

Oracle 19c Database Administration I

(Oracle DBA I)

Agenda

Day 1:

1. Introduction to Oracle Database: 6
2. Accessing an Oracle Database: 15
3. Creating an Oracle Database by Using DBCA: 36
4. Creating an Oracle Database by Using a SQL Command: 48
5. Starting Up and Shutting Down a Database Instance: 56

Day 2:

6. Managing Database Instances: 67
7. Oracle Net Services Overview: 90
8. Configuring Naming Methods: 100
9. Configuring and Administering the Listener: 115
10. Configuring a Shared Server Architecture: 126

Agenda

Day 3:

11. Configuring Oracle Connection Manager for Multiplexing and Access Control: 136
12. Creating PDBs from Seed: 155
13. Using Other Techniques to Create PDBs: 164
14. Managing PDBs: 182
15. Database Storage Overview: 193

Day 4:

- 16: Creating and Managing Tablespaces: 211
17. Improving Space Usage: 230
18. Managing Undo Data: 259
19. Creating and Managing User Accounts: 278
20. Configuring Privilege and Role Authorization: 296

Agenda

Day 5:

- 21: Configuring User Resource Limits: 316
- 22: Implementing Oracle Database Auditing: 329
- 23. Introduction to Loading and Transporting Data: 361

Bonus Content / Time Permitting

- 24. Loading Data: 370
- 25: Transporting Data: 381
- 26: Using External Tables to Load and Transport Data: 413
- 27: Automated Maintenance Tasks: Overview: 423
- 28: Managing Tasks and Windows: 433
- 29: Database Monitoring and Tuning Performance Overview: 442
- 30: Monitoring Database Performance: 455
- 31: Database Processes: 470
- 32: Managing Memory: 490
- 33: Analyzing SQL and Optimizing Access Paths: 517

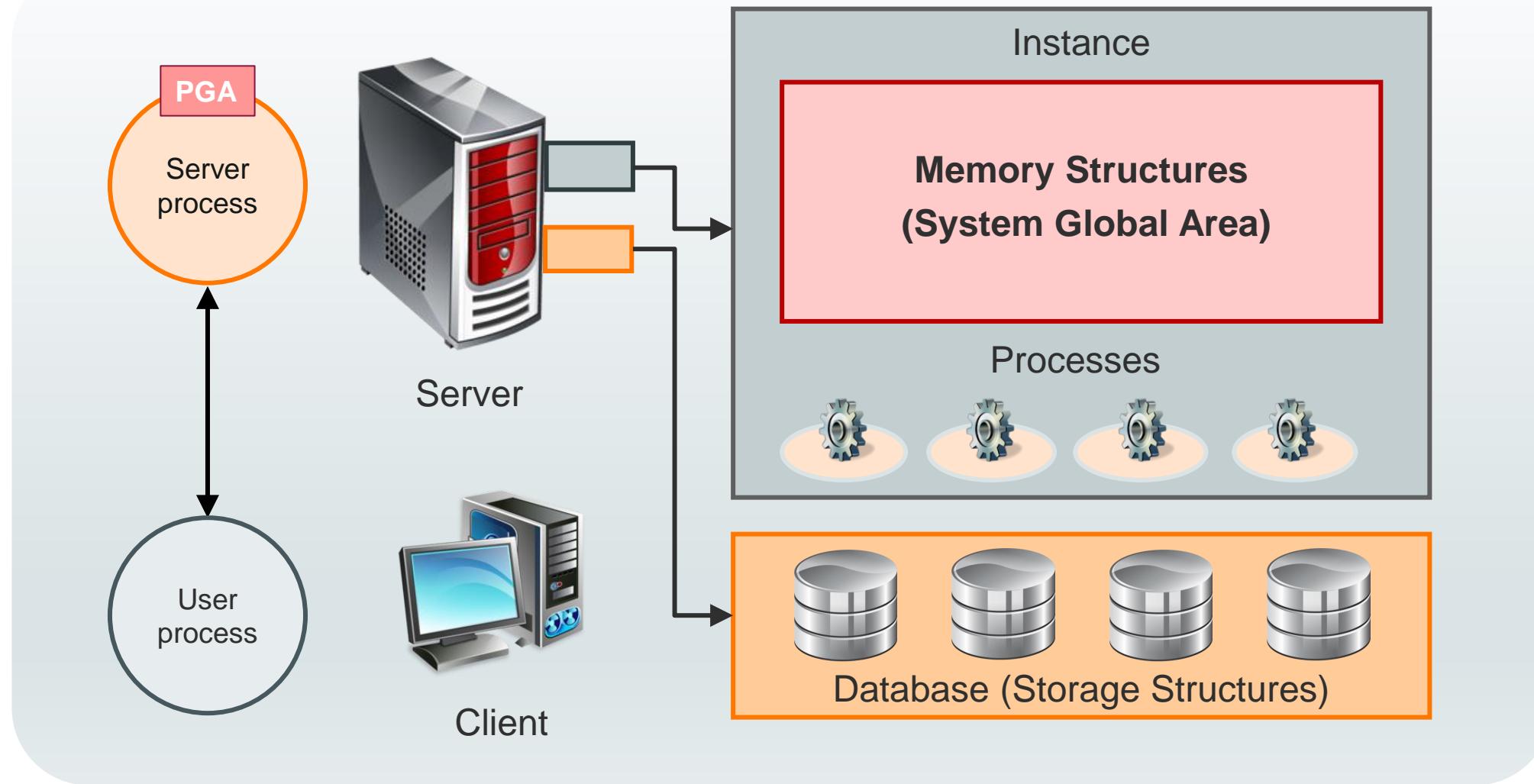
1. Introduction to Oracle Database

Objectives

After completing this lesson, you should be able to:

- List the major architectural components of Oracle Database
- Describe multitenant architecture
- Describe database sharding

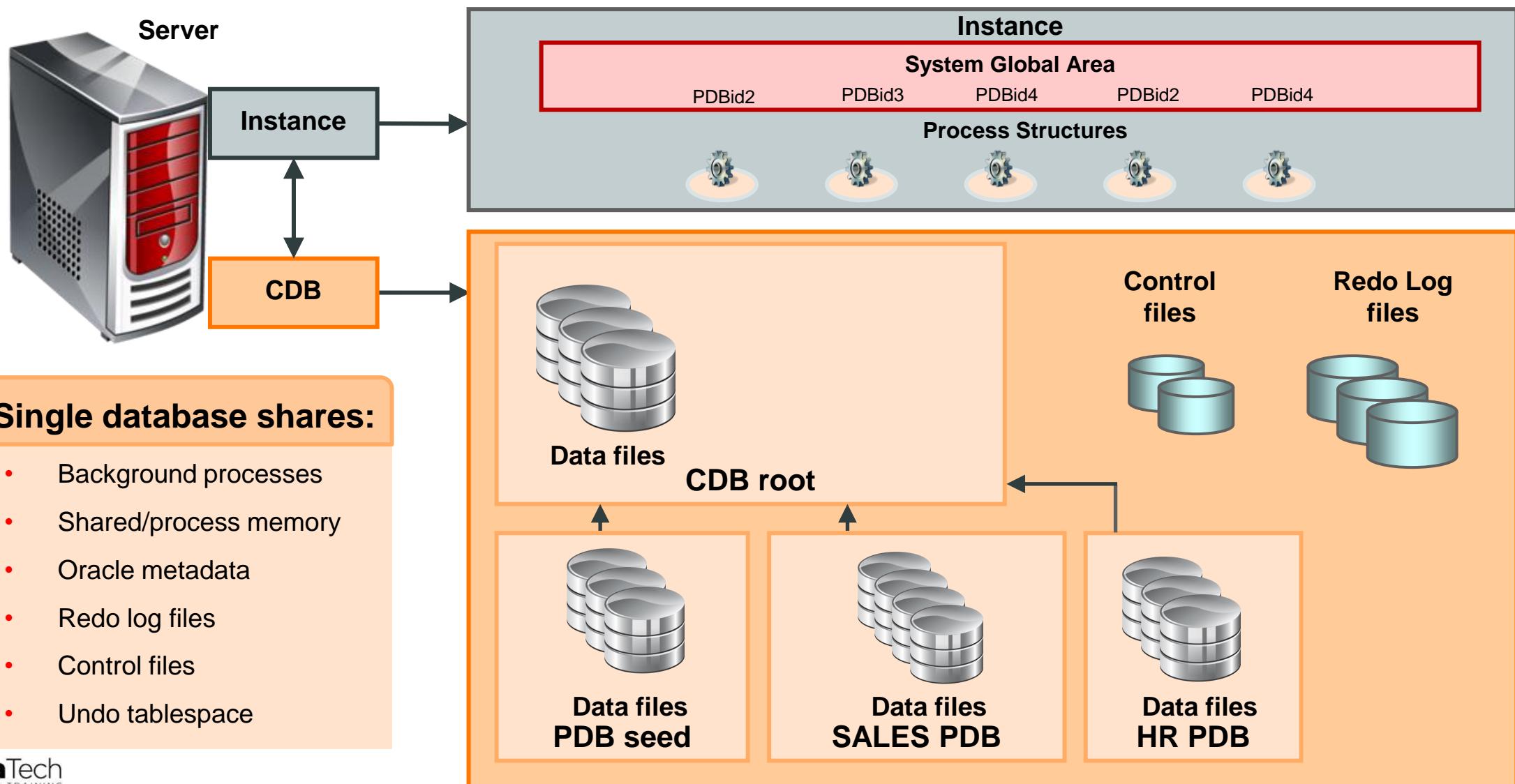
Oracle Database Server Architecture: Overview



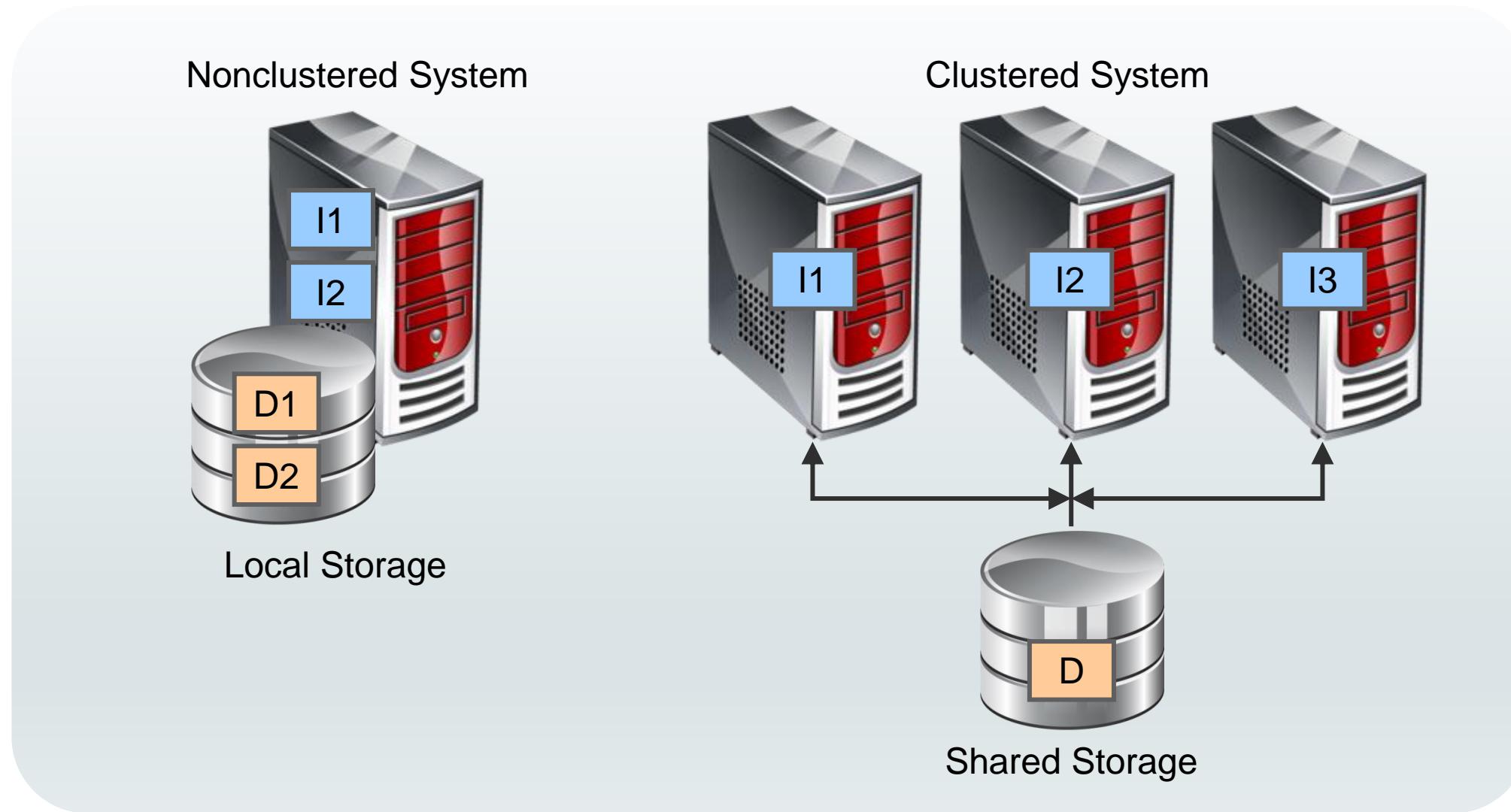
Oracle Multitenant Container Database: Introduction

- *Container*: A logical collection of data or metadata within the multitenant architecture
- *Pluggable database (PDB)*: A portable collection of schemas, schema objects, and nonschema objects
- Containers in a *multitenant container database (CDB)*:
 - CDB root container (also known as the *root*)
 - System container (includes the root and all PDBs)
 - Application containers
 - Root PDB
 - User-created PDBs
- All pluggable databases share:
 - Background processes
 - Shared memory management and some memory structures
 - Some of the Oracle metadata

Oracle Multitenant Container Database: Architecture

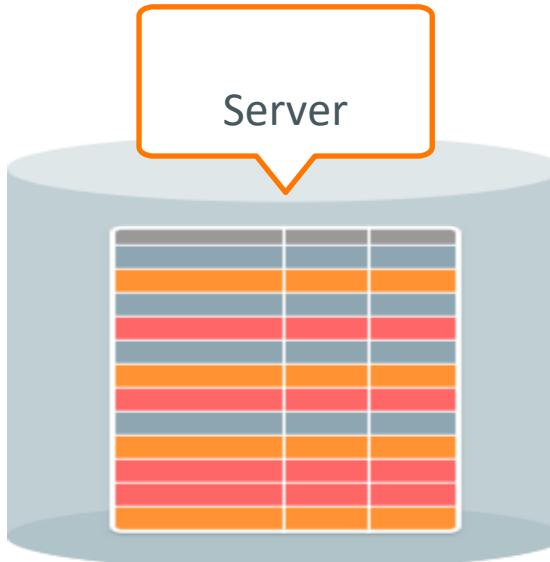


Oracle Database Instance Configurations

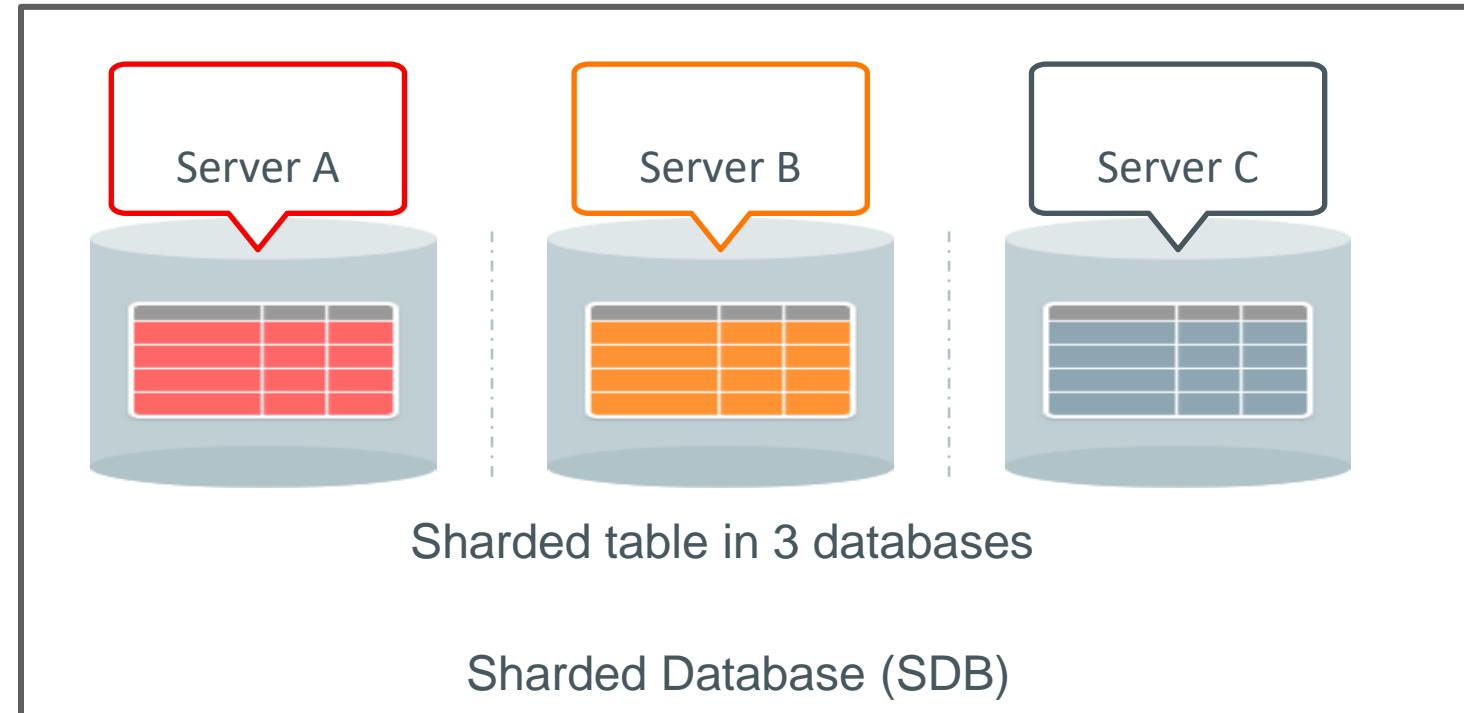


Database Sharding: Introduction

- A shared-nothing architecture for scalability and availability
- Horizontally partitioned data across independent databases
- Loosely coupled data tier without clusterware



Unsharded table in 1 database



Oracle Database Server: Interactive Architecture Diagram

Access the Interactive Architecture Diagram on the Oracle Help Center Oracle Database “What’s New” page.

<https://tinyurl.com/yyepn9ma>

Summary

In this lesson, you should have learned how to:

- List the major architectural components of Oracle Database
- Describe multitenant architecture
- Describe database sharding

2. Accessing an Oracle Database

Objectives

After completing this lesson, you should be able to:

- Connect to an Oracle Database
- Describe the tools used to access an Oracle Database

Connecting to an Oracle Database Instance

- You connect client applications to an Oracle Database by connecting to its database instance, not its database.
- A user session is a logical entity that represents the state of the current user login to the database instance.
- Examples of connecting to an Oracle Database instance:
 - By using operating system authentication

```
$ sqlplus / as sysdba
```

- By using Easy Connect Syntax

```
SQL> connect hr/hr@host1.example.com:1521/db.example.com
```

Oracle Database Tools

- Oracle Database tools each have their own purpose, and some operations can be performed in more than one tool.
- Choosing a tool for a task often comes down to personal preference.
- Tools include:
 - SQL*Plus
 - SQL Developer
 - SQL Developer Command Line (SQLcl)
 - Database Configuration Assistant (DBCA)
 - Oracle Enterprise Manager Database Express (EM Express)
 - Oracle Enterprise Manager Cloud Control
 - Oracle Management Cloud Database Express (OMX)
 - Others such as Listener Control, Oracle Net Configuration Assistant, Oracle Net Manager, ADR Command Interpreter, SQL*Loader, Oracle Data Pump Import, and Oracle Data Pump Export

Database Tool Choices

Topic	SQL*Plus	SQL Developer	SQLcl	DBCA	EM Database Express	EM Cloud Control	Oracle Universal Installer
Create a CDB or PDB	Yes	Yes (PDB only)	Yes	Yes	Yes (PDB only)	Yes (PDB only)	Yes
Explore CDB instance, architecture, and PDBs	Yes	Yes	Yes	No	Yes	Yes	No

SQL*Plus

- Example 1: From a command line, you can start SQL*Plus, log in, and show the user that you're logged in as:

```
$ sqlplus / as sysdba  
...  
SQL> show user  
USER is "SYS"
```

- Example 2: Call a SQL script from the command line:

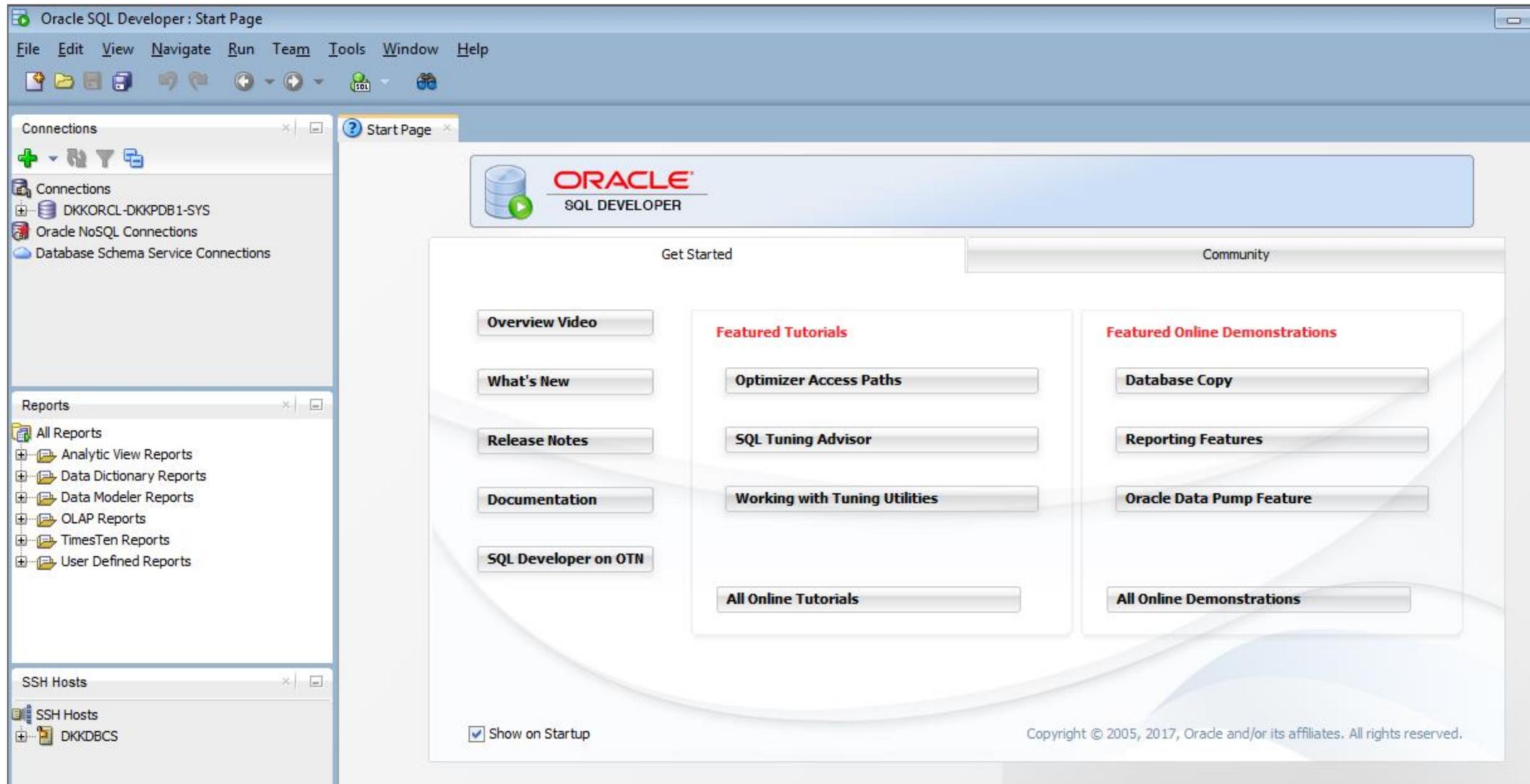
```
$ sqlplus hr/hr@HRPDB @script.sql
```

Username and password

Connect identifier

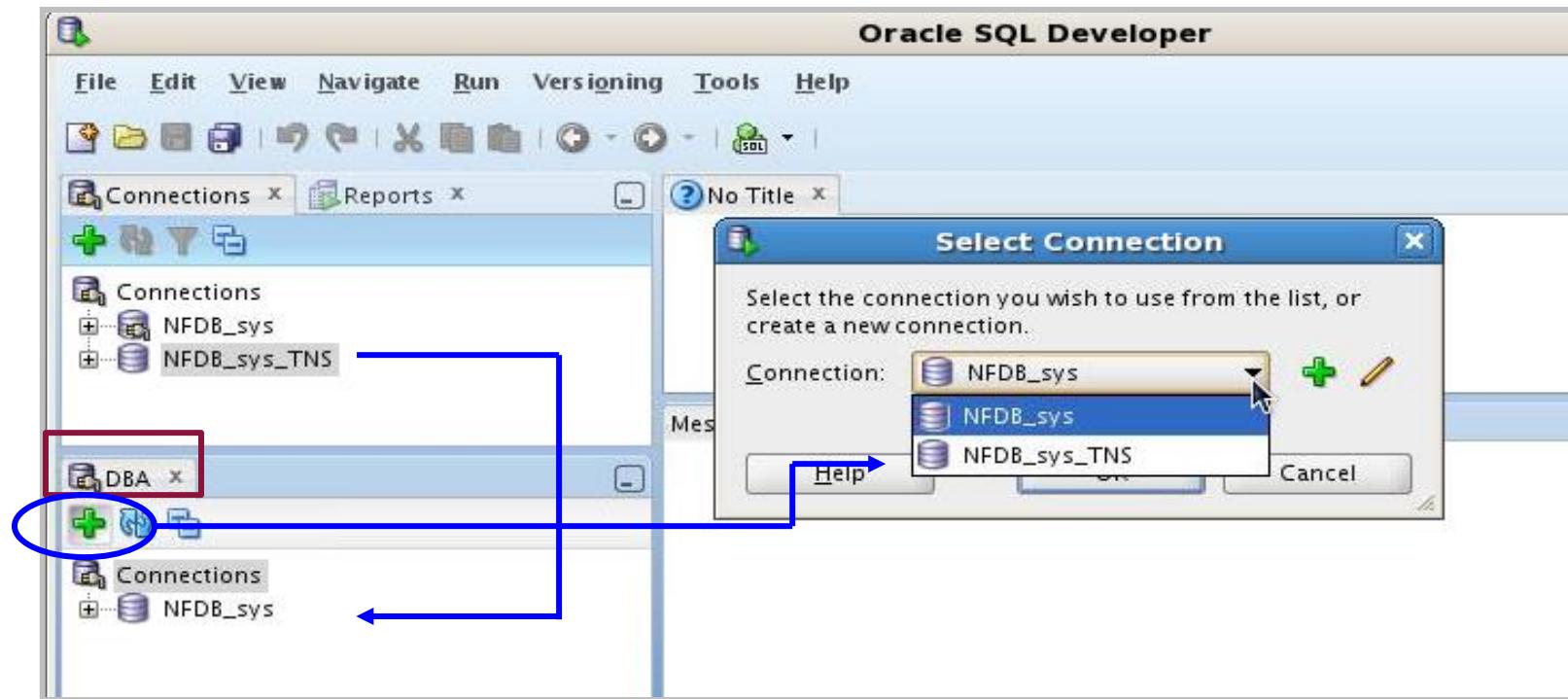
File name

Oracle SQL Developer



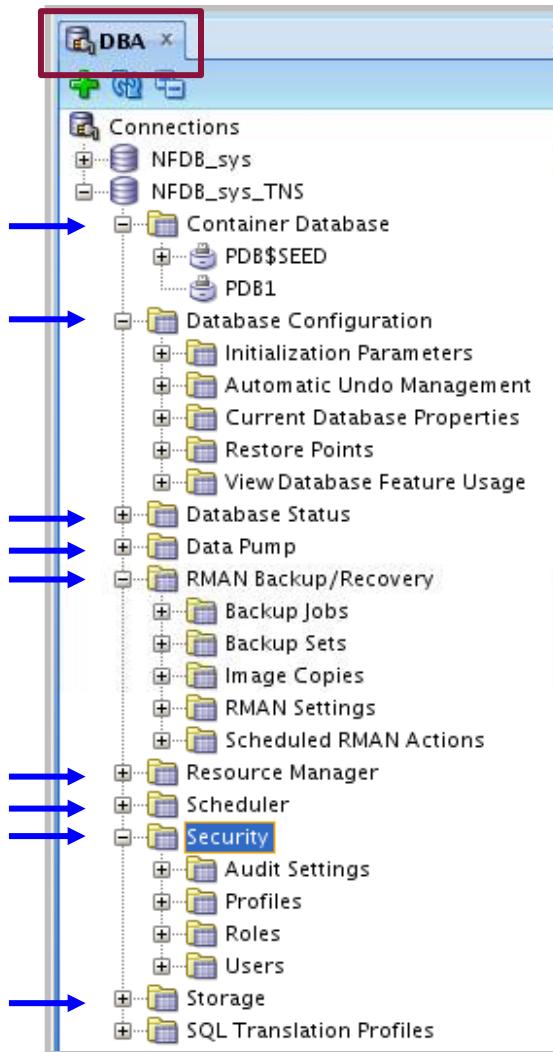
Oracle SQL Developer: Connections

Perform DBA operations in the DBA navigator by using DBA connections:

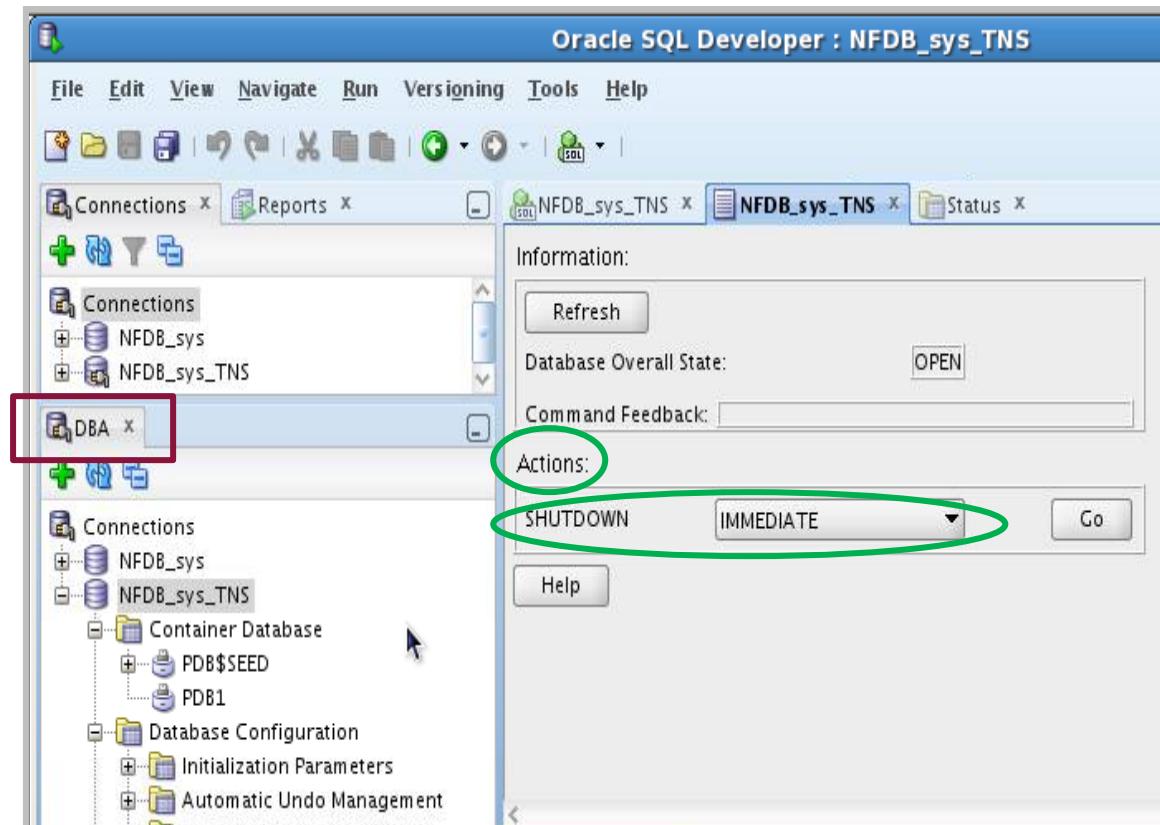


Oracle SQL Developer: DBA Actions

Using DBA features through DBA navigator



Performing DBA actions

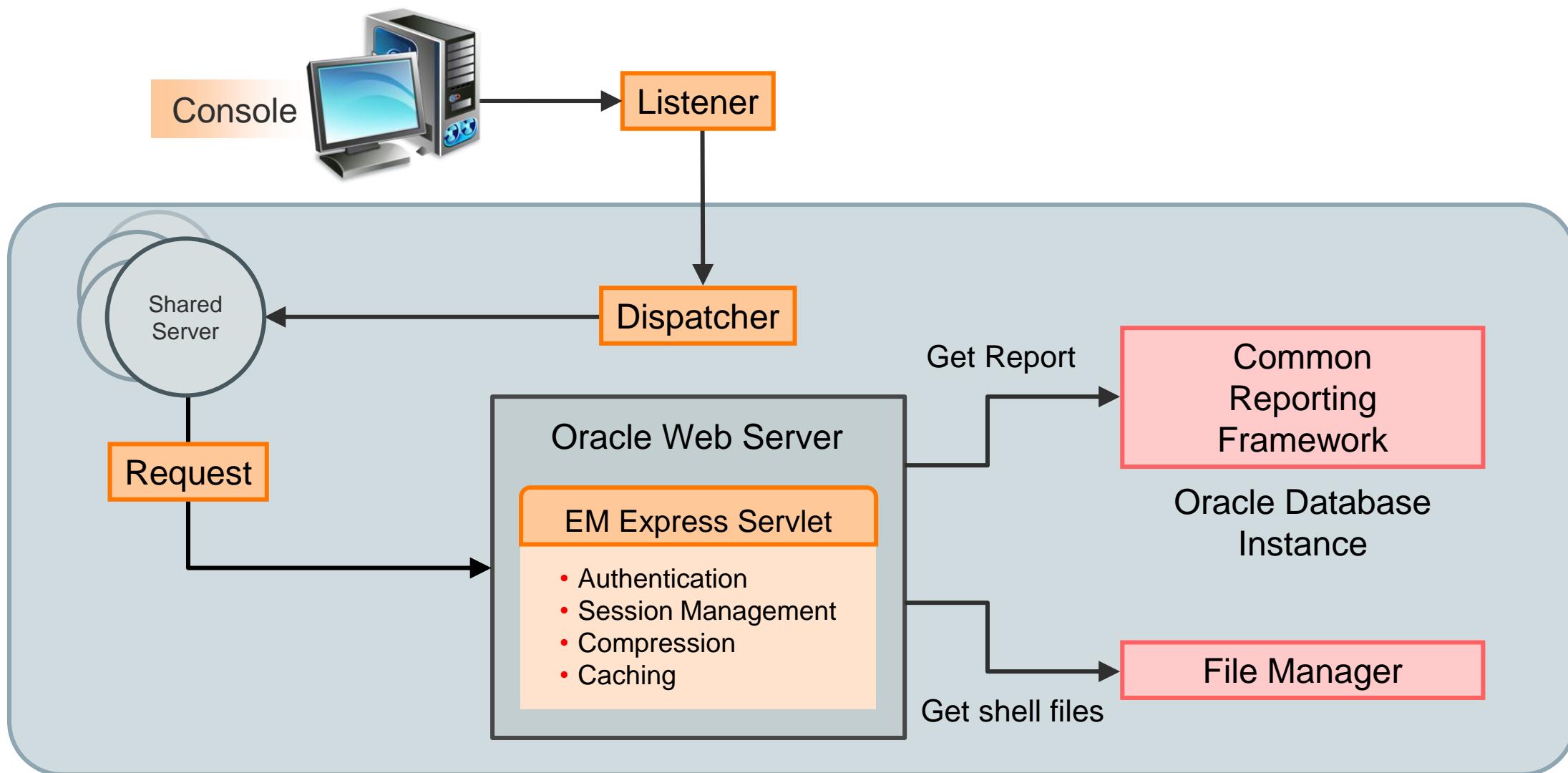


Database Configuration Assistant (DBCA)

- DBCA is a tool for creating and configuring an Oracle database.
- DBCA has two modes:
 - Interactive: Provides a graphical interface and guided workflow
 - Noninteractive/silent: Uses command-line arguments, a response file, or both
- DBCA can be launched by the Oracle Universal Installer or invoked after the software is installed.



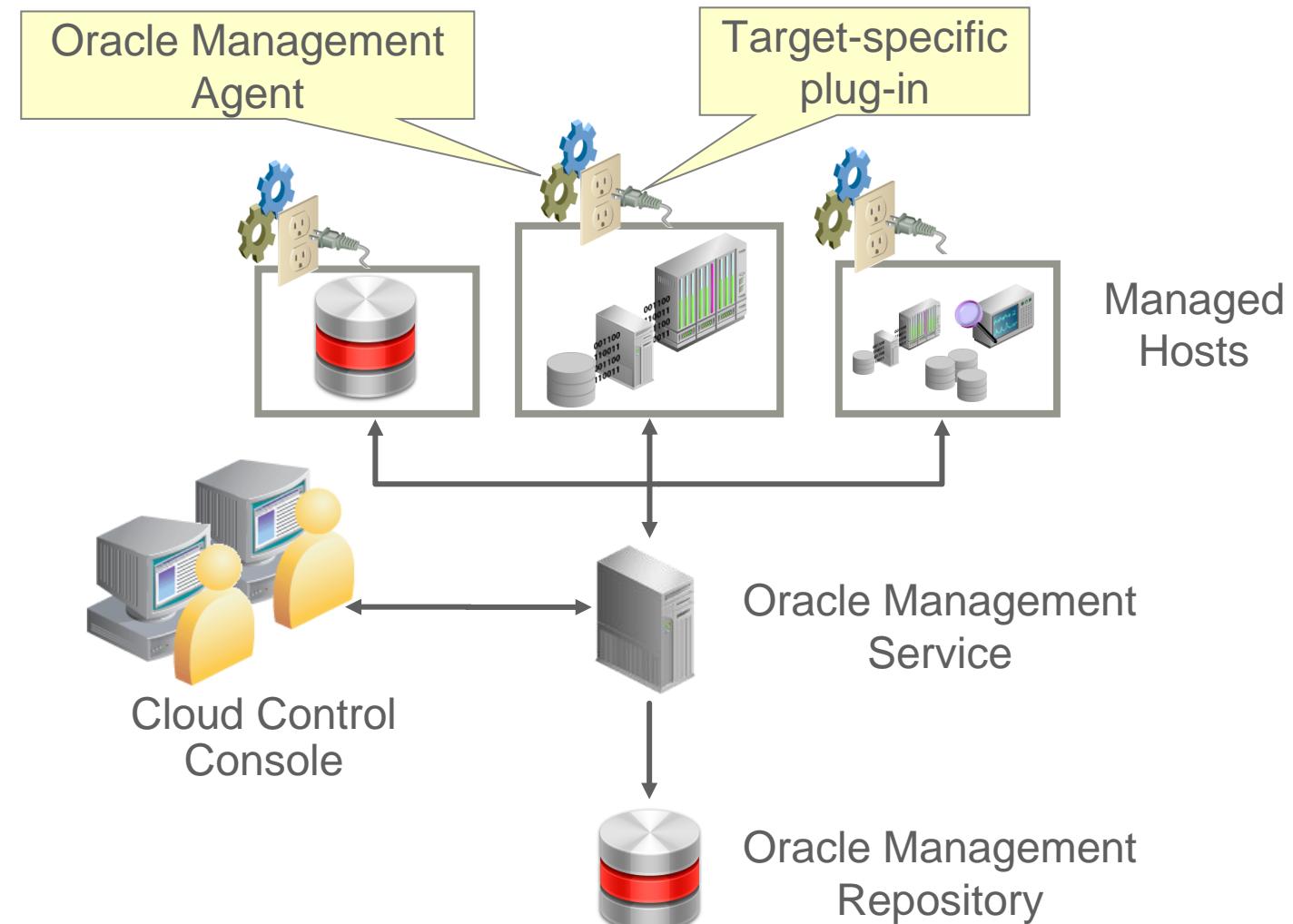
Oracle Enterprise Manager Database Express



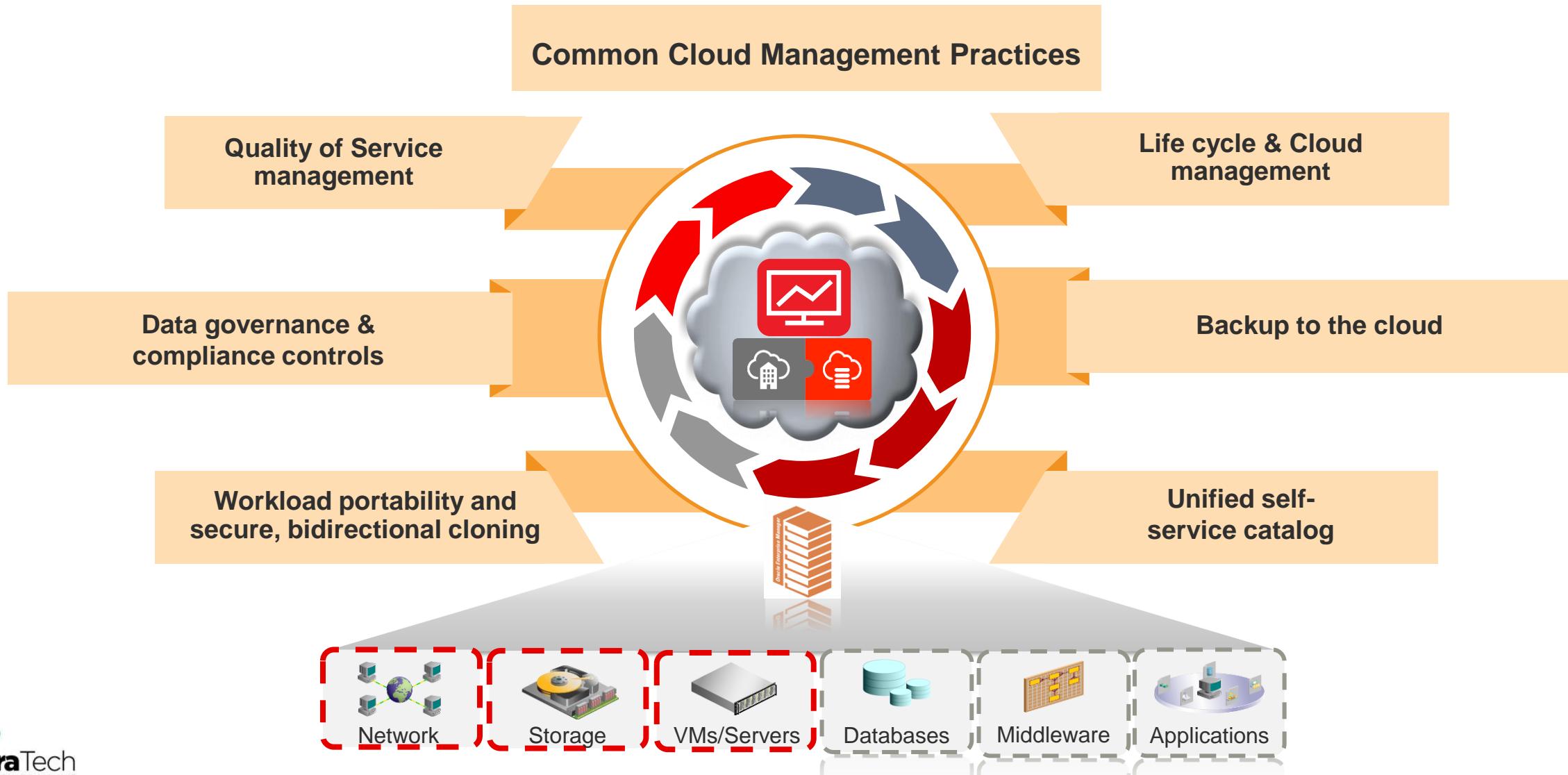
Enterprise Manager Cloud Control 13c Features



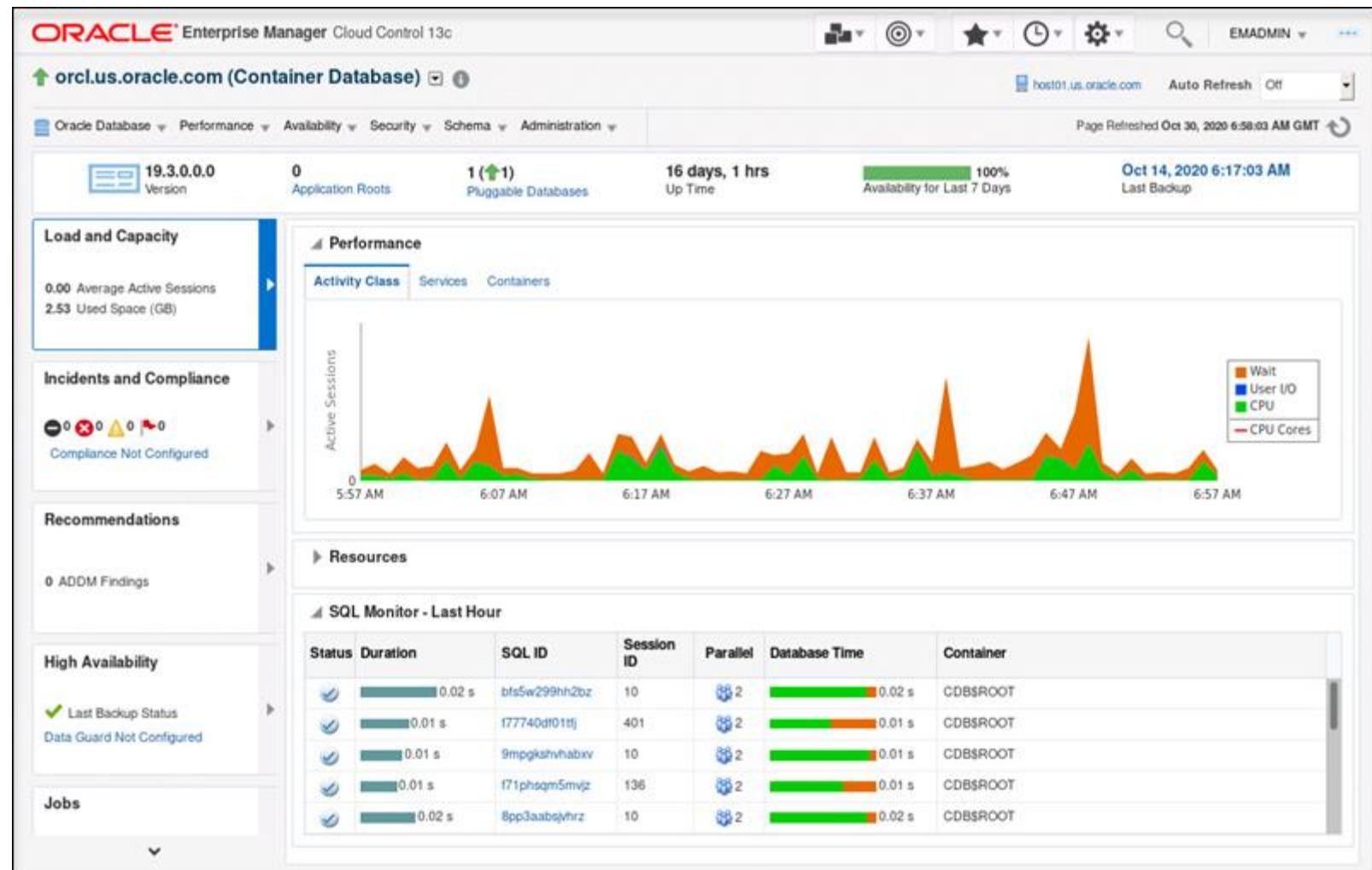
Oracle Enterprise Manager Component Overview



Single Pane of Glass for Enterprise Management



Oracle Enterprise Manager Database Management



Summary

In this lesson, you should have learned how to:

- Connect to an Oracle Database
- Describe the tools used to access an Oracle Database

3. Creating an Oracle Database by Using DBCA

Objectives

After completing this lesson, you should be able to:

- Create a database by using the Database Configuration Assistant (DBCA)
- Generate database creation scripts by using DBCA

Planning the Database

As a DBA, you must plan:

- The logical storage structure of the database and its physical implementation:
 - 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

Choosing a Database Template

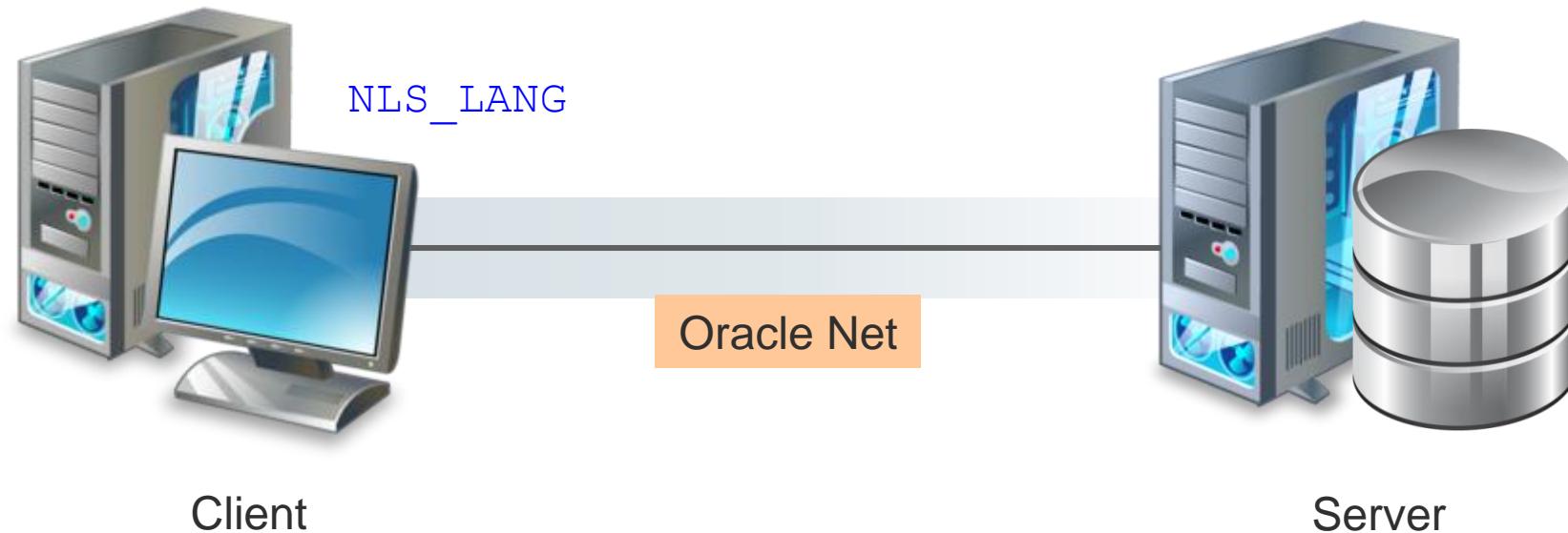
- General purpose or transaction processing:
 - Online transaction processing (OLTP) system, for example, a retail billing system for a software house or a nursery
- Custom:
 - Multipurpose database (perhaps combined OLTP and data warehouse functionality)
- Data warehouse:
 - Research and marketing data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

Choosing the Appropriate Character Set

- 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.
- The Oracle database supports different classes of character-encoding schemes:
 - Single-byte character sets
 - 7-bit
 - 8-bit
 - Multibyte character sets, including Unicode
- In general, Unicode is recommended because it is the most flexible character set.

