# Practices for Lesson 6: Managing Database Instances

Practices for Lesson 6: Overview

Overview

In these practices, you will learn more about initialization parameters. You will also learn how to view diagnostic information.

Practice 6-1: Investigating Initialization Parameter Files

Overview

In this practice, you investigate how the Oracle Database server uses initialization parameter files to start the database instance.

Assumptions

You are logged in as the oracle user.

The orclcdb database instance has been started.

Tasks

Use oraenv to set the environment variables for the orclcdb database.

Start SQL\*Plus and connect to the root container as the SYS user with the SYSDBA

privilege.

Locate the default SPFILE for your database instance by using the SHOW PARAMETER command. The results show that the SPFILE is in the ORACLE\_HOME/dbs directory. The output in the code box has been formatted for legibility.

View the init.ora file. This is the sample text initialization parameter file (PFILE) provided with the Oracle Database installation.

Use the SQL\*Plus HOST command to return to the operating system prompt.

Change to the $ORACLE\_HOME/dbs directory and use the ls command to list the contents of the directory.

Notice that the SPFILE (spfileorclcdb.ora) and init.ora files are stored here. The naming convention for an SPFILE is spfile<SID>.ora.

Use the cat or more command to view the contents of the sample text initialization parameter file (PFILE), init.ora. Then exit from the HOST shell back to SQL\*Plus.

# wbridge 12/03/91 - use of %c in archive format is discouraged

# ghallmar 12/02/91 - add global\_names=true, db\_directory=us.acme.com

################################################################ ##############

# Example INIT.ORA file #

# This file is provided by Oracle Corporation as a starting point for

# customizing the Oracle Database installation for your site. #

# NOTE: The values that are used in this file are example values only.

# You may want to adjust those values for your specific requirements.

# You might also consider using the Database Configuration Assistant

# tool (DBCA) to create a server-side initialization parameter file

# and to size your initial set of tablespaces. See the # Oracle Database 2 Day DBA guide for more information.

################################################################ ###############

# Change '<ORACLE\_BASE>' to point to the oracle base (the one you specify at

# install time)

db\_name='ORCL' memory\_target=1G processes = 150

audit\_file\_dest='<ORACLE\_BASE>/admin/orcl/adump' audit\_trail ='db'

db\_block\_size=8192 db\_domain=''

db\_recovery\_file\_dest='<ORACLE\_BASE>/fast\_recovery\_area'

The initorclcdb.ora file does not exist. So create it in SQL\*Plus. The '!'

character is a shortcut for the host command.

Now use the cat or more command to view the text initialization parameter file,

initorclcdb.ora.

Return to SQL\*Plus.

If the database server doesn't find an SPFILE, then the text initialization parameter file will be used. Now you’ll set up a test to see how the search works when you start the database instance.

Shut down the database instance in IMMEDIATE mode.

Use the HOST command or ! to return to an operating system prompt.

Change to the $ORACLE\_HOME/dbs directory.

Rename the spfileorclcdb.ora file to spfileorclcdb.ora\_original . Renaming this file will take it out of the search order for parameter files when you start up the database instance. Instead, the database server will automatically find the initORCLCDB.ora file (PFILE) to start the database instance.

Return to SQL\*Plus.

Start the database instance by using the STARTUP command.

Verify that the database instance was started with your PFILE by issuing the SHOW PARAMETER spfile command. The value is null, which means the database instance was started with a PFILE.

Configure the database instance to once again start with the SPFILE.

Shut down the database instance in IMMEDIATE mode.

Use the HOST command to return to the operating system.

Change to the $ORACLE\_HOME/dbs directory.

Rename the orig\_spfileorclcdb.ora file to spfileorclcdb.ora.

Return to SQL\*Plus.

Start the database instance by using the STARTUP command.

Verify that the database instance was started with the SPFILE.

Exit SQL\*Plus

Practice 6-2: Viewing Initialization Parameters by Using SQL\*Plus

Overview

In this practice, you view initialization parameters (parameters) by using SQL\*Plus. You do this in two ways:

By using the SHOW PARAMETER command

By querying the following views: V$PARAMETER, V$SPPARAMETER, V$PARAMETER2,

and V$SYSTEM\_PARAMETER

Assumptions

You are logged in as the oracle OS user.

