ORACLE®

# DATABASE ADMINISTRATION

## LAB WORK BOOK

NURIZAH BINTI MAHMOR
NOR ANISAH BINTI MOHD SAAD

ORACLE®

# DATABASE ADMINISTRATION

## LAB WORKBOOK

**DEPARTMENT OF INFORMATION TECHNOLOGY AND COMMUNICATION**
**POLITEKNIK UNGKU OMAR**

**DEPARTMENT OF POLYTECHNIC EDUCATION AND COMMUNITY COLLEGES MINISTRY OF HIGHER EDUCATION**

# PREFACE

Thankful to Allah since by His mercy, we were able to finish the Oracle Database Administration Lab Workbook. We learned a lot of useful writing skills while putting this Lab Workbook together. On this occasion, we want to express our sincere gratitude to all those who contributed to the creation, production, and publication of our Lab Workbook. All the guidance and knowledge that was shared helped us a lot in the effort to produce this Lab Workbook. We would also like to say a million thanks to the management of Politeknik Ungku Omar, the Head of the Information Technology and Communication (ICT) Department who has given us the opportunity and moral support. Sincere appreciation also addressed to all parties who have been involved in making this Lab Workbook a success either directly or indirectly. We greatly appreciate all the help you have given because, without your help and support, this Lab Workbook could not be produced and published.

# DECLARATION & DISCLAIMER



**e-ISBN: 978-629-7635-06-4**

**ORACLE DATABASE ADMINISTRATION LAB WORKBOOK**

**WRITER:**
NURIZAH BINTI MAHMOR (nurizah@puo.edu.my)
NOR ANISAH BINTI MOHD SAAD (anisaad@puo.edu.my)

**PUBLISHER:**
POLITEKNIK UNGKU OMAR,
JALAN RAJA MUSA MAHADI,
31400 IPOH,
PERAK DARUL RIDZUAN
Tel: 05-545 7656/ 05-545 7622
https://www.puo.edu.my

**WRITER**

NURIZAH BINTI MAHMOR
NOR ANISAH BINTI MOHD SAAD

**EDITOR**

MUNIRAH BINTI ABDULLAH

**ILUSTRATOR**

NOR ANISAH BINTI MOHD SAAD

# WRITER'S DETAILS

**NURIZAH BINTI MAHMOR**
SENIOR LECTURER
JABATAN TEKNOLOGI MAKLUMAT & KOMUNIKASI
POLITEKNIK UNGKU OMAR
nurizah@puo.edu.my
012-5412858

Nurizah Binti Mahmor holds a Master's Degree in Software Engineering from Universiti Teknikal Malaysia (UTEM). She started her carrier as a lecturer in 2005 and currently teaching in Politeknik Ungku Omar. Currently, she is the Head od Student Affair Department's Division. Her subject interest revolves around Database, Programming, Web Development and etc.

**NOR ANISAH BINTI MOHD SAAD**
SENIOR LECTURER
JABATAN TEKNOLOGI MAKLUMAT & KOMUNIKASI
POLITEKNIK UNGKU OMAR
anisaad@puo.edu.my
019-5719676

Nor Anisah Binti Mohd Saad holds a Bachelor's Degree in Computer Science (Software Engineering) from Universiti Sains Malaysia (USM). She started her carrier as a lecturer in 2006 and currently teaching in Politeknik Ungku Omar. Her subject interest revolves around database, Software Development and etc

# CONTENT

# LAB ACTIVITY 1

# LAB ACTIVITY 1: Introduction to Oracle Database Architecture – Part 1

Duration: 3 Hours

## Learning Outcomes

This activity encompasses activities 1A, 1B and 1C.

By the end of this laboratory session, you should be able to:

1. Identify major components of Oracle Server
2. Explain major elements of Oracle Server components.

## Activity 1A


**Activity Outcome: Identify major components of Oracle Server.**

Oracle server consists of two components; Oracle instance and Oracle database. Briefly explain in your own words the definitions below.

Oracle Instance

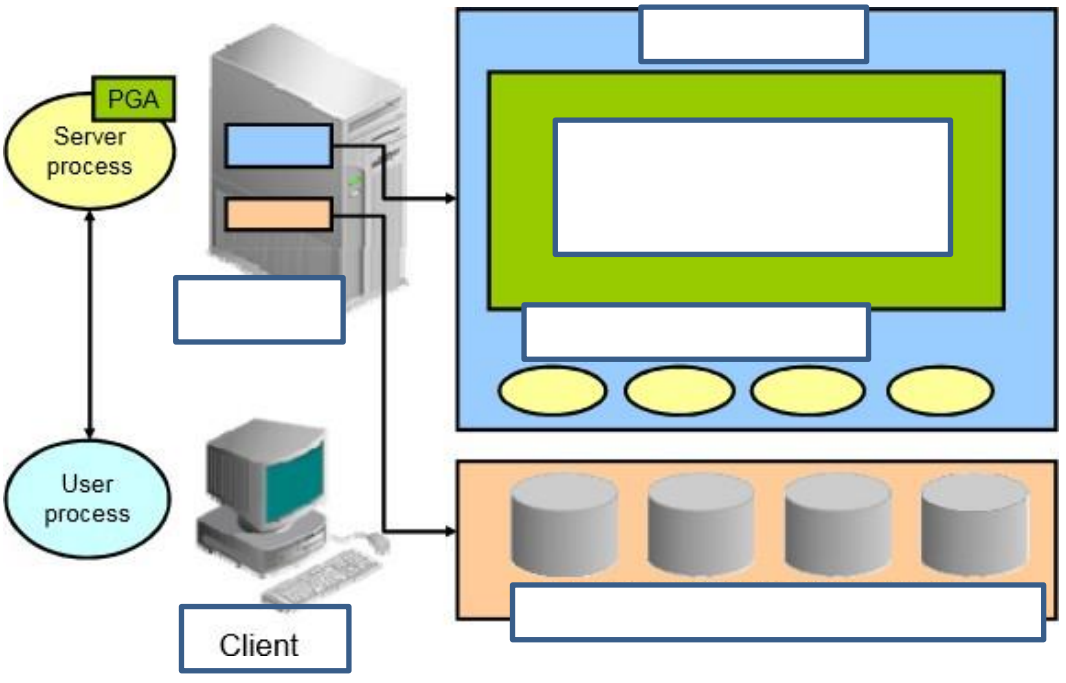
Oracle Database

## Activity 1B


**Activity Outcome: Identify major elements of Oracle Server components.**

Fill in the suitable element for the Oracle Server Architecture below.

| Processes | Client | Database |
|---|---|---|
| Memory (SGA) | Server | Instance |




ORACLE SERVER ARCHITECTURE

## Activity 1C


**Activity Outcome: Explain major elements of Oracle Server components.**

Briefly explain the following elements of Oracle Server components.

| ORACLE DATABASE | ORACLE INSTANCE |
|---|---|
| **Shared Global Area (SGA) :**<br>• Shared pool<br>• Database buffer cache<br>• Redo log buffer<br>• Large pool<br>• Java pool<br>• Streams pool | **Logical Structure :**<br>• Tablespace<br>• Segment<br>• Extent<br>• Data blocks |

| ORACLE DATABASE | ORACLE INSTANCE |
|---|---|
| **Background Processes :**<br>• Database Writer process (DBWn)<br>• Log Writer process (LGWR)<br>• Checkpoint process (CKPT)<br>• System monitor process (SMON)<br>• Process monitor process (PMON)<br>• Recoverer process (RECO)<br>• Manageability monitor process (MMON)<br>• Archiver processes (ARCn) | **Physical Structure :**<br>• Data file<br>• Log file<br>• Control file<br>• Online redo log file<br>• Archive file<br>• Password file<br>• Parameter file |

# LAB ACTIVITY 1: Introduction to Oracle Database Architecture – Part 2

Duration: 3 Hours

## Learning Outcomes

This activity encompasses activities 1A, 1B and 1C.

By the end of this laboratory session, you should be able to:

1. Installing Oracle Software
2. Create Listener
3. Create database

## Activity 1A


**Activity Outcome: Install Oracle Database 19c on Windows**

**Step 1: Download Oracle Database 19c software for Windows**

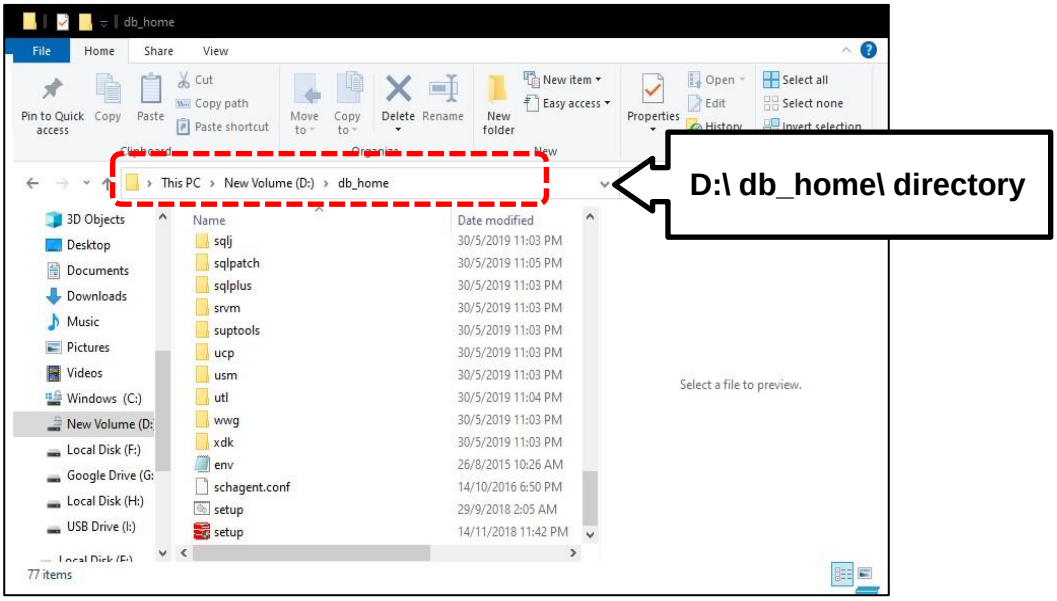
Our first step is to get the Oracle 19c software for Windows from the official Oracle download page. On this page you will find many packages. In this guide, we will focus on the Oracle Database 19c (19.3) for Microsoft Windows x64 (64-bit) package. If you want to access your future Oracle database remotely, you can also download the Oracle Database 19c Client (19.3) for Microsoft Windows x64 (64-bit) or Oracle Database 19c Client (19.3) for Microsoft Windows (32- bit) according to the architecture of the client computer.

Downloading Oracle 19c software is free, however, to use it in the production environment requires a license.

**Step 2: Launch the setup wizard.**

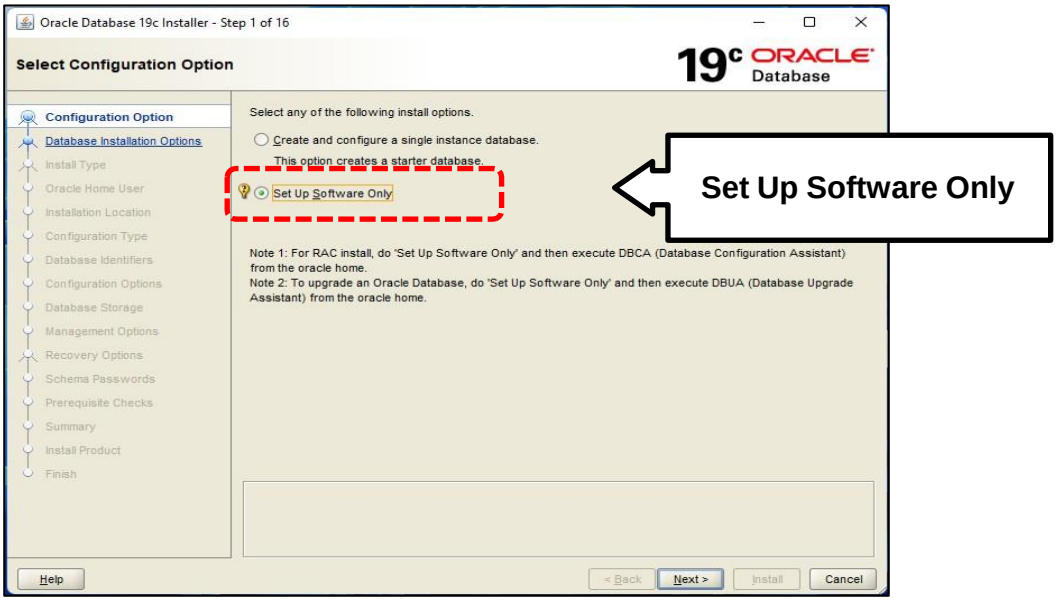
- Once the download is complete
- **unzip** the package then **copy it to the root** of your disk; (**Suggested D:/ directory**)
- **rename the folder**, choose a shorter name (e.g. **db_home**);
- then launch the setup.

Note: the installation wizard may take several minutes to open. So be patient.




**Step 3: Choose database installation options.**

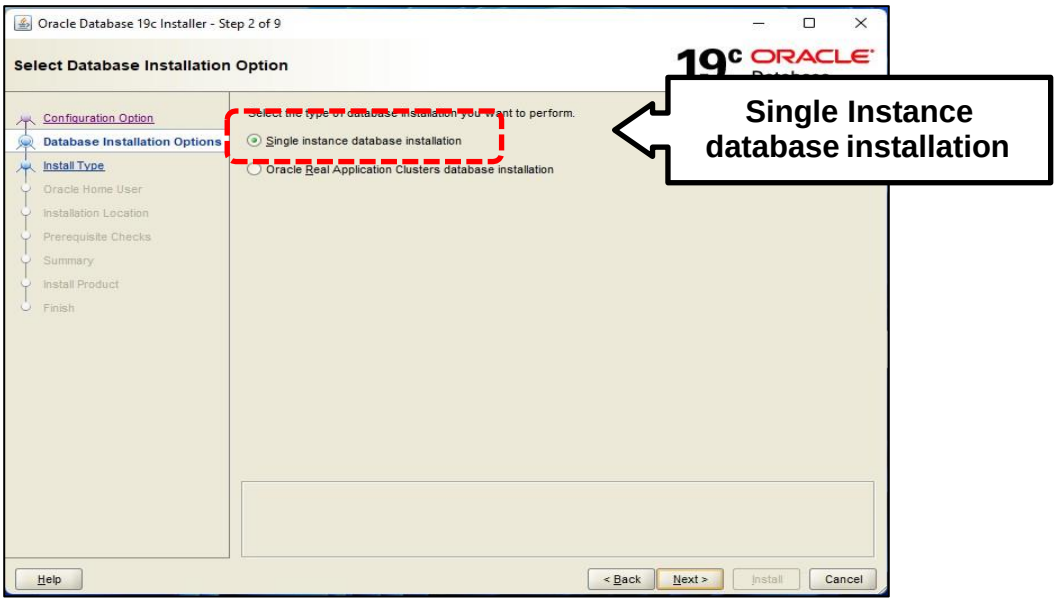
This step is very important. It allows you to choose the database installation options. You can install Oracle software and create a database at the same time (Create and configure a single instance database). Since we only want to install the Oracle 19c software and its components, we will opt for **"Set up Software Only"**. This option installs the essential components for creating and administering a database. You can also use it to upgrade an older version of Oracle (example: 12c or 18c) or install RAC.



**Step 4: Select database installation type.**

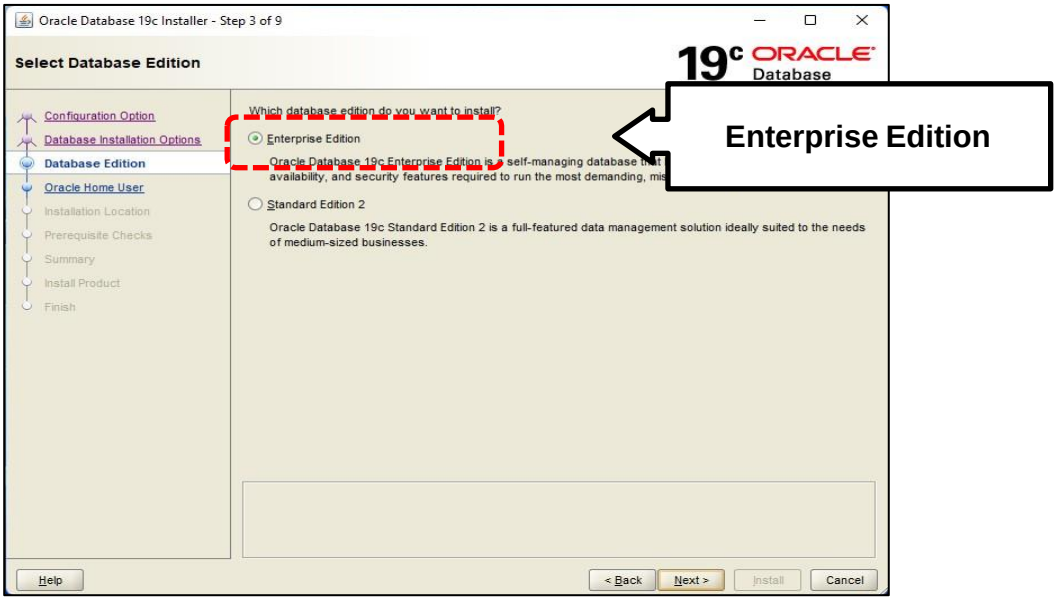
The first option allows you to install a single instance database on your server. Oracle Database with the Oracle Real Application Clusters (RAC) option allows multiple instances running on different servers to access the same physical database stored on shared storage. AS we want to install Oracle 19c on a single server, we select option 1.

You can create a database after installation by using Oracle Database Configuration Assistant (Oracle DBCA).




**Step 5: Choose database edition.**

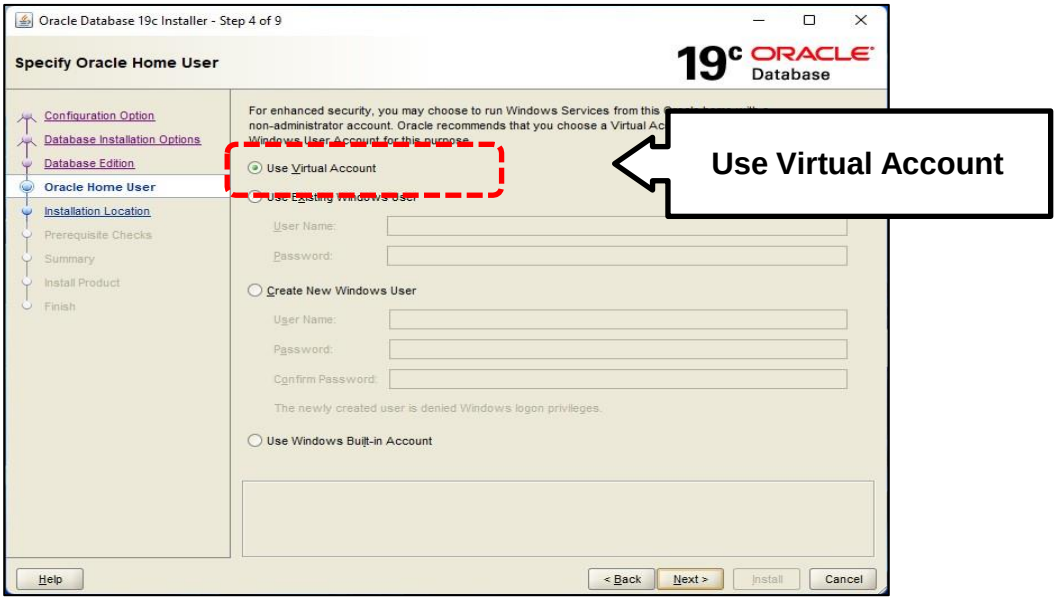
The version of the database to install depends on your needs. For an application developer or and medium-size companies, the standard version covers practically all needs. To take full advantage of Oracle 19c, you can install the Enterprise version. Obviously, the Enterprise version requires more resources (storage, RAM, CPU).




**Step 6: Specify Oracle home user.**

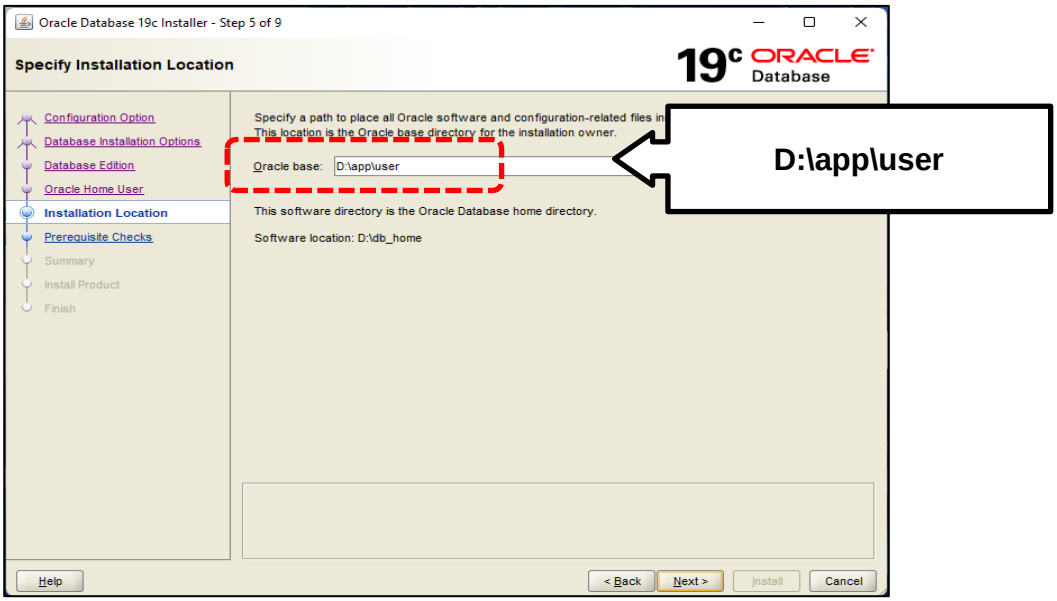
During Oracle Database installation, you can specify an optional Oracle home user associated with the Oracle home. Oracle home user can be a Windows built-in account (LocalSystem for Server and LocalService for Client), virtual account, or a regular (not an administrator) Windows account. If you specify an existing user as the Oracle home user, then the Windows user account you specify can either be a Windows domain user or a Windows local user.

A Windows user account need not be created by the administrator if a virtual account or a Windows built-in account is used during installation. If you specify a non-existing user as the Oracle home user, then the Windows user account you specify must be a Windows local user. The installer creates this account automatically to run the Windows services for the Oracle home. Do not log in using this account to perform administrative tasks.



