

UNIT 2

Managing Database Instances

(5 Hours) — Database Administration (CACS405)

*Note: These notes are prepared for study purposes and **may contain errors or omissions**. Syntax, file paths and screen names differ between Oracle versions and operating systems (examples here follow Oracle 19c on Linux). Please verify important points with your textbook, class notes and the official Oracle documentation, and test commands on a practice database, never on a production system.*

2.1 Oracle Database Installation

Installing Oracle Database is usually done on Linux (or Windows) using the **Oracle Universal Installer (OUI)**. In Oracle 19c the software image is unzipped directly into the **Oracle Home** directory and then `runInstaller` is launched. Careful preparation of the operating system avoids most installation errors.

2.1.1 Pre-installation Requirements

- **Hardware:** enough RAM (minimum about 2 GB, more is better), swap space, and disk space for software and data files; check the certification matrix for the OS version.
- **Operating system packages, kernel parameters and shell limits** configured (on Oracle Linux, the preinstall package sets most of these).
- **OS user and groups:** a software owner (commonly `oracle`) and groups `oinstall` (inventory group) and `dba` (OSDBA).
- **Directories:** follow the Optimal Flexible Architecture (OFA) layout for `ORACLE_BASE`, `ORACLE_HOME` and data file locations.
- **Networking:** fixed hostname/IP entry in `/etc/hosts`; required ports (default listener port 1521) open in the firewall.

Term	Meaning	Example
ORACLE_BASE	Top of the Oracle directory tree	/u01/app/oracle
ORACLE_HOME	Location of the installed Oracle software	/u01/app/oracle/product/19.0.0/dbhome_1
ORACLE_SID	System identifier of the instance on this server	ORCL
Oracle Inventory	Central list of installed Oracle software	/u01/app/oraInventory

2.1.2 Installation Steps (Linux, summary)

- Log in as the `oracle` user; set `ORACLE_BASE`, `ORACLE_HOME` and `PATH` in the profile.
- Unzip the database software into `ORACLE_HOME` and run `./runInstaller`.
- Choose **Set Up Software Only** or **Create and configure a single instance database**; choose the edition (Enterprise/Standard).
- Enter the base location, inventory location and OS groups (OSDBA, OSOPER, etc.).
- Review the prerequisite check results and fix any failures; then start the install.
- When prompted, run the root scripts as **root**: `orainstRoot.sh` and `root.sh`.
- Verify: `sqlplus / as sysdba` and check `SELECT banner FROM v$version;`

Note: Screen names and the exact steps differ by version; always read the installation guide for your release.

2.2 Database Creation

A database can be created in two ways: with the **Database Configuration Assistant (DBCA)**, which is the recommended graphical or silent-mode tool, or manually with the `CREATE DATABASE` statement.

2.2.1 Using DBCA

- Run `dbca`; choose **Create a database** and Typical or Advanced configuration.
- Give the **global database name** and SID; choose whether it is a container database (CDB) with pluggable databases (Unit 5).
- Select storage (file system or ASM), the fast recovery area and archiving options.
- Set memory (SGA/PGA), character set (e.g. `AL32UTF8`), and passwords for SYS and SYSTEM.
- Optionally register with Enterprise Manager and generate creation scripts; then finish.

DBCA can also run non-interactively (`dbca -silent -createDatabase ...`) using a response file, which is convenient for automation.

2.2.2 Manual Creation (outline)

- Set the environment (ORACLE_SID, ORACLE_HOME) and create the directories and a password file if needed.
- Create a minimal parameter file (DB_NAME, CONTROL_FILES, memory settings, DB_BLOCK_SIZE).
- Start the instance with STARTUP NOMOUNT.
- Run CREATE DATABASE; then run the catalog scripts (catalog.sql, catproc.sql) to build the data dictionary.

