Managing Database Instance
Initialization Parameters & Parameters Files

- When you start a database instance , it reads instance configuration parameters (initialization parameters) from initialization parameters file (Parameter file)

- Parameter files location on most platform is : $ORACLE_HOME/dbs
- You can use one of the following types of parameter files to start the instance

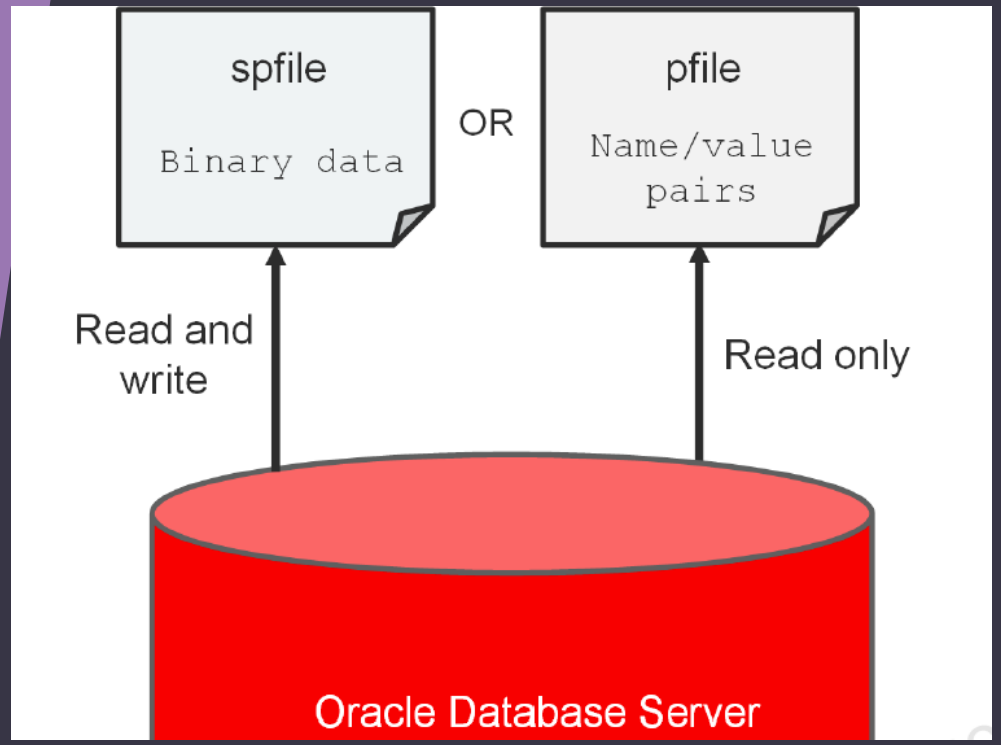
1. Server parameter file (Spfile)

- It is binary file (you can not edit it manually)
- Database can read it and write on it
- It is the preferred choice
- You can use alter system command for changing Parameters
- It is automatically created by DBCA when you create CDB.
- Default name for spfile is : spfile<sid>.ora *example: spfileorcl.ora*

2. Text initialization parameter file (pfile)

- it is a text file containing parameter values in name/value pairs
- The database can only read it
- You should edit it manually and restart the instance to refresh the parameter values
- Sample pfile name : *init.ora*

Managing Database Instance



Search order for a parameter file

SQL> Startup;

1. Spfile<sid>.ora example: spfileorcl.ora
2. Spfile.ora
3. Init<sid>.ora (pfile) example initorcl.ora

Note: you can create pfile from spfile but you should name it Init<sid>.ora in order the instance can use use it automatically incase the spfile not available.

If you name it another name, then you have to specify it during startup

SQL> Startup pfile=pfilename

Managing Database Instance

Uses of Initialization Parameters.

- Optimize performance by adjusting memory structures.
- Set database-wide defaults, such as the amount of space initially allocated for a context area when it is created (context area :area for processing an SQL statement).
- Set database limits, such as the maximum number of database users.
- Specify names of files or directories required by the database.

Note: All initialization parameters are optional. Oracle has a default value for each parameter

Managing Database Instance

Types of Initialization Parameters.

❑ Basic Parameters

- Most databases should only need to have the database basic initialization parameters set to run properly and efficiently.
- Basic parameters around 30 parameters
- *Examples*: DB_BLOCK_SIZE , SGA_TARGET, CONTROL_FILES

❑ Advanced Parameters

- Sometimes we need to change theses parameters to enhance performance.
- Some of these parameters need expert DBAs
- Example: DB_CACHE_SIZE

Derived parameters: their values are calculated from the values of other parameters.
Example: sessions Derived from processes example: sessions = (1.5 * PROCESSES) + 22

Operating System-Dependent Parameters
Example : DB_BLOCK_SIZE

Note: the best way to know more is : Database Reference manual from oracle (12c,18c…)

Managing Database Instance

About Modifying Initialization Parameters.

