2024

Oracle 19c Database Administration

DAY 1

Day 1 Topics

Oracle Database Architecture

Installation for Windows (Windows 8,10,11, Server2012, Server2016, Server2019, Server2022)

a. Download and install:
   https://www.oracle.com/database/technologies/oracle19c-windows-downloads.html

Installation on RHEL

a. Set static ip
b. Set hostname

```
$ sudo hostnamectl set-hostname <hostname>
$ hostname
```

c. Add ip and hostname to hosts file

```
$ sudo vim /etc/hosts
```

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
```

d. Set SELinux to permissive

```
$ getenforce                    //check selinux status
$ sudo setenforce 0
```

e. Update and install prereqs

```
$ sudo dnf update
$ sudo dnf install oracle-database-preinstall-19c
$ reboot
```

f. Download the rpm package
   https://www.oracle.com/database/technologies/oracle19c-linux-downloads.html

g. Navigate to the file and install

```
$ sudo dnf -y localinstall <filename>.rpm
$ sudo rpm -i <filename>.rpm
```

h. Check Configuration file

```
$ cat /etc/sysconfig/oracledb_ORCLCDB-19c.conf
$ sudo /etc/init.d/oracledb_ORCLCDB-19c configure          //run configuration script
```

i. After installing oracle database 19c, you will get a new user “oracle”

```
$ cat /etc/passwd               //check if oracle user exists
$ passwd oracle                 //reset oracle user password
$ su oracle                     //switch user to oracle
$ ps | grep ora                 //check if there are running processes or database instance
```

j. Configure oracle user account profile

```
$ su oracle
$ sudo vim .bash_profile
```

```
# add to the end
umask 022
export ORACLE_SID=ORCLCDB
export ORACLE_BASE=/opt/oracle
export ORACLE_HOME=/opt/oracle/product/19c/dbhome_1
export PATH=$PATH:$ORACLE_HOME/bin
```

```
$ source .bash_profile
```

k. Access database using sqlplus

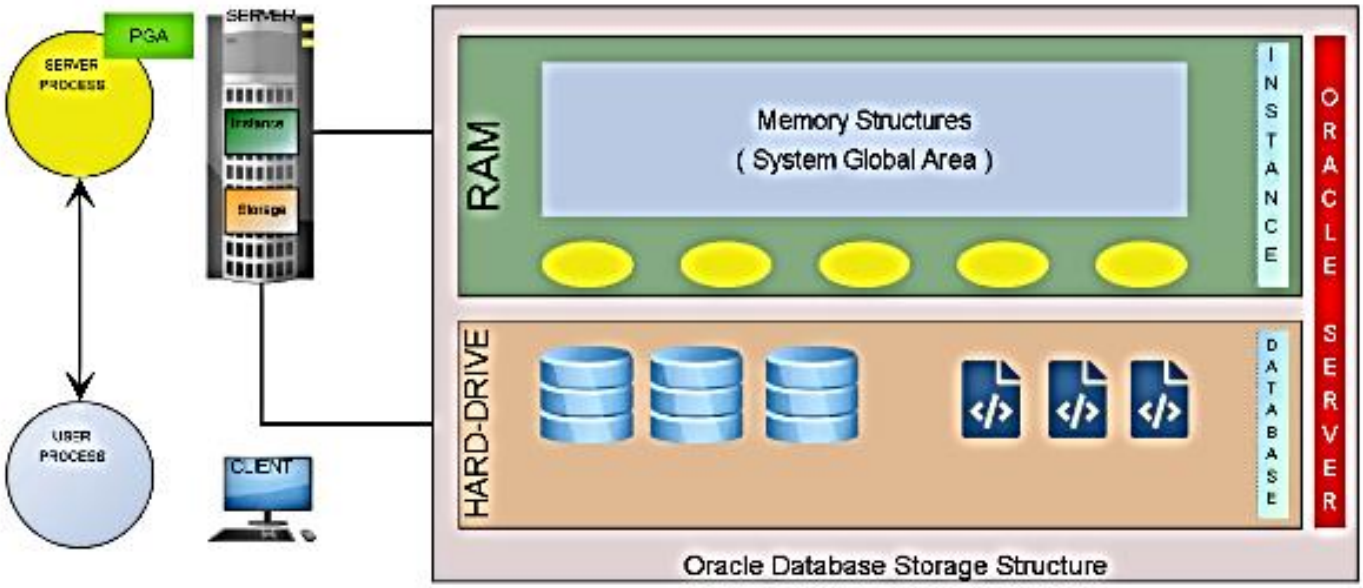
```
$ sqlplus / as sysdba
```

l. Commonly used sqlplus commands

```
sql> startup                    // start the instance
sql> shutdown immediate         // stop the instance
sql> exit                       // exit sqlplus
sql> clear screen               // clear sqlplus console
sql> set hist on                // enable command history temporarily for the session
sql> hist 2 run                 // run the 2nd command in history
sql> connect username/password@database          // connect to another database
sql> disconnect
sql> show user                  // currently connected user
sql> select * from employees    // select query to a table
sql> describe employees         // show table structure
sql> spool output.txt           // output to file
sql> spool off
sql> variable var1 number       // create and set data type
sql> print var1                 // show variable value
sql> define var1=24             // set value to variable
sql> undefine var1              // remove value to variable
sql> define var1=24             // set value to variable
sql> define var1=24             // set value to variable
sql> define var1=24             // set value to variable
sql> CONNECT sys/password@CDB_NAME AS SYSDBA       // Connect to CDB
sql> CONNECT sys/password@PDB_NAME AS SYSDBA       // Connect to PDB

// Create a PDB
sql>    CREATE PLUGGABLE DATABASE pdb_name
        ADMIN USER pdb_admin IDENTIFIED BY password
        FILE_NAME_CONVERT = ('/path/to/cdb/datafile/', '/path/to/pdb/datafile/');

sql> show pdbs                  // show all pdbs in the instance
sql> ALTER PLUGGABLE DATABASE pdb_name OPEN;                    // Open a PDB
sql> ALTER PLUGGABLE DATABASE pdb_name CLOSE IMMEDIATE;         // Close a PDB
sql> ALTER PLUGGABLE DATABASE ALL OPEN;                         // Open All PDBs
sql> ALTER PLUGGABLE DATABASE ALL CLOSE IMMEDIATE;              // Close All PDBs
sql> ALTER SESSION SET CONTAINER = pdb_name;                    // Switch to a PDB
sql> ALTER SESSION SET CONTAINER = CDB$ROOT;                    // Switch to the Root Container
sql> SELECT con_id, name, open_mode FROM v$pdbs;                // List All PDBs
sql> SHOW CON_NAME;                                             // Show Current Container
```

```
// Unplug a PDB
sql> ALTER PLUGGABLE DATABASE pdb_name CLOSE IMMEDIATE;
sql> ALTER PLUGGABLE DATABASE pdb_name UNPLUG INTO '/path/to/pdb_name.xml';
sql> DROP PLUGGABLE DATABASE pdb_name KEEP DATAFILES;       // Drop an Unplugged PDB