Tasks

View Basic Parameters

In this section, you view basic parameters by using the SHOW PARAMETER command. Basic parameters are those parameters that you are likely to modify.

Be sure your environment is using the orclcdb database.

Start SQL\*Plus and connect to the root container as the SYS user with the SYSDBA

privilege.

View the values of the DB\_NAME and DB\_DOMAIN parameters. Together, these values create the global database name.

View the value of the DB\_NAME parameter. This parameter specifies the current database identifier of up to eight characters. If you have multiple databases, the value of this parameter should match the Oracle instance identifier of each one to avoid confusion with other databases running on the system.

View the value of the DB\_DOMAIN parameter. In a distributed database system, DB\_DOMAIN specifies the logical location of the database within the network structure. You should set this parameter if this database is or ever will be part of a distributed system. There is no default value.

View the DB\_RECOVERY\_FILE\_DEST and DB\_RECOVERY\_FILE\_DEST\_SIZE parameters. These parameters set the location of the fast recovery area and its size.

The DB\_RECOVERY\_FILE\_DEST parameter specifies the default location for the fast recovery area. The fast recovery area contains multiplexed copies of current control files and online redo logs, as well as archived redo logs, flashback logs, and Recovery Manager (RMAN) backups. If you specify a value for DB\_RECOVERY\_FILE\_DEST, you must also specify a value for the DB\_RECOVERY\_FILE\_DEST\_SIZE initialization parameter.

The DB\_RECOVERY\_FILE\_DEST\_SIZE parameter specifies (in bytes) the hard limit on the total space to be used by target database recovery files created in the fast recovery area.

SQL> show parameter db\_recovery\_file\_dest

db\_recovery\_file\_dest\_size big integer 14970M

SQL>

View the SGA\_TARGET and SGA\_MAX\_SIZE parameters.

SGA\_TARGET specifies the total amount of SGA memory available to a database instance, and SGA\_MAX\_SIZE sets a maximum size for the SGA.

If you set the SGA\_TARGET parameter, you enable the Automatic Shared Memory Management (ASMM) feature. The Oracle Database server will automatically distribute memory among the various SGA memory pools (buffer cache, shared pool, large pool, java pool, and streams pool), ensuring the most effective memory utilization. Note, the log buffer pool, other buffer caches (such as KEEP and RECYCLE), other block sizes, fixed SGA, and other internal allocations must be manually sized and are not affected by ASMM. The memory allocated to these pools is deducted from the total available memory for SGA\_TARGET when ASMM is enabled.

The manageability monitor process (MMON) computes the values of the automatically tuned memory pools to support ASMM.

In addition to SGA\_TARGET and SGA\_MAX\_SIZE, you can set minimum nonzero values for each memory pool if an application component needs a minimum amount of memory to function properly. ASMM will treat those values as minimum levels.

The range of values for SGA\_TARGET can be from 64 MB to an operating system-dependent value.

View the UNDO\_TABLESPACE parameter. This parameter specifies the undo tablespace to be used when an instance starts. Oracle Database creates and manages information that is used to roll back, or undo, changes to the database. Such information consists of records of the actions of transactions, primarily before they are committed. These records are collectively referred to as undo and are stored in the undo tablespace. The results below indicate that the undo tablespace in your environment is UNDOTBS1.

View the COMPATIBLE parameter. This parameter specifies the release with which Oracle must maintain compatibility. It enables you to use a new release of Oracle, while at the same time guaranteeing backward compatibility with an earlier release. This is helpful if it becomes necessary to revert to the earlier release. By default, the value for the compatible entry for this parameter is equal to the version of the Oracle Database that you have installed.

View the CONTROL\_FILES initialization parameter. This parameter specifies one or more control files, separated by commas, and including paths. One to eight file names are listed. Oracle strongly recommends that you multiplex and mirror control files. The output has been formatted for legibility.

View the PROCESSES, SESSIONS, and TRANSACTIONS initialization parameters.