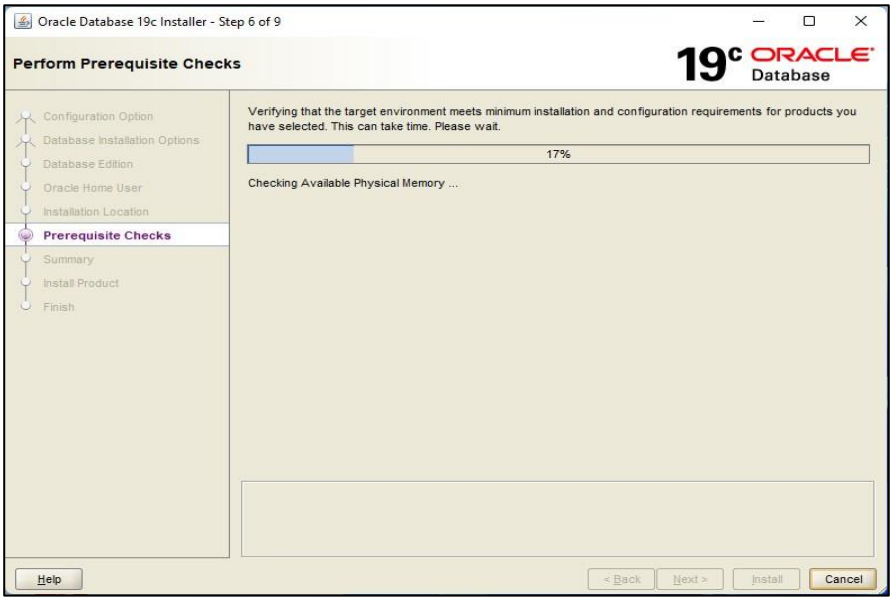

**Step 7: Specify the location of Oracle software.**

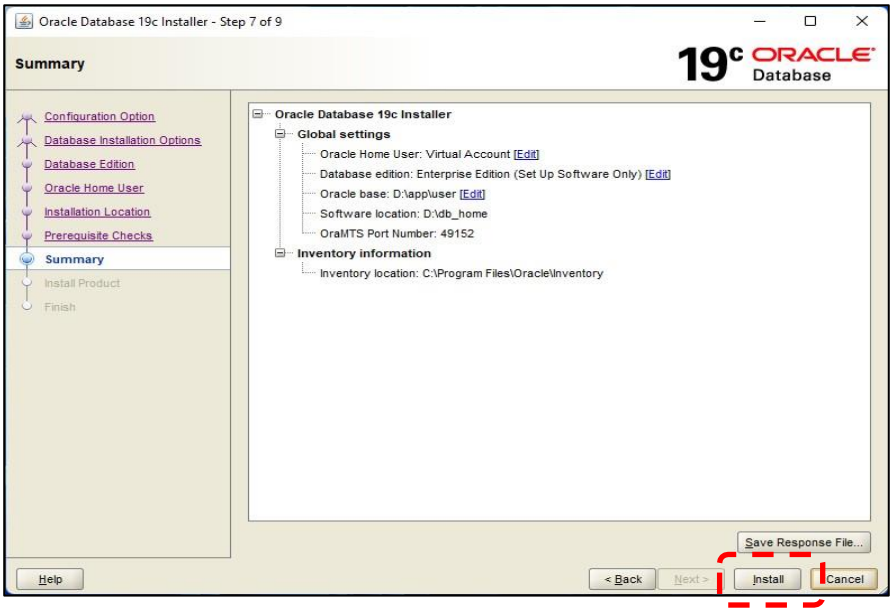
In a default Windows installation, the Oracle base directory appears as follows: ***DRIVE_LETTER:\app\username*** where *username* is the Oracle installation user if you choose Windows built-in account, else it is the Oracle Home user (standard Windows user account). You can change this directory at your convenience or leave it as default.



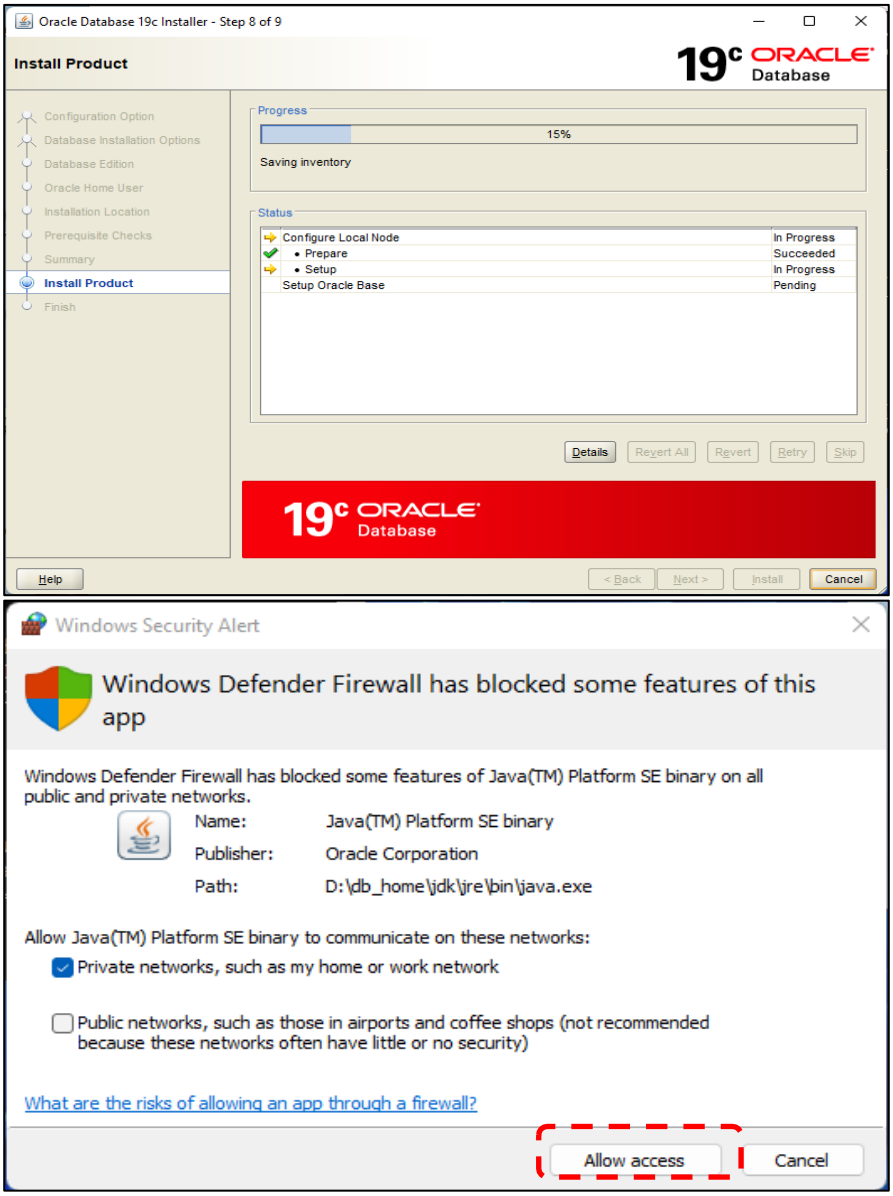
**Step 8: Minimum requirements checks, summary and end of the installation.**

After these initial phases of configuration, let's check the installation prerequisites.
If, however, there are errors, try to readjust the minimum installation requirements and start again.

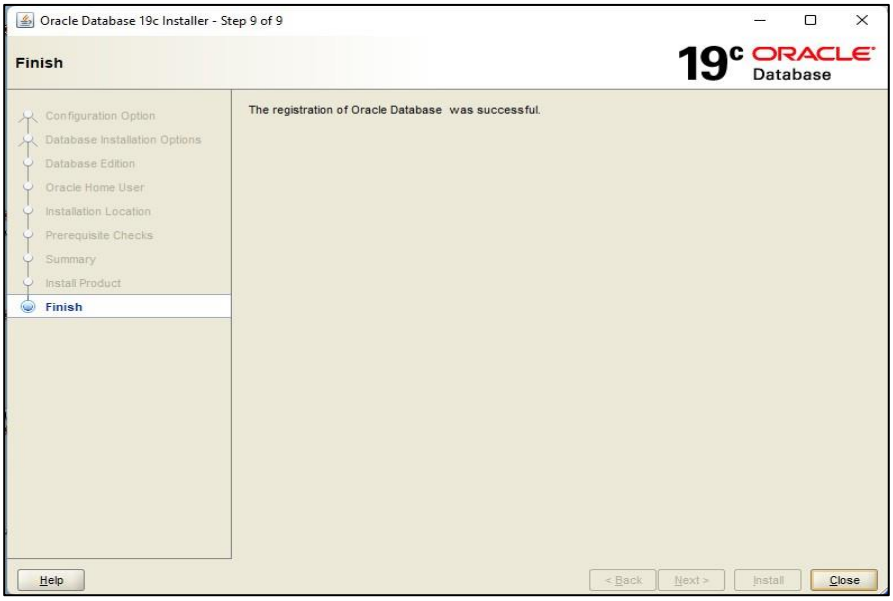


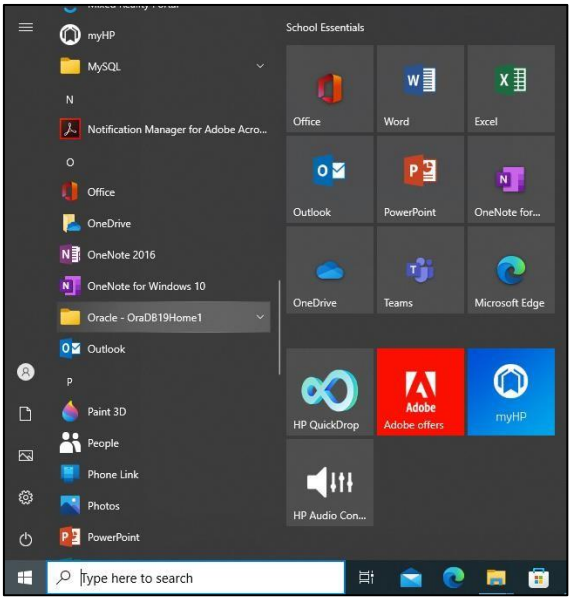

Click **install** if no error occurs.




That’s it. You can close the installation wizard now.




You can look at the different components of your installation from the Windows start menu.



## Activity 1B

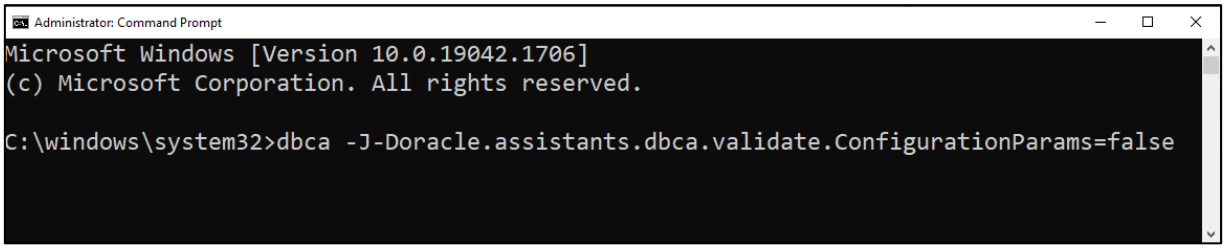

**Activity Outcome: Create a database in Oracle 19c on Windows**

**Step 1: Launch DBCA**

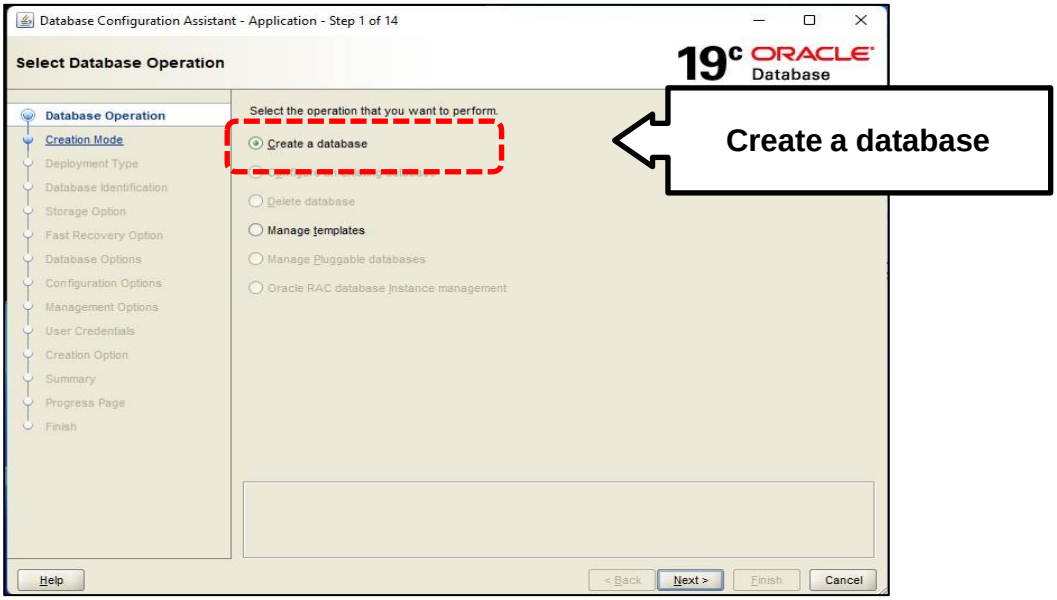
The Oracle DBCA tool is available after installing the Oracle 19c software. To launch it, you must log into Windows as an administrator and use one of the methods below:

- open it from the Windows Start menu;
- run the ***dbca*** command from the Windows command prompt;
- execute the following combination**: "Windows + R"**, then type ***dbca***.

(Preferably, use command prompt – Run as Administrator to launch DBCA)




**Step 2: Select "Create a database" and click "Next"**

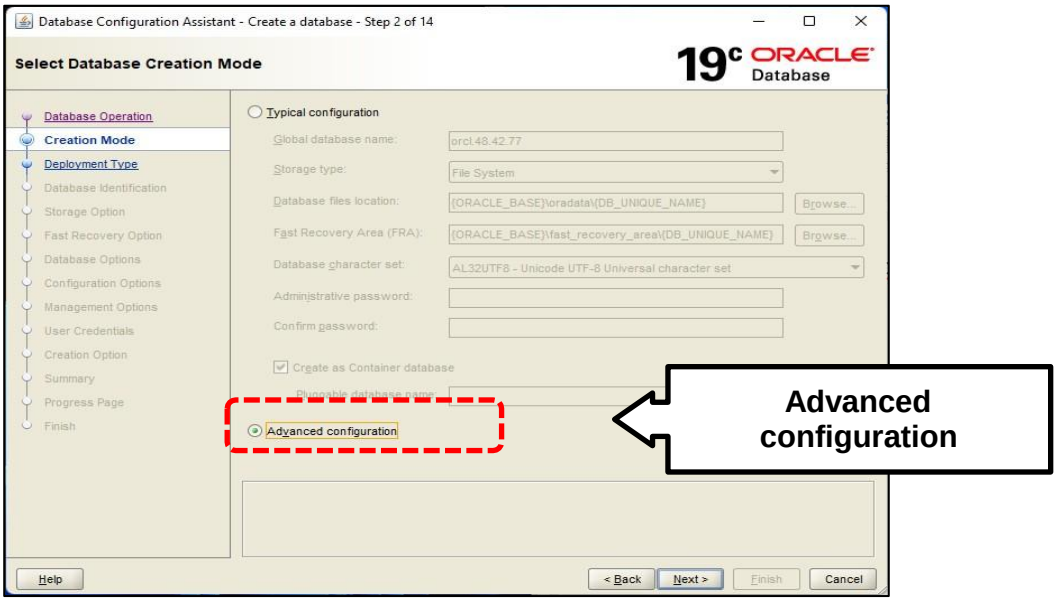


**Step 3: Select database configuration mode**

DBCA enables you to create a database with typical configuration or with advanced configuration:

- in “Typical configuration” mode, you can set up your database quickly using Oracle prebuilt templates.
- in “Advanced configuration” mode, you can customize storage locations, management options, database options, configuration option, user credentials, etc. It allows you to have full control of your database configuration.

In the following, we use the Advanced configuration mode.




**Step 4: Select database deployment type**

This step enables you to select the type of database and template to use to create the database. You can select:

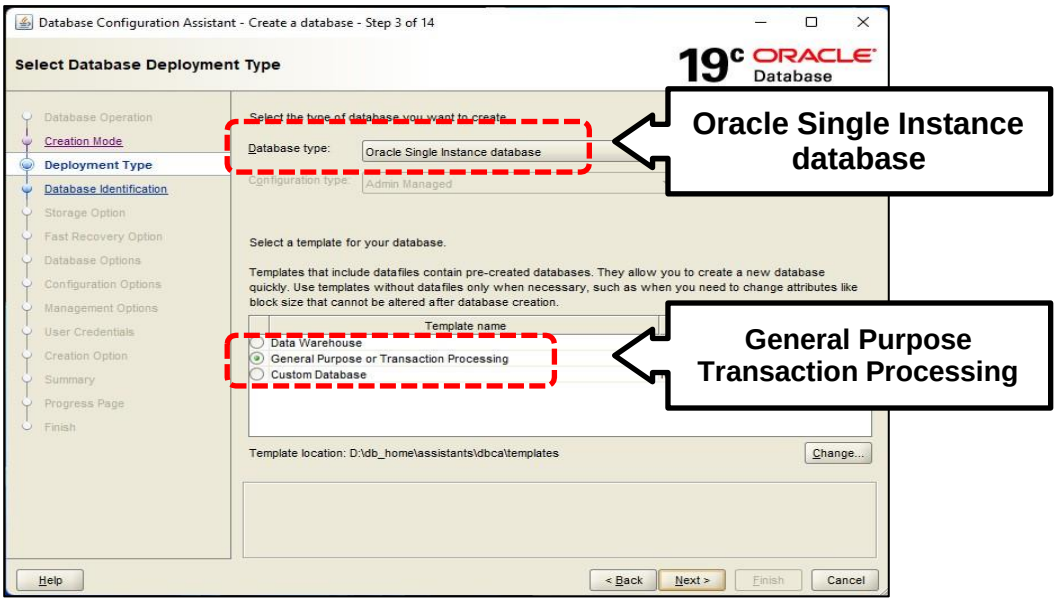
- Oracle Single Instance Database
- RAC database
- RAC node database

for the database type and

- Data Warehouse
- General Purpose or Transaction Processing
- Custom Database

for the database template.

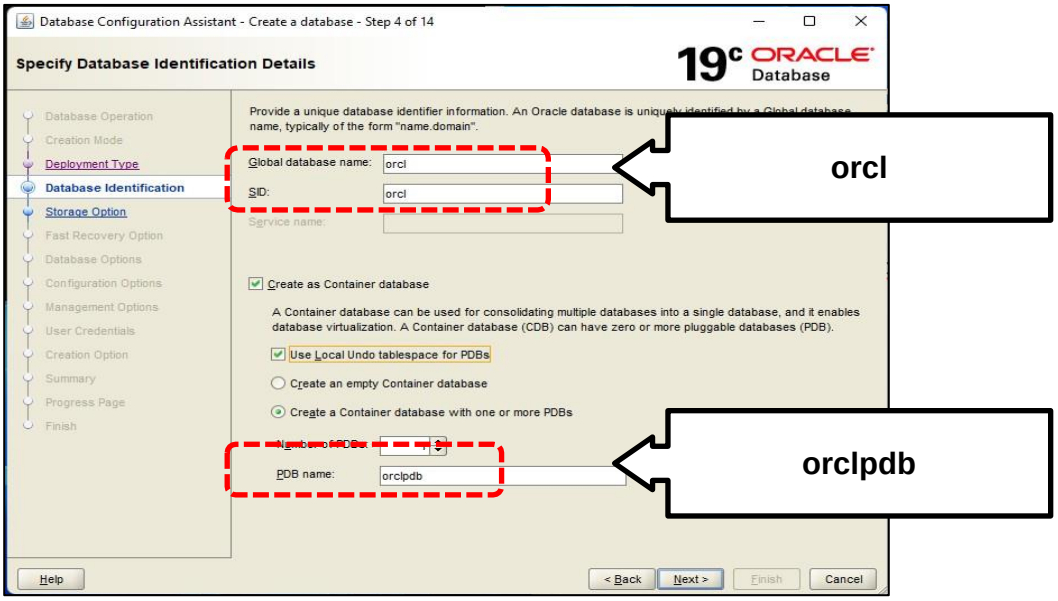
In this guide, use **"Oracle Single Instance Database"** and **"General Purpose or Transaction Processing"** as a template. However, you can select the type and template suited to the type of workload your database will support. For more information on templates, click on **"View details"** next to each template.




**Step 5: Specify database identification details**

In this step, provide the global database name, something like **"database.domaine_name"**. You don't need to fill the SID. It is created automatically from the global database name.

If you want to create a multitenant container database (CDB), then check Create as Container Database and specify the number of pluggable databases the CDB can support. You can also create an empty CDB.

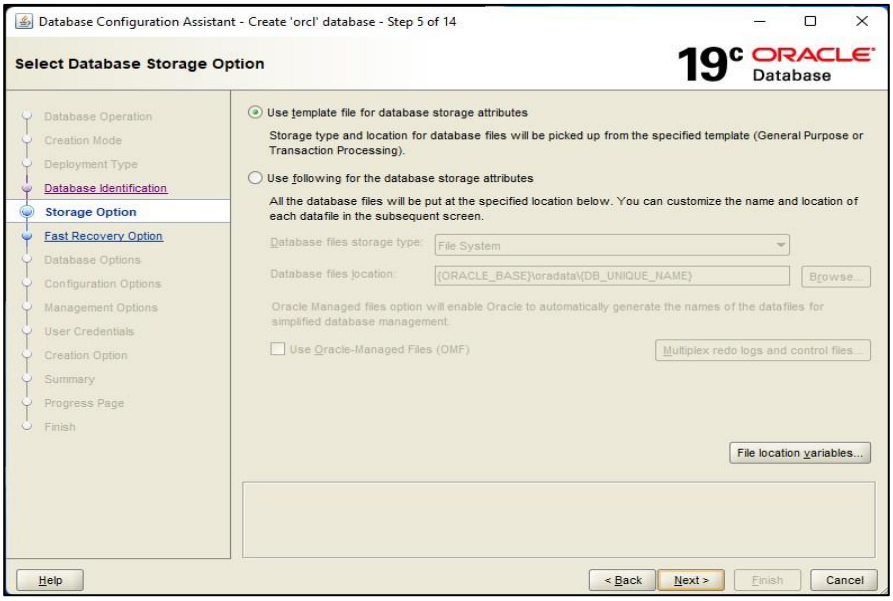



**Step 6: Select database storage option**

You can customize the database storage options in different ways. Make sure your select **"Use template file for database storage attributes"** and click **"Next"**.

