Oracle Database 12*c*: Install and Upgrade Workshop

Student Guide

D77766GC10
Edition 1.0
July 2013
D82686

Authors

Donna Keesling
Dominique Jeunot
James Spiller

Technical Contributors and Reviewers

Roy Swonger
Ira Singer
Darryl Balaski
Mark Fuller
Carol Tagliaferri
Harald Van Breederode
Joel Goodman
Prakash Jashnani
Gerlinde Frenzen

Editor

Arijit Ghosh

Graphic Designer

Divya Thallap

Publishers

Syed Imtiaz Ali
Giri Venugopal

Contents

1 Oracle Database 12*c*: Overview
Objectives 1-2
Oracle Database Innovation 1-3
Enterprise Cloud Computing 1-4
Oracle Database Server Architecture: Overview 1-5
Oracle Database Instance Configurations 1-6
Connecting to the Database Instance 1-7
Oracle Database Memory Structures 1-8
Process Architecture 1-10
Process Structures 1-12
Database Storage Architecture 1-14
Logical and Physical Database Structures 1-16
Tablespaces and Data Files 1-18
SYSTEM and SYSAUX Tablespaces 1-19
Database User Accounts 1-20
Predefined Administrative Accounts 1-22
Administrative Privileges 1-23
Interacting with an Oracle Database: Memory, Processes, and Storage 1-24
Multitenant Architecture: Introduction 1-26
Multitenant Architecture 1-27
Automatic Storage Management 1-28
ASM Instance 1-29
ASM Components: ASM Instance – Primary Processes 1-31
Interaction Between Database Instances and ASM 1-32
Oracle Net Services: Overview 1-33
Oracle Net Listener: Overview 1-34
Oracle Database Management Tools 1-35
Oracle Automatic Storage Management (Oracle ASM) Tools 1-37
Summary 1-38
Practice 1: Overview 1-39

2 Oracle Software Installation Basics
Objectives 2-2
Planning Your Installation 2-3
Configuring Oracle Linux with Oracle RDBMS Pre-Install RPM 2-5

Operating System Groups and Users 2-7
Environment Variables 2-8
Configuring the Oracle Software Owner Environment 2-9
Using Oracle Universal Installer (OUI) 2-10
Installation Option: Silent Mode 2-11
Quiz 2-12
Summary 2-14

3 Installing Oracle Grid Infrastructure for a Standalone Server
Objectives 3-2
Overview of Oracle Grid Infrastructure for a Standalone Server 3-3
Oracle Restart 3-4
ASM Storage Components 3-6
Configuring Storage for Oracle Automatic Storage Management (ASM) 3-7
Oracle Grid Infrastructure Installation: System Requirements 3-8
Creating Operating System Groups and Users 3-9
Types of Installations 3-10
Installing the Oracle Grid Infrastructure for Standalone Server 3-12
Upgrading Existing Oracle ASM Instances 3-13
Summary 3-14
Practice 3: Overview 3-15

4 Installing Oracle Database Software
Objectives 4-2
Oracle Database Installation: System Requirements 4-3
Creating Operating System Groups and Users 4-4
Types of Installations 4-6
Quiz 4-7
Summary 4-8
Practice 4: Overview 4-9

5 Creating an Oracle Database by Using DBCA
Objectives 5-2
Planning the Database 5-3
Types of Databases 5-4
Choosing the Appropriate Character Set 5-5
Understanding How Character Sets Are Used 5-7
Setting the NLS_LANG Initialization Parameter 5-8
Using the DBCA to Create a Database 5-9
Creating a Container Database by Using DBCA 5-10
Creating a Database Design Template 5-11

Using the DBCA to Delete a Database 5-12
Using the DBCA for Additional Tasks 5-14
Summary 5-15
Practice 5: Overview 5-16

6 Oracle Restart
Objectives 6-2
Oracle Restart 6-3
Oracle Restart Process Startup 6-5
Controlling Oracle Restart 6-6
Choosing the Correct SRVCTL Utility 6-8
Oracle Restart Configuration 6-9
Using the SRVCTL Utility 6-10
Obtaining Help for the SRVCTL Utility 6-11
Starting Components by Using the SRVCTL Utility 6-12
Stopping Components by Using the SRVCTL Utility 6-13
Viewing Component Status 6-14
Displaying the Oracle Restart Configuration for a Component 6-15
Manually Adding Components to the Oracle Restart Configuration 6-16
Quiz 6-17
Summary 6-18
Practice 6: Overview 6-19

7 Introduction to Upgrading to Oracle Database 12*c*
Objectives 7-2
Defining Upgrade and Data Migration 7-3
Upgrade Methods 7-4
Database Upgrade Assistant: Advantages and Disadvantages 7-5
Manual Upgrade: Advantages and Disadvantages 7-6
Upgrade Paths 7-7
Upgrade Method Limitations 7-8
Upgrading to a New Release of Oracle Database 7-9
Migration Methods 7-10
Using the Export and Import Method 7-11
Summary 7-12
Practice 7: Overview 7-13

8 Preparing to Upgrade to Oracle Database 12*c*
Objectives 8-2
Preparing to Upgrade 8-3
Planning the Upgrade 8-4

Developing a Test Plan 8-5
Performance Testing 8-6
Requirements for Databases Using Oracle Label Security or Oracle Database Vault 8-7
Requirement for Databases Using Oracle Warehouse Builder 8-8
Using the Pre-Upgrade Information Tool 8-9
Backing Up the Database 8-10
Summary 8-11
Practice 8: Overview 8-12

9 Upgrading to Oracle Database 12*c*
Objectives 9-2
Upgrading by Using the Database Upgrade Assistant (DBUA) 9-3
Key DBUA Features 9-4
Manually Upgrading to Oracle Database 12*c* 9-7
Migrating a Pre-12.1 or 12.1 Non-CDB to CDB 9-9
Plugging a Non-CDB Database into a CDB 9-10
Summary 9-11
Practice 9: Overview 9-12

10 Performing Post-Upgrade Tasks
Objectives 10-2
Required Tasks After Database Upgrade 10-3
Recommended Tasks After Database Upgrade 10-4
Understanding Auditing Implementation 10-6
Enabling Unified Auditing 10-7
Administering the Roles Required for Auditing 10-8
Summary 10-9
Practice 10: Overview 10-10

11 Migrating Data by Using Oracle Data Pump
Objectives 11-2
Oracle Data Pump: Overview 11-3
Data Pump Export and Import Clients: Overview 11-4
Data Pump Utility: Interfaces and Modes 11-5
Migrating by Using Oracle Data Pump 11-6
Importing by Using a Network Link 11-7
Creating a Container Database by Using DBCA 11-8
Creating a New PDB from PDB$SEED 11-9

Exporting from a Non-CDB and Importing into a PDB 11-10
Summary 11-11
Practice 11: Overview 11-12

1

Oracle Database 12*c*: Overview

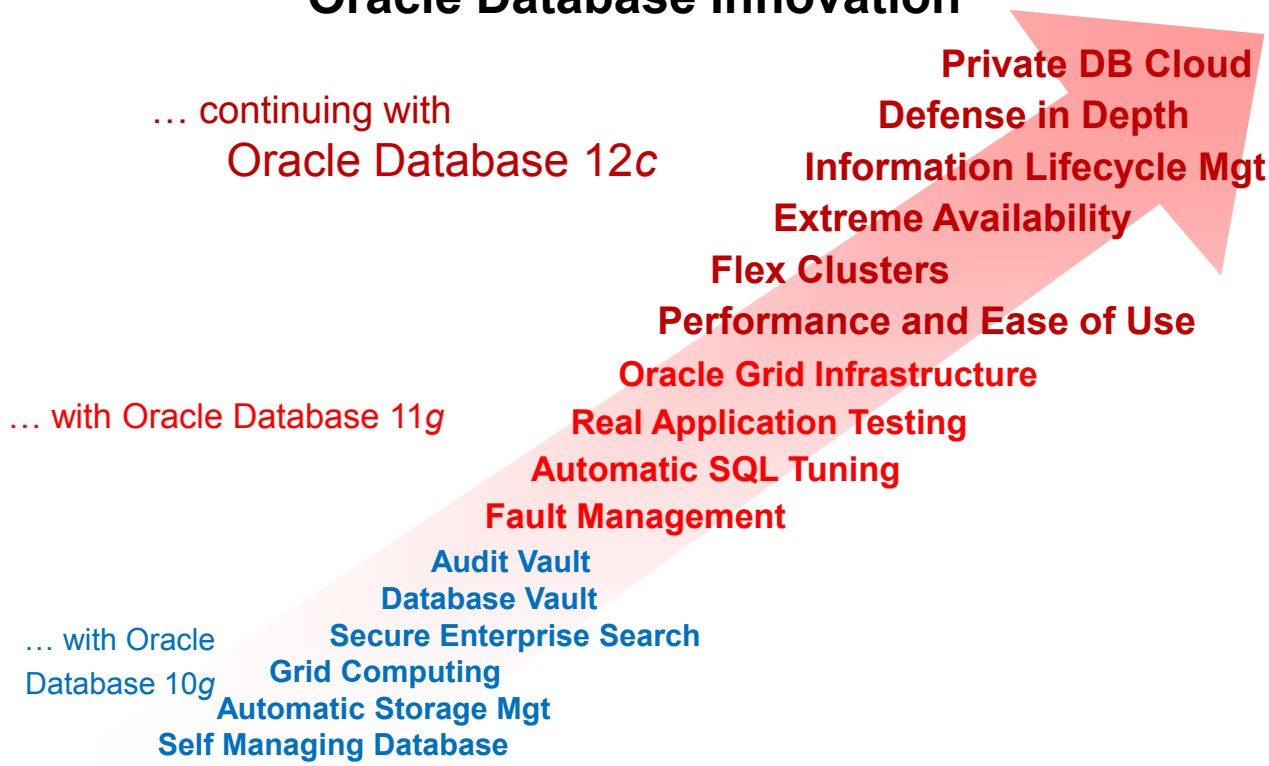
Objectives

After completing this lesson, you should be able to:

- List the major architectural components of an Oracle Database server
- Describe the memory structures and background processes of an Oracle Database instance
- Correlate the logical and physical storage structures of an Oracle database
- Describe the architecture of Oracle Automatic Storage Management (ASM)
- Describe the primary function of Oracle Net Services and the listener
- List the tools that are used to administer an Oracle Database instance and an ASM instance

ORACLE

Oracle Database Innovation

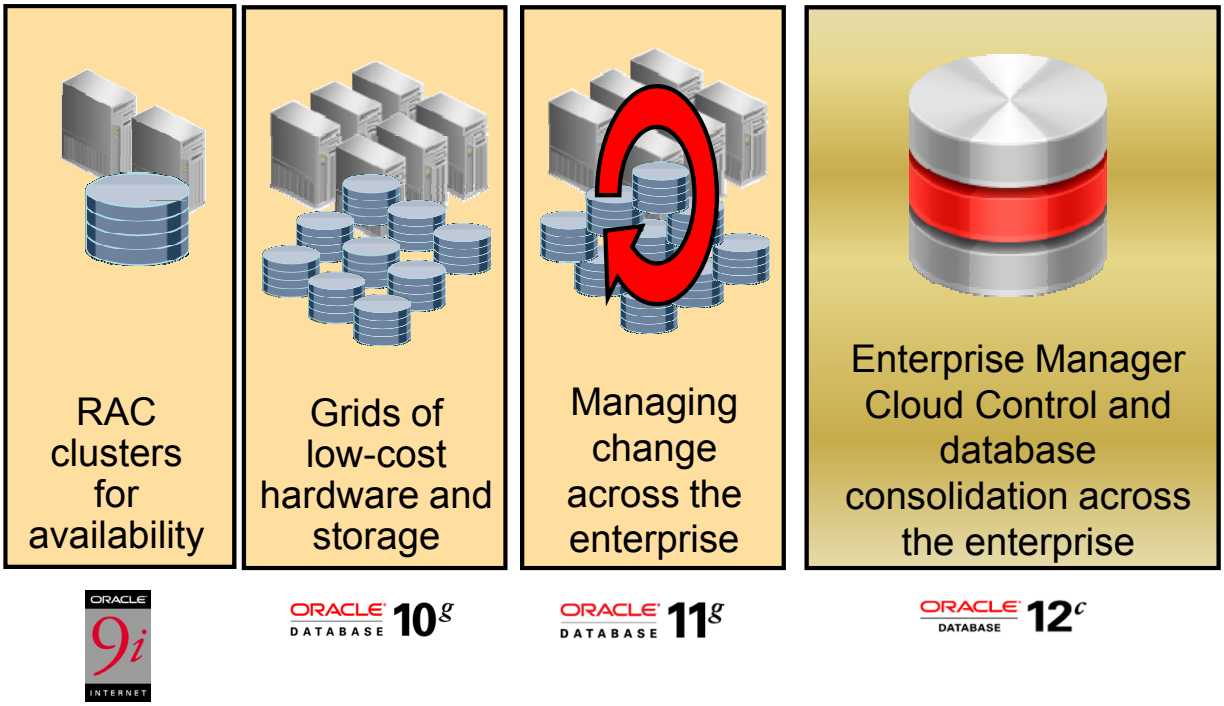


As a result of its early focus on innovation, Oracle has maintained the lead in the industry with a large number of trend-setting products.

Some of the marquee areas in the Oracle Database 12*c* release are the following:

- Private Database Cloud
- Defense in Depth including Oracle Data Redaction, Real Application Security
- Information Lifecycle Management (ILM), which includes hot/cold data classification, declarative compression and tiering, In-database Archiving, and Valid-Time Temporal
- Flex Clusters
- Extreme Availability, which includes Data Guard Far-Sync and Application Continuity
- Lower Cost Migrations
- Performance and Ease of Use, which includes “just-in-time” optimizations, attribute clustering, and zone maps for Exadata only

Enterprise Cloud Computing



Oracle Database 10*g* was the first database management system designed for grid computing.

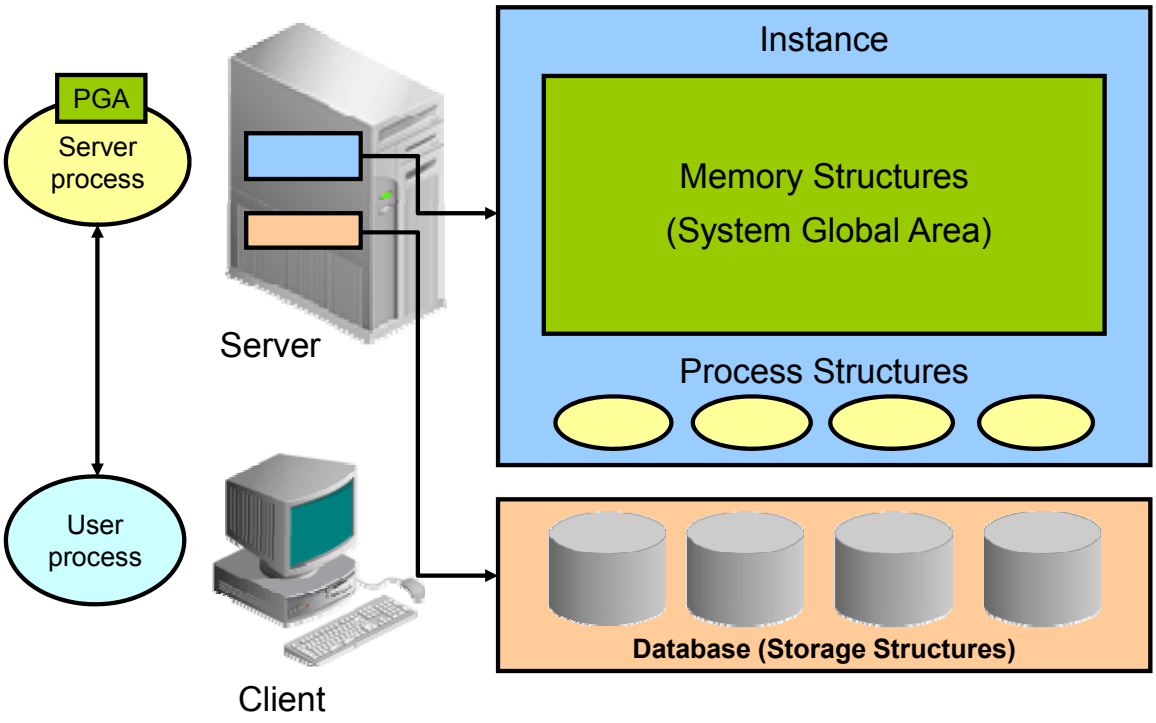
Oracle Database 11*g* consolidates and extends Oracle's unique ability to deliver the benefits of grid computing, transforming data centers from silos of isolated system resources to shared pools of servers and storage.

Oracle Database 12*c* and Enterprise Manager Cloud Control are designed for cloud computing. Cloud computing creates a complete, pre-integrated, off-the-shelf private cloud solution that allows you to quickly transform the enterprise data center into a private cloud.

The key benefits are the following:

- Reduce server sprawl and improve CPU utilization by consolidating on fewer servers.
- Reduce the amount of time a DBA spends installing and configuring databases, by automating deployment of standard database configurations.
- A single console manages the entire Cloud life cycle—plan, set up, deliver, and operate.
- Prevent resource hogging by setting quotas for individual users.
- Forecast future resource needs by analyzing trending reports.
- Compute chargeback based on performance and configuration metrics.

Oracle Database Server Architecture: Overview



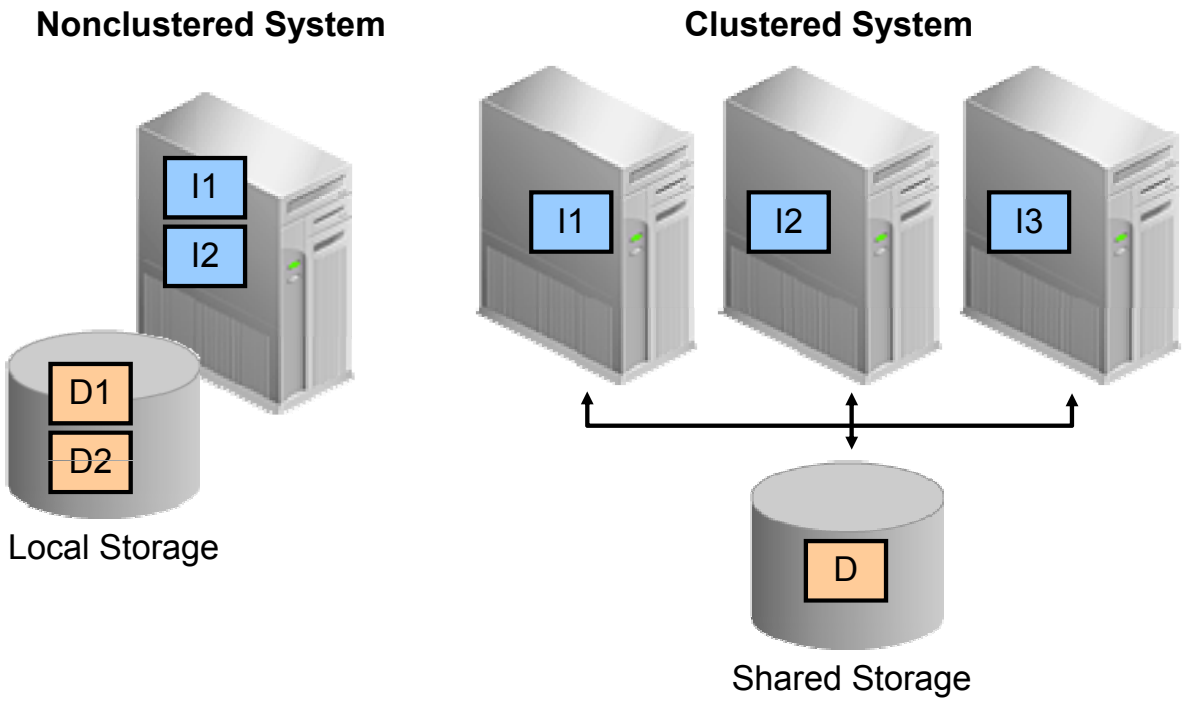
There are three major structures in Oracle Database server architecture: memory structures, process structures, and storage structures. A basic Oracle database system consists of an Oracle database and a database instance.

The database consists of both physical structures and logical structures. Because the physical and logical structures are separate, the physical storage of data can be managed without affecting access to logical storage structures.

The instance consists of memory structures and background processes associated with that instance. Every time an instance is started, a shared memory area called the System Global Area (SGA) is allocated and the background processes are started. Processes are jobs that work in the memory of computers. A process is defined as a “thread of control” or a mechanism in an operating system that can run a series of steps. After starting a database instance, the Oracle software associates the instance with a specific physical database. This is called *mounting the database*. The database is then ready to be opened, which makes it accessible to authorized users.

Note: Oracle Automatic Storage Management (ASM) uses the concept of an instance for the memory and process components, but is not associated with a specific database.

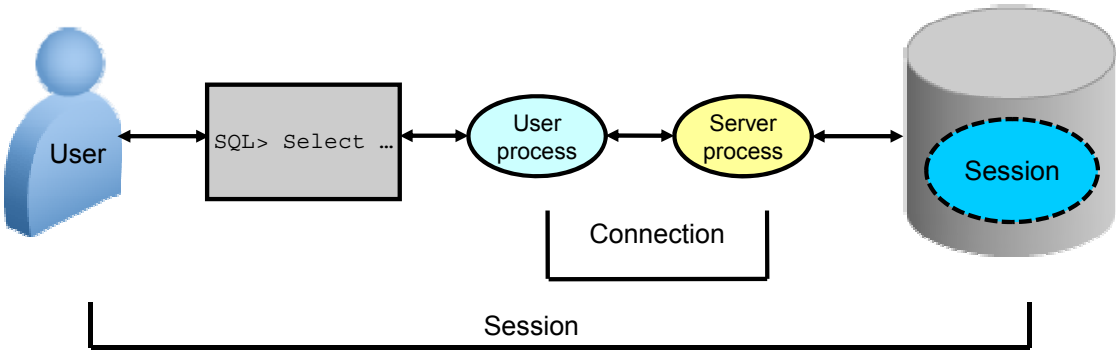
Oracle Database Instance Configurations



Each database instance is associated with only one physical database. If there are multiple databases on the same server, then there is a separate and distinct database instance for each physical database. A database instance cannot be shared. A Real Applications Cluster (RAC) database usually has multiple instances on separate servers for the same shared database. In this model, the same database is associated with each RAC instance, which meets the requirement that only one database is associated with an instance.

Connecting to the Database Instance

- **Connection:** Communication between a user process and an instance
- **Session:** Specific connection of a user to an instance through a user process



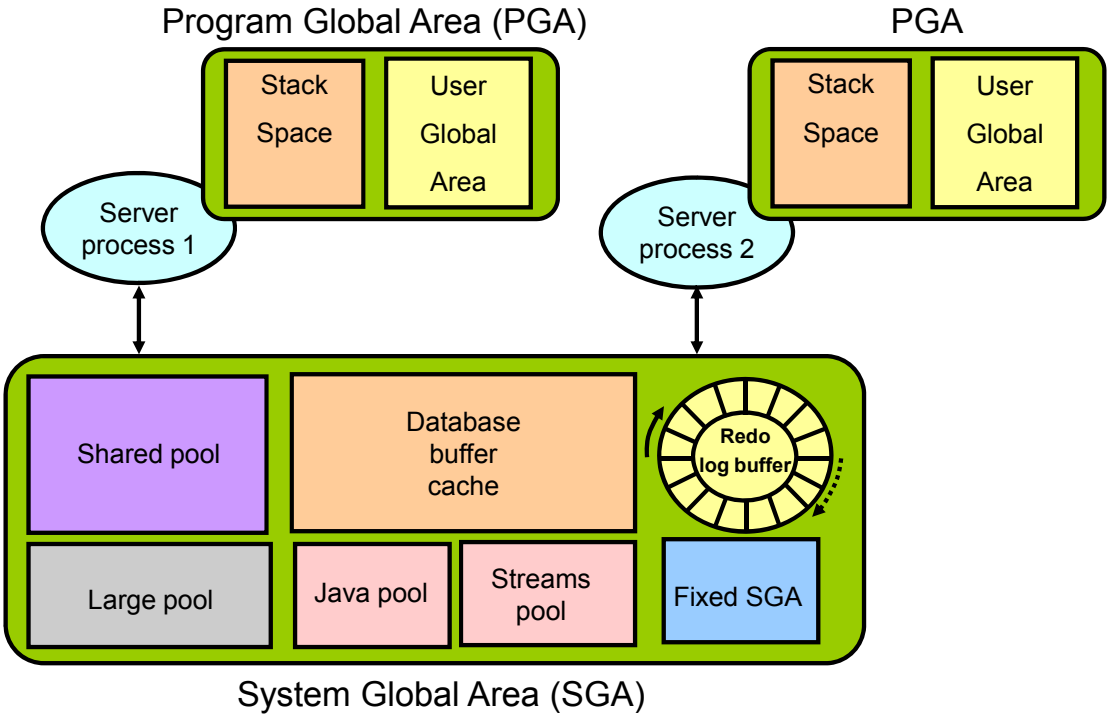
Connections and sessions are closely related to user processes but are very different in meaning.

A *connection* is a communication pathway between a user process and an Oracle Database instance. A communication pathway is established using available interprocess communication mechanisms (on a computer that runs both the user process and Oracle Database) or network software (when different computers run the database application and Oracle Database, and communicate through a network).

A *session* represents the state of a current user login to the database instance. For example, when a user starts SQL*Plus, the user must provide a valid username and password, and then a session is established for that user. A session lasts from the time a user connects until the user disconnects or exits the database application.

Multiple sessions can be created and exist concurrently for a single Oracle database user using the same username. For example, a user with the username/password of `HR`/`HR` can connect to the same Oracle Database instance several times.

Oracle Database Memory Structures



Oracle Database creates and uses memory structures for various purposes. For example, memory stores program code being run, data that is shared among users, and private data areas for each connected user.
Two basic memory structures are associated with an instance:

- **System Global Area (SGA):** Group of shared memory structures, known as SGA components, that contain data and control information for one Oracle Database instance. The SGA is shared by all server and background processes. Examples of data stored in the SGA include cached data blocks and shared SQL areas.
- **Program Global Areas (PGA):** Memory regions that contain data and control information for a server or background process. A PGA is nonshared memory created by Oracle Database when a server or background process is started. Access to the PGA is exclusive to the server process. Each server process and background process has its own PGA.

