Data Guard Basics

Marco Mischke, Robotron Datenbank Software GmbH

Wenn es um Hochverfügbarkeit geht, ist der erste Gedanke meist RAC. Dieser deckt jedoch nur ein Teilgebiet ab, nämlich die Server-Redundanz. Um Desaster-Szenarien über Rechenzentrumsgrenzen hinaus abzudecken, führt an der Vorhaltung einer Standby-Datenbank und damit an Data Guard praktisch kein Weg vorbei. Der Artikel zeigt die Grundlagen, um die häufigsten Stolpersteine bereits von Beginn an umgehen zu können.

Zuerst einmal stellt sich die Frage, warum man Data Guard überhaupt benötigt. Der Real Application Cluster ist doch bereits hochverfügbar durch die Server-Redundanz und das Storage ist ebenfalls gespiegelt, sei es nun mittels ASM oder direkt im Storage selbst. Typischerweise befinden sich diese Komponenten allerdings in einem Serverraum, wenn nicht sogar im gleichen Schrank. Das trifft insbesondere auf Engineered Systems wie die Oracle Database Appliance zu.

Für die Ausfallsicherheit über Rechenzentrumsgrenzen hinweg ist ein zweites System notwendig, das gegebenenfalls die Aufgaben übernehmen kann. An diesem Punkt kommt Data Guard ins Spiel. Man hält auf einem entfernten System eine Kopie der Produktiv-Datenbank durch permanentes Nachspielen der in den Redologs aufgezeichneten Transaktionen auf dem aktuellen Stand. Diese Umgebung mit Primär- und Standby-Datenbank wird durch Data Guard verwaltet und ermöglicht geplantes sowie ungeplantes Umschalten zwischen beiden Datenbanken. Im Desaster-Fall lässt sich somit innerhalb von Minuten die volle Funktionalität durch die Aktivierung des Standby-Systems wiederherstellen.

Namensgebung

Bei der Einrichtung einer Data-Guard-Umgebung ist einiges zu beachten. Das beginnt bereits bei der Namensgebung. Dazu muss man wissen, welche Namen und Parameter es überhaupt gibt *(siehe Tabelle 1)*.

In einer Data-Guard-Umgebung hat die Standby-Datenbank den gleichen „db_name" wie die Primär-Datenbank, da sie eine byteweise Kopie ist. Die Unterscheidung der beiden Systeme erfolgt daher über den „db_unique_name". Oftmals werden die Rollen in den „db_unique_name" eingebaut, also beispielsweise „DBPRIM" und „DBSTBY". Es ist aber zu beachten, dass man die Rollen auch tauschen kann – dann sind diese Namen irreführend. Man sollte daher die Namen unbedingt unabhängig von der Rolle festlegen und stattdessen ein Kürzel zur Identifizierung des Rechenzentrums verwenden, also „DBRZ1" und „DBRZ2". Der Service-Name wird dann entsprechend unabhängig von Rollen etc. vergeben, um das Konstrukt vor den Client-Anwendungen zu maskieren. *Abbildung 1* zeigt die Namensgebung in einem Data-Guard-Verbund mit zwei RAC-Systemen.

Diese Art der Namensgebung sollte also von Anfang an konsequent verfolgt werden, auch wenn man zu Beginn noch keinen Data Guard verwenden will. Man ist jedoch bereits für zukünftige Entwicklungen gerüstet. Weiterhin empfiehlt es sich, die Namen in Großschreibung zu verwenden, insbesondere bei Benutzung von Oracle Managed Files (OMF). Für einige Pfade werden die Namen automatisch in Großschreibung umgewandelt, andere nicht, sodass man mit einer generellen Großschreibung für mehr Übersicht sorgen kann.

