# Oracle 数据库备份与恢复总结

# Oracle 数据库备份与恢复总结

关于 Oracle 数据库的备份与恢复，网上有不少文章。经过了不少项目，以及我在给公司做培训时也有一些总结，现在总结在一起贴出来。以下方法，有一些可能不能完全归纳为备份与恢复，但是作为开发 DBA，有时也是很有用的。对于生产库，一般还是推荐使用 ARCHIVELOG 模式下的 OS 备份或 RMAN 方法，要求比较高的可能还必须用到 RAC 等并行处理的架构，这可是一个很大的主题了，在此不作讨论。

这里列出来，只是自己的一个备忘录以备需要时查看，有好多不全面或者不当的地方，欢迎各位补充、批评指正！同时，本文借鉴了网上的一些相关文章，希望大侠们不要见怪，此处一并谢过。

- exp/imp (导出与导入 装库与卸库)
- SQL*Loader
- User Managed Backup and Recovery (用户管理的备份与恢复)
- RMAN
- Flashback
- LogMiner
- 备份与恢复的规划

# 1. exp/imp (导出与导入 装库与卸库)

## 1.1 基本命令

### 1. 获取帮助

```
$ exp help=y
$ imp help=y
```

### 2. 三种工作方式

**（1）交互式方式**

```
$ exp      // 然后按提示输入所需要的参数
```

**（2）命令行方式**

```
$ exp user/pwd@dbname file=/oracle/test.dmp full=y     // 命令行中输入所需的参数
```

**（3）参数文件方式**

$ exp parfile=username.par  // 在参数文件中输入所需的参数

参数文件 username.par 内容

userid=username/userpassword
buffer=8192000
compress=n
grants=y
file=/oracle/test.dmp
full=y

### 3. 三种模式

**（1）表方式，将指定表的数据导出/导入。**

**导出：**

导出一张或几张表：

$ exp user/pwd file=/dir/xxx.dmp log=xxx.log tables=table1,table2

导出某张表的部分数据

$ exp user/pwd file=/dir/xxx.dmp log=xxx.log tables=table1 query=\"where col1=\'…\' and col2 \<…\"

**导入：**

导入一张或几张表

$ imp user/pwd file=/dir/xxx.dmp log=xxx.log tables=table1,table2 fromuser=dbuser touser=dbuser2 commit=y ignore=y

**（2）用户方式，将指定用户的所有对象及数据导出/导入。**

**导出：**

$ exp user/pwd file=/dir/xxx.dmp log=xxx.log owner=(xx,yy)

只导出数据对象，不导出数据 (rows=n )

$ exp user/pwd file=/dir/xxx.dmp log=xxx.log owner=user rows=n

**导入：**

$ imp user/pwd file=/dir/xxx.dmp log=xxx.log fromuser=dbuser touser=dbuser2 commit=y ignore=y

**（3）全库方式，将数据库中的所有对象导出/导入**

**导出：**

$ exp user/pwd file=/dir/xxx.dmp log=xxx.log full=y commit=y ignore=y

**导入：**

$ imp user/pwd file=/dir/xxx.dmp log=xxx.log fromuser=dbuser touser=dbuser2

## 1.2 高级选项

### 1. 分割成多个文件

以多个固定大小文件方式导出：这种做法通常用在表数据量较大，单个 dump 文件可能会超出文件系统的限制的情况

$ exp user/pwd file=1.dmp,2.dmp,3.dmp,… filesize=1000m  log=xxx.log full=y

以多个固定大小文件方式导入

$ imp user/pwd file=1.dmp,2.dmp,3.dmp,… filesize=1000m tables=xxx fromuser=dbuser touser=dbuser2  commit=y ignore=y

## 2. 增量导出/导入

// oracle 9i 以后 exp 不再支持 inctype

必须为 SYS 或 SYSTEM 才可执行增量导出导入

**增量导出:** 包括三个类型:

（1）“完全”增量导出（Complete） // 备份整个数据库

$ exp user/pwd file=/dir/xxx.dmp log=xxx.log inctype=complete

（2）“增量型”增量导出  导出上一次备份后改变的数据。

$ exp user/pwd file=/dir/xxx.dmp log=xxx.log inctype=incremental

（3） “累计型”增量导出（Cumulative）  只导出自上次“完全”导出之后数据库中变化了的信息。

$ exp user/pwd file=/dir/xxx.dmp log=xxx.log inctype=cumulative

**增量导入:**

$ imp usr/pwd FULL=y inctype=system/restore/inctype

其中:

SYSTEM:  导入系统对象

RESTORE: 导入所有用户对象

## 3. 以SYSDBA进行导出/导入

1. 用于 Oracle 技术支持
2. 用于表空间传输

例：

$ imp \'usr/pwd@instance as sysdba\' tablespaces=xx transport_tablespace=y file=xxx.dmp datafiles=xxx.dbf

$ imp file=expdat.dmp userid=”””sys/password as sysdba””” transport_tablespace=y “datafile=(c:tempapp_data,c:tempapp_index)”

## 4. 表空间传输 (速度快)

表空间传输是 8i 新增加的一种快速在数据库间移动数据的一种办法，是把一个数据库上的格式数据文件附加到另外一个数据库中，而不是把数据导出成 dmp 文件，这在有些时候是非常管用的，因为传输表空间移动数据就象复制文件一样快。

**1.关于传输表空间有一些规则 (10g 前):**

Ø 源数据库和目标数据库必须运行在相同的硬件平台上。

- Ø 源数据库与目标数据库必须使用相同的字符集。
- Ø 源数据库与目标数据库一定要有相同大小的数据块
- Ø 目标数据库不能有与迁移表空间同名的表空间
- Ø SYS 的对象不能迁移
- Ø 必须传输自包含的对象集
- Ø 有一些对象，如物化视图，基于函数的索引等不能被传输

(同字节序文件的跨平台可以用更换数据文件的文件头的方法)

(10g 支持跨平台的表空间传输，只要操作系统字节顺序相同，就可以进行表空间传输。需要使用 RMAN 转换文件格式，略)

**2. 检测一个表空间是否符合传输标准的方法：**

SQL > exec sys.dbms_tts.transport_set_check('tablespace_name',true);

SQL > select * from sys.transport_set_violations;

如果没有行选择，表示该表空间只包含表数据，并且是自包含的。对于有些非自包含的表空间，如数据表空间和索引表空间，可以一起传输。

**3. 简要使用步骤：**

如果想参考详细使用方法，也可以参考 ORACLE 联机帮助。

1.设置表空间为只读（假定表空间名字为 APP_Data 和 APP_Index）

SQL > alter tablespace app_data read only;

SQL > alter tablespace app_index read only;

2.发出 EXP 命令

SQL> host exp userid=""""sys/password as sysdba"""" transport_tablespace=y

tablespaces=(app_data, app_index)

以上需要注意的是

·为了在 SQL 中执行 EXP，USERID 必须用三个引号，在 UNIX 中也必须注意避免"/"的使用

·在 816 和以后，必须使用 sysdba 才能操作

·这个命令在 SQL 中必须放置在一行（这里是因为显示问题放在了两行）

3.拷贝.dbf 数据文件（以及.dmp 文件）到另一个地点，即目标数据库

可以是 cp(unix)或 copy(windows)或通过 ftp 传输文件（一定要在 bin 方式）

4.把本地的表空间设置为读写

$ alter tablespace app_data read write;

$ alter tablespace app_index read write;

5.在目标数据库附加该数据文件 (直接指定数据文件名)

(表空间不能存在，必须建立相应用户名或者用 fromuser/touser)

$ imp file=expdat.dmp userid=""""sys/password as sysdba""""

transport_tablespace=y datafiles=("c:\app_data.dbf,c:\app_index.dbf")

tablespaces=app_data,app_index tts_owners=hr,oe

6.设置目标数据库表空间为读写

$ alter tablespace app_data read write;

$ alter tablespace app_index read write;

# 1.3 优化

## 1. 加快exp速度

加大 large_pool_size，可以提高 exp 的速度
采用直接路径的方式(direct=y)，数据不需要经过内存进行整合和检查.
设置较大的 buffer，如果导出大对象，小 buffer 会失败。
export 文件不在 ORACLE 使用的驱动器上
不要 export 到 NFS 文件系统
UNIX 环境：用管道模式直接导入导出来提高 imp/exp 的性能

## 2. 加快imp速度

建立一个 indexfile，在数据 import 完成后在建立索引
将 import 文件放在不同的驱动器上
增加 DB_BLOCK_BUFFERS
增加 LOG_BUFFER
用非归档方式运行 ORACLE：ALTER DATABASE NOARCHIVELOG;
建立大的表空间和回滚段，OFFLINE 其他回滚段，回滚段的大小为最大表的 1/2
使用 COMMIT=N
使用 ANALYZE=N
单用户模式导入
UNIX 环境：用管道模式直接导入导出来提高 imp/exp 的性能

## 3. 通过unix/Linux PIPE管道加快exp/imp速度

**通过管道导出数据:**

1.通过 mknod -p 建立管道

$ mknod /home/exppipe p  // 在目录/home 下建立一个管道 exppipe 注意参数 p

2.通过 exp 和 gzip 导出数据到建立的管道并压缩

$ exp test/test file=/home/exppipe & gzip < /home/exppipe > exp.dmp.gz

$ exp test/test tables=bitmap file=/home/newsys/test.pipe &
gzip < /home/newsys/test.pipe > bitmap.dmp.gz

3.导出成功完成之后删除建立的管道

$ rm -rf /home/exppipe

**导出脚本:**

```
###UNIX 下 ORACLE 数据库通过 PIPE 管道进行备份
###### using "export" and "tar" command to bakup oracle datebase #######

trap "" 1 #nohup
LOGFILE=/opt/bakup/log/bakup_ora.log
```

```
export LOGFILE
DUMPDIR=/archlog_node1
export DUMPDIR
exec >$LOGFILE 2>&1

echo
echo ' Begin at ' `date`
echo

#           clear old result file
cd $DUMPDIR
if [ -f exp.dmp.Z ]
then
            echo "clear old result file"
            rm exp.dmp.Z
fi

#           make pipe
mkfifo exp.pipe
chmod a+rw exp.pipe

#           gain the dmp.Z file
compress < exp.pipe > exp.dmp.Z &
su -u oracle -c "exp userid=ll/ll file=$DUMPDIR/exp.pipe full=y buffer=20000000"

echo
echo '    exp end at '`date`
echo

#           rm pipe
rm exp.pipe

#           tar the dmp.Z file to tape
mt -f /dev/rmt/0 rew
tar cvf /dev/rmt/0 exp.dmp.Z

echo
echo '    tar end at '`date`
echo
```

**通过管道导入生成的文件:**

1.通过 mknod -p 建立管道

$ mknod /home/exppipe p

2.导入生成的压缩文件

$ imp test/test file=/home/exppipe fromuser=test touser=macro &

gunzip < exp.dmp.gz > /home/exppipe

3.删除管道

$ rm –fr /home/exppipe

### 4. 全库导入的一般步骤

**注意：**在导出时，需要通过toad或其他工具提取源数据库创建主键和索引的脚本

1. 先全库加 rows=n 把结构导进去

$ imp system/manager file=exp.dmp log=imp.log full=y rows=n indexes=n

2. 使业务用户的触发器失效/删除主键和唯一索引

```
spool drop_pk_u.sql
select 'alter table '||table_name||' drop constraint '||constraint_name||';'
from user_constraints
where constraint_type in ('P','U');
/
spool off
spool disable_trigger.sql
select 'alter trigger '||trigger_name||' disable;'
from user_triggers;
/
spool off

@drop_pk_u.sql
@disable_trigger.sql
```

3. 以 ignore=y 全库导入

$ imp system/manager file=exp.dmp log=imp.log full=y ignore=y

4. 通过 toad 或其他工具提取源数据库创建主键和索引的脚本,在目标数据库中创建主键和索引。使触发器生效。

## 1.4 常见问题

### 1. 字符集问题

ORACLE 多国语言设置是为了支持世界范围的语言与字符集，一般对语言提示，货币形式，排序方式和 CHAR,VARCHAR2,CLOB,LONG 字段的数据的显示等有效。ORACLE 的多国语言设置最主要的两个特性就是国家语言设置与字符集设置，国家语言设置决定了界面或提示使用的语言种类,字符集决定了数据库保存与字符集有关数据（如文本）时候的编码规则。

ORACLE 字符集设定，分为数据库字符集和客户端字符集环境设置。在数据库端，

字符集在创建数据库的时候设定，并保存在数据库 props$表中。

在客户端的字符集环境比较简单，主要就是环境变量或注册表项 NLS_LANG，注意 NLS_LANG 的优先级别为：参数文件<注册表<环境变量<alter session。如果客户端字符集和服务器端字符集不一样，而且字符集的转换也不兼容，那么客户端的数据显示与导出/导入的与字符集有关的数据将都是乱码。

使用一点点技巧，就可以使导出/导入在不同的字符集的数据库上转换数据。这里需要一个 2 进制文件编辑工具即可，如 uedit32。用编辑方式打开导出的 dmp 文件，获取 2、3 字节的内容，如 00 01，先把它转换为 10 进制数，为 1，使用函数 NLS_CHARSET_NAME 即可获得该字符集：

```
SQL> select nls_charset_name(1) from dual;
NLS_CHARSET_NAME(1)
------------------
US7ASCII
```

可以知道该 dmp 文件的字符集为 US7ASCII，如果需要把该 dmp 文件的字符集换成 ZHS16GBK，则需要用 NLS_CHARSET_ID 获取该字符集的编号：

```
SQL> select nls_charset_id('zhs16gbk') from dual;
NLS_CHARSET_ID('ZHS16GBK')
--------------------------
852
```

把 852 换成 16 进制数，为 354，把 2、3 字节的 00 01 换成 03 54，即完成了把该 dmp 文件字符集从 us7ascii 到 zhs16gbk 的转化，这样，再把该 dmp 文件导入到 zhs16gbk 字符集的数据库就可以了。

## 2. 版本问题

Exp/Imp 很多时候，可以跨版本使用，如在版本 7 与版本 8 之间导出导入数据，但这样做必须选择正确的版本，规则为：

·总是使用 IMP 的版本匹配数据库的版本，如果要导入到 816，则使用 816 的导入工具。

·总是使用 EXP 的版本匹配两个数据库中低的那个版本，如在 815 与 816 之间互导，则使用 815 的 EXP 工具。

imp 和 exp 版本不能往上兼容： imp 可以导入低版本 exp 生成的文件，不能导入高版本 exp 生成的文件

# 2. SQL*Loader

## 2.1 基本知识

Oracle 的 SQL*LOADER 可以将外部格式化的文本数据加载到数据库表中。通常与 SPOOL 导出文本数据方法配合使用。

### 1．命令格式

SQLLDR keyword=value [,keyword=value,...]

例：

$ sqlldr user/pwd control=emp.ctl data=emp.dat bad=emp.bad log=emp.log

### 2．控制文件

SQL*LOADER 根据控制文件可以找到需要加载的数据。并且分析和解释这些数据。

控制文件由三个部分组成,具体参数参考帮助文档：

1. 全局选件，行，跳过的记录数等；
2. INFILE 子句指定的输入数据；
3. 数据特性说明。

comment: --注释

例：

```
load data
infile *
append   --除了 append 外，还有 insert、replace、truncate 等方式
into table emp
fields terminated by '|'
(
no          float external,
    name        char(20),
    age         integer external,
    duty        char(1),
    salary      float external,
    upd_ts      date(14) 'YYYYMMDDHH24MISS'
)
begindata
100000000003|Mulder|000020|1|000000005000|20020101000000
100000000004|Scully|000025|2|000000008000|20020101235959
```

控制文件中 infile 选项跟 sqlldr 命令行中 data 选项含义相同，如使用 infile *则表明数据在本控制文件以 begin data 开头的区域内。

一些选项：

FIELDS TERMINATED BY WHITESPACE

FIELDS TERMINATED BY x'09'

FILLER_1 FILLER, // 指定某一列将不会被装载

DEPTNO position(1:2), DNAME position(*:16), // 指定列的位置

SEQNO RECNUM //载入每行的行号

SKIP n  // 指定导入时可以跳过多少行数据

## 3．数据文件

按控制文件数据格式定义的数据行集，例：

100000000001|Tom|000020|1|000000005000|20020101000000

100000000002|Jerry|000025|2|000000008000|20020101235959

固定格式、可变格式、流记录格式：

**固定格式：**

当数据固定的格式（长度一样）时且是在文件中得到时，要用 INFILE "fix n"

load data

infile 'example.dat' "fix 11"

into table example

fields terminated by ',' optionally enclosed by '"'

(col1 char(5),

col2 char(7))

example.dat:

001, cd, 0002,fghi,

00003,lmn,

1, "pqrs",

0005,uvwx,

**可变格式:**

当数据是可变格式（长度不一样）时且是在文件中得到时，要用 INFILE "var n"。如：

load data

infile 'example.dat' "var 3"

into table example

fields terminated by ',' optionally enclosed by '"'

(col1 char(5),

col2 char(7))

example.dat:

009hello,cd,010world,im,

012my,name is,

流记录格式: // Stream-recored format:

load data infile 'xx.dat' "str '|\n'"

```
        into table xx field terminated by ',' optionally enclosed by ""
(col1 char(5), col2 char(7))

example.dat:
        hello, ccd,|
        world, bb,|
```

### 4. 坏文件

bad=emp.bad

坏文件包含那些被 SQL*Loader 拒绝的记录。被拒绝的记录可能是不符合要求的记录。

### 5. 日志文件及日志信息

log=emp.log

当 SQL*Loader 开始执行后，它就自动建立 日志文件。日志文件包含有加载的总结，加载中的错误信息等。

## 2.2 高级选项

### 1. Conventional Path Load 与 Direct Path Load

**Conventional-path Load:**

通过常规通道方式上载。

特点：commit, always gen redo logs, enforce all constraints, fire insert triggers, can load into cluster, other user can make change

rows：每次提交的记录数

bindsize：每次提交记录的缓冲区

readsize：与 bindsize 成对使用，其中较小者会自动调整到较大者

sqlldr 先计算单条记录长度，乘以 rows，如小于 bindsize，不会试图扩张 rows 以填充 bindsize；如超出，则以 bindsize 为准。

命令为：

```
$ sqlldr dbuser/oracle control=emp.ctl log=emp.log rows=10000 bindsize=8192000
```

**Direct-Path Load:**

通过直通方式上载，可以跳过数据库的相关逻辑，不进行 SQL 解析，而直接将数据导入到数据文件中。

特点: save, conditionly gen redo logs, enforce PK UK NN, not fire triggers, can not load into cluster, other user can not make change

