12. User-managed and RMAN-based backups.

Abstract: A *physical backup* is a copy of the physical database files, and it can be performed in two ways. The first is through the Recovery Manager (RMAN) tool that Oracle provides. The second way is by performing a user-managed, or non-RMAN, backup. This lesson focuses on both types of backup. The user-managed backup has been used for years to back up the Oracle database. The OS backup script is a totally customized solution, and therefore, it has the variations and inconsistencies associated with custom solutions. This backup usually consists of an OS backup created with a scripting language or batch commands. Even though the user-managed backup has been historically helpful, the current trend shows that most larger database sites are now using RMAN to conduct their backups. The reason for this is because of RMAN's extended capabilities and because it can be used consistently regardless of platform. However, the OS backup is still useful to the DBA. You can use it to train yourself so that you understand the physical backup fundamentals. Further, many storage subsystem providers still utilize user-managed backups in conjunction with third mirror capabilities to expedite large backups in a short period of time. After you have learned as much as you can from the user-managed backup, move on to RMAN, which builds on these fundamentals. In this lesson, you will learn the various physical backup methods with both user-managed backups and RMAN backups.

Contents

Objectives:

- Describe user-managed backup and recovery operations.
- Discuss backup issues associated with read tablespaces.
- Perform closed database backups.
- Perform open database backups.
- Back up the control file.
- Perform cleanup after a failed online backup.
- Use the DBVERIFY utility to detect corruption.
- Identify types of RMAN specific backups.
- …

1. User-Managed Backup and Recovery Operations

As mentioned, different sites customize their user-managed backups and recovery operations to suit different requirements. This customization is possible because this type of backup is generally a script written in a Unix shell or using Windows NT/2000/XP batch commands. As a result, these user-managed backups and recovery operations must be managed as custom code, and significant testing must be conducted to validate their functionality.

This description shows both the benefits and drawbacks of using this type of backup. Though this ability to customize can allow user-managed backups to be designed for unique situations in addition to significant testing and validation, such customizations could cause errors that could invalidate the entire backup or recovery process.

Despite such possible side effects, user-managed backups have been in use for many years in the Oracle environment. In addition to their broad usage, these backups also provide the building blocks you need to understand the entire Oracle backup and recovery process (including RMAN). This is why every DBA should be comfortable with user-managed backup and recovery operations. This knowledge will allow them to make the correct decisions in a failure situation whether the site is using user-managed or RMAN-based backup and recovery.

2. Working with Read-Only Tablespaces

The backup and recovery of a *read-only tablespace* requires unique procedures in certain situations. The backup and recovery process changes depending on the state of the tablespace at the time of backup and the time of recovery, and the state of the control file. This section discusses the implications of each of these situations.

A backup of a *read-only tablespace* requires different procedures from those used in a backup of a *read-write tablespace*. The read-only tablespace is, as its name suggests, marked read-

only; in other words, all write activity is disabled. Therefore, the system change number (SCN) does not change after the tablespace has been made read-only, as long as it stays read-only. This means that after a database failure, no recovery is needed for a tablespace marked read-only if the tablespace was read-only at the time of the backup.

The read-only tablespace could simply be restored and no archived logs would get applied during the recovery process. If a backup is restored that contains a tablespace that was in read-write mode at the time of the backup but is in read-only at the time of failure, then a recovery would need to be performed. This is because changes would be made during read-write mode. Archived logs would be applied up until the tablespace was made read-only.

The state of the control file also affects the recovery process of read-only tablespaces. During recovery of a backup control file, or recovery when there is no current control file, read-only tablespaces should be taken offline or you will get an ORA-1233 error. The control file cannot be created with a read-only tablespace online. The data file or data files associated with the read-only tablespace must be taken offline before the recovery command is issued. After the database is recovered, the read-only tablespace can be brought online. We will look at examples of the recovery of read-only tablespace in the next lesson.

3. Understanding Closed and Opened Database Backups

A *closed backup* is a backup of a database that is not in the opened state. Usually this means that the database is completely shut down. This kind of backup is also called a cold, or offline, backup. An *opened backup* is a backup of a database in the opened state. In this state, the database is completely available for access. An opened backup is also called a hot, or online, backup.

The main implication of a closed backup is that the database is unavailable to users until the backup is complete. One of the preferable prerequisites of doing a cold backup is that the database should be shut down with the NORMAL or IMMEDIATE options so that the database is in a consistent state. In other words, the database files are stamped with the same SCN at the same point in time. This is called a *consistent backup*. Because no recovery information in the archived logs needs to be applied to the data files, no recovery is necessary for a consistent, closed backup. Figure 1 is an example of a closed backup.

The main implication of an opened backup is that the database is available to users during the backup. The backup is accomplished with the command ALTER TABLESPACE <TABLESPACE_NAME> BEGIN BACKUP, an OS copy command, and the command ALTER TABLEPACE <TABLESPACE_NAME> END BACKUP. Figure 2 is an example of an opened backup.

During an opened backup, the database is in an inconsistent state; in other words, the SCN information for the data files and control files is not necessarily consistent. Therefore, this is referred to as an *inconsistent backup*.