Parameter	Bedeutung	Data Guard relevant
db_name	Formt aus Datendateien die Datenbank	Ja
instance_name	Identifiziert die Prozesse	Nein
db_unique_name	Eindeutiger Name der Datenbank zur Identifizierung	Ja
service_name	Abstraktionsschicht für Applikationen	Ja
ORACLE_SID	Ermittelt das Parameter-File sowie die Passwort-Datei beim Instance-Start	Nein
global_name	Globaler Name der Datenbank (für Datenbank-Links)	Nein

Tabelle 1

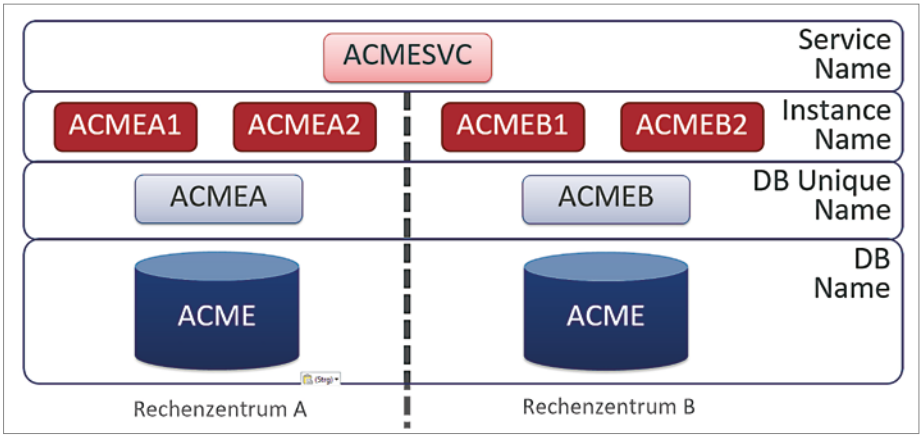

Abbildung 1: Beispielhafte Namensgebung

```
SQL> shutdown immediate
SQL> startup mount
SQL> alter database archivelog;
SQL> alter database open;
SQL> alter database force logging;
SQL> alter database flashback on;
```

Listing 1

```
SQL> alter database add standby logfile thread 1 group 10 size 500M;
SQL> alter database add standby logfile thread 1 group 11 size 500M;
SQL> alter database add standby logfile thread 1 group 12 size 500M;
SQL> alter database add standby logfile thread 1 group 13 size 500M;
```

Listing 2

```
SID_LIST_LISTENER =
  (SID_LIST =
      (SID_DESC =
          (GLOBAL_DBNAME = db12cb.robotron.de)
          (ORACLE_HOME = /u01/app/oracle/product/12.1.0.2/db)
          (SID_NAME= db12cb)
      )
  )
```

Listing 3

Vorbereitung der Quell-Datenbank

Bevor man nun eine Data-Guard-Umgebung aufbaut, bereitet man die Quell-Datenbank soweit es geht vor, um den Aufwand möglichst gering zu halten. Die Parameter zur Namensgebung wurden bereits erläutert. Dazu kommen noch das Aktivieren des Archivelog-Modus und vor allem das Erzwingen des Logging. Ansonsten können durch Nologging-Operationen Informationen zu Transaktionen an den Redologs vorbei geschrieben werden,

```
connect target sys/oracle@db12ca
connect auxiliary sys/oracle@db12cb

startup clone nomount

duplicate target database for standby
from active database
spfile
   set db_name 'db12c'
   set db_unique_name 'db12cb '
   set db_create_file_dest '/u01/app/oracle/oradata '
   set db_create_online_log_dest_1 '/u01/app/oracle/oradata '
   set db_recovery_file_dest '/u01/app/oracle/fra '
   set control_files ' '
   set diagnostic_dest '/u01/app/oracle '
   set audit_file_dest '/u01/app/oracle/admin/db12cb/adump '
   set dg_broker_start 'FALSE '
   set filesystemio_options 'ASYNCH '
   reset log_archive_config
   reset log_archive_dest_1
   reset log_archive_dest_2
   reset fal_server
   reset fal_client
   reset db_domain
   reset memory_target
   set sga_target '1200M '
   set pga_aggregate_target '200M '
noresume
dorecover
;
```

