12. User-managed and RMAN-based incomplete recovery.

Abstract: Incomplete database recovery requires an understanding of the redo log and ARCHIVELOG processes, the synchronization of the Oracle database, and the options allowed for performing an incomplete recovery. This lesson discusses the incomplete recovery process and the commands associated with each incomplete recovery option. It also includes an example that shows how to perform an incomplete recovery due to lost or corrupted current redo log files. In addition to the user-managed incomplete recovery, this lesson demonstrates how to use RMAN for this process. The RMAN examples covered include how to use the SET UNTIL TIME and UNTIL SEQUENCE clauses. Incomplete recovery is the only method of recovery for certain types of failures. It is important that you understand when and how to use incomplete recovery methods. Anytime there is a failure that will cause you not to apply all the changes back to the databases, you will need to use one of the incomplete recovery methods that will be discussed in this lesson.

Contents

1. Describing the User-Managed Incomplete Recovery 2
2. Performing an Incomplete Database Recovery 3
   2.1. Cancel-Based Recovery 3
   2.2. Time-Based Recovery 6
   2.3. Change-Based Recovery 6
3. Recovering after Losing Current Redo Logs 7
4. Performing an RMAN-Based Incomplete Recovery Using UNTIL TIME 9
5. Performing RMAN-Based Incomplete Recovery Using UNTIL SEQUENCE 15
6. Summary 21
References 21

Objectives:

- Describe the steps of incomplete recovery.
- Perform an incomplete database recovery.
- Identify the loss of current online redo log files.
- Perform an incomplete database recovery using UNTIL TIME.
- Perform an incomplete database recovery using UNTIL SEQUENCE.

1. Describing the User-Managed Incomplete Recovery

Incomplete recovery occurs when the database is not recovered entirely to the point at which the database failed. This is a partial recovery of the database in which some archived logs are applied to the database, but not all of them. With this type of recovery, only a portion of the transactions gets applied.

There are three types of incomplete media recovery:

- Cancel-based
- Time-based
- Change-based

Each of these options allows recovery to a point in time prior to the failure. The main reason why there are three options is so that the DBA can have better flexibility and control of where recovery is halted during the recovery process. Each of these options is described in detail in the next section.

