【体系结构】有关 Oracle SCN 知识点的整理

1.1 BLOG 文档结构图

1.2 前言部分

1.2.1 导读和注意事项

各位技术爱好者，看完本文后，你可以掌握如下的技能，也可以学到一些其它你所不知道的知识，~O(∩_∩)O~：

① Oracle 中的 SCN 是什么？（重点）

② 如何查询 SCN?（重点）

③ SCN 有哪些分类？（重点）

④ SCN 和系统恢复的关系？（重点）

④ 实例恢复和介质恢复的区别是什么？RAC 中的实例恢复是什么样的？（重点）

⑥ SCN 和时间的转换

⑦ SMON_SCN_TIME 系统表的认识

⑧ 不完全恢复的一些分类及其写法

Tips：

① 本文在 itpub（http://blog.itpub.net/26736162）、博客园（http://www.cnblogs.com/lhrbest）和微信公众号（xiaomaimiaolhr）上有同步更新。

② 文章中用到的所有代码、相关软件、相关资料及本文的 pdf 版本都请前往小麦苗的 360 云盘下载，我的 360 云盘地址见：http://blog.itpub.net/26736162/viewspace-1624453/。

③ 若网页文章代码格式有错乱，请尝试以下办法：①使用 360 浏览器，②去博客园地址阅读③下载 pdf 格式的文档来阅读。

④ 在本篇 BLOG 中，代码输出部分一般放在一行一列的表格中。其中，需要特别关注的地方我都用灰色背景和粉红色字体来表示，比如在下边的例子中，thread 1 的最大归档日志号为 33，thread 2 的最大归档日志号为 43 是需要特别关注的地方；而命令一般使用黄色背景和红色字体标注；对代码或代码输出部分的注释一般采用蓝色字体表示。

```
  List of Archived Logs in backup set 11
  Thrd Seq     Low SCN    Low Time            Next SCN   Next Time
  ---- ------- ---------- ------------------- ---------- ---------
  1    32      1621589    2015-05-29 11:09:52 1625242    2015-05-29 11:15:48
  1    33      1625242    2015-05-29 11:15:48 1625293    2015-05-29 11:15:58
  2    42      1613951    2015-05-29 10:41:18 1625245    2015-05-29 11:15:49
  2    43      1625245    2015-05-29 11:15:49 1625253    2015-05-29 11:15:53
[ZHLHRDB1:root]:/>lsvg -o
T_XLHRD_APP1_vg
rootvg
[ZHLHRDB1:root]:/>
00:27:22 SQL> alter tablespace idxtbs read write;
====》2097152*512/1024/1024/1024=1G
```

本文如有错误或不完善的地方请大家多多指正，ITPUB 留言或 QQ 皆可，您的批评指正是我写作的最大动力。

1.2.2 本文简介

由于写书遇到了 SCN 的概念，所以就找了点资料，整理了一下有关 SCN 的一些知识。顺便复习了一下 SCN 和数据库恢复的关系。

1.3 Oracle SCN

1.3.1 简介

SCN(System Change Number，系统改变号)是一个由系统内部维护的序列号。当系统需要更新的时候自动增加，它是系统中维持数据的一致性和顺序恢复的重要标志，是数据库非常重要的一种数据结构。SCN 的最大值是 0xffff.ffffffff。在数据库中 SCN 作为一种时钟机制来标记数据库动作，比如当事务的发生，数据库会用一个 SCN 来标记它。同时这个 SCN 在数据库全局也是唯一的，它随时间的增长而增长除非重建数据库。

在数据库中，SCN 可以说是无处不在，数据文件头，控制文件，数据块头，日志文件等等都标记着 SCN。也正是这样，数据库的一致性维护和 SCN 密切相关。不管是数据的备份，恢复都是离不开 SCN 的。

1.3.2 官方文档

A **system change number (SCN)** is a logical, internal time stamp used by Oracle Database. SCNs order events that occur within the database, which is necessary to satisfy the ACID properties of a transaction. Oracle Database uses SCNs to mark the SCN before which all changes are known to be on disk so that recovery avoids applying unnecessary redo. The database also uses SCNs to mark the point at which no redo exists for a set of data so that recovery can stop.

SCNs occur in a monotonically increasing sequence. Oracle Database can use an SCN like a clock because an observed SCN indicates a logical point in time and repeated observations return equal or greater values. If one event has a lower SCN than another event, then it occurred at an earlier time with respect to the database. Several events may share the same SCN, which means that they occurred at the same time with respect to the database.

Every transaction has an SCN. For example, if a transaction updates a row, then the database records the SCN at which this update occurred. Other modifications in this transaction have the same SCN. When a transaction commits, the database records an SCN for this commit.

Oracle Database increments SCNs in the system global area (SGA). When a transaction

modifies data, the database writes a new SCN to the undo data segment assigned to the transaction. The log writer process then writes the commit record of the transaction immediately to the online redo log. The commit record has the unique SCN of the transaction. Oracle Database also uses SCNs as part of its instance recovery and media recovery mechanisms.

怎么理解这个“SCN（系统变更号）是供 Oracle 数据库使用的一个逻辑的、内部的时间戳”呢？要理解这个先需要理解 Oracle 中的事务(Transaction)和数据一致性(Data Consistency)的概念。

先说说数据一致性的概念。数据一致性指的是数据的可用性。比如说管理一个财务的系统，需要从 A 账户将 100 元转入到 B 账户，正常的操作是从 A 账户减去 100 元，然后给 B 账户加上 100 元，如果这两步操作都正常完成了，那我们可以说完成转账操作之后的数据是一致可用的；但是如果在操作的过程中出了问题，A 账户的 100 元给减掉了，但是 B 账户却没有加上 100 元，这样的情况下产生的结果数据就有问题了，因为部分操作的失败导致了数据的不一致而不可用，在实际中肯定是要避免这种让数据不一致的情况发生的。在 Oracle 数据库中，保证数据一致性的方法就是事务。

事务是一个逻辑的、原子性的作业单元，通常由一个或者是多个 SQL 组成，一个事务里面的所有 SQL 操作要么全部失败回滚(Rollback)，要么就是全部成功提交(Commit)。就像上面转账的例子，为保证数据的一致性，就需要将转账的两步操作放在一个事务里面，这样不管哪个操作失败了，都需要将所有已进行的操作回滚，以保证数据的可用性。进行事务管理是数据库区别于别的文件系统的一个最主要的特征，在数据库中事务最主要的作用就是保证了数据的一致性，每次事务的提交都是将数据库从一种一致性的状态带入到另外一种一致性的状态中，SCN 就是用来对数据库的每个一致状态进行标记的，每当数据库进入到一个新的一致的状态，SCN 就会加 1，也就是每个提交操作之后，SCN 都会增加。也许你会想为什么不直接记录事务提交时候的时间戳呢？这里面主要是涉及了两个问题，一个是时间戳记录的精度有限，再一个就是在分布式系统中记录时间戳会存在系统时钟同步的问题，详细的讨论可以查看 Ordering Events in Oracle。

