Oracle DataGuard物理备库安装配置

## 0. 一些重要概念

### 1)、filesystem和 asm的区别？

在安装Oracle数据库时，有选项要求选择存储的类型。

自动存储管理 (ASM) 是 Oracle Database 的一个特性，它为数据库管理员提供了一个在所有服务器和存储平台上均一致的简单存储管理接口。作为专门为 Oracle 数据库文件创建的垂直集成文件系统和卷管理器，ASM 提供了直接异步 I/O 的性能以及文件系统的易管理性。ASM 提供了可节省 DBA 时间的功能，以及管理动态数据库环境的灵活性，并且提高了效率。ASM 的主要优点有：

简化和自动化了存储管理

提高了存储利用率和敏捷性

提供可预测的性能、可用性和可伸缩性

### 2)、startup 和 shutdown的参数的区别？

STARTUP 命令

STARTUP OPEN：STARTUP缺省的参数就是OPEN，打开数据库，允许数据库的访问。当前实例的控制文件中所描述的所有文件都已经打开。

STARTUP MOUNT：MOUNT数据库，仅仅给DBA进行管理操作，不允许数据库的用户访问。仅仅只是当前实例的控制文件被打开，数据文件未打开。

STARTUP NOMOUNT：仅仅通过初始化文件，分配出SGA区，启动数据库后台进程，没有打开控制文件和数据文件。不能任何访问数据库。

STARTUP PFILE= filename：以filename为初始化文件启动数据库，不是采用缺省初始化文件。

STARTUP FORCE：中止当前数据库的运行，并开始重新正常的启动数据库。

STARTUP RESTRICT：只允许具有RESTRICTED SESSION权限的用户访问数据库。

STARTUP RECOVER：数据库启动，并开始介质恢复。

SHUTDOWN 命令

SHUTDOWN有四个参数：NORMAL、TRANSACTIONAL、IMMEDIATE、ABORT。缺省不带任何参数时表示是NORMAL。

SHUTDOWN NORMAL：不允许新的连接、等待会话结束、等待事务结束、做一个检查点并关闭数据文件。启动时不需要实例恢复。

SHUTDOWN TRANSACTIONAL：不允许新的连接、不等待会话结束、等待事务结束、做一个检查点并关闭数据文件。启动时不需要实例恢复。

SHUTDOWN IMMEDIATE：不允许新的连接、不等待会话结束、不等待事务结束、做一个

检查点并关闭数据文件。没有结束的事务是自动rollback的。启动时不需要实例恢复。

SHUTDOWN ABORT：不允许新的连接、不等待会话结束、不等待事务结束、不做检查点且没有关闭数据文件。启动时自动进行实例恢复。

当数据库正常关闭时（Shutdown Immediate,shutdown normal,shutdown tracsactional）,Oracle 会把SGA内容全部写回磁盘后才关闭数据库，这时内存和磁盘就完全同步了。 所以正常关闭数据库后数据不会丢失，但是如果数据库是异常关闭（突然断线，shutdown abort），内存中的数据来不及同步到磁盘，这是就会产生了数据不一致，Oracle 在次打开数据库时，就需要进行实例恢复。

### 3)、oracle spfile和pfile的区别？

pfile 默认的名称为“init+例程名.ora”文件路径：$ORACLE\_HOME /dbs，这是一个文本文件，可以用任何文本编辑工具编辑，可以放在客户端和服务端，从oracle 10g开始就不使用pfile。

spfile 默认的名称为“spfile+例程名.ora”文件路径：$ORACLE\_HOME/dbs以二进制文本形式存在，只能放在oracle服务器端，可以被rman备份。不能用vi编辑器对其中参数进行修改，。但可以通过在线系统命令 alter system set 参数名=值 scope=参数2进行修改；

两个文件可以用命令 CREATE PFILE FROM SPFILE或CREATE SPFILE FROM PFILE来互相创建

可以使用CREATE PFILE='/tmp/initDB11G\_stby.ora' FROM SPFILE;备份

### 4)、归档模式与非归档模式？

归档模式下会生成归档日志文件，其中保留了数据库的改动信息。

在这种模式下可以获得如下好处:

可以进行完全、不完全恢复：由于对数据库所做的全部改动都记录在日志文件中，如果发生硬盘故障等导致数据文件丢失的话，则可以利用物理备份和归档日志完全恢复数据库，不会丢失任何数据。

可以进行联机热备：所谓联机热备，就是在数据库运行状态下，对数据库进行备份。备份时用户对数据库的使用不受任何影响。

可以实施 Data Guard：可以部署 1 个或多个备用数据库，从而最大限度地提供灾难保护手段。

可以实施 Stream：利用 Stream 技术，可以实现最简单的单向复制到复杂的双向复制、多向复制， 提供更加灵活的数据冗余方案。

表空间可以脱机：可以备份部分数据库，比如重要的表空间。

能够增量备份：只需做一次完全备份，以后只备份发生改变的数据，可以提高备份速度。

更多的优化选项：随着 Oracle 版本升级，在联机热备方面不断有新的优化策略出现。

使用归档模式的缺点在于：

需要更多的磁盘空间保存归档日志；

DBA 会有更多的管理工作，包括维护归档空间、备份归档日志。

非归档模式不生成归档日志，从数据安全角度来说，这种模式缺点是主要的，而优点可以忽略不计。

非归档模式的优点包括：

DBA 的管理工作减少，因为非归档模式不产生归档日志，因此 DBA 不用考虑对归档的管理；性能会有提升。

非归档模式的缺点包括：

只能进行脱机备份，也就是所谓的“ 冷备份”，和联机备份的“ 热备份” 相对应，数据库必须完全关闭后备份，在备份过程中数据库不可用；

必须备份整个数据库，不能只备份部分数据库；

不能增量备份，对于 TB 级数据库（VLDB） ，这是一个非常大的缺点；

只能部分恢复，如果数据文件丢失需要恢复，DBA 只能恢复最后一次的完全备份，而之后的所有 数据库改变全部丢失。

非归档模式转换成归档模式：

1 登录数据库查看数据库的归档状态

SQL> archive log list;

如果出现信息：

Database log mode No Archive Mode

Automatic archival Disabled

则为非归档模式

2 关闭数据库

SQL>shutdown immediate;

3 启动数据库到mount状态

SQL>startup mount;

4 启动归档模式

SQL>alter database archivelog; (非归档模式为：alter database noarchivelog)

5 打开数据库

SQL>alter database open;

查看归档模式状态

SQL>archive log list

如果出现信息：

Database log mode Archive Mode

Automatic archival Enabled

成功改成归档模式

### 5)、db\_name、db\_unique\_name、instance\_name、service\_names、ORACLE\_SID、global\_names、DBid等的区别？

这些参数可以通过show parameter name看到。其中：

db\_name(数据库名)

db\_name必须是一个不超过8个字符的文本串。在数据库创建过程中，db\_name被记录在数据文件，日志文件和控制文件中。如果数据库实例启动过程中参数文件中的db\_name和控制文件中的db\_name名称不一样，则数据库不能启动。db\_name是最具有稳定意义的参数，官网说不能修改，修改后会导致数据库不能启动。

db\_name 也就是数据库的名字标示。这里，数据库里可能有多个实例，比如RAC里的多节点，这多个节点是不同的实例，但是却有相同的名字，他们的 db\_name是相同的但是Instance\_name是不同的。db\_name会保持在数据文件头里，所以更改db\_name不能仅仅修改parameter，还需要用nid 来进行更改，并且更改后还需要手工做些工作，使其生效。

但是服务器参数(spfile)中仅仅设置了db\_name那么其它的name比如db\_unique\_name,instance\_name,service\_names的值是怎么出来的？

db\_unique\_name(数据库唯一名)

db\_unique\_name这在另一个HA的应用，Dataguard中会经常提及的，和db\_name不一样的作用，在DG里，要求物理DG，主从库都有一样的db\_name，虽然他们和RAC不一样，并不是同一个库。这里是数据库的唯一名字。但是他们的db\_unique\_name是不一样的，用以进行不同的标示。db\_unique\_name的会影响到Service\_names，也会影响到动态监听的时候的service\_name

service\_names(数据库服务名)

service\_names 在tnsnames.ora中用的最多。

service\_names=DB\_UNIQUE\_NAME.DB\_DOMAIN

我的环境中db\_unique\_name,db\_domanin都没有设置，而db\_unique\_name在单实例下默认值为：db\_name = ORCL

所以service\_names=ORCL

这也就解释了刚开始提出的问题：没有在spfile中设置instance\_name,service\_names 但是动态注册时怎么发生的？

instance\_name (数据库实例名)

用于和操作系统之间的联系，用于对外部连接时使用。在操作系统中要取得与数据库之间的交互，必须使用数据库实例名。该值允许修改，但一般设置与数据库名相同即可

Instance\_name的默认值就是oracle SID。 一般跟数据库库名称相同，也可以不相同。

Instance\_name简单讲就是ORACLE\_SID,oracle里通过ORSCLE\_SID来管理不同的数据库实例。

服务器参数spfile中没有设置instance\_name,所以取默认值oracle\_sid,我这里sid 为ORCL 所以instance\_name=ORCL

oracle\_sid(数据库实例名)

SID的全称为site identifier，Oracle\_SID则为Oracle site identifier.

在Unix/Linux系统中，SID和ORACLE\_HOME在一起哈希后得到一个唯一的值作为SGA的key。

当oracle实例启动时，在操作系统上的fork进程则根据Oracle\_SID来创建相关后台进程。

Oracle 11g 支持Oracle\_SID的长度为12位，db\_name的长度为8位

作用：

用于区别同一台主机上不同的Oracle实例

决定实例所启动后台进程的名称。(实例由SGA和后台进程组成)

决定了参数文件的名称。如spfile.ora,init.ora

决定后台进程产生的相关跟踪文件、日志文件等。

如alert\_.log,\_arc1\_.trc,\_ora\_.trc

同一主机上不同的$ORACLE\_HOME，可以创建相同的ORACLE\_SID

由此可以得出

ORACLE\_HOME相同时，可以使用不同的ORACLE\_SID

ORACLE\_HOME不同时，可以使用相同的ORACLE\_SID

dbid(数据库id)

DBID 可以看成是db\_name在数据库内部的表示。 DBID是在创建数据库时，用db\_name 结合一种算法来创建的。 具体用什么算法，不太清楚。它存在与数据文件和控制文件，用于表示数据文件的归属。 所以这个DBID 是唯一的。 对于不同的数据库，DBID 是不同的，但是db\_name 有可能相同。

用身份证打个比方： 可以有同名的人，但是它的省份证号码肯定是不同的。

查看dbid：

select dbid from v$database;

global\_names (全局数据库名)

Global\_names 是一个布尔值，global\_names的作用是创建db link时是否强制使用远程数据库的global\_name,如果global\_names=true,则db link name必须要求是remote database的global\_name,否则创建之后db link 不能连同，缺省值是false。多用于分布式系统。

global\_name命名规则= db\_name+[db\_domain]

也就是由db\_name.db\_domain构成。

查看global\_name

SELECT \* FROM GLOBAL\_NAME;

**总结：**

DB 相关的： DBID, ORCLE\_SID

PFILE中的参数：DB\_NAME，DB\_DOMAIN, INSTANCE\_NAME，

DB\_UNIQUE\_NAME,SERVICE\_NAMES,

GLOBAL\_NAME，GLOBAL\_NAMES

Listener.ora中参数：

SID\_NAME，GLOBAL\_DBNAME

Tnsnames.ora中参数：

SERVICE\_NAME，SID

### 6)、最大保护模式、最高性能模式？

Data Guard 提供三种数据保护模式：最大保护（Maximum Protection），最高可用（Maximum Availability）和 最高性能（Maximum Performance）。

对DG而言，最大性能模式是默认的保护模式。当我们完成了DG安装之后，就自动进入了Maximum Performance模式。

如果按照对数据的保护程度或者说主从库数据的同步性 由低到高排序，三种保护模式的顺序应该是：最高性能、最高可用、最大保护 。参见如下文章，讲的很详细

[【DataGuard】Oracle DataGuard 数据保护模式切换](http://blog.itpub.net/29475508/viewspace-1248530/)

### 7)、联机重做日志(online redo log)、归档重做日志(archvie log)和备用重做日志(standby redo log 需要另外配置)、告警日志(Alert log)、跟踪日志(Trace files)、审计日志(audit log)

**Oracle数据库有联机重做日志**，它记录了数据的所有变化(DML,DDL或管理员对数据所作的结构性更改等)；也提供恢复机制(对于意外删除或宕机利用日志文件实现数据恢复)；且可以被分组管理。

一般的，数据库会有两个及以上联机重做日志，这里我们以两个为例进行说明。当联机重做日志1被写满后，数据库会切换到另一个联机重做日志2中继续记录。那么日志2也被写满了怎么办呢？

话说前文讲到联机日志1和2，当两个日志都被写满后，数据库会根据不同的工作模式（归档模式和非归档模式），采取不同的反应。

处于非归档模式时，当联机重做日志2被写满后，会切换到联机重做日志1中继续写。那么日志1中之前的记录怎么办呢？当然是直接删除了。如此反复，日志中的内容也在被反复的删除和记录。

处于归档模式时，当联机重做日志1 写满后，会切换到联机重做日志2，此时数据库会将联机重做日志1备份到指定的目录下。这样当联机重做日志2被写满后切换到日志1时，虽然也会将日志1的内容删除，但是因为之前有了备份。妈妈再也不用担心我的操作记录丢失了。（当然，日志2被写满后也是会备份的）

联机重做日志组

由一个或多个相同的联机日志文件组成一个联机重做日志组

至少两个日志组，每组一个成员(建议每组两个成员，分散放开到不同的磁盘)

由LGWR后台进程同时将日志内容写入到一个组的所有成员

LGWR的触发条件

在事务提交的时候(COMMIT)

Redo Log Buffer 三分之一满

Redo Log Buffer 多于一兆的变化记录

在DBWn写入数据文件之前

联机重做日志成员

重做日志组内的每一个联机日志文件称为一个成员

一个组内的每一个成员具有相同的日志序列号(log sequence number),且成员的大小相同

每次日志切换时，Oracle服务器分配一个新的LSN号给即将写入日志的日志文件组

LSN号用于唯一区分每一个联机日志组和归档日志

处于归档模式的联机日志，LSN号在归档时也被写入到归档日志之中

**归档重做日志**，归档日志archive log是当数据库运行在归档模式下时，一个redo log file(group)写满后，由ARCn进程将联机重做日志的内容备份到归档日志文件下，然后这个redo log file(group)才能被下一次使用。不管数据库是否是归档模式，重做日志是肯定要写的。而只有数据库在归档模式下，重做日志才会备份，形成归档日志。

一般来说，归档日志结合全备份，用于数据库出现问题后的恢复使用。

归档日志和重做日志都是物理上的文件，只是存放的目录不同，而且重做日志的文件名不变，而归档日志的文件名是备份时系统生成的。

联机重做日志备份为归档日志后，系统就会把重做日志的内容清空，但文件依然存在，准备下一次使用。

联机重做日志纪录了你所有做过的dml语句，重做日志循环使用，写满一轮后就要覆盖前面的。如果你是用热备模式，当重做日志写满一个后就将内容写入归档日志，以备将来恢复数据用。

只有数据库运行在归档模式并且初始化参数ARCHIVE\_LOG\_START等于TRUE时，ARCn进程才能被启动，进行自动归档。

如果数据库运行在归档模式但ARCHIVE\_LOG\_START等于FALSE时，需要DBA手工归档。（这种方式可能会因为归档不及时导致数据库被迫挂起，不实用！）

联机日志文件,一般数据库有几个日志文件(例如有三个,编号分别为1,2,3)先写1,当1满时再写2,当2满时再写3,当3满时1就归 档出来,产生一个文件写到磁盘上,这个文件就叫归档日志文件.1归档出来后,新的联机日志文件又写到1中,将原来的覆盖,(即联机日志是循环使用的).一 般当产生一个检查点或联机日志写满一定程度时会产生一个归档日志文件.