命令为：

```
$ sqlldr dbuser/oracle control=emp.ctl log=emp.log direct=true
```

### 2. SPOOL导出文本数据方法

导入的数据文件可以用 SPOOL 导出文本数据方法生成。

SQL*PLUS 环境设置

SET NEWPAGE NONE HEADING OFF SPACE 0 PAGESIZE 0

SET TRIMOUT ON TRIMSPOOL ON LINESIZE 2500

注：LINESIZE 要稍微设置大些，免得数据被截断，它应和相应的 TRIMSPOOL 结合使用防止导出的文本有太多的尾部空格。

但是如果 LINESIZE 设置太大，会大大降低导出的速度，另外在 WINDOWS 下导出最好不要用 PLSQL 导出，速度比较慢，直接用 COMMEND 下的 SQLPLUS 命令最小化窗口执行。对于字段内包含很多回车换行符的应该给与过滤，形成比较规矩的文本文件。

通常情况下，我们使用 SPOOL 方法，将数据库中的表导出为文本文件，如下述：

```
set trimspool on
set linesize 120 pagesize 2000 newpage 1 heading off    term off
spool  路径+文件名
select col1||','||col2||','||col3||','||col4||'..' from tablename;
spool off
```

## 2.3 脚本

### 1. 将表中数据记录导出为字段值用分隔符'|'分开的.dat文件

```
#!/bin/ksh
#####################################################################
##    名称: unloadtable
##    功能:  本 shell 用于将表中数据记录导出
##              导出为字段值用分隔符'|'分开的.dat 文件
##    编者：
##    日期: 2006.03.18
#####################################################################

if [ $# -ne 3 ]
then
      echo "usage:unloadtable tablename username password."
      exit 0
fi

##准备工作
echo "set heading off      " >/tmp/$1.col
echo "set pagesize 0" >>/tmp/$1.col
echo "set linesize 800     " >>/tmp/$1.col
echo "set feedback off     " >>/tmp/$1.col
```

```
echo "set tab off              " >>/tmp/$1.col
echo "select column_name||','' from user_tab_columns where lower(table_name)='$1' order by column_id; " >> /tmp/$1.col

##产生 select 语句
echo "set heading off      " >/tmp/$1.sel
echo "set pagesize 0" >>/tmp/$1.sel
echo "set linesize 800     " >>/tmp/$1.sel
echo "set feedback off     " >>/tmp/$1.sel
echo "set tab off              " >>/tmp/$1.sel
echo "select " >>/tmp/$1.sel
echo `sqlplus -s $2/$3 < /tmp/$1.col` |sed "s/,/||'|'||/g" |sed "s/||$//g"|sed "s/date/\"date\"/g" >>/tmp/$1.sel

##生成 dat 文件
#echo "from $1;\n/" >>/tmp/$1.sel    由于 / 导致多执行一次 select
echo "from $1;\n" >>/tmp/$1.sel
sqlplus -s $2/$3 < /tmp/$1.sel >$1_tmp.dat
#awk '{if(FNR!=1) print $0}' $1_tmp.dat >$1.dat        FNR 选项使得第一条记录选不出
awk '{print $0}' $1_tmp.dat >$1.dat
rm -f $1_tmp.dat
```

**2. 将数据导入到相应表中**

```
#!/bin/ksh
#########################################################################
##    名称:loadtable
##    功能:本 shell 用于将已经准备好的.dat 数据文件导入相应的表中
##            .dat 文件各个字段值用分隔符'|'分开。
##    编者:
##    日期: 2006.03.18
#########################################################################
if [ $# -ne 3 ]
then
        echo "usage:loadtable tablename username    password."
        exit 0
fi

##准备工作
echo "set heading off      " >/tmp/$1.colsql
```

```
echo "set pagesize 0" >>/tmp/$1.colsql
echo "set linesize 800    " >>/tmp/$1.colsql
echo "set feedback off    " >>/tmp/$1.colsql
echo "set tab off             " >>/tmp/$1.colsql
echo "select column_name||',' from user_tab_columns where lower(table_name)='$1' order by column_id; " >> /tmp/$1.colsql

##产生 ctl 文件
echo "load data" >/tmp/$1.ctl
echo "infile *" >>/tmp/$1.ctl
echo "into table $1" >>/tmp/$1.ctl
echo "fields terminated by '|'" >>/tmp/$1.ctl
echo `sqlplus -s $2/$3 < /tmp/$1.colsql` |sed "s/,$/)/g" |sed "s/^/(/g" >>/tmp/$1.ctl

##开始导入数据
echo "truncate table $1;" >/tmp/$1.sql
sqlplus $2/$3 < /tmp/$1.sql
sqlldr $2/$3 data=$1.dat control=/tmp/$1.ctl log=/tmp/$1.log
```

# 3. OS 备份/用户管理的备份与恢复(User Managed Backup and Recovery)

用户管理的备份与恢复也称 OS 物理备份，是指通过数据库命令设置数据库为备份状态，然后用操作系统命令，拷贝需要备份或恢复的文件。这种备份与恢复需要用户的参与手工或自动完成。

对于使用 OS 拷贝备份的数据文件，可以使用 DBVERTIFY 进行检验。DBVERTIFY 是一个外部工具，主要用于校验数据文件或备份的数据文件的数据块是否正确。

例： dbv /u01/oradata/oracle/users01.dbf BLOCKSIZE=8192

参数说明：

| 关键字 | 说明 | （默认） |
|---|---|---|
| FILE | 要检验的文件 | （NONE） |
| START | 起始块 | （文件的第一个块） |
| END | 结束块 | （文件的最后一个块） |
| BLOCKSIZE | 逻辑块大小 | （2048） |
| LOGFILE | 输出日志 | （NONE） |
| FEEDBACK | 显示进程 | （0） |

Recover 还可以进行测试，检测恢复的错误，错误信息记载在 alert_SID.log 文件中，通过测试，我们可以知道该恢复操作是否能正常完成。

```
SQL> RECOVER TABLESPACE sales TEST;
SQL> RECOVER DATABASE UNTIL CANCEL TEST;
```

## 3.1 相关设置

### 3.1.1 设置ARCHIVELOG与NONARCHIVELOG模式

重做日志组是以循环方式使用的，重做日志组会被覆盖重做日志信息就会丢失。为了保存历史以来的重做日志，数据库可以运行在日志归档模式下（archivelog mode）。

在日志归档模式下，当日志组撤换到下一个组时后台进程 ARCn 将上一个日志文件复制到另一个地方（oracle 10g 使用快速恢复区会归档到该区）保存。数据库默认为非归档模式（noarchivelog mode）。

设置 ARCHIVELOG 模式步骤：

1. 关闭数据库，备份已有的数据，改变数据库的运行方式是对数据库的重要改动，所以要对数据库做备份，对可能出现的问题作出保护。

2. 修改初试化参数: 使用 PFILE，修改初始化参数文件 init[SID].ora

```
log_archive_start=true                  #启动自动归档
log_archive_format=ARC%T%S.arc    #归档文件格式
log_archive_dest=/arch12/arch          #归档路径
```

3. 启动 Instance 到 Mount 状态，即加载数据库但不打开数据库：

SQL > startup mount;

4. 发出修改命令

SQL > alter database archivelog;

SQL > alter database open;

设置 NONARCHIVELOG 模式步骤同上，只需修改相应参数值即可。

### 3.1.2 LOGGING 与 NOLOGGING

表空间、表、索引、分区可以设置为 NOLOGGING，用于快速装入数据（Direct Load）。在插入数据时只写入最小的重做日志和回滚数据。在归档数据库模式下，执行 Direct Load 操作后应立即进行备份，否则不能使用之前的备份进行恢复。另外，用户可以设置数据库的强制日志模式，使用所有操作都记入日志。

LOGGING 与 NOLOGGING 的区别：

| LOGGING | NOLOGGING |
|---|---|
| 所有的更改写入 REDO | 最小写入 REDO LOG |
| 从最近备份中完全恢复 | 不能从最近备份中完全恢复 |
| 不需要增加备份 | 需要增加备份 |

NOLOGGING 的操作：

CREATE TABLE … NOLOGGING AS SELECT 语句

INSERT /*+APPEND*/ INTO <表> NOLOGGING SELECT 语句

INSERT /*+ PARALLEL(<表>,<n>)达式*/ INTO <表> NOLOGGING SELECT 语句

SQL*LOADER 的 DIRECT 方法

例：

```
SQL>CREATE TABLE emp1 NOLOGGING AS SELECT * FROM emp;
SQL>SELECT name,unrecoverable_time FROM V$DATAFILE;

SQL>INSERT /*+ APPEND */ INTO emp1 NOLOGGING SELECT* * FROM emp;
SQL>SELECT name,unrecoverable_time FROM V$DATAFILE;

SQL>ALTER DATABASE NO FORCE LOGGING;
```

### 3.1.3 归档路径

在归档模式下进行自动归档时，或者在恢复时设置归档所在的位置，需要设置归档路径初始化参数：

LOG_ARCHIVE_DEST_n=”LOCATION=path MANDATORY|OPTIONAL REOPEN=n”

LOG_ARCHIVE_DEST_n=”SERVICE=standby MANDATORY|OPTIONAL REOPEN=n”

# 3.2 NONARCHIVELOG 模式

## 3.2.1 脱机冷备与恢复

冷备份发生在数据库已经正常关闭的情况下，当正常关闭时会提供给我们一个完整的数据库。冷备份是将关键性文件拷贝到另外位置的一种说法。对于备份 Oracle 信息而言，冷备份是最快和最安全的方法。

冷备份的优点：

1．是非常快速的备份方法（只需拷贝文件）

2．容易归档（简单拷贝即可）

3．容易恢复到某个时间点上（只需将文件再拷贝回去）

4．能与归档方法相结合，作数据库“最新状态”的恢复。

5．低度维护，高度安全。

冷备份的不足：

1．单独使用时，只能提供到“某一时间点上”的恢复。

2．在实施备份的全过程中，数据库必须要作备份而不能作其它工作。也就是说，在冷备份过程中，数据库必须是关闭状态。

3．若磁盘空间有限，只能拷贝到磁带等其它外部存储设备上，速度会很慢。

4．不能按表或按用户恢复。

如果可能的话（主要看效率），应将信息备份到磁盘上，然后启动数据库（使用户可以工作）并将所备份的信息拷贝到磁带上（拷贝的同时，数据库也可以工作）。冷备份中必须拷贝的文件包括：

1．所有数据文件

2．所有控制文件

3．所有联机 REDO LOG 文件

4．参数化参数 Init.ora 文件（可选）。

## 3.2.2 案例

**1．9i 脱机冷备/恢复的例子：**

（1） 关闭数据库

```
$ sqlplus /nolog
SQL> connect /as sysdba
SQL> shutdown normal;
```

（2） 用拷贝命令备份/恢复全部的时间文件、重做日志文件、控制文件、初始化参数文件

```
SQL > host cp    xx    xx;
```

可以使用以下冷备脚本：

```
#!/bin/bash
#####################################################################
##    名称:    coldback_gen.sh
```

```
##   功能:   本 shell 用于生成冷备份脚本, 进行冷备份
                                    同时生成相应的恢复命令
##              可以修改后在生成后立即执行
##   编者:
##   日期:   2006.12.13.
#########################################################################

##设置变量
##设置临时文件名
tempsql=./backup.sql
##设置备份文件存放路径
backdate=`date -u +%Y%m%d`
backupdir=/u04/oracle/coldback/$backdate

mkdir $backupdir
##设置备份脚本文件名
backupsh=$backupdir/coldback.sh
rcvrsh=$backupdir/recovery.sh

echo "正在生成冷备份脚本[$backupsh]..."

##检查 ORACLE 数据库是否启动
oraisrun=`ps -ef|grep -c ora_`
if [ "$oraisrun" = "0" ] || [ "$oraisrun" = "1" ]
then
echo "ORACLE 数据库尚未启动，请先启动 ORACLE"
echo ""
exit
fi

##准备工作
echo "set heading off      " > $tempsql
echo "set feedback off    " >>$tempsql
echo "set tab off             " >>$tempsql
echo "set verify off       " >>$tempsql
echo "set pagesize 0"         >>$tempsql
echo "set linesize 800    " >>$tempsql
echo "select '#!/bin/bash' from dual;"         >> $tempsql
echo "select "                    from dual;"         >> $tempsql
echo  "select  '##  备份脚本生成时间:  "  `date  +%Y 年%m 月%d 日-%H:%M:%S`  "'
from dual;" >> $tempsql
echo "select '##  备份目的路径: $backupdir' from dual; "         >> $tempsql
echo "select " from dual; " >> $tempsql
```

```
echo ""
echo "select 'echo "开始进行脱机冷备..."' from dual; "              >> $tempsql
echo "select 'echo "备份目的路径: $backupdir "' from dual; "      >> $tempsql


##这里不直接关闭数据库，提示用户手工关闭为好  如果需要直接关闭，请修改
echo "select 'orarun='||'\`'||'ps -ef|grep -c ora_'||'\`' from dual;" >>$tempsql
echo "select 'if [ "\$orarun" != "0" ] && [ "\$orarun" != "1" ]    ' from dual;" >>$tempsql
echo "select 'then' from dual; " >>$tempsql
echo "select 'echo " " ' from dual;" >>$tempsql
echo "select 'echo "ORACLE 数据库已启动，请先关闭 ORACLE 数据库"' from dual;
" >>$tempsql
echo "select 'echo " " ' from dual;" >>$tempsql
echo "select 'exit' from dual; "      >>$tempsql
echo "select 'fi'      from dual; "      >>$tempsql

echo "select 'echo " " ' from dual; "    >> $tempsql
echo "select 'echo "正在备份控制文件..."' from dual; "    >> $tempsql
echo "select 'cp ' ||name||' $backupdir' from v\$controlfile; "    >> $tempsql
echo "select 'echo "控制文件备份完毕!"' from dual; "    >> $tempsql
echo "select 'echo " " ' from dual; "    >> $tempsql
echo "select 'echo "正在备份数据文件..."' from dual; "    >> $tempsql
echo "select 'cp ' ||name||' $backupdir' from v\$datafile;        "    >> $tempsql
echo "select 'echo "数据文件备份完毕!"' from dual; "    >> $tempsql
echo "select 'echo "正在备份联机日志..."' from dual; "    >> $tempsql
echo "select 'echo " " ' from dual; "    >> $tempsql
echo "select 'cp '||member||' $backupdir' from v\$logfile;        "    >> $tempsql
echo "select 'echo "联机日志备份完毕!"' from dual;"    >> $tempsql
echo "select 'echo " " ' from dual; " >> $tempsql
echo "select 'echo "脱机冷备完毕!"' from dual;"    >> $tempsql
echo "select 'echo " " ' from dual; " >> $tempsql
echo "select 'echo " " ' from dual; " >> $tempsql


##生成冷备份执行脚本
sqlplus -s ' / as sysdba' < $tempsql > $backupsh
rm -f $tempsql
chmod +x $backupsh
cp $backupsh .


echo "正在生成冷备对应的恢复脚本[$rcvrsh]..."
```

```
##准备工作
echo "set heading off      " > $tempsql
echo "set feedback off     " >>$tempsql
echo "set tab off          " >>$tempsql
echo "set verify off       " >>$tempsql
echo "set pagesize 0"         >>$tempsql
echo "set linesize 800     " >>$tempsql
echo "select '#!/bin/bash' from dual;"         >> $tempsql
echo "select "                    from dual;"         >> $tempsql
echo  "select  '## 恢复脚本生成时间:  " `date  +%Y 年%m 月%d 日-%H:%M:%S`  "'
from dual;" >> $tempsql
echo "select '##  恢复文件所在路径: $backupdir' from dual; "         >> $tempsql
echo "select " from dual; " >> $tempsql
echo ""
echo "select 'echo "开始进行文件的复制恢复..."' from dual; "            >> $tempsql
echo "select 'echo "恢复文件所在的路径: $backupdir "' from dual; "     >> $tempsql


##这里不直接关闭数据库，提示用户手工关闭为好   如果需要直接关闭，请修改
echo "select 'orarun='||\`||'ps -ef|grep -c ora_'||\` from dual;" >>$tempsql
echo "select 'if [ "\$orarun" != "0" ] && [ "\$orarun" != "1" ]    ' from dual;" >>$tempsql
echo "select 'then' from dual; " >>$tempsql
echo "select 'echo " " ' from dual;" >>$tempsql
echo "select 'echo "ORACLE 数据库已启动，请先关闭 ORACLE 数据库"' from dual;
" >>$tempsql
echo "select 'echo " " ' from dual;" >>$tempsql
echo "select 'exit' from dual; "     >>$tempsql
echo "select 'fi'     from dual; "     >>$tempsql

echo "select 'echo " " ' from dual; "    >> $tempsql
echo "select 'echo "正在恢复控制文件..."' from dual; "    >> $tempsql
echo           "select          'cp          '||'$backupdir'||'/'||substr(name,instr(name,'/',-1)+1,
length(name)-instr(name,'/',-1) )||' '||name from v\$controlfile; "    >> $tempsql
echo "select 'echo "控制文件恢复完毕!"' from dual; "    >> $tempsql
echo "select 'echo " " ' from dual; "    >> $tempsql
echo "select 'echo "正在恢复数据文件..."' from dual; "    >> $tempsql
echo           "select          'cp          '||'$backupdir'||'/'||substr(name,instr(name,'/',-1)+1,
length(name)-instr(name,'/',-1) )||' '||name from v\$datafile;       "    >> $tempsql
echo "select 'echo "数据文件恢复完毕!"' from dual; "    >> $tempsql
echo "select 'echo "正在恢复联机日志..."' from dual; "    >> $tempsql
echo "select 'echo " " ' from dual; "    >> $tempsql
echo        "select        'cp         '||'$backupdir'||'/'||substr(member,instr(member,'/',-1)+1,
length(member)-instr(member,'/',-1) )||' '||member from v\$logfile;       "    >> $tempsql
```

```
echo "select 'echo "联机日志恢复完毕!"' from dual;"    >> $tempsql
echo "select 'echo " " ' from dual; " >> $tempsql
echo "select 'echo "脱机冷备恢复完毕!"' from dual;"    >> $tempsql
echo "select 'echo " " ' from dual; " >> $tempsql
echo "select 'echo " " ' from dual; " >> $tempsql


##生成冷备恢复的执行脚本
sqlplus -s ' / as sysdba' < $tempsql > $rcvrsh

rm -f $tempsql
chmod +x $rcvrsh
cp $rcvrsh .

echo "生成脱机冷备备份与恢复脚本完毕!"
echo "请检查脚本文件: [$backupsh]"
echo "                [$rcvrsh]"
echo ""

#如果需要生成后立即执行备份，可增加关闭数据库的操作，然后将以一几行的注
释#去掉即可
#./$backupsh
#echo "备份执行完毕，请检查！"
#echo ""
```

（3） 重启 Oracle 数据库

$ sqlplus /nolog

SQL> connect /as sysdba

SQL> startup

**2．如果自从上次脱机冷备后，数据文件错误，联机日志没有被覆盖，可模拟不完全恢复。**

1. SQL> shutdown;
2. $ cp .. ..;  // 只恢复出错的数据文件
3. SQL> startup mount;
4. SQL> recover database;
5. SQL> alter database open;