If you want to specify your own location to store database files, select **"Use following for the database storage attributes"** option. With this option, you need to choose how the database files will be managed:

- File System option: your operating system will manage your database files.
- Automatic Storage Management (ASM) option: you place your data files in Oracle Automatic Storage Management (Oracle ASM) disk groups.
- Oracle-Managed Files (OMF) option: Oracle Database will directly manage operating system files comprising an Oracle database.

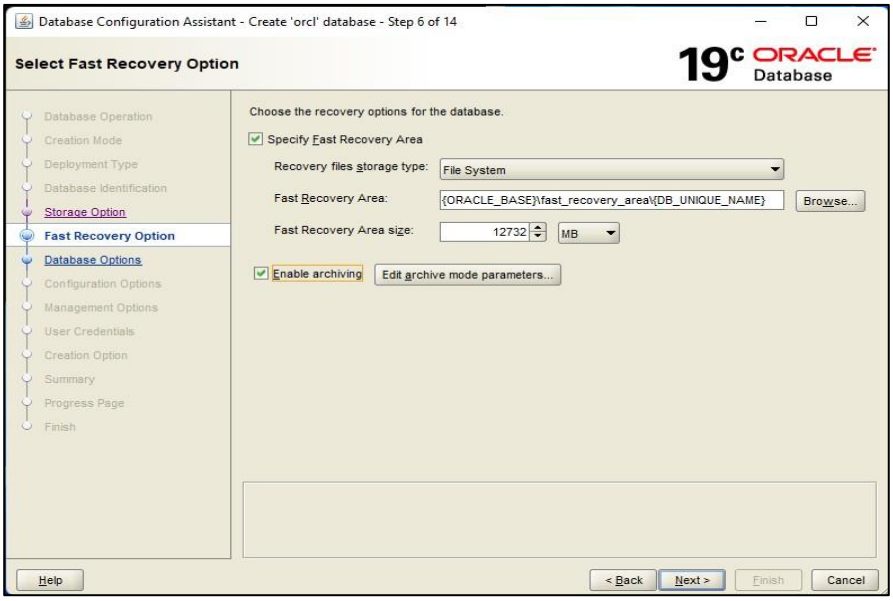



### Step 7: Select Fast recovery option

The fast recovery is an essential component of your database. In fact, it allows you to recover your data if a system failure occurs. It is a location in which Oracle Database can store and manage files related to backup and recovery.

Check **"Specify Fast Recovery Area"** to specify a backup and recovery area and its directory location, file storage type and size.

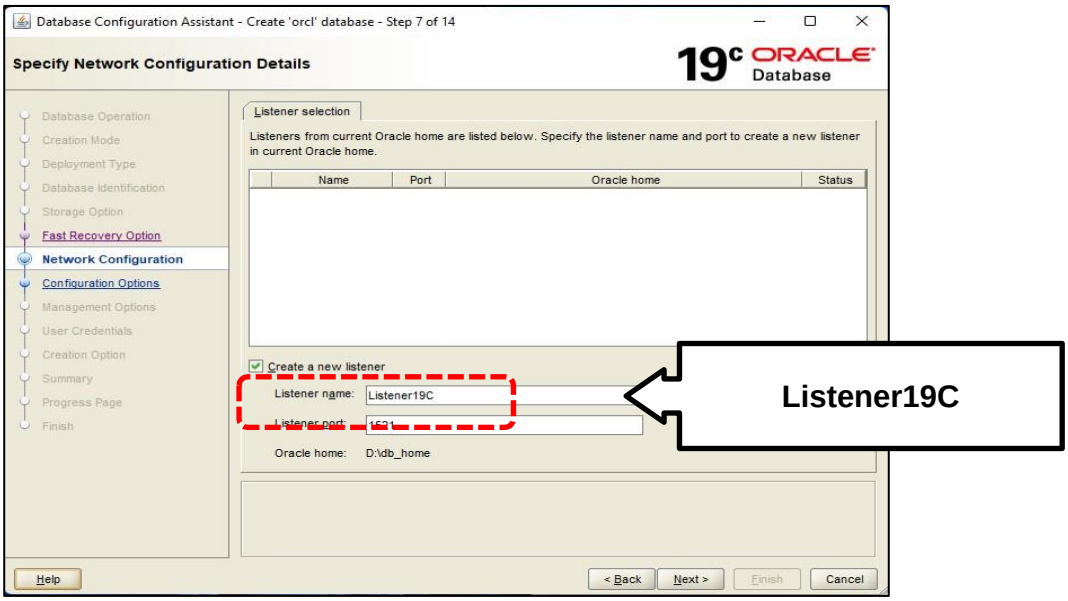
You can also to enable the archiving of database online redo log files, which Oracle uses to recover a database with the **"Enable archiving"** option.



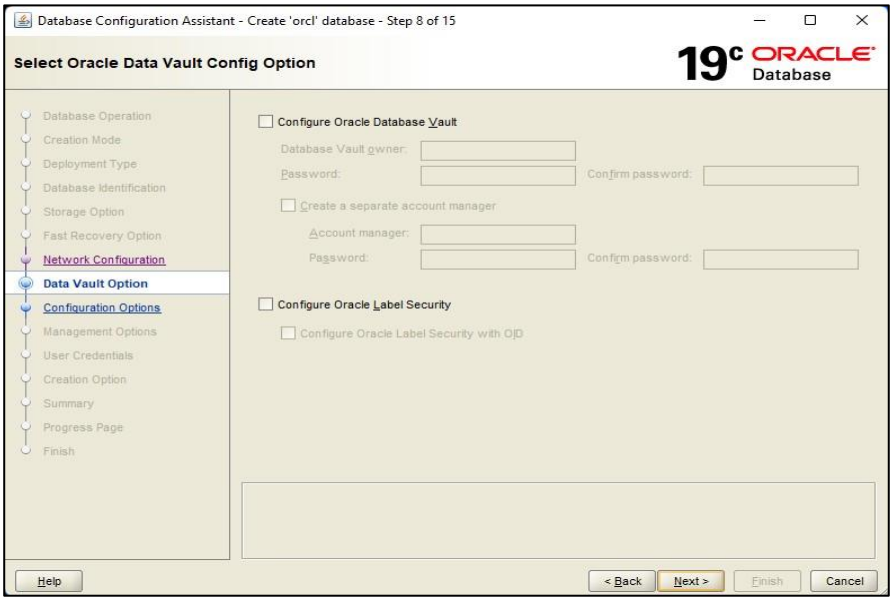

**Step 8: Specify network configuration details**

Configuring a listener is mandatory if you want to access your database remotely. A listener receives incoming client connection requests and manages the traffic of these requests to the database server.

In this step, you can select among the listeners in the current Oracle home or create a new one by providing the listener name and a port number. Then click **"Next"** to continue.



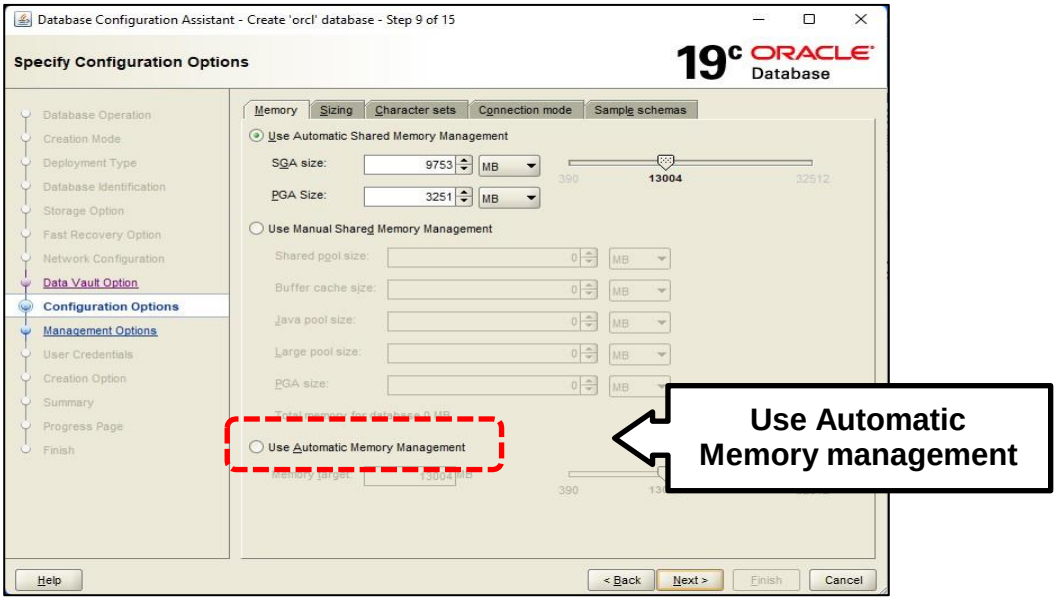
You can configure Oracle Database Vault and Oracle Label Security in the next window, or you can click Next to continue through DBCA without configuring Oracle Database Vault and Oracle Label Security.




**Step 9: Specify configuration options**

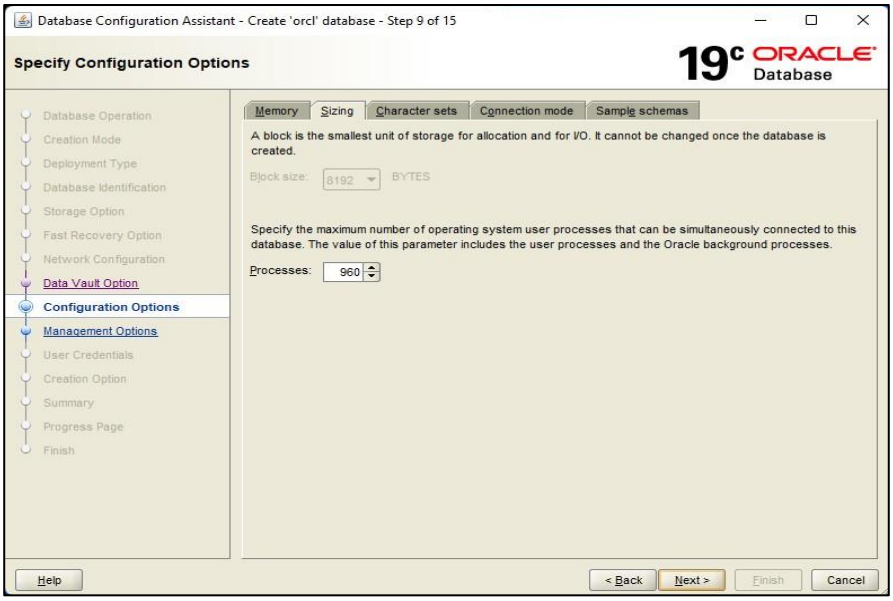
**Memory allocation:** The Memory tab enables you to control how the database manages its memory. You can either use:

- Automatic Shared Memory Management if you want to allocate specific amounts of memory to the SGA and aggregate PGA to your database instance;
- Manual Shared Memory Management if you want to allocate specific memory amount for each SGA component and the aggregate PGA;
- Automatic Memory Management if you want Oracle to automatically tune the memory components of the SGA, and allocates memory to individual PGAs as needed.

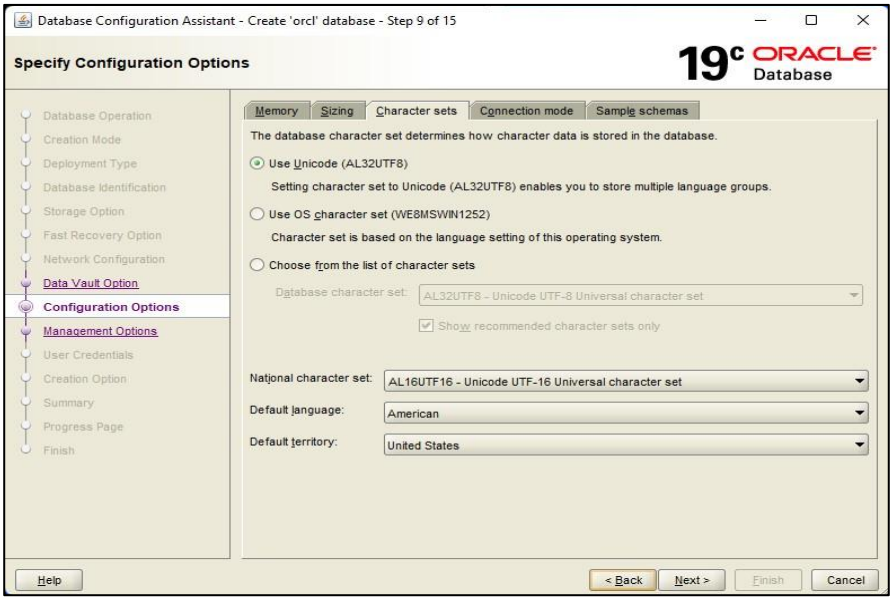



**Block size and processes:** this tab allows you to set the database data block size and the maximum number of user processes that can simultaneously connect to the database.

The maximum number of processes depend on many parameters. The value you select should allow for all background processes, user processes, and parallel execution processes. A small value for the maximum number of processes may result to the database not running.



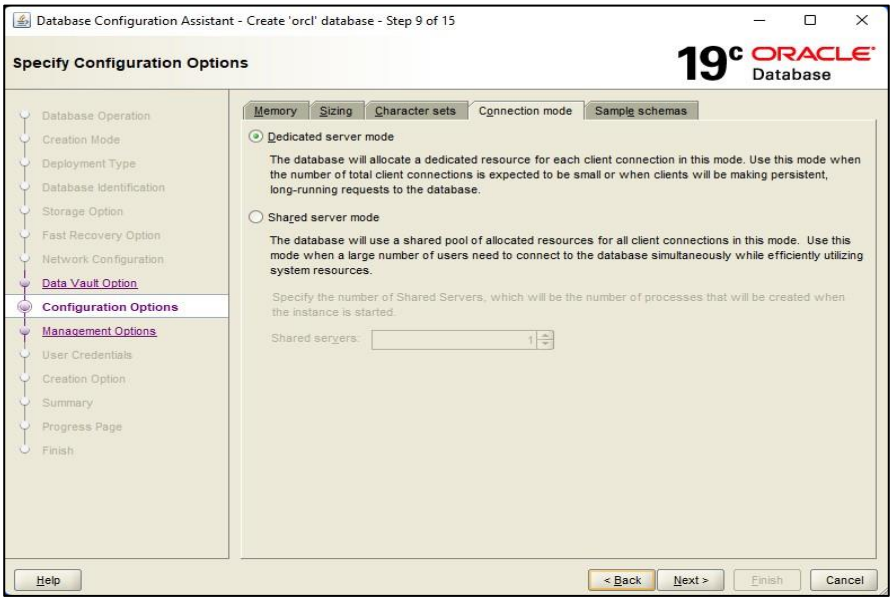

**Character sets:** Use this tab to determine how character data is stored in the database. Select Unicode (AL32UTF8) as the database character set.



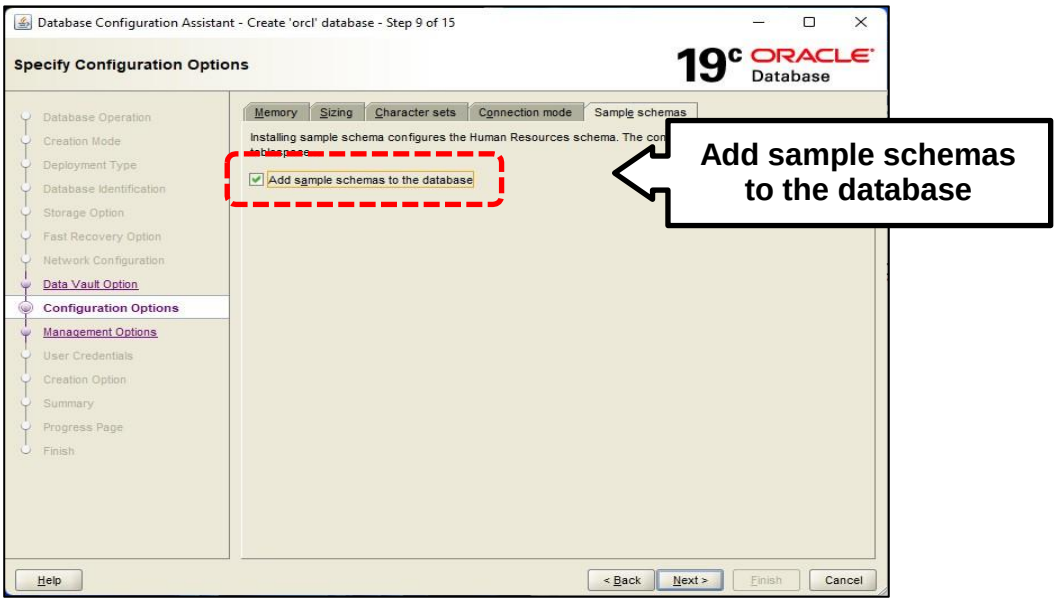

**Connection mode:** this tab enables you to select the database connection mode.

In Dedicated server mode, each user process is associated with a dedicated server process. This option is suitable when the number of clients is small.

In Shared serve mode, several client connections share a database-allocated pool of resources. This mode is the best option when client load is expected to cause a strain on memory and other system resources. If you select this mode, then make sure you provide the number of Shared servers.



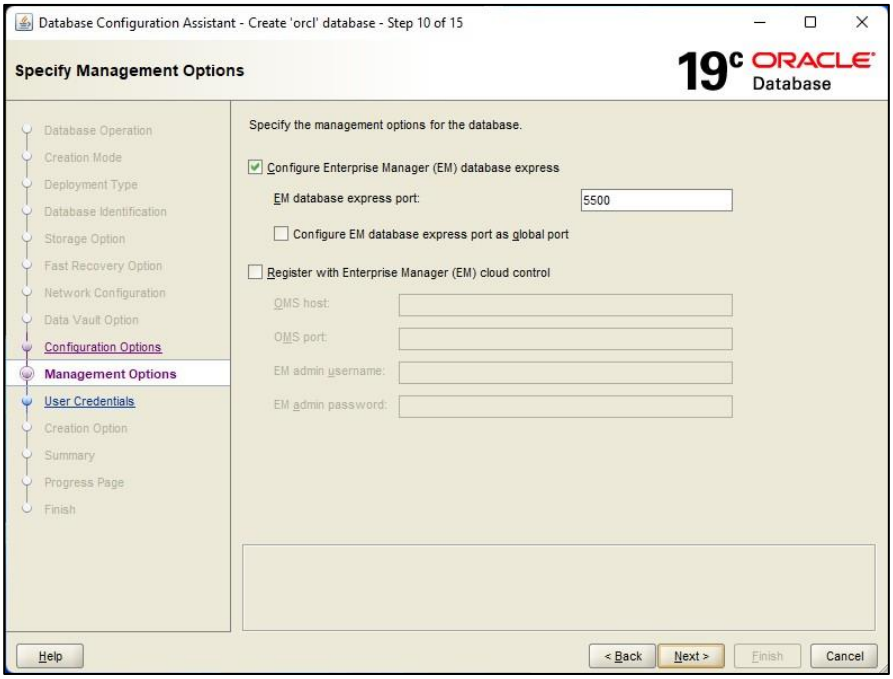
**Sample schemas:** this tab allows you to include the sample schemas like HR and OE in your database. Select **"Add sample schemas to the database"** if you want to use them later.




**Step 10: Specify management options**

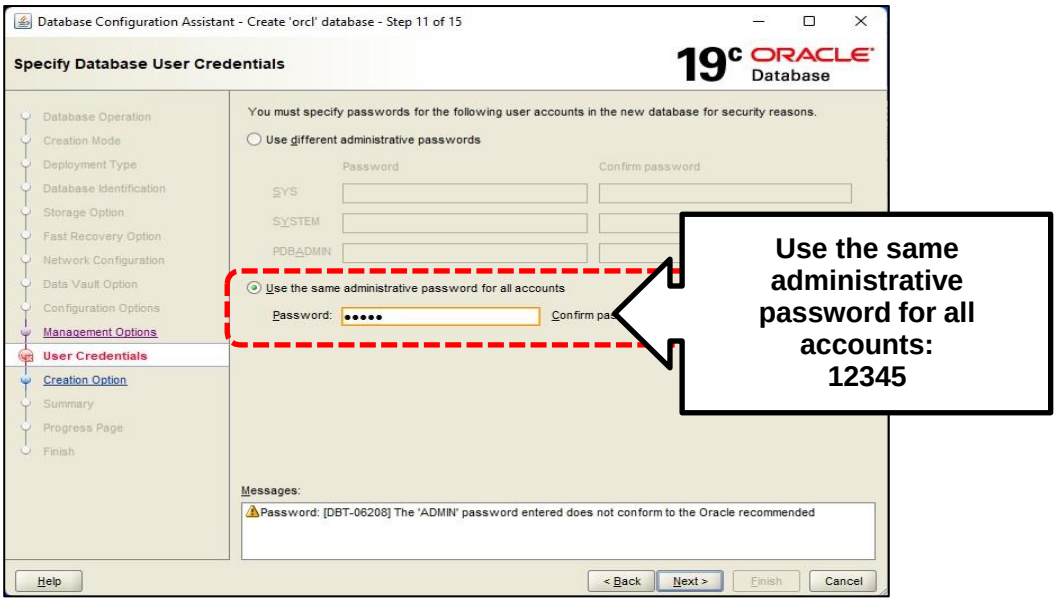
Enterprise Manager provides Web-based management tools for Oracle databases. You can select Configure Enterprise Manager (EM) express and click **"Next"**.

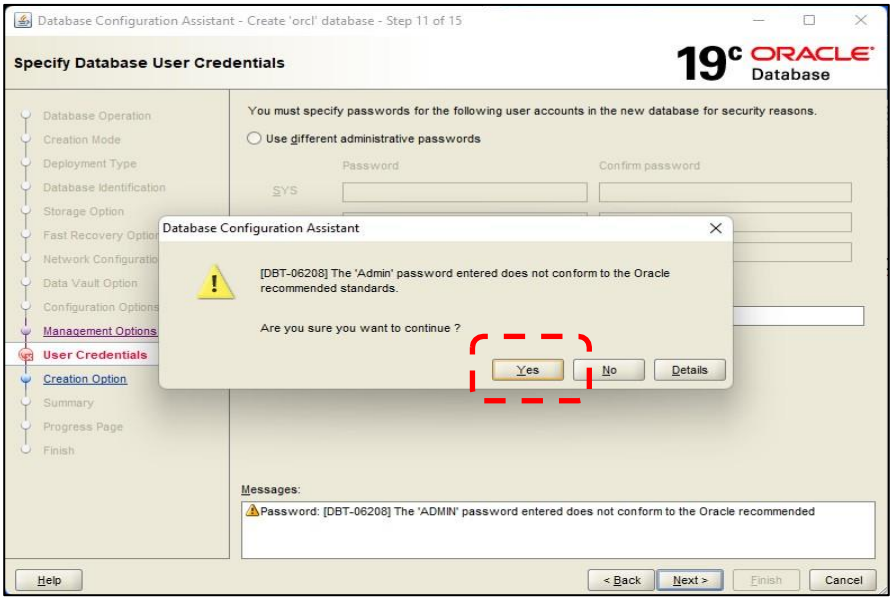
If Enterprise Manager cloud control is installed on your host computer, then you can choose Register with Enterprise Manager (EM) cloud control and provide the necessary connection details (host, port number, username, and password).




