4. Control and Redo Log Files

Abstract: This practical lesson presents two important components of the Oracle database: the control file and the redo log files. The control file keeps information about the physical structure of the database. The redo log files record all changes made to data. These two files are critical for database recovery in case of a failure. You can multiplex both the control and the redo log files. In this lesson you will learn more about control files, putting the database in ARCHIVELOG mode, controlling checkpoints, and managing redo logs and control files with Oracle Managed Files (OMF).

Contents

Objectives:

- Explain the uses of the control file
- Describe the contents of the control file
- Multiplex and manage the control file
- Manage the control file with Oracle Managed Files (OMF)
- Obtain control file information
- Explain the purpose of online redo log files
- Describe the structure of online redo log files
- Control log switches and checkpoints
- Multiplex and maintain online redo log files
- Manage online redo log files with Oracle Managed Files (OMF)

1. Maintaining the Control File

You can think of the **control file** as a metadata repository for the physical database. It has the structure of the database—the data files and redo log files that constitute a database. The control file is a binary file, created when the database is created, and is updated with the physical changes whenever you add or rename a file.

The control file is updated continuously and should be available at all times. Don’t edit the contents of the control file; only Oracle processes should update its contents. When you start up the database, Oracle uses the control file to identify the data files, redo log files, and open them. Control files play a major role when recovering a database.

The contents of the control file include the following:

- Database name to which the control file belongs. A control file can belong to only one database.
- Database creation timestamp.
- Data files—name, location, and online/offline status information.
- Redo log files—name and location.
- Redo log archive information.
- Tablespace names.
- Current *log sequence number*, a unique identifier that is incremented and recorded when an online redo log file is switched.
- Most recent checkpoint information. A *checkpoint* occurs when all the modified database buffers in the SGA are written to the data files. The *system change number* (SCN), a number sequentially assigned to each transaction in the database, is also recorded in the control file against the data file name that is taken offline or made read-only.
- Begin and end of undo segments.
- Recovery Manager's (RMAN's) backup information. RMAN is the Oracle utility you use to back up and recover databases.

The control file size is determined by the MAX clauses you provide when you create the database: MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES, and MAXINSTANCES. Oracle pre-allocates space for these maximums in the control file. Therefore, when you add or rename a file in the database, the control file size does not change.

When you add a new file to the database or relocate a file, an Oracle server process immediately updates the information in the control file. Back up the control file after any structural changes. The log writer process (LGWR) updates the control file with the current log sequence number. The checkpoint process (CKPT) updates the control file with the recent checkpoint information. When the database is in ARCHIVELOG mode, the archiver process (ARCn) updates the control file with archiving information such as the archive log file name and log sequence number.

The control file contains two types of *record sections*: reusable and not reusable. Recovery Manager information is kept in the reusable section. Items such as the names of the backup data files are kept in this section, and once this section fills up, the entries are re-used in a circular fashion.

Multiplexing Control Files

Since the control file is critical for the database operation, Oracle recommends a minimum of two control files. You duplicate the control file on different disks either by using the multiplexing feature of Oracle or by using the mirroring feature of your operating system. The next two sections discuss the two ways you can implement the multiplexing feature: using *init.ora* and using an SPFILE.

Multiplexing Control Files Using init.ora

Multiplexing is defined as keeping a copy of the same control file in different locations. Copying the control file to multiple locations and changing the CONTROL_FILES parameter in the initialization file *init.ora* to include all control file names specifies the multiplexing of the control file. The following syntax shows three multiplexed control files.

```
CONTROL_FILES = ('/ora01/oradata/MYDB/ctrlMYDB01.ctl',
        '/ora02/oradata/MYDB/ctrlMYDB02.ctl',
        '/ora03/oradata/MYDB/ctrlMYDB03.ctl')
```

By storing the control file on multiple disks, you avoid the risk of a single point of failure. When multiplexing control files, updates to the control file can take a little longer, but that is insignificant when compared with the benefits. If you lose one control file, you can restart the database after copying one of the other control files or after changing the CONTROL_FILES parameter in the initialization file.

When multiplexing control files, Oracle updates all the control files at the same time, but uses only the first control file listed in the CONTROL_FILES parameter for reading.

When creating a database, you can list the control file names in the CONTROL_FILES parameter, and Oracle creates as many control files as are listed. You can have a maximum of eight multiplexed control file copies.

If you need to add more control file copies, do the following:

1. Shut down the database.
2. Copy the control file to more locations by using an operating system command.
3. Change the initialization parameter file to include the new control file name(s) in the parameter CONTROL_FILES.
4. Start up the database.

After creating the database, you can change the location of the control files, rename the control files, or drop certain control files. You must have at least one control file for each database. To add, rename, or delete control files, you need to follow the preceding steps. Basically, you shut down the database, use the operating system copy command (copy, rename, or drop the control files accordingly), edit the CONTROL_FILES parameter in *init.ora* and start up the database.

Multiplexing Control Files Using an SPFILE

Multiplexing using an SPFILE is similar to multiplexing using *init.ora*; the major difference being how the CONTROL_FILES parameter is changed.

Follow these steps:

1. Alter the SPFILE while the database is still open:

```
SQL> ALTER SYSTEM SET CONTROL_FILES =
    '/ora01/oradata/MYDB/ctrlMYDB01.ctl',
    '/ora02/oradata/MYDB/ctrlMYDB02.ctl',
    '/ora03/oradata/MYDB/ctrlMYDB03.ctl',
    '/ora04/oradata/MYDB/ctrlMYDB04.ctl' SCOPE=SPFILE;
```

   This parameter change will only take effect after the next instance restart by using the SCOPE=SPFILE qualifier. The contents of the binary SPFILE are changed immediately, but the old specification of CONTROL_FILES will be used until the instance is restarted.

2. Shut down the database.

```
SQL> SHUTDOWN NORMAL
```

3. Copy an existing control file to the new location:

```
$ cp /ora01/oradata/MYDB/ctrlMYDB01.ctl
     /ora01/oradata/MYDB/ctrlMYDB04.ctl
```

4. Start the instance.

```
SQL> STARTUP
```

TIP: If you lose one of the control files, you can shut down the database, copy a control file, or change the CONTROL_FILES parameter and restart the database.