必须启用归档模式才能将重做日志备份成为归档日志文件；

**备用重做日志**，备用重做日志类似于"联机重做日志"。不过备用重做日志是用于存储从主数据库接收的重做条目。

如果数据库运行在"最大保护"或"最大可用性"模式时，必须在备用库配置备用联机日志。配置备用联机重做日志可以做备用库的实时恢复。这在DataGuard恢复操作是很有帮助。

额外福利：

DataGuard主备机同步步骤：

1.主库生成重做数据，传输到备用库。

2.备用库系统通过"RFS进程(远程文件服务)"接收。

3.RFS进程(远程文件服务)将重做条目写入备用归档日志或备用重做日志(如果配置备用重做日志)中。

4.备用库通过归档日志或备用重做日志文件来恢复(实时应用情况下，备用重做日志写满后直接应该而不是归档后再应用)。

DataGuard恢复操作最关键的结构是：联机重做日志(redo log)、归档重做日志(archvie log)和备用重做日志(standby redo log 需要另外配置)。

联机重做日志(redo log):

主备库都包含redolog,物理备库因为不会以"读写"方式打开，所以不使用联机日志，但是联机重做日志是存在的。仅仅是不使用而已。

归档重做日志(archive log):

归档日志是主备库之间保持同步的主要文件，主库通过传送归档日志到备库而完成两台数据库间的同步。

备用重做日志(standby redo log):

备用重做日志类似于"联机重做日志"。不过备用重做日志是用于存储从主数据库接收的重做条目。

如果数据库运行在"最大保护"或"最大可用性"模式时，必须在备用库配置备用联机日志。配置备用联机重做日志可以做备用库的实时恢复。

备用重做日志的优点：

可以存放在裸设置上；

与联机重做日志一样可以多重备用重做日志，以提高重做日志的可靠性；

如果主数据库出现故障，备用重做日志可以恢复比归档日志更多的数据；

主数据库上的ARCn或LGWR进程直接传送数据到远程备机的备用重做日志，省去了注册归档日志的步骤。

**告警日志**，ORACLE警告日志记录一些信息和操作ORACLE数据库时的错误警告。

当数据库出现问题时，警告日志很快会给你指出问题所在。

比如说表不能增加存储空间，回滚段问题等等都包含在警告日志中。正因为如此，我们每天都要检查警告日志文件，看看数据库有没有什么异常？

alert日志: 在系统初始化参数文件设置的background\_dump\_dest对应的就是它的位置。

Select name ,value from v$parameter where name='background\_dump\_dest';

**跟踪日志**，Trace file（追踪文件)是以trc为后续的文本文件,它记录了各种sql操作及所消耗的时间等,根据trace文件我们就可以了解哪些sql导致了系统的性能瓶颈，进而采取恰当的方式调优.

trace日志：一般放seesion追踪的信息，对应系统初始化参数文件参数user\_dump\_dest对应的就是它的位置。

Select name ,value from v$parameter where name='user\_dump\_dest';

**审计日志**，审计（Audit)用于监视用户所执行的数据库操作，并且Oracle会将审计跟踪结果存放到OS文件（默认位置为$ORACLE\_BASE/admin/$ORACLE\_SID/adump/）或数据库（存储在system表空间中的 SYS.AUD$表中，可通过视图dba\_audit\_trail查看）中。默认情况下审计是没有开启的。

不管你是否打开数据库的审计功能，以下这些操作系统会强制记录：用管理员权限连接Instance；启动数据库；关闭数据库。

audit日志：审计的信息，对应系统初始化参数文件参数audit\_file\_dest对应的就是它的位置。

### 8)、redo与undo

REDO记录transaction logs，分为online和archived。以恢复为目的。

比如，机器停电，那么在重启之后需要online redo logs去恢复系统到失败点。

比如，磁盘坏了，需要用archived redo logs和online redo logs去恢复数据。

比如，truncate一个表或其他的操作，想恢复到之前的状态，同样也需要。

REDO 是为了重新实现你的操作，而UNDO相反，是为了撤销你做的操作，比如你得一个TRANSACTION执行失败了或你自己后悔了，则需要用 ROLLBACK命令回退到操作之前。

### 9)、控制文件

控制文件（Control File）是Oracle的物理文件之一，它记录了数据库的名字、数据文件的位置等信息。控制文件的重要性在于，一旦控制文件损坏，数据库将会宕机。控制文件是一个很小的二进制文件，用户不能编辑控制文件，控制文件的修改由Oracle自动完成。那么，我们能够查看控制文件里面的具体信息吗？答案是可以的，我们可以通过转储文件来获得控制文件的详细信息。

可以查看控制文件的位置：

select value from v$parameter where name ='control\_files';

VALUE

--------------------------------------------------------------------------------

/opt/oracle/oradata/master/control01.ctl, /opt/oracle/flash\_recovery\_area/master

/control02.ctl

### 10)、物理备份(热备与冷备)与逻辑备份(导出条件)

物理备份： 指对数据文件，控制文件，联机日志文件等文件进行物理拷贝的方法，这种方法是在文件层进行的，通过冗余的文件备份来体统数据保护。物理备份又可分为联机备份（也叫作联机热备）和脱机备份（也称冷备）。

逻辑备份： 利用Oracle 提供的导出工具把重要数据导出到文件，以后恢复时在利用工具把数据重新导入到数据库中，这种保护是在数据层进行的。 该类工具有： exp/imp, expdp/impdp（10g 以后版本）。

物理备份是最强健的数据保护方也，也是备份策略中首选的方法，逻辑备份只能作为物理备份的补充手段，不足以保护数据丢失。

物理备份又可分为用户管理备份（User-Managed Backup）和RMAN备份（Recovery Manager）。 前者是联合使用SQL 命令和OS的cp 命来进行文件备份。 Rman 备份指利用RMAN 工具来进行备份。

Rman 备份有几点好处：

1. 增量备份

2. 数据块恢复：可以在新进行数据块恢复，不必进行数据文件恢复，提高系统的可用性。

3. 压缩备份

4. 加密备份

具体用法参考blog：

RMAN 备份与恢复 实例

<http://blog.csdn.net/tianlesoftware/archive/2009/10/20/4699320.aspx>

Oracle Rman 命令详解(List report backup configure)

<http://blog.csdn.net/tianlesoftware/archive/2009/12/13/4976998.aspx>

Oracle Rman跨resetlogs版本恢复

<http://blog.csdn.net/tianlesoftware/archive/2009/10/17/4682463.aspx>

ORACLE 数据库逻辑备份 简单 EXP/IMP

<http://blog.csdn.net/tianlesoftware/archive/2009/10/24/4718366.aspx>

Oracle 10g EXPDP和IMPDP使用说明

<http://blog.csdn.net/tianlesoftware/archive/2009/10/16/4674224.aspx>

12)、恢复

Oracle恢复可以分成实例恢复（Instance Recovery）,介质恢复（Media Recovey），其中介质恢复又可分为完全恢复（Complete Recovery）和不完全恢复（Incomplete Recovery）。

Instance Recovery -- 由Oracle 自动完成，无需DBA 干预

如果实例异常关闭（宕机，shutdown abort），并且数据文件，控制文件，联机日志都没有丢失。在下次启动时，要利用联机日志的内容进行恢复，这种恢复就是实例恢复（Instance Recovery）。

Instance Recovery 主要包括3个阶段：

1） 根据联机日志内容进行Rollover。

2） 打开数据库，提供服务

3） SMON 或者用户进程进行Rollback。

Media Recovery

    如果发生数据文件丢失或者破坏，就需要使用备份和归档日志来进行恢复， 这种恢复就是 介质恢复，它需要有备份，归档日志，联机日志一起才能完成。又分为完全恢复和不完全恢复两种。

完全恢复

它是把数据库恢复到发生故障时的状态，名字中的完全指没有任何数据损失，要实现这个目标，必须满足一定的条件： 备份，从备份之后的所有归档日志，联机日志都可用。

完全恢复是最简单的一种恢复， 只需要两个命令： restore database 和 recover database。

这2个命令的具体用法参考我的Blog： RMAN 备份与恢复 实例， 里面有详细的例子

<http://blog.csdn.net/tianlesoftware/archive/2009/10/20/4699320.aspx>

在此补充一点知识：RECOVER DATABASE UNTIL CANCEL 和 RECOVER DATABASE UNTIL CANCEL USING BACKUP CONTROLFILE区别

1） RECOVER DATABASE UNTIL CANCEL  
==> DATAFILE HEADER SCN一定会小于CONTROLFILE的DATAFILE SCN  
如果你有进行RESTORE DATAFILE，则该RESTORE的DATAFILE HEADER SCN一定会小于目前CONTROLFILE的DATAFILE SCN，此时会无法开启数据库，必须进行media recovery。 重做archive log直到该datafile header的SCN=current scn

2） RECOVER DATABASE UNTIL CANCEL USING BACKUP CONTROLFILE;

==> DATAFILE HEADER SCN一定会大于CONTROLFILE的DATAFILE SCN

如果只是某TABLE被DROP掉，没有破坏数据库整体数据结构，还可以用NCOMPLETE RECOVERY解决 如果是某个TABLESPACE OR DATAFILE被DROP掉，因为档案结构已经破坏，目前的CONTROL FILE内已经没有 该DATAFILE的信息，就算你只RESTORE DATAFILE然后进行INCOMPLETE RECOVERY也无法救回被DROP的DATA FILE。

只好RESOTRE 之前备份的CONTROL FILE(里头被DROP DATAFILE Metadata此时还存在)，不过RESTOREC CONTROL FILE后 此时Oracle会发现CONTROL FILE内的SYSTEM SCN会小于目前的DATAFILE HEADER SCN，也不等于目前储存于LOG FILE内的SCN， 此时就必须使用RECOVER DATABASE UNTIL CANCEL USING BACKUP CONTROLFILE到DROP DATAFILE OR DROP TABLESPACE之前的SCN。

不完全恢复

不完全恢复是指数据库无法恢复到发生故障那一点的状态，而只能恢复到之前一段时间的状态，这就以为着承受一定量的数据损失。

Oracle 运行时包括参数文件，控制文件，数据文件，联机日志，那么哪些文件会导致不完全恢复呢？

参数文件只是一个文本文件，丢失没有关系。控制文件通常有多个文件相互冗余。 而且在做全库备份时，控制文件会被自动备份，故所有文件都损坏，也可以通过备份进行恢复，即使没有备份，也可以通过重建控制文件来恢复，也不会造成数据丢失。如果是数据文件损坏，只要有备份和备份后的完整的日志文件，也可以完成恢复，不会造成数据丢失。

联机日志比较特殊，通过前面的介绍，我们知道在数据库异常关机的情况下，它可能造成数据丢失。

我们来看一下联机日志损坏的恢复方法：

先用SQL 查看一下出问题的联机日志是什么状态：

SQL> Select thread#,group#,status from v$log;

1）如果是Inactive 状态的联机日志，因为它里面的记录已经同步到数据文件，所以只需要把该日志删掉即可。

2）如果是Active/ current 状态的连接日志， 因为他们里面有记录没有同步到数据文件，可以通过如下方式来恢复：

（1）关闭所有实例

（2）在受损实例上，启动到mount状态

（3）执行alter database open resetlogs

（4）如果在第三步出现错误，并其实需要不完全恢复，就执行一下： recover database until cancel

（5）实例启动成功后，启动其他实例

（6）立即对数据库进行一次全备。

说明： 在做了alter database open resetlogs;会把online redelog file清空，数据文件丢失.所以这个时候要做一个全备份。resetlogs命令表示一个数据库逻辑生存期的结束和另一个数据库逻辑生存期的开始，每次使用resetlogs命令的时候，SCN不会被重置，不过oracle会重置日志序列号，而且会重置联机重做日志内容.这样做是为了防止不完全恢复后日志序列会发生冲突（因为现有日志和数据文件间有了时间差）。

Crash Recovery -- 由Oracle 自动完成。

Instance Recovery是实例发生Crash 后进行的Recovery，这种恢复是在故障节点进行，而RAC 中的Crash Recovery 是某个实例发生Crash 后在其他实例上进行的Recovery。 这种Recovery 有一个特殊要求：在健康节点执行Crash Recovery时，必须要保证故障节点不能在对共享数据进行操作，也就是要对故障节点进行IO 隔离（IO Fencing），这是由CSS服务来保证的。

在Crash Recovery过程中PCM Lock起到了重要作用，恢复实例（执行Recovery动作的实例）根据数据块的PCM-Lock 状态来决定数据块是否需要进行恢复。

Crash Recovery 可分为3个阶段：

1）First-Pass Log Read

2）Recovery Claim Locking

3）Second-Pass Log Read

Online Block Recovery

Online Block Recovery 是RAC所特有的， 如果某个用户进程在修改数据时异常死掉，导致SGA的Data buffer数据不一致，或者说Data Buffer被破坏，这时就会触发Online Block Recovery，这个动作可有PMON进程或者前台进程完成。这个恢复过程需要一个恢复起点， 这个起点就是最近的Past Image。

### 11)、Active Standby和Read Only、Snapshot Standby？

[【DataGuard】Oracle 11g DataGuard 新特性之 Active Standby：Real-Time Apply+Query](http://blog.itpub.net/29475508/viewspace-1244579/)

Active Data Guard, 是Oracle Database Enterprise Edition的一个功能，需要额外授权。

在Oracle 11g之前，物理备库（physical Standby）在应用redo的时候，数据库需要处于mount状态。从11g开始，应用redo的时候，物理备库可以处于read-only模式，这就称为Active Data Guard 。通过Active Data Guard，可以在物理备库进行查询或者导出数据，从而减少对主库的访问和压力。

如需启用Active Data Guard, 只需要将备库以 read-only 模式打开，而且执行 ALTER DATABASE RECOVER MANAGED STANDBY DATABASE语句就可以。

需要注意的是：主库和备库的COMPATIBLE 参数至少要设置为11.0.0。

查看COMPATIBLE参数：

SQL>show parameter compatible;

修改COMPATIBLE参数：

SQL>ALTER SYSTEM SET compatible='10.1.0.2.0' SCOPE=SPFILE;

如下：在备库执行：

SQL>startup mount;

SQL>alter database open read only;

SQL>ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;

如果已经启用了Active Data Guard，备库的V$DATABASE会显示为"READ ONLY WITH APPLY':

SQL> SELECT open\_mode FROM V$DATABASE;

OPEN\_MODE

--------------------

READ ONLY WITH APPLY

为了保证备库数据的实时性，需要在备库启动real-time apply:

SQL>ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE;

查看adg数据库状态(分别在主库和备库上执行)：

SELECT database\_role, open\_mode FROM v$database;

Oracle 11g DataGuard 的Snapshot Standby数据库功能，可以让物理备库从只读状态 切换到可读写状态。在snapshot standby状态下，备库可以进行读写操作，同时备库还会正常接收主库的归档和日志信息，但不会应用这些redo日志。

当备库读写状态下任务完成后，还可以从Snapshot Standby数据库角色切换回Physical standby备库角色，恢复与主库数据同步。

切换回物理备库后，在Snapshot standby状态下写入备库的数据将不再存在。

总结：

1、切换Snapshot standby前，物理备库需设置快速回复区db\_recovery\_file\_dest。

2、物理备库切换为Snapshot standby状态前，需取消备库的日志应用。

3、物理备库切换为Snapshot standby时，会创建一个restore point，当需要从snapshot standby状态切回原来的状态时，会回到这个restore point的状态，期间在snapshot standby 的备库上写入的数据将不再存在。

[【DataGuard】Oracle 11g DataGuard 新特性之 Snapshot Standby Database](http://blog.itpub.net/29475508/viewspace-1256816/)