How Are Character Sets Used?

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



Setting NLS_LANG Correctly on the Client

- No conversion occurs, because it does not seem to be required.
- Invalid data is entered into the database.



Using the Database Configuration Assistant

Task	Description
Create a database	Create and configure a new database
Create and manage database design templates	Create an XML file containing information required to create a new database
Delete a database	Shut down the database instance and delete all database files
Manage pluggable databases in a multitenant architecture	Create, delete, and unplug PDBs
Change the configuration of a database or PDB	Configure options and security settings

Using DBCA in Silent Mode

- Create a new database named ORCL using the General Purpose template

```
$ORACLE_HOME/bin/dbca -silent -createDatabase  
-templateName General_Purpose.dbc -gdbname ORCL  
-sid ORCL -createAsContainerDatabase true  
-numberOfPDBs 1 -pdbName pdb1 -useLocalUndoForPDBs true  
-responseFile NO_VALUE -characterSet AL32UTF8  
-totalMemory 1800 -sysPassword Welcome_1  
-systemPassword Welcome_1 -pdbAdminPassword Welcome_1  
-emConfiguration DBEXPRESS -dbsnmpPassword Welcome_1  
-emExpressPort 5500 -enableArchive true  
-recoveryAreaDestination /u03/app/oracle/fast_recovery_area  
-recoveryAreaSize 15000 -datafileDestination /u02/app/oracle/oradata
```

- Delete a database named ORCL

```
$ORACLE_HOME/bin/dbca -silent -deleteDatabase -sourceDB ORCL -sid ORCL  
-sysPassword Welcome_1
```

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

Practice Overview

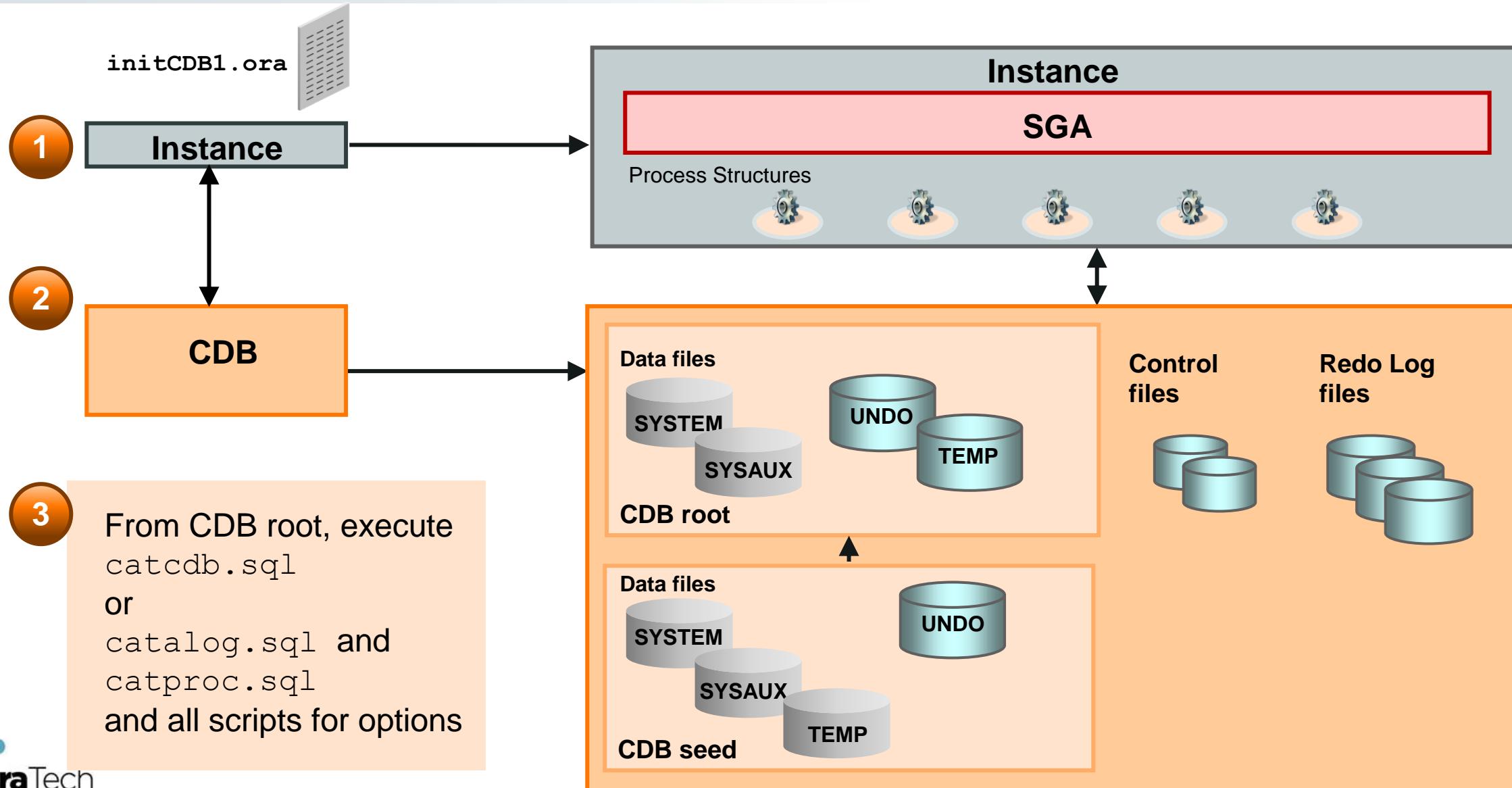
- Creating a New CDB

4. Creating an Oracle Database by Using a SQL Command

Objectives

After completing this lesson, you should be able to create a container database (CDB) by using the CREATE DATABASE command.

Creating a Container Database (CDB)



Creating a CDB by Using a SQL Command: Example

1. Start up the instance :

a. Set ORACLE_SID=CDB1.

b. Create the initCDB1.ora file and set parameters:

- CONTROL_FILES to CDB control file names
- DB_NAME to a CDB name
- ENABLE_PLUGGABLE_DATABASE to TRUE

```
SQL> CONNECT / AS SYSDBA  
SQL> STARTUP NOMOUNT
```

1. Create the database:

```
SQL> CREATE DATABASE cdb1 ENABLE PLUGGABLE DATABASE ...  
      SEED FILE_NAME_CONVERT = ('/oracle/dbs','/oracle/seed');
```

CDB\$ROOT + PDB\$SEED created

1. Execute the \$ORACLE_HOME/rdbms/admin/catcdb.sql SQL script.

Using the SEED FILE_NAME_CONVERT Clause

```
SQL> CREATE DATABASE cdb1
  USER SYS IDENTIFIED BY p1 USER SYSTEM IDENTIFIED BY p2
  LOGFILE GROUP 1 ('/u01/app/oradata/CDB1/red01a.log',
                    '/u02/app/oradata/CDB1/red01b.log') SIZE 100M,
  GROUP 2 ('/u01/app/oradata/CDB1/red02a.log',
                    '/u02/app/oradata/CDB1/red02b.log') SIZE 100M
  CHARACTER SET AL32UTF8 NATIONAL CHARACTER SET AL16UTF16
  EXTENT MANAGEMENT LOCAL DATAFILE
    '/u01/app/oradata/CDB1/system01.dbf' SIZE 325M
  SYSAUX DATAFILE '/u01/app/oradata/CDB1/sysaux01.dbf' SIZE 325M
  DEFAULT TEMPORARY TABLESPACE tempts1
    TEMPFILE '/u01/app/oradata/CDB1/temp01.dbf' SIZE 20M
  UNDO TABLESPACE undotbs
    DATAFILE '/u01/app/oradata/CDB1/undotbs01.dbf' SIZE 200M
  ENABLE PLUGGABLE DATABASE
  SEED FILE_NAME_CONVERT =('/u01/app/oradata/CDB1','/u01/app/oradata/CDB1/seed');
```

Using the ENABLE PLUGGABLE DATABASE Clause

Without **SEED FILE_NAME_CONVERT**:

- OMF: **DB_CREATE_FILE_DEST** = '/u02/app/oradata'

```
SQL> CONNECT / AS SYSDBA
SQL> STARTUP NOMOUNT
SQL> CREATE DATABASE cdb2
  USER SYS IDENTIFIED BY p1 USER SYSTEM IDENTIFIED BY p2
  EXTENT MANAGEMENT LOCAL
  DEFAULT TEMPORARY TABLESPACE temp
  UNDO TABLESPACE undotbs
  DEFAULT TABLESPACE users
ENABLE PLUGGABLE DATABASE;
```

- Or initialization parameter: **PDB_FILE_NAME_CONVERT** =
'/u02/app/oradata/CDB2','/u02/app/oradata/seed'

Summary

In this lesson, you should have learned how to create a CDB by using the CREATE DATABASE command.

Practice Overview

- Creating a New CDB

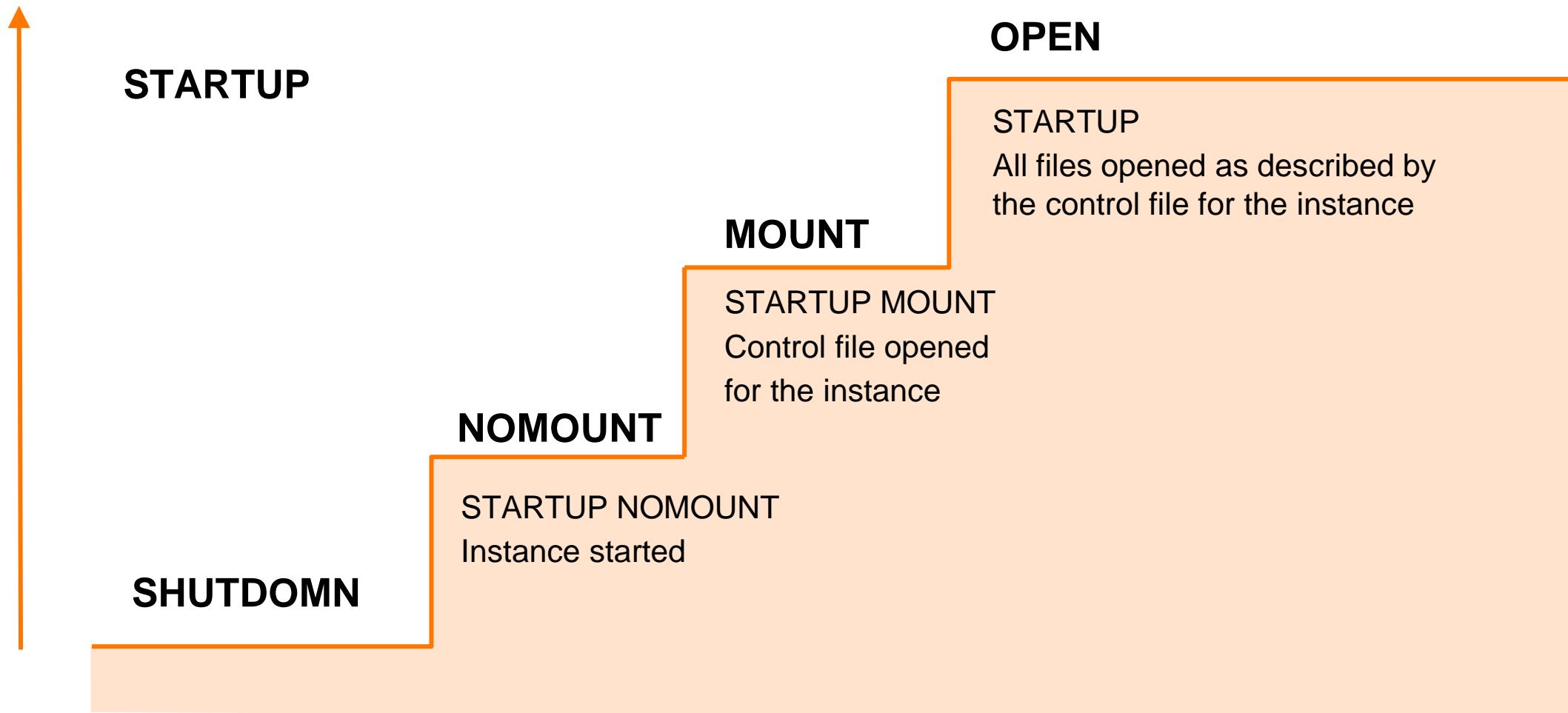
5. Starting Up and Shutting Down a Database Instance

Objectives

After completing this lesson, you should be able to:

- Start up and shut down Oracle databases
- Open and close PDBs

Starting the Oracle Database Instance



Shutting Down an Oracle Database Instance

- Sometimes you need to shut down the database instance (for example, to change a static parameter or patch the database server).
- Use the `SHUTDOWN` command to shut down the database instance in various modes: `ABORT`, `IMMEDIATE`, `NORMAL`, and `TRANSACTIONAL`.

	ABORT	IMMEDIATE	NORMAL	TRANSACTIONAL
Allows new connections	No	No	No	No
Waits until current sessions end	No	No	Yes	No
Waits until current transactions end	No	No	Yes	Yes
Forces a checkpoint and closes files	No	Yes	Yes	Yes

Comparing SHUTDOWN Modes

- On the way down:
- Uncommitted changes rolled back, for IMMEDIATE
 - Database buffer cache written to data files
 - Resources released

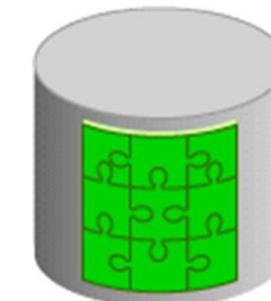
During:

SHUTDOWN NORMAL
or
SHUTDOWN TRANSACTIONAL
or
SHUTDOWN IMMEDIATE

On the way up:

- No instance recovery

Consistent database



Comparing SHUTDOWN Modes

On the way down:

- Modified buffers not written to data files
- Uncommitted changes not rolled back



During:

SHUTDOWN ABORT
or
Instance failure
or
STARTUP FORCE

On the way up:

- Online redo log files used to reapply changes
- Undo segments used to roll back uncommitted changes
- Resources released

Inconsistent database

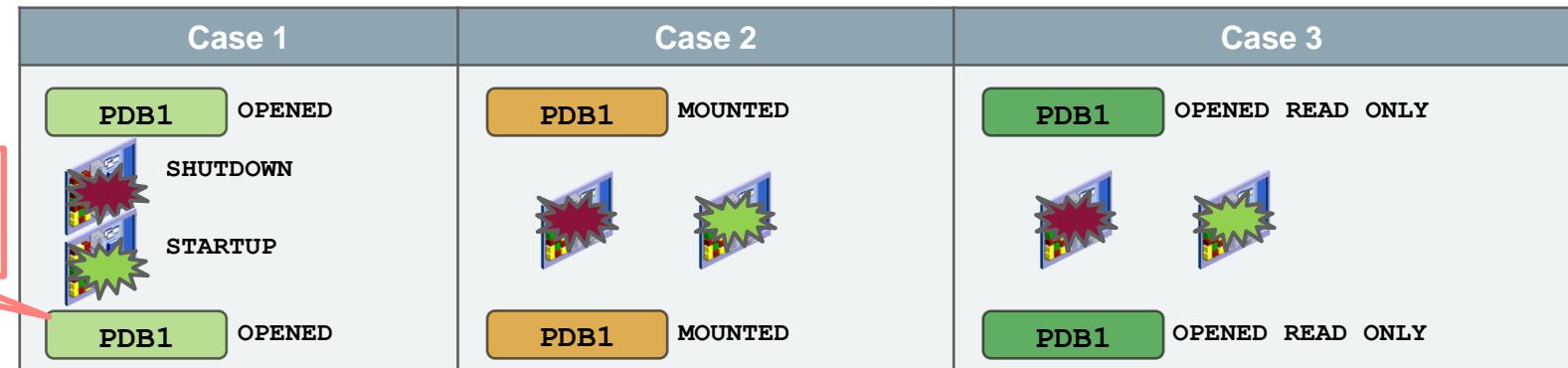
Opening and Closing PDBs

- Open/close a PDB to open/close its data files.
- A PDB has four open modes:
 - READ WRITE (the PDB is fully started/opened)
 - READ ONLY
 - MIGRATE
 - MOUNTED (the PDB is shut down/closed)
- Use the ALTER PLUGGABLE DATABASE command or STARTUP and SHUTDOWN commands to open and close PDBs.
 - Example: SQL> ALTER PLUGGABLE DATABASE PDB1 OPEN;
- The ALTER PLUGGABLE DATABASE command lets you change from any open mode to another.
- To use the STARTUP command, the PDB must be in MOUNTED mode.

Configuring PDBs to Automatically Open

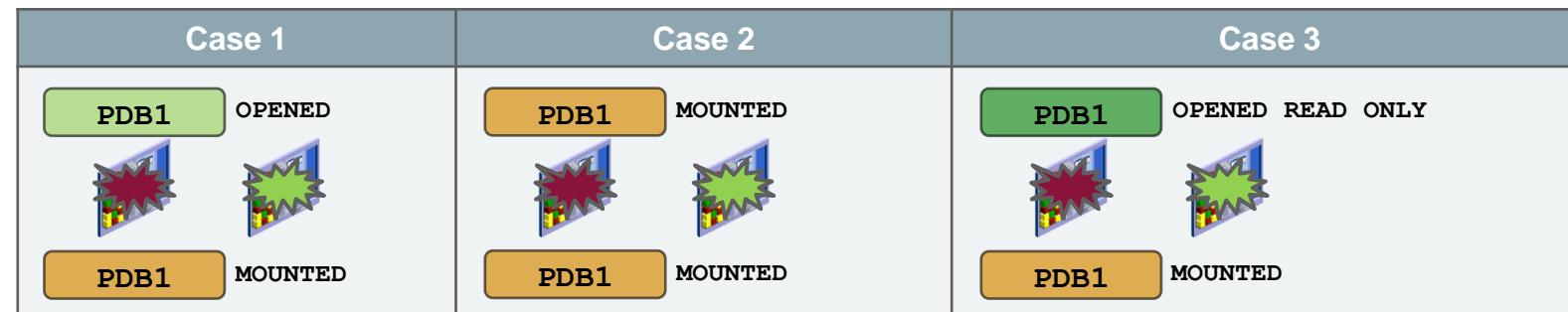
- Automatically keep the PDB's state after CDB STARTUP:

```
SQL> ALTER PLUGGABLE DATABASE pdb1 SAVE STATE;
```



- Automatically discard the PDB's state after CDB STARTUP:

```
SQL> ALTER PLUGGABLE DATABASE pdb1 DISCARD STATE;
```



Summary

In this lesson, you should have learned how to:

- Start up and shut down Oracle databases
- Open and close PDBs

Practice Overview

- Shutting Down and Starting Up the Oracle Database Instance

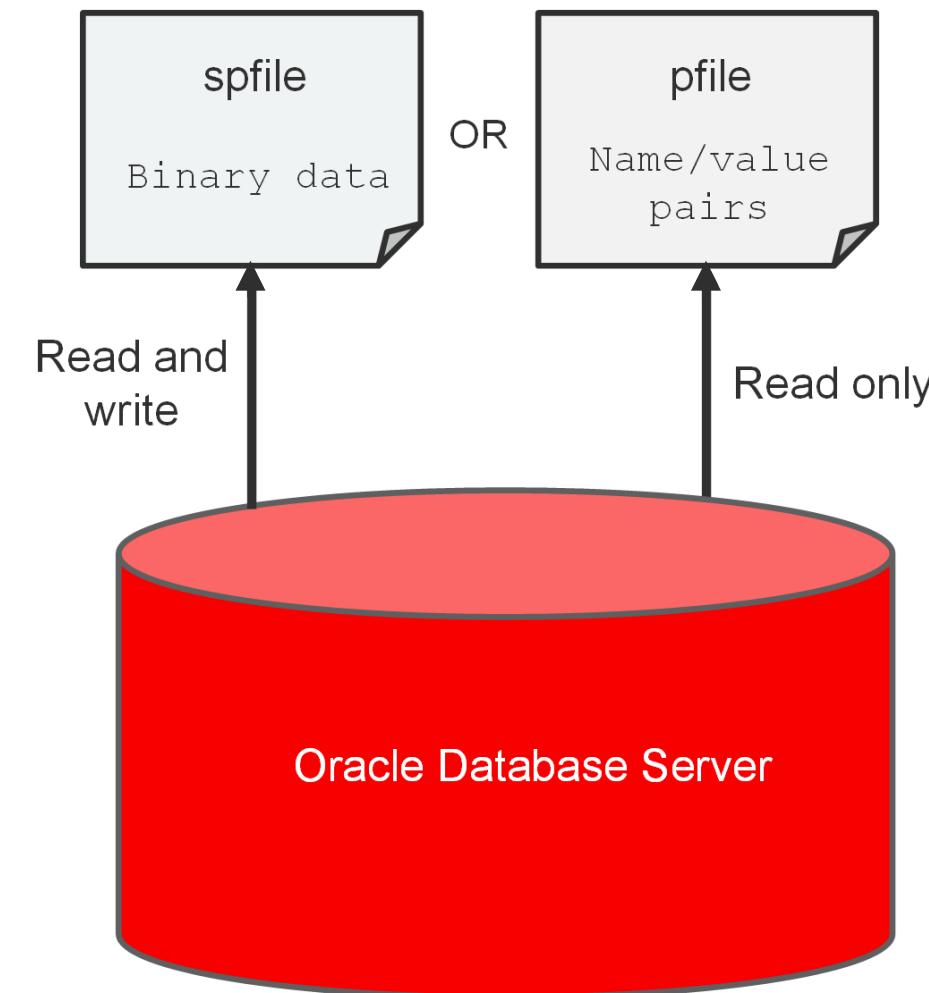
6. Managing Database Instances

Objectives

After completing this lesson, you should be able to:

- Describe initialization parameter files and initialization parameters
- View and modify initialization parameters in SQL*Plus
- Work with the Automatic Diagnostic Repository (ADR)
- Query dynamic performance views

Working with Initialization Parameters



Initialization Parameters

- Initialization parameters (parameters):
 - Set database limits
 - Set database-wide defaults
 - Specify files and directories
 - Affect performance
- Parameters can be of two types: basic or advanced.
 - Tune around 30 basic parameters to get reasonable database performance.
 - Example of a basic parameter: SGA_TARGET
 - Example of an advanced parameter: DB_CACHE_SIZE
- Derived parameters calculate their values from the values of other parameters.
 - Example: SESSIONS is derived from PROCESSES.
- Some parameter values or value ranges depend on the host operating system.
 - Example: DB_BLOCK_SIZE

Modifying Initialization Parameters

- Modify parameters to set capacity limits or improve performance.
 - Use Enterprise Manager or SQL*Plus (`ALTER SESSION` or `ALTER SYSTEM`).
- Query `V$PARAMETER` for an initialization parameter to learn whether you can make:
 - Session-level changes (`ISSES_MODIFIABLE` column)
 - System-level changes (`ISSYS_MODIFIABLE` column)
 - PDB-level changes (`ISPDB_MODIFIABLE` column)
- Use the `SCOPE` clause with the `ALTER SYSTEM` command to tell the system where to update the system-level parameter:
 - MEMORY
 - SPFILE
 - BOTH
- Use the `DEFERRED` keyword to set or modify the value of the parameter for future sessions that connect to the database.

Viewing Initialization Parameters

Ways to view initialization parameters in SQL*Plus:

- Issue the SHOW PARAMETER command.
 - Example: Find out about all the parameters whose names contain the word “para.”

```
SQL> SHOW PARAMETER para
```

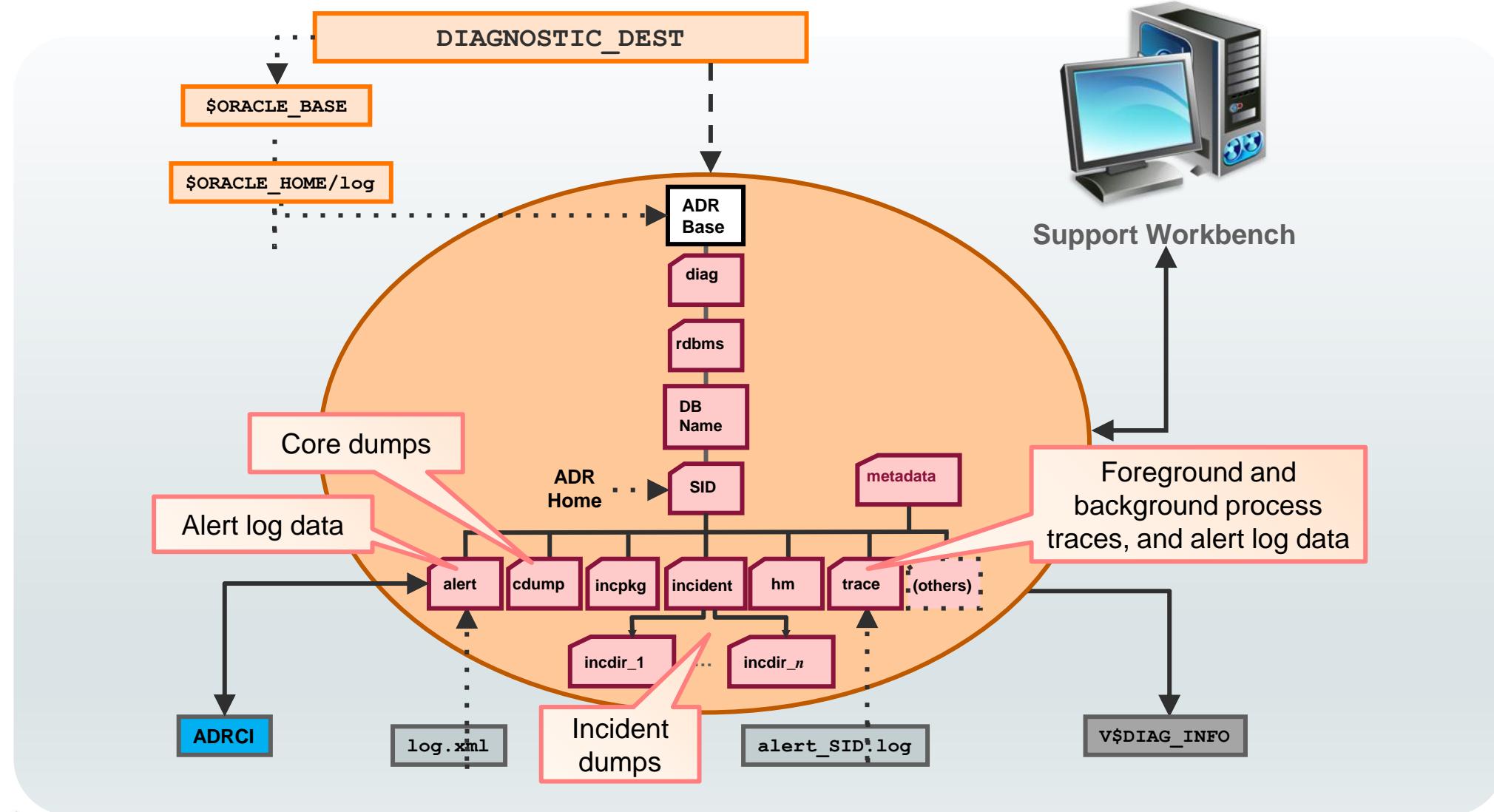
- Query the following views:
 - V\$PARAMETER
 - V\$PARAMETER2
 - V\$SPPARAMETER
 - V\$SYSTEM_PARAMETER
 - V\$SYSTEM_PARAMETER2

Working with the Automatic Diagnostic Repository

The Automatic Diagnostic Repository (ADR):

- Is a file-based repository outside the database
- Is a system-wide central tracing and logging repository
- Stores database diagnostic data such as:
 - Traces
 - Alert log
 - Health monitor reports

Automatic Diagnostic Repository



Viewing the Alert Log

- The alert log file is a chronological log of messages about the database instance and database, such as:
 - Any nondefault initialization parameters used at startup
 - All internal errors (ORA-600), block corruption errors (ORA-1578), and deadlock errors (ORA-60) that occurred
 - Administrative operations, such as the SQL statements CREATE, ALTER, DROP DATABASE, and TABLESPACE, and the Enterprise Manager or SQL*Plus statements STARTUP, SHUTDOWN, ARCHIVE LOG, and RECOVER
 - Several messages and errors relating to the functions of shared server and dispatcher processes
 - Errors during the automatic refresh of a materialized view
- Query V\$DIAG_INFO to find the location of the alert log.
 - The path to alert_SID.log corresponds to the Diag Trace entry.
 - The path to log.xml corresponds to the Diag Alert entry.
- You can view the alert log in a text editor or in ADRCI.

Using Trace Files

- Trace files contain:
 - Error information (contact Oracle Support Services if an internal error occurs)
 - Information that can provide guidance for tuning applications or an instance
- Each server and background process can write to an associated trace file.
- Trace file names for background processes are named after their processes.
 - Exception: Trace files generated by job queue processes
- Oracle Database includes an advanced fault diagnosability infrastructure for preventing, detecting, diagnosing, and resolving problems.
- When a critical error occurs:
 - An incident number is assigned to the error
 - Diagnostic data for the error (such as trace files) is immediately captured and tagged with the incident number
 - Data is stored in the ADR
- ADR files can be automatically purged by setting retention policy parameters.

Administering the DDL Log File

- Enable the capture of certain DDL statements to a DDL log file by setting `ENABLE_DDL_LOGGING` to TRUE
- The DDL log contains one log record for each DDL statement.
- Two DDL logs contain the same information:
 - XML DDL log: `log.xml` written to
 `$ORACLE_BASE/diag/rdbms/<dbname>/<SID>/log/ddl`
 - Text DDL: `ddl_<sid>.log` written to
 `$ORACLE_BASE/diag/rdbms/<dbname>/<SID>/log`
- Example:

```
$ more ddl_orcl.log
Thu Nov 15 08:35:47 2016
diag_adl:drop user app_user
```

Querying Dynamic Performance Views

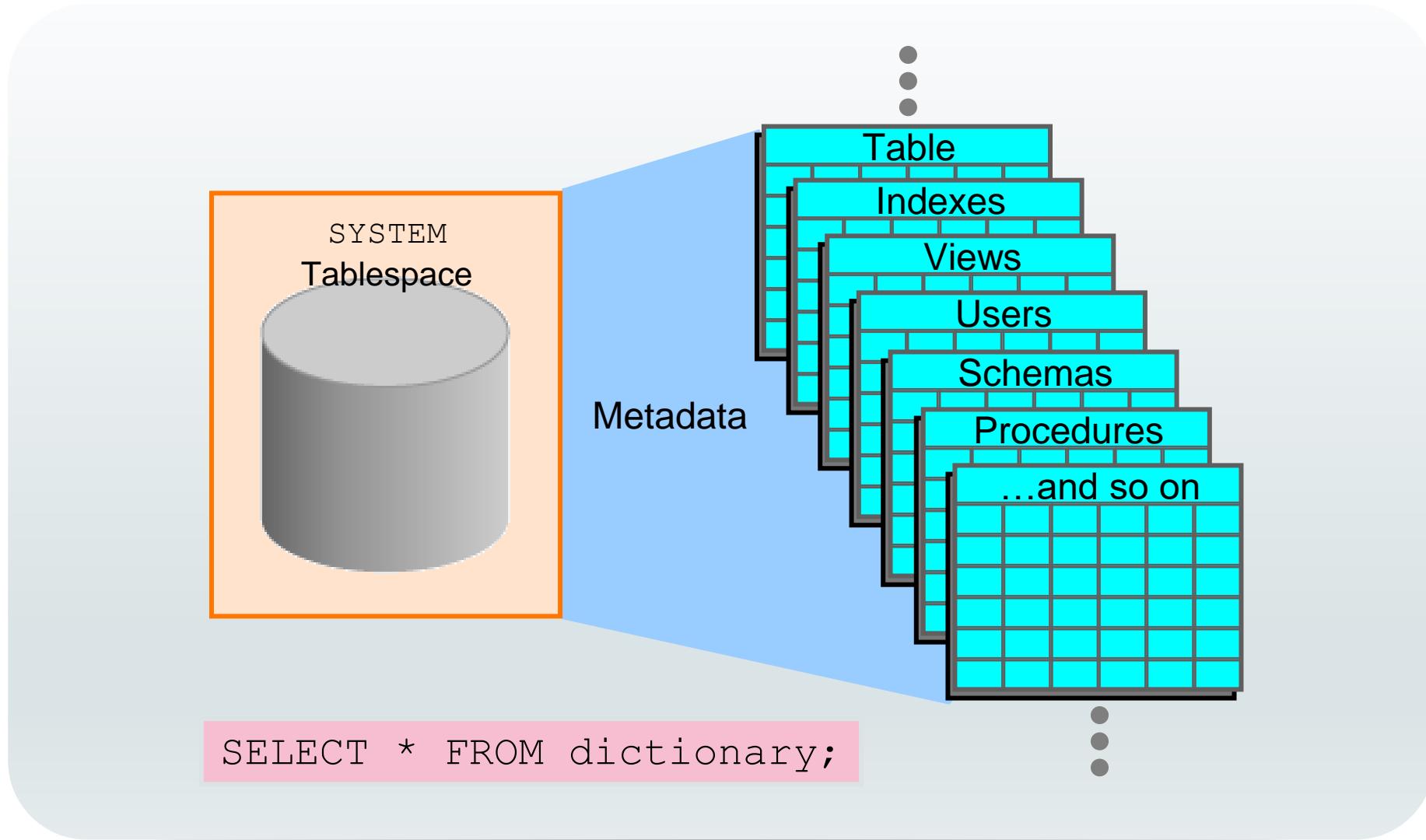
- Dynamic performance views provide access to information about the changing states of instance memory structures:
 - Sessions, file states, and locks
 - Progress of jobs and tasks
 - Backup status, memory usage, and allocation
 - System and session parameters
 - SQL execution
 - Statistics and metrics
- Dynamic performance views start with the prefix V\$.
- Example query: Which current sessions have logged in from the EDXX9P1 computer on the last day?

```
SQL> SELECT * FROM V$SESSION
  2 WHERE machine = 'EDXX9P1'
  3 AND logon_time > SYSDATE - 1;
```

Considerations for Dynamic Performance Views

- These views are owned by the SYS user.
- Views provide information depending on the stage (NOMOUNT, MOUNT, or OPEN).
- You can query V\$FIXED_TABLE to see all the view names.
- These views are often referred to as “v-dollar views.”
- Read consistency is not guaranteed on these views because the data is dynamic.

Data Dictionary: Overview



Querying the Oracle Data Dictionary

CDB_ All objects in the CDB across all PDBs

DBA_ All objects in a container or PDB

ALL_ Objects accessible by the current user

USER_ All objects owned by current user

Summary

In this lesson, you should have learned how to:

- Describe initialization parameter files and initialization parameters
- View and modify initialization parameters in SQL*Plus
- Work with the Automatic Diagnostic Repository (ADR)
- Query dynamic performance views

Practice Overview

- Investigating Initialization Parameter Files
- Viewing Initialization Parameters by Using SQL*Plus
- Modifying Initialization Parameters by Using SQL*Plus
- Viewing Diagnostic Information

7. Oracle Net Services: Overview

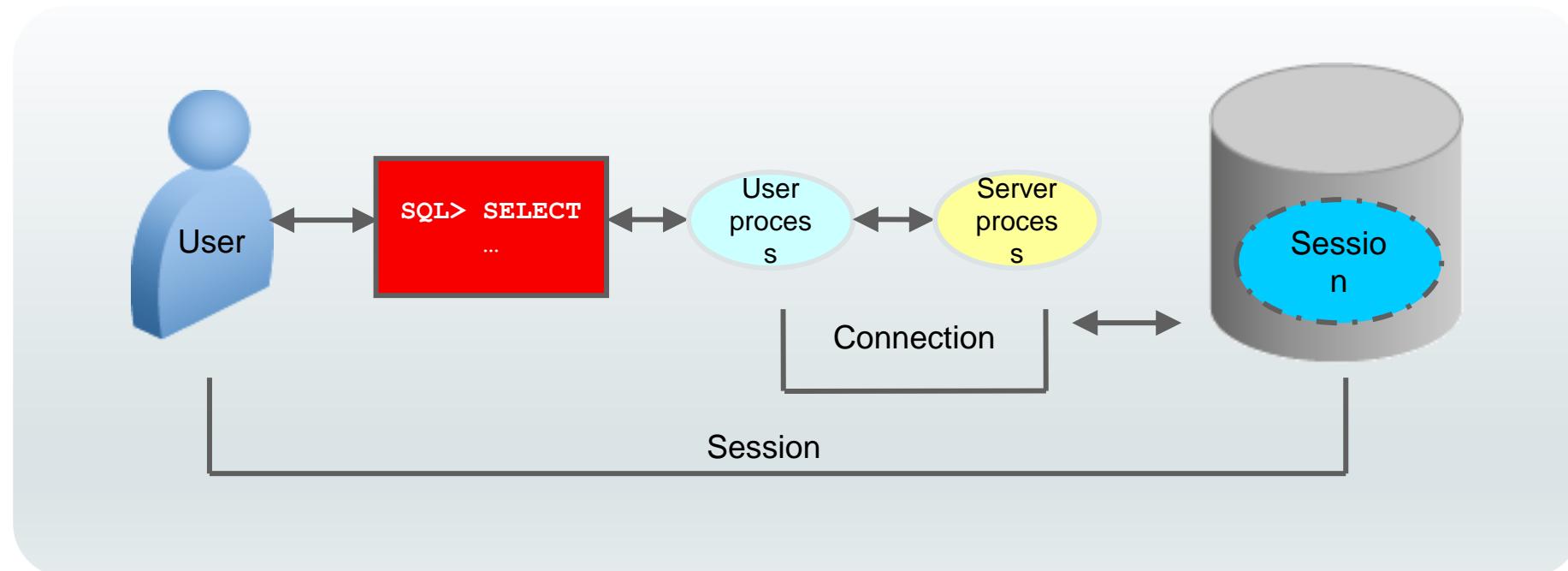
Objectives

After completing this lesson, you should be able to:

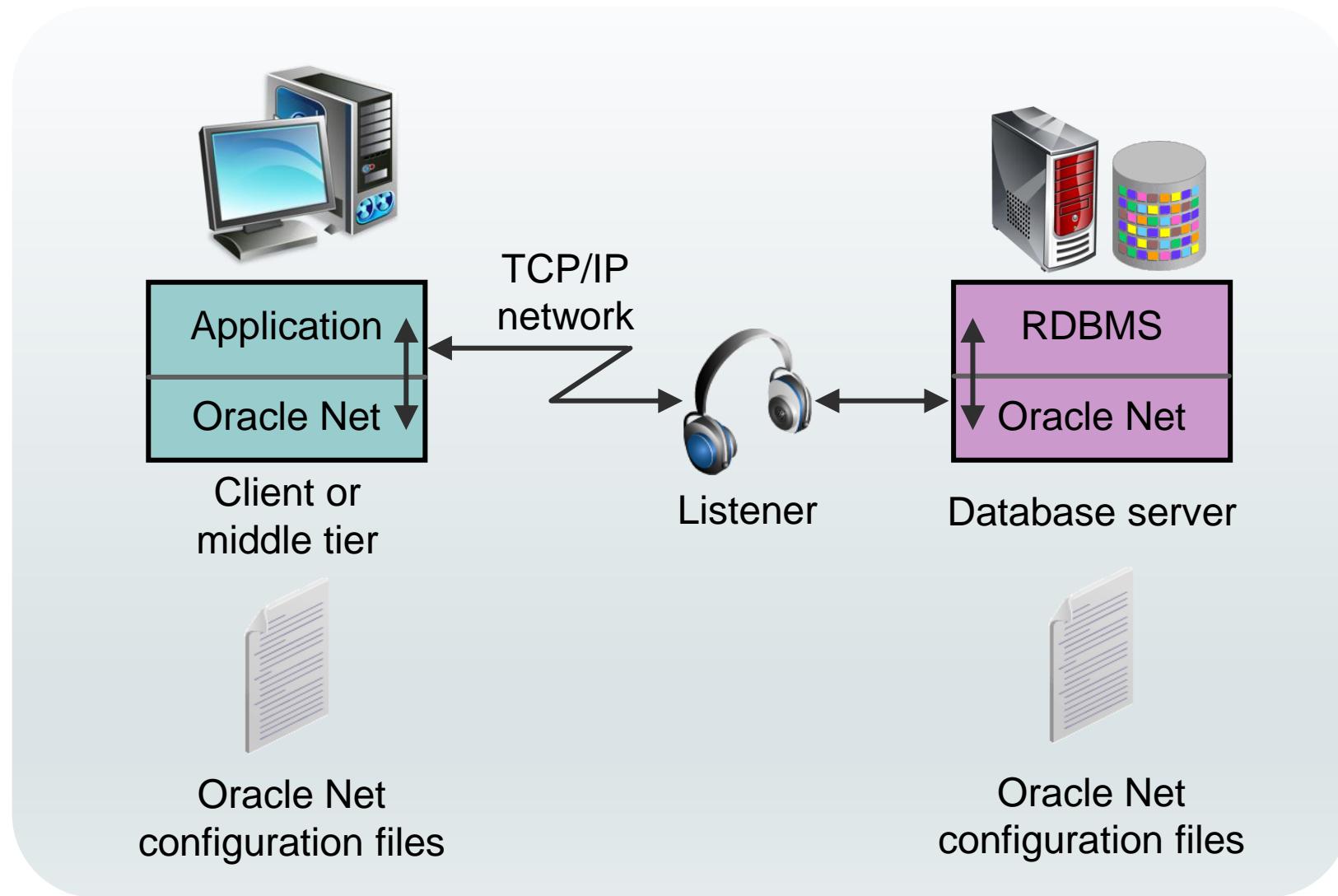
- List the components of Oracle Net Services
- Explain how listeners work
- Describe the tools that are used to administer Oracle Net Services
- Explain the difference between dedicated and shared server configurations

Connecting to the Database Instance

- **Connection:** Communication between a user process and an instance
- **Session:** Represents the state of a current user login to the database instance



Oracle Net Services: Overview



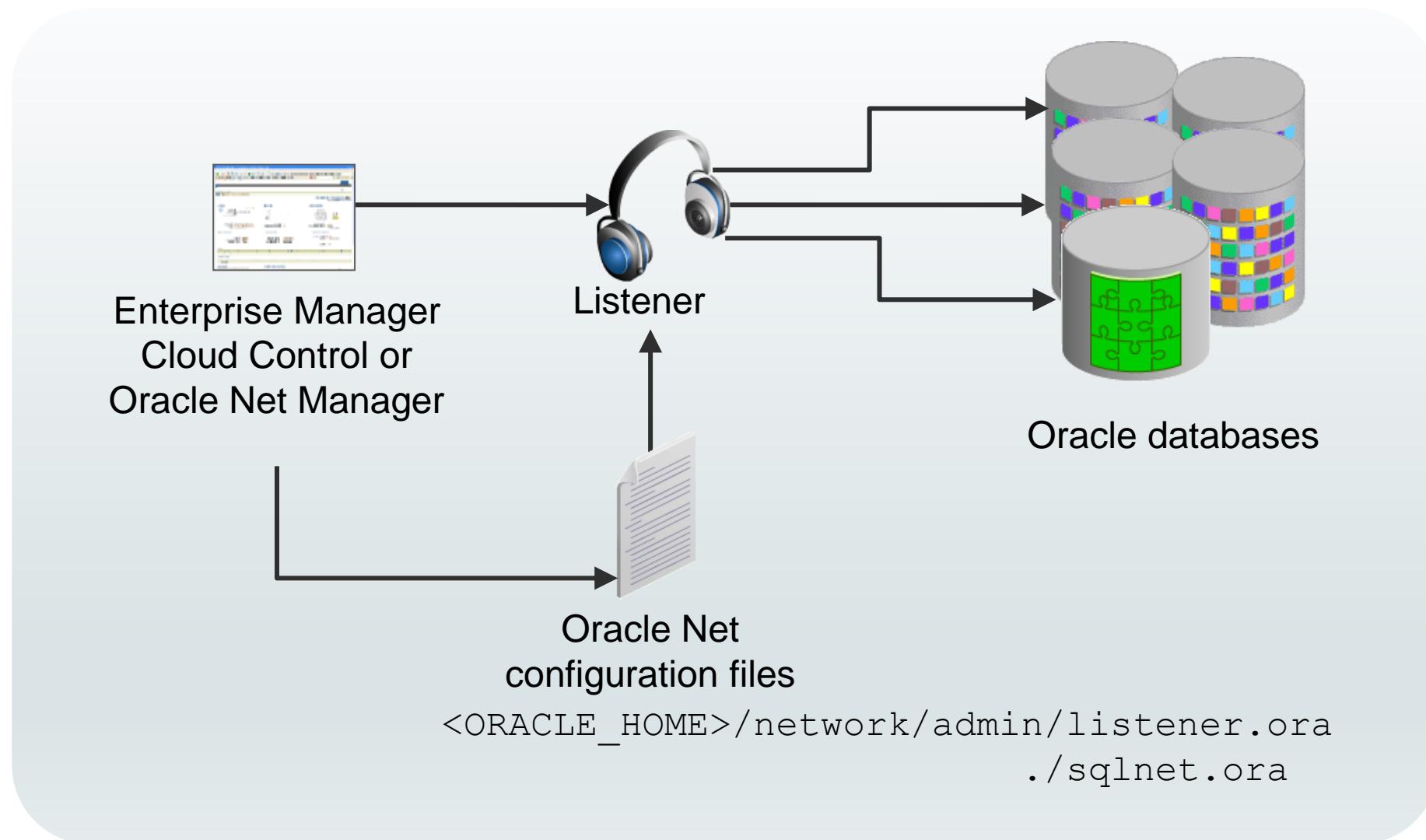
Defining Oracle Net Services Components

Component	Description	File
Listeners	A process that resides on the server whose responsibility is to listen for incoming client connection requests and manage traffic to the server	listener.ora
Naming methods	A resolution method used by a client application to resolve a connect identifier to a connect descriptor when attempting to connect to a database service	
Naming (net service name)	A simple name (connect identifier) for a service that resolves to a connect descriptor to identify the network location and identification of a service	tnsnames.ora (local configuration)
Profiles	A collection of parameters that specifies preferences for enabling and configuring Oracle Net features on the client or server	sqlnet.ora

Tools for Configuring and Managing Oracle Net Services

- Enterprise Manager Cloud Control
- Oracle Net Manager
- Oracle Net Configuration Assistant
- Listener Control Utility

Oracle Net Listener: Overview



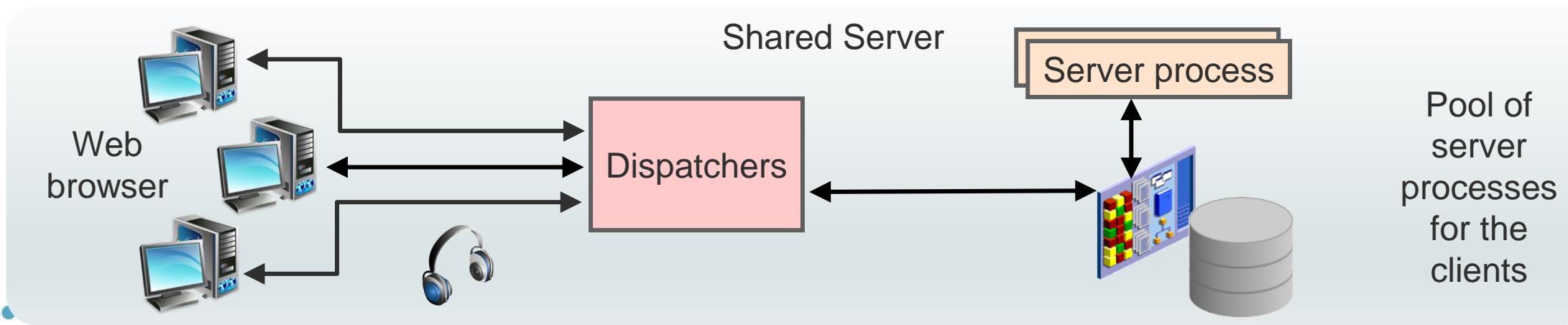
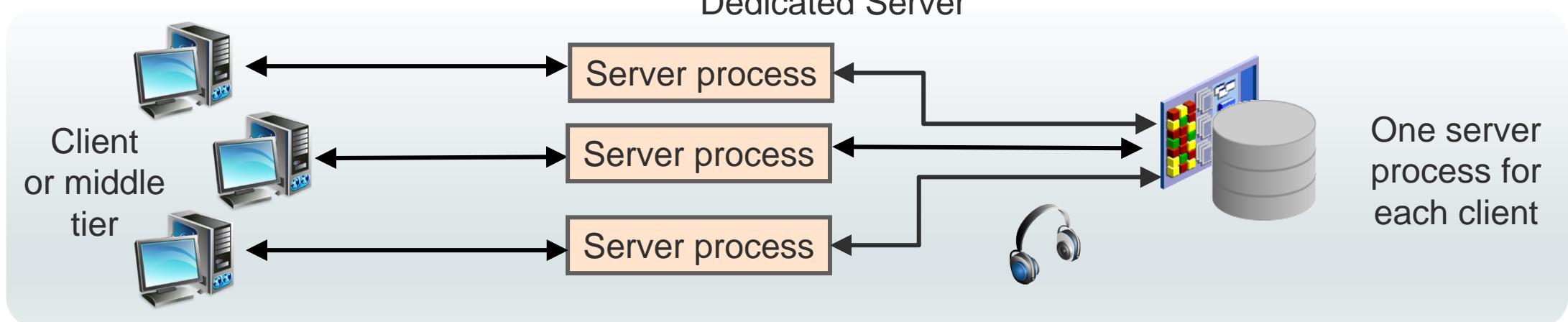
The Default Listener

- During the creation of an Oracle database, the Oracle Net Configuration Assistant utility creates a local listener named LISTENER.
- LISTENER is automatically populated with available database services through a feature called dynamic service registration.
- LISTENER listens on the following TCP/IP protocol address:

ADDRESS= (PROTOCOL=tcp) (HOST=host_name) (PORT=1521))

- Without any configuration, you can access your database instance immediately through LISTENER.
- If the listener name is LISTENER and it cannot be resolved, a protocol address of TCP/IP and a port number of 1521 is assumed.