Using OMF to Manage Control Files

Using OMF can make the creation and maintenance of control files much easier. To use OMF-created control files, do not specify the CONTROL_FILES parameter in *init.ora*, but instead make sure that the parameter DB_CREATE_ONLINE_LOG_DEST_n is specified *n* times starting with 1. Therefore, *n* is the number of desired control files to be created. The actual names of the control files are system generated and can be found in the alert logs located in $ORACLE_HOME/admin/bdump.

To add more copies of the control file later, use the method described in the previous section, “Multiplexing Control Files Using an SPFILE.”

Creating New Control Files

You can create a new control file by using the CREATE CONTROLFILE command. You will need to create a new control file if you lose all the control files that belong to the

database, if you want to change any of the MAX clauses in the CREATE DATABASE command, or if you want to change the database name. You must know the data file names and redo log file names to create the control file. Follow these steps to create the new control file:

1. Prepare the CREATE CONTROLFILE command. You should have the complete list of data files and redo log files. If you omit any data files, they can no longer be a part of the database. The following is an example of the CREATE CONTROLFILE command.

```
CREATE CONTROLFILE SET DATABASE "ORACLE"
        NORESETLOGS NOARCHIVELOG
        MAXLOGFILES 32
        MAXLOGMEMBERS 2
        MAXDATAFILES 32
        MAXINSTANCES 1
        MAXLOGHISTORY 1630
LOGFILE
        GROUP 1 'C:\ORACLE\DATABASE\LOG2ORCL.ORA' SIZE 500K,
        GROUP 2 'C:\ORACLE\DATABASE\LOG1ORCL.ORA' SIZE 500K
DATAFILE
        'C:\ORACLE\DATABASE\SYS1ORCL.ORA',
        'C:\ORACLE\DATABASE\USR1ORCL.ORA',
        'C:\ORACLE\DATABASE\RBS1ORCL.ORA',
        'C:\ORACLE\DATABASE\TMP1ORCL.ORA',
        'C:\ORACLE\DATABASE\APPDATA1.ORA',
        'C:\ORACLE\DATABASE\APPINDX1.ORA'
;
```

   The options in this command are similar to the CREATE DATABASE command. The NORESETLOGS option specifies that the online redo log files should not be reset.

2. Shut down the database.
3. Start up the database with the NOMOUNT option. Remember, to mount the database, Oracle needs to open the control file.

4. Create the new control file with a command similar to the preceding example. The control files will be created using the names and locations specified in the initialization parameter CONTROL_FILES.
5. Open the database by using the ALTER DATABASE OPEN command.
6. Shut down the database and back up the database.

These steps are very basic. Depending on the situation, you might have to perform additional steps.

NOTE: You can generate the CREATE CONTROLFILE command from the current database by using the command ALTER DATABASE BACKUP CONTROLFILE TO TRACE. The control file creation script is written to the USER_DUMP_DEST directory.

After creating the control file, determine whether any of the data files listed in the dictionary are missing in the control file. If you query the V$DATAFILE view, the missing files will have the name MISSING*nnnn*. If you created the control file by using the RESETLOGS option, the missing data files cannot be added back to the database. If you created the control file with the NORESETLOGS option, the missing data file can be included in the database by performing a media recovery.

You can back up the control file when the database is up by using the command

```
ALTER DATABASE BACKUP CONTROLFILE TO '<filename>' REUSE;
```

Another way to back up a control file is by using the command

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE;
```

This command places the contents of the control file into a text-format trace file, located in USER_DUMP_DEST, together with some extraneous information that must be edited out before using it to re-create the control file:

```
Dump file H:\Oracle9i\admin\or90\udump\ORA01568.TRC
Wed Oct 10 22:00:05 2001
ORACLE V9.0.1.1.1 - Production vsnsta=0
vsnsql=10 vsnxtr=3
Windows 2000 Version 5.0 Service Pack 2, CPU type 586
Oracle9i Enterprise Edition Release 9.0.1.1.1 - Production
```



```
With the Partitioning option
JServer Release 9.0.1.1.1 - Production
Windows 2000 Version 5.0 Service Pack 2, CPU type 586
Instance name: or90
Redo thread mounted by this instance: 1

Oracle process number: 13

Windows thread id: 1568, image: ORACLE.EXE

*** SESSION ID:(8.39) 2001-10-10 22:00:05.000
*** 2001-10-10 22:00:05.000
# 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 "OR90" NORESETLOGS NOARCHIVELOG
    MAXLOGFILES 50
    MAXLOGMEMBERS 5
    MAXDATAFILES 100
    MAXINSTANCES 1
    MAXLOGHISTORY 113
LOGFILE
    GROUP 1 'H:\ORACLE9I\ORADATA\OR90\REDO01.LOG' SIZE 100M,
    GROUP 2 'H:\ORACLE9I\ORADATA\OR90\REDO02.LOG' SIZE 100M,
    GROUP 3 'H:\ORACLE9I\ORADATA\OR90\REDO03.LOG' SIZE 100M
# STANDBY LOGFILE
DATAFILE
```



```
        'H:\ORACLE9I\ORADATA\OR90\SYSTEM01.DBF',
        'H:\ORACLE9I\ORADATA\OR90\UNDOTBS01.DBF',
        'H:\ORACLE9I\ORADATA\OR90\CWMLITE01.DBF',
        'H:\ORACLE9I\ORADATA\OR90\DRSYS01.DBF',
        'H:\ORACLE9I\ORADATA\OR90\EXAMPLE01.DBF',
        'H:\ORACLE9I\ORADATA\OR90\INDX01.DBF',
        'H:\ORACLE9I\ORADATA\OR90\TOOLS01.DBF',
        'H:\ORACLE9I\ORADATA\OR90\USERS01.DBF',
        'H:\ORACLE9I\ORADATA\OR90\OEM_REPOSITORY.DBF'
CHARACTER SET WE8MSWIN1252
;
# Recovery is required if any of the datafiles are restored backups,
# or if the last shutdown was not normal or immediate.
RECOVER DATABASE
# 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
'H:\ORACLE9I\ORADATA\OR90\TEMP01.DBF' REUSE;
# End of tempfile additions.
#
```

Oracle recommends backing up the control file whenever you make a change to the database structure, such as adding data files, renaming files, or dropping redo log files.

Querying Control File Information

The Oracle data dictionary holds all the information about the control file. The view V$CONTROLFILE lists the names of the control files for the database. The STATUS column should always be NULL; when a control file is missing, the STATUS would be INVALID,

but that should never occur because when Oracle cannot update one of the control files, the instance crashes – you can start up the database only after copying a good control file.

For example, in order to obtain control information you do as following:

```
SQL> SELECT * FROM V$CONTROLFILE;

STATUS NAME

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

       /ora01/oradata/MYDB/ctrlMYDB01.ctl

       /ora02/oradata/MYDB/ctrlMYDB02.ctl

       /ora03/oradata/MYDB/ctrlMYDB03.ctl

3 rows selected.