# 3.3 ARCHIVELOG 模式

## 3.3.1 脱机冷备与恢复

同 NONARCHIVELOG 模式

### 3.3.2 联机热备

联机热备是在数据库运行的情况下进行备份的方法。热备份要求数据库在 Archivelog 方式下操作，并需要大量的档案空间。

**热备份的优点：**

1．可在表空间或数据文件级备份，备份时间短。
2．备份时数据库仍可使用，支持 24*7 不间断运行。
3．可达到秒级恢复（恢复到某一时间点上）。
4．可对几乎所有数据库实体作恢复。
5．恢复是快速的，在大多数情况下在数据库仍工作时恢复。

**热备份的不足是：**

1．不能出错，否则后果严重。
2．若热备份不成功，所得结果不可用于时间点的恢复。
3．因难于维护，所以要特别仔细小心，不允许“以失败而告终”。

**注意：**在热备过程中系统会生成更多的重做日志和回滚数据。所以必须在数据库较空闲时才进行备份。

**备份内容：**

(1) 数据文件: 一个表空间一个表空间地备份

```
sql> alter tablespace users begin backup;
sql> $copy '/xx/xx.dbf '     '/yy/yy.dbf' ;
sql> alter tablespace users end backup;
sql> alter system checkpoint;
```

**(只读表空间直接拷贝，不用 begin backup)**

(2) 备份归档 log 文件

（1）临时停止归档进程  log_archive_max_processes=0
（2）log 下那些在 archive redo log 目标目录中的文件
（3）重新启动 archive 进程
（4）备份归档的 redo log  文件

(3) 备份联机的控制文件：

```
sql> alter database backup controlfile to '/xx/xx.ctl';
```

(4) 备份初始化文件  配置文件  等：

```
sql> $copy   .. .. ;
```

**热备脚本：hotback.sql**

```
Rem  热备份脚本    for Linux
Rem  执行该脚本必须保证数据库处于归档模式
Rem db.dw.dm@gmail.com 2007-03-17

Rem
Rem  设置 SQL*Plus 环境参数
```

```
Rem

set feedback off
set pagesize 0
set heading off
set verify off
set linesize 100
set trimspool on

Rem  设置备份相关的路径   For Linux
Rem  设置数据文件备份路径
define datafile_dir    = '/u05/oracle/hotback/datafile'
Rem  设置归档日志文件备份路径
define archlog_dir = '/u05/oracle/hotback/archlog'
Rem  设置控制文件备份路径
define controlfile_dir = '/u05/oracle/hotback/controlfile'

Rem  设置生成的备份脚本名
define hotback    = '/u05/oracle/hotback/open_hot_backup.sql'
define spoolfile = '/u05/oracle/hotback/spool.tmp'
define cpy = 'cp'


prompt *** Spooling to &hotback


Rem  产生备份数据文件、归档日志文件的命令

set serveroutput on size 1000000
spool &hotback
prompt spool &spoolfile

prompt archive log list;;
prompt alter system switch logfile;;
prompt alter system archive log all;;

DECLARE
  CURSOR cur_tablespace IS
    SELECT tablespace_name
       FROM dba_tablespaces
       ORDER BY tablespace_name;

  CURSOR cur_datafile (tn VARCHAR) IS
```

```
    SELECT file_name
      FROM dba_data_files
     WHERE tablespace_name = tn
     ORDER BY file_name;

  CURSOR cur_arch_dest IS
    SELECT value
      FROM v$parameter
           WHERE    name = 'log_archive_dest';
BEGIN
   FOR ct IN cur_tablespace LOOP
      IF ct.tablespace_name!='TEMP' then
            dbms_output.put_line ('alter tablespace '||ct.tablespace_name||' begin backup;');
      FOR cd IN cur_datafile (ct.tablespace_name) LOOP
            dbms_output.put_line ('host &cpy '||cd.file_name||' &datafile_dir');
      END LOOP;
      dbms_output.put_line ('alter tablespace '||ct.tablespace_name||' end backup;');
      end if;
   END LOOP;

   FOR dest IN cur_arch_dest LOOP
      dbms_output.put_line ('host &cpy '|| dest.value || '/* &archlog_dir ');
   END LOOP;

END;
/

Rem 产生备份控制文件的命令

prompt alter system archive log current;;
prompt alter database backup controlfile to trace;;
prompt alter database backup controlfile to '&controlfile_dir/control.bak' REUSE;;

prompt archive log list;;
prompt prompt ***Hot Backup Finish***;
prompt spool off
spool off;


host rm -f &spoolfile
Rem 执行生成的脚本文件
Rem @&hotback
Rem host del &hotback
```

### 3.3.3 联机热备的恢复

#### 3.3.3.1 完全恢复

一般步骤:

1. 通过以下信息，找到故障数据文件
   alert.log
   background trace file
   v$recover_file  v$recovery_log
   通过这两个视图可以了解详细的需要恢复的数据文件与需要使用到的归档日志。
2. 将故障数据文件对应的表空间 offline
   SQL> alter tablespace xxx offline;
3. restore and recover
   SQL> host cp .. ..;
   SQL> [alter database] recover database/tablespace/datafile 'xx';
4. 将表空间 online
   SQL> alter tablespace xxx online;

#### 3.3.3.2 不完全恢复

不完全恢复的方法只能恢复到过去某个时间点/SCN 的数据库状态。

**一些限制:**

1. 必要条件
   一个有效的 online/offline 备份(包含所有的数据文件)
   自从备份到故障前的所有归档日志
   有可能需要控件文件 (所有控件文件丢失，数据库结构已改变)
   SQL> recover database ... using backup controlfile;
2. 只能恢复到所有备份数据文件的最大 SCN 以后，
3. 恢复后需要 resetlog，所以需要在恢复后马上备份

**三种不完全恢复的方法:**

1. 基于变化的不完全恢复  Change-based Recovery
2. 基于用户干涉(取消)的不完全恢复  Cancel-based Recovery
3. 基于时间的不完全恢复  Time-based Recovery

**获得信息:**

alert.log

可以通过 LogMiner 获得精确的时间/SCN，一般在备机上恢复，再 exp/imp 到生产机。

查看需要恢复的文件，以及相关的提示信息

SQL> select * from v$recover_file;

SQL> select * from v$datafile;

查看二者的 change#, 确定对应的在 v$log_history 中的范围，从而确定需要那个日志文

件序列

**设置归档日志文件的路径：** LOG_ARCHIVE_DEST

**设置 log 在不同的路径：**

SQL> SET LOGSOURCE 'xx';

SQL> alter system archive log start to 'xx';

**恢复步骤：**

1. 关闭数据库，启动到 MOUNT 状态
   SQL> shutdown;
   SQL> startup mount;
2. 恢复数据文件、日志文件、归档日志文件
   SQL> host cp .. ..;
   SQL> archive log list;
   SQL> archived log ==>LOG_ARCHIVE_DEST
3. 执行恢复命令

   基于变化:
   SQL> recover database until change 9999;

   基于时间:
   SQL> revover database until time '2001-12-01 14:02:23' using backup controlfile;

   基于取消:
   SQL> recover database until cancel;
4. 重置日志，恢复后需要马上备份
   SQL> alter database open resetlogs;

# 3.4 分类案例

## 3.4.1 控件文件的备份与恢复

### 一、备份

1. 镜像控制文件

   手工备份  // 每当数据库结构发生变化时立即备份

   1.数据库关闭时，OS 命令拷贝。

   2. 联机备份
      SQL> alter database backup controlfile to 'ctl.bak';
      SQL> alter database backup controlfile to trace;

### 二、恢复

1. 损坏一个控制文件：从镜像拷贝或修改 initSID.ora 取消损坏的控制文件。
2. 损坏所有的控制文件：

   利用备份的控制文件恢复，拷贝或在命令中恢复：
   SQL> recover database ...using backup controlfile;

   手工重建控制文件：

NOMOUNT 状态下执行
SQL> CREATE CONYTROLFILE..; //注意联机日志和数据文件的路径和文件名
SQL> alter database open resetlogs;

### 3.4.2 联机日志文件的备份与恢复

**一、备份**

1. 镜像在不同的磁盘上。 //如果有镜像备份，不用恢复
2. 非归档模式下，在数据库关闭时用 OS 命令拷贝备份。
3. 归档模式下，手工或自动归档。

**二、恢复** // 在恢复后一定要重做备份

丢失日志组成员:

在有多个镜像时，一般不会报错，如果需要恢复，可以先删除再增加。

1. 删除： SQL> alter database drop logfile member 'xx';
2. 新增： SQL> alter database add logfile member 'xx' to group 2;

( 如果丢失当前日志组成员： 可以先 alter system switch logfile; 再进行操作。)
以下恢复方法都是指丢失所有日志组成员的情况下的恢复。

**一、丢失非当前联机日志**

1． 重启数据库到 Mount 状态：
2． 重建丢失的日志：用命令清空日志组的方法
//已归档，重建该日志
SQL> alter database clear logfile group 2;
//归档模式下如果没有归档
SQL> alter database clear unarchived logfile group 2;

**二、丢失当前联机日志**

1. 如果数据库正常关闭：
日志中没有未决事务需要实例恢复，同非当前联机日志方法。
2. 如果是非正常关闭数据库的情况，未决事务需要实例恢复：
如果有备份，可通过备份进行不完全恢复。// until cancel
没有备份，进行强制性恢复 // 最后的办法，可能导致数据库的不一致
3. 如果数据库当前为 Open 状态，有活动的事务：
尝试能否 Export 或热备份，如果可以，赶紧备份
检查非当前日志是否正常，可先做 clear，然后尝试是否能 switch log，能否正常关闭 DB，如果可以 switch log 正常关闭 DB，方法同 1，否则同 2

### 3.4.3 回滚数据文件的恢复

**1. 从可用备份中恢复**

非归档模式下的恢复会有数据丢失
归档模式下，有可用备份，可完全恢复(需要关闭数据库)

**2. 没有可用备份时强行恢复:**

// offline drop，删除重建。

// 需要先注释 undo_tablespace，或者重新指定一个系统回滚段表空间，然后再操作

1. 数据库正常关闭  没有未决的事务
    1. shutdown，修改 init 参数文件，注释 undo_tablespace
    2. SQL>startup restrict mount;
    3. SQL>alter database datafile 2 offline drop;
    4. SQL>alter database open;
    5. SQL>drop tablespace xxx including contents;
    6. 重建回滚段表空间
    7. shutdown，修改 init 参数文件，去掉注释设置新的 undo_tablespace
    8. SQL> alter system disable restricted session;
2. 非正常关闭  强制恢复  隐含参数：_CORRUPTED_ROLLBACK_SEGMENTS
    1. shutdown，修改 init 参数文件，删除 undo_tablespace
    2. SQL>startup restrict mount;
    3. SQL>alter database datafile 2 offline drop;
    4. SQL>alter database open;
    5. SQL>drop tablespace xxx including contents;
        如果出错：回滚段中有活动事务
            SQL>drop rollback segment rbs0; rbs1, 2 ...
        在第 1 步中，加入隐含参数
        _CORRUPTED_ROLLBACK_SEGMENTS
                = (_SYSSMU1$,_SYSSMU2$,_SYSSMU3$, ...)
    6. 重建回滚段表空间，online
    7. shutdown，修改 init 参数文件，去掉注释设置新的 undo_tablespace
        去掉隐含参数
    8. SQL>alter system disable restricted session;

### 3.4.5 临时数据文件的恢复

方法：  先将用户临时表空间置为其他，然后删除重建

1. SQL> shutdown
2. SQL> startup restrict mount;
3. SQL> alter user xxx temporary tablespace TEMP2;
4. SQL> alter database open;
5. SQL> drop tablespace temp including contents;
6. 重建临时表空间
7. 重新分配给各用户
8. SQL> alter system disable restricted session;

如果是默认的临时表空间，需要先将默认临时表空间置为其他

SQL> alter database default temporary tablespace temp_2;

# 4. RMAN(备份与恢复管理器)

RMAN 是 ORACLE 提供的一个备份与恢复的工具，可以用来备份和还原数据库文件、归档日志和控制文件。它也可以用来执行完全或不完全的数据库恢复。

RMAN 可以由命令行接口或者 OEM 的 Backup Manager GUI 来控制。

## 4.1 基本知识

### 4.1.1 RMAN的组件、概念

**1. RMAN 主要包括以下组件：**

**Target Database: (目标数据库)**

就是需要 RMAN 对其进行备份与恢复的数据库，RMAN 可以备份数据文件，控制文件，归档日志，spfile。(注意：RMAN 不能用于备份联机日志、初始化参数文件和口令文件)

**Server Session:  (服务器会话)**

RMAN 启动数据库上的 Oracle 服务器进程，将建立一个与目标数据库的会话。由目标数据库上的服务器进程进行备份、还原、恢复的实际操作。

**服务器进程**

RMAN 的服务进程是一个后台进程，用于与 RMAN 工具与数据库之间的通信，也用于 RMAN 工具与磁盘/磁带等 I/O 设置之间的通信，服务进程负责备份与恢复的所有工作，在如下情况将产生一个服务进程：

- Ø 当连接到目标数据库
- Ø 分配一个新的通道

**Channel:  (通道)**

一个通道是 RMAN 和目标数据库之间的一个连接，"allocate channel"命令在目标数据库启动一个服务器进程，同时必须定义服务器进程执行备份或者恢复操作使用的 I/O 类型。

通道控制命令可以用来：

- Ø 控制 RMAN 使用的 O/S 资源，影响并行度
- Ø 指定 I/O 带宽的限制值（设置 limit read rate 参数）
- Ø 定义备份片大小的限制（设置 limit kbytes）
- Ø 指定当前打开文件的限制值（设置 limit maxopenfiles）

**recovery catalog: (恢复目录)**

用来保存备份与恢复信息的一个数据库，不建议创建在目标数据库上。RMAN 利用恢复目录记载的信息去判断如何执行需要的备份恢复操作。

如果不采用恢复目录，备份信息可以存在于目标数据库的 control file 中。

如果存放在目标数据库的 control file 中，控件文件会不断增长，不能保存 RMAN 的 Script。

CONTROL_FILE_RECORD_KEEP_TIME (default=7)：控件文件中 RMAN 信息保存的最短时间。

使用恢复目录的优势： 可以存储脚本，记载较长时间的备份恢复操作。

**RMAN Repository: (RMAN 恢复目录数据库)**

存放 recovery catalog(恢复目录)的数据库。建议为恢复目录数据库创建一个单独的数据库。

**MML: (媒体管理库)**

Media Management Layer (MML)是第三方工具或软件，用于管理对磁带的读写与文件的跟踪管理。如果你想直接通过 RMAN 备份到磁带上，就必须配置媒体管理层，媒体管理层的工具如备份软件可以调用 RMAN 来进行备份与恢复。

**Command Line and Script (命令行接口与脚本)**

( 见 4.1.2 )

**2．概念述语**

**Backup Sets (备份集合)**

备份集合的特性：包括一个或多个数据文件或归档日志，以 oracle 专有的格式保存，有一个完全的所有的备份片集合构成，构成一个完全备份或增量备份。

**Backup Pieces (备份片)**

一个备份集由若干个备份片组成。每个备份片是一个单独的输出文件。一个备份片的大小是有限制的；如果没有大小的限制， 备份集就只由一个备份片构成。备份片的大小不能大于使用的文件系统所支持的文件长度的最大值。

**Image Copies 镜像备份**

镜像备份是独立文件（数据文件、归档日志、控制文件）的备份。它很类似操作系统级的文件备份。它不是备份集或备份片，也没有被压缩。

**Full backup Sets 全备份集合**

全备份是一个或多个数据文件中使用过的数据块的的备份。没有使用过的数据块是不被备份的，也就是说，oracle 进行备份集合的压缩。

**Incremental backup sets 增量备份集合**

增量备份是指备份一个或多个数据文件的自从上一次同一级别的或更低级别的备份以来被修改过的数据块。 与完全备份相同，增量备份也进行压缩。

**File multiplexing**

多个数据文件可以在一个备份集中。

**Recovery catalog resyncing 恢复目录同步**

使用恢复管理器执行 backup、copy、restore 或者 switch 命令时，恢复目录自动进行更新，但是有关日志与归档日志信息没有自动记入恢复目录。需要进行目录同步。使用 resync catalog 命令进行同步。

RMAN> resync catalog;

**Incarnation 对应物**

在不完全恢复完成之后，通常需要使用 resetlogs 选项来打开数据库。resetlogs 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。数据库的逻辑生存期也被称为一个对应物(incarnation)。每次使用 resetlogs 选项来打开数据库后都会创建一个新的数据库对应物。

### 4.1.2 RMAN的使用：命令行接口与脚本

数据库状态：

RMAN恢复目录数据库： 必须OPEN

目标数据库： 根据不同情况，必须MOUNT或OPEN

**1. 使用不带恢复目录的 RMAN**

设置目标数据库的 ORACLE_SID ，执行：

```
$ rman nocatalog
RMAN> connect target
RMAN> connect target user/pwd>@db
```

**2. 使用带恢复目录的 RMAN**

```
$ rman catalog rman/rman
RMAN> connect target  //连接本地数据库作为目标数据库
RMAN> connect target user/pwd>@db //连接远程数据库
或
$ rman catalog rman/rman target user/pwd>@db
```

**3. 命令行接口**

1、单个执行

```
RMAN> backup database;
```

2、运行一个命令块

```
RMAN> RUN {
2> copy datafile 10 to
3> '/oracle/prod/backup/prod_10.dbf';
4> }
```

3、运行存储在恢复目录中的脚本：

```
RMAN> RUN  { EXECUTE SCRIPT backup_whole_db };
```

4、运行外部脚本：

```
$ rman catalog rman/rman target / @backup_db.rman
```

```
$ rman cmdfile=backup.rman msglog=backup.log
RMAN> @backup_db.rman
RMAN> RUN  { @backup_db.rman }
```

如果在 cron 中执行，注意在脚本中设置正确的环境变量，例：

```
#set env
export ORACLE_HOME=/opt/oracle/product/9.2
export ORACLE_SID=test
export NLS_LANG="AMERICAN_AMERICA.zhs16gbk"
export PATH=$PATH:$ORACLE_HOME/bin
rman cmdfile=backup_db.rman
```

**4. 使用脚本**

**创建或者取代脚本：**

```
RMAN> create script alloc_disk {
    2> # Allocates one disk
    3> allocate channel dev1 type disk;
    4> setlimit channel dev1 kbytes 2097150 maxopenfiles 32 readrate 200;
    5> }
RMAN> replace script rel_disk {
    2> # releases disk
    3> release channel dev1;
    5> }
```