This requires recovery of the data files by applying archived logs to bring the data files to a consistent state.

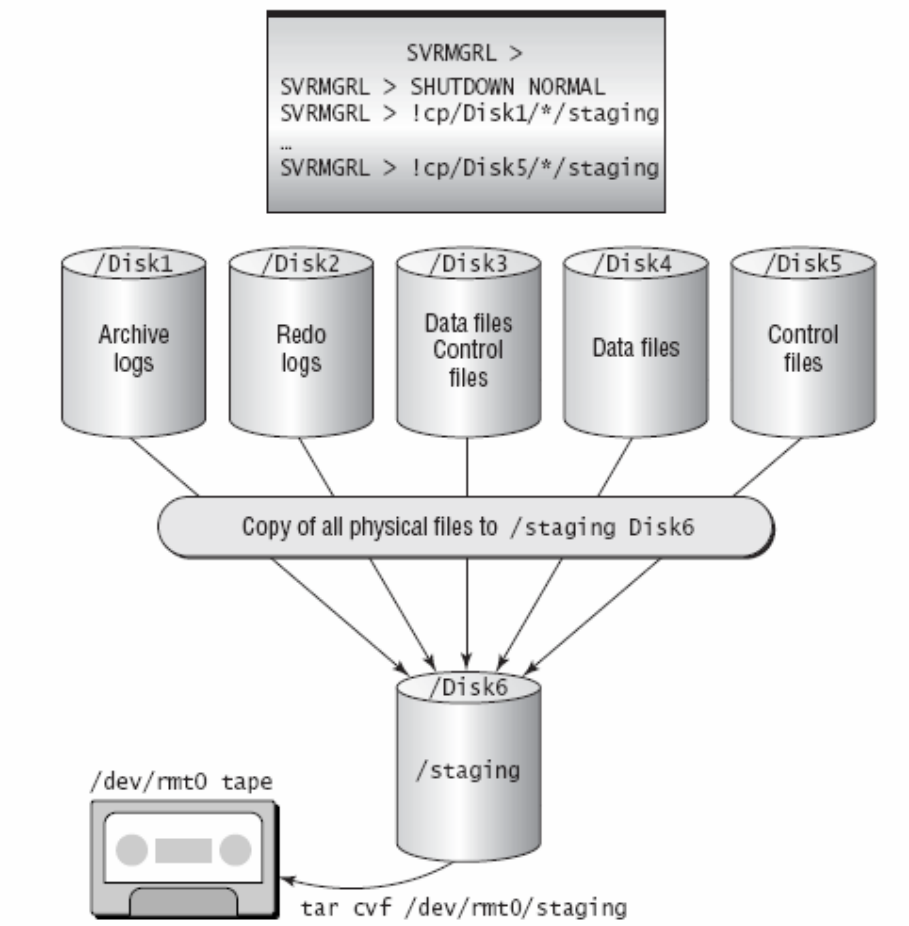

FIGURE 1. Physical backup utilizing the cold, offline, or closed backup approach in Unix.