SQL>
```

You can also use the SHOW PARAMETER command to retrieve the names of the control files.

```
SQL> show parameter control_files

NAME TYPE VALUE

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

control_files string  H:\Oracle9i\oradata\or90\CONTROL01.CTL,

                      H:\Oracle9i\oradata\

                      or90\CONTROL02.CTL, H:\Oracle9i\

                      oradata\or90\CONTROL03.CTL
```

The other data dictionary view that gives information about the control file is V$CONTROLFILE_RECORD_SECTION, which displays the control file record sections. The record type, record size, total records allocated, number of records used, and the index position of the first and last records are in this view. For a listing of the record types, record sizes, and usage, run the following query.

```
SQL> SELECT TYPE, RECORD_SIZE, RECORDS_TOTAL, RECORDS_USED

2 FROM V$CONTROLFILE_RECORD_SECTION;

TYPE   RECORD_SIZE RECORDS_TOTAL RECORDS_USED

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

DATABASE                 192     1        1

CKPT PROGRESS            4084    1        0
```

```
REDO THREAD              104     1       1
REDO LOG                 72      32      3
DATAFILE                 180     254     8
FILENAME                 524     319     11
TABLESPACE               68      254     8
TEMPORARY FILENAME       56      254     0
RMAN CONFIGURATION       1       1       0
LOG HISTORY              36      1815    1217
OFFLINE RANGE            56      291     0
ARCHIVED LOG             584     13      0
BACKUP SET               40      408     0
BACKUP PIECE             736     510     0
BACKUP DATAFILE          116     563     0
BACKUP REDOLOG           76      107     0
DATAFILE COPY            660     519     0
BACKUP CORRUPTION        44      371     0
COPY CORRUPTION          40      408     0
DELETED OBJECT           20      408     0
PROXY COPY               852     575     0
RESERVED4                1       8168    0
22 rows selected.
SQL>
```

Other data dictionary views read information from the control file. Table 1 lists and describes these dynamic performance views. You can access these views when the database is mounted, that is, before opening the database.

Table 1. Dictionary Views That Read from the Control File.

View Name	Description
V$ARCHIVED_LOG	Archive log information such as size, SCN, timestamp, etc.
V$BACKUP	Backup status for individual datafiles that constitute the database.

V$BACKUP_DATAFILE	Contains filename, timestamp, etc. of the data files backed up using RMAN.
V$BACKUP_PIECE	Information about backup pieces, updated when using RMAN.
V$BACKUP_REDOLOG	Information about the archived log files backed up using RMAN.
V$BACKUP_SET	Information about complete, successful backups using RMAN.
V$DATABASE	Database information such as name, creation timestamp, archive log mode, SCN, log sequence number, etc.
V$DATAFILE	Information about the data files associated with the database.
V$DATAFILE_COPY	Information about data files copied during a hot backup or using RMAN.
V$DATAFILE_HEADER	Data file header information; the filename and status are obtained from the control file.
V$LOG	Online redo log group information.
V$LOGFILE	Files or members of the online redo log group.
V$THREAD	Information about the log files assigned to each instance.

2. Maintaining and Monitoring Redo Log Files

Redo logs record all changes to the database. The redo log buffer in the SGA is written to the redo log file periodically by the LGWR process. The redo log files are accessed and are open during normal database operation; hence they are called the online redo log files. Every Oracle database must have at least two redo log files. The LGWR process writes to these files in a circular fashion. For example, say there are three online redo log files. The LGWR process writes to the first file, and when this file is full, it starts writing to the second file, and then to the third file, and then again to the first file (overwriting the contents).

Online redo log files are filled with redo records. A *redo record*, also called a redo entry, is made up of a group of *change vectors*, each of which is a description of a change made to a single block in the database. Redo entries record data that you can use to reconstruct all changes made to the database, including the undo segments. When you recover the database by using redo log files, Oracle reads the change vectors in the redo records and applies the changes to the relevant blocks.

LGWR writes redo information from the redo log buffer to the online redo log files under a variety of circumstances:

- A user commits a transaction, even if this is the only transaction in the log buffer.
- The redo log buffer becomes one-third full.
- When there is approximately 1MB of changed records in the buffer. This total does not include deleted or inserted records.

NOTE: LGWR always writes its records to the online redo log file *before* DBW*n* writes new or modified database buffer cache records to the datafiles.

Each database has its own online *redo log groups*. A log group can have one or more *redo log members* (each member is a single operating system file). If you have a Real Application Cluster configuration, in which multiple instances are mounted to one database, each instance will have one online redo thread. That is, the LGWR process of each instance writes to the same online redo log files, and hence Oracle has to keep track of the instance from where the database changes are coming. For single instance configurations, there will be only one thread, and that thread number is 1. The redo log file contains both committed and uncommitted transactions. Whenever a transaction is committed, a system change number is assigned to the redo records to identify the committed transaction.

The redo log group is referenced by an integer; you can specify the group number when you create the redo log files, either when you create the database or when you create the control file. You can also change the redo log configuration (add/drop/rename files) by using database commands. The following example shows a CREATE DATABASE command.

```
CREATE DATABASE "MYDB01"
LOGFILE '/ora02/oradata/MYDB01/redo01.log' SIZE 10M,
        '/ora03/oradata/MYDB01/redo02.log' SIZE 10M;
```

Two log file groups are created here; the first file will be assigned to group 1, and the second file will be assigned to group 2. You can have more files in each group; this practice is known as the multiplexing of redo log files, which we'll discuss later. You can specify any group number — the range will be between 1 and MAXLOGFILES. Oracle recommends that all redo log groups be the same size. The following is an example of creating the log files by specifying the groups.