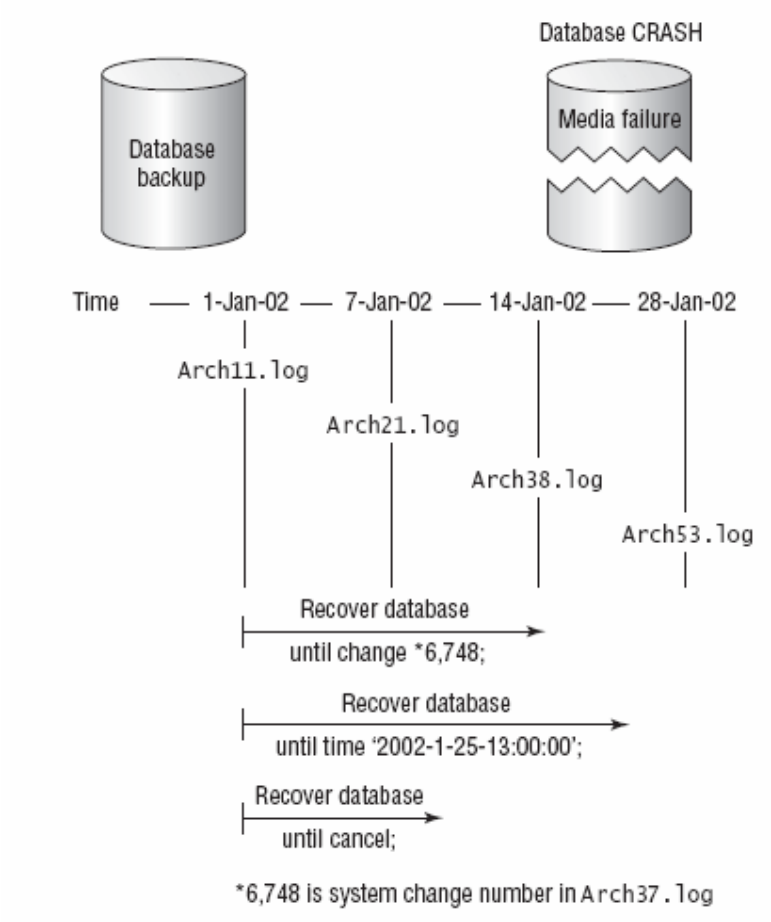

FIGURE. 1. Incomplete recovery in ARCHIVELOG mode for a media failure on January 28th

Incomplete recovery is performed any time you don't want to apply all the archived and nonarchived log files that are necessary to bring the database up to the time of failure. As a result, the database is essentially not completely recovered; transactions remain missing. Figure 1 illustrates the different types of incomplete recovery.

Incomplete recovery should be performed when the DBA wants or is required to recover the database prior to the point of time when the database failed. Some of the circumstances that might call for this type of recovery include data file corruption, redo log corruption, or the loss of a table due to user error.

In some cases, incomplete recovery is the only option available to you. In a failure situation that involves the loss or corruption of the current and inactive nonarchived redo log files, recovering the database without the transactions in these files is the only option. If a complete recovery were performed instead, the error would be reintroduced as a result of the recovery process.

2. Performing an Incomplete Database Recovery

This section details the three types of incomplete database recovery: cancel-based, time-based, and change-based. Each of these methods is used for different circumstances, and you should be aware of when each is appropriate.

2.1. Cancel-Based Recovery

In *cancel-based recovery*, you cancel the recovery before the point of failure. Cancel-based recovery provides the least flexibility and control of the stopping point during the recovery process. In this type of recovery, you apply archived logs during the recovery process. At some point before the recovery is complete, you enter the CANCEL command. At this point, recovery halts, and no more archived logs are applied.

The following is a sample of cancel-based incomplete recovery.

```
SQL> recover database until cancel;
```

One example of when you would use a cancel-based incomplete recovery is when you need to restore a backup of a lost data file from a hot backup. To do this, you perform the following steps:

1. Make sure that the database is shutdown by using the SHUTDOWN command from SQL*Plus.

```
SQL> shutdown abort
```

2. Make sure that current copies of the data files, control files, and parameter files exist in case there are errors that arise during the recovery process. If these files exist, you will be able to restart the recovery process, if needed, without introducing any new errors that might have resulted from a previous failed recovery.

3. Make sure that a current backup exists because copied files from the current backup will replace the failed data files, online redo log files, or control files.

4. Restore the data files from the backup location to the proper location. You do this by issuing an operating system–specific command. In Unix, you would use a *cp* command, as in this example:

```
cp /stage/data01.dbf /oracle/database/tst9/data01.dbf
cp /stage/system01.dbf /oracle/database/tst9/system01.dbf
cp /stage/rbs01.dbf /oracle/database/tst9/rbs01.dbf
cp /stage/temp01.dbf /oracle/database/tst9/temp01.dbf
cp /stage/users01.dbf /oracle/database/tst9/users01.dbf
cp /stage/tools01.dbf /oracle/database/tst9/tools01.dbf
cp /stage/indx01.dbf /oracle/database/tst9/indx01.dbf
```

