Public

SAP SQL Anywhere
DBミラーリング環境構築 WorkShop

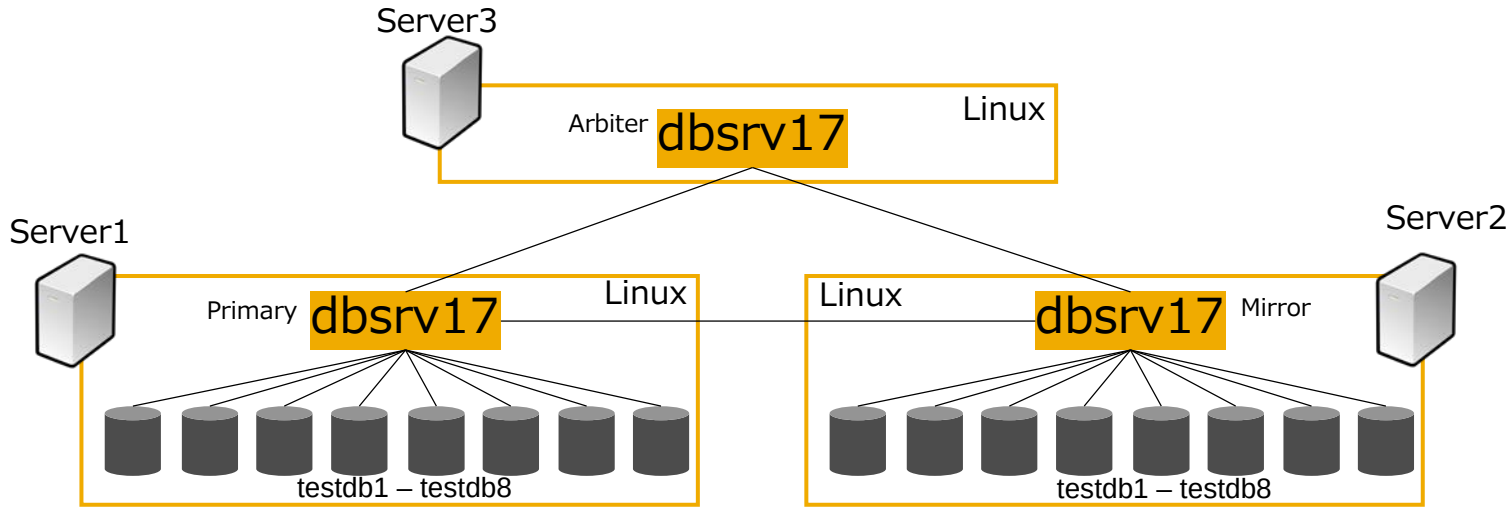
SAPジャパン株式会社
ソリューション統括本部
デジタル・エンタープライズ・プラットフォーム部
境　直人

2017/3/10

テスト環境

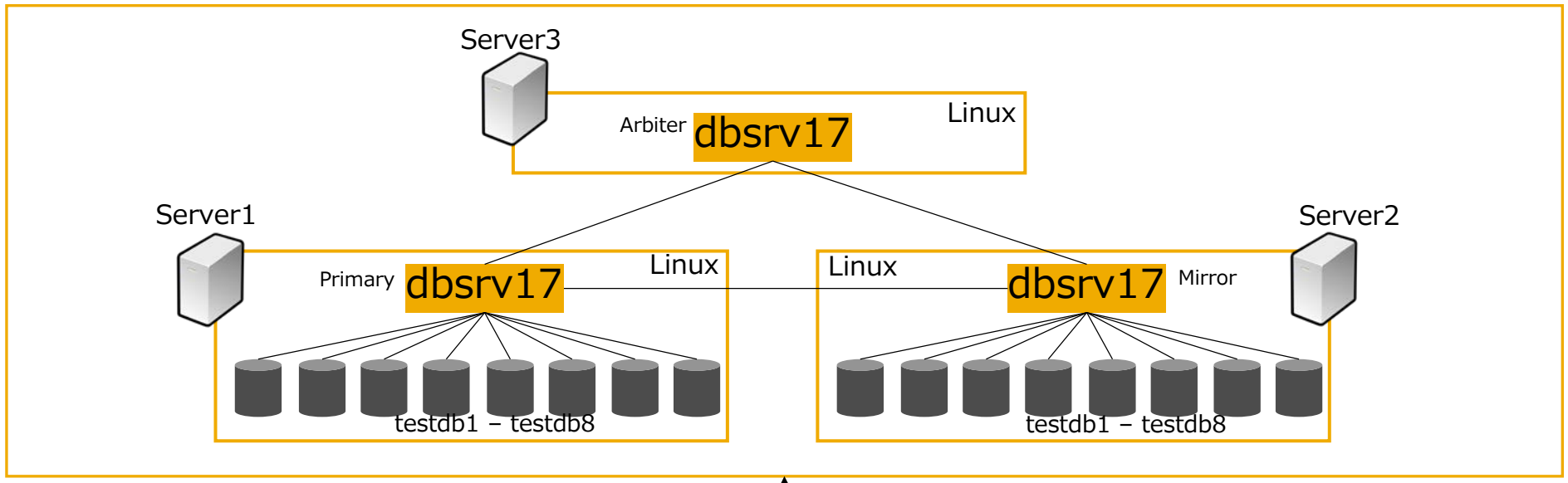
テスト環境

- **環境はSQLA17 on Linuxを想定しています。**
  - 3台のLinuxにそれぞれSQLA17がインストールしてある想定です。
- **1DBサーバに8DB、これを１アービターで監視するというシナリオです。**
- **ホスト名はServer1,2,3、データベース名はtestdb1,2,3,…8としています。**
  - ホスト名は実際に使用するもので置き換えて下さい。
    - Server1=Primary,Server2=Mirror,Server3=Arbiter
  - DBファイルは/DB/testdb<n>/testdb<n>.dbとして配置
  - 各ホストにはsybaseユーザーでログイン、このユーザーの.bashrcにはsource sa_config.shを仕込み済み



テスト環境の注意

テスト環境ではポート10000を使用して各サーバ間で通信を行います。ファイアーウォール等からポート10000を除外して下さい。



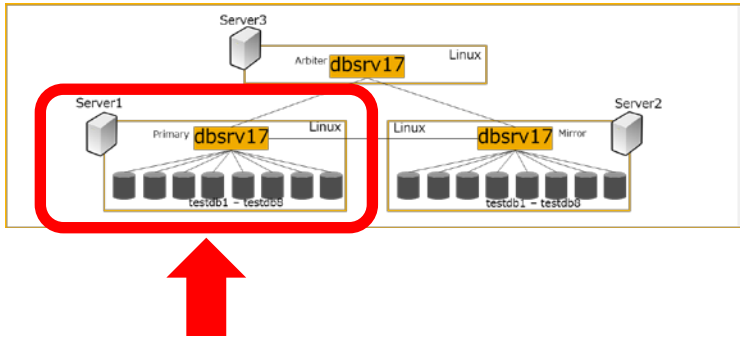
また、テストでクライアントから接続するので、この3台とは別に1台SQL Anywhere Clientがインストールされたマシンを用意するのが好ましいです。

SQL Anywhere 17 Client

ミラーリング環境の作成

1.　DBを8個作成 (on Server1)

Server1上でテスト用にDBを8個作成します。newdemo.shはdemo.dbを作成するスクリプトです。引数で違う名前のDBを作成できます。



--- rootで
mkdir /DB
chmod a+wrx /DB

---sybaseユーザーで
mkdir /DB/testdb<n>
cd /DB/testdb<n>
newdemo.sh testdb<n>

} <n>を1-8に変更して8回

```
[sybase@ip-172-31-15-51@Server1 testdb1]$ newdemo.sh testdb1
SQL Anywhere Erase Utility Version 17.0.4.2053
Database "testdb1.db" not found
SQL Anywhere Initialization Utility Version 17.0.4.2053
CHAR collation sequence:  UTF8BIN(CaseSensitivity=Ignore)
CHAR character set encoding:  UTF-8
NCHAR collation sequence:
UCA(CaseSensitivity=Ignore;AccentSensitivity=Ignore;PunctuationSensitivity=Primary)
NCHAR character set encoding:  UTF-8
Database is not encrypted
Creating system tables
Creating system views
Setting option values
Database "testdb1.db" created successfully
```