View the PROCESSES parameter. This parameter specifies the maximum number of operating system user processes that can simultaneously connect to an Oracle server. This value should allow for all background processes and user processes. The default values of the SESSIONS and TRANSACTIONS initialization parameters are derived from the PROCESSES parameter. Therefore, if you change the value of PROCESSES, you should evaluate whether to adjust the values of those derived parameters. The range of values is from six to an OS-dependent value. The default value is dynamic and dependent on the number of CPUs.

View the SESSIONS parameter. This parameter specifies the maximum number of sessions that can be created in the system. Because every login requires a session,

this parameter effectively determines the maximum number of concurrent users in the system. Notice in the results that the session entry has a value of 472. You should always set this parameter explicitly to a value equivalent to your estimate of the maximum number of concurrent users, plus the number of background processes, plus approximately 10% for recursive sessions.

View Advanced Parameters

In this section, you use the SHOW PARAMETER command to view advanced parameters.

View the TRANSACTIONS parameter. This is an advanced parameter and seldom needs any adjustment. This parameter specifies how many rollback segments to bring online when the UNDO\_MANAGEMENT initialization parameter is equal to MANUAL. A transaction is assigned to a rollback segment when the transaction starts, and it can't change for the life of the transaction. A transaction table exists in the rollback segment header with limited space, limiting how many transactions a single segment can support. Therefore, X number of concurrent transactions require at least Y number of rollback segments. With Oracle Automatic Undo Management, the database creates rollback segments, brings them online, takes them offline, and drops them as needed.

View the configuration for the DB\_FILES initialization parameter. This parameter specifies the maximum number of database files that can be opened for this database. The range of values is OS-dependent.

View the COMMIT\_LOGGING parameter. This parameter is used to control how redo is batched by the Log Writer process. There is no default value, as shown below. You can modify this parameter in a PDB.

View the COMMIT\_WAIT parameter. This parameter is used to control when the redo for a commit is flushed to the redo logs. There is no default value.

View the SHARED\_POOL\_SIZE parameter. This parameter specifies the size of the shared pool in bytes. The shared pool contains objects such as shared cursors, stored procedures, control structures, and parallel execution message buffers. The range of values is OS-dependent. The default value is zero if the SGA\_TARGET parameter is set. Otherwise, the value is 128 MB for a 64-bit platform or 48 MB for a 32-bit platform.

View the DB\_BLOCK\_SIZE parameter. This parameter specifies the standard Oracle database block size (in bytes) and is used by all tablespaces by default. Its value is set during database creation and cannot be subsequently changed. The range of values is from 2048 to 32768 (OS-dependent). The default value is 8192.

View the DB\_CACHE\_SIZE initialization parameter. You configure this parameter to specify the size of the standard block buffer cache (default buffer pool). The range of values is at least 4 MB times the number of CPUs. Smaller values are automatically rounded up to this value. The default value is zero if the SGA\_TARGET initialization parameter is set, otherwise the larger of 48 MB or (4 MB\*CPU\_COUNT).

View the UNDO\_MANAGEMENT parameter. This parameter specifies the undo space management mode that the system should use. When set to AUTO, the instance is started in automatic undo management mode. Otherwise, it is started in rollback undo mode. In rollback undo mode, undo space is allocated as rollback segments. In automatic undo mode, undo space is allocated as undo tablespaces. The value is AUTO or MANUAL. If the UNDO\_MANAGEMENT parameter is omitted when the instance is started, the default value AUTO is used.

View the MEMORY\_TARGET and MEMORY\_MAX\_TARGET parameters. MEMORY\_TARGET specifies the Oracle system-wide usable memory. The database server tunes memory to the MEMORY\_TARGET value, reducing or enlarging the SGA and PGA as needed. MEMORY\_MAX\_TARGET sets a maximum value for MEMORY\_TARGET.