// Plug an Existing PDB
sql>     CREATE PLUGGABLE DATABASE pdb_name USING '/path/to/pdb_name.xml'
         FILE_NAME_CONVERT = ('/path/to/source/', '/path/to/destination/');

sql> BACKUP PLUGGABLE DATABASE pdb_name;                    // Backup a PDB
sql> RECOVER PLUGGABLE DATABASE pdb_name;                   // Recover a PDB
```

Oracle Database Server Architecture



Database Server 3 Main Components

- ✓ Database Instance in memory (the SGA and the background processes)
- ✓ Files that store the data and database information
- ✓ DB software (binary/executable files for functioning of Oracle)

After starting a database instance, the Oracle software associates the instance with a specific database. This is called *mounting the database*. The database is then ready to be opened, which makes it accessible to authorized users.

Oracle Database Instance Configurations

Configuring an Oracle Database instance involves several steps, including setting up the initialization parameters, configuring memory management, and tuning performance. Here's a comprehensive guide to Oracle Database instance configurations:

Initialization Parameters
Initialization parameters control the behavior of an Oracle Database instance. These parameters can be set in the init.ora or spfile.ora files.

File Locations:

- PFILE (init.ora): Plain text file located at $ORACLE_HOME/dbs/init<SID>.ora.
- SPFILE (spfile.ora): Binary file located at $ORACLE_HOME/database/spfile<SID>.ora.

Basic Initialization Parameters:

DB_NAME: Name of the database.

```
DB_NAME = mydb
```

DB_DOMAIN: Domain name of the database.

```
DB_DOMAIN = example.com
```

INSTANCE_NAME: Name of the instance.

```
INSTANCE_NAME = mydb1
```

CONTROL_FILES: Locations of control files.

```
CONTROL_FILES = ('/path/to/control01.ctl', '/path/to/control02.ctl')
```

Memory Management
Oracle provides two types of memory management: Automatic Memory Management (AMM) and Manual Memory Management.

Automatic Memory Management (AMM):

MEMORY_TARGET: Total amount of memory allocated to the database.

```
MEMORY_TARGET = 2G
```

MEMORY_MAX_TARGET: Maximum memory target.

```
MEMORY_MAX_TARGET = 2G
```

Manual Memory Management:

SGA_TARGET: Size of the System Global Area.

```
SGA_TARGET = 1G
```

PGA_AGGREGATE_TARGET: Size of the Program Global Area.

```
PGA_AGGREGATE_TARGET = 500M
```

Configuring Shared Pool and Buffer Cache

Shared Pool Size:

```
SHARED_POOL_SIZE = 256M
```

Database Buffer Cache Size:

```
DB_CACHE_SIZE = 512M
```

Redo Log and Archive Log

Redo Log File Size:

```
LOG_FILE_SIZE = 100M
```

Archive Log Mode:

Enable Archive Log Mode:

```
ALTER SYSTEM SET log_archive_dest_1='LOCATION=/path/to/archive/';
ALTER SYSTEM SET log_archive_format='%t_%s_%r.dbf';
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
```

Diagnostic and Tuning

Diagnostic Destination:

```
DIAGNOSTIC_DEST = /path/to/diag
```

Automatic Workload Repository (AWR):

Configure AWR snapshot interval and retention:

```
EXEC DBMS_WORKLOAD_REPOSITORY.modify_snapshot_settings(interval => 60, retention => 7*24*60);
```

Performance Tuning

Optimizer Mode:

```
OPTIMIZER_MODE = ALL_ROWS
```

Cursor Sharing:

```
CURSOR_SHARING = FORCE
```

Parallel Execution:

```
PARALLEL_MAX_SERVERS = 16
PARALLEL_MIN_SERVERS = 4
```

Undo Management

Automatic Undo Management:

```
UNDO_MANAGEMENT = AUTO
UNDO_TABLESPACE = undotbs1
```

View Parameters

```
SHOW PARAMETERS;
SHOW PARAMETERS DB_CACHE_SIZE;
```

Network Configuration

Listener Configuration (listener.ora):

```
SID_LIST_LISTENER =
 (SID_LIST =
  (SID_DESC =
   (SID_NAME = mydb)
   (ORACLE_HOME = /path/to/oracle_home)
   (GLOBAL_DBNAME = mydb.example.com)
  )
 )

LISTENER =
 (DESCRIPTION_LIST =
  (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost)(PORT = 1521))
  )
 )
```

TNS Configuration (tnsnames.ora):

(text)

```
mydb =
 (DESCRIPTION =
  (ADDRESS_LIST =
   (ADDRESS = (PROTOCOL = TCP)(HOST = myhost)(PORT = 1521))
  )
  (CONNECT_DATA =
   (SERVICE_NAME = mydb.example.com)
  )
 )
```

Backup and Recovery Configuration

Enable Flashback Database:

```
ALTER DATABASE FLASHBACK ON;
```

Configure RMAN (Recovery Manager):

```
RMAN> CONFIGURE CONTROLFILE AUTOBACKUP ON;
RMAN> CONFIGURE RETENTION POLICY TO REDUNDANCY 2;
```

User and Security Management

Create Users and Assign Roles:

```
CREATE USER myuser IDENTIFIED BY password;
GRANT CONNECT, RESOURCE TO myuser;
```

View Users

```
DESCRIBE DBA_USERS;          // see also ALL_USERS
SELECT USERNAME FROM DBA_USERS
```

Profile Management:
View user and their profile

```
SELECT USERNAME, PROFILE FROM DBA_USERS
```

Check the resource limits of a profile

```
select resource_name, limit from dba_profiles where profile='DEFAULT';
```

Create profile

```
CREATE PROFILE myprofile LIMIT
  SESSIONS_PER_USER 5
  CPU_PER_SESSION 10000
  CONNECT_TIME 60
  IDLE_TIME 10;

ALTER USER myuser PROFILE myprofile;
```

Comprehensive Example: Initialization Parameter File (init.ora)
Here's an example of a complete initialization parameter file:

```
DB_NAME = mydb
DB_DOMAIN = example.com
INSTANCE_NAME = mydb1
CONTROL_FILES = ('/path/to/control01.ctl', '/path/to/control02.ctl')
MEMORY_TARGET = 2G
MEMORY_MAX_TARGET = 2G
SHARED_POOL_SIZE = 256M
DB_CACHE_SIZE = 512M
LOG_FILE_SIZE = 100M
DIAGNOSTIC_DEST = /path/to/diag
OPTIMIZER_MODE = ALL_ROWS
CURSOR_SHARING = FORCE
PARALLEL_MAX_SERVERS = 16
PARALLEL_MIN_SERVERS = 4
UNDO_MANAGEMENT = AUTO
UNDO_TABLESPACE = undotbs1
```

Connecting to the Database Instance

Connecting to an Oracle Database instance involves understanding various methods and tools available for establishing a connection. Here's a comprehensive guide on how to connect to an Oracle Database instance using different methods:

Using SQL*Plus
SQL*Plus is a command-line tool provided by Oracle to interact with the database.

1. Open SQL*Plus:

```
sqlplus
```

2. Connect using a username and password:

```
CONNECT username/password@host:port/service_name
```

Example:

```
CONNECT scott/tiger@localhost:1521/orclpdb1
```

3. Connect using Easy Connect Naming Method:

```
CONNECT username/password@//host:port/service_name
```

Example:

```
CONNECT scott/tiger@//localhost:1521/orclpdb1
```

4. Connect using TNS alias:
Ensure your `tnsnames.ora` file is configured properly.