**Step 11: Specify database user credentials**

Provide passwords for the administrative accounts SYS and SYSTEM and the Oracle home user account. You can specify a password for each administrative account or use the same password for all accounts. For security reasons, I recommend to set different password for each administrative account.





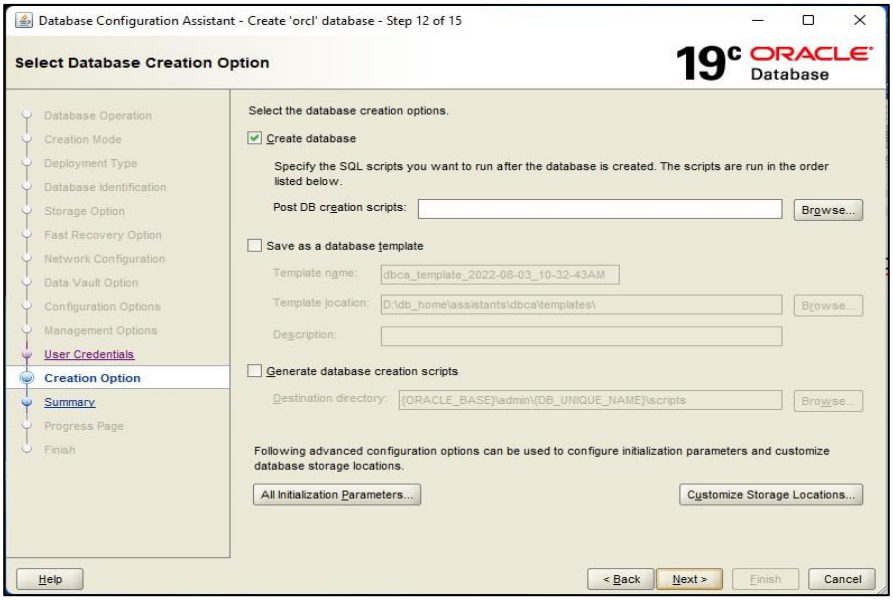

Note: Choose the second option to Use the same administrative password for all accounts. Set the Password and Confirm password to 12345 / admin. (Please do remember this Password for further process)

### Step 12: Select database creation option

In this step, you can select any of the following options for creating the database:

- **"Create database"** to create your database now;
- **"Save as a database template"** to save the database definition as a template to use at a later time;
- **"Generate database creation scripts"** to generate a SQL database creation script that you can run at a later time.

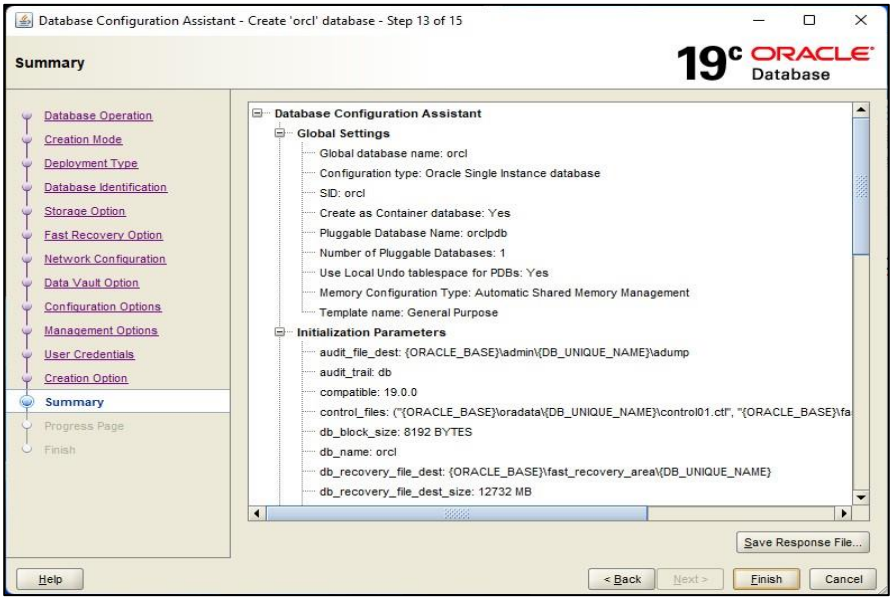
Moreover, you can adjust the server initialization parameters, relocate or replicate your database files (control files, redo logs, etc.).

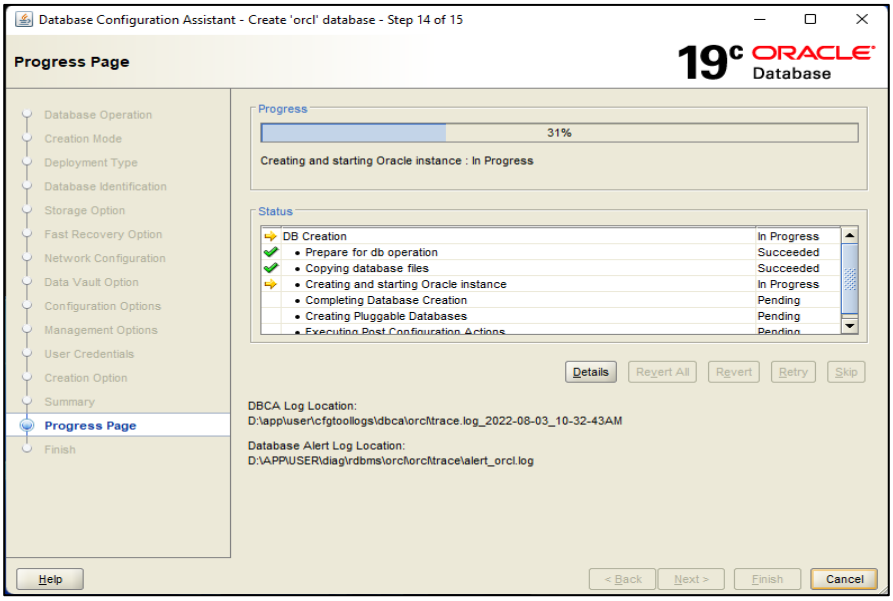



### Step 13: Oracle database creation summary

This step enables you to review the summary information. To change any of these options, click **"Back"** and return to the window where you can modify the option.

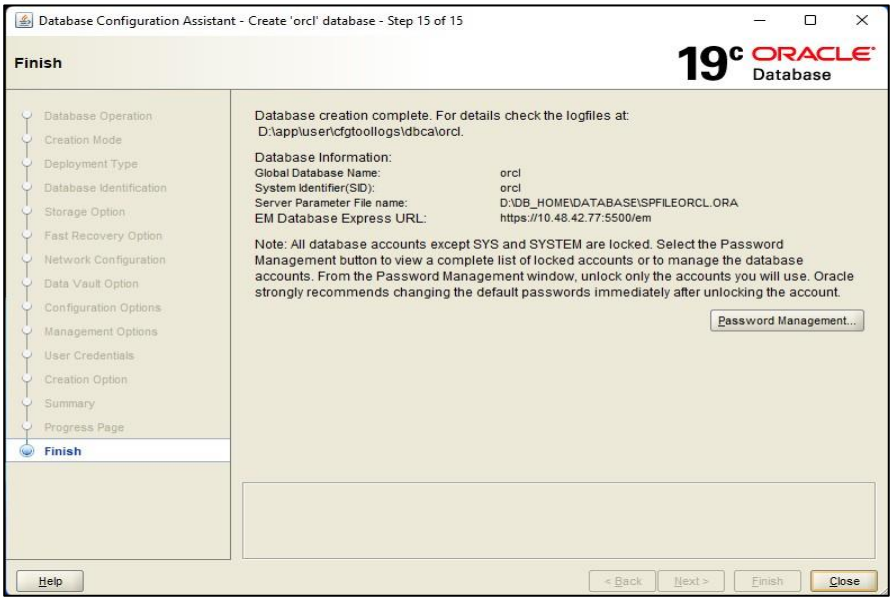
Click **"Finish"** to start the creation of the database.






Voilà! You have successfully created your Oracle 19c database with DBCA.

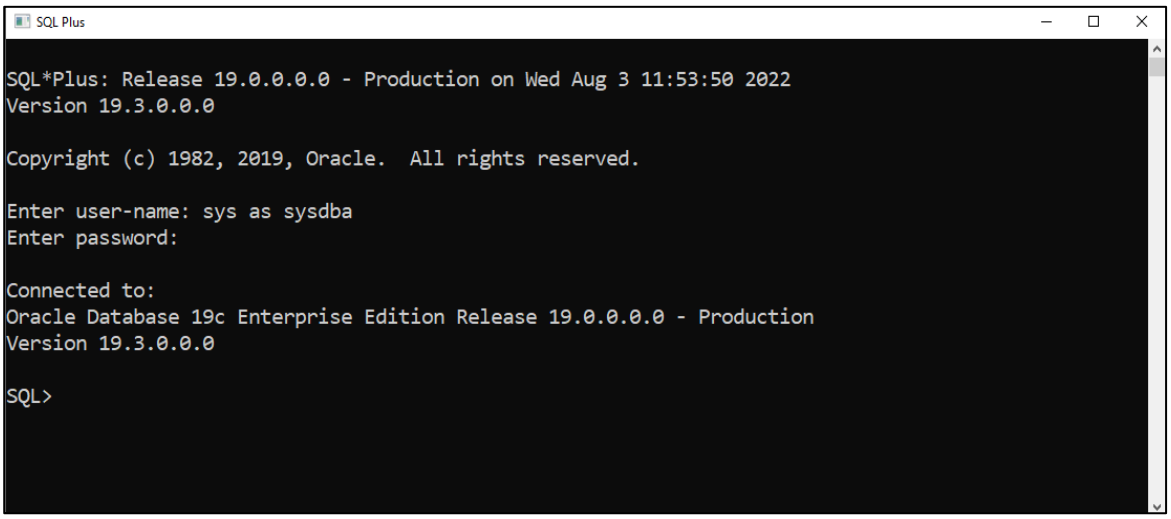
Before closing the window, make sure you copy the **"Enterprise Manager Database Express URL"**.




### Step 14: Test the database

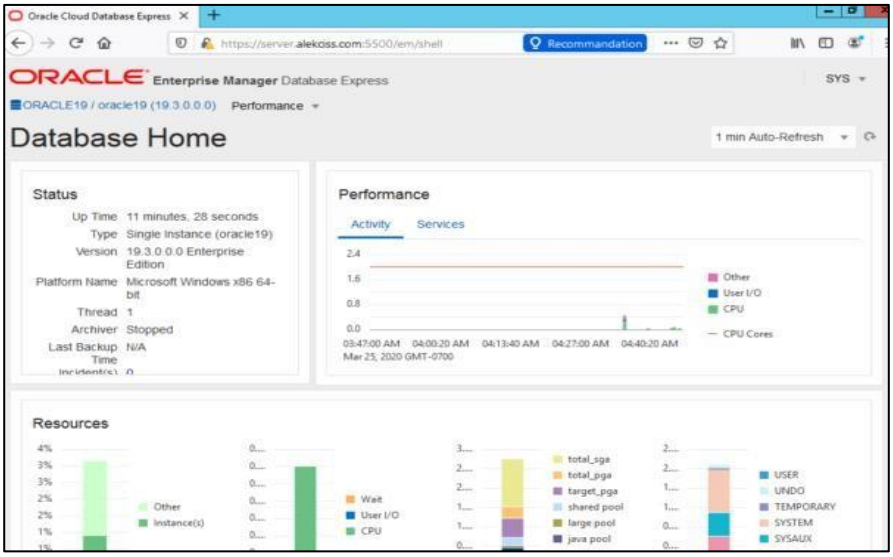
To check if your database is running properly, you can:

Start a new **SQLPlus** session with an administrative account.




or connect to EM Express with SYS administrative account.


You should see the following screen if everything is okay


Oracle Cloud Database Express
Recommandation
ORACLE Enterprise Manager Database Express
SYS
ORACLE19 / oracle19 (19.3.0.0.0)
Performance
Database Home
1 min Auto-Refresh
Status
Up Time 11 minutes, 28 seconds
Type Single Instance (oracle19)
Version 19.3.0.0.0 Enterprise Edition
Platform Name Microsoft Windows x86 64-bit
Thread 1
Archiver Stopped
Last Backup Time N/A
Performance
Activity
Services
Other
User I/O
CPU
CPU Cores
Mar 25, 2020 GMT-0700
Resources
Other
Instance(s)
Wait
User I/O
CPU
total_sga
total_pga
target_pga
shared pool
large pool
java pool
USER
UNDO
TEMPORARY
SYSTEM
SYSAUX

That's it.

## Activity 1C

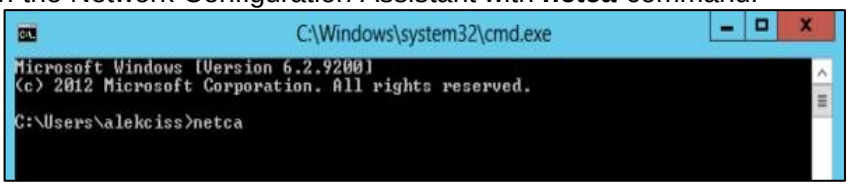

**Activity Outcome: Create a Listener in Oracle Database 19c**

A listener is a process allowing to serve a connection to a database instance via a network protocol (TCP / IP, IPC, etc.). It receives incoming client connection requests and manages the traffic of these requests to the database server. Creating a listener is then necessary if you want your users to access the database remotely.

You can create a listener in Oracle Database 19c with Network Configuration Assistant or Oracle Network Manager. These tools are available after you install Oracle Database 19c software.

**Create a listener in Oracle Database 19c with NETCA**

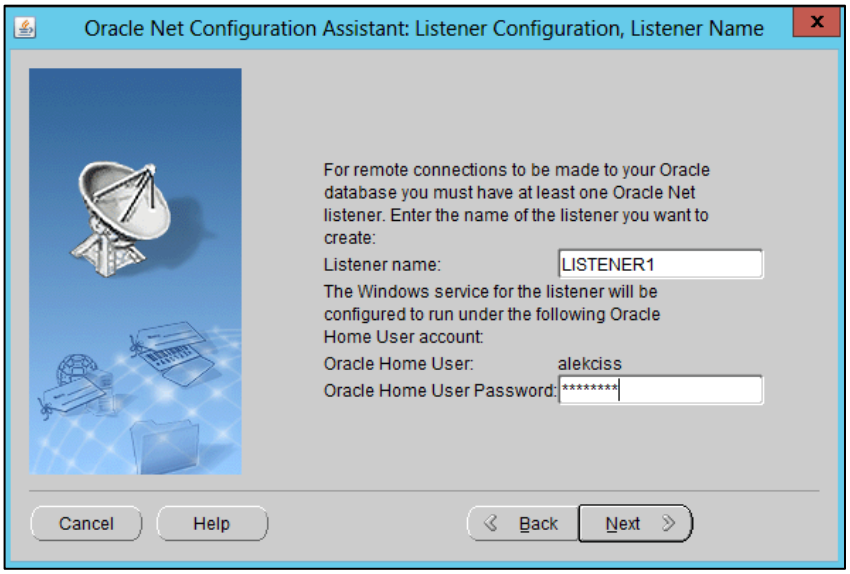
**Step 1:** Launch the Network Configuration Assistant with ***netca*** command.



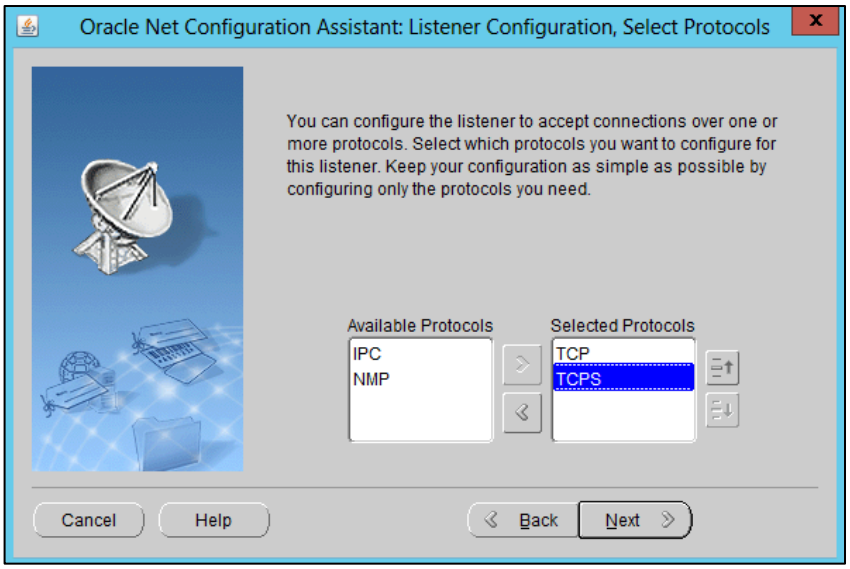

**Step 2: "Listener Configuration"** and click **"Next"**.

**Step 3:** Select "**Add**" then click "**Next**".

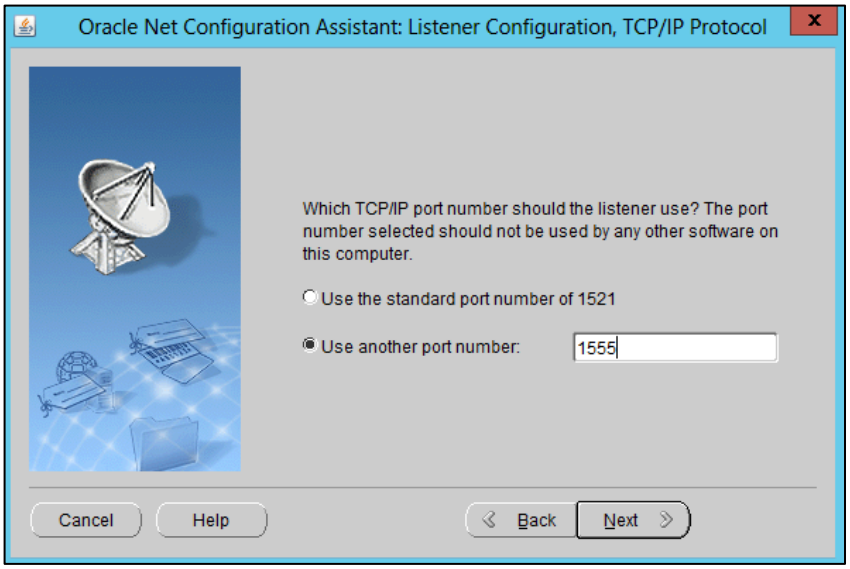

**Step 4:** Give a name to the listener to easily identify it from other configured listeners and provide Oracle Home user password.

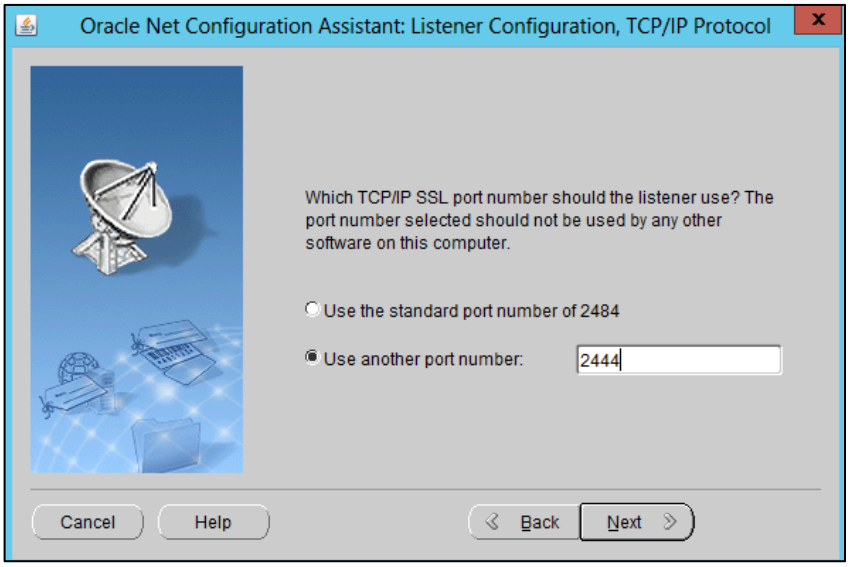


**Step 5:** Choose which protocols client applications will use to access the database. For TCP and secure TCP (TCPS), a port number is required.