```
CREATE DATABASE "MYDB01"
LOGFILE GROUP 1 '/ora02/oradata/MYDB01/redo01.log' SIZE 10M,
        GROUP 2 '/ora03/oradata/MYDB01/redo02.log' SIZE 10M;
```

Log Switch Operations

The LGWR process writes to only one redo log file group at any time. The file that is actively being written to is known as the current log file. The log files that are required for instance recovery are known as the active log files. The other log files are known as inactive. Oracle automatically recovers an instance when starting up the instance by using the online redo log files. Instance recovery may be needed if you do not shut down the database properly or if your computer crashes.

The log files are written in a circular fashion. A log switch occurs when Oracle finishes writing to one file and starts writing to the next file. A log switch always occurs when the current redo log file is completely full and log writing must continue. You can force a log switch by using the ALTER SYSTEM command. A manual log switch may be necessary when performing maintenance on the redo log files by using the ALTER SYSTEM SWITCH LOGFILE command. Figure 1 shows how LGWR writes to the redo log groups in a circular fashion.

Whenever a log switch occurs, Oracle allocates a sequence number to the new redo log file before writing to it. As stated earlier, this number is known as the log sequence number. If there are lots of transactions or changes to the database, the log switches can occur too frequently. Size the redo log file appropriately to avoid frequent log switches. Oracle writes to the alert log file whenever a log switch occurs.

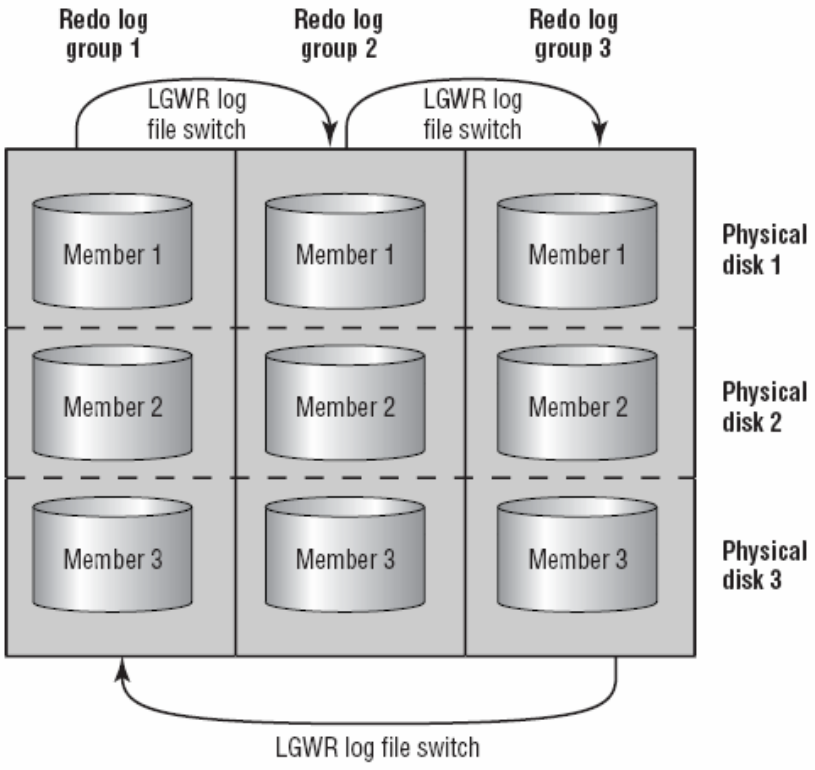

Fig. 1. Redo log file usage.

TIP: Redo log files are written sequentially on the disk, so the I/O will be fast if there is no other activity on the disk (the disk head is always properly positioned). Keep the redo log files on a separate disk for better performance. If you have to store a data file on the same disk as the redo log file, do not put the SYSTEM, UNDOTBS, or any very active data or index tablespace file on this disk.

Database checkpoints are closely tied to redo log file switches. A checkpoint is an event that flushes the modified data from the buffer cache to the disk and updates the control file and data files. The CKPT process updates the headers of data files and control files; the actual blocks are written to the file by the DBW*n* process. A checkpoint is initiated when the redo log file is filled and a log switch occurs, when the instance is shut down with NORMAL, TRANSACTIONAL, or IMMEDIATE, when a tablespace status is changed to readonly or put into BACKUP mode, when a tablespace or datafile is taken offline, or when other values specified by certain parameters (discussed later) are reached.

You can force a checkpoint if needed. Forcing a checkpoint ensures that all changes to the database buffers are written to the data files on disk.

```
ALTER SYSTEM CHECKPOINT;
```

Another way to force a checkpoint is by forcing a log file switch.

```
ALTER SYSTEM SWITCH LOGFILE;
```

The size of the redo log affects the checkpoint performance. If the size of the redo log is smaller compared with the number of transactions, a log switch occurs often and so does the checkpoint. The DBW*n* process writes the dirty buffer blocks whenever a checkpoint occurs. This situation might reduce the time required for instance recovery, but it might also affect the runtime performance. You can adjust checkpoints primarily by using the initialization parameter FAST_START_MTTR_TARGET. This parameter replaces the deprecated parameters FAST_START_IO_TARGET and LOG_CHECKPOINT_TIMEOUT in previous versions of the Oracle database. It is used to ensure that recovery time at instance startup (if required) will not exceed a certain number of seconds.

For example, setting FAST_START_MTTR_TARGET to 600 ensures that system recovery will not take more than 600 seconds, by writing redo log buffer blocks more often, writing dirty database buffer cache entries more often, and so on.

TIP: Setting the parameter LOG_CHECKPOINTS_TO_ALERT to TRUE logs each checkpoint activity to the alert log file, which is useful for determining if checkpoints are occurring at the desired frequency.

REAL WORLD SCENARIO: Redo Log Troubleshooting

In the case of redo log groups, it's best to be generous with the number of groups and the number of members for each group. After estimating the number of groups that would be appropriate for your installation, add one more. I can remember many database installations in which I was trying to be overly cautious about disk space usage, not putting things into perspective, and realizing that the slight additional work involved in maintaining either additional or larger redo logs is small in relation to the time needed to fix a problem when the number of users and concurrent active transactions increase.

The space needed for additional log file groups is minimal and is well worth the effort up front to avoid the dreaded "Checkpoint not complete" message in the alert log and potentially increasing the recovery time in the event of an instance failure.

Multiplexing Log Files

You can keep multiple copies of the online redo log file to safeguard against damage to these files. When multiplexing online redo log files, LGWR concurrently writes the same redo log information to multiple identical online redo log files, thereby eliminating a single point of redo log failure. All copies of the redo file are the same size and are known as a *group*, which is identified by an integer. Each redo log file in the group is known as a *member*. You must have at least two redo log groups for normal database operation.

When multiplexing redo log files, it is preferable to keep the members of a group on different disks, so that one disk failure will not affect the continuing operation of the database. If LGWR can write to at least one member of the group, database operation proceeds as normal; an entry is written to the alert log file. If all members of the redo log file group are not available for writing, Oracle shuts down the instance. An instance recovery or media recovery may be needed to bring up the database.

You can create multiple copies of the online redo log files when you create the database. For example, the following statement creates two redo log file groups with two members in each.

