10. Instance and Media Recovery Structures.

Abstract: Oracle uses a wide variety of processes and structures to provide a robust set of recovery options. A process is a daemon, or background program, that performs certain tasks. A structure is either a physical or logical object that is part of the database, such as files or database objects themselves. The processes consist of log writer (LGWR), system monitor (SMON), process monitor (PMON), checkpoint (CKPT), and archiver (ARCn). The available structures include redo logs, rollback segments, control files, and data files. This lesson provides details on these, including how different combinations of these processes and structures are used to recover from different kinds of failures. This lesson also introduces methods you can use to tune instance recovery, including the ability to set approximate limits on the length of time needed to recover from instance failure. We discuss and provide examples of these methods. In order to understand the backup and recovery process, you must first understand processes and structures. This knowledge will help you, as the DBA, make real-life decisions about backup and recovery situations. For example, if you understand how the database physical structures get synchronized, you will be able to make sense of the recovery process. If you truly understand how processes and structures interact with backup and recovery procedures, you are on the road to having a solid understanding of the backup and recovery process.

Contents

1. Oracle Recovery Processes, Memory Components, and File Structures 2
   1.1. Recovery Processes....2
   1.2. Memory Structures 3
   1.3. File Structures....4
2. Checkpoints, Redo Logs, and Archived Logs....8
3. Ways to Tune the Instance Recovery....10
4. Summary 12
References....12

Objectives:

- Describe the Oracle processes, memory structures, and files relating to recovery.
- Identify the importance of checkpoints, redo log files, and archived log files.
- Describe ways to tune instance recovery.

1. Oracle Recovery Processes, Memory Components, and File Structures

Oracle recovery processes, memory components, and file structures all work together during recovery in order to maintain the physical integrity of the database. Oracle recovery processes interact with Oracle memory components to coordinate data blocks that are written to disk. These blocks of data reside in memory for faster access at different times of database operations. As these memory structures change, coordination with the physical and logical structures occurs so that the database can remain consistent.

Basically, recovery processes coordinate what blocks (and other information) need to be read (or modified from the data blocks in memory) and then written to disk or to the physical and logical structures in the form of online redo logs, archived logs, or data files. Each process has specific tasks that it fulfills.

These physical and logical structures are like memory structures in that they are made up of data blocks. But the *physical structures* are static structures that consist of files in the OS file system. Data blocks and other information are written to these physical structures to make them consistent. The *logical structures* are temporarily used to hold parts of information for intermediate time periods, or until the processes can permanently record the appropriate information in the physical structures.

1.1. Recovery Processes

Oracle has five major processes related to recovery. These processes include log writer, system monitor, process manager, checkpoint, and archiver. Let's look at each of these processes in more detail.

Log writer (**LGWR**) The log writer (LGWR) process writes redo log entries from the redo buffers. A redo log entry is any change, or transaction, that has been applied to the database, committed or not. (To commit means to save or permanently store the results of the transaction to the database.) The LGWR process is mandatory and is started by default when the database is started.

System monitor (**SMON**) The system monitor (SMON) process performs a varied set of functions. SMON is responsible for instance recovery, and it also performs temporary segment cleanup. It is a mandatory process and is started by default when the database is started.

Process monitor (**PMON**) The process monitor (PMON) process performs recovery of failed user processes. This is a mandatory process and is started by default when the database is started.

Checkpoint (**CKPT**) The checkpoint (CKPT) process performs checkpointing in the control files and data files. Checkpointing is the process of stamping a unique counter in the control files and data files for database consistency and synchronization. As of Oracle8, the CKPT process was made mandatory and is started by default.

Archiver (**ARCn**) The archiver (ARCn) process performs the copying of the online redo log files to archived log files. The ARCn process is enabled only if the init.ora

file's parameter LOG_ARCHIVE_START is set to TRUE or with the ARCHIVE LOG START command. This isn't a mandatory process.