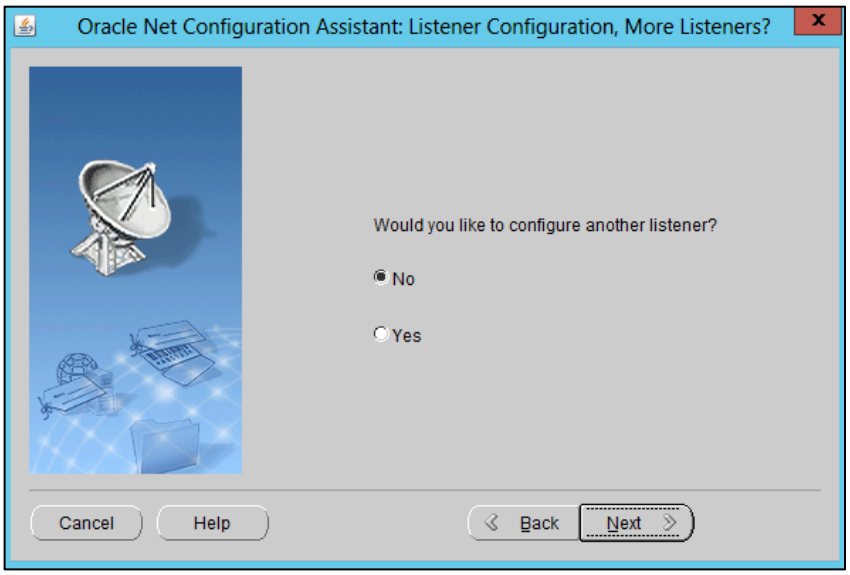



**Step 6:** Provide port numbers for TCP and TCPS protocols. Make sure port numbers are greater than 1024.





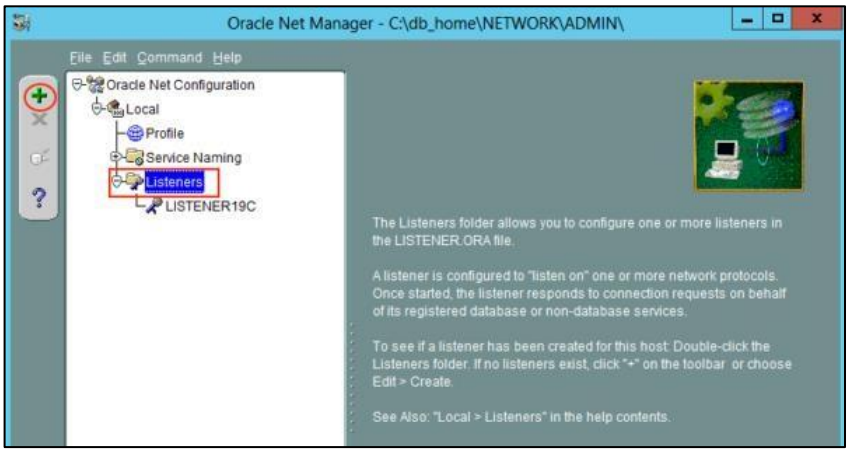

**Step 7:** Finish the listener configuration.



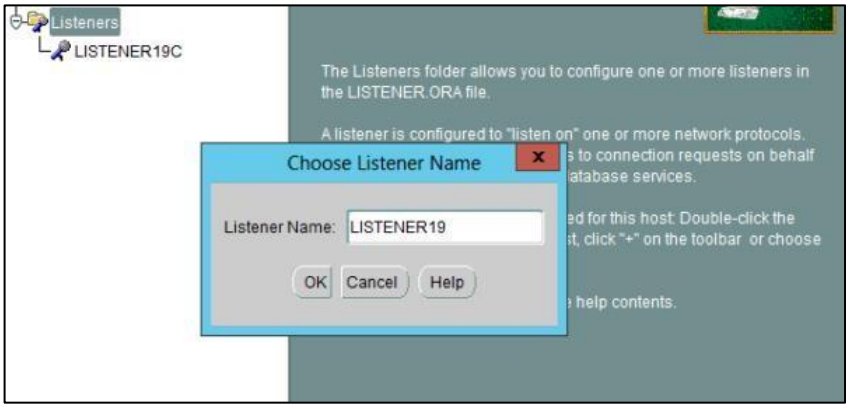
## Create a listener in Oracle Database 19c using Oracle Network Manager

To create a lister with Oracle Network Manager, you can execute the following steps.

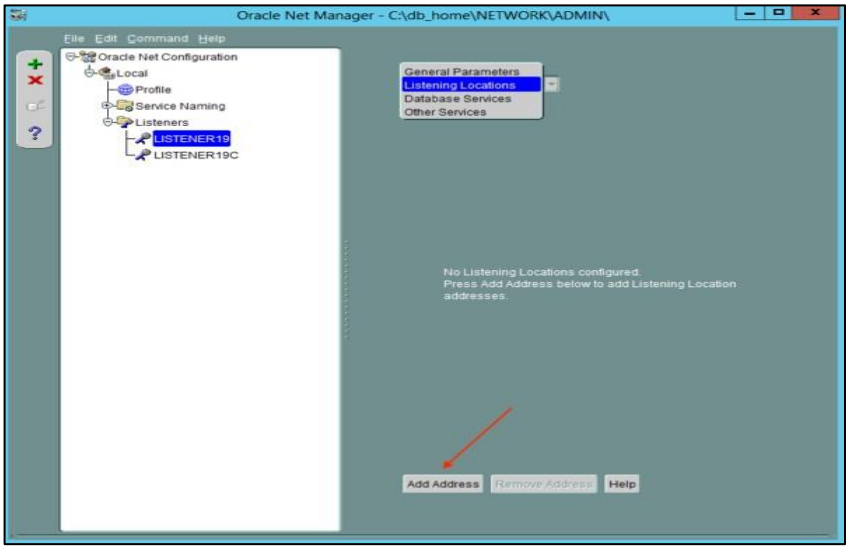
**Step 1:** Start Oracle Network Manager from Windows start menu. Then, click on **"Listeners"** and the "+" button.



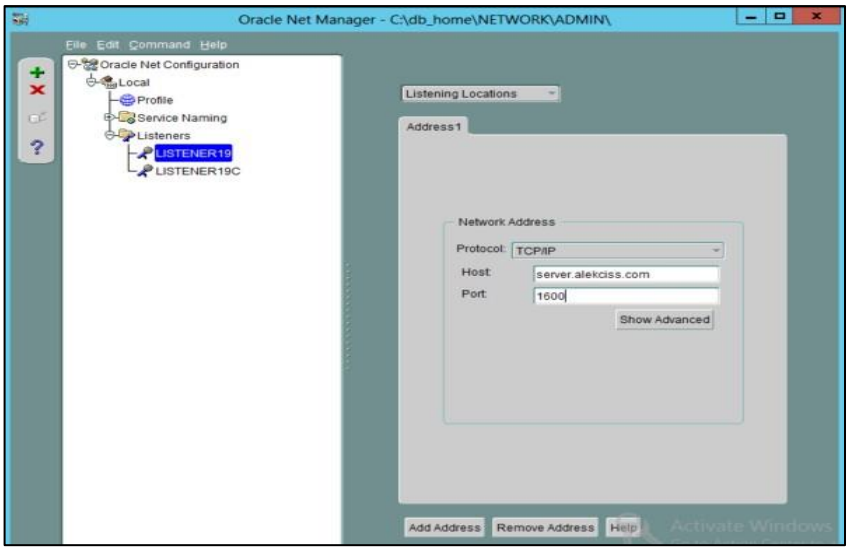

**Step 2:** Give a name to the listener and click **"OK"**.



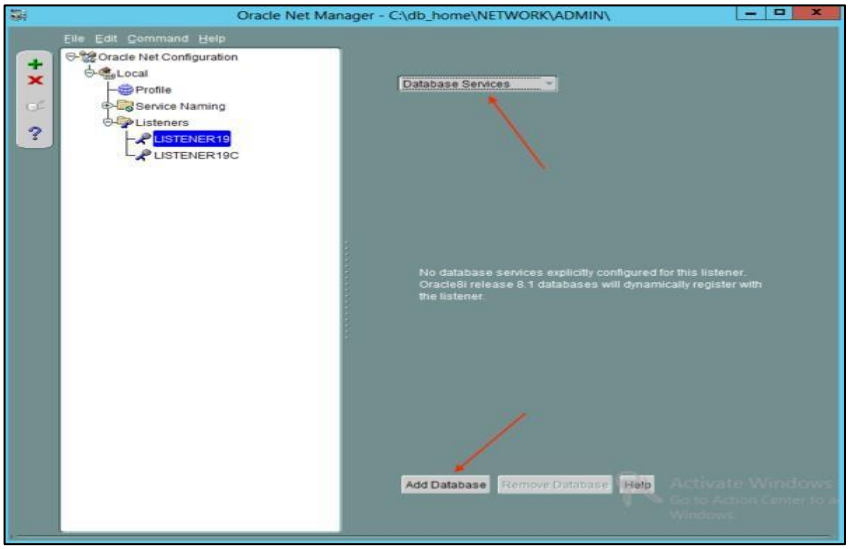
**Step 3:** Next, select **"Listening Locations"** and click on **"Add Address"** to indicate the server remote access details.



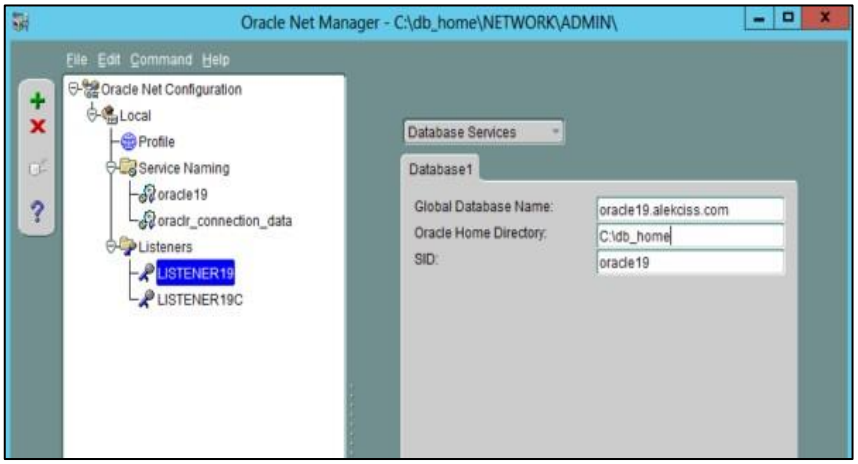

**Step 4:** Select a protocol (eg.**"TCP/IP"**) and provide the server hostname (or IP address) and a port number. You can repeat the process to add another protocol (eg. **"TCP/IP with SSL"**) with **"Add Address"** button.



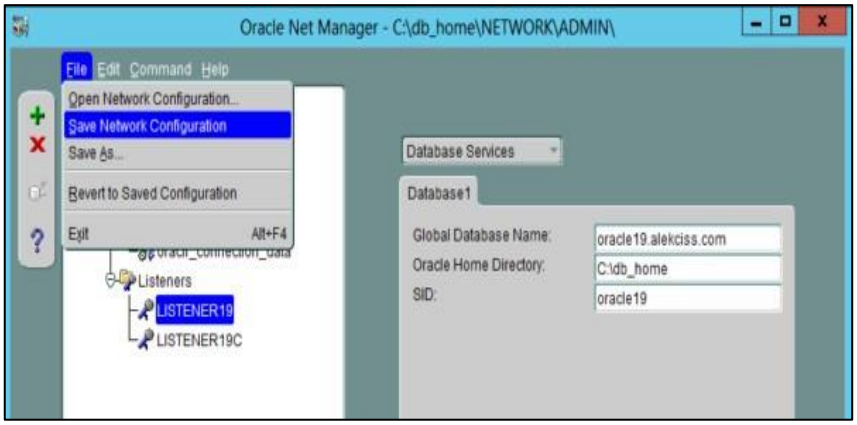
**Step 5:** Configure the database to use the listener. To do so, click on the listener name, on the left, then choose **"Database Services"**, then click **"Add Database"**.




**Step 6:** Provide the database global name, the Oracle Home Directory and the SID.



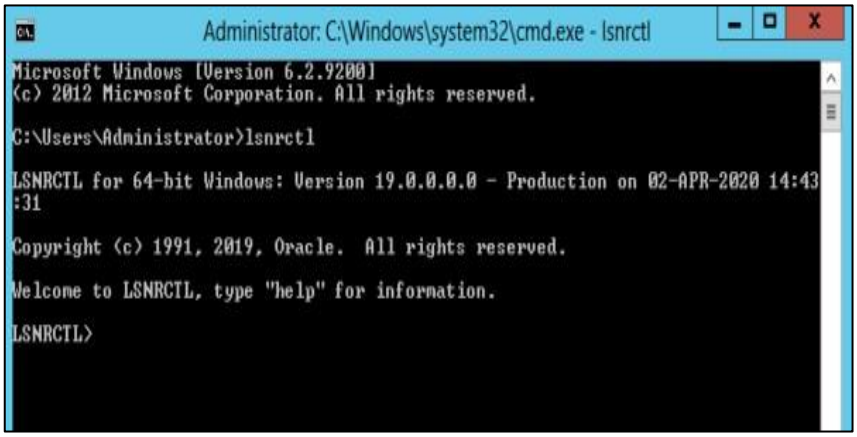
**Step 7:** Save the listener configuration and exit.



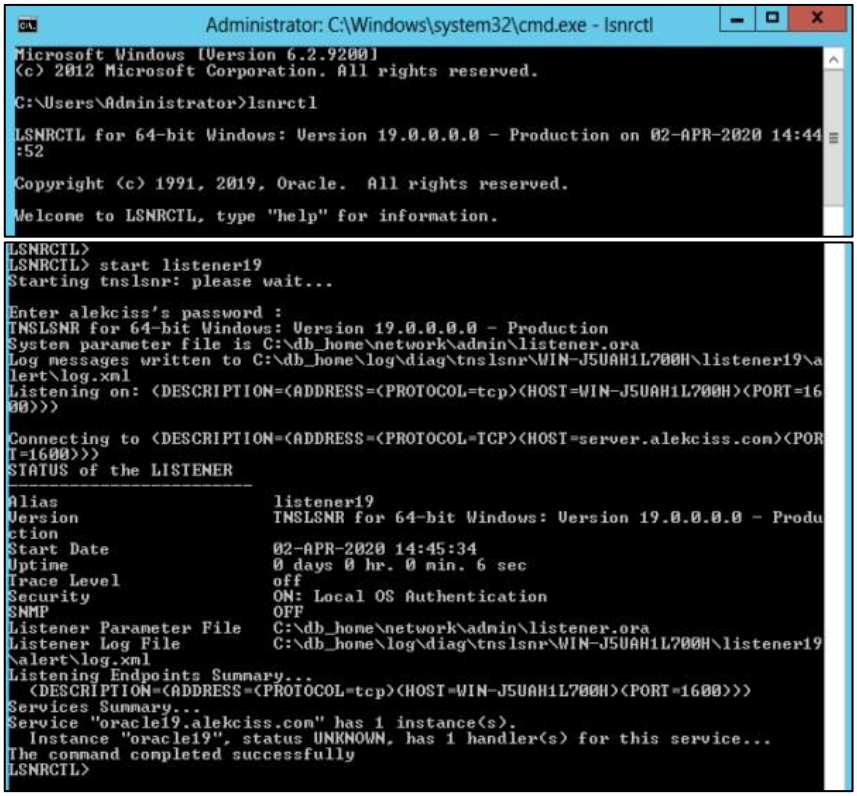

## Start a listener in Oracle Database 19c

In order to use a listener, you will need to start it. Oracle Database comes with a simple tool to control your listeners: Listener Control or LSNRCTL in short. You can use it to

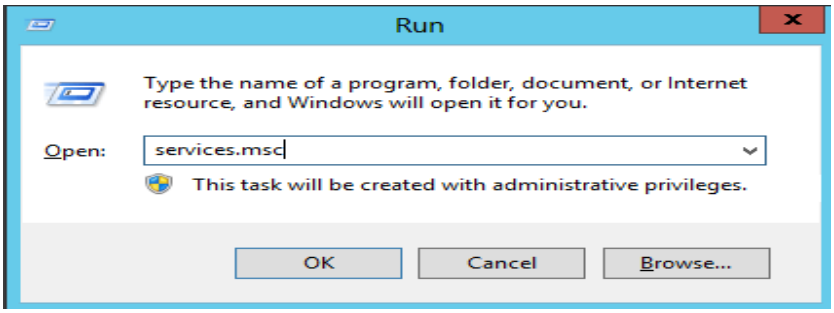
- check the status of a listener;
- start or reload a listener;
- stop a listener.



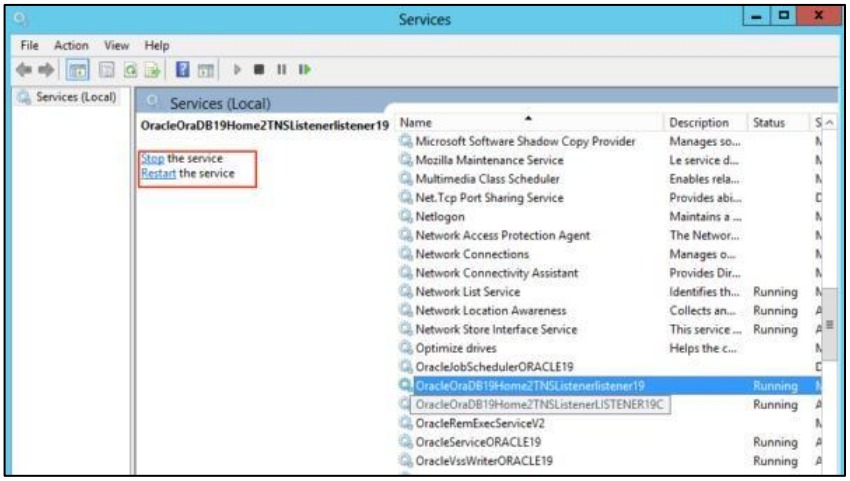

To start [stop, reload] a listener, you can run the command ***start listener_name*** [***stop listener_name, reload listener_name***].




You can also control the listener from Windows services. Use the following combination: "***Windows + R***", then type ***services.msc***.




Look for the listener and use the left panel to start, stop or reload it.


Services
File Action View Help
Services (Local)
Services (Local)
OracleOraDB19Home2TNSListenerlistener19
Stop the service
Restart the service
Name
Description
Status
Microsoft Software Shadow Copy Provider
Manages so...
Mozilla Maintenance Service
Le service d...
Multimedia Class Scheduler
Enables rela...
Net.Tcp Port Sharing Service
Provides abi...
Netlogon
Maintains a ...
Network Access Protection Agent
The Networ...
Network Connections
Manages o...
Network Connectivity Assistant
Provides Dir...
Network List Service
Identifies th...
Running
Network Location Awareness
Collects an...
Running
Network Store Interface Service
This service ...
Running
Optimize drives
Helps the c...
OracleJobSchedulerORACLE19
OracleOraDB19Home2TNSListenerlistener19
Running
OracleOraDB19Home2TNSListenerLISTENER19C
Running
OracleRemExecServiceV2
OracleServiceORACLE19
Running
OracleVssWriterORACLE19
Running

That's it.

# LAB ACTIVITY 2

# LAB ACTIVITY 2: Oracle Database Structure – Part 1

Duration: 3 Hours

## Learning Outcomes

This activity encompasses activities 2A, 2B and 2C.

By the end of this laboratory session, you should be able to:

1. Identify listener and listener configuration files
2. Connecting to an Oracle Database
3. Testing Oracle Net Connectivity

## Activity 2A


**Activity Outcome: Identify listener and listener configuration files**

i. Explain each term below:

**Oracle Net**

**Listener**

**SQL *Plus**

ii. Open the listener configuration files:

1. Got to Oracle Home (C:\db_home) or (D:\db_home)  the location where the db_home is in your computer
2. Open folder network  admin
3. Open the listener.ora file
4. Write down
   a) Host : _____
   b) Port : _____
   c) Protocol : _____
   d) Service name : _____

## Activity 2B


**Activity Outcome: Connecting to an Oracle Database**

**A. Explain each naming methods below for name resolution during connecting to an Oracle database.**

| Naming Method | Explanation |
|---|---|
| Easy connect naming | |
| Local naming | |
| Directory naming | |
| External naming | |

## B. Connecting to an Oracle Database using SQL *Plus (Command Line)

1. Open a Windows terminal (command prompt) and enter the SQL *Plus command:

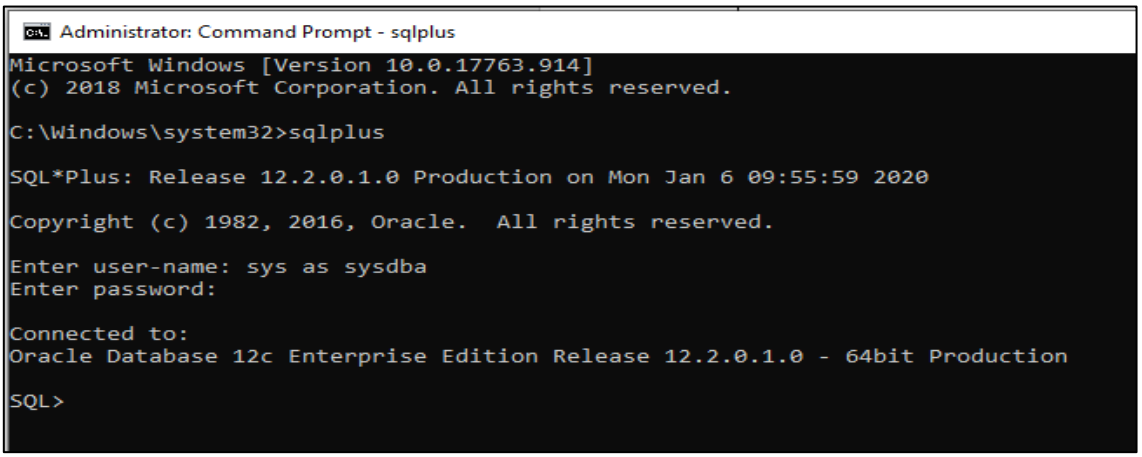
```
sqlplus
```

2. When prompted, enter your Oracle Database username and password. If you do not know your Oracle Database username and password, ask your Database Administrator.
3. Alternatively, enter the SQL*Plus command in the form:

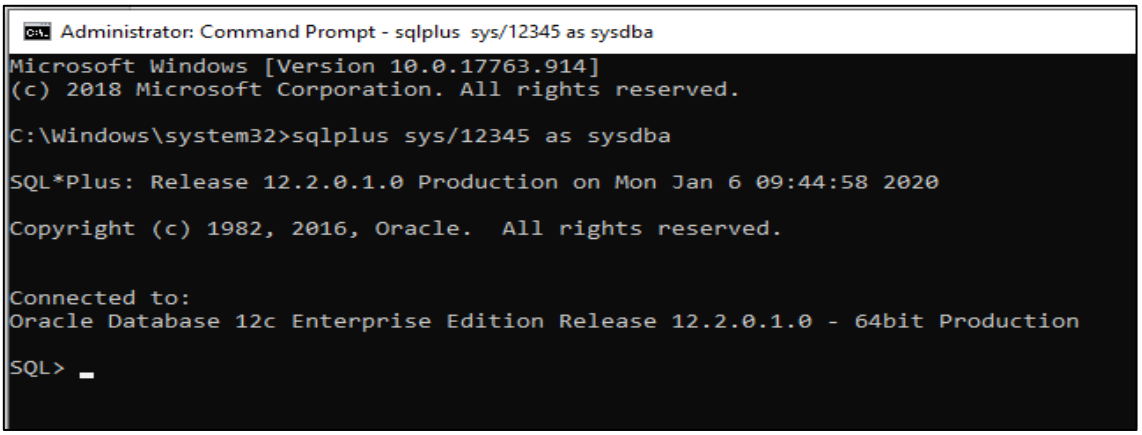
```
sqlplus username/password
```

4. SQL *Plus starts and connects to the default database.

   Now you can start entering and executing SQL, PL/SQL and SQL*Plus statements and commands at the SQL> prompt.




Or



To view current user

```
SQL > show user;
```

To view current database

```
SQL> show con_name;
```

To view other user in the database (connected as sysdba)

```
a) SQL> select username from dba_users;
b) SQL> select username from all_users;
c) SQL> select username from user_users;
```

**Task : Find the difference between dba_users, all_users and user_users.**

| dba_users | all_users | user_users |
|---|---|---|
| | | |

**C. Connecting to an Oracle Database as user HR (sample schema provided by an Oracle)**

1. Using SQL *Plus, connect to Oracle Database as **sys** user.
2. Display the current connection name / database.

```
SQL> show con_name;
```

3. At the SQL> prompt, unlock the HR account and reset its password:

```
SQL> alter user HR account unlock identified by hr;
```

The system responds:

```
User altered.
```

The **HR** account is unlocked and its password is **hr**.