```
CONNECT username/password@tns_alias
```

Example:

```
CONNECT scott/tiger@orcl
```

5. Connect as SYSDBA or SYSOPER:

```
CONNECT sys/password@host:port/service_name AS SYSDBA
```

Example:

```
CONNECT sys/password@localhost:1521/orclpdb1 AS SYSDBA
```

Using Oracle SQL Developer
Oracle SQL Developer is a graphical tool for database development.
1. Open Oracle SQL Developer.

2. Create a new connection:
- Click on the New Connection button.
- Enter the connection details:
  - Connection Name: Any name for your connection.
  - Username: Your database username.
  - Password: Your database password.
  - Hostname: The database host (e.g., `localhost`).
  - Port: The database port (default is `1521`).
  - SID or Service Name: The database SID or service name.
- Click Test to verify the connection.
- Click Connect to establish the connection.

Using JDBC (Java Database Connectivity)
JDBC allows Java applications to connect to an Oracle Database.

1. Include the Oracle JDBC Driver in your project.
- Download the Oracle JDBC driver from the Oracle website and add it to your project's classpath.

2. Write Java code to connect to the database:

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class OracleJDBCExample {
    public static void main(String[] args) {
        String jdbcUrl = "jdbc:oracle:thin:@localhost:1521:orclpdb1";
        String username = "scott";
        String password = "tiger";

        try {
            Connection connection = DriverManager.getConnection(jdbcUrl, username, password);
            System.out.println("Connected to the database!");
            connection.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Using Python with cx_Oracle

cx_Oracle is a Python library that allows interaction with Oracle databases.

1. Install cx_Oracle:

```
pip install cx_Oracle
```

2. Write Python code to connect to the database:

```
import cx_Oracle

dsn = cx_Oracle.makedsn("localhost", 1521, service_name="orclpdb1")
connection = cx_Oracle.connect(user="scott", password="tiger", dsn=dsn)

print("Connected to the database!")
connection.close()
```

Using Oracle Net Services (NetCA and Net Manager)

Oracle Net Services configuration involves setting up the listener and configuring client-side connections.

1. Configure the Listener (listener.ora):

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = orcl)
      (ORACLE_HOME = /u01/app/oracle/product/19.0.0/dbhome_1)
    )
  )

LISTENER =
```

```
  (DESCRIPTION_LIST =
   (DESCRIPTION =
    (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
   )
  )
```

2. Configure Client-Side Connections (tnsnames.ora):

```
 ORCL =
  (DESCRIPTION =
   (ADDRESS_LIST =
    (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
   )
   (CONNECT_DATA =
    (SERVICE_NAME = orcl)
   )
  )
```

3. Start the Listener:

```
 lsnrctl start
```

4. Connect using SQL*Plus with TNS alias:

```
 CONNECT scott/tiger@ORCL
```

Using Oracle Data Pump for Remote Connections

Oracle Data Pump allows importing and exporting data remotely.

1. Export Data:

```
 expdp username/password@//localhost:1521/orclpdb1 DIRECTORY=exp_dir DUMPFILE=expdp.dmp
LOGFILE=expdp.log SCHEMAS=schema_name
```

2. Import Data:

```
 impdp username/password@//localhost:1521/orclpdb1 DIRECTORY=imp_dir DUMPFILE=expdp.dmp
LOGFILE=impdp.log SCHEMAS=schema_name
```

Oracle Database Memory Structures

Oracle 19c, like its predecessors, utilizes a sophisticated memory architecture designed to manage and process data efficiently. Understanding these memory structures is crucial for database administrators to optimize performance and ensure stability. Here's a comprehensive guide to the memory structures in Oracle 19c:

Overview of Oracle Memory Structures

Oracle Database uses two main memory structures:

- ✓ System Global Area (SGA)
- ✓ Program Global Area (PGA)

System Global Area (SGA)
The SGA is a shared memory region that contains data and control information for one Oracle Database instance. It is allocated when the database instance starts and deallocated when the instance shuts down. The SGA consists of several components:

Key Components of the SGA:

- Database Buffer Cache
- Shared Pool
- Redo Log Buffer
- Large Pool
- Java Pool
- Streams Pool
- Data Dictionary Cache
- Result Cache
- In-Memory Column Store (optional)

1. Database Buffer Cache
    - Purpose: Stores copies of data blocks read from data files. Frequently accessed data blocks are cached here to improve performance.
    - Tuning Parameter: `DB_CACHE_SIZE`

2. Shared Pool
    - Purpose: Caches various types of information that can be shared among users, such as SQL statements, PL/SQL code, and data dictionary information.
    - Components:
        - Library Cache: Stores executable forms of SQL and PL/SQL.
        - Data Dictionary Cache: Stores metadata about database objects.
    - Tuning Parameter: `SHARED_POOL_SIZE`

3. Redo Log Buffer
    - Purpose: Temporarily stores redo entries (change vectors) before they are written to the redo log files. Ensures recovery of committed transactions.
    - Tuning Parameter: `LOG_BUFFER`

4. Large Pool
    - Purpose: Provides memory for large memory allocations, such as Oracle backup and recovery operations, and I/O server processes.
    - Tuning Parameter: `LARGE_POOL_SIZE`

5. Java Pool
    - Purpose: Used for all session-specific Java code and data within the JVM.
    - Tuning Parameter: `JAVA_POOL_SIZE`

6. Streams Pool
    - Purpose: Used by Oracle Streams to store information related to capture and apply processes.
    - Tuning Parameter: `STREAMS_POOL_SIZE`

7. Data Dictionary Cache
    - Purpose: Part of the shared pool; stores metadata about database objects.
    - Tuning Parameter: Managed within `SHARED_POOL_SIZE`

8. Result Cache
    - Purpose: Caches the results of SQL queries and PL/SQL functions for faster retrieval.
    - Tuning Parameter: `RESULT_CACHE_MAX_SIZE`

9. In-Memory Column Store (optional)
    - Purpose: Stores a copy of table data in a columnar format for analytic query performance improvements.
    - Tuning Parameter: `INMEMORY_SIZE`

Program Global Area (PGA)
The PGA is a memory region that contains data and control information for a single server process or background process. Unlike the SGA, the PGA is not shared and is allocated when a process is started.

Key Components of the PGA:
- Session Memory
- Private SQL Area
- Sort Area
- Hash Area
- Bitmap Merge Area

1. Session Memory
    - Purpose: Contains session variables, logon information, and other session-specific information.

2. Private SQL Area
    - Purpose: Contains data such as bind variable values and runtime buffers. Each session that issues a SQL statement has a private SQL area.

3. Sort Area
    - Purpose: Used for sort operations such as ORDER BY, GROUP BY, and creating indexes.

4. Hash Area
    - Purpose: Used for hash join operations.

5. Bitmap Merge Area
    - Purpose: Used for bitmap index merge operations.

Automatic Memory Management (AMM)
Oracle can automatically manage the sizes of the SGA and PGA memory areas using Automatic Memory Management (AMM).

Parameters for AMM:
1. MEMORY_TARGET: Total memory allocated to Oracle for both SGA and PGA.
2. MEMORY_MAX_TARGET: Maximum size to which MEMORY_TARGET can be set dynamically.

