# Oracle Database

# Managing Database Instances

An Oracle database system consists of an Oracle database and an Oracle instance. An Oracle instance (also known as a database instance) contains the set of Oracle Database background processes that operate on the stored data and the shared allocated memory that those processes use to do their work. In this tutorial you learn how to manage your Oracle Database instance.
In this document you will learn the following topics:



## What is Initialization Parameter Files?

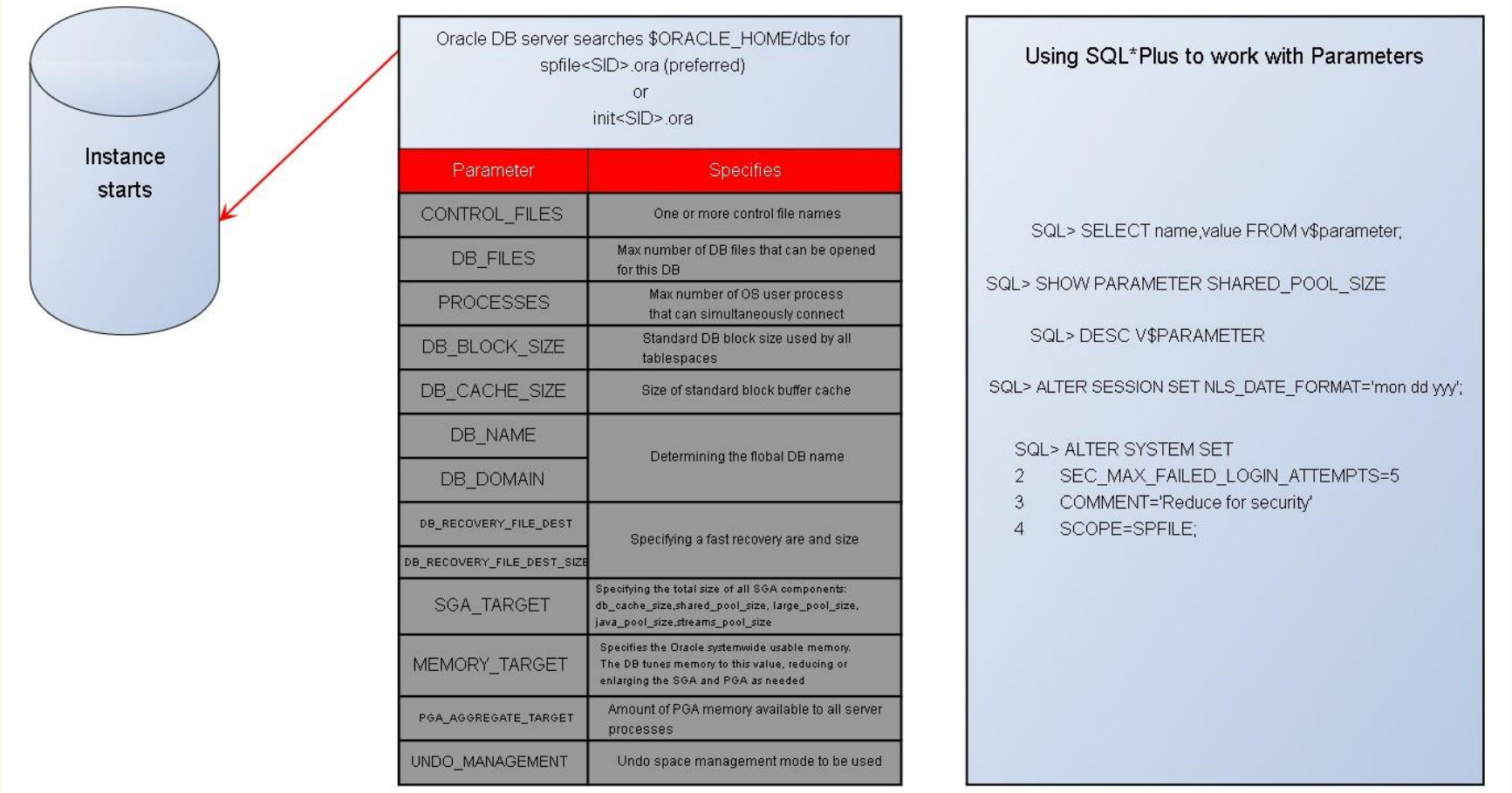
The properties of an Oracle instance are specified using instance initialization parameters. When the instance is started, an initialization parameter file is read, and the instance is configured accordingly. Parameter files used to start your database instance can be of 2 different types: **Server parameter file** (SPFILE) - a binary file that is written to and read by the database server. You can't edit it manually. or **Text initialization parameter** file (PFILE) - a text file containing parameter values in name/value pairs, which the database server can read to start the database instance. Unlike an SPFILE, the database server cannot write to and alter a PFILE. Therefore, to change parameter values in a PFILE you must manually edit the PFILE in a text editor and restart the database instance to refresh the parameter values.
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:

```
SQL> show parameter spfile
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
spfile                               string      /u01/app/oracle/product/19.0.0/db_1/dbs/spfileorcl.ora
```

Check ORACLE_HOME/dbs directory. There should be present SPFILE and PFILE files. You can view the text PFILE with less editor:

```
[oracle@oracle dbs]$ ls /u01/app/oracle/product/19.0.0/db_1/dbs/*.ora
/u01/app/oracle/product/19.0.0/db_1/dbs/init.ora
/u01/app/oracle/product/19.0.0/db_1/dbs/spfileorcl.ora
[oracle@oracle dbs]$ less /u01/app/oracle/product/19.0.0/db_1/dbs/init.ora
```

An example of some parameters you can modify in Database Instance:



The list of all initialization parameters with their description can be found in the following link: Initialization Parameter Descriptions
Some parameters are derived, meaning their values are calculated from the values of other parameters. For example if we look up **PROCESSES** parameter in oficial documentation (1.282 PROCESSES ) we will see the definition for this parameter: **PROCESSES** specifies the maximum number of operating system user processes that can simultaneously connect to Oracle and additional information about derived parameters - The default values of the **SESSIONS** and **TRANSACTIONS** parameters are derived from the PROCESSES parameter.

## How to modify Initialization Parameters?

Before any parameter changes it is a good practice to do a backup of parameter configuration that we can use to start a database instance in case it can not be started after our changes or in case of some problems with **SPFILE**. You can create a text initialization parameter file from the SPFILE by using the **CREATE PFILE** command:

```
[oracle@oracle ~]$ . oraenv
ORACLE_SID = [oracle] ? orcl
The Oracle base has been set to /u01/app/oracle
[oracle@oracle ~]$ sqlplus / as sysdba
SQL*Plus: Release 19.0.0.0.0 - Production on Wed Jun 30 17:54:13 2021
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> create pfile='/home/oracle/pfile_20210630.conf' from spfile;
File created.
SQL> Disconnected
[oracle@oracle ~]$ less /home/oracle/pfile_20210630.conf
```

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. At first **SHUTDOWN** the database instance if it is running:

```
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
```

Rename the spfileORCL.ora file to original_spfileorcl.ora 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 look for the initORCL.ora file (PFILE) to start the database instance.

```
[oracle@oracle dbs]$ cd $ORACLE_HOME/dbs
[oracle@oracle dbs]$ pwd
/u01/app/oracle/product/19.0.0/db_1/dbs
[oracle@oracle dbs]$ mv spfileorcl.ora original_spfileorcl.ora
```

Try to start the Database Instance with the **STARTUP** command. You will get an error because Database Instance haven't found any SPFILE ( spfileORCL.ora ) nor PFILE ( initORCL.ora ) at $ORACLE_HOME/dbs. File name conventions are important for Database Instance

```
[oracle@oracle dbs]$ sqlplus / as sysdba
SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jul 2 10:12:16 2021
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> startup;
ORA-01078: failure in processing system parameters
LRM-00109: could not open parameter file '/u01/app/oracle/product/19.0.0/db_1/dbs/initorcl.ora'
SQL>
```

Copy the backup configuration file we created earlier to the place where oracle database instance is looking for it in the error above **$ORACLE_HOME/dbs** and with proper name - initorcl.ora

```
[oracle@oracle dbs]$ cp /home/oracle/pfile_20210630.conf $ORACLE_HOME/dbs/initorcl.ora
[oracle@oracle dbs]$ ls -lha /u01/app/oracle/product/19.0.0/db_1/dbs/initorcl.ora
-rw-r--r--. 1 oracle oinstall 1.3K Jul 2 10:34 /u01/app/oracle/product/19.0.0/db_1/dbs/initorcl.ora
[oracle@oracle dbs]$
```

---

Now if you try to start DB Instance again it should be successfully started from PFILE:

```
[oracle@oracle dbs]$ sqlplus / as sysdba
SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jul 2 10:43:51 2021
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> startup
ORACLE instance started.
Total System Global Area 327151856 bytes
Fixed Size                 9134320 bytes
Variable Size            289406976 bytes
Database Buffers          25165824 bytes
Redo Buffers               3444736 bytes
Database mounted.
Database opened.
SQL>
```