```
CREATE DATABASE "MYDB01"
LOGFILE
GROUP 1 ('/ora02/oradata/MYDB01/redo0101.log',
        '/ora03/oradata/MYDB01/redo0102.log') SIZE 10M,
GROUP 2 ('/ora02/oradata/MYDB01/redo0201.log',
        '/ora03/oradata/MYDB01/redo0202.log') SIZE 10M;
```

The maximum number of log file groups is specified in the clause MAXLOGFILES, and the maximum number of members is specified in the clause MAXLOGMEMBERS. You can separate the filenames (members) by using a space or a comma.

Creating New Groups

You can create and add more redo log groups to the database by using the ALTER DATABASE command. The following statement creates a new log file group with two members.

```
ALTER DATABASE ADD LOGFILE
GROUP 3 ('/ora02/oradata/MYDB01/redo0301.log',
        '/ora03/oradata/MYDB01/redo0402.log') SIZE 10M;
```

If you omit the GROUP clause, Oracle assigns the next available number. For example, the following statement also creates a multiplexed group.

```
ALTER DATABASE ADD LOGFILE
        ('/ora02/oradata/MYDB01/redo0301.log'
        '/ora03/oradata/MYDB01/redo0402.log') SIZE 10M;
```

To create a new group without multiplexing, use the following statement.

```
ALTER DATABASE ADD LOGFILE
        '/ora02/oradata/MYDB01/redo0301.log' REUSE;
```

You can add more than one redo log group by using the ALTER DATABASE command—just use a comma to separate the groups.

TIP: If the redo log files you create already exist, use the REUSE option and don't specify the size. The new redo log size will be same as that of the existing file.

Adding New Members

If you forgot to multiplex the redo log files when creating the database or if you need to add more redo log members, you can do so by using the ALTER DATABASE command. When adding new members, you do not specify the file size, because all group members will have the same size.

If you know the group number, using the following statement will add a member to group 2.

```
ALTER DATABASE ADD LOGFILE MEMBER
        '/ora04/oradata/MYDB01/redo0203.log' TO GROUP 2;
```

You can also add group members by specifying the names of other members in the group, instead of specifying the group number. Specify all the existing group members with this syntax.

```
ALTER DATABASE ADD LOGFILE MEMBER
        '/ora04/oradata/MYDB01/redo0203.log' TO
        ('/ora02/oradata/MYDB01/redo0201.log',
        '/ora03/oradata/MYDB01/redo0202.log');
```

Adding New Members Using Storage Manager

Adding redo log members is even easier with Storage Manager. Figure 2 shows the screen from Storage Manager that you can use to add new redo log members to a redo log group.

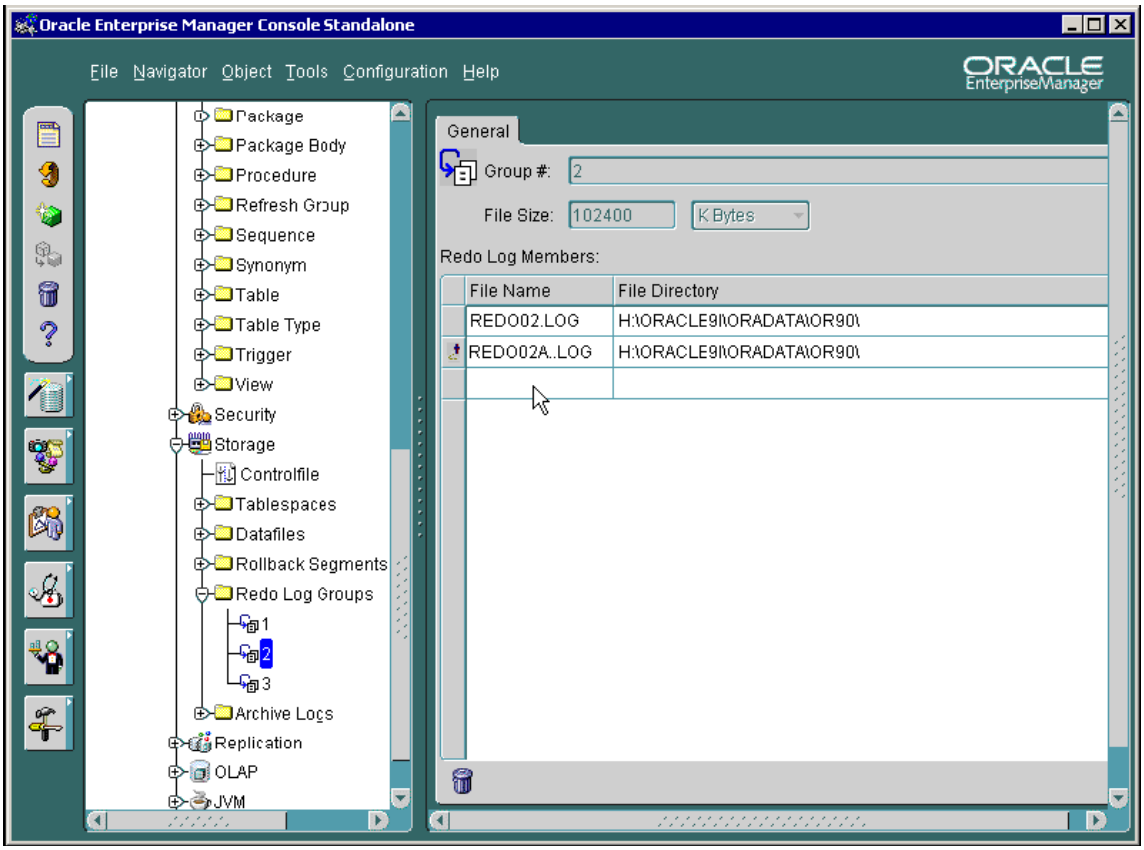

Fig. 2. Storage Manager.

Renaming Log Members

If you want to move the log file member from one disk to another or just want a more meaningful name, you can rename a redo log member. Before renaming the online redo log members, the new (target) online redo files should exist. The SQL commands in Oracle

change only the internal pointer in the control file to a new log file; they do not change or rename the operating system file. You must use an operating system command to rename or move the file. Follow these steps to rename a log member:

1. Shut down the database (a complete backup is recommended).
2. Copy/rename the redo log file member to the new location by using an operating system command.
3. Start up the instance and mount the database (STARTUP MOUNT).
4. Rename the log file member in the control file. Use ALTER DATABASE RENAME FILE '<old_redo_file_name>' TO '<new_redo_file_name>';
5. Open the database (ALTER DATABASE OPEN).
6. Back up the control file.

Dropping Redo Log Groups

You can drop a redo log group and its members by using the ALTER DATABASE command. Remember that you should have at least two redo log groups for the database to function normally. The group that is to be dropped should not be the active group or the current group—that is, you can drop only an inactive log file group. If the log file to be dropped is not inactive, use the ALTER SYSTEM SWITCH LOGFILE command.

To drop the log file group 3, use the following SQL statement.