- Modify parameters to set capacity limits or improve performance.
  - Use EM Express or SQL*Plus (`ALTER SESSION` or `ALTER SYSTEM`).
- Query the `V$PARAMETER` view for a particular initialization parameter to learn whether you can make:
  - Session-level changes (`ISSES_MODIFIABLE` column)
  - System-level changes (`ISSYS_MODIFIABLE` column)
  - PDB-level changes (`ISPDB _MODIFIABLE` column)
- Use the `SCOPE` clause with the `ALTER SYSTEM` command to tell the system *where* to update the system-level parameter:
  - `MEMORY` indicates that the change is made in memory, takes effect immediately, and persists until the database is shut down.
  - `SPFILE` indicates that the change is made in the server parameter file. The new setting takes effect when the database is next shut down and started up again
  - `BOTH` indicates that the change is made in memory and in the server parameter file (**this is the default**)
  - `DEFERRED` Effective for future sessions

Default scope in alter statement

Alter system set *parameter=value*

- If a server parameter file was used to start up the database, then BOTH is the default.

Alter system set parameter=value same as Alter system set parameter=value scope=both

- If a parameter file was used to start up the database, then MEMORY is the default, as well as the only scope you can specify.

Alter system set parameter=value same as Alter system set parameter=value scope=memory

CONTAINER clause in alter system

Alter system set *parameter=value container =CURRENT | ALL*

- You can specify the CONTAINER clause when you set a parameter value in a CDB.
- A CDB uses an inheritance model for initialization parameters in which PDBs inherit initialization parameter values from the root
- A PDB can override the root's setting for some parameters (ISPDB_MODIFIABLE is TRUE)
- Note: when the PDB override the root's setting , a new record will be added to V$system_PARAMETER
- If you specify CONTAINER = ALL, then the parameter setting applies to all containers in the CDB, including the root and all of the PDBs. The current container must be the root.
- If you specify CONTAINER = CURRENT, then the parameter setting applies only to the current container.

But When the current container is the root, the parameter setting applies to the root and to any PDB with an inheritance property of true for the parameter (no override done before by the pluggable)

- if you omit this clause, then CONTAINER = CURRENT is the default.

Automatic Diagnostic Repository (ADR)

The Automatic diagnostic repository (ADR):

- Is a file-based repository outside the database
- Is a system-wide central tracing and logging repository
- Stores database diagnostic data such as:
  - Traces
  - Alert log
  - Health monitor reports

Typical installations will have the ADR_BASE set to the ORACLE_BASE

```
[oracle@test ~]$ echo $ORACLE_BASE
/u01/app/oracle
```

- You can make sure from parameter diagnostic_dest
- The ADR home path is `<ADR Base>/diag/product_type/db_id/instance_id`

Automatic Diagnostic Repository (ADR)
The alert log

The **alert log** is a chronological log of messages and errors, and includes the following items:

- Any non default initialization parameter used at startup.
- All internal errors (ORA-00600), block corruption errors (ORA-01578), and deadlock errors (ORA-00060) that occur
- Administrative operations, such as some CREATE, ALTER, and DROP statements and STARTUP, SHUTDOWN, and ARCHIVELOG statements
- Messages and errors relating to the functions of shared server and dispatcher processes.
- Errors occurring during the automatic refresh of a materialized view
- The locations of the various diagnostics directories can be displayed using the V$DIAG_INFO view.

```
SELECT name, value FROM v$diag_info;
```

- You can view the alert log by text editor or using ADRCI

Automatic Diagnostic Repository (ADR)
Trace files

- Trace files contain:
  - Error information (contact Oracle Support Services if internal error occurs)
  - Information that can provide guidance for tuning applications or an instance
- Each server and background process can write to an associated trace file.
- Trace file names for background processes are named after their processes.
  - Exception: Trace files generated by job queue processes
- Oracle Database includes an advanced fault diagnosability infrastructure for preventing, detecting, diagnosing, and resolving problems.
- When a critical error occurs:
  - An incident number is assigned to the error.
  - Diagnostic data for the error (such as trace files) is immediately captured and tagged with the incident number.
  - Data is stored in the ADR
- ADR files can be automatically purged with retention policy parameters.

background process name

orcl_dbw0_27147.trc

Server process name

orcl_ora_3341.trc

Automatic Diagnostic Repository (ADR)
Trace files

Purging Mechanism

The purging mechanism allows you to specify a retention policy stating:

- How old ADR contents should be before they are automatically deleted.
  - The long retention period is used for the relatively higher-value diagnostic data, such as incidents and alert log. (Default value is 365 days)
  - The short retention period is used for traces and core dumps. (Default value is 30 days).

  Older items are deleted first. The long retention period items are typically older than any of the items in the short retention period. So a mechanism is used in which the time periods are "scaled," so that roughly the same percentage of each gets deleted. Some components use these periods in slightly different ways. For instance, IPS, the packaging facility, uses the short retention period to determine when to purge packaging metadata and the staging directory contents. However, the age of the data is based on when the package was completed, not when it was originally created.

- The size-based retention to specify a target size for an ADR home. When purging, the old data, determined by the time-based retention periods, is deleted first. If the size of the ADR home is still greater than the target size, diagnostics are automatically deleted until the target size is no longer exceeded.