While trying to unlock the HR user , if you are getting an error saying, **"user HR does not exist"**

```
SQL> alter user hr account unlock identified by hr;

ORA-01918: user 'HR' does not exist
```

Edit **tnsnames.ora** file located at **%db_home%\network\admin**

Add TNS ORCLPDB like below-highlighted block

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

Open CMD as Administrator and execute below command

```
C:\>lsnrctl reload
```

Open CMD and execute SQLPlus with below command to login as SYSDBA

```
C:\WINDOWS\system32> sqplus / as sysdba
```

Run below SQL commands to Unlock your Schema

```
SQL> ALTER SESSION SET container = ORCLPDB;
Session altered.
```

Check if the pluggable database is opened?

```
SQL> SELECT name, open_mode FROM v$pdbs;
```

To open your pluggable database Skip If Already open

```
SQL> ALTER PLUGGABLE DATABASE open;
Pluggable database altered.
```

Unlock HR user

```
SQL> alter user hr account unlock identified by hr;
User altered.
```

4. Now you can connect to Oracle Database as user HR with the password 123.

```
SQL> conn hr@orclpdb/hr
```

5. You can view the objects that belong to the HR schema by querying the static data dictionary view USER_OBJECTS.

```
SQL> select object_name, object_type
     from user_objects
     order by object_type, object_name;
```

Setting the format of results:

```
SQL> set pagesize 100
SQL> column object_name format a25
SQL> column object_type format a25
SQL> select object_name, object_type
     from user_objects
     order by object type, object_name;
```

6. View tables belongs to HR

```
SQL> select table_name from user_tables;
```

7. View employees table properties

```
SQL> describe employees
```

8. View employees table data

```
SQL> set pagesize 150
SQL> column first_name format a25
SQL> column last_name format a25
SQL> column phone_number format a20

SQL> select last_name, first_name, phone_number
   2 from employees order by last_name;
```

## Activity 2C


**Activity Outcome: Testing Oracle Net connectivity**

i. Testing the Oracle Net connectivity

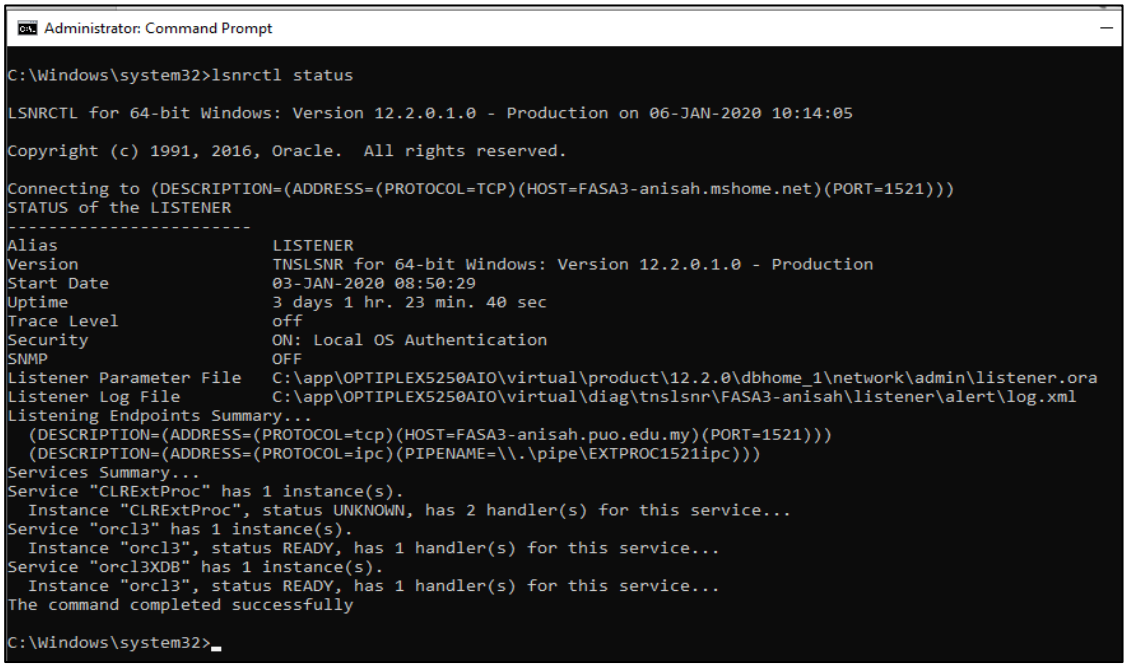
Open a Windows terminal (command prompt) and enter the command:

```
tnsping orcl
```

ii. Check the status of listener

Open a Windows terminal (command prompt) and enter the command:

```
lsnrctl status
```




iii. Start / Stop the listener

Open a Windows terminal (command prompt) and enter the command:

```
lsnrctl start

lsnrctl stop
```

# LAB ACTIVITY 2: Oracle Database Structure – Part 2

Duration: 3 Hours

## Learning Outcomes

This activity encompasses activities 2D and 2E.

By the end of this laboratory session, you should be able to:

1. Identify Memory Structure Component and their specific functions
2. Identify the SQL syntax use to access memory component structure

## Activity 2D


**Activity Outcome: Identify memory structure component and their specific functions**

i. List and explain briefly of two memory structures available in instance:

ii. State one major differences for both memory structure listed above:

| Memory Structure Type | | |
|---|---|---|
| Differences | | |

iii. What is the benefit of having more than ONE instance connected to the database?

iv. What is the reason of only a copy of data block from the datafiles are transfer to the database buffer cache to be access and manipulate during user process?

v. Explain briefly below terms :

| Terms | Explanation |
|---|---|
| LRU | |
| UGA | |
| Touch Count | |
| Data Block | |
| Cache Hit | |
| Cache Miss | |
| Circular Buffer | |

## Activity 2E


**Activity Outcome: Identify the SQL syntax use to access memory component structure**

State the SQL statement use to access information of below memory from instance.

| Memory | SQL statement |
|---|---|
| Size of shared pool | |
| Size of redo log buffer | |
| Size of large pool | |
| Size of java pool | |
| Size of stream pool | |
| Numbers of buffer in database buffer cache | |
| Size of buffer in database buffer cache | |

# LAB ACTIVITY 2: Oracle Database Structure – Part 3

Duration: 3 Hours

## Learning Outcomes

This activity encompasses activities 2F, 2G and 2H.

By the end of this laboratory session, you should be able to:

1. Identify Extents, Segments and Tablespaces.
2. Identify memory Management.
3. Identify Data Dictionary.

## Activity 2F


**Activity Outcome: Identify Extents, Segments and Tablespaces.**

Type the following SELECT statement:

```
SQL > conn sys/12345 as sysdba
SQL > SELECT DISTINCT segment_type FROM dba_segments;
```

a) List the most common types of segments.

b) Do you see the most common types of segments listed in your result set?

c) Briefly describe the kind of information each type of segment stores or is used for.

| | |
|---|---|
| Data segments | |
| Index segments | |
| Rollback segments (Undo segments) | |
| Temporary segments | |

d) To view list of used blocks and empty blocks for the user tables, type the following command:

```
SQL > conn HR@orclpdb/hr

SQL > SELECT blocks as BLOCKS_USED, empty_blocks
      FROM user_tables
      WHERE table_name='DEPARTMENTS';
```

e) To view used space and free space for the user segments, type the following command:

(to view all the segment name in user segment in MB)

```
SQL > SELECT segment_name from user_segments;
```

(to view the used space in database)

```
SQL > SELECT sum(bytes)/1024/1024 from user_segments;
```

(to view the free space in database in MB)

```
SQL > SELECT sum(bytes)/1024/1024 from user_free_space;
```

f) A tablespace may contain multiple extents allocated to segments and one or more free extents. What is a free extents?

g) How does an extent become free?

Type the following SELECT statement:

```
SQL > conn sys/12345 as sysdba
SQL > SELECT tablespace_name, status FROM dba_tablespaces;
```

h) List the tablespaces you see, along with their current status.

## Activity 2G


**Activity Outcome: Identify Memory Management**

**AUTOMATIC MEMORY MANAGEMENT (AMM) : MEMORY TARGET & MEMORY MAX TARGET**

a) To view the information about parameter target, type the following command:

```
SQL > show parameter target
```

b) To view the information about parameter memory_target, type the following command:

```
SQL > show parameter memory_target
```

c) To view the information about parameter memory_max_target, type the following command:

```
SQL > show parameter memory_max_target
```

We can increase the value of MEMORY_TARGET parameter to MEMORY_MAX_TARGET, but if we will try to increase it more from the size of max parameter it will throw error.

**SYSTEM GLOBAL AREA (SGA)**

The view (V$sga) display summary information about the system global area (SGA).

a) To view the information available for sga, type the following command:

```
SQL > desc v$sga
```

b) To view name and value (in MB) for each parameter in SGA, type the following command:

```
SQL > select name, value/1024/1024 Size_MB from v$sga;
```

c) To view the total size of SGA in MB, type the following command:

```
SQL > select sum(value)/1024/1024 Total_size_In_MB
    2 from V$sga;
```

**FREE SPACE**

The View (V$SGASTAT) displays detailed information on free space in the system global area (SGA).

a) To view the information available for sga, type the following command:

```
SQL > desc v$sgastat
```

Let's have a short explanation of all column of the v$sgastat parameter:

i) POOL - Designates the pool in which the memory in NAME resides:

*Shared pool = Memory is allocated from the shared pool

*Large pool = Memory is allocated from the large pool

*Java pool = Memory is allocated from the Java pool

*Stream pool = Memory is allocated from the stream pool

ii) NAME - SGA component name

iii) BYTES - The Memory size in bytes

b) To view in detailed view sga free memory space, type the following command:

```
SQL > select pool, name,
      round(bytes/1024/1024,0) free_memory_in_MB
      from v$sgastat where name like '%free memory%';
```

c) To view total amount of free size in SGA, type the following command:

```
SQL > select sum(bytes/1024/1024) Free_Memory_In_MB
    2 from V$sgastat where Name Like '%free memory%';
```

**PROGRAM GLOBAL AREA (PGA)**

The view (V$process) display summary information about the program global area (SGA).

a) To view the information available for sga, type the following command:

```
SQL > desc v$process
```

b) To view maximum size of PGA, type the following command:

```
SQL > select sum(pga_max_mem)/1024/1024 "TOTAL MAX PGA (MB)"
    2 from v$process;
```

c) To view more detailed breakdown of PGA memory usage, type the following command:

```
SQL > SELECT spid, program,
    2 pga_max_mem/1024/1024 max,
      pga_used_mem/1024/1024 used,
      pga_alloc_mem/1024/1024 alloc,
      pga_freeable_mem/1024/1024 free
      from v$process;
```

d) Format the display for the output,

```
SQL > set pagesize 150
SQL > column spid format a15
SQL > column program format a20
```

Review the detailed breakdown of PGA memory usage by using command form c)

## Activity 2H


**Activity Outcome: Identify Data Dictionary Views**

Querying from the data dictionary views including user_tables, all_tables, and dba_tables.

```
SQL > conn HR@orclpdb/hr

SQL > select table_name from user_tables;
```

a) What is the result?

```
SQL > select table_name from all_tables;
```

b) What is the result?

```
SQL > select table_name from dba_tables;
```

c) What is the result?

d) Why HR cannot view the dba_tables view?

Now, type this coding.

```
SQL > conn sys/12345 as sysdba;

SQL > grant select on dba_tables to HR;

SQL > conn HR/hr

SQL > select table_name from dba_tables;
```

e) What is the result?

f) Which data dictionary view can be used to find the names of all tables in the database?

a. USER_TABLES
b. ALL_TABLES
c. DBA_TABLES
d. ANY_TABLES

g) Find differences between data dictionary views and dynamic performance views.

| Data dictionary views | Dynamic performance views |
|---|---|
| | |

# LAB ACTIVITY 3

# LAB ACTIVITY 3: Managing Tables, Indexes and Constraints – Part 1

Duration: 3 Hours

## Learning Outcomes

This activity encompasses activities 3A, 3B, 3C and 3D.

By the end of this laboratory session, you should be able to:

1. View database storage information.
2. Managing tablespace.
3. Managing Online Redo Log.
4. Managing undo data.

## Activity 3A


**Activity Outcome: View database storage information.**

1. Run the command prompt and open SQL Plus platform.

```
C:\ …>sqlplus sys/12345 as sysdba
```

2. In the SQL environment important data dictionary table is called **dba_data_file**

```
SQL > desc dba_data_files;
```

3. Before we proceed, let us set the output view layout first:

```
SQL > SET LINESIZE 150;
SQL > COLUMN TABLESPACE_NAME FORMAT A30;
SQL > COLUMN FILE_NAME FORMAT A50;
```

4. Now let us view the content

```
SQL > select tablespace_name, file_name, bytes
    2 from dba_data_files;
```

5. Change the unit into Mbyte for a better view of the size

```
SQL > select tablespace_name, file_name, bytes/1024/1024
    2 from dba_data_files;
```

## Activity 3B


**Activity Outcome: Managing tablespace**

### A. CREATE TABLESPACE

1. Create new tablespace

```
SQL > create tablespace tbs1
    2 datafile 'D:\app\P300\oradata\orcl\tbs1.dbf' size 1m;
```

**(Please take note that path for directory may vary, depends on your machine)**

2. Then, run this command again.

```
SQL > select tablespace_name, file_name, bytes/1024/1024
    2 from dba_data_file;
```

3. Observe the output. Can you see your data file?

4. Now, let us check the availability of the free space.

```
SQL > select tablespace_name, bytes
      from dba_free_space
      where tablespace_name='TBS1';
```

5. Check again, the size of your tablespace occupied as the table created in step above:

```
SQL > select tablespace_name, file_name, bytes
    2 from dba_data_files;
```

6. Create a table for your tablespace.

```
SQL > create table mytable (id int) tablespace TBS1;
```

Observe the output.

## B. MODIFYING THE TABLESPACE

1. We are going increase the size of existing tablespace.

```
SQL > alter database
      datafile 'D:\app\P300\oradata\orcl\tbs1.dbf'
      resize 10m;
```

2. Then, run this command again.

```
SQL > select tablespace_name, file_name, bytes/1024/1024
    2 from dba_data_file;
```

Observe the output.

**EXERCISE**

Scan this QR code.

Then, try to follow steps to create tablespace.
Snap screenshot for each step, paste on MS Word and submit the steps to your lecturer as
**Lab3Part1_AlterTablespace.docx** document.

**Please take note you have to put your directory path in front of your tablespace name. For example:**

```
1 CREATE TABLESPACE tbs2
2     DATAFILE 'tbs2_data.dbf'
3     SIZE 5m;
```

You should write like this:

```
SQL> CREATE TABLESPACE tbs2
DATAFILE 'D:\app\P300\oradata\orcl\tbs2_data.dbf'
SIZE 5m;
```

## C. DROP THE TABLESPACE

1. Before we proceed, let us delete the existing tablespace we've created

```
SQL> drop tablespace tbs1 including contents and datafiles;
```

**EXERCISE**

Scan this QR code.

Then, try to follow steps to create tablespace.
Snap screenshot for each steps, paste on MS Word and submit the steps to your lecturer as **Lab3Part1_DropTablespace.docx** document.

## Activity 3C


**Activity Outcome: Managing Online Redo Log**

1. Run the command prompt and open SQL Plus platform.

```
C:\ …>sqlplus/12345 as sysdba
```

2. In the SQL environment check the existing logfile

```
SQL > select * from v$logfile;
```

3. Now let us view together with the size

```
SQL > select * from v$log;
```

4. Change the unit into Mbyte for a better view of the size

```
SQL > select group#, bytes/1024/1024, status from v$log;
```

5. Edit the existing logfile to have TWO logfile in group 1.

```
SQL > alter database add logfile member
    2 'D:\app\....\oradata\orcl\redo01b.log' to group 1;
```

6. In the SQL environment, check again the existing logfile

```
SQL > set wrap off
SQL > select * from v$logfile;
```

7. Observe the output. Do you notice the status to be INVALID? Why?

8. Now, we will change the status from invalid to valid

```
SQL > alter system switch logfile;
```

Run this SQL statement for three times

9. View your status after execution

```
SQL > select * from v$logfile;
```

10. Let us now delete one group of logfile. This group must be in an inactive status. Check our logfile status.

```
SQL > select * from v$log;
```

11. Choose the inactive group to delete.

```
SQL > alter database drop logfile group 3;
```

12. View your status after execution

```
SQL > select * from v$logfile;
```

13. Let us now add new logfile. Run this SQL statement

```
SQL > alter database add logfile member
    2 'D:<location>\orcl\redo02b.log' to group 2;
SQL > alter database add logfile member
    2'D:<location>\orcl\redo02c.log' to group 2;
SQL > alter database add logfile member
    2 'D:<location>\orcl\redo01c.log' to group 1;
```

14. Check output after execution.

```
SQL > select * from v$logfile order by group#;
```

Observe the output.

## Activity 3D


**Activity Outcome: Managing Undo Data**

1. Run the command prompt and open SQL Plus platform.

```
C:\ …>sqlplus/12345 as sysdba
```

2. In the SQL environment check the existing tablespace (This is to track either your tablespace of UNDOTBS1 already exist)

```
SQL > select tablespace_name, contents, status
    2 from dba_tablespaces;
```

3. Now let us view the size of this tablespace

```
SQL > select tablespace_name, file_name, bytes/1024/1024
    2 from dba_data_files;
```

4. Now let view parameter that is available available in undo management

```
SQL > show parameter undo;
```

5. Adding undo tablespace to the existing database.

```
SQL > create undo tablespace undotbs2
      datafile 'D: <location>\oradata\orcl\undotbs2a.dbf'
      size 5m reuse autoextend on;
```

6. Check the output.

```
SQL > select tablespace_name, contents, status
    2 from dba_tablespaces;
```

7. Check to see which undo tablespace is the active one :

```
SQL > show parameter undo;
```

8. Change the active tablespace to undotbs2

```
SQL > alter system set undo_tablespace=undotbs2;
```

9. Check again the active undo tablespace after execution

```
SQL > show parameter undo;
```

10. Now to proof that only one undo tablespace can be active at one time:

```
SQL > select segment_name, owner, tablespace_name, status
    2 from dba_rollback_segs;
```

Observe the output.

11. Managing the retention period in undo tablespace.

```
SQL > show parameter undo;
```

12. Change the retention value

```
SQL > alter system set undo_retention=2400;
```

Observe the output.

13. Managing the retention period in undo tablespace to be guarantee. Check the guarantee status:

```
SQL > select tablespace_name, retention from dba_tablespaces;
```

14. Change the guarantee status for the active undo tablespace

```
SQL > alter tablespace undotbs2 retention guarantee;
```

Observe the output. Note: the retention guarantees ONLY applicable to the UNDO tablespace.

# LAB ACTIVITY 3: Managing Tables, Indexes and Constraints – Part 2

Duration: 3 Hours

## Learning Outcomes

This activity encompasses activities 3E, 3F and 3G.

By the end of this laboratory session, you should be able to:

1. Managing Schema objects.
2. Managing Tables.
3. Managing Views.

## Activity 3E


**Activity Outcome: Managing schema objects**

1. Log in as **`sys/12345 as sysdba`**

2. Display all user

```
SQL > SELECT username FROM dba_users ORDER BY username;
```

   Observe the output.

3. Log in as **`HR.`** View all object under **`HR`** schema

```
SQL > SELECT username FROM user_users;
SQL > set pagesize 50
SQL > column object_name format A25
SQL > column object_type format A25
SQL > select object_name, object_type from user_objects;
```

   All objects for HR schema will be display. Observe the output.

4. Log in as **`sys/12345 as sysdba`**

5. View quantity of object under **`sys`** schema

```
SQL > select count(object_name) from user_objects;
```

6. View table employees under **`HR`** schema

```
SQL > select employee_id, first_name from HR.employees;
```

   Observe the output.

## Activity 3F


**Activity Outcome: Managing Tables**

1. Log in as **HR**

2. View all objects under **HR** schema

```
SQL > set pagesize 50
SQL > column object_name format A25
SQL > column object_type format A25
SQL > select object_name, object_type
      from user_objects
      where object_type = 'TABLE';
```

   Observe the output.

3. Create table **Student**

```
SQL > create table student (s_id int, s_name varchar(50),
SQL > s_class varchar(10),primary key(s_id));
SQL > desc student
SQL > select object_name, object_type from user_objects;
```

   Observe the output. Student appear as a table in HR objects.

4. Alter table **Student**

```
SQL > alter table student add age int;
SQL > desc student
```

   Observe the output. Try explore alter + modify and alter + drop by yourself.

5. Insert data into table **Student**

```
SQL > insert into student values (1, 'Amir', '1A', 18);
```

   Insert another 5 dummy data.

6. View data for table **Student**

```
SQL > select * from student;
```

7. View data from sys schema. Log in as **sys/12345 as sysdba**.

```
SQL > select * from HR.student;
```

8. Delete data from table student and drop table student.

```
SQL > delete from HR.student;
```

```
SQL > select * from HR.student;
```

What is the result?

```
SQL > desc HR.student;
```

What is the result?

```
SQL > drop table HR.student;
SQL > desc HR.student;
```

What is the result?

9. Create table as select.

```
SQL > create table HR.emp1 as select * from HR.employees;
SQL > desc HR.emp1;
SQL > select count(*) from HR.emp1;
```

What is the result?

## Activity 3G


**Activity Outcome: Managing Views**

1. Log in as **HR**. View all objects under **HR** schema

```
SQL > set pagesize 50
SQL > column object_name format A25
SQL > column object_type format A25
SQL > select object_name, object_type from user_objects;
```

   Observe the output.

2. Create dummy table

```
SQL > create table emp2 as select * from employees;
```

3. Create view

```
SQL > create view emp_view
      as select employee_id, first_name, job_id
      from emp2;
SQL > Set pagesize 150
SQL > select * from emp_view;
```

   What is the result?

```
SQL > select * from emp_view where job_id="SH_CLERK';
```

   What is the result?

4. Create dummy table

```
SQL > create table dep2 as select * from departments;
```

5. Create or replace view emp_view

```
SQL > create or replace view emp_view as select
SQL > e.employee_id, e.first_name, d.department_name
SQL > from emp2 e, dep2 d
SQL > where e.department_id = d.department_id;

SQL > select * from emp_view;
```

   What is the result?

6. Drop view

```
SQL > drop view emp_view;
```

# LAB ACTIVITY 3: Managing Tables, Indexes and Constraints – Part 3

Duration: 3 Hours

## Learning Outcomes

This activity encompasses activities 3H, 3I and 3J.

By the end of this laboratory session, you should be able to:

1. Managing Indexes.
2. Managing Sequences.
3. Managing Synonyms.

## Activity 3H


**Activity Outcome: Managing Indexes**

An index is a performance-tuning method of allowing faster retrieval of records. An index creates an entry for each value that appears in the indexed columns. By default, Oracle creates B-tree indexes.