### 12)、如果使用的是非catalog，在rman 连接时，加上nocatalog关键字？

（1）如果使用的是非catalog，在rman 连接时，加上nocatalog关键字，如：

rman targetsys/oracle@DG\_PD auxiliary sys/oracle@DG\_ST nocatalog

（2） 在执行duplicate的时候，如果源库和目标库目录相同，那么在duplicate 时，需要加上nofilenamecheck，如下：

RMAN> duplicate target database for standby from active database do recover nofilenamecheck;

### 13)、Oracle Listener 动态注册与静态注册

(1)、什么是注册

注册就是将数据库作为一个服务注册到监听程序。客户端不需要知道数据库名和实例名，只需要知道该数据库对外提供的服务名就可以申请连接到数据库。这个服务名可能与实例名一样，也有可能不一样。

　　在数据库服务器启动过程中，数据库服务器会向监听程序注册相应的服务（无论何时启动一个数据库，默认地都有两条信息注册到监听器中：数据库服务器对应的实例和服务。）

相当于是这样：在数据库服务器和客户端之间有一监听程序（Listener），在监听程序中，会记录相应数据库对应的服务名（一个数据库可能对应有多个服务名），当客户端需要连接数据库时，只需要提供服务名，就可以建立客户端和服务器之间的连接。

(2)、动态注册

动态注册是在instance启动的时候PMON进程根据init.ora中的instance\_name,service\_names两个参数将实例和服务动态注册到listener中。

动态监听：

　　oracle监听器运行后，oracle实例在open时，会动态向监听程序注册其service\_names和instance\_name。

　　其中instance\_name默认是：db\_name

而service\_names默认是：db\_name.db\_domain

(3)、静态注册

静态注册就是实例启动时读取listener.ora文件的配置，将实例和服务注册到监听程序。无论何时启动一个数据库，默认地都有两条信息注册到监听器中：数据库服务器对应的实例和服务。

静态注册时，listener.ora中的GLOBAL\_DBNAME向外提供服务名，listener.ora中的SID\_NAME提供注册的实例名。

静态监听：

　　oracle实例运行后，监听程序启动时，根据listener.ora的配置注册相应的服务。

　　其中global\_dbname对应的是oracle对外的服务名，即初始化参数里的service\_names

而sid\_name对应的是oralce实例的名称，即初始化参数里的instance\_name

(4)、查询某服务是静态注册还是动态注册

可以使用命令lsnrctl status来查看某服务是静态注册还是动态注册。

　　实例状态为UNKNOWN值时表明此服务是静态注册的设置。这时监听器用来表明它不知道关于该实例的任何信息，只有当客户发出连接请求时，它才检查该实例是否存在。

动态注册的数据库通过状态信息中的状态READY或状态BLOCKED（对于一个备用数据库）来指明。不管关闭何时数据库，动态注册的数据库都会动态地从 监听器注销，而与之相关的信息将从状态列表中消失。这样，不管数据库是在运行还是已经关闭，监听器总是知道它的状态。该信息将被用于连接请求的回退（fallback）和负载平衡。

既然有动态监听为什么还要静态监听呢？原因如下：

　　1.监听器不是最早启动，oracle实例先启动

　　2.监听器重启

　　3.oracle实例没有open

### 14)、表空间和数据文件关系

Oracle数据库在逻辑上划分为一个或者多个表空间（TABLESPACE）。Oracle表空间是包含物理数据文件的逻辑实体。一个表空间中可以包含一个或者多个数据文件。而数据文件是Oracle数据库的物理组成部分，是物理概念。一个数据文件只能隶属于一个表空间。Oracle数据库中每一个表都属于一个给定的表空间中，通过表空间的逻辑形式管理数据表。

数据库，表空间，及数据文件关系密切，但同时又有很多区别：

一个Oracle数据库是由一个或多个表空间（tablespace）的逻辑存储单位构成的，这些表空间共同来存储数据库的数据

Oracle数据库的每个表空间由一个或多个被称为数据文件（datafile）的物理文件构成，这些文件由Oracle所在的操作系统管理。

数据库的数据实际存储在构成各个表空间的数据文件中。例如，一个最简单的Oracle数据库至少包含一个表空间及一个数据文件。再例如，一个数据库可能含有三个表空间，每个表空间由两个数据文件构成（即总共有六个数据文件）。

SQL>select ts#,name from v$tablespace;#查询表空间

表空间与数据文件之间的对应关系

SQL>select t.ts#,t.name,d.ts#,d.name from v$tablespace t,v$datafile d where t.ts#=d.ts#;

为什么使用表空间

表空间允许更容易将空间限额分配给不同的数据库用户

表空间允许你执行基于表空间的部分备份和恢复

由于像数据仓库分区表这样的大对象可以分布在几个表空间上，因此，可以通过使表空间横跨几个磁盘和控制器来提高性能

可以使一个表空间脱机而不必停止整个个数据库

表空间是分配数据库空间的一种便利方法

通过使用表空间级的导入和导出实用程序可以导入和导出指定的应用数据

## Oracle数据库安装

### 1)、增加swap分区大小

1. 查看当前分区情况

free -m

2. 增加 swap 大小, 4G 左右

dd if=/dev/zero of=/var/swap bs=1024 count=4096000

3. 设置交换文件

mkswap /var/swap

4. 立即激活启用交换分区

swapon /var/swap

5. 添加系统引导时自启动运行

vi /etc/fstab

添加一行

swap /var/swap swap defaults 0 0

查看系统内存：

grep MemTotal /proc/meminfo

查看交换空间：

grep SwapTotal /proc/meminfo

查看系统架构：

uname -m

查看系统版本：

cat /proc/version

### 2)、确认依赖包

binutils-2.17.50.0.6

compat-libstdc++-33-3.2.3

compat-libstdc++-33-3.2.3 (32 bit)

elfutils-libelf-0.125

elfutils-libelf-devel-0.125

gcc-4.1.2

gcc-c++-4.1.2

glibc-2.5-24

glibc-2.5-24 (32 bit)

glibc-common-2.5

glibc-devel-2.5

glibc-devel-2.5 (32 bit)

glibc-headers-2.5

ksh-20060214

libaio-0.3.106

libaio-0.3.106 (32 bit)

libaio-devel-0.3.106

libaio-devel-0.3.106 (32 bit)

libgcc-4.1.2

libgcc-4.1.2 (32 bit)

libstdc++-4.1.2

libstdc++-4.1.2 (32 bit)

libstdc++-devel 4.1.2

make-3.81

sysstat-7.0.2

unixODBC-2.2.14-11.el6 (x86\_64) or later

unixODBC-2.2.14-11.el6.i686 or later

unixODBC-devel-2.2.14-11.el6 (x86\_64) or later

unixODBC-devel-2.2.14-11.el6.i686 or later

libXp

$yum install -y binutils compat-libstdc++-33 elfutils-libelf elfutils-libelf-devel gcc gcc-c++ glibc glibc-common glibc-devel libaio libaio-devel libgcc libstdc++ libstdc++-devel make numactl sysstat libXp unixODBC unixODBC-devel pdksh\*

$yum install binutils\* compat-libstdc\* elfutils-libelf\* gcc\* glibc\* ksh\* libaio\* libgcc\* libstdc\* make\* numactl-devel\* sysstat\* compat\* elfutils\* unix\* libXp\* pdksh\* -y

确认哪些包已安装

rpm -q package\_name

########

手工安装pdksh-5.2.14-37.el5\_8.1.x86\_64.rpm

wget http://mirror.centos.org/centos/5/os/x86\_64/CentOS/pdksh-5.2.14-37.el5\_8.1.x86\_64.rpm

yum install pdksh-5.2.14-37.el5\_8.1.x86\_64.rpm

### 3)、创建所需的用户组(在root帐号下操作)：

$groupadd oinstall;

$groupadd dba;

$groupadd oper;

创建所需的帐号(在root帐号下操作)：

使用命令 id oracle; 查看 oracle 帐号是否已存在，若未存在则使用如下命令进行创建

$useradd -g oinstall -G dba,oper oracle; #初始群组为 oinstall，有效群组为 dba、oper

$passwd oracle; #为oracle帐号设置密码(安装的时候需要登录)

若oracle帐号已存在，则使用命令 usermod -a -G dba,oper oracle 将其加入群组 dba 和 oper中

### 4)、修改安装环境

vi /home/oracle/.bash\_profile

文件内容如下：

export ORACLE\_BASE=/opt/oracle

export ORACLE\_HOME=$ORACLE\_BASE/product/11.2.0/db\_1

export ORACLE\_SID=orcl

export NLS\_LANG=AMERICAN\_AMERICA.UTF8

export PATH=/usr/lib64:$ORACLE\_HOME/bin:$PATH

umask 022

export ORACLE\_TERM=xterm

export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$ORACLE\_HOME/lib:$ORACLE\_HOME/lib32:/lib:/usr/lib:$ORACLE\_HOME/rdbms/lib

export SHLIB\_PATH=$ORACLE\_HOME/lib32:$ORACLE\_HOME/rdbms/lib32

export CLASSPATH=$ORACLE\_HOME/JRE:$ORACLE\_HOME/jlib:$ORACLE\_HOME/rdbms/jlib:$ORACLE\_HOME/network/jlib

增加Oracle 用户权限

$chown oracle:dba /home/oracle/.bash\_profile

若允许远程连接到Oracle,需要设置远程服务器端的Oracle。

设置"Database Configuration Assistant"，在进行第6步：连接模式 时选择：“共享服务器模式” 。

并且下面的 “共享服务器” 选择 3 以上。然后点击“确定”或“是” 。

### 5)、修改内核参数和资源限制(在root帐号下操作)：

编辑内核参数配置文件 vi /etc/sysctl.conf，在该文件下修改或添加如下行：

fs.file-max = 6815744

fs.aio\_max\_nr=1048576

kernel.shmall = 2097152

kernel.shmmax = 2147483648

kernel.shmmni = 4096

kernel.sem = 250 32000 100 128

net.ipv4.ip\_local\_port\_range = 9000 65500

net.core.rmem\_default = 262144

net.core.rmem\_max = 4194304

net.core.wmem\_default = 262144

net.core.wmem\_max = 1048576

再执行以下命令使以上配置立即生效：

$/sbin/sysctl -p;

确认是否执行正确：

/sbin/sysctl –a

修改系统资源限制(在root帐号下操作)：

编辑系统资源限制配置文件 vi /etc/security/limits.conf，在该文件下添加如下行：

oracle soft nproc 2047

oracle hard nproc 16384

oracle soft nofile 1024

oracle hard nofile 65536

oracle soft stack 10240

编辑文件 vi /etc/pam.d/login，添加如下行(11gR1有要求，R2没有要求)：

session required pam\_limits.so

编辑 vi /etc/profile 文件，添加如下行(11gR1有要求，R2没有要求)：

if [ $USER = "oracle" ]; then

if [ $SHELL = "/bin/ksh" ]; then

ulimit -p 16384

ulimit -n 65536

else

ulimit -u 16384 -n 65536

fi

fi

该配置在用户oracle登录时会立即生效，如果当前 oracle 用户已经登录可退出后重新登录使之生效。

### 6)、关闭SELinux(在root帐号下操作)：

编辑SELinux配置文件 vi /etc/selinux/config，将SELINUX的值设为 disabled，如下：

$SELINUX=disabled;

修改该文件可使重启系统后不启动SELinux。关闭当前已开启的SELinux使用如下命令：

$setenforce 0

修改/etc/hosts文件(在root帐号下操作)：

### 7)、编辑文件 vi /etc/hosts，添加一行：

$192.168.229.128 master

$192.168.229.129 slave

若不进行此操作会在安装时弹出警告，并且可能影响监听程序的正常运行，所以建议做这个修改。

查看主机名与hosts文件是否一致，不一致需要修改为一致

cat /etc/sysconfig/network

### 8)、创建Oracle所需的目录(在root帐号下操作)：

把oracle安装在 /opt/oracle 目录下，所以需创建该目录：

$mkdir -p /opt/oracle;

$chown -R oracle:oinstall /opt/oracle;

$chmod -R 755 /opt/oracle;

安装时还需要设置 Inventory 目录，所以需创建该目录：

$mkdir -p /opt/oraInventory;

$chown -R oracle:oinstall /opt/oraInventory;

$chmod -R 755 /opt/oraInventory;

### 9)、开始安装(以oracle身份登录)：

安装包相关：

解压zip安装文件：

$unzip linux.x64\_11gR2\_database\_1of2.zip

$unzip linux.x64\_11gR2\_database\_2of2.zip

执行该程序开始安装：

$sh /home/oracle/database/runInstaller;

$su

$sh /opt/oraInventory/orainstRoot.sh

$sh /opt/oracle/product/11.2.0/db\_1/root.sh

配置监听程序

$netca

配置数据库

$dbca

需注意选择字符集时 选择utf-8

查看已经启动的Oracle进程：

$ps –ef | grep oracle

Database infomation

global db name:master

global db name:slave

sid:orcl

service name ：master

server parameter file name:spfileorcl.ora

db control url is https://master:1158/em

启动监听服务：

启动监听：lsnrctl start

查看监听：lsnrctl status

停止监听：lsnrctl stop

启动数据库：

以管理员模式登录：

sqlplus / as sysdba

启动数据库

startup;

关闭防火墙，以便外网能访问

su

chkconfig iptables off

### 10)、常用开启关闭命令

启动和关闭数据库

启动oracle数据库

$su - oracle #切换到oracle用户

$sqlplus /nolog #进入sqlplus

$conn / as sysdba #以超级管理员权限登录

$startup #启动数据库实例

$quit #退出

$lsnrctl start #启动监听

startup参数说明：

startup #不带任何参数，启动数据库实例并打开数据库,一般选择这种启动方式

startup nomount #只启动数据库实例，不打开数据库，一般用于创建新的数据库时使用

startup mount #启动数据库实例，并加载数据库，但不打开数据库，一般用于修改数据库名称等管理时使用

关闭oracle数据库

$su - oracle #切换到oracle用户

$lsnrctl stop #关闭监听器

$sqlplus /nolog #进入sqlplus

$shutdown immediate #关闭数据库实例

$quit #退出

shutdown参数说明：

normal #等待所有的用户断开连接，执行命令后不允许新连接

immediate #等待用户完成当前的语句后，再断开用户连接，不允许新用户连接

transactional #等待用户完成当前的事务后断开连接，不允许新用户连接

abort #强行断开连接并直接关闭数据库

前面三种方法不会导致数据库出错，最后一种方法非特殊状况不建议使用，

一般使用shutdown immediate关闭数据库

备注：如果执行shutdown immediate出现ORA-01012: not logged on错误

请先执行以下语句

ps -ef|grep ora\_dbw0\_$orcl

kill -9 pid #pid为进程号

其中orcl 是数据库的SID

oracle几个重要配置文件：

listener.ora

tnsnames.ora

init\*.ora

### 11)、设置Oracle开机启动

说明：linux下默认安装完Oracle后不会实现开机自启动

1、su - root #切换到root用户操作

/data/oracle/product/11.2.0/db\_1/root.sh #执行脚本，生成/etc/oratab文件

/data/oraInventory/orainstRoot.sh

2、vi /etc/oratab #编辑

找到：$ORACLE\_SID:$ORACLE\_HOME:<N|Y>:

取消前面的#注释，修改为：

orcl:/data/oracle/product/11.2.0/db\_1:Y #orcl为oracle数据库实例名称

:wq! #保存退出

3、vi /data/oracle/product/11.2.0/db\_1/bin/dbstart #编辑

找到：ORACLE\_HOME\_LISTNER=$1

修改为：ORACLE\_HOME\_LISTNER=$ORACLE\_HOME

4、vi /data/oracle/product/11.2.0/db\_1/bin/dbshut #编辑

找到：ORACLE\_HOME\_LISTNER=$1

修改为：ORACLE\_HOME\_LISTNER=$ORACLE\_HOME

5、创建oracle启动脚本

vi /etc/init.d/oracle #编辑，添加以下代码

##########################################################

#!/bin/sh

# chkconfig: 0356 99 10

# description: Startup Script for Oracle Databases

# Script path：/etc/init.d/oracle

# Author：www.osyunwei.com by qihang01

#

ORACLE\_BASE=/data/oracle

ORACLE\_HOME=$ORACLE\_BASE/product/11.2.0/db\_1

ORACLE\_SID=orcl

PATH=$ORACLE\_HOME/bin:/usr/sbin:$PATH

export ORACLE\_BASE ORACLE\_HOME ORACLE\_SID PATH

#

ORA\_OWNR="oracle"

#

if [ ! -f $ORACLE\_HOME/bin/dbstart -o ! -d $ORACLE\_HOME ]

then

echo "Oracle startup: cannot start"

exit 1

fi

#

case "$1" in

start)

su - $ORA\_OWNR -lc $ORACLE\_HOME/bin/dbstart

echo "Oracle Start Succesful!"