コピペ用全コマンド、sybaseユーザーで実行する。

```
mkdir /DB/testdb1
cd /DB/testdb1
newdemo.sh testdb1
mkdir /DB/testdb2
cd /DB/testdb2
newdemo.sh testdb2
mkdir /DB/testdb3
cd /DB/testdb3
newdemo.sh testdb3
mkdir /DB/testdb4
cd /DB/testdb4
newdemo.sh testdb4
mkdir /DB/testdb5
cd /DB/testdb5
newdemo.sh testdb5
mkdir /DB/testdb6
cd /DB/testdb6
newdemo.sh testdb6
mkdir /DB/testdb7
cd /DB/testdb7
newdemo.sh testdb7
mkdir /DB/testdb8
cd /DB/testdb8
newdemo.sh testdb8
```

ポイント

この例ではデモDBを使用していますが、自分で作成したアプリケーションのDBを使用することも出来ます。その場合はDBファイルとログファイルをセットで該当ディレクトリにコピーして下さい。

- Dbinit直後のDBを使用する場合はログファイルが出来ていませんので、一度dbengなどで起動して、ログファイルを作成させてから使用するようにして下さい。

ミラーリングの設定

以下のような設定にするとします。

Group1（これはこの8DBの集まりをこう呼ぶだけで設定内で意味はない）								
	testdb1	testdb2	testdb3	testdb4	testdb5	testdb6	testdb7	testdb8
Server1DB サーバ名	group1_server1 Port=10000							
Server2DB サーバ名	group1_server2 Port=10000							
プライマリの論理サーバー名	testdb1_primary	testdb2_primary	testdb3_primary	testdb4_primary	testdb5_primary	testdb6_primary	testdb7_primary	testdb8_primary
ミラーの論理サーバー名	testdb1_mirror	testdb2_mirror	testdb3_mirror	testdb4_mirror	testdb5_mirror	testdb6_mirror	testdb7_mirror	testdb8_mirror
DBの配置場所	**Server1:** /DB/testdb1/testdb1.db & log **Server2:** /DB/testdb1/testdb1.db & log	**Server1:** /DB/testdb1/testdb1.db & log **Server2:** /DB/testdb1/testdb1.db & log	**Server1:** /DB/testdb1/testdb1.db & log **Server2:** /DB/testdb1/testdb1.db & log	**Server1:** /DB/testdb1/testdb1.db & log **Server2:** /DB/testdb1/testdb1.db & log	**Server1:** /DB/testdb1/testdb1.db & log **Server2:** /DB/testdb1/testdb1.db & log	**Server1:** /DB/testdb1/testdb1.db & log **Server2:** /DB/testdb1/testdb1.db & log	**Server1:** /DB/testdb1/testdb1.db & log **Server2:** /DB/testdb1/testdb1.db & log	**Server1:** /DB/testdb1/testdb1.db & log **Server2:** /DB/testdb1/testdb1.db & log
アービター (on server3)	group1_arbiter Port=10000 Authentication = ‘abc’							

2.　8DBを起動

dbspawn dbsrv17 -n group1_server1 -x "tcpip(port=10000)" -su <passwd>
/DB/testdb1/testdb1.db -xp on /DB/testdb2/testdb2.db -xp on /DB/testdb3/testdb3.db -xp on
/DB/testdb4/testdb4.db -xp on /DB/testdb5/testdb5.db -xp on /DB/testdb6/testdb6.db -xp on
/DB/testdb7/testdb7.db -xp on /DB/testdb8/testdb8.db -xp on –o group1_server1.log

を実行してDBを起動して下さい。上記はバッチなどにしたほうが良いでしょう。

dbspawn：SQL Anywhereバックグラウンド起動ユーティリティ

　dbsrv17 -udオプションと違い、このコマンドを経由するとDBがバックグラウンドで正常起動して受信可能になるまで待機してからコマンドの成否応答（0（成功）かそうでない値)を返す。

- -udの場合もバックグラウンド起動するが、コマンドの応答としては「dbsrv17プロセスの起動」が対象となることに注意。起動直後にエラーになって終了した場合も０が返されるのでDBが起動したかはわからない。

-su <passwd>：ユーティリティDBの使用とパスワード指定

　「動作中のインスタンスに後からDBを追加」などがあり得るので使用したほうが良いように思います。指定しないとインスタンスを停止しないとDBが追加できなくなります。

参考：dbspawn使用時の停止方法

Dbspawn使用時はdbstopでDBを停止する必要があります。参考までに今回の環境の場合は以下のコマンドで停止できます

Server1:

dbstop -c "Host=localhost:10000;Server=group1_server1;uid=DBA;pwd=sql“

Server2 :

dbstop -c "Host=localhost:10000;Server=group1_server2;uid=DBA;pwd=sql“

Server3

dbstop -c "DBN=utility_db;UID=DBA;PWD=passwd;HOST=localhost:10000;Server=group1_arbiter"

＊他のマシンからも停止は可能です。

ミラーリング環境ではミラー側から停止させるのが基本です。（プライマリーを先に停止させてもフェールオーバーが発生します。）

3. ミラーリング設定

Interactive SQLでtestdb1に接続して以下のSQLを実行します。

<server1>/<server2>/<server3>は実際のホスト名で置き換えて下さい。（次スライドも参照）

Interactive SQL での接続文字列　：　“UID=DBA;PWD=sql;HOST=<server1>:10000;SERVER=group1_server1;DBN=testdb1"

```
CREATE MIRROR SERVER group1_server1
AS PARTNER
connection_string='SERVER=group1_server1;host=<server1>:10000'
state_file='/DB/server1state.txt';

CREATE MIRROR SERVER testdb1_primary
AS PRIMARY
connection_string='SERVER=testdb1_primary;host=<server1>:10000,<server2>:10000';

CREATE MIRROR SERVER group1_server2
AS PARTNER
connection_string='SERVER=group1_server2;host=<server2>:10000'
state_file='/DB/server2state.txt';

CREATE MIRROR SERVER testdb1_mirror
AS MIRROR
connection_string='SERVER=testdb1_mirror;host=<server1>:10000, <server2>:10000';

CREATE MIRROR SERVER group1_arbiter
AS ARBITER
connection_string='SERVER=group1_arbiter;HOST=<server3>:10000';
SET MIRROR OPTION authentication_string='abc';
```

testdb1の部分を　testdb2,testdb3…..testdb8に変更して8DBに対して実行します。

チェック！

ここで/DBにserver1state.txtが出来ていることを確認して下さい。

```
[sybase /DB]$ ls
group1_server1.log  testdb1  testdb3  testdb5  testdb7
server1state.txt    testdb2  testdb4  testdb6  testdb8
```

中身も確認して、8個分のDBエントリがあることを確認して下さい。

4.　バックアップ作成とServer2へのコピー

ミラーリング設定が完了したDBファイルのバックアップを作成し、Server2へ配置します。
以下はServer2 からServer1の各DBのバックアップを取得し、各ディレクトリに配置しています。

Server2上で実行します。

```
mkdir /DB/testdb1
cd /DB/testdb1
dbbackup -c "UID=DBA;PWD=sql;HOST=<server1>:10000;SERVER=group1_server1;DBN=testdb1" ./
mkdir /DB/testdb2
cd /DB/testdb2
dbbackup -c "UID=DBA;PWD=sql;HOST=<server1>:10000;SERVER=group1_server1;DBN=testdb2" ./
mkdir /DB/testdb3
cd /DB/testdb3
dbbackup -c "UID=DBA;PWD=sql;HOST=<server1>:10000;SERVER=group1_server1;DBN=testdb3" ./
mkdir /DB/testdb4
cd /DB/testdb4
dbbackup -c "UID=DBA;PWD=sql;HOST=<server1>:10000;SERVER=group1_server1;DBN=testdb4" ./
mkdir /DB/testdb5
cd /DB/testdb5
dbbackup -c "UID=DBA;PWD=sql;HOST=<server1>:10000;SERVER=group1_server1;DBN=testdb5" ./
mkdir /DB/testdb6
cd /DB/testdb6
dbbackup -c "UID=DBA;PWD=sql;HOST=<server1>:10000;SERVER=group1_server1;DBN=testdb6" ./
mkdir /DB/testdb7
cd /DB/testdb7
dbbackup -c "UID=DBA;PWD=sql;HOST=<server1>:10000;SERVER=group1_server1;DBN=testdb7" ./
mkdir /DB/testdb8
cd /DB/testdb8
dbbackup -c "UID=DBA;PWD=sql;HOST=<server1>:10000;SERVER=group1_server1;DBN=testdb8" ./
```