The SGA is the memory area that contains data and control information for the instance. The SGA includes the following data structures:

- **Shared pool:** Caches various constructs that can be shared among users
- **Database buffer cache:** Caches blocks of data retrieved from the database
- **Redo log buffer:** Caches redo information (used for instance recovery) until it can be written to the physical redo log files stored on the disk
- **Large pool:** Optional area that provides large memory allocations for certain large processes, such as Oracle backup and recovery operations, and I/O server processes
- **Java pool:** Used for all session-specific Java code and data in the Java Virtual Machine (JVM)
- **Streams pool:** Used by Oracle Streams to store information required by capture and apply
- **Fixed SGA:** An internal housekeeping area containing general information about the state of the database and the instance, and information communicated between processes

When you start the instance, the amount of memory allocated for the SGA is displayed.

A Program Global Area (PGA) is a memory region that contains data and control information for each server process. An Oracle server process services a client's requests. Each server process has its own private PGA that is allocated when the server process is started. Access to the PGA is exclusive to that server process, and the PGA is read and written only by the Oracle code acting on its behalf. The PGA is divided into two major areas: stack space and the user global area (UGA).

With the dynamic SGA infrastructure, the sizes of the database buffer cache, the shared pool, the large pool, the Java pool, and the Streams pool can change without shutting down the instance.

The Oracle Database server uses initialization parameters to create and manage memory structures. The simplest way to manage memory is to allow the database to automatically manage and tune it for you. To do so (on most platforms), you only have to set a target memory size initialization parameter (`MEMORY_TARGET`) and a maximum memory size initialization parameter (`MEMORY_MAX_TARGET`).

Process Architecture

- User process
  - Is the application or tool that connects to the Oracle database
- Database processes
  - Server process: Connects to the Oracle instance and is started when a user establishes a session
  - Background processes: Are started when an Oracle instance is started
- Daemon/Application processes
  - Networking listeners
  - Grid Infrastructure daemons

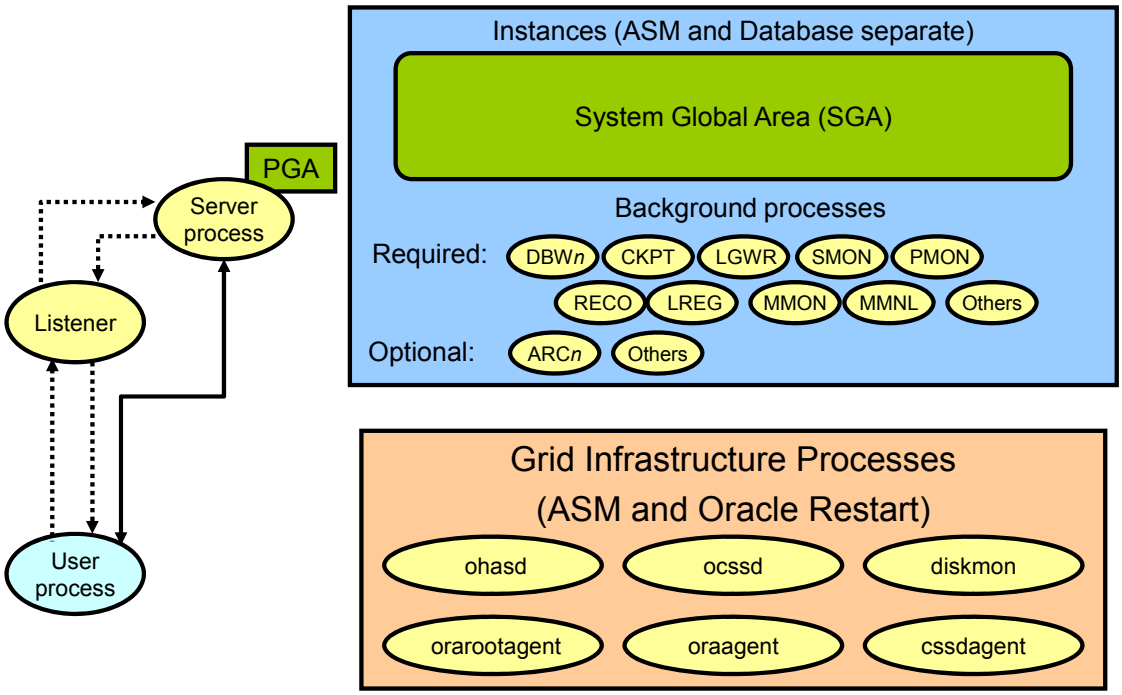
The processes in an Oracle Database system can be divided into three major groups:

- User processes that run the application or Oracle tool code
- Oracle Database processes that run the Oracle Database server code (including server processes and background processes)
- Oracle daemons and application processes not specific to a single database

When a user runs an application program or an Oracle tool such as SQL*Plus, the term *user process* is used to refer to the user's application. The user process may or may not be on the database server machine. Oracle Database also creates a *server process* to execute the commands issued by the user process. In addition, the Oracle server also has a set of *background processes* for an instance that interact with each other and with the operating system to manage the memory structures, asynchronously perform I/O to write data to disk, and perform other required tasks. The process structure varies for different Oracle Database configurations, depending on the operating system and the choice of Oracle Database options. The code for connected users can be configured as a dedicated server or a shared server.

- **Dedicated server:** For each session, the database application is run by a user process that is served by a dedicated server process that executes Oracle database server code.
- **Shared server:** Eliminates the need for a dedicated server process for each connection. A dispatcher directs multiple incoming network session requests to a pool of shared server processes. A shared server process serves any client request.

Process Structures



ORACLE

Server Processes

Oracle Database creates server processes to handle the requests of user processes connected to the instance. The user process represents the application or tool that connects to the Oracle database. It may be on the same machine as the Oracle database, or it may exist on a remote client and use a network to reach the Oracle database. The user process first communicates with a listener process that creates a server process in a dedicated environment.

Server processes created on behalf of each user's application can perform one or more of the following:

- Parse and run SQL statements issued through the application.
- Read necessary data blocks from data files on disk into the shared database buffers of the SGA (if the blocks are not already present in the SGA).
- Return results in such a way that the application can process the information.

Background Processes

To maximize performance and accommodate many users, a multiprocess Oracle Database system uses some additional Oracle Database processes called *background processes*. An Oracle Database instance can have many background processes.

The background processes commonly seen in non-RAC, non-ASM environments can include the following:

- Database Writer process (DBW*n*)
- Log Writer process (LGWR)
- Checkpoint process (CKPT)
- System monitor process (SMON)
- Process monitor process (PMON)
- Recoverer process (RECO)
- Listener registration process (LREG)
- Manageability monitor process (MMON)
- Manageability monitor lite process (MMNL)
- Job queue coordinator (CJQ0)
- Job slave processes (J*nnn*)
- Archiver processes (ARC*n*)
- Queue monitor processes (QMN*n*)
- The process spawner process (PSP0)

Other background processes may be found in more advanced configurations such as RAC. See the `V$BGPROCESS` view for more information about background processes.

Some background processes are created automatically when an instance is started, whereas others are started as required.

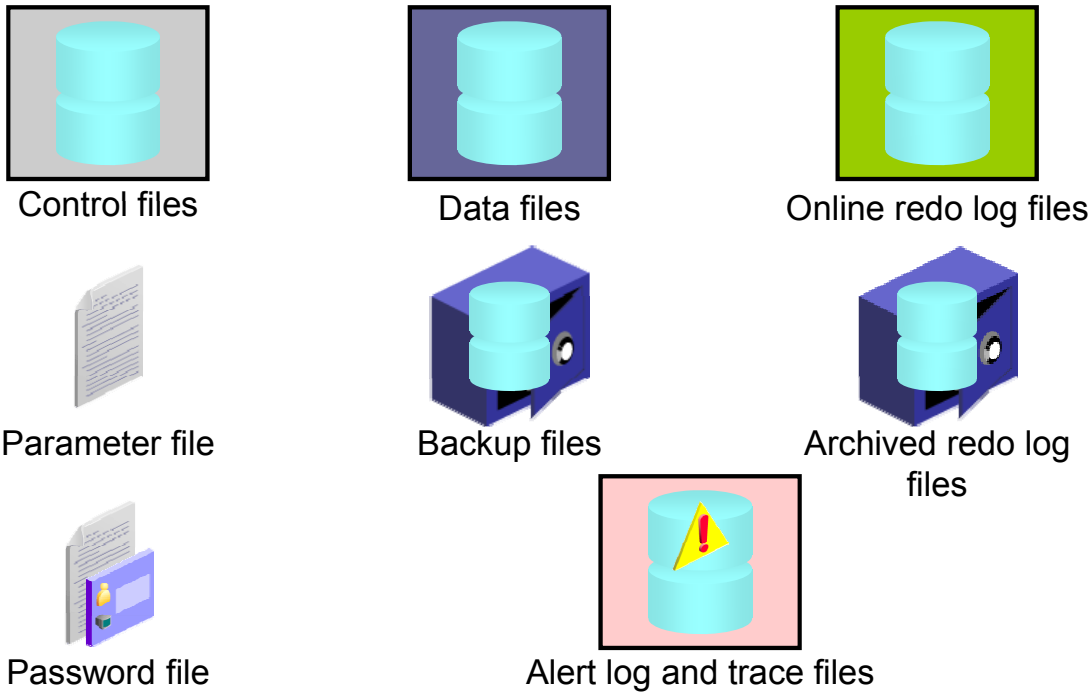
Other process structures are not specific to a single database, but rather can be shared among many databases on the same server. The Grid Infrastructure and networking processes fall into this category.

Oracle Grid Infrastructure processes on Linux and UNIX systems include the following:

- `ohasd` (Oracle High Availability Service daemon): Is responsible for starting Oracle Clusterware processes
- `ocssd`: Cluster Synchronization Service daemon
- `diskmon` (Disk Monitor daemon): Is responsible for input and output fencing for Oracle Exadata Storage
- `cssdagent`: Starts, stops, and check the status of the CSS daemon, `ocssd`
- `oraagent`: Extends clusterware to support Oracle-specific requirements and complex resources
- `orarootagent`: Is a specialized Oracle agent process that helps manage resources owned by root, such as the network.

Note: For a more detailed list of the background processes, consult the *Oracle Database Reference* guide.

Database Storage Architecture



The files that comprise an Oracle database are as follows:

- **Control files:** Each database has one unique control file that contains data about the database itself (that is, physical database structure information). Multiple copies may be maintained to protect against total loss. It can also contain metadata related to backups. The control file is critical to the database. Without the control file, the database cannot be mounted or opened.
- **Data files:** Contain the user or application data of the database, as well as metadata and the data dictionary
- **Online redo log files:** Allow for instance recovery of the database. If the database server crashes and does not lose any data files, the instance can recover the database with the information in these files.

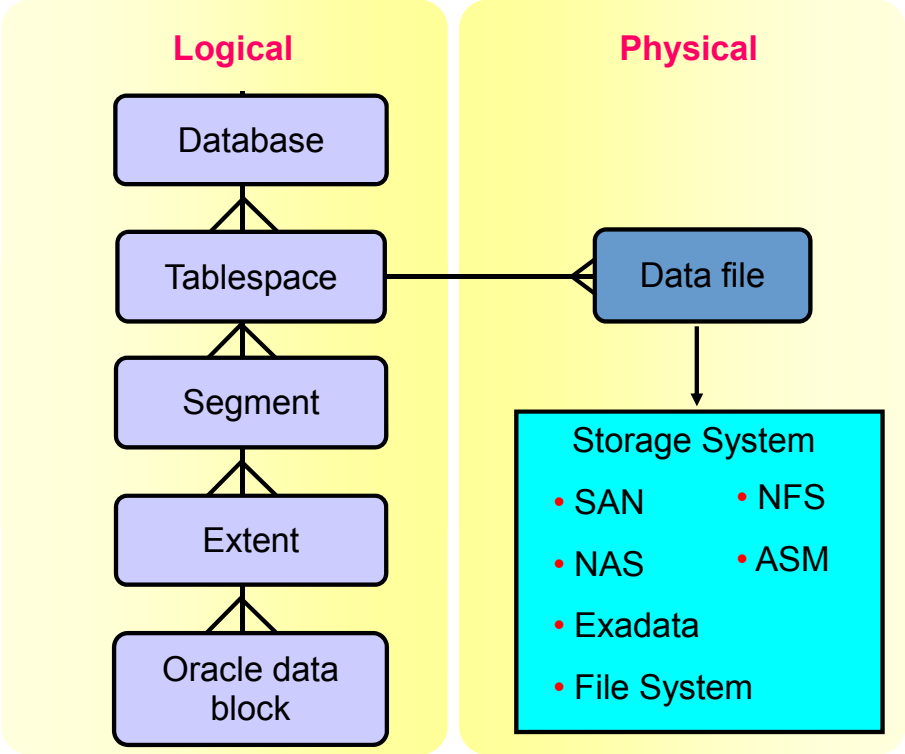
The following additional files are used during the operation of the database:

- **Parameter file:** Is used to define how the instance is configured when it starts up
- **Password file:** Allows users using the `SYSDBA`, `SYSOPER`, `SYSBACKUP`, `SYSDG`, `SYSKM`, and `SYSASM` roles to connect remotely to the instance and perform administrative tasks
- **Backup files:** Are used for database recovery. You typically restore a backup file when a media failure or user error has damaged or deleted the original file.

- **Archived redo log files:** Contain an ongoing history of the data changes (redo) that are generated by the instance. Using these files and a backup of the database, you can recover a lost data file. That is, archive logs enable the recovery of restored data files.
- **Trace files:** Each server and background process can write to an associated trace file. When an internal error is detected by a process, the process dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database administrator, whereas other information is for Oracle Support Services.
- **Alert log file:** These are special trace entries. The alert log of a database is a chronological log of messages and errors. It is recommended that you review the alert log periodically.

Note: Parameter, password, alert, and trace files are covered in other lessons.

Logical and Physical Database Structures



The database has logical structures and physical structures.

Databases, Tablespaces, and Data Files

The relationship among databases, tablespaces, and data files is illustrated in the slide. Each database is logically divided into two or more tablespaces. One or more data files are explicitly created for each tablespace to physically store the data of all segments in a tablespace. If it is a `TEMPORARY` tablespace, it has a temporary file instead of a data file. A tablespace's data file can be physically stored on any supported storage technology.

Tablespaces

A database is divided into logical storage units called *tablespaces*, which group related logical structures or data files together. For example, tablespaces commonly group all of an application's segments to simplify some administrative operations.

Data Blocks

At the finest level of granularity, an Oracle database's data is stored in *data blocks*. One data block corresponds to a specific number of bytes of physical space on the disk. A data block size is specified for each tablespace when it is created. A database uses and allocates free database space in Oracle data blocks.

Extents

The next level of logical database space is an *extent*. An extent is a specific number of contiguous Oracle data blocks (obtained in a single allocation) that are used to store a specific type of information. Oracle data blocks in an extent are logically contiguous but can be physically spread out on disk because of RAID striping and file system implementations.

Segments

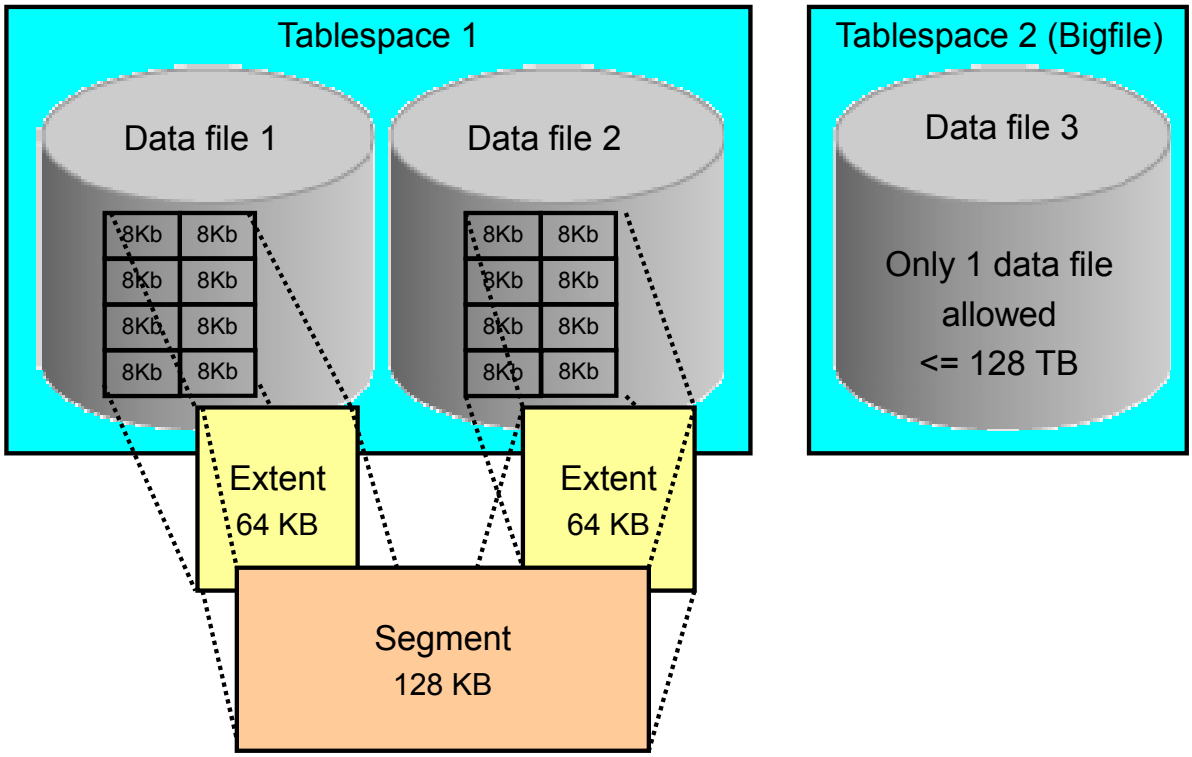
The level of logical database storage above an extent is called a *segment*. A segment is a set of extents allocated for a certain logical structure. Example:

- **Data segments:** Each nonclustered, non-index-organized table has a data segment, with the exception of external tables, global temporary tables, and partitioned tables (in which each table has one or more segments). All of the table's data is stored in the extents of its data segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in the cluster's data segment.
- **Index segments:** Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.
- **Undo segments:** One `UNDO` tablespace is created for each database instance. This tablespace contains numerous undo segments to temporarily store undo information. The information in an undo segment is used to generate read-consistent database information and, during database recovery, to roll back uncommitted transactions for users.
- **Temporary segments:** Temporary segments are created by the Oracle database when a SQL statement needs a temporary work area to complete execution. When the statement finishes execution, the temporary segment's extents are returned to the database for future use. Specify either a default temporary tablespace for every user, or a default temporary tablespace that is used database-wide.

Note: There are other types of segments not listed here. There are also schema objects such as views, packages, triggers, and so on that are not considered segments even though they are database objects. A segment owns its respective disk space allocation. The other objects exist as rows stored in a system metadata segment.

The Oracle Database server dynamically allocates space. When the existing extents of a segment are full, additional extents are added. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on the disk, and they can come from different data files belonging to the same tablespace.

Tablespaces and Data Files



A database is divided into *tablespaces*, which are logical storage units that can be used to group related logical structures. One or more data files are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace.

The graphic in the slide illustrates tablespace 1, composed of two data files. A segment of 128 KB size, composed of two extents is spanning the two data files. The first extent of size 64 KB is in the first data file and the second extent, also of size 64 KB is in the second data file. Both extents are formed from contiguous 8 KB Oracle blocks.

Note: You can also create bigfile tablespaces, which have only one file that is often very large. The file may be any size up to the maximum that the row ID architecture permits. The maximum size is the block size for the tablespace multiplied by 2^{36}, or 128 TB for a 32 KB block size. Traditional smallfile tablespaces (which are the default) may contain multiple data files, but the files cannot be as large. For more information about bigfile tablespaces, see the *Oracle Database Administrator's Guide*.

SYSTEM and SYSAUX Tablespaces

- The SYSTEM and SYSAUX tablespaces are mandatory tablespaces that are created at the time of database creation. They must be online.
- The SYSTEM tablespace is used for core functionality (for example, data dictionary tables).
- The auxiliary SYSAUX tablespace is used for additional database components.
- The SYSTEM and SYSAUX tablespaces should not be used for application data.

Each Oracle database must contain a SYSTEM tablespace and a SYSAUX tablespace. They are automatically created when the database is created. The system default is to create a smallfile tablespace. You can also create bigfile tablespaces, which enable the Oracle database to manage ultra-large files.

A tablespace can be online (accessible) or offline (not accessible). The SYSTEM tablespace is always online when the database is open. It stores tables that support the core functionality of the database, such as the data dictionary tables.

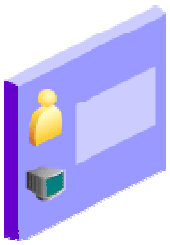
The SYSAUX tablespace is an auxiliary tablespace to the SYSTEM tablespace. The SYSAUX tablespace stores many database components, and it must be online for the correct functioning of all database components. The SYSTEM and SYSAUX tablespaces are not recommended for storing an application's data. Additional tablespaces can be created for this purpose.

Note: The SYSAUX tablespace may be taken offline to perform tablespace recovery, whereas this is not possible for the SYSTEM tablespace. Neither of them may be made read-only.

Database User Accounts

Each database user account has:

- A unique username
- An authentication method
- A default tablespace
- A temporary tablespace
- A user profile
- An initial consumer group
- An account status



A schema:

- Is a collection of database objects that are owned by a database user
- Has the same name as the user account

ORACLE

To access the database, a user must specify a valid database user account and successfully authenticate as required by that user account. Each database user has a unique database account.
Oracle recommends this to avoid potential security holes and provide meaningful data for certain audit activities. However, users may sometimes share a common database account. In these rare cases, the operating system and applications must provide adequate security for the database. Each user account has:

- **Unique username:** Usernames cannot exceed 30 bytes, cannot contain special characters, and must start with a letter.
- **Authentication method:** The most common authentication method is a password. Oracle Database supports password, global, and external authentication methods (such as biometric, certificate, and token authentication).
- **Default tablespace:** This is a place where a user creates objects if the user does not specify some other tablespace. Note that having a default tablespace does not imply that the user has the *privilege* of creating objects in that tablespace, nor does it mean that the user has a *quota* of space in that tablespace in which to create objects. Both of these are granted separately.

- **Temporary tablespace:** This is a place where temporary objects, such as sorts and temporary tables, are created on behalf of the user by the instance. No quota is applied to temporary tablespaces.
- **User profile:** This is a set of resource and password restrictions assigned to the user.
- **Initial consumer group:** This is used by the Resource Manager.
- **Account status:** Users can access only “open” accounts. The account status may be “locked” and/or “expired.”

Schemas: A *schema* is a collection of database objects that are owned by a database user. Schema objects are the logical structures that directly refer to the database’s data. Schema objects include such structures as tables, views, sequences, stored procedures, synonyms, indexes, clusters, and database links. In general, schema objects include everything that your application creates in the database.

Note: A database user is not necessarily a person. It is a common practice to create a user that owns the database objects of a particular application, such as HR. The database user can be a device, an application, or just a way to group database objects for security purposes. The personal identifying information of a person is not needed for a database user.

Predefined Administrative Accounts

- SYS:
  - Owns the data dictionary and the Automatic Workload Repository (AWR)
  - Used for startup and shutdown of the database instance
- SYSTEM: Owns additional administrative tables and views
- SYSBACKUP: Facilitates Oracle Recovery Manager (RMAN) backup and recovery operations
- SYSDG: Facilitates Oracle Data Guard operations
- SYSKM: Facilitates Transparent Data Encryption wallet operations

The SYS and SYSTEM accounts have the database administrator (DBA) role granted to them by default. In addition, the SYS account has all privileges with ADMIN OPTION and owns the data dictionary. To connect to the SYS account, you must use the AS SYSDBA clause for a database instance and AS SYSASM for an Automatic Storage Management (ASM) instance. Any user that is granted the SYSDBA privilege can connect to the SYS account by using the AS SYSDBA clause. Only "privileged" users who are granted the SYSDBA, SYSOPER, SYSASM , SYSBACKUP, or SYSDG privileges are allowed to start up and shut down instances. The SYSTEM account does not have the SYSDBA privilege. SYSTEM is also granted the AQ_ADMINISTRATOR_ROLE and MGMT_USER roles. The SYS and SYSTEM accounts are required accounts in the database. They cannot be dropped.

Best practice: Applying the principle of least privilege, these accounts are not used for routine operations. Users who need DBA privileges have separate accounts with the required privileges granted to them.

The SYSBACKUP, SYSDG, and SYSKM users are created to facilitate separation of duties for database administrators. Each of these provides a designated use for an administrative privilege by the same name. You should create a user and grant the appropriate administrative privilege to that user.