```
ALTER DATABASE DROP LOGFILE GROUP 3;
```

When an online redo log group is dropped from the database, the operating system files are not deleted from disk. The control files of the associated database are updated to drop the members of the group from the database structure. After dropping an online redo log group, make sure that the drop is completed successfully, and then use the appropriate operating system command to delete the dropped online redo log files.

Dropping Redo Log Members

Similar to conditions for dropping a redo log group, you can drop only the members of an inactive redo log group. Also, if there are only two groups, the log member to be dropped should not be the last member of a group. You can have a different number of members for each redo log group, though it is not advised. For example, say you have three log groups, each with two members. If you drop a log member from group 2, and a failure occurs to the sole member of group 2, the instance crashes. So even if you drop a member for maintenance reasons, ensure that all redo log groups have the same number of members.

To drop the log member, use the DROP LOGFILE MEMBER clause of the ALTER DATABASE command.

```
ALTER DATABASE DROP LOGFILE MEMBER
```



```
'/ora04/oradata/MYDB01/redo0203.log';
```

The operating system file is not removed from the disk; only the control file is updated. Use an operating system command to delete the redo log file member from disk.

> *TIP*: If a database is running in ARCHIVELOG mode, redo log members cannot be deleted unless the redo log group has been archived.

Clearing Online Redo Log Files

Under certain circumstances, a redo log group member (or all members of a log group) may become corrupted. To solve this problem, you can drop and re-add the log file group or group member. It is much easier, however, to use the ALTER DATABASE CLEAR LOGFILE command. The following example clears the contents of redo log group 3 in the database:

```
ALTER DATABASE CLEAR LOGFILE GROUP 3;
```

Another distinct advantage of this command is that you can clear a log group even if the database has only two log groups, and only one member in each group. You can also clear a log group member even if it has not been archived yet by using the UNARCHIVED keyword. In this case, it is advisable to do a full database backup at the earliest convenience, because the unarchived redo log file is no longer usable for database recovery.

Managing Online Redo Log Files with OMF

OMF simplifies online redo log management. As with all OMF-related operations, be sure that the proper initialization parameters are set. If you are multiplexing redo logs in three locations, be sure to set the parameters DB_CREATE_ONLINE_LOG_DEST_1 through _3.

To add a new log file group, use the following command:

```
ALTER DATABASE ADD LOGFILE;
```

The filenames for the three new operating systems files are generated automatically. Be sure to set each DB_CREATE_ONLINE_LOG_DEST_*n* parameter to path names on different physical volumes.

Archiving Log Files

You know that online redo log files record all changes to the database. Oracle lets you copy these log files to a different location or to an offline storage medium. The process of copying is called *archiving*. The archiver process (ARC*n*) does this archiving. By archiving the redo log files, you can use them later to recover a database, update the standby database, or use the LogMiner utility to audit the database activities.

When an online redo log file is full, and LGWR starts writing to the next redo log file, ARC*n* copies the completed redo log file to the archive destination. It is possible to specify more than one archive destination. The LGWR process waits for the ARC*n* process to complete the copy operation before overwriting any online redo log file.

When the archiver process is copying the redo log files to another destination, the database is said to be in ARCHIVELOG mode. If archiving is not enabled, the database is said to be in NOARCHIVELOG mode. For production systems, for which you cannot afford to lose data, you must run the database in ARCHIVELOG mode so that in the event of a failure, you can recover the database to the time of failure or to a point in time. You can achieve this ability to recover by restoring the database backup and applying the database changes by using the archived log files.

Setting the Archive Destination

You specify the archive destination in the initialization parameter file. To change the archive destination parameters during normal database operation, you use the ALTER SYSTEM command. The following parameters are associated with archive log destinations and the archiver process:

LOG_ARCHIVE_DEST Specifies the destination to write archive log files. This location should be a valid directory on the server where the database is located. You can change the archiving location specified by this parameter by using ALTER SYSTEM SET LOG_ARCHIVE_DEST = '*<new_location>*';

LOG_ARCHIVE_DUPLEX_DEST Specifies a second destination to write the archive log files. This destination must be a location on the server where the database is located. This destination can be either a must succeed or a best-effort archive destination, depending on how many archive destinations must succeed. You specify the minimum successful number of archive destinations in the parameter LOG_ARCHIVE_MIN_SUCCEED_DEST. You can change the archiving location specified by this parameter by using ALTER SYSTEM SET LOG_ARCHIVE_DUPLEX_DEST = '*<new_location>*';

LOG_ARCHIVE_DEST_*n* Using this parameter, you can specify as many as five archiving destinations. These archive locations can be either on the local machine or on a remote machine where the standby database is located. When these parameters are used, you cannot use the LOG_ARCHIVE_DEST or LOG_ARCHIVE_DUPLEX_DEST parameters to specify the archiving location. The syntax for specifying this parameter in the initialization file is as follows:

```
LOG_ARCHIVE_DEST_n = "null_string" |
((SERVICE = <tnsnames_name> |
        LOCATION = '<directory_name>')
[MANDATORY | OPTIONAL]
[REOPEN [= <integer>]])
```


For example, LOG_ARCHIVE_DEST_1 = ((LOCATION='/archive/MYDB01') MANDATORY REOPEN = 60) specifies a location for the archive log files on the local machine at /archive/MYDB01. The MANDATORY clause specifies that writing to this location must succeed. The REOPEN clause specifies when the next attempt to write to this location should be made, when the first attempt did not succeed. The default value is 300 seconds.

Here is another example, which applies the archive logs to a standby database on a remote computer.

