Unit 2: Log File Management

- Introduction to Control and Redo Log Files
- Managing the control files
- Maintaining and monitoring redo log files
- Multiplexing redo log files
- Archiving log files

Control Files

It is a small binary file that records the physical structure of the database. It includes *(can be written for application of Control files)*:

- Database Name
- Names and locations of associated datafiles and redo log files
- Timestamp of database creation
- Current log sequence number
- Checkpoint information

Control file must be available for writing by the Oracle Database server whenever the database is open. Without control file, the database cannot be mounted and recovery is difficult.

It is created at the same time as database. By default, at least one copy of control file is created but we should create two or more copies of control file during database creation and also if we lose a control file or want to change particular settings in the control files.

Managing Control Files: Managing Size of Control files:

The main determinants of the size of a control file are the values set for the

- MAXDATAFILES,
- MAXLOGFILES,
- MAXLOGMEMBERS,
- MAXLOGHISTORY, and
- MAXINSTANCES

Parameters in the CREATE DATABASE statement that created the associated database. Increasing the values of these parameters increases the size of a control file of the associated database.

Creating Control Files:

- Creating initial control files
- Creating additional copies, renaming and relocating control files
- Creating new control files

Creating Initial control file

Initial control file is created when you issue the CREATE DATABASE statement. The names of control files are specified by CONTROL_FILES parameter in the initialization parameter file during database creation.

Example:

CONTROL_FILES = (/u01/oracle/prod/control01.ctl,
/u02/oracle/prod/control02.ctl,
/u03/oracle/prod/control03.ctl)

If file with same name already exist, you must specify CONTROL FILE REUSE clause in the CREATE DATABASE statement or else error will occur. If size of old control file differ from size parameter of new one, you cannot use REUSE clause.

Creating Additional Copies (Multiplexing), Renaming and Relocating control files:

Steps for multiplexing or renaming a control file:

1. Shut down the database
2. Copy existing control file from old location to new location using operating system commands
   eg:$cp/u01/oracle/ica/control.ora,
   /u02/oracle/ica/control.ora
3. Edit the CONTROL_FILES parameter in database initialization parameter file to add the new control file name, or renaming control filename, or specifying new location for multiplexing
   eg.for multiplexing: CONTROL_FILES=/u01/oracle/ica/control.ora,
   /u02/oracle/ica/control.ora
4. Restart the database

Creating new control file

You can create a new control file for database using CREAE CONTROLFILE

```
CREATE CONTROLFILE
   SET DATABASE prod
   LOGFILE GROUP 1 ('/u01/oracle/prod/redo01_01.log',
                    '/u01/oracle/prod/redo01_02.log'),
           GROUP 2 ('/u01/oracle/prod/redo02_01.log',
                    '/u01/oracle/prod/redo02_02.log'),
           GROUP 3 ('/u01/oracle/prod/redo03_01.log',
                    '/u01/oracle/prod/redo03_02.log')
   RESETLOGS
   DATAFILE '/u01/oracle/prod/system01.dbf' SIZE 3M,
            '/u01/oracle/prod/rbs01.dbs' SIZE 5M,
            '/u01/oracle/prod/users01.dbs' SIZE 5M,
            '/u01/oracle/prod/temp01.dbs' SIZE 5M
   MAXLOGFILES 50
   MAXLOGMEMBERS 3
   MAXLOGHISTORY 400
   MAXDATAFILES 200
   MAXINSTANCES 6
   ARCHIVELOG;
```