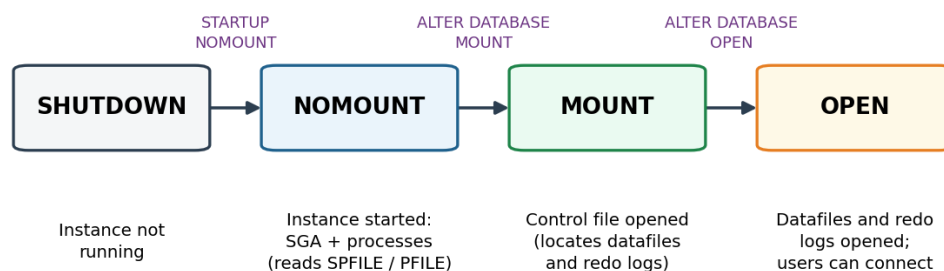
```
CREATE DATABASE orcl
  USER SYS IDENTIFIED BY sys_password
  USER SYSTEM IDENTIFIED BY system_password
  LOGFILE GROUP 1 ('/u01/oradata/ORCL/redo01.log') SIZE 100M,
           GROUP 2 ('/u01/oradata/ORCL/redo02.log') SIZE 100M,
           GROUP 3 ('/u01/oradata/ORCL/redo03.log') SIZE 100M
  CHARACTER SET AL32UTF8
  EXTENT MANAGEMENT LOCAL
  DATAFILE '/u01/oradata/ORCL/system01.dbf' SIZE 700M
  SYSAUX DATAFILE '/u01/oradata/ORCL/sysaux01.dbf' SIZE 550M
  DEFAULT TEMPORARY TABLESPACE temp
           TEMPFILE '/u01/oradata/ORCL/temp01.dbf' SIZE 20M
  UNDO TABLESPACE undotbs1
           DATAFILE '/u01/oradata/ORCL/undotbs01.dbf' SIZE 200M;
```

Listing 2.1: Simplified CREATE DATABASE statement

Every database needs at least the **SYSTEM** (data dictionary), **SYSAUX** (auxiliary components), an **UNDO** tablespace (for rollback and read consistency) and a **TEMP** tablespace (for sorts).

2.3 Starting Up and Shutting Down an Oracle Instance

To use the database, the instance must pass through several stages. Startup and shutdown require the SYSDBA or SYSOPER privilege.



SHUTDOWN reverses these stages: close database → dismount → stop instance

Figure 2.1: Stages of database startup

Stage	What happens	Typical use
NOMOUNT	Instance is created: parameter file is read, SGA allocated, background processes started. No database is associated yet.	Create a database; re-create control files; restore a control file.
MOUNT	Instance reads the control file and locates the datafiles and redo logs, but they are not yet opened.	Enable/disable ARCHIVELOG; full database recovery; rename or move datafiles.
OPEN	All online datafiles and redo logs are opened; users can connect and work. Instance recovery runs here if needed.	Normal operation.

```

SQL> CONNECT / AS SYSDBA
SQL> STARTUP;                                -- goes through NOMOUNT, MOUNT, OPEN
SQL> STARTUP NOMOUNT;
SQL> ALTER DATABASE MOUNT;
SQL> ALTER DATABASE OPEN;
SQL> STARTUP RESTRICT;                       -- only users with RESTRICTED SESSION
SQL> STARTUP MOUNT;
SQL> ALTER DATABASE OPEN READ ONLY;
SQL> STARTUP FORCE;                           -- SHUTDOWN ABORT followed by STARTUP

```

Listing 2.2: Startup options

2.3.1 Shutdown Modes

Mode	New connections	Existing sessions / transactions	Recovery on next start?
NORMAL	Not allowed	Waits for all users to disconnect	No
TRANSACTIONAL	Not allowed	Waits for active transactions to finish, then disconnects users	No
IMMEDIATE	Not allowed	Rolls back active transactions and disconnects users	No
ABORT	Not allowed	Instance stops at once; nothing is rolled back or checkpointed	Yes — instance recovery by SMON

SHUTDOWN IMMEDIATE is the most commonly used mode. Use **ABORT** only when other modes fail, because the next startup needs instance recovery. For a consistent (cold) backup use **NORMAL**, **TRANSACTIONAL** or **IMMEDIATE**, never **ABORT**.

Note: In a multitenant database (Unit 5), each pluggable database can also be opened and closed independently of the CDB.

2.4 Oracle Network Components and Communication Between Databases

Clients reach the database through **Oracle Net Services**. The main components are:

- **Listener** (`lsnrctl`) — a process on the server that receives connection requests and passes them to the database (configured in `listener.ora`).
- **Net service names** — aliases that a client resolves to host, port and service name (configured in `tnsnames.ora` or using Easy Connect).
- **sqlnet.ora** — client/server profile settings such as naming methods and timeouts.

Unit 6 explains these in detail. Here we see how one database communicates with another: through a **database link**.