```
LOG_ARCHIVE_DEST_2 = (SERVICE=STDBY01) OPTIONAL REOPEN;
```

Here STDBY01 is the Oracle Net connect string used to connect to the remote database. Since writing is optional, the database activity continues even if ARC*n* could not write the archive log file. It tries the writing operation again since the REOPEN clause is specified.

LOG_ARCHIVE_MIN_SUCCEED_DEST Specifies the number of destinations the ARCn process should successfully write at a minimum to proceed with overwriting the online redo log files. The default value of this parameter is 1. If you are using the LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST parameters, setting this value to 1 makes LOG_ARCHIVE_DEST mandatory and LOG_ARCHIVE_DUPLEX_DEST optional. If you set the parameter to 2, writing to both destinations must be successful. If you are using the LOG_ARCHIVE_DEST_n parameter, the LOG_ARCHIVE_MIN_SUCCEED_DEST parameter cannot exceed the total number of enabled destinations. If this parameter value is less than the number of MANDATORY destinations, the parameter is ignored.

LOG_ARCHIVE_FORMAT Specifies the format in which to write the filename of the archived redo log files. You can provide a text string and any of the predefined variables. The variables are as follows:

%s Log sequence number

%S Log sequence number, zero filled

%t Thread number

%T Thread number, zero filled

For example, specifying the LOG_ARCHIVE_FORMAT = 'arch_%t_%s' generates the archive log file names as arch_1_101, arch_1_102, arch_1_103, and so on; 1 is the thread number, and 101, 102, and 103 are log sequence numbers. Specifying the format as arch_%S generates filenames such as arch_00000101, arch_00000102, and so on; the number of leading zeros depends on the operating system.

LOG_ARCHIVE_MAX_PROCESSES Specifies the maximum number of ARC*n* processes Oracle should start when starting up the database. By default the value is 1.

LOG_ARCHIVE_START Specifies whether Oracle should enable automatic archiving. If this parameter is set to FALSE, none of the ARC*n* processes are started. You can override this parameter by using the command ARCHIVE LOG START or ARCHIVE LOG STOP.

Setting ARCHIVELOG

Specifying these parameters does not start writing the archive log files; you should place the database in ARCHIVELOG mode to enable archiving of the redo log files. You can specify the ARCHIVELOG clause while creating the database. However, most DBAs prefer to create the database first and then enable ARCHIVELOG mode. To enable ARCHIVELOG mode, follow these steps:

1. Shut down the database. Set up the appropriate initialization parameters.
2. Start up and mount the database.
3. Enable ARCHIVELOG mode by using the command ALTER DATABASE ARCHIVELOG.
4. Open the database by using ALTER DATABASE OPEN.

To disable ARCHIVELOG mode, follow these steps:

1. Shut down the database.
2. Start up and mount the database.
3. Disable ARCHIVELOG mode by using the command ALTER DATABASE NOARCHIVELOG.
4. Open the database by using ALTER DATABASE OPEN.

You can enable automatic archiving by setting the parameter `LOG_ARCHIVE_START = TRUE`. If you set the parameter to `FALSE`, Oracle does not start the ARC*n* process. Therefore, when the redo log files are full, the database will hang, waiting for the redo log files to be archived. You can initiate the automatic archive process by using the command `ALTER SYSTEM ARCHIVE LOG START`, which starts the ARC*n* processes; to manually archive all unarchived logs, use the command

```
ALTER SYSTEM ARCHIVE LOG ALL.
```

Querying Log and Archive Information

You can query the redo log file information from the SQL command ARCHIVE LOG LIST or by querying the dynamic performance views.

The ARCHIVE LOG LIST command shows whether the database is in ARCHIVELOG mode, whether automatic archiving is enabled, the archival destination, and the oldest, next, and current log sequence numbers.

```
SQL> archive log list
Database log mode              Archive Mode
```

```
Automatic archival              Enabled
Archive destination             C:\Oracle\oradata\ORADB02\archive
Oldest online log sequence      194
Next log sequence to archive    196
Current log sequence            196
SQL>
```

TIP: The view V$DATABASE shows whether the database is in ARCHIVELOG mode or in NOARCHIVELOG mode.

V$LOG

This dynamic performance view contains information about the log file groups and sizes and its status. The valid status codes in this view and their meanings are as follows:

UNUSED New log group, never used.

CURRENT Current log group.

ACTIVE Log group that may be required for instance recovery.

CLEARING You issued an ALTER DATABASE CLEAR LOGFILE command.

CLEARING_CURRENT Empty log file after issuing the ALTER DATABASE CLEAR LOGFILE command.

INACTIVE The log group is not needed for instance recovery.

Here is a query from V$LOG:

```
SQL> SELECT * FROM V$LOG;
GROUP# THREAD# SEQUENCE# BYTES MEMBERS
---------- ---------- ---------- ---------- ----------
ARCHIVED STATUS FIRST_CHANGE# FIRST_TIM
-------- ---------------- ------------ ---------
         1          1        196          1048576      2
NO       CURRENT                 56686        30-JUL-01
         2          1        194          1048576      2
YES      INACTIVE                36658        28-JUL-01
         3          1        195          1048576      2
```

```
YES   INACTIVE                36684        28-JUL-01

SQL>
```

V$LOGFILE

The V$LOGFILE view has information about the log group members. The filenames and group numbers are in this view. The STATUS column can have the value INVALID (file is not accessible), STALE (file's contents are incomplete), DELETED (file is no longer used), or blank (file is in use).

```
SQL> SELECT * FROM V$LOGFILE
       2       ORDER BY GROUP#;

       GROUP# STATUS TYPE MEMBER
--------- ------- ------- ----------------------------------
       1             ONLINE     C:\ORACLE\ORADATA\ORADB02\REDO11.LOG
       1             ONLINE     D:\ORACLE\ORADATA\ORADB02\REDO12.LOG
       2  STALE      ONLINE     C:\ORACLE\ORADATA\ORADB02\REDO21.LOG
       2             ONLINE     D:\ORACLE\ORADATA\ORADB02\REDO22.LOG
       3             ONLINE     C:\ORACLE\ORADATA\ORADB02\REDO31.LOG
       3             ONLINE     D:\ORACLE\ORADATA\ORADB02\REDO32.LOG

6 rows selected.

