Data Continuity Solution in Fault-tolerant Information Systems

Roman Danel, Zdeněk Neustupa
Institute of Economics and Control Systems
Faculty of Geology and Mining
VŠB – Technical University Ostrava
Ostrava, Czech Republic
roman.danel@vsb.cz, zdenek.neustupa@vsb.cz

***Abstract*: The paper deals with fault tolerant information systems in raw material industry. Database is a heart of every information (control) system. To store the system in the category of fault tolerant (or disaster tolerant) systems, the goal is to solve the database continuity. Current database systems offer technology for online data mirroring. The paper brings details about solution of database mirroring using Microsoft SQL Server, with practical experiences on real projects.**

Keywords: Fault tolerant Control System, Database Mirroring, Continuity Management, Microsoft SQL Server, automatic failover

I. INTRODUCTION

In this paper we are talking about control and information systems in industry. We see increase of demands on high availability, reliability and security. One way of accomplishing these demands is design a system that falls into the category of so-called "fault tolerant" systems.

What is a fault tolerant system? The system meets requirements for resistance for any faults (power outage, hardware failure, network failure…). One possible solution is the duplication of system: production system and its copy on backup system. If the production system fails, control is switched to the backup system, without significant interruption of management of technological process.

Note: Disaster tolerant system is fault tolerant system with high level of resistance and security. The backup system must be placed in different site (at minimum in other building) to be disaster resistant (such as fire, flood etc.). But in terms of software architecture, there is no difference between fault tolerant and disaster tolerant system.

One of the most important problems considering the building of fault tolerant control (information) systems is how to manage continuity of the data stored into relational databases. Strict Real Time Control System with no ties to the technological process history is possible simply switch to backup system. But what we should do when the system contains a database with continuous data? Imagine a system with periodic sampling of production data working 7x24 (metallurgical production, coal preparation, gas flow…) used for balance or time-dependent statistical reports. Interruption of the continuous time series data has fatal consequences on system results.

What can we do? The oldest and the standard method is to use database backup and restore. This mechanism is a standard part of every database system. After switching control to the backup system, we just restore the data into database from last database backup. But how old is your last backup? A few minutes? Or hours? Days? There is a gap in the continuous data depending on how old the last backup is and this is not what we are talking about – fault tolerant system.

II. DATABASE MIRROR

To solve mentioned problem – insuring real on-line data continuity in relational databases, we have to use other tools. One of the options is to use database replication. This is an effective method, but difficult to administering and maintenance.

One possibility, which the most known database systems have started offering recently, is technology called "database mirror". It's a software solution for increasing database availability. Database mirroring transfers data directly from one server to another. In next chapter I'll try to explain the solution of fault-tolerant database system using Microsoft Database Mirror technology. In database mirroring, one SQL Server instance continuously sends a database's transaction log records to a copy of the database on another standby SQL Server instance. Microsoft introduced this technology in the year 2005 (when "SQL Server 2005" database system has been launched into the market).

There is thousands of articles about the database mirror technology in Internet nowadays. But there is missing guide covering all the steps for deploying this technology. This is a key goal of this paper.

III. HOW IT WORKS?

Note that we are talking about standard relational databases based on transactions (a mechanism to insure data integrity). Database system working with transactions has a possibility to turn back a failed operation (performing a rollback operation) and is able to bring data to the previous (stable) state. The history of action executed by database system to guarantee transactional properties must be

978-1-61284-361-2/11/$26.00 ©2011 IEEE

somewhere stored. In Microsoft SQL Server it's a file called transaction log. Based on the information stored in the transaction log the database system is able to do rollback of unsuccessful database operation.

In database mirroring, there are two SQL Server connected together, the source server is called principal server and the destination server is called mirror server. As the principal server writes changes into log file (block of data from log buffer in memory to disk), it simultaneously sends that block of log records to the mirror instance. Because the mirror database replays the principal's transaction log records, it duplicates the database changes on the principal database. This made a true on-line backup of all data changes including modification of data structure and all data definitions language statements.

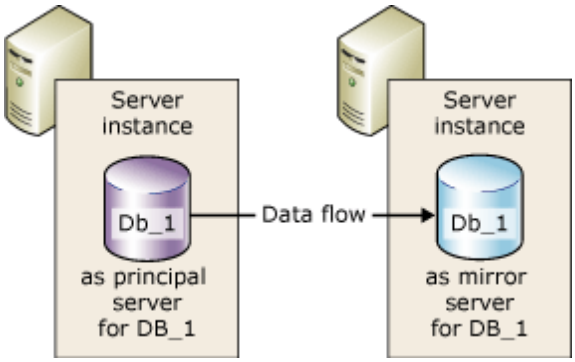

Figure 1. Database mirror overview [2].