;;

stop)

#

su - $ORA\_OWNR -lc $ORACLE\_HOME/bin/dbshut

echo "Oracle Stop Succesful!"

;;

restart)

$0 stop

$0 start

;;

\*)

echo $"Usage: `basename $0` {start|stop|restart}"

exit 1

esac

exit 0

##########################################################

:wq! #保存退出

chown oracle.oinstall /etc/init.d/oracle #更改文件所有者

chmod 755 /etc/init.d/oracle #添加执行权限

ln -s /etc/init.d/oracle /etc/rc3.d/S99oracle #添加系统启动级别3软连接

ln -s /etc/init.d/oracle /etc/rc5.d/S99oracle #添加系统启动级别5软连接

ln -s /etc/init.d/oracle /etc/rc0.d/K01oracle #添加系统关机软连接

ln -s /etc/init.d/oracle /etc/rc6.d/K01oracle #添加系统重启软连接

chkconfig --add oracle #添加系统服务

chkconfig oracle on #设置开机启动

备注：# chkconfig: 0356 99 10

0356：定义运行级别

99：开机启动优先级（数字越大启动越晚）

10：关机停止优先级（数字越小关闭越早）

系统运维 www.osyunwei.com 温馨提醒：qihang01原创内容 版权所有,转载请注明出处及原文链接

6、恢复操作系统标识

vi /etc/redhat-release #编辑

把：redhat-4

修改为：CentOS release 5.8 (Final)

:wq! #保存退出

7、更改系统默认启动级别为3

vi /etc/inittab #编辑

id:3:initdefault: #修改5为3

:wq! #保存退出

8、测试

shutdown -r now #重启系统

service oracle start #启动oracle

service oracle stop #关闭oracle

/etc/init.d/oracle restart #重启oracle

su - oracle #切换到oracle用户

dbstart #启动oracle

dbshut #关闭oracle

ps -efw | grep ora #查看oracle进程

ps -efw | grep LISTEN | grep -v gre #查看oracle端口监听

lsnrctl status #查看oracle运行状态

至此，CentOS下安装Oracle 11g R2教程完成。

## 安装Oracle物理备库

### 0)、软件基础环境

DG1: CentOS 6.8 x64 Oracle 11g

DB\_NAME :master SID\_NAME = orcl

IP:192.168.229.128

DG2: CentOS 6.8 x64 Oracle 11g

DB\_NAME:slave SID\_NAME = orcl

IP:192.168.229.129

判断DG是否已经安装：

select \* from v$option where parameter = 'Oracle Data Guard';

### 1)、主库前期准备开启归档及强制日志

sqlplus / as sysdba

SQL>startup mount

SQL>alter database archivelog; ----开启归档

SQL>archive log list;

SQL>alter database force logging; ---强制日志模式

SQL>select name, force\_logging from v$database;

--如果需要在主库添加或者删除数据文件时，这些文件也会在备份添加或删除，使用如下：

sql>alter system set standby\_file\_management='AUTO';

sql>show parameter standby--默认此参数是manual手工方式

SQL>alter database open;

### 2)、主库创建standby log files(备用日志文件）

Create Redo Logs

sql>alter database add standby logfile group 11 '/opt/oracle/oradata/master/standby11.log' size 50M;

sql> alter database add standby logfile group 12 '/opt/oracle/oradata/master/standby12.log' size 50M;

sql> alter database add standby logfile group 13 '/opt/oracle/oradata/master/standby13.log' size 50M;

sql> alter database add standby logfile group 14 '/opt/oracle/oradata/master/standby14.log' size 50M;

sql> show parameter REMOTE\_LOGIN\_PASSWORDFILE--检查REMOTE\_LOGIN\_PASSWORDFILE值是否为 EXCLUSIVE

如果值不是EXCLUSIVE，则：

sql> alter system set remote\_login\_passwordfile=exclusive scope=spfile;

### 3)、db\_name和db\_unique\_name

默认db\_name和db\_unique\_name和实例名是一致的，这里分别为master,slave

需要注意在DG中主库和从库的db\_unique\_name是不能一致的，需要区分开的。

这里我们设置主库的db\_unique\_name为master，从库为slave

sql>show parameter db\_unique\_name

设置：

sql>alter system set db\_unique\_name=master scope=spfile;

--注意虽然默认db\_unique\_name和db\_name是一致的，但是需要显式设置，否则在spfile中没有此参数

### 4)、开启闪回数据库

强烈建议开启数据库闪回功能。闪回允许你将数据库还原到以前的某一时间点。当发生故障转移时，这个功能非常有用，它能让你将老的主库闪回到故障前，然后将其转换为备库。如果没有启用闪回功能，你就必须重建备库，意味着要再复制一次数据文件。

除了这个好处，闪回还能在某些情况下让你避免从备份恢复数据。

(1)快速恢复区(Flash/Fast Recovery Area)，默认是配置的,但是需要确认这个区域的磁盘够大，至少300G以上（默认3G）

sql>show parameter db\_recovery\_file\_dest

可以修改位置：sql>alter system set db\_recovery\_file\_dest='新路径';

更改大小：sql>alter system set db\_recovery\_file\_dest\_size=400G;

(2)查看是否启用，默认是不开启的

sql>select flashback\_on from v$database;

开启：sql>alter database flashback on;

FLASHBACK\_ON 这列的值应该是 YES。如果你碰到 ORA-01153 报错，那一定是在备库进行此操作。你需要先取消重做日志应用，启用闪回日志，然后重新启用日志应用。

在主库启用闪回日志，不会同步备库也启用。你必须手动在主库和备库上均启用闪回日志。如果不启用闪回日志，当出现故障转移时，你将需要完全重新开始创建一个备库。

### 5)、SQL\*NET设置

在创建备库前，要确认两台服务器的数据库之间能通信，如果我们要用 RMAN 的 duplicate from active database 命令创建备库的话。我们需要配置监听和 TNS 名。你可以手动配置，也可以使用网络配置工具(netca)。虽然数据库会自动注册监听，但如果要使用 RMAN 的 duplicate 命令创建备库，备库必须首先处于 NOMOUNT 状态。在 NOMOUNT 状态下，数据库实例不会自动注册监听，你必须配置静态监听。另外必须要注意的一点是，NOMOUNT 状态下的数据库必须使用专用模式(dedicated server)连接。

两台服务器上的 TNS 名字文件必须配置好，让主备库能用 LOG\_ARCHIVE\_DEST\_N 和 FAL\_SERVER 参数（稍后会介绍这些参数）中的服务名(Service Names）找到对方。

$cd $ORACLE\_HOME/network/admin/

配置监听：

$vi listener.ora

主库加入：

SID\_LIST\_LISTENER=

(SID\_LIST =

(SID\_DESC =

(GLOBAL\_DBNAME = master) #----设置成当前库，这个为数据库名

(ORACLE\_HOME = /opt/oracle/product/11.2.0/db\_1)

(SID\_NAME = orcl) #----ORACLE\_SID

)

)

备库加入：

SID\_LIST\_LISTENER=

(SID\_LIST =

(SID\_DESC =

(GLOBAL\_DBNAME = slave) #----设置成当前库

(ORACLE\_HOME = /opt/oracle/product/11.2.0/db\_1)

(SID\_NAME = orcl) #----ORACLE\_SID

)

)

配置服务名：

$vi tnsnames.ora

主库：

MASTER =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = master)(PORT = 1521))

(CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = master)

)

)

SLAVE =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.229.129)(PORT = 1521))

(CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = slave)

)

)

备库：

SLAVE =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = slave)(PORT = 1521))

(CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = slave)

)

)

MASTER =

(DESCRIPTION =

(ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.229.128)(PORT = 1521))

(CONNECT\_DATA =

(SERVER = DEDICATED)

(SERVICE\_NAME = master)

)

)

注意：修改完后记得重启listener。

$ lsnrctl reload

启动监听后tnsping测试监听服务是否可用

先开放1521端口

$vi /etc/sysconfig/iptables

添加一行：

-A INPUT -m state --state NEW -m tcp -p tcp --dport 1521 -j ACCEPT

重启防火墙：

$/etc/init.d/iptables restart

测试tnsping：

$tnsping slave

### 6)、修改主库的参数

配置主库归档日志位置信息LOG\_ARCHIVE\_DEST\_1

$ mkdir -p /opt/oracle/oradata/master/archivelog/

sqlplus / as sysdba

//SQL>alter system set DB\_UNIQUE\_NAME=master scope=spfile;

配置归档位置：

SQL>alter system set LOG\_ARCHIVE\_DEST\_1='LOCATION=/opt/oracle/oradata/master/archivelog/ VALID\_FOR=(ALL\_LOGFILES,ALL\_ROLES) DB\_UNIQUE\_NAME=master' scope=spfile;

这个命令指定快速恢复区作为归档位置，此归档位置用于在所有数据库角色下归档所有的日志文件。

然后配置重做日志传输到备库：

SQL>alter system set LOG\_ARCHIVE\_DEST\_2='SERVICE=slave lgwr ASYNC NOAFFIRM VALID\_FOR=(ONLINE\_LOGFILES,PRIMARY\_ROLE) DB\_UNIQUE\_NAME=slave' scope=spfile;

这条语句说，如果这是主库，就使用服务名 slave传输在线日志，目标库名叫slave。

另一个要设置的参数是 FAL\_SERVER。这个参数指定当日志传输出现问题时，备库到哪里去找缺少的归档日志。它用在备库接收的到的重做日志间有缺口的时候。这种情况会发生在日志传输出现中断时，比如你需要对备库进行维护操作。在备库维护期间，没有日志传输过来，这时缺口就出现了。设置了这个参数，备库就会主动去寻找那些缺少的日志，并要求主库进行传输。

SQL> alter system set FAL\_SERVER='slave' scope=spfile;

然后我们要让主库知道 Data Guard 配置里的另外一个库的名字：

SQL>alter system set LOG\_ARCHIVE\_CONFIG='DG\_CONFIG=(master,slave)' scope=spfile;

SQL>alter system set LOG\_ARCHIVE\_FORMAT='%s\_%t\_%r.arc' scope=spfile;--设置归档日志格式

SQL>ALTER SYSTEM SET LOG\_ARCHIVE\_MAX\_PROCESSES=30;

SQL>alter system set LOG\_ARCHIVE\_DEST\_STATE\_1=ENABLE;

SQL>alter system set LOG\_ARCHIVE\_DEST\_STATE\_2=ENABLE;

alter system set archive\_lag\_target=1800; --强制日志归档，如果数据库30分未归档，则强制归档。

----可以设置网络延迟时间，以及是否启用压缩特性

alter system set log\_archive\_dest\_2 = 'service=slave LGWR ASYNC valid\_for=(ONLINE\_LOGFILES,PRIMARY\_ROLE) db\_unique\_name=slave compression=enable net\_timeout=20 ';

----如果主备库文件存放位置不一样，则需要使用db\_file\_name\_convert和log\_file\_name\_convert参数，且nofilenamecheck参数弃掉

#DB\_FILE\_NAME\_CONVERT='/opt/oracle/oradata/slave','/opt/oracle/oradata/master' SCOPE=SPFILE; #如数据文件路径不一致，则需加该参数， ##且以备库在前，本地库在后

#LOG\_FILE\_NAME\_CONVERT='/opt/oracle/oradata/slave/','/opt/oracle/oradata/master/' SCOPE=SPFILE; #如

STANDBY\_FILE\_MANAGEMENT=AUTO #自动

### 7)、主库生成standby控制文件与spfile文件

生成standby控制文件：

sql>shutdown immediate

sql>startup mount#只允许管理员操作数据库

sql>alter database create standby controlfile as '/tmp/db11g\_stby.ctl';#控制文件

SQL>create pfile from spfile;#如果执行将会在$ORACLE\_HOME/dbs生成一个“init+例程名.ora”文件，可以通过文本编辑器编辑。与：

sql>create pfile='/tmp/initDB11G\_stby.ora' from spfile;

效果一样一样的

sql>ALTER DATABASE OPEN;

### 8)、Copy the files from the primary to the standby server.(参数文件、密码文件)

备库创建主库对应文件夹

//$ mkdir -p /opt/oracle/oradata/master/archivelog/--日志文件目录

//$ mkdir -p $ORACLE\_BASE/oradata/master –数据文件目录

$ mkdir -p $ORACLE\_BASE/fast\_recovery\_area/master

$ mkdir -p $ORACLE\_BASE/admin/master/adump

将主库文件复制到备库：

$ scp -l 8192 -rp oracle@192.168.229.128:$ORACLE\_BASE/oradata/master/ $ORACLE\_BASE/oradata/

-l是limit限制，这样最大是8192/8=1M速度，是为了解决stalled问题

-rp 循环子目录文件

拷贝控制文件并重命名：

$scp oracle@192.168.229.128:/tmp/db11g\_stby.ctl $ORACLE\_BASE/oradata/master/control01.ctl

$ cp $ORACLE\_BASE/oradata/master/control01.ctl $ORACLE\_BASE/fast\_recovery\_area/master/control02.ctl

# Remote login password file.

$ scp oracle@192.168.229.128:$ORACLE\_HOME/dbs/orapworcl $ORACLE\_HOME/dbs

$ cp $ORACLE\_HOME/dbs/orapwmaster orapwslave####如果主备库的实例名不一样需要将密码文件拷贝一份重命名(重要)

$ # Parameter file.

