Data Guard Basics

Autor(en): Marco Mischke, Robotron Datenbank Software GmbH

Wenn es um Hochverfügbarkeit geht, ist der erste Gedanke meist RAC. Dieser deckt jedoch nur ein Teilgebiet, nämlich die Serverredundanz, ab. Um Desaster Szenarien über Rechenzentrumsgrenzen abzudecken, führt an der Vorhaltung einer Standby Datenbank und damit an Data Guard praktisch kein Weg vorbei. Folgender Artikel soll die Grundlagen erklären 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 Serverredundanz und das Storage ist ebenfalls gespiegelt, sei es nun mittels ASM oder direkt im Storage selbst. Aber typischerweise befinden sich diese Komponenten in einem Serverraum wenn nicht sogar im gleichen Schrank. Das trifft insbesondere auf die Engineered Systems wie die Oracle Database Appliance zu. Man benötigt für eine Ausfallsicherheit über Rechenzentrumsgrenzen hinweg ein zweites System, dass gegebenenfalls die Aufgaben übernehmen kann. An diesem Punkt kommt Data Guard ins Spiel. Man hält auf einem entfernten System eine Kopie der Produktivdatenbank durch permanentes Nachspielen der in den Redologs aufgezeichneten Transaktionen auf dem aktuellen Stand. Diese Umgebung mit Primär- und Standbydatenbank wird durch Data Guard verwaltet und ermöglicht geplante sowie ungeplante Umschaltungen zwischen beiden Datenbanken. Im Desasterfall kann also innerhalb von Minuten die volle Funktion durch die Aktivierung des Standby-Systems wiederhergestellt werden.

Namensgebung

Bei der Einrichtung einer DataGuard Umgebung muss einiges beachtet werden, das beginnt bereits bei der Namensgebung. Dazu muss man welche Namen und Parameter es überhaupt gibt.

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 Parameterfile sowie Passwortdatei beim Instance Start	Nein
global_name	Globaler Name der Datenbank (für Datenbanklinks)	Nein

In einer Data Guard Umgebung hat die Standbydatenbank den gleichen db_name wie die Primärdatenbank, 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 z.B. „DBPRIM“ und „DBSTBY“. Man muss aber 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 Servicename wird dann entsprechend unabhängig von Rollen usw. vergeben um das Konstrukt vor den Clientanwendungen zu maskieren. Abbildung 1 verdeutlicht die Namensgebung in einem Data Guard Verbund mit zwei RAC Systemen.

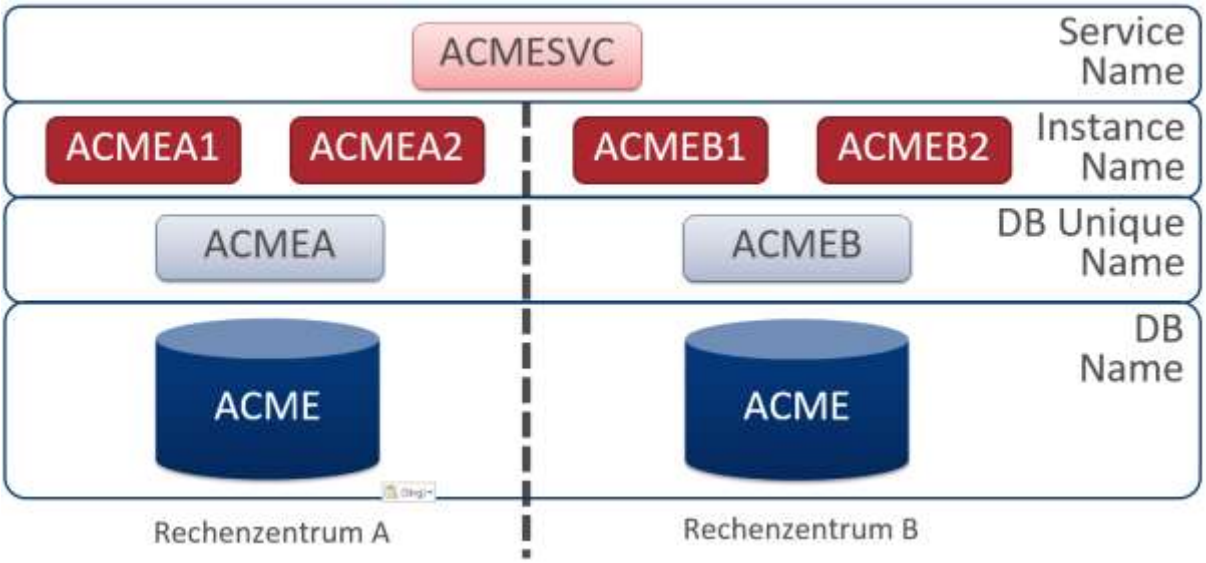

Abbildung 1: Beispielhafte Namensgebung

Diese Art der Namensgebung sollte also von Anfang konsequent verfolgt werden, auch wenn man zu Beginn noch keinen Data Guard verwenden will. Man ist aber für

zukünftige Entwicklungen gerüstet. Weiterhin empfiehlt es sich, die Namen in Großschreibung zu verwenden, insbesondere bei Verwendung von Oracle Managed Files (OMF). Für einige Pfade werden die Namen automatisch in Großschreibung umgewandelt, andere nicht, so dass man mit einer generellen Großschreibung für mehr Übersicht sorgen kann.