In the Windows 2000/XP environments, the processes are threads of the main Oracle executable. In the Unix environment, an example of each of these processes can be seen by typing the Unix command **ps –ef | grep < $ORACLE_SID >.$ORACLE_SID** is a Unix environment variable that identifies the Oracle system identifier. In this case, $ORACLE_SID is orc9.

```
oracle@octilli:~ > ps -ef|grep orc9
oracle 2077 1 0 00:26 ? 00:00:00 ora_pmon_orc9
oracle 2079 1 0 00:26 ? 00:00:00 ora_dbw0_orc9
oracle 2081 1 0 00:26 ? 00:00:00 ora_lgwr_orc9
oracle 2083 1 0 00:26 ? 00:00:00 ora_ckpt_orc9
oracle 2085 1 0 00:26 ? 00:00:00 ora_smon_orc9
oracle 2087 1 0 00:26 ? 00:00:00 ora_reco_orc9
oracle 2097 1 0 00:26 ? 00:00:00 ora_arc0_orc9
```

1.2. Memory Structures

There are two Oracle memory structures relating to recovery: log buffers and data block buffers. The *log buffers* are the memory buffers that record the changes, or transactions, to data block buffers before they are written to online redo logs or disk. Online redo logs record all changes to the database, whether the transactions are committed or rolled back.

The *data block buffers* are the memory buffers that store all the database information. A data block buffer stores mainly data that needs to be queried, read, changed, or modified by users. The modified data block buffers that have not yet been written to disk are called *dirty buffers*. At some point, Oracle determines that these dirty buffers must be written to disk. When this happens, a checkpoint occurs.

Both Oracle memory structures can be viewed in a number of ways. The most common method is by performing a SHOW SGA command from SQL. This displays all of the memory sizes of the database that you are connected to. See the following example:

```
oracle@octilli:/oracle/admin/orc9/pfile > sqlplus /nolog
SQL*Plus: Release 9.0.1.0.0 - Production on Tue Sep 25
00:38:13 2001
(c) Copyright 2001 Oracle Corporation. All rights reserved.
SQL> connect /as sysdba
Connected.
SQL> show sga
```

```
Total System Global Area      235693104 bytes
Fixed Size                    279600 bytes
Variable Size                 167772160 bytes
Database Buffers              67108864 bytes
Redo Buffers                  532480 bytes
SQL>
```

When you look at this code example, you will see that the database buffers are approximately 67MB, and the redo buffers are approximately 512KB. When the SHOW SGA command is run, the data block buffers are referred to as *database buffers*, and the log buffers are referred to as *redo buffers*. These values are extremely small and are suitable for a sample database or for testing. Data block buffers can be about 100MB to 300MB for average-sized databases with a few hundred users.

In this System Global Area (SGA) output, Variable Size pertains to the SHARED_POOL and LARGE_POOL values in the init.ora file. The SHARED_POOL value stores parsed versions of SQL and PL/SQL so that it may be reused. The LARGE_POOL value provides large memory sizes for shared servers session memory as well as backup and recovery operations. The init.ora file's DBWR_IO_SLAVES and BACKUP_TAPE_IO_SLAVES parameters are examples of backup and recovery operations that will use the LARGE_POOL memory. In the SGA output above, Fixed Size pertains to a few less-critical parameters in the init.ora.

1.3. File Structures

The Oracle file structures relating to recovery include the online redo logs, archived logs, control files, data files, and parameter files. The redo logs consist of files that record all the changes to the database. The archived logs consist of files that are copies of the redo logs; these exist so that a historical record of all the changes made to the database can be utilized if necessary. Control files are binary files that contain the physical structure of the database, such as operating system filenames of all files that make up the database. Data files are physical structures of the database that make up a logical structure called a tablespace. All data is stored within some type of object within a tablespace. Parameter files are the files that contain the initialization parameters for the database. These are the files that set the initial settings or values of database resources when the database is started. Let's look at each of these in more detail.

Redo Logs

The *redo logs* consist of files that record all of the changes to the database. Recording all changes is one of the most important activities in the Oracle database from the recovery standpoint. The redo logs get information written to them before all other physical structures in the database. The purpose of the redo log is to protect against data loss in the event of a failure.