**删除脚本：**

```
RMAN> DELETE SCRIPT Level0Backup;
```

**查看脚本：**

```
RMAN> PRINT SCRIPT Level0Backup;
```

**运行脚本：**

```
RMAN> RUN { EXECUTE SCRIPT backup_whole_db };
```

**5. 运行 OS 命令**

RMAN 支持通过执行 host 命令暂时退出 RMAN 的命令提示符而进入到操作系统的命令环境。

**6. 执行 SQL 语句**

在 RMAN 的命令提示符后输入 SQL 命令，然后在一对单引号(双引号亦可)中输入要执行的 SQL 语句，例如：

```
RMAN> SQL 'ALTER SYSTEM CHECKPOINT';
```

对于 SELECT 语句，无法得到结果。可以先执行 host 再用 SQLPLUS。

# 4.2 RMAN 的配置

## 4.2.1 建立Recovery Catalog恢复目录

**(1) 在目录数据库中创建恢复目录所用表空间：**

SQL> create tablespace rman_ts datafile '/xxx/rman_ts.dbf' size 20M;

**(2) 在目录数据库中创建 RMAN 用户并授权：**

SQL> create user rman identified by rman default tablespace rman_ts temporary tablespace temp quota unlimited on rman_ts;

SQL> grant connect, resource, recovery_catalog_owner to rman;

**(3) 在目录数据库中创建恢复目录**

$ rman catalog rman/rman

RMAN> create catalog tablespace rman_ts;

**(4) 登记目标数据库：**

一个恢复目录可以注册多个目标数据库，注册目标数据库的命令为：

$ RMAN catalog rman/rman target user/pwd @rcdb;

RMAN> register database;

## 4.2.2 查看RMAN的默认设置SHOW命令

必须连接目标数据库

```
RMAN> show all
RMAN> show channel;                    // 通道分配
RMAN> show device type;                // IO 设备类型
RMAN> show retention policy;           // 保存策略
RMAN> show datafile backup copies;     // 多个备份的拷贝数目
RMAN> show maxsetsize;                 // 备份集大小的最大值
RMAN> show exclude;                    // 不必备份的表空间
RMAN> show backup optimization;        // 备份的优化
```

## 4.2.3 配置RMAN的默认设置

**1. 配置备份集文件的格式 (format)**

RMAN> configure channel device type disk format '/u05/oracle/rmanback/%U';

备份文件可以自定义各种各样的格式，如下

%c 备份片的拷贝数

%d 数据库名称

%D 位于该月中的第几天 (DD)

%M 位于该年中的第几月 (MM)

%F 一个基于 DBID 唯一的名称,这个格式的形式为 c-IIIIIIIIII-YYYYMMDD-QQ, 其中 IIIIIIIII 为该数据库的 DBID，YYYYMMDD 为日期，QQ 是一个 1-256 的

序列
%n 数据库名称，向右填补到最大八个字符
%u 一个八个字符的名称代表备份集与创建时间
%p 该备份集中的备份片号，从 1 开始到创建的文件数
%U 一个唯一的文件名，代表%u_%p_%c
%s 备份集的号
%t 备份集时间戳
%T 年月日格式(YYYYMMDD)

**2. 配置默认 IO 设备类型 ( device type )**

IO 设备类型可以是磁盘或者磁带，在默认的情况下是磁盘，可以通过如下的命令进行重新配置。

RMAN> configure default device type to disk;
RMAN> configure default device type to sbt;

注意，如果换了一种 IO 设备，相应的配置也需要做修改，如

RMAN> configure device type sbt parallelism 2;

**3. 配置自动分配的通道 ( Chanel )**

RMAN> configure channel device type disk format
'/U01/ORACLE/BACKUP/%U

在运行块中，手工指定通道分配，这样的话，将取代默认的通道分配。

```
RMAN> Run {
        allocate channel cq type disk format='/u01/backup/%u.bak';
        …
      }
```

通道的一些特性：

| | |
|---|---|
| 读的速率限制 | Allocate channel ……rate = integer |
| 最大备份片大小限制 | Allocate channel …… maxpiecesize = integer |
| 最大并发打开文件数（默认 16） | Allocate channel …… maxopenfile = integer |

**4. 配置默认的保存策略 ( Retention Policy)**

保存策略是管理备份与副本有效期或者是否有效的一种方法。恢复数据库的时候 Oracle 不考虑失效的备份。我们可以定义两种保存策略：恢复窗口备份保存策略（recovery window backup retention policy ）和冗余备份保存策略（redundancy backup retention policy ）

备份策略保持 分为两个保持策略：
一个是时间策略，决定至少有一个备份能恢复到指定的日期
一个冗余策略，规定至少有几个冗余的备份。

**恢复窗口备份保存策略**

这种保存策略类型的使用基于数据库可能恢复到的最早的日期。 例如，假设今天是星期一，此前存在 3 个备份。第一个备份在昨天生成的，第二个备份是上星期四生成的，而最后一个备份是 10 天前备份的。假如恢复窗口是 7 天，那么昨天和上星期四的备份是有效备份，而 10 天前的备份会成为废弃备份。下面的命令将恢复窗口配置

为 7 天：

RMAN> configure retention policy to recovery window of 7 days;

**冗余备份保存策略**

使用这种保存策略，RMAN 会从最新备份开是保留 N 个数据备份，其余的废弃。例如，如果有四个备份，而冗余数是 3，那么最早的那个备份将被废弃。下面的命令将备份策略设置为 3：

RMAN> configure retention policy to redundancy 3;

设置 NONE 可以把使备份保持策略失效，Clear 将恢复默认的保持策略

RMAN> configure retention policy to none;

**例：**

保证至少有一个备份能恢复到 Sysdate-5 的时间点上，之前的备份将标记为 Obsolete

RMAN> configure retention policy to recovery window of 5 days;

至少需要有三个冗余的备份存在，如果多余三个备份以上的备份将标记为冗余

RMAN> configure retention policy to redundancy 5;

### 5. 配置多个备份的拷贝数目( backup copies )

如果觉得单个备份集不放心，可以设置多个备份集的拷贝，如：

RMAN> configure datafile backup copies for device type disk to 2;

RMAN> configure archivelog backup copies for device type disk to 2;

如果指定了多个拷贝，可以在通道配置或者备份配置中**指定多个拷贝地点：**

```
RMAN> configure channel device type disk format
                '/u01/backup/%U', '/u02/backup/%U';
RMAN> backup datafile n format '/u01/backup/%U', '/u02/backup/%U';
```

### 6. 设置并行备份( ARALLELISM )

RMAN 支持并行备份与恢复，也可以在配置中指定默认的并行程度。如：

RMAN> configure device type disk parallelism 4;

指定在以后的备份与恢复中，将采用并行度为 4，同时开启 4 个通道进行备份与恢复，当然也可以在 RUN 的运行块中手工分配多个通道来决定备份与恢复的并行程度。并行的数目决定了开启通道的个数。如果指定了通道配置，将采用指定的通道，如果没有指定通道，将采用默认通道配置。

还可以在 BACKUP 命令中使用指定 FILESPERSET 或者指定(datafile 1,4,5 channel c1 tag=DF1)(datafile 2,3,6 channel c2 tag=DF2)

### 7. 设置控制文件自动备份 (autobackup on)

通过如下的命令，可以设置控制文件的自动备份

RMAN> configure controlfile autobackup on;

对于没有恢复目录的备份策略来说，这个特性是特别有效的，控制文件的自动备份发生在任何 backup 或者 copy 命令之后，或者任何数据库的结构改变之后。

可以用如下的配置指定控制文件的备份路径与格式

RMAN> configure controlfile autobackup format for type disk to '%f';

在备份期间，将产生一个控制文件的快照，用于控制文件的读一致性，如下配置：

```
RMAN> configure snapshot controlfile name to
        '/u01/app/oracle/product/9.0.2/dbs/snapcf_U02.f';
```

**8. 设置备份优化选项（optimization）**

可以在配置中设置备份的优化，如

```
RMAN> configure backup optimization on;
```

如果优化设置打开，将对备份的数据文件、归档日志或备份集运行一个优化算法。

### 4.2.4 RMAN 会话的设置

set 命令与 configure 命令很相似，但是 set 命令设置不是永久的。set 命令定义只应用于当前 RMAN 会话的设置。

可以用于 RUN 代码之外的命令有：

**set echo on | off**　　// 显示或关闭 RMAN 显示

**set DBID dbidn**　　// 指定一个数据库的数据库标识符。

下面的 set 命令只能在 RUN 代码中使用：

**set newname：**

用于 TSPITR 或者数据库复制操作，指定新的数据库文件名，将数据库移动到新的系统中并且文件名不同的时候可以用此命令。

**set maxcorrupt for datafile：**

用于定义 RMAN 操作失败之前允许的数据块讹误的数量

**set archivelog destination：**

可以修改存储归档的重做日志 archive_log_dest_1 的目的地。

**set 命令和 until 子句：**

可以定义数据库时间点恢复操作所使用的具体的时间点、SCN 或者日志序列号，例：

```
set until time "to_date('2005/08/01 13:00:00','yyyy/mm/dd hh24:mi:ss')";
```

**set backup copies：**

使用该命令可以定义为备份集的每个备份片创建的镜像副本数。

**例：**

```
RMAN> RUN{
        set maxcorrupt for datafile 3 to 10;
        set backup copies = 2;
        backup database;
      }
```

## 4.3 Copy 镜像拷贝与恢复

Copy 镜像拷贝命令可以创建数据库数据文件、归档重做日志或者控制文件的精确副本。RMAN 副本与这些文件的区别仅在于名称和（或）位置的区别。功能相当于用户管理的备份恢复中的热备份。备份副本的好处是恢复比较快，恢复时可以不用拷贝，指定新位置即可。

Copy 镜像拷贝至少要在 mount 状态下运行。

Copy 镜像拷贝可作为增量备份的 Level 0
Oracle10g 开始，允许使用单条命令"backup as copy"进行数据库拷贝。

### 4.3.1 备份

生成数据文件副本：

```
RMAN> copy datafile 3 to 'd:\backup\datafilecopy\users01.dbf.bak';
RMAN> copy datafile 'd:\oracle\oradata\ora9i\users01.dbf' to
            'd:\backup\datafilecopy\users01.dbf.bak';
```

生成控制文件副本：

```
RMAN> copy current controlfile to … ;
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP is ON;
```

备份 ARCHIVELOG 副本：

```
// 只能一个一个的来备份，而不能指定一个范围
RMAN> copy archivelog 'd:\oracle\oradata\arc\ming_351.arc' to
      'd:\oracle\orabackup\arc\ming_351.arc';
```

**并行设置：**

1. RMAN> configure device type ... parallelism = 3;
   //only 2 channel, one for writting data to disk
2. 手工分配多个通道
3. 在命令中指定多个文件
   RMAN> copy datafile 'xx' to 'xx2', datafile 'yy' to 'yy2',  ...;

**块检查：**

CHECK LOGICAL 选项
MAXCORRUPT 参数
V$COPY_CORRUPTION

在复制操作中，Oracle 服务器进程对每个块执行校验和计算以检测是否有块损坏。RMAN 在还原副本时也要核对校验和。该过程称为物理损坏检测。可以使用 NOCHECKSUM 选项取消校验和操作，从而加快复制进程。如果数据库已在维护块校验和，则此选项无效。缺省情况下，禁用逻辑损坏的错误检查。

可以使用 CHECK LOGICAL 选项测试通过了物理损坏检查的数据和索引块，查看它们是否存在逻辑损坏，如行片或索引条目损坏。如果检测到任何块存在逻辑损坏，则将该块记录到服务器进程的警报日志和跟踪文件中。

可以使用 MAXCORRUPT 参数设置逻辑和物理损坏的阈值。只要在某个文件中检测到的逻辑和物理损坏总和低于该值，则 RMAN 命令完成，同时 Oracle 将损坏块的范围植入到 V$COPY_CORRUPTION 视图。如果超出 MAXCORRUPT，则该命令终止，并且不植入视图。 当并行度比较高时，占用的计算机资源较多，但备份操作完成速度较快。缺省情况下将启用对物理损坏的错误检查。有关在备份过程中遇到的损坏数据文件块的信息将记录在控制文件和警报日志中。

### 4.3.2 恢复

查看所有的 Copy 镜像拷贝：

RMAN> list copy;

数据文件副本还原：

还原时可以 offline 数据文件所属表空间，然后利用 OS 拷贝命令恢复副本。

还可以用 restore (datafile num) from datafilecopy 命令来从数据文件副本中还原数据文件，然后再用 recover 命令来恢复。例如：

RMAN> sql "alter database datafile 5 offline";

RMAN> restore (datafile 5) from datafilecopy;

RMAN> recover datafile 5;

RMAN> sql "alter database  datafile 5 online";

注意，上面的圆括号很重要，如果没有，restore 命令就会执行失败。

# 4.4 Backup 备份与恢复

Backup 备份命令生成 Backup sets (备份集合)，以 oracle 专有的格式保存，由一个完全的所有的备份片集合构成，构成一个完全备份或增量备份。

## 4.4.1 BACKUP 备份命令选项

**设置标记(TAG)**

RMAN> backup database tag= 'test backup';

**限制备份集大小**

RMAN> backup database maxsetsize=100M;

**只备份新增部分**

RMAN> backup incremental level 0 database;

**备份控制文件  同时备份 SPFILE**

RMAN> backup current controlfile;

RMAN> configure controlfile autobackup on;  // 默认是 off

也可以在备份数据库或者文件的时候加上 include current controlfile 选项。例如：

RMAN> backup database include current controlfile;

备份时如果包含了 SYSTEM 表空间，将自动备份控件文件和 SPFILE

RMAN> backup file 1;

使用自动备份进行恢复:

RMAN> restore spfile/controlfile to '/xx/xx' from autobackup;

**备份归档日志(9i)**

RMAN> backup archivelog all;

RMAN> backup ... plus archivelog; // 在备份其他时同时备份归档日志

plus archivelog 隐含如下步骤：

- 运行一个 ALTER SYSTEM ARCHIVELOG CURRENT 命令

- Ø 运行 BACKUP ARCHIVELOG ALL 命令。注意如果备份优化被启用，RMAN 只会备份未备份过的日志
- Ø 备份 BACKUP 命令中定义的文件
- Ø 运行 ALTER SYSTEM ARCHIVELOG CURRENT 命令
- Ø 备份所有的剩下的归档日志

**备份完后删除归档日志**

RMAN> backup ... ARCHIVELOG all delete all input;

**修改备份集的保存策略**

例如：将备份设置为永久有效

RMAN> backup database keep forever logs|nologs;

设置为有效期 180 天

RMAN> backup database keep until time='sysdate+180';

**重写 configure exclude / noexclude**

通过 configure exclude 可以配置 RMAN 不备份上次备份以来没有发生变化的数据文件。如果要确保 RMAN 备份这些数据文件，可以在 backup 命令中添加 noexclude 选项。

例如：

RMAN> backup database noexclude;

**跳过脱机的、不可存取的或者只读的数据文件**

RMAN> backup database skip offline skip inaccessible skip readonly;

**强制备份只读的数据文件**

RMAN> backup database force;

**备份指定周期内没有备份的数据文件**

RMAN> backup database not backed up;

RMAN> backup database not backed up since time='sysdate-2';

**在备份操作期间检查逻辑讹误**

RMAN> backup check logical database;  //在检查逻辑错误的同时进行备份

RMAN> backup validate check logical database;  //只检查

**建立压缩备份集**

RMAN> backup as compressed backupset tablespace users

FORMAT='D:\BACKUP\%d_%s.dbf';

## 4.4.2 RESTORE/RECOVER恢复命令选项

**数据库恢复**

RMAN> restore/revover database ;

**表空间恢复**

RMAN> restore/revover tablespace xx ;

**只读表空间的恢复**

默认情况下，即使丢失了只读的数据文件，RMAN 也不会在执行完全数据库还原操作时候还原只读的数据文件。要在完全恢复期间还原只读的数据文件，就必须在 RESTORE 命令中使用 CHECK READONLY 参数：

RMAN> RESTORE DATABASE CHECK READONLY;

**恢复 SPFILE/控制文件**

使用自动备份恢复 SPFILE/控制文件

RMAN> startup nomount;
RMAN> set dbid=153910023
RMAN> restore controlfile from autobackup
RMAN> restore spfile/controlfile to '/xx/xx' from autobackup ;

或

RMAN> restore controlfile from '/arch/ct_c-2347671489-20060630-00';

联机状态：目标数据库 MOUNT 或 OPEN

RMAN> restore controlfile to 'd:\temp\control01.ctl';

**归档重做日志的还原**

RMAN> RESTORE ARCHIVELOG ALL;
RMAN> RESTORE ARCHIVELOG FROM LOGSEQ=1 UNTIL LOGSEQ=20;
RMAN> RESTORE ARCHIVELOG FROM LOGSEQ=1;

也可以用 SET 命令来指定归档日志的还原位置，例如：

```
RMAN> run
 {
       set archivelog destination to "d:\temp";
   restore archivelog all;
 }
```

**数据文件副本还原**

RMAN> sql "alter datafile 5 offline";
RMAN> restore (datafile 5) from datafilecopy;
RMAN> recover datafile 5;
RMAN> sql "alter datafile 5 online";

请注意，上面的圆括号很重要

**还原检查与恢复测试**

与备份检查一样，还原操作也可以检查是否能正常 restore 或者是否该备份集是否有效。如::

```
RMAN> RESTORE DATABASE VALIDATE;
RMAN> VALIDATE BACKUPSET 218;
RMAN> RESTORE DATABASE VALIDATE CHECK LOGICAL ;
```

**从指定的 tag 恢复：**

```
RMAN> RESTORE FROM tag=’xxxx’;
```

**不完全恢复的还原：**

1. set until time/SCN/
2. RMAN> restore database until scn 1000;
   RMAN>restore database "to_date('2005/08/01 13:00:00','yyyy/mm/dd hh24:mi:ss')";
   RMAN> restore database until sequence 100 thread 1;

**块级别的恢复**

块恢复 Block Media Recovery (BMR)，块是恢复的最小单元，通过块可以减少恢复时间，而且数据文件可以在线。恢复块的时候，必须指定具体的块号，如：

```
RMAN> blockrecover datafile 6 block 3;
```

具体请见 4.7.4 块媒体恢复 Block Media Recovery (BMR)

### 4.4.3 非归档模式下的 BACKUP 备份与恢复

恢复目录： 打开
目标数据库：例程启动，数据库加载，mount 不能 open

因为目标数据库不在归档模式下，所以当进行备份/恢复操作的时候数据库无法打开。目标数据库只能在 MOUNT 状态不能 Open，所以属于脱机备份。

非归档模式不备份 redo 日志，只有完全备份和 readonly/offline 表空间和数据文件备份是有意义的，所以非归档模式最好不用 RMAN 进行备份，备份语法与归档模式相同，所以这里只做简单介绍。

#### 4.4.3.1 全库备份

例： 使用默认的设置脱机全备份的语句

```
RMAN> shutdown immediate;
RMAN> startup mount;
RMAN> backup database;
RMAN> startup;
```