5. Server2のDBの起動

Server2上で下記のコマンドで各DBを起動します。

dbspawn dbsrv17 -n group1_server2 -x "tcpip(port=10000)" -su <passwd> /DB/testdb1/testdb1.db -xp on /DB/testdb2/testdb2.db -xp on /DB/testdb3/testdb3.db -xp on /DB/testdb4/testdb4.db -xp on /DB/testdb5/testdb5.db -xp on /DB/testdb6/testdb6.db -xp on /DB/testdb7/testdb7.db -xp on /DB/testdb8/testdb8.db -xp on -o /DB/group1_server2.log

/DB/group1_server2.logにはペアのDBとの接続が確立して同期が行われた旨が表示されます。

```
I. 12/27 07:09:02. Database "testdb3" mirroring:  determining mirror role ...
I. 12/27 07:09:02. Database "testdb3" mirroring:  synchronizing ...
I. 12/27 07:09:02. Database "testdb3" transaction log current offset is 1152649
I. 12/27 07:09:02. Database "testdb3" (testdb3.db) started as copy node at Tue Dec 27 2016 07:09
I. 12/27 07:09:02. Database "testdb2" mirroring:  determining mirror role ...
I. 12/27 07:09:02. Database "testdb2" mirroring:  synchronizing ...
I. 12/27 07:09:02. Database "testdb2" transaction log current offset is 1152649
I. 12/27 07:09:02. Database "testdb2" (testdb2.db) started as copy node at Tue Dec 27 2016 07:09
I. 12/27 07:09:02. Database "testdb8" mirroring:  determining mirror role ...
I. 12/27 07:09:02. Database "testdb8" mirroring:  synchronizing ...
I. 12/27 07:09:02. Database "testdb8" transaction log current offset is 1152649
I. 12/27 07:09:02. Database "testdb8" (testdb8.db) started as copy node at Tue Dec 27 2016 07:09
I. 12/27 07:09:02. Database "testdb4" mirroring:  determining mirror role ...
I. 12/27 07:09:02. Database "testdb4" mirroring:  synchronizing ...
I. 12/27 07:09:02. Database "testdb4" transaction log current offset is 1152683
I. 12/27 07:09:02. Database "testdb4" (testdb4.db) started as copy node at Tue Dec 27 2016 07:09
I. 12/27 07:09:02. Database "testdb6" mirroring:  determining mirror role ...
I. 12/27 07:09:02. Database "testdb6" mirroring:  synchronizing ...
I. 12/27 07:09:02. Database "testdb6" transaction log current offset is 1152649
```

逆にServer1の/DB/group1_server1.logにはこれまでmirrorに接続できない旨が出ており、

チェック！

ここで/DBにserver2state.txtが出来ていることを確認して下さい。

```
[sybase /DB]$ ls
group1_server2.log  testdb1  testdb3  testdb5  testdb7
server2state.txt    testdb2  testdb4  testdb6  testdb8
```

中身も確認して、8個分のDBエントリがあることを確認して下さい。
これで完了です。

6.　アービターの起動

Serve3にログインして、下記のコマンドを実行してアービターを起動します。

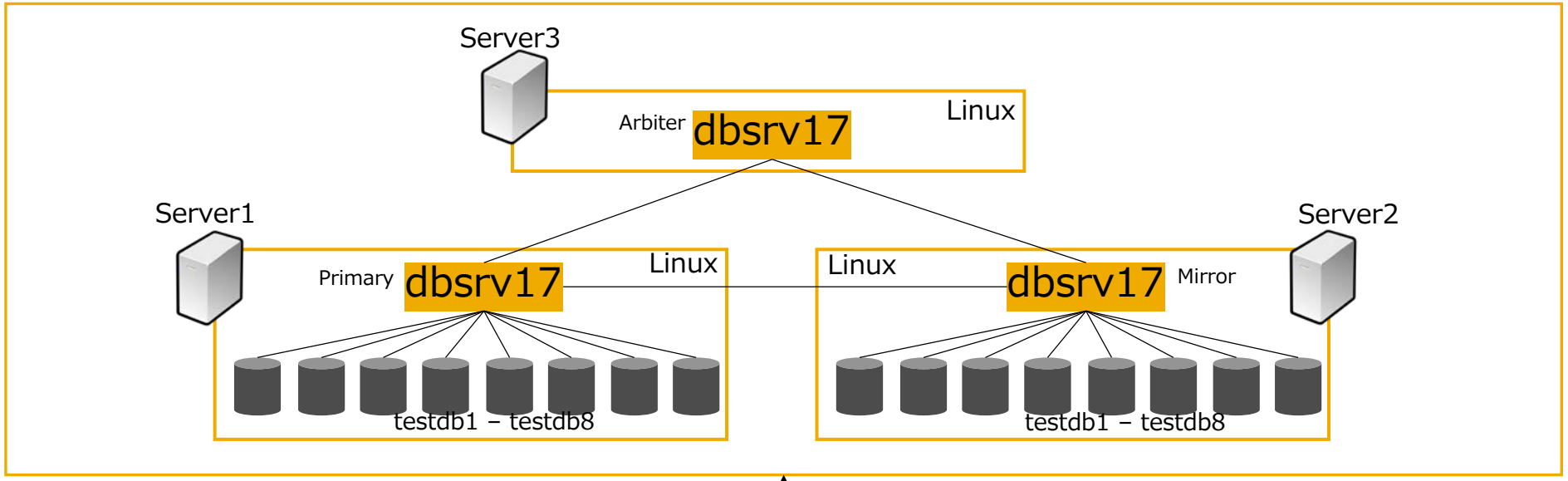
dbspawn dbsrv17 -n group1_arbiter -su passwd -x "tcpip(port=10000)" -xf /DB/group1_arbiterstate.txt -xa "AUTH=abc,abc,abc,abc,abc,abc,abc,abc;DBN=testdb1,testdb2,testdb3,testdb4,testdb5,testdb6,testdb7,testdb8" -o /DB/group1_arbiterlog.txt