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.

```
SQL> SHOW PARAMETER spfile
```

```
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
spfile                               string
SQL>
```

Let's now create SPFILE from PFILE with **CREATE SPFILE** command:

```
[oracle@oracle dbs]$ sqlplus / as sysdba
SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jul 2 11:03:10 2021
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle. All rights reserved.
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
SQL> CREATE SPFILE='/u01/app/oracle/product/19.0.0/db_1/dbs/spfileorcl.ora' FROM
PFILE='/u01/app/oracle/product/19.0.0/db_1/dbs/initorcl.ora';
File created.
SQL> exit
# lets check all parameter files we have after our manipulations:
[oracle@oracle dbs]$ ls /u01/app/oracle/product/19.0.0/db_1/dbs/*ora
/u01/app/oracle/product/19.0.0/db_1/dbs/init.ora  # example
pfile         /u01/app/oracle/product/19.0.0/db_1/dbs/original_spfileorcl.ora # original spfile
we renamed
/u01/app/oracle/product/19.0.0/db_1/dbs/initorcl.ora # backup
pfile /u01/app/oracle/product/19.0.0/db_1/dbs/spfileorcl.ora # restored  just now spfile
```

Now let's **SHUTDOWN** and **STARTUP** our Oracle DB Instance to make sure it can be started from newly created **SPFILE**:

```
[oracle@oracle ~]$ sqlplus / as sysdba
SQL> SHUTDOWN IMMEDIATE
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL>
SQL> startup
ORACLE instance started.
Total System Global Area 327151856 bytes
Fixed Size                 9134320 bytes
Variable Size            289406976 bytes
Database Buffers          25165824 bytes
Redo Buffers               3444736 bytes
Database mounted.
Database opened.
SQL> show parameter spfile
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
spfile                               string      /u01/app/oracle/product/19.0.0/db_1/dbs/spfileorcl.ora
SQL>
```

Good! You have learned how you can create PFILE from SPFILE and SPFILE from PFILE to restore the work of database instance. Also you would like to perform some regular backups of SPFILE to have the ability to restore from backups in case you need it. Now we can move on to changing db parameters.

Before modifying a parameter, you also should query the **V$PARAMETER** view to learn about how you can modify a parameter.

```
[oracle@oracle ~]$ sqlplus / as sysdba
SQL> set markup csv on;
SQL> SELECT name, value, isses_modifiable, issys_modifiable, ispdb_modifiable from V$PARAMETER where name in
('nls_date_format','sec_max_failed_login_attempts');
"NAME","VALUE","ISSES_MODIFIABLE","ISSYS_MODIFIABLE","ISPDB_MODIFIABLE"
"nls_date_format",,"TRUE","FALSE","TRUE"
"sec_max_failed_login_attempts","5","FALSE","FALSE","FALSE"
SQL>
```

## Modify a Session-Level Parameter

Pay attention to the values of the **ISSES_MODIFIABLE** column. A value of **TRUE** means you can change the parameter for your current session with the **ALTER SESSION** command. Changes are applied to your current session immediately (dynamically) and expire when you end your session and there is no impact on other client sessions or database instance in general. In this session, you modify the **NLS_DATE_FORMAT** parameter that defines the default date format to use with the **TO_CHAR** and **TO_DATE** functions. When your session ends, your modification expires, and the parameter is returned to its default value.

```
# Start PDB1 if it is down like in my case:
SQL> show pdbs;
    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           MOUNTED
SQL> alter pluggable database PDB1 open;
Pluggable database altered.
SQL> alter session set container=PDB1;
Session altered.
# Run a simple query against the sample data to view an example of the current default date format in use.
SQL> SELECT last_name, hire_date FROM hr.employees;
LAST_NAME                 HIRE_DATE
```

```
--------------------------------
King                 17-JUN-03
Kochhar               21-SEP-05
De Haan               13-JAN-01
Hunold               03-JAN-06
Ernst                21-MAY-07
Austin               25-JUN-05
Pataballa             05-FEB-06
Lorentz              07-FEB-07
Greenberg              17-AUG-02
Faviet               16-AUG-02
Chen                 28-SEP-05
...
# Modify the NLS_DATE_FORMAT parameter
SQL> ALTER SESSION SET NLS_DATE_FORMAT ='mon dd yyyy';
Session altered.
# Rerun the query against the HR.EMPLOYEES table. Notice that the date format has changed from dd-mon-rr to mon dd yyyy.
SQL> SELECT last_name, hire_date FROM hr.employees;
LAST_NAME              HIRE_DATE
---------------------------------
King                 jun 17 2003
Kochhar                sep 21 2005
De Haan                jan 13 2001
Hunold               jan 03 2006
Ernst                may 21 2007
Austin               jun 25 2005
Pataballa             feb 05 2006
Lorentz              feb 07 2007
Greenberg              aug 17 2002
Faviet               aug 16 2002
Chen                 sep 28 2005
```

## Modify a Dynamic System-Level Parameter

The **ISSYS_MODIFIABLE** column value in **V$PARAMETER** view tells you when a system-level change to the parameter, made by using the ALTER SYSTEM command, takes effect:

1. **IMMEDIATE** means the change will take effect immediately and be applied to all current sessions.
2. **DEFERRED** means the change will take effect only for newly created sessions.
3. **FALSE** Parameters with a value of FALSE are referred to as static parameters. For static parameters, you need to restart the database instance for changes to be applied.

In this example, you modify the **JOB_QUEUE_PROCESSES** parameter that 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.

```
[oracle@oracle ~]$ sqlplus / as sysdba
SQL*Plus: Release 19.0.0.0.0 - Production on Sat Jul 3 11:53:27 2021
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle.  All rights reserved.
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
# let's check if we can change parameter immediately without instance restart (ISSYS_MODIFIABLE column):
SQL> set markup csv on;
SQL> SELECT name, isses_modifiable, issys_modifiable,value FROM v$parameter WHERE name = 'job_queue_processes';
"NAME","ISSES_MODIFIABLE","ISSYS_MODIFIABLE","VALUE"
"job_queue_processes","FALSE","IMMEDIATE","20"
# ISSYS_MODIFIABLE=IMMEDIATE means the change can be applied immediately
SQL> ALTER SYSTEM SET job_queue_processes=15 SCOPE=BOTH;
# The new value for the parameter has been applied:
SQL> show parameter job_queue_processes;
"NAME","TYPE","VALUE"
"job_queue_processes","integer","15"
SQL>
```

Pay attention to the **SCOPE** part of the command above. The **SCOPE** clause lets you specify when the change takes effect:

1. **MEMORY**: This value tells the system to make the parameter change in memory only. The change will take effect immediately, and will be lost after you restart the database instance. If you are using pfile (not recommended), then this is the only scope you can specify.
2. **SPFILE**: This value tells the system to make the parameter change in the server parameter file SPFILE only. The new setting takes effect after the database restart.
3. **BOTH**: This value tells the system to make the parameter change in both memory and in the SPFILE. The change will take effect immediately and persist after you restart the database instance.

## Modify a Static System-Level Parameter

In this example, you modify the SEC_MAX_FAILED_LOGIN_ATTEMPTS that 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.

```
SQL> ALTER SYSTEM SET SEC_MAX_FAILED_LOGIN_ATTEMPTS=15 SCOPE=SPFILE;
System altered.
# Pay attention that the value for the parameter has not been changed.. You need to restart the Database instance for the new
value to be applied.
```

```
SQL> SHOW PARAMETER sec_max
"NAME","TYPE","VALUE"
"sec_max_failed_login_attempts","integer","5"
```

Pay attention we used **SCOPE=SPFILE** for **SEC_MAX_FAILED_LOGIN_ATTEMPTS** as the only option to change this parameter. If we try to use another **SCOPE** for parameter with **ISSYS_MODIFIABLE=FALSE** we get an error:

```
SQL> set markup csv on;
SQL> SELECT name, value, isses_modifiable, issys_modifiable, ispdb_modifiable from V$PARAMETER where name in ('sec_max_failed_login_attempts');
"NAME","VALUE","ISSES_MODIFIABLE","ISSYS_MODIFIABLE","ISPDB_MODIFIABLE"
"sec_max_failed_login_attempts","5","FALSE","FALSE","FALSE"
SQL> ALTER SYSTEM SET SEC_MAX_FAILED_LOGIN_ATTEMPTS=15 SCOPE=BOTH;
ALTER SYSTEM SET SEC_MAX_FAILED_LOGIN_ATTEMPTS=5 SCOPE=BOTH
                *
ERROR at line 1:
ORA-02095: specified initialization parameter cannot be modified
```

When you issue the **ALTER SYSTEM** statement while connected to a PDB, you can modify only initialization parameters for which the **ISPDB_MODIFIABLE** column is **TRUE** in the **V$SYSTEM_PARAMETER** view. The initialization parameter value takes effect only for the PDB. For any initialization parameter that is not set explicitly for a PDB, the PDB inherits the CDB root's parameter value.