Comparing Dedicated and Shared Server Architecture



Summary

In this lesson, you should have learned how to:

- List the components of Oracle Net Services
- Explain how listeners work
- Describe the tools that are used to administer Oracle Net Services
- Explain the difference between dedicated and shared server configurations

8. Configuring Naming Methods

Objectives

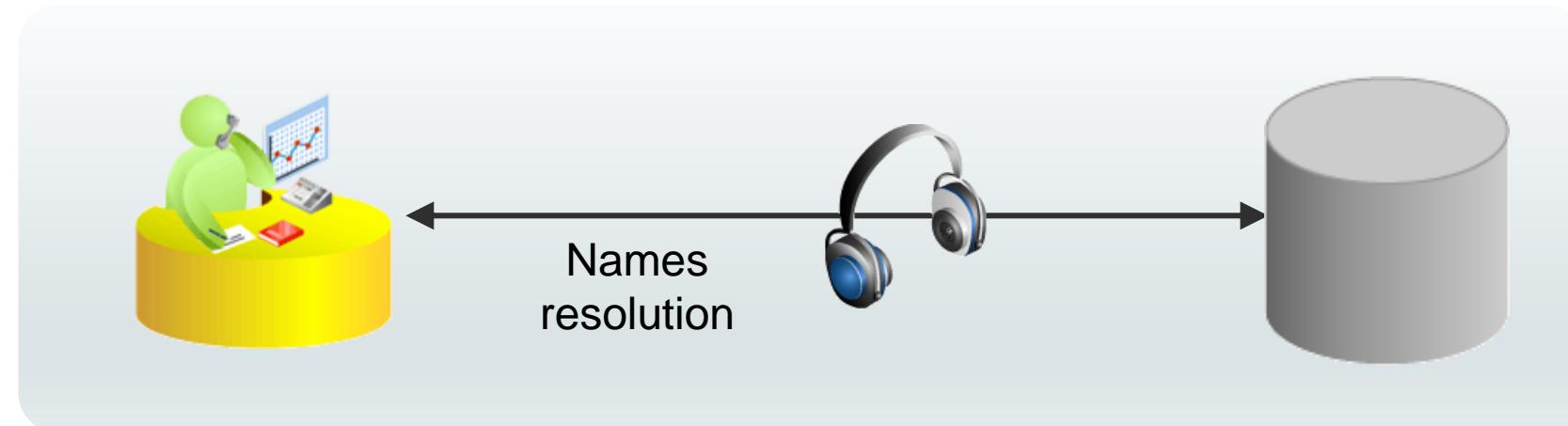
After completing this lesson, you should be able to:

- Describe Oracle Net Services naming methods
- Configure local naming for database connections

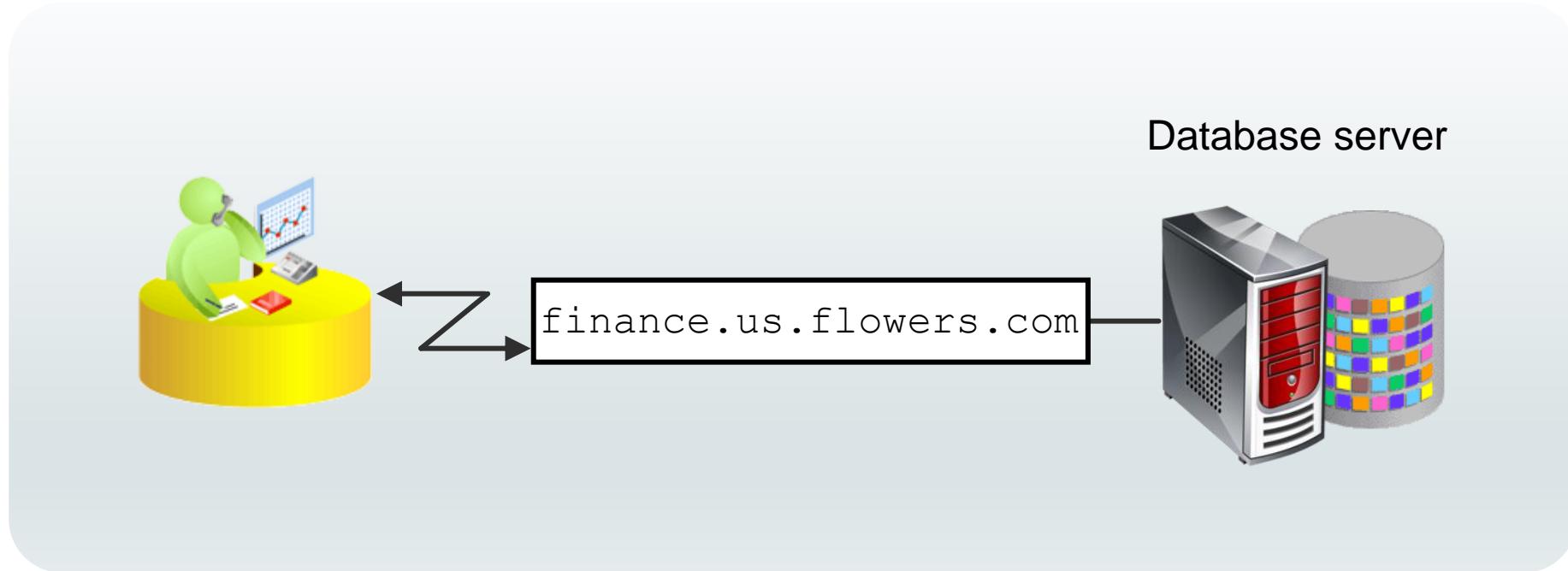
Establishing Oracle Network Connections

To make a client or middle-tier connection, Oracle Net requires the client to know the:

- Host where the listener is running
- Port that the listener is monitoring
- Protocol that the listener is using
- Name of the service that the listener is handling



Connecting to an Oracle Database Instance



Name Resolution

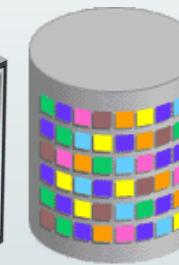


CONNECT jsmith/jspass@finflowers

Name resolution

```
finflowers = (DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp) (HOST=flowers-server) (PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=finance.us.flowers.com) ))
```

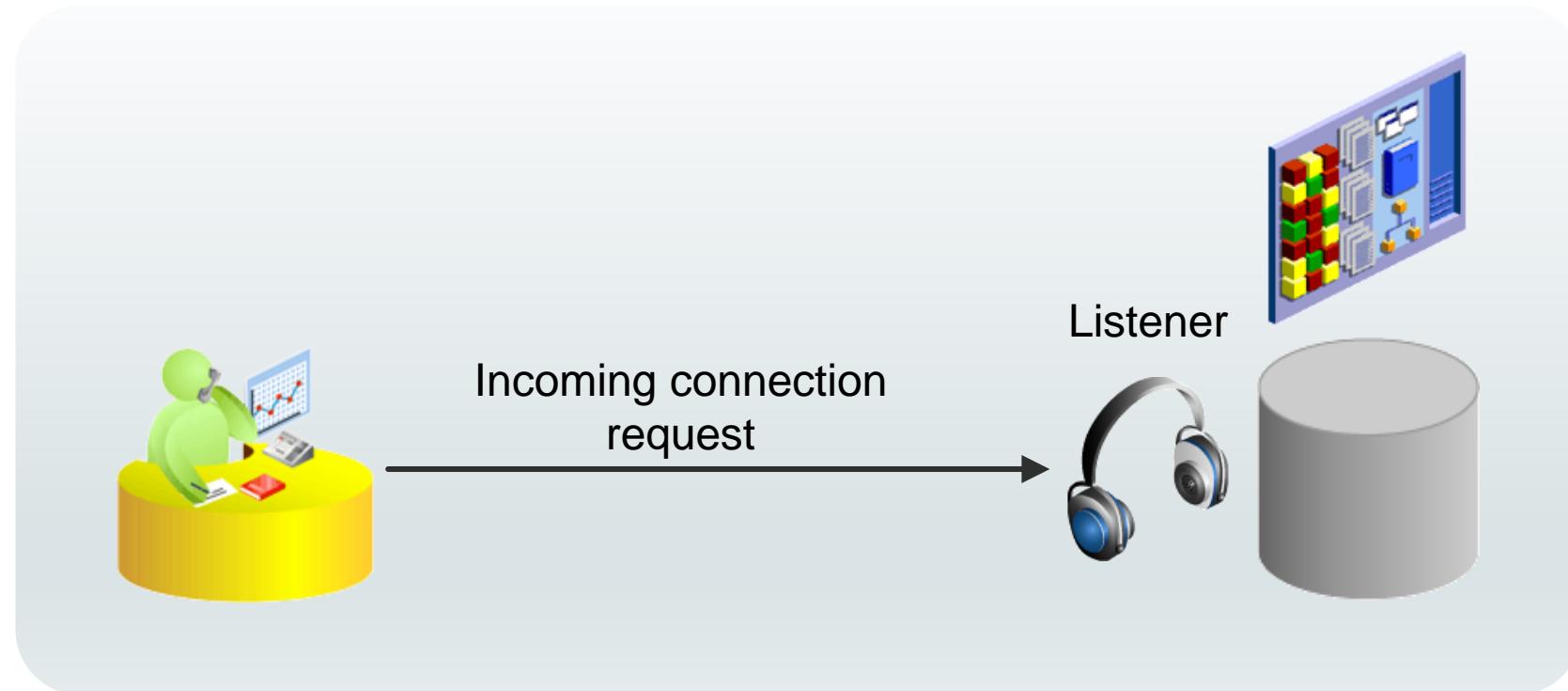
LISTENER
port 1521



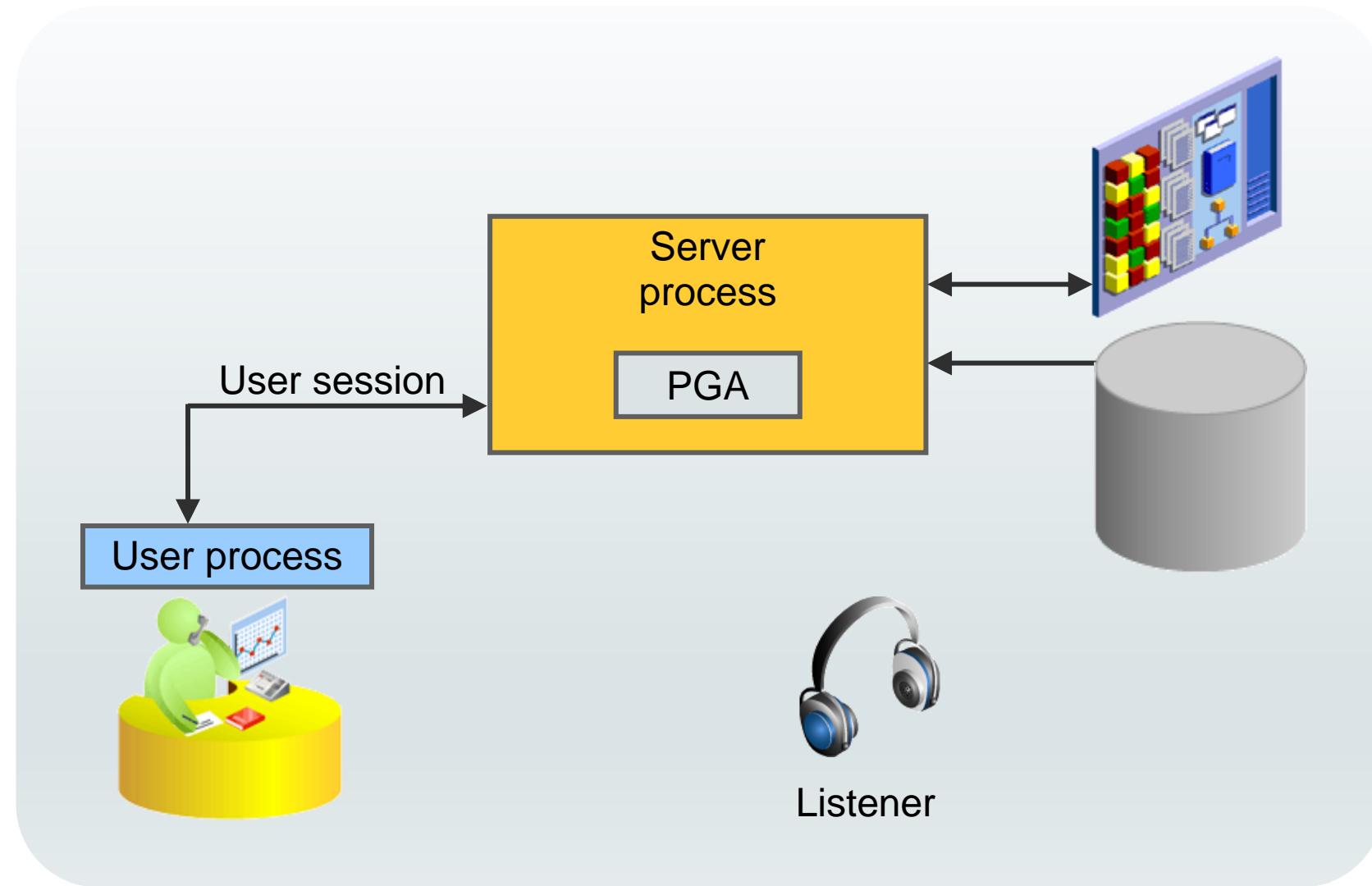
finance

flowers-server

Establishing a Connection



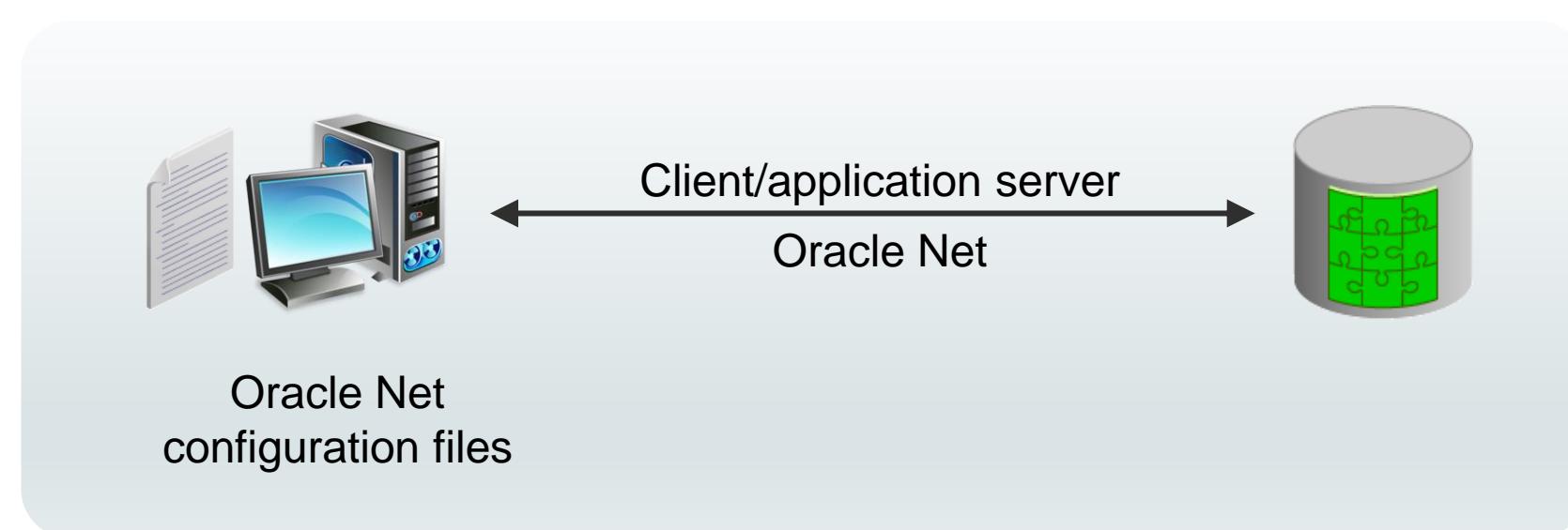
User Sessions



Naming Methods

Oracle Net supports several methods for resolving connection information:

- **Easy connect naming:** Uses a TCP/IP connect string
- **Local naming:** Uses a local configuration file
- **Directory naming:** Uses a centralized LDAP-compliant directory server



Easy Connect

- Is enabled by default
- Requires no client-side configuration
- Supports only TCP/IP (no SSL)
- Offers no support for advanced connection options such as:
 - Connect-time failover
 - Source routing
 - Load balancing

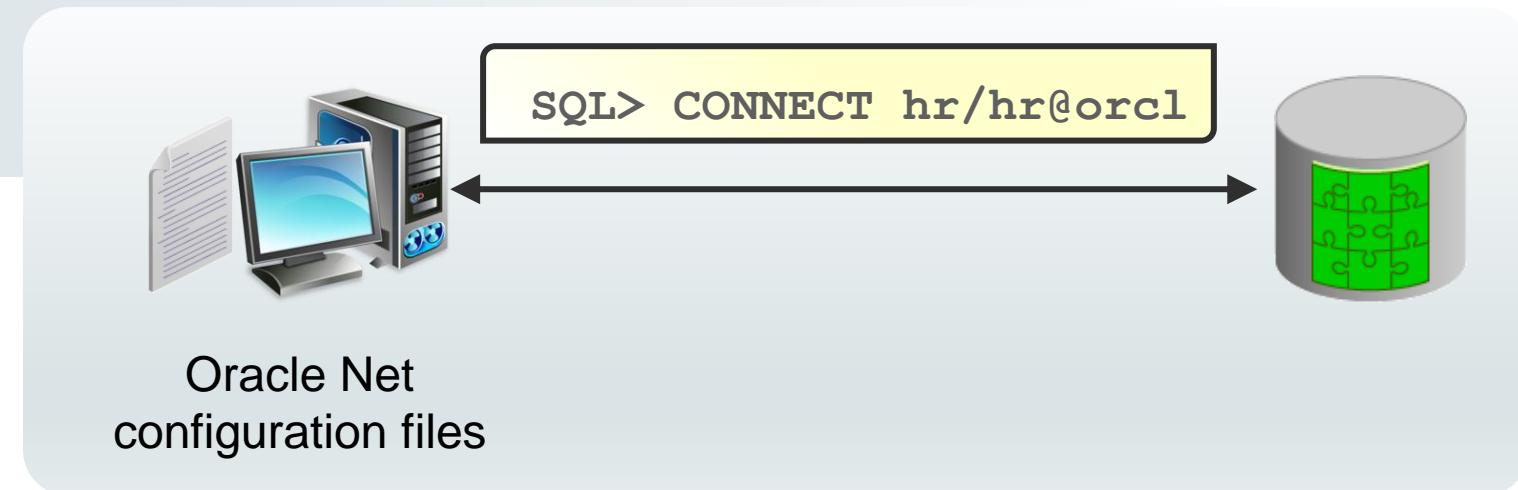
```
SQL> CONNECT hr/hr@db.us.oracle.com:1521/dba11g
```



No Oracle Net
configuration files

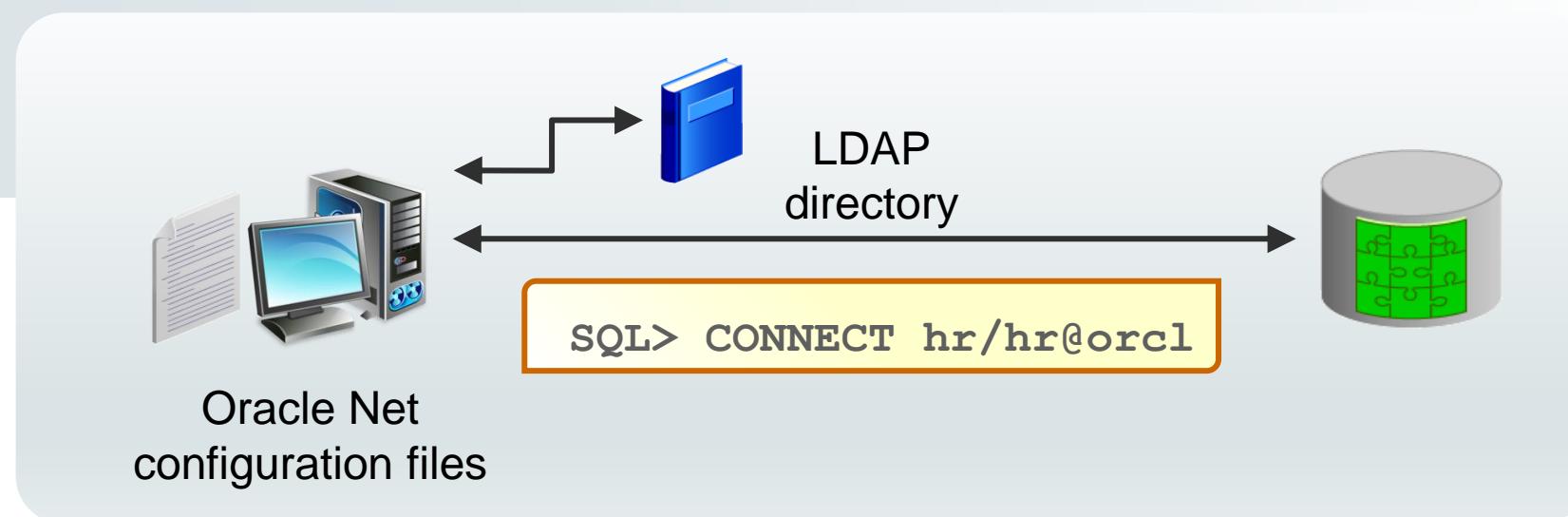
Local Naming

- Requires a client-side names-resolution file
- Supports all Oracle Net protocols
- Supports advanced connection options such as:
 - Connect-time failover
 - Source routing
 - Load balancing

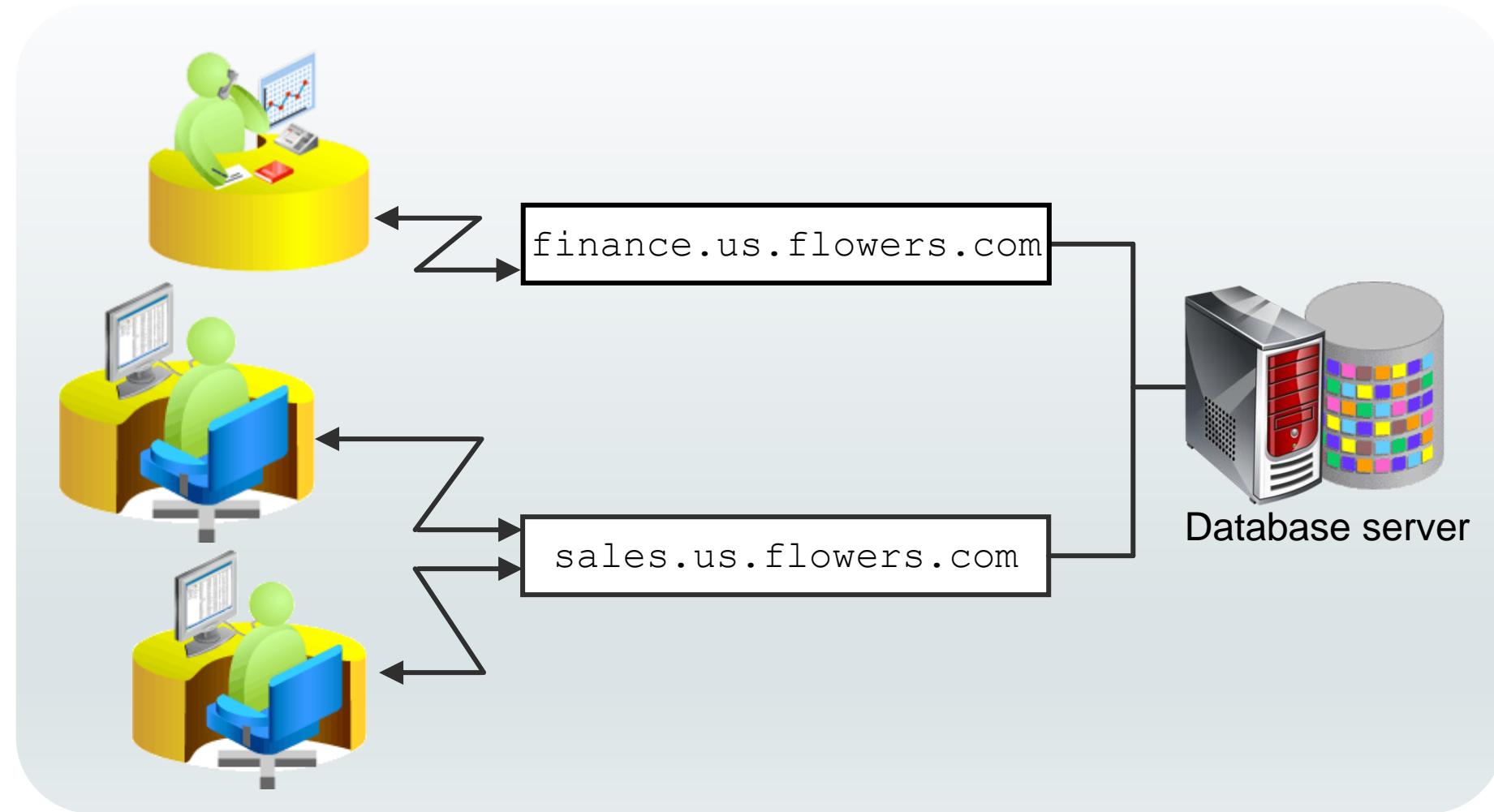


Directory Naming

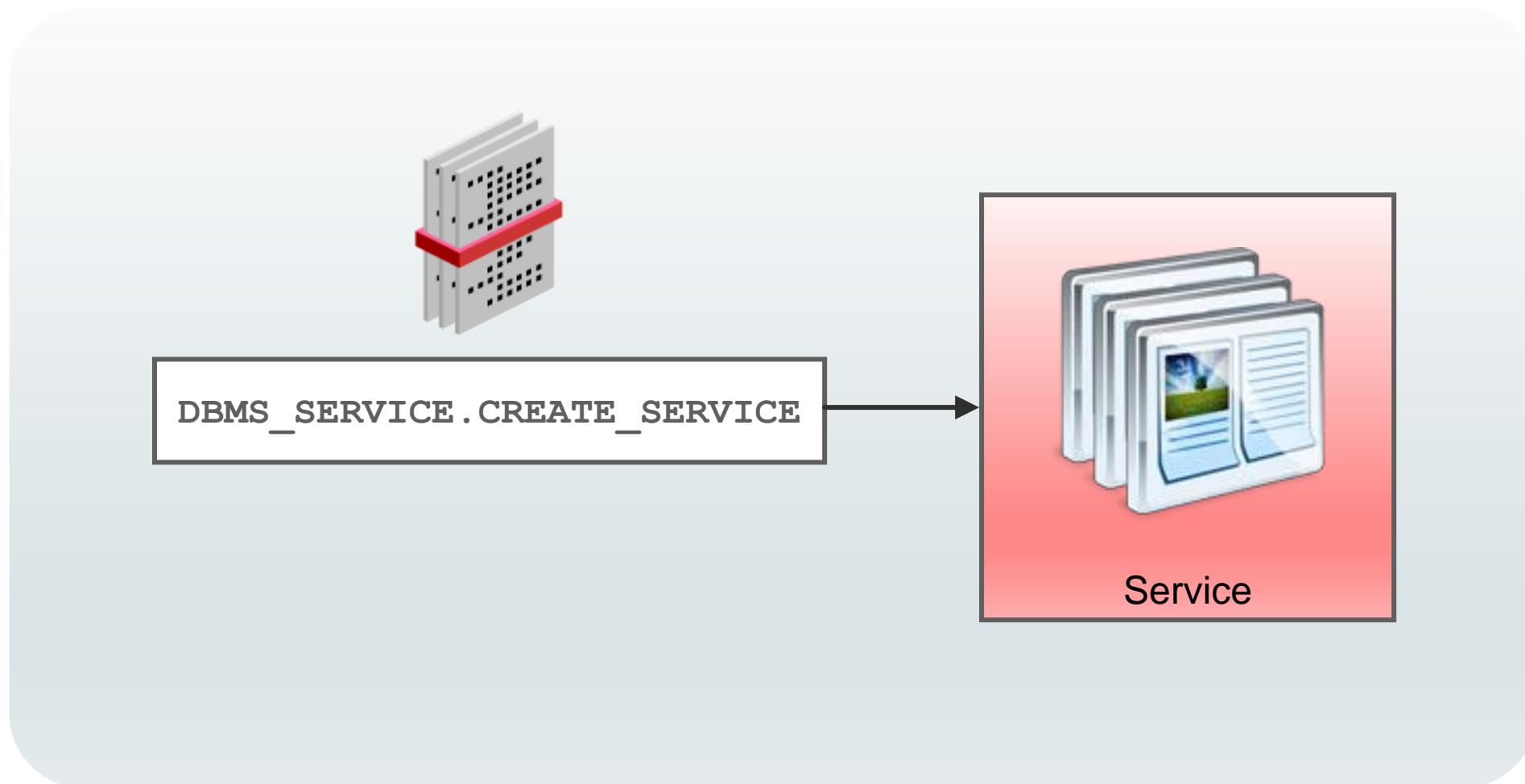
- Requires LDAP with Oracle Net names resolution information loaded:
 - Oracle Internet Directory
 - Microsoft Active Directory Services
- Supports all Oracle Net protocols
- Supports advanced connection options



Using Database Services to Manage Workloads



Creating Database Services



Summary

In this lesson, you should have learned how to:

- Describe Oracle Net Services naming methods
- Configure local naming for database connections

Practice Overview

This practice covers the following topics:

- Configuring the Oracle Network to Access a Database
- Creating a Net Service Name for a PDB

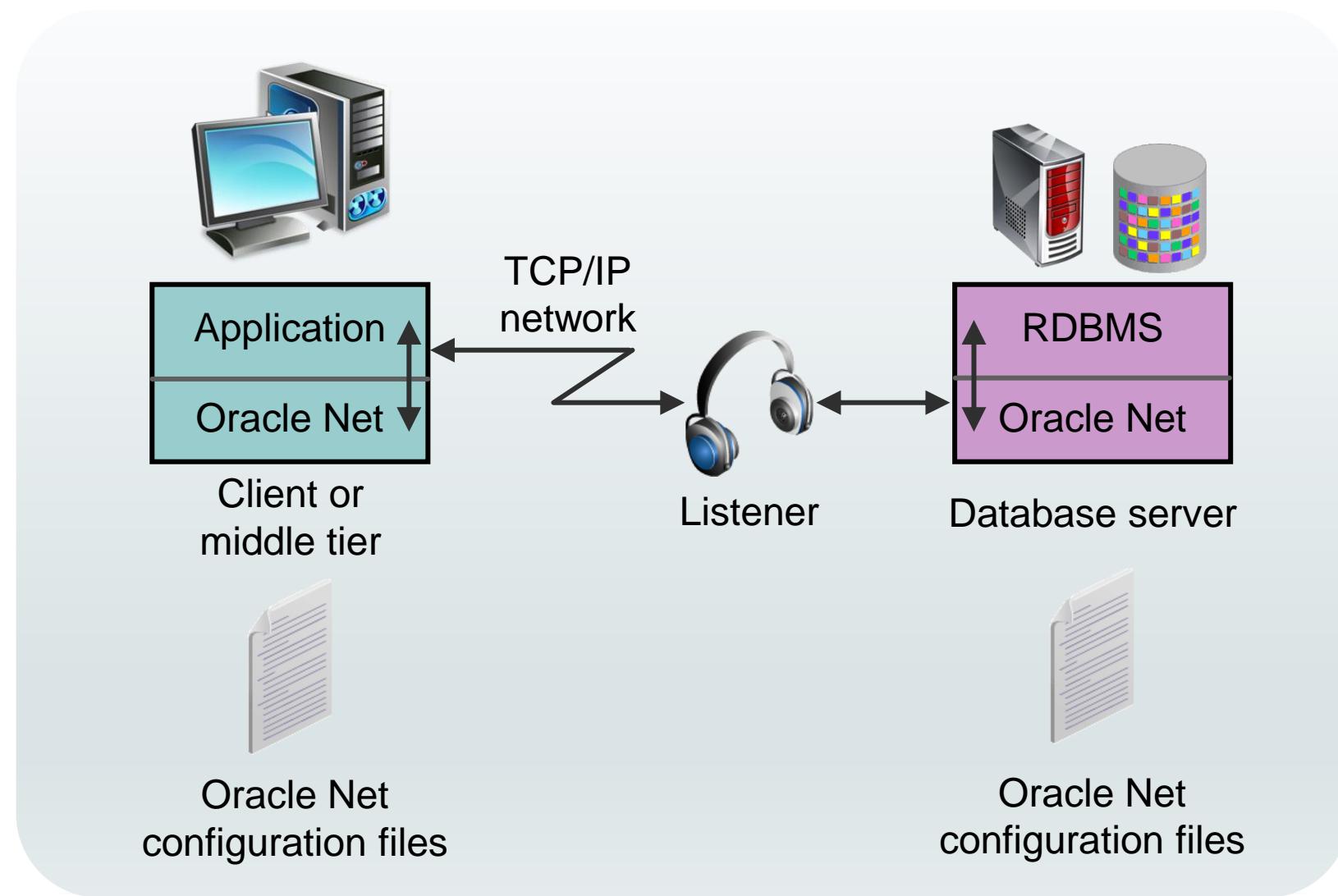
9. Configuring and Administering the Listener

Objectives

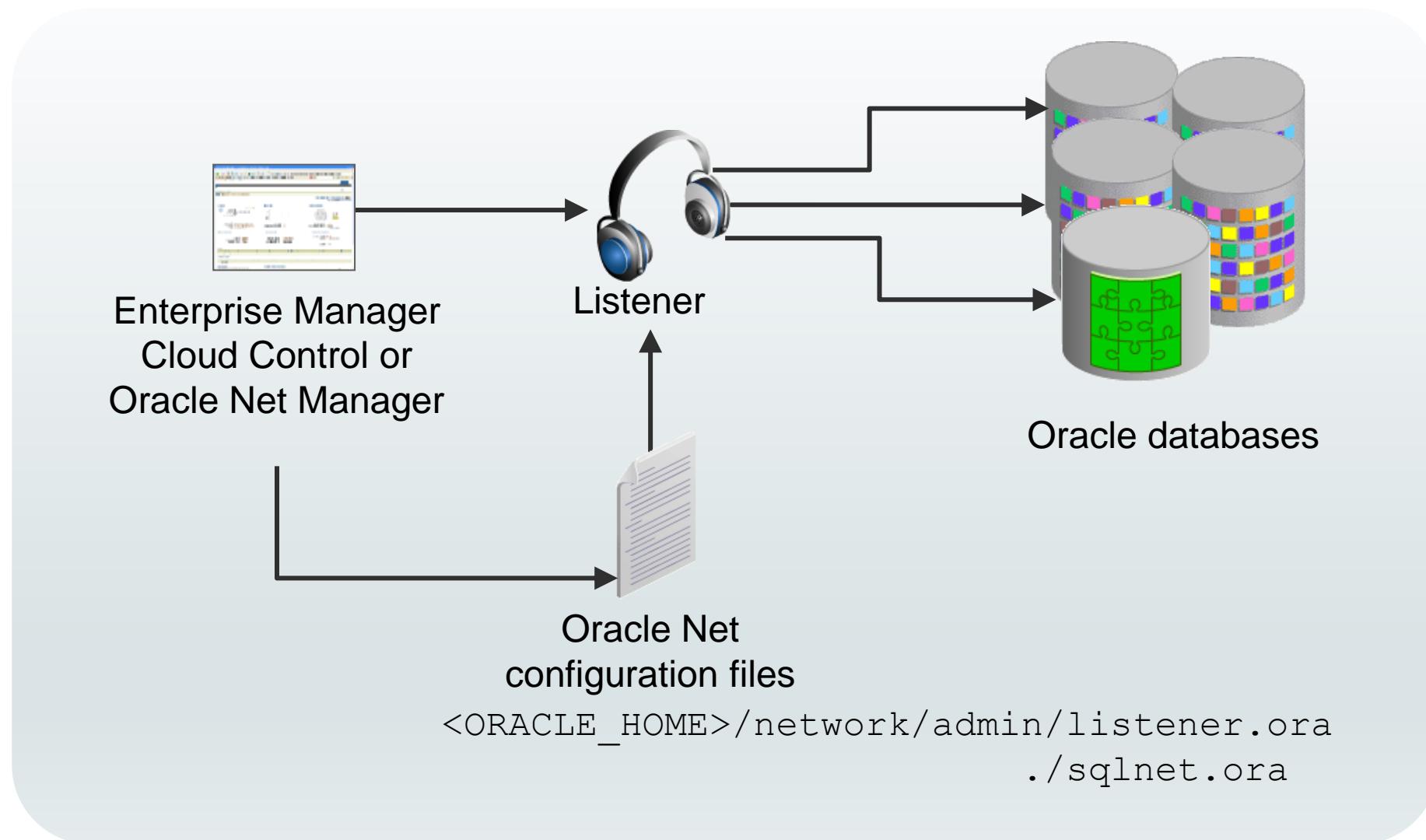
After completing this lesson, you should be able to:

- Explain how listeners work
- Configure listeners for dynamic or static service registration

Review: Oracle Net Services Overview



Oracle Net Listener: Overview



The Default Listener

- During the creation of an Oracle database, the Oracle Net Configuration Assistant creates a local listener named LISTENER.
- LISTENER is automatically populated with available database services through a feature called *dynamic service registration*.
- LISTENER listens on the following TCP/IP protocol address:

ADDRESS=(PROTOCOL=tcp) (HOST=host_name) (PORT=1521))

- Without any configuration, you can access your database instance immediately through LISTENER.

Configuring Dynamic Service Registration

- By default, an Oracle database instance is configured to use dynamic service registration (service registration), which allows the Oracle database instance to identify its available services to listeners automatically.
- The LREG process polls the listeners to see if they are running and, if so, registers database service information to them.
- Dynamic service registration registers, by default, all PDB services to the same listener. If you stop that listener, you stop access to all the PDB services.
- General steps to configure dynamic service registration:
 - Make sure that the `INSTANCE_NAME`, `LOCAL_LISTENER`, `REMOTE_LISTENER`, and `SERVICE_NAMES` initialization parameters are properly configured.
 - Configure protocol addresses (end points) in the server-side `tnsnames.ora` file.
- Use the `ALTER SYSTEM REGISTER` command to initiate service registration immediately after the listener is started.

Configuring Static Service Registration

- Static service registration is a method for configuring listeners to obtain their service information manually.
 - You can create a listener for a particular PDB.
 - Static service registration might be required for some services, such as external procedures and heterogeneous services (for non-Oracle systems).
- With static registration, the listener has no knowledge of whether its database services exist or not. It only knows that it supports them. The Listener Control utility shows the services status as UNKNOWN.
- General steps to configure static service registration:
 1. In `listener.ora`, define a listener and its protocol addresses.
 2. In `listener.ora`, also create a `SID_LIST_<listener name>` section that lists the database services for the listener.

Summary

In this lesson, you should have learned how to:

- Explain how listeners work
- Configure listeners for dynamic or static service registration

Practice Overview

This practice covers the following topics:

- Exploring the Default Listener
- Creating a Second Listener
- Connecting to a Database Service Using the New Listener

10. Configuring a Shared Server Architecture

Objectives

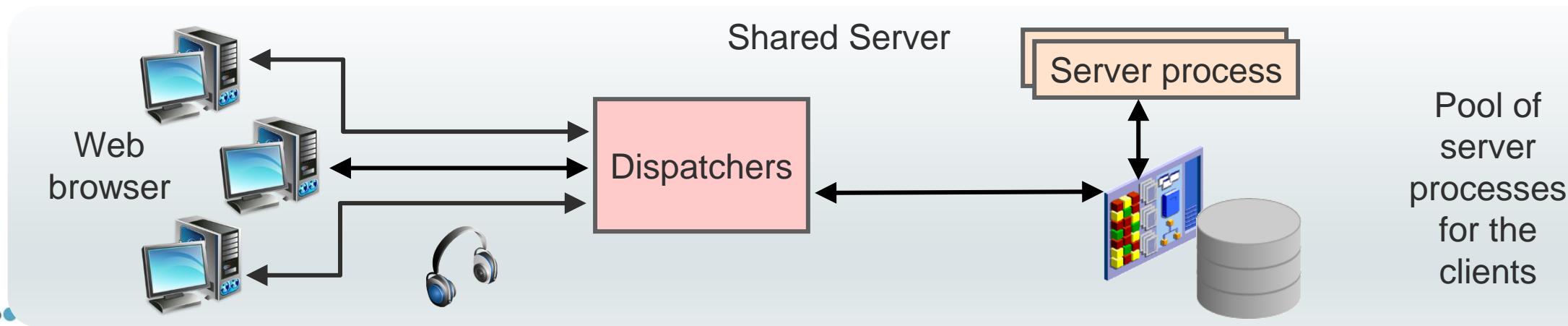
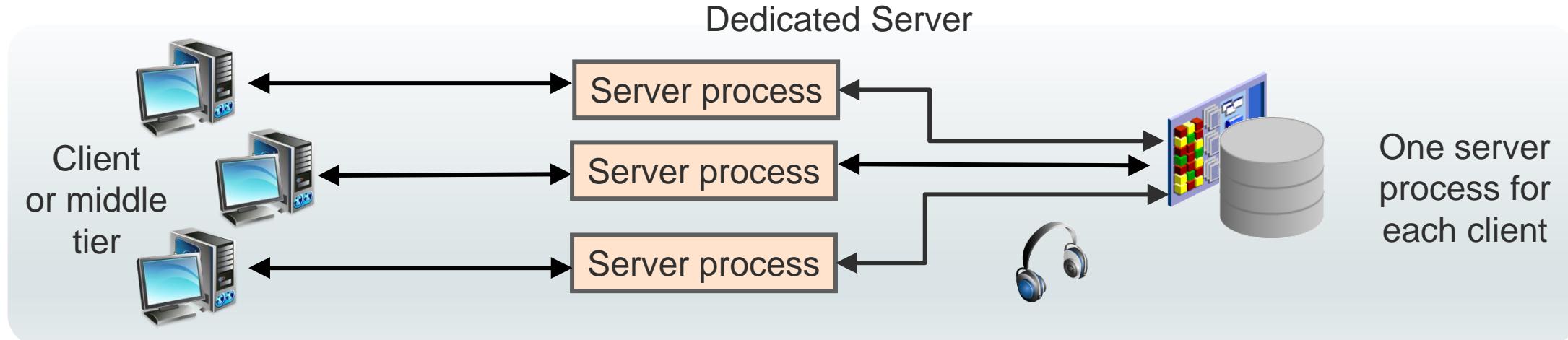
After completing this lesson, you should be able to:

- Explain the difference between dedicated and shared server configurations
- Enable shared server
- Control shared server operations

Shared Server Architecture: Overview

- When should you consider configuring the shared server architecture?
 - System is overloaded.
 - System has limited memory.
- How does the shared server architecture address these issues?
 - Client processes share server processes.
 - Few processes are required to support the client load.