/DB/group1_arbiterstate.txtの中身を確認して、8個分のDBエントリがあることを確認して下さい。
これで完了です。

ミラーリングDBへの接続

ミラーリングDBへ接続

作成したミラーリングDBへ接続してみます。この場合、実際のクライアントを想定してこの3台以外の別のマシンから接続テストをすることが望ましいです。



Interactive SQLで接続してみましょう！

このマシンから接続

SQL Anywhere 17 Client

接続文字列

（今回の場合）接続文字列は以下が基本になります。

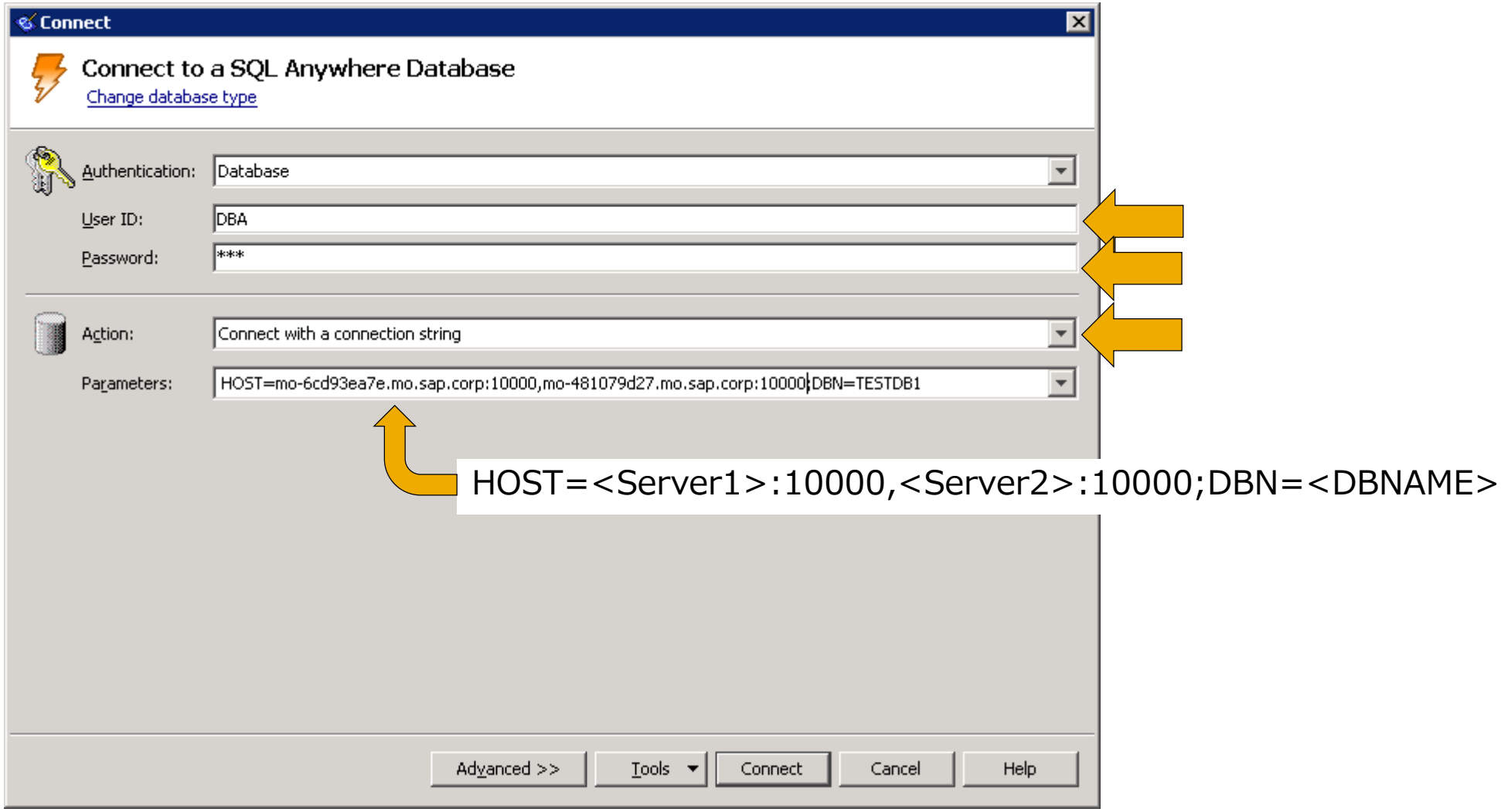
HOST=<Server1>:10000<Server2>:10000;DBN=<DBNAME>;UID=<UserID>;PWD=<PWD>

以下の様にすることでプライマリではなくミラーに接続を行うことも出来ます。

HOST=<Server1>:10000,<Server2>:10000;DBN=<DBNAME>;UID=<UserID>;PWD=<PWD>;NodeType=Mirror

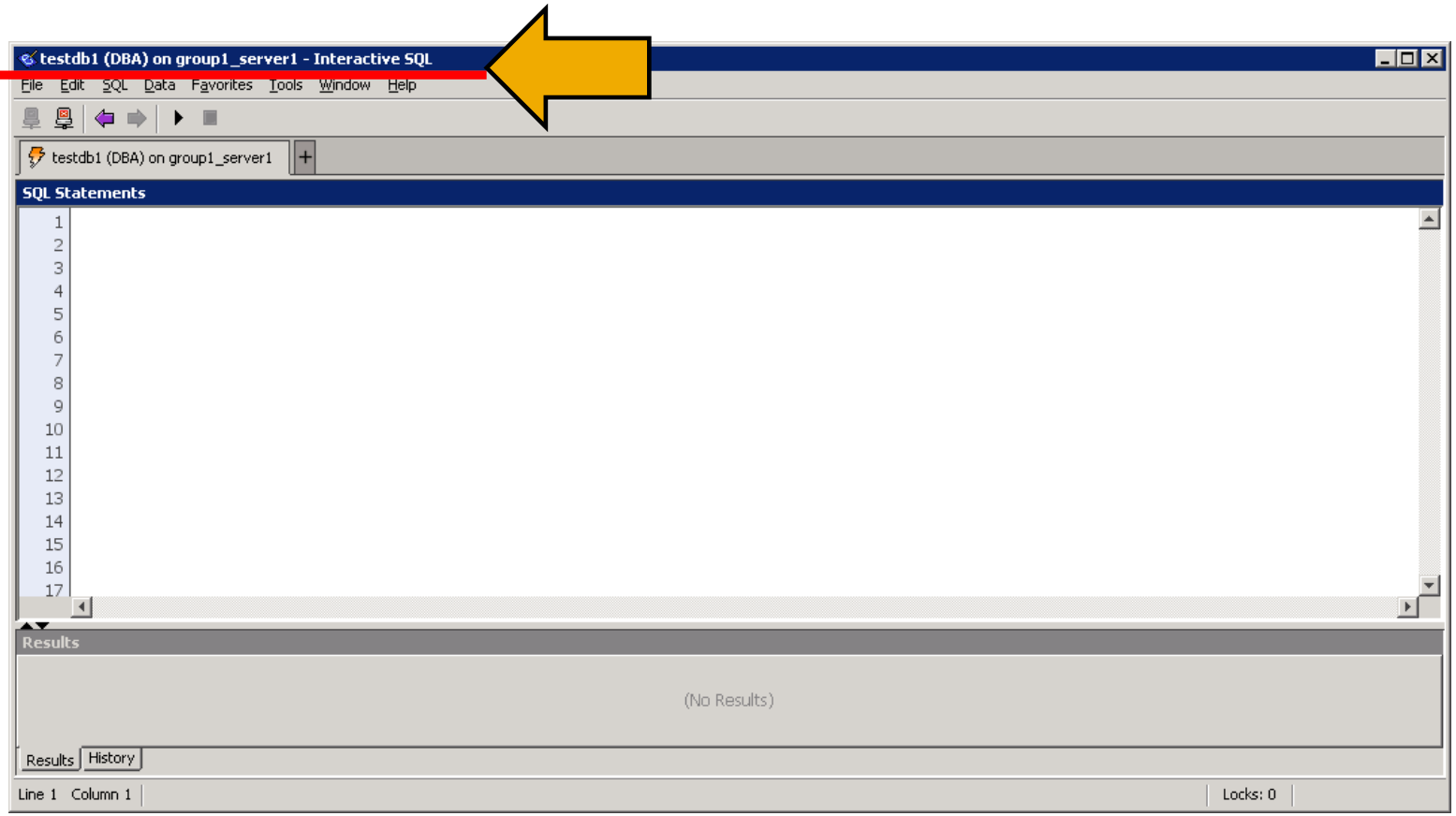
- 但し、ミラーが存在しない（1サーバーダウンした状態）もあることに注意が必要です。

