Datenbankreplikation in der Standard Edition

Markus Jendrossek
GKL Gemeinsame Klassenlotterie der Länder
Standort Hamburg

Schlüsselworte

Standard Edition, Replikation, Standby, Bash

Einleitung

Oracle Data Guard bietet eine Reihe von Werkzeugen, die das Einrichten, Administrieren und Monitoren von Physical-Standby-Datenbanken ermöglichen. Dadurch kann ein Datenverlust im Katastrophenfall oder bei Datenkorruption verhindert werden. Bei entsprechender Konfiguration kann die Replikation auch mit Zeitversatz durchgeführt werden. Dadurch kann auf einen älteren Datenstand zurückgegriffen werden.

So viele Vorteile Data Guard bietet, hat es doch einen entscheidenden Nachteil: das Werkzeug ist Bestandteil der Oracle Enterprise Edition und daher nur gegen Aufpreis erhältlich. Zum Glück ist die Basisfunktionalität der Standby-Datenbank bereits in der Standard Edition enthalten und kann dort ebenfalls eingesetzt werden.

Das hier beschriebene Konzept basiert darauf, regelmäßig Archivelogs von der Quelldatenbank auf die Zieldatenbank zu schieben und dort mittels rman einzuspielen. Das ganze wird über selbst geschriebene Skripte gesteuert.

Einrichtung

Vorbereitung Quellumgebung

In einer RAC-Installation wird als erstes eine gesharte Archivelog-Destination benötigt. NFS fällt aus, da das Schreiben von Archivelogs ebenfalls nur in der Enterprise Edition gestattet ist.
Alternativ lässt sich ein ACFS-share einrichten. Das hat zudem den Vorteil, dass ACFS über die normalerweise sowieso installierte Oracle Clusterware einzurichten ist, also keinen durch Drittsoftware erzeugten Overhead erzeugt.

Es folgen nun die einzelnen Schritte zur Einrichtung eines ACFS-shares. Es wird angenommen, dass die Clusterware inklusive ASM Treibern bereits installiert und mit dem aktuellsten PSU versehen ist.

Neue ASM Disk erstellen

```
[root@pontius ~]# fdisk -uc /dev/sdc
[root@pontius ~]# oracleasm createdisk ACFS /dev/sdc1
```

(bei RAC auf übrigen Knoten danach “oracleasm scandisks” ausführen)

An ASM Instanz anmelden (sys as sysasm) und eine Diskgroup erstellen

```
SQL> create diskgroup acfs_dg external redundancy disk 'ORCL:ACFS';
SQL> alter diskgroup acfs_dg set attribute ‘compatible.asm'='12.1.0.0.0';
SQL> alter diskgroup acfs_dg mount;
```

ACFS Volume erstellen

```
ASMCMD> volcreate —G acfs_dg -s 4G acfs_vol
ASMCMD> volinfo -G acfs_dg acfs_vol
    Diskgroup Name: ACFS_DG
```

```
        Volume Name: ACFS_VOL
        Volume Device: /dev/asm/acfs_vol-457
```

Dateisystem erstellen und mounten

```
[root@pontius ~]# mkdir /acfs
[root@pontius ~]# mkfs -t acfs /dev/asm/acfs_vol-457
[root@pontius ~]$ acfsutil registry -a /dev/asm/acfs_vol-457 /acfs/
```

Alternativ können die vorigen Schritte auch grafisch mit dem asmca durchgeführt werden.

Dann muss nur noch der Datenbank gesagt werden, dass sie die Archivelogs auch auf das ACFS schreiben soll.

```
SQL> alter system set log_archive_dest_2='LOCATION=/acfs' scope=both;
```

Zudem sollte sichergestellt sein, dass der user oracle sich ohne Passworteingabe von der Quell- auf die Zielumgebung per SSH verbinden kann. Das kann durch Erstellen eines SSH Schlüsselpaares erreicht werden. Auch die DNS Auflösung sollte funktionieren.

Wenn auf der Quellumgebung ein regelmäßig rman-Backup läuft, das auch Archivelogs löscht, sollten die entsprechenden rman-Skripte angepasst werden. Ansonsten kann es passieren, dass das Backup Archivelogs aus der ACFS Umgebung löscht, die von der Replikation noch benötigt werden.

Vorbereitung Zielumgebung

Als Zielumgebung wird hier eine single instance genutzt. Für RAC sollte das Vorgehen aber im Wesentlichen identisch sein.
Auf der Zielseite wird als erstes die Clusterware-Software installiert. Die sollte mindestens den selben Versionsstand haben, wie die Quellumgebung.