## How to view Initialization Parameters?

We can query the **V$PARAMETER** view or run **SHOW PARAMETER** command. In this example we filter results only for parameters that contain “pool” word-part in their names:

```
SQL> SELECT name, value FROM v$parameter WHERE name LIKE '%pool%';
"NAME","VALUE"
"shared_pool_size","0"
"large_pool_size","0"
"java_pool_size","0"
"streams_pool_size","0"
"shared_pool_reserved_size","14260633"
"memoptimize_pool_size","0"
"buffer_pool_keep",
"buffer_pool_recycle",
"olap_page_pool_size","0"
9 rows selected.
SQL> show parameter pool;
"NAME","TYPE","VALUE"
"buffer_pool_keep","string",
"buffer_pool_recycle","string",
"java_pool_size","big integer","0"
"large_pool_size","big integer","0"
"memoptimize_pool_size","big integer","0"
"olap_page_pool_size","big integer","0"
"shared_pool_reserved_size","big integer","14260633"
"shared_pool_size","big integer","0"
"streams_pool_size","big integer","0"
```

Other views that contain parameter information include:

- **V$SPPARAMETER**: Displays information about the contents of the server parameter file SPFILE
- **V$PARAMETER2** : Displays information about the parameters that are currently in effect for the session. A new session inherits parameter values from the database instance-wide values displayed in the **V$SYSTEM_PARAMETER2** view.
- **V$SYSTEM_PARAMETER**: Displays information about the parameters that are currently in effect for the database instance.

Let’s review some important initialization parameters. To view the values of the **DB_NAME** and **DB_DOMAIN** parameters that together, create the global database name

```
SQL> SHOW PARAMETER db_name
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_name                              string      orcldb
SQL> SHOW PARAMETER db_domain
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_domain                            string
SQL>
```

Let’s view the configuration for the **DB_FILES** initialization parameter that specifies the maximum number of database files that can be opened for this database - 200 files in our case.

```
SQL> SHOW PARAMETER db_files
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
db_files                             integer     200
SQL>
```

**DB_RECOVERY_FILE_DEST** and **DB_RECOVERY_FILE_DEST_SIZE** parameters set the location of the fast recovery area and its size. 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.

```
SQL> SHOW PARAMETER db_recovery_file_dest
NAME                                 TYPE        VALUE
```

```
db_recovery_file_dest                string      /u01/app/oracle/product/19.0.0/db_1/myreco
db_recovery_file_dest_size           big integer 12732M
```

**UNDO_TABLESPACE** 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**.

```
SQL> SHOW PARAMETER undo_tablespace
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
undo_tablespace                      string      UNDOTBS1
SQL>
```

**COMPATIBLE** 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. - for example when you upgrade a database to a new version compatible shows you the version you can roll back to. By default, the value for the compatible entry for this parameter is equal to the version of the Oracle Database that you have installed.

```
SQL> SHOW PARAMETER compatible
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
compatible                           string      19.0.0
noncdb_compatible                    boolean     FALSE
SQL>
```

CONTROL_FILES initialization parameter specifies one or more control files, separated by commas, and including paths. Oracle strongly recommends that you multiplex and mirror control files.

```
SQL> SHOW PARAMETER control_files
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
control_files                        string      /u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/control01.ctl,
                                                 /u01/app/oracle/product/19.0.0/db_1/myreco/ORCLDB/control02.ctl
```

**PROCESSES** 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 by the following formula: *sessions=1.5*processes+22; transactions=sessions*1.1*. Therefore, if you change the value of PROCESSES, you should evaluate whether to adjust the values of those derived parameters. The default value is dynamic and dependent on the number of CPUs.

```
SQL> SHOW PARAMETER processes
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
aq_tm_processes                      integer     1
db_writer_processes                  integer     1
gcs_server_processes                 integer     0
global_txn_processes                 integer     1
job_queue_processes                  integer     20
log_archive_max_processes            integer     4
processes                            integer     300

SQL>
```

**SESSIONS** 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.

```
SQL> SHOW PARAMETER sessions
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
java_max_sessionspace_size           integer     0
java_soft_sessionspace_limit         integer     0
license_max_sessions                 integer     0
license_sessions_warning             integer     0
sessions                             integer     472
shared_server_sessions               integer
```

**TRANSACTIONS** 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 X 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. The maximum number of concurrent transactions is now restricted by undo tablespace size (UNDO_MANAGEMENT = AUTO) or the number of online rollback segments (UNDO_MANAGEMENT = MANUAL).

```
SQL> SHOW PARAMETER transactions
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
transactions                         integer     519
transactions_per_rollback_segment    integer     5
SQL>
```

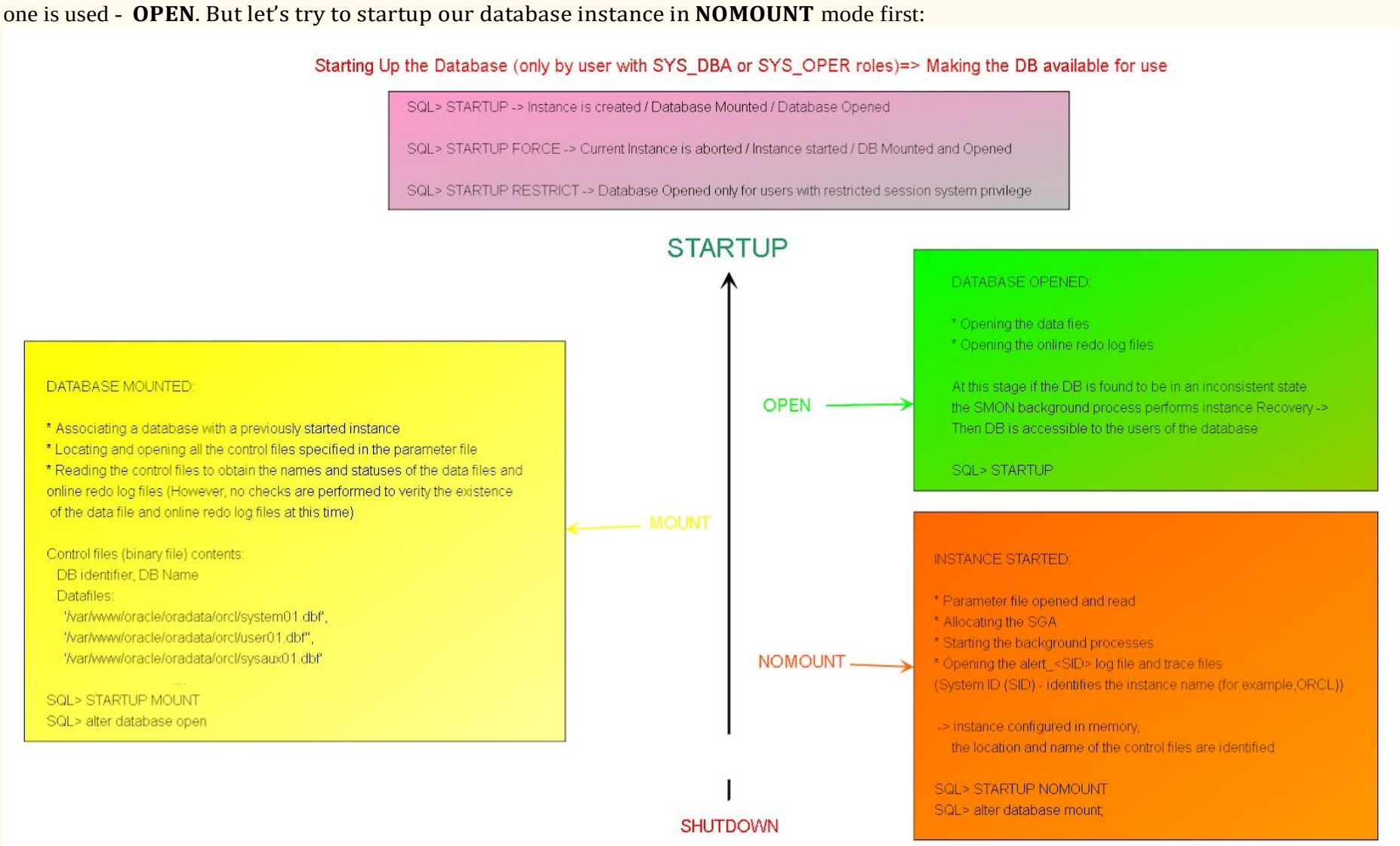
**UNDO_MANAGEMENT** parameter specifies the undo space management mode that the system should use. When set to AUTO, the instance is started in automatic undo management mode. Otherwise, it is started in rollback undo mode. In rollback undo mode, undo space is allocated as rollback segments. In automatic undo mode, undo space is allocated as undo tablespaces.

```
SQL> SHOW PARAMETER undo_management
NAME                                 TYPE        VALUE
------------------------------------ ----------- ------------------------------
undo_management                      string      AUTO
SQL>
```

## How to start Oracle Database Instance?

Before users can connect to a database instance, a database administrator must start the database instance. The database instance and database go through stages as the database is made available for access by users.