$ scp oracle@192.168.229.128:/tmp/initDB11G\_stby.ora /tmp/initDB11G\_stby.ora

### 9)、修改备库参数文件

1)、修改pfile文件

$vi /tmp/initDB11G\_stby.ora

orcl.\_\_db\_cache\_size=654311424

orcl.\_\_java\_pool\_size=16777216

orcl.\_\_large\_pool\_size=16777216

orcl.\_\_oracle\_base='/opt/oracle'#ORACLE\_BASE set from environment

orcl.\_\_pga\_aggregate\_target=654311424

orcl.\_\_sga\_target=956301312

orcl.\_\_shared\_io\_pool\_size=0

orcl.\_\_shared\_pool\_size=251658240

orcl.\_\_streams\_pool\_size=0

\*.audit\_file\_dest='/opt/oracle/admin/master/adump'#与主库保持一致，复制主库文件（这个路径应该是可以自定义的）

\*.audit\_trail='db'

\*.compatible='11.2.0.0.0'

\*.control\_files='/opt/oracle/oradata/master/control01.ctl','/opt/oracle/fast\_recovery\_area/master/control02.ctl'#控制文件一定要保证一致，复制主库文件（这两个路径应该是可以自定义的）

\*.db\_block\_size=8192

\*.db\_domain=''

\*.db\_name='master'####一定注意这里，增加一行主库dbname

\*.db\_recovery\_file\_dest='/opt/oracle/fast\_recovery\_area'

\*.db\_recovery\_file\_dest\_size=4070572032

\*.db\_unique\_name='SLAVE'#这里为standby库名

\*.diagnostic\_dest='/opt/oracle'

\*.dispatchers='(PROTOCOL=TCP) (SERVICE=orclXDB)'

\*.fal\_server='MASTER'

\*.log\_archive\_config='DG\_CONFIG=(master,slave)'

\*.log\_archive\_dest\_1='LOCATION=/opt/oracle/oradata/master/archivelog/ VALID\_FOR=(ALL\_LOGFILES,ALL\_ROLES) DB\_UNIQUE\_NAME=slave'

\*.log\_archive\_dest\_2='SERVICE=master lgwr ASYNC NOAFFIRM VALID\_FOR=(ONLINE\_LOGFILES,PRIMARY\_ROLE) DB\_UNIQUE\_NAME=master'

\*.log\_archive\_format='%s\_%t\_%r.arc'

\*.memory\_target=1600126976

\*.open\_cursors=300

\*.processes=150

\*.remote\_login\_passwordfile='EXCLUSIVE'

\*.standby\_file\_management='AUTO'

\*.undo\_tablespace='UNDOTBS1'

2)、恢复pfile文件

sql>CREATE SPFILE FROM PFILE='/tmp/initDB11G\_stby.ora';

$cd $ORACLE\_HOME/dbs

$cat initorcl.ora

orcl.\_\_db\_cache\_size=654311424

orcl.\_\_java\_pool\_size=16777216

orcl.\_\_large\_pool\_size=16777216

orcl.\_\_oracle\_base='/opt/oracle'#ORACLE\_BASE set from environment

orcl.\_\_pga\_aggregate\_target=654311424

orcl.\_\_sga\_target=956301312

orcl.\_\_shared\_io\_pool\_size=0

orcl.\_\_shared\_pool\_size=251658240

orcl.\_\_streams\_pool\_size=0

\*.audit\_file\_dest='/opt/oracle/admin/master/adump'#与主库保持一致，复制主库文件

\*.audit\_trail='db'

\*.compatible='11.2.0.0.0'

\*.control\_files='/opt/oracle/oradata/master/control01.ctl','/opt/oracle/fast\_recovery\_area/master/control02.ctl'#控制文件一定要保证一致，复制主库文件

\*.db\_block\_size=8192

\*.db\_domain=''

\*.db\_name='master'#一定注意这里，增加一行主库dbname

\*.db\_recovery\_file\_dest='/opt/oracle/fast\_recovery\_area'

\*.db\_recovery\_file\_dest\_size=4070572032

\*.db\_unique\_name='SLAVE'

\*.diagnostic\_dest='/opt/oracle'

\*.dispatchers='(PROTOCOL=TCP) (SERVICE=orclXDB)'

\*.fal\_server='MASTER'

\*.log\_archive\_config='DG\_CONFIG=(master,slave)'

\*.log\_archive\_dest\_1='LOCATION=/opt/oracle/oradata/master/archivelog/ VALID\_FOR=(ALL\_LOGFILES,ALL\_ROLES) DB\_UNIQUE\_NAME=slave'

\*.log\_archive\_dest\_2='SERVICE=master lgwr ASYNC NOAFFIRM VALID\_FOR=(ONLINE\_LOGFILES,PRIMARY\_ROLE) DB\_UNIQUE\_NAME=master'

\*.log\_archive\_format='%s\_%t\_%r.arc'

\*.memory\_target=1600126976

\*.open\_cursors=300

\*.processes=150

\*.remote\_login\_passwordfile='EXCLUSIVE'

\*.standby\_file\_management='AUTO'

\*.undo\_tablespace='UNDOTBS1'

3)、然后备库执行

sql>create pfile from spfile;

4)、也可以通过命令修改

SQL> startup nomount;

SQL> CREATE SPFILE FROM PFILE='/tmp/initDB11G\_stby.ora';

SQL> shutdown abort;

SQL> startup nomount;

SQL>alter system set db\_unique\_name=slave scope=spfile;

SQL>alter system set log\_archive\_config='DG\_CONFIG=(master,slave)' scope=spfile;

SQL>alter system set log\_archive\_dest\_1 ='LOCATION=/opt/oracle/oradata/master/archivelog/ VALID\_FOR=(ALL\_LOGFILES,ALL\_ROLES) DB\_UNIQUE\_NAME=slave' scope=spfile;

SQL>alter system set log\_archive\_dest\_2 ='SERVICE=master lgwr ASYNC NOAFFIRM VALID\_FOR=(ONLINE\_LOGFILES,PRIMARY\_ROLE) DB\_UNIQUE\_NAME=master' scope=spfile;

SQL>alter system set fal\_server=master scope=spfile;

SQL> shutdown abort;

SQL> startup nomount

sql>create pfile from spfile;

### 10)、启用物理备用数据库 (在备库上执行)

非active dataguard模式，启用备用数据库：

sql>startup nomount

sql>alter database mount standby database;

(1)启动 redo 应用

SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE DISCONNECT FROM SESSION;

或者，备库置于active dataguard模式下(需要将数据库打开处于只读模式)

SQL> alter database open ; ----默认是以read only模式打开数据库

SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE DISCONNECT FROM SESSION; (adg模式，且开启实时应用 real-time apply，查看数据库状态为READ ONLY WITH APPLY)

SQL> SELECT open\_mode FROM V$DATABASE;

这个命令指示备库开始使用备用日志文件进行恢复。DISCONNECT FROM SESSION它也告诉备库命令完成后回到命令行界面

using current logfile disconnect from session和disconnect from session区别：前者会实时同步日志（也就是active dataguard模式），即使该日志没有归档，也会应用到备库；而后者必须要归档后才会应用到备库

SQL> alter database open;

alter database open

\*

ERROR at line 1:

ORA-10456: cannot open standby database; media recovery session may be in

Progress

当前我们在应用日志同步的状态。在进行同步的时候，是不能open的。我们需要首先将这个日志同步动作cancel中。

SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;

此时只是暂时 redo 应用，并不是停止 Standby 数据库，standby 仍会保持接收只不过不会再应用接收到的归档，直到你再次启动 redo 应用为止（备库仍旧会接受日志，但不会应用到实例）。这时，主备库APPLIED列会显示为NO

SQL> select SEQUENCE#, FIRST\_TIME, NEXT\_TIME, APPLIED, ARCHIVED from V$ARCHIVED\_LOG order by FIRST\_TIME;

(2)停止standby

正常情况下，首先

SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;

然后再sql>shutdown immediate

当然也可以直接shutdown immediate

(3)备用服务器的管理模式与只读模式

<1>.启动到管理模式

SQL>shutdown immediate;

SQL>startup nomount;

SQL>alter database mount standby database;

SQL>alter database recover managed standby database disconnect from session;

<2>.启动到只读方式

SQL>shutdown immediate;

SQL>startup nomount;

SQL>alter database mount standby database;

SQL>alter database open read only;

<3>如果在管理恢复模式下到只读模式

SQL>alter database recover managed standby database cancel;

SQL> alter database open read only;

这个时候，可以给数据库增加临时数据文件（这个在热备份的时候是没有备份过来的）

如alter tablespace temp add tempfile '/u02/oradata/test/temp01.dbf' size 100M;

<4>.从只读方式到管理恢复方式

SQL> alter database recover managed standby database disconnect from session;

(4)应用物理备库的几点监控

如果上面出了问题或者我们不知道成功了没有，可以用下面的方法检测

<1>确认主备库里的归档目的地配置都是有效的

select DEST\_ID, STATUS, DESTINATION, ERROR from V$ARCHIVE\_DEST where DEST\_ID<=2;

目的地状态status应该显示为 VALID，注意如果上面没有执行redo应用会有一条error信息

<2>确认重做日志是否真的被应用了，在主库执行

select SEQUENCE#, FIRST\_TIME, NEXT\_TIME, APPLIED, ARCHIVED from V$ARCHIVED\_LOG order by FIRST\_TIME;

如果归档和日志应用均正常，APPLIED 和 ARCHIVED 列都应该是 YES。(如果没有应用redo，applied应该是NO)

很多教程里都让这个查询以 SEQUENCE# 列排序，但我不推荐。如果以 SEQUENCE# 列排序，当你做了一次故障转移后，序列号会再从1开始，这时使用这个查询，你将不能在结果最后看到最新的记录。我曾经很奇怪为什么查不到新记录，其实是因为新记录不是出现在最后，我没看到。所以，这个查询都是以 FIRST\_TIME 列排序.

<3>主库上检查是否有重做日志缺口

如果你发现日志没有被应用，那可能是重做日志有了缺口，这种情况下备库无法进行日志应用。但如果你的 FAL\_SERVER 参数设置正确，这应该不会有问题

select STATUS, GAP\_STATUS from V$ARCHIVE\_DEST\_STATUS where DEST\_ID = 2;

如果一切正常，应该返回 VALID 和 NO GAP .切记启用redo应用才能显示No GAP

<4>在主备库上执行以下查询查看数据库状态

sql>select \* from V$DATAGUARD\_STATUS order by TIMESTAMP;

<5>检查是否成功：

主库上查看日志传送情况：

sql>select dest\_name,status,error from v$archive\_dest;

应该log\_archive\_dest\_1和2状态应该是valid

切换几次日志：

sql>alter system switch logfile;

查看日志序号：

sql>select sequence# from v$archived\_log;

备库验证：

sql>select sequence#,applied from v$archived\_log;

**验证物理备库日志应用**

**1）主库上操作**

**SQL> conn / as sysdba;**

**SQL> create user abc identified by abc ;**

**SQL> grant dba to abc;**

**SQL> conn abc/abc**

**SQL> create table abc ( id integer , name char(10));**

**SQL> insert into abc values ( 0 , 'aaa' );**

**SQL> commit;**

**SQL> conn / as sysdba;**

**SQL> archive log list;**

**2）备库上验证**

**SQL> archive log list;**

**SQL> select sequence#,first\_time,next\_time,applied from v$archived\_log order by sequence#;**

##Verify the Physical Standby Database Is Performing Properly

1)Identify the existing archived redo log files.

On the standby database, query the V$ARCHIVED\_LOG view to identify existing files in the archived redo log. For example:

SQL> SELECT SEQUENCE#, FIRST\_TIME, NEXT\_TIME FROM V$ARCHIVED\_LOG ORDER BY SEQUENCE#;

SEQUENCE# FIRST\_TIME NEXT\_TIME

---------- ------------------ ------------------

8 11-JUL-07 17:50:45 11-JUL-07 17:50:53

9 11-JUL-07 17:50:53 11-JUL-07 17:50:58

10 11-JUL-07 17:50:58 11-JUL-07 17:51:03

3 rows selected.

2)Force a log switch to archive the current online redo log file.

On the primary database, issue the ALTER SYSTEM SWITCH LOGFILE statement to force a log switch and archive the current online redo log file group:

SQL> ALTER SYSTEM SWITCH LOGFILE;

3)Verify the new redo data was archived on the standby database.

On the standby database, query the V$ARCHIVED\_LOG view to verify the redo data was received and archived on the standby database:

SQL> SELECT SEQUENCE#, FIRST\_TIME, NEXT\_TIME FROM V$ARCHIVED\_LOG ORDER BY SEQUENCE#;

SEQUENCE# FIRST\_TIME NEXT\_TIME

---------- ------------------ ------------------

8 11-JUL-07 17:50:45 11-JUL-07 17:50:53

9 11-JUL-07 17:50:53 11-JUL-07 17:50:58

10 11-JUL-07 17:50:58 11-JUL-07 17:51:03

11 11-JUL-07 17:51:03 11-JUL-07 18:34:11

4 rows selected.

The archived redo log files are now available to be applied to the physical standby database.

4)Verify that received redo has been applied.

On the standby database, query the V$ARCHIVED\_LOG view to verify that received redo has been applied:

SQL> SELECT SEQUENCE#,APPLIED FROM V$ARCHIVED\_LOG ORDER BY SEQUENCE#;

SEQUENCE# APP

--------- ---

8 YES

9 YES

10 YES

11 IN-MEMORY

4 rows selected.

Note:

The value of the APPLIED column for the most recently received log file will be either IN-MEMORY or YES if that log file has been applied.

### 11)、将备库置于Active DataGuard模式，体验实时查询

在Oracle 11g之前，物理备库（physical Standby）在应用redo的时候，数据库需要处于mount状态。从11g开始，应用redo的时候，物理备库可以处于read-only模式，这就称为Active Data Guard 。通过Active Data Guard，可以在物理备库进行查询或者导出数据，从而减少对主库的访问和压力。

1. 备库上操作

1) 查看备库当前状态 mount

SQL> select open\_mode,database\_role,db\_unique\_name from v$database;

OPEN\_MODE DATABASE\_ROLE DB\_UNIQUE\_NAME

-------------------- ---------------- ------------------------------

MOUNTED PHYSICAL STANDBY DG

2) 取消备库的自动恢复

SQL> alter database recover managed standby database cancel;

数据库已更改。

3) OPEN备库为只读模式（Dataguard只能启动到readonly模式）

SQL> alter database open;

数据库已更改。

SQL> select open\_mode from v$database;

OPEN\_MODE

--------------------

READ ONLY

4）打开实时应用状态模式

SQL> alter database recover managed standby database using current logfile disconnect;

数据库已更改。

SQL> select open\_mode,database\_role,db\_unique\_name from v$database;

OPEN\_MODE DATABASE\_ROLE DB\_UNIQUE\_NAME

-------------------- ---------------- ------------------------------

READ ONLY WITH APPLY PHYSICAL STANDBY DG

2. 主库上操作执行DDL,DML操作

# su - oracle

$ sqlplus /nolog

SQL> conn / as sysdba;

SQL> create tablespace abc datafile '/opt/oracle/oradata/master/abc.dbf' size 10m autoextend on next 10m;

表空间已创建。

SQL> conn abc/abc

已连接。

SQL> insert into abc values (1 , 'bbb');

已创建 1 行。

SQL> commit;

提交完成。

SQL>

3. 备库上查看

SQL> select name from v$tablespace;

NAME

------------------------------

SYSTEM

SYSAUX

UNDOTBS1

USERS

TEMP

ABC

已选择6行。

SQL> conn abc/abc;

已连接。

SQL> select \* from abc;

ID NAME

---------- ----------

0 aaa

1 bbb

主备的修改很快在备库上应用。

### 12)、dataguard启动关闭顺序

(1)监听

先启从库再起主库

#lsnrctl start

(2)启动

先启从库：

sql>startup nomount

sql>alter database mount standby database;

sql>alter database recover managed standby database using current logfile disconnect from session;

在启主库

sql>startup

启动Oracle Enterprise Manager：

查看dbconsole进程的状态：

**emctl status dbconsole**

启动 dbconsole 进程：

emctl start dbconsole

(3)关闭：和开启正好相反

先关主库：

sql>shutdown immediate

再关从库：

sql>alter database recover managed standby database cancel;

sql>shutdown immediate;

### 13)、总结

其实最重要的就三个文件(密码文件、控制文件、pfile文件)，前两个与主备库一致，第三个db\_name与主库一致，其他部分配置需要按需修改。

对standby而言，standby redo log就是online redo log。Oracle推荐standby redo log大小要与Primary上redo log size相匹配的。

## 详细的冷备搭建DataGuard文档

[**oracle11g dataguard 完全手册**](http://www.cnblogs.com/tippoint/archive/2013/04/18/3029019.html)

[Oracle 11g DataGuard 物理备库配置及Active DataGuard测试](http://koumm.blog.51cto.com/703525/1263870)

[Creating a Physical Standby Database](http://docs.oracle.com/cd/B28359_01/server.111/b28294/create_ps.htm#i77130)

[Oracle 11g Dataguard（物理备库安装实验）](http://blog.itpub.net/9515241/viewspace-1807664/)

[Oracle 11g Dataguard broker配置](http://koumm.blog.51cto.com/703525/1275237)

## 主备切换(switchover)

### 1)、查看库所在的保护模式和身份

SQL > select database\_role,protection\_mode,protection\_level from v$database;