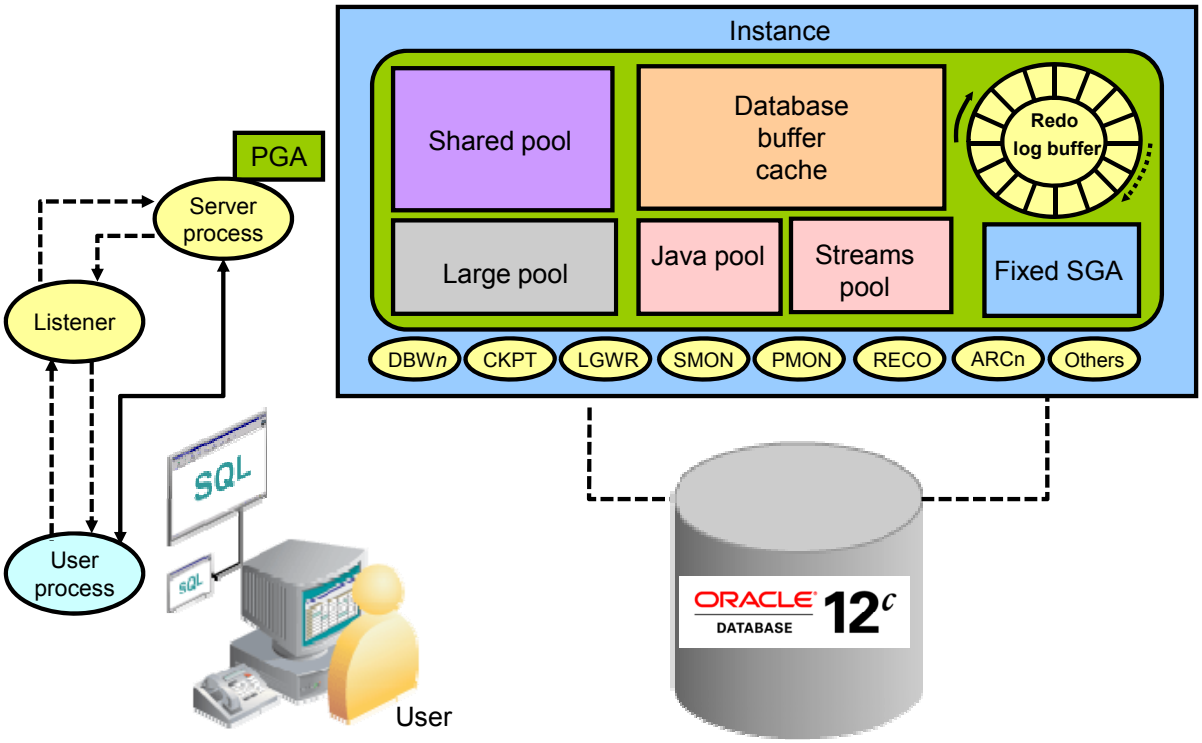
Administrative Privileges

Privilege	Description
SYSDBA	Standard database operations, such as starting and shutting down the database instance, creating the server parameter file (SPFILE), and changing the ARCHIVELOG mode Allows the grantee to view user data
SYSOPER	Standard database operations, such as starting and shutting down the database instance, creating the server parameter file (SPFILE), and changing the ARCHIVELOG mode
SYSBACKUP	Oracle Recovery Manager (RMAN) backup and recovery operations by using RMAN or SQL*Plus
SYSDG	Data Guard operations by using the Data Guard Broker or the DGMGRL command-line interface
SYSKM	Manage Transparent Data Encryption wallet operations

Oracle Database includes five administrative privileges that are provided to facilitate separation of duty. The SYSDBA and SYSOPER administrative privileges are used to perform a variety of standard database operations including starting up the database instance and shutting it down. Refer to the *Oracle Database Administrator's Guide* for a complete list of authorized operations for the SYSDBA and SYSOPER privileges.

SYSBACKUP, SYSDG, and SYSKM are new to Oracle Database 12*c* and are tailored for the specific administrative tasks of backup and recovery, Oracle Data Guard, and Transparent Data Encryption key management. In previous releases, the SYSDBA privilege was required for these tasks. These privileges enable you to connect to the database even if the database is not open. Refer to the *Oracle Database Security Guide* for a list of supported operations for the SYSBACKUP, SYSDG, and SYSKM privileges.

Interacting with an Oracle Database: Memory, Processes, and Storage



The following example describes Oracle database operations at the most basic level. It illustrates an Oracle database configuration in which the user and associated server process are on separate computers, connected through a network.

1. An instance has started on a node where Oracle Database is installed, often called the *host* or *database server*.
2. A user starts an application spawning a user process. The application attempts to establish a connection to the server. (The connection may be local, client/server, or a three-tier connection from a middle tier.)
3. The server runs a listener that has the appropriate Oracle Net Services handler. The listener detects the connection request from the application and creates a dedicated server process on behalf of the user process.
4. The user runs a DML-type SQL statement and commits the transaction. For example, the user changes the address of a customer in a table and commits the change.
5. The server process receives the statement and checks the shared pool (an SGA component) for any shared SQL area that contains an identical SQL statement. If a shared SQL area is found, the server process checks the user's access privileges to the requested data, and the existing shared SQL area is used to process the statement. If a shared SQL area is not found, a new shared SQL area is allocated for the statement so that it can be parsed and processed.

6. The server process retrieves any necessary data values, either from the actual data file (table) or from values stored in the database buffer cache.
7. The server process modifies data in the SGA. Because the transaction is committed, the Log Writer process (LGWR) immediately records the transaction in the redo log file. The Database Writer process (DBW*n*) writes modified blocks permanently to disk when it is efficient to do so.
8. If the transaction is successful, the server process sends a message across the network to the application. If it is not successful, an error message is transmitted.
9. Throughout this entire procedure, the other background processes run, watching for conditions that require intervention. In addition, the database server manages other users' transactions and prevents contention between transactions that request the same data.

Multitenant Architecture: Introduction

- *Multitenent container database (CDB):*
  - Oracle Database 12*c* architecture that consolidates databases
  - Has a database instance and database files at the physical level
- *Pluggable database (PDB)* is a:
  - Database consolidated within a CDB
  - Set of database schemas that appears logically to users and applications as a separate database
- All pluggable databases and the root container share:
  - Background processes
  - Shared/process memory
  - Oracle metadata

ORACLE

A pluggable database (PDB) is a set of database schemas and tablespaces that appears to users and applications as a separate database. But at the physical level, the multitenant container database (CDB) has a database instance and database files, just as a traditional database (non-CDB) does.

It is easy to plug a non-CDB into a CDB as a PDB.

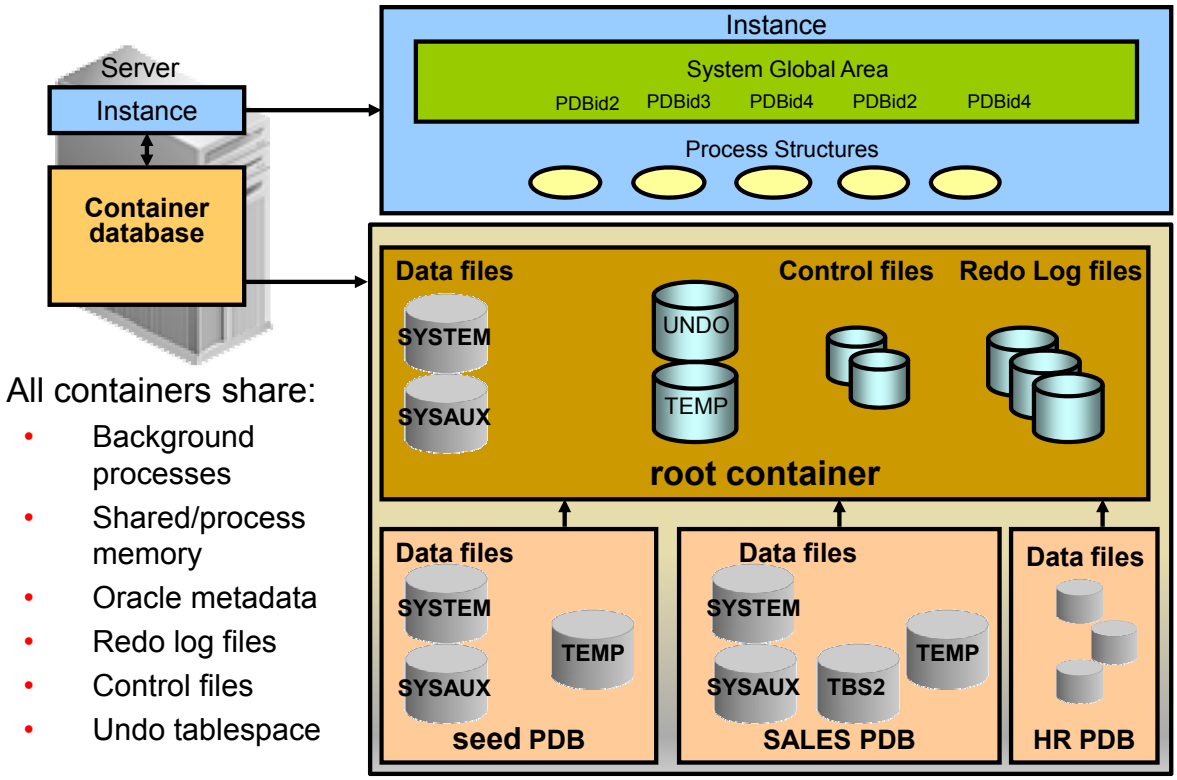
A CDB avoids redundancy of:

- Background processes
- Memory allocation
- Oracle metadata in several data dictionaries

A CDB grouping several PDBs has one instance, consequently one set of background processes, one SGA allocation, and one data dictionary in the root container that is common for all PDB. Each PDB maintains its own application data dictionary.

When applications need to be patched or upgraded, the maintenance operation is performed only once on the CDB and, consequently, all applications are updated at the same time.

Multitenant Architecture



All containers share:

- Background processes
- Shared/process memory
- Oracle metadata
- Redo log files
- Control files
- Undo tablespace

The graphic in the slide shows a CDB with four containers: the root, the seed, and two PDBs. There is only one instance for all the PDBs. The two applications (HR and SALES) each use a PDB and are maintained separately.

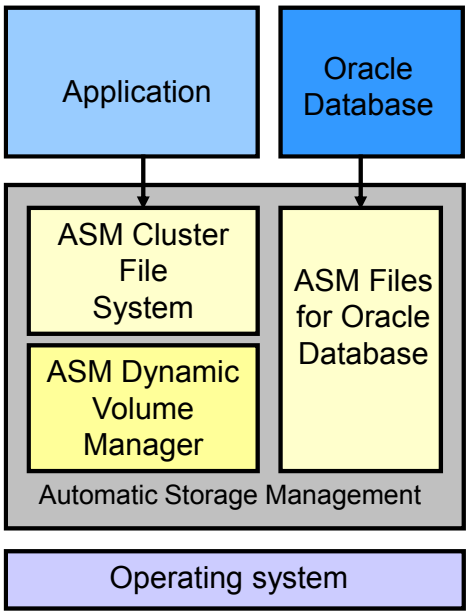
At the physical level, the CDB has a database instance and database files, just as a non-CDB does.

- The redo log files are common for the whole CDB. The redo log records are annotated with the identity of the PDB where a change occurs. All PDBs in a CDB share the ARCHIVELOG mode of the CDB.
- The control files are common for the whole CDB. The control files are updated to reflect any additional tablespace and data files of plugged PDBs.
- The UNDO tablespace is common for all containers.
- A temporary tablespace common to all containers is required. But each PDB can have its own temporary tablespace for its own local users.
- Each container (PDB) has its own data dictionary stored in its `SYSTEM` tablespace, containing its own metadata, and each PDB has a `SYSAUX` tablespace.
- The PDBs can create tablespaces within the PDB according to application needs.
- Each data file is associated with a specific container, which has a container name and container identifier.

Refer to the *Oracle Database 12c: Managing Pluggable Databases* course for detailed information.

Automatic Storage Management

- Is a portable and high-performance cluster file system
- Manages Oracle database files
- Manages application files with ASM Cluster File System (ACFS)
- Spreads data across disks to balance load
- Can mirror data to protect against failures
- Solves storage management challenges



ORACLE

Automatic Storage Management (ASM) provides vertical integration of the file system and the volume manager for Oracle database files. ASM can provide management for single symmetric multiprocessing (SMP) machines or across multiple nodes of a cluster for Oracle Real Application Clusters (RAC) support.

Oracle ASM Cluster File System (ACFS) is a multi-platform, scalable file system, and storage management technology that extends ASM functionality to support application files outside of the Oracle Database such as executables, reports, BFILEs, video, audio, text, images, and other general-purpose application file data.

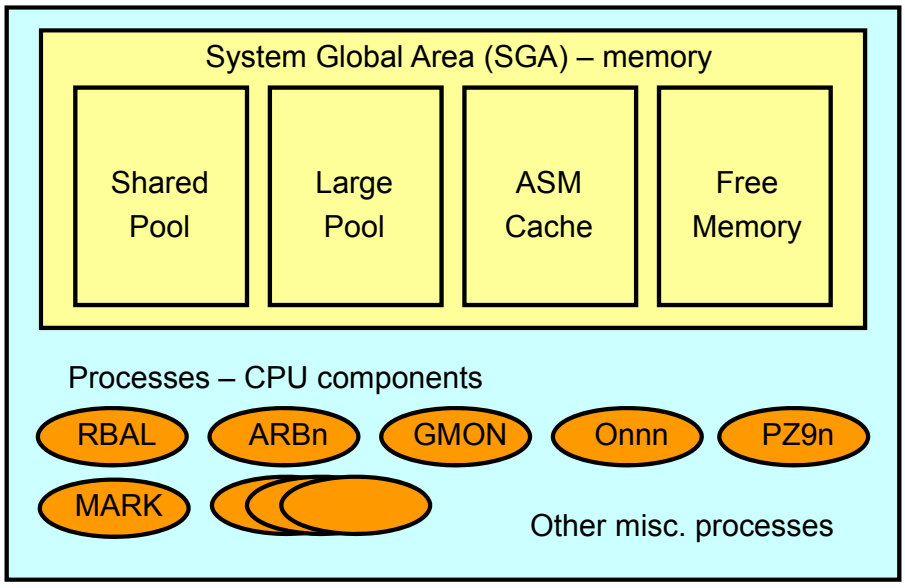
ASM distributes input/output (I/O) load across all available resources to optimize performance while removing the need for manual I/O tuning. ASM helps database administrators (DBAs) manage a dynamic database environment by enabling them to increase the database size without having to shut down the database to adjust storage allocation.

ASM can maintain redundant copies of data to provide fault tolerance, or it can be built on top of vendor-supplied storage mechanisms. Data management is done by selecting the desired reliability and performance characteristics for classes of data rather than with human interaction on a per-file basis.

ASM capabilities save the DBA's time by automating manual storage and thereby increasing the administrator's ability to manage more and larger databases with increased efficiency.

ASM Instance

The ASM instance is a combination of the process and memory components for ASM.



ASM Instance

Every time ASM or a database is started, a shared memory area called the system global area (SGA) is allocated and Oracle ASM or database background processes are started. The combination of the background processes and the SGA is called an Oracle ASM instance or an Oracle Database instance. The instance represents the CPU and RAM components of a running ASM environment.

The SGA in an ASM instance is different in memory allocation and usage than the SGA in a database instance. The SGA in the ASM instance is divided into four primary areas as follows:

- **Shared Pool:** Used for metadata information
- **Large Pool:** Used for parallel operations
- **ASM Cache:** Used for reading and writing blocks during rebalance operations
- **Free Memory:** Unallocated memory available

The minimum recommended amount of memory for an ASM instance is 256 MB. Automatic memory management is enabled by default on an ASM instance and will dynamically tune the sizes of the individual SGA memory components. The amount of memory that is needed for an ASM instance will depend on the amount of disk space being managed by ASM.

The second part of the ASM instance is the background processes. An ASM instance can have many background processes; not all of which are always present.

The background processes specific to ASM functionality are covered in the next slide. There are required background processes and optional background processes. Some of these processes may include the following:

- **ARC*n*:** The archiver processes
- **CKPT:** The checkpoint process
- **DBW*n*:** The database writer processes
- **DIAG:** The diagnosability process
- **Jnnn:** Job queue processes
- **LGWR:** The log writer process
- **PMON:** The process monitor process
- **PSP0:** The process spawner process
- **QMN*n*:** The queue monitor processes
- **RECO:** The recoverer process
- **SMON:** The system monitor process
- **VKTM:** The virtual keeper of time process
- **MMAN:** The memory manager process
- **LREG:** The listener registration process

The above list of processes is not complete. For the ASM instance, these processes will not always perform the same tasks as they would in a database instance. For example, the `LGWR` process in a database instance is responsible for copying change vectors from the log buffer section of the SGA to the online redo logs on disk. The ASM instance does not contain a log buffer in its SGA, nor does it use online redo logs. The `LGWR` process in an ASM instance copies logging information to an ASM disk group.

If ASM is clustered, then additional processes related to cluster management will be running in the ASM instance. Some of these processes include the following:

- **LMON:** The global enqueue service monitor process
- **LMD*n*:** The global enqueue service daemons
- **LMS*n*:** The global cache service processes
- **LCK*n*:** The lock processes

ASM Components: ASM Instance – Primary Processes

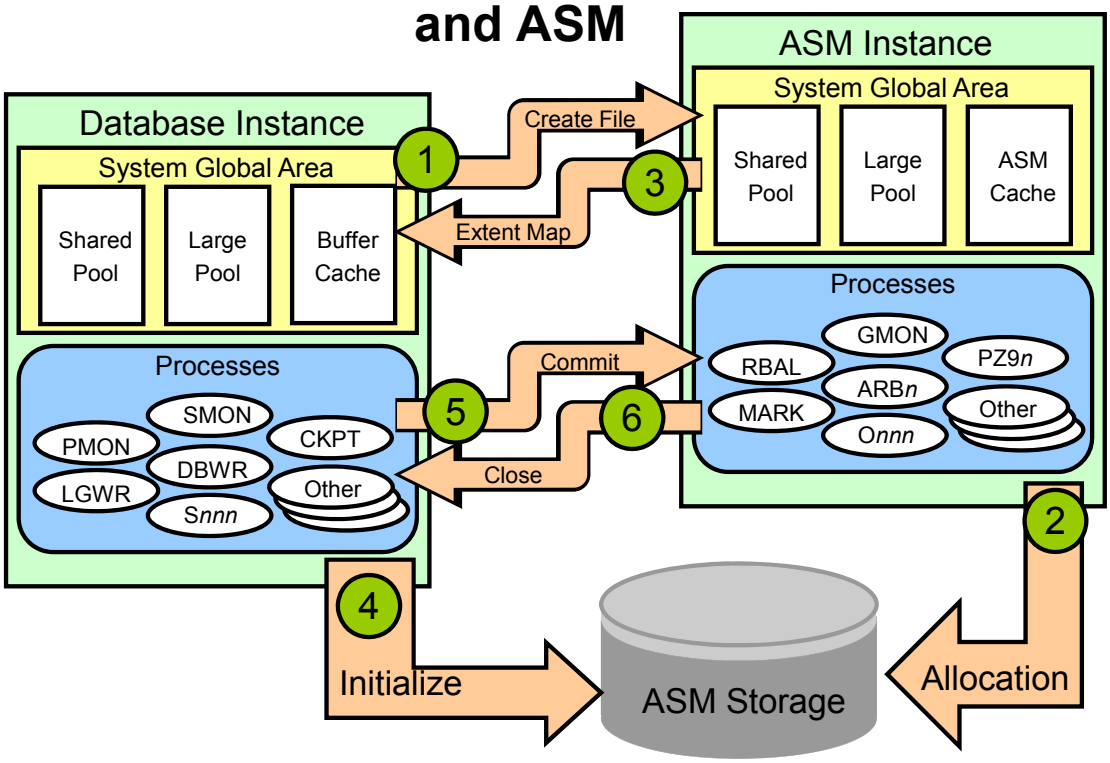
The ASM instance primary processes are responsible for ASM-related activities.

Process	Description
RBAL	Opens all device files as part of discovery and coordinates the rebalance activity
ARB*n*	One or more slave processes that do the rebalance activity
GMON	Responsible for managing the disk-level activities such as drop or offline and advancing the ASM disk group compatibility
MARK	Marks ASM allocation units as stale when needed
Onnn	One or more ASM slave processes forming a pool of connections to the ASM instance for exchanging messages

ORACLE

The ASM instance uses dedicated background processes for much of its functionality. The `RBAL` process coordinates rebalance activity for disk groups in an Automatic Storage Management instance. It performs a global open on Automatic Storage Management disks. The `ARBn` processes perform the actual rebalance data extent movements in an Automatic Storage Management instance. There can be many of these at a time, called `ARB0`, `ARB1`, and so on. The `GMON` process maintains disk membership in ASM disk groups. The `MARK` process marks ASM allocation units as stale following a missed write to an offline disk. The `Onnn` processes represent the server side of a client/server connection. These processes will appear the moment the instance is started, and will disappear after that. They form a pool of connections to the ASM instance for exchanging messages and only appear when needed.

Interaction Between Database Instances and ASM



ORACLE

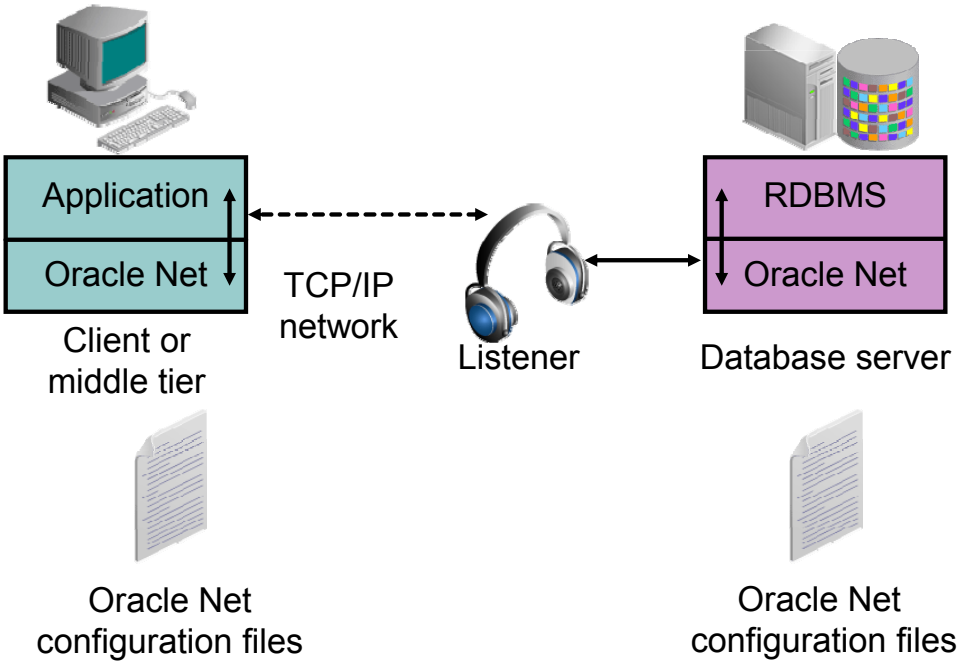
The file creation process provides an illustration of the interactions that take place between database instances and ASM. The file creation process occurs as follows:

1. The database requests file creation.
2. An ASM foreground process creates a Continuing Operation Directory (COD) entry and allocates space for the new file across the disk group.
3. The ASMB database process receives an extent map for the new file.
4. The file is now open and the database process initializes the file directly.
5. After initialization, the database process requests that the file creation is committed. This causes the ASM foreground process to clear the COD entry and mark the file as created.
6. Acknowledgment of the file commit implicitly closes the file. The database instance will need to reopen the file for future I/O.

This example reinforces two important points about the architecture of ASM:

- The Database Instance and ASM Instance work together in a coordinated fashion. A Database instance must interact with ASM to map database files to ASM extents. A Database instance also receives a constant stream of messages relating to ASM operations (such as disk group rebalancing) that may lock or move ASM extents.
- Database I/O is not channelled through the ASM instance. In fact, the database conducts I/O operations directly against ASM files, as illustrated in step 4 in the slide.

Oracle Net Services: Overview



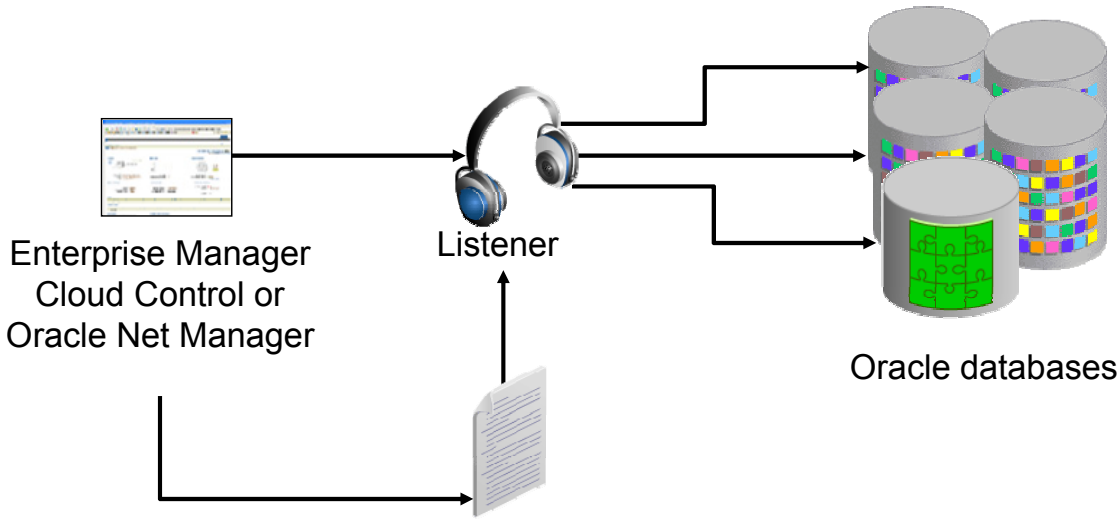
Oracle Net Services enables network connections from a client or middle-tier application to the Oracle server. After a network session is established, Oracle Net acts as the data courier for both the client application and the database server. It is responsible for establishing and maintaining the connection between the client application and database server, as well as exchanging messages between them. Oracle Net (or something that simulates Oracle Net, such as Java Database Connectivity) is located on each computer that needs to talk to the database server.

On the client computer, Oracle Net is a background component for application connections to the database.

On the database server, Oracle Net includes an active process called *Oracle Net Listener*, which is responsible for coordinating connections between the database and external applications.

The most common use of Oracle Net Services is to allow incoming database connections. You can configure additional net services to allow access to external code libraries (EXTPROC) and to connect the Oracle instance to non-Oracle data sources (such as Sybase, Informix, DB2, and SQL Server) through Oracle Heterogeneous Services.

Oracle Net Listener: Overview



Oracle Net Listener (or simply *the listener*) is the gateway to the Oracle instance for all nonlocal user connections. A single listener can service multiple database instances and thousands of client connections.

You can use Enterprise Manager Cloud Control or Oracle Net Manager to configure the listener and specify log file locations.

Advanced administrators can also configure Oracle Net Services by manually editing the configuration files, if necessary, with a standard operating system (OS) text editor such as `vi` or `gedit`.