Listing 4

Parameter	Wert	Bedeutung
standbyfile_management	auto	Datendateien, die auf der primären Datenbank erzeugt werden, werden automatisch auch auf der Standby-Datenbank angelegt
db_block_checking	medium	Führt logische Checks der Datenbank-Blöcke durch (bei „full“ auch für Index-Blöcke mit Performance Overhead)
db_block_checksum	typical	Berechnet vor dem Schreiben die Check-Summen für Daten- und Redo-Blöcke
db_lost_write_protect	typical	Zeichnet Cache-Reads in den Redologs auf, um SCNs vergleichen zu können

Tabelle 2

```
SQL> select force_logging, log_mode, flashback_on from v$database;

FORCE_LOGGING    LOG_MODE       FLASHBACK_ON
---------------  ------------   ------------
YES              ARCHIVELOG     NO
```

Listing 5

```
SQL> alter system set dg_broker_config_file1='/u01/app/oracle/oradata/DB12CA/dr1db12ca.dat';
SQL> alter system set dg_broker_config_file2='/u01/app/oracle/fra/DB12CA/dr2db12ca.dat';
SQL> alter system set dg_broker_start=true;
```

Listing 6

was ein Recovery unmöglich macht. Um ein Re-Instanziieren nach einem Failover zu vereinfachen, aktiviert man weiterhin Flashback Database *(siehe Listing 1)*.

Um die Standby-Datenbank möglichst synchron zur Primär-Datenbank halten zu können, sind Standby-Redologs erforderlich. Die Primär-Datenbank schreibt dann parallel zu den eigenen Redologs die Transaktionen auch in die Standby-Redologs der Standby-Datenbank. Diese kann die Transaktionen dann direkt einspielen und nicht erst beim Archivieren eines Redologs auf der Primärseite.

Es empfiehlt sich, eine Gruppe mehr an Standby-Redologs anzulegen, als es Online-Redolog-Gruppen gibt. Im Fall von RAC muss man selbstverständlich für jeden Thread entsprechende Gruppen anlegen. Am besten verwendet man einen separaten Zahlenbereich für die Gruppennummer, um die Dateien direkt unterscheiden zu können *(siehe Listing 2)*. Zur maximalen Sicherheit gegenüber Datenverlust sind noch einige weitere Parameter relevant, die in *Tabelle 2* aufgelistet sind.

Anlegen der Standby-Datenbank

Ist die Quell-Datenbank soweit vorbereitet, wird die zukünftige Standby-Datenbank per „RMAN DUPLICATE“ als Kopie erstellt. Auf dem Standby-Server müssen dazu lediglich die Datenbank-Software installiert und das Verzeichnis für „audit_file_dest“ angelegt sein. Um sich initial an der Zieldatenbank auch remote anmelden zu können, ist per „orapwd file=$ORACLE_HOME/dbs/orapwdb12cb password=oracle force=y“ ein Passwort-File anzulegen. Zudem benötigt man einen statischen Listener-Eintrag in der „listener.ora“ *(siehe Listing 3)*.

Nun kann das Duplikat gestartet werden, dabei setzt man am besten gleich alle Parameter mit den passenden Werten oder setzt diese zurück, um mögliche Probleme direkt auszuschließen *(siehe Listing 4)*. Überprüft man nach erfolgreicher

Duplizierung nun den Zustand der Kopie, so stellt man fest, dass alle Einstellungen übernommen wurden, mit Ausnahme des Flashback-Database *(siehe Listing 5)*. Flashback Database ist also auf der Kopie nochmals separat zu aktivieren.

Data-Guard-Konfiguration

Da die Quell- und Ziel-Datenbank nun vorhanden ist, kann man mit der Einrichtung des Data Guard beginnen. Die Konfiguration wird in zwei gespiegelten Dateien verwaltet, ähnlich den Kontroll-Dateien der Datenbank. Den Namen und Speicherort dieser Dateien legt man entsprechend über Initialisierungsparameter fest. Die eigentliche Arbeit in einer Data-Guard-Umgebung übernimmt der Data-Guard-Broker-Prozess, den man durch Initialisierungsparameter startet *(siehe Listing 6)*.