Steps for creating new control files

1. Make a list of all datafiles and redo log files of the database.
    - SELECT MEMBER FROM V$LOGFILE;
    - SELECT NAME FROM V$DATAFILE;
    - SELECT VALUE FROM V$PARAMETER WHERE NAME = 'control_files';
2. Shut down the database. If the database is open, shut down the database normally if possible. Use the IMMEDIATE or ABORT clauses only as a last resort.
3. Back up all datafiles and redo log files of the database.
4. Start up a new instance, but do not mount or open the database: STARTUP NOMOUNT
5. Create a new control file for the database using the CREATE CONTROLFILE statement.
6. Store a backup of the new control file on an offline storage device.
7. Edit the CONTROL_FILES initialization parameter for the database to indicate all of the control files now part of your database as
8. Recover the database if necessary. If you are not recovering the database, skip to
9. Open the database using one of the following methods:
   ALTER DATABASE OPEN;

Backing up Control Files:

Backing up control files is needed every time you change the physical structure of your database. Such structural changes include:

- Adding, dropping, or renaming datafiles.
- Adding or dropping a tablespace, or altering the read/write state of the tablespace.
- Adding or dropping redo log files or groups.

Use the ALTER DATABASE BACKUP CONTROLFILE statement to back up your control files. You have two options:

- Back up the control file to a binary file (duplicate of existing control file) using the following statement:
  ALTER DATABASE BACKUP CONTROLFILE TO '/oracle/backup/control.bkp';
- Produce SQL statements that can later be used to re-create your control file:

ALTER DATABASE BACKUP CONTROLFILE TO TRACE;

Recovering a control file using a current copy

- Recovering from Control File Corruption Using a Control File Copy.
- Recovering from Permanent Media Failure Using a Control File Copy.

Recovering from Control File Corruption Using a Control File copy

This procedure assumes that one of the control files specified in the CONTROL_FILES parameter is corrupted, the control file directory is still accessible, and that you have a multiplexed copy of the control file.

1. With the instance shut down, use an operating system command to overwrite the bad control file with a good copy:
   % cp /u03/oracle/prod/control03.ctl
   /u02/oracle/prod/control02.ctl
2. Start SQL*Plus and open the database:
   SQL> STARTUP

Recovering from Permanent Media Failure Using a Control File Copy.

This procedure assumes that one of the control files specified in the CONTROL_FILES parameter is inaccessible due to a permanent media failure and that you have a multiplexed copy of the control file.

1. With the instance shut down, use an operating system command to copy the current copy of the control file to a new, accessible location:
   %cp /u01/oracle/prod/control01.ctl
   /u04/oracle/prod/control03.ctl
2. Edit the CONTROL_FILES parameter in the initialization parameter file to replace the bad location with the new location:
   CONTROL_FILES = (/u01/oracle/prod/control01.ctl,
   /u02/oracle/prod/control02.ctl, /u04/oracle/prod/control03.ctl)
3. Start SQL*Plus and open the database:
   SQL> STARTUP

Dropping Control Files

When we want to drop control files from the database, for example, if the location of a control file is no longer appropriate Remember that the database should have at least two control files at all times.

1. Shut down the database.
2. Edit the CONTROL_FILES parameter in the database initialization parameter file to delete the old control file name.
3. Restart the database.

Note: This operation does not physically delete the unwanted control file from the disk. Use operating system commands to delete the unnecessary file after you have dropped the

control file from the database.

REDO LOG FILES

Redo log files are those files that logs history of all changes that are made to database. Each redo log file consists of redo records that holds a group of change vectors, each of which describes or represents a change made to single block in the database. Redo records are buffered in a circular fashion in the redo log buffer of the SGA and are written to one of the redo log files by the Log Writer (LGWR) database background process.
Whenever something changes in a data file, oracle records the changes in redo file. If database crashes, the RDBMS can redo all changes on data file which will take back the database to the state it was when the last redo record was written.

There are **two types** of redo log files:

1. Online Redo Logs
2. Archived Redo logs

Oracle Database uses only one redo log files at a time to store redo records written from the redo log buffer. The redo log file that LGWR is actively writing to is called the **current redo log file.**
Redo log files that are required for instance recovery are called **active redo log files.**
Redo log files that are no longer required for instance recovery are called **inactive redo** log files.