1. Log in sqlplus as **HR**

2. Create dummy table name **dep**

```
SQL > create table dep as select * from departments;
```

3. Create index for **department_id**

```
SQL > create index dep_idx on dep(department_id);
```

4. View the details without synonyms

```
SQL > select object_name, object_type from user_objects;
```

Now, you can see the index **dep_idx** as one of your schema object.

5. Alter the index name

```
SQL > alter index dep_idx rename to dep_1_idx;
SQL > select object_name, object_type from user_objects;
```

Now, you can see the index **dep_1_idx** as one of your schema object.

6. Drop the index

```
SQL > drop index dep_1_idx;
SQL > select object_name, object_type from user_objects;
```

Now, you cannot see the index **dep_1_idx** as one of your schema object.

7. Klik this link to know how indexes used in your database.

**https://youtu.be/fsG1XaZEa78**

## Activity 3I


**Activity Outcome: Managing Sequences**

In Oracle, you can create an auto number field by using sequences. A sequence is an object in Oracle that is used to generate a number sequence. This can be useful when you need to create a unique number to act as a primary key.

1. Log in sqlplus as **HR**

2. Create table **info**

```
SQL > create table info
SQL > (id int, name varchar(100),primary key(id));
```

3. Create sequence name **info_seq**

```
SQL > create sequence info_seq
SQL >  minvalue  1
SQL > start with 1
SQL > increment by 1
SQL > cache 20;
SQL > set pagesize 50
SQL > column object_name format A25
SQL > column object_type format A10
SQL > select object_name, object_type from user_objects;
```

Now, you can see the sequence **info_seq** as one of your schema object.

4. Insert data into info table by using sequencs **info_seq**

```
SQL > insert into info values (info_seq.nextval, 'Aina');
```

(repeat this steps for 5 times)

5. View the results

```
SQL > select * from info;
```

Observe the output.

6. Alter the sequence

```
SQL > alter sequence info_seq increment by 10;
SQL > insert into info values (info_seq.nextval, 'Aina');
```

(repeat this steps for 5 times)

```
SQL > select * from info;
```

Observe the output.

7. Drop the sequence

```
SQL > drop sequence info_seq;
SQL > select object_name, object_type from user_objects;
```

Now, you cannot see the sequence **info_seq** as one of your schema object.

## Exercise:

Create new dummy table and create sequence for the table. The sequences will start from 2 and end at 10, increment by 2. Put the screenshot as an output

## Activity 3J


**Activity Outcome: Managing Synonyms**

A synonym is an alternative name for objects such as tables, views, sequences, stored procedures, and other database objects. You generally use synonyms when you are granting access to an object from another schema and you don't want the users to have to worry about knowing which schema owns the object. To rename column, we can use alias (as) in select statements.

1. Log in sqlplus as **HR**

2. Create dummy table name **emp**

```
SQL > create table emp as select * from employees;
```

3. View the details without alias

```
SQL > select employee_id, first_name, job_id from emp;
```

4. Display details with alias (new name for column)

```
SQL > select employee_id as EMPLOYEE_ID,
SQL > first_name as EMPLOYEE_NAME,
SQL > job_id as EMPLOYEE_JOB from emp;
```

   Observe the output.

5. Login as **sys/12345 as sysdba**. Create synonym for table **emp** in **HR** schemas

```
SQL > create synonym e for hr.emp;
```

6. View the synonym

```
SQL > select object_name, object_type
SQL > from all_objects where object_name = 'E';
```

   Now, you can see the synonym **e** as one of your schema objects.

7. Display details in table emp in HR schema using synonym. Observe the output.

```
SQL > select * from e;
```

8. Drop synonym

```
SQL > drop synonym e;
SQL > select object_name, object_type
SQL > from all_objects where object_name = 'E';
```

   Now, you cannot see the synonym **e** as one of your schema objects.

# LAB ACTIVITY 4

# LAB ACTIVITY 4: Creating User – Part 1

Duration: 3 Hours

## Learning Outcomes

This activity encompasses activities 4A, 4B and 4C.

By the end of this laboratory session, you should be able to:

1. Create and View database user information.
2. Create privilege to a specific user.
3. Grant All Privilege to user.

The **CREATE USER** statement allows you to create a new database user which you can use to log in to the Oracle database.

The basic syntax of the CREATE USER statement is as follows:

```
CREATE USER username
    IDENTIFIED BY password
    [DEFAULT TABLESPACE tablespace]
    [QUOTA {size | UNLIMITED} ON tablespace]
    [PROFILE profile]
    [PASSWORD EXPIRE]
    [ACCOUNT {LOCK | UNLOCK}];
```

Where;

- username - Specify the name of the user to be created
- password - Specify a password for the local user to use to log on to the database
- tablespace - Specify the tablespace of the objects such as tables and views that the user will create
- quota - Specify the maximum of space in the tablespace that the user can use
- profile - Assign a profile to a newly created user. If you skip this clause, Oracle will assign the DEFAULT profile to the user.
- password expire - Use the PASSWORD EXPIRE if you want to force the user to change the password for the first time the user logs in to the database.
- Account {lock | unlock} - Use ACCOUNT LOCK if you want to lock user and disable access.

## Activity 4A


**Activity Outcome: Create and View database user information.**

1. Run the command prompt and open SQL*Plus platform.

```
C:\ …>sqlplus sys/12345 as sysdba
```

2. In the SQL environment important data dictionary table is called dba_users;

```
SQL > desc dba_users;
```

3. Before we proceed, let us set the output view layout first

```
SQL > SET LINESIZE 200
SQL > COLUMN USERNAME FORMAT A30
SQL > COLUMN DEFAULT_TABLESPACE FORMAT A30
SQL > COLUMN PROFILE format A30
SQL > COLUMN AUTHENTICATION_TYPE format A15
```

4. Now let us view the list of users with the OPEN status

```
SQL > SELECT username, default_tablespace, profile,
      authentication_type FROM dba_users
      WHERE account_status = 'OPEN';
```

5. Observe the output.

6. Create a new local user named **johnny** with the password **johnny**

```
SQL > CREATE USER johnny IDENTIFIED BY johnny;
```

7. Again, view the list of users. Noted that, the new user (johnny) has been added into the list.

8. Let us use the johnny account to log in the database

```
SQL > conn johnny/johnny
```

9. Noted that, Oracle issued the following error:

```
ERROR: ORA-01045:
user JOHNNY lacks CREATE SESSION privilege; logon denied
Warning: You are no longer connected to ORACLE.
```

10. To enable the user john to log in, you need to grant the CREATE SESSION system privilege to the user johnny by using the following statement:

```
SQL> conn sys/12345 as sysdba
SQL> GRANT CREATE SESSION TO johnny;
```

Now, the user johnny should be able to log in the database.

## Activity 4B


**Activity Outcome: Create privileges to a specific user**

### A. GRANT SYSTEM AND OBJECT PRIVILEGES TO A USER

1. Launch SQL*Plus and log in to the Oracle database using the user johnny.
   Note that we assigned the user john the CREATE SESSION system privilege, so it should be able to log in.

```
SQL > conn johnny/johnny
```

2. Use the user johnny to log in to the Oracle Database and create a new table

```
SQL > CREATE TABLE t1(id NUMBER PRIMARY KEY);
```

   Observe the output. Noted that, Oracle issued the following error:

```
CREATE TABLE t1(id NUMBER PRIMARY KEY)
*
ERROR at line 1:
ORA-01031: insufficient privileges
```

3. So, how to overcome the issues? Yes, we need to grant CREATE TABLE system privilege to the user johnny by using the following statement:

```
SQL> conn sys/12345 as sysdba
Connected.
SQL> GRANT CREATE TABLE TO johnny;

Grant succeeded.
```

4. Now, let us connect as user johnny and then create new table t1 (refer to step 2). Noted that table created successfully.

5. The following statement shows the privileges of the current user:

```
SQL> SELECT * FROM session_privs;
```

   Observe the output. It will list out the privilege for user johnny.

6. Use the user johnny to insert a new row into the t1 table:

```
SQL> INSERT INTO t1(id) VALUES(10);
```

Observe the output. Noted that Oracle issued an error **ORA-01950: no privileges on tablespace 'USERS'**. Explain the issue.

This is because the user johnny has a quota of zero on the USERS tablespace.

To fix this, you use the ALTER USER command to change the quota of the user johnny on the USERS tablespace:

```
SQL > conn sys/12345 as sysdba
Connected.

SQL > ALTER USER johnny QUOTA UNLIMITED ON USERS;

User altered.
```

7. Now, the user johnny should be able to insert a row into the t1 table and display the data as well:

```
SQL > conn johnny/johnny
Connected.

SQL > INSERT INTO t1(id) VALUES(10);

1 row created.

SQL > SELECT * FROM t1;
        ID
----------
        10
```

## B. ASSIGN PRIVILEGES WITH ADMIN OPTION

1. Create a new user called jackie and grant the user the CREATE SESSION so that the user can log in:

```
SQL > conn sys/12345 as sysdba
Connected.

SQL > CREATE USER jackie IDENTIFIED BY jackie
    2 QUOTA UNLIMITED ON users;
```

2. Then, grant the CREATE TABLE system privilege to johnny, but this time, use the WITH ADMIN OPTION:

```
SQL> GRANT CREATE TABLE TO johnny WITH ADMIN OPTION;
```

Now, the user johnny can grant the CREATE TABLE system privilege to another user e.g. jackie.

3. Next, login as johnny and grant the CREATE TABLE system privilege to jackie:

```
SQL > conn johnny/johnny
Connected.

SQL > GRANT CREATE TABLE TO jackie;
```

4. Login as jackie:

```
SQL > conn jackie/jackie
```

Noted that, Oracle issued the following error:

```
ERROR: ORA-01045:
user JACKIE lacks CREATE SESSION privilege; logon denied
Warning: You are no longer connected to ORACLE.
```

Login as sys and then grant CREATE SESSION to jackie

```
SQL > conn sys/12345 as sysdba
Connected.

SQL > GRANT CREATE SESSION TO jackie;
```

Now, login as Jackie and then create a new table:

```
SQL > CREATE TABLE t2(id NUMBER PRIMARY KEY);
```

**EXERCISE**

1) Oracle CREATE USER

Scan this QR code.

**Or** Click Here

Follow the procedure and steps stated in the Lab Activity for Creating User.

2) Oracle REVOKE

Scan this QR code.

**Or** Click Here

Follow the procedure and steps stated in the Lab Activity on how to use the Oracle REVOKE statement to revoke system and object privileges from a specific user.

3) Oracle ALTER USER

Scan this QR code.

**Or** Click Here

Follow the procedure and steps stated in the Lab Activity on how to use the Oracle ALTER USER statement to modify the authentication or database resource of a database user.

4) Oracle DROP USER

Scan this QR code.

**Or** Click Here

Follow the procedure and steps stated in the Lab Activity on how to use the Oracle DROP USER to delete a user from the database.

## Activity 4C


**Activity Outcome: Grant All Privileges to a User**

1. Run the command prompt and open SQL*Plus platform.

```
C:\ …>sqlplus sys/12345 as sysdba
```

2. Create a new user called superman with a password by using the following CREATE USER statement:

```
SQL > CREATE USER superman IDENTIFIED BY superman;

User created.
```

3. Second, use the GRANT ALL PRIVILEGES statement to grant all privileges to the super user:

```
SQL > GRANT ALL PRIVILEGES TO superman;

Grant succeeded.
```

4. Third, log in to the Oracle Database as the superman user and query the superman user's privileges:

```
SQL > conn superman/superman
Connected.

SQL > SELECT * FROM session_privs ORDER BY privilege;
```

5. Observe the output.

6. To grant all privileges to an existing user, you just need to use the GRANT ALL PRIVILEGES statement. For example, the following statement grants all privileges to the user jackie:

```
SQL > GRANT ALL PRIVILEGES TO jackie;
```

# LAB ACTIVITY 4: Managing Roles – Part 2

Duration: 3 Hours

## Learning Outcomes

This activity encompasses activities 4D, 4E, 4F and 4G.

By the end of this laboratory session, you should be able to:

1. Create role.
2. Set role.
3. Alter role.
4. Drop role

**ROLES:**

- Role is a set of privileges that can be granted to users or to other roles
- We can add privileges to roles and then grant the role to a user.

## Activity 4D


**Activity Outcome: Create role**

A role is a group of privileges. Instead of granting individual privileges to users, you can group related privileges into a role and grant this role to users. Roles help manage privileges more efficiently.

To create a new role, you use the CREATE ROLE statement. The basic syntax of the CREATE

```
CREATE ROLE role_name
[IDENTIFIED BY password]
[NOT IDENTIFIED]
```

In this syntax:

- First, specify the name of the role that you want to create.
- Second, use IDENTIFIED BY password option to create a local role and indicate that the user, who was granted the role, must provide the password to the database when enabling the role.
- Third, use NOT IDENTIFIED to indicate that the role is authorized by the database and the user, who was granted this role, don't need a password to enable the role.

After a role is created, it is empty. To grant privileges to a role, you use the GRANT statement:

```
GRANT {system_privileges | object_privileges} TO role_name;
```

In addition, you can use the GRANT statement to grant privileges of a role to another role:

```
GRANT role_name TO another_role_name;
```

## A. Using Oracle CREATE ROLE without a password

1. Run the command prompt and open SQL*Plus platform. Log in as sysdba, then grant create role to user HR.

```
C:\ …>sqlplus sys/12345 as sysdba

SQL > GRANT CREATE ROLE To HR
SQL > conn hr@orclpdb/123
```

2. Create a new role named mdm (master data management) in the sample database:

```
SQL > CREATE ROLES mdm;
```

3. Show tables in HR schema:

```
SQL > SELECT table_name from user_tables ORDER_BY table_name;

TABLE_NAME
_____________________

COUNTRIES
DEPARTMENT
S
EMPLOYEES
JOBS
JOB_HISTOR
Y
LOCATIONS
REGIONS
```

4.

```
SQL > GRANT SELECT, INSERT, UPDATE, DELETE
      ON COUNTRIES
      TO mdm;

SQL > GRANT SELECT, INSERT, UPDATE, DELETE
      ON DEPARTMENTS
      TO mdm;

SQL > GRANT SELECT, INSERT, UPDATE, DELETE
      ON EMPLOYEES
      TO mdm;

SQL > GRANT SELECT, INSERT, UPDATE, DELETE
      ON JOBS
      TO mdm;

SQL > GRANT SELECT, INSERT, UPDATE, DELETE
      ON JOB_HISTORY
      TO mdm;
```

```
SQL > GRANT SELECT, INSERT, UPDATE, DELETE
      ON LOCATIONS
      TO mdm;

SQL > GRANT SELECT, INSERT, UPDATE, DELETE
   ON REGIONS
   TO mdm;
```

5. Create a new user named alice and grant the CREATE SESSION privilege to alice:

```
SQL > CREATE USER alice IDENTIFIED BY alice;
User created.

SQL > GRANT CREATE SESSION TO alice;
Grant succeeded.
```

6. Log in to the database as alice, and attempt to query data from the hr.employees table:

```
SQL > conn alice/alice
SQL > SELECT * FROM hr.employees;
```

7. Observe the output. Noted that Oracle issued the following error:

```
ORA-00942: table or view does not exist
```

8. Go back to the first session and grant alice the mdm role:

```
SQL > GRANT mdm TO alice;
```

9. Go to the alice's session and enable role using the SET ROLE statement:

```
SQL > SET ROLE mdm;
```

10. To query all roles of the current user, you use the following query:

```
SQL > SELECT * FROM session_roles;
```

11. Observe the output.

```
ROLE
------
MDM
```

Now, alice can manipulate data in the master data tables such as departments and employees.

12. Re-try to make an attempt to query data from the hr.employees table:

```
SQL > SELECT * FROM hr.employees;
```

Observe the output. Set the display format appropriately.

13. Insert a records to hr.employess:

```
SQL > INSERT INTO hr.employees
      (employee_id, last_name, email, hire_date, job_id)
      VALUES (999,'Salmah','Salmah','01-Jun-21','AC_ACCOUNT');
1 row created.
```

**B. Using Oracle CREATE ROLE to create a role with IDENTIFIED BY password**

1. Connect as user HR then create a new role named order_entry with the password xyz123:

```
SQL > conn hr@orclpdb/123
Connected.

SQL > CREATE ROLE region_entry IDENTIFIED BY xyz123;
```

2. Grant object privileges on the orders and order_items tables to the order_entry role:

```
SQL > GRANT SELECT, INSERT, UPDATE, DELETE
      ON regions
      TO region_entry;
```

3. Grant the region_entry role to the user alice:

```
SQL> GRANT order_entry TO alice;
```

4. Log in as alice and enable the region_entry role by using the SET ROLE statement:

```
SQL > SET ROLE
      region_entry IDENTIFIED BY xyz123,
    mdm;
Role set.
```

5. Use the following statement to get the current roles of alice:

```
SQL > SELECT * FROM session_roles;
```

6. Observe the output. The current roles of alice are as follow:

```
ROLE
------
MDM
REGION_ENTRY
```

## Activity 4E


**Activity Outcome: Set role**

The SET ROLE statement allows you to enable and disable roles for your current

session. Here is the basic syntax of the SET ROLE statement:

```
SET ROLE role;
```

In this syntax, you just need to specify the role that was previously granted to your account.

If the role requires a password, you use the following syntax:

```
SET ROLE role IDENTIFIED BY password;
```

It is possible to enable multiple roles at once like the following statement:

```
SET ROLE role1, role2, ...;
```

Or

```
SET ROLE
    role1,
    role2 IDENTIFIED BY password,
    ...;
```

To enable all roles previously granted to your account, you use the following syntax:

```
SET ROLE ALL;
```

The session_roles data dictionary view provides the currently enabled roles in your current session:

```
SELECT * FROM session_roles;
```

1. In hr session, create a user named scott and grant him the CREATE SESSION privilege so that he can log in the database:

```
SQL > CREATE USER scott IDENTIFIED BY scott;
User created.

SQL > GRANT CREATE SESSION TO scott;
Grant succeeded.
```

2. Create a new table create two roles called warehouse_manager and warehouse_staff:

```
SQL > CREATE ROLE warehouse_staff;
Role created.

SQL > CREATE ROLE warehouse_manager IDENTIFIED BY xyz123;
Role created.
```

3. Grant object privileges on inventories table to the warehouse_staff role:

```
SQL > GRANT SELECT, INSERT, UPDATE, DELETE
   ON employees
 TO warehouse_staff;
Grant succeeded.
```

4. Grant object privileges on warehouses table to the warehouse_manager role:

```
SQL > GRANT SELECT, INSERT, UPDATE, DELETE
   ON departments
 TO warehouse_manager;
Grant successded.
```

5. Grant privileges of the warehouse_staff role to warehouse_manager role:

```
SQL > GRANT warehouse_staff to warehouse_manager;
Grant succeeded.
```

6. Grant the role warehouse_manager to scott and warehouse_staff to alice:

```
SQL > GRANT warehouse_manager TO scott;
Grant succeeded.

SQL > GRANT warehouse_staff TO alice;
Grant succeeded.
```

7. Log in to the database as scott and enable the warehouse_manager role:

```
SQL > conn scott/scott;

SQL > SET ROLE warehouse_manager IDENTIFIED BY xyz123;
```

8. View the current roles of scott:

```
SQL > SELECT * FROM session_roles;
```

```
ROLE
------------------
WAREHOUSE_STAFF
WAREHOUSE_MANAGER
```

The user scott has two roles: warehouse_manager who was directly granted and warehouse_staff that was indirectly granted via the warehouse_manager role.

9. To disable all roles of scott, you use this statement:

```
SQL > SET ROLE NONE;
```

## Activity 4F


**Activity Outcome: Alter role**

The ALTER ROLE statement allows you to modify the authorization needed to enable a

role. Here is the basic syntax of the ALTER ROLE statement:

```
ALTER ROLE role_name
    { NOT IDENTIFIED | IDENTIFIED BY password }
```

In this syntax:

- First, specify the name of the role that you want to change.
- Second, use the corresponding action such as NOT IDENTIFIED to not using a password, or IDENTIFIED BY password to change the password of the role.

To execute the ALTER ROLE statement, your account must either have been granted the role with ADMIN OPTION or have the ALTER ANY ROLE system privilege.

Note that it is not possible to change a NOT IDENTIFIED role to a IDENTIFIED BY password role if the role has been granted to another role.

**Using the ALTER ROLE statement to change an IDENTIFIED BY password role to a NOT IDENTIFIED role.**

1. Run the command prompt and open SQL*Plus platform.

```
C:\ …>sqlplus sys/12345 as sysdba
```

2. Create a new role called db_designer:

```
SQL > CREATE ROLE db_designer IDENTIFIED BY abcd1234;
Role created.
```

3. Grant the CREATE TABLE and CREATE VIEW system privileges to the db_designer role:

```
SQL > GRANT CREATE TABLE, CREATE VIEW TO db_designer;
Grant succeeded.
```

4. Create a user called michael:

```
SQL > CREATE USER michael IDENTIFIED BY michael;
User created.
```

5. Grant the db_designer and connect roles to the user michael:

```
SQL > GRANT db_designer, connect TO michael;
Grant succeeded. connect
```

6. The following query returns the roles granted to the user michael:

```
SQL > SELECT * FROM dba_role_privs
 2 WHERE grantee = 'MICHAEL';
```

Observe the output:

7. View all role granted to michael:

```
SQL > SELECT granted_role
       FROM dba_role_privs
       WHERE grantee = MICHAEL';
```

Observe the output.

8. Vew all the privilege set to role db_designer.

```
SQL > SELECT * FROM role_sys_privs
    2  WHERE ROLE = 'DB_DESIGNER';
```

Observe the output.

9. Log in to the Oracle Database using the user michael and set the role of michael to db_designer:

```
SQL > SET ROLE db_designer IDENTIFIED BY abcd1234;
Role set.he
```

10. View the current roles of michael:

```
SQL > SELECT * FROM session_roles;
```

Observe the output.

11. Go back to the first session and change the role to a NOT IDENTIFIED role:

```
SQL > ALTER ROLE db_designer NOT IDENTIFIED;
```

12. Go the user michael's session and reissue the SET ROLE statement again. This time we don't need a password since the role has been changed:

```
SQL > SET ROLE db_designer;
```

## Activity 4G


**Activity Outcome: Drop role**

The DROP ROLE statement allows you to remove a role from the database.

Here is the syntax of the DROP ROLE statement:

```
DROP ROLE role_name;
```

In this syntax, you specify the name of the role that you want to drop after the DROP ROLE keywords.

When you drop a role, Oracle revokes it from all users and roles that have been previously granted. In addition, Oracle deletes the role from the database.

To drop a role, you must have the DROP ANY ROLE system privilege or have been granted the role with the ADMIN OPTION.