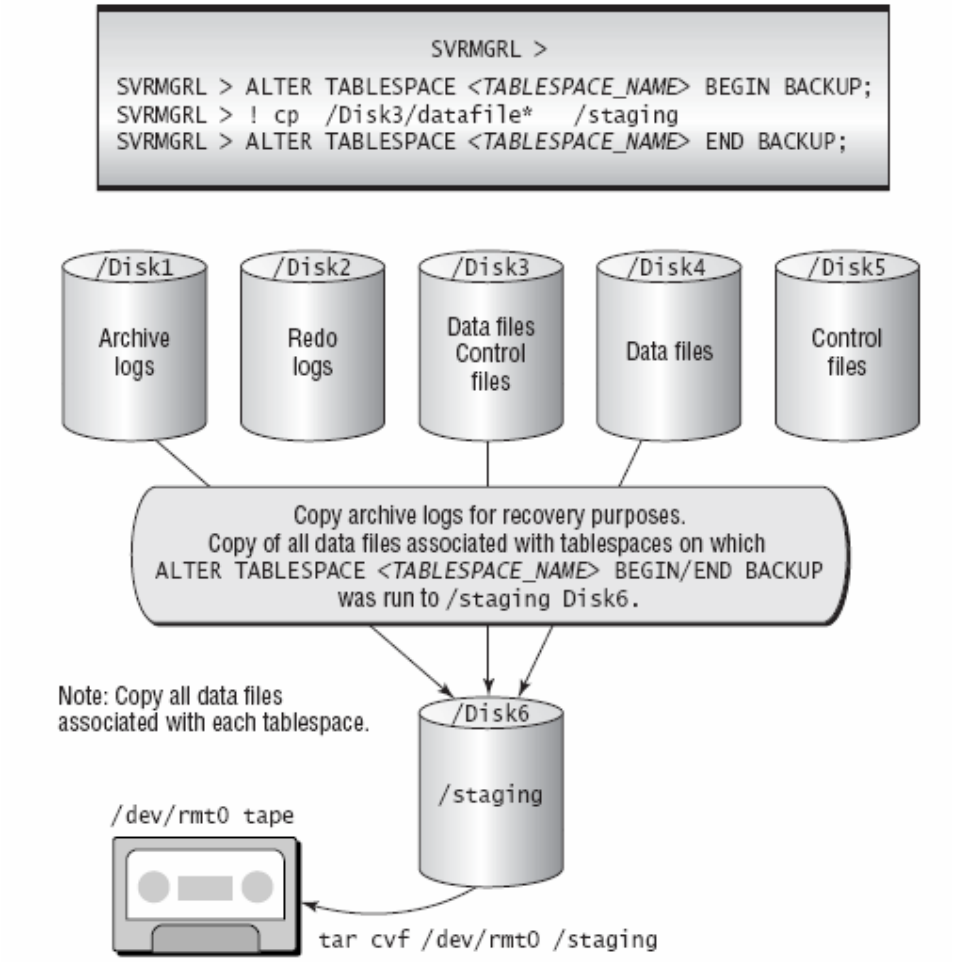

FIGURE 2. Physical backup utilizing the hot, online, or opened backup approach in Unix.

4. Performing Closed and Opened Database Backups

A closed backup and *opened backup* are performed in a similar manner — by executing OS copy commands. The closed backup can be performed just like a standard OS backup after the database has been shut down. Basically, the closed backup just makes a copy of all the necessary physical files that make up the database; these include the data files, online redo logs, control files, and parameter files.

The opened backup is also executed by issuing OS copy commands. These commands are issued between an ALTER TABLESPACE <TABLESPACE_NAME> BEGIN BACKUP command and an ALTER TABLESPACE <TABLESPACE_NAME> END BACKUP command. The opened backup requires only a copy of the data files. The ALTER TABLESPACE <TABLESPACE_NAME> BEGIN BACKUP command causes a checkpoint to the data file or data files in the tablespace. This causes all dirty blocks to be written to the data file, and the data-file header is stamped with the SCN consistent with those data blocks. All other changes occurring in the data file from Data Manipulation Language (DML), such

as INSERT, UPDATE, and DELETE statements, get recorded in the data files and redo logs in almost the same way as during normal database operations. However, the data-file header SCN does not get updated until the ALTER TABLESPACE <TABLESPACE_NAME> END BACKUP command gets executed and another checkpoint occurs. To distinguish what blocks are needed in a recovery situation, Oracle writes more information in the redo logs during the period that the ALTER TABLESPACE <TABLESPACE_NAME> BEGIN BACKUP command is executed.

This is one reason why data files are fundamentally different from other Oracle database files, such as redo logs, control files, and init.ora files, when it comes to backups. Redo logs, control files, and init.ora files can be copied with standard OS copy commands without performing any preparatory steps such as the ALTER TABLESPACE <TABLESPACE_NAME> commands. This is completely true in the Unix environment. This is partially true in the Windows NT/2000/XP environment because of the locking that is performed on open files.

NOTE: Even though hot backups can be performed at any time when the database is opened, it is a good idea to perform hot backups when there is the lowest DML activity. This will prevent excessive redo logging, which could impair database performance.

Here are the steps you need to take to perform a closed database backup:

1. Shut down the database that you want to back up. Make sure that a SHUTDOWN NORMAL, IMMEDIATE, or TRANSACTIONAL command is used, and not a SHUTDOWN ABORT.

```
oracle@octilli:/opt/oracle > sqlplus /nolog
SQL*Plus: Release 9.0.1.0.0 - Production on Sat Sep 29 00:11:37 2001
(c) Copyright 2001 Oracle Corporation. All rights reserved.
SQL> connect /as sysdba
Connected.
Database closed.
Database dismounted.
ORACLE instance shut down.
```

2. Once the database is shut down, perform an OS copy of all the data files, parameter files, and control files to a disk or a tape device. In Unix, perform the following commands to copy the data files, parameter files, and control files to a disk staging location where they await copy to tape.

```
cp /oracle/product/9.0.1/oradata/*.dbf /staging/cold
        # datafiles
cp /oracle/admin/orc9/pfile/* /staging/cold
        # INIT.ora files
cp /oracle/oradata/*.ctl /staging/cold
        # control files location 1
```

```
cp /oracle/product/9.0.1/oradata/*.ctl /staging/cold
        # control files location 2
cp /oracle/oradata/orc9/*.log /staging/cold
        # online redo logs group 1
```

3. Restart the database and proceed with normal database operations.

```
SQL> startup;
```

Here are the steps you would use to perform an opened database backup. The database is available to users during these operations, although the response time for users may be decreased depending on what tablespace is being backed up.