SCN 在数据库中是一个单一的不断的随着数据库一致性状态的改变而自增的序列。正如一个时间戳代表着时间里面的某一个固定的时刻点一样，每一个 SCN 值也代表着数据库在运行当中的一个一致性的点，大的 SCN 值所对应的事务总是比小 SCN 值的事务发生的更晚。因此把 SCN 说成是 Oracle 数据库的逻辑时间戳是很恰当的。

1.3.3 SCN 的分类

严格来说 SCN 是没有分类的，之所以会有不同类型的 SCN 并不是说这些 SCN 的概念不一样，而是说不同分类的 SCN 代表的意义不一样，**不管什么时候 SCN 所指代的都是数据库的某个一致性的状态。**就像我们给一天中的某个时间点定义上班时间、另外的某个时间点定义成下班时间一样，数据库 Checkpoint 发生点的 SCN 被称为 Checkpoint SCN，仅此而已。

SCN 可以分为 4 类，系统检查点 SCN（System Checkpoint SCN）、文件检查点 SCN（Datafile Checkpoint SCN）、开始 SCN（Start SCN）和结束 SCN（Stop SCN），参考如下表格：

分类	英文	简介	
系统检查点SCN	System Checkpoint SCN	存在于控制文件中，在系统执行Checkpoint后，Oracle会更新当前控制文件中的System Checkpoint SCN。该SCN是全局范围的，当发生文件级别的SCN时，例如将表空间置于只读状态，则不会更新系统检查点SCN。	SELECT CHECKP
文件检查点SCN	Datafile Checkpoint SCN	存在于控制文件中，表示该数据文件最近一次执行检查点操作时的SCN，例如将表空间置为只读、BEGIN BACKUP或将某个数据文件设置为OFFLINE等。	SELECT FILE#,CHECKPO (CHECKPOINT_T FROM V$DATAFI
开始SCN（数据文件头SCN）	Start SCN	存在于各个数据文件头，在执行Checkpoint时，Oracle会更新存放在各个实际的数据文件头的Start SCN（注意绝对不会是控制文件中），这个SCN存在的目的是用于检查数据库启动过程中是否需要做MEDIA RECOVERY（介质恢复）。 在数据库的启动过程中，当System Checkpoint SCN=Datafile Checkpoint SCN=Start SCN的时候，Oracle数据库是可以正常启动的，而不需要做任何的MEDIA RECOVERY。而如果三者当中有一个不同的话，则需要做MEDIA RECOVERY。Oracle在启动过程中首先检查是否需要MEDIA RECOVERY，然后再检查是否需要INSTANCE RECOVERY。	SELECT FILE#, V$DATAFILE_HE
结束SCN	Stop SCN（End SCN）	存在于控制文件中，这个SCN存在的绝对意义主要是用来去验证数据库启动过程中是否需要做INSTANCE RECOVERY（实例恢复）。当End SCN不等于Start SCN的时候，数据库需要做实例恢复。如果数据库异常关闭的话，则End SCN号将为空，则需要做实例恢复。 在数据库正常运行的情况下，对于READ/WRITE的ONLINE数据文件这个SCN号为空，在数据库异常关闭的情况下，结束SCN号也为空，其它情况都存在具体的SCN值。	SELECT FILE#,

1.3.4 查询 4 种 SCN 常用的 SQL 语句

```
col status for a10
select GROUP# ,SEQUENCE#,STATUS,FIRST_CHANGE#,FIRST_TIME from v$log;

SELECT A.FILE#,
      A.NAME,
      (SELECT CHECKPOINT_CHANGE# FROM V$DATABASE) SYSTEM_CKPT_SCN,
      A.CHECKPOINT_CHANGE# DF_CKPT_SCN,
```

```
      A.LAST_CHANGE# END_SCN,
      B.CHECKPOINT_CHANGE# START_SCN,
      B.RECOVER,
      A.STATUS
  FROM V$DATAFILE A, V$DATAFILE_HEADER B
 WHERE A.FILE# = B.FILE#;
SELECT FILE#,ONLINE_STATUS,CHANGE#,ERROR FROM V$RECOVER_FILE;
```

1.3.4.1 文件检查点 SCN (Datafile Checkpoint SCN)

```
SYS@lhrdb>  select file#,checkpoint_change# from v$datafile;

    FILE# CHECKPOINT_CHANGE#
---------- ------------------
         1            9026292
         2            9026292
         3            9026292
         4            9026292
         5            9026292
         6            9026292
         7            9026292

7 rows selected.

SYS@lhrdb>  alter tablespace users read only;

Tablespace altered.

SYS@lhrdb> select file#,checkpoint_change# from v$datafile;

    FILE# CHECKPOINT_CHANGE#
---------- ------------------
         1            9026292
         2            9026292
         3            9026292
         4            9028165
         5            9026292
         6            9026292
         7            9026292

7 rows selected.

SYS@lhrdb> select checkpoint_change# from v$database;

CHECKPOINT_CHANGE#
------------------
           9026292
```

可以看到 4 号文件也就是 users 表空间所属的文件 scn 值和其他文件不一致，且比系统检查点的 scn 要大。

1.3.4.2 Stop SCN

Stop scn 记录在数据文件头上。当数据库处在打开状态时，stop scn 被设成最大值 0xffff.ffffffff。在

数据库正常关闭过程中，stop scn 被设置成当前系统的最大 scn 值。在数据库打开过程中，Oracle 会比较各文件的 stop scn 和 checkpoint scn，如果值不一致，表明数据库先前没有正常关闭，需要做恢复。

```
SYS@lhrdb> SELECT TABLESPACE_NAME,STATUS FROM DBA_TABLESPACES;

TABLESPACE_NAME                STATUS
------------------------------ ---------
SYSTEM                         ONLINE
SYSAUX                         ONLINE
UNDOTBS1                       ONLINE
TEMP                           ONLINE
USERS                          READ ONLY
EXAMPLE                        ONLINE
TS_MIG_CHAIN_LHR               ONLINE
TS_TESTBLOCKLHR                ONLINE

8 rows selected.

SYS@lhrdb>  SELECT FILE#,LAST_CHANGE# FROM  V$DATAFILE;

     FILE# LAST_CHANGE#
---------- ------------
         1
         2
         3
         4      9028165
         5
         6
         7

7 rows selected.
```