**A. DROP ROLE (1)**

1. Log in to the Oracle Database using the sys account.

```
SQL > conn sys/12345 as sysdba
```

2. Next, create a new role called developer:

```
SQL > CREATE ROLE developer;
```

3. Then, check if the role has been created successfully:

```
SQL > SELECT * from dba_roles
    2  WHERE role = 'DEVELOPER';
```

   Observe the output.

4. After that, drop the developer role:

```
SQL > DROP ROLE developer;
```

5. Check the dba_roles again using SELECT statement in no. 3.

**B. DROP ROLE (2)**

1. Still log into the Oracle database using sys account. Create a new role called auditor and grant the SELECT object privilege on the orders table in the sample database:

```
SQL > CREATE ROLE auditor;
Role created.

SQL > GRANT SELECT ON employees TO auditor;
Grant succeeded.
```

2. Create a new user named audi, grant the CREATE SESSION system privilege and the auditor role to audi:

```
SQL > CREATE USER audi IDENTIFIED BY audi;
User created.

SQL > GRANT CREATE SESSION TO auditor;
Grant succeeded.

SQL > GRANT auditor TO audi;
Grant succeeded.
```

3. Log in to the Oracle database as the audi user in the second session and issue the following command:

```
SQL > conn audi/audi
SQL > SELECT * FROM sys.employees;
```

Observe the output.

4. Query role of the audi user:

```
SQL > SELECT * FROM session_roles;
```

Observe the output.

```
 ROLE
------
AUDITOR
```

5. To view all privilege set to role auditor:

```
SQL > SELECT * FROM ROLE_SYS_PRIVS
    2  WHERE ROLE ='AUDITOR';
```

6. Go back to the first session and drop the role auditor:

```
SQL > DROP ROLE auditor;
```

Observe the output.

**EXERCISE**

1) Managing Roles

Scan this QR code.

**Or** Click Here.

Follow the procedures and steps stated in Lab Activity on how to manage roles in Oracle.

2) Managing Privileges

Scan this QR code.

**Or** Click Here.

Follow the procedures and steps stated in Lab Activity on how to manage privileges in Oracle.

# LAB ACTIVITY 5

# LAB ACTIVITY 5: Backup and Recovery

Duration: 3 Hours

## Learning Outcomes

This activity encompasses activities 5A until 5G.

By the end of this laboratory session, you should be able to:

1. Understand RMAN.
2. Creating Recovery Catalog Owner.
3. Creating Recovery Catalog.
4. Backing Up a Database.
5. Reporting RMAN Operation.
6. Diagnose and Repair Failures with Data Recovery Advisor
7. Restore and Recover Database Files

## Activity 5A


**Activity Outcome: Understand RMAN Connection**

Scan this QR code.

**Start and Interact with RMAN**

## Activity 5B


**Activity Outcome: Creating the Recovery Catalog Owner**

RMAN can be used either with or without a recovery catalog. A recovery catalog is a schema stored in a database that tracks backups and stores scripts for use in RMAN backup and recovery situations. Generally, an experienced DBA would suggest that the Enterprise Manager instance schema and RMAN catalog schema be placed in the same utility database on a server separate from the main servers. The RMAN schema generally only requires 15 megabyte per year per database backed up.

The RMAN schema owner is created in the RMAN database using the following steps:

```
SQL*Plus: Release 19.0.0.0.0 - Production on Fri Jun 11 21:56:14 2021
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle.  All rights reserved.

Enter user-name: sys as sysdba
Enter password:

Connected to:
Oracle Database 19c Enterprise Edition Release 19.0.0.0.0 - Production
Version 19.3.0.0.0

SQL> CREATE USER rman IDENTIFIED BY cat
  2  TEMPORARY TABLESPACE temp
  3  DEFAULT TABLESPACE users
  4  QUOTA UNLIMITED ON users;

User created.
```

Grant the recovery_catalog_owner role to the user. This role provides all of the privileges required to maintain and query the recovery catalog.

```
SQL> GRANT RECOVERY_CATALOG_OWNER TO rman;

Grant succeeded.
```

## Activity 5C


**Activity Outcome: Creating the Recovery Catalog**

Before you perform any operations using RMAN, you must connect to a target database. The RMAN client is started by issuing the **`rman`** command at the command prompt of your operating system.

**Connecting to the Target Database, Recovery Catalog and Creating Recovery Catalog:**

```
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>rman

Recovery Manager: Release 19.0.0.0.0 - Production on Fri Jun 11 22:18:15 2021
Version 19.3.0.0.0

Copyright (c) 1982, 2019, Oracle and/or its affiliates.  All rights reserved.

RMAN> connect target sys/12345

connected to target database: ORCL (DBID=1595310094)

RMAN> connect CATALOG rman/cat

connected to recovery catalog database

RMAN> create CATALOG;

recovery catalog created
```

**Registering a Database in the Recovery Catalog:**

```
RMAN> register database;

database registered in recovery catalog
starting full resync of recovery catalog
full resync complete
```

Verify that the registration was successful by running **`REPORT SCHEMA`**:

```
RMAN> report schema;
```

**Exit from RMAN:**

```
RMAN> exit

Recovery Manager complete.
```

The following example appends the output from an RMAN session to a text file at `/tmp/msglog.log`

```
RMAN> TARGET / LOG /tmp/msglog.log APPEND
```

## Showing the Default RMAN Configuration

The RMAN backup and recovery environment is preconfigured for each target database. The configuration is persistent and applies to all subsequent operations on this target database, even if you exit and restart RMAN.

RMAN configuration settings can specify backup devices, set up connections to those devices (known as channels), set policies affecting backup strategy, and more.

**To show the current configuration for a database:**

```
RMAN> show all;

RMAN configuration parameters for database with db_unique_name ORCL are:
CONFIGURE RETENTION POLICY TO REDUNDANCY 1; # default
CONFIGURE BACKUP OPTIMIZATION OFF; # default
CONFIGURE DEFAULT DEVICE TYPE TO DISK; # default
CONFIGURE CONTROLFILE AUTOBACKUP ON; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 1 BACKUP TYPE TO BACKUPSET; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR LOAD TRUE ; # default
CONFIGURE RMAN OUTPUT TO KEEP FOR 7 DAYS; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO 'D:\DB_HOME\DATABASE\SNCFORCL2.ORA'; # default
```

The output lists the CONFIGURE commands to re-create this configuration.

## Activity 5D


**Activity Outcome: Backing Up a Database**

Use the BACKUP command to back up files. RMAN backs up data to the configured default device for the type of backup requested.

By default, RMAN creates backups on disk. If a fast recovery area is enabled, and if you do not specify the FORMAT parameter, then RMAN creates backups in the recovery area and automatically gives them unique names.

**Backing Up a Database in ARCHIVELOG Mode**

If a database runs in ARCHIVELOG mode, then you can back up the database while it is open.

**To back up the database and archived redo logs while the database is open:**

```
RMAN> CONNECT TARGET sys/12345
target database Password: password
connected to target database: ORCL (DBID=1595310094)

RMAN> BACKUP DATABASE PLUS ARCHIVELOG;
```

**Backing Up a Database in NOARCHIVELOG Mode**

If a database runs in NOARCHIVELOG mode, then the only valid database backup is a consistent backup. For the backup to be consistent, the database must be mounted after a consistent shutdown.

**To make a consistent database backup:**

- Start RMAN and connect to a target database
- Shut down the database consistently and then mount it. For example, enter the following commands to guarantee that the database is in a consistent state for a backup:

```
RMAN> SHUTDOWN IMMEDIATE;
RMAN> STARTUP FORCE DBA;
RMAN> SHUTDOWN IMMEDIATE;
RMAN> STARTUP MOUNT;
```

- Run the BACKUP DATABASE command.
  For example, enter the following command at the RMAN prompt to back up the database to the default backup device:

```
RMAN> BACKUP DATABASE;
```

  The following variation of the command creates image copy backups of all data files in the database:

```
RMAN> BACKUP AS COPY DATABASE;
```

- Open the database and resume normal operations. The following command opens the database:

```
RMAN> ALTER DATABASE OPEN;
```

**Making Incremental Backups**

Incremental backups capture block-level changes to a database made after a previous incremental backup.

If you specify BACKUP INCREMENTAL, then RMAN creates an incremental backup of a database. Incremental backups are generally smaller and faster to make than full database backups. Recovery with incremental backups is faster than using redo logs alone.

The starting point for an incremental backup strategy is a level 0 incremental backup, which backs up all blocks in the database. An incremental backup at level 0 is identical in content to a full backup, however, unlike a full backup the level 0 backup is considered a part of the incremental backup strategy.

A level 1 incremental backup contains only blocks changed after a previous incremental backup. If no level 0 backup exists in either the current or parent database incarnation when you run a level 1 backup, then RMAN makes a level 0 backup automatically.

**To make incremental backups of the database:**

- Start RMAN and connect to a target database
- Run the BACKUP INCREMENTAL command.
  The following example creates a level 0 incremental backup to serve as a base for an incremental backup strategy:

```
BACKUP INCREMENTAL LEVEL 0 DATABASE;
```

The following example creates a level 1 cumulative incremental backup:

```
BACKUP INCREMENTAL LEVEL 1 CUMULATIVE DATABASE;
```

The following example creates a level 1 differential incremental backup:

```
BACKUP INCREMENTAL LEVEL 1 DATABASE;
```

## Making Incrementally Updated Backups

Incrementally updated backups enable you to implement an efficient incremental forever backup strategy.

**To implement an incrementally updated backup strategy:**

- Start RMAN and connect to a target database
- Run the RECOVER COPY and BACKUP INCREMENTAL commands.
  The following script, run on a regular basis, is all that is required to implement a strategy basedon incrementally updated backups.

```
RECOVER COPY OF DATABASE
   WITH TAG 'incr_update';
BACKUP
   INCREMENTAL LEVEL 1
   FOR RECOVER OF COPY WITH TAG 'incr_update' DATABASE;
```

## Validating Database Files and Backups

RMAN validation checks a backup to determine whether it can be restored. Validation also checksfor corrupt blocks and missing files.

Use the **VALIDATE** command to confirm that all database files exist, are in their correct location, and are free of physical corruption. The **CHECK LOGICAL** option also checks for logical block corruption.

**To validate database files:**

- Start RMAN and connect to a target database
- Run the **BACKUP VALIDATE** ... command for the desired files.
  For example, enter the following commands to validate all database files and archived redo logfiles for physical and logical corruption:

```
BACKUP VALIDATE CHECK LOGICAL
   DATABASE ARCHIVELOG ALL;
```

You can also use the **VALIDATE** command to check individual data blocks, as shown in the following example:

```
VALIDATE DATAFILE 4 BLOCK 10 TO 13;
```

You can also validate backup sets, as shown in the following example:

```
VALIDATE BACKUPSET 3;
```

You specify backup sets by primary key, which is shown in the output of the **LIST BACKUP** command.

## Activity 5E


**Activity Outcome: Reporting on RMAN Operations**

RMAN can use the information stored in the RMAN repository to generate reports on backup activities.

Use the **RMAN LIST** and **REPORT** commands for reporting on backup operations. Use the **SHOW ALL** command to display the current RMAN configuration. In addition, RMAN provides a comprehensive set of views for generating custom reports.

### Listing Backups

The **LIST BACKUP** and **LIST COPY** commands display information about backups and data file copies listed in the repository.

**To list backups and copies:**

- Start RMAN and connect to a target database
- Run the LIST command at the RMAN prompt.
  You can display specific objects, as in the following examples:

```
LIST BACKUP OF DATABASE;
LIST COPY OF DATAFILE 1, 2;
LIST BACKUP OF ARCHIVELOG FROM SEQUENCE 10;
LIST BACKUPSET OF DATAFILE 1;
```

### Reporting on Database Files and Backups

The **REPORT** command performs more complex reporting analysis than the **LIST** command.

**To generate reports of database files and backups:**

- Start RMAN and connect to a target database.
- Run the REPORT command at the RMAN prompt.
  The following example reports backups that are obsolete according to the currently configured backup retention policy:

```
REPORT OBSOLETE;
```

The following example reports the data files and temp files in the database:

```
REPORT SCHEMA;
```

## Maintaining RMAN Backups

RMAN repository metadata is always stored in the control file of the target database. The RMAN maintenance commands use this metadata when managing backups.

## Cross-checking Backups

Use the **CROSSCHECK** command to synchronize the logical records of RMAN backups and copies with the files on storage media.

If a backup is on disk, then **CROSSCHECK** determines whether the header of the file is valid. If a backup is on tape, then RMAN queries the RMAN repository for the names and locations of the backup pieces. It is a good idea to crosscheck backups and copies before deleting them.

**To crosscheck all backups and copies on disk:**

- Start RMAN and connect to a target database.
- Run the `CROSSCHECK` command, as shown in the following example:

```
CROSSCHECK BACKUP;
CROSSCHECK COPY;
```

## Deleting Obsolete Backups

The **DELETE** command removes RMAN backups and copies from disk and tape, updates thestatus of the files to DELETED in the control file repository, and removes the records from therecovery catalog (if you use a catalog).

If you run RMAN interactively, and if you do not specify the **NOPROMPT** option, then **DELETE** displays a list of files and prompts for confirmation before deleting any file in the list. The **DELETEOBSOLETE** command is useful because RMAN deletes backups and data file copies recorded in the RMAN repository that are obsolete, that is, no longer needed. You can use options on the **DELETE** command to specify what is obsolete or use the configured backup retention policy.

**To delete obsolete backups and copies:**

- Start RMAN and connect to a target database
- Run the `DELETE OBSOLETE` command, as shown in the following example:

```
DELETE OBSOLETE;
```

## Activity 5F


**Activity Outcome: Diagnosing and Repairing Failures with Data Recovery Advisor**

Data Recovery Advisor is an Oracle Database tool that provides an infrastructure for diagnosing persistent data failures, presenting repair options to the user, and executes repairs at the user's request.

### Listing Failures and Determining Repair Options

A failure is a persistent data corruption detected by the Health Monitor. Examples include physical and logical data block corruptions and missing data files.

Each failure has a failure priority and failure status. The priority can be CRITICAL, HIGH, or LOW. The status can be OPEN or CLOSED.

You can run the **LIST FAILURE** command to show all known failures. If failures exist, then run the **ADVISE FAILURE** command in the same session to determine repair options. The **ADVISE FAILURE** output shows both manual and automated repair options. First try to fix the problem manually. If you cannot fix the problem manually, then review the automated repair section.
An automated repair option describes a server-managed repair for one or more failures. Repairsare consolidated when possible so that a single repair can fix multiple failures. The repair optionindicates which repair is performed and whether data is lost by performing the repair operation.

The following illustrates the commands to list failures and determine repair options. The output indicates the file name of a repair script containing RMAN commands.

```
RMAN> LIST FAILURE;

Database Role: PRIMARY
List of Database Failures
=========================

Failure ID Priority Status    Time Detected Summary
---------- -------- --------- ------------- -------
142        HIGH     OPEN      23-APR-13     One or more non-system
                                            datafiles are missing
101        HIGH     OPEN      23-APR-13     Datafile 1:
                                           '/disk1/oradata/prod/system01.dbf'
                                           contains one or more corrupt blocks


RMAN> ADVISE FAILURE;
```

```
Database Role: PRIMARY
List of Database Failures
=========================

Failure ID Priority Status       Time Detected Summary
---------- -------- ------------ ------------- -----------
142        HIGH     OPEN         23-APR-13     One or more non-system
                                                datafiles are missing
101        HIGH     OPEN         23-APR-13     Datafile 1:
                                                '/disk1/oradata/prod/system01.dbf'
                                                Contains one or more corrupt blocks

analyzing automatic repair options; this may take some time
using channel ORA_DISK_1
analyzing automatic repair options complete

Mandatory Manual Actions
========================
no manual actions available

Optional Manual Actions
=======================
1. If file /disk1/oradata/prod/users01.dbf was unintentionally
renamedor moved, restore it

Automated Repair Options
========================
Option Repair Description
------ ------------------
1 Restore and recover datafile 28; Perform block media recovery of
block 56416 in file 1
Strategy: The repair includes complete media recovery with no data loss
Repair script: /disk1/oracle/log/diag/rdbms/prod/prod/hm/reco_660500184.hm
```

## Repairing Failures

Use the RMAN **REPAIR FAILURE** command to repair failures that were detected.

After running **LIST FAILURE** and **ADVISE FAILURE** in an RMAN session, you can run **REPAIR FAILURE** to execute a repair option. If you execute **REPAIR FAILURE** with no other command options, then RMAN uses the first repair option of the most recent **ADVISE FAILURE** command in the current session. Alternatively, specify the repair option number obtained from the most recent **ADVISE FAILURE** command.

```
RMAN> REPAIR FAILURE;
```

By default, **REPAIR FAILURE** prompts for confirmation before it begins executing. After executinga repair, Data Recovery Advisor re-evaluates all existing failures on the possibility that they may also have been fixed. Data Recovery Advisor always verifies that failures are still relevant and automatically closes fixed failures. If a repair fails to complete because of an error, then the error triggers a new assessment and re-evaluation of existing failures and repairs.

## Activity 5G


**Activity Outcome: Restoring and Recovering Database Files**

Use the **RESTORE** and **RECOVER** commands for RMAN restore and recovery of physical database files. Restoring data files is retrieving them from backups as needed for a recovery operation. Media recovery is the application of changes from redo logs and incremental backups to a restored datafile to bring the data file forward to a desired SCN or point in time.

### Preparing to Restore and Recover Database Files

To recover the database because a media failure damages database files, then first ensure that you have the necessary backups.

You can use the **RESTORE ... PREVIEW** command to report, but not restore, the backups that RMAN can use to restore to the specified time. RMAN queries the metadata and does not actually read the backup files. The database can be open when you run this command.

**To preview a database restore and recovery:**

- Start RMAN and connect to the target database
- Optionally, list the current tablespaces and data files, as shown in the following command:

```
RMAN> REPORT SCHEMA;
```

- Run the **RESTORE DATABASE** command with the **PREVIEW** option.
  The following command specifies **SUMMARY** so that the backup metadata is not displayed in verbose mode (sample output included):

```
RMAN> RESTORE DATABASE PREVIEW SUMMARY;

Starting restore at 21-MAY-
13 allocated channel:
ORA_DISK_1
channel ORA_DISK_1: SID=80 device type=DISK

List of Backups
===============
Key     TY LV S Device Type Completion Time #Pieces #Copies Compressed Tag
11      B  F  A DISK        18-MAY-
13      1  2  NO  TAG20070518T181114
13      B  F  A DISK        18-MAY-
13      1  2  NO  TAG20070518T181114
using channel ORA_DISK_1

List of Archived Log Copies for database with db_unique_name PROD
=====================================================================
Key     Thrd Seq     S Low
Time 47 1    18      A 18-MAY-
13
        Name: /disk1/oracle/dbs/db1r_60ffa882_1_18_0622902157.arc

Media recovery start SCN is 586534
Recovery must be done beyond SCN 587194 to clear datafile
fuzziness validation succeeded for backup piece
Finished restore at 21-MAY-13
```

## Recovering the Whole Database

Use the **RESTORE DATABASE** and **RECOVER DATABASE** commands to recover the whole database.

You must have previously made backups of all needed files. This scenario assumes that you can restore all data files to their original locations. If the original locations are inaccessible, then use the **SET NEWNAME** command.

**To recover the whole database:**

- Prepare for recovery as explained earlier.
- Place the database in a mounted state.
  The following example terminates the database instance (if it is started) and mounts the database:

```
RMAN> STARTUP FORCE MOUNT;
```

- Restore the database.
  The following example uses the preconfigured disk channel to restore the database:

```
RMAN> RESTORE DATABASE;
```

- Recover the database, as shown in the following example:

```
RMAN> RECOVER DATABASE;
```

- Open the database, as shown in the following example:

```
RMAN> ALTER DATABASE OPEN;
```

## Recovering Tablespaces

Use the **RESTORE TABLESPACE** and **RECOVER TABLESPACE** commands on individual tablespaces when the database is open. In this case, you must take the tablespace that needs recovery offline, restore and then recover the tablespace, and bring the recovered tablespace online.

If you cannot restore a data file to its original location, then use the RMAN **SET NEWNAME** command within a RUN block to specify the new file name and location.

Afterward, use a **SWITCH DATAFILE ALL** command to update the control file to reflect the new names for all data files for which a **SET NEWNAME** has been issued in the **RUN** command.

**To recover an individual tablespace when the database is open:**

- Prepare for recovery as explained earlier.
- Take the tablespace to be recovered offline.
  The following example takes the USERS tablespace offline:

```
RMAN> ALTER TABLESPACE users OFFLINE;
```

- Restore and recover the tablespace.
  The following RUN command, which you execute at the RMAN prompt, sets a new name forthe data file in the USERS tablespace:

```
RUN
{
   SET NEWNAME FOR DATAFILE '/disk1/oradata/prod/users01.dbf'
   TO '/disk2/users01.dbf';
   RESTORE TABLESPACE users;
   SWITCH DATAFILE ALL; #update control file with new file names
   RECOVER TABLESPACE users;
}
```

- Bring the tablespace online, as shown in the following example:

```
RMAN> ALTER TABLESPACE users ONLINE;
```

You can also use **RESTORE DATAFILE** and **RECOVER DATAFILE** for recovery at the data file level.

## Recovering Individual Data Blocks

RMAN can recover individual corrupted data file blocks.

When RMAN performs a complete scan of a file for a backup, any corrupted blocks are listed in **V$DATABASE_BLOCK_CORRUPTION**. Corruption is usually reported in alert logs, trace files, or results of SQL queries.

**To recover data blocks:**

- Start RMAN and connect to the target database.
- Obtain the block numbers of the corrupted blocks if you do not have this information.

```
RMAN> SELECT NAME, VALUE FROM V$DIAG_INFO;
```

- Run the RECOVER command to repair the blocks.
  The following RMAN command recovers all corrupted blocks:

```
RMAN> RECOVER CORRUPTION LIST;
```

  You can also recover individual blocks, as shown in the following example:

```
RMAN> RECOVER DATAFILE 1 BLOCK 233, 235
      DATAFILE 2 BLOCK 100 TO 200;
```

# REFERENCES


Kumar. R A (2020). *Oracle Database 2 Day DBA, 19c.* Oracle. (ISBN: E96197-09)

Doran. M, Potineni. P, Bhatiya. R (2023). *Oracle Database Database Administrator's Guide, 19c.* Oracle. (ISBN: E96348-17)

Ramya. P. (2023). *Oracle Database Backup and Recovery User's Guide, 19c*. Oracle. (ISBN: E96241-10)

OracleTututorial.com website https://www.oracletutorial.com/oracle-administration/

Oracle Database 19c website https://docs.oracle.com/en/database/oracle/oracle-database/19/index.html

# ORACLE DATABASE ADMINISTRATION LAB WORKBOOK