1. To determine all the tablespaces that make up the database, query the V$TABLESPACE and V$DATAFILE dynamic views. The following is the SQL statement that identifies tablespaces and their associated data files.

```
select a.TS#,a.NAME,b.NAME from v$tablespace a, v$datafile b where
    a.TS#=b.TS#;
```

2. Determine what all the data files are that make up each tablespace. Each tablespace can be made up of many data files. All data files associated with the tablespace need to be copied when the tablespace is in backup mode. Perform the above query by connecting to SQLPLUS.

```
oracle@octilli:~ > sqlplus /nolog
SQL*Plus: Release 9.0.1.0.0 - Production on Sat Sep 29 10:44:22 2001
(c) Copyright 2001 Oracle Corporation. All rights reserved.
SQL>
SQL> connect /as sysdba
Connected.
SQL> select a.TS#,a.NAME,b.NAME from v$tablespace a, v$datafile b
  where a.TS#=b.TS#;
TS#   NAME        NAME
----- ---------- -------------------------------------------------
0     SYSTEM     /oracle/product/9.0.1/oradata/orc9/system01.dbf
1     UNDOTBS    /oracle/product/9.0.1/oradata/orc9/undotbs01.dbf
2     CWMLITE    /oracle/product/9.0.1/oradata/orc9/cwmlite01.dbf
3     DRSYS      /oracle/product/9.0.1/oradata/orc9/drsys01.dbf
4     EXAMPLE    /oracle/product/9.0.1/oradata/orc9/example01.dbf
5     INDX       /oracle/product/9.0.1/oradata/orc9/indx01.dbf
7     TOOLS      /oracle/product/9.0.1/oradata/orc9/tools01.dbf
8     USERS      /oracle/product/9.0.1/oradata/orc9/users01.dbf
```

3. Put the tablespaces in backup mode.

```
SQL> alter tablespace users begin backup;
Tablespace altered.
```

4. Perform an OS copy of each data file associated with the tablespace in backup mode.

```
SQL> ! cp /oracle/product/9.0.1/oradata/orc9/users01.dbf /staging/cold
```

5. End backup mode for the tablespace.

```
SQL> alter tablespace users end backup;
Tablespace altered.
```

These series of commands can be repeated for every tablespace and associated data file that make up the database. The database must be in ARCHIVELOG mode to execute the ALTER TABLESPACE <TABLESPACE_NAME> BEGIN and END backup commands. Typically, this type of backup is done with a scripting language in Unix or a third-party GUI utility in the Windows environment. Using Unix shell scripts, a list of data files and tablespaces are dumped to a file listing and parsed into svrmgr and cp commands so that all tablespaces and data files are backed up together.

NOTE: During an opened or closed backup, it is a good idea to get a backup control file, all archived logs, and a copy of the parameter files. These can be packaged with data files so that all necessary or potentially necessary components for recovery are grouped together. This is called a *whole database backup*.

5. Identifying User-Managed Control-File Backups

There are two types of user-managed control-file backups. The first type is performed by executing a command that creates a binary copy of the existing control file in a new directory location. For example, the following command performs a binary copy of the control file.

```
SQL> alter database backup controlfile to '/staging/control.ctl.bak';
```

The second type of control-file backup creates an ASCII copy of the current control file as a trace file in the USER_DUMP_DEST location. The USER_DUMP_DEST parameter should be set in your init.ora file. In a configuration compliant with Optimal Flexible Architecture (OFA), this will be the udump directory. The backup of the control file can be performed by executing the following command:

```
SQL> alter database backup controlfile to trace;
```

The output of the trace file looks like this:

```
/oracle/admin/orc9/udump/ora_4976.trc
Oracle9i Enterprise Edition Release 9.0.1.0.0 – Production
With the Partitioning option
JServer Release 9.0.1.0.0 - Production
ORACLE_HOME = /oracle/product/9.0.1
System name: Linux
Node name: octilli
Release: 2.4.4-4GB
Version: #1 Fri May 18 14:11:12 GMT 2001
Machine: i686
Instance name: orc9
Redo thread mounted by this instance: 1
Oracle process number: 13
Unix process pid: 4976, image: oracle@octilli (TNS V1-V3)

*** SESSION ID:(8.7) 2001-09-29 00:29:20.499
*** 2001-09-29 00:29:20.499
# The following commands will create a new control file and use it
# to open the database.
# Data used by the recovery manager will be lost.
# Additional logs may be required for media recovery of offline data files.
# Use this only if the current version of all online logs are available.
STARTUP NOMOUNT
CREATE CONTROLFILE REUSE DATABASE "ORC9" NORESETLOGS
ARCHIVELOG
        MAXLOGFILES 50
        MAXLOGMEMBERS 5
        MAXDATAFILES 100
        MAXINSTANCES 1
        MAXLOGHISTORY 226
```

```
LOGFILE
        GROUP 1 '/oracle/oradata/orc9/redo01.log' SIZE 100M,
        GROUP 2 '/oracle/oradata/orc9/redo02.log' SIZE 100M,
        GROUP 3 '/oracle/oradata/orc9/redo03.log' SIZE 100M
# STANDBY LOGFILE
DATAFILE
        '/oracle/product/9.0.1/oradata/orc9/system01.dbf',
        '/oracle/product/9.0.1/oradata/orc9/undotbs01.dbf',
        '/oracle/product/9.0.1/oradata/orc9/cwmlite01.dbf',
        '/oracle/product/9.0.1/oradata/orc9/drsys01.dbf',
        '/oracle/product/9.0.1/oradata/orc9/example01.dbf',
        '/oracle/product/9.0.1/oradata/orc9/indx01.dbf',
        '/oracle/product/9.0.1/oradata/orc9/tools01.dbf',
        '/oracle/product/9.0.1/oradata/orc9/users01.dbf'
CHARACTER SET WE8ISO8859P1
;
# Recovery is required if any of the datafiles are restored backups,
# or if the last shutdown was not normal or immediate.
RECOVER DATABASE
# All logs need archiving and a log switch is needed.
ALTER SYSTEM ARCHIVE LOG ALL;
# Database can now be opened normally.
ALTER DATABASE OPEN;
# Commands to add tempfiles to temporary tablespaces.
# Online tempfiles have complete space information.
# Other tempfiles may require adjustment.
ALTER TABLESPACE TEMP ADD TEMPFILE '/oracle/product/9.0.1/
oradata/orc9/temp01.dbf' REUSE;
# End of tempfile additions.
```