5. Start the database in MOUNT mode.

```
SQL> startup mount
```

6. Verify that all the data files you need to recover are online. The following query shows the status of each data file that is online (with the exception of the system data file, which is always online; status equals system).

```
SQL> select file#,status,enabled,name from v$datafile;
FILE#       STATUS ENABLED  NAME
---------- ------- ---------- ------------------------
1 SYSTEM    READ WRITE      /oracle/database/tst9/system01.dbf
2 ONLINE    READ WRITE      /oracle/database/tst9/rbs01.dbf
3 ONLINE    READ WRITE      /oracle/database/tst9/temp01.dbf
4 ONLINE    READ WRITE      /oracle/database/tst9/users01.dbf
5 ONLINE    READ WRITE      /oracle/database/tst9/tools01.dbf
6 ONLINE    READ WRITE      /oracle/database/tst9/data01.dbf
7 ONLINE    READ WRITE      /oracle/database/tst9/indx01.dbf
7 rows selected.
```

7. Perform an incomplete recovery by using the UNTIL CANCEL clause in the RECOVER DATABASE command.

```
SQL> recover database until cancel;
```

8. Open the database with the RESETLOGS option.

```
SQL> alter database open resetlogs;
```

You must use the RESETLOGS clause with the ALTER DATABASE OPEN command for all types of incomplete recovery. The RESETLOGS option ensures that the log files applied in recovery can never be used again by resetting the log sequence and rebuilding the existing online redo logs. This process permanently deactivates all transactions that exist in the nonarchived log files so that they can never be recovered. At the same time, it resynchronizes log files with the data files and control files. If these transactions were not purged, the log files would create bad archived logs. This is the main reason why a backup of the control file, data files, and redo logs should be done prior to performing an incomplete recovery.

9. Perform a new cold or hot backup of the database. Existing backups are no longer valid.

NOTE: Remember, after the ALTER DATABASE OPEN RESETLOGS command is applied, the previous log files and backed-up data files are useless for this newly recovered database. This is because a gap exists in the log files. The old backup data files and logs can never again be synchronized with the database. Thus, a complete backup must be performed after an incomplete recovery of any type.

REAL WORLD SCENARIO

Using Incomplete Recovery to Move a Database

Recovery operations can be used as tools to perform activities other than the typical recovery from failure. For this reason, you need to be familiar with the backup and recovery features and capabilities associated with an incomplete recovery.

Incomplete recovery options, such as the backup control file being used in conjunction with the RECOVER DATABASE USING BACKUP CONTROLFILE UNTIL CANCEL command, can be useful when you are trying to move databases from one location to another. When you use such options, you can move databases for any purpose, such as moving a database for testing, or just moving a database to a new server. You must make sure that if you are moving a database to a new server, the Oracle database software you are using and the OS on the new server are similar.

This approach of moving databases is performed by taking the hot or cold backup of the

database you want to move and moving the data files and initialization files to the new location. Then you would change the backup control file to location references of all the physical database files, such as redo logs and data files. After you have done this, you need to validate your ORACLE_SID and make sure that it is sourced to the correct database; you will then need to execute the backup control file at the SQL prompt as sysdba. This will generate a new database on a new server and in different locations.

Please refer to the Oracle documentation for the exact steps to perform this task and always try this process in a test environment first. As a DBA, you will be responsible for setting up test database environments for numerous reasons. You will find that the ability to move and set up databases and applications on different servers for testing and upgrade purposes is a must-have skill. Every time there is any significant upgrade, it will need to be tested on a different environment than the production environment. This approach of moving data files and re-creating the control file works well for many testing and upgrading situations.

2.2. Time-Based Recovery

In *time-based recovery*, the DBA recovers the database to a point in time before the point of failure. Time-based recovery provides more flexibility and control than the cancel-based option does. The cancel-based option's granularity is the size of a redo log file; in other words, when you are applying a redo log file, you get all the transactions in that file, regardless of the time period over which that log was filled.