例：不使用默认的设置执行脱机备份操作 ，在备份命令中指定备份选项

```
RMAN> shutdown immediate;
RMAN> startup mount;
RMAN> run {
      allocate channel c1 type disk format '/xxx/ming_%U';
```

```
        allocate channel c2 type disk format '/xxx/ming_%U';
        backup full tag full_db_backup format "/backups/db_t%t_s%s_p%p" (database);
        backup current controlfile;
        release channel c1 ;
        release channel c2 ;
    }
```

在这个例子中，我们分配了两个通道，备份位置是在/xxx。默认情况下，如果备份数据文件 1 (SYSTEM 表空间)，控制文件和参数文件也会备份。

可以通过下面的命令显示恢复目录中记载的备份集信息：

RMAN> list backupset of database;

#### 4.4.3.2 全库备份的恢复

```
$ rman target /
RMAN> startup mount
RMAN> restore database;
RMAN> recover database;
RMAN> alter database open resetlogs;
```

#### 4.4.3.3 表空间备份

只有 readonly/offline 表空间的备份才有意义。

```
RMAN> run {
  2> allocate channel dev1 type disk;
  3> backup
  4> tag tbs_users_read_only
  5> format "/oracle/backups/tbs_users_t%t_s%s"
  6> (tablespace users);
  7> }
```

使用下列命令来显示恢复目录中该表的备份信息：

RMAN> list backupset of tablespace users;

#### 4.4.3.4 表空间备份的恢复

```
RMAN> RUN{
        sql "alter tablespace xx offline immediate;"
        restore tablespace xx;
        recover tablespace xx;
```

```
        sql "alter tablespace xx online;"
}
```

#### 4.4.3.5 备份控制文件

```
RMAN> run {
  2> allocate channel dev1 type "SBT_TAPE";
  3> backup
  4> format "cf_t%t_s%s_p%p"
  5> tag cf_monday_night
  6> (current controlfile);
  7> release channel dev1;
  8> }
```

注：数据库完全备份将自动备份控制文件。(或者备份时加 include current controlfile)

### 4.4.4 归档模式下的BACKUP备份与完全恢复

要用 RMAN 进行联机备份操作，数据库就必须位于 ARCHIVELOG 模式。恢复目录必须打开，目标数据库例程必须启动，数据库加载或者打开。

这部分只介绍完全恢复，不完全恢复见下一节。

#### 4.4.4.1 整库备份与恢复

**备份命令：**

只备份数据文件 (如果 configure controlfile autobackup on; 将自动包括控件文件，SPFILE)：

RMAN> backup database;

同时备份归档日志，然后将备份后的归档日志删除

RMAN> backup database plus archivelog delete input;

明确指定同时备份控件文件：

```
RMAN> run{
      allocate channel c1 type disk;
      sql 'alter system archive log current';
      backup full database include current controlfile tag 'dbfull'
      format '/u06/oracle/rmanback/full_%u_%s_%p';
      sql 'alter system archive log current';
      release channel c1;
}
```

可以用 RMAN 的 plus archvielog 选项简化数据库备份：

```
RMAN> run {
2> backup database
3> format '/xxfull%d_%T_%s'
4> plus archivelog
5> format '/xx/arch_%d_%T_%s'
6> delete all input;
7> }
```

**完全恢复：**

目标数据库必须是 mount 状态

```
$ rman target /
RMAN> startup mount
RMAN> restore database;
RMAN> recover database;
RMAN> alter database open;
```

#### 4.4.4.3 表空间的备份与恢复

**备份命令：**

RMAN> backup tablespace users ;

例：

```
RMAN> RUN{
        allocate channel c1 type disk;
        backup tablespace users tag 'ts_users' format '/oracle/rmanback/ts_%u_%s_%p' ;
        release channel c1;
}
```

**恢复：**

如果我们只丢失了特定的表空间的数据文件，那么我们可以选择只恢复这个表空间，而不是恢复整个数据库。表空间恢复可以在不关闭数据库的情况下进行，只需要将需要恢复的表空间 offline。

```
RMAN> RUN{
      sql "alter tablespace xx offline immediate;"
      restore tablespace xx;
      recover tablespace xx;
      sql "alter tablespace xx online;"
}
```

恢复到一个不同的位置：

```
RMAN> RUN{
      sql "alter tablespace xx offline immediate;"
      SET NEWNAME for datafile 1 to '/xx';
      restore tablespace xx;
      switch datafile 1;
      recover tablespace xx;
      sql "alter tablespace tbs1 online;"
}
```

#### 4.4.4.3 数据文件的备份与恢复

**备份命令：**

```
RMAN> backup datafile 3;
RMAN> backup datafile 'D:\ORACLE\ORADATA\TEST\TEST.DB';
```

**恢复命令：**

数据文件恢复与表空间恢复类似。假设数据文件号为 5 的文件丢失，文件名是：'E:\ORACLE\ORADATA\USERS.DBF'， 那么我们恢复的时候可以指定文件号，也可以指定文件名。

```
RMAN> run {
2> allocate channel dev1 type disk;
3> sql "alter tablespace users offline immediate";
4> restore datafile 4;   --或者  restore 'E:\ORACLE\ORADATA\USERS.DBF'
5> recover datafile 4;
6> sql "alter tablespace users online";
7> release channel dev1;
8> }
```

恢复到一个不同的位置：

```
$ rman target /
RMAN> startup mount
RMAN> RUN{
            sql "alter tablespace users offline immediate";
            SET NEWNAME for datafile 9 to '/xx/user01.dbf';
            restore datafile 9;
            switch datafile all;
            recover datafile 9;
            sql "alter tablespace users online";
      }
```

#### 4.4.4.4 归档重做日志的备份与恢复

**备份：**

整库备份的同时，备份所有归档 (以及联机日志)：

```
RMAN> backup database plus archivelog;
```

备份所有归档：

```
RMAN> backup archivelog all;
```

备份两天来的归档：

```
RMAN> backup archivelog from time='sysdate-2' [to time='xxx'] ;
```

备份从 sequence 1 开始的归档：

```
RMAN> backup archivelog from sequence 1 [to sequence ='n];
```

备份没有三次备份的归档：

```
RMAN> backup archivelog not backed up 3 times;
```

备份所有归档，然后删除归档：

```
RMAN> backup archivelog all delete input;
```

**恢复：**

显示恢复目录中的归档日志：

```
RMAN> list backupset of archivelog all;
```

一般情况下，在 RMAN 的普通恢复过程中，不必恢复归档的重做日志。不过偶尔也需要恢复重做日志，例如我们用 Log Miner 来从归档中查找一些东西。

RMAN 命令举例：

```
RMAN> RESTORE ARCHIVELOG ALL;
RMAN> RESTORE ARCHIVELOG FROM LOGSEQ=1 UNTIL LOGSEQ=20;
RMAN> RESTORE ARCHIVELOG FROM LOGSEQ=1;
```

可以用 SET 命令来指定归档日志的还原位置，例如：

```
RMAN> run{
        set archivelog destination to "d:\temp";
        restore archivelog all;
  }
```

需要注意的是，即使新的归档日志目录不同于默认的归档日志目录，如果 Oracle 判定日志已存在，也不会恢复该归档日志文件。

#### 4.4.4.5 联机日志的备份

联机日志不能用 RMAN 来备份，可以先将其归档，再备份。为了实现这点，必须在 RMAN 中执行归档命令语句：

```
RMAN> run {
  2> allocate channel dev1 type disk;
  3> sql "alter system archive log current";
  4> backup (archivelog from time "sysdate-1" all delete input)
  5> "format "/oracle/backups/log_t%t_s%s_p%p";
  6> release channel dev1;
  7> }
```

上面的脚本可以在执行完一个完整的联机数据库备份后执行，确保所有的重做日志可以将数据库恢复到一个一致性的状态。

当然，也可以在全库备份时使用 plus archvielog 选项，将自动完成联机日志的备份。

#### 4.4.4.6 控制文件和服务器参数文件的备份与恢复

**备份：**

// 设置文件名格式

RMAN> set controlfile autobackup format for device type disk to 'ctl_%F';

1. RMAN> configure controlfile autobackup on;  // backup database 时将自动备份
2. RMAN> backup current controlfile;
3. RMAN> backup .... include control file;
4. RMAN> backup file 1;  // system datafile 自动备份

**恢复 SPFILE：**

SPFILE（PFILE）的丢失/损坏，对数据库不会产生致命影响，可以从其他方式恢复。不过既然 RMAN 的备份计划中包括了 SPFILE 的备份，那么就可以使用 RMAN 来还原 SPFILE 了。

```
$ rman target / catalog "rman/rman@db"
RMAN> set dbid=153910023   // SET DBID 这个步骤是不能省略的，否则会报错。
RMAN> restore spfile from autobackup [MAXDAYS 100];  // 或者 restore spfile;
RMAN> startup force
```

或者从某个备份集恢复:

RMAN> restore spfile from backupset bs_num 命令。

使用 dbms_backup_restore 包恢复服务器参数文件：

在一些不常见的情况下，我们可能需要直接使用 dbms_backup_restore 包来恢复 spfile。当然这个包也可以用来恢复其它数据，是常规办法都没有用的时候的一个利器。这个包可以在数据库 NOMOUNT 状态下使用。假设我们有一个自动备份文件 C-2600315304-20060829-02，我们需要从这里恢复数据，那么可以通过执行下面的脚本来完成：

```
SQL>
DECLARE
  DEVTYPE VARCHAR2(256);
  DONE BOOLEAN;
BEGIN
  DEVTYPE:=DBMS_BACKUP_RESTORE.deviceallocate(NULL);
  DBMS_BACKUP_RESTORE.restoresetdatafile;
  DBMS_BACKUP_RESTORE.restorespfileto('/back/SPFILE.ORA');
  DBMS_BACKUP_RESTORE.restorebackuppiece(
                        '/back/C-2600315304-20060829-02',DONE=>done);
  DBMS_BACKUP_RESTORE.devicedeallocate(NULL);
END;
```

```
/
```

**恢复控制文件：**

RMAN> startup nomount;

RMAN> set dbid=153910023

RMAN> restore controlfile from autobackup

或

RMAN> restore controlfile from '/arch/ct_c-2347671489-20060630-00';

联机状态：目标数据库 MOUNT 或 OPEN

RMAN> restore controlfile to 'd:\temp\control01.ctl';

然后再执行恢复数据库的其他步骤：

RMAN> restore database;

RMAN> recover database;

RMAN> alter database open resetlogs;

使用 dbms_backup_restore 包恢复控制文件：

```
SQL>
    DECLARE
    DEVTYPE VARCHAR2(256);
    DONE BOOLEAN;
BEGIN
    DEVTYPE:=DBMS_BACKUP_RESTORE.deviceallocate(NULL);
    DBMS_BACKUP_RESTORE.restoresetdatafile;
    DBMS_BACKUP_RESTORE.restorecontrolfileto('/back/CONTROL01.CTL');
    DBMS_BACKUP_RESTORE.restorebackuppiece
                ('/back/C-2600315304-20060829-00',DONE=>done);
    DBMS_BACKUP_RESTORE.devicedeallocate(NULL);
END;
/
```

#### 4.4.4.7 备份集的备份的备份与恢复

**备份：**

备份所有备份集：

RMAN> backup backupset all;

备份指定备份集：

RMAN> backup backupset bs_num;

**恢复：（这种备份只是增加一个镜像，不用恢复）**

主要用于改变备份集的位置，或者创建多个镜像备份，比如将备份集从硬盘备份到磁

带。如果要详细了解，见我在 ITPUB 上发的帖子：

http://www.itpub.net/viewthread.php?tid=1007747&page=1&extra=

### 4.4.5 归档模式下的不完全恢复

不完全恢复就意味着有数据的丢失。引起不完全恢复的原因有很多，如丢失了联机日志或某个归档日志。另外如果出现了严重损害数据库的用户错误，比如某用户错误的删除了某个重要的数据，那么数据库也要恢复到这个错误操作之前。

不完全恢复会影响整个数据库，需要在 MOUNT 状态下进行。在不完全恢复完成之后，通常需要使用 resetlogs 选项来打开数据库。resetlogs 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。数据库的逻辑生存期也被称为一个对应物(incarnation)。每次使用 resetlogs 选项来打开数据库后都会创建一个新的数据库对应物，这对于恢复操作来说非常重要。每次使用 resetlogs 后，SCN 计数器不会被重置，但是 Oracle 会重置联机日志序列号，同时还会重置联机重做日志内容。因此执行了 resetlogs 之后，应该立即重新备份整个数据库，否则恢复起来相当麻烦。 （注:Oracle 10g 中已经可以在 resetlogs 之后不备份数据库，恢复的时候能够穿越 resetlogs）

使用 RMAN 执行不完全恢复操作时需要完成的一个工作是建立恢复目标。恢复目标是要终止恢复进程的点，可以是时间点、指定的 SCN 或者一个日志序列号。我们可以在 run 代码中使用 set 命令和 until time、until scn、until sequence 参数。

也可以选择在 RESTORE 和 RECOVER 命令中直接使用 UNTIL TIME、UNTIL SCN、或者 UNTIL SEQUENCE 参数，这样就可以避免使用 run 代码。例如；

```
startup mount;
restore database "to_date('2005/08/01 13:00:00','yyyy/mm/dd hh24:mi:ss')";
recover database "to_date('2005/08/01 13:00:00','yyyy/mm/dd hh24:mi:ss')";
alter database open resetlogs;
```

#### 4.4.5.1 基于 SCN 的恢复

如果知道数据库出错前的 SCN，可以将数据库还原到指定 SCN 状态。

```
$ startup mount;
RMAN> run{
      allocate channel d1 type disk;
      restore database until scn 1317011;          --或者  set until scn 1317011
      recover database until scn 1317011;
      sql 'alter database open resetlogs';
      release channel d1;
}
```

#### 4.4.5.2 基于时间的恢复

下面使用 set until time 命令为 2005 年 8 月 1 日下午 1 点的恢复目标：

```
$ startup mount;
```

```
RMAN> run{
      set until time "to_date('2005/08/01 13:00:00','yyyy/mm/dd hh24:mi:ss')";
      restore database;
      recover database;
      alter database open resetlogs;
}
```

执行上面的命令时， RMAN 会查找与恢复目标时间最近、但是不包含恢复目标时间及以后时间的备份集，并且从这个备份中还原数据库。如果数据库非归档模式，那么恢复操作会在备份集的时间点停止；否则 RECOVER 命令会应用恢复目标之前的归档重做日志或需要的增量备份。

#### 4.4.5.3 基于日志序列的恢复

RMAN 允许用户将数据库恢复到指定的归档重做日志序列号。如果归档的重做日志中有间隙（某个归档日志文件或备份损坏或丢失） ，使用这种方法就很方便。间隙通常意味着我们只能将数据库还原到间隙开始的地方。

```
SQL> startup mount;

RMAN> restore database until sequence 100 thread 1;      --not include 100
RMAN> recover database until sequence 100 thread 1;

SQL> alter database open resetlogs;
```

```
RMAN> RUN {
2> SET UNTIL SEQUENCE 120 THREAD 1;
4> RESTORE DATABASE;
5> RECOVER DATABASE;    --recovers through log 119 not include 120
6> ALTER DATABASE OPEN RESESTLOGS;
7> }
```

## 4.5. RMAN 查看信息　　List 与 Report

### 4.5.1 恢复目录相关视图

恢复目录本身有一组视图，用于存放目标数据库与备份信息，可以用 RMAN 用户登录数据库进行查看，例：

RC_DATABASE

RC_DATAFILE
RC_STORED_SCRIPT
RC_STORED_SCRIPT_LINE
RC_TABLESPACE

### 4.5.2 RMAN 动态性能视图

以下是目标数据库上与 RMAN 备份有关系的一些动态性能视图，可以用 SYS 用户进行查询。

V$ARCHIVED_LOG
V$BACKUP_CORRUPTION
V$COPY_CORRUPTION
V$BACKUP_DEVICE
V$CONTROLFILE_RECORD_SECTION
V$BACKUP_DATAFILE　用于通过确定各数据文件中的块数来创建大小相同的备份集。通过它也可以找出数据文件中已损坏的块数。
V$BACKUP_REDOLOG　显示在备份集中存储的归档日志。
V$BACKUP_SET　显示已经创建的备份集。
V$BACKUP_PIECE　显示为备份集创建的备份片。

这里还有一个视图，可以大致的监控到 RMAN 备份进行的程度。如通过如下的 SQL 脚本，将获得备份的进度。

```
SQL> SELECT SID, SERIAL#, CONTEXT, SOFAR, TOTALWORK,
2 ROUND(SOFAR/TOTALWORK*100,2) "%_COMPLETE"
3 FROM V$SESSION_LONGOPS
4 WHERE OPNAME LIKE 'RMAN%'
5 AND OPNAME NOT LIKE '%aggregate%'
6 AND TOTALWORK != 0
7 AND SOFAR <> TOTALWORK;
```

要在备份过程中将某一进程与一个通道关联起来，请：

1. 启动恢复管理器并连接到目标数据库和恢复目录（与后者的连接是可选的）。

```
rman target / catalog rman/rman@rcat
```

2. 在分配通道后，设置 COMMAND ID 参数，然后复制所需的对象。

```
run {
      allocate channel t1 type disk;
      set command id to 'rman';
      copy datafile 1 to '/u01/backup/df1.cpy';
      release channel t1;}
```

3. 查询 V$SESSION_LONGOPS 视图以获得复制的状态。

```
SELECT sid, serial#, context, sofar, totalwork
```

```
        round(sofar/totalwork*100,2) "% Complete",
   FROM v$session_longops
   WHERE opname LIKE 'RMAN:%'
   AND opname NOT LIKE 'RMAN: aggregate%';
```

4. 使用 SQL*Plus 并查询 V$PROCESS 和 V$SESSION 以获得 SID 和 SPID。然后，  使用

操作系统实用程序来监视进程或线程。

```
SELECT sid, spid, client_info
FROM v$process p, v$session s
WHERE p.addr = s.paddr
AND client_info LIKE '%id=rman%';
```

### 4.5.3 List

List 命令是一种在数据库控制文件或者恢复目录中查询备份的历史信息的方法。List 提供了一组信息，可以提供各种备份的信息，如对应物、备份集、归档日志备份、控制文件备份等等。

**列出对应物：**

RMAN> list incarnation;

**列出备份概要信息：**

RMAN> list backup summary;

**按备份类型列出备份：**

RMAN> list backup by file;

**获得备份的详细信息，包括备份片的物理文件名：**

RMAN> list backup;

RMAN> list backupset bs#;

**或者按照 TAG 来查：**

RMAN> list backup tag=tab_number;

**列出过期的备份：**

RMAN> list expired backup;

**按照表空间和数据文件来列出备份：**

列出 USERS 表空间的备份：

RMAN> list backup of tablespace USERS;

列出文件 5 的备份：

RMAN> list backup of datafile 5;