In a PFILE, if you omit MEMORY\_MAX\_TARGET and include a value for MEMORY\_TARGET, the database automatically sets MEMORY\_MAX\_TARGET to the value of MEMORY\_TARGET. If you omit the line for MEMORY\_TARGET and include a value for MEMORY\_MAX\_TARGET, the MEMORY\_TARGET parameter defaults to zero. After startup, you can dynamically change MEMORY\_TARGET to a nonzero value if it does not exceed the value of MEMORY\_MAX\_TARGET. For MEMORY\_TARGET, values range from 152 MB to MEMORY\_MAX\_TARGET.

View the MEMORY\_TARGET parameter.

View the MEMORY\_MAX\_TARGET parameter.

View the PGA\_AGGREGATE\_TARGET parameter. This parameter specifies the amount of Program Global Area (PGA) memory available to all server processes attached to the database instance. This memory does not reside in the System Global Area (SGA). The database uses this parameter as a target amount of PGA memory to use. When setting this parameter, subtract the SGA from the total memory on the system that is available to the Oracle instance. The minimum value is 10 MB, and the maximum value is 4096 GB minus. The default value is 10 MB or 20% of the size of the SGA, whichever is greater.

Query Views for Parameter Values

In this section, you query views to learn about parameters.

Query the data dictionary to find views that contain the word "parameter." The query below returns 66 rows. Not all of these views contain information about initialization parameters. Among these rows are the V$PARAMETER, V$SPPARAMETER, V$PARAMETER2, and V$SYSTEM\_PARAMETER views, which you'll examine next.

Explore the V$PARAMETER view. This view displays the current parameter values in the current session.

View the columns in the V$PARAMETER view by using the DESCRIBE command. This command returns column names, whether null values are allowed (NOT NULL is displayed if the value cannot be null), and column data types.

The results below contain a column named ISSYS\_MODIFIABLE. This column is important because it tells you whether a parameter is static or dynamic. If its value is FALSE, then the parameter is static; otherwise it's dynamic. To change a static parameter, you must shut down and restart the database; however, you can modify a dynamic parameter in real time while the database is online.

Query NAME, ISSYS\_MODIFIABLE, and VALUE in the V$PARAMETER view. The query returns many rows.

The TRANSACTIONS parameter is static as indicated by FALSE in the ISSYS\_MODIFIABLE column. The PLSQL\_WARNINGS parameter is dynamic as indicated by IMMEDIATE in the ISSYS\_MODIFIABLE column.

Optional: Before entering the following command, you can enter SET PAUSE ON to cause a pause after each page output. Press Enter to display each next page. After all pages have been displayed, you can issue the SET PAUSE OFF command to stop this feature.

Query the V$PARAMETER view again, but this time be more specific. Include a WHERE clause to specify all parameters that contain the word "pool." The query returns all of the parameters that contain the string "pool."

**Note**: The values shown may vary from the values displayed in the output.

Explore the V$SPPARAMETER view. This view contains information about the contents of the server parameter file. If a server parameter file was not used to start the instance, each row of the view will contain FALSE in the ISSPECIFIED column.

View the columns in the V$SPPARAMETER view by using the DESCRIBE command.

Query NAME and VALUE in the V$SPPARAMETER view. Browse the rows returned by the query.

Explore the V$PARAMETER2 view. This view contains information about the initialization parameters that are currently in effect for the session. For parameters with more than one value assigned such as the control\_files parameter, each parameter value will be listed as a row in the view's output. A new session inherits parameter values from the instance-wide values displayed in the V$SYSTEM\_PARAMETER2 view.

View the columns in the V$PARAMETER2 view by using the DESCRIBE command.

Query NAME and VALUE in the V$PARAMETER2 view. Browse the rows returned by the query.

Explore the V$SYSTEM\_PARAMETER view. This view contains information about the initialization parameters that are currently in effect for the instance.

View the columns in the V$SYSTEM\_PARAMETER view by using the DESCRIBE

command.

Query NAME and VALUE in the V$SYSTEM\_PARAMETER view. Browse the rows returned by the query.

Exit SQL\*Plus.

Practice 6-3: Modifying Initialization Parameters by Using SQL\*Plus

Overview

In this practice, you modify the following kinds of initialization parameters (parameters) with SQL\*Plus:

Session-level parameter

Dynamic system-level parameter

Static system-level parameter

Assumptions

You are connected to the compute node as the oracle user. The Oracle environment is set to access the orclcdb instance.