In time-based recovery, you apply archived logs to the database up to a designated point in time. This point could be in the middle of the archived log, but not necessarily apply to the whole archived log. After you have applied these logs, you can control the recovery process to a time prior to a fatal action in the database, such as data block corruption or the loss of a database object due to user error. Below is a sample of the time-based, incomplete recovery.

```
SQL> recover database until time '2001-9-30:22:55:00';
```

You can use the preceding example to restore lost data files and then use time-based recovery in place of cancel-based recovery. To do this, you restore all the necessary data files from a hot backup, as before. The only change is that you use the UNTIL TIME clause in step 7 instead of an UNTIL CANCEL clause, as shown here:

7. Perform an incomplete recovery by using the UNTIL TIME clause.

```
SQL> recover database until time '2001-9-30:22:55:00';
```

All other steps remain the same.

2.3. Change-Based Recovery

In *change-based recovery*, you recover to a system change number (SCN) before the point of failure. This type of incomplete recovery gives you the most control.

As you have already learned, the SCN is what Oracle uses to uniquely identify each committed transaction. The SCN is a number that orders the transactions consecutively in the

redo logs as each transaction occurs. This number is also recorded in transaction tables within the rollback segments, control files, and data file headers. The SCN coordination between the transactions and these files synchronizes the database to a consistent state.

Each redo log is associated with a low and high SCN number. This SCN information can be seen in the V$LOG_HISTORY view below. Notice the low and high SCN numbers in the FIRST_CHANGE# and NEXT_CHANGE# columns for each log sequence or log file.

```
SQLWKS> select sequence#,first_change#,next_change#,first_time
2> from v$log_history where sequence# > 10326;
SEQUENCE# FIRST_CHAN  NEXT_CHANG  FIRST_TIME
---------- ----------  ----------  --------------------
10327      60731807    60732514    30-SEP-01
10328      60732514    60732848    30-SEP-01
10329      60732848    60747780    30-SEP-01
10330      60747780    60748140    30-SEP-01
4 rows selected.
```

All transactions between these SCNs are included in these logs. Oracle determines what should be recovered by using the SCN information that is recorded in transaction tables within the rollback segments, control files, and data file headers.

To perform a change-based recovery, you can use the previous example of incomplete database recovery but utilize change-based recovery in place of cancel-based or time-based recovery. To do this, restore all the needed data files from a hot backup, and then just use the following step 7 instead of the one shown previously.

7. Perform an incomplete recovery by using the UNTIL CHANGE clause.

```
SQL> recover database until change 60747681;
```

3. Recovering after Losing Current Redo Logs

Incomplete recovery is necessary if there is a loss of the current and/or inactive nonarchived redo log files. If this scenario occurs, it means that you don't have all the redo log files up to the point of failure, so the only alternative is to recover prior to the point of failure.

> *NOTE*: Oracle has made improvements to compensate for this failure by giving you the ability to mirror copies of redo logs or to create group members on different filesystems.

Some common error messages that might be seen in the alert log are ORA-00255, ORA-00312, ORA-00286, and ORA-00334. Each of these error messages indicates a problem writing to the online redo log files.

To perform incomplete recovery after the redo log files have been lost, you do the following:

1. Start Server Manager and execute a CONNECT INTERNAL command.

```
SQL> connect / as sysdba ;
```

2. Execute a SHUTDOWN command and copy all data files.

```
SQL>shutdown;
cp /stage/data01.dbf /oracle/database/tst9/data01.dbf
cp /stage/system01.dbf /oracle/database/tst9/system01.dbf
cp /stage/rbs01.dbf /oracle/database/tst9/rbs01.dbf
cp /stage/temp01.dbf /oracle/database/tst9/temp01.dbf
cp /stage/users01.dbf /oracle/database/tst9/users01.dbf
cp /stage/tools01.dbf /oracle/database/tst9/tools01.dbf
cp /stage/indx01.dbf /oracle/database/tst9/indx01.dbf
```

3. Execute a STARTUP MOUNT command to read the contents of the control file.

```
SQL> startup mount;
```

4. Execute a RECOVER DATABASE UNTIL CANCEL command to start the recovery process.

```
SQL> recover database until cancel;
```