Comparing Dedicated and Shared Server Architecture: Review



Enabling Shared Server

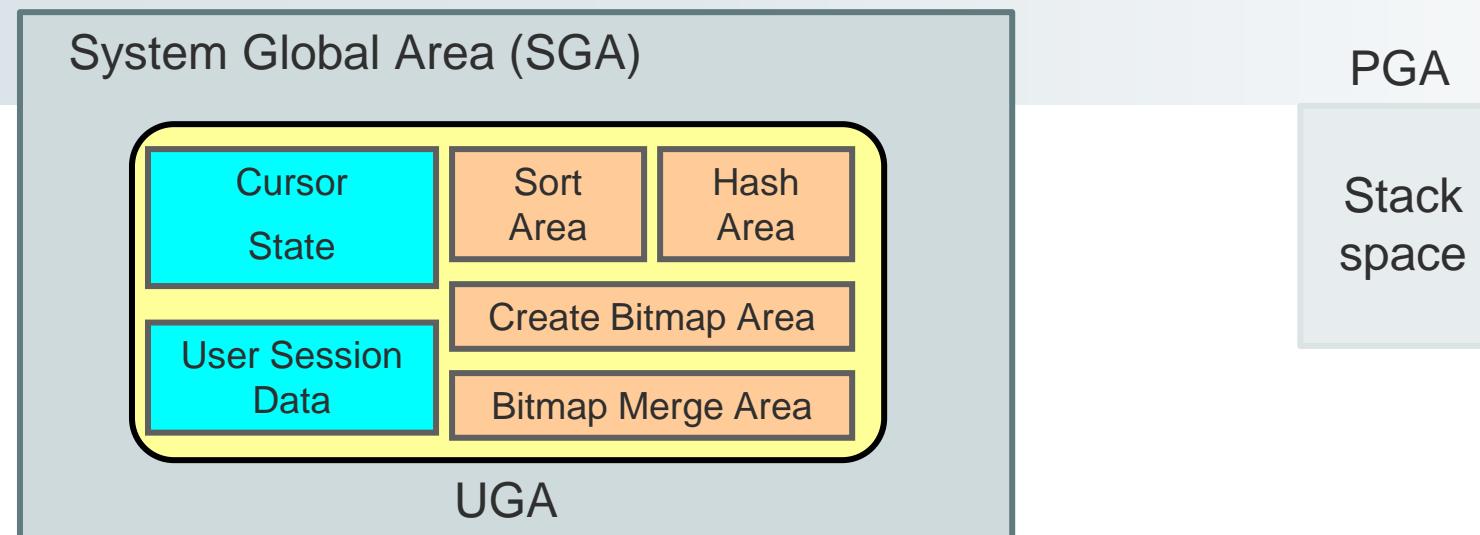
- `SHARED_SERVERS` specifies the minimum number of shared servers that will be created when the instance is started.
- Enable shared server by setting the `SHARED_SERVERS` initialization parameter to a value greater than 0 in the initialization parameter file or by using the `ALTER SYSTEM` command.
- If `SHARED_SERVERS` is not in the initialization parameter file and `DISPATCHERS` is set to 1 or greater, then shared server is enabled by default.

Controlling Shared Server Operations

Initialization Parameter	Used To
DISPATCHERS	Configure dispatcher processes
SHARED_SERVERS	Specify the initial number of shared servers to start and the minimum number of shared servers to keep
MAX_SHARED_SERVERS	Specify the maximum number of shared servers that can run simultaneously
SHARED_SERVER_SESSIONS	Specify the total number of shared server user sessions that can run simultaneously
CIRCUITS	Set a maximum limit on the number of virtual circuits that can be created in shared memory

SGA and PGA Usage

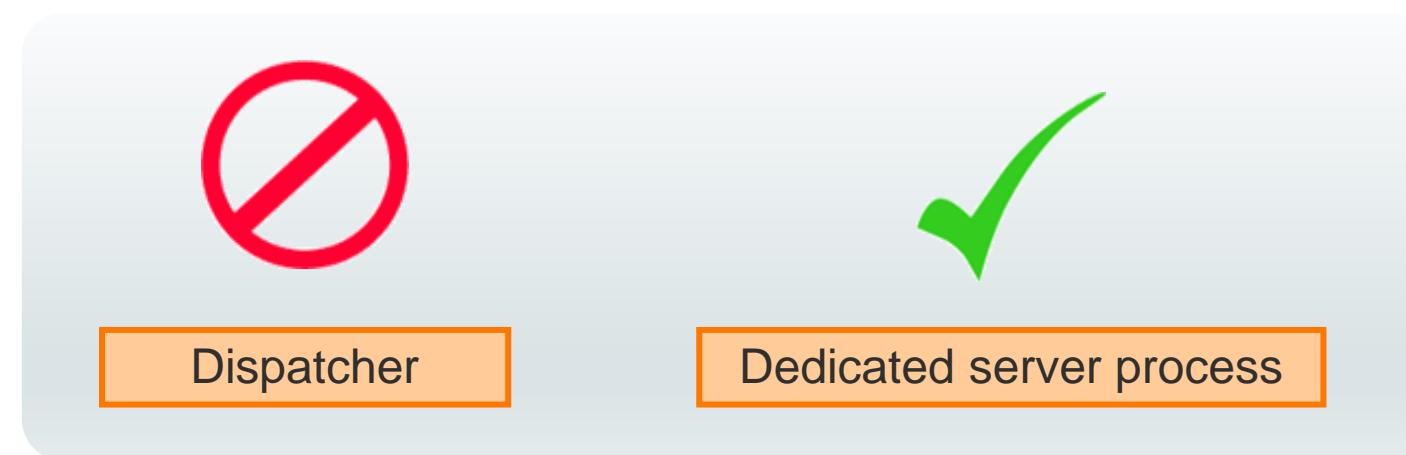
- Oracle Shared Server: User session data is held in the SGA.
- Be sure to consider shared server memory requirements when sizing the SGA.



Shared Server Configuration Considerations

Certain types of database work must not be performed using shared servers:

- Database administration
- Backup and recovery operations
- Batch processing and bulk-load operations
- Data warehouse operations



Summary

In this lesson, you should have learned how to:

- Explain the difference between dedicated and shared server configurations
- Enable shared server
- Control shared server operations

Practice Overview

This practice covers the following topics:

- Configuring Shared Server Mode
- Configuring Clients to Use a Shared Server

11. Configuring Oracle Connection Manager for Multiplexing and Access Control

Objectives

After completing this lesson, you should be able to:

- Identify the capabilities of Oracle Connection Manager
- Describe the Oracle Connection Manager architecture
- Configure Oracle Connection Manager for session multiplexing
- Use the Oracle Connection Manager Control utility to administer Oracle Connection Manager

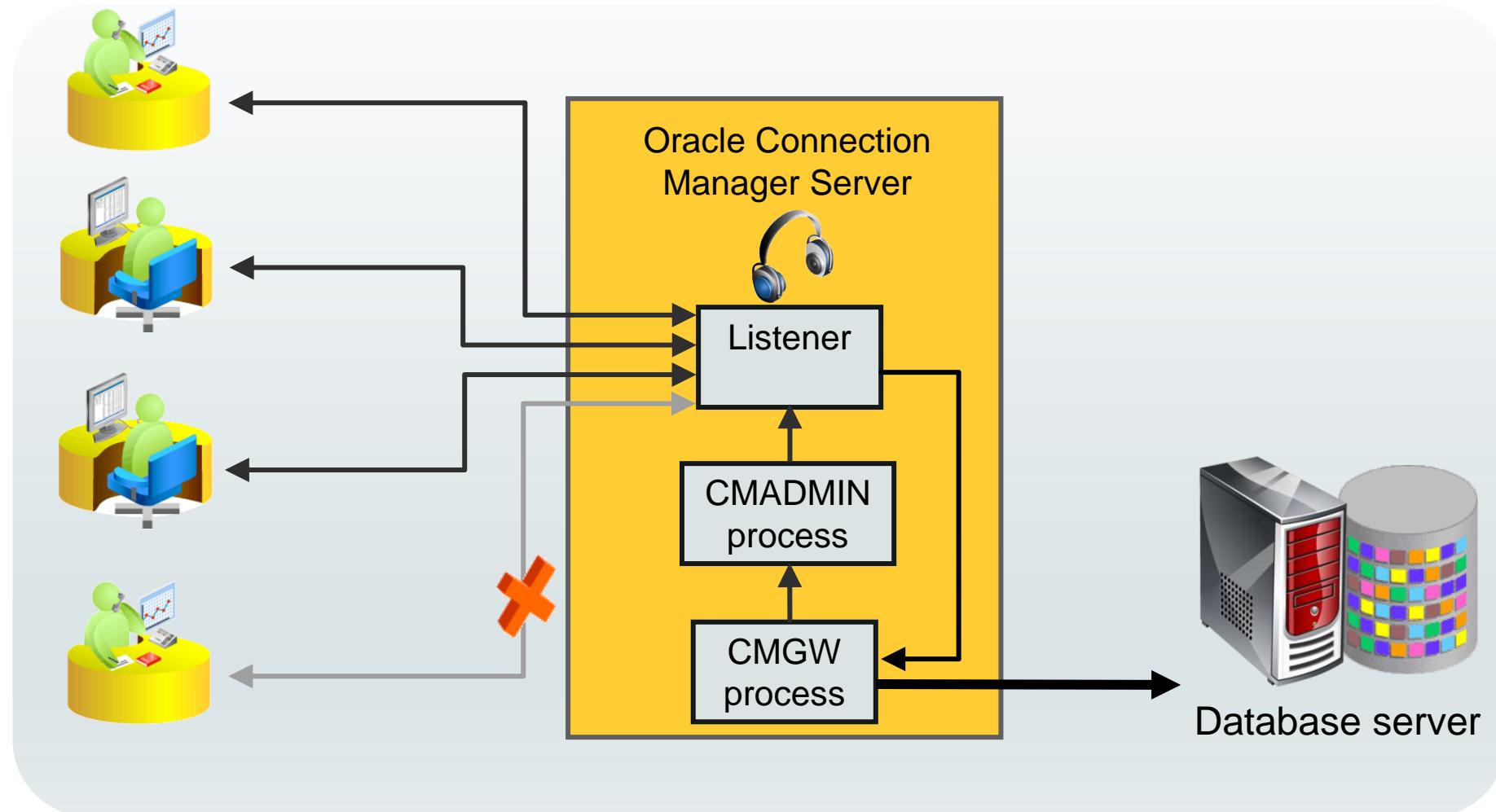
Oracle Connection Manager: Overview

- Typically resides on a separate server
- Functions as a proxy server
- Can be configured to perform:
 - Access control filtering
 - Session multiplexing

Oracle Connection Manager Processes

- Gateway process: CMGW
 - Receives client connections
 - Uses rules to evaluate connection and determine whether to allow access
 - Forwards requests to the next hop if access is allowed
 - Also multiplexes client connections
- Administrative process: CMADMIN
 - Multithreaded process
 - Performs administrative functions
- The processes are started with Oracle Connection Manager Control utility.

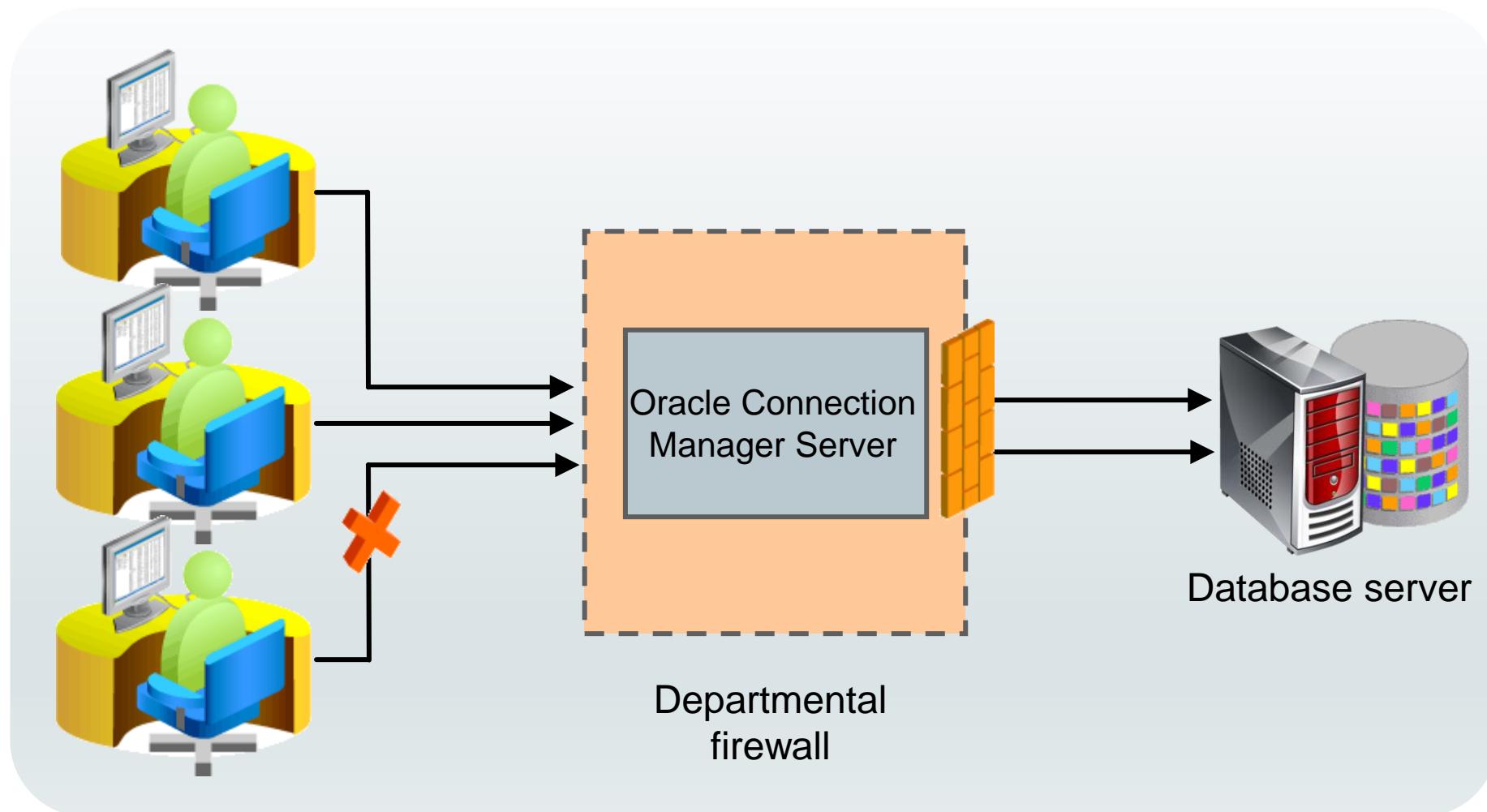
Oracle Connection Manager: Architecture



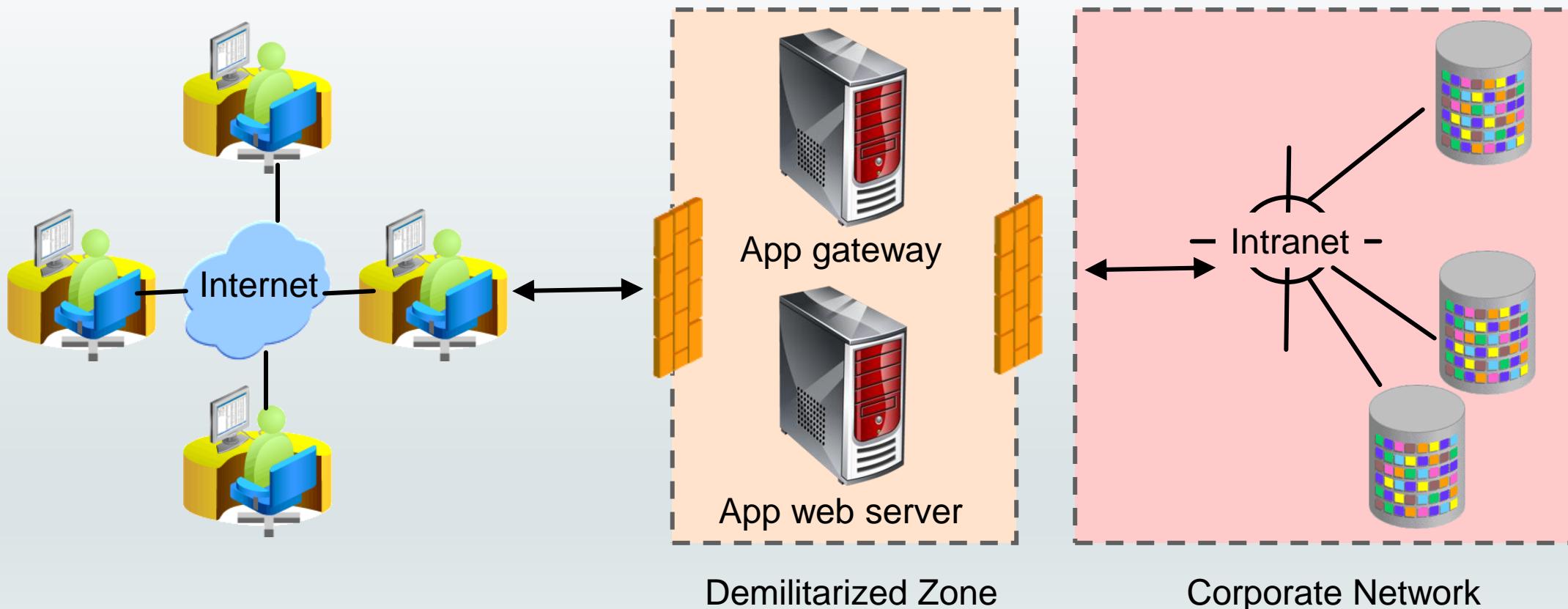
Using Filtering Rules

- Control access to Oracle databases using filtering rules
- Client access based on:
 - Source host names or IP addresses for client
 - Destination host names or IP addresses for servers
 - Destination database service names
 - Client use of Oracle Advanced Security
- Specified through the `CMAN_RULES` parameter in the `cman.ora` file

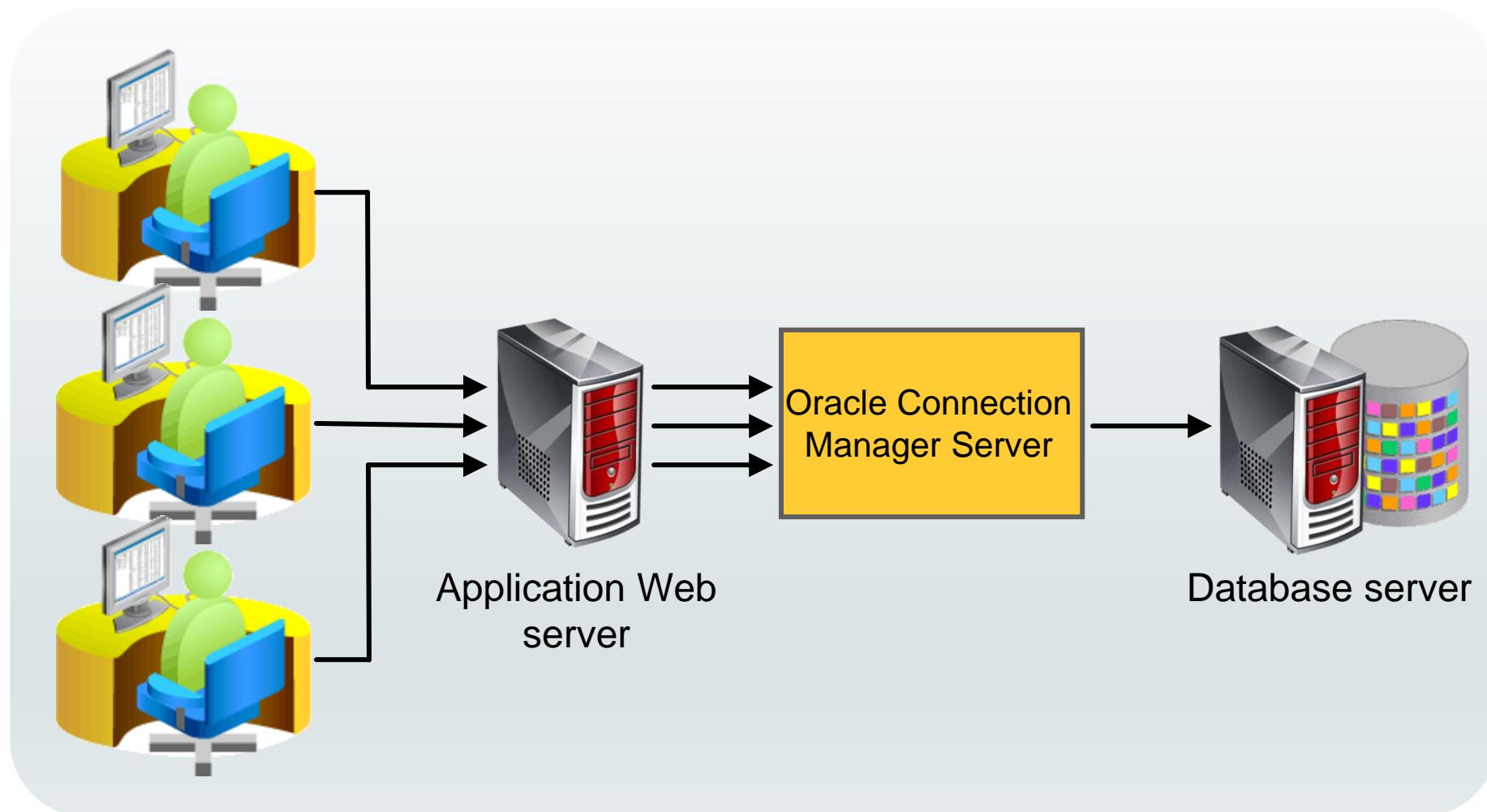
Implementing Intranet Access Control



Implementing Internet Access Control



Using Session Multiplexing



Configuring Oracle Connection Manager

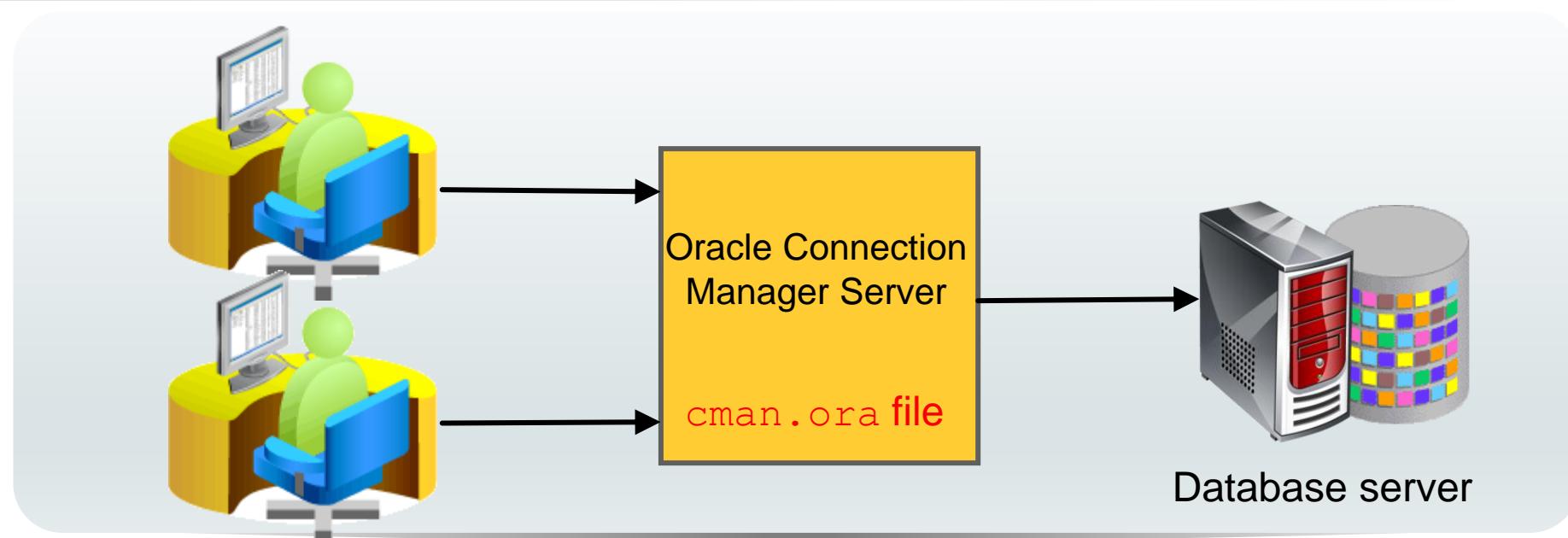
Three-step process:

1. Configure the `cman.ora` file on the Connection Manager server.
2. Configure clients with the protocol address of the Oracle Connection Manager listener.
3. Configure the database server for session multiplexing (optional).

Configuring the cman.ora File

Define the following in the `cman.ora` file:

- Protocol address of the Oracle Connection Manager listener (`ADDRESS`)
- Access control rules (`RULE_LIST`)
- Performance parameters (`PARAMETER_LIST`)



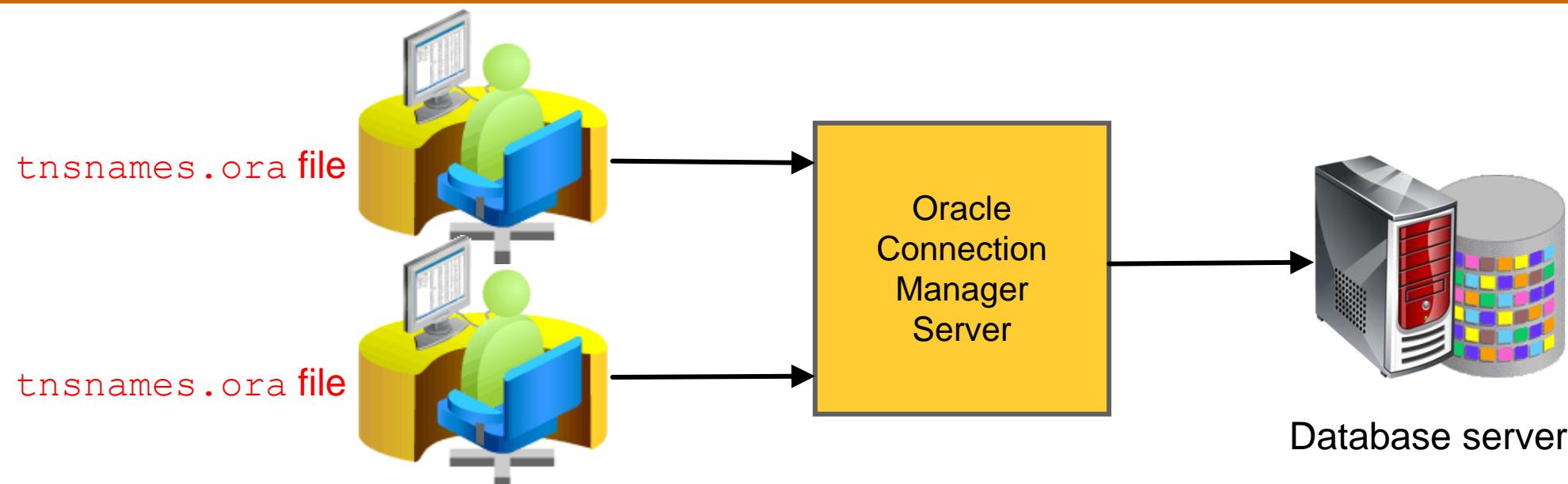
Example of a cman.ora File

```
CMAN01=
  (CONFIGURATION=
    (ADDRESS=(PROTOCOL=tcp) (HOST=proxysvr) (PORT=1521) )
    (RULE_LIST=
      (RULE=(SRC=192.0.2.32/24) (DST=fin-server) (SRV=*) (ACT=accept)
        (ACTION_LIST=(AUT=on) (MCT=120) (MIT=30)))
      (RULE=(SRC=192.0.2.32) (DST=proxysvr) (SRV=cmon) (ACT=accept)))
    (PARAMETER_LIST=
      (LOG_LEVEL=2)
      (TRACING=on)
      (MAX_GATEWAY_PROCESSES=8)
      (MIN_GATEWAY_PROCESSES=3)))
```

Configuring Clients

Configure the `tnsnames.ora` file with a connect descriptor that specifies the protocol address of Oracle Connection Manager.

```
finance=
  (DESCRIPTION=
    (ADDRESS= (PROTOCOL=tcp) (HOST=cman-serv) (PORT=1521))
    (CONNECT_DATA= (SERVICE_NAME=FINDB01)))
```

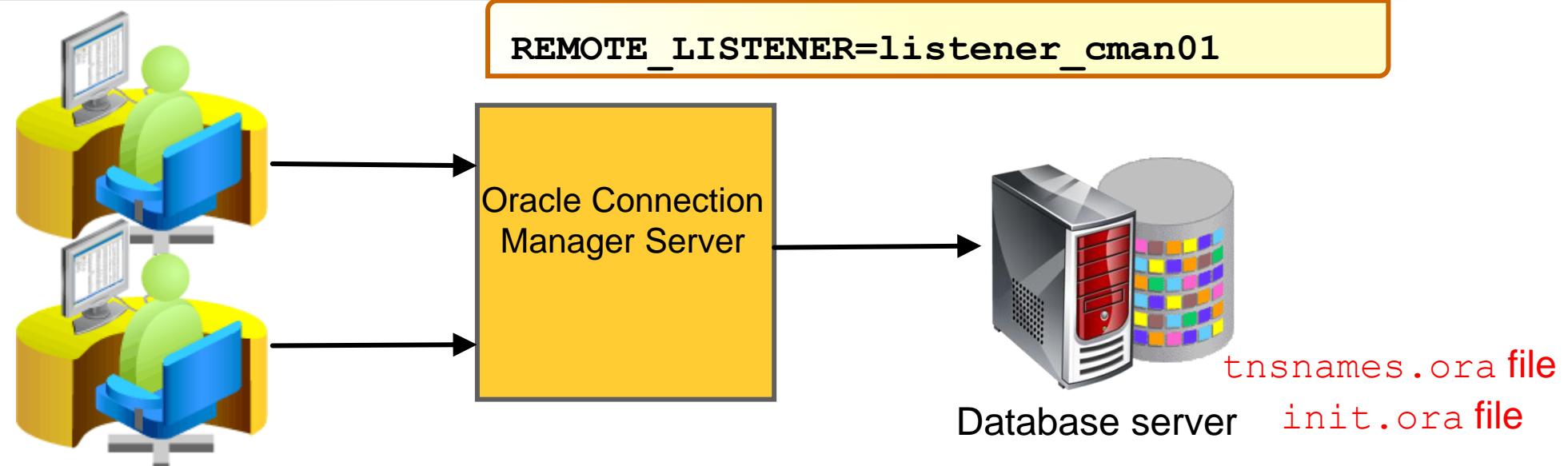


Configuring the Database Server

- In the tnsnames.ora file, add a service name entry for Oracle Connection Manager.

```
listener_cman01=
...
(ADDRESS=(PROTOCOL=tcp) (HOST=proxyserver1) (PORT=1521)))
```

- In the initialization parameter file, specify an alias for Oracle Connection Manager.

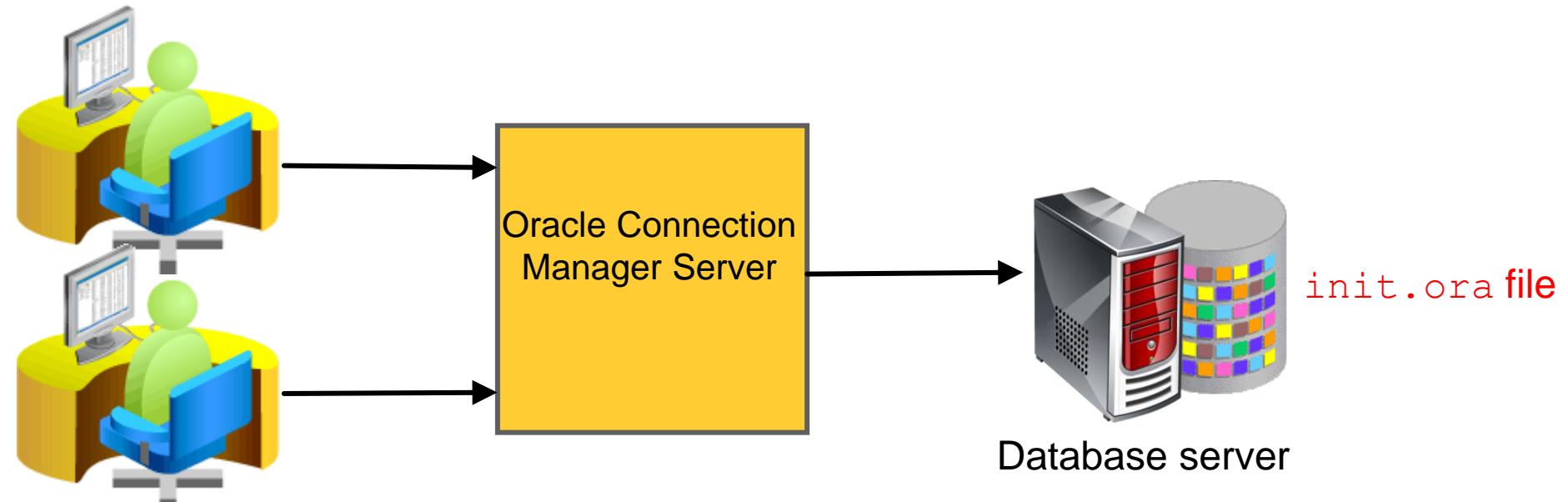


Configuring the Database Server for Multiplexing (Optional)

In the initialization parameter file, set attributes for the `DISPATCHERS` parameter:

- PROTOCOL
- MULTIPLEX

```
DISPATCHERS = "(PROTOCOL=TCP) (MULTIPLEX=ON)"
```



Using the Oracle Connection Manager Control Utility

- Basic syntax for the utility: CMCTL *command* [*process_type*]
- Types of commands supported:
 - Operational (START, STOP)
 - Modifier (SET, SET LOG LEVEL)
 - Informational (SHOW, SHOW ADDRESS, STATUS)
 - Utility operational (EXIT, HELP, QUIT)
- *process_type* is the name of the Oracle Connection Manager process:
 - cman: Both the gateway and administrative processes (default)
 - cm: The gateway process
 - adm: The administrative process

Review of Oracle Connection Manager Features

- **Access control filtering** through the CMAN_RULES parameter in the cman.ora file
- **Session multiplexing** through the PROTOCOL and MULTIPLEX attributes of the DISPATCHERS initialization parameter

Summary

In this lesson, you should have learned how to:

- Identify the capabilities of Oracle Connection Manager
- Describe the Oracle Connection Manager architecture
- Configure Oracle Connection Manager for session multiplexing
- Use the Oracle Connection Manager Control utility to administer Oracle Connection Manager

Practice Overview

This practice covers the following topics:

- Installing Oracle Instant Client
- Configuring the `cman.ora` File
- Configuring the Database for Oracle Connection Manager
- Configuring Clients for Oracle Connection Manager
- Configuring the Oracle Database Server for Session Multiplexing

12. Creating PDBs from Seed

Objectives

After completing this lesson, you should be able to create a new PDB from the PDB seed.

Provisioning New Pluggable Databases

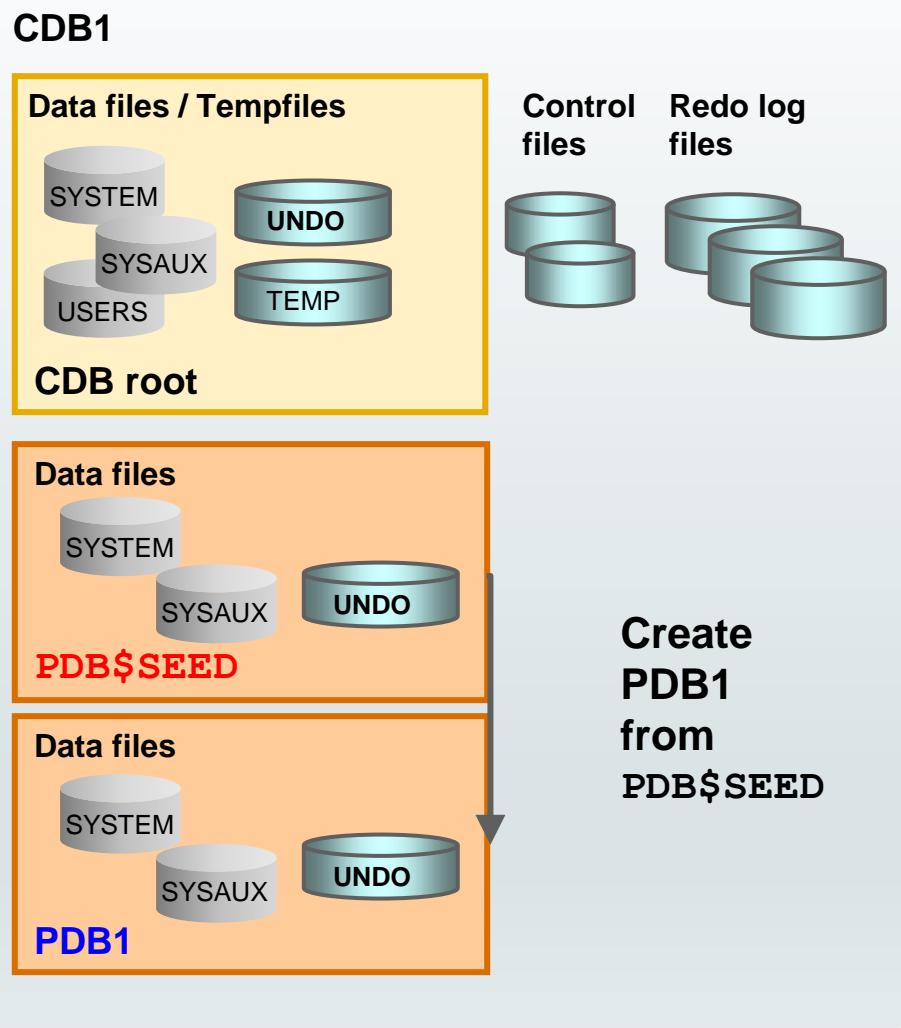
- Create a new PDB from the PDB seed.
- Plug an unplugged PDB into the same CDB or into another CDB.
- Plug a non-CDB in a CDB.
- Clone a PDB from another PDB (local or remote CDB, hot or cold).
- Relocate a PDB from a CDB into another CDB.
- Proxy a PDB from another PDB.

Tools

To provision new PDBs, you can use:

- SQL*Plus
- SQL Developer
- Enterprise Manager Cloud Control
- Enterprise Manager Database Express
- Database Configuration Assistant (DBCA)
 - Clone from PDB seed
 - Clone from an existing PDB
 - Plug an unplugged PDB

Creating a New PDB from PDB\$SEED



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

Using the FILE_NAME_CONVERT Clause

Create a new PDB from the seed using **FILE_NAME_CONVERT**:

1. Connect to the CDB root as a common user with the CREATE PLUGGABLE DATABASE system privilege:

```
SQL> CREATE PLUGGABLE DATABASE pdb1
      ADMIN USER admin1 IDENTIFIED BY p1 ROLES=(CONNECT)
      FILE_NAME_CONVERT = ('PDB$SEEDdir', 'PDB1dir');
```

1. Use views to verify:

```
SQL> CONNECT / AS SYSDBA
SQL> SELECT * FROM cdb_pdbs;
SQL> SELECT * FROM cdb_tablespaces;
SQL> SELECT * FROM cdb_data_files;
SQL> ALTER PLUGGABLE DATABASE pdb1 OPEN RESTRICTED;
SQL> CONNECT sys@pdb1 AS SYSDBA
SQL> CONNECT admin1@pdb1
```

Using OMF or the **PDB_FILE_NAME_CONVERT** Parameter

- Use OMF: **DB_CREATE_FILE_DEST** = '/u01/app/oradata/CDB1/**pdb1**'

Or

- Use the initialization parameter: **PDB_FILE_NAME_CONVERT** = '/u01/app/oradata/CDB1/seed','/u01/app/oradata/CDB1/**pdb1**'

```
SQL> CREATE PLUGGABLE DATABASE pdb1
      ADMIN USER pdb1_admin IDENTIFIED BY p1 ROLES=(CONNECT);
```

Or

- Use the clause in the CREATE PLUGGABLE DATABASE command:
CREATE_FILE_DEST = '/u01/app/oradata/CDB1/**pdb1**'

Summary

In this lesson, you should have learned how to create a new PDB from the PDB seed.

Practice Overview

- Creating a PDB from the PDB Seed

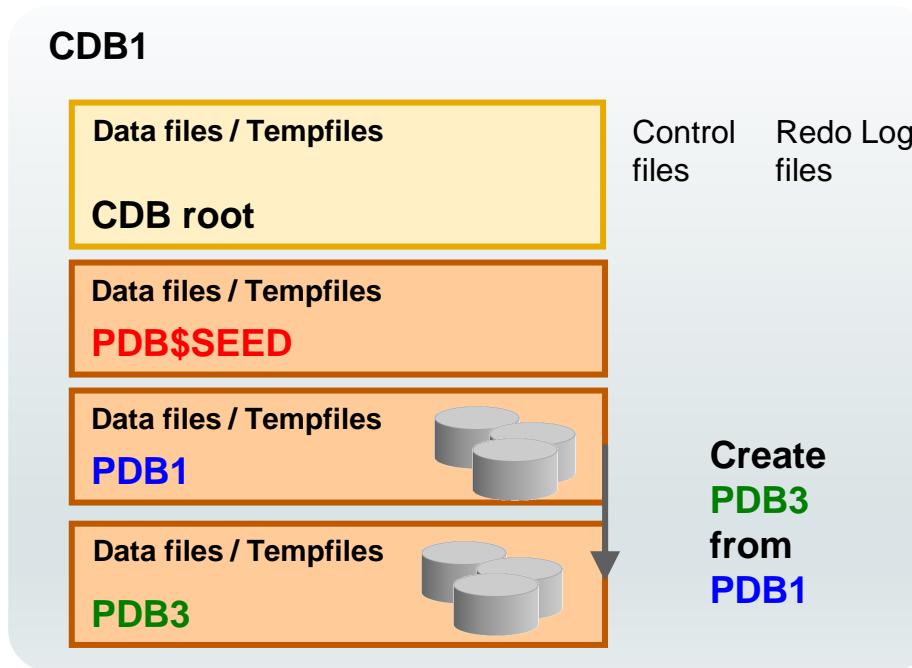
13. Using Other Techniques to Create PDBs

Objectives

After completing this lesson, you should be able to:

- Clone a regular PDB
- Unplug and plug or clone a non-CDB
- Unplug and plug a regular PDB
- Perform hot cloning
- Perform near-zero downtime PDB relocation
- Create and use a proxy PDB

Cloning Regular PDBs



PDB3 owns:

- SYSTEM, SYSAUX, UNDO tablespaces
- Full catalog
- SYS, SYSTEM common users
- Same local administrator name
- New service name

1. Define the data file location:

- Set `DB_CREATE_FILE_DEST= 'PDB3dir'`
- Set `PDB_FILE_NAME_CONVERT='PDB1dir', 'PDB3dir'`
- Use the `CREATE_FILE_DEST= 'PDB3dir'` clause

2. Connect to the CDB root to close **PDB1**.

3. Clone **PDB3** from **PDB1**.

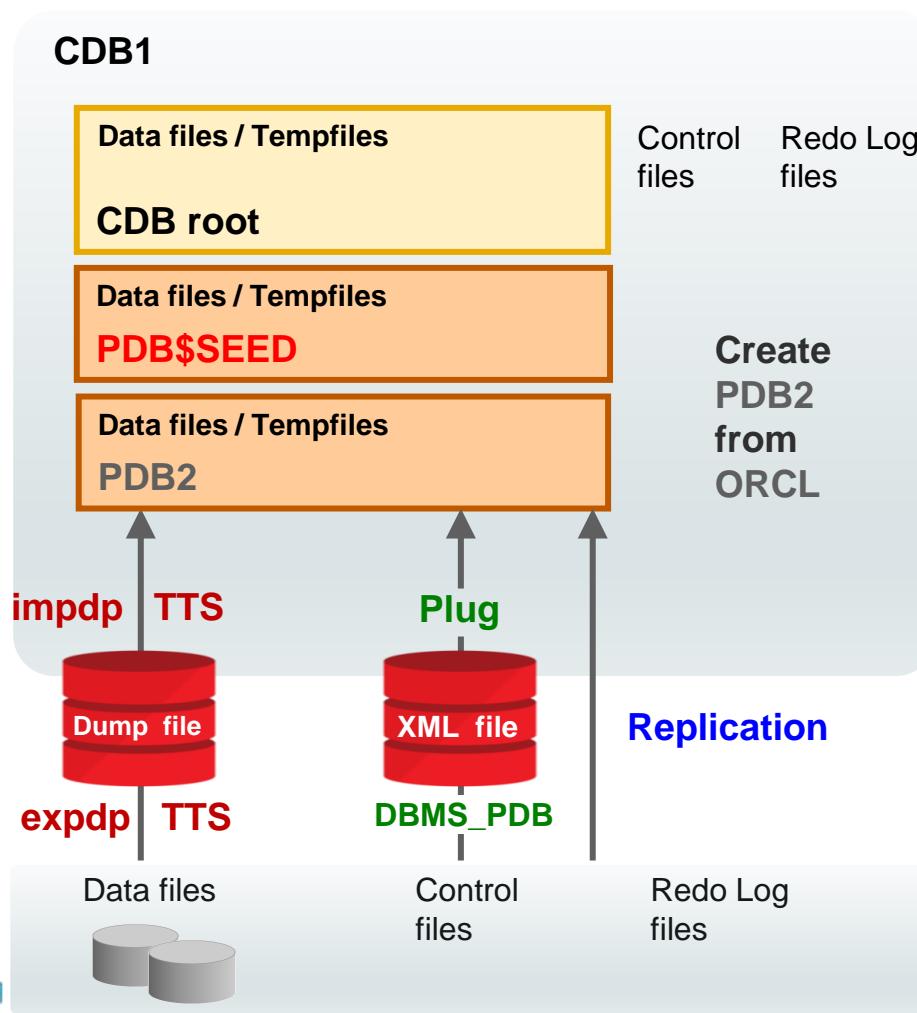
```
SQL> CREATE PLUGGABLE DATABASE pdb3 FROM pdb1  
      CREATE_FILE_DEST = 'PDB3dir';
```

1. Open **PDB3** in read write mode.

```
SQL> ALTER PLUGGABLE DATABASE pdb3 OPEN;
```

Note: Clone metadata only by using NO DATA.