SQL > alter database set standby database to maximize protection; --最大保护

SQL > alter database set standby database to maximize availability; --最高可用性

SQL > alter database set standby database to maximize performance; --最高性能

一定要在最高性能模式下，才能执行主备切换

### 2)、故障转移配置

现在你已经配好了一个物理备库，你可能想试试主备切换(switchover)，甚至故障转移(failover)，但你先得确定客户端会跟着切换和转移。

我们需要配置数据库和客户端来支持这些功能。要确定你的客户端能连接到正确的数据库，你要在数据库里配置一个支持故障转移的服务，并配置客户端的 TNS，让它知道如何在一个 Data Guard 集群里找到主库。

(1)主库创建自动转移服务:

此服务在数据库出现故障时会发送通知给客户端，允许查询语句在故障转移发生后继续运行。我使用命名 SID\_RW 显示这时一个可读写的数据库（主库）。

begin

DBMS\_SERVICE.CREATE\_SERVICE (

service\_name => 'orcl\_RW',

network\_name => 'orcl\_RW',

aq\_ha\_notifications => TRUE,

failover\_method => 'BASIC',

failover\_type => 'SELECT',

failover\_retries => 30,

failover\_delay => 5);

end;

(2)创建存储过程确保只在主库运行

我们创建一个存储过程来实现此目的，如果当前数据库是主库它就启动此服务，如果是备库就停止。

create or replace procedure cmc\_taf\_service\_proc

is

v\_role VARCHAR(30);

begin

select DATABASE\_ROLE into v\_role from V$DATABASE;

if v\_role = 'PRIMARY' then

DBMS\_SERVICE.START\_SERVICE('orcl\_RW');

else

DBMS\_SERVICE.STOP\_SERVICE('orcl\_RW');

end if;

end;/

若编译执行报错，为权限问题

SQL > GRANT SELECT ANY DICTIONARY TO SYSTEM/PUBLIC;

SQL > GRANT EXECUTE ON DBMS\_SERVICE TO SYSTEM/PUBLIC;

(3)创建触发器（2个）来确保服务可以运行

创建两个触发器，让数据库在启动和角色转换时运行此存储过程。注意有的文档说是建立一个，那可能会导致你在重启你的数据库时，它不会重启故障转移服务

create or replace TRIGGER cmc\_taf\_service\_trg\_startup

after startup on database

begin

cmc\_taf\_service\_proc;

end;

create or replace TRIGGER cmc\_taf\_manage\_trg\_rolechange

after db\_role\_change on database

begin

cmc\_taf\_service\_proc;

end;

（4）启动服务

我们执行一次存储过程，确定服务正在运行，并归档当前日志，让以上更改同步到备库。

SQL> exec cmc\_taf\_service\_proc;

SQL> alter system archive log current;

查看服务信息

SQL> show parameter service\_names

(5)客户端配置TNS

服务名存在还不够，你必须配置客户端的 TNS 名去连接它。客户端的 TNS 名应该类似如下：

orcl\_RW =

(DESCRIPTION =

(ADDRESS\_LIST=

(ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.229.128)(PORT = 1521))

(ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.229.129)(PORT = 1521))

)

(CONNECT\_DATA = (SERVICE\_NAME = orcl\_RW)

(FAILOVER\_MODE=(TYPE=SELECT)(METHOD=BASIC)(RETRIES=30)(DELAY=5))

)

)

当你的客户端使用新 TNS 名后，它们能在主备切换和故障转移操作后找到主库。如果客户端在运行一个查询，并且没有 DML 是在一个交易中，那在发生切换操作后，只要主备转换和故障转移在超过最大重试次数前完成，这个查询会继续工作，只是会有延迟。你应该多做几次切换实验，以确定 RETRIES 和 DELAY 参数如何设置合适。

如果有一个正在进行中的交易，当客户端连接到新的主库后，查询将报错（`ORA-25403: transaction must roll back），并回滚。

### 3)、主备切换switchover

(1)主库确认没有日志缺口

SQL> select STATUS, GAP\_STATUS from V$ARCHIVE\_DEST\_STATUS where DEST\_ID <= 2;

应该返回 VALID 和 NO GAP。如果不是NO GAP，是LOG SWITCH GAP ，是不能进行主备切换的。

将GAP\_STATUS状态由LOG SWITCH GAP修改为NO GAP：

SQL> shutdown immediate;

SQL> startup;

SQL> ALTER SYSTEM FLUSH REDO TO 'slave';#刷新日志到备库

GAP\_STATUS字段值意义：

NO GAP：没有gap

LOG SWITCH GAP：有gap，日志还有切换完成，不需要人工干预

RESOLVABLE GAP：有gap，但是可以自动解决，不需要人工干预

UNRESOLVABLE GAP：有gap，不能自动解决，需要人工干预

LOCALLY UNRESOLVABLE GAP：有gap，不能从本机获取日志自动解决，可能可以从其它机器获取日志自动解决

(2)查询v$tempfile视图确认备库的临时文件和主库一样。

(3)确认所有重做日志都已在备库应用，查询备库

查询备库之前，先确认重做日志是否真的被应用了，在主库执行

select SEQUENCE#, FIRST\_TIME, NEXT\_TIME, APPLIED, ARCHIVED from V$ARCHIVED\_LOG order by FIRST\_TIME;

SQL> select NAME, VALUE, DATUM\_TIME from V$DATAGUARD\_STATS;# 查询备库

不应该返回 transport lag 或 apply lag, finish time 应该为0.

关于V$DATAGUARD\_STATS是这样描述的：

该动态性能视图显示出在主库上产生了多少重做日志数据，但是还没有被备库所应用。

所以，通过查询该视图可以基本确定如果万一主库出现崩溃的话，备库上将丢失多少重做日志数据。

我们可以在一套Dataguard环境下的任一备库的实例上从该视图里获取相关信息，

然而，在主库的实例上查询该视图返回的信息都将是空。也就是说，只可以从备库的实例上查询V$DATAGUARD\_STATS，从主库实例上是看不到任何有用信息的。

NAME：

apply lag,该值表示在通过在备库上应用主库传递过来的重做日志与主库同步所延迟的时间。

transport lag,该值表示在单位时间内主库上产生的重做日志还没有传输到备库上，或者主库上产生的重做日志还没有被备库所应用。

apply finish time,该值表示在备库上完成应用重做日志所需要的时间。

estimated startup time,该值表示启动和打开物理备库所需要的时间，该字段不是适用于逻辑备库。

standby has been open,该值表示物理备库自从上次启动以来，是否以OPEN READ ONLY方式打开过？

该参数值如果是Y，现在需要做FAILOVER，那么就需要先将该物理备库shutdown然后以OPEN READ WRITE方式打开。

11g的dataguard可以一边OPEN READ ONLY，一边执行redo apply,也就是11g 的ACTIVE Dataguard。

VALUE：给出各个参数的值。如第1个查询中的，apply finish time值为+00 00:00:00.1，说明该物理备库需要0.1秒的时间来完成应用剩余的重做日志数据。

UNIT：各个参数的时间单元。

TIME\_COMPUTED：物理备库上估算各个参数的本地时间。

DATUM\_TIME：在物理备库上获取元数据来估算 APPLY LAG 和 TRANSPORT LAG 这两个参数值的本地时间。如果从多次查询中看到该时间值对应的APPLY LAG 和 TRANSPORT LAG 这两个参数值保持不变的话，那么就说明该物理备库已经停止从主库接收到重做数据！该字段是11g中新出现的。

(4)确认主库可以进行角色切换，查询主库

SQL> select SWITCHOVER\_STATUS from V$DATABASE;

如果返回 TO STANDBY 或 SESSIONS ACTIVE，那么主库就可以进行切换

附一份DG主备库切换状态说明：

v$database

NOT ALLOWED - On a primary database, this status indicates that there are no valid and enabled standby databases. On a standby database, this status indicates that a switchover request has not been received from the primary database.

SESSIONS ACTIVE - The database has active sessions. On a physical standby database, the WITH SESSION SHUTDOWN SQL clause must be specified to perform a role transition while in this state. On a logical standby database, a role transition can be performed while in this state, but the role transition will not complete until all current transactions have committed.

SWITCHOVER PENDING - On a physical standby database, this status indicates that a switchover request has been received from the primary database and is being processed. A physical standby database cannot switch to the primary role while in this transient state.

SWITCHOVER LATENT - On a physical standby database, this status indicates that a switchover request was pending, but the original primary database has been switched back to the primary role.

TO PRIMARY - The database is ready to switch to the primary role.

TO STANDBY - The database is ready to switch to either the physical or logical standby role.

TO LOGICAL STANDBY - The database has received a data dictionary from a logical standby database and is ready to switch to the logical standby role.

RECOVERY NEEDED - On a physical standby database, this status indicates that additional redo must be applied before the database can switch to the primary role.

PREPARING SWITCHOVER - On a primary database, this status indicates that a data dictionary is being received from a logical standby database in preparation for switching to the logical standby role. On a logical standby database, this status indicates that the data dictionary has been sent to the primary database and other standby databases.

PREPARING DICTIONARY - On a logical standby database, this status indicates that the data dictionary is being sent to the primary database and other standby databases in preparation for switching to the primary role.

FAILED DESTINATION - On a primary database, this status indicates that one or more standby destinations are in an error state.

RESOLVABLE GAP - On a primary database, this status indicates that one or more standby databases have a redo gap that can be automatically resolved by fetching the missing redo from the primary database or from another standby database.

UNRESOLVABLE GAP - On a primary database, this status indicates that one or more standby databases have a redo gap that cannot be automatically resolved by fetching the missing redo from the primary database or from another standby database.

LOG SWITCH GAP - On a primary database, this status indicates that one or more standby databases are missing redo due to a recent log switch.

以下是一些消除主备库之间GAP的命令和说明：

--主库，将所有未传送的redo传送给从库，target\_db\_name使用DB\_UNIQUE\_NAME 。

ALTER SYSTEM FLUSH REDO TO ‘target\_db\_name’;

--验证备库

SELECT UNIQUE THREAD# AS THREAD, MAX(SEQUENCE#) OVER (PARTITION BY thread#) AS LAST from V$ARCHIVED\_LOG;

--如果必要，拷贝归档日志到从库，并进行注册

ALTER DATABASE REGISTER PHYSICAL LOGFILE 'filespec1';

--重复上一步，知道确认所有归档完毕。

SELECT THREAD#, LOW\_SEQUENCE#, HIGH\_SEQUENCE# FROM V$ARCHIVE\_GAP;

--查看目标日志传输路径状态和GAP状态

SELECT STATUS, GAP\_STATUS FROM V$ARCHIVE\_DEST\_STATUS WHERE DEST\_ID = 2;

--在目标备库上，停止日志应用

ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;

--在目标备库上

ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;

--如果日志确定丢失，可以采用激活方式，但这样会有数据丢失。

--ALTER DATABASE ACTIVATE PHYSICAL STANDBY DATABASE;

--验证目标备库

SELECT SWITCHOVER\_STATUS FROM V$DATABASE;

--开始切换，如果状态为“TO PRIMARY”，则WITH SESSION SHUTDOWN从句可以去掉。

ALTER DATABASE COMMIT TO SWITCHOVER TO PRIMARY WITH SESSION SHUTDOWN;

--打开新主库

ALTER DATABASE OPEN;

(5)切换

<1>切换主库为备库命令为：

如果执行为TO STANDBY执行

SQL> alter database commit to switchover to physical standby ;

#注意，在主库切换为备库时，是不能有多余一个的sesson连接的，不然会报错。

SQL> alter database commit to switchover to physical standby ;

alter database commit to switchover to physical standby

\*

ERROR at line 1:

ORA-01093: ALTER DATABASE CLOSE only permitted with no sessions connected

可以先杀掉会话，

select username,machine,count(\*) from v$session group by username,machine;#查询不同用户的会话

select sid,serial#,username, machine,status from v$session;#找到用户会话的sid,serial#

ALTER SYSTEM KILL SESSION 'v\_sid,v\_serial#' immediate;#杀掉会话

再执行

SQL> alter database commit to switchover to physical standby ;

或则在执行时加上with session shutdown：

SQL>ALTER DATABASE COMMIT TO SWITCHOVER TO PHYSICAL STANDBY WITH SESSION SHUTDOWN;

SQL> shutdown immediate;

SQL> startup mount;

如果为SESSIONS ACTIVE执行

SQL> alter database commit to switchover to physical standby with session shutdown;

SQL> shutdown immediate;

SQL> startup mount;

<2>查询备库是否可以切换为主库，查询备库：

SQL> select SWITCHOVER\_STATUS from V$DATABASE;

如果返回 TO PRIMARY 或 SESSIONS ACTIVE，就可以切换。如果返回 SWITCHOVER LATENT 或 SWITCHOVER PENDING，就要去检查告警日志，

看有什么问题，一般是需要应用一些日志。如果是需要应用日志的话，在备库执行如下命令：

SQL> recover standby database using backup controlfile;

完成应用后，会变成 TO PRIMARY 或 SESSIONS ACTIVE状态。

或者alter database recover managed standby database disconnect from session;

alter database recover managed standby database cancel；

切换备库为主库了：

SQL> select SWITCHOVER\_STATUS from V$DATABASE;

如果是TO\_PRIMARY 执行下面的

sql> alter database commit to switchover to primary;

SQL> alter database open;

如果是SESSIONS ACTIVE，执行下面的：

SQL> alter database commit to switchover to primary with session shutdown;

SQL> alter database open;

备库(现在的备库即101)上启用日志应用：

SQL> alter database recover managed standby database using current logfile disconnect from session;

此时在现在的备库101上执行：

sql>select SWITCHOVER\_STATUS,DATABASE\_ROLE from V$DATABASE; --not allowed PHYSICAL STANDBY

ora-16014错误，删除standby logfile然后alter database clear logfile清理数据库

从现在102主库上alter system switch logfile手工触发去传输standby logfile

注意的是：如果现在把101在切换回主库，102成从库。按照正常操作，先切换现在的主库102成从库，当操作完成的时候，101的 SWITCHOVER\_STATUS就成

to\_primary也就是说可以切换成主库了。

现在的主库102上执行：

sql> select SWITCHOVER\_STATUS,DATABASE\_ROLE from V$DATABASE; --RESOLVABLE GAP PRIMARY

检查alert日志发现ora-00312错误，删除错误的log日志,后状态为TO STANDBY，PRIMARY

sql>startup mount;

sql>alter database clear logfile '\*\*\*\*\*.LOG';

sql>alter database open

（6）问题

如果主备库的sequence#不同，而且主库切换switch logfile也不能传送到备库，可用下面的办法

手工同步归档日志文件到备库，然后在备库执行

sql>alter database register logfile '/opt/oracle/oradata/orcl/archive1/'

会报错，不需要管，会提示可以注册的已经注册进去了

sql>select sequence#,first\_change#,next\_change#,applied,activation# from v$archived\_log;

检查主备库sequence#，然后主库切换logfile可以看到已经可以同步了

至此，整个SWITCHOVER就做完了，SWITCHOVER是不会丢失数据的(不同于FAILOVER)，如果FAILOVER能做到没有GAP，也可以不丢数据，并且正常使用SWITCHOVER切换

### 4)、总结

主备切换需要在满足以下条件时，才能成功切换：

(1) select database\_role,protection\_mode,protection\_level from v$database;##保证最高性能模式 (MAXIMUM PERFORMANCE)

(2) select STATUS, GAP\_STATUS from V$ARCHIVE\_DEST\_STATUS where DEST\_ID <= 2;# 主库确认没有日志缺口 (NO GAP)

(3) select SWITCHOVER\_STATUS from V$DATABASE;# 确认主库可以进行角色切换 (TO STANDBY)