IV. CONDITION FOR A MIRROR

To set up a database mirror between two instances of Microsoft SQL Server, we must meet several conditions:

- Firstly, we have to use Microsoft SQL Server 2005/2008 Standard Edition or Enterprise Edition. SQL Server Workgroup Edition doesn't support a mirror technology
- The database on both instances that will be mirrored must have the same name
- The principal database must be set to a full recovery model (recovery models control transaction log maintenance; there are three models available – simple, full and bulk-copy). There is one restrictions only - in full recovery mode log records that result from bulk-logged operations cannot be sent to the mirror database (for example high speed data loading…).
- The mirror database must be initialized from a restore of the principal database with NORECOVERY options. Because the mirror database is in a recovering state after that, it cannot be accessed directly by the users. Direct access is available after the database mirror disconnection and then recovery state must be turn off.

V. SET UP A DATABASE MIRROR

Before we could run the database mirror, some prepare task is need to be done.

Database mirror works under TCP/IP connection through TCP/IP endpoints with an authentication using certificates. The servers involved in the database mirroring session must trust each other (in non-trusted domains, the mirror doesn't work without a certificates).

So first we should generate certificates and create a login, in following steps:

1. Create a master key (master key is used for certificate)
2. Create certificate
3. Create login
4. Export certificate to a file
5. Move and import certificate

Let's suppose that SERVER_P is a principal server (production server in fault tolerant solution) and SERVER_M is a mirror server (on backup server).

Example of code:

```
use master;
create master key encryption by
password='xxxx';
create certificate SERVER_P_cert with
       subject='SERVER_P certificate',
       start_date='20110101';

create login HOST_SERVER_M_login  with
password = 'xxxx';

create user HOST_SERVER_M_user for login
HOST_SERVER_M_login;

/* backup certificate on SERVER_P for
   Transfer to SERVER_M */
backup certificate SERVER_P_cert to file =
'c:\sql\certif\SERVER_P_cert.txt';

/* load certificate from SERVER_M */
create certificate HOST_SERVER_M_cert
    authorization HOST_SERVER_M_user
    from file =
   'C:\sql\certif\SERVER_M_cert.txt';

/* check of installed certificates */
select * from sys.certificates;
```

Followed requirements must be met:

- All the statements are run in "master" database.
- Execution of statements requires you have system administrator rights
- Mind the default settings – one of the most common problems with the database mirror is that the database mirror failed after certain time of successful

working due to certificate lifetime expiration!! Good practice is to prolong a certificate lifetime.

The same steps should be done on mirror server.

Now we are ready to create and establish a TCP/IP endpoints (endpoint is an object that enables SQL Server communicate over the network).

```
create ENDPOINT [mirror_ep] STATE=STARTED
   AS TCP (LISTENER_PORT = 5022,
   LISTENER_IP = ALL)
   FOR DATA_MIRRORING
   (ROLE = ALL, AUTHENTICATION = CERTIFICATE
   SERVER_P_cert, ENCRYPTION=REQUIRED
   ALGORITHM AES);
```

In this example we are working on principal server and we are using default TCP port 5022. The same command should be done on mirrored server (with SERVER_M_cert certificate created by the commands on previous page).

Further step is to grant connection to the endpoint:

```
   GRANT CONNECT ON ENDPOINT::Mirror_ep TO
[HOST_SERVER_M_login];
```

This command allows access to SERVER_P endpoint from SERVER_M (mirror server) via certificate SERVER_P_cert (which must be loaded on SERVER_M).

Both servers are ready for establish the database mirror now. Activation of the database mirror requires these steps:

1. Full backup of principal database
2. Transaction log backup of principal database
3. Restore of full backup on mirror server (in recovery mode)
4. Restore of transaction log on mirror database
5. Start the mirror

In example here, we do these commands:

```
   /* full backup of database "db_name" */
   backup database db_name to
       disk='c:\data\mssql\db_name.bak' with
       format;

   /* backup of transaction log */
   backup log db_name to
       disk='c:\data\mssql\b_name_log.bak'
       with format;

   /* restore database in recovery mode on
       SERVER_M */
   restore database db_name from
       disk='C:\data\mssql\db_name.bak'
       with norecovery, replace;

   /* restore transaction log on SERVER_M */
   restore log db_name from
       disk='c:\data\mssql\db_name_log.bak'
       with norecovery;

   /* start the mirror on SERVER_M */
   alter database db_name
       set partner='tcp://SERVER_P:5022';

   /* start the mirror on SERVER_P */
   alter database db_name
       set partner='tcp://SERVER_M:5022';
```

Final notes:

- When we do the database and log backup of principal database, the best practice is to do truncation of log file before
- Successfully established mirror is indicated by "synchronizing" status in the list of databases in SQL Server Management Studio.
- "Partner not found" error message after "alter" command is likely to incorrect access point settings (something wrong with endpoint, login or certificate…)
- Mirrored database is now in recovery mode and isn't accessible for the users

VI. SWITCH TO BACKUP SERVER

The significant differences between database replication and database mirror are:

- Database mirror doesn't require any changes or settings in the database structure
- Database mirror in SQL Server could be run (according Microsoft license policy) under one SQL Server license (running on principal and mirror server both).

Database mirror could be also created in High Availability Operating Mode using third server called "witness". In this mode, database mirror is self-monitoring with automatic failover. Clients accessing SQL server using "SQL Server Native Driver" are automatically redirected after principal server fails to its partner server [2].