Dann wird ein Backup der Quelldatenbank erstellt (pfile, controlfile, datafiles)

```
SQL> create pfile='/tmp/pfile_pondb' from spfile;
SQL> alter database create standby controlfile as '/tmp/control_pondb';
rman> backup database plus archivelog;
```

Die daraus resultierenden Dateien werden auf die Zielumgebung kopiert und dann daraus die Zieldatenbank restored.

Im pfile werden dafür zunächst folgende Parameter entfernt: „instance_number“, „control_files“, „cluster_database“ und „log_archive_dest_2“.
Das pfile dann unter $ORACLE_HOME/dbs/initpondb1.ora ablegen.

Dann das controlfile restoren

```
[oracle@pilatus ~]$ export ORACLE_SID=pondb1

RMAN> startup nomount
RMAN> restore controlfile from '/tmp/control_pondb';
```

Der rman gibt die Namen der wiederhergestellten Dateien aus. Damit kann man dann im pfile den Parameter “control_files” wieder anlegen und die Instanz mounten.

```
RMAN> alter database mount;
```

Danach dann die Datafiles restoren

```
RMAN> catalog start with '/rman/';
RMAN> restore database;
RMAN> recover database;
```

Datenbank an dieser Stelle **nicht** öffnen

```
SQL> shutdown immediate;
SQL> startup nomount;
SQL> alter database mount standby database;
```

Zum Schluss braucht man noch ein Verzeichnis, in das die Archivelogs von der Quelle kopiert werden sollen.

```
[root@pilatus ~]$ mkdir /replicate
[root@pilatus ~]$ chown -R oracle:oinstall /replicate
```

Replikation manuell durchführen

Die Replikation lässt sich durch die folgenden Befehle manuell sehr einfach durchführen.

```
[oracle@pontius ~]$ scp /acfs/pondb/*.dbf oracle@pilatus:/replicate/

[oracle@pilatus ~]$ rman
RMAN> connect target
RMAN> catalog start with '/replicate';
RMAN> recover database;
```

Da man das aber regelmäßig tun sollte, ist es sinnvoll, das Ganze über Skripte zu steuern.

Replikation skriptgesteuert durchführen

im beiliegenden zip-File liegen fünf Dateien, die auf die Ziel- bzw. Quellumgebung abgelegt werden. Die Pfade sind beliebig, jedoch sollten die Skripte sich alle im selben Verzeichnis befinden.
Angepasst werden muss nur die parameter_<sid> Datei. Sollen mehrere Datenbanken repliziert werden, muss für jede Datenbank solch eine Datei angelegt und konfiguriert werden.

Aufgerufen wird das ganze als user oracle durch das Wrapperscript:

```
[oracle@pontius ~]$ cd /u01/scripts/
[oracle@pontius scripts]$ ./replicate_db.sh
Thu Aug 24 08:11:31 CEST 2017: start replication of databases  pondb
Thu Aug 24 08:11:35 CEST 2017: databases pondb  have been replicated
succesfully
```

Das Skript sucht nach parameter_<sid> Dateien im lokalen Verzeichnis und ermittelt so, welche Datenbanken repliziert werden sollen. Danach ruft es für jede Datenbank `copy_arclog.sh` auf.
Dieses Skript verbindet sich mit der Zielumgebung und ruft dort für jeden Archivethread `get_last_seq.sh` auf, um die Sequenz des zuletzt eingespielten Archivelogs je Thread zu ermitteln. Dann macht es auf der Quellseite einen Logswitch. Danach sucht es nach dem Archivelog mit der nächst höheren Sequenznummer und kopiert dieses auf die Zielumgebung. Das passiert in einer Schleife so lange, bis keine Archivelogs mehr gefunden werden.
Danach wird ein rman-Skript erstellt, auf die Zielumgebung kopiert und schließlich remote über den Aufruf von `apply_arclog.sh` ausgeführt und somit die Archivelogs mit rman eingespielt. Das rman-Skript enthält die SCN, bis zu der die Datenbank mit den soeben kopierten Archivelogs restored werden kann.
Zum Schluss werden Archivelogs aus der ACFS-Umgebung auf der Querseite sowie auf der Zielseite gelöscht.

Auf der Quellseite wird zudem ein Unterordnen `logs` erstellt. Dort loggt das Skript nach `replicate_db_<sid>-current.log` den aktuellsten Lauf. Am Ende wird der Inhalt von current an `replicate_sb_<sid>-<datum>.log` angehängt. Somit gibt es für jeden Tag ein Logfile, das dann alle Läufe des jeweiligen Tages enthält.