You can use the **STARTUP** command in SQL*Plus with the options shown in the slide for each stage. If we do not specify any option the default one is used - **OPEN**. But let's try to startup our database instance in **NOMOUNT** mode first:




```
[oracle@oracle ~]$ sqlplus / as sysdba
# If your database is running at the moment you can shut it down with SHUTDOWN command
SQL> show con_name
ERROR:
ORA-01034: ORACLE not available
Process ID: 0
Session ID: 0 Serial number: 0
SP2-1545: This feature requires Database availability.
SQL> startup nomount
ORACLE instance started.
"Total System Global Area",327151856,"bytes"
"Fixed Size",9134320,"bytes"
"Variable Size",289406976,"bytes"
"Database Buffers",25165824,"bytes"
"Redo Buffers",3444736,"bytes"
SQL>
```

During this step, the Oracle software locates the parameter file (SPFILE or PFILE), allocates memory to the System Global Area (SGA), starts the background processes, and opens the alert log and trace files . At this stage, the database instance is started; however, users cannot access it yet. You would usually start in NOMOUNT mode if you were creating a database, re-creating control files, or performing certain backup and recovery tasks.

We can check if SPFILE file was used with the following command:

```
SQL> SELECT name, value FROM v$parameter WHERE name = 'spfile';
"NAME","VALUE"
"spfile","/u01/app/oracle/product/19.0.0/db_1/dbs/spfileorcl.ora"
```

Mount the database by using the ALTER DATABASE MOUNT command. During this step, the database instance mounts the database. This means that the database instance locates and opens all the control files specified in the initialization parameter file and reads the control files to obtain the names and statuses of the data files and online redo log files. The database instance does not, however, verify the existence of the data files and online redo log files at this time. You must mount the database, but not open it when you want to rename data files, enable/disable online redo log file archiving options, or perform a full database recovery.

```
SQL> ALTER DATABASE MOUNT;
Database altered.
SQL>
```

Open the database by using the **ALTER DATABASE** command. During this step, the database instance opens the data files for the CDB and online redo log files and checks the consistency of the database. When the database is open, all users can access the database instance.

```
SQL> ALTER DATABASE OPEN;
Database altered.
SQL> show con_name;
CON_NAME
------------------------------
CDB$ROOT
SQL> show pdbs;
    CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY NO
         3 PDB1                           MOUNTED
SQL>
```

Did you expect PDB1 to be open? By default, PDBs are mounted when a CDB is opened. But we can create a database event trigger to open all PDBs after startup.

```
SQL> CREATE OR REPLACE TRIGGER open_pdbs
  AFTER STARTUP ON DATABASE
BEGIN
   EXECUTE IMMEDIATE 'ALTER PLUGGABLE DATABASE ALL OPEN';
END open_pdbs;
/
Trigger created.
SQL> set markup csv on;
SQL> SELECT trigger_name, trigger_type, triggering_event, trigger_body FROM dba_triggers WHERE trigger_name LIKE
'OPEN%';
"TRIGGER_NAME","TRIGGER_TYPE","TRIGGERING_EVENT","TRIGGER_BODY"
"OPEN_PDBS","AFTER EVENT","STARTUP ","BEGIN
   EXECUTE IMMEDIATE 'ALTER PLUGGABLE DATABASE ALL OPEN';
END open_pdbs;
```

Let's shutdown and startup our database instance to make sure the PDB1 database will be opened automatically ( by a trigger ) now. OPEN_MODE should be READ WRITE

```
SQL> shutdown;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup;
ORACLE instance started.
"Total System Global Area",327151856,"bytes"
"Fixed Size",9134320,"bytes"
"Variable Size",289406976,"bytes"
"Database Buffers",25165824,"bytes"
"Redo Buffers",3444736,"bytes"
Database mounted.
Database opened.
SQL> show pdbs;
"CON_ID","CON_NAME","OPEN MODE","RESTRICTED"
2,"PDB$SEED","READ ONLY","NO"
3,"PDB1","READ WRITE","NO"
```

**V$DATABASE** displays information about the database from the control file. You can see that after we started up db without specifying any options the **OPEN_MODE** is **READ WRITE**. It is the same as **OPEN**

```
SQL> select dbid, name, open_mode from v$database;
"DBID","NAME","OPEN_MODE"
2500688251,"ORCLDB","READ WRITE"
SQL>
```

Let's shutdown the database instance and try to start it up, not in default mode. For example in **MOUNT** mode. In the following example I will also show how you can switch from **MOUNT** to **OPEN** mode with **ALTER DATABASE** command:

```
SQL> shutdown;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup MOUNT;
ORACLE instance started.
"Total System Global Area",327151856,"bytes"
"Fixed Size",9134320,"bytes"
"Variable Size",289406976,"bytes"
"Database Buffers",25165824,"bytes"
"Redo Buffers",3444736,"bytes"
Database mounted.
SQL> select dbid, name, open_mode from v$database;
"DBID","NAME","OPEN_MODE"
2500688251,"ORCLDB","MOUNTED"
SQL> alter database open;
```

```
Database altered.
SQL> select dbid, name, open_mode from v$database;
"DBID","NAME","OPEN_MODE"
2500688251,"ORCLDB","READ WRITE"
```

Let's close Database Instance one more time to start up from the backup configuration - pfile we created earlier. This could be helpful in some recovery scenarios - when the Database Instance is not able to be started after some changes:

```
SQL> shutdown;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> startup pfile='/home/oracle/pfile_20210630.conf';
ORACLE instance started.
Total System Global Area 327151856 bytes
Fixed Size                  9134320 bytes
Variable Size             289406976 bytes
Database Buffers           25165824 bytes
Redo Buffers                3444736 bytes
Database mounted.
Database opened.
SQL>
```

To check that our database instance didn't started from spfile we can run the following query and make sure there is empty value for spfile parameter:

```
SQL> SELECT name, value FROM v$parameter WHERE name = 'spfile';
"NAME","VALUE"
"spfile",
```

Also we learned earlier that **V$SPPARAMETER** contains information about the contents of the server parameter file **SPFILE**. In our case a server parameter file was not used to start the instance, then each row of the view will contain **FALSE** in the **ISSPECIFIED** column. We can confirm this by running the following query to check all the values for the **ISSPECIFIED** column and count for each value - all parameters have FALSE values for the **ISSPECIFIED** column. It is expected:

```
SQL> SELECT ISSPECIFIED, COUNT(*) FROM V$SPPARAMETER GROUP BY ISSPECIFIED;
"ISSPECIFIED","COUNT(*)"
"FALSE",448
```

## How to open and close PDBs?

Starting up a PDB and opening a PDB mean the same thing. When you open a PDB, the database server opens the data files for that PDB.
A PDB has four open modes:

- READ WRITE (the PDB is fully started/opened)
- READ ONLY
- MIGRATE
- MOUNTED (the PDB is shut down/closed)

You can use the **ALTER PLUGGABLE DATABASE** command to open and close a PDB. Pay attention on the **OPEN_MODE** value after opening and closing PDB1.

```
SQL> show pdbs;
    CON_ID CON_NAME          OPEN MODE  RESTRICTED

---------- ------------------------------------------------------------
         2 PDB$SEED                  READ ONLY  NO
         3 PDB1                        MOUNTED
SQL> ALTER PLUGGABLE DATABASE PDB1 OPEN;
Pluggable database altered.
SQL> show pdbs;
    CON_ID CON_NAME          OPEN MODE  RESTRICTED
---------- ------------------------------------------------------------
         2 PDB$SEED                  READ ONLY  NO
         3 PDB1                        READ WRITE NO
SQL> ALTER PLUGGABLE DATABASE PDB1 CLOSE;
Pluggable database altered.
SQL> show pdbs;
    CON_ID CON_NAME          OPEN MODE  RESTRICTED
---------- ------------------------------------------------------------
         2 PDB$SEED                  READ ONLY  NO
         3 PDB1                        MOUNTED
SQL>
```

## What is an alert log file?

Each database instance has an alert_SID.log file. The alert log file is a chronological log of messages about the database instance and database.
Query **V$DIAG_INFO** view to find the location of the alert log.

```
SQL> select * from v$diag_info;
"INST_ID","NAME","VALUE","CON_ID"
1,"Diag Enabled","TRUE",0
1,"ADR Base","/u01/app/oracle",0
1,"ADR Home","/u01/app/oracle/diag/rdbms/orcldb/orcl",0
1,"Diag Trace","/u01/app/oracle/diag/rdbms/orcldb/orcl/trace",0
1,"Diag Alert","/u01/app/oracle/diag/rdbms/orcldb/orcl/alert",0
```

```
1,"Diag Incident","/u01/app/oracle/diag/rdbms/orcldb/orcl/incident",0
1,"Diag Cdump","/u01/app/oracle/diag/rdbms/orcldb/orcl/cdump",0
1,"Health Monitor","/u01/app/oracle/diag/rdbms/orcldb/orcl/hm",0
1,"Default Trace File","/u01/app/oracle/diag/rdbms/orcldb/orcl/trace/orcl_ora_4357.trc",0
1,"Active Problem Count","2",0
1,"Active Incident Count","229",0
1,"ORACLE_HOME","/u01/app/oracle/product/19.0.0/db_1",0
12 rows selected.
```