Oracle Database Management Tools

- SQL*Plus provides an interface to your database so that you can:
  - Perform database management operations
  - Execute SQL commands to query, insert, update, and delete data in your database
- SQL Developer
  - Is a graphical user interface for accessing your instance of Oracle Database
  - Supports development in both SQL and PL/SQL
  - Is available in the default installation of Oracle Database
- Oracle Enterprise Manager Database Express
- Oracle Enterprise Manager Cloud Control
- Listener Control Utility (LSNRCTL)

SQL*Plus is a command-line program that you use to submit SQL and PL/SQL statements to an Oracle database. You can submit statements interactively or as SQL*Plus scripts. SQL*Plus is installed with the database and is located in your `$ORACLE_HOME/bin` directory.

You can start SQL*Plus from the command line, or from the Start menu on a Windows client.

SQL Developer is a graphical user interface for accessing your instance of Oracle Database. SQL Developer supports development in both the SQL and PL/SQL languages. It is available in the default installation of Oracle Database.

With SQL Developer, you can browse database objects, run SQL statements and SQL scripts, and edit and debug PL/SQL statements. You can also run any number of provided reports, as well as create and save your own.

Oracle Enterprise Manager Database Express is a lightweight administration tool. It provides an out-of-box, browser-based management solution for a single Oracle database (or database cluster), including performance monitoring, configuration management, administration, diagnostics, and tuning.

Oracle Enterprise Manager Database Express uses a web-based console, communicating with the built-in web server available in XML DB.

Oracle Enterprise Manager Cloud Control is a browser-based management tool that provides monitoring and management capabilities for Oracle and non-Oracle components. Using Enterprise Manager Cloud Control, you can monitor the health of all application components, the hosts that they run on, and the key business processes that they support.

The Listener Control Utility (`LSNRCTL`) is used to start, stop, and view the status of the listener process.

Oracle Automatic Storage Management (Oracle ASM) Tools

- Oracle ASM Configuration Assistant (ASMCA): Used to install and configure Oracle ASM instances, disk groups, and volumes
- Oracle ASM Command-Line Utility (ASMCMD): Used to manage Oracle ASM instances, disk groups, and volumes

You can use Oracle ASM Configuration Assistant (ASMCA) to install and configure Oracle ASM instances, disk groups, volumes, and Oracle Automatic Storage Management Cluster File System (Oracle ACFS).

Oracle ASM Command-Line Utility (ASMCMD) is a command-line utility that you can use to manage:

- Oracle ASM instances
- Disk groups
- File access control for disk groups
- Files and directories within disk groups
- Templates for disk groups
- Volumes

Summary

In this lesson, you should have learned how to:

- List the major architectural components of an Oracle Database server
- Describe the memory structures and background processes of an Oracle Database instance
- Correlate the logical and physical storage structures of an Oracle database
- Describe the architecture of Oracle Automatic Storage Management (ASM)
- Describe the primary function of Oracle Net Services and the listener
- List the tools that are used to administer an Oracle Database instance and an ASM instance

ORACLE

Practice 1: Overview

This practice covers the following topics:

- Listing existing instances and Oracle homes
- Determining locations of data files, control files, and redo log files
- Determining the character set of existing databases
- Determining existing listeners

2

Oracle Software Installation Basics

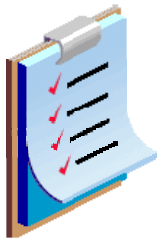
Objectives

After completing this lesson, you should be able to:

- Plan an Oracle Database and Oracle Grid Infrastructure for standalone server software installation
- Use the Oracle RDBMS Pre-Install RPM
- Configure the Oracle software owner environment
- Invoke the appropriate Oracle Universal Installer (OUI)

Planning Your Installation

- What Oracle software are you installing?
- Does the hardware involved meet the minimum required specifications?
- Is there a recommended order of installation when multiple products are involved?
- Are there prerequisite steps that must be performed by someone other than the DBA?



ORACLE

Before you begin installing Oracle software you should ask yourself the following questions to help you plan your installation:

- **What Oracle Software Are You Installing?**
  Perform an installation of Oracle Grid Infrastructure to install the components required for ASM as well as Oracle Restart.
- **Does the Hardware Involved Meet the Minimum Required Specifications?**
  Identify all hardware that will be involved in the installation process and ensure that the minimum suggested specifications are met.
- **Is There a Recommended Order of Installation When Multiple Products Are Involved?**
  Whenever possible it is recommended that the Oracle Grid Infrastructure software be installed before the Oracle Database software. Performing the installation in this order means that a newly created database can be configured to use ASM disk groups and this database is automatically registered with Oracle Restart. If Oracle Grid Infrastructure is installed after Oracle Database then manual configuration steps are required to register the database with Oracle Restart. Migration steps are required if you want this existing database to use ASM disk groups for storage.

- **Are There Prerequisite Steps That Must be Performed by Someone Other Than the DBA?**
  Following recommended separation of duty guidelines, the DBA is probably not responsible for configuring the hardware and storage devices that are to be used for your installation. Before installing Oracle Grid Infrastructure, there are required configuration steps that a Storage Administrator would perform to configure the required disk partitions. For more information, see the *Oracle Database Installation Guide* appropriate to your operating system.

  In addition, if this is the first time Oracle software is installed on your system, specific operating system groups and users may need to be created. Additional information about the operating system groups and users is provided in later lessons.

New with 12c

Configuring Oracle Linux with Oracle RDBMS Pre-Install RPM

To configure a server using Oracle Linux and the Oracle RDBMS Pre-Install RPM:

1. Install Oracle Linux.
2. Register your Linux distribution with Oracle Unbreakable Linux Network (ULN), or download and configure the yum repository for your system.
3. Install the RPM:
   - Oracle RDBMS Pre-Install RPM for Oracle Linux 6
   - Oracle Validated RPM with the RPM for Oracle Linux 5

Oracle recommends that you install Oracle Linux 6 or Oracle Linux 5, and use Oracle RPMs to configure your operating systems for Oracle Database and Oracle Grid Infrastructure installations. For Oracle Linux 6, install the Oracle RDBMS Server 12*c* Release 1 Pre-Install RPM package. For Oracle Linux 5, install the Oracle Validated RPM.

Oracle RPMs automatically install any additional packages that are needed to install Oracle Grid Infrastructure and Oracle Database, and configure your server operating system automatically, including setting kernel parameters and other basic operating system requirements for installation.

Perform the following steps to configure a server using Oracle Linux and the Oracle RDBMS Pre-Install RPM:

1. Install Oracle Linux.
2. Register your Linux distribution with Oracle Unbreakable Linux Network (ULN), or download and configure the yum repository for your system using the Oracle public yum repository for your Oracle Linux release.
3. Install the Oracle RDBMS Pre-Install RPM or Oracle Validated RPM with the RPM for your Oracle Grid Infrastructure and Oracle Database releases, and update your Linux release.

Refer to the *Oracle Database Installation Guide* for additional information.

Note: Other platforms have pre-installation requirements that are described in the platform-specific installation guide.

New with 12c

Operating System Groups and Users

Oracle Pre-Install RPM configures:

- `oracle`: Oracle Database installation owner
- `oinstall`: Oracle Inventory group
- `dba`: Oracle administrative privileges group

The Oracle Pre-Install RPM configures the following operating system users and groups:

- Oracle Database installation owner named `oracle`
- Oracle Inventory group named `oinstall`
- Oracle administrative privileges group named `dba`

You can use these users and groups for your installation, or create your own users and groups to enforce job role separation. Additional information about using job role separation is provided in the next two lessons.

Environment Variables

The following Oracle environment variables are suggested by the OUI during installation:

- `ORACLE_BASE`: Base of the Oracle directory structure
- `ORACLE_HOME`: Oracle Grid Infrastructure home directory or Oracle Database home directory depending on the product that is being installed

During installation of Oracle software, you are prompted for values that determine the value of the following environment variables:

- `ORACLE_BASE`: Specifies the base of the Oracle directory structure for all Oracle software and configuration-related files
- `ORACLE_HOME`: Specifies the location for storing Oracle software files separate from configuration files

Configuring the Oracle Software Owner Environment

Prior to installing the Oracle software, configure the Oracle software owner environment:

- Set the default file mode creation mask (`umask`) to `022` in the shell startup file.
- Unset Oracle environment variables (`ORACLE_HOME`, `ORACLE_BASE`, `ORACLE_SID`, and `TNS_ADMIN`).
- Remove `$ORACLE_HOME/bin` from your `PATH` environment variable.

ORACLE

If you have an existing Oracle installation on your system, and you are using the same user account to perform a new installation, unset the `ORACLE_HOME`, `ORACLE_BASE`, `ORACLE_SID`, `TNS_ADMIN` environment variables, and any other environment variables for Oracle software homes.

Using Oracle Universal Installer (OUI)

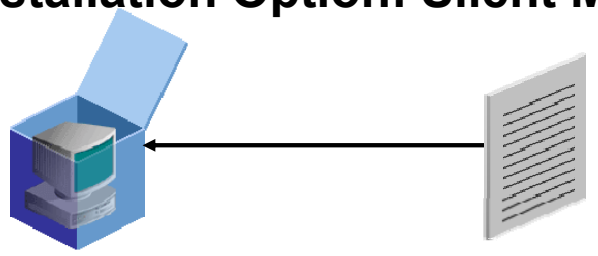
Oracle Universal Installer (OUI) is a Java application that performs component-based installations and enables different levels of integrated bundle, suite, and Web-based installations as well as complex logic in a single package. The installation engine is easily portable across all Java-enabled platforms, and platform-specific issues can be encapsulated from the overall installation process.

OUI provides the following capabilities for addressing software management and distribution:

- Automatic dependency resolution and complex logic handling
- Installation from the Web
- Component and suite installations
- Support for multiple Oracle homes
- NLS or globalization support
- Support for distributed installations
- Unattended “silent” installations that use response files

The example in the slide shows invoking OUI to install Oracle Grid Infrastructure.

Installation Option: Silent Mode



To install and configure Oracle products with OUI in silent mode, perform the following steps:

1. Prepare a response file by:
   - Using one of the file templates that is delivered with the Oracle software
   - Recording a response file during an installation process using the OUI by clicking Save Response File on the summary page
2. Run OUI in silent or response file mode.
   `./runInstaller -silent -responsefile <filename>`
   If required, run ASMCA, DBCA, and NetCA in silent mode.

To install and configure Oracle products by using OUI in silent or response file mode:

1. Prepare a response file. This can be done using one of the file templates that are provided for each product and installation type, such as `enterprise.rsp`, `standard.rsp`, and `netca.rsp`. If these templates are not appropriate for your scenario, you can use OUI in interactive mode to record a response file that you can edit and then use to complete silent-mode or response file-mode installations. This is done by clicking Save Response File on the summary page during an interactive installation using the OUI.
2. Run OUI in silent or response file mode. Run `$ORACLE_BASE/oraInventory/orainstRoot.sh` and `$ORACLE_HOME/root.sh` at the end of the installation. If you completed a software-only installation, run Oracle Net Configuration Assistant (NetCA) and the Database Configuration Assistant (DBCA) in silent or noninteractive mode if required.

For more information, see your OS-specific *Oracle Database Installation Guide*.

Quiz

The universal installer performs all required configuration for installing Oracle software.

a. True
b. False

Answer: b

Quiz Comments

The operating system users and groups, for example, must be created on UNIX and Linux before installing.

Quiz

A response file is:

a. A binary file that can be edited using a binary editing program
b. A binary file that can be created by the installer program
c. A text file that cannot be edited, but can be created by the installer program
d. A text file that can be edited with a text editor

Answer: d

Summary

In this lesson, you should have learned how to:

- Plan an Oracle Database and Oracle Grid Infrastructure for standalone server software installation
- Use the Oracle RDBMS Pre-Install RPM
- Configure the Oracle software owner environment
- Invoke the appropriate Oracle Universal Installer (OUI)

3

Installing Oracle Grid Infrastructure for a Standalone Server

Objectives

After completing this lesson, you should be able to:

- Install Oracle Grid Infrastructure for a standalone server
- Upgrade an existing Oracle Grid Infrastructure installation
- Use ASMCA to upgrade an existing Oracle Automatic Storage Management (ASM) instance

Overview of Oracle Grid Infrastructure for a Standalone Server

- Provides the infrastructure to include your single-instance database in an enterprise grid architecture
- Oracle Grid Infrastructure for a standalone server includes:
  - Oracle Restart
  - Oracle Cluster Synchronization Services (CSS)
  - Oracle Automatic Storage Management (ASM)
- Infrastructure products are combined into a single set of binaries that is installed into an Oracle Grid Infrastructure (Oracle Restart) home.
- Oracle Grid Infrastructure for a standalone server should be installed before you install and create a database.

ORACLE

Oracle Grid Infrastructure for a standalone server provides the infrastructure to include your single-instance database in an enterprise grid architecture. The Oracle Grid Infrastructure home includes Oracle Restart and Oracle Automatic Storage Management (Oracle ASM) software.

When you install Oracle Grid Infrastructure for a standalone server, the Oracle Universal Installer (OUI) configures a single-node version of Oracle Cluster Synchronization Services (CSS). CSS is a daemon process that enables synchronization between an Oracle ASM instance and the database instances that rely on it for database file storage. The CSS daemon is installed in and runs from the Oracle Grid Infrastructure home.

You should install Oracle Grid Infrastructure for a standalone server before you install and create the Oracle database. Otherwise, you will need to manually register the database with Oracle Restart.

Oracle Restart

Oracle Restart implements a high availability solution for stand-alone Oracle databases.

- Can monitor and restart the following components:
  - Database instances
  - Oracle Net listener
  - Database services
  - Automatic Storage Management (ASM) instance
  - ASM disk groups
  - Oracle Notification Services (ONS/eONS)
- Runs periodic check operations to monitor the health of the components

Oracle Restart is designed to improve the availability of your Oracle Database. It implements a high availability solution for single instance (nonclustered) environments only. For Oracle Real Application Cluster (Oracle RAC) environments, the functionality to automatically restart components is provided by Oracle Clusterware. Oracle Restart can monitor the health and automatically restart the following components:

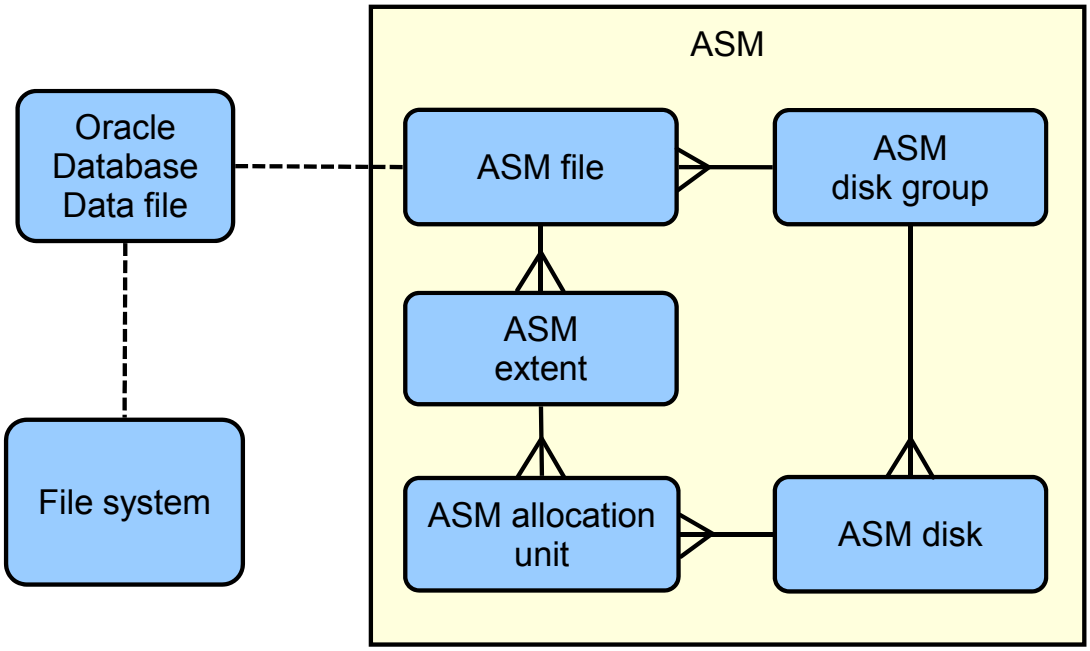
- Database instances
- Oracle Net listener
- Database services
- ASM instance
- ASM disk groups
- Oracle Notification Services (ONS/eONS): Service for sending Fast Application Notification (FAN) events to integrated clients upon failover. The eONS is used by Oracle Enterprise Manager to receive notification of change in status of components managed by Oracle Restart.

Restarting an ASM disk group means mounting it. The ability to restart ONS is applicable only in Oracle Data Guard installations for automatic failover of connections between primary and standby databases through FAN.

Oracle Restart ensures that the components are started in the proper order, in accordance with component dependencies. If a component must be shut down, it ensures that the dependent components are cleanly shut down first.

Note: Oracle Restart runs out of the Oracle Grid Infrastructure home and is dependent on Oracle Cluster Synchronization Services (CSS).

ASM Storage Components



The diagram illustrates the relationships between an Oracle Database data file and the ASM storage components. The crow's foot notation represents a one-to-many relationship. An Oracle Database data file has a one-to-one relationship with either a file stored on the operating system in a file system or an ASM file.

An Oracle ASM disk group is a collection of one or more Oracle ASM disks managed as a logical unit. The data structures in a disk group are self-contained using some of the space for metadata needs. Oracle ASM disks are the storage devices provisioned to an Oracle ASM disk group and can be physical disk or partitions, a Logical Unit Number (LUN) from a storage array, a logical volume (LV), or a network-attached file. Each ASM disk is divided into many ASM allocation units, the smallest contiguous amount of disk space that ASM allocates. When you create an ASM disk group, you can set the ASM allocation unit size to 1, 2, 4, 8, 16, 32, or 64 MB depending on the disk group compatibility level. One or more ASM allocation units form an ASM extent. An Oracle ASM extent is the raw storage used to hold the contents of an Oracle ASM file. An Oracle ASM file consists of one or more file extents. Variable extent sizes of 1*AU size, 4*AU size, and 16*AU size are used for supporting very large ASM files.

Configuring Storage for Oracle Automatic Storage Management (ASM)

To configure storage for use by Oracle ASM:

- Determine the number of devices and the amount of free disk space that you require
- Create DAS or SAN disk partitions for Oracle ASM
- Configure storage device path persistence by some means:
  - Oracle ASMLIB, only on some Linux operating systems
  - `udev` (device file naming scheme) on all Linux operating systems
  - OS-specific device management

ORACLE

To identify the storage requirements for using Oracle ASM, you must determine the number of devices and the amount of free disk space that you require. You need to determine whether you want to use Oracle ASM for Oracle Database files (data files, control files, and redo log files), recovery files, or both. You need to choose the Oracle ASM redundancy level to use for each Oracle ASM disk group that you create. The redundancy level determines how Oracle ASM mirrors files in the disk group and determines the number of disks and amount of disk space that is required.

In order to use a DAS or SAN disk in Oracle ASM, the disk must have a partition table. Oracle recommends creating exactly one partition for each disk.

Oracle recommends that you use Oracle Automatic Storage Management library driver (Oracle ASMLIB) for device persistence, where it is available. Oracle ASMLIB simplifies the configuration and management of block disk devices by eliminating the need to rebind block disk devices used with Oracle ASM each time the system is restarted. With Oracle ASMLIB, you define the range of disks you want to have made available as Oracle ASM disks. Oracle ASMLIB maintains permissions and disk labels that are persistent on the storage device, so that the label is available even after an operating system upgrade.

Refer to the *Oracle Database Installation Guide* for detailed information about these configuration tasks.

Oracle Grid Infrastructure Installation: System Requirements

- Memory requirements for Linux:
  - 4 GB for the ASM instance and Oracle Restart
  - Swap space:
    - 4 GB – 16 GB RAM, swap space = RAM
    - 16 GB + RAM, swap space = 16 GB
- Disk space requirements for Linux:
  - 6.9 GB disk space
  - 1 GB of disk space in the `/tmp` directory

A standard Oracle Grid Infrastructure for standalone server installation requires a minimum of 4 GB RAM.

The amount of swap space required is dependent on the amount of RAM:

- Between 4 GB and 16 GB: Swap space should be equal to the size of the RAM
- More than 16 GB: 16 GB

At least 6.9 GB of disk space is required for the installation and at least 1 GB of space in the `/tmp` directory is required.

Creating Operating System Groups and Users

Create custom configuration groups and users based on job role separation:

- Groups:
  - Oracle Inventory group (`oinstall`)
  - Oracle Grid Infrastructure groups for job role separation:
    - OSDBA (`asmdba`)
    - OSASM (`asmadmin`)
    - OSOPER (`asmoper`)
- Users (software owners):
  - Oracle Grid Infrastructure/Oracle Restart: `grid`

When you install Oracle software on a system for the first time, the Oracle Universal Installer (OUI) creates a file named `oraInst.loc`. This file identifies the Oracle Inventory group (typically, `oinstall`) and the path of the Oracle Inventory directory.

If you have an existing central Oracle Inventory, you should use that Oracle Inventory for all Oracle software installations. Ensure that all Oracle software users that will perform installations have permissions to write to this directory.

Create the following operating system groups:

- OSDBA: Administrative access to Oracle ASM instances
- OSASM: Granted `SYSASM` privileges to administer Oracle ASM
- OSOPER: Granted `SYSOPER` privileges (limited administrative privileges)

Oracle recommends that you create separate software owners for Oracle Database and Oracle Grid Infrastructure for a standalone server. Typically, the owner for Oracle Grid Infrastructure is named `grid`. The `grid` user must be a member of the Oracle Inventory group (`oinstall`), the OSDBA (`asmdba`) group, and the OSDBA group for each database.

Types of Installations

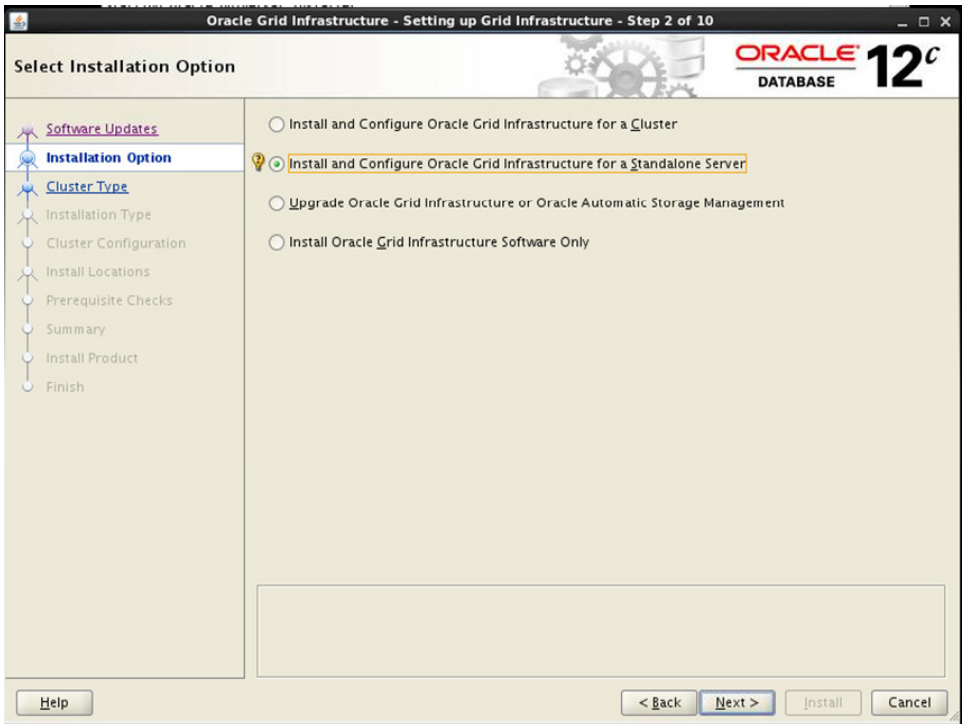
- Software-only installation
  - Copies the binaries to the specified location
  - Requires manual configuration
- Installation Oracle Grid Infrastructure, followed by an installation of Oracle Database
  - Database is added to the Oracle Restart configuration and is automatically restarted when required
- Installation of Oracle Grid Infrastructure following the installation of Oracle Database
  - Requires you to manually add the database, the listener, the Oracle ASM instance, and other components to the Oracle Restart configuration
- Upgrade of Oracle Grid Infrastructure and Oracle ASM

You can perform the following types of installations:

- Software-only installation to install the software binaries for Oracle Restart and Automatic Storage Management. Manual configuration is required to enable the software after this type of installation.
- Installation of Oracle Grid Infrastructure for a standalone server with a new database installation. With this type of installation, the database is automatically added to the Oracle Restart configuration and is automatically restarted when required. Perform the following steps:
  1. Install Oracle Grid Infrastructure for a standalone server to install Oracle Restart and Oracle ASM.
  2. Configure Oracle ASM with at least one disk group.
  3. Install Oracle Database and use ASM disk groups for the database files.
- Installation of Oracle Grid Infrastructure for a standalone server on a host with an existing database. Perform the following steps:
  1. Select “Configure Oracle Grid Infrastructure for a Standalone Server” as the installation option when you install Oracle Grid Infrastructure.
  2. Manually add the database, the listener, the Oracle ASM instance, all Oracle ASM disk groups, and any database services to the Oracle Restart configuration.

- Upgrade of Oracle Grid Infrastructure and Oracle Automatic Storage Management (ASM): If you have an earlier release of Oracle ASM on your system, the OUI automatically defaults to upgrade mode.