The term *redo log* includes many subclassifications. Redo logs consist of *online redo logs*, *offline redo logs* (also called archived logs), *current online redo logs*, and *non-current online redo logs*. Each is described below.

Online redo logs Logs that the log buffers are writing to in a circular fashion. These logs are written and rewritten.

Offline redo logs, **or archived logs** Copies of the online redo logs before they are written over by the LGWR.

Current online redo logs Logs that are currently being written to and therefore are considered active.

Non-current redo logs Online redo logs that are not currently being written to and therefore are inactive.

Each database has at least two sets of online redo logs, but Oracle recommends at least three sets. You will see why when we discuss archived logs in the next section. Redo logs record all the changes that are made to the database; these changes result from the LGWR writing out log buffers to the redo logs at particular events or points in time.

As mentioned previously, redo logs are written in a circular fashion. That is, if there are three sets of logs, log 1 gets written to first until it is full. Then Oracle moves to log 2, and it gets written to until it is full. Then Oracle moves to log 3, and it gets written to until it is full. Oracle then goes back to log 1, writes over the existing information, and repeats this process over again. Here is a listing of the logs from the V$LOGFILE view.

```
SQL> select group#, member from v$logfile;

GROUP#     MEMBER
---------- ----------------------------------------
         3 /oracle/oradata/orc9/redo03.log
         2 /oracle/oradata/orc9/redo02.log
         1 /oracle/oradata/orc9/redo01.log

3 rows selected.
```

Figure 1 shows an example of the circular process of redo file generation, which writes to one log at a time, starting with log 1, then log 2, then log 3, and then back to log 1 again.

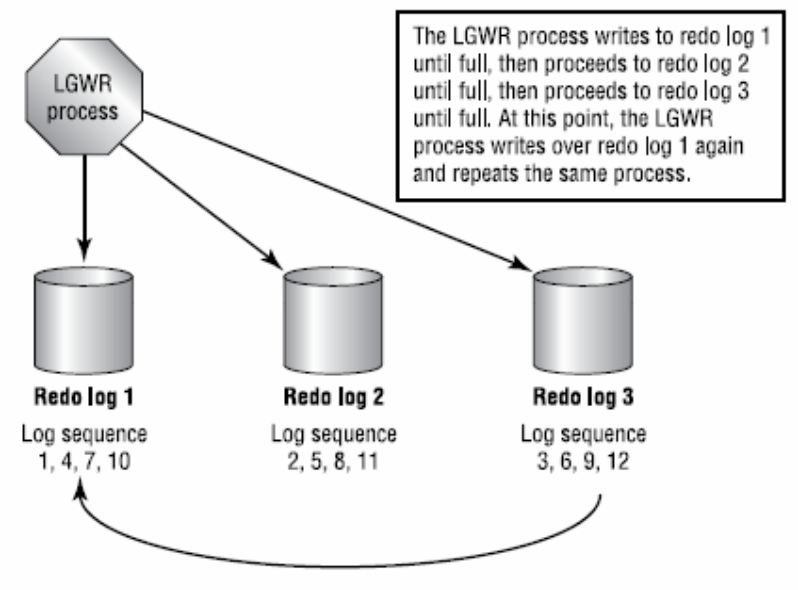

FIGURE 1. The circular process of redo file generation.

Redo logs contain values called system change numbers (SCNs) that uniquely identify each committed transaction in the database. SCNs are like a clock of events that have occurred in the database, and they are one of the major synchronization elements in recovery. Each data-file header and control file is synchronized with the current highest SCN.

Archived Logs

Archived logs are non-current online redo logs that have been copied to a new location offline. This location is the value of the init.ora file's LOG_ARCHIVE_DEST parameter. Archived logs are created if the database is in ARCHIVELOG mode rather than NOARCHIVELOG mode. Archived logs are also referred to as offline redo logs.

An archived log is created when a current online redo log is completed or filled, and before an online redo log needs to be written to again. Remember, redo logs are written to in a circular fashion. If there are only two redo log sets and you are in ARCHIVELOG mode, the LGWR may have to wait or halt the writing of information to redo logs while an archived log is being copied. If it doesn't, the LGWR process would overwrite the archive information, making the archived log useless. If at least three redo log groups are available, the archived log will have enough time to be created and not cause the LGWR to wait for an available online redo log. This is under average transaction volumes. Some large or transaction-intensive databases may have 10 to 20 log sets to reduce the contention on redo log availability.