5. Apply the necessary archived logs up to, but not including, the lost or corrupted log.

6. Open the database and reset the log files.

```
SQL> alter database open resetlogs;
```

7. Switch the log files to see whether the new logs are working.

```
SQL> alter system switch logfile;
```

8. Shut down the database.

```
SQL> shutdown normal;
```

9. Execute STARTUP and SHUTDOWN NORMAL commands to validate that the database is functional by making sure that the alert logs after these commands are executed.

```
SQL> startup;
SQL> shutdown normal;
```

10. Perform a cold backup or hot backup.

NOTE: If you need to recover archived logs from a different location, you can just change the LOG_ARCHIVE_DEST location to the new location of the archived log files. This may occur in a recovery situation in which you recover your archived logs from tape to a staging location on disk.

4. Performing an RMAN-Based Incomplete Recovery Using UNTIL TIME

In this example you will perform one type of incomplete recovery using RMAN. You will recover the database to a particular point in time. To do so, you will create a user, called TEST, and two tables with date and time data stored in them. You will then perform a database backup in ARCHIVELOG mode and recover to a time between 2:31, when the data was stored in the first table, and 3:59, when the data was stored in the second table. You accomplish all of this with the SET UNTIL TIME clause in RMAN; this clause is required to perform incomplete recovery. Thus, when the database is recovered, you should not see the second table's data. When the recovery is completed and validated, you will need to register the database in the recovery catalog. Let's walk through this example:

1. Source your ORACLE_SID to tst9, which is your target database, so that the database can be started in MOUNT or OPENED mode with SQL.

```
oracle@octilli:~ > . oraenv
ORACLE_SID = [tst9] tst9
oracle@octilli:~ >

SQL> startup mount
```

```
Oracle instance started
database mounted
Total System Global Area 75854976 bytes
Fixed Size 279680 bytes
Variable Size 71303168 bytes
Database Buffers 4194304 bytes
```

Or

```
SQL> startup mount
ORACLE instance started.
Total System Global Area 75854976 bytes
Fixed Size 279680 bytes
Variable Size 71303168 bytes
Database Buffers 4194304 bytes
Redo Buffers 77824 bytes
Database mounted.
SQL> alter database open;
```

2. Connect to RMAN, the target database, and the recovery catalog in one step.

```
oracle@octilli:~ > rman target / catalog rman/rman@rcat
Recovery Manager: Release 9.0.1.0.0 - Production
(c) Copyright 2001 Oracle Corporation. All rights
reserved.
connected to target database: TST9 (DBID=1268700551)
connected to recovery catalog database
RMAN>
```

3. Create a user TEST and the two tables, which will be used throughout this example. Data will be added to the first table. The results of this data insertion can be seen in the SELECT statement.

```
SQL> create user test identified by test
```

```
   2> default tablespace users
   3> temporary tablespace temp;
Statement processed.
SQL> grant connect,resource to test;
Statement processed.
SQL> connect test/test
Connected.
SQL> create table t1 (c1 number, c2 char(50));
Statement processed.
SQL>insert into t1 values (1, to_char(sysdate, 'HH:MI DD-MON-YYYY'));
SQL>commit;
SQL> create table t2 (c1 number, c2 char(50));
Statement processed.
SQL> connect system/manager
SQL> alter system switch logfile;
Statement processed.
SQL> select * from t1;
C1      C2
------- --------------------------------------------
1       02:31 04-OCT-2001
1 row selected.
```