Installing the Oracle Grid Infrastructure for Standalone Server



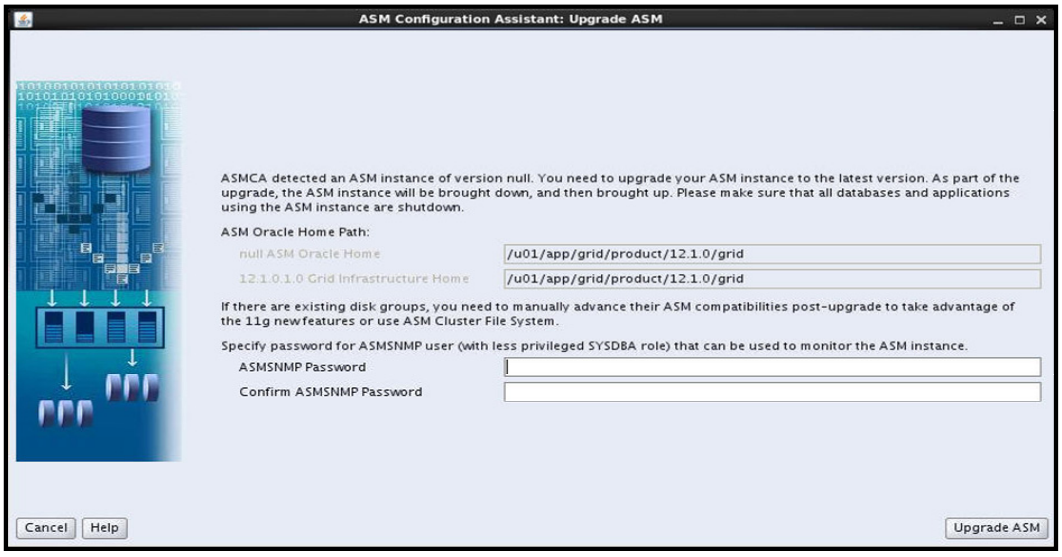
To install the Oracle Grid Infrastructure software by using the Oracle Universal Installer (OUI), log on to your computer as a member of the administrative group that is authorized to install the Oracle software and to create and manage the database. Enter `./runInstaller` from the relevant directory on the Oracle Database 12*c* installation media or download directory to start the OUI.

On the Download Software Updates page you can supply My Oracle Support credentials so that the OUI can download updates for the installation. You can also specify that the OUI use software updates that you have previously downloaded or skip the updates completely.

The next page that is displayed is the Select Installation Option page, as shown in the screenshot. Select the option appropriate to your situation and click Next to proceed with your installation.

Upgrading Existing Oracle ASM Instances

Use ASMCA to upgrade an existing ASM installation



If you have an Oracle ASM installation from an earlier release installed on your server, or in an existing Oracle Restart installation, you can use Oracle Automatic Storage Management Configuration Assistant (Oracle ASMCA) to upgrade the existing Oracle ASM instance to 12*c*, and subsequently configure disk groups, Oracle ASM volumes and Oracle ASM file systems.

After the installation of the Oracle Grid Infrastructure Software only, Oracle ASMCA detects that there is a prior Oracle ASM version installed in another Oracle ASM home. After installing the Oracle ASM 12*c* binaries, you can start Oracle ASMCA to upgrade the existing Oracle ASM instance.

Note: An Oracle ASM instance should be upgraded by using OUI. OUI automatically defaults to upgrade mode when it detects an Oracle ASM instance at a previous release level.

Summary

In this lesson, you should have learned how to:

- Install Oracle Grid Infrastructure for a Standalone Server
- Upgrade an existing Oracle Grid Infrastructure installation
- Use ASMCA to upgrade an existing Oracle Automatic Storage Management (ASM) instance

Practice 3: Overview

This practice covers the following topics:

- Upgrading and configuring Oracle Grid Infrastructure and ASM
- Using the upgraded ASM instance
- Determining existing listeners

4

Installing Oracle Database Software

Objectives

After completing this lesson, you should be able to:

- Describe operating system groups and users that are required for Oracle Database software installation
- Install Oracle Database software

Oracle Database Installation: System Requirements

- Memory requirements for Linux:
  - Minimum 1 GB (2 GB recommended) for the database instance
  - Swap space:
    - 1 GB – 2 GB RAM, swap space = 1.5 times the size of RAM
    - 2 GB – 16 GB RMAN, swap space = size of RAM
    - 16 GB + RAM, swap space = 16 GB
- Disk space requirements for Linux:
  - 6.4 GB for the Oracle Database software (Enterprise Edition) and data files
  - 1 GB of disk space in the `/tmp` directory
  - 4 GB (default) for the fast recovery area (optional)

ORACLE

An Oracle Database installation can be completed on a computer with 1 GB of RAM and 1.5 GB of swap space or larger.

The amount of swap space required is dependent on the amount of RAM (for example, for 2 GB RAM you need 3 GB of swap space).

Storage requirements for Oracle Database Enterprise Edition software files and data files are listed in the slide. Standard Edition requires 6.1 GB of disk space.

Refer to operating system–specific installation guides for system requirements for other operating systems.

Creating Operating System Groups and Users

Create custom configuration groups and users based on job role separation:

- Groups:
  - Oracle Inventory group (`oinstall`)
  - Standard Oracle Database groups:
    - OSDBA (`dba`)
    - OSOPER (`oper`)
  - Extended Oracle Database groups:
    - OSBACKUPDBA (`backupdba`)
    - OSDGDBA (`dgdba`)
    - OSKMDBA (`kmdba`)
- Users (software owners):
  - Oracle Database: `oracle`

Before you install the Oracle Database software you need to create operating system groups and users as described in the slide. If you installed Oracle Grid Infrastructure for a standalone server you may have already created some of the required groups and users.

If you used the Oracle Pre-Install RPM to provision your Linux operating system for an Oracle Grid Infrastructure and Oracle Database installation, it configured the Oracle database installation owner (`oracle`), an Oracle Inventory group (`oinstall`), and an Oracle administrative privileges group (`dba`).

Create the following operating system groups:

- OSDBA (`dba`): Database administrative privileges (`SYSDBA` privilege)
- OSOPER (`oper`): Limited set of database administrative privileges (`SYSOPER` privilege)

Optionally, create the following operating system groups that are more task-specific and less privileged than OSDBA:

- OSBACKUPDBA (`backupdba`): Limited set of database backup and recovery related administrative privileges (`SYSBACKUP` privilege)
- OSDGDBA (`dgdba`): Limited set of privileges to administer and monitor Oracle Data Guard (`SYSDG` privilege)

- OSKMDBA (`kmdba`): Limited set of privileges for encryption key management (`SYSKM` privilege)

Be sure to add the Oracle software installation to the above groups.

Oracle recommends that you create separate software owners for Oracle Database and Oracle Grid Infrastructure for a standalone server. Typically, the owner for Oracle Database is named `oracle`.

Types of Installations

- Software-only installation
  - Copies the binaries to the specified location
  - Use Database Configuration Assistant (DBCA) to create the database
- Installation of the software and creation of a database
- Upgrade an existing database

You can perform the following types of installations:

- **Install database software only:** This option only installs the database binaries. You must use the Oracle Database Configuration Assistant (DBCA) after the software installation to create the database. Additional information about DBCA is presented in the next lesson.
- **Create and configure a database:** This option installs the software and creates a new database.
- **Upgrade an existing database:** This option installs the software binaries in a new Oracle home. After the installation, you can upgrade the existing database. Additional information about upgrading a database is provided in later lessons.

Quiz

During Oracle Database software installation you can specify groups for:

a. the `osoper` group
b. the `osasm` group
c. the `osdba` group
d. the `osadmin` group

ORACLE

Answer: a, c

Summary

In this lesson, you should have learned how to:

- Describe operating system groups and users that are required for Oracle Database software installation
- Install Oracle Database software

Practice 4: Overview

This practice covers installing the Oracle Database software by using Oracle Universal Installer.

5

Creating an Oracle Database by Using DBCA

Objectives

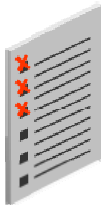
After completing this lesson, you should be able to do the following:

- Create a database by using the Database Configuration Assistant (DBCA)
- Generate database creation scripts by using DBCA
- Manage database design templates by using DBCA
- Configure database options by using DBCA

Planning the Database

As a DBA, you must plan:

- The logical storage structure of the database and its physical implementation:
  - How many disk drives do you have? What type of storage is being used?
  - How many data files will you need? (Plan for growth.)
  - How many tablespaces will you use?
  - What types of information will be stored?
  - Are there any special storage requirements due to type or size?
- Overall database design
- Database backup strategy



It is important to plan how the logical storage structure of the database will affect system performance and various database management operations. For example, before creating any tablespaces for your database, you should know how many data files will make up the tablespace, what type of information will be stored in each tablespace, and on which disk drives the data files will be physically stored. Information such as the availability of network attached storage (NAS) and the bandwidth for the private storage network are important. If storage area networks (SAN) are going to be used, knowing how the logical volumes are configured and the stripe size is useful.

When planning the overall logical storage of the database structure, take into account the effects that this structure will have when the database is actually created and running. You may have database objects that have special storage requirements due to type or size.

In distributed database environments, this planning stage is extremely important. The physical location of frequently accessed data dramatically affects application performance.

During the planning stage, develop a backup strategy for the database. You can alter the logical storage structure or design of the database to improve backup efficiency. Backup strategies are introduced in a later lesson.

Types of Databases

- General purpose or transaction processing:
  - Online transaction processing (OLTP) system, for example a retail billing system for a software house or a nursery
- Custom:
  - Multipurpose database (perhaps combined OLTP and data warehouse functionality)
- Data warehouse:
  - Research and marketing data
  - State or federal tax payments
  - Professional licensing (doctors, nurses, and so on)

Different types of databases have their own specific instance and storage requirements. Your Oracle Database software includes templates for the creation of these different types of databases. Characteristics of these examples are the following:

- **General purpose:** For general purpose or transaction processing usage such as working with transactions and storing them for a medium length of time
- **Custom:** For customized databases that do not fit into the general purpose or data warehouse template
- **Data warehouse:** For storing data for long periods and retrieving them in read operations

Choosing the Appropriate Character Set

- Oracle Database supports different classes of character-encoding schemes:
  - Single-byte character sets
    - 7-bit
    - 8-bit
  - Multibyte character sets, including Unicode
- The character set is chosen at the time of database creation. Choose the character set that best meets your business requirements now and in the future because it can be difficult to change character sets later on.
- In general Unicode is recommended because it is the most flexible character set.

ORACLE

When computer systems process characters, they use numeric codes instead of the graphical representation of the character. An *encoded character set* maps numeric codes to characters that a computer or terminal can display and receive. Different character sets support different character repertoires. Because character sets are typically based on a particular writing script, they can support more than one language. However, script-based character sets are restricted in the sense that they are limited to groups of languages based on similar scripts. Universal character sets encompass most major scripts of the modern world and provide a more useful solution to multilingual support. For information about the Unicode standards, see the website at http:www.unicode.org.

Oracle Database supports three classes of encoding schemes: Single-byte, Varying-width multibyte, and Universal. Choose the correct character set that best meets your business requirements now and in the future because it can be difficult to change character sets later on. For best performance, choose a character set that avoids character set conversion and uses the most efficient encoding for the languages desired. Single-byte character sets result in better performance than multibyte character sets, and they also are the most efficient in terms of space requirements. However, single-byte character sets limit how many languages you can support. To choose your correct database character set, evaluate your current and future business requirements, as well as technical requirements (for example, the XML and

Java standards require Unicode). In general, Oracle recommends the use of Unicode for all new databases, because it is the most flexible character set and avoids future conversions.

Single-Byte Character Sets

In a single-byte character set, each character occupies one byte. Single-byte 7-bit encoding schemes can define up to 128 (2^7) characters; single-byte 8-bit encoding schemes can define up to 256 (2^8) characters.

Examples of Single-Byte Schemes

7-bit character set:

- American Standard Code for Information Interchange (ASCII) 7-bit American (`US7ASCII`)

8-bit character set:

- International Organization for Standards (ISO) 8859-1 West European (`WE8`*`i`*`SO8859P1`)
- DEC 8-bit West European (`WE8DEC`)
- Extended Binary Coded Decimal Interchange Code (EBCDIC) Code Page 1144 8-bit Italian (`I8EBCDIC1144`)

Multibyte Character Sets

A varying-width multibyte character set is represented by one or more bytes per character. Multibyte character sets are commonly used for Asian language support. Some multibyte encoding schemes use the value of the most significant bit to indicate whether a byte represents a single byte or is part of a series of bytes representing a character. However, other character-encoding schemes differentiate single-byte from multibyte characters. A shift-out control code, sent by a device, indicates that any successive bytes are double-byte characters until a shift-in code is encountered. Shift-sensitive encoding schemes are used primarily on IBM platforms.

Unicode is a universal encoded character set that enables information from any language to be stored using a single character set. Unicode provides a unique code value for every character, regardless of the platform, program, or language.

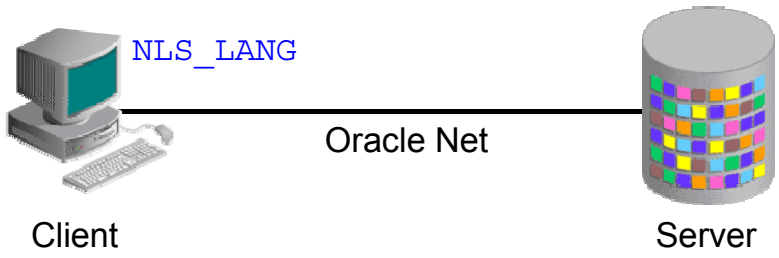
The Unicode standard has been adopted by many software and hardware vendors. Many operating systems and browsers now support Unicode. Unicode is required by standards such as XML, Java, JavaScript, LDAP, and WML. It is also synchronized with the ISO/IEC 10646 standard.

Examples of Varying-Width Multibyte Schemes

- Shift-JIS 16-bit Japanese (`JA16SJIS`)
- MS Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001 (`ZHT16HKSCS`)
- Unicode 4.0 UTF-8 Universal character set (`AL32UTF8`) - a variable-width type of encoding and also a strict superset of ASCII.

Understanding How Character Sets Are Used

- Oracle Net compares the client NLS_LANG setting to the character set on the server.
- If needed, conversion occurs automatically and transparently.



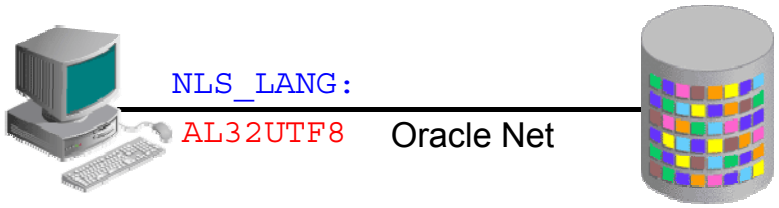
The NLS_LANG parameter defines a client terminal's character-encoding scheme. Different clients can use different encoding schemes. Data passed between the client and the server is converted automatically between the two encoding schemes. The database's encoding scheme should be a superset, or equivalent, of all the clients' encoding schemes. The conversion is transparent to the client application.

When the database character set and the client character set are the same, the database assumes that the data being sent or received is of the same character set, so no validations or conversions are performed.

Character set conversion may be required in a client/server environment, if a client application resides on a different platform than the server and if the platforms do not use the same character-encoding schemes. Character data passed between the client and the server must be converted between the two encoding schemes. Character conversion occurs automatically and transparently through Oracle Net.

Setting the NLS_LANG Initialization Parameter

Example:



Client
Windows English
Code page: WE8MSWIN1252

Server
Database character set:
AL32UTF8

No conversion occurs, because it does not seem to be required.

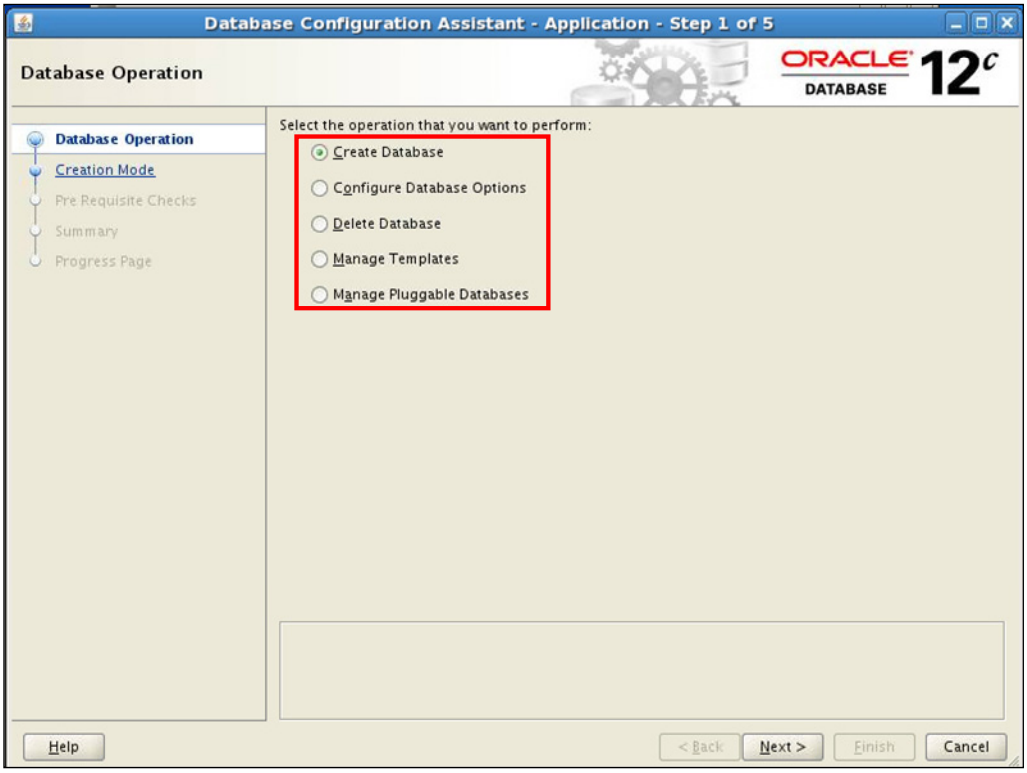
Issue: Invalid data are entered into the database.

Invalid data usually enters a database when the NLS_LANG parameter is not set properly on the client. The NLS_LANG value should reflect the encoding of the incoming data.

- When the NLS_LANG parameter is set properly, the database can automatically convert incoming data from the client operating system.
- When the NLS_LANG parameter is not set properly, the data entering the database is not converted properly.

For example, suppose that the database character set is AL32UTF8, the client is an English Windows operating system (code page: WE8MSWIN1252), and the NLS_LANG setting on the client is AL32UTF8. Data entering the database is encoded in WE8MSWIN1252 and is not converted to AL32UTF8 data because the NLS_LANG setting on the client matches the database character set. Thus the Oracle Database server assumes that no conversion is necessary, and invalid data is entered into the database.

Using the DBCA to Create a Database



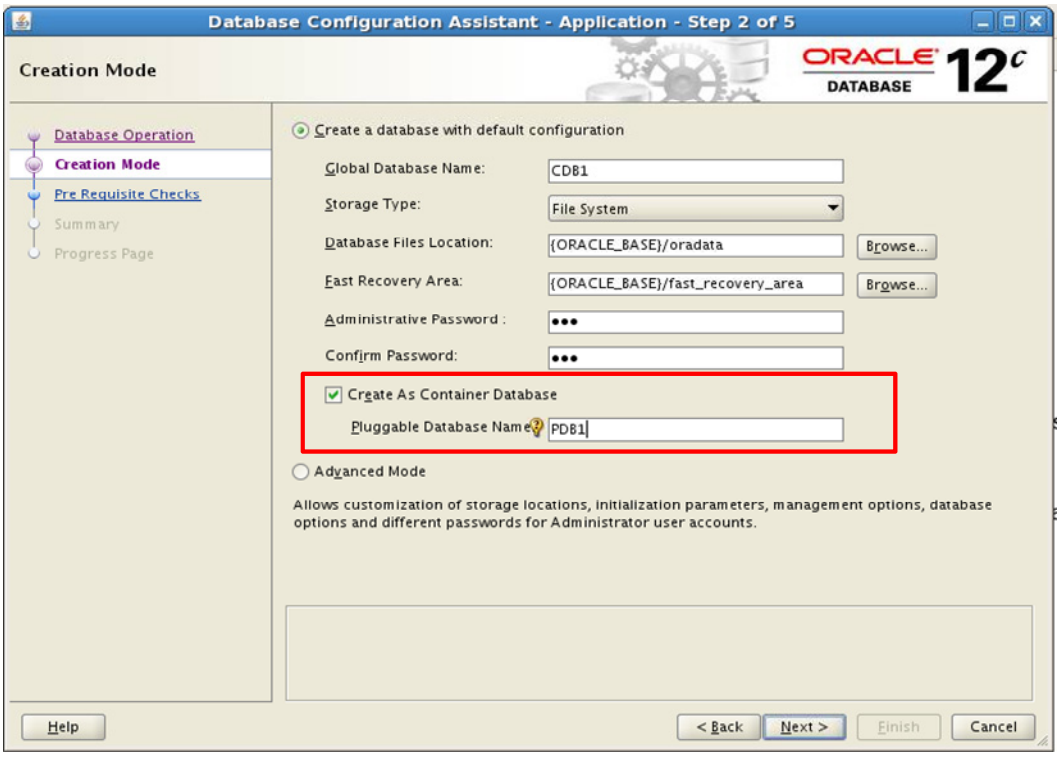
The Database Configuration Assistant (DBCA) offers several choices to assist you with various operations. You can use the DBCA to create, change the configuration of, or delete a database. You can also create a database from a list of predefined templates or use an existing database as a sample to create a new database or template. The “Manage Pluggable Databases” option enables you to create, configure, unplug, and delete PDBs.

The DBCA provides several options to allow you to create a database to meet your needs. The DBCA provides a series of pages where you enter configuration information. On most pages, the DBCA provides default settings that you can accept if they apply.

Invoke DBCA to create a database as follows:

1. Log on to your computer as a member of the OS DBA group that is authorized to install the Oracle software. If required, set environment variables and enter `dbca` to invoke the DBCA.
2. Select “Create Database” and click Next.

Creating a Container Database by Using DBCA

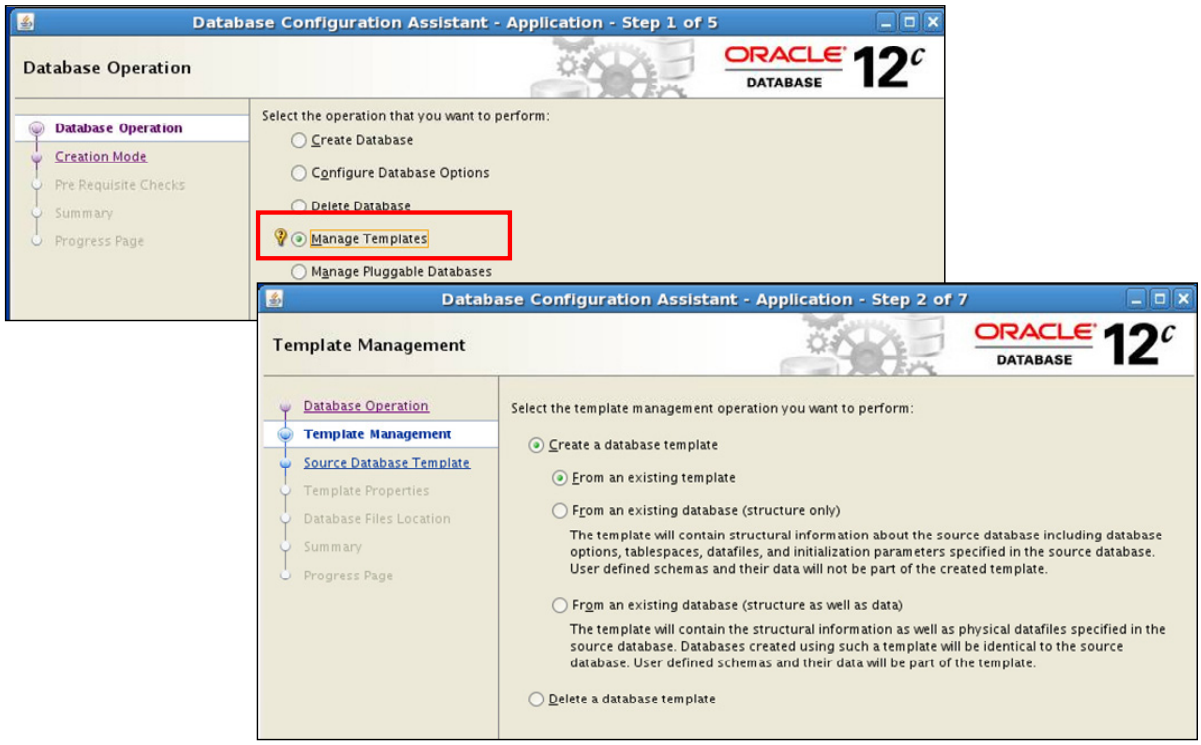


To create a container database by using DBCA, select “Create As Container Database.”

You also have to provide a pluggable database name when you select the “Create a Database with Default Configuration” check box. If you select “Advanced Mode,” you can create an empty multitenant container database with only the root and seed containers.

In Advanced Mode, you can register the CDB with Enterprise Manager Cloud Control and set passwords for the `SYS` and `SYSTEM` users.

Creating a Database Design Template



A template is a predefined database definition that you use as a starting point for a new database. If you do not create a template as part of the database creation process, you can do it at any time by invoking the DBCA and choosing the Manage Templates operation.

There are three ways to create a template:

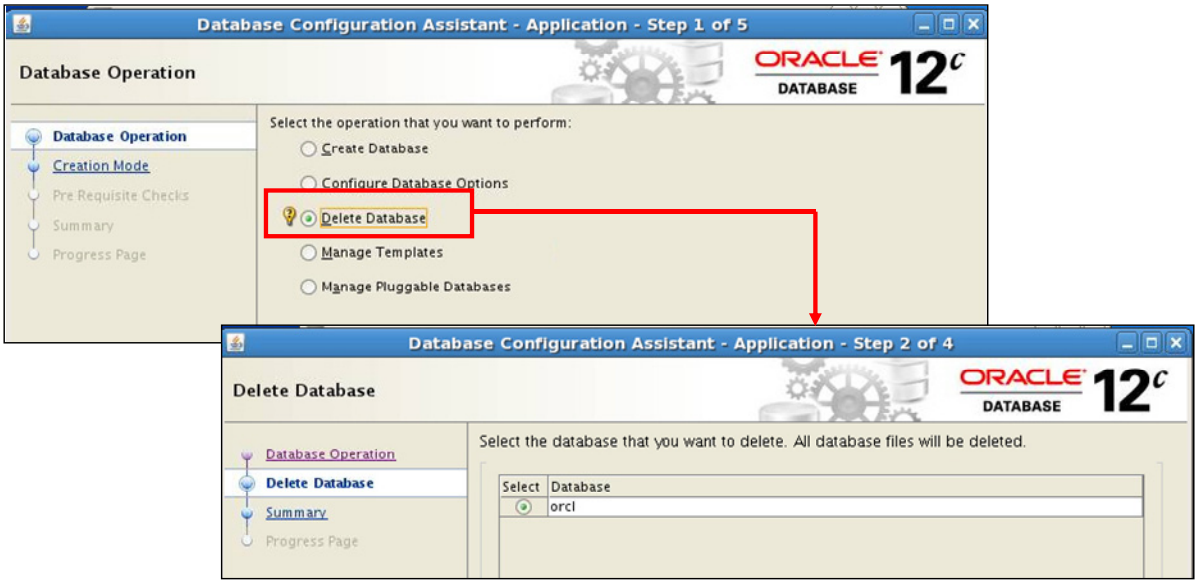
- From an existing template
- From an existing database (structure only)
- From an existing database (structure as well as data)

The DBCA guides you through the steps to create a database design template.