Tasks

Modify a Session-Level Parameter

In this section, you modify the NLS\_DATE\_FORMAT parameter. This parameter defines the default date format to use with the TO\_CHAR and TO\_DATE functions. The NLS\_TERRITORY parameter determines the default value of NLS\_DATE\_FORMAT. NLS\_DATE\_FORMAT is one of the National Language Support (NLS) parameters that you can customize just for your session, therefore making it a session-level parameter. When your session ends, your modification expires, and the parameter is returned to its default value.

Start SQL\*Plus and log in to the database as the SYS user with the SYSDBA privilege.

Learn about the NLS\_DATE\_FORMAT parameter by querying the V$PARAMETER view. Include a WHERE clause to narrow down the query to just the NLS\_DATE\_FORMAT parameter. Remember that in the V$PARAMETER view, the parameter names are in lowercase.

Find out the default date format for the database by querying the NLS\_TERRITORY parameter in the V$PARAMETER view. Include a WHERE clause to narrow down the query to just the NLS\_TERRITORY parameter. Remember that in the V$PARAMETER view, the parameter names are in lowercase.

Note: NLS\_TERRITORY is set at installation and can be changed with ALTER SESSION.

Connect to ORCLPDB1. Run a simple query against the sample data to view an example of the current default date format in use.

Switch to ORCLPDB1 by using the ALTER SESSION command.

Query the LAST\_NAME and HIRE\_DATE columns in the HR.EMPLOYEES table. Notice the date format is dd-mon-rr.

Modify the NLS\_DATE\_FORMAT parameter to use the format mon dd yyyy by using the

ALTER SESSION command.

Rerun the query against the HR.EMPLOYEES table. Notice that the date format has changed from dd-mon-rr to mon dd yyyy , also that case of the output.

Query the NLS\_DATE\_FORMAT parameter again by using the SHOW PARAMETER command. The value column now reflects the custom date format.

Disconnect from ORCLPDB1 to end your session.

Connect to ORCLPDB1 again as the SYSTEM user by using the Easy Connect syntax. See *Course Practice Environment: Security Credentials* for the SYSTEM user password. The easy connect syntax is //<full hostname>:<port number>/<service name>.

Get the full hostname.

Get the listener port number. The grep command reduces the lines to just those using TCP or TCPS protocol. Look for the port using TCP.

Connect as a user with sysdba privileges.

Query the V$SERVICES view. Discover the network name.

**Note:** The network name is the name of the fully qualified database instance and is formed by default from the instance\_name and db\_domain\_name. In this practice environment, the network name is the same as the database service name.

Using values from steps 9a through 9d, construct and use the Easy Connect string to connect to the orclpdb1 PDB. Replace <service\_name> in the connect command with the name value from the previous step. , change ***password*** to the password shown in the *Course Practice Environment: Security Credentials*.

**Note:** The '//' characters preceding the <full hostname> are optional.

Rerun the query against the HR.EMPLOYEES table. The date format has reverted to the default format dd-mon-rr. A session-level parameter change only lasts for the duration of the session. A connect command creates a new session.

Query the NLS\_DATE\_FORMAT parameter again by using the SHOW PARAMETER command. The VALUE column no longer has the custom date format.

Modify a Dynamic System-Level Parameter

In this section, you modify the JOB\_QUEUE\_PROCESSES parameter. This parameter specifies the maximum number of job slaves per database instance that can be created for the execution of DBMS\_JOB jobs and Oracle Scheduler (DBMS\_SCHEDULER) jobs.

Exit SQL\*Plus, and connect to the root container with the SYSDBA privilege. If you try to update the JOB\_QUEUE\_PROCESSES parameter from PDB1, you'll get an error. Also, you'll need the SYSDBA privilege to restart the database instance later on.

Learn about the JOB\_QUEUE\_PROCESSES parameter by querying the V$PARAMETER view. Include a WHERE clause to narrow down the query to just the JOB\_QUEUE\_PROCESSES parameter. Remember that in the V$PARAMETER view, the parameter names are in lowercase.