Example Configuration:

```
ALTER SYSTEM SET MEMORY_TARGET=4G SCOPE=BOTH;
ALTER SYSTEM SET MEMORY_MAX_TARGET=4G SCOPE=BOTH;
```

Automatic Shared Memory Management (ASMM)

If you prefer more control over the SGA, you can use Automatic Shared Memory Management (ASMM). This allows Oracle to automatically adjust the sizes of the various SGA components based on workload.

Parameters for ASMM:

1. SGA_TARGET: Total size of all SGA components.
2. SGA_MAX_SIZE: Maximum size to which SGA_TARGET can be set dynamically.

Example Configuration:

```
ALTER SYSTEM SET SGA_TARGET=2G SCOPE=BOTH;
ALTER SYSTEM SET SGA_MAX_SIZE=2G SCOPE=BOTH;
```

Manual Memory Management

If you opt for manual memory management, you will need to set the sizes for each individual component of the SGA and PGA.

Example Configuration:

```
ALTER SYSTEM SET DB_CACHE_SIZE=1G SCOPE=BOTH;
ALTER SYSTEM SET SHARED_POOL_SIZE=500M SCOPE=BOTH;
ALTER SYSTEM SET LARGE_POOL_SIZE=128M SCOPE=BOTH;
ALTER SYSTEM SET JAVA_POOL_SIZE=64M SCOPE=BOTH;
ALTER SYSTEM SET STREAMS_POOL_SIZE=64M SCOPE=BOTH;
ALTER SYSTEM SET PGA_AGGREGATE_TARGET=1G SCOPE=BOTH;
```

Memory Advisors

Oracle provides several memory advisors to help tune the memory components:

1. SGA Target Advisor
2. PGA Target Advisor

These advisors provide recommendations based on the workload and performance metrics.

Example of Using SQL*Plus for Memory Advisors

PGA Advisor Example:

```
SELECT
  round(pga_target_for_estimate/1024/1024) as target_mb,
  estd_pga_cache_hit_percentage as cache_hit_perc,
  estd_overalloc_count as over_alloc_count
FROM
  v$pga_target_advice;
```

SGA Advisor Example:

```
SELECT
  sga_size,
  estd_physical_reads AS physical_reads
FROM
  v$sga_target_advice;
```

Memory Advisor Example:

```
SELECT
  memory_size_for_estimate,
  estd_db_time,
  estd_physical_read_time,
  estd_physical_write_time
FROM
  v$memory_target_advice
ORDER BY
  memory_size_for_estimate;
```

Using Oracle Enterprise Manager (OEM)
Oracle Enterprise Manager provides a graphical interface for accessing memory advisors.

1) Log in to OEM:
    a. Connect to Oracle Enterprise Manager.
    b. Navigate to Memory Advisors:

2) Go to the Performance tab.
    a. Click on Advisors Central.
    b. Select Memory Advisors.

3) View Recommendations:
    a. Select the desired advisor (PGA, SGA, Memory).
    b. View the recommendations and implement suggested changes.

Creating DBCS Database Deployments

Automated Database Provisioning and Creating a Database Deployment

In Oracle 19c, automated database provisioning can be performed using Oracle's Database Creation Assistant (DBCA) in silent mode or using Oracle Enterprise Manager Cloud Control. Here are the steps to perform automated database provisioning using both methods:

Using DBCA in Silent Mode
Prepare the Response File: Create a response file with all the necessary parameters for the database creation. This file is a plain text file that contains key-value pairs.

Example of a response file (dbca.rsp):

```
[GENERAL]
RESPONSEFILE_VERSION = "19.0.0"
```

```
OPERATION_TYPE = "createDatabase"

[CREATEDATABASE]
GDBNAME = "orcl"
SID = "orcl"
TEMPLATE_NAME = "General_Purpose.dbc"
CHARACTERSET = "AL32UTF8"
NATIONALCHARACTERSET = "AL16UTF16"
TOTALMEMORY = "2048"
SYSPASSWORD = "sys_password"
SYSTEMPASSWORD = "system_password"
```

Run DBCA in Silent Mode: Use the dbca command with the -silent flag and the response file to create the database.

```
dbca -silent -responseFile /path/to/dbca.rsp
```

Using Oracle Enterprise Manager Cloud Control

Oracle Enterprise Manager (OEM) Cloud Control provides a more graphical and centralized approach for database provisioning. Here's how you can automate provisioning using OEM:

1. Create a Provisioning Profile:
    - Navigate to the Enterprise menu, select Provisioning and Patching, then Database Provisioning.
    - Create a new profile by capturing an existing database configuration or using a template.

2. Create a Deployment Procedure:
    - Go to the Deployments tab and select Database Provisioning.
    - Create a new deployment procedure using the profile you created.

3. Schedule the Provisioning Job:
    - Schedule the job to run immediately or at a specified time.
    - You can also set up recurring schedules for regular provisioning tasks.

4. Monitor and Manage:
    - Use the OEM interface to monitor the status of your provisioning jobs.
    - Manage and adjust the procedures as needed based on your requirements.

Example Response File for Automated Database Creation

Here's a more detailed example of a response file (dbca_response.rsp) for DBCA:

```
[GENERAL]
RESPONSEFILE_VERSION = "19.0.0"
OPERATION_TYPE = "createDatabase"

[CREATEDATABASE]
GDBNAME = "orcl.example.com"
SID = "orcl"
TEMPLATENAME = "General_Purpose.dbc"
SYSPASSWORD = "Welcome1"
SYSTEMPASSWORD = "Welcome1"
```

```
DATAFILEDESTINATION = "/u01/app/oracle/oradata"
RECOVERYAREADESTINATION = "/u01/app/oracle/flash_recovery_area"
STORAGETYPE = "FS"
CHARACTERSET = "AL32UTF8"
NATIONALCHARACTERSET= "AL16UTF16"
TOTALMEMORY = "4096"
```

Running the DBCA Command
Once you have your response file prepared, you can run the following command to initiate the automated provisioning:

```
dbca -silent -responseFile /path/to/dbca_response.rsp
```

This approach allows for full automation of the database creation process, making it suitable for environments where rapid provisioning is required.

Accessing an Oracle Database

Connecting to an Oracle Database Instance

In Oracle 19.3c, there are several ways to connect to an Oracle database instance. These methods can be used based on the specific requirements and environments. Here are the common methods:

1. SQL*Plus
SQL*Plus is a command-line tool that comes with Oracle Database for running SQL and PL/SQL commands.

Connecting using SQL*Plus

```
sqlplus username/password@hostname:port/SID
```

Example:

```
sqlplus system/Welcome1@localhost:1521/orcl
```

2. Oracle SQL Developer
Oracle SQL Developer is a free integrated development environment that simplifies the management of Oracle databases in both traditional and Cloud deployments.

Connecting using Oracle SQL Developer

1) Open SQL Developer.
2) Click on the "New Connection" button.
3) Enter the connection details such as:
- Connection Name
- Username
- Password
- Hostname
- Port
- SID or Service Name

4) Click "Connect."