可以看到除了 USERS 表空间的结束 SCN 不为空，其他数据文件的结束 SCN 为空。

将数据库至于 MOUNT 状态，由于该状态下所有的数据文件都不可写，故 MOUNT 状态下所有的数据文件都具有结束 SCN。

```
SYS@lhrdb> startup mount
ORACLE instance started.

Total System Global Area 1720328192 bytes
Fixed Size                 2247072 bytes
Variable Size            452986464 bytes
Database Buffers        1258291200 bytes
Redo Buffers              6803456 bytes
Database mounted.
SYS@lhrdb> SELECT FILE#,LAST_CHANGE# FROM  V$DATAFILE;

     FILE# LAST_CHANGE#
---------- ------------
         1      9048847
         2      9048847
         3      9048847
         4      9028165
         5      9048847
         6      9048847
         7      9048847
```

```
7 rows selected.

SYS@lhrdb> alter tablespace users read write;

Tablespace altered.

SYS@lhrdb> SELECT FILE#,LAST_CHANGE# FROM  V$DATAFILE;

    FILE# LAST_CHANGE#
---------- ------------
        1
        2
        3
        4
        5
        6
        7

7 rows selected.

SYS@lhrdb> startup force mount
ORACLE instance started.

Total System Global Area 1720328192 bytes
Fixed Size                2247072 bytes
Variable Size           452986464 bytes
Database Buffers       1258291200 bytes
Redo Buffers              6803456 bytes
Database mounted.
SYS@lhrdb> SELECT FILE#,LAST_CHANGE# FROM  V$DATAFILE;

    FILE# LAST_CHANGE#
---------- ------------
        1
        2
        3
        4
        5
        6
        7

7 rows selected.
```

1.3.4.3 HIGH AND LOW SCN

ORACLE 的 REDO LOG 会顺序纪录数据库的各个变化。一组 REDO LOG 文件写满后，会自动切换到下一组 REDO LOG 文件。则上一组 REDO LOG 的 HIGH SCN 就是下一组 REDO LOG 的 LOW SCN。在 CURRENT LOG 中 HIGH SCN 为无穷大。

在视图 V$LOG_HISTORY 中，SEQUENCE#代表 REDO LOG 的序列号，FIRST_CHANGE#表示当前 REDO LOG 的 LOW SCN，列 NEXT_CHANGE#表示当前 REDO LOG 的 HIGH SCN。

可通过查询 V$LOG_HISTORY 查看 LOW SCN 和 HIGH SCN。

```
SYS@lhrdb> set pagesize 9999
SYS@lhrdb> SELECT RECID,SEQUENCE#,FIRST_CHANGE#,NEXT_CHANGE# FROM
V$LOG_HISTORY WHERE ROWNUM<=6;
    RECID  SEQUENCE# FIRST_CHANGE# NEXT_CHANGE#
---------- ---------- ------------- ------------
      272       272      7486197     7510243
      273       273      7510243     7527538
      274       274      7527538     7539409
      275       275      7539409     7556740
      276       276      7556740     7572195
      277       277      7572195     7581847
6 rows selected.
```

查看 CURRNET REDO LOG 中的 HIGH SCN

```
SYS@lhrdb> COL MEMBER FORMAT A50
SYS@lhrdb> SELECT VF.MEMBER,V.STATUS,V.FIRST_CHANGE# FROM V$LOGFILE VF,V$LOG V
  2    WHERE VF.GROUP#=V.GROUP#
  3    AND V.STATUS='CURRENT';

MEMBER                                             STATUS           FIRST_CHANGE#
-------------------------------------------------- ---------------- -------------
+DATA/lhrdb/onlinelog/group_4.798.923841413        CURRENT                9069089
+DATA/lhrdb/onlinelog/group_4.797.923841415        CURRENT                9069089

SYS@lhrdb> ALTER SYSTEM DUMP LOGFILE '+DATA/lhrdb/onlinelog/group_4.797.923841415';

System altered.

SYS@lhrdb> oradebug setmypid
Statement processed.
SYS@lhrdb> oradebug tracefile_name
/oracle/app/oracle/diag/rdbms/lhrdb/lhrdb/trace/lhrdb_ora_8388948.trc
```

查看转储文件的内容：

```
DUMP OF REDO FROM FILE '+DATA/lhrdb/onlinelog/group_4.797.923841415'
 Opcodes *.*
 RBAs: 0x000000.00000000.0000 thru 0xffffffff.ffffffff.ffff
 SCNs: scn: 0x0000.00000000 thru scn: 0xffff.ffffffff
 Times: creation thru eternity
 FILE HEADER:
       Compatibility Vsn = 186647552=0xb200400
       Db ID=959319562=0x392e0e0a, Db Name='LHRDB'
       Activation ID=959339270=0x392e5b06
       Control Seq=96545=0x17921, File size=204800=0x32000
       File Number=4, Blksiz=512, File Type=2 LOG
 descrip:"Thread 0001, Seq# 0000001090, SCN 0x0000008a6221-0xffffffffffff"
 thread: 1 nab: 0xffffffff seq: 0x00000442 hws: 0x2 eot: 1 dis: 0
 resetlogs count: 0x36a23c8c scn: 0x0000.000e20dc (925916)
 prev resetlogs count: 0x3155bebd scn: 0x0000.00000001 (1)
 Low  scn: 0x0000.008a6221 (9069089) 10/11/2016 16:46:41
 Next scn: 0xffff.ffffffff 01/01/1988 00:00:00
 Enabled scn: 0x0000.000e20dc (925916) 07/07/2016 19:39:56
 Thread closed scn: 0x0000.008a6221 (9069089) 10/11/2016 16:46:41
 Disk cksum: 0xc14c Calc cksum: 0xc14c
 Terminal recovery stop scn: 0x0000.00000000
```

```
Terminal recovery  01/01/1988 00:00:00
Most recent redo scn: 0x0000.00000000
Largest LWN: 0 blocks
End-of-redo stream : No
Unprotected mode
Miscellaneous flags: 0x800000
Thread internal enable indicator: thr: 0, seq: 0 scn: 0x0000.00000000
Zero blocks: 0
Format ID is 1
redo log key is 47e6cd1abd3a43fd864d2b94ae9a8128
redo log key flag is 5
Enabled redo threads: 1
```

当前最新的数据库 scn 值可通过如下命令查看：

```
SYS@lhrdb> select dbms_flashback.get_system_change_number from dual;
GET_SYSTEM_CHANGE_NUMBER
------------------------
                 9069555
```

如果需要进行实例恢复，则需要恢复的记录为 9069089 至 9069555 中 redo log 中的记录。

1.3.5 SCN 号于数据库的启动、关闭

Scn 号与 Oracle 数据库恢复过程有着密切的关系，只有很好地理解了这层关系，才能深刻地理解恢复的原理。CKPT 进程在 checkpoint 发生时，将当时的 SCN 号写入数据文件头和控制文件，同时通知 DBWR 进程将数据块写到数据文件。

CKPT 进程也会在控制文件中记录 RBA(redo block address)，以标志 Recovery 需要从日志中哪个地方开始。

1.在数据库的启动过程中，当 System Checkpoint SCN=Datafile Checkpoint SCN=Start SCN 的时候，Oracle 数据库是可以正常启动的，而不需要做任何的 MEDIA RECOVERY。而如果三者当中有一个不同的话，则需要做 MEDIA RECOVERY。Oracle 在启动过程中首先检查是否需要 MEDIA RECOVERY，然后再检查是否需要 INSTANCE RECOVERY。