2.4.1 Database Links

A **database link** is a schema object that stores the connection information needed to access objects in a remote database. Once it exists, remote tables are used as `table_name@link_name` in SQL.

```
CREATE DATABASE LINK remote_hr
  CONNECT TO hr IDENTIFIED BY hr_pwd
  USING 'REMOTEDB';           -- net service name of remote DB

SELECT emp_name FROM employees@remote_hr;

CREATE PUBLIC DATABASE LINK sales_link USING 'SALESDB';  -- uses current user

DROP DATABASE LINK remote_hr;
SELECT db_link, host FROM user_db_links;
```

Listing 2.3: Creating and using a database link

Note: The parameter `GLOBAL_NAMES` and the privilege `CREATE DATABASE LINK` affect how links are named and created. Storing a password in a link is a security consideration.

2.5 Using Dynamic Performance Views

Dynamic performance views (names start with `vs`) show the current state of the instance and database. They are built on internal memory structures and the control file, not on real tables, so their content changes continuously and is reset when the instance restarts. `GVs` views show the same information for all instances in a RAC cluster. The **data dictionary views** (`DBA_`, `ALL_`, `USER_`, and `CDB_` in a CDB) are different: they describe permanent metadata stored in the `SYSTEM` tablespace.

View	Information provided
V\$INSTANCE	Instance name, status (STARTED/MOUNTED/OPEN), host, startup time.
V\$DATABASE	Database name, DBID, log mode (ARCHIVELOG/NOARCHIVELOG), open mode.
V\$VERSION	Oracle release banner.
V\$SGA, V\$SGAINFO	SGA components and sizes.
V\$PARAMETER, V\$SPF_PARAMETER	Current parameter values / values stored in the SPFILE.
V\$SESSION, V\$PROCESS	Connected sessions and server/background processes.
V\$DATAFILE, V\$TABLESPACE, V\$LOGFILE, V\$CONTROLFILE	Files that make up the database.
V\$SQL, V\$SQLAREA	SQL statements in the shared pool with execution statistics.
V\$DIAG_INFO	Locations of ADR diagnostic files.

```

SELECT instance_name, status, host_name, startup_time FROM v$instance;
SELECT name, open_mode, log_mode FROM v$database;
SELECT username, status, osuser, machine, program FROM v$session
WHERE type = 'USER';
SELECT name, value FROM v$parameter WHERE name LIKE 'sga%';

```

Listing 2.4: Querying dynamic performance views

Example 2.1

Find out whether the database is in ARCHIVELOG mode and whether it is open.

Solution

```

SELECT name, log_mode, open_mode FROM v$database;

```

LOG_MODE shows ARCHIVELOG or NOARCHIVELOG, and OPEN_MODE shows READ WRITE or MOUNTED, etc.

2.6 Using the Automatic Diagnostic Repository (ADR)

The **Automatic Diagnostic Repository (ADR)** is a file-based repository stored outside the database, so it is available even when the database is down. It holds diagnostic data such as the alert log, trace files, incident dumps and health monitor reports. Its top-level location is set by the parameter `DIAGNOSTIC_DEST` (by default it is `ORACLE_BASE`). Each product instance has an **ADR home**.

```

$ORACLE_BASE/diag/rdbms/<db_unique_name>/<instance_name>/
  alert/      -- alert log in XML format
  trace/      -- text alert log (alert_<SID>.log) and trace files
  incident/   -- incident dumps for serious errors (e.g. ORA-00600)
  cdump/      -- core dumps
  hm/         -- health monitor reports

```

Listing 2.5: ADR home directory layout

Benefits of ADR: a uniform structure for all Oracle products, automatic collection of data about critical errors (**problems** and **incidents**), and the ability to package diagnostics for Oracle Support. The command-line tool **ADRCI** is used to view and manage ADR data.

```

$ adrci
adrci> SHOW HOMES
adrci> SET HOMEPATH diag/rdbms/orcl/orcl
adrci> SHOW ALERT -TAIL 50      -- last 50 lines of the alert log
adrci> SHOW INCIDENT
adrci> SHOW PROBLEM
adrci> EXIT

```

Listing 2.6: Basic ADRCI commands

2.7 Using the Alert Log and Trace Files

The **alert log** is a chronological record of important events in the instance. It is the first place a DBA looks when something goes wrong. It records instance startup and shutdown, non-default parameters, ALTER and CREATE commands issued by administrators, redo log switches, tablespace changes, deadlocks, and serious errors such as ORA-00600. The text version is named alert_<SID>.log and is stored in the **trace** directory of the ADR home.