3. JDBC
Java applications connect to Oracle Database using JDBC (Java Database Connectivity).

Example JDBC Connection

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class OracleJDBCExample {
   public static void main(String[] argv) {
      try {
         Class.forName("oracle.jdbc.driver.OracleDriver");

         Connection connection = DriverManager.getConnection(
            "jdbc:oracle:thin:@localhost:1521:orcl", "username", "password");

         System.out.println("Connected to Oracle database!");

      } catch (ClassNotFoundException | SQLException e) {
         e.printStackTrace();
      }
   }
}
```

4. ODBC
ODBC (Open Database Connectivity) allows C and C++ applications to connect to Oracle Database.

ODBC Connection String

```
Driver={Oracle in OraHome19};Dbq=localhost:1521/orcl;Uid=username;Pwd=password;
```

5. Oracle Net Services (TNS)
Oracle Net Services provide enterprise-wide connectivity solutions in distributed, heterogeneous computing environments.

TNS Names Configuration
In the tnsnames.ora file:

```
ORCL =
 (DESCRIPTION =
   (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)(PORT = 1521))
   (CONNECT_DATA =
     (SERVER = DEDICATED)
     (SERVICE_NAME = orcl)
   )
 )
```

Connecting using TNS

```
sqlplus username/password@ORCL
```

6. Python with cx_Oracle
cx_Oracle is a Python extension module that enables access to Oracle Database.

Example Python Connection using cx_Oracle

```
import cx_Oracle
connection = cx_Oracle.connect("username/password@localhost:1521/orcl")

print("Connected to Oracle Database")
```

Oracle Database Tools

Oracle 19.3c offers a wide range of tools for managing, developing, and administering Oracle databases. Here are some of the primary tools available:

1. SQL*Plus
   - Description: A command-line tool used to run SQL and PL/SQL commands, manage database objects, and execute scripts.
   - Usage: Ideal for quick, manual interactions with the database.

2. Oracle SQL Developer
   - Description: A free graphical integrated development environment (IDE) for database development and management.
   - Features: Supports running SQL queries, designing database schemas, managing database objects, and developing PL/SQL code.

3. Oracle Enterprise Manager (OEM) Cloud Control
   - Description: A comprehensive web-based management tool for monitoring and managing Oracle databases and other Oracle products.
   - Features: Performance monitoring, alerting, automated maintenance, and provisioning.

4. Database Configuration Assistant (DBCA)
   - Description: A graphical tool to create, configure, and delete Oracle databases.
   - Features: Supports silent mode operations for automated database creation.

5. Oracle Data Pump
   - Description: A utility for fast data movement between Oracle databases.
   - Components: expdp for export, impdp for import.
   - Features: High-speed data transfer, parallel execution, and advanced filtering.

6. Oracle Recovery Manager (RMAN)
   - Description: A command-line tool for backup and recovery operations.
   - Features: Full and incremental backups, point-in-time recovery, and integration with Oracle Enterprise Manager.

7. Oracle Universal Installer (OUI)
   - Description: A tool for installing Oracle software.
   - Features: GUI-based and silent mode installation options.

8. Oracle Net Manager
   - Description: A tool to configure Oracle Net Services.
   - Features: Manage listener configurations, naming methods, and directory usage.

9. Oracle Enterprise Manager Database Express (EM Express)
   - Description: A web-based management tool included with Oracle Database 19c.
   - Features: Lightweight, for database performance management, and simple administrative tasks.

10. Oracle SQL*Loader
    - Description: A tool to load data from external files into Oracle Database tables.
    - Features: Supports various data formats, direct path loads, and efficient data processing.

11. Oracle APEX (Application Express)
    - Description: A low-code development platform for building web applications.
    - Features: Integrated development environment, SQL Workshop, and user interface design tools.

12. Oracle Listener Control Utility (LSNRCTL)
    - Description: A command-line utility to manage Oracle Listeners.
    - Features: Start, stop, and manage database listener processes.

13. Oracle Wallet Manager
    - Description: A tool to manage security credentials on Oracle databases.
    - Features: Create, manage, and store encryption keys and certificates.

14. Oracle Clusterware and Oracle Grid Infrastructure
    - Description: Tools and utilities for managing Oracle Real Application Clusters (RAC) and Oracle Automatic Storage Management (ASM).
    - Features: High availability, resource management, and cluster configuration.

15. Oracle Data Guard Broker
    - Description: A management and automation tool for Oracle Data Guard configurations.
    - Features: Simplified setup, management, and monitoring of Data Guard environments.

16. Oracle GoldenGate
    - Description: A comprehensive software package for real-time data integration and replication.
    - Features: Data capture, transformation, and delivery with high availability.

17. Oracle OLAP (Online Analytical Processing)
    - Description: Tools for multidimensional analysis within Oracle databases.
    - Features: Cube building, advanced analytics, and data warehousing capabilities.

SQL*Plus

Connecting to SQL*Plus

1. Open a terminal or command prompt.
2. Run the sqlplus command with your username and password.

```
sqlplus username/password@hostname:port/SID
```

Example:

```
sqlplus system/Welcome1@localhost:1521/orcl
```

Basic Commands

- Connect to Database:

```
CONNECT username/password@hostname:port/SID;
```

- Run SQL Commands:

```
SELECT * FROM employees;
```

- Execute Scripts:

```
@path/to/script.sql
```

- Exit:

```
EXIT;
```

Oracle SQL Developer

Download and Installation

1) Download SQL Developer from the Oracle website.
2) Extract the downloaded file and run sqldeveloper.exe (on Windows) or the corresponding executable on other OS.

Connecting to a Database

1) Open SQL Developer.
2) Click the "New Connection" button.
3) Enter the connection details:
    a. Connection Name: A name for your connection.
    b. Username: Your database username.
    c. Password: Your database password.
    d. Hostname: The hostname of your database server.
    e. Port: The port number (default is 1521).
    f. SID/Service Name: The SID or Service Name of your database.
4) Click "Connect".

Running Queries

1) Open a new SQL Worksheet.
2) Type your SQL query.
3) Click the "Run" button (or press F5).

SQL Developer Command Line (SQLcl)

Download and Installation

1) Download SQLcl from the Oracle website https://www.oracle.com/database/sqldeveloper/technologies/sqlcl/download/
2) Unzip the downloaded file.

3) Add the SQLcl directory to your system's PATH.

Connecting to a Database

1) Open a terminal or command prompt.
2) Run the sql command with your connection details:

```
sql username/password@hostname:port/SID
```

Example:

```
sql system/Welcome1@localhost:1521/orcl
```

Basic Commands

Run SQL Commands:

```
SELECT * FROM employees;
```

Execute Scripts:

```
@path/to/script.sql
```

Exit:

```
EXIT;
```

Database Configuration Assistant (DBCA)

Running DBCA

1) Open a terminal or command prompt.
2) Run the dbca command.