Change the JOB\_QUEUE\_PROCESSES parameter value to 15 by using the ALTER SYSTEM command. Set SCOPE equal to BOTH so that the change happens in both the database instance memory (which makes the change immediate) and in the SPFILE (which makes the change permanent).

Use the SHOW PARAMETER command to verify that the JOB\_QUEUE\_PROCESSES parameter value is now equal to 15. Notice that only job was entered with the SHOW PARAMETER command instead of the full name, job\_queue\_processes. Remember, when you use the SHOW PARAMETER command, you don't have to enter the full name. The database server will find all parameters that contain the letters job. In this example, the database server found three parameters that contain the letters job. The query result indicates that the job\_queue\_processes value in memory is now 15.

Verify that the new value for the JOB\_QUEUE\_PROCESSES parameter persists after the database instance is restarted.

Shut down the database instance with the IMMEDIATE mode.

Start the database instance by using the STARTUP command.

View the configuration for the JOB\_QUEUE\_PROCESSES parameter again by using the SHOW PARAMETER command. The value is 15, which proves that your change to the parameter persisted after the database instance was restarted.

Modify a Static System-Level Parameter

In this section, you modify the SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter. This parameter specifies the number of authentication attempts that can be made by a client on a connection to the server process. These login attempts can be for multiple user accounts in the same connection. After the specified number of failure attempts, the connection will be automatically dropped by the server process.

Learn about the SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter by querying the V$PARAMETER view. Include a WHERE clause to narrow down the query to just the SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter. Remember that in the V$PARAMETER view, the parameter names are in lowercase. The query results below have been formatted for easier viewing.

Change the SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter value to 2 by using the ALTER SYSTEM command. Include the comment 'Reduce for tighter security' and set the scope equal to SPFILE so that the change is made only in the SPFILE. When you specify SCOPE as SPFILE or as BOTH, an optional COMMENT clause lets you associate a text string with the parameter update. The comment is written to the SPFILE.

What happens if you set SCOPE=BOTH?

Now set SCOPE=SPFILE and include the comment.

View the SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter value by using the SHOW PARAMETER command. The query result indicates that the value hasn't been updated yet. It's still equal to 3 because you need to restart the database instance for the change to take effect, which is required for static parameters.

Restart the database and then verify that the new value for the

SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter is updated.

Shut down the database instance with the IMMEDIATE mode.

Start the database instance by using the STARTUP command.

View the SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter value again by using the SHOW PARAMETER command. The query result indicates that the parameter's value was successfully changed to 2.

View the NAME and UPDATE\_COMMENT columns in the V$PARAMETER view for the SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter. Notice that the comment you added is stored in this view. The results below are formatted for easier reading.

Change the SEC\_MAX\_FAILED\_LOGIN\_ATTEMPTS parameter value back to its original value.

Exit SQL\*Plus and close the terminal window.

Practice 6-4: Viewing Diagnostic Information

Overview

In this practice, you perform the following tasks:

Examine the structure of the Automatic Diagnostic Repository (ADR)

View the alert log two ways—first through a text editor and then using the Automatic Diagnostic Repository Command Interpreter (ADRCI)

Enable DDL logging and log some DDL statements in the DDL log file

The alert log is a file that provides a chronological log of database messages and errors. It is automatically created and stored, by default, in the Automatic Diagnostic Repository (ADR) on the database server in the $ORACLE\_BASE/diag/rdbms/<db\_name>/<SID>/trace directory.

ADRCI is an Oracle command-line utility that enables you to investigate problems, view health check reports, and package and upload first-failure data to Oracle Support. You can also use the utility to view the names of the trace files in the ADR and to view the alert log. ADRCI has a rich command set that you can use interactively or in scripts.

The DDL log file contains one log record for each DDL statement.

Assumptions

You are logged in as the oracle user.

The Oracle environment is set to access the orclcdb database instance.