## 故障转移(failover)

一般情况下执行failover都是主库已经game over。故障转移将备库转换为主库，但不把原主库（有故障，无法正常工作）切换为备库。当故障转移发生后，你必须重建主库，或者使用闪回数据库功能将主库回退到故障发生前，然后转换其为备库并启用日志应用。

### 1)、执行failover的前提

(1)检查归档文件是否连续

查询待转换 standby 数据库的 V$ARCHIVE\_GAP 视图，确认归档文件是否连接：

SQL> SELECT THREAD#, LOW\_SEQUENCE#, HIGH\_SEQUENCE# FROM V$ARCHIVE\_GAP;

未选定行

如果返回的有记录， 按照列出的记录号复制对应的归档文件到待转换的 standby 服务器。 这一步非常重要，必须确保所有已生成的归档文件均已存在于 standby 服务器，不然可能会数据不一致造成转换时报错。

文件复制之后，通过下列命令将其加入数据字典：

SQL>ALTER DATABASE REGISTER PHYSICAL LOGFILE 'filespec1';

(2)检查归档文件是否完整

分别在 primary/standby 执行下列语句：

SQL> select distinct thread#,max(sequence#) over(partition by thread#) a from v$archived\_log;

该语句取得当前数据库各线程已归档文件最大序号，如果 primary 与 standby 最大序号不相同，必须将多出的序号对应的归档文件复制到待转换的 standby 服务器。 不过既然是 failover， 有可能 primary 数据库此时已经无法打开，甚至无法访问,那只好听天由命了，但是如果可以到mounted，则可以传输归档日志。

所以应该在primary机器crashed之前做好准备。

假如此时结果是：

THREAD# A

---------- ----------

1 87

从库最大为87，主库也执行，如果也是87则ok，如果不一致需要scp 然后执行register

$scp oracle@192.168.229.128:/opt/oracle/oradata/master/archivelog/ /opt/oracle/oradata/slave/archivelog/

在备库注册丢失的日志文件:

SQL> alter database register logfile '/opt/oracle/oradata/slave/archivelog/9\_1\_919095053.arc';

(3)、如果当前备库是处于最大保护（maximum protection）模式，要进行故障转移，必须先修改为最大性能（maximum performance）模式。

SQL>select database\_role,protection\_mode,protection\_level from v$database;

修改方法：

SQL> alter database set standby database to maximize performance;

### 2)、执行切换

(1)在备库停止日志应用：

SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE CANCEL;

(2)结束应用任何日志：

SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE FINISH;

或者：

SQL> alter database recover managed standby database finish force;

FORCE 关键字将会停止当前活动的 RFS 进程，以便立刻执行 failover。

(3)转换备库为主库

SQL> alter database commit to switchover to primary with session shutdown;

SQL> alter database open;

注意这个时候主库已经不在dataguard中了。

### 3)、使用闪回重建数据库

(1)获得原备库转换为主库时的 SCN。查询现主库102：

SQL> SELECT to\_char(STANDBY\_BECAME\_PRIMARY\_SCN) from V$DATABASE;

(2)在原主库101上执行：

SQL> SHUTDOWN IMMEDIATE;

SQL> STARTUP MOUNT;

SQL> FLASHBACK DATABASE TO SCN &standby\_became\_primary\_scn;输入之前查询到的SCN号

如果没有开始闪回日志，最后一条命令会报错

ORA-38726: Flashback database logging is not on.，

无法进行闪回。你就需要重新从新主库复制数据，重建原主库为备库。

如果命令成功执行，原主库101就可以使用新主库102的日志进行恢复。将原主库101转换成物理备库，并启动日志应用进程：

SQL> ALTER DATABASE CONVERT TO PHYSICAL STANDBY;

SQL> SHUTDOWN IMMEDIATE;

SQL> STARTUP MOUNT;

SQL> ALTER DATABASE RECOVER MANAGED STANDBY DATABASE USING CURRENT LOGFILE DISCONNECT FROM SESSION;

### 4)、客户端故障转移

我很喜欢的一个功能是，当主备切换或故障转移发生后，客户端能够自动重连。在你的系统里，做下述实验，看结果如何。

master是主库，slave是备库。客户端使用支持故障转移的 orcl\_RW 服务名。

首先，用 SYSTEM 用户和 orcl\_RW 服务名在 SQL\*Plus 里登录主库：

SQL> connect system@orcl\_rw

SQL> select db\_unique\_name from v$database;

查询应该返回主库名 orcl。然后做一次主备切换，在将备库转换为主库

alter database commit to switchover to primary with session shutdown; 这一步时，主备库均处于 MOUNT 状态。

然后执行查询：

SQL> select db\_unique\_name from v$database;

这时查询应该挂住，这是因为客户端正在尝试寻找主库，但当前又没有可用的主库。然后继续完成主备切换。

当主备切换完成后，客户端应该会重连并重新执行查询，查询完成后成功返回结果 slave，因为现在主库已经切换为 slave，不再是master。

另一个很酷的测试方法是，执行一个执行时间非常长的查询，当查询结果返回，屏幕一直滚动时开始主备切换。

你应该会看到屏幕暂停滚动一段时间，当切换完成后，又会继续滚动。

### 5)、活动数据卫士（active dataguard)

警告！活动数据卫士功能需要单独的授权。虽然打开此功能很容易，如果没有授权，你不应该使用它。

活动数据卫士是 11g 的新功能，它允许你的物理备库在应用日志时处于只读打开状态。这明显是一个很有用的功能。

能够允许主库有一个物理备库作为备份，并能在保持备库数据更新的同时读取备库，这是一个很好的功能。

调整备库为“READ ONLY WITH APPLY”状态，这体现的便是Oracle 11g物理Active Data Guard功能中的“Active”真实含义。

在实际工作中，这个功能非常好用，经常用于检查实时数据同步情况。

(1)查看备库openmode

sql>select open\_mode from v$database;--MOUNTED

(2)取消备库的自动恢复

sql>alter database recover managed standby database cancel;

sql>alter database open;

sql>select open\_mode from v$database; --这时候为READ ONLY

(3)read only下备库恢复

sql>alter database recover managed standby database using current logfile disconnect;

sql>select open\_mode from v$database; --这时为READ ONLY WITH APPLY

### 6)、11g standby的一些变化

(1)standby数据库可以直接执行

startup --就是read only打开数据库

startup mount --就是mount standby数据库

(2)read only的情况下可以执行managed recover standby命令。

SQL>startup

SQL>alter database recover managed standby database disconnect;

### 7)、关于备份的引申

有了 Data Guard，你的 RMAN 备份在主库或者备库都可以执行。但既然你已经配置了物理备库，你应该减轻点主库的负载。

基本上，能在主库执行的标准的备份命令或脚本，也能在备库执行，但也有几个值得注意地方。这些 Oracle 官方文档都有，我只提几个关键的事情：

首先，你应该使用恢复目录（Recovery Catalog）。这是因为主库需要知道备库已经存在了哪些备份文件。你不需要在恢复目录中注册备库，恢复目录能认出它是备库。

其次，你不能备份备库的控制文件，所以不要在主库关掉所有的备份，至少需要在主库备份控制文件和参数文件。

备份和恢复可以写一整篇文章，我只是讲下我是如何配置备份的，让你能从这里开始，修改并形成自己的策略。测试下，确认你能从你当前实现的备份设置中恢复。在运行备份前，你需要配置一些基本的东西。

(1)确认开启控制文件和 spfile 自动备份：

RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;

(2)根据你的需要设置备份文件保留策略：

RMAN> CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 3 DAYS;

(3)如果一个文件已经有备份，并且检查点 SCN 相同，就不备份：

RMAN> CONFIGURE BACKUP OPTIMIZATION ON;

(4)只在主库的归档日志已经在备库应用（或者配置为已经传输到备份）后才删除：

RMAN> CONFIGURE ARCHIVELOG DELETION POLICY TO APPLIED ON ALL STANDBY;

(5)允许 RMAN 在主备间重新同步：

RMAN> CONFIGURE DB\_UNIQUE\_NAME P10AC CONNECT IDENTIFIER ‘master’;

RMAN> CONFIGURE DB\_UNIQUE\_NAME P11AC CONNECT IDENTIFIER 'slave';

在主库我仍然备份归档日志。首先，在主备库都备份归档提供了冗余。其次，当发生需要恢复的事件（比如数据文件下线等）后，我在主库已经有归档了。我需要删除过期的归档，以清理磁盘空间。

在 Data Guard 环境下，不能使用标准的在单机删除归档的命令，两者有一点小区别。因为我们必须使用恢复目录，

我创建了一个全局脚本（global script）：

create global script dg\_primary\_arch

{

backup archivelog all;

delete noprompt archivelog all completed before 'sysdate-.5';

delete noprompt backup of archivelog all completed before 'sysdate-2';

}

在备库我运行标准的全库和归档备份，并删除过期的备份集。在 Data Guard 环境下，将备份归档包含在备份全库的命令里，会经常导致报错

RMAN-08137: WARNING: archived log not deleted, needed for standby or upstream capture process。

为避免此报错，你应该将备份归档放在单独的命令里。我还是创建一个全局脚本：

create global script dg\_standby\_full

{

backup database plus archivelog;

delete noprompt archivelog all completed before 'sysdate-1';

delete noprompt obsolete;

}

另外一个有用的技巧是，如果可能，使用共享文件系统进行备份。这样你在两台服务器上都可以访问备份文件。

这样，当你需要恢复时，你不需要从另一台服务器上复制文件了。但如果使用共享文件系统的话，你的归档备份虽然有两份，却都放在一个文件系统里，如果硬盘出现故障，两份备份都会丢失。

参考文档：

1.《Oracle 11g Data Guard 物理备库快速配置指南》

2.《三思笔记：一步一步学dataguard》

3.《Red\_HatOracle11gDataGuard配置宝典》

4.其他网络资料

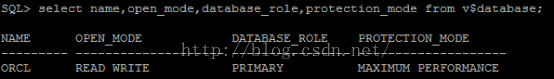
## Oracle逻辑备库配置

Oracle Data Guard逻辑备库是利用主库的一个备份首先建立一个物理备库，然后再将其转换为逻辑备库。这之后主库将日志传递到备库，备库利用logminer从主库的日志中解析出主库所执行过的SQL，在备库上重新执行一遍，从而保证与主库的数据在逻辑上保持一致。与物理备库相对应的是，物理备库使用的是redo apply，逻辑备库使用的是sql apply。因此逻辑备库仅仅保证数据与主库是在逻辑上是一致的，从而逻辑备库可以处于open状态下并进行相应的DML操作。

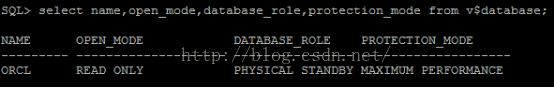
在转换physical备库到logical备库之前，先来看看主备库的情况：

1. SQL> **select** **name**,open\_mode,database\_role,protection\_mode **from** v$**database**;

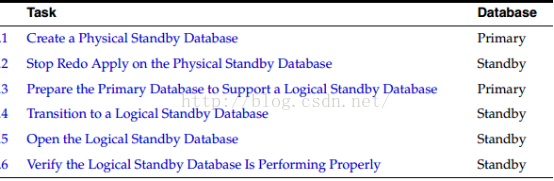
* Primary库



* Physical备库



### 1.1 Step-by-Step Instructions for Creating a Logical Standby Database



   由于逻辑Standby是通过SQL应用来保持与Primary数据库的同步。SQL应用与REDO应用是有很大的区别，REDO应用实际上是在物理Standby端进行RECOVER；SQL应用则是分析重做日志文件中的REDO信息，并将其转换为SQL语句，在逻辑Standby端执行，因此需要**确认操作的对象和语句是否能被逻辑Standby支持。**

* 检查primary数据库是否有不被逻辑standby支持的对象

1. SQL> **SELECT** \* **FROM** DBA\_LOGSTDBY\_UNSUPPORTED;

http://img.blog.csdn.net/20160506143518397?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

* 查看primary数据库中不含有主键或唯一键索引的表

1. SQL> **SELECT** OWNER, TABLE\_NAME **FROM** DBA\_LOGSTDBY\_NOT\_UNIQUE
3. 2> **WHERE** (OWNER, TABLE\_NAME) NOT IN
5. 3> (**SELECT** **DISTINCT** OWNER, TABLE\_NAME **FROM** DBA\_LOGSTDBY\_UNSUPPORTED)
7. 4> AND BAD\_COLUMN = 'Y';

   注意BAD\_COLUMN列值，该列有两个值：

   Y ：表示该表中有采用大数据类型的字段，比如LONG啦，CLOB啦之类。如果表中除log列某些行记录完全匹配，则该表无法成功应用于逻辑standby。standby会尝试维护这些表，不过你必须保证应用不允许。

   N ：表示该表拥有足够的信息，能够支持在逻辑standby的更新，不过仍然建议你为该表创建一个主键或者唯一索引/约束以提高log应用效率。

   假设在某张表中你可以确认数据是唯一的，但是基于效率方面的考虑，不想为其创建主键或唯一约束，怎么办呢？没关系，Oracle早想到了这一点，你可以创建一个DISABLE的Primary-Key Rely约束。

1. **Add** a disabled **primary**-**key** RELY **constraint**.（摘自君三思）

   维护逻辑standby与primary的数据库同步是通过sql应用实现，SQL应用转换的SQL语句在执行时，对于insert还好说，对于update,delete操作则必须能够唯一定位到数据库待更新的那条记录。如果primary库中表设置不当，可能就无法确认唯一条件。所以，Oracle 建议为表创建一个主键或非空的唯一索引/约束，以尽可能确保sql应用能够有效应用redo数据，更新逻辑standby数据库。

   如果能够确认表中的行是唯一的，那么可以为该表创建rely的主键，RELY约束并不会造成系统维护主键的开销，主你对一个表创建了rely约束，系统则会假定该表中的行是唯一，这样能够提供sql应用时的性能。但是需要注意，由于rely的主键约束只是假定唯一，如果实际并不唯一的话，有可能会造成错误的更新哟。

   创建rely的主键约束非常简单，只要在标准的创建语句后加上RELY DISABLE即可，例如：

1. SQL> **ALTER** **TABLE** mytab **ADD** **PRIMARY** **KEY** (id, **name**) RELY DISABLE;

**注 意:**创建了Rely约束后，Oracle会假定该列是唯一的（给DBA足够的信任），不过并不会对该列的值进行唯一性的验证，因此该列是否唯一只能由DBA来主动维护。

#### 1.1.1 Create a Physical Standby Database

参详物理standby创建

#### 1.1.2 Stop Redo Apply on the Physical Standby Database

   Physical备库切换到logical备库，需要在primary库构建LogMiner字典及开启supplemental日志，在这之前应先停用physical备库的MRP进程，以避免提前应用含LogMiner字典的REDO数据，造成转换为逻辑Standby后，SQL应用时没有LogMiner字典数据的REDO数据而影响到逻辑Standby与Primary的正常同步。

1. SQL> **ALTER** **DATABASE** RECOVER MANAGED STANDBY **DATABASE** CANCEL;

备库恢复传输redo：

1. SQL>  recover managed standby **database** disconnect **from** session;

http://img.blog.csdn.net/20160506144624633?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

#### 1.1.3 Prepare the Primary Database to Support a Logical Standby Database

* 一是将主库LOG\_ARCHIVE\_DEST\_1参数中的VALID\_FOR属性改为仅仅联机重做日志有效，而不包括备用重做日志。当primary为主库时，用于存放primary产生的arch，当primary被切换为备库角色后，用于存放自身作为备库产生的归档。

1. LOG\_ARCHIVE\_DEST\_1='LOCATION=USE\_DB\_RECOVERY\_FILE\_DEST VALID\_FOR=(ONLINE\_LOGFILES,ALL\_ROLES) DB\_UNIQUE\_NAME=primary'