2.那什么时候需要做 INSTANCE RECOVERY 呢？其实在正常 OPEN 数据库的时候，Oracle 会将记录在控制文件中的每一个数据文件头的 End SCN 都设置为#FFFFFF(NULL)，那么如果数据库进行了正常关闭比如(shutdown or shutdown immediate)这个时候，系统会执行一个检查点，这个检查点会将控制文件中记录的各个数据文件头

的 End SCN 更新为当前 online 数据文件的各个数据文件头的 Start SCN，也就是 End SCN=Start SCN，如果再次启动数据库的时候发现二者相等，则直接打开数据库，并再次将 End SCN 设置为#FFFFFF(NULL)，那么如果数据库是异常关闭，那么 CHECKPOINT 就不会执行，因此再次打开数据库的时候 End SCN<>Start SCN 这个时候就需要做实例恢复。如果数据库异常关闭的话，则 END SCN 号将为 NULL.则需要做 instance recovery。

1.3.5.1 为什么需要 System checkpoint SCN 号与 Datafile Checkpoint SCN 号

为什么 ORACLE 会在控制文件中记录 System checkpoint SCN 号的同时，还需要为每个数据文件记录 Datafile Checkpoint SCN 号？

原因有二：

1.对只读表空间，其数据文件的 Datafile Checkpoint SCN、Start SCN 和 END SCN 号均相同。这三个 SCN 在表空间处于只读期间都将被冻结。

2.如果控制文件不是当前的控制文件，则 System checkpoint 会小于 Start SCN 或 END SCN 号。记录这些 SCN 号，可以区分控制文件是否是当前的控制文件。

```
SYS@lhrdb> alter tablespace users read only;

Tablespace altered.

SYS@lhrdb> SELECT A.FILE#,
  2         A.NAME,
  3         (SELECT CHECKPOINT_CHANGE# FROM V$DATABASE) SYSTEM_CKPT_SCN,
  4         A.CHECKPOINT_CHANGE# DF_CKPT_SCN,
  5         A.LAST_CHANGE# END_SCN,
  6         B.CHECKPOINT_CHANGE# START_SCN,
  7         B.RECOVER,
  8         A.STATUS
  9    FROM V$DATAFILE A, V$DATAFILE_HEADER B
 10   WHERE A.FILE# = B.FILE#;

    FILE# NAME                                                SYSTEM_CKPT_SCN DF_CKPT_SCN    END_SCN  START_SCN REC STATUS
--------- -------------------------------------------------- --------------- ----------- ---------- ---------- --- ----------
        1 +DATA/lhrdb/datafile/system.347.916601927                 9225394     9225394                9225394 NO  SYSTEM
        2 +DATA/lhrdb/datafile/sysaux.340.916601927                9225394     9225394                9225394 NO  ONLINE
        3 +DATA/lhrdb/datafile/undotbs1.353.916601927              9225394     9225394                9225394 NO  ONLINE
        4 +DATA/lhrdb/datafile/users.445.916601927                 9225394     9229175    9229175     9229175 NO  ONLINE
        5 +DATA/lhrdb/datafile/example.416.916602001               9225394     9225394                9225394 NO  ONLINE
        6 +DATA/lhrdb/datafile/ts_mig_chain_lhr.471.919677645      9225394     9225394                9225394 NO  ONLINE
        7 /oracle/app/oracle/oradata/lhrdb/testblocklhr01.dbf      9225394     9225394                9225394 NO  ONLINE

7 rows selected.
```

1.3.5.2 recover database using backup controlfile

当有一个 Start SCN 号超过了 System Checkpoint SCN 号时，则说明控制文件不是当前的控制文件，因此在做 recover 时需要采用 using backup controlfile。这是为什么需要记录 SystemCheckpoint SCN 的原因之一。

这里需要一提的是，当重建控制文件的时候，System Checkpoint SCN 为 0，Datafile Checkpoint SCN 的数据来自于 Start SCN。根据上述的描述，此时需要采用 using backup controlfile 做 recovery.

1.3.6 查看系统当前 SCN

Oracle 数据库提供了两种直接查看系统当前 SCN 的方法，一个是 V$DATABASE 中的 CURRENT_SCN 列，另外一个就是通过 DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER 得到。

```
SYS@ORACNSL1> COL SCN1 FOR 9999999999999
SYS@ORACNSL1> COL SCN2 FOR 9999999999999
SYS@ORACNSL1> COL SCN3 FOR 9999999999999
SYS@ORACNSL1> SELECT CURRENT_SCN SCN1,DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER
SCN2,TIMESTAMP_TO_SCN(SYSDATE) SCN3 FROM V$DATABASE;
         SCN1          SCN2       SCN3
-------------- -------------- ----------
    1495460388     1495460388 1495460387
```

一般情况下，SCN1 和 SCN2 的结果一致，但在系统比较繁忙的时候可能 SCN2 比 SCN1 稍微大一点，比如大 1。

在 oracle 9i 中要麻烦些，V$DATABASE 视图中没有 CURRENT_SCN 这列，只有通过查询 X$KTUXE 视图来得到。

```
SYS@lhrdb> SELECT MAX(KTUXESCNW*POWER(2,32)+KTUXESCNB) SCN FROM X$KTUXE;
       SCN
----------
   8764198
```

1.3.7 SCN 与时间的相互转换（SCN_TO_TIMESTAMP 与 TIMESTAMP_TO_SCN）

Oracle 10g 提供了两个新函数对于 SCN 和时间戳进行相互转换，这两个函数是 SCN_TO_TIMESTAMP、TIMESTAMP_TO_SCN，通过对 SCN 和时间戳进行转换，Oracle 极大地方便了很多备份和恢复过程。

一个 SCN 值总是发生在某一个特定的时刻的，只不过由于粒度的不一样，通常会存在多个 SCN 对应同一个时间戳。Oracle 中提供了两个函数以供我们进行 SCN 和时间的互换：

- SCN_TO_TIMESTAMP(scn_number) 将 SCN 转换成时间戳。
- TIMESTAMP_TO_SCN(timestamp) 将时间戳转换成 SCN。

通过这两个函数，最终 Oracle 将 SCN 和时间的关系建立起来，在 Oracle 10g 之前，是没有办法通过函数转换得到 SCN 和时间的对应关系的，一般可以通过 logmnr 分析日志获得。但是这种转换要依赖于数据库内部的数据记录（SMON_SCN_TIME），对于久远的 SCN 则不能转换，请看以下举例：

```
SYS@lhrdb>  SELECT MIN(FIRST_CHANGE#) SCN,DBMS_FLASHBACK.GET_SYSTEM_CHANGE_NUMBER FROM V$ARCHIVED_LOG;

      SCN GET_SYSTEM_CHANGE_NUMBER
---------- ------------------------
   7527538                8763206

SYS@lhrdb> SELECT SCN_TO_TIMESTAMP(7527538) SCN FROM DUAL;
select scn_to_timestamp(7527538) scn from dual
       *
ERROR at line 1:
ORA-08181: specified number is not a valid system change number
ORA-06512: at "SYS.SCN_TO_TIMESTAMP", line 1

SYS@lhrdb> select min(scn) from smon_scn_time;

  MIN(SCN)
----------
   8622517

SYS@lhrdb>  select scn_to_timestamp(8622517) timestamp from dual;

TIMESTAMP
---------------------------------------------------------------------------
08-OCT-16 04.30.26.000000000 AM

SYS@lhrdb>  select scn_to_timestamp(8622516) timestamp from dual;
 select scn_to_timestamp(8622516) timestamp from dual
        *
ERROR at line 1:
ORA-08181: specified number is not a valid system change number
ORA-06512: at "SYS.SCN_TO_TIMESTAMP", line 1
```