Check out the information that is written to alert log file:

```
[oracle@oracle ~]$ less /u01/app/oracle/diag/rdbms/orcldb/orcl/trace/alert_orcl.log
```

It is a good practice to familiarize yourself with the information written in the alert log file on startup and shutdown operations. I recommend saving these records to some file as an example of a successful start. And in the case of issues with starting up our database instance we can compare logs with saved ones and make some conclusions.
Create 2 sessions to the DB server. Shutdown the database instance with **SHUTDOWN** command if it is up. In one session run the following command to capture and save all records that will appear in alert log since we run this command:

```
[oracle@oracle ~]$ tail -f $ORACLE_BASE/diag/rdbms/orcldb/orcl/trace/alert_orcl.log | tee
$ORACLE_HOME/example_of_successfull_pdb_open_operation.log
```

In another session start up the database instance with **STARTUP** command:

```
SQL> startup
ORACLE instance started.
Total System Global Area 327151856 bytes
Fixed Size                 9134320 bytes
Variable Size            289406976 bytes
Database Buffers          25165824 bytes
Redo Buffers               3444736 bytes
Database mounted.
Database opened.
```

Review information in $ORACLE_HOME/example_of_successfull_pdb_open_operation.log

```
[oracle@oracle ~]$ less $ORACLE_HOME/example_of_successfull_pdb_open_operation.log
```

## What are Trace 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. If an internal error occurs and information is written to a trace file, the administrator should contact Oracle Support Services. When a critical error occurs, an incident number is assigned to it; diagnostic data for the error (such as trace files) is immediately captured and tagged with this number. The data is then stored in the Automatic Diagnostic Repository (ADR)—a file-based repository outside the database—where it can later be retrieved by incident number and analyzed.

## What is an Automatic Diagnostic Repository?

The Automatic Diagnostic Repository (ADR) is a file-based tracing and logging central repository for database diagnostic data such as traces, the alert log, health monitor reports, and more. It is stored outside of any database and available for problem diagnosis when the database is down.
To access, navigate and manage the contents of ADR we use the Automatic Diagnostic Repository Command Interpreter (ADRCI) tool.

```
[oracle@oracle ~]$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base remains unchanged with value /u01/app/oracle
[oracle@oracle ~]$ adrci
ADRCI: Release 19.0.0.0.0 - Production on Thu Jul 1 13:15:53 2021
Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.
ADR base = "/u01/app/oracle"
adrci>
```

To show you the ADR homes in the current ADRCI session you can run **SHOW HOMES** command:

```
adrci> show homes;
ADR Homes:
diag/rdbms/orcldb/orcl
diag/clients/user_oracle/host_2310821773_110
diag/tnslsnr/oracle/listener
```

---

To make our Oralce Database Instance HOME current home we use **SET HOME** command:

```
adrci> set home diag/rdbms/orcldb/orcl
adrci>
```

To view the contents of it's alert log file run **SHOW ALERT** command. The **VI** editor is used. Here is some basic operation you may want to perform in **VI** editor:

- You can press the Esc key and then press Shift + G **to move the cursor to the end of the file** in **VI** editor**.**
- Type the forward slash (/) and followed by the string (or word) you want **to search**.
- **To quit** press the Esc key and then type ":q" and press Enter

```
adrci> show alert
ADR Home = /u01/app/oracle/diag/rdbms/orcldb/orcl:
********************************************************************************
Output the results to file: /tmp/alert_5105_14019_orcl_1.ado
```

---

To see the list of all available commands run the **HELP** command:

```
adrci> help
 HELP [topic]
  Available Topics:
        CREATE REPORT
        ECHO
        ESTIMATE
        EXIT
        HELP
        HOST
        IPS
        PURGE
        RUN
        SELECT
        SET BASE
        SET  BROWSER
        SET CONTROL
        SET ECHO
        SET EDITOR
        SET HOMES | HOME | HOMEPATH
        SET TERMOUT
        SHOW ALERT
        SHOW BASE
        SHOW CONTROL
        SHOW HM_RUN
        SHOW HOMES | HOME | HOMEPATH
        SHOW INCDIR
        SHOW INCIDENT
        SHOW LOG
        SHOW PROBLEM
        SHOW REPORT
        SHOW TRACEFILE
        SPOOL
 There are other commands intended to be used directly by Oracle, type
 "HELP EXTENDED" to see the list
adrci>
```

To see the list of  all the incidents associated with the current ADR home you can use **SHOW INCIDENT** command:

```
adrci> show incident;
ADR Home = /u01/app/oracle/diag/rdbms/orcldb/orcl:
*************************************************************************
INCIDENT_ID          PROBLEM_KEY                                 CREATE_TIME
                                                                 ----------------------
60225                ORA 4031                                    2021-05-22 15:31:03.302000 -04:00
60226                ORA 4031                                    2021-05-22 16:01:02.849000 -04:00
60273                ORA 4031                                    2021-05-22 16:01:17.612000 -04:00
60227                ORA 4031                                    2021-05-22 16:05:02.815000 -04:00
60228                ORA 4031                                    2021-05-22 17:01:02.802000 -04:00
60229                ORA 4031                                    2021-05-22 17:01:09.348000 -04:00
60230                ORA 4031                                    2021-05-22 17:01:14.029000 -04:00
60231                ORA 4031                                    2021-05-22 17:02:00.141000 -04:00
60232                ORA 4031                                    2021-05-22 17:02:04.710000 -04:00
62678                ORA 4031                                    2021-05-22 18:17:03.404000 -04:00
62679                ORA 4031                                    2021-05-22 18:17:08.842000 -04:00
62680                ORA 4031                                    2021-05-22 18:17:10.625000 -04:00
62846                ORA 4031                                    2021-05-22 18:17:16.912000 -04:00
62847                ORA 4031                                    2021-05-22 18:17:20.642000 -04:00
65108                ORA 4031                                     2021-05-23  06:26:30.748000  -04:00
65109                ORA 4031                                    2021-05-23 06:27:26.289000 -04:00
65110                ORA 4031                                    2021-05-23 06:28:27.822000 -04:00
65111                ORA 4031                                    2021-05-23 06:32:18.972000 -04:00
65112                ORA 4031                                    2021-05-23 06:33:18.341000 -04:00
74950                ORA 4031                                    2021-06-04 11:19:38.569000 -04:00
74951                ORA 4031                                    2021-06-04 11:19:44.427000 -04:00
74952                ORA 4031                                    2021-06-04 11:19:50.054000 -04:00
74942                ORA 4031                                    2021-06-04 11:20:37.246000 -04:00
74943                ORA 4031                                    2021-06-04 11:20:42.300000 -04:00
77002                ORA  7445  [qmhProcessRequestData]          2021-06-04 12:18:30.437000 -04:00
25 rows fetched
adrci>
```