If you no longer need a specific template use the “Delete a database template” option on the Template Management page of the DBCA.

Note: Templates you created will appear in the Database Templates list when you create a new database using the DBCA.

Using the DBCA to Delete a Database



Start the DBCA by entering `dbca` in a terminal window. To delete a database, perform the following steps:

1. On the “Database Operation” page, select Delete a Database. Then click Next.
2. Select the database that you want to delete and click Next.
3. Click Finish on the Summary page.
4. Click Yes to confirm your deletion.
5. When the deletion is completed, you will be asked whether you want to perform another operation. Answer accordingly.

Note: The database being deleted must be opened so that DBCA can connect to the database to determine file location information.

Dropping a database involves removing its data files, redo log files, control files, and initialization parameter files. You can drop a database manually using the `DROP DATABASE` SQL statement. The `DROP DATABASE` statement deletes all control files and all other database files listed in the control file. To use the `DROP DATABASE` statement successfully, all of the following conditions must apply:

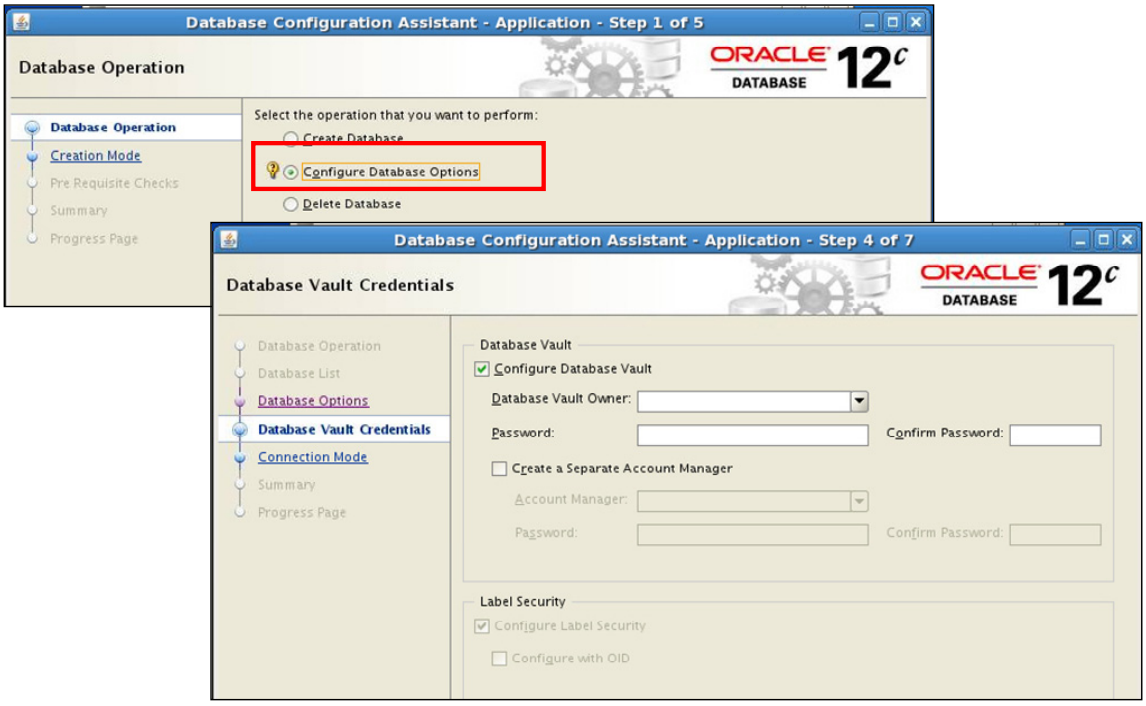
- The database must be mounted and closed.
- The database must be mounted exclusively (not in shared mode).
- The database must have been started in `RESTRICT` mode.

An example of these statements is:

```
STARTUP RESTRICT FORCE MOUNT;
DROP DATABASE;
```

The `DROP DATABASE` statement has no effect on archived log files, nor does it have any effect on copies or backups of the database. It is best to use Recovery Manager (RMAN) to delete such files. If the database is on raw disks, the actual raw disk special files are not deleted.

Using the DBCA for Additional Tasks



You can use the DBCA to configure database options such as Oracle Database Vault and Oracle Label Security. Not all options are installed by default during database software installation, so it may be necessary to install additional software prior to attempting to configure a database to use certain options.

Note

- For more information about Oracle Database Vault, see the *Oracle Database Vault Administrator's Guide*.
- For more information about Oracle Label Security, see the *Oracle Label Security Administrator's Guide*.

Summary

In this lesson, you should have learned how to:

- Create a database by using the Database Configuration Assistant (DBCA)
- Generate database creation scripts by using DBCA
- Manage database design templates by using DBCA
- Configure database options by using DBCA

Practice 5: Overview

This practice covers the following topics:

- Creating the ORCL database by using the DBCA
- Unlocking the HR schema
- Creating the ORCL database design template by using the DBCA
- Creating database creation scripts by using the DBCA
- Creating a CDB and PDB

6

Oracle Restart

Objectives

After completing this lesson, you should be able to:

- Start and stop Oracle Restart
- Use Oracle Restart to manage components

Oracle Restart

Oracle Restart implements a high availability solution for stand-alone Oracle databases.

- Can monitor and restart the following components:
  - Database instances
  - Oracle Net listener
  - Database services
  - Automatic Storage Management (ASM) instance
  - ASM disk groups
  - Oracle Notification Services (ONS/eONS)
- Runs periodic check operations to monitor the health of the components
- Runs out of the Oracle Grid Infrastructure home, which you install separately from Oracle Database homes

ORACLE

Oracle Restart is designed to improve the availability of your Oracle database. It implements a high availability solution for single instance (nonclustered) environments only. For Oracle Real Application Cluster (Oracle RAC) environments, the functionality to automatically restart components is provided by Oracle Clusterware. Oracle Restart can monitor the health and automatically restart the following components:

- Database instances
- Oracle Net listener
- Database services
- ASM instance
- ASM disk groups
- Oracle Notification Services (ONS/eONS): Service for sending Fast Application Notification (FAN) events to integrated clients upon failover. The eONS is used by Oracle Enterprise Manager to receive notification of change in status of components managed by Oracle Restart.

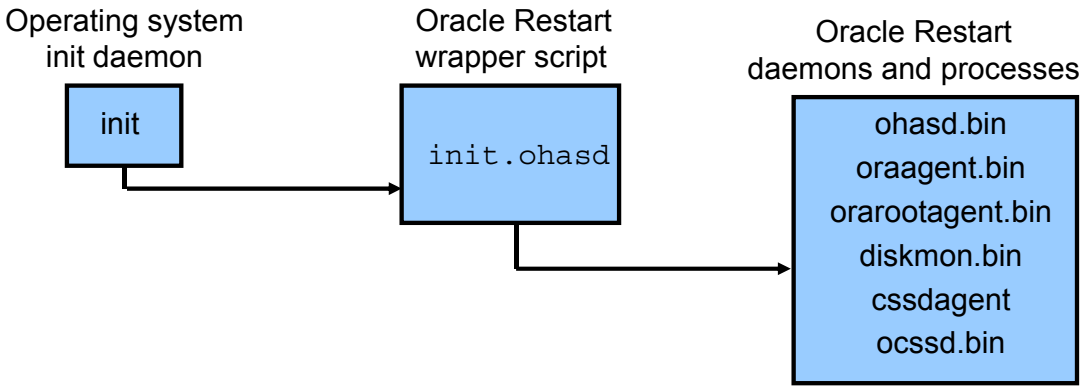
Restarting an ASM disk group means mounting it. The ability to restart ONS is applicable only in Oracle Data Guard installations for automatic failover of connections between primary and standby databases through FAN.

Oracle Restart ensures that the components are started in the proper order, in accordance with component dependencies. If a component must be shut down, it ensures that the dependent components are cleanly shut down first.

Oracle Restart operates inside Oracle Grid Infrastructure home.

Oracle Restart Process Startup

- Oracle Restart is started by the OS init daemon.



- During installation Oracle Restart is registered with the operating system so that it is started whenever the system starts up.

ORACLE

The wrapper script sets up the environment variables and then starts the Oracle Restart daemons and processes.

When a command is used to stop Oracle Restart, the daemons will be stopped, but the wrapper script process will remain running.

The wrapper script is started with the respawn action so it will be restarted whenever it is terminated. In addition, the respawn action causes the init process to restart the daemons if they fail.

Some of the Oracle Restart daemons will be running under the `root` user with real-time priority, and others will be running under the Grid Infrastructure owner with user-mode priorities after they are started. On a Windows platform, operating system services are used instead of wrapper initialization scripts and the daemons are executable binaries.

Note: Executing the wrapper script directly is not supported.

Controlling Oracle Restart

The Oracle Clusterware Control (CRSCTL) utility can be used to control the state of Oracle Restart.

- To display the Oracle Restart configuration:

```
$ crsctl config has
```

- To enable or disable the automatic restart of Oracle Restart:

```
$ crsctl [ enable | disable ] has
```

- To start or stop Oracle Restart:

```
$ crsctl [ start | stop ] has
```


The CRSCTL utility can be used to control the state of Oracle Restart. It can be used to determine if the autostart capability is enabled or disabled as follows:

```
$ crsctl config has
CRS-4622: Oracle High Availability Services autostart is enabled.
```

The control files for `ohas` are used to control the state of `ohas` and determine if automatic restart is enabled or disabled. These are known as `SCLS_SCR` files. For Linux, the location of the control files is defined to be:

```
/etc/oracle/scls_scr/$HOST/<Oracle Restart owner> and
/etc/oracle/scls_scr/$HOST/root
```

If the CRSCTL utility is used to stop Oracle Restart, all components currently managed by Oracle Restart will also be stopped.

```
$ crsctl stop has
CRS-4549: Stopping resources.
CRS-2673: Attempting to stop 'ora.diskmon' on 'host01'
CRS-2673: Attempting to stop 'ora.DATA.dg' on 'host01'
CRS-2673: Attempting to stop 'ora.LISTENER.lsnr' on 'host01'
CRS-2677: Stop of 'ora.DATA.dg' on 'host01' succeeded
CRS-2673: Attempting to stop 'ora.asm' on 'host01'
CRS-2675: Stop of 'ora.diskmon' on 'host01' succeeded
CRS-2677: Stop of 'ora.LISTENER.lsnr' on 'host01' succeeded
CRS-2677: Stop of 'ora.asm' on 'host01' succeeded
CRS-2673: Attempting to stop 'ora.cssd' on 'host01'
CRS-2677: Stop of 'ora.cssd' on 'host01' succeeded
CRS-2673: Attempting to stop 'ora.diskmon' on 'host01'
CRS-2677: Stop of 'ora.diskmon' on 'host01' succeeded
CRS-4133: Oracle High Availability Services has been stopped.
```

In the example above, there is no Oracle Database software installed. If Oracle databases had been registered with Oracle Restart or they had been using ASM, they would have been stopped also.

When starting Oracle Restart with the CRSCTL utility, each component that is started is not displayed to standard output.

```
$ crsctl start has
CRS-4123: Oracle High Availability Services has been started.
```

Note: Invoking the wrapper script directly to start the Oracle Grid Infrastructure processes is not supported.

Choosing the Correct SRVCTL Utility

- Invoke the Server Control (SRVCTL) utility from the Oracle Grid Infrastructure home when working with ASM instances, disk groups, listeners, and ONS.

```
$ export ORACLE_HOME=/u01/app/grid/product/12.1.0/grid
$ $ORACLE_HOME/bin/srvctl command component options
```

- Invoke the SRVCTL utility from the Oracle Database home when working with the database or database instance.

```
$ export
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
$ $ORACLE_HOME/bin/srvctl command component options
```


Oracle Restart includes the SRVCTL utility that you use to start, stop, and manage Oracle Restart components. After the Oracle Database software is installed in addition to the Oracle Grid Infrastructure software, there will be a copy of the SRVCTL utility in each `ORACLE_HOME` location. You need to determine the correct `ORACLE_HOME` location in which to run the SRVCTL utility. You need to run the SRVCTL utility from the Grid Infrastructure software home directory when managing the ASM instance, ASM disk groups, Oracle Net listeners, and the ONS. You need to run the SRVCTL utility from the Oracle Database software home directory when managing the Oracle database instances. To determine the currently mapped location of the SRVCTL utility, use the `which` command as follows:

```
$ which srvctl
/u01/app/oracle/product/12.1.0/dbhome_1/bin/srvctl
```

Note: For the Oracle Net listener, the assumption is that Oracle Grid Infrastructure was installed before the Oracle Database software. If Oracle Restart is added to an existing Oracle Database installation, the Oracle Net listener could be running from the Oracle Database home directory. In that case, you should use the SRVCTL utility from the Oracle Database home to manage the Oracle Net listener.

Refer to the *Oracle Database Administrator's Guide* for additional information about SRVCTL.

Oracle Restart Configuration

Oracle utilities will automatically update the Oracle Restart configuration.

Create operations and the Oracle Restart configuration	Automatically added to configuration?
Create a database by using OUI or DBCA	YES
Create a database by using a SQL statement	NO
Create an ASM instance by using OUI, DBCA, or ASMCA	YES
Create a disk group (any method)	YES
Add a listener by using NETCA	YES
Create a database service by using SRVCTL	YES
Create a database service by modifying `SERVICE_NAMES` initialization parameter	NO
Create a database service by using `DBMS_SERVICE.CREATE_SERVICE`	NO

ORACLE

Oracle Restart maintains a list of all the components that it manages, and maintains configuration information for each component. All of this information is collectively known as the "Oracle Restart configuration." When Oracle Restart is installed, many operations that create Oracle components using Oracle utilities will automatically add the components to the Oracle Restart configuration. If a component is created manually without using an Oracle utility, then SRVCTL commands can be used to add it to the Oracle Restart configuration if desired. The table in the slide shows which create operations automatically add the component to the Oracle Restart configuration and which create operations do not update the Oracle Restart configuration.

Note: The same principle applies to delete operations.

Using the SRVCTL Utility

- The SRVCTL utility is used to start, stop, and manage Oracle Restart components with the following syntax:

```
$ srvctl command component options
```

- The following command and components are supported:

Commands	`add config disable downgrade enable getenv modify remove setenv start status stop unsetenv upgrade`
Components	`asm db dg filesystem home lsnr serv ons`

ORACLE

When Oracle Restart is in use, it is strongly recommended that you use the SRVCTL utility to start, stop, and manage all Oracle Restart components. The SRVCTL utility is recommended for the following reasons:

- All dependencies between components are maintained. This enables Oracle Restart to start or to stop any dependent components first.
- Components are started according to their Oracle Restart configuration.
- Environment variables stored in the Oracle Restart configuration for the components are set.

Oracle Restart components can also be started with other utilities such as the listener control (LSNRCTL) utility or SQL*Plus, but the benefits listed above may not be obtained with other utilities.

The options that are allowed vary with each command and component combination. The SRVCTL utility syntax is as follows:

```
srvctl command component options
```

where:

- *command* is a verb such as `start`, `stop`, or `remove`
- *component* is the object on which SRVCTL performs the command, such as a database
- *options* extend the use of the preceding command to include additional parameters

Obtaining Help for the SRVCTL Utility

The SRVCTL utility provides online help for its commands, components, and options.

- For help with general usage:

```
$ srvctl -h
```

- For help on a particular command:

```
$ srvctl command -h
```

- For help on a particular command and component:

```
$ srvctl command component -h
```


The SRVCTL utility provides detailed online help for its commands, components, and options. To display the online Help, you use the help (`-h`) option to display usage information. If the help (`-h`) option is the only parameter specified, SRVCTL displays a general outline of all commands with the most common options used for each command and component combination. This will not be a complete list of all supported options. For more detailed and complete information, the help (`-h`) option can be used for a specific command, or for a specific command and component combination.

Starting Components by Using the SRVCTL Utility

Oracle recommends that the SRVCTL utility be used to start all components.

- Examples of starting individual components:

```
$ srvctl start database -d PROD -o mount
$ srvctl start listener -l crmlistener
$ srvctl start service -d PROD -s "service1,service2"
$ srvctl start diskgroup -g "DATA,FRA"
$ srvctl start asm
$ srvctl start eons -v
$ srvctl start ons
```

- Example of starting all Oracle Restart components in a specified Oracle home:

```
$ srvctl start home -o oracle_home -s state_file
```


The SRVCTL utility can be used to start individual components, along with any dependent components that are necessary. For example, the `srvctl start database -d PROD` command may also start the listener, the ASM instance, and multiple disk groups if those components have been defined as being managed by Oracle Restart and are listed as dependent components to the `PROD` database.

The SRVCTL utility can also be used to start all components that are associated with a specified Oracle home and have been configured for Oracle Restart with the following command:

```
srvctl start home -o /u01/app/oracle/product/12.1.0/dbhome_1 -s
/usr/local/bin/group_state_file
```

The state file contains the current state information for the components in the Oracle home and is created when the `srvctl status home` command is executed. It is indicated with the state file option (`-s`), and must specify the complete path of the state file. The state file can be created in any directory.

Note: The options shown in the slide represent the most common options and are not a complete list. You can use the help option (`-h`) for a complete list of all available options for each command.

Stopping Components by Using the SRVCTL Utility

Oracle recommends that the SRVCTL utility be used to stop all components.

- Examples of stopping individual components:

```
$ srvctl stop database -d PROD -o transactional
$ srvctl stop listener -l crmlistener -f
$ srvctl stop service -d PROD -s "service1,service2"
$ srvctl stop diskgroup -g "DATA,FRA" -f
$ srvctl stop asm -o immediate -f
$ srvctl stop eons -v
$ srvctl stop ons
```

- Example of stopping all Oracle Restart components in a specified Oracle home:

```
$ srvctl stop home -o oracle_home -s state_file -f
```

ORACLE

The SRVCTL utility can be used to stop individual components, along with any dependent components that must be stopped. For example, the `srvctl stop diskgroup -g "DATA -f"` command will force an unmount of the diskgroup even if files are open in it. It will also stop all database instances that depend on the `DATA` disk group.

The SRVCTL utility can also be used to stop all components that are associated with a specified Oracle home and have been configured for Oracle Restart with the following command:

```
srvctl stop home -o /u01/app/oracle/product/12.1.0/dbhome_1 -s
/usr/local/bin/group_state_file -f
```

This can be very useful when it is necessary to stop all components such as when you need to apply a patch to the software binaries.

Note: The options shown in the slide represent the most common options and are not a complete list. You can use the help option (`-h`) for a complete list of all available options for each command.

Viewing Component Status

- You can use the `status` command to view the running status for any component managed by Oracle Restart.
- Display the running status for a database:

```
$ srvctl status database -d orcl
Database is running.
```

- Display the listener status:

```
$ srvctl status lsnr
Listener LISTENER is enabled
Listener LISTENER is running on node(s): host01
```


You can use the SRVCTL utility to view the running status (running or not running) for any component managed by Oracle Restart. Additional information is displayed for some components.

The format of the command is as follows:

```
srvctl status object [options]
```

object can be one of the following:

- `asm`: ASM instance
- `database (db)`: Database instance
- `diskgroup (dg)`: Oracle ASM disk group
- `filesystem`: Oracle ASM file system
- `home`: Oracle home or Oracle Clusterware home
- `listener(lsnr)`: Oracle Net listener
- `ons`, `eons`: Oracle Notification Services
- `service (serv)`: Database service

Refer to the *Oracle Database Administrators Guide* for a list of options for each of the listed objects.

Displaying the Oracle Restart Configuration for a Component

- You can use the `config` command to display the Oracle Restart configuration of a component.
- Display the Oracle Restart configuration for a database:

```
$ srvctl config database -d orcl
Database unique name: orcl
Database name: orcl
Oracle home: /u01/app/oracle/product/12.1.0/dbhome_1
Oracle user: oracle
Spfile: +DATA/orcl/spfileorcl.ora
Domain: example.com
Start options: open
Stop options: immediate
Database role:
Management policy: automatic
Disk Groups: DATA,FRA
Services: east,sales
```


You can use the SRVCTL utility to display the Oracle Restart configuration of a component with the `config` command. The `config` command is valid for the `database`, `service`, `asm`, `listener`, `ons`, and `eons` components.

The configuration for an Oracle Restart component can be modified with the SRVCTL utility `modify` command. The following syntax shows an example of how to modify the database with the unique name of `orcl` to use a different, nonstandard directory for the server parameter file (SPFILE).

```
srvctl modify database -d orcl -p
/usr/local/oracle/spfileorcl.ora
```

Manually Adding Components to the Oracle Restart Configuration

Components can be manually added to the Oracle Restart configuration with the `add` command.

- To define a new listener that was not created with NETCA, use the following syntax:

```
$ srvctl add listener -l MYLISTENER -p TCP:1525 -o
/u01/app/oracle/product/12.1.0/grid
```

- To specify a nondefault location for the new listener's networking files, use the following syntax:

```
$ srvctl setenv listener -l MYLISTENER -t
TNS_ADMIN=/usr/local/oracle
```

ORACLE

The SRVCTL utility can be used to manually add components to the Oracle Restart configuration with the `add` command. If the component was created with an Oracle utility such as NETCA, DBCA, ASMCA, or OUI, it would have been automatically added to the Oracle Restart configuration and it would not be necessary to manually add it.

The slide illustrates manually adding a new listener called `MYLISTENER` to the Oracle Restart configuration. The listener will use the software binaries of the Grid home installation, but will depend on a nonstandard location for the networking files. For this example, it is assumed that the `listener.ora` networking file has been created in the `/usr/local/oracle` directory. The `setenv` command of the `srvctl` utility is used to define environment variables that may be needed for specific components. The `TNS_ADMIN` environment variable is set to the nondefault location of the `listener.ora` file, and is defined only for the listener named `MYLISTENER`. This will not have an impact on any other listeners that may already exist and that use different directories for the networking files.

The SRVCTL utility includes a `remove` command to manually delete a component from the Oracle Restart configuration. To delete the listener created above, use the following syntax:

```
srvctl remove listener -l mylistener -f
```

This will also remove the environment variable that was associated with the listener.

Quiz

You invoke the SRVCTL utility from the Oracle Grid Infrastructure home when working with:

a. Listeners
b. ASM instances
c. Database instances
d. ASM disk groups

Answer: a, b, and d

Summary

In this lesson you should have learned how to:

- Start and stop Oracle Restart
- Use Oracle Restart to manage components

Practice 6: Overview

This practice covers the following topics:

- Stopping and starting Oracle Restart
- Using SRVCTL to manage components:
  - Viewing current configuration
  - Adding components to the configuration
  - Viewing component status
  - Stopping and starting components
  - Modifying component configuration (environment variables)
  - Adding services for a PDB

7

Introduction to Upgrading to Oracle Database 12*c*

Objectives

After completing this lesson, you should be able to:

- Describe upgrade methods
- Describe data migration methods
- Describe the upgrade process

Defining Upgrade and Data Migration

- *Upgrading*:
  - Transforms an existing Oracle Database environment into an Oracle Database 12*c* environment
  - Does not affect user data
- *Migrating data*:
  - Moving data from one Oracle database to another database
  - Does not include upgrading to a new release of Oracle Database

Upgrade refers to transforming an existing Oracle Database environment into a new release of Oracle Database, in the case of this course to Oracle Database 12*c*.

Data migration refers to moving data from one Oracle Database into a database that was created in advance for the migrated data. Migration does not involve an upgrade of the Oracle Database software.

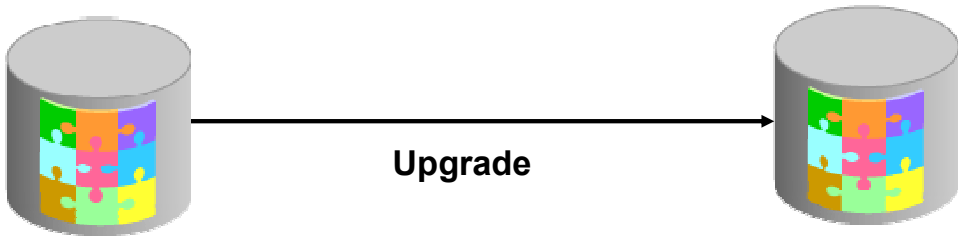
Note: Best practice is to perform migration and upgrade operations separately.

This lesson provides details on how you can upgrade to Oracle Database 12*c* and how you can migrate data from an Oracle database to an Oracle Database 12*c*.

Upgrade Methods

The upgrade methods are:

- Database Upgrade Assistant (DBUA): Recommended for major release and patch release upgrades
- Manual method using SQL scripts and utilities



Oracle Database 11*g* **Oracle Database 12*c***

You can ease the process of upgrading a database to Oracle Database 12*c* through careful planning and use of the Oracle Database tools. The method you choose to upgrade depends on several factors. Oracle Database supports the following methods for upgrading a database:

- **Using the Database Upgrade Assistant (DBUA):** This tool can be launched by the Oracle Universal Installer (OUI) or as a stand-alone tool at any time in the future to upgrade a database. DBUA provides a graphical user interface (GUI) that guides you through the upgrade of a database.
- **Performing a manual upgrade:** A manual upgrade is a command-line upgrade of a database by using SQL scripts and utilities. Though a manual upgrade gives you finer control over the upgrade process, it is susceptible to errors if any of the upgrade or preupgrade steps are either not followed or are performed out of order.

Note: DBUA is the preferred method of upgrading a database. It is highly recommended that you use DBUA to upgrade to Oracle Database 12*c*.