列出文件 E:\ORACLE\USERS.DB 的备份：

RMAN> list backup of datafile ' E:\ORACLE\USERS.DB ';

**列出控制文件的备份：**

RMAN> list backup of controlfile;

**列出归档日志的备份：**

RMAN> list archivelog all;

RMAN> list backup of archivelog all;

**列出副本：**

RMAN> list copy  列出所有的副本。

RMAN> list copy of controlfile  列出控制文件副本

RMAN> list copy of archivelog all  列出所有归档日志副本

RMAN> list copy of database  列出数据库所有数据文件的副本

### 4.5.4 Report

Report 命令被用于判断数据库的当前可恢复状态和提供数据库备份的特定信息，可以检测哪些文件需要备份，哪些备份能被删除以及那些文件能不能获得的信息。可以报告数据库的所有能备份数据文件对象，包括数据文件名、文件号、表空间、文件大小、是否含有回滚段等。

RMAN> report schema

或者

RMAN> REPORT SCHEMA AT TIME 'SYSDATE-14';

RMAN> REPORT SCHEMA AT SCN 1000;

RMAN> REPORT SCHEMA AT SEQUENCE 100 THREAD 1;

**报告需要备份的数据文件**

RMAN> report need backup [ redundancy | days | incremental n];

**报告过期了的数据文件或者不可用的备份与拷贝**

RMAN> Report obsolete [orphan]

**报告最近没有备份的数据文件**

RMAN> report need backup days=10;  // 恢复需要超过 10 天的归档日志

RMAN> report need backup incremental=3; // 恢复时需要超过 3 增量的文件报表

文件 增量 名称

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

8  15  E:\ORACLE\ORADATA\MING_RECOVER\MINGDICT.DB

9  15  E:\ORACLE\ORADATA\MING_RECOVER\MINGLOB.DB

这个报告中，列出的数据文件，在进行恢复的时候，需要从 3 个以上的增量备份文件中恢复。我们知道如果需要从很多文件中恢复，会影响恢复速度，可以根据情况来重新备份这些文件。

**报告备份冗余或恢复窗口**

我们可以执行 report need backup redundancy 来确定为满足冗余备份策略而需要备份的文件。例如：

RMAN> report need backup redundancy=2;  // 文件冗余备份少于 2 个

我们也可以按照恢复窗口来查找需要备份的文件。比如我们要求恢复窗口小于 2 天，那么用下面的命令：

RMAN> report need backup recovery window of 2 days;  //文件报表的恢复需要超过 2 天的归档日志

这个命令等同于：report need backup days=2;

# 4.6 RMAN 的管理与维护

如果数据库做了 alter database open resetlogs; 就需 reset database，如果有库结构变化，就需要 resync catalog

### 4.6.1 加入目录数据库

如果存在以前创建的备份数据想注册到目标数据库，可以采用如下手工方式加入到恢复目录中，

RMAN> CATALOG datafilecopy '/oracle/ .... /system01.dbf';
RMAN> CATALOG controlfilecopy  '/oracle/CONTROL01.CTL.BCK';
RMAN> CATALOG archivelog  '/oracle/arc001.log';

**10g 新特性：**

1. 可以手工加入手工加入备份片，所以，只要你的备份还在，归档还在，即使 catalog database 崩溃，控制文件重建，照样可以用这些备份来做恢复。

RMAN> CATALOG backupiece '/oracle/xxxx';

2. 扫描整个目录，如果备份片或者归档日志文件太多，可以放到一个目录中，一次性扫描就行。

RMAN> CATALOG START WITH '/xxx/arch_logs';

### 4.6.2 恢复目录的建立、升级与删除

恢复目录的建立请见：4.2.1 建立 Recovery Catalog 恢复目录

当恢复目录管理的某个数据库进行了升级，只要版本不高于恢复目录就没有问题，下面的命令查询恢复目录的版本 (RMAN 用户)：

SQL> select version from rcver;

如果要高于恢复目录，那么恢复目录就必须进行升级。用下面的命令即可：

RMAN> upgrade catalog;

恢复目录可以采用如下命令删除

RMAN> DROP CATALOG;

### 4.6.3 同步或重置 CROSSCHECK命令(交叉校验)

如果目标数据库物理对象发生了变化，如添加了一个数据文件，需要用如下命令同步：

RMAN> resync catalog;

如果目标数据库 reset 了数据库，需要用如下命令同步

RMAN> reset database;

(必须使用 catalog 数据库)

打开数据库的时候，每次使用 resetlogs 参数都会创建数据库的一个新对应物。如果这个操作是在 rman 中完成的，rman 会自动更新恢复目录。如果是在 rman 之外完成，如 sql*plus，那么就必须手工重置对应物。如：

RMAN> reset database;

另外有的时候，我们需要改成之前的某个对应物，可以用 list incarnation 来显示出对应物列表之后用下面的命令：

RMAN> reset database to incarnation incarnation_num;

当手工删除了数据库的归档文件后，要执行以下脚本同步

RMAN> allocate channel for maintenance type disk;

RMAN> change archivelog all crosscheck;

RMAN> release channel;

当手工删除了数据库的 RMAN 备份后，要执行以下脚本来同步

RMAN> allocate channel for maintenance type disk;

RMAN> crosscheck backup;

RMAN> delete expire backup;

RMAN> release channel;

**关于 CROSSCHECK 交叉校验命令**

该命令用于核对磁盘和磁带上的备份文件，以确保 RMAN 资料库与备份文件保持同步。该命令只会检查 RMAN 资料库所记载的备份文件。当执行 CROSSCHECK 命令时，如果资料库记录不匹配于备份文件的物理状态，那么该命令会更新资料库记录的状态信息。当使用 CROSSCHECK 命令核对备份文件之后，备份文件的状态会包括 AVALIABLE、UNAVALIABLE 和 EXPIRED 三种，如果备份文件处于 EXPIRED 状态，则说明物理文件已经被手工删除或者损坏。注意，如果备份文件处于 EXPIRED 状态，应该使用 DELETE 命令删除该备份文件。

**核对所有备份集**

RMAN>CROSSCHECK BACKUP;

**核对所有数据文件的备份集**

RMAN> CROSSCHECK BACKUP OF DATABASE;

**核对特定表空间的备份集**

RMAN>CROSSCHECK BACKUP OF TABLESPACE SYSTEM;

**核对特定数据文件的备份集**

RMAN>CROSSCHECK BACKUP OF DATAFILE 4;

**核对控制文件的备份集**

RMAN>CROSSCHECK BACKUP OF CONTROLFILE;

**核对 SPFILE 的备份集**

RMAN> CROSSCHECK BACKUP OF SPFILE;

**核对归档日志的备份集**

RMAN> CROSSCHECK BACKUP OF ARCHIVELOG SEQUENCE 3;

**核对所有映像副本**

RMAN> CROSSCHECK COPY;

**核对所有数据文件的映像副本**

RMAN> CROSSCHECK COPY OF DATABASE;

**核对特定表空间的映像副本**

RMAN> CROSSCHECK COPY OF TABLESPACE USERS;

**核对特定数据文件的映像副本**

RMAN> CROSSCHECK COPY OF DATAFILE 4;

**核对控制文件的映像副本**

RMAN> CROSSCHECK COPY OF CONTROLFILE;

**核对归档日志的映像副本**

RMAN> CROSSCHECK COPY OF ARCHIVELOG SEQUENCE 4;

### 4.6.4 修改备份的可用状态、保存策略 Change命令

Change 命令可以修改备份的状态是可用（available）还是不可用（unavailable）。对于不可用的备份，还原与恢复期间不会被考虑到，不过执行 delete expired 命令期间也不会删除这些记录。

例如：

```
RMAN> change backup of database tag='GOLD' unavailable;
RMAN> change copy of database like '%GOLD%' available;
RMAN> change archivelog 'd:\arc\arch_001.arc' unavailable;
RMAN> change backupset 4981 available;
RMAN> change backup of database available;
RMAN> change archivelog all available;
RMAN> change archivelog all backed up 5 times unavailable;
```

当一个给定的备份或者副本根据备份的保存策略的标准而被废弃的时候，RMAN 并不会自动删除这个备份或者副本，而只是标记这个备份为废弃。我们可以用 report obsolete 命令来查看标记为废弃的备份。可以使用 change 命令来将一个备份修改为永久保留的备份，也可以修改为要保存多少天的备份。还可以使用 change … nokeep 来手工丢弃一个备份。

举例：

将 4421 备份集标记为废弃

RMAN> change backupset 4421 nokeep;

将 4421 备份集标记为 7 天内有效

RMAN> change backupset 4421 keep until time 'sysdate+7' logs;

将 4421 备份集标记为永久有效

RMAN> change backupset 4421 keep forever logs;

废弃的备份集并不真正的删除，如果需要物理删除，则可以用下面的命令：

RMAN> delete obsolete;

### 4.6.5 查看与删除过时的备份信息

列出已经过时的备份：

RMAN> report obsolete;

定义 delete 通道：

RMAN> allocate channel for delete/maintenance type disk;

删除过时的备份信息

RMAN> allocate channel for maintenance type disk;

RMAN> change backupset id delete;

RMAN> release channel;

当手工删除了数据库的 RMAN 备份文件后，要执行以下脚本进行同步：

RMAN> allocate channel for maintenance type disk;

RMAN> crosscheck backup;

RMAN> delete expired backup;  -- 删除过期的备份

RMAN> delete obsolete;  -- 删除废弃的备份

RMAN> release channel;

在 8i 和 8i 之前的版本只能用 change…delete 命令来删除物理备份。

RMAN> change archivelog until logseq=500 delete;

### 4.6.6 恢复目录记录的删除

如果不加以维护，具有 DELETE 状态的旧备份会一直驻留在恢复目录中。为了解决这个问题，Oracle 提供了?/rdbms/admin/prgrmanc.sql，这个脚本可以删除恢复目录中具有 DELETE 状态的记录。如果想删除旧的对应物，那么我们可以删除 dbinc 表的记录。例如，如果要删除对应物是 2 的记录，则可以执行下面的语句：

SQL> delete from dbinc where dbinc_key=2;

### 4.6.7 备份RMAN数据库

RMAN 自己的数据库也需要备份，但是本身很小，而且不是经常发生变化，所以在每次 RMAN 备份完成后，可以对 RMAN 数据库备份。

$ EXP RMAN/RMAN OWNER=RMAN FILE=RMAN.DMP ROWS=Y GRANTS=Y COMPRESS=Y CONSISTENT=Y

### 4.6.8 备份检查  验证备份的可恢复性

我们可以通过 Validate 命令来检查是否能备份，如数据文件是否存在，是否存在坏块不能被备份，通过使用 RESTORE DATABASE VALIDATE; 和 RESTORE DATABASE VALIDATE CHECK LOGICAL; 可以检查最新的备份是否可恢复。这些命令并不真正的执行恢复，而是检查备份中是否有讹误。如果使用 CHECK LOGICAL 选项，还将检查数据和索引段中是否存在逻辑讹误。

RMAN> RESTORE DATABASE VALIDATE;

RMAN> RESTORE DATABASE VALIDATE CHECK LOGICAL ;

RMAN> VALIDATE BACKUPSET 218;

RMAN> VALIDATE BACKUPSET bs CHECK LOGICAL ;

RMAN> BACKUP VALIDATE DATABASE ARCHIVELOG ALL;

对于数据库与数据文件，可以从指定的 tag 恢复：

RMAN> RESTORE DATAFILE 1 FROM TAG='tag name' ;

### 4.6.9 登记目标数据库：

一个恢复目录可以注册多个目标数据库，注册目标数据库的命令为：

$ RMAN catalog rman/rman target user/pwd @db;

RMAN> register database

### 4.6.10 注销数据库

注销数据库不是简单的在 RMAN 提示下反注册就可以了，需要运行一个程序包，过程如下：

1. 连接目标数据库，获得目标数据库 ID

   $ RMAN target internal/password catalog rman/rman@rcdb;

2. 以 RMAN 用户登录，查询恢复目录，得到更详细的信息

   SQL> SELECT db_key, db_id FROM db WHERE db_id = 1231209694;

   DB_KEY  DB_ID

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

   1  1237603294

   1 row selected.

3. 运行过程 dbms_rcvcat.unregisterdatabase 注销数据库，如

SQL> EXECUTE dbms_rcvcat.unregisterdatabase(1 , 1237603294)

### 4.6.11 重新启动备份

对于异常结束了的备份，很多人可能不想再重新开始备份了吧，特别是备份到 90%以上，因为异常原因终止了该备份，那怎么办呢？RMAN 提供一个重新开始备份的方法，通过简单的命令，你就可以只备份那不到 1%的数据了。

RMAN> backup not backed up since time 'sysdate-14' database plus archivelog;

### 4.6.12 脚本及自动运行

脚本的自动/定时运行可以通过 Windows 计划任务，UNIX/Linux 的 Crontab/at 命令，或者第三方软件实现，详细介绍有待补充。

(未完成)

1、 编写 rman 批处理文件

2、 编写 Shell 脚本

set ORACLE_SID =xxxx 或 export $ORACLE_SID

rman target / msglog  /xxx.log cmdfile=/xxx/backup.rman

3、 设定执行计划

# 4.7 高级主题

## 4.7.1 使用RMAN备份集恢复DB到其他机器 (1常规方法)

**用途：**

利用生产 DB 的备份集建立测试环境或备用数据库

（ 设置成相同的 DBID，不需要连接原始 DB，只需要利用 **Catalog** ）

**步骤：**

**1．准备工作，配置目标 DB 环境**

- 在目标 DB 上设置 ORACLE 相关的环境变量：记下生产 DB 的 ORACLE_SID，DBID

  $ set ORACLE_SID=ming
- 在目标 DB 创建与生产 DB 相同的目录，将生产 DB 的 pfile 文件 FTP 到目标 DB，或者，在第二步中可以从备份集中进行恢复得到 PFILE。
- 在目标 DB 上创建生产 DB 的 RMAN 备份集存放的目录，没有使用 catalog 情况时这点必然保持一样
- 移植备份集(确定包含控制文件，数据文件，归档文件)到目标 DB
- 配置网络连接

**2．目标 DB 在 nomount 状态下恢复 pfile 和 controlfile**

连接恢复目录和目标数据库：

$ RMAN target / catalog rman/rman@rcdb

RMAN > set DBID= 3324789823

RMAN > startup nomount

RMAN> restore spfile to pfile '/home/oracle/pfile' [from 'backupset name'];

RMAN> restore controlfile [to  'xx '] from '/xx/ ORADB_ctl_20070111_c-xx';

(控制文件的恢复可以参考 4.4.4.6 中其他方法)

**3．启动目标 DB 到 mount，在目标 DB 上 Restore 和 Recover**

RMAN> alter database mount; --可能会提示找不到密码文件，不用理会

如果备份集的位置改变了，需要先进行交叉校验：

RMAN> catalog start with '/newdir';

RMAN> crosscheck backup;

RMAN> report schema;

然后 restore / recover ：

```
RMAN> run {
      allocate channel ch_disk_db_1 device type disk;
      restore database;
      recover database until sequence=2412 thread=1;
                --最后一个 arachivelog 的 sequence
      release channel ch_disk_db_1;
```

}

**4．Resetlogs 打开目标 DB**

联机日志文件并没有恢复，所以需要 resetlogs

SQL> alter database open resetlogs;

**5．重建临时表空间，重建密码文件，立即备份数据库**

**重建临时表空间：**

因为备份时不备份临时文件，v$tempfile 中查询为空，所以可以增加一个文件即可：

SQL> alter tablespace temp add tempfile '/u05/oracle/oradata/crm/temp01.dbf' size 10M;

**重建密码文件**

可以通过 orapwd 命令重建密码文件

**备注：恢复时有时遇到的一个问题**

RMAN-03002  ORA-27064 错误

解决办法：  把数据文件 resize 到 oracle 的 block size 的整数倍

SQL> alter database datafile 'xxx' resize n MB;

### 4.7.2 使用RMAN备份集恢复DB到其他机器 (2复制方法)

使用 RMAN DUPLICATE 命令可以在保留目标数据库的基础上依靠目标数据库(Target Database)的备份创建一个副本数据库(Duplicate Database) 。该副本即可与目标数据库完全相同，也可仅包含目标数据库表空间的一部子集。目标站点(Target Site)以及副本站点(Duplicate Site)甚至可以在同一台机器上。

（ 生成新的唯一的 DBID，如果控制文件中仍保存有备份信息，可以不连接 Catalog ）

**步骤：**

**1．准备工作，配置副本 DB 环境**

1. 备份主数据库（包括数据文件、控制文件以及所有归档），并把该备份集拷贝到副本数据库机器同样的目录下。

2. 拷贝主数据库的初始化参数文件 pfile 到副本数据库机器上，并根据需要作相应修改，创建密码文件。

3. 配置主数据库到副本数据库的连接。

**2．启动副本数据库到 nomount 下，目录数据库必须 MOUNT (或 OPEN)**

$ sqlplus /nolog

SQL> conn / as sysdba

SQL> startup nomount

**3．运行 RMAN，分别连接主数据库与副本数据库实例**

控制文件中保存有备份信息，可以不连接 Catalog

在副本数据库上：

$ rman target sys/change_on_install@MING auxiliary /

**4．运行复制命令**

如果没有配置自动分配通道的话，需要手工指定至少 1 条辅助通道。

DUPLICATE 命令将自动完成：将还原所有数据文件，重新创建控制文件，并利用新的参数文件启动恢复数据库到一致状态，最后用 resetlog 方式打开数据库，并重建 redolog

```
RMAN> RUN
{
    ALLOCATE AUXILIARY CHANNEL aux1 DEVICE TYPE DISK;
    duplicate target database to crm nofilenamecheck;
    RELEASE CHANNEL aux1;
}
```

**5．重建临时表空间，立即备份数据库**

SQL> alter tablespace temp add tempfile '/u05/oracle/oradata/crm/temp01.dbf' size 10M;

**附：Duplicate 复制命令的一些高级用法：**

**1．NOFILENAMECHECK**

异机恢复需要指定 NOFILENAMECHECK，可以跳过文件名检测，避免 oracle 错误的领会你的操作意图。

**2．指定 PFILE**

如果辅助实例不是用 SPFILE 启动，需要指定 PFILE 参数，PFILE 文件必须放在运行 RMAN 的客户端上。

**3．跳过不需要复制的表空间**

要跳过只读表空间，在执行 duplicate 命令时指定 SKIP READONLY 子句即可。

要跳过离线表空间，在执行 duplicate 命令之前将其置为 OFFLINE NORMAL 状态即可，RMAN 复制时会跳过只读或离线表空间的数据文件。

要跳过正常的表空间，指定 SKIP TABLESPACE ts_name, …

**4．恢复到以前的时间点**

要恢复到以前的时间点，在 duplicate 命令中指定 UNTIL TIME 'SYSDATE-1'

**5．创建过程中重命名控制文件**

可以使用 restore controlfile from …;或者 dbms_backup_restore 包恢复出控件文件，然后在初始化参数文件 PFILE 中进行正确的设置。