从上面的例子可以看出 Oracle 能够转换的最小 SCN 也就是 SMON_SCN_TIME.scn 的最小值。

```
SYS@lhrdb> SELECT SCN_TO_TIMESTAMP(8763206) SCN FROM DUAL;

SCN
---------------------------------------------------------------------------
```

```
10-OCT-16 05.22.40.000000000 PM

SYS@lhrdb> SELECT TIMESTAMP_TO_SCN(TO_TIMESTAMP('10-OCT-16 05.22.40.000000000 PM','DD-Mon-RR HH:MI:SS.FF AM')) SCN FROM DUAL;

      SCN
----------
   8763206

SYS@lhrdb> SELECT TIMESTAMP_TO_SCN(TO_TIMESTAMP('2016-10-10 17:22:40','YYYY-MM-DD HH24:MI:SS')) SCN FROM DUAL;

      SCN
----------
   8763206

SYS@lhrdb> SELECT TO_CHAR(SCN_TO_TIMESTAMP(8763206), 'YYYY-MM-DD HH24:MI:SS') CHR_DATE,TIMESTAMP_TO_SCN(SCN_TO_TIMESTAMP(8763206)) DT FROM DUAL;

CHR_DATE                 DT
-------------------- ----------
2016-10-10 17:22:40    8763206
```

对于时间到 SCN 的转换，Oracle 只能定位到 3 秒以内，3 秒内的时间都被转换成同一个 SCN：

```
SYS@lhrdb> SELECT TIMESTAMP_TO_SCN(TO_TIMESTAMP('2016-10-10 17:22:40','YYYY-MM-DD HH24:MI:SS')) SCN FROM DUAL;

      SCN
----------
   8763206

SYS@lhrdb> SELECT TIMESTAMP_TO_SCN(TO_TIMESTAMP('2016-10-10 17:22:41','YYYY-MM-DD HH24:MI:SS')) SCN FROM DUAL;

      SCN
----------
   8763206

SYS@lhrdb> SELECT TIMESTAMP_TO_SCN(TO_TIMESTAMP('2016-10-10 17:22:42','YYYY-MM-DD HH24:MI:SS')) SCN FROM DUAL;

      SCN
----------
   8763206

SYS@lhrdb> SELECT TIMESTAMP_TO_SCN(TO_TIMESTAMP('2016-10-10 17:22:43','YYYY-MM-DD HH24:MI:SS')) SCN FROM DUAL;

      SCN
----------
   8763213

SYS@lhrdb> SELECT TIMESTAMP_TO_SCN(TO_TIMESTAMP('2016-10-10 17:22:39','YYYY-MM-DD HH24:MI:SS')) SCN FROM DUAL;

      SCN
----------
   8763205
```

1.3.7.1 SMON_SCN_TIME

```
SELECT * FROM DBA_TABLES D WHERE D.TABLE_NAME = 'SMON_SCN_TIME';
```

	OWNER	TABLE_NAME	TABLESPACE_NAME
1	SYS	SMON_SCN_TIME	SYSAUX

Oracle 在内部都是使用 scn，即使你指定的是 as of timestamp，oracle 也会将其转换成 scn，系统时间标记与 scn 之间存在一张表，即 SYS 下的 SMON_SCN_TIME。

```
SYS@lhrdb> set linesize 80
SYS@lhrdb> desc sys.smon_scn_time
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 THREAD                                             NUMBER
 TIME_MP                                            NUMBER
 TIME_DP                                            DATE
 SCN_WRP                                            NUMBER
 SCN_BAS                                            NUMBER
 NUM_MAPPINGS                                       NUMBER
 TIM_SCN_MAP                                        RAW(1200)
 SCN                                                NUMBER
 ORIG_THREAD                                        NUMBER
```

每隔 5 分钟，系统产生一次系统时间标记与 scn 的匹配并存入 sys.smon_scn_time 表，该表中记录了最近 1440 个系统时间标记与 scn 的匹配记录，由于该表只维护了最近的 1440 条记录，因此如果使用 as of timestamp 的方式则只能 flashback 最近 5 天内的数据（假设系统是在持续不断运行并无中断或关机重启之类操作的话）。

查看 SCN 和 timestamp 之间的对应关系：

```
SELECT SCN, TO_CHAR(TIME_DP, 'YYYY-MM-DD HH24:MI:SS') TIME_DP
  FROM SYS.SMON_SCN_TIME T
 ORDER BY T.SCN DESC;
```

	SCN	TIME_DP
1	9229645	2016-10-13 02:10:33
2	9229527	2016-10-13 02:05:19
3	9229405	2016-10-13 02:00:18
4	9229140	2016-10-13 01:56:12
5	9229022	2016-10-13 01:51:00
6	9228911	2016-10-13 01:46:09
7	9228792	2016-10-13 01:41:00
8	9228680	2016-10-13 01:36:00
9	9228569	2016-10-13 01:31:06
10	9228465	2016-10-13 01:26:00
11	9228286	2016-10-13 01:21:12
12	9228143	2016-10-13 01:16:00
13	9227972	2016-10-13 01:11:09
14	9227862	2016-10-13 01:06:00
15	9227507	2016-10-13 01:00:42
16	9227393	2016-10-13 00:56:00
17	9227279	2016-10-13 00:50:42
18	9227167	2016-10-13 00:45:54
19	9227057	2016-10-13 00:40:42

有关表 SMON_SCN_TIME 的更多内容可以参考我的 BLOG：http://blog.itpub.net/26736162/viewspace-2126291/，David 大神写的，非常全面，我就不画蛇添足了。

1.3.8 实例恢复（INSTANCE RECOVERY）和介质恢复（MEDIA RECOVERY）

REDO LOG 是 Oracle 为确保已经提交的事务不会丢失而建立的一个机制。实际上 REDO LOG 的存在是为两种场景准备的，一种我们称之为实例恢复（INSTANCE RECOVERY），一种我们称之为介质恢复（MEDIA RECOVERY）。

REDO LOG 的数据是按照 THREAD 来组织的，对于单实例系统来说，只有一个 THREAD，对于 RAC 系统来说，可能存在多个 THREAD，每个数据库实例拥有一组独立的 REDO LOG 文件，拥有独立的 LOG BUFFER，某个实例的变化会被独立的记录到一个 THREAD 的 REDO LOG 文件中。