Um eine regelmäßige Replikation durchzuführen, muss einfach nur `replicate_db.sh` per cronjob ausgeführt werden. Im RAC ist eine Ausführung als externes Programm des dbms_scheduler besser, dann ist sichergestellt, dass die DB auf der Instanz auch gerade läuft.

Monitoring

Es ist jederzeit möglich, die Zieldatenbank read only zu öffnen.

```
SQL> alter database open read only;
```

Dann kann man sich einloggen und den Datenstand abfragen. Währenddessen funktioniert die Replikation natürlich nicht. Um diese wieder zum Laufen zu bringen, die Datenbank einfach wieder im „mount standby status" starten.

```
SQL> shutdown immediate;
SQL> startup nomount;
SQL> alter database mount standby database;
```

Abfragen gegen das Data Dictionary funktionieren auch im „mount standby status". Um zum Beispiel zu ermitteln, wie weit die Standby-Datenbank aktuell (in Tagen) hinterher hängt, kann man dieses SQL auf dem Ziel ausführen.

```
SQL> select (sysdate-max(checkpoint_time)) from v$datafile_header;
```

Troubleshooting

Datenbank Patches

Das Einspielen von PSU erfolgt nach der von oracle empfohlenen Methode “standby first”.

Als erstes sollte die Replikation angehalten werden (also zB den cronjob auskommentieren). Danach auf der Zielseite die Clusterware herunterfahren und das Grid-PSU wie gewohnt mit `opatchauto apply` einspielen. Das Datenbank-PSU wird dann nicht mit opatchauto, sondern mit `opatch apply` eingespielt. Dadurch werden keine Änderungen an der Datenbank vorgenommen, sondern nur die Binaries gepatcht.
Wenn die Patches eingespielt sind, die Datenbank in „mount standby" bringen. Jetzt kann die Replikation wieder gestartet werden.

Für das Einspielen auf der Quellseite muss beachtet werden, dass hier ACFS genutzt wird und daher der entsprechende Abschnitt in der Dokumentation des PSU befolgt werden muss.
Dann sowohl das Grid- aus auch das DB-PSU mit `opatchauto apply` einspielen.
Dadurch wird auf der Quellseite der Patch auch in die Datenbank selber eingespielt (was auf der Zielseite übersprungen wurde). Wenn nun die Replikation wieder gestartet wird, werden die Änderungen am Data Dictionary mit auf die Zielseite repliziert.

Datafile hinzufügen

Legt man auf der Quellseite einen neuen Tablespace an (oder fügt ein Datafile an einen bestehenden hinzu), erhält man bei der Replikation eine Fehlermeldung

```
RMAN-11003: failure during parse/execution of SQL statement: alter
database recover logfile '/replication/1_106_929286367.dbf'
ORA-00283: Recovery Session wegen Fehlern abgebrochen
ORA-01274: Datendatei, die ursprunglich als "/u01/app/oracle/
oradata/pondb/myts.dbf" erstellt wurde, kann nicht hinzugefugt
werden
```

Oracle bietet dazu in der Enterprise Edition den Parameter `standby_file_management=AUTO`. Dadurch würde rman das neue Datafile der in der Standart Edition nicht erlaubt ist, muss man das Datafile auf der Zielseite manuell anlegen.

Der Tablespace wird angelegt, das Datafile jedoch unter einem Dummy-Namen erstellt

```
SQL> select name from v$datafile;
/u01/app/oracle/product/12.1.0/dbhome_1/dbs/UNNAMED00006
/u01/app/oracle/oradata/pondb/system01.dbf
/u01/app/oracle/oradata/pondb/sysaux01.dbf
/u01/app/oracle/oradata/pondb/undotbs01.dbf
/u01/app/oracle/oradata/pondb/example01.dbf
/u01/app/oracle/oradata/pondb/users01.dbf
```

Um das zu korrigieren, braucht man als erstes die Größe des Originalfiles auf der Quellseite. Den Originalnamen des Datafiles kann man aus der rman-Meldung entnehmen

```
SQL> select BYTES/1024/1024/1024, name from v$datafile;
BYTES/1024/1024/1024 NAME
-------------------- ---------
2                    /u01/app/oracle/oradata/pondb/myts.dbf
```

Jetzt auf der Zielseite das Datafile mit der eben ermittelten Größe aus dem Dummy-File kopieren

```
SQL> alter database create datafile '$ORACLE_HOME/dbs/UNNAMED00006'
as ‘u01/app/oracle/oradata/pondb/myts.dbf' size 2G;
```

Danach kann die Replikation wieder gestartet werden.