Natürlich muss man das sowohl auf der Primär- wie auch auf der Standby-Datenbank einstellen. Anschließend kann die Konfiguration per „dgmgrl" erstellt werden. Man verwendet für die Verbindung zu Primär- und Standby-Datenbank am besten die Easy-Connect- Syntax, da man so unabhängig von Einstellungen und Fehlern in der tnsnames.ora oder Ähnlichem ist: „DGMGRL> connect sys/oracle@"oel6u4:1521/db12ca.robotron.de"". Initial wird die Konfiguration mit einer Primär-Datenbank erstellt *(siehe Listing 7)*, als Nächstes fügt man die Standby-Datenbank hinzu *(siehe Listing 8)*.

Nun stellt man den Redolog-Transport in beide Richtungen auf „synchrone Übertragung", um die Transaktionen möglichst ohne Zeitverzug in die Standby-Redologs zu übertragen *(siehe Listing 9)*.

Als Letztes stellt man noch den Protection Mode auf die gewünschte Stufe. Typischerweise ist das „Maximum Availability", dabei werden die Transaktionen nach Möglichkeit synchron übertragen, die Primär-Datenbank stoppt aber nicht, wenn die Standby-Datenbank gerade einmal nicht erreichbar ist. Die Konfiguration ist damit vollständig und kann per „DGMGRL> enable configuration" aktiviert werden. Ab diesem Moment überträgt die Primär-Datenbank ihren Redolog-Stream zur Standby-Datenbank, wo die Transaktionen direkt eingespielt werden. Den Status kann man nach einigen Sekunden entsprechend überprüfen *(siehe Listing 10)*. Damit ist die Konfiguration abgeschlossen.

```
DGMGRL> create configuration db12c as
> primary database is db12ca
> connect identifier is 'oel6u4:1521/db12ca.robotron.de';
```

Listing 7

```
DGMGRL> add database db12cb as
> connect identifier is 'oel6u4:1521/db12cb.robotron.de'
> maintained as physical;
```

Listing 8

```
DGMGRL> edit database db12ca set property logxptmode='SYNC';
DGMGRL> edit database db12cb set property logxptmode='SYNC';
```

Listing 9

```
DGMGRL> show configuration verbose

Configuration - db12c

  Protection Mode: MaxPerformance
  Members:
  db12ca - Primary database
      db12cb - Physical standby database

  Properties:
      FastStartFailoverThreshold       = '30'
      OperationTimeout                 = '30'
      TraceLevel                       = 'USER'
      FastStartFailoverLagLimit        = '30'
      CommunicationTimeout             = '180'
      ObserverReconnect                = '0'
      FastStartFailoverAutoReinstate   = 'TRUE'
      FastStartFailoverPmyShutdown     = 'TRUE'
      BystandersFollowRoleChange       = 'ALL'
      ObserverOverride                 = 'FALSE'
      ExternalDestination1             = ''
      ExternalDestination2             = ''
      PrimaryLostWriteAction           = 'CONTINUE'

Fast-Start Failover: DISABLED

Configuration Status:
SUCCESS
```

Listing 10

```
srvctl add database -db db12ca -dbname db12c \
  -oraclehome /u01/app/oracle/product/12.1.0.2/db \
  -domain robotron.de \
  -spfile /u01/app/oracle/product/12.1.0.2/db/dbs/spfiledb12ca.ora \
  -pwfile /u01/app/oracle/product/12.1.0.2/db/dbs/orapwdb12ca \
  -role PRIMARY -startoption open -stopoption immediate
```

Listing 11

Betriebsthemen

Um die beteiligten Datenbanken immer automatisch mit dem Server zu starten und besonders um die korrekten Start-Optionen zu gewährleisten, sollte die Oracle Grid Infrastructure verwendet werden. Auch wenn weder Cluster noch

```
srvctl add database -db db12cb -dbname db12c -oraclehome /u01/app/oracle/product/12.1.0.2/db \
   -domain robotron.de \
   -spfile /u01/app/oracle/product/12.1.0.2/db/dbs/spfiledb12cb.ora \
   -pwfile /u01/app/oracle/product/12.1.0.2/db/dbs/orapwdb12cb \
   -role PHYSICAL_STANDBY -startoption mount -stopoption immediate
```