1.3.8.1 实例恢复

对于单实例的系统，实例恢复一般是在数据库实例异常故障后数据库重启时进行，当数据库执行了 SHUTDOWN ABORT 或者由于操作系统、主机等原因宕机重启后，在执行 ALTER DATABASE OPEN 的时候，就会自动做实例恢复。而在 RAC 环境中，如果某个实例宕机了，那么剩下的实例将会代替宕掉的实例做实例恢复。除非是所有的实例都宕机了，这样的话，第一个执行 ALTER DATABASE OPEN 的实例将会做实例恢复。这也是在 RAC 环境中，REDO LOG 是实例私有的组件，但是 REDO LOG 的文件必须存放在共享存储上的原因。

一、 RAC 中的实例恢复

一个单实例数据库或者 RAC 数据库所有实例失败之后，第一个打开数据库的实例会自动执行实例恢复。这种形式的实例恢复称为 Crash 恢复。一个 RAC 数据库的一部分但不是所有实例失败后，在 RAC 中幸存的实例自动执行失败实例的恢复称为实例恢复。一般而言，在崩溃或关机退出之后第一个打开数据库的实例将自动执行崩溃恢复。

根据 Crash 恢复和实例恢复的不同，由幸存实例或者第一个重启的实例读取失败实例生成的联机 Redo 日志和 UNDO 表空间数据，使用这些信息确保只有已提交的事务被写到数据库中，回滚在失败时候活动的事务，并释放事务使用的资源。

```
[ZFZHLHRDB1:oracle]:/oracle>crsctl stat res -t
--------------------------------------------------------------------------------
NAME           TARGET  STATE        SERVER                   STATE_DETAILS
--------------------------------------------------------------------------------
Local Resources
--------------------------------------------------------------------------------
ora.DATA.dg
               ONLINE  ONLINE       zfzhlhrdb1
               ONLINE  ONLINE       zfzhlhrdb2
ora.LISTENER.lsnr
               ONLINE  ONLINE       zfzhlhrdb1
               ONLINE  ONLINE       zfzhlhrdb2
ora.LISTENER_LHRDG.lsnr
               ONLINE  ONLINE       zfzhlhrdb1
               ONLINE  ONLINE       zfzhlhrdb2
ora.asm
               ONLINE  ONLINE       zfzhlhrdb1               Started
               ONLINE  ONLINE       zfzhlhrdb2               Started
ora.gsd
               OFFLINE OFFLINE      zfzhlhrdb1
               OFFLINE OFFLINE      zfzhlhrdb2
ora.net1.network
               ONLINE  ONLINE       zfzhlhrdb1
               ONLINE  ONLINE       zfzhlhrdb2
ora.ons
               ONLINE  ONLINE       zfzhlhrdb1
               ONLINE  ONLINE       zfzhlhrdb2
ora.registry.acfs
               ONLINE  ONLINE       zfzhlhrdb1
               ONLINE  ONLINE       zfzhlhrdb2
--------------------------------------------------------------------------------
Cluster Resources
--------------------------------------------------------------------------------
ora.LISTENER_SCAN1.lsnr
      1        ONLINE  ONLINE       zfzhlhrdb1
ora.cvu
      1        ONLINE  ONLINE       zfzhlhrdb1
ora.lhrdb.db
      1        ONLINE  ONLINE       zfzhlhrdb1               Open
ora.oc4j
      1        ONLINE  ONLINE       zfzhlhrdb1
ora.raclhr.db
      1        ONLINE  ONLINE       zfzhlhrdb2               Open
      2        ONLINE  ONLINE       zfzhlhrdb1               Open
ora.scan1.vip
      1        ONLINE  ONLINE       zfzhlhrdb1
ora.zfzhlhrdb1.vip
      1        ONLINE  ONLINE       zfzhlhrdb1
ora.zfzhlhrdb2.vip
      1        ONLINE  ONLINE       zfzhlhrdb2
[ZFZHLHRDB1:oracle]:/oracle>srvctl stop instance -d raclhr -i raclhr1 -o abort
[ZFZHLHRDB1:oracle]:/oracle>srvctl status db -d raclhr
Instance raclhr1 is not running on node zfzhlhrdb1
Instance raclhr2 is running on node zfzhlhrdb2
```

abort 掉实例 1 后：

实例一的告警日志：

```
Thu Oct 13 15:51:30 2016
Shutting down instance (abort)
License high water mark = 60
USER (ospid: 4194780): terminating the instance
Instance terminated by USER, pid = 4194780
Thu Oct 13 15:51:32 2016
Instance shutdown complete
```

实例二的告警日志：

```
Thu Oct 13 15:51:31 2016
Reconfiguration started (old inc 4, new inc 6)
List of instances:
 2 (myinst: 2)
 Global Resource Directory frozen
 * dead instance detected - domain 0 invalid = TRUE
 Communication channels reestablished
 Master broadcasted resource hash value bitmaps
 Non-local Process blocks cleaned out
Thu Oct 13 15:51:31 2016
 LMS 0: 0 GCS shadows cancelled, 0 closed, 0 Xw survived
Thu Oct 13 15:51:31 2016
 LMS 1: 0 GCS shadows cancelled, 0 closed, 0 Xw survived
 Set master node info
 Submitted all remote-enqueue requests
 Dwn-cvts replayed, VALBLKs dubious
 All grantable enqueues granted
 Post SMON to start 1st pass IR
Thu Oct 13 15:51:31 2016
Instance recovery: looking for dead threads
 Submitted all GCS remote-cache requests
 Post SMON to start 1st pass IR
 Fix write in gcs resources
Reconfiguration complete
Beginning instance recovery of 1 threads
 parallel recovery started with 7 processes
Started redo scan
Completed redo scan
 read 18 KB redo, 14 data blocks need recovery
Started redo application at
 Thread 1: logseq 235, block 68352
Recovery of Online Redo Log: Thread 1 Group 1 Seq 235 Reading mem 0
  Mem# 0: +DATA/raclhr/onlinelog/group_1.362.916601361
  Mem# 1: +DATA/raclhr/onlinelog/group_1.361.916601361
Completed redo application of 0.01MB
Completed instance recovery at
 Thread 1: logseq 235, block 68389, scn 9725527
 14 data blocks read, 14 data blocks written, 18 redo k-bytes read
Thu Oct 13 15:51:33 2016
minact-scn: Inst 2 is now the master inc#:6 mmon proc-id:25100420 status:0x7
minact-scn status: grec-scn:0x0000.00000000 gmin-scn:0x0000.009417d9 gcalc-scn:0x0000.009417e3
minact-scn: master found reconf/inst-rec before recscn scan old-inc#:6 new-inc#:6
Thread 1 advanced to log sequence 236 (thread recovery)
Redo thread 1 internally disabled at seq 236 (SMON)
Thu Oct 13 15:51:34 2016
Thread 2 advanced to log sequence 265 (LGWR switch)
  Current log# 4 seq# 265 mem# 0: +DATA/raclhr/onlinelog/group_4.349.916601715
  Current log# 4 seq# 265 mem# 1: +DATA/raclhr/onlinelog/group_4.348.916601715
Thu Oct 13 15:51:35 2016
Archived Log entry 493 added for thread 1 sequence 235 ID 0x441b1480 dest 1:
Thu Oct 13 15:51:35 2016
ARC0: Archiving disabled thread 1 sequence 236
```

```
Archived Log entry 494 added for thread 1 sequence 236 ID 0x441b1480 dest 1:
Thu Oct 13 15:51:35 2016
Archived Log entry 495 added for thread 2 sequence 264 ID 0x441b1480 dest 1:
minact-scn: master continuing after IR
```