To see more details about some specific incident we run **SHOW INCIDENT** command with specifying **INCIDENT ID**. This information can be requested by Oracle Support. But at first lets check the correct syntax for this command with **HELP SHOW INCIDENT** command:

```
adrci> HELP SHOW INCIDENT
  Usage: SHOW INCIDENT [-p <predicate_string>]
                       [-mode BASIC|BRIEF|DETAIL]
                       [-last <num> | -all]
                       [-orderby (field1, field2, ...) [ASC|DSC]]
```

```
Purpose: Show the incident information. By default, this command will
         only show the last 50 incidents which are not flood controlled.
Options:
   [-p <predicate_string>]: The predicate string must be double-quoted.
   [-mode BASIC|BRIEF|DETAIL]: The different modes of showing incidents.
   BASIC will show the basic information of non-flooded controlled
   incidents, which is the default mode. In this mode, only the following
   fields can be used in the predicate clause:
      INCIDENT_ID               number
      PROBLEM_KEY                text(550)
      CREATE_TIME               timestamp
   BRIEF will display incident information from the incident relation.
   In this mode, the fields can appear in the predicate are:
      INCIDENT_ID               number
      PROBLEM_ID                number
      CREATE_TIME               timestamp
      CLOSE_TIME                timestamp
      STATUS                  number
      FLAGS                   number
      FLOOD_CONTROLLED              number
      ERROR_FACILITY             text(10)
      ERROR_NUMBER               number
      ERROR_ARG1                text(64)
      ERROR_ARG2                text(64)
      ERROR_ARG3                text(64)
      ERROR_ARG4                text(64)
      ERROR_ARG5                text(64)
      ERROR_ARG6                text(64)
      ERROR_ARG7                text(64)
      ERROR_ARG8                text(64)
      SIGNALLING_COMPONENT          text(64)
      SIGNALLING_SUBCOMPONENT        text(64)
      SUSPECT_COMPONENT             text(64)
      SUSPECT_SUBCOMPONENT          text(64)
      ECID                    text(64)
      IMPACT                    number
      CON_UID                   number
   DETAIL will display all incident-related information, such as incident
   files. The fields can appear in the predicate is the same as the ones
   in the brief mode.
   [-last <num> | -all]: This option allows users to either select
   the last <num> of qualified incidents to show or to show all the
   qualified incidents. If this option is not specified, this command
   will only show 50 incidents.
   [-orderby (field1, field2, ...) [ASC|DSC]]: If specified, the results
   will be ordered by the specified fields' values. By default, it will be
   in the ascending order unless "DSC" is specified. Note that the field
   names that can be specified here are from the "INCIDENT" relation.
 Examples:
   show incident
   show incident -mode detail
   show incident -mode detail -p "incident_id=123"
adrci> show incident -mode detail -p "incident_id=77002"
ADR Home = /u01/app/oracle/diag/rdbms/orcldb/orcl:
*************************************************************************
**********************************************************
INCIDENT INFO RECORD 1
**********************************************************
  INCIDENT_ID                77002
  STATUS                   ready
  CREATE_TIME                2021-06-04 12:18:30.437000 -04:00
  PROBLEM_ID                 2
  CLOSE_TIME                 <NULL>
  FLOOD_CONTROLLED               none
  ERROR_FACILITY              ORA
  ERROR_NUMBER                7445
  ERROR_ARG1                 qmhProcessRequestData
  ERROR_ARG2                 SIGSEGV
  ERROR_ARG3                 ADDR:0x630
  ERROR_ARG4                 PC:0xB02B9D7
  ERROR_ARG5                 Address not mapped to object
  ERROR_ARG6                 <NULL>
  ERROR_ARG7                 <NULL>
  ERROR_ARG8                 <NULL>
  ERROR_ARG9                 <NULL>
  ERROR_ARG10                 <NULL>
  ERROR_ARG11                 <NULL>
  ERROR_ARG12                 <NULL>
```

```
SIGNALLING_COMPONENT          XDB_Protocols
SIGNALLING_SUBCOMPONENT       <NULL>
SUSPECT_COMPONENT             <NULL>
SUSPECT_SUBCOMPONENT          <NULL>
ECID                          <NULL>
IMPACTS                       0
CON_UID                       1
PROBLEM_KEY                   ORA 7445 [qmhProcessRequestData]
FIRST_INCIDENT                77002
FIRSTINC_TIME                 2021-06-04 12:18:30.437000 -04:00
LAST_INCIDENT                 77002
LASTINC_TIME                  2021-06-04 12:18:30.437000 -04:00
IMPACT1                       0
IMPACT2                       0
IMPACT3                       0
IMPACT4                       0
KEY_NAME                      PdbName
KEY_VALUE                     CDB$ROOT
KEY_NAME                      Module
KEY_VALUE                     oracle@oracle (S001)
KEY_NAME                      ProcId
KEY_VALUE                     46.10
KEY_NAME                      Service
KEY_VALUE                     SYS$USERS
KEY_NAME                      SID
KEY_VALUE                     459.60167
KEY_NAME                      Client ProcId
KEY_VALUE                     oracle@oracle.2018_140711314912192
OWNER_ID                      1
INCIDENT_FILE                 /u01/app/oracle/diag/rdbms/orcldb/orcl/trace/orcl_s001_2018.trc
OWNER_ID                      1
INCIDENT_FILE                 /u01/app/oracle/diag/rdbms/orcldb/orcl/incident/incdir_77002/orcl_s001_2018_i77002.trc
1 row fetched
adrci>
```

**SHOW PROBLEM** command will show you information about unique problems in the current ADR home:

```
adrci> show problem
ADR Home = /u01/app/oracle/diag/rdbms/orcldb/orcl:
*************************************************************************
PROBLEM_ID          PROBLEM_KEY                                 LAST_INCIDENT        LASTINC_TIME
-------------------------------------------------------------------------------------------------------------------------------
-----
1                   ORA 4031                                    77000                2021-06-04 12:11:11.793000 -04:00
2                   ORA 7445 [qmhProcessRequestData]            77002                2021-06-04 12:18:30.437000 -04:00
2 rows fetched
```

## How to manage the size of the Automatic Diagnostic Repository?

Sometimes we are facing space related issues due to the huge number of trace file generation. Automatic purging can help us in this situation

- Set homepath to your Oracle Database Instance

```
adrci> show homes
ADR Homes:
diag/rdbms/orcldb/orcl
diag/clients/user_oracle/host_2310821773_110
diag/tnslsnr/oracle/listener
adrci> set home diag/rdbms/orcldb/orcl
adrci>
```

- The automatic purging runs on schedule defined in retention policy .To check current policy we can run the following query against **ADR_CONTROL** view. The long retention period is used for the relatively higher-value diagnostic data, such as incidents and alert log (default value is 365 days or 8760 hours). The short retention period is used for traces and core dumps (default value is 30 days or 720 hours).

```
adrci> select SHORTP_POLICY,LONGP_POLICY from ADR_CONTROL;
ADR Home = /u01/app/oracle/diag/rdbms/orcldb/orcl:
*************************************************************************
SHORTP_POLICY       LONGP_POLICY
---------------------------------------
720                 8760
1 row fetched
```

- To change the retention policy we use the **SET CONTROL** command.

```
adrci> set control (SHORTP_POLICY=120);
adrci> set control (LONGP_POLICY=720);
# To check the result:
adrci> select SHORTP_POLICY,LONGP_POLICY,LAST_AUTOPRG_TIME,LAST_MANUPRG_TIME from ADR_CONTROL;
ADR Home = /u01/app/oracle/diag/rdbms/orcldb/orcl:
*************************************************************************
SHORTP_POLICY       LONGP_POLICY       LAST_AUTOPRG_TIME                LAST_MANUPRG_TIME
```

```
---------- ---------- ----------------------------------- ----------------------------------------
120        720        2021-06-28 12:07:08.806208 -04:00
1 row fetched
adrci>
```

## How to purge alert logs and trace files manually?

- Set homepath to your Oracle Database Instance

```
adrci> show homes
ADR Homes:
diag/rdbms/orcldb/orcl
diag/clients/user_oracle/host_2310821773_110
diag/tnslsnr/oracle/listener
adrci> set home diag/rdbms/orcldb/orcl
adrci>
```

- Let's purge diagnostic data that is 1 day old (1440 minutes).

```
adrci> PURGE -age 1440 -type ALERT
adrci> PURGE -age 1440 -type TRACE
```

- To remove all data older than one minute use:

```
adrci> PURGE -age 1 -type ALERT
adrci> SHOW ALERT
ADR Home = /u01/app/oracle/diag/rdbms/orcldb/orcl:
*************************************************************************
No readable alert log in selected home
adrci> exit
```

**How to enable the capture of DDL statements to a DDL log file?**

Data definition language (DDL) refers to the set of SQL commands that can create and manipulate the structures of a database. DDL statements are used to create, change, and remove objects including indexes, triggers, tables, and views. Common DDL statements include:

- CREATE (generates a new table)
- ALTER (alters table)
- DROP (removes a table from the database)

You can enable the capture of certain DDL statements to a DDL log file by setting **ENABLE_DDL_LOGGING** initialization parameter to TRUE

```
SQL> set markup csv on;
SQL> SELECT name, value, isses_modifiable, issys_modifiable, ispdb_modifiable from V$PARAMETER where name in
('enable_ddl_logging');
"NAME","VALUE","ISSES_MODIFIABLE","ISSYS_MODIFIABLE","ISPDB_MODIFIABLE"
"enable_ddl_logging","FALSE","TRUE","IMMEDIATE","TRUE"
# ISSYS_MODIFIABLE=TRUE means we can change this parameter without restart
SQL> ALTER SYSTEM SET enable_ddl_logging=TRUE SCOPE=BOTH;
System altered.
SQL> SELECT name, value, isses_modifiable, issys_modifiable, ispdb_modifiable from V$PARAMETER where name in
('enable_ddl_logging');
"NAME","VALUE","ISSES_MODIFIABLE","ISSYS_MODIFIABLE","ISPDB_MODIFIABLE"
"enable_ddl_logging","TRUE","TRUE","IMMEDIATE","TRUE"
SQL>
```

Let's now run some DDL statement inside PDB1 to make sure our change works and we can see DDL statements in logs:

```
SQL> alter session set container=PDB1;
Session altered.
SQL> show pdbs;
"CON_ID","CON_NAME","OPEN MODE","RESTRICTED"
3,"PDB1","MOUNTED",
SQL> alter pluggable database open;
Pluggable database altered.
SQL> show pdbs;
"CON_ID","CON_NAME","OPEN MODE","RESTRICTED"
3,"PDB1","READ WRITE","NO"
SQL> create user monica identified by PSSWD_2021;
User created.
SQL> drop user monica;
User dropped.
```