Migrating Data from a Non-CDB into a CDB



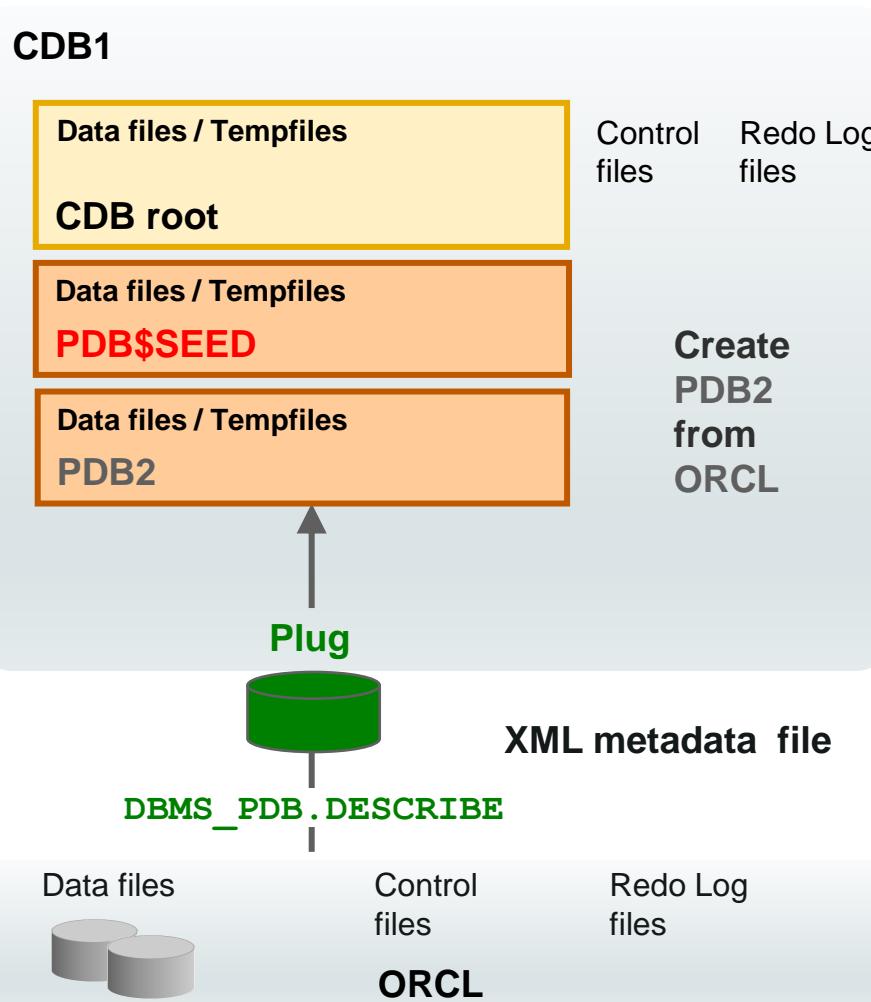
Possible methods:

- **Data Pump (TTS or TDB or full export/import)**
- **Plugging** (XML file definition with `DBMS_PDB`)
- **Cloning**
- **Replication**

Entities created in the new PDB:

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

Plugging a Non-CDB into CDB Using DBMS_PDB



1. Open ORCL in READ ONLY mode.

```
SQL> EXEC DBMS_PDB.DESCRIBE ('/tmp/ORCL.xml')
```

2. Connect to the target CDB root as a common user with the CREATE PLUGGABLE DATABASE privilege.
2. Plug in the unplugged ORCL as PDB2.

```
SQL> CREATE PLUGGABLE DATABASE PDB2  
      USING '/tmp/ORCL.xml';
```

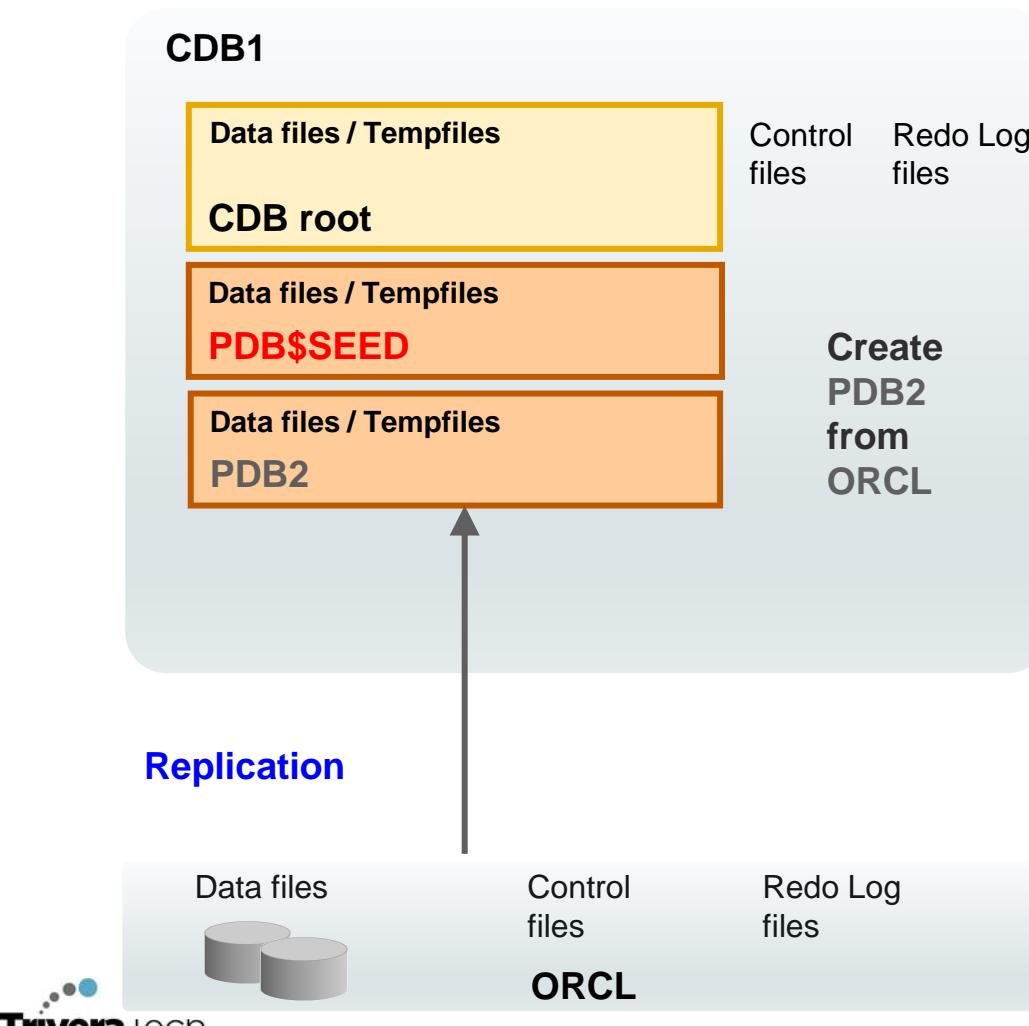
4. Run the noncdb_to_pdb.sql script in PDB2.

```
SQL> CONNECT sys@PDB2 AS SYSDBA  
SQL> @$ORACLE_HOME/rdbms/admin/noncdb_to_pdb
```

4. Open PDB2.

Note: The STATUS of the PDB is CONVERTING.

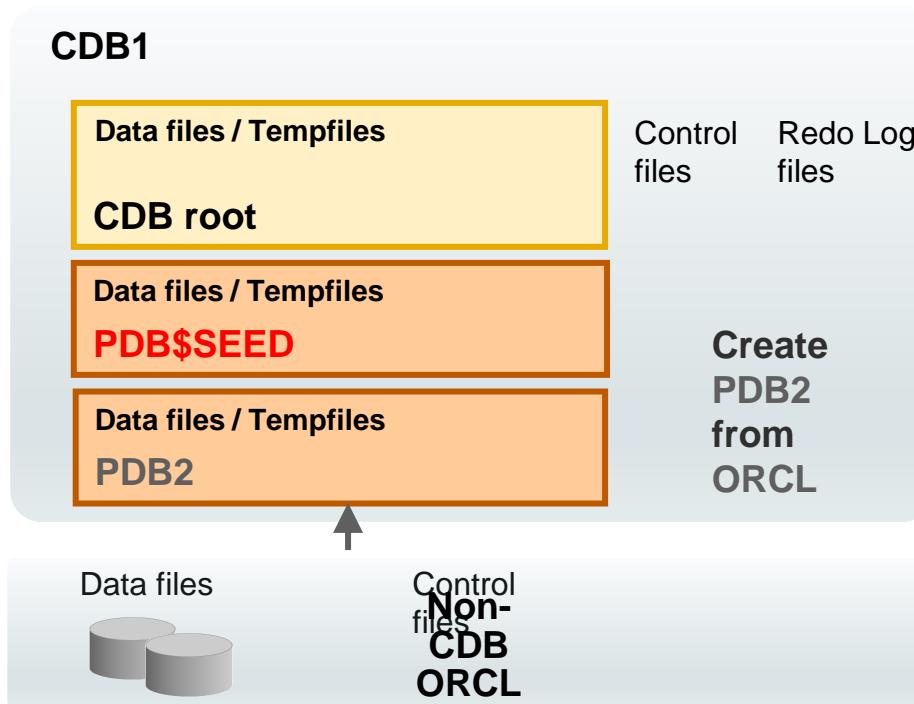
Replicating a Non-CDB into a CDB by Using GoldenGate



1. Connect to the CDB root as a common user with the CREATE PLUGGABLE DATABASE privilege.
2. Create new **PDB2** (from PDB\$SEED).
3. Open **PDB2** in read/write mode.
4. Configure an Oracle GoldenGate unidirectional replication environment from **ORCL** to **PDB2**.
5. Check application data.

```
SQL> CONNECT sys@PDB2
SQL> SELECT * FROM dba_tables;
SQL> SELECT * FROM HR.EMP;
```

Cloning a Non-CDB or Remote PDB



PDB_ORCL owns:

- SYSTEM, SYSAUX, UNDO tablespaces
- Full catalog
- A temporary tablespace
- SYS, SYSTEM common users
- New service name

1. Set ORCL in READ ONLY mode.
2. Connect to the CDB to create the database link:

```
SQL> CREATE DATABASE LINK link_orcl  
CONNECT TO system IDENTIFIED BY ***  
USING 'orcl';
```

1. Clone the non-CDB:

```
SQL> CREATE PLUGGABLE DATABASE pdb_orcl  
FROM NON$CDB@link_orcl  
CREATE_FILE_DEST = '.../PDB_orcl';
```

1. Run the noncdb_to_pdb.sql script.

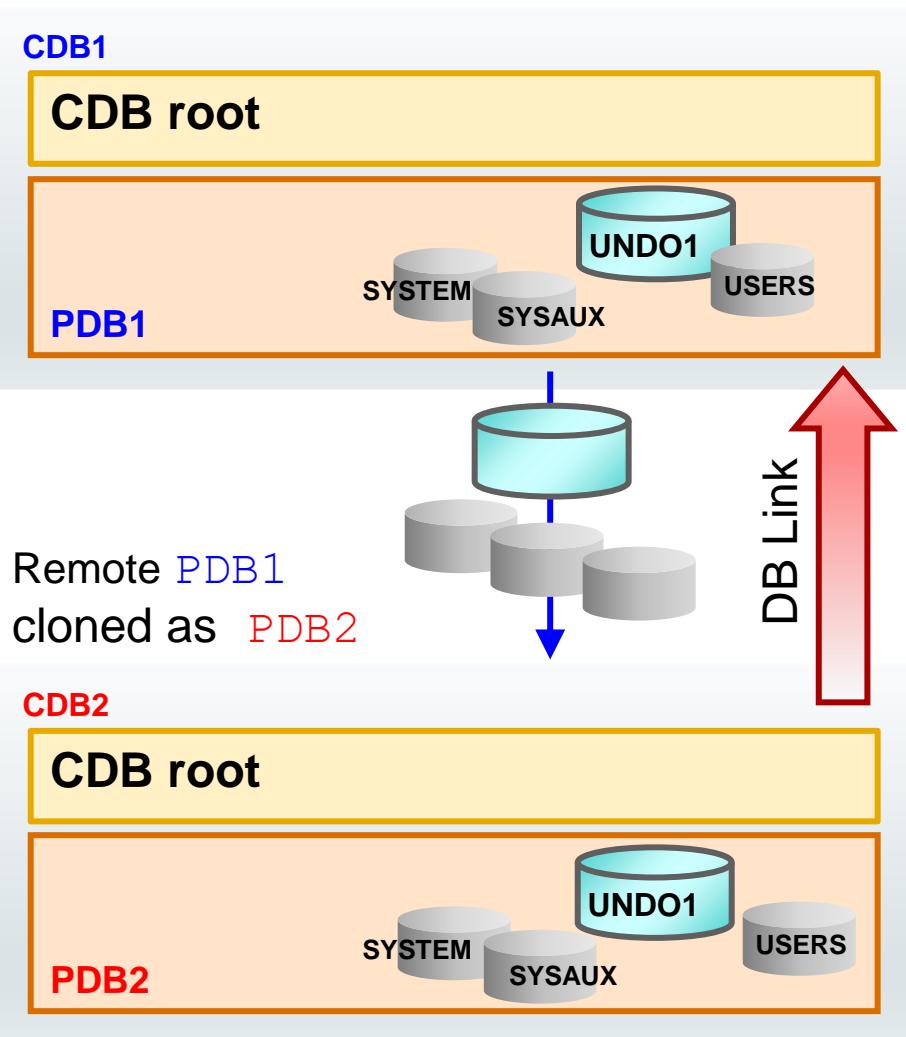
```
SQL> CONNECT sys@pdb_orcl AS SYSDBA  
SQL> @$ORACLE_HOME/rdbms/admin/noncdb_to_pdb
```

1. Open PDB_ORCL in read/write mode.

```
SQL> ALTER PLUGGABLE DATABASE pdb_orcl OPEN;
```

Using DBCA to Clone a Remote PDB

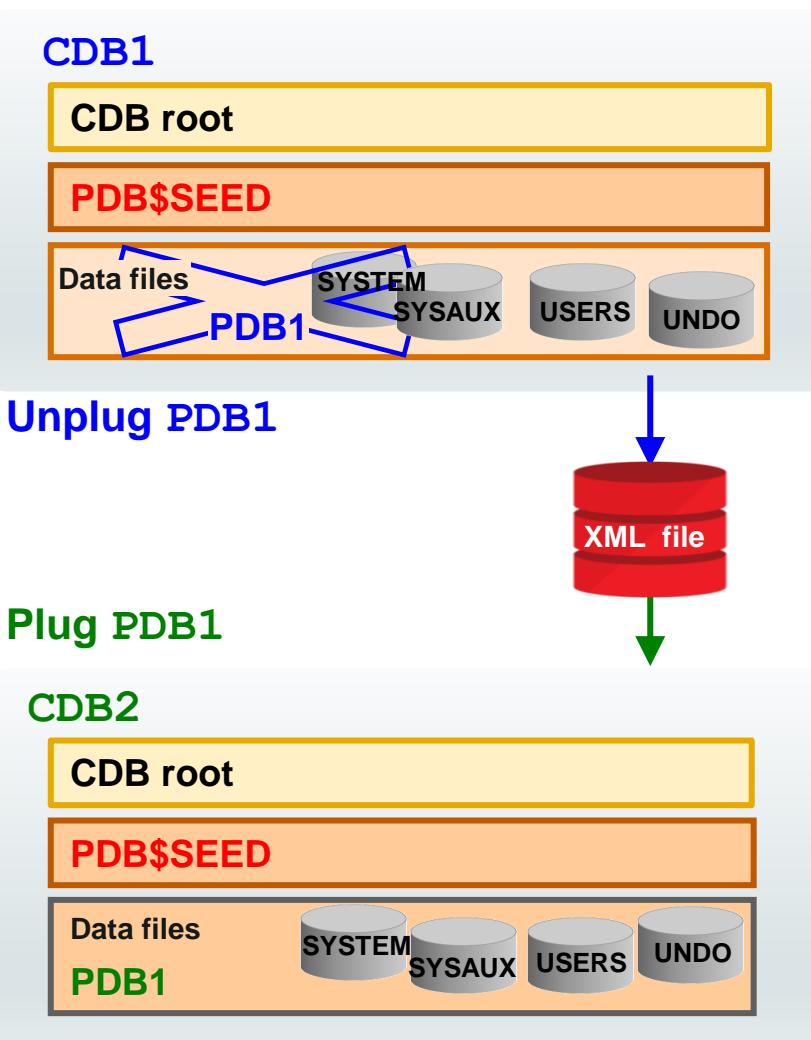
Remote source PDB1



1. Create a common user with privileges in the remote CDB **CDB1**.
2. Use DBCA to clone the remote **PDB1** from **CDB1** to **PDB2**.

```
$ dbca -silent -createPluggableDatabase  
-createFromRemotePDB -remotePdbName PDB1  
-remoteDBConnString CDB1  
-sysDBAUserName system  
-sysDBAPassword password  
-remoteDBSYSDBAUserName SYS  
-remoteDBSYSDBAUserPassword password  
-dbLinkUsername c##remote_user  
-dbLinkUserPassword password  
-sourceDB CDB2 -pdbName PDB2
```

Plugging an Unplugged Regular PDB into CDB



Unplug **PDB1** from **CDB1**:

1. Connect to **CDB1** as a common user.
2. Verify that **PDB1** is closed.

```
SQL> ALTER PLUGGABLE DATABASE pdb1  
UNPLUG INTO 'xmlfile1';
```

3. Drop **PDB1** from **CDB1**.

Plug **PDB1** into **CDB2** :

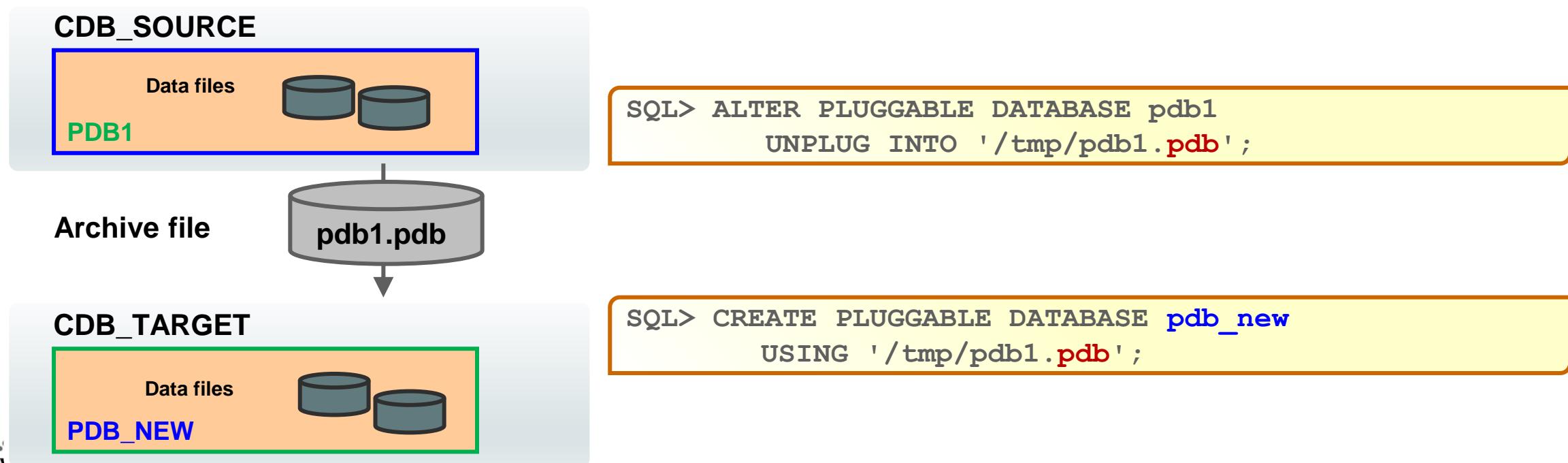
1. Connect to **CDB2** as a common user.
2. Use the `DBMS_PDB` package to check the compatibility of **PDB1** with **CDB2**.

```
SQL> CREATE PLUGGABLE DATABASE pdb1  
USING 'xmlfile1' NOCOPY;
```

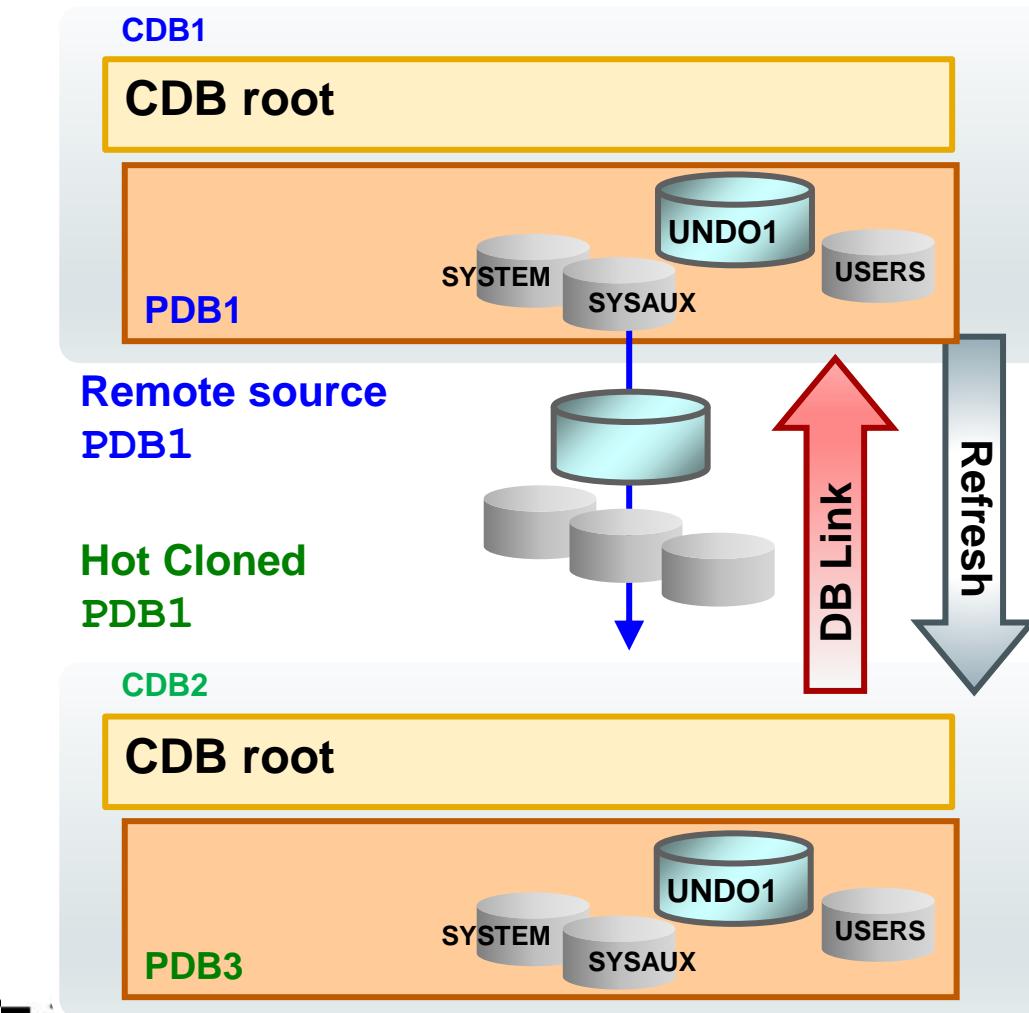
3. Open **PDB1** in read/write mode.

Plugging in a PDB Using an Archive File

1. Unplugging a PDB into a single archive file includes:
 - XML file
 - Data files
2. Plugging in the PDB requires only the archive file.



Cloning Remote PDBs in Hot Mode



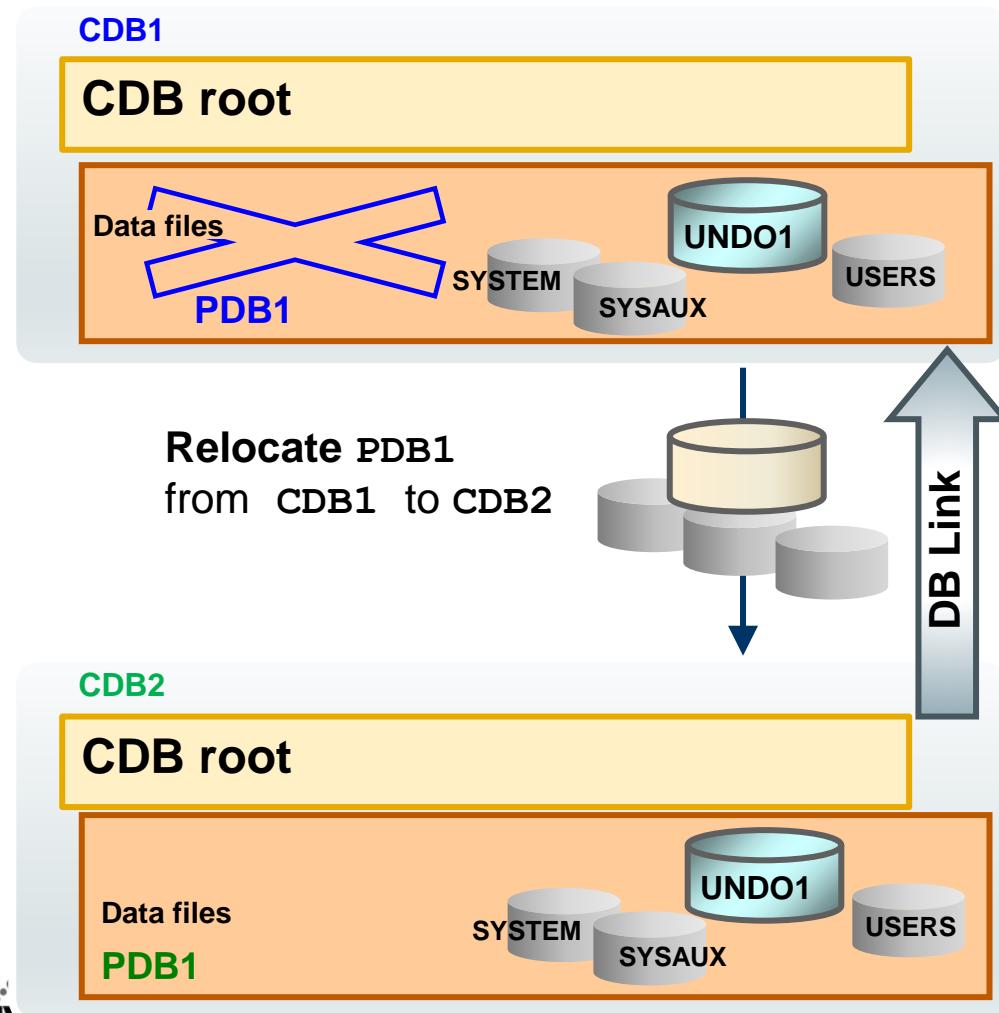
Remote source PDB still up and fully functional:

1. Connect to the target **CDB2** root to create the database link to **CDB1**.
2. Switch the shared undo mode to local undo mode in both the CDBs.
3. Clone the remote **PDB1** to **PDB3**.
4. Open **PDB3** in read-only or read/write mode.

Incremental refreshing:

- Manual
- Automatic (predefined interval)

Near-Zero Downtime PDB Relocation



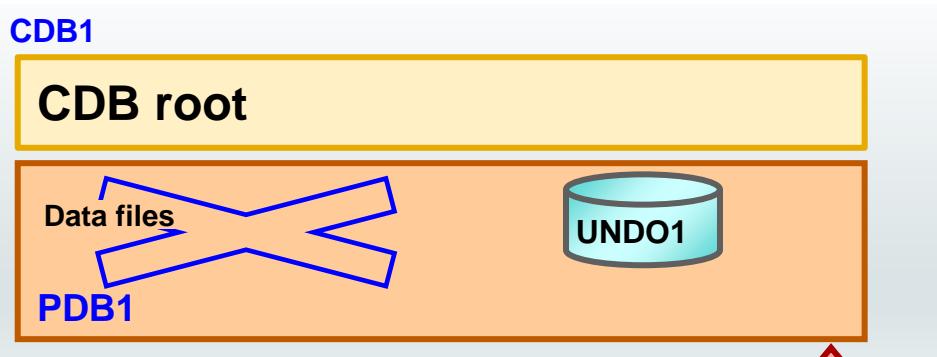
Use a single statement to relocate **PDB1** from **CDB1** into **CDB2**:

1. Switch the shared undo mode to local undo mode in both CDBs.
2. Set ARCHIVELOG mode in both CDBs.
3. Grant SYSOPER to the user connected to **CDB1** via the database link created in **CDB2**.
4. Connect to **CDB2** as a common user to create the database link.
5. Use the CREATE PLUGGABLE DATABASE statement with the RELOCATE clause.
6. Open **PDB1** in read/write mode.

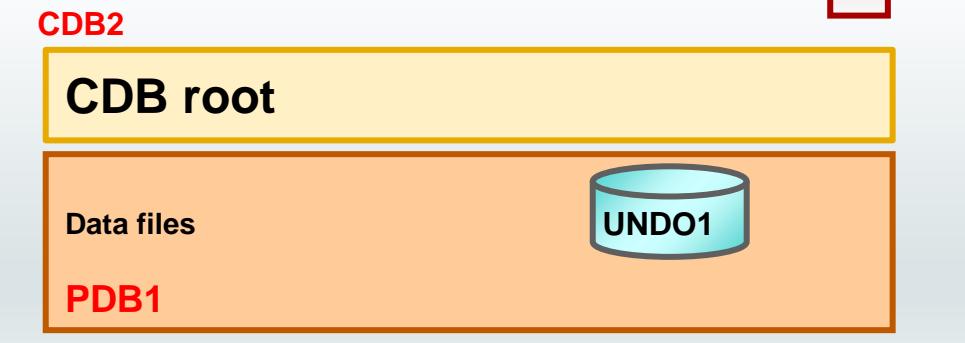
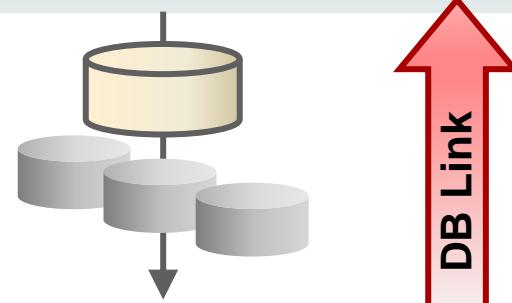
There is no need to:

- Unplug the PDB from the source CDB
- Copy or transfer the data files to a new location
- Plug the PDB in the target CDB
- Drop the source PDB from the source CDB

Using DBCA to Relocate a Remote PDB



Relocate PDB1
from **CDB1**
 to **CDB2**

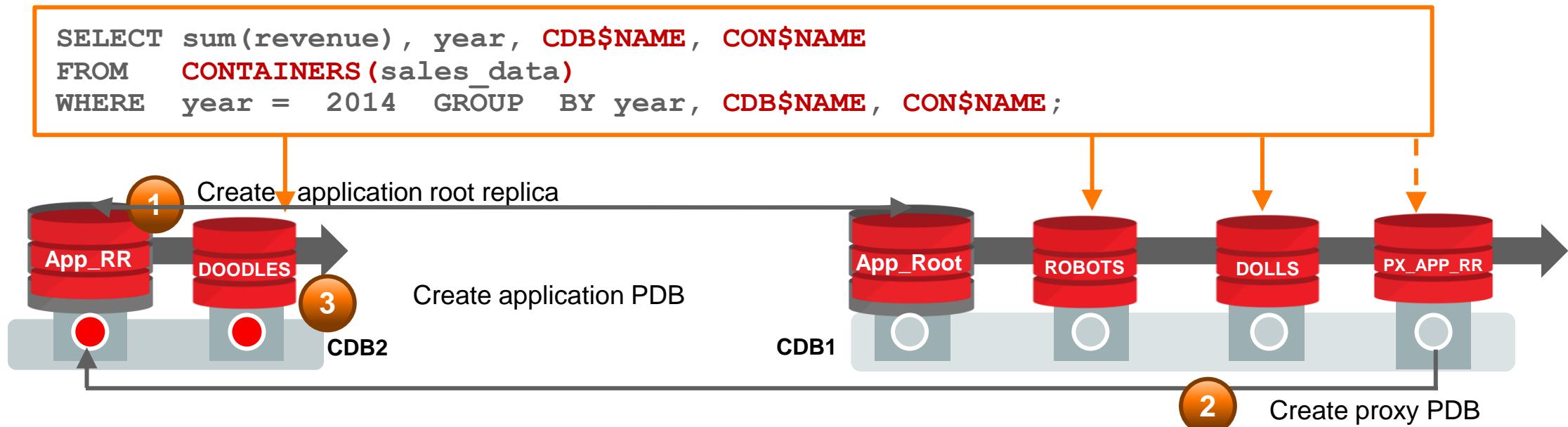


Use DBCA to relocate the remote **PDB1** from **CDB1** into **CDB2**.

```
$ export ORACLE_SID=CDB2
```

```
$ dbca -silent -relocatePDB  
-remotePDBName PDB1 -remoteDBConnectionString CDB1  
-sysDBAUserName system  
-sysDBAPassword password  
-remoteDBSYSDBAUserName SYS  
-remoteDBSYSDBAUserPassword password  
-dbLinkUsername c##remote_user  
-dbLinkUserPassword password  
-sourceDB CDB2 -pdbName PDB1
```

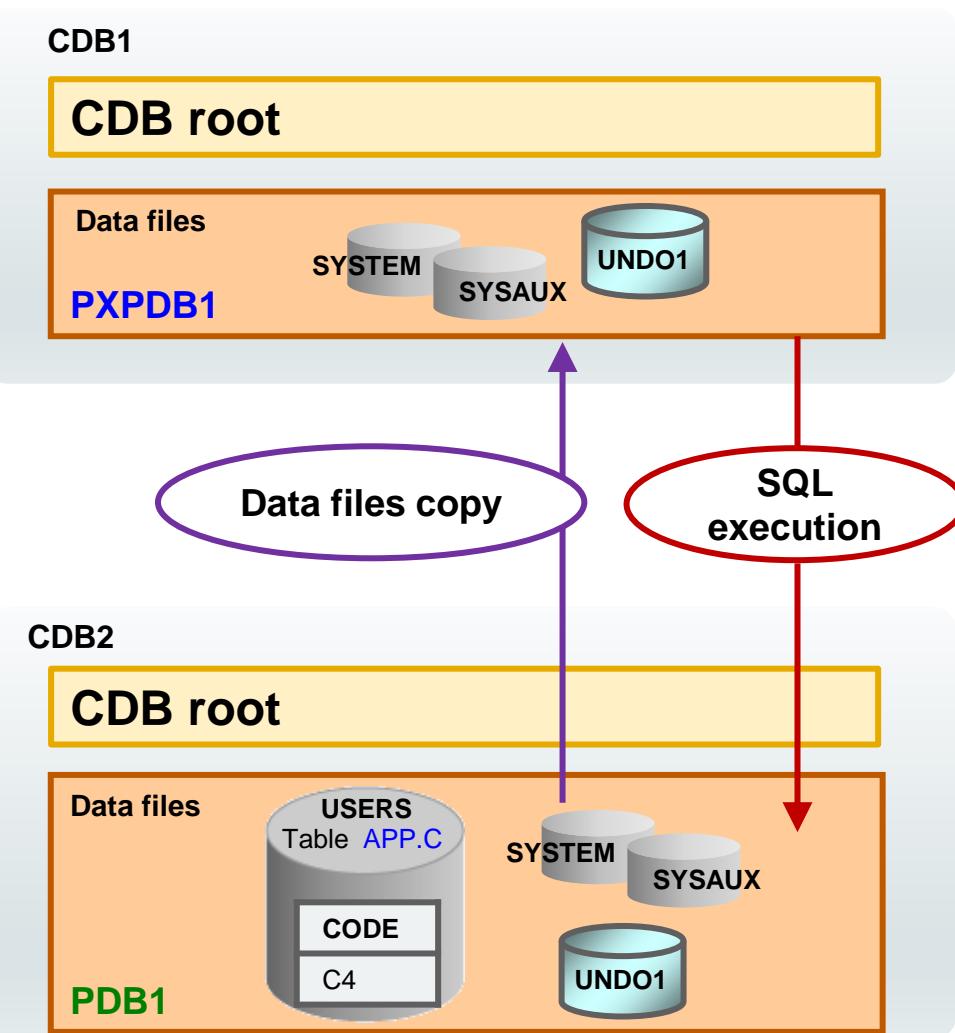
Proxy PDB: Query Across CDBs Proxying Root Replica



- Retrieves rows from the shared table whose data is stored in application PDBs in the application root and replicas in CDBs

Revenue	Year	CDB\$NAME	CON\$NAME
15000000	2014	CDB1	ROBOTS
20000000	2014	CDB2	DOODLES
10000000	2014	CDB1	DOLLS

Creating a Proxy PDB



A proxy PDB allows execution in a proxied PDB.

1. Switch the shared undo mode to local undo mode in both CDBs.
2. Set the ARCHIVELOG mode in both CDBs.
3. Connect to **CDB1** and create a database link (to **CDB2**).
4. Create the **PXPDB1** proxy PDB in **CDB1** as a view referencing the entire proxied **PDB1** in **CDB2**.

```
SQL> CONNECT sys@cdb1 AS SYSDBA
SQL> CREATE PLUGGABLE DATABASE pxpdb1 AS PROXY
      FROM pdb1@link_cdb2;
```

4. Execute all the statements in the **PXPDB1** proxy PDB context to have them executed in the proxied **PDB1** PDB in **CDB2**.

```
SQL> CONNECT sys@pxpdb1 AS SYSDBA
SQL> ALTER PLUGGABLE DATABASE pxpdb1 OPEN;
SQL> SELECT * FROM app.c;
```

Summary

In this lesson, you should have learned how to:

- Clone a regular PDB
- Unplug and plug or clone a non-CDB
- Unplug and plug a regular PDB
- Perform hot cloning
- Perform near-zero downtime PDB relocation
- Create and use a proxy PDB

Practice Overview

- Cloning Remote PDBs in Hot Mode
- Relocating PDBs

14. Managing PDBs

Objectives

After completing this lesson, you should be able to:

- Change the modes and settings of PDBs
- Evaluate the impact of parameter value changes
- Configure host name and port number per PDB
- Drop PDBs

Changing the PDB Mode

After closing a PDB, open it in:

- Restricted read/write mode

```
SQL> CONNECT sys@pdb1 AS SYSDBA  
SQL> ALTER PLUGGABLE DATABASE CLOSE;
```

```
SQL> ALTER PLUGGABLE DATABASE OPEN RESTRICTED;
```

```
SQL> SELECT name, open_mode FROM v$pdbs;  
  
NAME          OPEN_MODE  RES  
-----  
PDB1          READ WRITE YES
```

- Read-only mode

```
SQL> CONNECT / AS SYSDBA  
SQL> ALTER PLUGGABLE DATABASE ALL OPEN READ ONLY;
```

Modifying PDB Settings

- Bring a PDB data file online

```
SQL> ALTER PLUGGABLE DATABASE DATAFILE '/u03/pdb1_01.dbf' ONLINE;
```

- Change the PDB default tablespace

```
SQL> ALTER PLUGGABLE DATABASE DEFAULT TABLESPACE pdb1_tbs;
```

- Change the PDB default temporary tablespace

```
SQL> ALTER PLUGGABLE DATABASE DEFAULT TEMPORARY TABLESPACE temp_tbs;
```

- Set the PDB storage limit

```
SQL> ALTER PLUGGABLE DATABASE STORAGE (MAXSIZE 2G);
```

- Change the global name

```
SQL> ALTER PLUGGABLE DATABASE RENAME GLOBAL_NAME TO pdbAPP1;
```

Impact of Changing Initialization Parameters

- A single server parameter file (SPFILE) per CDB
- PDB value changes:
 - Only when ISPDB_MODIFIABLE=TRUE
 - Loaded in memory after PDB close
 - Stored in dictionary after CDB shutdown

```
SQL> CONNECT sys@pdb1 AS SYSDBA
Connected.
SQL> ALTER SYSTEM SET ddl_lock_timeout=10;
System altered.
SQL> SHOW PARAMETER ddl_lock_timeout
```

NAME	TYPE	VALUE
ddl_lock_timeout	boolean	10

Changing Initialization Parameters: Example

```
SQL> CONNECT sys@pdb2 AS SYSDBA  
  
SQL> ALTER SYSTEM SET ddl_lock_timeout=20 SCOPE=BOTH;  
  
SQL> ALTER PLUGGABLE DATABASE CLOSE;  
SQL> ALTER PLUGGABLE DATABASE OPEN;
```

```
SQL> CONNECT / AS SYSDBA  
SQL> SELECT value, isdpdb_modifiable, con_id FROM v$system_parameter  
      WHERE name = 'ddl_lock_timeout';
```

VALUE	ISPDB	CON_ID
0	TRUE	0
10	TRUE	3
20	TRUE	4

Using the ALTER SYSTEM Command in a PDB

- Some statements change the way a PDB operates:

ALTER SYSTEM Affecting the PDB only	Objects Impacted
ALTER SYSTEM FLUSH SHARED_POOL	Only for objects of the PDB
ALTER SYSTEM FLUSH BUFFER_CACHE	Only for buffers of the PDB
ALTER SYSTEM ENABLE/DISABLE RESTRICTED SESSION	Only for sessions of the PDB
ALTER SYSTEM KILL SESSION	Only for sessions of the PDB
ALTER SYSTEM SET <i>parameter</i>	Only for parameter of the PDB

- Some ALTER SYSTEM statements can be executed in a PDB but affect the whole CDB:

ALTER SYSTEM CHECKPOINT	Affects all data files except those in read only or offline
-------------------------	-------------------------------------------------------------

- All other ALTER SYSTEM statements affect the entire CDB and must be executed by a common user in the CDB root.

ALTER SYSTEM SWITCH LOGFILE	Operation not allowed from within a pluggable database
-----------------------------	--------------------------------------------------------

Configuring Host Name and Port Number per PDB

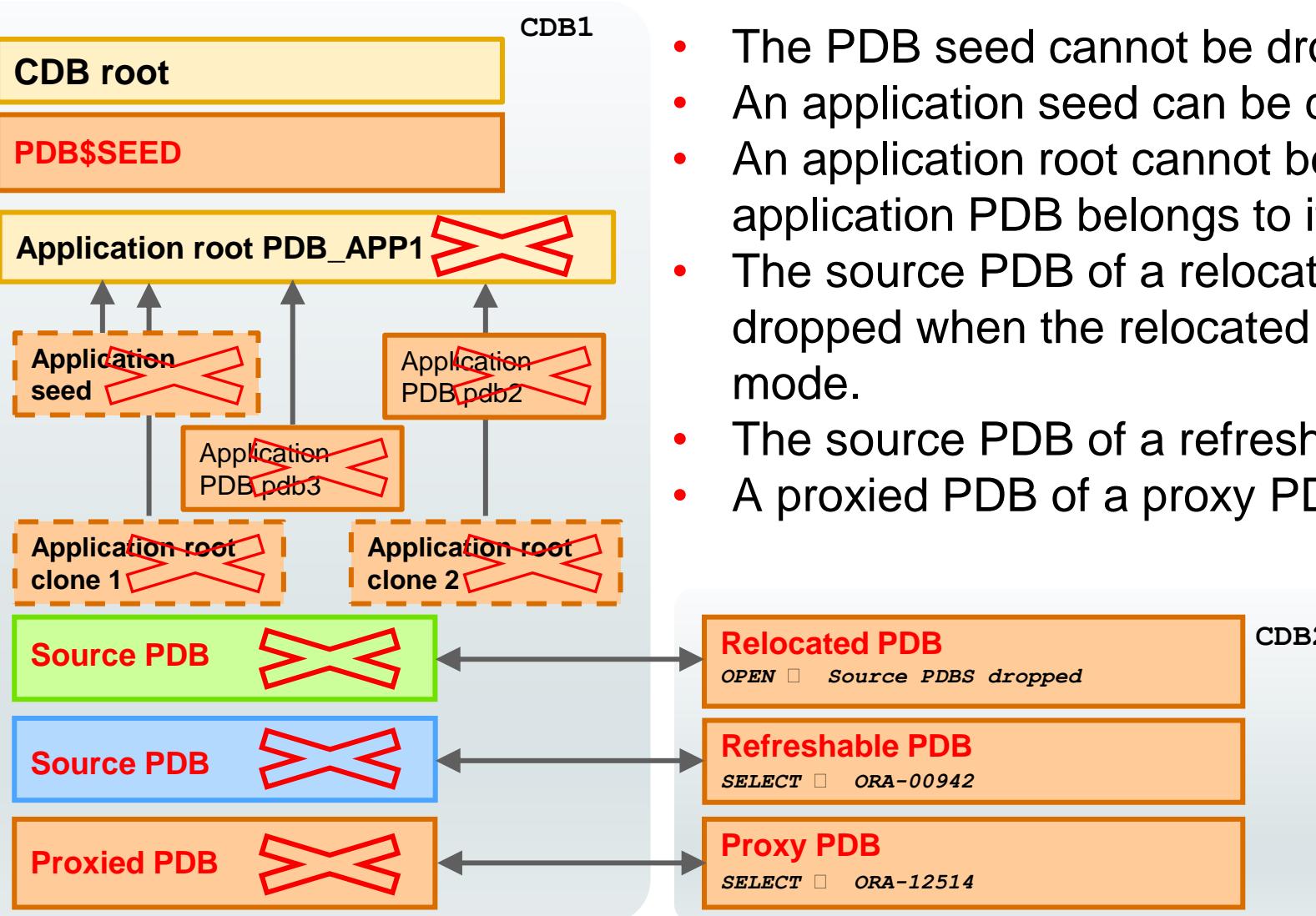
- The host name and port number settings for a PDB are important only if proxy PDBs will reference the PDB.

```
SQL> ALTER PLUGGABLE DATABASE CONTAINERS HOST = <host_name>;  
SQL> ALTER PLUGGABLE DATABASE CONTAINERS PORT = <port_nb>;
```

- The host name and port number can be reset to their default:

```
SQL> ALTER PLUGGABLE DATABASE CONTAINERS HOST RESET;  
SQL> ALTER PLUGGABLE DATABASE CONTAINERS PORT RESET;
```

Dropping PDBs



- The PDB seed cannot be dropped.
- An application seed can be dropped.
- An application root cannot be dropped as long as an application PDB belongs to it.
- The source PDB of a relocated PDB is automatically dropped when the relocated PDB is opened in RW mode.
- The source PDB of a refreshable PDB can be dropped.
- A proxied PDB of a proxy PDB can be dropped.

The **DROP** operation updates controlfiles:

1. Removes PDB data files
2. Retains data files (default)

Summary

In this lesson, you should have learned how to:

- Change the different modes and settings of PDBs
- Evaluate the impact of parameter value changes
- Configure host name and port number per PDB
- Drop PDBs

Practice Overview

- Renaming a PDB
- Setting Parameter Values for PDBs

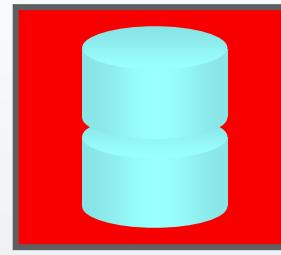
15. Database Storage Overview

Objectives

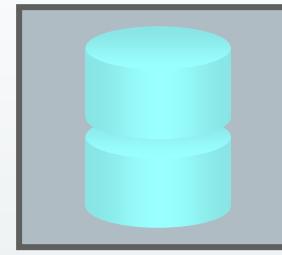
After completing this lesson, you should be able to:

- Describe logical and physical storage structures in an Oracle database
- Describe the purpose of each of the default tablespaces
- Describe the storage of data in blocks
- List the advantages of deferred segment creation

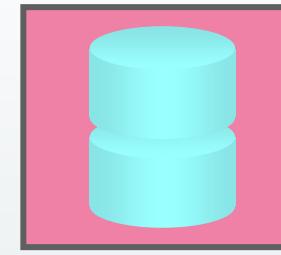
Database Storage Architecture



Control files



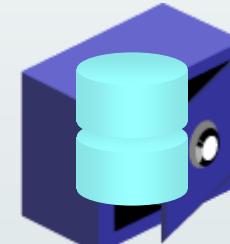
Data files



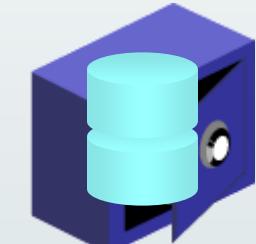
Online redo log files



Initialization
parameter file



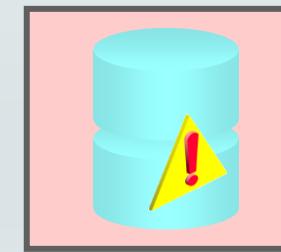
Backup files



Archived redo
log files

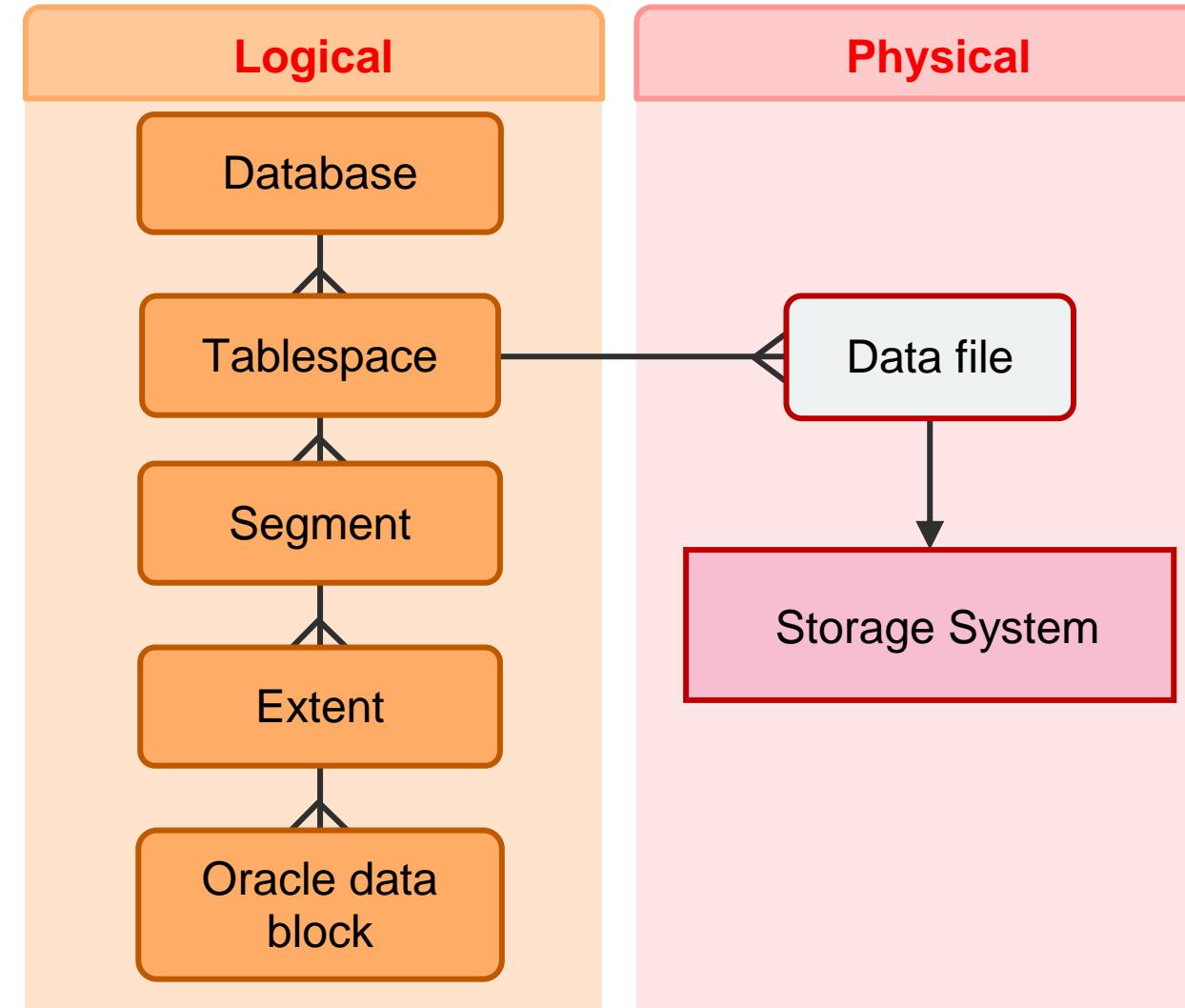


Password file



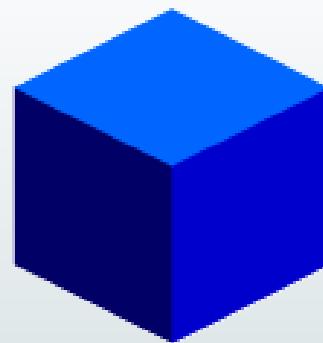
Log and trace
files

Logical and Physical Database Structures



Segments, Extents, and Blocks

- Segments exist in a tablespace.
- Segments are collections of extents.
- Extents are collections of data/undo blocks.
- Data/undo blocks are mapped to disk blocks.