Tasks

View the ADR Directories

The Automatic Diagnostics Repository (ADR) is a hierarchical file-based repository for handling diagnostic information. You can navigate the contents of ADR by using your operating system's command line, file browsing tools, or Oracle's ADR Command Interpreter (ADRCI). ADRCI is preferred for many tasks.

In this section, you locate the XML and text-only versions of the alert log by querying the

V$DIAG\_INFO view.

Start SQL\*Plus and log in to the database as the SYS user with the SYSDBA privilege.

View the locations of the various diagnostics directories in the ADR. The results below have been formatted for easier reading.

The path that corresponds to the Diag Alert entry in the NAME column is for the XML version. This path is /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/alert.

The path that corresponds to the Diag Trace entry is for the text-only version. This path is /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace.

Exit SQL\*Plus.

View the Alert Log

View the XML version of the alert log. The log.xml file is the XML version of the alert log.

Browse to the /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/alert

directory.

List the contents of the directory. Notice that there is a log.xml file in this directory.

Use cat or more to scroll through the file. Notice that it is a chronological log of messages about non-default initialization parameters used at startup, errors, SQL statements, and so on. Oracle Database uses the alert log to keep a record of these

events as an alternative to displaying the information on an operator’s console.

View the text-only version of the alert log.

Change to the /u01/app/oracle/diag/rdbms/orclcdb/orclcdb/trace

directory.

The alert\_orclcdb.log (format is alert\_SID.log) file is the text-only version. In this directory, you also have server process trace files (TRC files) and trace map files (TRM files). Each server and background process can write to an associated trace file. When a process detects an internal error, it dumps information about the error to its trace file. Trace map files contain structural information about trace files and are used for searching and navigation.

Open the file with an editor or use a command such as tail to view the contents of the alert log.

Change your directory to the home Directory

Use ADRCI to View the Alert Log

Start the ADRCI tool. Recall that you set the Oracle environment variables at the beginning of this practice; however, only the ORACLE\_HOME environment variable needs to be set prior to starting ADRCI. If you ever need to set just that one variable, you can do so by entering the following at the command prompt: export PATH=$PATH:$ORACLE\_HOME/bin.

View the alert log by using the SHOW ALERT command. The show alert command will prompt you for which alert log file to display, unless you are in the database’s diagnostic directory. Choose the alert log for the orclcdb database:

Note: that the alert log file in the vi editor, by default.

Enter **G** (uppercase) to move to the bottom of the alert file.

Enter **/Starting ORACLE/** and press return. Press **N** (uppercase) to search from the bottom of the file to find the last time the instance was started. The following will be similar to your alert log. Note: Here lowercase and uppercase are important because vi distinguishes them, unless you ignore them by setting :set ic.

Search forward by entering **/ ALTER** to find the line that starts with ALTER DATABASE MOUNT. Here lowercase and uppercase are important because vi distinguishes them.

Search forward again by entering **/ ALTER** to find the line that starts with ALTER DATABASE OPEN. Notice that the stages that the database goes through during startup are MOUNT and OPEN.

Exit the vi editor by entering **:q!** and pressing **Enter**.

At Alert log list, enter **Q** to leave alert log list.

Exit adrci.

Log DDL Statements in the DDL Log File

Determine if DDL logging is enabled in ORCLPDB1. If not, enable it by setting the value for the ENABLE\_DDL\_LOGGING initialization parameter to TRUE.

Start SQL\*Plus and log in to the database as the SYS user with the SYSDBA privilege.

Switch to PDB1.

Issue the SHOW PARAMETER command to view the value for ENABLE\_DDL\_LOGGING. In Oracle Database Cloud Service, ENABLE\_DDL\_LOGGING is set to TRUE by default. The default value for ENABLE\_DDL\_LOGGING is FALSE in non-Cloud installations.

If DDL logging was not enabled, enable it for just this session by using the ALTER SESSION command.

Create and drop a table to generate statements that will be logged.

Exit SQL\*Plus.

Change to the directory where the text version of the DDL log file resides.

List the contents of the log directory.

View the ddl\_orclcdb.log file by using the cat command.

Change to the ddl directory and list the contents. The XML version of the DDL log file (log.xml) is located here.

Close the terminal window.