Interactive SQLからの接続



HOST=<Server1>:10000,<Server2>:10000;DBN=<DBNAME>

Interactive SQLでは接続したDB名とサーバー名がウインドウのタイトル部分に表示されます。



SQLでは

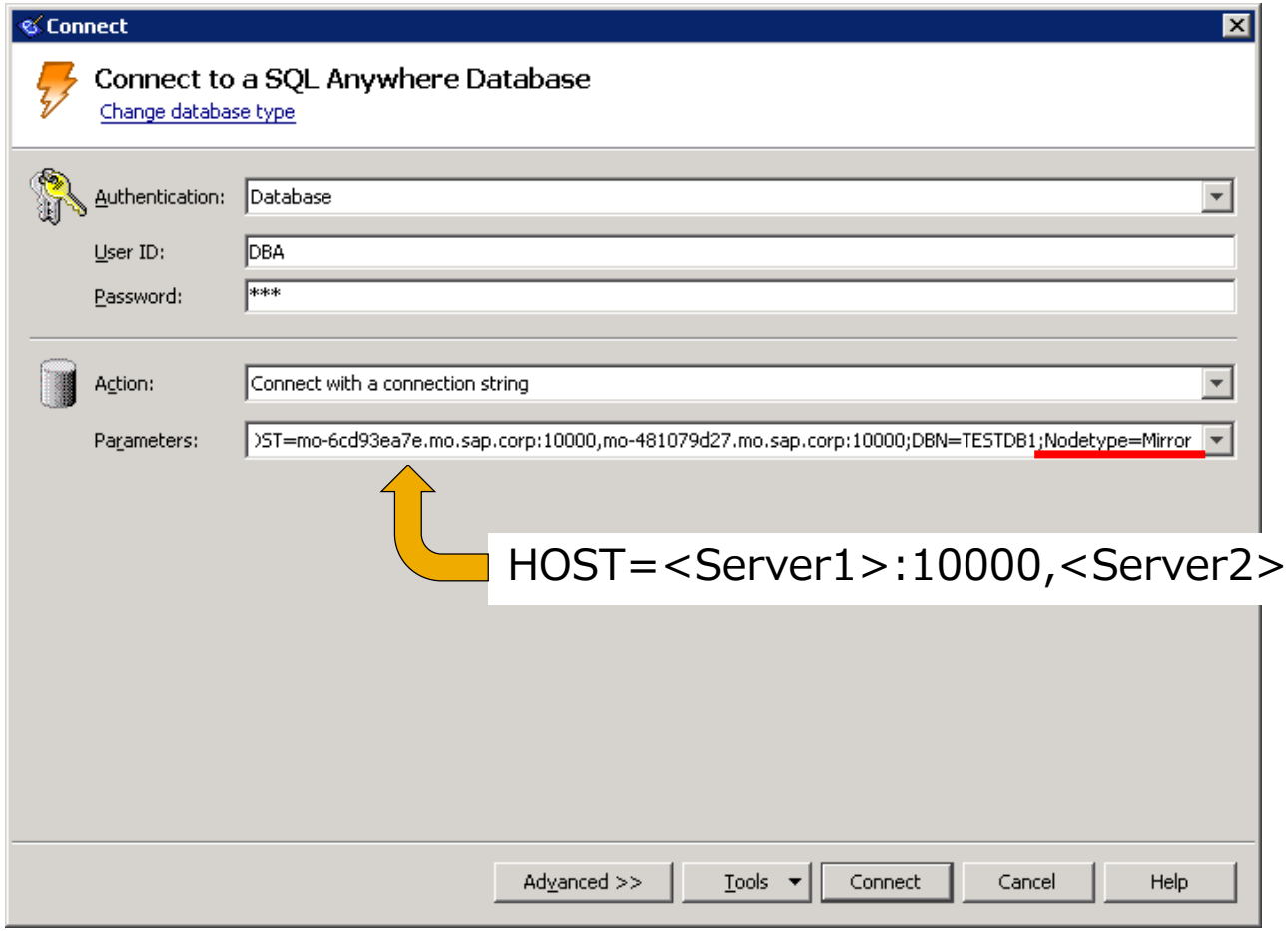
SELECT property('ServerName');
SELECT db_property('Name');

で同じ値を取得することが出来ます。

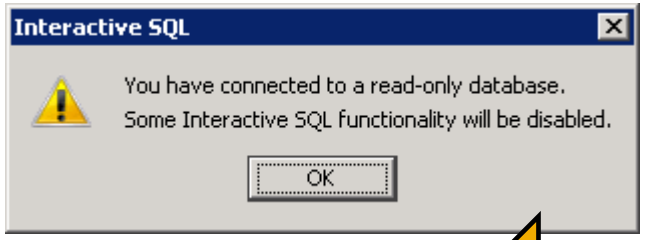
SELECT property(‘Machine_Name’);
SELECT property('TcpIpAddresses');
SELECT db_property('MirrorRole');

も確認してどのような値が返るか確認してください。

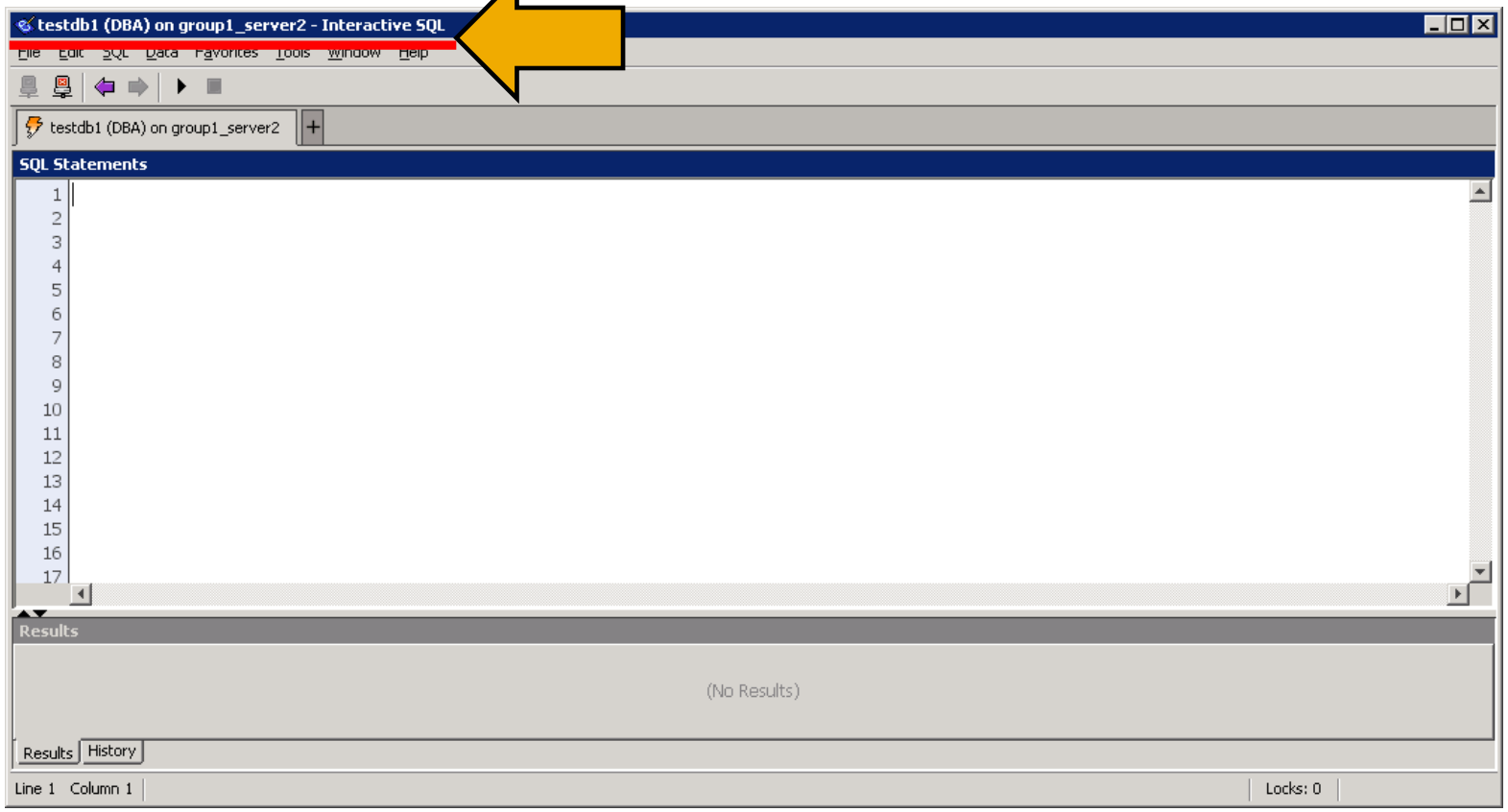
一旦切断するか、もう一つInteractive SQLを起動して、今度は接続文字列に追加指定を入れて接続します。