Segment



Extents

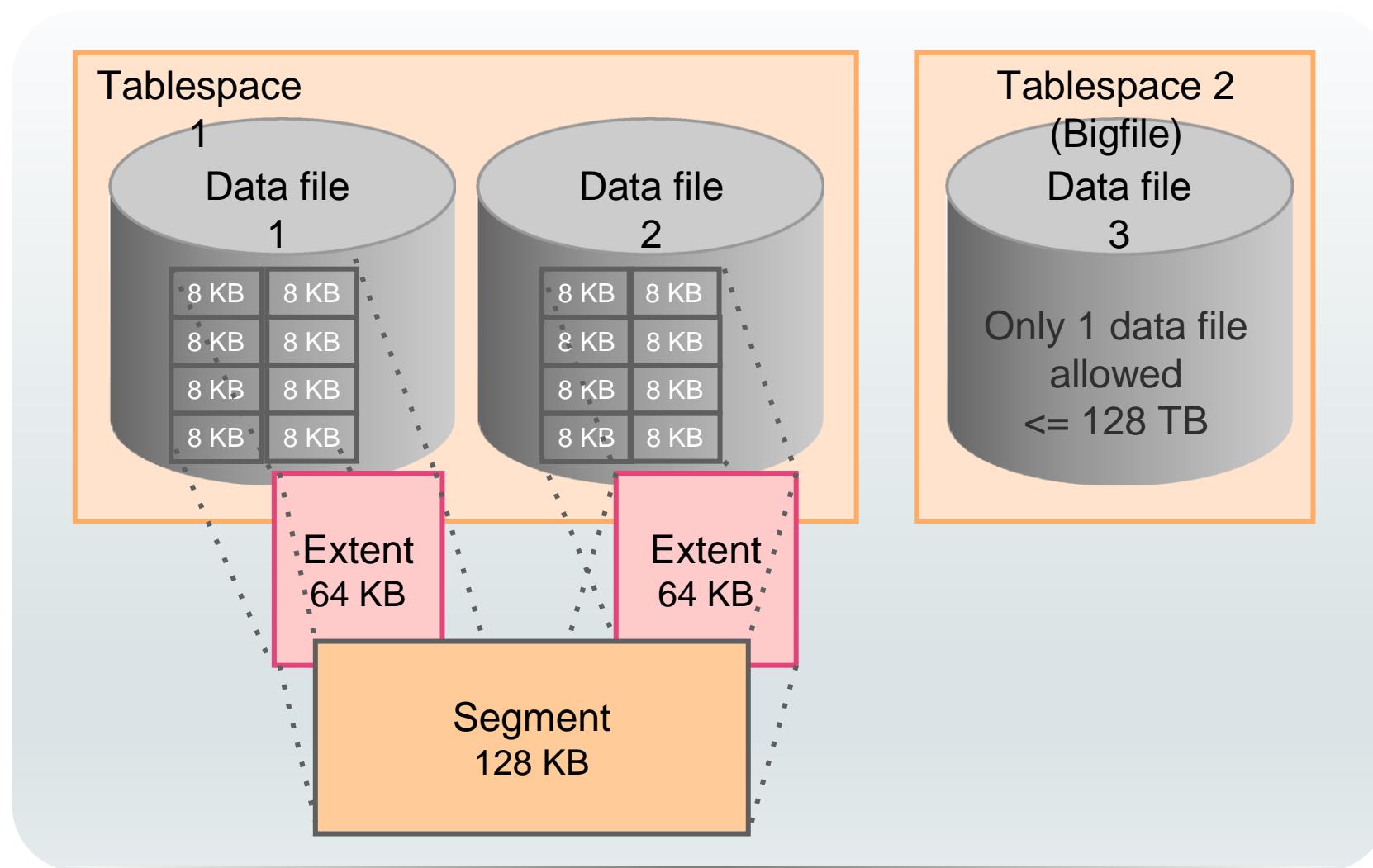


Data/undo
blocks



Disk blocks
(File System
Storage)

Tablespaces and Data Files



Default Tablespaces in a Multitenant Container Database

Tablespace	Description
SYSTEM	The SYSTEM tablespace is used for core functionality. In the root container, it contains Oracle-supplied metadata. In a PDB, it contains user metadata.
SYSAUX	The SYSAUX tablespace is an auxiliary tablespace to the SYSTEM tablespace and helps reduce the load on the SYSTEM tablespace. It exists in the root container and in each PDB.
TEMP	The TEMP tablespace contains schema objects only for a session's duration. There is one default temporary tablespace for the CDB root and one for each application root, application PDB, and PDB.
UNDO	The UNDO tablespace stores the data needed to roll back, or undo, changes to the database. One active undo tablespace exists in the root container. It is recommended that you have a local undo tablespace in each PDB.
USERS	The USERS tablespace stores user objects and data. It is created by default in the root container only.

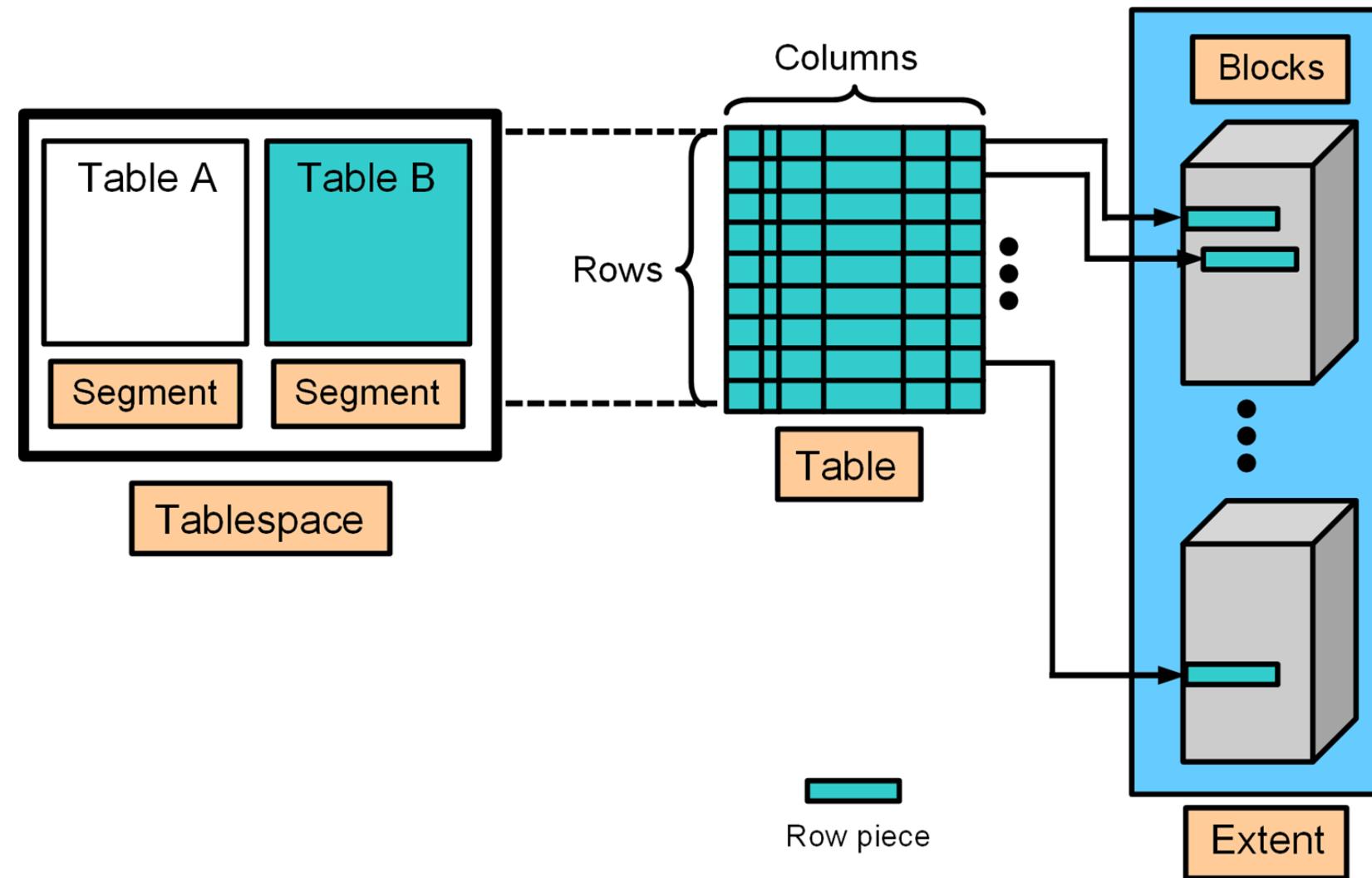
SYSTEM and SYSAUX Tablespaces

- 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.

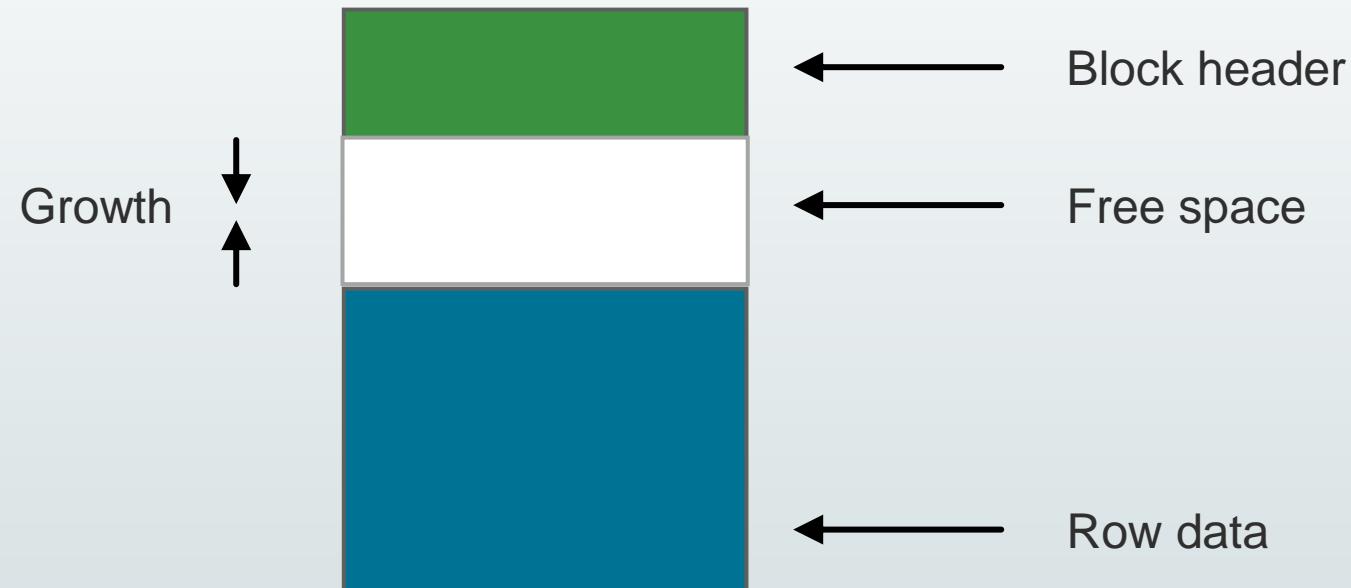
Types of Segments

- A segment is a set of extents allocated for a certain logical structure.
- The different types of segments include:
 - Table and cluster
 - Index
 - Undo
 - Temporary
- Segments are dynamically allocated by the Oracle Database server.

How Table Data Is Stored

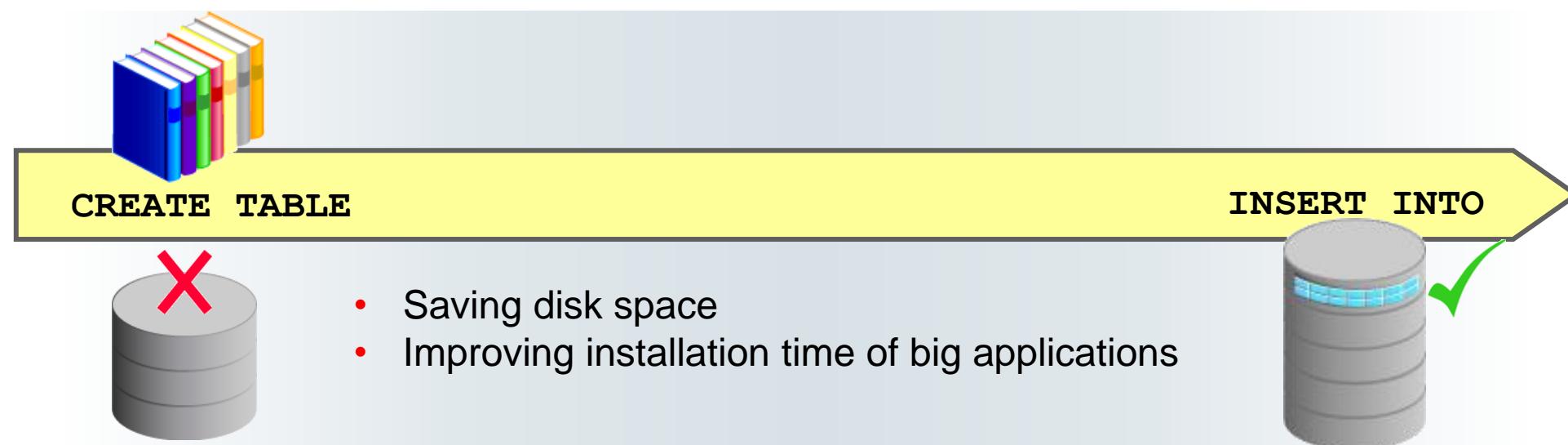


Database Block Content



Understanding Deferred Segment Creation

- DEFERRED_SEGMENT_CREATION = TRUE is the default.
- Deferred segment is the default for tables, indexes, and partitions.
- Segment creation takes place as follows:
 - Table creation > Data dictionary operation
 - DML > Segment creation



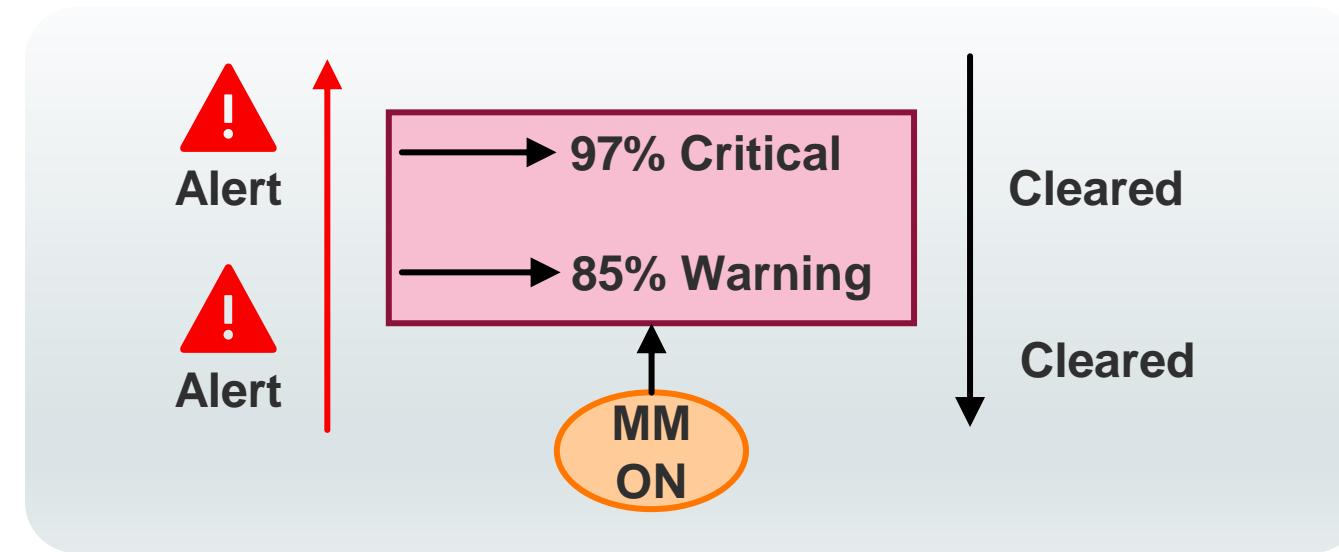
- Saving disk space
- Improving installation time of big applications

Controlling Deferred Segment Creation

- With the DEFERRED_SEGMENT_CREATION parameter:
 - Initialization parameter file
 - ALTER SESSION command
 - ALTER SYSTEM command
- With the SEGMENT CREATION clause:
 - IMMEDIATE
 - DEFERRED (default)

```
CREATE TABLE SEG_TAB3(C1 number, C2 number)
SEGMENT CREATION IMMEDIATE TABLESPACE SEG_TBS;
CREATE TABLE SEG_TAB4(C1 number, C2 number)
SEGMENT CREATION DEFERRED;
```

Monitoring Tablespace Space Usage



- Read-only and offline tablespaces: Do not set up alerts.
- Temporary tablespace: Threshold corresponds to space currently used by sessions.
- Undo tablespace: Threshold corresponds to space used by active and unexpired extents.
- Auto-extensible files: Threshold is based on the maximum file size.

Oracle Database Storage Structures: Interactive Architecture Diagram

Access the Interactive Architecture Diagram on the Oracle Help Center Oracle Database “What’s New” page.

<https://tinyurl.com/yyepn9ma>

Summary

In this lesson, you should have learned how to:

- Describe logical and physical storage structures in an Oracle database
- Describe the purpose of each of the default tablespaces
- Describe the storage of data in blocks
- List the advantages of deferred segment creation

16. Creating and Managing Tablespaces

Objectives

After completing this lesson, you should be able to:

- Create, alter, and drop tablespaces
- View tablespace information
- Implement Oracle Managed Files (OMF)
- Move and rename online data files

Creating Tablespaces

- A tablespace is an allocation of space in the database that can contain schema objects.
- Create a tablespace with the `CREATE TABLESPACE` statement or use a graphical tool.
- You can create three types of tablespaces:
 - Permanent tablespace: Contains persistent schema objects. Objects in permanent tablespaces are stored in data files.
 - Undo tablespace: Is a type of permanent tablespace used by Oracle Database to manage undo data in automatic undo management mode
 - Temporary tablespace: Contains schema objects only for the duration of a session. Objects in temporary tablespaces are stored in temp files.

Creating a Tablespace: Clauses

Include one or more of the following clauses to define various aspects of the tablespace:

Clause	Description
DATAFILE or TEMPFILE	Used to specify the name, location, and initial size of the data file or temp file
ONLINE or OFFLINE	Used to make the tablespace available (or not available) immediately after creation
BLOCKSIZE	Used to specify a nonstandard block size
EXTENT MANAGEMENT	Used to specify how the extents of the tablespace will be managed and where the metadata for allocated and unallocated extents is to be stored
LOGGING	Used to specify the default logging attributes of objects in the tablespace
SEGMENT MANAGEMENT	Used to specify how free space in the segments in the tablespace should be tracked (bitmaps or free lists)

Creating Permanent Tablespaces in a CDB

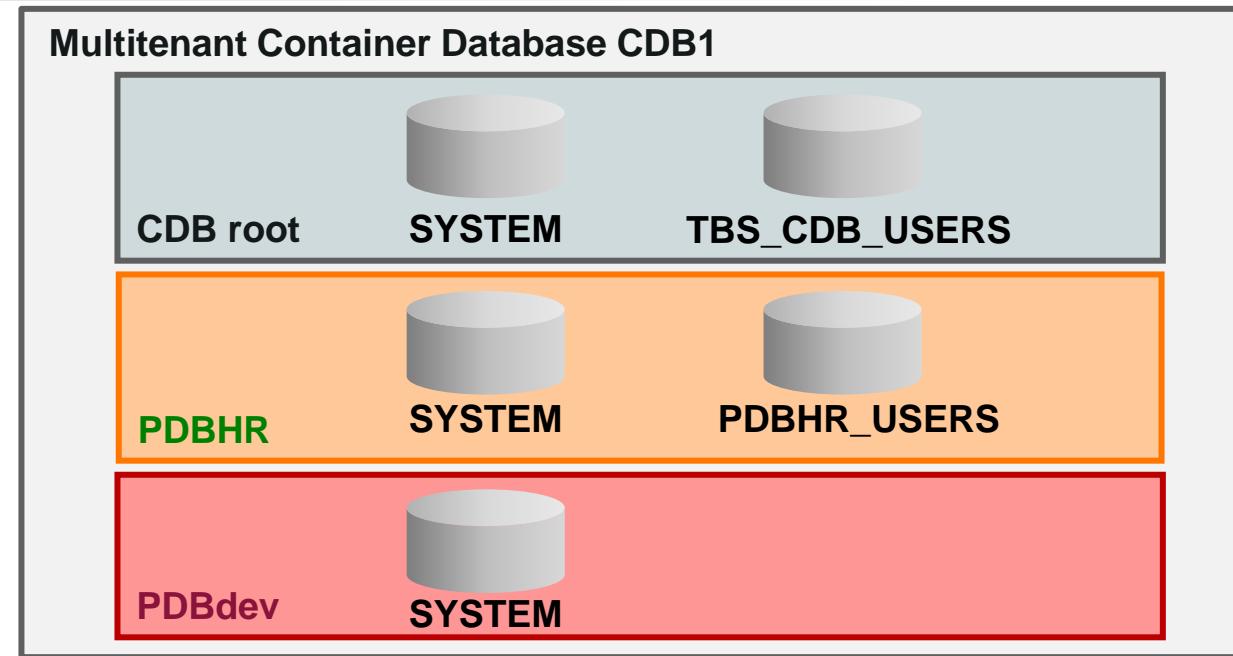
- Tablespace creation during CDB creation:
 - With DBCA: USERS tablespace created in the CDB root
 - With CREATE DATABASE statement with USER_DATA TABLESPACE clause: Your defined tablespace created in the CDB root
- Create a permanent tablespace in the CDB root:

```
SQL> CONNECT system@cdb1
SQL> CREATE TABLESPACE tbs_CDB_users
      DATAFILE '/u1/app/oracle/oradata/cdb/cdb_users01.dbf' SIZE 100M;
```

- Create a permanent tablespace in a PDB:

```
SQL> CONNECT system@PDB1
SQL> CREATE TABLESPACE tbs_PDB1_users
      DATAFILE '/u1/app/oracle/oradata/cdb/pdb1/users01.dbf' SIZE 100M;
```

Defining Default Permanent Tablespaces



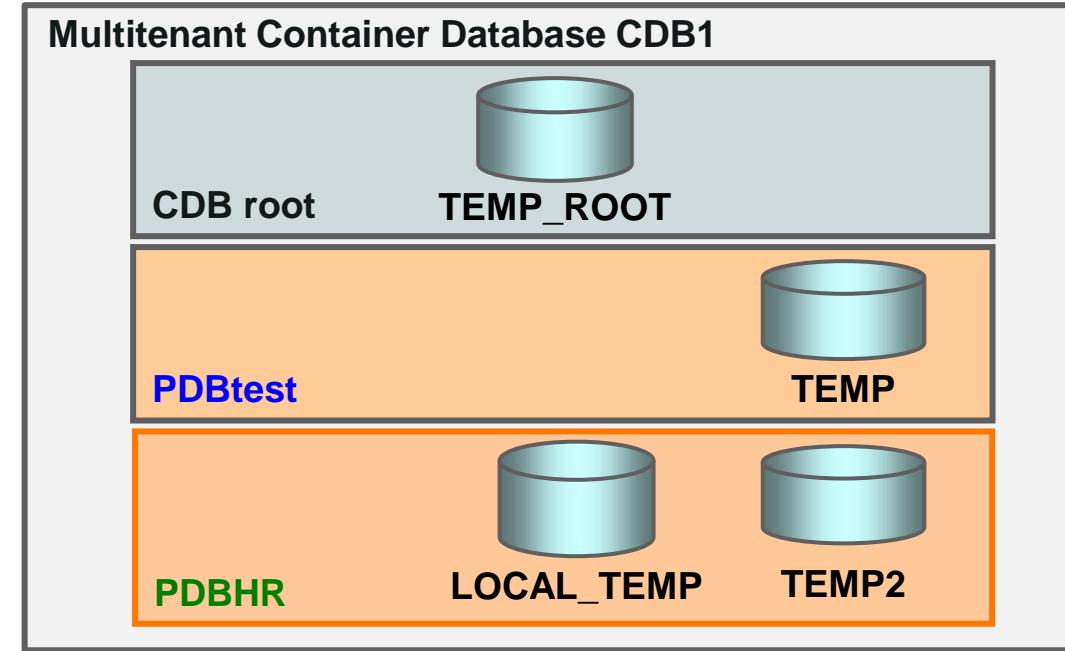
- In the CDB

```
SQL> CONNECT system@cdb1
SQL> ALTER DATABASE DEFAULT TABLESPACE tbs_CDB_users;
```

- In the PDB

```
SQL> CONNECT pdb1_admin@pdbhr
SQL> ALTER PLUGGABLE DATABASE DEFAULT TABLESPACE pdbhr_users;
```

Temporary Tablespaces



- Only one default temporary tablespace or tablespace group is allowed per CDB or PDB.
- Each PDB can have temporary tablespaces or tablespace groups.
- Define the default temporary tablespace in a PDB:

```
SQL> CONNECT pdb1_admin@pdbhr
SQL> ALTER DATABASE DEFAULT TEMPORARY TABLESPACE local_temp;
```

Altering and Dropping Tablespaces

- When you create a tablespace, it is initially a read/write tablespace.
- Use the `ALTER TABLESPACE` statement to take a tablespace offline or online, add data files or temp files to it, or make it a read-only tablespace.
- A tablespace can be in one of three different statuses or states:
 - Read Write
 - Read Only
 - Offline with one of the following options:
 - NORMAL
 - TEMPORARY
 - IMMEDIATE
- Add space to an existing tablespace by either adding data files to the tablespace or changing the size of an existing data file.
- Use the `DROP TABLESPACE` statement to drop a tablespace and its contents from the database if you no longer need its content.

Viewing Tablespace Information

Tablespace and data file information can be obtained by querying the following views:

- Tablespace information:
 - CDB_TABLESPACES and DBA_TABLESPACES
 - V\$TABLESPACE
- Data file information:
 - CDB_DATA_FILES and DBA_DATA_FILES
 - V\$DATAFILE
- Temp file information:
 - CDB_TEMP_FILES and DBA_TEMP_FILES
 - V\$TEMPFILE
- Tables in a tablespace:
 - ALL_TABLES

Implementing Oracle Managed Files (OMF)

- Specify file operations in terms of database objects rather than file names.

Parameter	Description
DB_CREATE_FILE_DEST	Defines the location of the default file system directory for data files and temporary files
DB_CREATE_ONLINE_LOG_DEST_n	Defines the location for redo log files and control file creation
DB_RECOVERY_FILE_DEST	Gives the default location for the fast recovery area

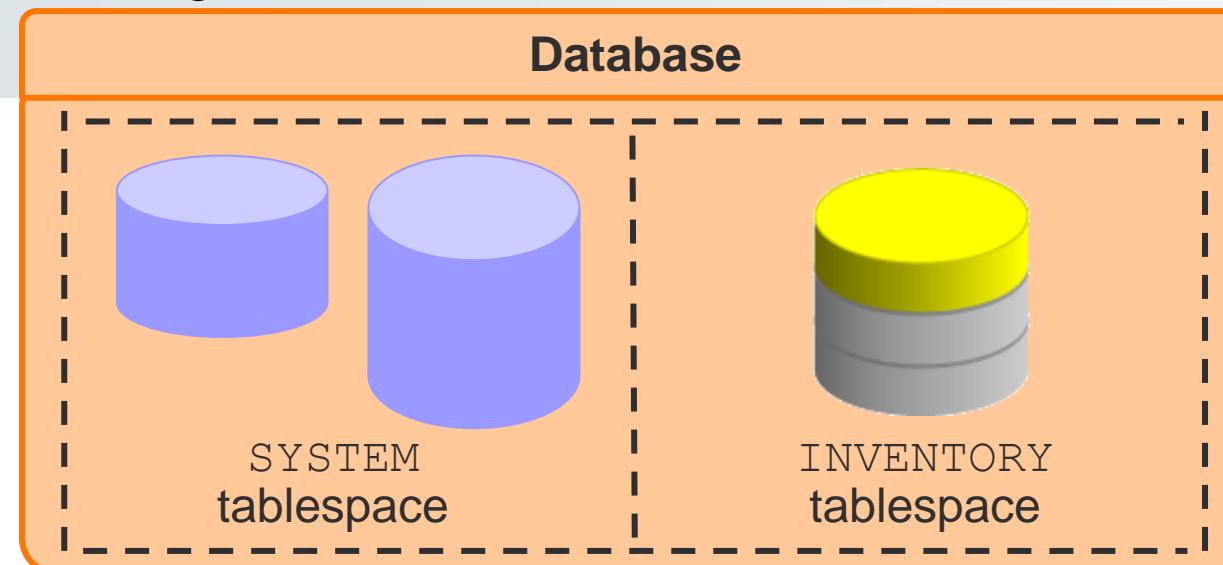
- Example:

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST='/u01/app/oracle/oradata';
SQL> CREATE TABLESPACE tbs_1;
```

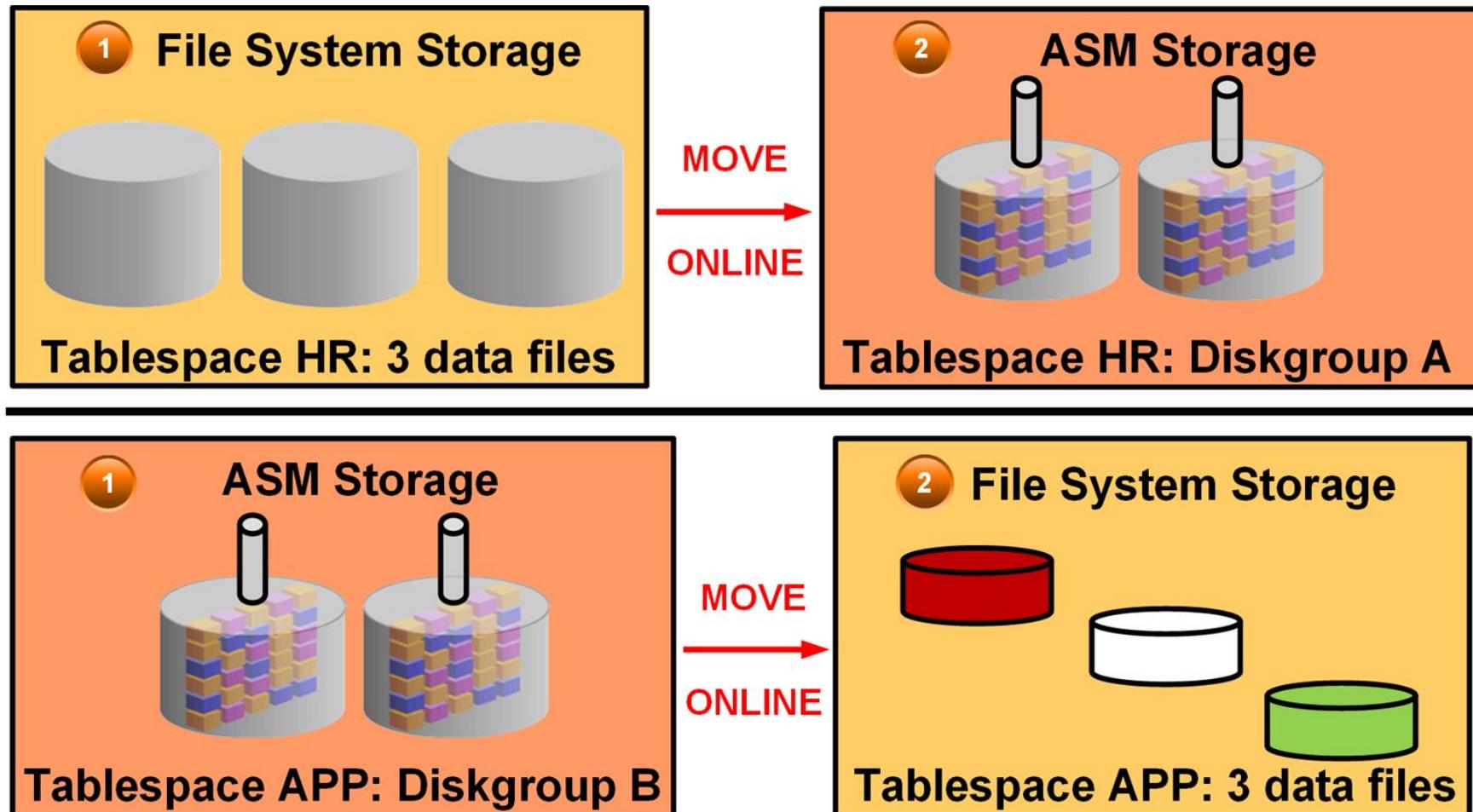
Enlarging the Database

You can enlarge the database in the following ways:

- Create a new tablespace.
- Add a data file to an existing smallfile tablespace.
- Increase the size of a data file.
- Provide for the dynamic growth of a data file.



Moving or Renaming Online Data Files



Examples: Moving and Renaming Online Data Files

- Relocating an online data file:

```
SQL> ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf'  
2 TO '/disk2/myexample01.dbf' ;
```

- Copying a data file from a file system to Automatic Storage Management (ASM):

```
SQL> ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf'  
2 TO '+DiskGroup2' KEEP;
```

- Renaming an online data file:

```
SQL> ALTER DATABASE MOVE DATAFILE '/disk1/myexample01.dbf'  
2 TO '/disk1/myexample02.dbf' ;
```

Summary

In this lesson, you should have learned how to:

- Create, alter, and drop tablespaces
- View tablespace information
- Implement Oracle Managed Files (OMF)
- Move and rename online data files

Practice Overview

- Viewing Tablespace Information
- Creating a Tablespace
- Managing Temporary and Permanent Tablespaces

17. Improving Space Usage

Objectives

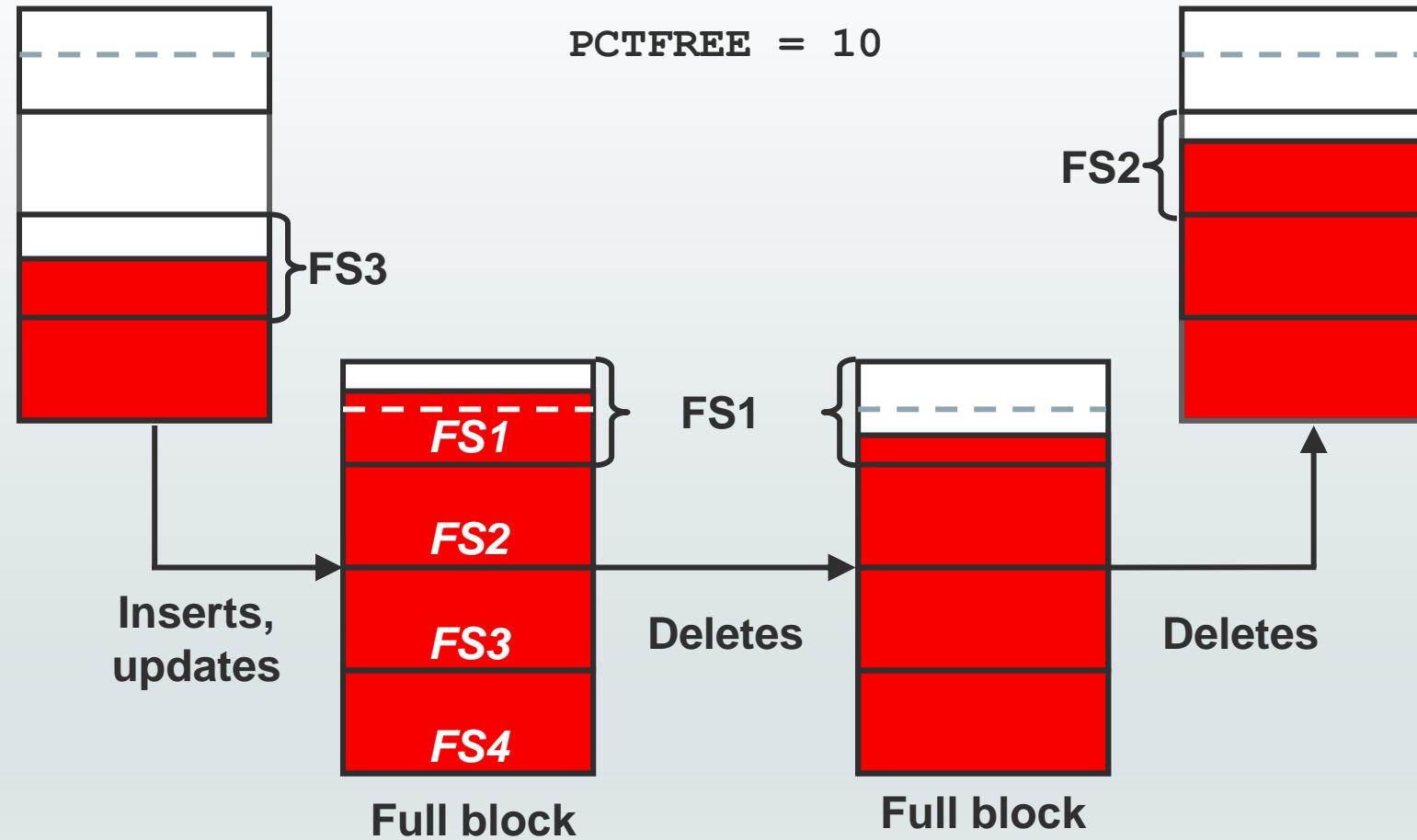
After completing this lesson, you should be able to:

- Describe and use Oracle Database features that save space
- Create private temporary tables
- Save space by using compression
- Reclaim wasted space from tables and indexes by using the segment shrink functionality
- Manage resumable space allocation

Space Management Features

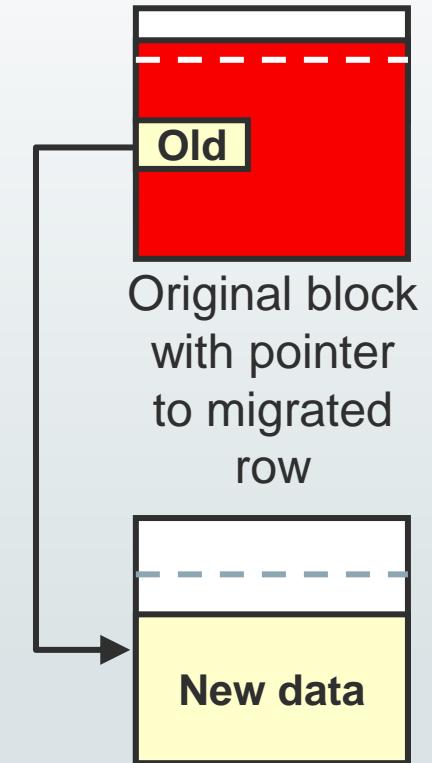
- Space is automatically managed by the Oracle Database server. It generates alerts about potential problems and recommends possible solutions.
- Space management features include:
 - Oracle Managed Files (OMF)
 - Free-space management with bitmaps (“locally managed”) and automatic data file extension
 - Proactive space management (default thresholds and server-generated alerts)
 - Space reclamation (shrinking segments, online table redefinition)
 - Capacity planning (growth reports)

Block Space Management

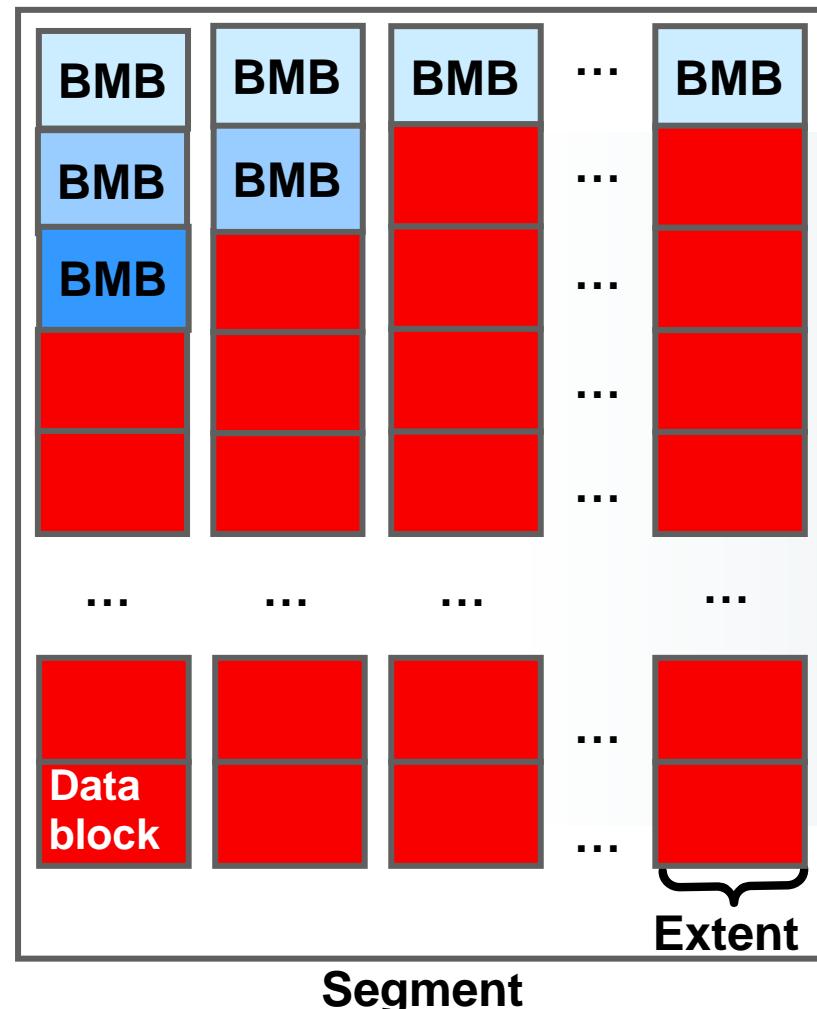


Row Chaining and Migration

- On update: Row length increases, exceeding the available free space in the block.
- Data needs to be stored in a new block.
- Original physical identifier of row (ROWID) is preserved.
- The Oracle Database server needs to read two blocks to retrieve data.
- Segment Advisor finds segments containing the migrated rows.
- There is automatic coalescing of fragmented free space inside the block.



Free Space Management Within Segments



- Tracked by bitmaps in segments
- Benefits:
 - More flexible space utilization
 - Runtime adjustment
 - Multiple process search of bitmap blocks (BMBs)

Allocating Extents

- Searching the data file's bitmap for the required number of adjacent free blocks
- Sizing extents with storage clauses:
 - UNIFORM
 - AUTOALLOCATE
- Viewing the extent map
- Obtaining deallocation advice

Using Unusable Indexes

- Consider using unusable indexes to improve the performance of bulk loads.
- Unusable indexes are ignored by the optimizer.
- When an unusable index is created, no segment is created:

```
CREATE INDEX test_i1 ON seg_test(c) UNUSABLE
```

- When an existing index is altered to unusable, the segment is dropped:

```
ALTER INDEX test_i UNUSABLE
```

- An unusable index can be rebuilt to make it valid again:

```
ALTER INDEX test_i REBUILD
```

Using Temporary Tables

- Temporary tables contain data for the duration of a transaction or session.
- Types of temporary tables:
 - Global: Table definition is visible to all sessions; content is specific to a session.
 - Private: Table definition is visible only to the creating session.
- Segment for a temporary table is allocated with the first `INSERT` or `CREATE TABLE AS SELECT` statement.
- Table definition persists after a `ROLLBACK`.
- Transaction-specific temporary table can only be used by one transaction at a time.

Creating Global Temporary Tables

- Create a global temporary table by using the CREATE GLOBAL TEMPORARY TABLE statement.
- Specify whether the global temporary table applies to a transaction or session by using the ON COMMIT clause:
 - Transaction-specific (default): ON COMMIT DELETE ROWS
 - Session-specific: ON COMMIT PRESERVE ROWS
- Example:

```
SQL> CREATE GLOBAL TEMPORARY TABLE trans_buff_area(date1 DATE,...)
      > ON COMMIT DELETE ROWS;
```

Creating Private Temporary Tables

- Create a private temporary table by using the CREATE PRIVATE TEMPORARY TABLE statement.
- The table name must start with ORA\$PTT_ :

PRIVATE_TEMP_TABLE_PREFIX = ORA\$PTT_

```
SQL> CREATE PRIVATE TEMPORARY TABLE ORA$PTT_mine (c1 DATE, ... c3 NUMBER(10,2));
```

- The CREATE PRIVATE TEMPORARY TABLE statement does not commit a transaction.
- Two concurrent sessions may have a private temporary table with the same name but different shape.
- Private temporary table definition and contents are automatically dropped at the end of a session or transaction.