---

Let's check DDL logs. You can see the **DROP USER** statement in it:

```
[oracle@oracle ~]$ cat /u01/app/oracle/diag/rdbms/orcldb/orcl/log/ddl_orcl.log
diag_adl:alter pluggable database open
2021-07-02T05:00:21.579168-04:00
diag_adl:truncate table wri$_adv_addm_pdbs
diag_adl:drop user monica
```

## What are Dynamic Performance Views?

The Oracle Database server maintains a dynamic set of data about the operation and performance of the database instance.
To see all views you can query the **V$FIXED_TABLE** view**:**

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

---

Here you are some example of questions we can answer with the help of Dynamic Performance Views:

- To see the list of background processes that are running:

```
SQL> select * from V$BGPROCESS;
```

- To see current sessions logged in to Database Instance we can query **V$SESSION** view. To see the list of columns in this view we can use **DESCRIBE** command

```
SQL> desc v$session;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 SADDR                                              RAW(8)
 SID                                                NUMBER
 SERIAL#                                            NUMBER
 AUDSID                                             NUMBER
 PADDR                                              RAW(8)
 USER#                                              NUMBER
 USERNAME                                           VARCHAR2(128)
 COMMAND                                            NUMBER
 OWNERID                                            NUMBER
 TADDR                                              VARCHAR2(16)
 LOCKWAIT                                           VARCHAR2(16)
 STATUS                                             VARCHAR2(8)
 SERVER                                             VARCHAR2(9)
...
SQL>
SELECT
  s.type,
  p.name,
  p.inst_id,
  s.status,
  s.server,
  s.machine,
  s.module,
  count(*) cnt
FROM
  gv$session s join gv$pdbs p on p.con_id = s.con_id and p.inst_id=s.inst_id
 GROUP BY s.type, p.name, p.inst_id, s.status, s.server, s.machine, s.module
 ORDER BY 1,2,3;
no rows selected
```

We got “NO ROWS SELECTED” result because there is no user connected to DB Instance right now but if we open another terminal and create another connection to PDB1 and repeat our query in first terminal we would see the connection from the 2nd Terminal

```
# 2nd TERMINAL:
[oracle@oracle ~]$ sqlplus / as sysdba
SQL*Plus: Release 19.0.0.0.0 - Production on Sat Jul 3 18:56:37 2021
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle.  All rights reserved.
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
SQL> show pdbs;
      CON_ID CON_NAME                       OPEN MODE  RESTRICTED
---------- ------------------------------ ---------- ----------
         2 PDB$SEED                       READ ONLY  NO
         3 PDB1                           READ WRITE NO
SQL> alter session set container=PDB1;
Session altered.
```

```
# 1st TERMINAL:
SQL> set markup csv on;
SQL> SELECT
  s.type,
p.name,
p.inst_id,
s.status,
s.server,
s.machine,
s.module,
count(*) cnt
FROM
  gv$session s join gv$pdbs p on p.con_id = s.con_id and p.inst_id=s.inst_id
 GROUP BY s.type, p.name, p.inst_id, s.status, s.server, s.machine, s.module
 ORDER BY 1,2,3;

"TYPE","NAME","INST_ID","STATUS","SERVER","MACHINE","MODULE","CNT"
"USER","PDB1",1,"INACTIVE","DEDICATED","oracle","sqlplus@oracle (TNS V1-V3)",1
SQL>
```

- To see information about file states. Pay attention there are files that belongs to CDB$ROOT ( CON_ID=1), PDB$SEED (CON_ID=2) and PDB1 (CON_ID=3)

```
SQL> DESC V$DATAFILE
SQL> select FILE#, CREATION_TIME, STATUS, NAME, CON_ID from v$datafile;
"FILE#","CREATION_TIME","STATUS","NAME","CON_ID"
1,"17-APR-19","SYSTEM","/u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/system01.dbf",1
3,"17-APR-19","ONLINE","/u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/sysaux01.dbf",1
4,"17-APR-19","ONLINE","/u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/undotbs01.dbf",1
5,"17-FEB-21","SYSTEM","/u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/pdbseed/system01.dbf",2
6,"17-FEB-21","ONLINE","/u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/pdbseed/sysaux01.dbf",2
7,"17-APR-19","ONLINE","/u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/users01.dbf",1
8,"17-FEB-21","ONLINE","/u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/pdbseed/undotbs01.dbf",2
9,"17-FEB-21","SYSTEM","/u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/pdb1/system01.dbf",3
10,"17-FEB-21","ONLINE","/u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/pdb1/sysaux01.dbf",3
11,"17-FEB-21","ONLINE","/u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/pdb1/undotbs01.dbf",3
12,"17-FEB-21","ONLINE","/u01/app/oracle/product/19.0.0/db_1/mydbfiles/ORCLDB/pdb1/users01.dbf",3
11 rows selected.
```

- What are sessions IDs of those sessions that are currently holding a lock that is blocking another user, and how long have those locks been held?

```
SQL> SELECT sid,ctime FROM V$LOCK WHERE block > 0;
```

- For which SQL statements ( and their associated numbers of executions ) is the CPU time consumed greater than 200000 microseconds?

```
SQL> SELECT sql_text, executions FROM V$SQL WHERE cpu_time > 200000;
```

## How to shut down an Oracle Database Instance?

To shut down the database instance you can use the **SHUTDOWN** command with different options to shut down the database instance in various modes: **ABORT, IMMEDIATE, TRANSACTIONAL**, or **NORMAL**. The difference between these modes is in how NEW connections,current sessions or transactions are processed during the shutdown process.

The **SHUTDOWN ABORT** is not recommended and only used if the other shutdown modes don't work. The **SHUTDOWN ABORT** has a similar effect as you unplug the power of the server. The database will be in an inconsistent state and instance recovery is required on the next startup, which occurs automatically.

| Shutdown Modes | ABORT | IMMEDIATE | TRANSACTIONAL | NORMAL |
|---|---|---|---|---|
| Allows new connections | NO | NO | NO | NO |
| Waits until current sessions end | NO | NO | NO | YES |
| Waits until current transactions end | NO | NO | YES | YES |
| Forces a checkpoint and closes files | NO | YES | YES | YES |
| | Inconsistent database<br><br>* Modified buffers not written to data files<br>* Uncommited changes not rolled back<br><br>On startup:<br><br>* Online redo log files used to reapply changes<br>* Undo segments used to roll back uncommited changes<br><br>* Resources released<br><br>SQL> shutdown abort | Consistent database<br><br>* Uncommittted chages rolled back, for IMMEDIATE<br>* Database buffer cache written to data files<br>* Resources released<br>* No instance recover on startup<br><br>SQL> shutdown<br>SQL> shutdown transactional<br>SQL> shutdown immediate | | |

The **SHUTDOWN IMMEDIATE** is the most common and practical way to shut down the Oracle database. The **SHUTDOWN IMMEDIATE** does not wait for the current users to disconnect from the database or current transactions to complete. During the **SHUTDOWN IMMEDIATE**, all the connected sessions are disconnected immediately, all uncommitted transactions are rolled back, and the database completely shuts down. After issuing the **SHUTDOWN IMMEDIATE** statement, the database will not accept any new connection. The statement will also close and dismount the database. Unlike the **SHUTDOWN ABORT** option, the **SHUTDOWN IMMEDIATE** option does not require an instance recovery on the next database startup.

**NORMAL** is the default shutdown mode if no mode is specified with the **SHUTDOWN** command. But in the case of some issue you want to restart the database as quickly as possible but with **NORMAL** mode it can take a very long time to do it because Oracle server waits for all users to disconnect before completing the shutdown.

During this mode of shutdown, the database instance closes the database — all data files and online redo log files are closed. Next, the database instance dismounts the database — all control files associated with the database instance are closed. Lastly, the Oracle software shuts down the database instance—background processes are terminated, and the System Global Area (SGA) is removed from memory. When a database instance shuts down in normal mode, the database instance waits for all users to disconnect before completing the shutdown, and no new connections are allowed. Control is not returned to the session that initiates a database shutdown until shutdown is complete.

Let's try to **SHUTDOWN** DB Instance with one open session. Open terminal connection to DB and login as sysdba user

```
[oracle@oracle ~]$ sqlplus / as sysdba
SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jul 2 05:16:42 2021
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle.  All rights reserved.
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
```

Open another terminal session, check the OPEN_MODE = READ WRITE for CDB and try to SHUTDOWN DB Instance in NORMAL mode - without any options

```
SQL> select dbid,name,open_mode from v$database;
```

```
      DBID NAME      OPEN_MODE
---------- --------- --------------------
2500688251 ORCLDB    READ WRITE
SQL>
SQL> shutdown;
```

You should observe that the **SHUTDOWN** command hangs and does not provide any output. Now you can go to the 1st terminal session and logout from SQL*PLUS console:

```
SQL> exit
Disconnected from Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
```

In the 2nd terminal session you should see that after you closed the 1st session the SHUTDOWN command was able to complete and now you can not query v$database view because instance is closed

```
SQL> shutdown;
Database closed.
Database dismounted.
ORACLE instance shut down.
SQL> select dbid,name,open_mode from v$database;
select dbid,name,open_mode from v$database
*
ERROR at line 1:
ORA-01034: ORACLE not available
Process ID: 0
Session ID: 0 Serial number: 0
SQL>
```

Try to repeat the same procedure but with **SHUTDOWN IMMEDIATE** command. Now you should see that the SHUTDOWN process happens even with a connected session.