If you have enabled archiving (the database is in ARCHIVELOG mode), then the database cannot reuse or overwrite an active online log file until one of the archiver background processes (ARC*n) has archived its contents. If archiving is disabled (the* database is in NOARCHIVELOG mode), then when the last redo log file is full, LGWR continues by overwriting the first available active file.

LOG SWITCHES AND LOG SEQUENCE NUMBER

A log switch is the point at which the database stops writing to one redo log file and begins writing to another.
Normally, a log switch occurs when the current redo log file is completely filled and writing must continue to the next redo log file. However, you can configure log switches to occur at regular intervals, regardless of whether the current redo log file is completely filled. You can also force log switches manually.
Oracle Database assigns each redo log file a new **log sequence number** every time a log switch occurs and LGWR begins writing to it. When the database archives redo log files, the archived log retains its log sequence number. A redo log file that is cycled back for use is given the next available log sequence number.
Each online or archived redo log file is uniquely identified by its log sequence number. During crash, instance, or media recovery, the database properly applies redo log files in ascending order by using the log sequence number of the necessary archived and redo log files.

MAINTAINING AND MONITORING REDO LOG FILES

- Multiplexing Redo Log Files.
- Placing Redo Log Members on Different Disks.
- Setting the Size of Redo Log Members.
- Choosing the Number of Redo Log Files.
- Controlling Archive Lag.

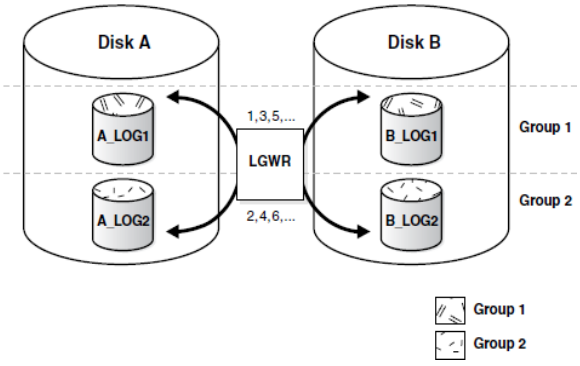
Multiplexing Redo Log Files

Redo log files must be multiplexed in separate location (better in separate disks) in order to protect it against the failure involving the redo log itself. The redundancy of redo log file can help protect against I/O errors, file corruption and so on.
Multiplexing is implemented by creating group that consists of redo log file and its multiplexed copies where each group is defined by a number, such as group 1, group 2, etc.

Placing Redo log Members on Different Disks

When setting up a multiplexed redo log, place members of a group on different physical disks. If a single disk fails, then only one member of a group becomes unavailable to LGWR and other members remain accessible to LGWR, so the instance can continue to function.

Figure 10–2 Multiplexed Redo Log Files



Setting size of Redo log members

Size of Redo log file should be set in such a way that a filled group can be archived to a single unit of storage media with least amount of space on the medium left unused. All members of the same multiplexed redo log group must be of same size. The minimum size permitted for a redo log file is 4 MB.

Choosing number of Redo Log files

The MAXLOGFILES parameter used in CREATE DATABASE statement determines the maximum number of groups of redo log files for each database. The group value can range

from 1 to MAXLOGFILES. If MAXLOGFILES is not specified for the CREATE DATABASE statement, then the database uses an operating system specific default value.

Controlling Archive Lag

In primary/standby database configuration, changes are made available to standby database by archiving redo logs at primary site and then shipping them to standby database. The changes that are being applied to standby database can lag the changes that are occurring on primary database as the standby database must wait for the changes in primary database redo log to be archived and then shipped to it.
To limit this lag, you can set ARCHIVE_LAG_TARGET initialization parameter and specify in seconds how long that lag can be.
eg: ARCHIVE_LAG_TARGET=1800