* 二是专门为备用重做日志添加一个新的归档路径LOG\_ARCHIVE\_DEST\_3，也就是说联机日志与备用日志分开，仅当primary库转为备库时有效，当primary库为备库角色时，用于存放从primary库接收到的STANDBY\_LOGFILES。

1. LOG\_ARCHIVE\_DEST\_3='LOCATION=USE\_DB\_RECOVERY\_FILE\_DEST VALID\_FOR=(STANDBY\_LOGFILES,STANDBY\_ROLE) DB\_UNIQUE\_NAME=primary'
3. LOG\_ARCHIVE\_DEST\_STATE\_3=ENABLE

* 生成LogMiner字典

   生成LogMiner字典，这样redo日志中的变化才能被LogMiner字典的SQL Apply组件正确解析。

1. SQL> **EXECUTE** DBMS\_LOGSTDBY.BUILD;

http://img.blog.csdn.net/20160506145101469?watermark/2/text/aHR0cDovL2Jsb2cuY3Nkbi5uZXQv/font/5a6L5L2T/fontsize/400/fill/I0JBQkFCMA==/dissolve/70/gravity/Center

   另外主键和唯一约束/索引列的supplemental日志是自动生成的，The supplemental logging information ensures each update contains enough information to logically identify each row that is modified by the statement.对于11.2以上的版本，supplemental日志会随primary库传至physical备库，而这之前的版本需要检查primary库与physical 备库是否生成supplemental日志，如果没有则主库通过如下语句生成，然后再执行switchover 或 failover传至physical 备库。

1. SQL> **ALTER** **DATABASE** **ADD** SUPPLEMENTAL LOG DATA (**PRIMARY** **KEY**, **UNIQUE** **INDEX**) COLUMNS;

#### 1.1.4 Transition to a Logical Standby Database

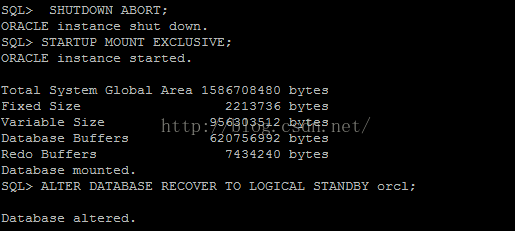
* physical备库转logical备库

若physical备库为RAC模式，则需先进行：

1. SQL> **ALTER** SYSTEM **SET** CLUSTER\_DATABASE=**FALSE** SCOPE=SPFILE;
3. SQL> SHUTDOWN ABORT;
5. SQL> STARTUP MOUNT EXCLUSIVE;

转physical备库为logical备库：

1. SQL> **ALTER** **DATABASE** RECOVER **TO** LOGICAL STANDBY db\_name;



* 备库参数配置

1. SQL> SHUTDOWN;
3. SQL> STARTUP MOUNT;
5. LOG\_ARCHIVE\_DEST\_1='LOCATION=USE\_DB\_RECOVERY\_FILE\_DEST VALID\_FOR=(ALL\_LOGFILES,ALL\_ROLES) DB\_UNIQUE\_NAME=standby'

   与physical备库所不同的是，logical备库被open后会产生自己的重做日志(redo)，因此我们需要配置参数LOG\_ARCHIVE\_DEST\_n。对于逻辑备库，此时存在三种日志文件，即online redo log，archived redo log，standby redolog。

1. LOG\_ARCHIVE\_DEST\_3='LOCATION=USE\_DB\_RECOVERY\_FILE\_DEST VALID\_FOR=(STANDBY\_LOGFILES,STANDBY\_ROLE) DB\_UNIQUE\_NAME=standby';
3. LOG\_ARCHIVE\_DEST\_STATE\_3=ENABLE

#### 1.1.5 Open the Logical Standby Database

* 由于逻辑standby与primary数据库事务并不一致，因此第一次打开时必须指定resetlogs子句

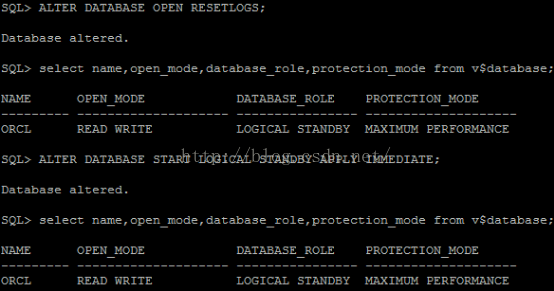
1. SQL> **ALTER** **DATABASE** **OPEN** RESETLOGS;

* 应用redo数据

1. SQL> **ALTER** **DATABASE** START LOGICAL STANDBY APPLY IMMEDIATE;

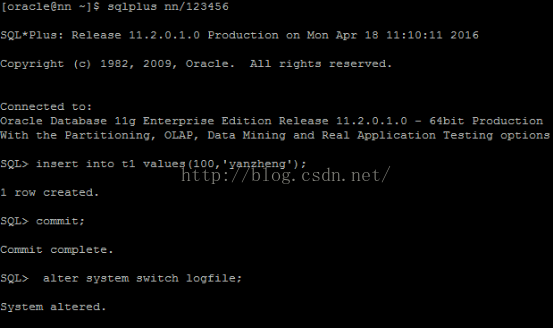
* 查询数据库状态

1. SQL> **select** **name**,open\_mode,database\_role,protection\_mode **from** v$**database**;

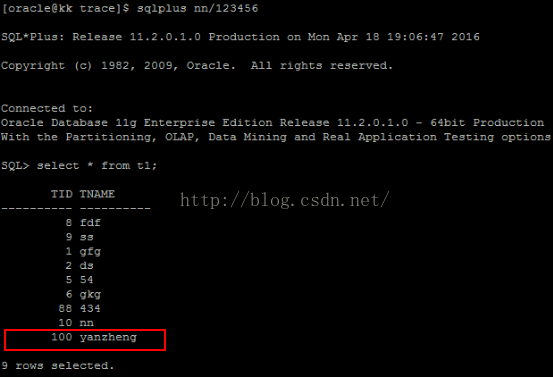


### 1.2 Verify the Logical Standby Database Is Performing Properly Standby

* Primary库NN用户T1表插入数据



* Logical standby库查询



至此，ORACLE 11G 之DATAGUARD搭建逻辑standby成功！

## RMAN(备份)常用命令

### 1)、RMAN同时连接主库与备库

在主库机器上执行（主库为启动startup，备库为nomount状态）

rman target sys/admin1234@master auxiliary sys/admin1234@slave nocatalog

RMAN>

### 2)、RMAN通过网络在线duplicate复制主库数据到备库

RMAN> duplicate target database for standby from active database nofilenamecheck;

RMAN> exit

### 3)、RMAN常用命令详解

Todo <http://blog.csdn.net/tianlesoftware/article/details/4976998>

### 4)、备份策略

## Dataguard broker配置与应用

### 1)、查询主备库switchover状态

SQL> select database\_role,switchover\_status from v$database;

### 2)、查询主备库dg\_broker\_start参数

SQL> show parameter dg\_broker\_start;

SQL> alter system set dg\_broker\_start = true;

### 3)、listener文件中加入静态监听

主库：

(SID\_DESC =

(GLOBAL\_DBNAME = master\_DGMGRL)

(ORACLE\_HOME = /opt/oracle/product/11.2.0/db\_1)

(SID\_NAME = orcl)

)

备库：

(SID\_DESC =

(GLOBAL\_DBNAME = slave\_DGMGRL)

(ORACLE\_HOME = /opt/oracle/product/11.2.0/db\_1)

(SID\_NAME = orcl)

)

重启监听

$ lsnrctl reload

查看监听状态

$ lsnrctl status

### 4)、broker主备切换配置（在主库上操作）

说明：dgmgrl可以安装到非主备库服务器上，因为但心主备库当掉，dgmgrl也当掉，无法对主备进行监控。

可以在第三方主机上安装oracle客户端，配置连接主备库的tnsname.ora服务器，即可远程启动dgmgrl进程。

# 配置broker

$ dgmgrl sys/admin1234

# 创建配置文件：

DGMGRL> create configuration DGORCLDB as primary database is master connect identifier is master;

create configuration '随意起名' as primary database is 'db\_unique\_name' connect identifier is 'tnsname.ora里连接主库';

DGMGRL> add database slave as connect identifier is slave maintained as physical;

add database 'db\_unique\_name' as connect identifier is 'tnsname.ora连接备库' maintained as physical;

# 启用配置

DGMGRL> enable configuration;

# 查看配置信息

DGMGRL> show configuration;

DGMGRL> show database master;

DGMGRL> show database slave;

### 5)、采用dataguard broker 测试switchover

# 主库情况

SQL> select open\_mode,database\_role,db\_unique\_name from v$database;

OPEN\_MODE DATABASE\_ROLE DB\_UNIQUE\_NAME

-------------------- ---------------- ------------------------------

READ WRITE PRIMARY MASTER

# 备库情况

SQL> select open\_mode,database\_role,db\_unique\_name from v$database;

OPEN\_MODE DATABASE\_ROLE DB\_UNIQUE\_NAME

-------------------- ---------------- ------------------------------

MOUNTED PHYSICAL STANDBY SLAVE

# 主备库切换测试

$ dgmgrl sys/admin1234

DGMGRL> show configuration;

# 主备库切

DGMGRL> switchover to slave;##主库db\_unique\_name:master，备库db\_unique\_name:slave

Performing switchover NOW, please wait...

Operation requires a connection to instance "orcl" on database "slave"

Connecting to instance "orcl"...

Connected.

New primary database "slave" is opening...

Operation requires startup of instance "orcl" on database "master"

Starting instance "orcl"...

ORACLE instance started.

Database mounted.

Switchover succeeded, new primary is "slave"

DGMGRL> show configuration

#主备测试

# 主库上，主备已经切换了备库上了。

SQL> conn / as sysdba;

SQL> select open\_mode,database\_role,db\_unique\_name from v$database;

# 备库上，备库上已经切换成了主库了。

SQL> conn / as sysdba;

SQL> select open\_mode,database\_role,db\_unique\_name from v$database;

SQL> conn abc/abc

SQL> select \* from abc;

# 要切换后的主库上新插入一条记录。

SQL> insert into abc values ( 5 , 'new' );

SQL> commit;

SQL> select \* from abc;

备库再切到主库（切回）

# 备库再切到主库

DGMGRL> connect sys@slave

Password:

Connected.

DGMGRL> show configuration;

DGMGRL> switchover to master;

Performing switchover NOW, please wait...

Operation requires a connection to instance "orcl" on database "master"

Connecting to instance "orcl"...

Connected.

New primary database "master" is opening...

Operation requires startup of instance "orcl" on database "slave"

Starting instance "orcl"...

# 主库上SQL查看，主备已经切换了

SQL> conn / as sysdba;

SQL> select open\_mode,database\_role,db\_unique\_name from v$database;

OPEN\_MODE DATABASE\_ROLE DB\_UNIQUE\_NAME

-------------------- ---------------- ------------------------------

READ WRITE PRIMARY master

SQL>

SQL> conn abc/abc

SQL> select \* from abc;

在主库上插入的记录也有了，实现了数据无损切换。

备库上SQL查看，备库上已经切回来了。

SQL> conn / as sysdba;

SQL> select open\_mode,database\_role,db\_unique\_name from v$database;

OPEN\_MODE DATABASE\_ROLE DB\_UNIQUE\_NAME

-------------------- ---------------- ------------------------------

MOUNTED PHYSICAL STANDBY SLAVE

SQL>

### 6)、broker fast failover测试

Oracle 11g Dataguard fast failover配置，需要主备数据库开启闪回功能，闪回功能开启本文略过。闪回开启需要启动到mount状态时，主备库的监听不要随意关闭。

1、dgmgrl查看主备库状态

$ dgmgrl sys/admin1234

DGMGRL> show configuration;

#查看故障转移情况， DISABLED没有启用。

DGMGRL> show fast\_start failover

快速启动故障转移: DISABLED

Fast-Start Failover: DISABLED

Threshold: 30 seconds

Target: (none)

Observer: (none)

Lag Limit: 30 seconds

Shutdown Primary: TRUE

Auto-reinstate: TRUE

Observer Reconnect: (none)

Observer Override: FALSE

Configurable Failover Conditions

Health Conditions:

Corrupted Controlfile YES

Corrupted Dictionary YES

Inaccessible Logfile NO

Stuck Archiver NO

Datafile Offline YES

Oracle Error Conditions:

(none)

2、启动快速启动故障转移

DGMGRL> enable fast\_start failover;

3、启动快速启动故障转移observer观察程序

DGMGRL> start observer;

说明：start observer后，观察程序不会在后台运行，就在前台显示。

在实际使用过程中，需要单独在服务器上启动，在后台自动运行，不能关闭，否则主备库就无法自动监控运行状态。就无法使用快速启动故障转移功能。

4、在另一个窗口执行查看

$ dgmgrl sys/admin1234

DGMGRL> show fast\_start failover;

快速启动故障转移: ENABLED

Fast-Start Failover: ENABLED

Threshold: 30 seconds

Target: slave

Observer: master

Lag Limit: 30 seconds

Shutdown Primary: TRUE

Auto-reinstate: TRUE

Observer Reconnect: (none)

Observer Override: FALSE

Configurable Failover Conditions

Health Conditions:

Corrupted Controlfile YES

Corrupted Dictionary YES

Inaccessible Logfile NO

Stuck Archiver NO

Datafile Offline YES

Oracle Error Conditions:

(none)

DGMGRL> show configuration

Configuration - dgorcldb

Protection Mode: MaxPerformance

Databases:

master - Primary database

slave - (\*) Physical standby database

Fast-Start Failover: ENABLED

Configuration Status:

SUCCESS

至此dgmgrl 配置的fast\_start failover已经配置好，下面模拟故障切换过程。

可以通过shutdown abort模拟数据库意外垮掉的情况，

注1：主库shutdown immediate是不会启动fast start failover功能的。

注2：oracle 11g dataguard fast start failover切换不需要主备库运行在最大可用模式。

5、模拟测试主库意外垮掉

1)主库上：

$ sqlplus / as sysdba;

SQL> shutdown abort;

观察器显示日志：显示执行主备切换过程

22:36:56.61 Friday, August 19, 2016

Initiating Fast-Start Failover to database "slave"...

Performing failover NOW, please wait...

Failover succeeded, new primary is "slave"

22:37:11.12 Friday, August 19, 2016

2) 备库上

登录备库，查看数据库状态已经切换回主库角色

SQL> select open\_mode,database\_role,db\_unique\_name from v$database;

OPEN\_MODE DATABASE\_ROLE DB\_UNIQUE\_NAME

-------------------- ---------------- ------------------------------

READ WRITE PRIMARY SLAVE

6、恢复原主库

DGMGRL> REINSTATE DATABASE master;

7、原主库再次启动，将自动重建新备库(原主库)

说明：原主库再次启动时，角色不会自动切换回，除非手动切换一次switchover到master。

SQL>startup mount; /\* 启动 master的 instance到 mount\*/

/\* observer开始自动重建新备库 开始自动重建新备库 \*/

DGMGRL> show configuration

Configuration - dgorcldb

Protection Mode: MaxPerformance

Databases:

slave - Primary database

master - (\*) Physical standby database

Fast-Start Failover: ENABLED

Configuration Status:

SUCCESS

观察器显示：

22:40:53.73 Friday, August 19, 2016

Initiating reinstatement for database "master"...

Reinstating database "master", please wait...

Operation requires shutdown of instance "orcl" on database "master"

Shutting down instance "orcl"...

ORA-01109: database not open

Database dismounted.

ORACLE instance shut down.

Operation requires startup of instance "orcl" on database "master"

Starting instance "orcl"...

ORACLE instance started.

Database mounted.

Continuing to reinstate database "master" ...

Reinstatement of database "master" succeeded

22:41:47.18 Friday, August 19, 2016

### 7)、总结

## 使用Oem管理数据库

包括Grid Control和Database Control

Oracle 站点卫士 (Oracle Site Guard)

## LogMiner简介