Archived logs are the copies of the online redo logs; the archived logs get applied to the database in certain types of recovery. Archived logs build the historical transactions, or changes, back in the database to make it consistent to the desired recovery point.

Control Files

A *control file* is a binary file that stores the information about the physical structures that make up the database. The physical structures are OS objects, such as all OS filenames. The control file also stores the highest SCN to assist in the recovery process. This file stores information about the backups if you are using RMAN, the database name, and the date the database was created.

You should always have at least two control files on different disk devices. Having this duplication is called multiplexing your control files. Multiplexing control files is done in an init.ora file's CONTROL_FILES parameter.

Every time a database is mounted, the control file is accessed to identify the data files and redo logs that are needed for the database to function. All new physical changes to the database get recorded in the control file by default.

Data Files

Data files are physical files stored on the file system. All Oracle databases have at least one data file, but usually more. Data files are where the physical and the logical structures meet. Tablespaces are logical structures and are made up of one or more data files. All logical objects reside in tablespaces.

Logical objects are those that do not exist outside of the database, such as tables, indexes, sequences, and views.

Data files are made up of blocks. These *data blocks* are the smallest unit of measure in the database. The logical objects, such as tables and indexes, are stored in the data blocks, which reside in the data files.

The first block of every file is called a *header block*. The header block of the data file contains information such as the file size, block size, and associated tablespace. It also contains the SCN for recovery purposes.

Here is the output from the V$DATAFILE view, showing the data files that make up a sample database.

```
SQL> select file#, status, substr(name,0,50) from v$datafile;

FILE#      STATUS     SUBSTR(NAME,0,50)
---------- ------- --------------------------------------------------
         1 SYSTEM     /oracle/product/9.0.1/oradata/orc9/system01.dbf
         2 ONLINE     /oracle/product/9.0.1/oradata/orc9/undotbs01.dbf
         3 ONLINE     /oracle/product/9.0.1/oradata/orc9/cwmlite01.dbf
         4 ONLINE     /oracle/product/9.0.1/oradata/orc9/drsys01.dbf
         5 ONLINE     /oracle/product/9.0.1/oradata/orc9/example01.dbf
         6 ONLINE     /oracle/product/9.0.1/oradata/orc9/indx01.dbf
```

```
         7      ONLINE      /oracle/product/9.0.1/oradata/orc9/tools01.dbf
         8      ONLINE      /oracle/product/9.0.1/oradata/orc9/users01.dbf
8 rows selected.
SQL>
```

Parameter Files

There are two parameter files that make up the file structures of the Oracle9i database. The *parameter file* is the file that contains the initialization parameters of the database that are used upon startup. The standard parameter file, called *init<ORACLE_SID>.ora*, contains parameters required for instance startup. The ORACLE_SID equals the name of the oracle system identifier or database name. This parameter file is ASCII and can be edited.

The second parameter file, called *spfile<ORACLE_SID>.ora*, is the server parameter file that stores persistent parameters that are required for instance startup and those that are modified when the database is started.

The server parameter file is stored in binary format, which is a new feature in Oracle9i.

Either of these files can be used to start the Oracle database, but the spfile.ora files are the default for starting the database. If these files are not found, then the init.ora files are used.

The default locations for these files are in the following directory structures. We will demonstrate both Unix and Windows NT/2000 locations.

Unix	$ORACLE_HOME/dbs
Windows NT/2000	%ORACLE_HOME%\database

2. Checkpoints, Redo Logs, and Archived Logs

Now that you have a basic understanding of the various Oracle processes, file structures, and memory structures used for recovery, it's time to see how these interrelate. As you learned earlier, checkpoints, redo logs, and archived logs are significant to all aspects of recovery.

The *checkpoint* is an event that determines the synchronization or consistency of all transactions on disk. The checkpoint is implemented by storing a unique number—the SCN (again, this stands for system change number)—in the control files, header of the data files, online redo logs, and archived logs.