**6．创建过程中重命名在线日志文件**

1．执行 duplicate 命名时指定 LOGFILE 子句，手工设置 redo logs 文件名。

例：

```
LOGFILE
  GROUP 1 ('?/oradata/newdb/redo01_1.log',
```

'?/oradata/newdb/redo01_2. log') SIZE 200K,
)

2．副本数据库初始化参数文件 PFILE 中设置 LOG_FILE_NAME_CONVERT
LOG_FILE_NAME_CONVERT = 'string1' , 'string2' , 'string3' , 'string4' , ...;
将 strin1 替换为 string2，string3 替换为 string4

**7．创建过程中重命名数据文件，或改变数据库文件的新的路径**

1．在初始化参数中重命名：
DB_FILE_NAME_CONVERT=’xxxx’,’yyyy’

2．在 Duplicate Duplicate Duplicate 命令中重命名数据文件：
duplicate target database to newdb
DB_FILE_NAME_CONVERT=('xxxx', 'yyyyy ');

3．使用 SET NEWNAME 命令重命名数据文件
SET NEWNAME FOR DATAFILE 1 TO '/xxx/xxx.dbf';

4．使用 CONFIGURE AUXNAME 命令重命名数据文件
CONFIGURE AUXNAME FOR DATAFILE 1 TO '/xxxx/xxx.dbf';

5．重命名临时文件路径
SET NEWNAME FOR TEMPFILE '/xxx/xxx.dbf' TO '/yyy/xxx.dbf';

**例：一个比较复杂的复制命令如下：**

```
RMAN> RUN
{
    allocate auxiliary channel newdb1 device type sbt;
    duplicate target database to newdb
    DB_FILE_NAME_CONVERT=('/h1/oracle/dbs/trgt/','/h2/oracle/oradata/newdb/')
     UNTIL TIME 'SYSDATE-1'     # specifies incomplete recovery
     SKIP TABLESPACE cmwlite, drsys, example       # skip desired tablespaces
     PFILE = ?/dbs/initNEWDB.ora
    LOGFILE
        GROUP 1 ('?/oradata/newdb/redo01_1.log',
                 '?/oradata/newdb/redo01_2. log') SIZE 200K,
        GROUP 2 ('?/oradata/newdb/redo02_1. log'',
                 '?/oradata/newdb/redo02_2. log'') SIZE 200K
        GROUP 3 ('?/oradata/newdb/redo03_1. log'',
                 '?/oradata/newdb/redo03_2. log'') SIZE 200K REUSE;
}
```

### 4.7.3 表空间时间点恢复(TSPITR)

用户可能错误的删除了几个表，而且还截断了几个表，如果进行整库恢复可能代价比较

高，这时，我们可以执行表空间时间点恢复（tablespace point-in-time recovery， TSPITR）。为了掌握如何执行 TSPITR，应该理解下面这些术语：

**辅助实例(auxiliary instance)：** 这是我们创建的临时实例，RMAN 使用这个实例执行 TSPITR。完成 TSPITR 后，这个实例可以删除。

**辅助集(auxiliary set)：** 辅助实例上的文件集。包括控制文件、回滚段或者重做段的表空间、系统表空间、联机日志文件等，还可以选择辅助实例的临时表空间。

**恢复集(recovery set)：** 要执行 TSPITR 的表空间/数据文件集

**目标数据库：** 实际执行 TSPITR 的数据库

TSPITR 的本质是，RMAN 将要进行时间点恢复的表空间按照设定的条件恢复到辅助实例上，然后再从辅助实例将表空间传到目标数据库，从而避免整个数据库的恢复，减少恢复时间，而且这期间其他表空间仍然可用。( 用户管理的备份与恢复也可以使用 TSPITR)

检查TSPITR之后丢失的对象，被丢失对象信息的获取：

```
SQL> Select owner, name
        From TS_PITR_OBJECTS_TO_BE_DROPPED
        Where tablespace_name = 'USER01' And creation_time >
            to_date( '2004-09-27 11:25:21', 'yyyy-mm-dd hh24:mi:ss' );
```

**进行 TSPITR 的基本步骤是（与有些书上的步骤略有不同）：**

(0) 检查需要恢复的表空间的数据文件备份和控制文件备份是否存在

(1) 验证表空间的可传输性

(2) 准备辅助实例

(3) 执行实际的 TSPITR

(4) 在目标数据库上执行 TSPITR 后的操作。

**1．验证表空间的可传输性**

RMAN 使用 Oracle 可传送的表空间特性来执行 TSPITR，因此表空间本身必须是可传送的。许多因素都导致表空间不可传送。

**TSPITR 限制**

不能还原包含 SYS 用户对象

不能执行 TSPITR 恢复具有复制主表的表空间

不支持使用快照日志的表空间

不能还原含有回滚段的表空间

不能还原含有分区对象（该对象的分区跨越多个表空间）的表空间

表空间中不能含有 VARRAY、嵌套表、外部表对象

TSPITR 不能用于恢复删除的表空间。

可以通过 TS_PITR_CHECK 视图来判断表空间是否可传送。如果表空间不可传送，那么执行 TSPITR 时就会失败。下面是查询 USERS 表空间是否可传送的语句：

```
SQL> SELECT OBJ1_NAME "Object Owner",obj1_name "Object Name",
        obj1_type "Object Type",ts1_name "Tablespace Name",reason
        FROM TS_PITR_CHECK  WHERE ts1_name='USERS';
```

如果未选定行，说明 USERS 表空间可传输。

**2．准备 TSPITR 的辅助实例 (AUXILIARY INSTANCE)**

启动 TSPITR 之前，需要准备辅助实例。这是一个不需要 RMAN 参与的手工过程。执行下面的步骤来创建辅助实例：

- 创建口令文件
- 创建辅助实例的参数文件：
  编辑辅助实例参数文件的一个简单方法是把目标数据库的参数文件改一下即可。如果目标数据库用的是 SPFILE，那么在 SQLPLUS 中执行 CREATE PFILE FROM SPFILE 就可以生成参数文件。
  **注意：**
  control_files 设置为辅助实例上的文件名
  db_name = 目标实例
  lock_name_space 如果辅助实例与目标数据库同一主机，必须设置
  instance_name service_names 取一新名，如 AUX
  db_file_name_convert 数据文件名的转换
  log_file_name_convert 日志文件名的转换
  注释掉 log_archive_start 参数

- 如果在 Windows NT 上运行 Oracle，需要使用 oradim 来添加数据库服务
- 启动辅助实例，如有必要，配置 tnsnames.ora，测试是否连通。

### 3．执行实际的 TSPITR

例子： 恢复用户误删除的某个重要的表。将实例 MING 的 USERS 表空间恢复到 SCN 是 3818161 的时候。

**启动辅助实例：**

```
C:\>SET ORACLE_SID=AUX1
C:\>SQLPLUS /NOLOG
SQL> CONNECT / AS SYSDBA
SQL> STARTUP NOMOUNT;
```

请注意，SET ORACLE_SID=AUX1 很重要，这样就可以不用配置 TNSNAMES.ORA

**执行实际的 TSPITR**

首先要连到目标数据库和辅助实例。可以联到恢复目录，也可以不连到恢复目录：

```
C:\> SET ORACLE_SID=AUX1
C:\> RMAN TARGET /@MING CATALOG RMAN/RMAN@RECO AUXILIARY /
RMAN> RECOVER TABLESPACE USERS UNTIL SCN 233646;
  …
```

等待完成表空间的恢复。当然也可以用 UNTIL TIME 或者 UNTIL SEQUENCE 子句，例：

```
RMAN> run {
        allocate auxiliary channel c1 device type disk;
        recover tablespace user02 until logseq 9;
}
```

( 这里有些版本有一个 ORACLE 的 BUG，执行到最后可能会报错，需要打补丁 ?? )

**可以改变辅助集/恢复集数据文件的位置和名称：**

set newname for datafile 5 to 'c:\demo\user01.dbf';

configure auxname for datafile 1 to '/backup/xxx.dbf';

在初始化参数文件中用 DB_FILE_NAME_CONVERT 进行转换

**10g 中增加了 auxiliary destination ，更方便**

RMAN> recover tablespace user01 until logseq 19 auxiliary destination 'd:\auxiliary';

**4．执行 TSPITR 后的操作**

首先应该重新连接 RMAN 与目标数据库，并且备份刚恢复的表空间。完成备份后，需要将表空间联机(RMAN 在 TSPITR 后使表空间脱机)。最后，我们还要关闭或删除辅助数据库。

### 4.7.4 块媒体恢复 Block Media Recovery (BMR)

产生块损坏的原因一般是间断或随机的 I/O 错误或者是内存的块错误。要恢复的坏块信息可以从报警与跟踪文件，表与索引的分析，DBV 工具或第三方媒体管理工具以及具体的查询语句中获得。

1. DBV 工具 dbv file=EYGLE.DBF blocksize=8192

2. RMAN> backup validate datafile 4; -- 或者 BACKUP VALIDATE DATABASE

备份的坏块信息保存在 V$BACKUP_CORRUPTION、V$COPY_CORRUPTION 和 V$DATABASE_BLOCK_CORRUPTION 视图中。

**V$BACKUP_CORRUPTION** 显示历史讹误的视图

**V$DATABASE_BLOCK_CORRUPTION** 显示当前数据块讹误的视图。

一旦修正了数据库的块讹误，就需要重新运行 BACKUP VALIDATE DATABASE 命令，然后确认 V$DATABASE_BLOCK_CORRUPTION 中没有其他讹误。

RMAN 可以备份包含损坏数据块的数据文件，通过设置 set maxcorrupt 可以跳过指定个数的坏块来避免备份失败。

RMAN> set maxcorrupt for datafile 1 to 0;

在 Oracle9i 中可以用 RMAN 来执行块级的恢复，而且恢复期间数据文件可以是联机状态。RMAN 通过 Block Media Recovery（简称 BMR）来执行块级恢复操作。假设我们在查询一个 Oracle 表的时候接收到下面的错误：

ERROR 位于第 1 行:

ORA-01578: ORACLE 数据块损坏（文件号 5，块号 97）

ORA-01110: 数据文件 5: 'E:\xxxx.dbf'

那么我们就可以在 RMAN 中用 BLOCKRECOVER 命令来修复：

RMAN> BLOCKRECOVER DATAFILE 5 BLOCK 97;

启动 blockrecover 于 03-9 月 -06

正在启动全部恢复目录的 resync

完成全部 resync

使用通道 ORA_DISK_1

通道 ORA_DISK_1: 正在从数据文件副本 E:\xxxx.bak 恢复块

正在开始介质的恢复
完成介质的恢复
完成 blockrecover 于 03-9 月 -06

**恢复 V$DATABASE_BLOCK_CORRUPTION 视图中列出的坏块：**

RMAN> blockrecover corruption list [ restore until time 'sysdate - 10' ];

**恢复指定坏块：**

RMAN> blockrecover datafile 2 block 12,13 datafile 9 block 19;
RMAN> blockrecover tablespace system dba 44404,44405 from tag "weekly_backup";
RMAN> blockrecover tablespace system dba 44404,44405 restore until time 'sysdate-2';

### 4.7.5 使用恢复目录恢复前一个对应物

在不完全恢复完成之后，通常需要使用 resetlogs 选项来打开数据库。resetlogs 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。数据库的逻辑生存期也被称为一个对应物(incarnation)。每次使用 resetlogs 选项来打开数据库后都会创建一个新的数据库对应物。

使用 RMAN 可以进行穿越 resetlogs 的恢复，即可以恢复到前一个对应物。10g 版本增强了这方面的功能。

**(1) 使用恢复目录的情况**

使用恢复目录的情况下，恢复前一个对应物是非常简单的。

**首先，用 list incarnation;来显示有哪些对应物：**

RMAN> list incarnation;

数据库 Incarnations 列表

| DB 关键字 | Inc 关键字 | DB 名 | DB ID | CUR | 重置 SCN | 重置时间 |
|---|---|---|---|---|---|---|
| 1 | 2 | MING | 2600315304 | NO | 1 | 20-4 月 -06 |
| 1 | 854 | MING | 2600315304 | NO | 3407561 | 19-8 月 -06 |
| 1 | 4368 | MING | 2600315304 | NO | 3794049 | 29-8 月 -06 |
| 1 | 4437 | MING | 2600315304 | NO | 3794728 | 29-8 月 -06 |
| 1 | 4639 | MING | 2600315304 | YES | 3794935 | 29-8 月 -06 |

**接下来，决定恢复到哪个对应物**

比如，决定恢复到上面的 4437，

方法：在 NOMOUNT 状态下用下面的命令：

RMAN> reset database to incarnation 4437;

下面是具体的恢复步骤的一个演示（输出略去）：

RMAN> shutdown immediate;
RMAN> startup nomount;
RMAN> reset database to incarnation 4437;
RMAN> restore controlfile;
RMAN> alter database mount;
RMAN> restore database until scn 3794934;

RMAN> recover database until scn 3794934;

RMAN> alter database open resetlogs;

上面的例子中，我们最大可恢复到的 SCN 就是 3794934，因为 4437 的下一个对应物的重置 SCN 是 3794935。

**(2) 不使用恢复目录的情况**

不使用恢复目录的情况下，要恢复前一个对应物就必须要能恢复到前一对应物的控制文件。不使用恢复目录，我们就无法使用 RESET DATABASE TO incarnation_num 命令。

**首先看看有哪些对应物：**

RMAN> list incarnation;

数据库 Incarnations 列表

| DB 关键字 | Inc 关键字 | DB 名 | DB ID | CUR | 重置 SCN | 重置时间 |
|---|---|---|---|---|---|---|
| 1 | 1 | MING | 2600315304 | NO | 3501920 | 20-8 月 -06 |
| 2 | 2 | MING | 2600315304 | NO | 3785052 | 29-8 月 -06 |
| 3 | 3 | MING | 2600315304 | NO | 3794049 | 29-8 月 -06 |
| 4 | 4 | MING | 2600315304 | NO | 3794728 | 29-8 月 -06 |
| 5 | 5 | MING | 2600315304 | NO | 3794935 | 29-8 月 -06 |
| 6 | 6 | MING | 2600315304 | YES | 3976301 | 03-9 月 -06 |

假设我们要恢复到 5，6 之间，那么就要看看有没有控制文件的备份的 SCN 在 3794935 到 3976301。我们使用下面的命令：

RMAN> LIST BACKUP OF CONTROLFILE BY FILE;

控制文件备份列表

| CF Ckp SCN | Ckp 时间 | BS Key | S | 段数 | 副本数 | 标记 |
|---|---|---|---|---|---|---|
| 3976095 | 03-9 月 -06 | 34 | A | 2 | 1 | TAG20060903T170136 |
| 3959239 | 03-9 月 -06 | 30 | A | 1 | 1 | |
| 3959170 | 03-9 月 -06 | 29 | A | 1 | 1 | |
| 3958983 | 03-9 月 -06 | 28 | A | 1 | 1 | |
| 3958937 | 03-9 月 -06 | 27 | A | 1 | 1 | |
| 3816575 | 30-8 月 -06 | 11 | A | 2 | 1 | TAG20060830T064447 |

可以看出，上面的列表中，控制文件都是可以用的。接下来找到一个具体的含有控制文件的备份：

RMAN> LIST BACKUP OF CONTROLFILE;

备份集列表

==========

……

段名:E:\TEST\C-2600315304-20060830-07

控制文件包括: Ckp SCN: 3818213 Ckp 时间:30-8 月 -06

……

假设我们使用 E:\TEST\C-2600315304-20060830-07，那么接下来我们需要用 DBMS_BACKUP_RESTORE 包来从这个文件中恢复控制文件。（在 SQLPLUS 中运行）

```
SQL>SHUTDOWN IMMEDIATE;
SQL>STARTUP NOMOUNT;
SQL>DECLARE
      DEVTYPE VARCHAR2(256);
      DONE BOOLEAN;
BEGIN
    DEVTYPE:=DBMS_BACKUP_RESTORE.deviceallocate(NULL);
    DBMS_BACKUP_RESTORE.restoresetdatafile;
    DBMS_BACKUP_RESTORE.restorecontrolfileto('/xxx/CONTROL01.CTL');
    DBMS_BACKUP_RESTORE.restorebackuppiece('/xxx/C-2600315304-20060830-07',D
ONE=>done);
    DBMS_BACKUP_RESTORE.restoresetdatafile;
    DBMS_BACKUP_RESTORE.restorecontrolfileto('/xxx/CONTROL02.CTL');
    DBMS_BACKUP_RESTORE.restorebackuppiece('/xxx/C-2600315304-20060830-07',D
    ONE=>done);
    DBMS_BACKUP_RESTORE.restoresetdatafile;
    DBMS_BACKUP_RESTORE.restorecontrolfileto('/xxx/CONTROL03.CTL');
    DBMS_BACKUP_RESTORE.restorebackuppiece('/xxx/C-2600315304-20060830-07',D
    ONE=>done);
    DBMS_BACKUP_RESTORE.devicedeallocate(NULL);
END;
/
SQL>ALTER DATABASE MOUNT;
```

重新联到 RMAN 后 再来运行一下 LIST INCARNATION;

```
$ RMAN TARGET /@MING NOCATALOG;
RMAN> LIST INCARNATION;
```

数据库 Incarnations 列表

| DB 关键字 | Inc 关键字 | DB 名 | DB ID | CUR | 重置 SCN | 重置时间 |
|---|---|---|---|---|---|---|
| 1 | 1 | MING | 2600315304 | NO | 3501920 | 20-8 月 -06 |
| 2 | 2 | MING | 2600315304 | NO | 3785052 | 29-8 月 -06 |
| 3 | 3 | MING | 2600315304 | NO | 3794049 | 29-8 月 -06 |
| 4 | 4 | MING | 2600315304 | NO | 3794728 | 29-8 月 -06 |
| 5 | 5 | MING | 2600315304 | YES | 3794935 | 29-8 月 -06 |

可以看出来，我们已经恢复到想要的对应物上了。接下来：

```
RMAN> RESTORE DATABASE;
RMAN> RECOVER DATABASE;
RMAN> ALTER DATABASE OPEN RESETLOGS;
```

### 4.7.6 RMAN增量备份

通过增量备份，RMAN 允许用户只备份上次增量备份操作以来被修改过的数据块。N 级别增量备份备份从最近的 N 级别或者更小级别以来的所有更改过的数据块内容。0 级备份与全备份的不同就是 0 级备份可以作为其它增量备份的基础备份而全备份是不可以的。

**增量备份的优点：** 节约备份时间　节省备份磁带或者磁盘空间　降低网络带宽要求

**增量备份的缺点：** 恢复时间可能要长。

增量备份分为两种，一种是累积增量备份，一种是非累积增量备份。

**差异增量备份(Differential Incremental Backup)**

差异备份是默认的增量备份类型，备份上一次在同级或者更低级别进行备份以来所有有变化的数据块。

RMAN>backup incremental level n (incremental) database;