4. Back up the database.

```
RMAN> run {
2> allocate channel ch1 type disk;
3> backup database;
4> }

allocated channel: ch1
channel ch1: sid=10 devtype=DISK
```

```
Starting backup at 04-OCT-01
channel ch1: starting full datafile backupset
channel ch1: specifying datafile(s) in backupset
including current controlfile in backupset
input datafile fno=00001 name=/db01/oracle/tst9/system01.dbf
input datafile fno=00006 name=/db01/oracle/tst9/data01.dbf
input datafile fno=00002 name=/db01/oracle/tst9/rbs01.dbf
input datafile fno=00003 name=/db01/oracle/tst9/temp01.dbf
input datafile fno=00004 name=/db01/oracle/tst9/users01.dbf
input datafile fno=00007 name=/db01/oracle/tst9/indx01.dbf
input datafile fno=00005 name=/db01/oracle/tst9/tools01.dbf
channel ch1: starting piece 1 at 04-OCT-01
piece handle=/oracle/product/9.0.1/dbs/02d5p88p_1_1
comment=NONE
channel ch1: backup set complete, elapsed time:
00:01:49
Finished backup at 04-OCT-01
```

5. Back up all archived log files.

```
RMAN> run {
2> allocate channel ch1 type disk;
3> backup
4> format 'log_t%t_s%s_p%p'
5> (archivelog all);
6> }
```

6. Create the second table, t2, and add the date-time data to the table. This date-time is the same day, but at 3:59 in the afternoon. Assume that you have run some log switches to move the data to the archived logs.

```
SQL> connect test/test
Connected.
```

```
SQL>insert into t2 values (2, to_char(sysdate, 'HH:MI DD-MON-YYYY'));
1 row processed.
SQL>commit;
SQL> select * from t2;
C1       C2
-------- -----------------------------------------
1        03:59 04-OCT-2001
1 row selected.
SQL> connect system/manager
SQL> alter system switch logfile;
SQL> alter system switch logfile;
```

7. Back up the archived logs to disk with the appropriate RMAN script.

```
RMAN> run {
2> allocate channel ch1 type disk;
3> backup
4> format '/oracle/backups/log_t%t_s%s_p%p'
5> (archivelog all);
6> }
```

8. Restore the database to a point in time between 2:31 and 3:59. Then validate that you do not see the second table.

```
RMAN> run {
allocate channel ch1 type disk;
set until time 'OCT 04 2001 15:58:00';
restore database;
recover database;
sql 'alter database open resetlogs';
}

SQL> select * from t1;
```

```
C1       C2
-------- ----------------------------------------
1        02:31 04-OCT-2001
1 row selected.
SQL> select * from t2;
No rows selected.
SQL>
```

9. Once the database has been restored, it should be shut down normally. Then you should perform a startup to make sure the restore process was completed successfully.

```
SQL> shutdown
SQL> startup
```

10. Once the database has been validated, the database should be reregistered in the RMAN catalog. This must done every time there is an incomplete recovery and the SQL 'ALTER DATABASE OPEN RESETLOGS'; command is performed. To reregister the database, you must be connected to the target database and the recovery catalog.

```
oracle@octilli:~ > rman target / catalog rman/rman@rcat
Recovery Manager: Release 9.0.1.0.0 - Production
(c) Copyright 2001 Oracle Corporation. All rights reserved.
connected to target database: TST9 (DBID=1268700551)
connected to recovery catalog database
RMAN> reset database;
database registered in recovery catalog
starting full resync of recovery catalog
full resync complete
RMAN>
```

NOTE: A complete backup should be performed on any database that is opened with the RESETLOGS options because all previous archived logs are invalid with databases opened with these options.

5. Performing RMAN-Based Incomplete Recovery Using UNTIL SEQUENCE

In this example, you will perform another type of incomplete recovery using RMAN. You will recover the database to a particular log sequence number. To do this, you will need to use the SET UNTIL [LOGSEQ/SEQUENCE/SCN] clause to perform an incomplete recovery until a sequence point. In this example, you will recover to the log sequence just prior to a corrupt online redo log. When the recovery is completed and validated, you will need to register the database in the recovery catalog. Let's walk through this example:

1. Source ORACLE_SID to tst9, which is your target database, so that the database can be started in MOUNT or OPENED mode with SQL.

```
oracle@octilli:~ > . oraenv
ORACLE_SID = [tst9] tst9
oracle@octilli:~ >
SQL> startup mount
Oracle instance started
database mounted
Total System Global Area 75854976 bytes
Fixed Size 279680 bytes
Variable Size 71303168 bytes
Database Buffers 4194304 bytes
```