NOTE: The control-file backup to ASCII can be used as part of a common technique of moving production databases to test and development servers. This technique can be useful for testing backups.

6. Cleaning Up after Failed Online Backups

Online backups, whether they are user-managed or RMAN-based, perform the same database commands. One of these commands, the ALTER TABLESPACE <tablespace name> BEGIN BACKUP command, results in tablespaces being placed into backup mode. When the required data file(s) is completely copied, the ALTER TABLESPACE <tablespace name> END BACKUP command is then executed.

If there is a problem before the tablespace is taken out of backup mode, the tablespace may cause problems during recovery or it may lock up the next backup. If the database is shutdown with a tablespace in backup mode, the database will not start without taking the associated data files out of backup mode.

Sometimes RMAN may have problems if the next scheduled backup attempts to place the tablespace in backup mode when it is already in backup mode. This will cause the next RMAN backup to fail or hang while it is trying to place the tablespace in backup mode.

If an online backup fails, all tablespace associated data files should be checked to make sure that they are not in backup mode. This can be done by checking the V$BACKUP view. The following example shows that one data file is in backup mode. The tablespace associated with data file 4 should be taken out of backup mode. This tablespace should be identified and then altered out of backup mode with ALTER TABLESPACE <tablespace_name> END BACKUP. See the example below:

1. First, check to see if any data file is active; if one is, this means that the tablespace and its associated data files are in backup mode. In this case, data file number 4 is in backup mode because the status is ACTIVE.

```
select * from v$backup;

FILE#    STATUS          CHANGE# TIME
---------- ----------------- ---------- ---------
1        NOT ACTIVE              0
2        NOT ACTIVE              0
3        NOT ACTIVE              0
4        ACTIVE             279174 29-SEP-01
5        NOT ACTIVE              0
6        NOT ACTIVE              0
7        NOT ACTIVE              0
8        NOT ACTIVE         278815 29-SEP-01
```

```
8 rows selected.

SQL>
```

2. Next, find what tablespace is associated with data file 4 by executing the following SQL query. Note that data file 4 and tablespace 3 are associated with the DRSYS tablespace.

```
select substr(b.name,0,10) name,a.file#,a.ts#,status
from v$datafile a, v$tablespace b
where a.ts#=b.ts#
order by file#;
```

NAME	FILE#	TS#	STATUS
SYSTEM	1	0	SYSTEM
UNDOTBS	2	1	ONLINE
CWMLITE	3	2	ONLINE
DRSYS	**4**	**3**	**ONLINE**
EXAMPLE	5	4	ONLINE
INDX	6	5	ONLINE
TOOLS	7	7	ONLINE
USERS	8	8	ONLINE

```
8 rows selected.
```

3. Next, take this tablespace out of backup mode by executing the following command. Then query the V$BACKUP view again to verify that the data file status is not active.

```
alter tablespace DRSYS end backup;

Tablespace altered.

select * from v$backup;

     FILE# STATUS      CHANGE# TIME
---------- ------------------ ---------- ---------
         1 NOT ACTIVE              0
         2 NOT ACTIVE              0
```

```
3 NOT ACTIVE          0
4 NOT ACTIVE     279174 29-SEP-01
5 NOT ACTIVE          0
6 NOT ACTIVE          0
7 NOT ACTIVE          0
8 NOT ACTIVE     278815 29-SEP-01
8 rows selected.
```

REAL WORLD SCENARIO

Checking the Backup before Shutdown

Because RMAN backups called by Media Management vendor's software are conducted in the background, they tend to be forgotten. Another reason these backups may be forgotten is because rather than the DBA conducting them, such backup-related tasks may be executed and controlled by the backup coordinator, who may reside within the systems administrators group. As a result, when the backup terminates for some reason, you may not know about it, unless you have good communication set up. As a result, the backup may terminate in such a way that it leaves the database partially in backup mode with some tablespaces and the associated data files still active, or it leaves jobs incomplete and hanging in the recovery catalog.

What happens when the database goes down when a tablespace is in backup mode? If this happens, the data file is not checkpointed so that it is consistent with the rest of the database. Therefore, when the database is restarted, the data file is marked as inconsistent and in need of recovery. This situation can come as an unwanted surprise when you are bouncing the database for some reason.

You can remedy this situation without recovery by issuing the ALTER DATAFILE '<datafile name>' END BACKUP command to fix this. However, this situation can be avoided in the first place if you check the V$BACKUP view to validate that it is safe to shut down the database before you do so.