Creating redo log group

To create a new group of redo log files, use the SQL statement ALTER DATABASE with the ADD LOGFILE clause.
ALTER DATABASE ADD LOGFILE ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo') SIZE 4M;
You can also specify the number that identifies the group using the GROUP clause:
ALTER DATABASE ADD LOGFILE GROUP 10 ('/oracle/dbs/log1c.rdo', '/oracle/dbs/log2c.rdo') SIZE 4M;

Creating Redo log members:

In some cases, it might not be necessary to create a complete group of redo log files. A group could already exist, but not be complete because one or more members of the group were dropped (for example, because of a disk failure). In this case, you can add new members to an existing group.
To create new redo log members for an existing group
ALTER DATABASE ADD LOGFILE MEMBER '/oracle/dbs/log2b.rdo' **TO GROUP 2;**

RELOCATING AND RENAMING REDO LOG MEMBERS

You can use operating system commands to relocate redo logs, then use the ALTER DATABASE statement to make their new names (locations) known to the database.
To rename redo log members, you must have the ALTER DATABASE system privilege.
Before relocating your redo logs, or making any other structural changes to the database, completely back up the database in case you experience problems while performing the operation. As a precaution, after renaming or relocating a set of redo log files, immediately back up the database control file.
Steps:

1. Shut down the database. SQL>SHUTDOWN

2. Copy the redo log files to the new location.
3. Startup the database, mount, but do not open it. SQL> CONNECT / as SYSDBA
   SQL>STARTUP MOUNT
4. Rename the redo log members.
   Use the ALTER DATABASE statement with the RENAME FILE clause to rename the database redo log files.
   ALTER DATABASE
   RENAME FILE '/diska/logs/log1a.rdo', '/diska/logs/log2a.rdo' TO '/diskc/logs/log1c.rdo', '/diskc/logs/log2c.rdo';
5. Open the database for normal operation. The redo log alterations take effect when the database is opened.
   ALTER DATABASE OPEN;

DROPPING REDO LOG GROUP

To drop a redo log group, you must have the ALTER DATABASE system privilege.
Before dropping a redo log group, consider the following restrictions and precautions:

- An instance requires at least two groups of redo log files, regardless of the number of members in the groups. (A group comprises one or more members.)
- You can drop a redo log group only if it is inactive. If you need to drop the current group, first force a log switch to occur.
- Make sure a redo log group is archived (if archiving is enabled) before dropping it.

To see whether this has happened, use the V$LOG view. Example: SELECT GROUP#, ARCHIVED, STATUS FROM V$LOG;

GROUP#	ARC	STATUS
1	YES	ACTIVE
2	NO	CURRENT
3	YES	INACTIVE
4	YES	INACTIVE

Drop a redo log group with the SQL statement ALTER DATABASE with the DROP LOGFILE clause.
E.g. ALTER DATABASE DROP LOGFILE GROUP 3;

DROPPING REDO LOG MEMBERS

To drop a redo log member, you must have the ALTER DATABASE system privilege.
You can drop a redo log member only if it is *not part of an active or current group.* If you want to drop a member of an active group, first force a log switch to occur.
To drop specific inactive redo log members, use the ALTER DATABASE statement with the

DROP LOGFILE MEMBER clause.
The following statement drops the redo log /oracle/dbs/log3c.rdo: ALTER DATABASE DROP LOGFILE MEMBER '/oracle/dbs/log3c.rdo';

FORCING LOG SWITCHES

A log switch occurs when LGWR stops writing to one redo log group and starts writing to another. By default, a log switch occurs automatically when the current redo log file group fills.
You can force a log switch to make the currently active group inactive and available for redo log maintenance operations.
For example, you want to drop the currently active group, but are not able to do so until the group is inactive.
To force a log switch, you must have the ALTER SYSTEM privilege. Use the ALTER SYSTEM statement with the SWITCH LOGFILE clause.
The following statement forces a log switch: ALTER SYSTEM SWITCH LOGFILE

MANAGING ARCHIVED REDO LOG FILES

Oracle Database lets you save filled groups of redo log files to one or more offline destinations, known collectively as the **archived redo log.**
The process of turning redo log files into archived redo log files is called **archiving. This process is only possible if** the database is running in **ARCHIVELOG mode.**
You can choose automatic or manual archiving.