Or

```
SQL> startup mount
ORACLE instance started.
Total System Global Area 75854976 bytes
Fixed Size 279680 bytes
Variable Size 71303168 bytes
Database Buffers 4194304 bytes
Redo Buffers 77824 bytes
Database mounted.
SQL> alter database open;
```

2. Connect to RMAN, the target database, and the recovery catalog in one step.

```
oracle@octilli:~ > rman target / catalog rman/rman@rcat
Recovery Manager: Release 9.0.1.0.0 - Production
(c) Copyright 2001 Oracle Corporation. All rights
reserved.
connected to target database: TST9 (DBID=1268700551)
connected to recovery catalog database
RMAN>
```

3. Create a user TEST and the two tables, which will be used throughout this example. Data will be added to the first table. The results of this data insertion can be seen in the SELECT statement.

```
oracle@octilli:~ > sqlplus /nolog
SQL*Plus: Release 9.0.1.0.0 - Production on Fri Oct 5 00:33:53 2001
(c) Copyright 2001 Oracle Corporation. All rights reserved.
SQL> connect /as sysdba
Connected.
SQL> archive log list;
Database log mode Archive Mode
Automatic archival Enabled
Archive destination /oracle/admin/tst9/arch
Oldest online log sequence 262
Next log sequence to archive 264
Current log sequence 264
```

4. Back up the database and archive the log files.

```
RMAN> run {
2> allocate channel ch1 type disk;
3> backup database;
4> backup (archivelog all);
5> }
```

```
allocated channel: ch1
channel ch1: sid=9 devtype=DISK

Starting backup at 05-OCT-01
channel ch1: starting full datafile backupset
channel ch1: specifying datafile(s) in backupset
including current controlfile in backupset
input datafile fno=00001 name=/db01/oracle/tst9/system01.dbf
input datafile fno=00006 name=/db01/oracle/tst9/data01.dbf
input datafile fno=00002 name=/db01/oracle/tst9/rbs01.dbf
input datafile fno=00003 name=/db01/oracle/tst9/temp01.dbf
input datafile fno=00004 name=/db01/oracle/tst9/users01.dbf
input datafile fno=00007 name=/db01/oracle/tst9/indx01.dbf
input datafile fno=00005 name=/db01/oracle/tst9/tools01.dbf
channel ch1: starting piece 1 at 05-OCT-01
channel ch1: finished piece 1 at 05-OCT-01
piece handle=/oracle/product/9.0.1/dbs/05d5petc_1_1
comment=NONE
channel ch1: backup set complete, elapsed time:
00:01:48
Finished backup at 05-OCT-01
Starting backup at 05-OCT-01
current log archived
channel ch1: starting archive log backupset
channel ch1: specifying archive log(s) in backup set
input archive log thread=1 sequence=257 recid=1
stamp=442278586
input archive log thread=1 sequence=258 recid=2
stamp=442278595
```

```
input archive log thread=1 sequence=259 recid=3
stamp=442278598
input archive log thread=1 sequence=260 recid=4
stamp=442278603
input archive log thread=1 sequence=261 recid=5
stamp=442278607
input archive log thread=1 sequence=262 recid=6
stamp=442284653
input archive log thread=1 sequence=263 recid=7
stamp=442285081
channel ch1: starting piece 1 at 05-OCT-01
channel ch1: finished piece 1 at 05-OCT-01
piece handle=/oracle/product/9.0.1/dbs/06d5pf0q_1_1
comment=NONE
channel ch1: backup set complete, elapsed time:
00:00:01
Finished backup at 05-OCT-01
released channel: ch1
RMAN>
```