The checkpoint is performed by the CKPT process. One of the ways a checkpoint is initiated is by the data block writer (DBWR) process. The DBWR process initiates a checkpoint by writing all modified data blocks in the data buffers (dirty buffers) to the data files. After a checkpoint is performed, all committed transactions are written to the data files.

If the instance were to crash at this point, only new transactions that occurred after this checkpoint would need to be applied to the database to enable a complete recovery. Therefore, the checkpoint process determines which transactions from the redo logs need to

be applied to the database in the event of a failure and subsequent recovery. Remember that all transactions, whether committed or not, get written to the redo logs.

Other methods that cause a checkpoint to occur include any of the following commands: ALTER SYSTEM SWITCH LOGFILE, ALTER SYSTEM CHECKPOINT LOCAL, ALTER TABLESPACE BEGIN BACKUP, and ALTER TABLESPACE END BACKUP.

SCNs are recorded within redo logs at every log switch, at a minimum. This is because a checkpoint occurs at every log switch. Archived logs have the same SCNs recorded within them as the online redo logs because the archived logs are merely copies of the online redo logs.

Let's look at an example of how checkpointing, online redo logs, and archived logs are all interrelated. First, the ALTER TABLESPACE BEGIN BACKUP command is used to begin an online backup of a database.

> *NOTE*: An *online backup*, also called a hot backup, occurs while the database is still available or running.

The ALTER TABLESPACE BEGIN BACKUP command is followed by an OS command to copy the files, such as cp in Unix. Then, the command ALTER TABLESPACE END BACKUP is used to end the hot backup. As we just discussed, these ALTER TABLESPACE commands also cause a checkpoint to occur. The following is an example of the data file data01.dbf for the tablespace DATA being backed up:

```
SQL> connect /as sysdba

Connected.

SQL> alter tablespace data begin backup;

Tablespace altered.

SQL> ! cp data01.dbf /stage/data01.dbf

SQL> alter tablespace data end backup;

Tablespace altered.

SQL>
```

Note that the tablespace DATA was put in backup mode, and then the OS command was executed, copying the data file data01.dbf to the new directory /stage, where it awaits writing to tape. Finally, the tablespace DATA was taken out of backup mode. These steps are repeated for every tablespace in the database. This is a simplified example of a hot backup.

If a hot backup was taken on Sunday at 2 A.M., and the database crashed on Tuesday at 3 P.M., then the last checkpoint would have been issued after all the data files were backed up. This backup would be the last checkpointed disk copy of the database. Therefore, all the archived logs generated after the 2 A.M. backup was completed would need to be applied to the checkpointed database to bring it up to 3 P.M. Tuesday, the time of the crash. When you are making a hot backup of the database, you are getting a copy of the database that has been checkpointed for each data file. In this case, each data file has a different SCN stamped in the

header and each will need all applicable redo log entries made with a greater SCN applied to the data file to make the database consistent.

3. Ways to Tune the Instance Recovery

Instance failure is when the Oracle database instance abnormally terminates due to something like a power outage or a shutdown abort. *Instance recovery* is the automatic process that the Oracle database performs to ensure that the database is functioning properly and the data is consistent. This is also known as the *roll forward and roll backward* process. Upon startup of the Oracle database after an instance failure, Oracle reads the current online redo log and applies all the changes in that redo log to the database. Any uncommitted changes are then rolled back. Thus, the database is made consistent from the time of the outage.

The concept of defining an approximate set time for the instance recovery process is called bounded time recovery. *Bounded time recovery* means that the DBA controls or puts bounds on the time it takes for an instance to recover after instance failure. These bounds are controlled by using two initialization parameters.

There are two primary initialization parameters that will speed up the instance recovery process. The first parameter is called FAST_START_MTTR_TARGET. This parameter makes the database writer (DBW0 background process) write dirty blocks faster and at a predefined pace to meet agreed-upon recovery timeframes. This parameter can be set from 0 to 3600 seconds.