```
dbca
```

Creating a Database

1) Choose "Create a database".
2) Follow the prompts to specify your database settings:
   - ✓ Database Name: Enter the Global Database Name.
   - ✓ Database Configuration: Choose a template or customize the database settings.
   - ✓ Management Options: Configure Enterprise Manager settings.
   - ✓ Database Credentials: Set passwords for administrative accounts.
   - ✓ Storage Options: Specify storage locations and options.
   - ✓ Initialization Parameters: Set memory, character set, and other parameters.
   - ✓ Database Creation: Review and create the database.

Oracle Enterprise Manager Database Express

Accessing OEM Database Express

1) Ensure the database is running.
2) Open a web browser and go to the URL:

```
https://hostname:5500/em
```

Logging In

1) Enter the administrative username (e.g., SYS) and password.
2) Select the "Login" button.

Basic Operations

- ✓ Monitor Performance: Navigate to the "Performance" tab to monitor database performance.
- ✓ Manage Storage: Use the "Storage" tab to manage tablespaces, data files, and redo logs.
- ✓ User Management: Create and manage database users under the "Security" tab.
- ✓ Running SQL Queries: Use the "SQL" tab to run SQL statements.

Note: Oracle Enterprise Manager Express 19c missing menus

Missing menus like Configuration, Storage, Tablespaces, users etc. This is the default behavior now! Since flash become obsolete slowly, Oracle leaves old EM Express flash version and comes up with Java Jet and default is Java Jet anymore. This initial version doesn't have many menus that flash version has. Even so if you need to use flash version you can convert it back and convert again Java Jet version:

To use flash version of EM express, run the script below using sqlplus with sys user:

```
Sqlplus / as sysdba
SYS@orcl-1>@?/rdbms/admin/execemx emx
```

and convert it back to Java Jet:

```
Sqlplus / as sysdba
SYS@orcl-1>@?/rdbms/admin/execemx omx
```

Enterprise Manager Cloud Control

Installation and Setup

1) Download OEM Cloud Control:
2) Obtain the latest version of OEM Cloud Control from the Oracle website.
3) Follow the installation guide provided by Oracle to install OEM on your server. The installation typically involves setting up the Oracle Management Service (OMS) and Oracle Management Repository (OMR).

Notes:

- ✓ Must have 10GB RAM or higher
- ✓ Must be binded to PDB (not CDB)
- ✓ Disable password verification

  Connect to SQL*Plus:

  ```
  sqlplus / as sysdba
  ```

  Disable the Password Verification Function:

  ```
  ALTER PROFILE DEFAULT LIMIT PASSWORD_VERIFY_FUNCTION NULL;
  ```

  Verify the Change:

  ```
  SELECT PROFILE, RESOURCE_NAME, LIMIT
  FROM DBA_PROFILES
  WHERE RESOURCE_NAME = 'PASSWORD_VERIFY_FUNCTION';
  ```

- ✓ set the _allow_insert_with_update_check parameter to TRUE, you can use the ALTER SYSTEM command. Note that changing hidden parameters should be done with caution and typically under the guidance of Oracle support.

  Check parameter:

```
SELECT name, value, ISDEFAULT
FROM v$parameter
WHERE name = '_allow_insert_with_update_check';
```

  Set the Parameter:

```
ALTER SYSTEM SET "_allow_insert_with_update_check"=TRUE SCOPE=SPFILE;
```

  Restart the Database:

```
SHUTDOWN IMMEDIATE;
STARTUP;
```

- ✓ the PDB must be mounted and open (read/write)

Post-Installation Configuration:

1) Access the OEM console via a web browser. The URL is typically in the format:

```
https://<hostname>:<port>/em
```

2) Log in with the SYSMAN account or another administrative account.

Adding Targets

To manage databases and other targets, you need to add them to OEM Cloud Control.

1) Navigate to the Setup Menu:
   a. Click on Setup in the top right corner.
   b. Select Add Target > Auto Discovery Results.

2) Configure Auto Discovery:
   a. Run the auto discovery process to find databases and other targets on the network.
   b. Alternatively, you can manually add targets by selecting Add Target Manually.

3) Add a Database:
   a. Choose the option to add a database.
   b. Enter the connection details for the database (hostname, port, SID/Service Name, and credentials).
   c. Follow the prompts to complete the registration.

Managing Databases

1) Navigate to the Database Home Page:
   a. From the main dashboard, click on the Targets menu.
   b. Select Databases to view a list of managed databases.
   c. Click on the database you want to manage to open its home page.

2) Monitor Database Performance:
    a. Use the Performance menu to view real-time and historical performance metrics.
    b. Monitor CPU usage, memory usage, I/O activity, and more.

3) Manage Storage:
    a. Navigate to the Storage menu to manage tablespaces, data files, and ASM (if applicable).
    b. Perform tasks such as adding or resizing data files.

4) User and Security Management:
    a. Use the Security menu to manage users, roles, and privileges.
    b. Implement security policies and audit settings.

5) Backup and Recovery:
    a. Navigate to the Availability menu.
    b. Configure and manage RMAN backups, perform restore operations, and manage Data Guard configurations.

Automated Tasks and Alerts

1) Create and Schedule Jobs:
    a. From the main menu, navigate to Enterprise > Job > Library.
    b. Create new jobs or use predefined job templates.
    c. Schedule jobs to run at specified intervals.

2) Set Up Alerts:
    a. Navigate to Setup > Incidents > Incident Rules.
    b. Define rules to generate alerts based on specific conditions (e.g., performance thresholds, availability issues).
    c. Configure notifications to be sent via email or other channels.

Reporting

1) Generate Reports:
    a. Navigate to Reports > Reports Library.
    b. Use predefined reports or create custom reports based on your needs.
    c. Schedule reports to be generated and sent via email.

Patching and Provisioning

1) Patching:
    a. Navigate to Enterprise > Provisioning and Patching > Database Provisioning.
    b. Use OEM to automate the patching process for databases and other targets.

2) Provisioning:
    a. Use the Provisioning menu to clone databases, set up test environments, and deploy new database instances.
    b. Using the Command Line Interface (EMCLI)

OEM Cloud Control also provides a command-line interface (EMCLI) for scripting and automation.

1) Install EMCLI:
    a. Download and install the EMCLI client from the OEM console.
    b. Follow the setup instructions provided by Oracle.

2) Using EMCLI:
   a. Authenticate using your OEM credentials:

```
emcli login -username=SYSMAN
```

   b. Run EMCLI commands to manage targets, jobs, and other aspects of your environment. For example:

```
emcli add_target -name="mydb" -type="oracle_database" -
properties="host=myhost,port=1521,sid=mydb"
```

Managing DBCS Database Deployments

Managing the Compute Node

Connecting to the Compute Node

SSH Access:

- Obtain the SSH private key that matches the public key specified when the database deployment was created.
- Use an SSH client to connect to the compute node. The default username is opc.

```
ssh -i /path/to/private-key opc@compute-node-ip
```

Switch to the Oracle User:

- After connecting as opc, switch to the oracle user to manage the database.

```
sudo su - oracle
```

Monitoring and Managing Performance

1) Check System Resources:
   - Use standard Linux commands to monitor CPU, memory, and disk usage.

```
top
vmstat
iostat
df -h
```

2) Oracle Database Performance:

- Use Oracle tools like SQL*Plus or SQL Developer to monitor database performance.
- Connect to the database using sqlplus and run performance queries.

```
sqlplus / as sysdba
SELECT * FROM v$instance;
SELECT * FROM v$session;
```

3) Enterprise Manager:

- If Enterprise Manager is set up, use it to monitor database performance and system metrics.

Managing Storage

1) Check Disk Usage:

- Monitor the disk usage of the file systems.

```
df -h
```

2) Manage ASM Storage:

- If using Oracle ASM, use asmcmd or SQL commands to manage storage.