To test status of the database mirror, we can use follow command:

```
   select sdb.name, isnull(
       mirroring_state_desc, 'NOT ACTIVE'),
       isnull(mirroring_partner_name,
       'NONE'), isnull(mirroring_role_desc,
       'N/A'),isnull(mirroring_safety_level_d
       esc, 'N/A')
       from sys.database_mirroring sdm,
        sys.databases sdb
          where sdb.database_id =
          sdm.database_id and name =
           'db_name';
```

To realize failover solution, we could disconnect mirror and switch the mirrored database into usable status by the following commands:

```
   alter database db_name set partner off;
```

```
/* turn off the recovery mode */
restore database db_name;
```

Important technical note – after transferring the mirrored database on backup server to an operational status, we must refresh users internal ID (this is caused because we restore database from backup which was build on principal server – the backup contains principal server users ID).

Refresh user ID command:

```
use db_name;
exec  sp_change_users_login  'Update_one',
    user_name, user_name;
```

In Microsoft SQL Server 2008 there is a possibility to pause and resume the database mirror:

```
ALTER DATABASE db_name SET PARTNER
    SUSPEND;

ALTER DATABASE db_name SET PARTNER RESUME;
```

VII. EXPERIENCES WITH A REAL PROJECT

I have designed fault tolerant information system for OKD coal preparation plant. Once again – the fault tolerant system is system that ensures continuity in required time after any failure that can appear. It considers the events like power supply, damage of network, hardware failure (server, disks, processor, RAID, operating system). The minimal solution is to build two identical systems, backup by UPS, with double independent network. And what is the meaning of "required time"? It depends on character of controlled process. If you control a nuclear power plant, we will probably talk about time in range of milliseconds. In coal preparation plant, it has been established approximately on 5 minutes (this is a time of loading one wagon). This mean, that any failure should be solve during 5 minutes and during this period system MUST continue. Building of two system is technically easy task, but the database is the weakness part of the system (because database contains continuous time-series data – we don't want to have gap appeared after failure).

The best solution for database continuity is cluster failover, but in real commercial project, there is a budget you must meet. Cluster failover is quite expensive solution. Replication technology is difficult in maintenance and there was a demand for full automatic failover. From these reason – costs and automatic failover – I did a research in the field of database mirror to solve full automatic fail-over system.

This system has been successfully run on Darkov Coal Preparation Plant (OKD coal mine joint stock company, Karviná, Czech Republic) in the year 2007 [1]. With total budget of 2.7 million CZK for whole system (including all hardware, network, data concentrators, Etherlinks, PC, licenses, design, develop and implementation of information system including SCADA systems and data collection etc)., I had a really small budget for failover solution. This was one of the reasons why I choose database mirror technology.

After four years of working, there is clear that database mirror has been good choice. We noticed two hardware failures (problem with RAID controller) and the automatic swap to backup system after failure was successful.

With a mirror technology, there appeared two operational problems. First problem was that the database with quite high traffic (there is a system of alarms reporting from 1600 sensors working 24x7, it produces 5 millions record by the year) led to enormous increase of database logs (several GB per month). We solved this problem by shrinking the databases and theirs logs and I then created system of regular operational backup of logs to prevent theirs continuous increase.

Second problem consider the certificates. In this paper I described how to establish the mirror in this situation: you have SQL Server Standard, two instances without witness server role and this is working in domain. This is important – because in all articles about database mirror technology there is not specified exactly that in this case you MUST use certificates to ensure authentication connection via endpoint (note it's not necessary without a domain environment). OK, we created and launched certificates, and database mirror is working. There is no any notice, that certificates have default time expiration only one year and after this period, there is no message in system log, that certificates are out of date. Mirror is still working after certificates expiration, to the first reboot of system. After that, mirror couldn't be established, but again – without any message or warning why.

VIII. CONCLUSION

The database mirror is great tool for realize failover solution of relational database in fault-tolerant control systems. It allows having an on-line copy of all production data stored into database on production server, without danger of data loss. I concentrated on Microsoft SQL Server database system (as a demand of customer on the real project described above), but other systems offer similar tools also. As a next step I want to test and set-up mirror technology in Open Source databases (for example PostgreSQL). The goal is to build also a fault-tolerant control system based on Linux server platform with Open Source relational database.

REFERENCES

[1] Danel, R.: Automation and Control Systems at Coal Preparation Plants in Czech Republic. CINEST - International Symposium on Earth Science and Technology 2009, p. 515-516, Fukuoka, Japan 2009

[2] Database Mirroring in SQL Server 2005. Microsoft Technet Library. [cit. 10.1.2011]. Available at <http://technet.microsoft.com/en-us/library/cc917680.aspx>.

[3] Database Mirroring Overview. Microsoft MSDN [cit. 10.1.2011]. Available at <http://msdn.microsoft.com/en-us/library/ms189852.aspx>.

[4] Microsoft SQL SERVER 2005 Implementation and Maintenance Training Kit, Microsoft Press , 2006.