Database Upgrade Assistant: Advantages and Disadvantages

- Advantages:
  - Automates all tasks
  - Performs both release and patch set upgrades
  - Supports RAC, single instance, and ASM
  - Informs the user and fixes upgrade prerequisites
  - Automatically reports errors found in spool logs
  - Provides complete HTML report of the upgrade process
  - Command-line interface allows ISVs to automate
- Disadvantages:
  - Offers less control over individual upgrade steps

ORACLE

The Database Upgrade Assistant (DBUA):

- Guides you through the upgrade process and configures a database for the new release
- Automates the upgrade process and makes appropriate recommendations for the configuration options, such as tablespaces and redo logs
- Shows the upgrade progress for each component, while the upgrade is running
- Writes detailed trace and log files and produces a complete HTML report for later reference
- Automatically locks new user accounts in the upgraded database to enhance security
- Proceeds to create new configuration files (initialization parameter and listener files) in the new Oracle home
- Upgrades all the database and configuration files on all nodes in the cluster, in an Oracle RAC environment
- Supports a silent mode of operation where no user interface is presented to the user

If the DBA requires more control over the individual steps in the upgrade process, a manual upgrade is still possible. Usually, however, the manual upgrade method is more error prone, is harder to automate, and involves a greater amount of work than upgrading with the DBUA.

Manual Upgrade: Advantages and Disadvantages

- Advantages:
  - The DBA controls every step of the upgrade process.
- Disadvantages:
  - It involves more work for the DBA, who must:
    - Perform a manual space check for the `SYSTEM` and `SYSAUX` tablespaces
    - Manually adjust all obsolete or deprecated initialization parameters
    - Perform a user-driven backup of the database
  - It is subject to errors.

A manual upgrade consists of running SQL scripts and utilities from a command line to upgrade to Oracle Database 12*c*. Depending on the release of the database being upgraded, you may need to perform additional preupgrade steps. If you must upgrade to an intermediate release, check the release documentation for the release that you are upgrading to.

Though a manual upgrade gives you finer control over the upgrade process, it is susceptible to errors if any of the upgrade or preupgrade steps are either not followed or are performed out of order. In contrast, DBUA performs all the necessary preupgrade and upgrade steps.

Refer to the *Oracle Database Upgrade Guide* for a list of the scripts and the order in which they should be run.

Upgrade Paths

- Direct upgrade to Oracle Database 12*c* is supported for Oracle Database release 10.2.0.5, 11.1.0.7, or 11.2.0.2 and higher
- Upgrade to an intermediate release is required for Oracle Database release 11.2.0.1 and releases before 10.2.0.5

If your Oracle Database software is release 10.2.0.5, 11.1.0.7, or 11.2.0.2 or later, you can perform a direct upgrade to Oracle Database 12*c*.

If your Oracle Database software is earlier than 10.2.0.5 or is release 11.2.0.1, then you must upgrade to an intermediate Oracle Database release before upgrading to Oracle Database 12*c*. The intermediate release should be one of the releases that can be directly upgraded to Oracle Database 12*c*.

Upgrade Method Limitations

Both DBUA and the manual method have the following limitations:

- Target and source must be on the same platform.

ORACLE

Both DBUA and the manual upgrade method have certain limitations:

- The source and target are the same database. The upgrade is an in-place operation. The database stays on the same platform. No change in operating system (OS) or hardware can take place during the upgrade. The OS must be compatible with both versions of the database.

Upgrading to a New Release of Oracle Database

Step	Tasks
1. Prepare to upgrade	Choose an upgrade method Choose an Oracle home Develop a test plan
2. Upgrade a test database	Test the upgrade process using a test database
3. Test the upgraded database	Complete planned tests ensuring there are no issues. Repeat steps 1-3 if necessary.
4. Prepare the production database	Back up the production database
5. Upgrade the production database	Upgrade to the new release of Oracle Database
6. Tune and adjust the upgraded database	Implement new features Test applications

ORACLE

The table in the slide lists the major steps that you should perform to upgrade to a new release of Oracle Database.

Basic tasks of each step are as follows:

1. Prepare to upgrade: Become familiar with the new features in the release. Develop a test plan and backup strategy.
2. Upgrade a test database: Test the upgrade process by using a test database, not the production database system.
3. Test the upgraded database: Use the test plan developed in step 1 and test the upgraded database. If there are any issues with the upgraded system, address the issues and retest.
4. Prepare the production database: Prepare the production database system for the upgrade. Schedule downtime as necessary and perform a full or incremental backup.
5. Upgrade the production database: Use the procedures you tested in steps 2 and 3 to upgrade the production database. Back up the database after the upgrade.
6. Tune and adjust the upgraded production database: Implement new features. Test the applications to ensure they operate correctly. Tune the database as necessary.

Migration Methods

The migration methods are:

- Oracle Data Pump
- `CREATE TABLE AS SELECT` SQL statement

You can use Oracle Data Pump to perform a full or partial export from your database followed by a full or partial import into a new Oracle Database 12*c* database. The use of Oracle Data Pump is described in the lesson titled “Migrating Data by Using Oracle Data Pump.”

Note: There may be installations to be migrated that do not support Oracle Data Pump, such as versions before Oracle Database 10.1.0. In those cases legacy Export and Import may be used.

The `CREATE TABLE AS SELECT` SQL statement can be used to copy data from one database into a new Oracle Database 12*c* database. Refer to the *Oracle Database SQL Language Reference* for additional information about this command.

Using the Export and Import Method

- This method is required for certain situations:
  - Change in platform
  - Upgrade from unsupported versions
- Export and import are useful in certain cases:
  - Changes in character set
  - In combination with upgrade

You can use the Oracle Data Pump Export and Import utilities to perform a full or partial export from your existing database, followed by a full or partial import into a new Oracle Database 12*c* database. Export/Import can copy a subset of the data, leaving the database unchanged.

The Export and Import method is required in certain situations.

- If you want to change operating systems or hardware platforms, you cannot use the DBUA.
- In some situations, a significant effort is required to move the database to a version where the DBUA may be used. In these cases, the export and import method may be used.

The Export and Import method is useful in certain cases

- Character set conversion may be required in some upgrade scenarios.
- If there is a partial export of problem data, you can use the DBUA and import. This method can be used to solve character set issues involving a small number of tables.

Summary

In this lesson, you should have learned how to:

- Describe upgrade methods
- Describe data migration methods
- Describe the upgrade process

Practice 7: Overview

This practice covers the following topics:

- Creating a plan for each case:
  - Case A: Upgrade an 11*g* Release 2 database on file system to 12*c*
  - Case B: Upgrade an 11*g* Release 2 database on ASM to 12*c*
  - Case C: Upgrade an 11*g* Release 2 database on ASM to 12*c* and migrate to new OS or platform
  - Case D: Upgrade an11*g* Release 2 database to 12*c* with minimal downtime

ORACLE

8

Preparing to Upgrade to Oracle Database 12*c*

Objectives

After completing this lesson, you should be able to:

- List upgrade requirements when certain features or options are used in Oracle Database
- Use the pre-upgrade information tool before performing an upgrade
- Prepare the new Oracle home prior to performing an upgrade

Preparing to Upgrade

1. Familiarize yourself with the features of Oracle Database 12*c*.
2. Determine the upgrade path.
3. Choose an upgrade method.
4. Choose an OFA-compliant Oracle home directory.
5. Prepare a backup and recovery strategy.
6. Develop a test plan to test your database, applications, and reports.

ORACLE

In preparation of your upgrade to Oracle Database 12*c*, you should become familiar with the features of Oracle Database 12*c* that you want to use. The *Oracle Database New Features Guide 12c Release 1* provides a comprehensive list of the new features.

Planning the Upgrade

1. Choose an upgrade method.
2. Test the upgrade:
   a. Prepare the test source
   b. Upgrade
   c. Perform postupgrade steps
3. Record problems.
4. Record lessons learned.

Choosing an Upgrade Method

In many cases, the upgrade method chosen depends on a number of factors. The size of the database and the time required to perform the upgrade are two of the most important factors. There are other factors (such as platform, storage, and character sets) that may limit the choice of the upgrade method.

Testing the Upgrade

In most production environments, the cost of down time is unacceptably high. So, testing the upgrade becomes essential. As in any testing situation, the test environment should be as similar to the production environment as possible. You would then perform the upgrade from preupgrade to postupgrade.

Recording Problems

It is not unusual to find problems that were not anticipated in the plan during an upgrade. Ensure that these issues are documented, especially the solutions.

Recording Lessons Learned

In sites where there are multiple databases to be upgraded, a record of the lessons learned through the course of upgrade testing, and the upgrades themselves can be very valuable in reducing the time it takes to perform each upgrade.

Developing a Test Plan

Type of Testing	Description
Upgrade	Plan and test the upgrade path
Minimal	Move all or part of an application to the new database and run the application without enabling any new database features
Functional	Perform a set of tests in which new and existing features and functions of the system are tested after the upgrade
High Availability	Perform tests to ensure that the upgraded database system meets RTO and RPO business requirements
Integration	Perform tests to ensure that component interaction is correct
Performance	Perform tests to compare the performance of various SQL statements in the new database with the performance of those same statements in the current database
Volume/load stress	Perform tests of the entire upgraded Oracle database under high volume and loads

ORACLE

You should develop a planned set of tests to validate all stages of the upgrade process.

Your test plan should include the following types of testing:

- **Upgrade testing:** Planning and testing the upgrade path
- **Minimal testing:** Moving all or part of an application from the current database to the new database and running the application without enabling any new database features. This type of testing helps to identify any application startup or invocation problems.
- **Functional testing:** Performing a set of tests in which new and existing features and functions of the system are tested after the upgrade
- **High availability testing:** Performing tests to ensure that the upgraded database system meets Recovery Time Objective (RTO) and Recovery Point Objective (RPO) business requirements
- **Integration testing:** Performing tests to ensure that interactions among components of the system is correct
- **Performance testing:** Performing tests to compare the performance of various SQL statements in the new database with the performance of those same statements in the current database
- **Volume and load stress testing for Oracle Database upgrades:** Performing tests of the entire upgraded Oracle database under high volume and loads

Performance Testing

You can use the following Oracle Database features to conduct performance tests:

- Database Replay: Perform real-world testing of a database upgrade using a production workload before actually upgrading the production database
- SQL Performance Analyzer: Forecast the impact of system changes on a SQL workload

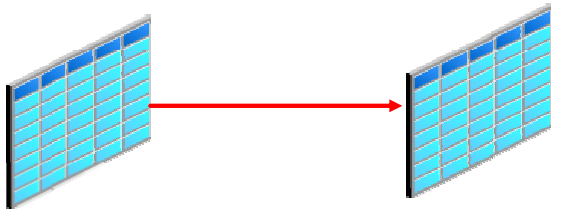
Performance tests are used to compare the performance of SQL statements in the upgraded database with the performance of the same statements in the current database. Prior to upgrading your database, you should ensure that you understand the performance profile of the application in the current database.

The following Oracle Database features can be used to conduct performance tests:

- **Database Replay:** You can use Database Reply to perform real-world testing of your production database workload before you upgrade the database. Database Replay captures the database workload on the current production system and enables you to replay it on a test system.
- **SQL Performance Analyzer:** Use SQL Performance Analyzer to forecast the impact of system changes on a SQL workload. SQL statements that will be impacted by the upgrade are identified enabling you to evaluate the impact of the upgrade.

Requirements for Databases Using Oracle Label Security or Oracle Database Vault

Execute the `olspreupgrade.sql` script to move the `AUD$` table from the `SYSTEM` schema to the `SYS` schema.



If you are upgrading from a release prior to Oracle Database release 12.1 that uses Oracle Label Security (OLS) and Database Vault, you must first execute the OLS preprocess script, `olspreupgrade.sql`, to process the `AUD$` table. The OLS preprocess script moves the `AUD$` table from the `SYSTEM` schema to the `SYS` schema.

To use the OLS preprocess script, copy `ORACLE_HOME/rdbms/admin/olspreupgrade.sql` from the newly installed Oracle home to the Oracle home of the database to be upgraded.

It is recommended that you archive your audit trail before executing the `olspreupgrade.sql` script. Refer to the *Oracle Database Security Guide* for detailed information about archiving the audit trail.

Requirement for Databases Using Oracle Warehouse Builder

- Oracle Warehouse Builder (OWB) is not installed as part of the software for Oracle Database 12*c*.
- OWB components that may exist in earlier releases are not upgraded as part of the Oracle Database upgrade process.
- You can use OWB release 11.2.0.3 with Oracle Database 12*c* as follows:
  - Add Oracle Database 12*c* access to an existing standalone OWB 11.2.0.3 installation.
  - Keep an existing OWB 11.2.0.3 Installation in-place with Oracle Database Release 11.2.0.3.
  - Use the standalone OWB 11.2.0.3 installation where available.

ORACLE

Oracle Warehouse Builder (OWB) is not installed as part of Oracle Database 12*c*, and OWB components that may exist in earlier releases are not upgraded as part of the Oracle Database upgrade process. However, you can use OWB release 11.2.0.3 with Oracle Database 12*c*. You cannot use OWB releases earlier than release 11.2.0.3 with Oracle Database 12*c*.

You can use OWB release 11.2.0.3 with Oracle Database 12*c* as follows:

- Add Oracle Database 12*c* access to an existing standalone OWB 11.2.0.3 installation: Oracle provides a patch update that enables OWB 11.2.0.3 to be used with Oracle Database 12*c*.
- Keep the existing OWB 11.2.0.3 installation in-place with Oracle Database Release 11.2.0.3: If OWB release 11.2.0.3 is running on a platform where a standalone installation is not available, you will need to keep the Oracle Database release 11.2.0.3 software in-place after the migration to Oracle Database 12*c* so that you can run OWB from this Oracle home.
- Use the standalone OWB 11.2.0.3 installation: If OWB 11.2.0.3 is running on a platform where a standalone installation is available (on Linux and Windows), you can install the standalone software and then remove the Oracle Database release 11.2.0.3 software.

Using the Pre-Upgrade Information Tool

After installing the Oracle Database 12*c* software you can optionally use the Pre-Upgrade Information Tool to perform configuration checks:

- Execute the `preupgrd.sql` script
- Review the information displayed by the Pre-Upgrade Information Tool
  - `preupgrade.log` contains the output of the Pre-Upgrade Information Tool
- Resolve issues
  - `preupgrade_fixups.sql` script addresses issues that can be fixed using SQL*Plus in the source database
  - `postupgrade_fixups.sql` script addresses issues that can be fixed after the database has been upgraded

After you have installed the software for Oracle Database 12*c*, you can analyze your database before upgrading it to the new release by using the Pre-Upgrade Information Tool. Execute the `preupgrd.sql` script from the environment of the database you are planning to upgrade.

The Pre-Upgrade Information Tool generates "fix-up" scripts that you can execute to resolve issues that are flagged in the source database:

- The `preupgrade_fixups.sql` script can be used to fix issues in the source database.
- The `postupgrade_fixups.sql` script can be used to fix issues after the database has been upgraded.

A log file named `preupgrade.log` is created containing the output of the Pre-Upgrade Information Tool and information about issues that require manual intervention.

Although it is not mandatory to use the Pre-Upgrade Information Tool prior to performing the upgrade, using it gives you an opportunity to address issues in advance. Before upgrading to the new release of Oracle Database, it is recommended that you analyze the information and warnings displayed by the Pre-Upgrade Information Tool.

Backing Up the Database

Perform an online backup by using RMAN

```
RUN
{
ALLOCATE CHANNEL chan_name TYPE DISK;
BACKUP DATABASE FORMAT 'some_backup_directory%U' TAG
   before_upgrade;
BACKUP CURRENT CONTROLFILE FORMAT 'controlfile location
   and name';
}
```


After running the Pre-Upgrade Information Tool, it is recommended that you back up the database. To minimize downtime, you may perform an online backup.

Summary

In this lesson, you should have learned how to:

- List upgrade requirements when certain features or options are used in Oracle Database
- Use the pre-upgrade information tool before performing an upgrade
- Prepare the new Oracle Home prior to performing an upgrade

Practice 8: Overview

This practice covers the following topics:

- Executing preliminary steps and the pre-upgrade script
- Performing the pre-upgrade information tool recommendations
- Completing prerequisite steps before upgrading
- Performing a full database backup
- Performing a transportable database export

9

Upgrading to Oracle Database 12*c*

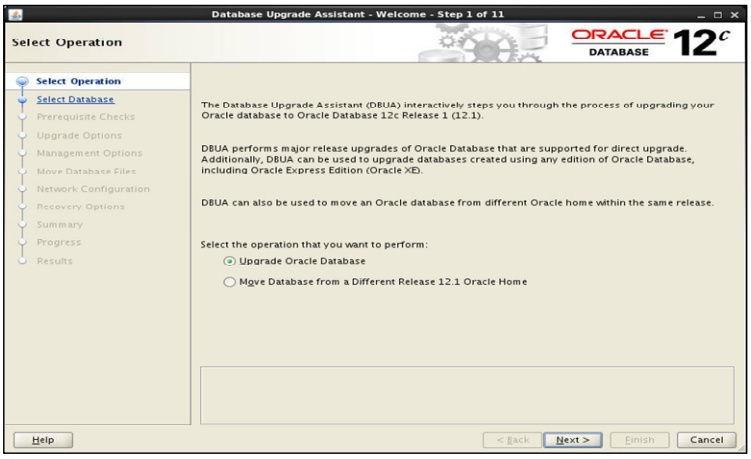
Objectives

After completing this lesson, you should be able to:

- Upgrade the database to Oracle Database 12*c* by using the Database Upgrade Assistant (DBUA)
- Perform a manual upgrade to Oracle Database 12*c* by using scripts and tools
- Migrate a non-CDB database to a CDB database

Upgrading by Using the Database Upgrade Assistant (DBUA)

- You can use the Database Upgrade Assistant (DBUA) to:
  - Upgrade to Oracle Database 12*c*
  - Move a database from an existing 12*c* Oracle home
- DBUA includes a GUI interface and silent mode



The DBUA guides you through the upgrade process and configures a database for the new release. The DBUA automates the upgrade process and makes appropriate recommendations for configuration options, such as tablespaces and redo logs.

The DBUA also supports a silent mode of operation. Silent mode enables you to use a single statement for the upgrade.

Key DBUA Features

- Provides for recoverability:
  - Performs a backup of the database before upgrade
  - Can restore the database after upgrade (if needed)
- Runs all the necessary scripts to perform the upgrade
- Displays the upgrade progress at a component level
- Performs configuration checks by executing the `preupgrd.sql` script

Before starting the upgrade process, it is strongly recommended that you back up your existing database, although the DBUA can perform a backup during the preupgrade stage.

If you use the DBUA to back up your database, it creates a copy of all your database files in the directory that you specify. The DBUA automatically performs this cold backup after it shuts down the database and before it begins performing the upgrade procedure. However, the cold backup does not compress your database files and the backup directory must be a valid file system path. In addition, the DBUA creates a batch file in the specified directory. You can use this batch file to restore the database files if needed.

During the upgrade, the DBUA automatically modifies or creates new required tablespaces and invokes the appropriate upgrade scripts. While the upgrade is running, the DBUA shows the upgrade progress for each component. It then creates new configuration files (parameter and listener files) in the new Oracle home. The DBUA provides a comprehensive summary of the preupgrade checks when finished.

Key DBUA Features

- Creates or updates configuration files:
  - Creates `init.ora` and SPFILE in the new Oracle home
  - Updates network configurations
  - Uses OFA-compliant locations
  - Updates the Oracle Internet Directory database information
- Oracle Enterprise Manager:
  - DBUA allows you to register a database with Enterprise Manager Cloud Control
  - Configure port for Enterprise Manager Database Express
- Provides logging and tracing:
  - Writes detailed trace and logging files to `$ORACLE_BASE/cfgtoollogs/dbua/<sid>/upgradeNN`

During the upgrade, the DBUA automatically modifies or creates a new `init.ora` and server parameter file (SPFILE) in the new Oracle home directory. In addition, it updates the network configurations, creates the required tablespaces, and invokes the appropriate upgrade scripts. While the upgrade is running, the DBUA shows the upgrade progress for each component. The DBUA writes detailed trace and log files in `$ORACLE_BASE/cfgtoollogs/dbua/<sid>/upgradeNN` and produces a complete HTML report for later reference.

Key DBUA Features

- Minimizes down time:
  - Recompiles packages in parallel
  - Does not require user interaction after upgrade starts
- Supports security features
  - Locks new users in the upgraded database
- Supports Real Application Clusters:
  - Upgrades all nodes
  - Upgrades all configuration files

After completing the preupgrade steps, the DBUA automatically archives redo logs and disables archiving during the upgrade phase.

To enhance security, the DBUA automatically locks the new user accounts in the upgraded database.

The DBUA is fully compliant with the Oracle RAC environments. In RAC environments, the DBUA upgrades all the database and configuration files on all nodes in the cluster.

Manually Upgrading to Oracle Database 12*c*

1. Execute the Pre-Upgrade Information Tool (`preupgrd.sql`).
2. Back up the source database.
3. Prepare the new Oracle home.
4. Log in to SQL*Plus and shut down the database instance.
5. Log in to SQL*Plus in the Oracle Database 12*c* home and execute the `STARTUP UPGRADE` command.
6. Execute the Parallel Upgrade Utility (`catctl.pl`).
7. Restart the database instance.
8. Execute the Post Upgrade Status Tool (`utlu121s.sql`).
9. Execute additional scripts (`catuppst.sql` and `utlrp.sql`).

As an alternative to using DBUA, you can manually upgrade to Oracle Database 12*c* by performing the following steps:

1. Run the Preupgrade Information Tool by executing the `preupgrd.sql` script in SQL*Plus. This script analyzes your database and generates scripts that you can use to fix problems. You must execute this script prior to running the Parallel Upgrade Utility or the `catctl.pl` upgrade script will terminate with errors.
2. Back up your database by using Recovery Manager (RMAN).
3. Prepare the new Oracle home by copying configuration files and modifying the initialization parameter file.
4. Log in to SQL*Plus and shut down the database instance.
5. Set the environment variables for the Oracle Database 12*c* Oracle home and log in to SQL*Plus. Execute the `STARTUP UPGRADE` command. The `UPGRADE` keyword enables you to open the database before it is upgraded. The command also causes logons to be restricted to only `AS SYSDBA` sessions, disables system triggers, and performs additional operations that prepare the environment for the upgrade.
6. At the operating system prompt, run the Parallel Upgrade Utility by executing the `catctl.pl` script from the new Oracle home. This utility provides parallel upgrade options that reduce downtime. The script can also be executed in serial mode.

7. Log in to SQL*Plus again and restart the database instance. You must restart the instance because the `catctl.pl` script shut down the instance.
8. Run the Post-Upgrade Status Tool by executing the `utlu121s.sql` script. The script provides a summary of the upgrade operation.
9. Execute the `catuppst.sql` script to perform additional upgrade operations. This script is run at this time because it does not require the database to be in UPGRADE mode. Execute the `utlrp.sql` script to recompile PL/SQL and Java code.

Migrating a Pre-12.1 or 12.1 Non-CDB to CDB

There are two methods: A, and B

A1) Upgrade an existing pre-12.1 non-CDB to 12*c*

A2) Plug non-CDB into a CDB

OR

B1) Pre-create a PDB in a CDB

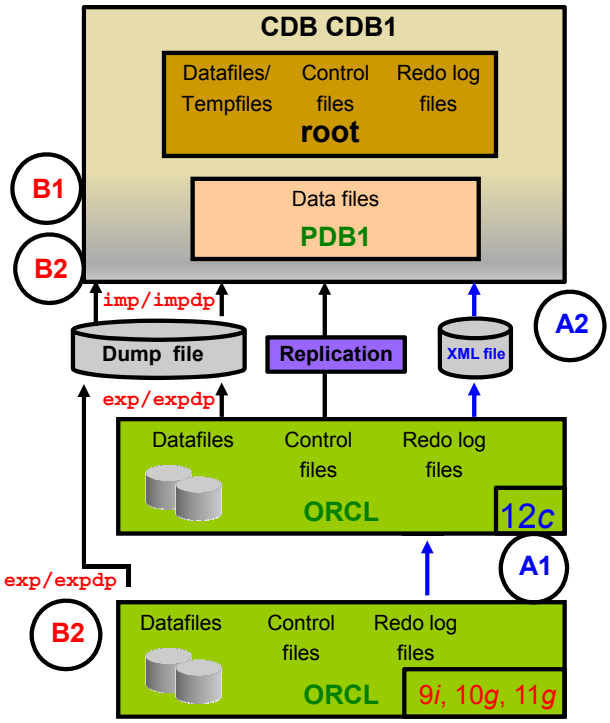
B2) Use 11*g* `expdp` / 12*c* `impdp`

or

Use 9*i* or 10*g* `exp` / 12*c* `imp`

or

Replication between non-CDB and PDB



There are two methods to migrate a non-CDB 11*g* database to a CDB.

The first method "A", consists of two steps:

A1) Upgrade the 11*g* database (or any other previous Oracle Database release) to 12*c* non-CDB

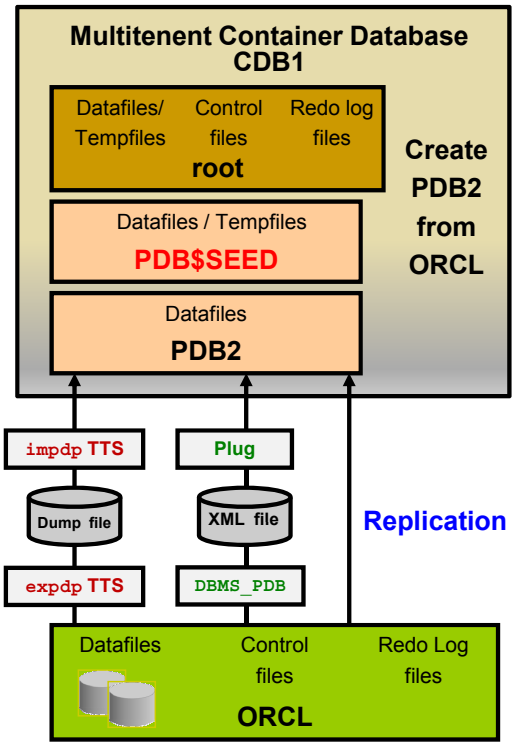
A2) Plug the 12*c* non-CDB into a CDB. Use the `DBMS_PDB.DESCRIBE` procedure to generate the XML file to plug the data files into the CDB as a new PDB. This is the fastest solution.