1.3.8.2 介质恢复

介质恢复是基于物理备份恢复数据，它是 Oracle 数据库出现介质故障时恢复的重要保障。介质恢复包括块恢复、数据文件恢复、表空间恢复和整个数据库的恢复。介质恢复主要是针对错误类型中的介质失败，如果是少量的块失败，那么可以使用介质恢复中的块恢复来快速修复；但如果是其它情况的丢失，那么需要根据具体情况，可使用数据文件恢复、表空间恢复甚至全库恢复，可以参考如下的表格：

错误分类	恢复解决方案
介质失败	如果是少量的块损坏，使用块介质恢复；如果是大量的块、数据文件、表空间的损坏，可能需要对损坏的数据文件或者表空间执行完全恢复；如果是归档 REDO 日志文件或者联机 REDO 日志文件的丢失，那么只需要不完全恢复方式。
逻辑损坏	如果是程序员错误导致出现的问题，可通过补丁应用修复问题。对于无法修复的问题，也可采用介质恢复手段来恢复数据。
用户错误	根据不同用户错误，选择不同的 FLASHBACK 技术恢复，使用 FLASHBACK 技术恢复用户错误是首选方案。如果 FLASHBACK 不能很好的恢复数据再考虑使用介质恢复或者表空间时间点恢复。

Oracle 数据库的介质恢复实际上包含了两个过程：数据库还原（RESTORE）与数据库恢复（RECOVER）。

数据库还原(RESTORE)是指利用备份的数据库文件来替换已经损坏的数据库文件或者将其恢复到一个新的位置。RMAN 在进行还原操作时，会利用恢复目录（有建立恢复目录的话就使用目标数据库的控制文件）来获取备份信息，并从中选择最合适的备份进行修复操作。选择备份时有两个原则：1、选择距离恢复目录时刻最近；2、优先选择镜像复制，其次才是备份集。

数据库恢复（RECOVER）是指数据文件的介质恢复，即为修复后的数据文件应用联机或归档日志，从而将修复的数据库文件更新到当前时刻或指定时刻下的状态。在执行恢复数据库时，需要使用 RECOVER 命令。

还原是将某个时间点数据文件的拷贝再拷贝回去，还原后的数据库处于不一致性的状态，或不是最新的状态，还需要执行恢复操作。恢复就是使用归档 REDO 日志文件和联机 REDO 日志文件将不一致的数据库应用到一致性状态。

需要注意的是，还原只是建立在数据库备份的基础版本上，例如，如果数据库备份包括 0 级备份和很多 1 级备份，还原只是应用 0 级备份，恢复过程会根据情况自动应用 1 级备份或 REDO 日志将数据库恢复到一致性的状态。

数据库的恢复过程根据恢复数据的程度又分为完全恢复（Complete Recovery）和不完全恢复（Incomplete Recovery）。

完全恢复是一种没有数据丢失的恢复方式，能够恢复到最新的联机 REDO 日志中已提交的数据。在传统恢复方式中，因介质失败破坏了数据文件之后，可以在数据库、表空间和数据文件上执行完全介质恢复。

不完全恢复是一种与完全恢复相反的恢复方式，是一种丢失数据的恢复方式，也称为数据库基于时间点恢复（Point-in-Time Recovery），是将整个数据库恢复到之前的某个时间点、日志序列号或者 SCN 号。通常情况下，若 FLASHBACK DATABASE 没有启用或者变得无效，可以执行不完全恢复撤销一个用户错误。不完全恢复不一定在原有的数据库环境执行，可以在测试环境下执行不完全恢复，将找回的数据再重新导入生产库中。不完全恢复根据备份情况恢复到与指定时间、日志序列号和 SCN 具有一致性的数据，之后的数据都将丢失。执行不完全恢复一方面是因为归档 REDO 日志、联机 REDO 日志的丢失不得不执行不完全恢复，另一方面可能是因为在某个时刻错误地操作了数据，过了一段时间之后才发现问题，而其它的恢复手段都无法恢复数据，这时也不得不使用不完全恢复来找回数据。执行不完全恢复必须从备份中还原所有的数据文件，备份文件必须是要恢复的时间点之前创建的。当恢复完成，使用 RESTLOGS 选项打开数据库，将重新初始化联机 Redo 日志，创建一个新的日志序列号流，日志序列号从 1 开始，RESETLOGS 之后的 SCN 还是在递增。

如果是完全恢复，那么数据库就是最新的一致性状态；如果是不完全恢复，那么数据库是非最新的一致性状态。对于非归档模式的数据库来说，不能执行不完全恢复。

不完全恢复的选项如下表所示：

不完全恢复方式	RMAN 选项	用户管理备份选项
恢复到某个时间点	UNTIL TIME	UNTIL TIME
恢复到某个日志序列号	UNTIL SUQUENCE	UNTIL CANCEL
恢复到某个 SCN 号	UNTIL SCN	UNTIL CHANGE