```
asmcmd lsdg
sqlplus / as sysasm
SELECT name, total_mb, free_mb FROM v$asm_diskgroup;
```

3) Add Storage:

- Use the Oracle Cloud Infrastructure console to add storage to your compute node if needed.

Patching and Updates

1) Apply Patches:

- Use the Oracle Cloud Infrastructure console to apply patches to your compute node and database.

2) Update Operating System Packages:

- Regularly update the OS packages to ensure security and stability.

```
sudo yum update -y
```

Backup and Recovery

1) Configure Backups:

- Ensure that backups are configured and running correctly. Use the Oracle Cloud Infrastructure console or RMAN.

2) Perform Manual Backups:

- Run RMAN commands to perform manual backups if needed.

```
rman target /
BACKUP DATABASE;
```

3) Restore and Recover:

- Use RMAN to restore and recover the database in case of failure.

```
rman target /
RESTORE DATABASE;
RECOVER DATABASE;
```

Security Management

1) Manage SSH Keys:

- Add or remove SSH keys as needed for secure access to the compute node.

```
vi ~/.ssh/authorized_keys
```

2) Firewall and Network Security:

- Use the Oracle Cloud Infrastructure console to manage security lists and firewall rules.

Automation and Scripting

1) Create Scripts for Routine Tasks:

- Write shell scripts or use cron jobs to automate routine maintenance tasks.

Example of a simple backup script:

```
#!/bin/bash
rman target / <<EOF
BACKUP DATABASE;
EXIT;
EOF
```

2) Schedule Jobs:

- Use cron to schedule scripts.

```
crontab -e
```

3) Add a cron job to run the backup script daily at 2 AM:

```
0 2 * * * /path/to/backup-script.sh
```

Scaling a Database Deployment

Scaling an Oracle Database deployment in the Oracle Cloud Infrastructure (OCI) involves adjusting the resources allocated to the database instance, such as increasing CPU, memory, or storage.

1. Scaling Compute Resources

Using the Oracle Cloud Infrastructure Console

a) Log in to the Oracle Cloud Infrastructure Console:

- Navigate to the Oracle Cloud Infrastructure console at https://cloud.oracle.com.

b) Go to the Database Section:

- In the navigation menu, select Oracle Database > Bare Metal, VM, and Exadata.

c) Select Your Database Deployment:

- Click on the database deployment you want to scale.

d) Edit Compute Shape:

- On the database details page, click on the Actions menu (three dots) in the upper right corner.
- Select Update from the drop-down menu.
- Under the Compute section, choose a new shape with more OCPUs and memory.
- Click Save Changes.

e) Confirm Changes:

- Review the changes and confirm. The database service will handle the scaling operation. This may involve a brief downtime depending on the type of scaling operation.

2. Scaling Storage

Using the Oracle Cloud Infrastructure Console

a) Navigate to the Database Section:

- In the OCI console, go to Oracle Database > Bare Metal, VM, and Exadata.

b) Select Your Database Deployment:
- Click on the database deployment you want to scale.

c) Add Storage:
- On the database details page, click on the Actions menu.
- Select Update Storage from the drop-down menu.
- Enter the additional storage amount you need and click Update Storage.

3. Scaling Database with Oracle Autonomous Database
If you're using Oracle Autonomous Database, the scaling process is simplified and can be done without downtime.

Using the Oracle Cloud Infrastructure Console
a) Navigate to Autonomous Database:
- In the OCI console, go to Oracle Database > Autonomous Database.

b) Select Your Autonomous Database:
- Click on the autonomous database you want to scale.

c) Scale Up/Down Resources:
- On the autonomous database details page, click Scale Up/Down.
- Adjust the number of OCPUs and the storage as needed.
- Click Save Changes.

4. Using OCI Command Line Interface (CLI)
You can also use the OCI CLI to scale your database deployment.

a) Install and Configure OCI CLI:
- If you haven't already, install and configure the OCI CLI following the instructions in the OCI CLI documentation.

b) Scale Compute Resources:
- Use the following command to change the shape of your database deployment.

```
oci db system update --db-system-id <DB_SYSTEM_OCID> --shape <NEW_SHAPE>
```

c) Scale Storage:
- Use the following command to increase the storage.

```
oci db system update --db-system-id <DB_SYSTEM_OCID> --data-storage-size-in-gbs <NEW_SIZE_IN_GB>
```

Patching DBCS

Pre-requisites
- ✓ Backup: Always take a full backup of your database before applying any patches.
- ✓ Review: Check the patch documentation for any prerequisites and the impact of the patch.

Using Oracle Cloud Infrastructure (OCI) Console
1) Log in to the Oracle Cloud Infrastructure Console:
- Navigate to the Oracle Cloud Infrastructure console at https://cloud.oracle.com.

2) Go to the Database Section:
- In the navigation menu, select Oracle Database > Bare Metal, VM, and Exadata.

3) Select Your Database Deployment:
- Click on the database deployment you want to patch.

4) Manage Patches:
- On the database details page, click on the Actions menu (three dots) in the upper right corner.
- Select Manage Patches.

5) Check for Available Patches:
- The system will list available patches for your database deployment.
- Review the list of available patches and select the appropriate one.

6) Apply Patch:
- Click Apply to start the patching process.
- Confirm the action and wait for the patching process to complete. This may involve downtime depending on the type of patch being applied.

Using Oracle Cloud Infrastructure (OCI) Command Line Interface (CLI)

1) Install and Configure OCI CLI:
- If you haven't already, install and configure the OCI CLI by following the instructions in the OCI CLI documentation.

2) List Available Patches:
- Use the following command to list available patches for your database system:

```
oci db patch list --db-system-id <DB_SYSTEM_OCID>
```

3) Apply Patch:
- Use the following command to apply a patch to your database system:

```
oci db patch apply --db-system-id <DB_SYSTEM_OCID> --patch-id <PATCH_ID>
```

- Replace <DB_SYSTEM_OCID> with the OCID of your database system and <PATCH_ID> with the ID of the patch you want to apply.

Post-Patching Steps

1) Verify the Patch:
- After the patching process is complete, verify that the patch has been successfully applied.

```
sqlplus / as sysdba
SELECT * FROM dba_registry_history ORDER BY action_time;
```

2) Test the Database:
- Perform any necessary testing to ensure that the database is functioning correctly after the patch.

3) Update Documentation:
- Document the patching process, including any issues encountered and how they were resolved.

Managing Database Instances

Working with Initialization Parameters

Oracle uses a parameter file to configure instance settings. There are two types of parameter files:

- SPFILE (Server Parameter File): Binary file that can be modified dynamically.
- PFILE (Parameter File): Text file that needs to be edited manually.

Viewing and Modifying Parameters

To view parameters:

```
SHOW PARAMETER <parameter_name>;
```

To modify parameters using SPFILE:

```
ALTER SYSTEM SET <parameter_name>=<value> SCOPE=SPFILE;
```

To modify parameters using PFILE, you need to edit the file manually and restart the database.

Examples:

Modifying Parameters Using SPFILE

The SPFILE allows you to dynamically change parameters without restarting the database. Here's how you can modify parameters using the SPFILE:

Check Current Value of a Parameter:

```
SHOW PARAMETER db_cache_size;
```

Modify the Parameter Value:

```
ALTER SYSTEM SET db_cache_size=500M SCOPE=SPFILE;
```

Verify the Change:

```
SHOW PARAMETER db_cache_size;
```

Example:

```
-- Connect to SQL*Plus as SYSDBA
sqlplus / as sysdba

-- Check the current value of db_cache_size
SHOW PARAMETER db_cache_size;

-- Modify the parameter value
ALTER SYSTEM SET db_cache_size=500M SCOPE=SPFILE;

-- Verify the change
SHOW PARAMETER db_cache_size;
```

Modifying Parameters Using PFILE

The PFILE is a text file, and changes to it require a restart of the database. Here's how to modify parameters using the PFILE:

1) Locate the PFILE:
   The PFILE is usually located in the $ORACLE_HOME/dbs directory and named init<DB_NAME>.ora.

2) Edit the PFILE:
   Open the PFILE in a text editor and modify the desired parameter.