HOST=<Server1>:10000,<Server2>:10000;DBN=<DBNAME>;NodeType=Mirror



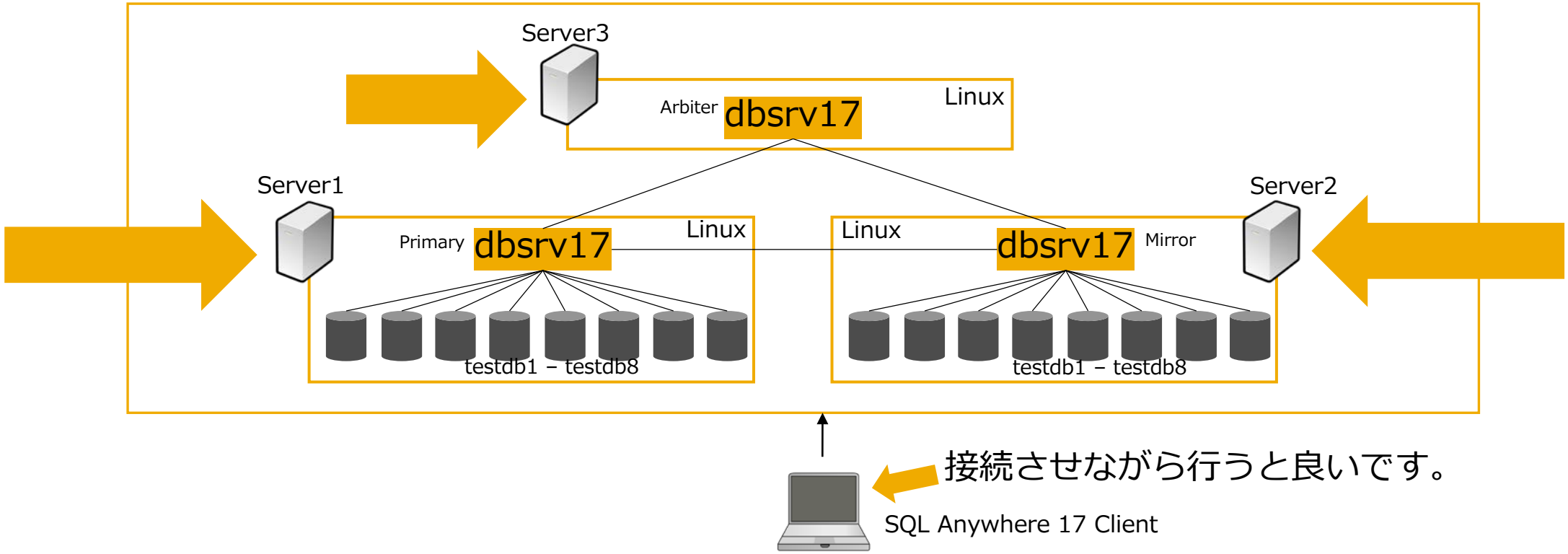
この接続設定はミラーに接続するというものです。ミラーはRead Onlyなのでinteractive SQLではこのような注意ダイアログが表示されます。



プライマリに接続したときと同様に各種プロパティ値を確認してみましょう

フェールオーバーテスト

各サーバで動作しているdbsrv17プロセスを停止させる、あるいはネットワークを切断する、マシンを停止させる等を行い、どのような挙動をするか確認して下さい。



参考：手動でのプライマリ・ミラーの切り替え

対象DB（プライマリ）に接続して

ALTER DATABASE SET PARTNER FAILOVER;

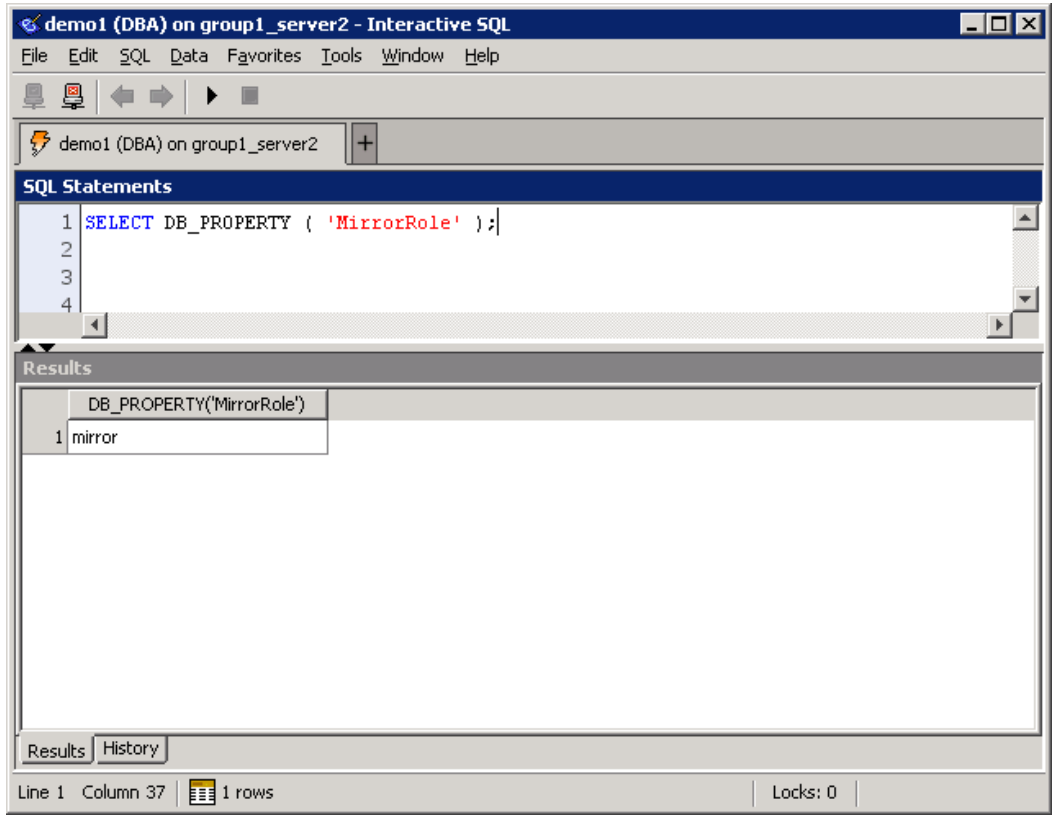
を実行するとフェールオーバーが発生し、プライマリとミラーが切り替わります。

- プライマリ・ミラーとも接続しているコネクションが強制切断されます。
- 実行時にミラーが停止している場合、コマンドはエラーになります。
- ミラー側で実行することは出来ません。

現在接続しているのはプライマリ・ミラーどちらか？

SELECT DB_PROPERTY ('MirrorRole');

で返ってきた文字列がPrimaryかMirrorかで判別できます。



プライマリ・ミラー・アービターの接続状況は？

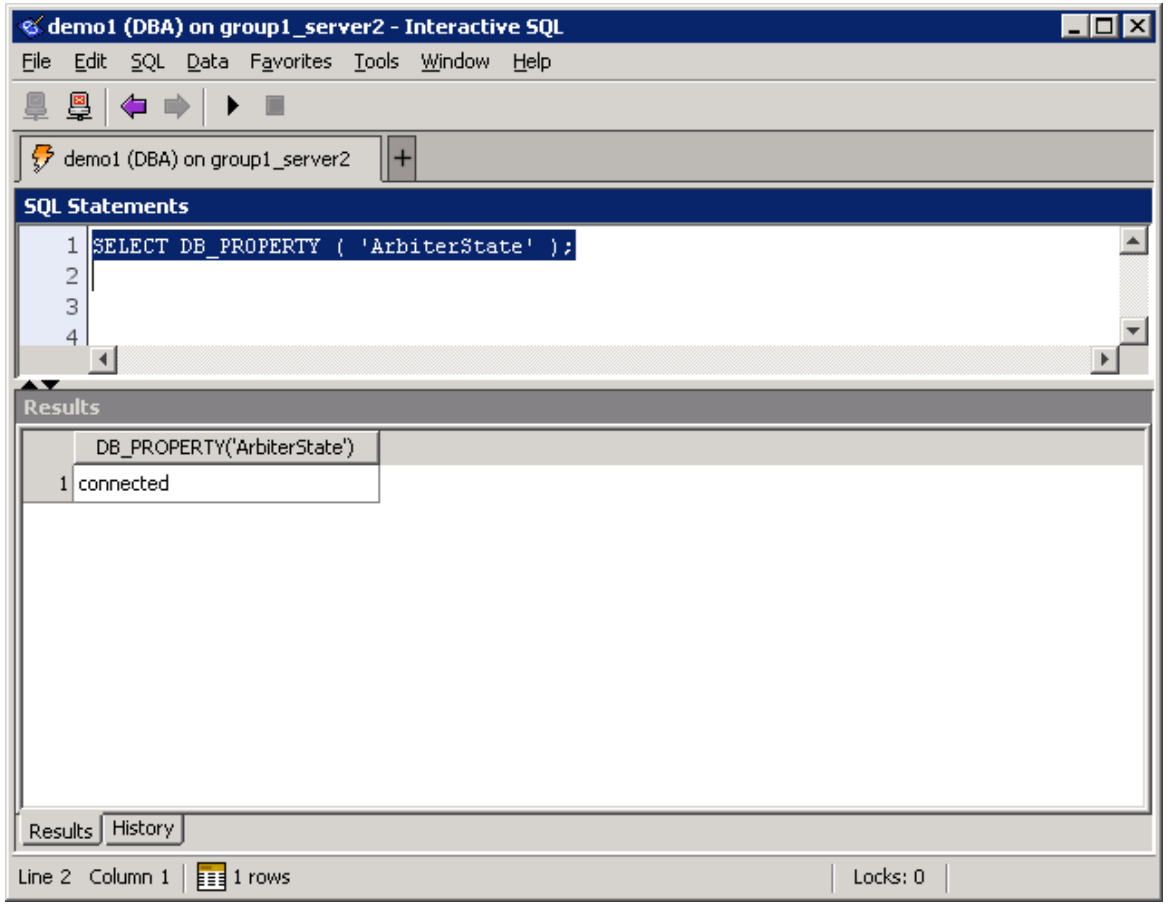
プライマリに接続し

SELECT DB_PROPERTY (‘PrimaryState');