不完全恢复的几种类型如下表所示（**注意：下图是小麦苗即将出版的书上的一个表格，现在提前分享给大家**）：

分类	简介	脚本

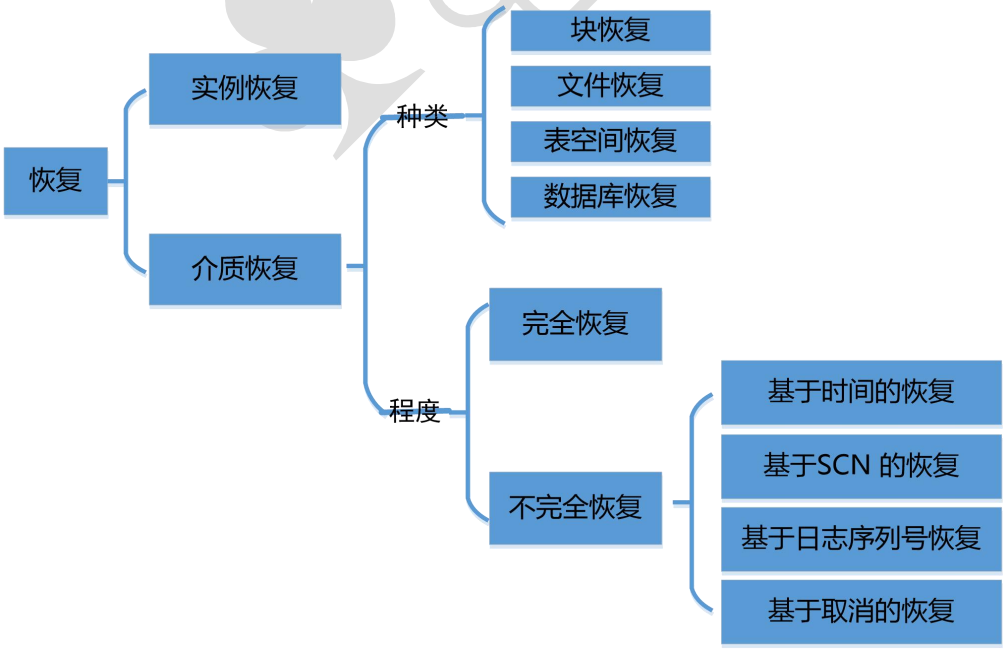
基于时间的恢复（Time-based Recovery）	将数据恢复到指定的时间点，SET UNTIL TIME时间可以使用多种表示方式，可以使用TO_DATE函数来表示时间，还可以使用SYSDATE-1方式来表示时间。	RUN {
		ALLOCATE CHANNEL C1 TYPE DISK;
		ALLOCATE CHANNEL C2 TYPE DISK;
		STARTUP FORCE MOUNT;
		SQL 'ALTER SESSION SET NLS_DATE_FORMAT="YYYY-MM-DD HH24:MI:SS"';
		SET UNTIL TIME = "TO_DATE('2015-01-19 14:20:20','YYYY-MM-DD HH24:MI:SS')";
		RESTORE DATABASE;
		RECOVER DATABASE;
		ALTER DATABASE OPEN RESETLOGS;
		RELEASE CHANNEL C1;
		RELEASE CHANNEL C2;
		}

基于SCN的恢复（Change-based Recovery）	将数据恢复到指定的SCN。	RUN {
		SHUTDOWN IMMEDIATE;
		STARTUP MOUNT;
		SET UNTIL SCN 324394;
		RESTORE DATABASE;
		RECOVER DATABASE;
		ALTER DATABASE OPEN RESETLOGS;
		}

基于日志序列号恢复（Log Sequence Recovery）	将数据恢复到指定的重做日志序列号（仅使用RMAN时有效）。基于序列号的不完全恢复须指定某个REDO线程的序列号，那么在这个序列号切换时间点之前的所有实例的归档日志都需要的，每个节点的负载不同，其他实例的序列号可能比指定的REDO线程序列号要大。	RUN {
		SHUTDOWN IMMEDIATE;
		STARTUP MOUNT;
		SET UNTIL SEQUENCE 10350 THREAD 1;
		RESTORE DATABASE;
		RECOVER DATABASE;
		ALTER DATABASE OPEN RESETLOGS;
		}

基于取消的恢复（Cancel-based Recovery）	当用户提交CANCEL后停止恢复（此选项在使用RMAN时无效）。	

综上所述，恢复的分类大致可以如下图所示的分类（**注意：下图也是小麦苗即将出版的书上的一个表格，现在提前分享给大家**）：



1.3.8.3 实例恢复和介质恢复的区别（注意：下图也是小麦苗即将出版的书上的一个表格，现在提前分享给大家）

分类	实例恢复（INSTANCE RECOVERY）	介质恢复（MEDIA RECOV
简介	数据库没有正常关闭，比如断电，或执行了以下命令：SHUTDOWN ABORT 、STARTUP FORCE，都会导致数据库实例在重启时执行实例恢复，具体是由SMON这个进程来完成这个任务。	介质恢复是当存储的数据文件出现故障（例如，数进行的，介质恢复无法自动进行，必须手工执行R RECOVER DATAFILE命令来实施。一般来说，介文件为起点进行恢复，因此在做介质恢复的时候 介质恢复是基于物理备份恢复数据，介质恢复技术故障时恢复的重要保障。
恢复的目的	在数据库发生故障时，确保BUFFER CACHE中的数据不会丢失，不会造成数据库的不一致。只有当联机REDO日志文件和UNDO表空间的介质没有被破坏才能确保实例恢复能成功。	当数据文件发生故障时，能够恢复数据。
利用资源	ONLINE REDO、ACTIVE REDO和UNDO	BACKUP SET ARCHIVE LOG、ONLINE AND IN
过程	利用REDO前滚（重做）；DB开启； 利用UNDO回滚	RESTORE; RECOVER
是否自动完成	自动完成	手工恢复，需要DBA干预，分为完全和不完全恢复
是否需要开启归档	不需要开启归档模式	需要开启归档模式
举例	用户在8：30：00触发了5个事务，分别是T1、T2、T3、T4、T5。在8：38：00分之前，T3和T5完成，在8：38：00产生了一个检查点事件，此时系统将对数据的更改都写入到数据文件中在8：38：27：00时，出现故障，导致实例异常关闭，在8：38：00至8：38：27之间的操作仅仅记录在REDO LOG中，并没有将这些更改写入到数据文件中。 当实例重新启动时，SMON将执行实例恢复： 例如：在8：43：00分重启实例，SMON执行实例恢复，整个执行过程可以分为两个部分：前滚和回滚。前滚是指8：38：00至8：38：27：00之间的操作应用到数据文件上，由于这些操作都记录在REDO LOG中，因此只需要从REDO LOG中读取这些操作并执行即可。前滚执行完毕后，数据库处于实例异常关闭前的状态，此时进入回滚阶段是指将未提交的事务回滚，即将示例中T1、T2和T4回滚至此，实例恢复执行完毕，8：45：00时，数据库正常打开。	在系统重启后发现数据库数据文件被rm删除了，通 DATABASE和RECOVER DATABASE命令进行介质恢

About Me

- 本文作者：小麦苗，只专注于数据库的技术，更注重技术的运用
- 本文在 itpub（http://blog.itpub.net/26736162）、博客园（http://www.cnblogs.com/lhrbest）和
- 本文 itpub 地址：http://blog.itpub.net/26736162/viewspace-2126407/
- 本文博客园地址：http://www.cnblogs.com/lhrbest/p/5961987.html
- 本文 pdf 版：http://yunpan.cn/cdEQedhCs2kFz （提取码：ed9b）

- 小麦苗云盘地址：http://blog.itpub.net/26736162/viewspace-1624453/
- QQ 群：230161599　　微信群：私聊
- 联系我请加 QQ 好友(642808185)，注明添加缘由
- 于 2016-10-08 15:00~ 2016-10-14 23:00 在中行完成
- 文章内容来源于小麦苗的学习笔记，部分整理自网络，若有侵权或不当之处还请谅解！
- 【版权所有，文章允许转载，但须以链接方式注明源地址，否则追究法律责任】

. . . . . .

手机长按下图识别二维码或微信客户端扫描下边的二维码来关注小麦苗的微信公众号：xiaomaimiaolhr,免费学习最