```
db_cache_size=500M
```

3) Restart the Database:

   a. Shutdown the Database:
   sqlplus / as sysdba

```
SHUTDOWN IMMEDIATE;
```

   b. Start the Database with the PFILE:

```
STARTUP PFILE='$ORACLE_HOME/dbs/init<DB_NAME>.ora';
```

Example:

Edit the PFILE:
-- Open the PFILE (initORCL.ora) and modify the parameter
db_cache_size=500M

Restart the Database:
-- Connect to SQL*Plus as SYSDBA
sqlplus / as sysdba

-- Shutdown the database
SHUTDOWN IMMEDIATE;

-- Start the database with the modified PFILE
STARTUP PFILE='$ORACLE_HOME/dbs/initORCL.ora';

Starting the Oracle Database Instance

To start an Oracle instance, you typically use SQL*Plus:

Connect to SQL*Plus as SYSDBA:

```
sqlplus / as sysdba
```

Start the instance:

```
STARTUP;
```

Shutting Down an Oracle Database Instance

Shutdown the instance:

```
SHUTDOWN IMMEDIATE;
```

Other shutdown options include:

```
SHUTDOWN NORMAL;          – waits for users to disconnect.
SHUTDOWN TRANSACTIONAL;   – waits for transactions to complete.
SHUTDOWN ABORT;           – immediate shutdown, no waiting.
```

Opening and Closing PDBs

Opening a PDB

Connect to the Container Database (CDB) as a privileged user:

```
sqlplus / as sysdba
```

Open a PDB:

To open a specific PDB, you need to first alter the session to the root container and then open the PDB.

```
-- Alter the session to the root container
ALTER SESSION SET CONTAINER = CDB$ROOT;

-- Open the PDB
ALTER PLUGGABLE DATABASE pdb_name OPEN;
```

To open all PDBs:

```
ALTER PLUGGABLE DATABASE ALL OPEN;
```

Closing a PDB

Connect to the Container Database (CDB) as a privileged user:

```
sqlplus / as sysdba
```

Close a PDB:

To close a specific PDB, you need to first alter the session to the root container and then close the PDB.

```
-- Alter the session to the root container
ALTER SESSION SET CONTAINER = CDB$ROOT;

-- Close the PDB
ALTER PLUGGABLE DATABASE pdb_name CLOSE IMMEDIATE;
```

To close all PDBs:

```
ALTER PLUGGABLE DATABASE ALL CLOSE IMMEDIATE;
```

```
-- Query the status of all PDBs
SELECT pdb_name, open_mode FROM cdb_pdbs;
```

Alert Log

The alert log is a special trace file that records significant events for the Oracle database, including:

- ✓ Startup and shutdown information.
- ✓ Errors (e.g., ORA-00600).
- ✓ Warnings.
- ✓ Administrative operations (e.g., CREATE, ALTER, DROP).

Location of the Alert Log

The alert log is typically located in the alert directory of the Oracle Diagnostic Destination (DIAGNOSTIC_DEST). By default, it's in the trace subdirectory:

```
$ORACLE_BASE/diag/rdbms/<DB_NAME>/<SID>/trace/alert_<SID>.log
```

Location on Windows

The alert log on Windows is located in the Oracle Diagnostic Destination directory, typically under the trace subdirectory. The path follows this structure:

```
%ORACLE_BASE%\diag\rdbms\<DB_NAME>\<SID>\trace\alert_<SID>.log
```

Example Path

If your Oracle base is C:\app\oracle and your database SID is ORCL, the path would be:

```
C:\app\oracle\diag\rdbms\orcl\orcl\trace\alert_orcl.log
```

Viewing the Alert Log

To view the alert log, you can use the following methods:

- Use the tail command in Unix/Linux:

  ```
  tail -f $ORACLE_BASE/diag/rdbms/<DB_NAME>/<SID>/trace/alert_<SID>.log
  ```

- Open the file directly in a text editor.

Trace Files

Trace files are generated by Oracle processes to provide detailed diagnostic information. They are used for debugging and diagnosing problems. Each Oracle process (e.g., DBWR, LGWR) can generate trace files.

Location of Trace Files

Trace files are located in the same directory as the alert log:

```
$ORACLE_BASE/diag/rdbms/<DB_NAME>/<SID>/trace/
```

Location on Windows

Trace files are also located in the Oracle Diagnostic Destination directory, under the trace subdirectory. The path follows this structure:

```
%ORACLE_BASE%\diag\rdbms\<DB_NAME>\<SID>\trace\<trace_filename>.trc
```

Example Path

For a database with SID ORCL:

```
C:\app\oracle\diag\rdbms\orcl\orcl\trace\<trace_filename>.trc
```

Types of Trace Files

- User Trace Files: Generated by user sessions.
- Background Trace Files: Generated by background processes like DBWR, LGWR, etc.

Viewing Trace Files

Similar to the alert log, you can view trace files using the tail command or by opening them in a text editor:

```
tail -f $ORACLE_BASE/diag/rdbms/<DB_NAME>/<SID>/trace/<trace_filename>.trc
```

DDL Log File

DDL log files record DDL statements executed in the database. This can be useful for auditing and monitoring schema changes.

Enabling DDL Logging

DDL logging is not enabled by default. To enable it, you need to set the following parameters in the sqlnet.ora file or dynamically:

Add the following line to the sqlnet.ora file:

```
DIAG_ADR_ENABLED = OFF
```

Enable DDL Logging Dynamically:

```
ALTER SYSTEM SET ENABLE_DDL_LOGGING=TRUE SCOPE=BOTH;
```

Location of DDL Log Files

The DDL log file is located in the log directory of the Oracle Diagnostic Destination:

```
$ORACLE_BASE/diag/rdbms/<DB_NAME>/<SID>/log/ddl.log
```

Location on Windows

DDL log files, if enabled, are located in the log directory of the Oracle Diagnostic Destination:

```
%ORACLE_BASE%\diag\rdbms\<DB_NAME>\<SID>\log\ddl.log
```

Example Path

For a database with SID ORCL:

```
C:\app\oracle\diag\rdbms\orcl\orcl\log\ddl.log
```

Viewing the DDL Log File

You can view the DDL log file using the tail command or by opening it in a text editor:

```
tail -f $ORACLE_BASE/diag/rdbms/<DB_NAME>/<SID>/log/ddl.log
```