Vorbereitung der Quelldatenbank

Bevor man nun eine Data Guard Umgebung aufbaut, bereitet man die Quelldatenbank soweit es geht vor um den Aufwand möglichst gering zu halten. Die Parameter zur Namensgebung wurden bereits erläutert. Dazu kommt 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 wodurch ein Recovery unmöglich wird. Um ein Reinstanziieren nach einem Failover zu vereinfachen, aktiviert man weiterhin Flashback Database.

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

Um die Standbydatenbank möglichst synchron zur Primärdatenbank halten zu können, werden Standby Redologs benötigt. Die Primärdatenbank kann dann parallel zu den eigenen Redologs die Transaktionen auch in die Standby Redologs der Standbydatenbank schreiben. Die Standbydatenbank kann die Transaktionen dann direkt einspielen und nicht erst beim Archivieren eines Redologs auf der Primärseite. Die Empfehlung ist, eine Gruppe mehr an Standby Redologs anzulegen, als es Online Redolog Gruppen gibt. Im Falle 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.

```
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;
```

Um maximale Sicherheit gegenüber Datenverlust zu erhalten, sind noch einige weitere Parameter relevant, die in folgender Tabelle aufgelistet sind.

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

Anlegen der Standby Datenbank

Ist die Quelldatenbank nun soweit vorbereitet, kann die zukünftige Standbydatenbank per RMAN DUPLICATE als Kopie erstellt werden. Auf dem Standby Server muss dazu lediglich die Datenbanksoftware installiert sein und das Verzeichnis für audit_file_dest angelegt werden. Um sich initial an der Zieldatenbank auch remote anmelden zu können, muss man ein Passwordfile anlegen:

```
orapwd  file=$ORACLE_HOME/dbs/orapwdb12cb password=oracle
force=y
```

Und man benötigt einen statischen Listener Eintrag in der listener.ora:

```
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)
    )
  )
```

Nun kann das Duplicate gestartet werden, dabei setzt man am besten gleich alle Parameter mit den passenden Werten oder setzt diese zurück, um mögliche Probleme direkt auszuschließen.

```
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
;
```

Überprüft man nach erfolgreicher Duplizierung nun den Zustand der Kopie, so stellt man fest, dass alle Einstellungen übernommen wurden mit Ausnahme des Flashback Database.

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

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

Flashback Database muss man also auf der Kopie nochmals separat aktivieren.

Data Guard Konfiguration

Nun, da die Quell- und Zieldatenbank vorhanden ist, kann man mit der Einrichtung des Data Guard beginnen. Die Konfiguration wird in zwei gespiegelten Dateien verwaltet, ähnlich den Kontrolldateien 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, auch diesen muss man mittels Initialisierungsparameter starten.

```
SQL> alter system set
dg_broker_config_file1='/u01/app/oracle/oradata/DB12CA/dr1db12
ca.dat';
SQL> alter system set
dg_broker_config_file2='/u01/app/oracle/fra/DB12CA/dr2db12ca.d
at';
SQL> alter system set dg_broker_start=true;
```

Natürlich muss man das Sowohl auf der Primär- wie auch auf der Standbydatenbank einstellen.
Nun kann die Konfiguration mittels „dgmgrl“ erstellt werden. Man verwendet für die Verbindung zu Primär- und Standbydatenbank 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 Primärdatenbank erstellt:

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

Als nächstes fügt man die Standbydatenbank hinzu:

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

Nun stellt man den Redolog Transport in beide Richtungen auf synchrone Übertragung um die Transaktionen möglichst ohne Zeitverzug in die Standby Redologs zu übertragen:

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

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 Primärdatenbank stoppt aber nicht, wenn die Standbydatenbank gerade einmal nicht erreichbar ist.

```
DGMGRL> edit configuration set protection mode as
MaxAvailability;
```

Die Konfiguration ist damit vollständig und kann aktiviert werden.

```
DGMGRL> enable configuration
```

Ab diesem Moment überträgt die Primärdatenbank ihren Redolog Stream zur Standbydatenbank wo die Transaktionen direkt eingespielt werden. Den Status kann man nach einigen Sekunden entsprechend überprüfen.

```
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
```

Damit ist die Konfiguration abgeschlossen.

Betriebsthemen

Um die beteiligten Datenbanken immer automatisch mit dem Server zu starten und besonders, um die korrekten Startoptionen zu gewährleisten, sollte die Oracle Grid Infrastructure verwendet werden. Auch wenn weder Cluster noch ASM zum Einsatz kommen sollen, macht das durchaus Sinn. Denn über die Datenbankressourcen in der Grid Infrastructure kann der richtige Modus zum Starten definiert werden. Hat man keine Lizenz für Active Data Guard, so darf man die Standbydatenbank nur im MOUNT starten. Per Standard würde die Datenbank aber Read Only geöffnet werden, was einen Lizenzverstoß bedeuten würde. Man legt also die Datenbankressourcen mit den passenden Einstellungen an, einmal für die Primärdatenbank:

```
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
```

Und einmal für die Standbydatenbank:

```
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
```

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 Startoptionen mit an. Bei einem Rollentausch startet also auch die neue Standbydatenbank nur im MOUNT.

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

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

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 enstprechenden TNS Alias eingerichten, der mit einer Description List beide Seiten im Data Guard Verbund definiert.

```
<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>))
    )
  )
```

Benutzen die Applikationen nun diesen TNS Alias, dann wird die Verbindung immer mit der aktuellen Primärdatenbank hergestellt ohne dass irgendwelche Anpassungen vorgenommen werden müssen.

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 Datenbanksystems, braucht man dann lediglich ein einziges Kommando um die Standbydatenbank zu aktivieren und weiterarbeiten zu können.

Kontakt:

Marco Mischke

marco.mischke@robotron.de