SQL>
```

V$THREAD

This view shows information about the threads in the database. A single instance database will have only one thread. This view shows the instance name, thread status, SCN status, log sequence numbers, timestamp of checkpoint, and so on.

```
SQL> SELECT THREAD#, GROUPS, CURRENT_GROUP#, SEQUENCE#
       2       FROM V$THREAD;

       THREAD# GROUPS CURRENT_GROUP# SEQUENCE#
---------- ---------- -------------- ----------
       1       3       1              199

SQL>
```

V$LOG_HISTORY

This view contains the history of the log information. It has the log sequence number, first and highest SCN for each log change, control file ID, and so on.

```
SQL> SELECT SEQUENCE#, FIRST_CHANGE#, NEXT_CHANGE#,
  2  TO_CHAR(FIRST_TIME,'DD-MM-YY HH24:MI:SS') TIME
  3  FROM V$LOG_HISTORY
  4  WHERE SEQUENCE# BETWEEN 50 AND 53;
SEQUENCE# FIRST_CHANGE# NEXT_CHANGE# TIME
---------- ------------- ----------- -----------------
50      22622        22709        28-07-01     19:15:22
51      22709        23464        28-07-01     19:15:26
52      23464        23598        28-07-01     19:15:33
53      23598        23685        28-07-01     19:15:39
SQL>
```

V$ARCHIVED_LOG

This view displays archive log information, including archive filenames, size of the file, redo log block size, SCNs, timestamp, and so on.

```
SQL> SELECT NAME, SEQUENCE#, FIRST_CHANGE#, NEXT_CHANGE#,
  2  BLOCKS, BLOCK_SIZE
  3  FROM V$ARCHIVED_LOG
  4  WHERE SEQUENCE# BETWEEN 193 AND 194;
NAME
------------------------------------------------------------
SEQUENCE# FIRST_CHANGE# NEXT_CHANGE# BLOCKS BLOCK_SIZE
--------- ------------- ------------ ------ ----------
C:\ORACLE\ORADATA\ORADB02\ARCHIVE\ARCH_00193.ARC
       193    36549        36658        722    512
C:\ORACLE\ORADATA\ORADB02\ARCHIVE\ARCH_00194.ARC
       194    36658        36684        39     512
SQL>
```

V$ARCHIVE_DEST

This view has information about the five archive destinations, their status, any failures, and so on. The STATUS column can have six values: INACTIVE (not initialized), VALID (initialized), DEFERRED (manually disabled by the DBA), ERROR (error during copy), DISABLED (disabled after error), and BAD PARAM (bad parameter value specified). The BINDING column shows whether the target is OPTIONAL or MANDATORY, and the TARGET column indicates whether the copy is to a PRIMARY or STANDBY database.

```
SQL> SELECT DESTINATION, BINDING, TARGET, REOPEN_SECS
  2      FROM V$ARCHIVE_DEST
  3      WHERE STATUS = 'VALID';
DESTINATION          BINDING    TARGET     REOPEN_SECS
-------------------- --------   ---------  -------
C:\ARCHIVE\ORADB02\archive  MANDATORY      PRIMARY    0
D:\ARCHIVE\ORADB02\archive  OPTIONAL       PRIMARY    180
SQL>
```

V$ARCHIVE_PROCESSES

This view displays information about the state of the 10 archive processes (ARC*n*). The LOG_SEQUENCE is available only if the STATE is BUSY.

```
SQL> SELECT * FROM V$ARCHIVE_PROCESSES;
PROCESS    STATUS     LOG_SEQUENCE    STATE
-------    ----------  ------------    -----
0          ACTIVE     0               IDLE
1          STOPPED    0               IDLE
2          STOPPED    0               IDLE
3          STOPPED    0               IDLE
4          STOPPED    0               IDLE
5          STOPPED    0               IDLE
6          STOPPED    0               IDLE
7          STOPPED    0               IDLE
8          STOPPED    0               IDLE
9          STOPPED    0               IDLE
```

```
10 rows selected.

SQL>
```

Summary

In this lesson we presented two important components of the Oracle database — the control file and the redo log files. The control file records information about the physical structure of the database along with the database name, tablespace names, log sequence number, checkpoint, and RMAN information. The size of the control file depends on the five MAX clauses you specify at the time of database creation: MAXLOGFILES, MAXLOGMEMBERS, MAXLOGHISTORY, MAXDATAFILES, and MAXINSTANCES.

Oracle provides a mechanism to multiplex the control file. The information is concurrently updated to all the control files. The parameter CONTROL_FILES in the parameter file specifies the control files at the time of database creation and afterward at database start-up. You can re-create the control files by specifying all the redo log files and data files. The V$CONTROLFILE view provides the names of the control files.

Redo log files record all changes to the database. The LGWR process writes the redo log buffer information from the SGA to the redo log files. The redo log file is treated as a group. The group can have more than one member. If more than one member is present in a group, the group is known as a multiplexed group, and the LGWR process writes to all the members of the group at the same time. Even if you lose one member, LGWR continues writing with the remaining members. You can use OMF to simplify the creation and maintenance of online redo log files.

The LGWR process writes to only one redo log file group at any time. The file that is actively being written to is known as the current log file. The log files that are required for instance recovery are known as active log files. The other log files are known as inactive. A log switch occurs when Oracle finishes writing one file and starts the next file. A log switch always occurs when the current redo log file is completely full and writing must continue.

A checkpoint is an event that flushes the modified data from the buffer cache to the disk and updates the control file and data files. The checkpoint process (CKPT) updates the headers of data files and control files; the DBW*n* process writes the actual blocks to the file. You can manually initiate a checkpoint or a log switch by using the ALTER SYSTEM SWITCH LOGFILE command.

By saving the redo log files to a different (or offline storage) location, you can recover a database or audit the redo log files. The ARC*n* process does the archiving when the database is in ARCHIVELOG mode. You specify the archive destination in the initialization parameter file. The dictionary views V$LOG and V$LOGFILE provide information about the redo log files.

References

[1] Oracle9i DBA Fundamentals I.