5. In this case, there is a corrupt online redo log sequence number 264, so you will need to recover to log sequence number 263.

```
RMAN> run {
2> allocate channel ch1 type disk;
3> set until logseq=263 thread=1;
4> restore database;
5> recover database;
6> sql "alter database open resetlogs";
7> }

allocated channel: ch1
```

```
channel ch1: sid=11 devtype=DISK
executing command: SET until clause

Starting restore at 05-OCT-01
channel ch1: starting datafile backupset restore
channel ch1: specifying datafile(s) to restore from
backup set
restoring datafile 00001 to /db01/oracle/tst9/system01.dbf
restoring datafile 00002 to /db01/oracle/tst9/rbs01.dbf
restoring datafile 00003 to /db01/oracle/tst9/temp01.dbf
restoring datafile 00004 to /db01/oracle/tst9/users01.dbf
restoring datafile 00005 to /db01/oracle/tst9/tools01.dbf
restoring datafile 00006 to /db01/oracle/tst9/data01.dbf
restoring datafile 00007 to /db01/oracle/tst9/indx01.dbf
channel ch1: restored backup piece 1
piece handle=/oracle/product/9.0.1/dbs/04d5peg1_1_1
tag=null params=NULL
channel ch1: restore complete
Finished restore at 05-OCT-01
Starting recover at 05-OCT-01
starting media recovery
archive log thread 1 sequence 262 is already on disk as
   file /oracle/admin/tst9/arch/archtst9_262.log
archive log filename=/oracle/admin/tst9/arch/archtst9_
262.log thread=1 sequence2
media recovery complete
Finished recover at 05-OCT-01
sql statement: alter database open resetlogs
released channel: ch1
RMAN>
```

6. Next, you can validate that the database was opened and the logs were reset, thus recovering before the corrupt log file sequence 263.

```
SQL> archive log list
Database log mode Archive Mode
Automatic archival Enabled
Archive destination /oracle/admin/tst9/arch
Oldest online log sequence 0
Next log sequence to archive 1
Current log sequence 1
SQL>
```

7. Once the database has been restored, shut it down normally. Then perform a startup to make sure that the restore process was completed successfully.

```
SQL> shutdown
SQL> startup
```

8. Once the database has been validated, it should be reregistered in the RMAN catalog with the RESET DATABASE command. This must be done every time there is an incomplete recovery and the SQL 'ALTER DATABASE OPEN RESETLOGS'; command is performed. You must be connected to the target database and the recovery catalog.

```
oracle@octilli:~ > rman target / catalog rman/rman@rcat
Recovery Manager: Release 9.0.1.0.0 - Production
(c) Copyright 2001 Oracle Corporation. All rights reserved.
connected to target database: TST9 (DBID=1268700551)
connected to recovery catalog database
RMAN> reset database;
database registered in recovery catalog
starting full resync of recovery catalog
full resync complete
RMAN>
```

6. Summary

Incomplete recovery allows you to recover a database to before the point where the database failed, or to the last available transaction at the time of the failure. In other words, as a result of this type of recovery, the recovered database is missing transactions or is incomplete.

You might need to perform incomplete database recovery for various reasons. The type of failure that requires such a recovery might be in the database, such as a corruption error or a dropped database object. This means that recovery would need to stop short of using all the archived logs that are available to be applied. If it didn't, the failure could be reintroduced to the database as the transactions were being read from archived redo logs. Another situation that might require an incomplete recovery is one in which your database has lost the current redo log files. Again, the reason this can only be solved with an incomplete recovery is because not all the previous transactions are available. At least one online log file is corrupted or lost.

The RMAN incomplete recovery process was also demonstrated with examples that used the SET UNTIL TIME and SEQUENCE commands. Incomplete recovery is a key component for certain types of failure; as a result, to be properly prepared for the test you must understand this concept. In the workplace, you can use these concepts in routine maintenance activities like those in which you may be required to move databases from one server to another.

References

[1] Oracle9i DBA Fundamentals II.