7. The DBVERIFY Utility

The Oracle *DBVERIFY utility* is executed by entering **dbv** at the command prompt. This utility has six parameters that can be specified at execution. The parameters are FILE, START, END, BLOCKSIZE, LOGFILE, and FEEDBACK. Table 1 describes these parameters.

TABLE 1. DBVERIFY Parameters.

Parameter	Description	Default Value for

		Parameter
FILE	Data file to be verified by the utility.	No default parameter
START	Starting block to begin verification.	First block in the data file
END	Ending block to end verification.	Last block in the data file
BLOCKSIZE	Block size of database. This should be the same as the init.ora parameter DB_BLOCK_SIZE.	2048
LOGFILE	Log file to store the results of running the utility.	No default parameter
FEEDBACK	Displays the progress of the utility by displaying a dot for each number of blocks processed.	0

This help information can also be seen by executing the DBV HELP=Y command. See the following example:

```
oracle@octilli:/db01/oracle/tst9 > dbv help=y

DBVERIFY: Release 9.0.1.0.0 - Production on Tue Oct 9 00:06:48 2001

(c) Copyright 2001 Oracle Corporation. All rights reserved.

Keyword         Description                    (Default)
------------------------------------------------------------------------------
FILE            File to Verify                 (NONE)
START           Start Block                    (First Block of File)
END             End Block                      (Last Block of File)
BLOCKSIZE       Logical Block Size             (2048)
LOGFILE         Output Log                     (NONE)
FEEDBACK        Display Progress                  (0)
PARFILE         Parameter File                 (NONE)
USERID          Username/Password              (NONE)
SEGMENT_ID      Segment ID (tsn.relfile.block) (NONE)
oracle@octilli:/db01/oracle/tst9 >
```

To run the DBVERIFY utility, the BLOCKSIZE parameter must match your database block size, or the following error will result:

```
oracle@octilli:/db01/oracle/tst9 > dbv file=data01.dbf

DBVERIFY: Release 9.0.1.0.0 - Production on Tue Oct 9 00:12:55 2001
```

```
(c) Copyright 2001 Oracle Corporation. All rights reserved.
DBV-00103: Specified BLOCKSIZE (2048) differs from actual (8192)
oracle@octilli:/db01/oracle/tst9 >
```

Once the BLOCKSIZE parameter is set to match the database block size, the DBVERIFY utility can proceed. There are two ways to run this utility: without the LOGFILE parameter specified, and with it specified.

Let's walk through each of these examples. First, this is what it looks like without the LOGFILE parameter set:

```
oracle@octilli:/db01/oracle/tst9 > dbv file=data01.dbf
BLOCKSIZE=8192
DBVERIFY: Release 9.0.1.0.0 - Production on Tue Oct 9 00:10:53 2001
(c) Copyright 2001 Oracle Corporation. All rights reserved.
DBVERIFY - Verification starting : FILE = data01.dbf
DBVERIFY - Verification complete
Total Pages Examined : 6400
Total Pages Processed (Data) : 0
Total Pages Failing (Data) : 0
Total Pages Processed (Index): 0
Total Pages Failing (Index): 0
Total Pages Processed (Other): 1
Total Pages Processed (Seg) : 0
Total Pages Failing (Seg) : 0
Total Pages Empty : 6399
Total Pages Marked Corrupt : 0
Total Pages Influx : 0
oracle@octilli:/db01/oracle/tst9 >
```

The following code demonstrates the DBVERIFY utility with the LOGFILE parameter set. The results of this command are written to the file data01.log and not to the screen. This can be displayed by editing the log file.

```
oracle@octilli:/db01/oracle/tst9 >dbv file=data01.dbf
BLOCKSIZE=8192 LOGFILE=data01.log
```

```
DBVERIFY: Release 9.0.1.0.0 - Production on Tue Oct 9 00:14:13 2001
(c) Copyright 2001 Oracle Corporation. All rights reserved.
oracle@octilli:/db01/oracle/tst9 >
```

8. Identifying RMAN-Specific Backups

There are three types of backups that are supported by the RMAN utility:

- Full or incremental
- Opened or closed
- Consistent or inconsistent

Each is described in the following sections.

8.1. Full or Incremental Backups

The full and incremental backups are differentiated by how the data blocks are backed up in the target database. The *full backup* backs up all the data blocks in the data files, modified or not. An *incremental backup* backs up only the data blocks in the data files that were modified since the last incremental backup.

The full backup cannot be used as part of an incremental backup strategy. The baseline backup for an incremental backup is a level 0 backup. A level 0 backup is a full backup at that point in time. Thus, all blocks, modified or not, are backed up, allowing the level 0 backup to serve as a baseline for future incremental backups. The incremental backups can then be applied with the baseline, or level 0, backup to form a full backup at some time in the future. The benefit of the incremental backup is that it is quicker, because not all data blocks need to be backed up.

There are two types of incremental backups: differential and cumulative, both of which back up only modified blocks. The difference between these two types of incremental backups is in the baseline database used to identify the modified blocks that need to be backed up.