Listing 12

```
srvctl add service -db <unique name> -service <service name> \
   -role primary -failovertype SELECT \
   -failovermethod BASIC -cardinality uniform \
   -failoverdelay 180 -failoverretry 5
```

Listing 13

```
<TNS alias>=
   (DESCRIPTION_LIST=
       (LOAD_BALANCE=off)
       (FAILOVER=on)
       (DESCRIPTION=
           (CONNECT_TIMEOUT=5)
           (TRANSPORT_CONNECT_TIMEOUT=3)
           (RETRY_COUNT=3)
           (ADDRESS_LIST=
               (LOAD_BALANCE=on)
               (ADDRESS=(PROTOCOL=TCP)
                   (HOST=<primary scan>)(PORT=1521))
       )
           (CONNECT_DATA=(SERVICE_NAME=<service name>))
       )
       (DESCRIPTION=
           (CONNECT_TIMEOUT=5)
           (TRANSPORT_CONNECT_TIMEOUT=3)
           (RETRY_COUNT=3)
           (ADDRESS_LIST=
               (LOAD_BALANCE=on)
               (ADDRESS=(PROTOCOL=TCP)
                   (HOST=<standby scan>)(PORT=1521))
       )
           (CONNECT_DATA=(SERVICE_NAME=<service name>))
   )
   )
```

Listing 14

ASM zum Einsatz kommen sollen, ergibt das durchaus Sinn, denn über die Datenbank-Ressourcen in der Grid Infrastructure kann der richtige Modus zum Starten definiert werden.

Wer keine Lizenz für Active Data Guard besitzt, darf die Standby-Datenbank nur im „MOUNT" starten. Per Standard würde die Datenbank aber „Read only" geöffnet werden, was einen Lizenzverstoß bedeutet. Man legt also die Datenbank-Ressourcen mit den passenden Einstellungen an, einmal für die Primär- *(siehe Listing 11)* und einmal für die Standby-Datenbank *(siehe Listing 12)*.

Bei geplanten oder ungeplanten Umschaltungen benutzt der Data Guard Broker dann die Grid Infrastructure zum Stoppen und Starten der Datenbanken. Das erspart zum einen gesonderte Konfigurationen für SQL*Net und zum anderen passt der Broker die Start-Optionen mit an. Bei einem Rollentausch startet also auch die neue Standby-Datenbank nur im „MOUNT".

Um die Data-Guard-Umgebung für die Applikationen unabhängig von der Rollenverteilung zugänglich zu machen, müssen sich die Applikationen über einen separat einzurichtenden Service verbinden. Dieser wird in der Grid Infrastructure auf beiden Seiten eingerichtet und startet abhängig von der Rolle der Datenbank. Zusätzlich kann der Service noch verschiedene Einstellungen für das Failover enthalten *(siehe Listing 13)*.

Diese Definition bewirkt, dass der Service nur startet, wenn die zugrunde liegende Datenbank die Rolle „PRIMARY" hat. Für die Applikationen muss man nun nur noch einen entsprechenden TNS-Alias einrichten, der mit einer Description-Liste beide Seiten im Data-Guard-Verbund definiert *(siehe Listing 14)*.

Benutzen die Applikationen nun diesen TNS-Alias, wird die Verbindung immer mit der aktuellen Primär-Datenbank hergestellt, ohne dass irgendwelche Anpassungen erforderlich sind.

Fazit

Mit Data Guard wird eine komplexe Technologie durch einfache Werkzeuge handhabbar. Man kann damit seine Datenbanken auf einfache Art und Weise auf ein anderes System spiegeln und dort aktuell halten. Im Desasterfall, also dem kompletten Ausfall des Datenbank-Systems, braucht man dann lediglich ein einziges Kommando, um die Standby-Datenbank zu aktivieren und weiterarbeiten zu können.

Marco Mischke
marco.mischke@robotron.de