Use of Archived Redo Log Files

- Recover a database.
- Update a standby database.
- Get information about the history of a database using the Log-Miner utility.

Running Database in NONARCHIVELOG mode

Running database in NOARCHIVELOG mode, disables archiving of the redo log. When a filled group becomes inactive after a log switch, the group is available for reuse by LGWR.
NOARCHIVELOG mode protects a database from instance failure but not from media failure.
In NOARCHIVELOG mode you cannot perform online tablespace backups, nor can you use online tablespace backups taken earlier while the database was in ARCHIVELOG mode.
To restore a database operating in NOARCHIVELOG mode, you can use only whole database backups taken while the database is closed. Therefore, if you decide to operate a database in NOARCHIVELOG mode, take whole database backups at regular, frequent intervals.

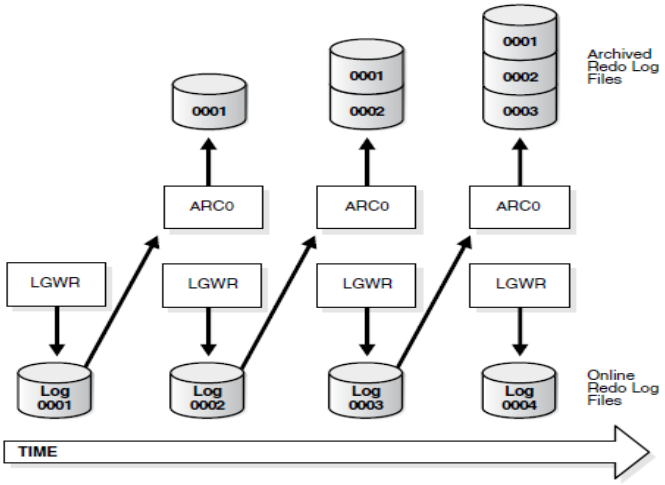
Running Database in ARCHIVELOG mode

Running database in ARCHIVELOG mode, enables the archiving of the redo log. A filled group becomes available for archiving immediately after a redo log switch occurs and can be reused by LGWR once they are finished archiving.

Advantages

- A database backup, together with online and archived redo log files, guarantees that you can recover all committed transactions in the event of an operating system or disk failure.
- If you keep an archived log, you can use a backup taken while the database is open and in normal system use.
- You can keep a standby database current with its original database by continuously applying the original archived redo logs to the standby.
- You can configure an instance to archive filled redo log files automatically, or you can archive manually. For convenience and efficiency, automatic archiving is usually best.

Figure 11–1 Redo Log File Use in ARCHIVELOG Mode



Steps for switching from NONARCHIVELOG mode to ARCHIVELOG mode

1. Shut down the database instance. SHUTDOWN
2. Back up the database:

   Before making any major change to a database, always back up the database to protect against any problems. This will be your final backup of the database in NOARCHIVELOG mode and can be used if something goes wrong during the change to ARCHIVELOG mode.
3. Edit the initialization parameter file to include the initialization parameters that specify the destinations for the archived redo log files.
4. Start a new instance and mount, but do not open, the database.

   STARTUP MOUNT

   To enable or disable archiving, the database must be mounted but not open.
5. Change the database archiving mode. Then open the database for normal operations.

   ALTER DATABASE ARCHIVELOG;

   ALTER DATABASE OPEN;
6. Shut down the database.

   SHUTDOWN IMMEDIATE

7. Back up the database:

Changing the database archiving mode updates the control file. After changing the database archiving mode, you must back up all of your database files and control file. Any previous backup is no longer usable because it was taken in NOARCHIVELOG mode.