The second parameter is called FAST_START_PARALLEL_ROLLBACK. This parameter determines the maximum number of processes that can exist for performing a parallel rollback. This parameter is useful when there are long running transactions involved in the instance recovery process. This causes the rollback aspect of instance recovery to process faster. This parameter can be set as HIGH, LOW, and FALSE. The following are sample init.ora file parameters:

```
fast_start_mttr_target          = 300
fast_start_parallel_rollback    = LO
```

There are three V$ sys user views that can be queried to monitor these parameters. These views are V$INSTANCE_RECOVERY, V$FAST_START_SERVERS, and V$FAST_START_TRANSACTIONS. Below is an example of the V$INSTANCE_RECOVERY view, which can be used to monitor and determine the length of recovery for the FAST_START_MTTR_TARGET parameter. TARGET_MTTR and ESTIMATED_MTTR show the estimated time in seconds to recover the database in the advent of an instance failure at that given time.

```
SQL*Plus: Release 9.0.1.0.0 - Production on Mon Sep 24
23:14:36 2001
(c) Copyright 2001 Oracle Corporation. All rights reserved.
Connected to:
```

```
Oracle9i Enterprise Edition Release 9.0.1.0.0 - Production
With the Partitioning option
JServer Release 9.0.1.0.0 - Production
SQL> select target_mttr,estimated_mttr from v$instance_recovery;
TARGET_MTTR      ESTIMATED_MTTR
----------------  -------------------
        29              17
SQL>
```

Real World Scenario

Instance Recovery in a Distributed Database Environment

Let's look at a real-life example to determine the circumstances that can cause data loss even though Oracle *instance recovery* accounts for all data loss properly.

A small manufacturing company generates their manufacturing information on an Oracle database. This database keeps track of raw materials, work in progress, and finished products. In order to complete the work required of it, the database needs to be available throughout the three shifts that the company runs to meet customer demand.

For financial purposes, on a regular basis, information from this manufacturing database is transferred to the financial database on another server. This is done through a variety of out-of-database transfers, which leaves the data exposed until it meanders its way back into the financial database.

In the midst of one of these transfers, the company experiences a severe power spike, which causes the servers to reboot. As a result, instance failure occurs on each database because data was in the process of being transferred from the manufacturing database to the financial database when the power spike occurred. In addition, the database transfer mechanism did not recover the data in transit.

At this point, the experienced database administrator can see that the issue here is a loosely distributed database environment. This could result from the company using different vendors for manufacturing and financial applications with customized application program interfaces (APIs).

Now that the power spike is over, the system administrator must get the servers restarted and the database administrator needs to validate that the databases are available again. When this is done, the administrators find out that the manufacturing database thinks that the data has been transferred to financial database, but the data never actually made it into the financial database before the instance failure.

As a result, the databases are out of synchronization with each other and the data inconsistencies must be manually tracked down. The analysts and business people must determine what did or didn't make it to the appropriate database by a series of ad hoc SQL

statements.

4. Summary

The recovery structures and processes available in Oracle allow significant recovery options for the DBA. These recovery structures consist of files, processes, memory buffers, and logical objects that reside in the database. In addition to these recovery structures, this lesson also identified the file structures—redo logs, archived logs, control files, data files, and parameter files—and the processes, which are PMON, SMON, LGWR, CKPT, and ARCn. It then discussed the memory structures, which consist of log buffers and data buffers. All of these structures play different roles in the recovery process, and they can be used for different types of recovery.

This lesson also described the checkpoint concept. During this discussion, you learned the importance of SCNs and how these identifiers help determine the consistency of the transactions that have been applied to the database.

In addition, you viewed the three main initialization parameters that you will need to tune the instance recovery process. It is through these that the DBA can constrain the time of the recover process to certain approximate limits. These parameters can be monitored by a series of V$ sys views.

This lesson lays the groundwork for making decisions in the backup and recovery process for later chapters and the real world. It is vital that you understand how the file structures and process that are discussed in this lesson function to make the database consistent or inconsistent in a failure situation. These topics can be thought of as the building blocks of the backup and recovery processes.

References

[1] Oracle9i DBA Fundamentals II.