The *differential incremental backup* backs up only data blocks modified since the most recent backup at the same level or lower. A differential incremental backup will determine which level 1 or level 2 backup has occurred most recently and back up only blocks that have changed since that backup. The differential incremental backup is the default incremental backup.

The *cumulative incremental backup* backs up only the data blocks that have changed since the most recent backup of the next lowest level—n – 1 or lower (with n being the existing level of backup). For example, if you are performing a level 2 cumulative incremental backup, the backup will copy data blocks only from the most recent level 1 backup. If no level 1 backup is available, then it will back up all data blocks that have changed since the most recent level 0 backup.

NOTE: Full backups do not mean the whole or complete database was backed up. In other words, a full backup can back up only part of the database and not all data files, control files, and logs.

8.2. Opened or Closed Backups

The opened and closed backups are differentiated by the state of the target database being backed up. The RMAN *opened backup* occurs when the target database is backed up while it is opened or available for use. This is similar to the non-RMAN hot backup that was demonstrated earlier in this lesson.

The RMAN *closed backup* occurs when the target database is mounted but not opened. This means the target database is not available for use during this type of backup. This is similar to the non-RMAN cold backup that was demonstrated earlier in this lesson.

8.3. Consistent or Inconsistent Backups

The consistent and inconsistent backups are differentiated by the state of the SCN in data file headers and in the control files. The *consistent backup* is a backup of a target database that is mounted but not opened and was shut down with either a SHUTDOWN IMMEDIATE, SHUTDOWN TRANSACTIONAL, or SHUTDOWN NORMAL option, but not the SHUTDOWN ABORT option. Also, the database must not have crashed prior to being mounted. This means that the SCN information in the data files matches the SCN information in the control files.

The *inconsistent backup* is the backup of the target database when it is opened but crashed prior to mounting, or when it was shut down with the SHUTDOWN ABORT option prior to mounting. This means that the SCN information in the data files does not match the SCN information in the control files.

9. Using RMAN's BACKUP and COPY Commands

There are two main backup sources that can be the basis for the RMAN recovery process: image copies and backup sets.

Image copies are actual copies of the database files, archived logs, or control files, and they are not stored in a special RMAN format — they can be stored only on disk. An image copy in RMAN is equivalent to an OS copy command such as cp or dd in Unix, or the COPY command in Windows NT/2000/XP. Thus, no RMAN restore processing is necessary to make image copies usable in a recovery situation. This can improve the speed and efficiency of the restore and recovery process in some cases. An image copy is performed by executing the RMAN COPY command.

On the other hand, database files in *backup sets* are stored in a special RMAN format and must be processed with the RESTORE command before these files are usable. This can take

more time and effort during the recovery process. Let's take a look at an example of using the BACKUP command, and then we will look at the RMAN COPY command in more detail.

9.1. Using RMAN BACKUP to Create Sets

The RMAN BACKUP command is used to perform a backup that creates a backup set.

When you are using the BACKUP command, the target database should be mounted or opened. You must manually allocate a channel for the BACKUP command to use during the backup process. Below is an example of this command in action. In this example, you are backing up the USERS tablespace and the current control file to a backup set.

```
oracle@octilli:/oracle/product/9.0.1/bin > rman

Recovery Manager: Release 9.0.1.0.0 - Production

(c) Copyright 2001 Oracle Corporation. All rights reserved.

RMAN> connect target

connected to target database: ORC9 (DBID=3960695)

RMAN> run

2> {allocate channel ch1 type disk;

3> backup tablespace users

4> include current controlfile;}


using target database controlfile instead of recovery catalog

allocated channel: ch1

channel ch1: sid=12 devtype=DISK


Starting backup at 30-SEP-01

channel ch1: starting full datafile backupset

channel ch1: specifying datafile(s) in backupset

input datafile fno=00008 name=/oracle/product/9.0.1/

oradata/orc9/users01.dbf

including current controlfile in backupset

channel ch1: starting piece 1 at 30-SEP-01

channel ch1: finished piece 1 at 30-SEP-01

piece handle=/oracle/product/9.0.1/dbs/01d5bspm_1_1
```

```
comment=NONE
channel ch1: backup set complete, elapsed time: 00:00:08
Finished backup at 30-SEP-01
released channel: ch1
```

NOTE: You cannot include archived logs and data files in a single backup. In other words, you will need to use the BACKUP command for the database or tablespace backup, and you will need to use it again for archived logs.

The BACKUP command has multiple options that can be specified. These options control performance, formatting, file sizes, and types of backups, to mention a few. Below are Tables 2 and 3, which describe the complete list of the BACKUP command's options and formats.

TABLE 2. BACKUP Command Options.

Option	Description
FULL	Causes the server session to copy all used blocks from data files into the backup set. The only blocks that do not get copied are blocks that have never been used. All archived log and redo log blocks are copied when the archived logs are designated for backup.
INCREMENTAL LEVEL INTEGER	Causes the server session to copy data blocks that have been modified since the last incremental n backup where n is any integer from 1 to 4.
FILESPERSET INTEGER	Determines how many files are in a backup set. When this option is used, the number of data files is compared to a determined number of files that are being backed up per channel allocated, and it takes the lower of the two. Using this option is another method for performing parallel backups.
DISKRATIO INTEGER	Forces RMAN to group data files in backup sets that are spread across a determined number of disk drives.
SKIP OFFLINE| READONLY | INACCESSIBLE	Excludes some data files or archived redo logs from the backup set. Some of the files it excludes include offline data files, read-only data files, or inaccessible data files and archived logs.