**累积增量备份(Cumulative Incremental Backup)**

包括上一次低级备份以来所有有变化的数据块。累计增量备份增加了备份的时间，但是因为恢复的时候，需要从更少的备份集中恢复数据，所以，为了减少恢复的时候，累计增量备份将比差异增量备份更有效。

RMAN>backup incremental level n (cumulative) database;

**9i ：** 不管怎么样增量备份，还是需要比较数据库中全部的数据块，这个过程其实是一个漫长的过程，而且由于增量备份形成多个不同的备份集，使得恢复变的更加不可靠而且速度慢，所以增量备份在版本 9 中仍然是鸡肋，除非是很大型的数据仓库系统，没有必要选择增量备份。

**10g ：** 在增量备份上做了很大的改进，可以使增量备份变成真正意义的增量，因为通过特有的增量日志，使得 RMAN 没有必要去比较数据库的每一个数据块，当然，代价就是日志的 IO 与磁盘空间付出。另外，10g 通过备份的合并，使增量备份的结果可以合并在一起，而完全的减少了恢复时间。

**1． 基本备份（0 级备份）**

Level 0 是增量备份策略的基础(注意与 Full 备份区别)。执行增量备份操作时，首先需要的是基本备份(incremental base backup)，增量备份必须有一个基本备份。基本备份的增量级别是 0，如果没有 0 级备份，那么其它级别的备份会自动转成基本备份。

**0 级备份举例：**

```
RMAN> run {
  2> allocate channel dev1 type disk;
  3> backup
  4> incremental level 0
  5> filesperset 4   #定义每个 backupset 的最大文件数
  6> format "/oracle/backups/sunday_level0_%t"
  7> (database);
  8> release channel dev1;
```

```
    9> }
```

使用 LIST 语句查看，数据库备份集的列表显示中， "type" 将显示 "Incremental"，"LV" 列将显示"0" 。

**2． N 级备份**

```
RMAN> run {
    2> allocate channel dev1 type disk;
    3> backup
    4> incremental level 1/2/3/4
    5> filesperset 4   #定义每个 backupset 的最大文件数
    6> format "/oracle/backups/sunday_level0_%t"
    7> (database);
    8> release channel dev1;
    9> }
```

**3． N 级备份的规划**

根据业务需求、数据量、恢复所需要的时间等方面考虑，制定合理的备份计划。

例： 一个典型的增量备份案例如下：

星期天晚上 - level 0 backup performed
星期一晚上 - level 2 backup performed
星期二晚上 - level 2 backup performed
星期三晚上 - level 1 backup performed
星期四晚上 - level 2 backup performed
星期五晚上 - level 1 backup performed
星期六晚上 - level 2 backup performed
星期天晚上 - level 0 backup performed

**4． 增量备份的恢复**

同普通的完全恢复： restore recover

RMAN 自动确定和使用需要的备份集。

### 4.7.7 RMAN备份的优化

可以在 RMAN 配置中设置备份的优化，如

RMAN> CONFIGURE BACKUP OPTIMIZATION ON;

如果优化设置打开，将对备份的数据文件、归档日志或备份集运行一个优化算法。

RMAN 备份操作主要是完成以下三个步骤

1、从磁盘上读取数据
2、在内存中处理数据块
3、写入数据到磁盘或磁带

以上的读写操作可以同步或异步的完成，在同步 I/O 操作中，一个时间只允许有一个 IO 操作，但是在异步 I/O 操作中，一个时间允许有多个 IO 操作。因此，备份与恢复的调优主要集中在以下几个方面：

1、 提高同步或异步 I/O 操作能力

在支持异步操作的操作系统上，可以通过设置 TAPE_AYSNCH_IO,DISK_ASYNCH_IO 和 BACKUP_TYPE_IO_SLAVES 来支持异步操作，提高写的能力。

2、 提高磁盘读能力

可以在 backup 命令后通过设置 DISKRATIO 来保证从多个磁盘上读取数据，保证连续的数据流。

3、 正确设置缓冲区与参数值

设置 LARGE_POOL_SIZE，使备份可以使用连续的缓冲池，通过设置 DB_FIL_DIRECT_IO_COUNT 可以提高缓冲区的利用。如果使用磁带备份，还可以设置 BACKUP_TYPE_IO_SLAVES 来提高磁带的写能力。

4、 采用并行备份

开辟多个通道，可以实现并行备份与恢复

### 4.7.8 DBMS_BACKUP_RESTORE 包

这个包是 RMAN 备份与恢复的核心，在调试模式下，RMAN 会输出它调用的每一条命令：

C:\> rman debug target /@tlgaxz catalog rman/rman@reco trace=debug.txt

**常用命令：**

| | |
|---|---|
| restoreSetDataFile | 指示还原操作的开始（但是不会执行实际的还原操作） |
| restoreSetDataFileTo | 定义要还原的数据文件和该文件的还原位置 |
| restoreControlFileTo | 定义控制文件的还原位置 |
| restoreSpfileTo | 定义要还原的 spfile 的位置 |
| restorebackuppiece | 执行实际的还原操作，这个函数的一个参数是备份文件名 |
| devicedeallocate | 释放 deviceallocate 函数所分配的设备 |
| deviceallocate | 分配用于连续 I/O 的设备 |
| applySetDataFile | 指示增量还原操作的开始 |
| applyDataFileTo | 定义数据文件的增量还原位置 |
| applybackuppiece | 执行实际的还原操作 |
| resetoreSetArchivelog | 指示归档日志还原操作的开始 |
| restoreArchivelog | 定义要还原的归档的重做日志序列和线程 |

**例：使用 dbms_backup_restore 包恢复服务器参数文件：**

在一些不常见的情况下，我们可能需要直接使用 dbms_backup_restore 包来恢复 spfile。当然这个包也可以用来恢复其它数据，是常规办法都没有用的时候的一个利器。这个包可以在数据库 NOMOUNT 状态下使用。假设我们有一个自动备份文件

C-2600315304-20060829-02，我们需要从这里恢复数据，那么可以通过执行下面的脚本来完成：

```
SQL>DECLARE
   DEVTYPE VARCHAR2(256);
   DONE BOOLEAN;
BEGIN
      DEVTYPE:=DBMS_BACKUP_RESTORE.deviceallocate(NULL);
      DBMS_BACKUP_RESTORE.restoresetdatafile;
      DBMS_BACKUP_RESTORE.restorespfileto('d:\spfile.ora');
      DBMS_BACKUP_RESTORE.restorebackuppiece('d:\C-2600315304-20060829-00',
DONE=>done);
      DBMS_BACKUP_RESTORE.devicedeallocate(NULL);
END;
/
```

# . Flashback

Flashback 在开发环境（有时生产环境的特殊情况下）是很有用的一个工具，

## 5.1 9i Flashback 简介

### 5.1.1 原理

当数据 update 或 delete 时，原来的数据会保存在 undo 表空间中，保存的最少时间是 UNDO_RETENTION。实际的保存时间与 undo 表空间的大小和数据更改的繁忙程度相关。UNDO_RETENTION 的参数（单位为秒）指定 Oracle 保存用于 flashback 查询的 undo 映像的时间。一般你可以将这个值设为一整天（864000 秒），这样你就能看到前一天全天的映像。当然，你的在线 undo 日志必须足够大，大到足以能保存一整天的 undo 日志数据，对于繁忙的 Oracle 系统，这个数值可以达到很大。

### 5.1.2 一些限制

- Ø 服务器必须配置成使用自动 undo 管理。
- Ø 在使用 Flashback 查询时不能使用 DDL 或者 DML。
- Ø Flashback 不取消 DDL 操作，例如 DROP 命令。

数据库管理员做一些必要的设置之后，一般用户才能使用 Flashback 查询功能：

```
SQL> ALTER SYSTEM SET UNDO_MANAGEMENT=AUTO
SQL> ALTER SYSTEM SET UNDO_RETENTION=86400
SQL> GRANT EXECUTE ON DBMS_FLASHBACK TO USERNAME;
```

### 5.1.3 获得SCN或时间点

在 Flashback 时，可以尝试多个 SCN，获取最佳值。

如果能得知具体时间，那么可以获得准确的数据闪回。

```
SQL> alter session set nls_date_format='YYYY-MM-DD HH24:MI:SS';
SQL> select sysdate from v$database;
```

捕捉提交的 SCN：  // 不知为什么，两种方法获得的 SCN 不一样

```
SQL> select dbms_flashback.get_system_change_number scn from dual;
SQL> select max(ktuxescnw * power(2,32) + ktuxescnb) SCN from x$ktuxe;
```

### 5.1.4 启用或禁用flashback查询

使用系统改变数（SCN）或者真实时间来指定 flashback 的时间点来获取数据映象。

**方法一：**

```
SQL> select * from [TABLE] as of scn 129292;
```

SQL> select * from [TABLE] as of timestamp to_timestamp('时间', '时间格式');

SQL> select * from saflog as of timestamp to_timestamp('2007-12-18 08:40:00', 'YYYY-MM-DD HH24:MI:SS');

**方法二：**

启用：

SQL> exec dbms_flashback.enable_at_system_change_number(112112);

SQL> exec dbms_Flashback.enable_at_time('28-AUG-02 11:00:00');

启用后看到的只是闪回的结果，实际上并未恢复数据。且闪回状态下不能做 DML 操作。可以先恢复到一个临时表中。见示例。

禁用：

SQL> execute dbms_flashback.disable();

### 5.1.5 示例：

```
declare
cursor c1 is select * from scott.emp_temp;
r_c1 scott.e%rowtype;
begin
loop
dbms_flashback.enable_at_system_change_number(49570);
if c1%isopen=false then open c1;end if;
fetch c1 into r_c1;
dbms_flashback.disable();
exit when c1%notfound;
update scott.emp_temp set sal=r_c1.sal where empno=r_c1.empno;
commit;
end loop;
exec dbms_flashback.disable();
close c1;
end;
/
```

## 5.2 10g Flashback 的增强

10g 的 Flashback 进行了增强和修改，通过回闪，用户可以完成许多不可能恢复的工作，目前 oracle10g 的回闪包括以下特性；

**1〉 oracle falshback Database.**

该特性允许 oracle 通过 Flashback database 语句，将数据库会滚到前一个时间点或者 scn 上，而不需要作时间点的恢复工作！

**2〉oracle falshback table.**

该特性允许 oracle 通过 flashback table 语句，将表会滚到前一个时间点或者 scn 上。

**3〉oracle falshback drop.**

该特性允许 oracle 把恢复 drop 掉的 table 或者索引。

**4〉oracle falshback version query.**

该特性可以得到特定的表在某一个时间段内的任何修改记录！

**5〉oracle falshback transaction query**

该特性可以限制用户在某一个事务级别上检查数据库的修改操作，适用于诊断问题、分析性能、审计事务。

(未完成，以后再更新吧)

# 6. LogMiner

## 6.1 LogMiner 的用途

Oracle LogMiner 是 Oracle 公司从产品 8i 以后提供的一个实际非常有用的分析工具，使用该工具可以轻松获得 Oracle 重作日志文件（归档日志文件）中的具体内容，特别是，该工具可以分析出所有对于数据库操作的 DML（insert、update、delete 等）语句，9i 后可以分析 DDL 语句，另外还可分析得到一些必要的回滚 SQL 语句。其中一个最重要的用途就是不用全部恢复数据库就可以恢复数据库的某个变化。该工具特别适用于调试、审计或者回退某个特定的事务。

LogMiner 工具即可以用来分析在线，也可以用来分析离线日志文件，即可以分析本身自己数据库的重作日志文件，也可以用来分析其他数据库的重作日志文件。

总的说来，LogMiner 工具的主要用途有：

1. 跟踪数据库的变化：可以离线的跟踪数据库的变化，而不会影响在线系统的性能。
2. 回退数据库的变化：回退特定的变化数据，减少 point-in-time recovery 的执行。
3. 优化和扩容计划：可通过分析日志文件中的数据以分析数据增长模式。
4. 确定数据库的逻辑损坏时间：准确定位操作执行的时间和 SCN ==> 基于时间和 SCN 的恢复
5. 确定事务级要执行的精细逻辑恢复操作 //取得相应的 UNDO 操作
6. 执行后续审计 //DML DDL DCL 执行时间、用户

**注意事项：**

1. LogMiner 可以帮你确定在某段时间所发的各种 DML，DDL 操作的具体时间和 SCN 号，它所依据的是归档日志文件及联机日志文件。
2. 它只能在 Oracle8i 及以后的版本中使用，不过它可以分析 Oracle8 的日志。
3. Oracle8i 只能用于分析 DML 操作，到 Oracle9i 则可以分析 DDL 操作了。
4. LogMiner 不支持索引组织表、Long、LOB 及集合类型。
5. 不支持 MTS 的环境
6. LogMiner 必须使用被分析数据库实例产生的字典文件，而不是安装 LogMiner 的数据库产生的字典文件，另外必须保证安装 LogMiner 数据库的字符集和被分析数据库的字符集相同。
7. 源数据库(Source Database)平台必须和分析数据库(Mining Database)平台一样

## 6.2 安装 LogMiner

要安装 LogMiner 工具，必须首先要运行下面这样两个脚本：

l. $ORACLE_HOME/rdbms/admin/dbmslm.sql

2. $ORACLE_HOME/rdbms/admin/dbmslmd.sql

这两个脚本必须均以 SYS 用户身份运行。其中第一个脚本用来创建

DBMS_LOGMNR 包，该包用来分析日志文件。第二个脚本用来创建 DBMS_LOGMNR_D 包，该包用来创建数据字典文件。

## 6.3 基本对象

**Source Database:** 日志所属的数据库

**Mining Database:** 执行 LogMiner 操作要使用的数据库，相同硬件平台，相同字符集，版本不低于 Source Database

**LogMiner 字典:** 将内部对象 ID 号和数据类型转换为对象名和外部数据格式，在 Source Database 上生成，有三种方式：

1. 使用源数据库数据字典

```
// 表结构无变化  S-DB 必须 OPEN，只能跟踪 DML 不能为 DDL
SQL> exec DBMS_LOGMNR.START_LOGMNR(
        OPTIONS=>DBMS_LOGMNR.DICT_FROM_ONLINE_CATALOG );
```

2. 摘取 LogMiner 字典到重做日志

```
// Source-DB 必须 OPEN，  Archivelog 模式
SQL> exec DBMS_LOGMNR_D.BUILD(
        OPTIONS=>DBMS_LOGMNR_D.STORE_IN_REDO_LOGS );
```

3. 摘取 LogMiner 字典到字典文件

配置字典文件所在目录:  静态参数: UTL_FILE_DIR

建立字典文件:

```
SQL> exec dbms_logmnr_d.build (
        dictionary_filename => 'logminer.dat',
        dictionary_location => 'F:\0_WorkSpace\1_Database\1_Oracle\Log_Miner' );
SQL> exec dbms_logmnr_d.build ('dict.ora', '/xxx',
        OPTIONS=>DBMS_LOGMNR_D.STORE_IN_FLAT_FILE);
```

**与 LogMiner 相关的数据字典：**

| | |
|---|---|
| v$loghist | 显示历史日志文件的一些信息 |
| v$logmnr_dictionary | 因 logmnr 可以有多个字典文件，显示字典文件信息 |
| v$logmnr_parameters | 显示 logmnr 的参数 |
| v$logmnr_logs | 显示用于分析的日志列表信息 |
| v$logmnr_contents | LogMiner 结果 |

## 6.4 使用 LogMiner 进行分析

### 6.4.1 设定用于LogMiner分析的日志文件存放的位置

设置 UTL_FILE_DIR，需要重启数据库。

设置 utl_file_dir=*  表示你能操作任何目录

在 initsid.ora 文件中加入 utl_file_dir 参数或者：

```
SQL> alter system set utl_file_dir='d:' scope=spfile;
```

```
SQL> shutdown immediate
SQL> starup
```

### 6.4.2 生成数据字典文件

```
SQL> BEGIN
  2      dbms_logmnr_d.build(
  3         dictionary_filename => 'logminer_dict.dat',
  4         dictionary_location => '/u01/arch'
  5      );
  6  END;
  7 /
```

dictionary_location 指的是 Logminer 数据字典文件存放的位置，它必须匹配 utl_file_dir 的设定。

dictionary_filename 指的是放于存放位置的字典文件的名字，名字可以任意取。

### 6.4.3 建立日志分析表

建立日志分析表数据库必须在 mount 或 nomount 状态

建立日志分析表,使用 dbms_logmnr.add_logfile()

```
SQL> BEGIN
  2      dbms_logmnr.add_logfile(
  3      options => dbms_logmnr.new,
  4      logfilename => '/u01/arch/arc_ctc_0503.arc'
  5      );
  6  END;
  7 /
```

其中的 options 有三种取值，

dbms_logmnr.new  用于建一个日志分析表

dbms_logmnr.addfile  用于加入用于分析的的日志文件

dbms_logmnr.removefile 用于移出用于分析的日志文件

### 6.4.4 添加用于分析的日志文件

```
SQL> BEGIN
  2      dbms_logmnr.add_logfile(
  3      options => dbms_logmnr.addfile,
  4      logfilename => '/u01/arch/arc_ctc_0504.arc'
  5      );
  6  END;
  7 /
```

可以用以下方法查询分析的日志文件包含的 scn 范围和日期范围：

```
SQL> select low_time,high_time,low_scn,next_scn from v$logmnr_logs;
```

### 6.4.5 启动 LogMiner 进行分析

```
SQL> BEGIN
  2      dbms_logmnr.start_logmnr(
  3      dictfilename => '/u01/arch/logminer_dict.dat',
  4      starttime => to_date('20030501 12:15:00','yyyymmdd hh24:mi:ss'),
  5      endtime => to_date('20030501 15:40:30','yyyymmdd hh24:mi:ss')
  6      );
  7  END;
  8 /
```

即分析 2003 年 5 月 1 日这天 12:15 至 15:40 这段时间，并把分析结果放到数据字典中以用于查询。

还有两个参数 startscn (起始 SCN 号) 及 endscn (终止 SCN) 号。

### 6.4.6 查看日志分析的结果

**查看 DML 操作:**

```
SELECT operation, sql_redo, sql_undo
    FROM V$logmnr_contents WHERE seg_name = 'QIUYB';

OPERATION       SQL_REDO                        SQL_UNDO
---------       --------------------------      --------------------------
INSERT          inser into qiuyb.qiuyb ...      delete from qiuyb.qiuyb...
```

其中 operation 指的是操作，sql_redo 指的是实际的操作，sql_undo 指的是用于取消的相反的操作。

**查看 DDL 操作:**

```
SELECT timstamp, sql_redo   FROM v$logmnr_contents
            WHERE upper(sql_redo) like '%TRUNCATE%';
```

### 6.4.7结束LogMiner的分析

```
SQL>BEGIN
  2      dbms_logmnr.end_logmnr();
  3    end;
  4 /
```