## What is a Data Dictionary?

Oracle data dictionary contains metadata or internal information about all objects in the database. Oracle Database accesses the data dictionary frequently during SQL statement parsing. This access is essential to the continuing operation of Oracle Database.

We can query the Data Dictionary to find information about users, objects, constraints and storage. Some example of questions we can answer by querying Data Dictionary views:

- The list of tables ( along with the names of tablespaces where they reside ) that have been created in your schema?

```
[oracle@oracle ~]$ sqlplus / as sysdba
SQL*Plus: Release 19.0.0.0.0 - Production on Sun Jul 4 06:45:46 2021
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle.  All rights reserved.
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
SQL> alter session set container=PDB1;
Session altered.
SQL> set markup csv on;
SQL> SELECT owner,table_name,tablespace_name from all_tables where owner='HR';
"OWNER","TABLE_NAME","TABLESPACE_NAME"
"HR","DEPARTMENTS","SYSAUX"
"HR","DEPARTMENTS_NEW","SYSAUX"
"HR","EMPLOYEES","SYSAUX"
"HR","JOBS","SYSAUX"
"HR","JOB_HISTORY","SYSAUX"
"HR","LOCATIONS","SYSAUX"
"HR","LOCATIONS2","SYSAUX"
"HR","REGIONS","SYSAUX"
"HR","COUNTRIES",
9 rows selected.
```

To answer this question we used the **ALL_TABLES** data dictionary view. If you want to list all tables for the current user you can use **USER_TABLES** that describes the relational tables owned by the current user. In our case we will see all tables owned by **SYSDBA** user. To see all tables owned by an HR user we need to connect as an HR user. Net listener should be running for this example - it can be started with the **LSNRCTL START** command in the shell console.

```
[oracle@oracle ~]$ lsnrctl start
[oracle@oracle ~]$ sqlplus hr/hr@localhost:1521/pdb1
SQL*Plus: Release 19.0.0.0.0 - Production on Sun Jul 4 06:52:27 2021
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle.  All rights reserved.
Last Successful login time: Fri Jun 04 2021 10:51:38 -04:00
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
SQL> set markup csv on;
SQL> SELECT table_name,tablespace_name from user_tables;
"TABLE_NAME","TABLESPACE_NAME"
"DEPARTMENTS_NEW","SYSAUX"
"LOCATIONS2","SYSAUX"
"REGIONS","SYSAUX"
```

```
"COUNTRIES",
"LOCATIONS","SYSAUX"
"DEPARTMENTS","SYSAUX"
"JOBS","SYSAUX"
"EMPLOYEES","SYSAUX"
"JOB_HISTORY","SYSAUX"
9 rows selected.
```

- Information about sequences in the database that you have access to:

```
[oracle@oracle ~]$ sqlplus / as sysdba
SQL*Plus: Release 19.0.0.0.0 - Production on Sun Jul 4 08:42:44 2021
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle.  All rights reserved.
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
SQL> alter session set container=PDB1;
Session altered.
SQL> set markup csv on;
SQL> SELECT
  sequence_name, min_value, max_value, increment_by
FROM
  all_sequences
WHERE sequence_owner IN ('HR');
"SEQUENCE_NAME","MIN_VALUE","MAX_VALUE","INCREMENT_BY"
"DEPARTMENTS_SEQ",1,9990,10
"EMPLOYEES_SEQ",1,1.0000E+28,1
"LOCATIONS_SEQ",1,9900,100
```

- What users in this database are currently able to log in:

```
[oracle@oracle ~]$ sqlplus / as sysdba
SQL*Plus: Release 19.0.0.0.0 - Production on Sun Jul 4 09:08:28 2021
Version 19.3.0.0.0
Copyright (c) 1982, 2019, Oracle.  All rights reserved.
Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0
SQL> alter session set container=PDB1;
Session altered.
SQL> set markup csv on;
SQL> SELECT username FROM dba_users where account_status='OPEN';
"USERNAME"
"SYS"
"SYSTEM"
"PDBADMIN"
"HR"
```

- To check information about indexes you can query DBA_INDEXES. To see what information you can view about all the indexes you can use DESCRIBE command:

```
SQL> DESCRIBE DBA_INDEXES
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OWNER                                     NOT NULL VARCHAR2(128)
 INDEX_NAME                                NOT NULL VARCHAR2(128)
 INDEX_TYPE                                         VARCHAR2(27)
 TABLE_OWNER                               NOT NULL VARCHAR2(128)
 TABLE_NAME                                NOT NULL VARCHAR2(128)
 TABLE_TYPE                                         VARCHAR2(11)
 UNIQUENESS                                         VARCHAR2(9)
 COMPRESSION                                        VARCHAR2(13)
...
```

- To show the names and descriptions of data dictionary tables and views:

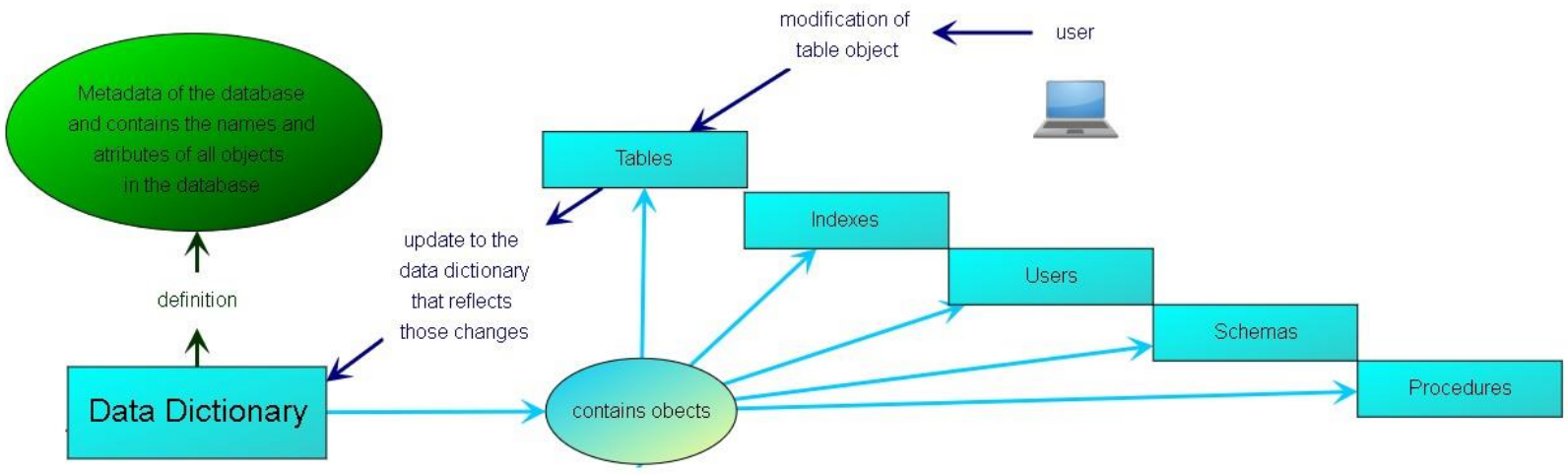
```
SQL> SELECT * FROM DICTIONARY;
```

- To view columns and their definitions:

```
SQL> SELECT * FROM DICT_COLUMNS;
```

You can refer to official documentation to see the description for any data dictionary view: [All Static Data Dictionary Views]

Remember when you update a table or any database object oracle database updates the data dictionary that reflects those changes. It is done automatically. You should never modify data dictionary tables or views directly by using SQL because this could lead to inconsistency.




| | Who Can Query | Contents | Subset of | Notes |
|---|---|---|---|---|
| CDB_ | DBA | Everything | N/A | display metadata for all objects in a CDB across all PDBs. |
| DBA_ | DBA | Everything | N/A | display metadata for all objects in a container or PDB |
| ALL_ | Everyone | Everything that the user has privileges to see | DBA_views | Includes user's own objects and other objects that the user has been granted privileges to see |
| USER_ | Everyone | Everything that the user owns | ALL_views | Includes user's own objects |

CDB_, DBA_, ALL_, and USER_ Views

The view prefixes, as shown in the picture above, indicate the data (and how much of that data) a given user can see.

- CDB_ views display metadata for all objects in a CDB across all PDBs.
- DBA_ views display metadata for all objects in a container or PDB.
- ALL_views display metadata for objects that the current user is privileged to see, whether the user owns them or not.
- USER_ views display metadata for all objects owned by the current user; that is, objects that are present in the user's own schema.

Only USER_ and ALL_ views are available to any user. The CDB_ and DBA_ views are restricted to DBA accounts.