MAXSETSIZE INTEGER	Specifies the maximum size of the backup set. Bytes are the default unit of measure, but kilobytes (K), megabytes (M), and gigabytes (G) can also be used.
DELETE INPUT	Deletes input files when the backup set has been created. This option should only be used when backing up archived logs, data file copies, or backup sets. This is equivalent to using the CHANGE and DELETE command for all input files.
INCLUDE CURRENT CONTROLFILE	Creates a copy of the current control file and places it into each backup set.

TABLE 3. BACKUP Command Formats.

Format	**Description**
%c	Specifies the copy number of the backup piece within the set of duplexed backup pieces.
%d	Specifies the database name.
%n	Specifies the database name, padded on the right with n characters equal to a length of 8 characters.
%p	Specifies the backup piece number within the backup set. This number starts at 1 and increases by 1 for each backup piece created.
%s	Specifies the backup number. This number starts at 1 and increases by 1 for each backup piece created.
%t	Specifies the backup set time stamp. The combination of %s and %t can be used to form a unique name for a backup set.
%u	Specifies an 8-character name that combines a compressed version of the backup set number and the time the backup set was created.
%U	This format parameter is equivalent to %u_%p_%c.

9.2. Using RMAN COPY to Create Image Copies

In this example, we will utilize the RMAN COPY command to create an image copy of various database files. In this example, you are backing up the system data file and the current control file as image copies to the /staging directory.

```
RMAN> run { allocate channel ch1 type disk;
2> copy
3> datafile 1 to '/staging/system01.dbf' ,
4> current controlfile to '/staging/control.ctl';}

allocated channel: ch1
channel ch1: sid=12 devtype=DISK

Starting copy at 30-SEP-01
channel ch1: copied datafile 1
output filename=/staging/system01.dbf recid=1
stamp=441840852
channel ch1: copied current controlfile
output filename=/staging/control.ctl
Finished copy at 30-SEP-01
released channel: ch1
```

10. Backing Up the Control File

A control file backup can be performed through the RMAN utility by executing the CURRENT CONTROLFILE command. Below is a brief example of using this command within the RMAN utility after being connected to the appropriate target database and RMAN catalog database.

NOTE: In the following example, note that the TAG command is being used to name the backed up control file controlfile_thurs within the recovery catalog. As you might guess from this name, the TAG command is used to assign a meaningful, logical name to backups or image copies. By performing this task, this command allows you to find backups more easily in LIST outputs. The TAG command name can also be used in SWITCH and RESTORE commands.

```
RMAN> run {
2> allocate channel ch1 type disk;
3> backup
4> format 'cf_t%t_s%s_p%p'
5> tag controlfile_thurs
6> (current controlfile);
7> release channel ch1;
8> }

allocated channel: ch1
channel ch1: sid=11 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
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/cf_t442286312_s7_p1
comment=NONE
channel ch1: backup set complete, elapsed time: 00:00:02
Finished backup at 05-OCT-01
released channel: ch1
RMAN>
```

11. Backing Up the Archived Redo Logs

Another important part of a complete database backup is including the archived logs. Without the archived redo logs, you cannot roll forward from online backups. Below is an example of a complete database backup file that includes the archived redo logs. In this example, the first activity that is performed is the backing up of the database. Next, all the redo logs that can be archived are flushed to archived logs in the file-system. After that, the archived logs are backed up in the RMAN catalog by using the BACKUP ARCHIVELOG ALL command; this

command backs up all available archived logs in the filesystem. Below is a brief example of using this command as it is used in complete database backup.

```
run {
allocate channel c1 type disk;
allocate channel c2 type disk;
backup database;
backup (archivelog all);
}
```

12. Summary

User-managed backups have been commonplace at Oracle sites for years. This type of backup mainly consists of OS-based commands, such as those produced from Korn or Bourne shells in the Unix environment. In the Windows environment, user-managed backups are available through the use of third-party GUI tools and batch commands. More recently, RMAN backups have become more popular with many sites (since the release of RMAN in Oracle8).

The fundamentals of backing up an Oracle database can be seen clearly when you analyze an OS-based backup. Some of these basic techniques are utilized in the hot, or opened, backup examples, and some are used in cold, or closed examples. If you have a solid understanding of these concepts, you should fully understand the backup process with RMAN-based backups.

RMAN-based backups can perform the same functions as user-managed backups. But RMAN backups tend to be more standardized as a result of their use of a common command set. We demonstrated some of the commands that are involved in performing various backup operations within RMAN in this lesson.

User-managed and RMAN-based backups are the two primary methods of backing up databases. You will be responsible for at least one of these types of backups, if not both, in a real-world DBA job. Understanding how to perform user-managed backup and recovery is a necessary prerequisite for understanding RMAN-based backup and recovery. This knowledge will help you understand what RMAN is trying to do and improve upon.

References

[1] Oracle9i DBA Fundamentals II.