```
SQL> CREATE PRIVATE TEMPORARY TABLE ORA$PTT_mine (c1 DATE ...)
      ON COMMIT PRESERVE DEFINITION;
```

```
SQL> DROP TABLE ORA$PTT_mine;
```

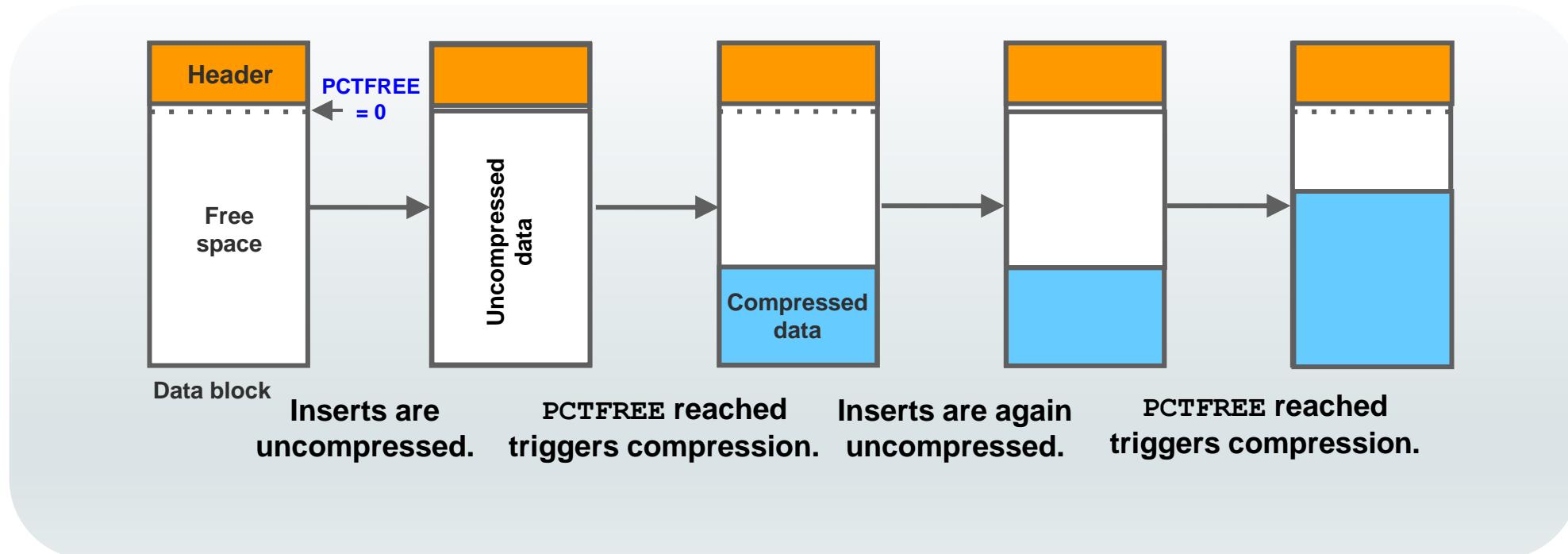
Table Compression: Overview

Reducing storage costs by compressing all data:

- Basic compression for direct-path insert operations: 10x
- Advanced row compression for all DML operations: 2–4x

Compression Method	Compression Ratio	CPU Overhead	CREATE and ALTER TABLE Syntax	Typical Applications
Basic table compression	High	Minimal	COMPRESS or ROW STORE COMPRESS BASIC	DSS
Advanced row compression	High	Minimal	ROW STORE COMPRESS ADVANCED	OLTP, DSS

Table Compression: Concepts



Compression for Direct-Path Insert Operations

- Is enabled with CREATE TABLE ... COMPRESS BASIC
- Is recommended for bulk loading data warehouses
- Maximizes contiguous free space in blocks

Advanced Row Compression for DML Operations

- Is enabled with CREATE TABLE ... ROW STORE COMPRESS ADVANCED
- Is recommended for active OLTP environments

Y	Y	Y	Y	Y
G	Y	Y	G	
G	Y	Y	Y	G

Uncompressed block

G	Y			
	Y	Y	Y	Y
G	Y	G		
G	Y	Y	Y	G

OLTP compression with the symbol table at the beginning of the block

Specifying Table Compression

- You can specify table compression for:
 - An entire heap-organized table
 - A partitioned table (each partition can have a different type or level of compression)
 - The storage of a nested table
- You cannot:
 - Specify basic and advanced row compression on tables with more than 255 columns
 - Drop a column if a table is compressed for direct loads, but you can drop it if the table is advance row compressed

Using the Compression Advisor

- Analyzes objects to give an estimate of space savings for different compression methods
- Helps in deciding the correct compression level for an application
- Recommends various strategies for compression
 - Picks the right compression algorithm for a particular data set
 - Sorts on a particular column for increasing the compression ratio
 - Presents tradeoffs between different compression algorithms

Resolving Space Usage Issues

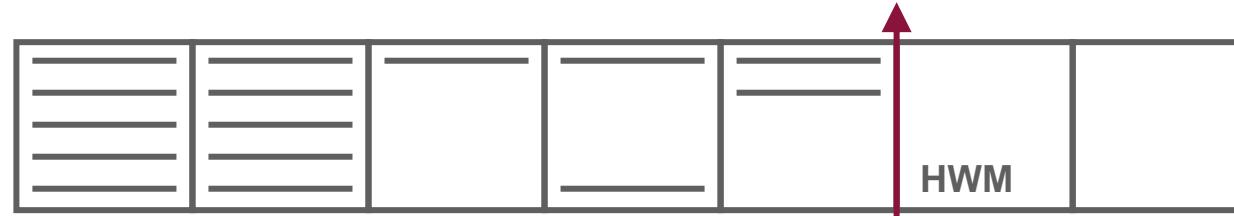


- Resolve space usage issues by:
 - Adding or resizing data files
 - Setting AUTOEXTEND to ON
 - Shrinking objects
 - Reducing UNDO_RETENTION
- Check for long-running queries in temporary tablespaces.

Reclaiming Space by Shrinking Segments

- Shrink is an online and in-place operation.
- It is applicable only to segments residing in ASSM tablespaces.
- Candidate segment types:
 - Heap-organized tables and index-organized tables
 - Indexes
 - Partitions and subpartitions
 - Materialized views and materialized view logs

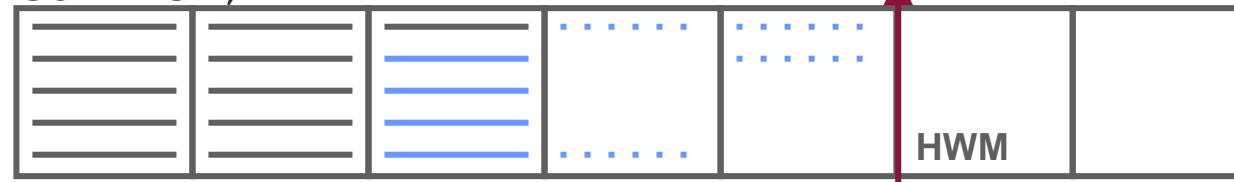
Shrinking Segments



1

```
ALTER TABLE employees SHRINK SPACE
```

```
COMPACT;
```



DML operations and queries can be issued during compaction.

2

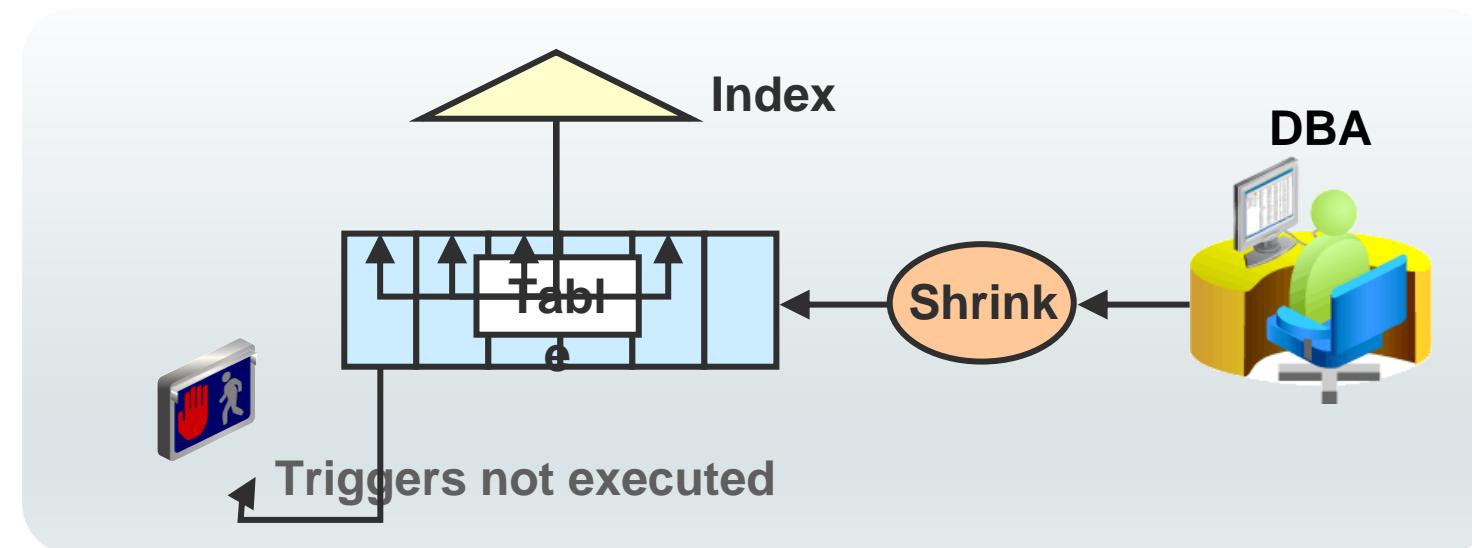
```
ALTER TABLE employees SHRINK SPACE;
```



DML operations are blocked when the HWM is adjusted.

Results of a Shrink Operation

- Improved performance and space utilization
- Indexes maintained
- Triggers not executed
- Number of migrated rows may be reduced
- Rebuilding secondary indexes on IOTs recommended



Managing Resumable Space Allocation

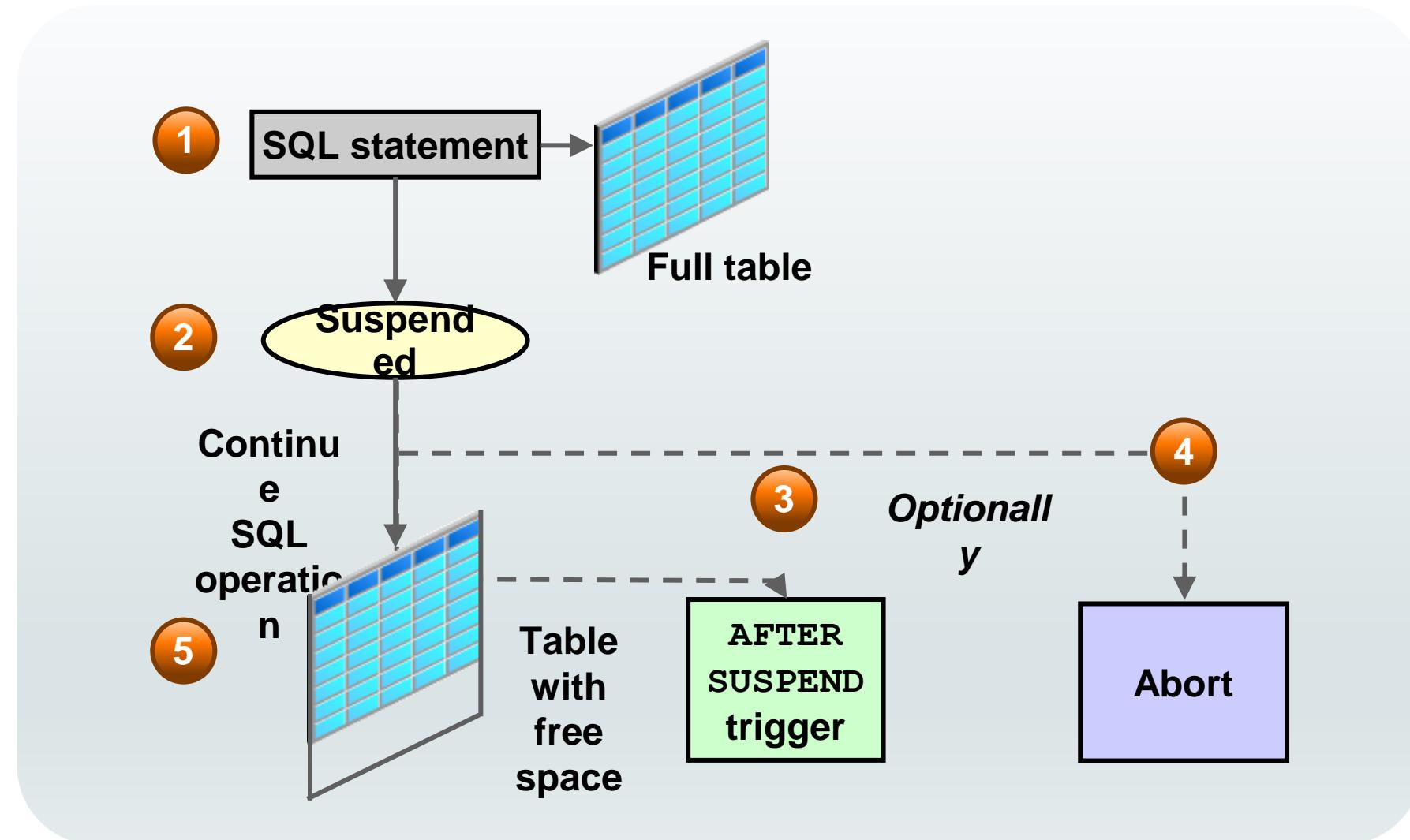
A resumable statement:

- Enables you to suspend large operations instead of receiving an error
- Gives you a chance to fix the problem while the operation is suspended, rather than starting over
- Is suspended for the following conditions:
 - Out of space
 - Maximum extents reached
 - Space quota exceeded
- Can be suspended and resumed multiple times

Using Resumable Space Allocation

- Queries, DML operations, and certain DDL operations can be resumed if they encounter an out-of-space error.
- A resumable statement can be issued through SQL, PL/SQL, SQL*Loader, and Data Pump utilities, or Oracle Call Interface (OCI).
- A statement executes in resumable mode only if its session has been enabled by one of the following actions:
 - The `RESUMABLE_TIMEOUT` initialization parameter is set to a nonzero value.
 - An `ALTER SESSION ENABLE RESUMABLE` statement is issued.

Resuming Suspended Statements



What Operations Are Resumable?

The following operations are resumable:

- Queries: SELECT statements that run out of temporary space (for sort areas)
- DML: INSERT, UPDATE, and DELETE statements
- The following DDL statements:
 - CREATE TABLE . . . AS SELECT
 - CREATE INDEX
 - ALTER INDEX . . . REBUILD
 - ALTER TABLE . . . MOVE PARTITION
 - ALTER TABLE . . . SPLIT PARTITION
 - ALTER INDEX . . . REBUILD PARTITION
 - ALTER INDEX . . . SPLIT PARTITION
 - CREATE MATERIALIZED VIEW

Summary

In this lesson, you should have learned how to:

- Describe and use Oracle Database features that save space
- Create private temporary tables
- Save space by using compression
- Reclaim wasted space from tables and indexes by using the segment shrink functionality
- Manage resumable space allocation

Practice Overview

- Managing Space in Tablespaces
- Using Compression
- Enabling the Resumable Space Allocation Feature

18. Managing Undo Data

Objectives

After completing this lesson, you should be able to:

- Explain DML and undo data generation
- Monitor and administer undo data
- Describe the difference between undo data and redo data
- Configure undo retention
- Guarantee undo retention
- Enable temporary undo

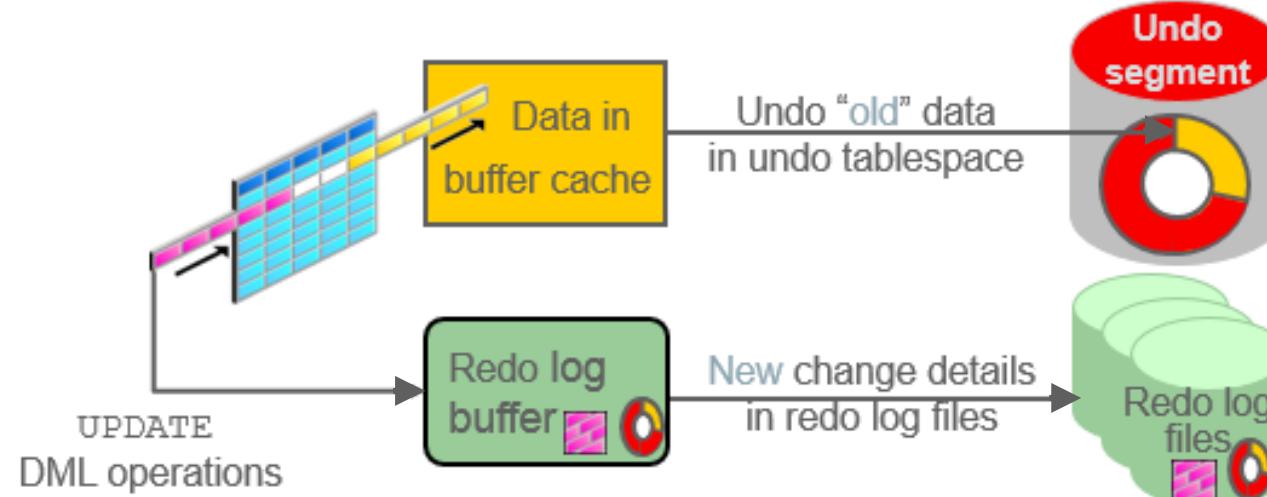
Undo Data: Overview

Undo data is:

- A record of the action of a transaction
- Captured for every transaction that changes data
- Retained at least until the transaction is ended
- Used to support:
 - Rollback operations
 - Read-consistent queries
 - Oracle Flashback Query, Oracle Flashback Transaction, and Oracle Flashback Table
 - Recovery from failed transactions

Transactions and Undo Data

- Each transaction is assigned to only one undo segment.
- An undo segment can service more than one transaction at a time.



Storing Undo Information

- Undo information is stored in undo segments, which are stored in an undo tablespace.
- Undo tablespaces:
 - Are used only for undo segments
 - Have special recovery considerations
 - May be associated with only a single instance
 - Require that only one of them be the current writable undo tablespace for a given instance at any given time

Comparing Undo Data and Redo Data

	Undo	Redo
Record of	How to undo a change	How to reproduce a change
Used for	Rollback, read consistency, flashback	Rolling forward of database changes
Stored in	Undo segments	Redo log files

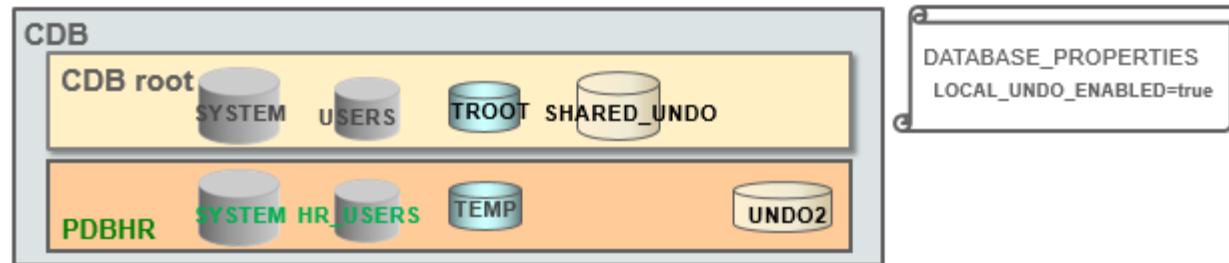


Managing Undo

- Automatic undo management:
 - Fully automated management of undo data and space in a dedicated undo tablespace
 - For all sessions
 - Self-tuning in AUTOEXTEND tablespaces to satisfy long-running queries
 - Self-tuning in fixed-size tablespaces for best retention
- DBA tasks in support of Flashback operations:
 - Configuring undo retention
 - Changing the undo tablespace to a fixed size
 - Avoiding space and “snapshot too old” errors

Comparing SHARED Undo Mode and LOCAL Undo Mode

- There are two undo modes in the multitenant architecture: SHARED and LOCAL.
 - There is only one SHARED undo tablespace (in CDB root).
 - There can be a LOCAL undo tablespace in each PDB.



- When is LOCAL undo mode required?
 - Hot cloning
 - Near-zero down time PDB relocation

```
SQL> STARTUP UPGRADE;  
SQL> ALTER DATABASE LOCAL UNDO ON;
```

Configuring Undo Retention

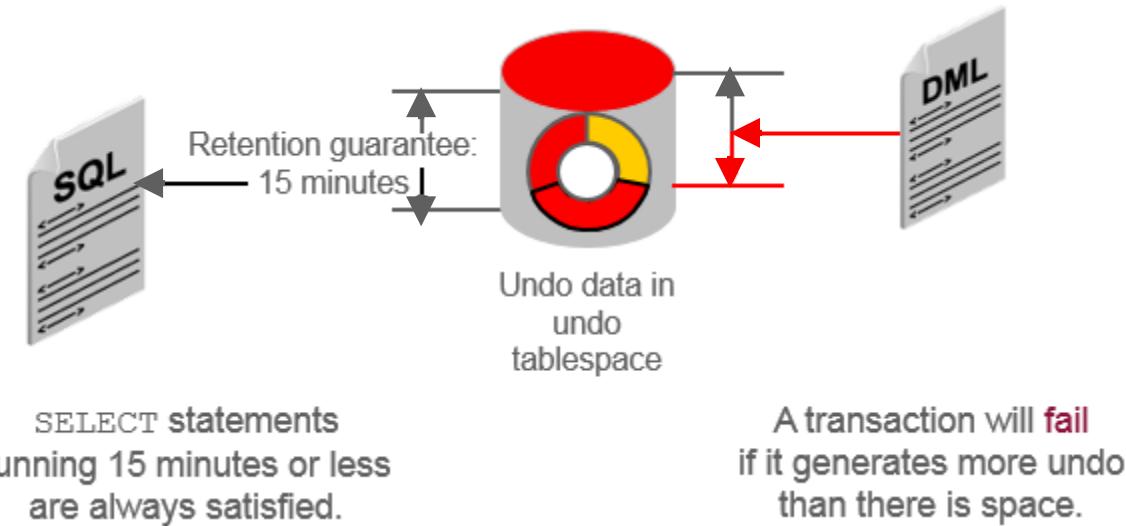
- `UNDO_RETENTION` specifies (in seconds) how long already committed undo information is to be retained.
- Set this parameter when:
 - The undo tablespace has the `AUTOEXTEND` option enabled
 - You want to set undo retention for LOBs
 - You want to guarantee retention

Categories of Undo

Category	Description
Active: Uncommitted undo information	Supports an active transaction and is never overwritten
Unexpired: Committed undo information	Is required to meet the undo retention interval
Expired: Expired undo information	Overwritten when space is required for an active transaction

Guaranteeing Undo Retention

```
SQL> ALTER TABLESPACE undotbs1 RETENTION GUARANTEE;
```

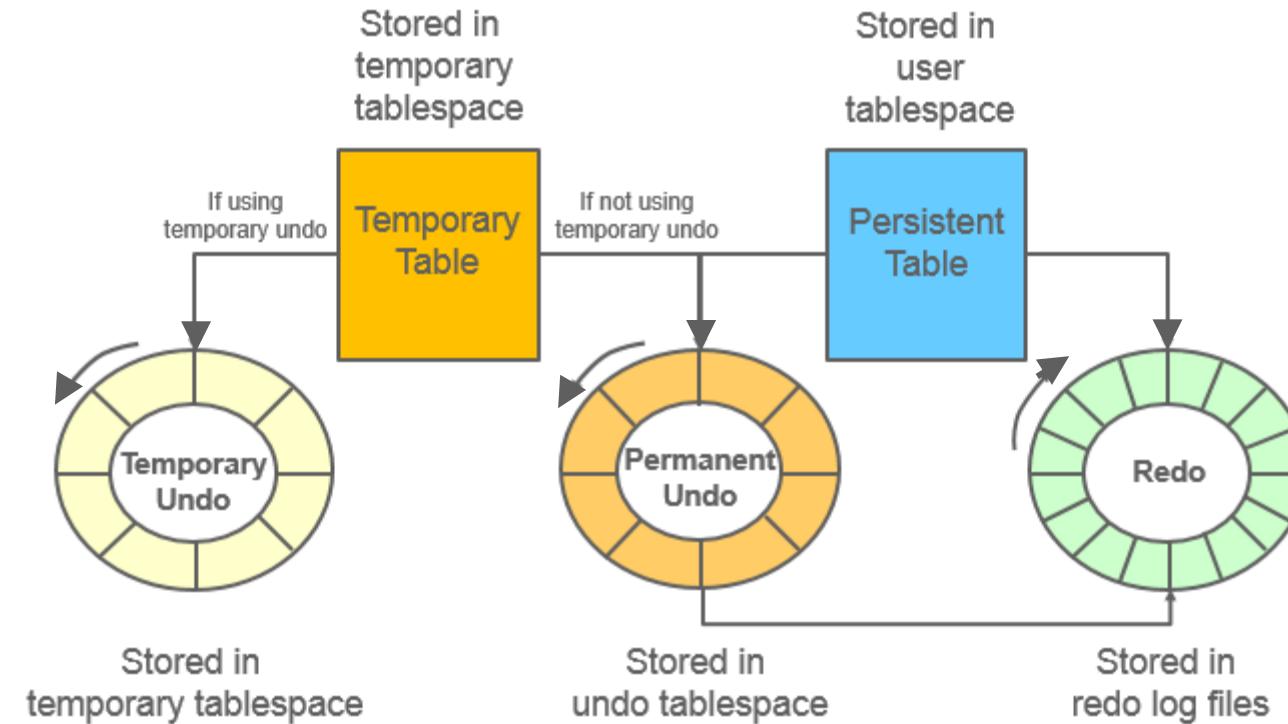


This example is based on an `UNDO_RETENTION` setting of 900 seconds (15 minutes).

Changing an Undo Tablespace to a Fixed Size

- Rationale:
 - Supporting Flashback operations
 - Limiting tablespace growth
- Steps:
 - Run the regular workload.
 - The self-tuning mechanism establishes the minimum required size.
 - (Optional) Use the Enterprise Manager Cloud Control Undo Advisor, which calculates the required size for future growth.
 - (Optional) Change the undo tablespace to a fixed size.

Temporary Undo: Overview



Temporary Undo Benefits

- Reduces the amount of undo stored in the undo tablespaces
- Reduces the amount of redo data written to the redo log
- Enables DML operations on temporary tables in a physical standby database with the Oracle Active Data Guard option

Enabling Temporary Undo

- Enable temporary undo for a session:

```
SQL> ALTER SESSION SET temp_undo_enabled = true;
```

- Enable temporary undo for the database instance:

```
SQL> ALTER SYSTEM SET temp_undo_enabled = true;
```

- Temporary undo mode is selected when a session first uses a temporary object.

Monitoring Temporary Undo

```
SQL> SELECT to_char(BEGIN_TIME,'dd/mm/yy hh24:mi:ss') "BEGIN TIME",
  2  txncount "TXNCNT", maxconcurrency, undoblkcnt, uscount "USCNT",
  3  nospaceerrcnt "NOSPEERRCNT"
  4  FROM v$tempundostat;
```

BEGIN TIME	TXNCNT	MAXCONCURRENCY	UNDOBLKCNT	USCNT	NOSPEERRCNT
---	-----	-----	-----	-----	-----
...					
19/08/12 22:19:44	0	0	0	0	0
19/08/12 22:09:44	0	0	0	0	0
...					
19/08/12 13:09:44	0	0	0	0	0
19/08/12 12:59:44	3	1	24	1	0
576 rows selected.					

```
SQL>
```

Summary

In this lesson, you should have learned how to:

- Explain DML and undo data generation
- Monitor and administer undo data
- Describe the difference between undo data and redo data
- Configure undo retention
- Guarantee undo retention
- Enable temporary undo

Practice Overview

- Managing Undo Tablespaces in a PDB

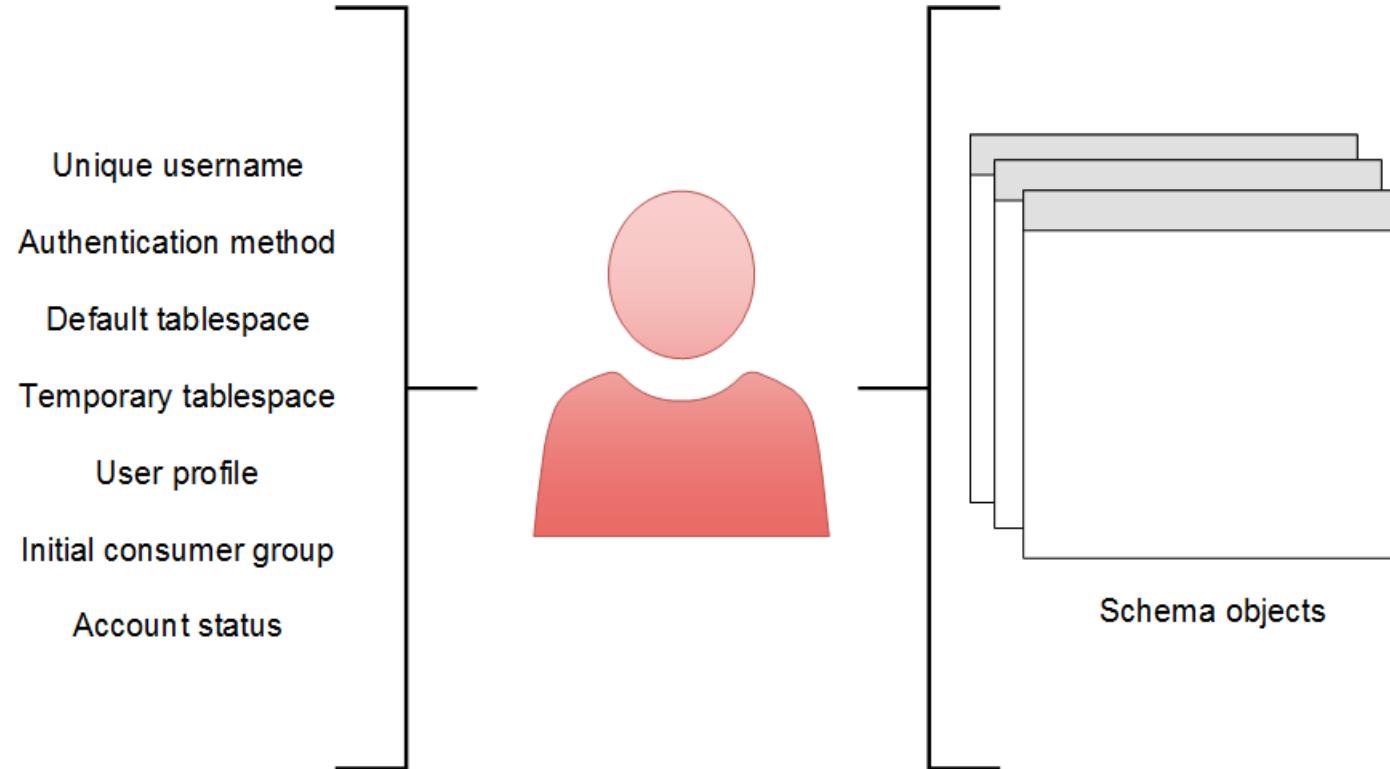
19. Creating and Managing User Accounts

Objectives

After completing this lesson, you should be able to:

- Create database users
- Explain the various authentication options for users
- Assign quota to users

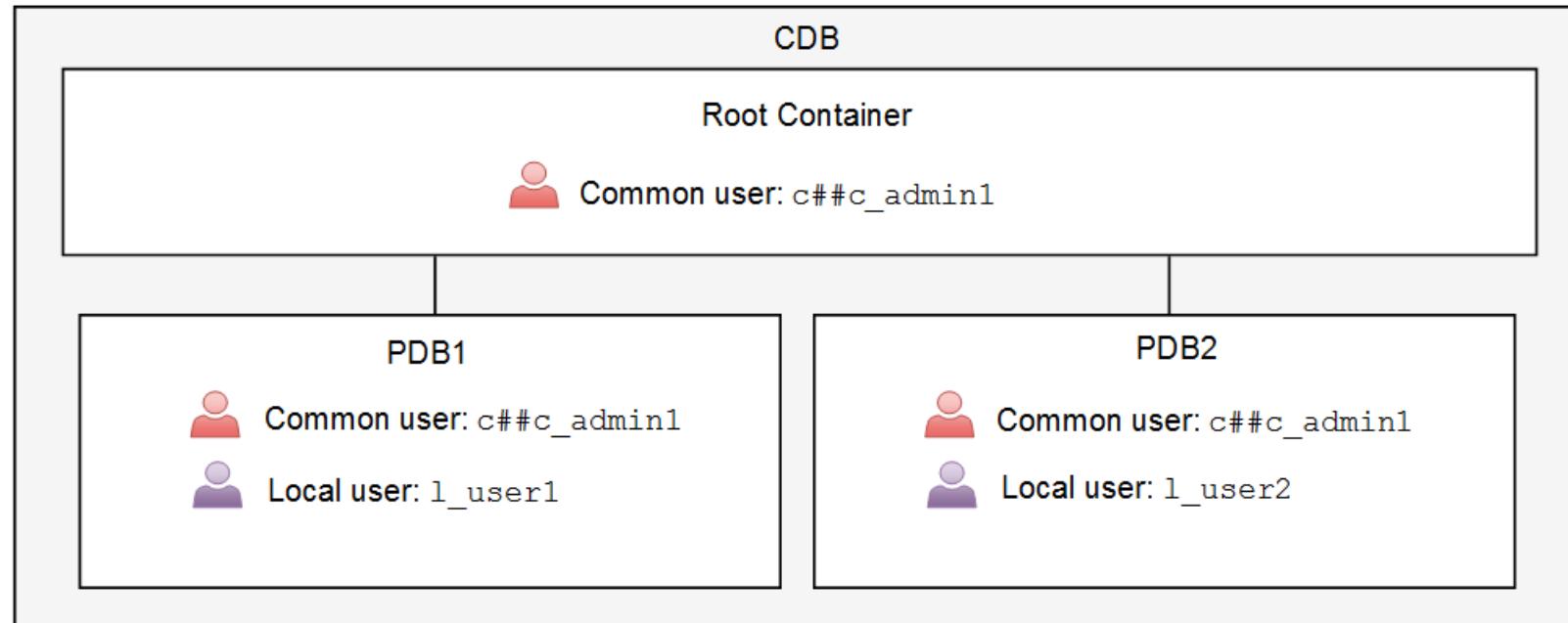
Database User Accounts



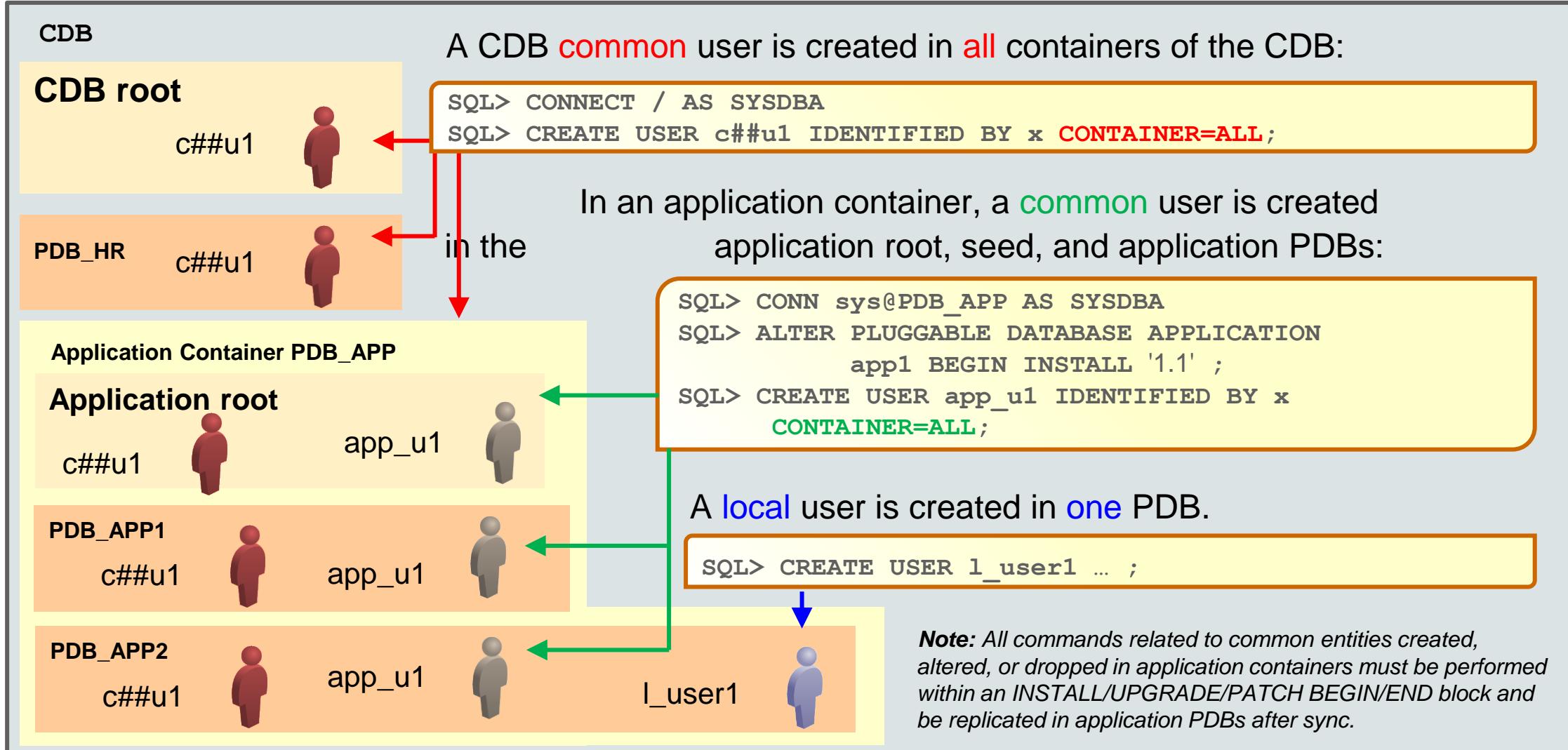
Oracle-Supplied Administrator Accounts

Account	Description
SYS	Super user. Owns the data dictionary and the Automatic Workload Repository (AWR). Used for starting up and shutting down 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
SYSRAC	For Oracle Real Application Clusters (RAC) database administration tasks
SYSMAN	For Oracle Enterprise Manager database administration tasks
DBSNMP	Used by the Management Agent component of Oracle Enterprise Manager to monitor and manage the database

Creating Oracle Database Users in a Multitenant Environment



Creating Common Users in the CDB and PDBs



Creating Schema Only Accounts

- It ensures that a user cannot log in to the instance.
- It enforces data access through the application.
- It secures schema objects by preventing the connected schema from dropping objects.
- A schema only account cannot connect through a database link.
- Oracle-supplied schemas created with the NO AUTHENTICATION clause are schema only accounts.
- Administrator privileges can be granted to and revoked from schema only accounts.

```
SQL> CREATE USER schema_noauth NO AUTHENTICATION;
```

Authenticating Users

- Every user, including administrators, must be authenticated when connecting to a database instance.
- Authentication verifies that the user is a valid database user and establishes a trust relationship for further interactions.
- Authentication also enables accountability by making it possible to link access and actions to specific identities.
- The following authentication methods are possible:
 - Password (usually for database users)
 - Operating system (OS) authentication
 - Password file (for system administrative privileged users only)
 - Strong authentication with Kerberos, SSL, or directory authentication
- A system administrative privileged user must use OS authentication, password file authentication, or strong authentication. These methods can authenticate when the database is available or unavailable (not started).

Using Password Authentication

- Create each user with an associated password that must be supplied when the user attempts to establish a connection.
- When setting up a password, you can expire the password immediately, which forces the user to change the password after first logging in.
- All passwords created in Oracle Database are case-sensitive by default.
- Passwords may contain multibyte characters and are limited to 30 bytes.
- Passwords are always automatically and transparently encrypted by using the Advanced Encryption Standard (AES) algorithm during network (client/server and server/server) connections before sending them across the network.
- A password management policy, controlled through user profiles, can be used to set a password expiration period, grace period for changing a password, and other attributes.

Using Password File Authentication

- You can use password file authentication for an Oracle database instance and for an Oracle Automatic Storage Management (Oracle ASM) instance.
- A password file stores database usernames and case-sensitive passwords for administrator users (common and local administrators).
- To prepare for password file authentication, you must:
 - Create the password file. DBCA creates a password file during execution.
 - Set the `REMOTE_LOGIN_PASSWORDFILE` initialization parameter
 - Grant system administrative privileges (for example, `GRANT SYSDBA TO mydba`)

Using OS Authentication

- Oracle Universal Installer creates operating system groups, assigns them specific names, and maps each group to a specific system privilege.
- Example: Members of the dba group are granted SYSDBA
- As a group member, you can be authenticated, enabled as an administrative user, and connected to a local database:

```
SQL> CONNECT / AS SYSDBA
```

```
SQL> CONNECT / AS SYSOPER
```

```
SQL> CONNECT / AS SYSBACKUP
```

```
SQL> CONNECT / AS SYSDG
```

```
SQL> CONNECT / AS SYSKM
```

```
SQL> CONNECT / AS SYSRAC
```

- If you are not a member of one of these OS groups, you will not be able to connect as an administrative user via OS authentication.

OS Authentication for Privileged Users

OS Group	UNIX or Linux User Group	Special System Privilege Granted to Members
Oracle Software Group (top level group)	oinstall	Allowed to create and delete database files on the OS. All database administrators belong to this group.
Database Administrator Group (OSDBA)	dba	SYSDBA (Connects you as the SYS user)
Database Operator Group (OSOPER) – optional	oper	SYSOPER (Connects you as the PUBLIC user)
Database Backup and Recovery Group (OSBACKUPDBA)	backupdba	SYSBACKUP
Data Guard Administrative Group (OSDGDBA)	dgdba	SYSDG
Encryption Key Management Administrative Group (OSKMDBA)	kmdba	SYSKM
Real Application Cluster Administrative Group (OSRACDBA)	rac	SYSRAC

Assigning Quotas

- A quota is a space allowance in a given tablespace.
- By default, a user has no quota on any of the tablespaces.
- Database accounts that need quota are those that own database objects (for example, accounts for applications).
- Only those activities that use space in a tablespace count against quota.
 - Oracle server checks quota when you create or extend a segment.
 - Activities that don't use space don't impact quota (example: CREATE VIEW).
 - You can be granted permission to use objects without needing any quota.
- Quota is not needed for assigned temporary tablespaces or undo tablespaces.
- A user's quota is replenished when objects are dropped with the PURGE clause or when objects in the recycle bin are purged.

Assigning Quotas

You have three options for providing quota for a user on a tablespace:

- UNLIMITED
- Value
- UNLIMITED TABLESPACE system privilege

Summary

In this lesson, you should have learned how to:

- Create database users
- Explain the various authentication options for users
- Assign quota to users

Practice Overview

- Creating Common and Local Users
- Creating a Local User for an Application
- Exploring OS and Password File Authentication

20. Configuring Privilege and Role Authorization

Objectives

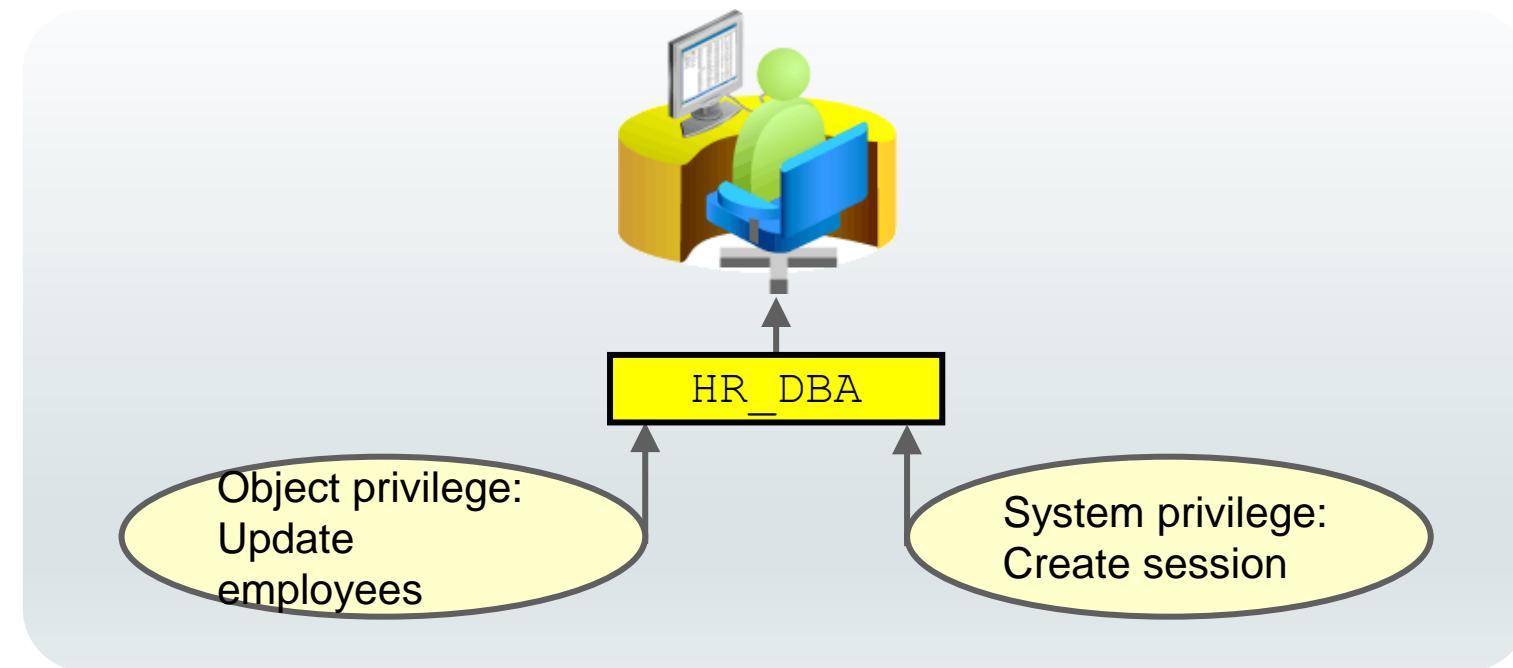
After completing this lesson, you should be able to:

- Grant system and object privileges to database users, commonly and locally
- Create roles
- Grant roles to users and other roles, commonly and locally
- Revoke privileges and roles from users and other roles

Privileges

There are two types of user privileges:

- System: Enables users to perform particular actions in the database
- Object: Enables users to access and manipulate a specific object



System Privileges

- Each system privilege allows a user to perform a particular database operation or class of database operations.
- Administrators have special system privileges.
- A system privilege with the ANY clause means the privilege applies to all schemas except SYS, not just your own.
- If you grant a system privilege with the ADMIN OPTION enabled, you enable the grantee to administer the system privilege and grant it to other users.

System Privileges for Administrators

Privilege	Description
SYSDBA	Perform all administrative tasks in the database, including create and drop a database, open and mount a database, start up and shut down an Oracle database, create an SPFILE, put a database in or remove a database from ARCHIVELOG mode, perform incomplete recovery operations, patch, and migrate. This privilege enables you to connect as the SYS user.
SYSOPER	Perform similar administration tasks as the SYSDBA privilege, but without the ability to look at user data. For example, you can start up and shut down the database, create an SPFILE, and perform complete recovery operations (not incomplete recovery operations).
SYSASM	Start up, shut down, and administer an Automatic Storage Management instance.
SYSBACKUP	Perform backup and recovery operations by using RMAN or SQL*Plus.
SYSDG	Perform Data Guard operations by using the Data Guard Broker or the DGMGRL command-line interface.
SYSKM	Manage Transparent Data Encryption wallet operations.
SYSRAC	Perform day-to-day administration tasks on an Oracle Real Application Clusters (RAC) cluster.

Object Privileges

- Allow a user to perform a particular action on a specific object.
- Without specific permission, users can access only their own objects.
- Object privileges can be granted by
 - The owner of an object
 - The administrator
 - Someone who has been explicitly given permission to grant privileges on the object
- The SQL syntax for granting object privileges is:

```
GRANT <object_privilege> ON <object> TO <grantee clause>
[WITH GRANT OPTION]
```

Granting Privileges in a Multitenant Environment

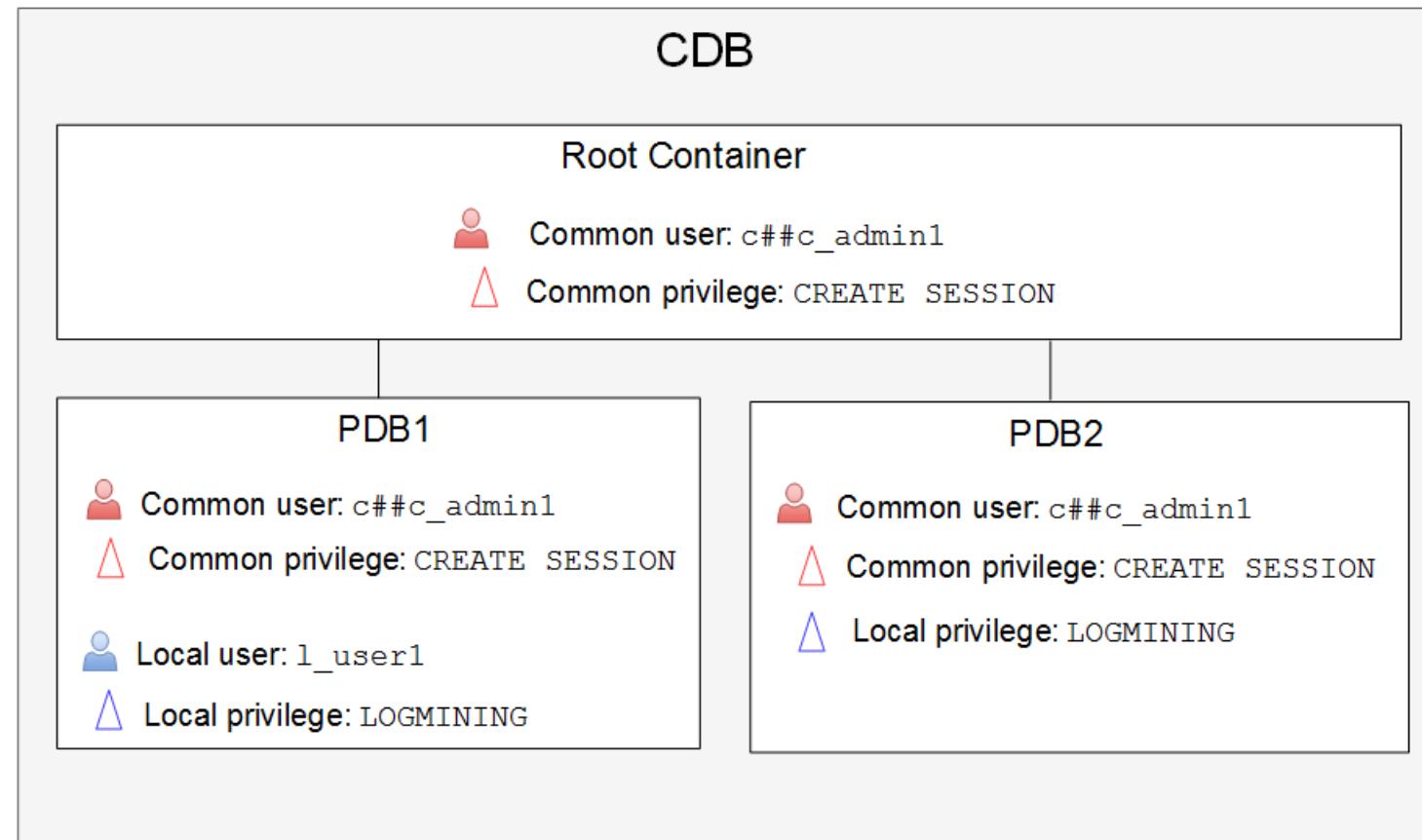
- Commonly: Grant the user a privilege in all containers of a CDB.