SELECT DB_PROPERTY (‘MirrorState');

SELECT DB_PROPERTY ('ArbiterState');

を実行します。切断されている場合は disconnectedが返り、接続されている場合はそれ以外の値が返ります。



ミラーリング環境でのバックアップ

ミラーリング環境におけるバックアップの基本

- **ミラーリング環境では、プライマリ・データベースとミラー・データベースのバックアップは共通です。**
  - 両方共同じデータベースです。別々に取る必要はありません。
- **バックアップは「プライマリ」側で取得されます。**
  - 従ってバックアップ先をプライマリ・ミラーで共通のパスでアクセスできるようにしておかないと、切り替わった場合に別のバックアップコマンドを使用する事になります。
- **ログのバックアップ等の戦略は非ミラーリング状態時と変わりません。**

オンラインバックアップの実行手法

オンラインバックアップの実行手法は「コマンドライン」と「SQLコマンド」の2種類

- **Dbbackupコマンド**
- **BACKUP DATABASE文**

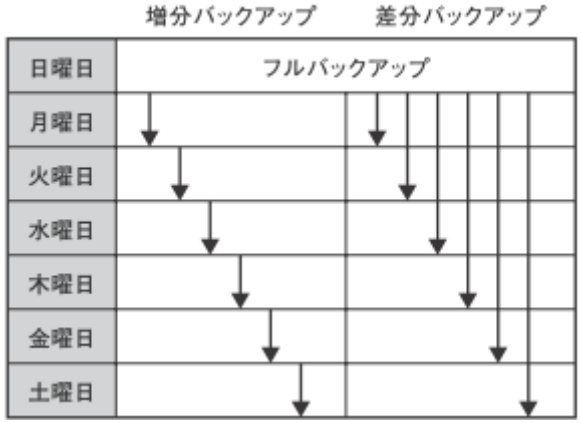
  - OSのコマンドラインとして実行するか、DBに接続して実行するか？
    - 管理ツールからのバッチ実行、夜間バッチ　→　コマンドラインからdbbackup
    - アプリケーション内のバックアップボタン　→　BACKUP　DATABASE文
    - SQL Anywhereのスケジュール機能　→　DB内の為BACKUP DATABASE文

両バックアップとも基本的に機能は変わらない。また、バックアップしたファイルも変わらないので、リストア時に組み合わせることも出来る。

- 例外として、dbbackupコマンドはクライアントマシンから実行してクライアントマシン側にバックアップを取得することが可能な機能がある。(-sを付けるとサーバー側出力となる。)

バックアップの種類

バックアップは3種類存在する。



- 差分はフルが無いと役に立たない
- 増分はその前までのバックアップが無いと役に立たない
- リストアはフルが一番速い（その他はフルに差分・増分を適用する形）

フルバックアップ

フルバックアップは以下のコマンドで行う。
- 現在のデータベースファイルとトランザクションログを指定したディレクトリに出力する。

BACKUP DATABASE文の例
BACKUP DATABASE DIRECTORY <backup_directory>;

dbbackupコマンドの例
$ dbbackup -c "dsn=SQL Anywhere 17 Demo;UID=DBA,pwd=password1" <backup_directory>

一般的なログバックアップ

一般的なバックアップ方法

差分バックアップは以下のように実行

- 現在のトランザクションログコピーをバックアップ先に出力
- 現在のトランザクションログはクリアしない。（そのまま大きくなる）

BACKUP DATABASE文の場合

```
BACKUP DATABASE DIRECTORY <backup_directory> TRANSACTION LOG ONLY;
```

dbbackupコマンドの場合

```
$ dbbackup -c "dsn=SQL Anywhere 17 Demo" -t <backup_directory>
```

増分バックアップは以下のように実行：これはDBミラーリング状態では使用できない。

- 現在のトランザクションログコピーをバックアップ先に出力し、現在のトランザクションログをクリアして新しいログとして開始する

BACKUP DATABASE文の場合

```
BACKUP DATABASE DIRECTORY <backup_directory> TRANSACTION LOG ONLY TRANSACTION LOG TRUNCATE;
```

dbbackupコマンドの場合

```
$ dbbackup -c "dsn=SQL Anywhere 17 Demo;uid=DBA;pwd=password1" -x -t <backup_directory>
```

ベストプラクティス１：
DBミラーリング環境でログバックアップ

トランザクションログのリネーム機構を利用する。

これは先の増分バックアップと同様だが、若干異なる動作が行われる。

- 現在のトランザクションログをYYMMDDxx.logにリネームする。（これは現在のトランザクションログのディレクトリに残存する）
  - xxはAA-ZZでカウントアップされる。
- 現在のトランザクションログのコピー（＝バックアップ）を指定したディレクトリに作成する
  - 下で示している例ではログはYYMMDDxx.logというファイル名にリネームしたものをバックアップ先にコピーします。
- 前のログファイルと同名で新しいログファイルを開始します。
- リネームの動作内容はミラー側でも反映されます。（ミラーのDBDIRにもYYMMDDxx.logは出来る）

BACKUP DATABASE文の場合

BACKUP DATABASE DIRECTORY <backup_directory> TRANSACTION LOG RENAME MATCH;

dbbackupコマンドの場合

$ dbbackup -c "dsn=SQL Anywhere 17 Demo" –t –r -n backup_directory

メリット：

- ログファイルを小さく出来る（DBミラーリング環境ではログのトランケートが不可なので肥大化し続ける）
- YYMMDDxx.logのログファイルがDBディレクトリに残るので、これをn日分保持できればミラーはn日前のフルバックアップからでも復旧出来る。

デメリット：

- 手動で古い、リネームされたログファイルは削除が必要。（ミラー側も）yymmddの日付を基準で削除する。

ミラーリング環境におけるバックアップの戦略

日に一度のフルバックアップ

dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql“ -r /<BackupDIR>/FULL/DAY/

- バックアップディレクトリにDBファイルとログファイルが出力される。
- もしこのときにログのリネームを行って、同時にログを小さくしたければ-r を付与。 **（推奨）**
  - DBディレクトリにはYYMMDDxx形式名のログファイルが作成される。バックアップディレクトリに出力されたものは元のファイル名のまま
  - -nは基本的に不要。付けた場合はリカバリー時に元のファイル名に手動で戻す必要がある。

日に数度のログバックアップ

dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql“ –t –r –n /<BackupDIR>/LOG

- バックアップディレクトリにログファイルがYYMMDDxx.logとリネームされて出力される。
- DBディレクトリにもYYMMDDxx.logが出来、ログファイルは中身がクリアされ新しくなる
- -nを付けないと元のファイル名と同じになるので出力先ディレクトリを変える必要がある。

適宜行う事項（通常フルバックアップ完了後）

- DBディレクトリに残存しているYYMMMDDxx.logファイルでフルバックアップより前の物を削除
- ミラー側にも残存しているので同様に行う
- バックアップディレクトリから不要な古いものを削除（保持要件に依存）