Trace files contain detailed diagnostic output. Background processes write trace files when they detect problems; server processes write them for serious errors, and a DBA or developer may request one by enabling SQL trace for a session. Incident trace files are linked to an incident number in ADR.

```

SELECT name, value FROM v$diag_info;      -- ADR locations, alert and trace paths
ALTER SESSION SET SQL_TRACE = TRUE;      -- older method (use DBMS_MONITOR now)
EXEC DBMS_MONITOR.SESSION_TRACE_ENABLE(session_id => 45, serial_num => 1234);

```

Listing 2.7: Finding diagnostic locations and enabling trace

File	Content	Written by
Alert log	Instance-wide events and errors	Instance and background processes
Background process trace	Details when a background process fails or reports a problem	DBWn, LGWR, SMON, etc.

File	Content	Written by
Server process trace	Errors or SQL trace data for one session	Server processes
Incident dump	Data collected for one critical error occurrence	Automatically by ADR

2.8 Managing Initialization Parameter Files

Initialization parameters control how the instance behaves (memory sizes, file locations, limits). They are stored in a parameter file that is read when the instance starts (at NOMOUNT).

Feature	PFILE	SPFILE
Format	Text file (init<SID>.ora)	Binary file (spfile<SID>.ora)
Edited with	Any text editor	ALTER SYSTEM command only (do not edit by hand)
Changes while running	Not recorded; needs manual edit and restart	Can be saved persistently with SCOPE=SPFILE or BOTH
Location	Client or server (default \$ORACLE_HOME/dbs)	Server-side (default \$ORACLE_HOME/dbs)

At startup Oracle looks in the default location (\$ORACLE_HOME/dbs on Linux) in this order: spfile<SID>.ora, then spfile.ora, then init<SID>.ora. A specific file can be forced with `STARTUP PFILE='path'`.

```
CREATE SPFILE FROM PFILE='/u01/app/oracle/admin/orcl/pfile/init.ora';
CREATE PFILE='/tmp/initorcl.ora' FROM SPFILE;          -- text copy for backup/editing

SHOW PARAMETER sga
ALTER SYSTEM SET sga_target = 2G SCOPE = BOTH;        -- dynamic parameter
ALTER SYSTEM SET processes = 400 SCOPE = SPFILE;      -- static: restart required
ALTER SESSION SET nls_date_format = 'DD-MON-YYYY';    -- session-level only

SELECT name, value, isses_modifiable, issys_modifiable
FROM   v$parameter WHERE name = 'processes';
```

Listing 2.8: Working with parameter files

SCOPE value	Effect
MEMORY	Changes the running instance only; lost at restart.
SPFILE	Changes the SPFILE only; takes effect at next restart. Required for static parameters.
BOTH	Changes the running instance and the SPFILE (default when an SPFILE is in use).

Important parameters include DB_NAME, DB_BLOCK_SIZE, CONTROL_FILES, SGA_TARGET, PGA_AGGREGATE_TARGET, PROCESSES, SESSIONS, UNDO_TABLESPACE,

DB_RECOVERY_FILE_DEST, COMPATIBLE and DIAGNOSTIC_DEST. **Static** parameters (such as DB_BLOCK_SIZE, DB_NAME) cannot be changed while the instance is running. **Dynamic** parameters can be changed with ALTER SESSION or ALTER SYSTEM.

Example 2.2

The DBA changed PROCESSES with SCOPE=SPFILE. When will it take effect and why?

Solution

It takes effect only after the instance is restarted. PROCESSES is a static parameter, so it cannot be changed in memory; SCOPE=SPFILE writes it to the SPFILE for the next startup.

Key Takeaways

- ✓ Install with OUI (runInstaller) after OS preparation; create databases with DBCA or CREATE DATABASE.
- ✓ Startup stages: **NOMOUNT** → **MOUNT** → **OPEN**. Shutdown modes: NORMAL, TRANSACTIONAL, IMMEDIATE, ABORT (needs recovery).
- ✓ Listener + net service names + sqlnet.ora form Oracle Net; a **database link** lets one database query another.
- ✓ v\$ dynamic performance views show live instance data; DBA_/ALL_/USER_ views show dictionary metadata.
- ✓ **ADR** stores diagnostics outside the database; use ADRCI, the **alert log** and trace files to troubleshoot.
- ✓ **SPFILE** (binary, dynamic) is preferred over PFILE (text); use ALTER SYSTEM SET ... SCOPE=MEMORY | SPFILE | BOTH.

Reminder: these notes may contain errors — cross-check with the official syllabus, your textbook and Oracle documentation.

— End of Unit 2 —