The second method, "B", consists of two steps:

B1) Pre-create a PDB in the CDB from the seed PDB. This operation establishes an Oracle Database 12*c* dictionary in the newly created PDB.

B2) Use Data Pump Export and Import (or the original Export and Import) or replication to load the 9*i*, 10*g*, and 11*g* data into the newly created PDB of the CDB.

Plugging a Non-CDB Database into a CDB



Three possible methods:

- TTS or TDB or full export/import
- XML file definition with `DBMS_PDB`
- Replication

Entities are created in the new PDB:

- Tablespaces: `SYSTEM`, `SYSAUX`
- A full catalog
- Common users: `SYS`, `SYSTEM`
- A local administrator (PDBA)
- A new default service

ORACLE

There are three possible methods to plug a non-CDB database into a CDB.

Whichever method is used, the non-CDB must be in a transactionally consistent state and opened in restricted mode.

It is appropriate to use Oracle Data Pump when:

- Both source and target databases are different endian.
- The source character set is not equal to the target character set , and is not a binary subset of the target.

Either use transportable tablespace (TTS), full conventional export / import, or full transportable database (TDB) provided that any user-defined object resides in a single user-defined tablespace. A Data Pump full transportable database does not support movement of XDB or AWR repositories. Only user-generated XML schemas are moved.

In other cases, using the `DBMS_PDB` package is the easiest option. The `DBMS_PDB` package constructs an XML file describing the non-CDB data files to plug into the CDB as a PDB. This method requires that the non-CDB is an Oracle 12*c* database.

If the `DBMS_PDB` package is not used, export/import is usually simpler than using GoldenGate replication, but export/import might require more down time during the switch from the non-CDB to the PDB.

Summary

In this lesson, you should have learned how to:

- Upgrade the database to Oracle Database 12*c* by using the Database Upgrade Assistant (DBUA)
- Perform a manual upgrade to Oracle Database 12*c* by using scripts and tools
- Migrate a non-CDB database to a CDB database

Practice 9: Overview

This practice covers the following topics:

- Upgrading an Oracle Database 11*g* Release 2 database to Oracle Database 12*c* by using DBUA
- Exporting a non-CDB application in preparation for a later practice
- Plugging a non-CDB database into a CDB

10

Performing Post-Upgrade Tasks

Objectives

After completing this lesson, you should be able to:

- Perform post-upgrade tasks
- Migrate to Unified Auditing

Required Tasks After Database Upgrade

Perform the following tasks as appropriate to your environment:

- For a manual upgrade:
  - Update the `ORACLE_HOME` and `PATH` operating system environment variables
  - Update the `oratab` file with the new Oracle home
- For all upgrade methods, update client scripts with the new Oracle home
- Upgrade the RMAN Recovery Catalog
- Configure the FTP and HTTP ports, and HTTP authentication for Oracle XML DB
- Enable Database Vault

After you upgrade to Oracle Database 12*c* and before you can consider the database operational, you must complete some postupgrade tasks. The tasks you need to complete depend on whether you performed a manual upgrade or used DBUA, and also on your configuration and use of features. Some common tasks include the following:

- Updating the environment variables following a manual upgrade: On Linux and UNIX operating systems, ensure that the `ORACLE_HOME` and `PATH` environment variables point to the new Oracle home.
- Updating the `oratab` file and client scripts with the new Oracle home: DBUA automatically points `oratab` to the new Oracle home. Regardless of the upgrade method you must check client scripts.
- Upgrading the RMAN Recovery Catalog: If the recovery catalog schema version is older than what is required by the RMAN client, you must upgrade it.
- Configuring the FTP and HTTP ports, and HTTP authentication for Oracle XML DB: The Oracle Database 12*c* DBCA does not configure ports for Oracle XML DB so you must manually configure them. You should also configure digest authentication for HTTP to take advantage of improved security features.
- Enabling Database Vault: Register Database Vault by using the `DVSYS.DBMS_MACADM.ENABLE_DV` procedure.

Recommended Tasks After Database Upgrade

It is recommended that you perform the following tasks after upgrading to Oracle Database 12*c*:

- Reset passwords to enforce case-sensitivity.
- Set threshold values for tablespace alerts.
- Implement new features as appropriate.
- Migrate to unified auditing.

You should consider the recommended tasks that are listed in the slide depending on the release you have migrated from and your use of various features. Additional information about each task follows and more detail can be found in the *Oracle Database Upgrade Guide 12c Release 1 (12.1)*:

- Reset passwords to enforce case-sensitivity: To take advantage of enforced case-sensitive passwords for releases earlier than 11.1.0.7, you must reset the passwords of existing users during the database upgrade. procedure. You can execute the `DBMS_VERIFIER.EXPIRE_ACCOUNTS_WITHOUT_LATEST_VERIFIER` procedure that forces users whose accounts do not yet have the latest verifier to change their passwords the next time they log in.
- Set threshold values for tablespace alerts: In an upgraded Oracle Database 12*c* database, Tablespace Alerts are disabled (the thresholds are set to null). Tablespaces in the database that are candidates for monitoring must be identified and the appropriate threshold values must be set. The default threshold values for a newly created Oracle Database 12*c* database are:
  - 85% full warning
  - 97% full critical

- Implement new features: Refer to the *Oracle Database New Features Guide* for a description of the new features in Oracle Database 12*c*. Attend the Oracle Database 12*c*: New Features for Administrators course to learn more about the new features.
- Migrate to unified auditing: Oracle Database 12*c* includes a new feature called the unified audit trail. With unified auditing, all Oracle Database audit trails (`SYS.AUD$` for the database audit trail, `SYS.FGA_LOG$` for fine-grained auditing, `DVSYS.AUDIT_TRAIL$` for Database Vault, and so on) are consolidated into one single audit trail. Additional information about unified auditing follows in this lesson.

Understanding Auditing Implementation

- *Mixed mode auditing* is the default when a new Oracle Database 12*c* database is created.
- Mixed mode auditing enables the use of:
  - Pre–Oracle Database 12*c* auditing features
  - *Unified auditing* features of Oracle Database 12*c*
- The recommendation from Oracle is to migrate to unified auditing.
- Query `V$OPTION` to determine if the database has been migrated to unified auditing:

```
SELECT value FROM v$option
WHERE parameter = 'Unified Auditing'
```

ORACLE

Prior to Oracle Database 12*c*, audit records from various sources were stored in different locations. Oracle Database 12*c* supports *unified auditing*, in which all audit records are stored in a single audit table.

When you create a new Oracle Database 12*c* database, mixed mode auditing is enabled. This mode enables you to use the auditing features available before Oracle Database 12*c* and also the unified auditing features. Mixed mode auditing is enabled by default through the `ORA_SECURECONFIG` predefined auditing policy for newly created databases.

If you are upgrading a database to Oracle Database 12*c*, you must manually migrate to unified auditing to use the unified auditing features.

Oracle Corporation recommends that you migrate to unified auditing.

Enabling Unified Auditing

1. In SQL*Plus, shut down the database instance:

```
SQL> SHUTDOWN IMMEDIATE
```

2. Shut down the listener:

```
$ lsnrctl stop
```

3. At the operating system prompt, enable the unified auditing executable:

```
$ cd $ORACLE_HOME/rdbms/lib
$ make -f ins_rdbms.mk uniaud_on ioracle ORACLE_HOME=$ORACLE_HOME
```

4. Restart the listener:

```
$ lsnrctl start
```

5. In SQL*Plus, restart the database instance:

```
SQL> STARTUP
```


Before enabling the unified auditing executable, shut down the database instance and the listener.

Change to the `$ORACLE_HOME/rdbms/lib` directory and execute the following command:

```
make -f ins_rdbms.mk uniaud_on ioracle ORACLE_HOME=$ORACLE_HOME
```

The `make` command is used to relink the Oracle executable with a different set of libraries to enable unified auditing.

After the `make` command completes, restart the listener and the database instance.

You can log in to SQL*Plus and verify that unified auditing has been enabled as follows:

```
SQL> select value
  2  from v$option
  3  where parameter = 'Unified Auditing';

VALUE
----------------------------------------------------------------
TRUE
```

Administering the Roles Required for Auditing

A user must be granted one of the following roles to perform auditing tasks:

- `AUDIT_ADMIN` enables the user to:
  - Create unified and fine-grained audit policies
  - Execute the `AUDIT` and `NOAUDIT` SQL statements
  - View audit data
  - Manage the audit trail (table in the `AUDSYS` schema)
- `AUDIT_VIEWER` enables the user to:
  - View and analyze audit data

ORACLE

Users must be granted the appropriate privilege to configure auditing and view audit data. To support separation of duty, two default roles are provided:

- `AUDIT_ADMIN`: Enables the grantee to configure auditing settings, create and administer audit policies (unified and fine-grained), and view and analyze audit data. This role is typically granted to a security administrator.
- `AUDIT_VIEWER`: Enables the grantee to view and analyze audit data. This role is typically granted to external auditors.

Summary

In this lesson, you should have learned how to:

- Perform post-upgrade tasks
- Migrate to Unified Auditing

Practice 10: Overview

This practice covers the following topics:

- Performing post-upgrade operations
- Enabling unified auditing
- Using unified auditing

11

Migrating Data by Using Oracle Data Pump

Objectives

After completing this lesson, you should be able to migrate data by using Oracle Data Pump.

Oracle Data Pump: Overview

As a server-based facility for high-speed data and metadata movement, Oracle Data Pump:

- Is callable via `DBMS_DATAPUMP`
- Provides the following tools:
  - `expdp`
  - `impdp`
  - GUI interface in Enterprise Manager Cloud Control
- Provides four data movement methods:
  - Data file copying
  - Direct path
  - External tables
  - Network link support
- Detaches from and re-attaches to long-running jobs
- Restarts Data Pump jobs

Oracle Data Pump enables high-speed data and metadata loading and unloading of Oracle databases. The Data Pump infrastructure is callable via the `DBMS_DATAPUMP` PL/SQL package. Thus, custom data movement utilities can be built by using Data Pump.

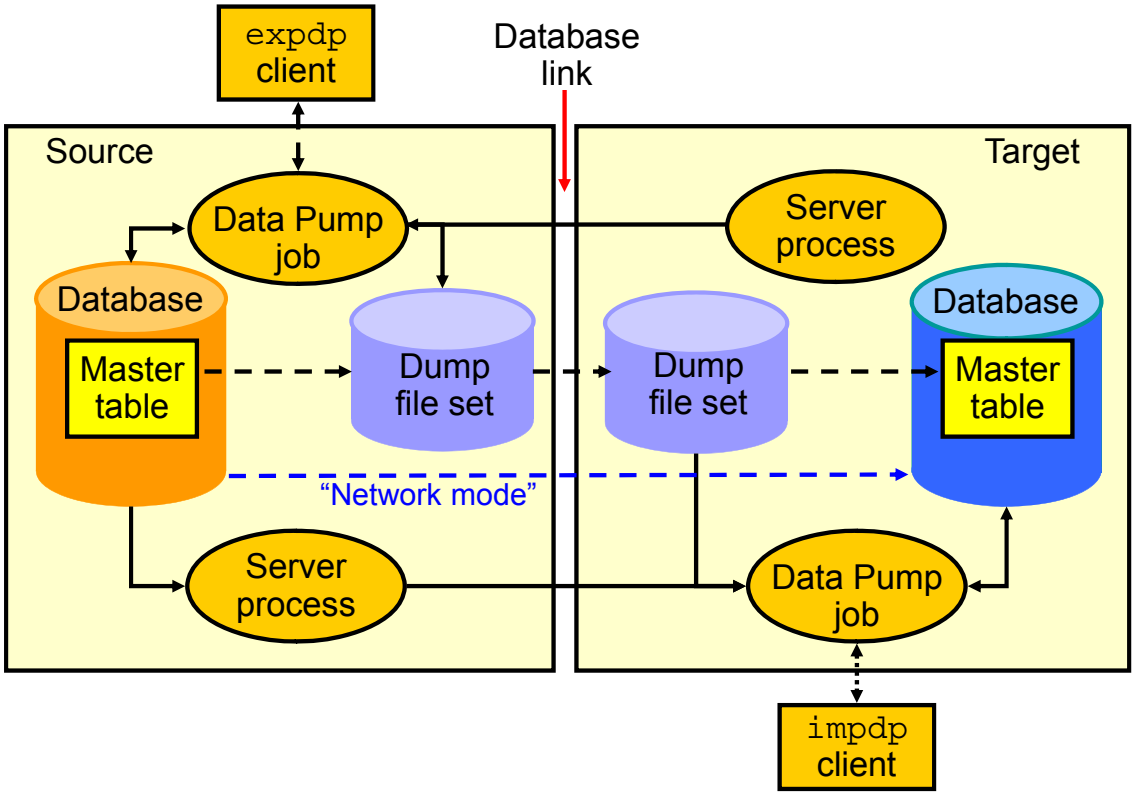
Oracle Database provides the following tools:

- Command-line export and import clients called `expdp` and `impdp`, respectively
- Export and import interface in Enterprise Manager Cloud Control

Data Pump automatically decides the data access methods to use; these can be either direct path or external tables. Data Pump uses direct path load and unload when a table's structure allows it and when maximum single-stream performance is desired. However, if there are clustered tables, referential integrity constraints, encrypted columns, or several other items, Data Pump uses external tables rather than direct path to move the data.

The ability to detach from and re-attach to long-running jobs without affecting the job itself enables you to monitor jobs from multiple locations while they are running. All stopped Data Pump jobs can be restarted without loss of data as long as the metainformation remains undisturbed. It does not matter whether the job is stopped voluntarily or involuntarily due to a crash.

Data Pump Export and Import Clients: Overview



Data Pump Export is a utility for unloading data and metadata into a set of operating system files called *dump file sets*. Data Pump Import is used to load metadata and data stored in an export dump file set into a target system.

The Data Pump API accesses its files on the server rather than on the client.

These utilities can also be used to export from a remote database directly to a dump file set, or to load the target database directly from a source database with no intervening files. This is known as *network mode*. This mode is particularly useful to export data from a read-only source database.

At the center of every Data Pump operation is the master table (MT), which is a table created in the schema of the user running the Data Pump job. The MT maintains all aspects of the job. The MT is built during a file-based export job and is written to the dump file set as the last step. Conversely, loading the MT into the current user's schema is the first step of a file-based import operation and is used to sequence the creation of all objects imported.

Note: The MT is the key to Data Pump's restart capability in the event of a planned or unplanned stopping of the job. The MT is dropped when the Data Pump job finishes normally.

Data Pump Utility: Interfaces and Modes

- Data Pump Export and Import interfaces:
  - Command line
  - Parameter file
  - Interactive command line
  - Enterprise Manager Cloud Control
- Data Pump Export and Import modes:
  - Full
  - Schema
  - Table
  - Tablespace
  - Transportable tablespace
  - Transportable database

ORACLE

You can interact with Data Pump Export and Import by using one of the following interfaces:

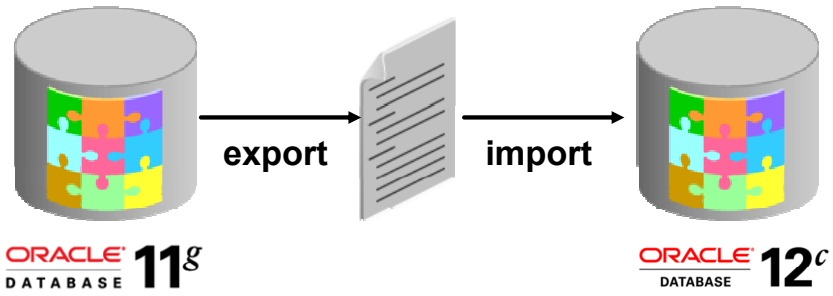
- **Command-line interface:** Enables you to specify most of the export parameters directly on the command line
- **Parameter file interface:** Enables you to specify all command-line parameters in a parameter file. The only exception is the `PARFILE` parameter.
- **Interactive-command interface:** Stops logging to the terminal and displays the export or import prompts, where you can enter various commands. This mode is enabled by pressing Ctrl + C during an export operation that is started with the command-line interface or the parameter file interface. Interactive-command mode is also enabled when you attach to an executing or stopped job.
- **GUI interface:** Select Schema > Database Export/Import. In the menu select the export or import operation you want to execute.

Data Pump Export and Import provide different modes for unloading and loading different portions of the database. The mode is specified on the command line by using the appropriate parameter. The available modes are listed in the slide and are the same as in the original export and import utilities.

Migrating by Using Oracle Data Pump

To migrate to Oracle Database 12*c* by using Oracle Data Pump:

1. Export data from the source database
2. Install Oracle Database 12*c* and create a database
3. Import data into the new Oracle Database 12*c* database



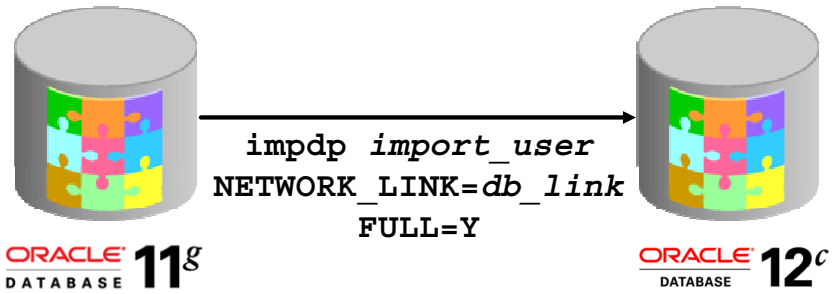
ORACLE

You can use Oracle Dump Export and Import to migrate your database to Oracle Database 12*c*. This method provides the following advantages:

- The database can remain available for user access during the upgrade. If a consistent snapshot of the database is required, the database must operate in restricted mode or must otherwise be protected from changes when the export executes.
- The upgraded database can be created on a different operating system or hardware platform.
- The database can be restructured as part of the import. You can create new tablespaces or modify existing tables, tablespaces, or partitions to be populated by imported data.

Importing by Using a Network Link

- Use Data Pump Import with a database link for a full database import
- User performing the Export must have the `DATAPUMP_EXP_FULL_DATABASE` role
- User performing the Import must have the `DATAPUMP_IMP_FULL_DATABASE` role

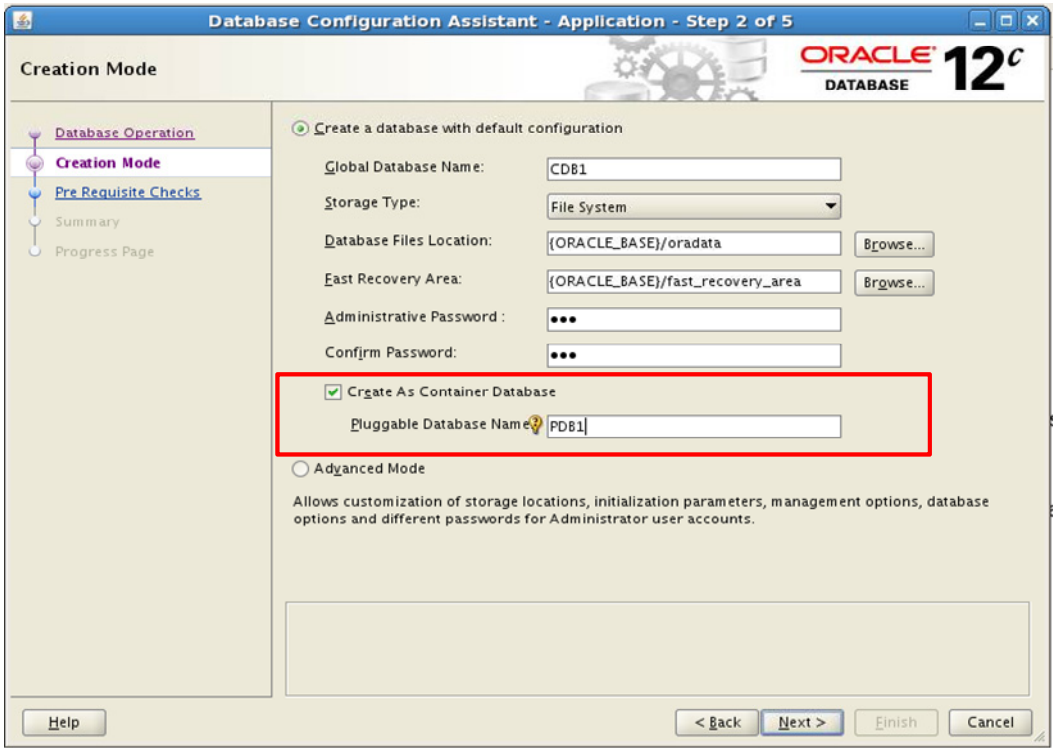


Data Pump Import can be used with a database link to perform a full database import. When you use this method, no intermediate dump files are created.

The user performing the export must be granted the `DATAPUMP_EXP_FULL_DATABASE` role. Specify this user when the database link is created.

The user performing the import must be granted the `DATAPUMP_IMP_FULL_DATABASE` role.

Creating a Container Database by Using DBCA

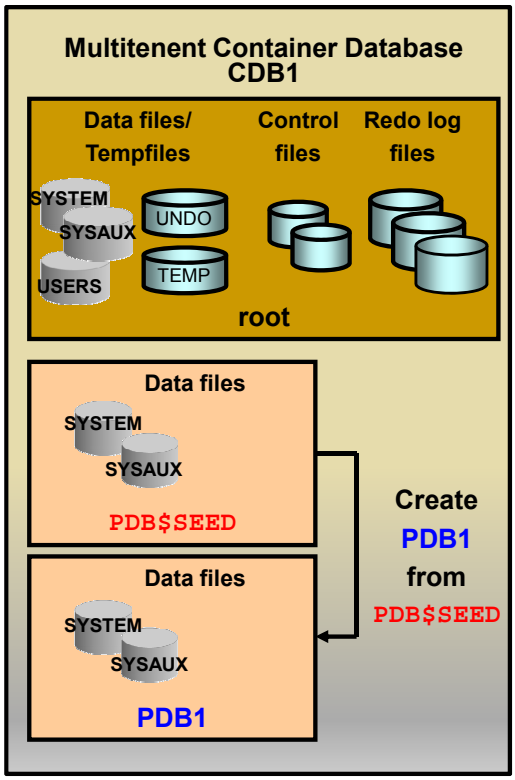


Another way to migrate a non-CDB database to a CDB is to create a multitenant container database and then import the database. To create a CDB by using DBCA, select “Create As Container Database.”

You also must provide a pluggable database name in the “Pluggable Database Name” field. If you select “Advanced Mode” you can create an empty CDB with only the root and seed containers.

In Advanced Mode, you can register the CDB with Enterprise Manager Cloud Control, configure the CDB for Enterprise Manager Database Express, and set passwords for the `SYS` and `SYSTEM` users.

Creating a New PDB from PDB$SEED



- Copies the data files from PDB$SEED data files
- Creates SYSTEM, SYSAUX tablespaces
- Creates a full catalog including metadata pointing to Oracle supplied objects
- Creates common users:
  - Superuser SYS
  - SYSTEM
- Creates a local user (PDBA) granted local PDB_DBA role
- Creates a new default service

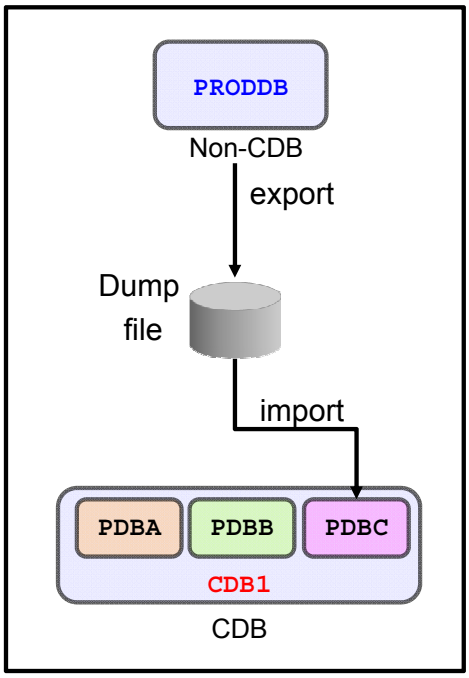
The creation of a new PDB from the seed is nearly instantaneous. The operation copies the data files from the READ ONLY seed PDB to the target directory defined in the CREATE PLUGGABLE DATABASE statement.

It creates tablespaces such as SYSTEM to store a full catalog including metadata pointing to Oracle-supplied objects, SYSAUX for local auxiliary data.

It creates default schemas and common users that exist in seed PDB, SYS who continues to have all superuser privileges, and SYSTEM who can administer the PDB.
It creates a local user (the PDBA) granted a local PDB_DBA role. Until the PDB SYS user grants privileges to the local PDB_DBA role, the new PDBA cannot perform any other operation than connecting to the PDB.

A new default service is also created for the PDB.

Exporting from a Non-CDB and Importing into a PDB



1. Export **PRODDB** with FULL clause:

```
$ expdp system@PRODDB FULL=Y
    DUMPFILE=proddb.dmp
```

2. Create PDBC if it does not exist in **CDB1**:

```
SQL> CONNECT sys@CDB1
SQL> CREATE PLUGGABLE DATABASE
  2  PDBC USING pdb$seed;
```

3. Open PDBC
4. Create a Data Pump directory in PDBC
5. Copy the dumpfile to the directory
6. Create same **PRODDB** tablespaces in PDBC for new local users objects
7. Import into PDBC with FULL and REMAP clauses:

```
$ impdp system@PDBC FULL=Y
    DUMPFILE=proddb.dmp
```

ORACLE

To export data from a non-CDB and import it into a PDB of a CDB, perform the steps as described in the slide.

The example in the slide describes how to perform a conventional full database export from the non-CDB and a conventional full database import into the PDB. You can also perform a full transportable, a tablespace, schema, or table-level export and import.

The tablespace export and import can be of either type: conventional or transportable.

The users exported from the non-CDB database are re-created as local users in the PDB.

The tablespaces for the new local users and objects must be created in the PDB before the import.

Summary

In this lesson, you should have learned how to:

- Migrate data by using Oracle Data Pump

Practice 11: Overview

This practice covers the following topics:

- Performing a transportable tablespace import
- Migrating to a PDB by using Data Pump