取得されるファイルのイメージ

ミラー側もDBDIRは同じになる。

時刻	コマンド	DBDIR	BACKUPDIR/FULL	BACKUPDIR/LOG
開始時点		Testdb1.db Testdb1.log		
2017/03/01 0:00 フルバックアップ	**dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql" -r /<BackupDIR>/FULL/20170301**	Testdb1.db **Testdb1.log(小さくなる)** **20170301AA.log**	**20170301/Testdb1.db** **20170301/Testdb1.log**	
2017/03/01 07:00 増分リネームバックアップ	**dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql" –t –r –n /<BackupDIR>/LOG**	Testdb1.db **Testdb1.log(小さくなる)** 20170301AA.log **20170301AB.log**	20170301/Testdb1.db 20170301/Testdb1.log	**20170301AB.log**
2017/03/01 17:00 増分リネームバックアップ	**dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql" –t –r –n /<BackupDIR>/LOG**	Testdb1.db **Testdb1.log(小さくなる)** 20170301AA.log 20170301AB.log **20170301AC.log**	20170301/Testdb1.db 20170301/Testdb1.log	20170301AB.log **20170301AC.log**
2017/03/02 0:00 フルバックアップ	**dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql" -r /<BackupDIR>/FULL/20170302**	Testdb1.db **Testdb1.log(小さくなる)** 20170301AA.log 20170301AB.log 20170301AC.log **20170302AA.log** この時点で消しても良い	20170301/Testdb1.db 20170301/Testdb1.log **20170302/Testdb1.db** **20170302/Testdb1.log**	
2017/03/02 07:00 増分リネームバックアップ	**dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql" –t –r –n /<BackupDIR>/LOG**	Testdb1.db **Testdb1.log(小さくなる)** 20170301AA.log 20170301AB.log 20170301AC.log 201700302AA.log **20170302AB.log**	20170301/Testdb1.db 20170301/Testdb1.log 20170302/Testdb1.db 20170302/Testdb1.log	20170301AB.log 20170301AC.log **20170302AB.log**

この状態からのリカバリー（ミラーの復旧）

通常プライマリがクラッシュするとミラーがプライマリになる。従ってこの状態から元プライマリを新ミラーとしてまず復旧する。

１．その時点でのフルバックアップ（DBファイル＋ログファイル）をミラーのディレクトリにコピー

- ディレクトリ内をクリアしてからコピー（メッセージログなどは残しても良い）

２．ミラーDBを起動する

- １サーバ８DB等で、1つのDBだけが停止状態になり、dbsrv17プロセスが残存している場合はユーティリティDBに接続してSTART DATABASE　～　MIRROR ON文で起動する。
- プライマリからログが転送され、ミラーと同期される
  - プライマリ側にミラーで使用するフルバックアップからのYYMMDDxx.logファイルが無ければならない。
  - 他の環境で単独でリカバリーを行ってログを適用したDBファイルを使用することは出来ない。（ログ適用完了で新たにログが発生するのでログが狂ってしまう）

1 では基本的に最終のフルバックアップのファイルを戻す。

時刻	コマンド	DBDIR	BACKUPDIR/FULL	BACKUPDIR/LOG
開始時点		Testdb1.db Testdb1.log		
2017/03/01 0:00 フルバックアップ	dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql" -r /<BackupDIR>/FULL/20170301	Testdb1.db Testdb1.log(小さくなる) 20170301AA.log	20170301/Testdb1.db 20170301/Testdb1.log	
2017/03/01 07:00 増分リネームバックアップ	dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql" -t -r -n /<BackupDIR>/LOG	Testdb1.db Testdb1.log(小さくなる) 20170301AA.log 20170301AB.log	20170301/Testdb1.db 20170301/Testdb1.log	20170301AB.log
2017/03/01 17:00 増分リネームバックアップ	dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql" -t -r -n /<BackupDIR>/LOG	Testdb1.db Testdb1.log(小さくなる) 20170301AA.log 20170301AB.log 20170301AC.log	20170301/Testdb1.db 20170301/Testdb1.log	20170301AB.log 20170301AC.log
2017/03/02 0:00 フルバックアップ	dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql" -r /<BackupDIR>/FULL/20170302	Testdb1.db Testdb1.log(小さくなる) 20170301AA.log 20170301AB.log 20170301AC.log 20170302AA.log	20170301/Testdb1.db 20170301/Testdb1.log 20170302/Testdb1.db 20170302/Testdb1.log	
2017/03/02 07:00 増分リネームバックアップ	dbbackup -c "HOST=localhost:10001;DBN=testdb1;UID=DBA;PWD=sql" -t -r -n /<BackupDIR>/LOG	Testdb1.db Testdb1.log(小さくなる) 20170301AA.log 20170301AB.log 20170301AC.log 201700302AA.log 20170302AB.log	20170301/Testdb1.db 20170301/Testdb1.log 20170302/Testdb1.db 20170302/Testdb1.log	20170301AB.log 20170301AC.log 20170302AB.log

プライマリのDBDIRにはその最終バックアップ時点以降のログファイルが残っていることが必要。その内容を復旧するミラーに転送してプライマリ⇔ミラーで同期が行われる。

クラッシュ時の復旧：一番確実な方法
ベストプラクティス２

ミラーを復旧させるのに一番確実な方法は初回にミラーをセットアップしたときと同様に、プライマリのフルバックアップを取得しミラー側で起動させること。

＊リカバリー手法が統一出来るので、これで運用しているパターンも多い

1. **起動しているプライマリに対し、フルバックアップを実行**

2. **バックアップしたファイルをミラー側のDBDIRに配置**
   - 残っているDBファイル・ログファイルは削除する

3. **ミラー側を起動**
   - プライマリと同期し、フルバックアップからの差分を適用してミラーリングが復帰する。

片肺状態でプライマリもクラッシュした時

片肺状態でプライマリもクラッシュし、バックアップからリストアが必要になったときは以下の方法で復旧する

1. **フルバックアップのDBファイルとログファイルを元のDBディレクトリにコピー**

2. **dbeng17 <dbfile> -ad <ログのバックアップ先>**
   - フルバックアップにログが適用されて最終バックアップ時点に復旧し、自動的に停止する
   - Dbeng17 testdbfile1.db –ad /<BackupDIR>/LOG
     - 指定したディレクトリ内のログを検索し、順に最新のログまで自動的に適用する。
     - Dbsrvでも良い。

3. **通常のコマンドで起動**
   - Dbsrv17かSTART DATABASE MIRROR ON文か

4. **“一番確実な方法”でミラーを復旧させる**

特殊パターン

以下のシナリオの場合、コミットされたトランザクションのロストが予想されるため、管理者の手で強制的に起動させることが必要。（通常の方法では起動できない）

シナリオ例：

1. **ミラーリングが確立している状態でミラー側がダウン**
2. **片肺運用状態でプライマリ側がダウン、HW的にクラッシュして起動不可**

この状態でミラー側をプライマリとして起動したいという場合

- **ミラーのままダウンしたので次回もミラーとして起動しようとする。が、プライマリはいない。**
- **ミラーのままダウンした=プライマリが単独で動いていた　なので、プライマリになってしまったらプライマリが単独で動いていた時間に発生したトランザクションはロストということになる。**
  - バックアップからリカバリしたから大丈夫ということは関係ない。

この場合はミラーを強制的にプライマリとして起動させるために、ミラー側のUtilityDBに接続し、「ALTER DATABASE ＜DB名> FORCE START; 」を実行する

1サーバー複数ミラーでの操作

このインスタンスに新規に作成したDBを追加する時

1. **DBISQL等でユーティリティDBに接続する。**
   - HOST=<Server1>:10000;DBN=Utility_DB;UID=DBA;PWD=<-suで指定した値>
   - ホストはどちらでも良い。（先に）プライマリにしたい方に接続する。

2. **START DATABASE ‘<DBFILE>’ を実行する**
   - ミラーリングさせる場合は “MIRROR ON”オプションをつける
   - このDB（だけ）に対して手順3からをセカンダリで行えばミラー環境にすることが出来る。
   - この方法でないと、DBサーバ全体を停止する必要が発生する。
   - 次回のdbsrv17での起動時の為に、起動スクリプトの修正は忘れずに。

ある「DBだけ」を停止させたい。

1. **DBISQL等でユーティリティDBに接続する。**
   - HOST=<Server1>:10000;DBN=Utility_DB;UID=DBA;PWD=<-suで指定した値>

2. **STOP DATABASE <Dbname>を実行する。**

※１DBサーバ複数DB構成ではdbstopでは全部停止することに注意

Thank you

Contact information:

F name L name
Title
Address
Phone number

F name L name
Title
Address
Phone number

© 2016 SAP SE or an SAP affiliate company. All rights reserved.