```
SQL> CONNECT / AS SYSDBA  
SQL> GRANT create session TO c##c_admin1 CONTAINER=ALL;
```

- Locally: Grant the user a privilege in a single PDB only.

```
SQL> CONNECT SYS@PDB1 AS SYSDBA  
SQL> GRANT logmining TO l_user1;
```

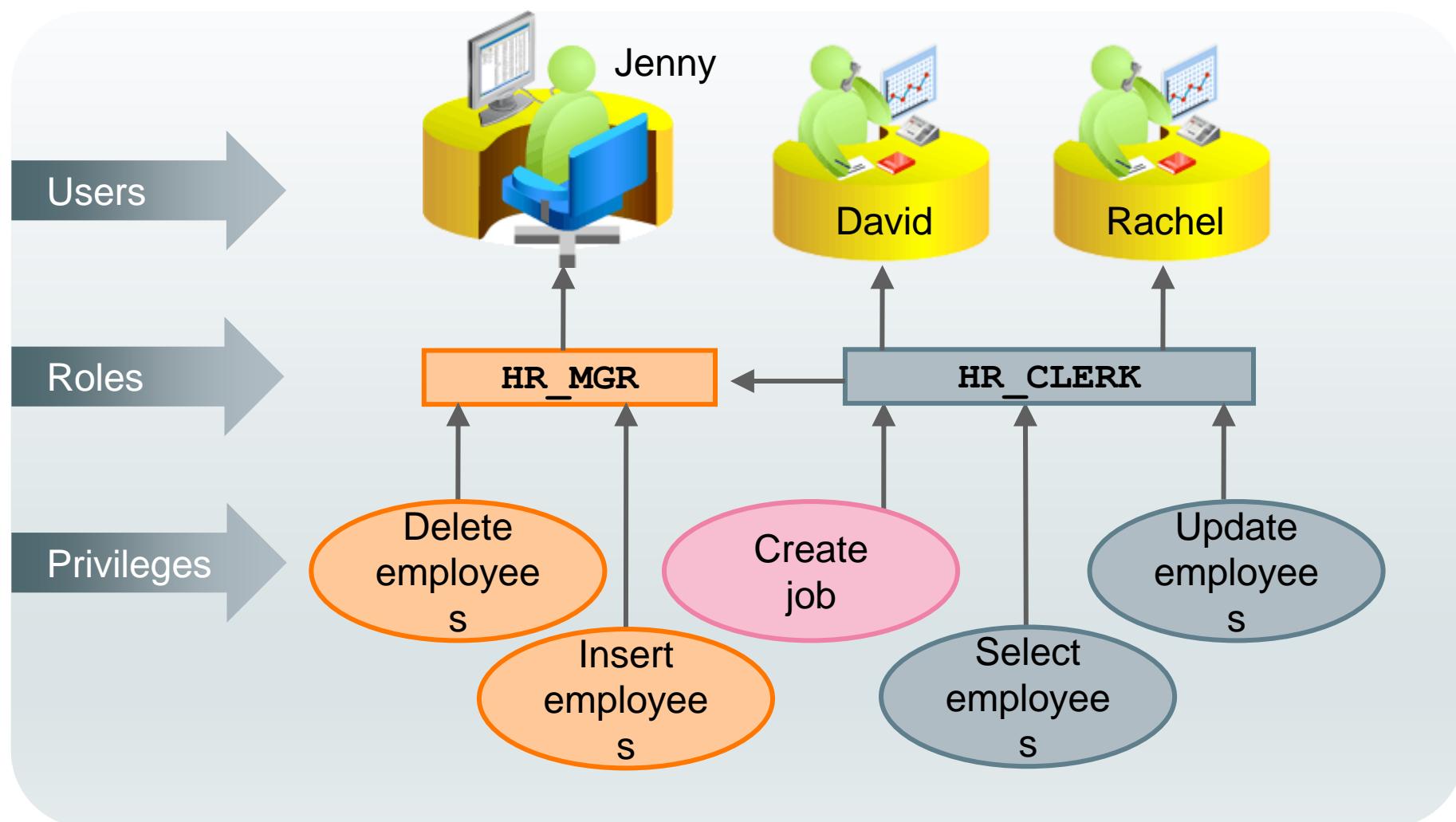
Granting Privileges: Example



Using Roles to Manage Privileges

- Roles:
 - Used to group privileges and roles together
 - Facilitate granting of multiple privileges or roles to users
- Benefits of roles:
 - Easier privilege management
 - Dynamic privilege management
 - Selective availability of privileges

Assigning Privileges to Roles and Assigning Roles to Users



Oracle-Supplied Roles

Account	Description
DBA	<p>Includes most system privileges and several other roles. Do not grant this role to nonadministrators.</p> <p>Users with this role can connect to the CDB or PDB only when it is open.</p>
RESOURCE	CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE
SCHEDULER_ADMIN	CREATE ANY JOB, CREATE EXTERNAL JOB, CREATE JOB, EXECUTE ANY CLASS, EXECUTE ANY PROGRAM, MANAGE SCHEDULER
SELECT_CATALOG_ROLE	SELECT privileges on data dictionary objects

Granting Roles in a Multitenant Environment

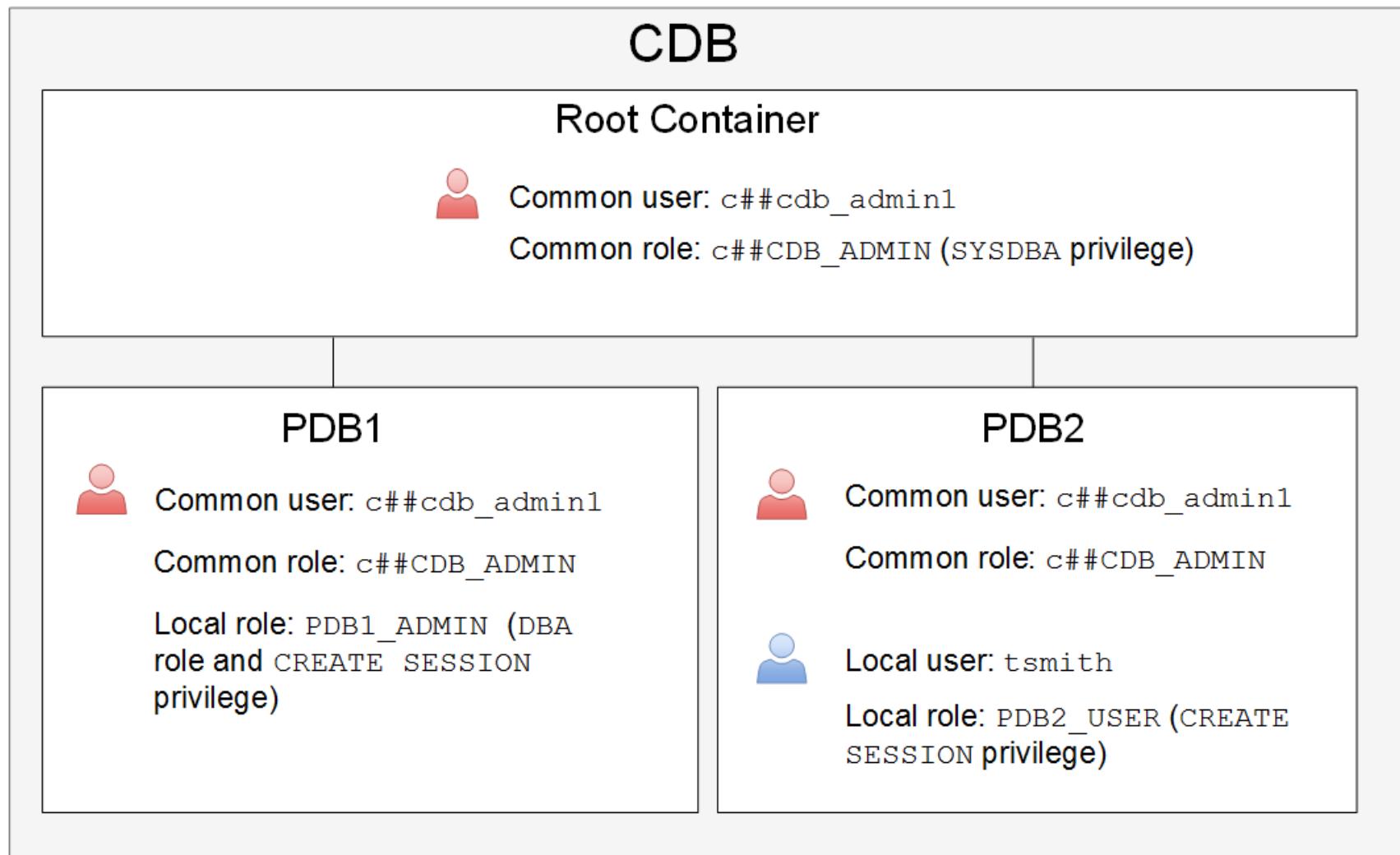
- Commonly: Grant the role to the user (or role) in all containers.

```
SQL> CONNECT / AS SYSDBA  
SQL> GRANT <common role> TO <common user or role> CONTAINER=ALL;
```

- Locally: Grant the role to a user (or role) in one PDB only.

```
SQL> CONNECT SYS@PDB1 AS SYSDBA  
SQL> GRANT <common or local role> TO <common or local user>;
```

Granting Roles: Example



Making Roles More Secure

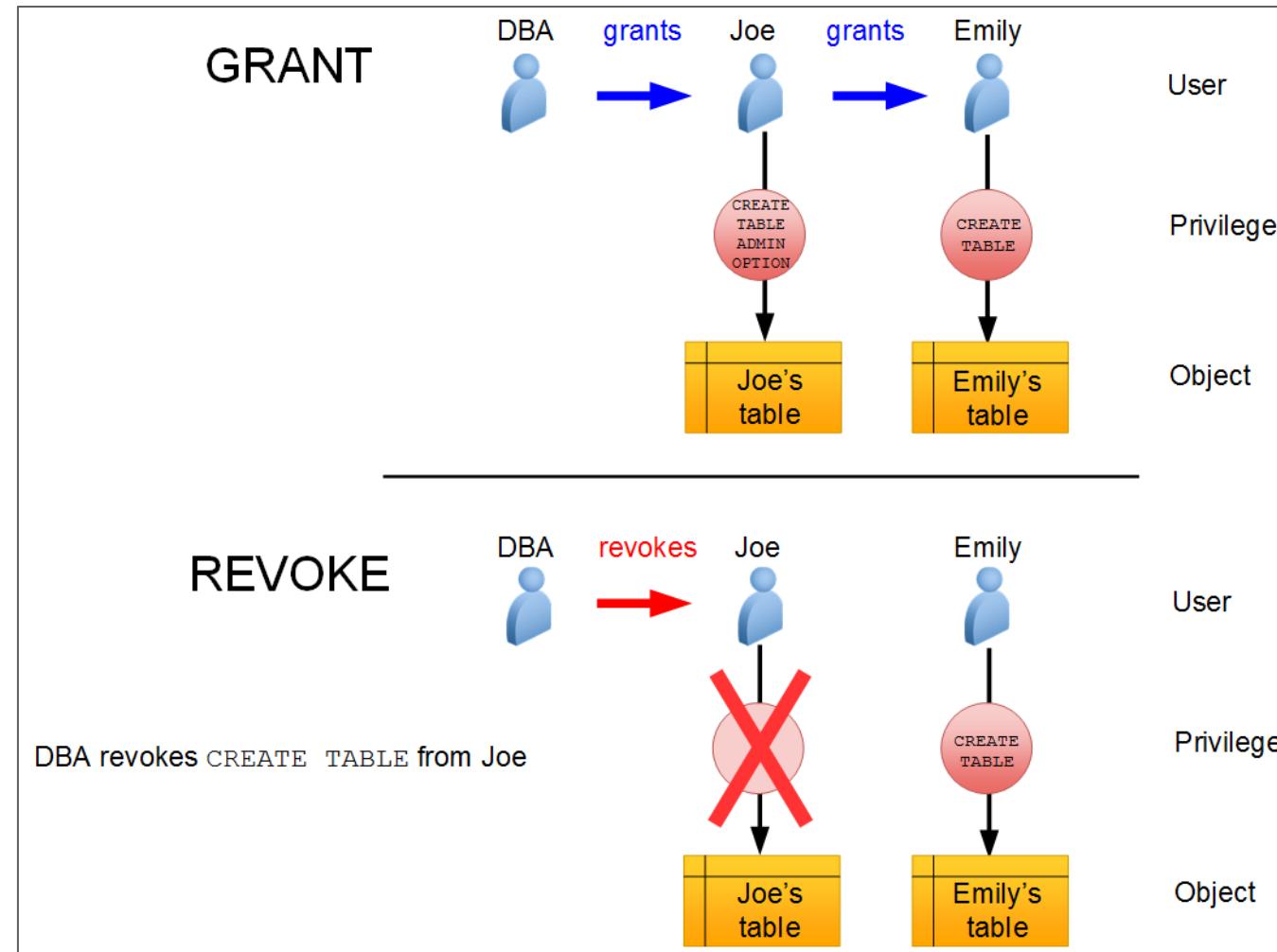
- Roles are usually enabled by default, which means that if a role is granted to a user, then that user can exercise the privileges given to the role immediately.
- Default roles are assigned to the user at connect time.
- Use the following security measures to make roles more secure:
 - Make a role nondefault.
 - Use role authentication.
 - Create application roles.

Revoking Roles and Privileges

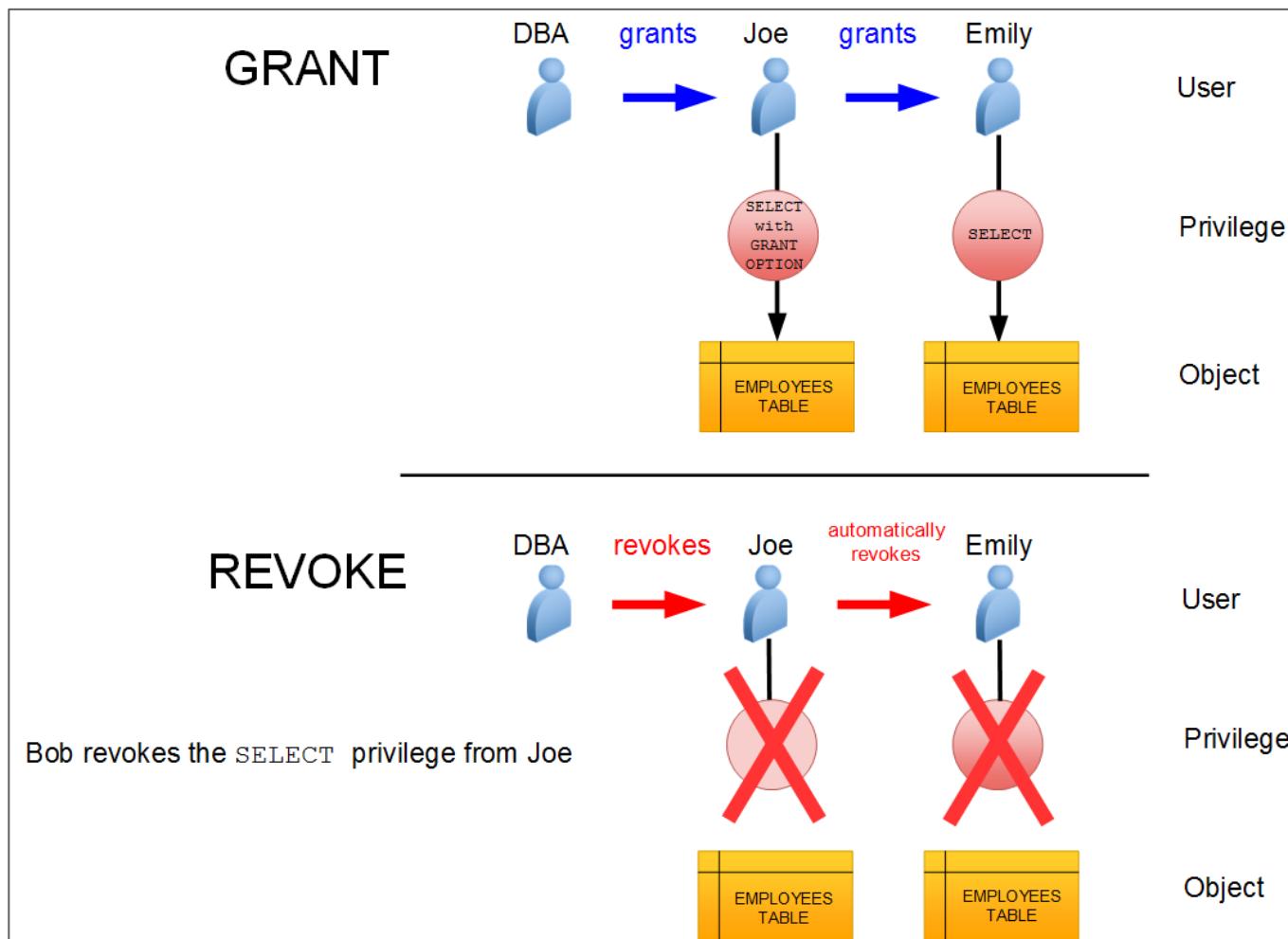
You can use the REVOKE statement to:

- Revoke system privileges from users and roles
- Revoke roles from users, roles, and program units
- Revoke object privileges for a particular object from users and roles

Granting and Revoking System Privileges



Granting and Revoking Object Privileges



Summary

In this lesson, you should have learned how to:

- Grant system and object privileges to database users, commonly and locally
- Create roles
- Grant roles to users and other roles, commonly and locally
- Revoke privileges and roles from users and other roles

Practice Overview

- Granting a Local Role (DBA) to PDBADMIN
- Using SQL Developer to Create Local Roles

21. Configuring User Resource Limits

Objectives

After completing this lesson, you should be able to:

- Create and assign profiles to:
 - Control resource consumption
 - Manage account status and password expiration
- Use Oracle-supplied password functions in profiles

Profiles and Users

- Users are assigned only one profile at a time.
- Profiles:
 - Control resource consumption
 - Manage account status and password expiration
- `RESOURCE_LIMIT` must be set to `TRUE` (default) for profiles to impose resource limitations.

Creating Profiles in a Multitenant Architecture

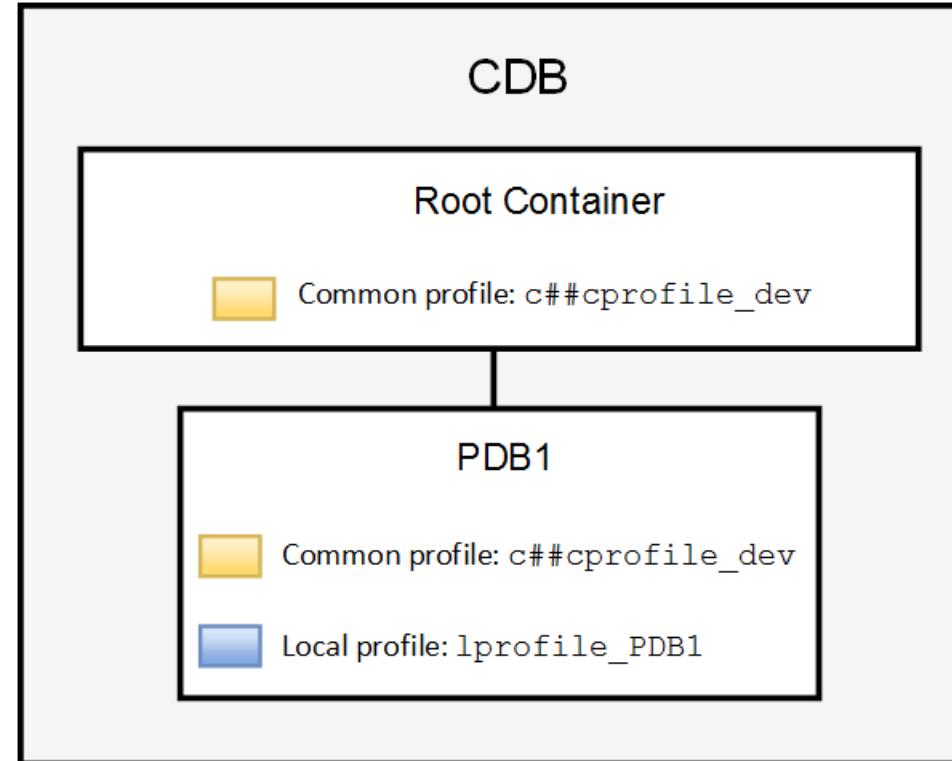
- Common profile: Replicated in all current and future containers

```
SQL> CREATE PROFILE c##cprofile_dev  
2 limit ... CONTAINER=ALL;
```

- Local profile: Created in a single PDB and can be used within that PDB only

```
SQL> CREATE PROFILE lprofile_PDB1  
2 limit ... ;
```

Creating Profiles: Example



Profile Parameters: Resources

- In a profile, you can control:
 - CPU resources: May be limited to a per-session or per-call basis
 - Network and memory resources (Connect time, Idle time, Concurrent sessions, Private SGA)
- Disk I/O resources: Limit the amount of data a user can read at the per-session level or per-call level.
- Profiles cannot impose resource limitations on users unless the `RESOURCE_LIMIT` initialization parameter is set to `TRUE`. With `RESOURCE_LIMIT` at its default value of `FALSE`, profile resource limitations are ignored.
- Profiles also allow composite limits, which are based on weighted combinations of CPU/session, reads/session, connect time, and private SGA.

Profile Parameters: Locking and Passwords

- In a profile, specific parameters control account locking, password aging and expiration, and password history.
- Profile password settings are always enforced.
- Account locking enables automatic locking of accounts for a set duration when users fail to log in to the system in the specified number of attempts or when accounts sit inactive for a predefined number of days (users have not attempted to log in to their accounts).
- Password aging and expiration enables user passwords to have a lifetime, after which the passwords expire and must be changed.
- Password history checks the new password to ensure that the password is not reused for a specified amount of time or a specified number of password changes.
- Password complexity verification makes a complexity check on the password to verify that it meets certain rules.

Oracle-Supplied Password Verification Functions

- Complexity verification checks that each password is complex enough to provide reasonable protection against intruders who try to break into the system by guessing passwords.
- You can create your own password verification functions.
- Oracle Database provides the following functions by default:
 - ORA12C_VERIFY_FUNCTION
 - ORA12C_STRONG_VERIFY_FUNCTION
 - ORA12C_STIG_VERIFY_FUNCTION
- Password complexity checking is not enforced for the SYS user.

Assigning Profiles in a Multitenant Architecture

- Commonly: The profile assignment is replicated in all current and future containers.

```
SQL> CONNECT / AS SYSDBA  
SQL> ALTER USER <common user> PROFILE <common profile> CONTAINER=ALL;
```

- Locally: The profile assignment occurs in one PDB (stand-alone or application container) only.

```
SQL> CONNECT SYS@PDB1 AS SYSDBA  
SQL> ALTER USER <common or local user> PROFILE <common or local profile>;
```

Summary

In this lesson, you should have learned how to:

- Create and assign profiles to:
 - Control resource consumption
 - Manage account status and password expiration
- Use Oracle-supplied password functions in profiles

Practice Overview

- Using SQL Developer to Create a Local Profile
- Using SQL Developer to Create Local Users
- Configuring a Default Role for a User

22. Implementing Oracle Database Auditing

Objectives

After completing this lesson, you should be able to:

- Describe DBA responsibilities for security and auditing
- Enable unified auditing
- Create unified audit policies
- Maintain the audit trail

Database Security

A secure system ensures the confidentiality of the data that it contains. There are several aspects of security:

- Restricting access to data and services
- Authenticating users
- Monitoring for suspicious activity

Monitoring for Compliance

- Monitoring or auditing must be an integral part of your security procedures.
- Review the following:
 - Mandatory auditing
 - Standard database auditing
 - Value-based auditing
 - Fine-grained auditing (FGA)

Types of Activities to be Audited

You can audit the following types of activities:

- User accounts, roles, and privileges
- Object actions
- Application context values
- Oracle Data Pump
- Oracle Database Real Application Security
- Oracle Database Vault
- Oracle Label Security
- Oracle Recovery Manager
- Oracle SQL*Loader direct path events

Mandatorily Audited Activities

The following activities are audited:

- CREATE/ALTER/DROP AUDIT POLICY
- AUDIT/NOAUDIT
- EXECUTE of:
 - DBMS_FGA
 - DBMS_AUDIT_MGMT
- ALTER TABLE **against AUDSYS audit trail table**
- **Top-level statements by administrative users (SYS, SYSDBA, SYSOPER, SYSASM, SYSPBACKUP, SYSDG, and SYSKM) until the database opens**

Understanding Auditing Implementation

- *Mixed mode auditing* is the default when a new database is created.
- Mixed mode auditing enables the use of:
 - Pre-Oracle Database 12c auditing features
 - *Unified auditing* features
- The recommendation from Oracle is to migrate to pure 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';
```

PARAMETER	VALUE
-----------	-------

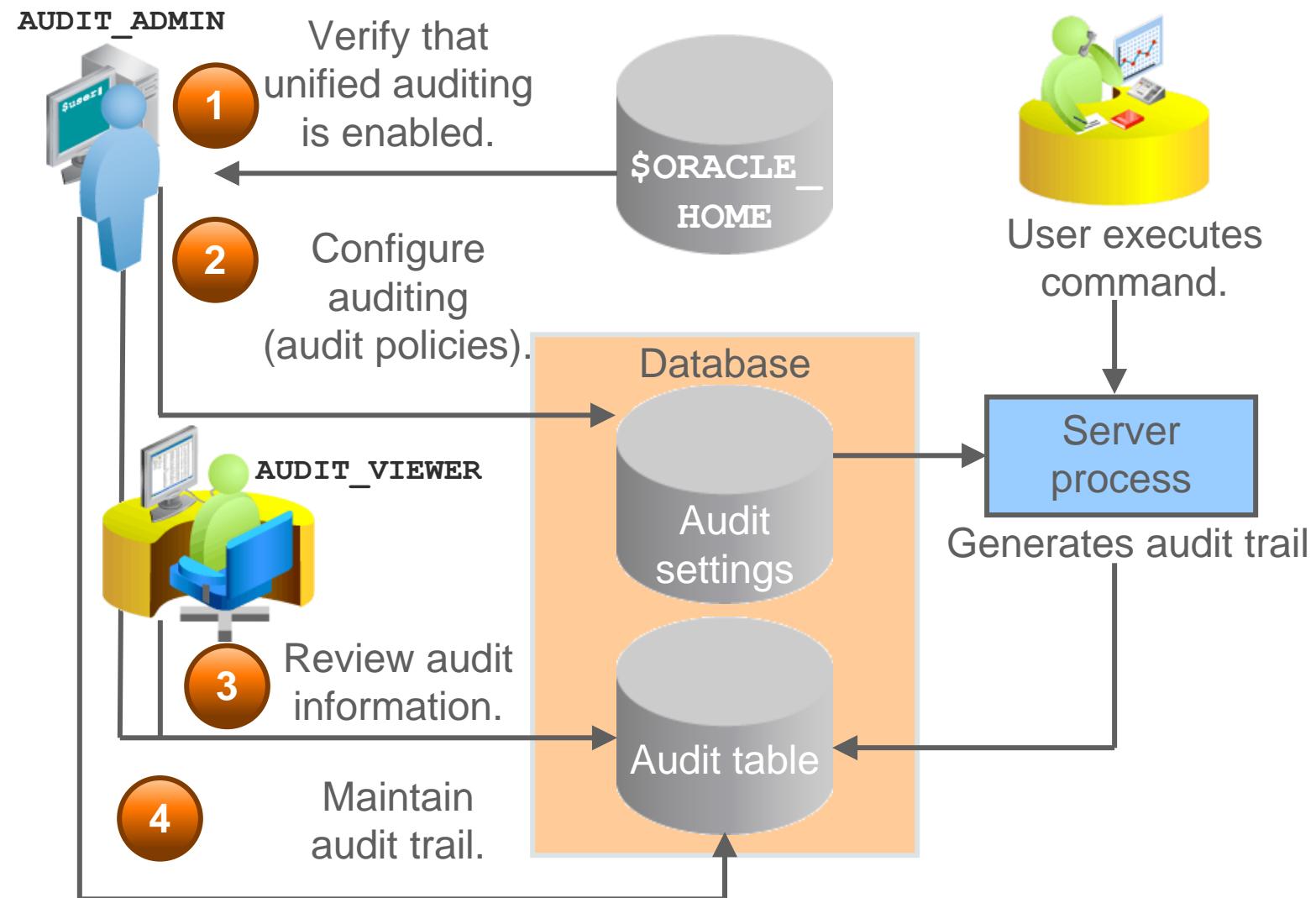
Unified Auditing	TRUE
------------------	------

Administering the Roles Required for Auditing

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

- 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
 - Execute the DBMS_AUDIT_UTIL PL/SQL package

Database Auditing: Overview



Configuring Auditing

Method	Description
Unified audit policies	Group audit settings into a policy
Predefined unified audit policies	Commonly used security-relevant audit settings
Fine-grained audit policies	Define specific conditions that must be met for auditing to take place

Creating a Unified Audit Policy

- Use the CREATE AUDIT POLICY statement to create a unified audit policy:

```
CREATE AUDIT POLICY select_emp_pol  
  ACTIONS select on hr.employees
```

- To simplify policy management, group related options into a single policy.

Creating an Audit Policy: System-Wide Audit Options

- System privileges:

```
CREATE AUDIT POLICY audit_syspriv_pol1  
PRIVILEGES SELECT ANY TABLE, CREATE LIBRARY
```

- Actions:

```
CREATE AUDIT POLICY audit_actions_pol2  
ACTIONS AUDIT, ALTER TRIGGER
```

- Roles:

```
CREATE AUDIT POLICY audit_role_pol3  
ROLES mgr_role
```

- System privileges, actions, and roles:

```
CREATE AUDIT POLICY audit_mixed_pol4  
PRIVILEGES DROP ANY TABLE  
ACTIONS      CREATE TABLE, DROP TABLE, TRUNCATE TABLE  
ROLES        emp_role
```

Creating an Audit Policy: Object-Specific Actions

Create audit policies based on object-specific options.

```
CREATE AUDIT POLICY audit_objpriv_pol5  
ACTIONS SELECT, UPDATE, LOCK ON hr.employees
```

```
CREATE AUDIT POLICY audit_objpriv_pol6  
ACTIONS ALL
```

```
CREATE AUDIT POLICY audit_objpriv_pol7  
ACTIONS EXECUTE, GRANT ON hr.raise_salary_proc
```

Creating an Audit Policy: Specifying Conditions

- Condition and evaluation **PER SESSION**

```
CREATE AUDIT POLICY audit_mixed_pol5
ACTIONS RENAME ON hr.employees,ALTER ON hr.jobs,
WHEN 'SYS_CONTEXT (''USERENV'', ''SESSION_USER'')='JIM'''
EVALUATE PER SESSION
```

- Condition and evaluation **PER STATEMENT**

```
CREATE AUDIT POLICY audit_objpriv_pol6
ACTIONS ALTER ON OE.ORDERS
WHEN 'SYS_CONTEXT(''USERENV'', ''CLIENT_IDENTIFIER'')='OE'''
EVALUATE PER STATEMENT
```

- Condition and evaluation **PER INSTANCE**

```
CREATE AUDIT POLICY audit_objpriv_pol7
ROLES dba
WHEN SYS_CONTEXT(''USERENV'', ''INSTANCE_NAME'')='sales'''
EVALUATE PER INSTANCE
```

Enabling and Disabling Audit Policies

- Enable audit policies:

- Apply to all users.

```
AUDIT POLICY audit_syspriv_pol1;
```

- Apply only to some users.

```
AUDIT POLICY audit_pol2 BY scott, oe;  
AUDIT POLICY audit_pol3 BY sys;
```

- Exclude some users.

```
AUDIT POLICY audit_pol4 EXCEPT jim, george;
```

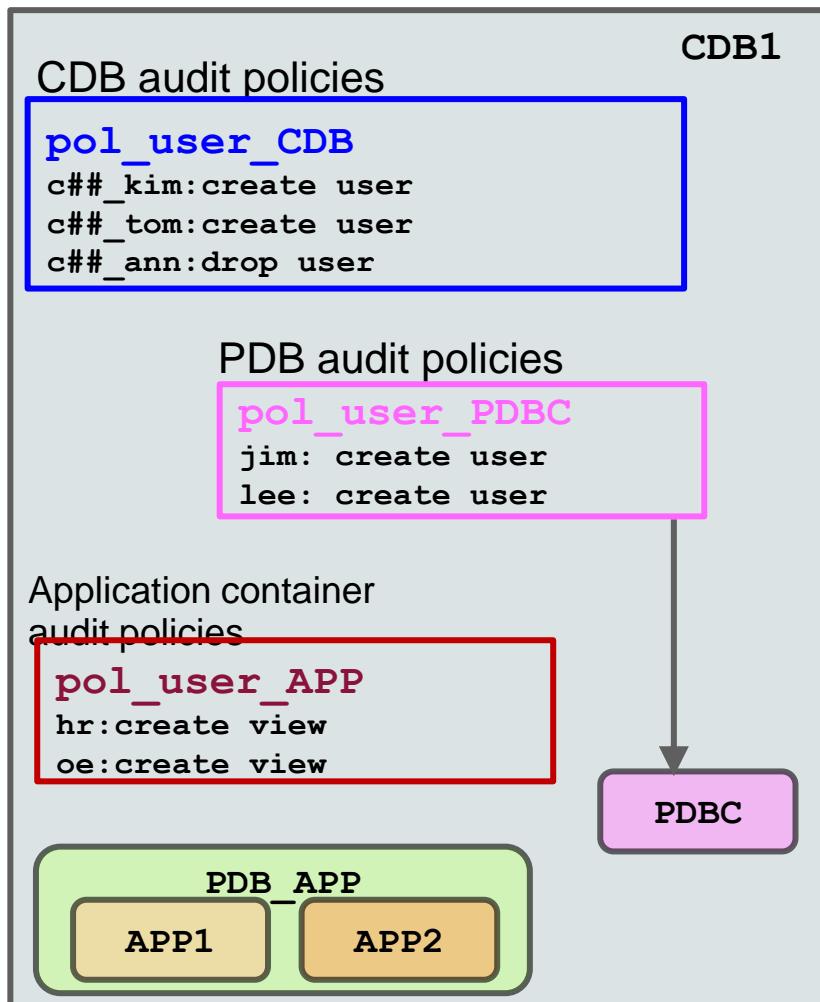
- Audit the recording based on failed or succeeded actions.

```
AUDIT POLICY audit_syspriv_pol1 WHENEVER SUCCESSFUL ;  
AUDIT POLICY audit_objpriv_pol2 WHENEVER NOT SUCCESSFUL ;
```

```
AUDIT POLICY auditpol5 BY joe WHENEVER SUCCESSFUL ;
```

- Disable audit policies by using the NOAUDIT command.

Auditing Actions in the CDB and PDBs



1. Connect to the CDB root or to an application root or to a regular PDB.
2. Create common or local unified audit policies:
 - For all PDBs (*connect to CDB root*)
 - For all application PDBs of an application container (*connect to the application root*)
 - For a regular PDB or a specific application PDB (*connect to the PDB*)
3. Enable/disable audit policies:
 - Define users or users being granted roles to be audited (*DBA role*)
 - Use AUDIT POLICY and NOAUDIT POLICY commands

Modifying a Unified Audit Policy

- Enabled and disabled unified audit policies can be modified.
- Use the ALTER AUDIT POLICY statement to modify a unified audit policy:

```
ALTER AUDIT POLICY select_emp_pol  
ADD ACTIONS select on hr.job_history
```

Auditing Top-Level Statements Only

Top-level statement unified auditing enables you to:

- Audit a top-level user or direct user activities in the database without collecting indirect user activity

```
SQL> CREATE AUDIT POLICY actions_all_pol ACTION ALL ONLY TOLEVEL;  
SQL> AUDIT POLICY actions_all_pol BY SYS;
```

```
SQL> CREATE AUDIT POLICY update_emp_pol ACTIONS UPDATE ON HR.EMPLOYEES ONLY TOLEVEL;  
SQL> AUDIT POLICY update_emp_pol;
```

- Minimize audit records

```
SQL> CONNECT user1@PDB1  
SQL> UPDATE hr.employees SET salary = salary * 0.1 WHERE empno = 100;
```

Direct Audited

```
SQL> EXEC hr.salary_emp_raise (empno => 100, increase => '0.1')
```

Not direct
 Not audited

Viewing Audit Policy Information

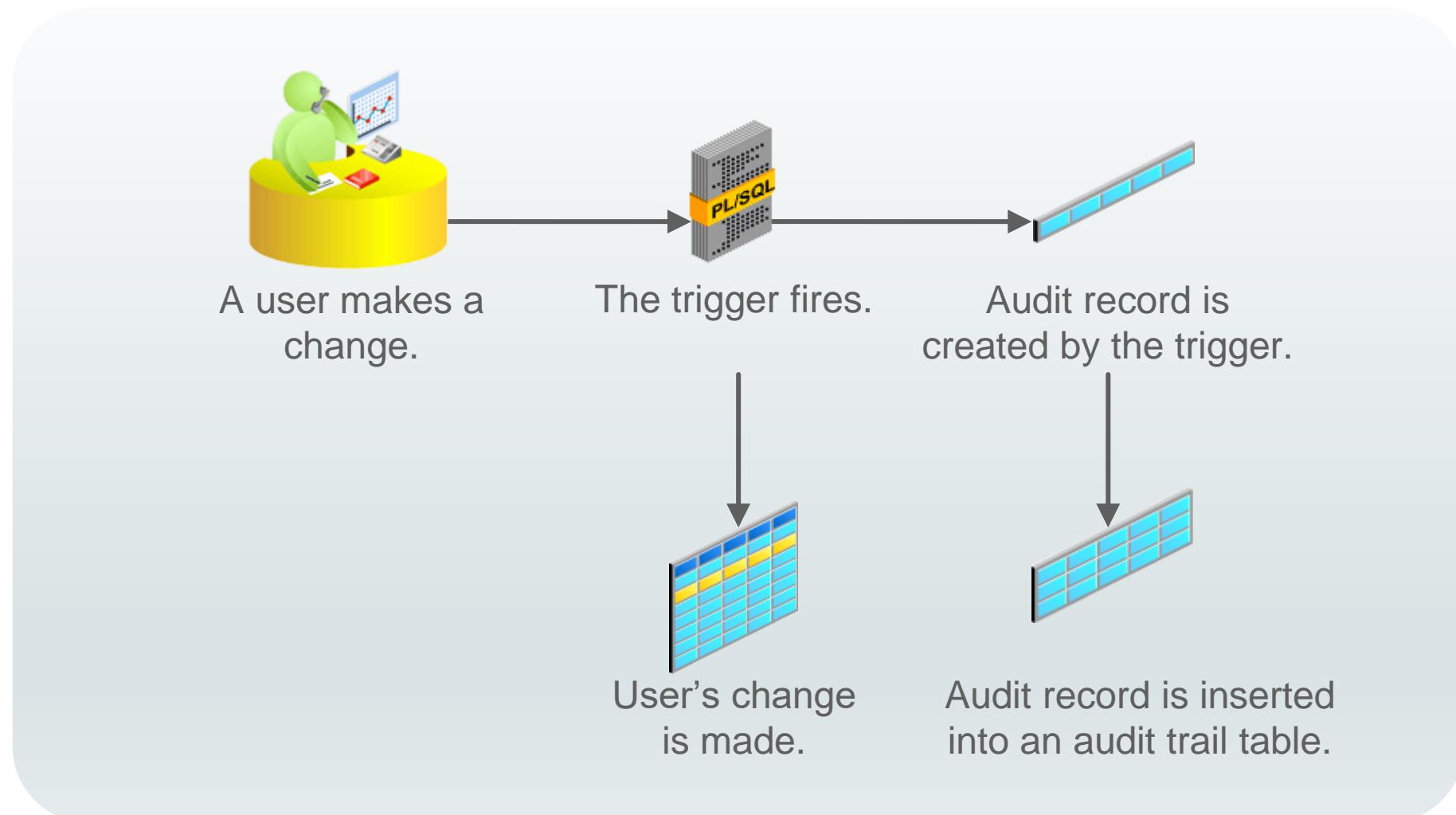
```
SQL> SELECT policy_name, audit_option, condition_eval_opt  
2  FROM    audit_unified_policies;
```

POLICY_NAME	AUDIT_OPTION	CONDITION_EVAL_OPT
POL1	DELETE	INSTANCE
POL2	TRUNCATE TABLE	NONE
POL3	RENAME	SESSION
POL4	ALL ACTIONS	STATEMENT

```
SQL> SELECT policy_name, enabled_opt, user_name, success, failure  
2  FROM    audit_unified_enabled_policies;
```

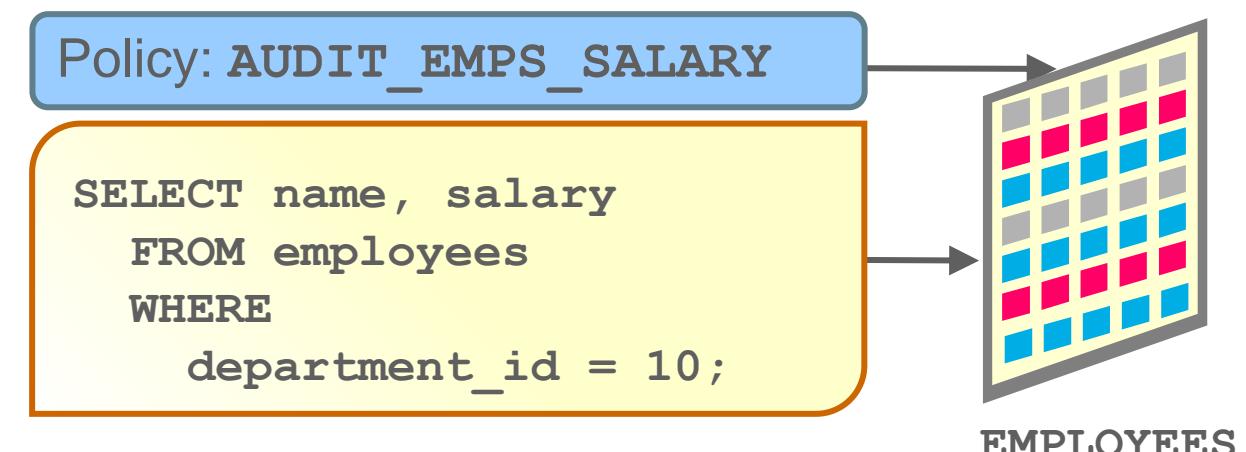
POLICY_NAME	ENABLED_USER_NAME	SUC	FAI
POL3	BY	PM	NO YES
POL2	EXCEPT	SYSTEM	NO YES
POL4	BY	SYS	YES
POL6	BY	ALL USERS	YES NO

Value-Based Auditing



Fine-Grained Auditing

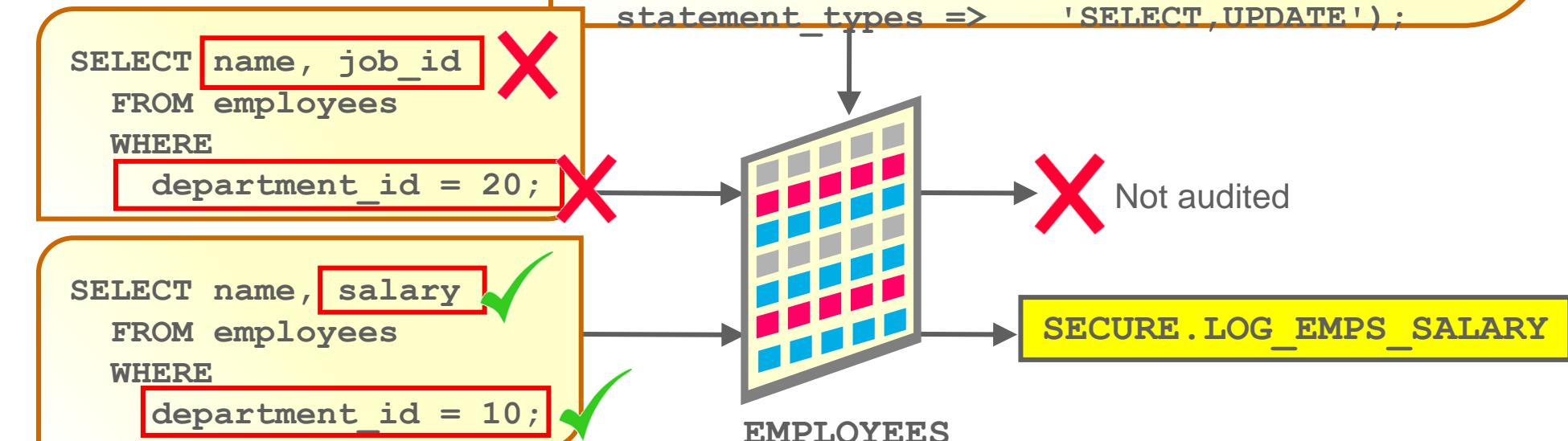
- Monitors data access on the basis of content
- Audits SELECT, INSERT, UPDATE, DELETE, and MERGE
- Can be linked to one or more columns in a table or view
- May execute a procedure
- Is administered with the DBMS_FGA package



FGA Policy

- Defines:
 - Audit criteria
 - Audit action
- Is created with
DBMS_FGA.ADD_POLICY

```
dbms_fga.add_policy (
    object_schema => 'HR',
    object_name => 'EMPLOYEES',
    policy_name => 'audit_emps_salary',
    audit_condition=> 'department_id=10',
    audit_column => 'SALARY,COMMISSION_PCT',
    handler_schema => 'secure',
    handler_module =>
        'log_emps_salary',
    enable => TRUE,
    statement_types => 'SELECT,UPDATE');
```



Audited DML Statement: Considerations

- Records are audited if the FGA predicate is satisfied and the relevant columns are referenced.
- DELETE statements are audited regardless of columns specified.
- MERGE statements are audited with the underlying INSERT, UPDATE, and DELETE generated statements.

```
UPDATE hr.employees  
SET salary = 1000  
WHERE commission_pct = .2;
```

Not audited because
none of the employees
are in department 10



```
UPDATE hr.employees  
SET salary = 1000  
WHERE employee_id = 200;
```

Audited because the
employee is in
department 10



FGA Guidelines

- To audit all rows, use a `null` audit condition.
- To audit all columns, use a `null` audit column.
- Policy names must be unique.
- The audited table or view must already exist when you create the policy.
- If the audit condition syntax is invalid, an ORA-28112 error is raised when the audited object is accessed.
- If the audited column does not exist in the table, no rows are audited.
- If the event handler does not exist, no error is returned and the audit record is still created.

Archiving and Purging the Audit Trail

- Periodically archive and purge the audit trail to prevent it from growing too large.
- Create an archive by:
 - Copying audit trail records to a database table
 - Using Oracle Audit Vault and Database Firewall
- Purge the audit trail by:
 - Creating and scheduling a purge job to run at a specified time by using the `DBMS_AUDIT_MGMT.CREATE_PURGE_JOB` PL/SQL procedure
 - Manually by using the `DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL` PL/SQL procedure

Purging Audit Trail Records

- Schedule an automatic purge job:

```
DBMS_AUDIT_MGMT.CREATE_PURGE_JOB  
(AUDIT_TRAIL_TYPE=> DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,  
AUDIT_TRAIL_PURGE_INTERVAL => 12,  
AUDIT_TRAIL_PURGE_NAME => 'Audit_Trail_PJ',  
USE_LAST_ARCH_TIMESTAMP => TRUE,  
CONTAINER => DBMS_AUDIT_MGMT.CONTAINER_CURRENT);
```

- Manually purge the audit records:

```
DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL(  
AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED)
```

Summary

In this lesson, you should have learned how to:

- Describe DBA responsibilities for security and auditing
- Enable unified auditing
- Create unified audit policies
- Maintain the audit trail

Practice Overview

- Enabling Unified Auditing
- Creating Audit Users
- Creating an Audit Policy

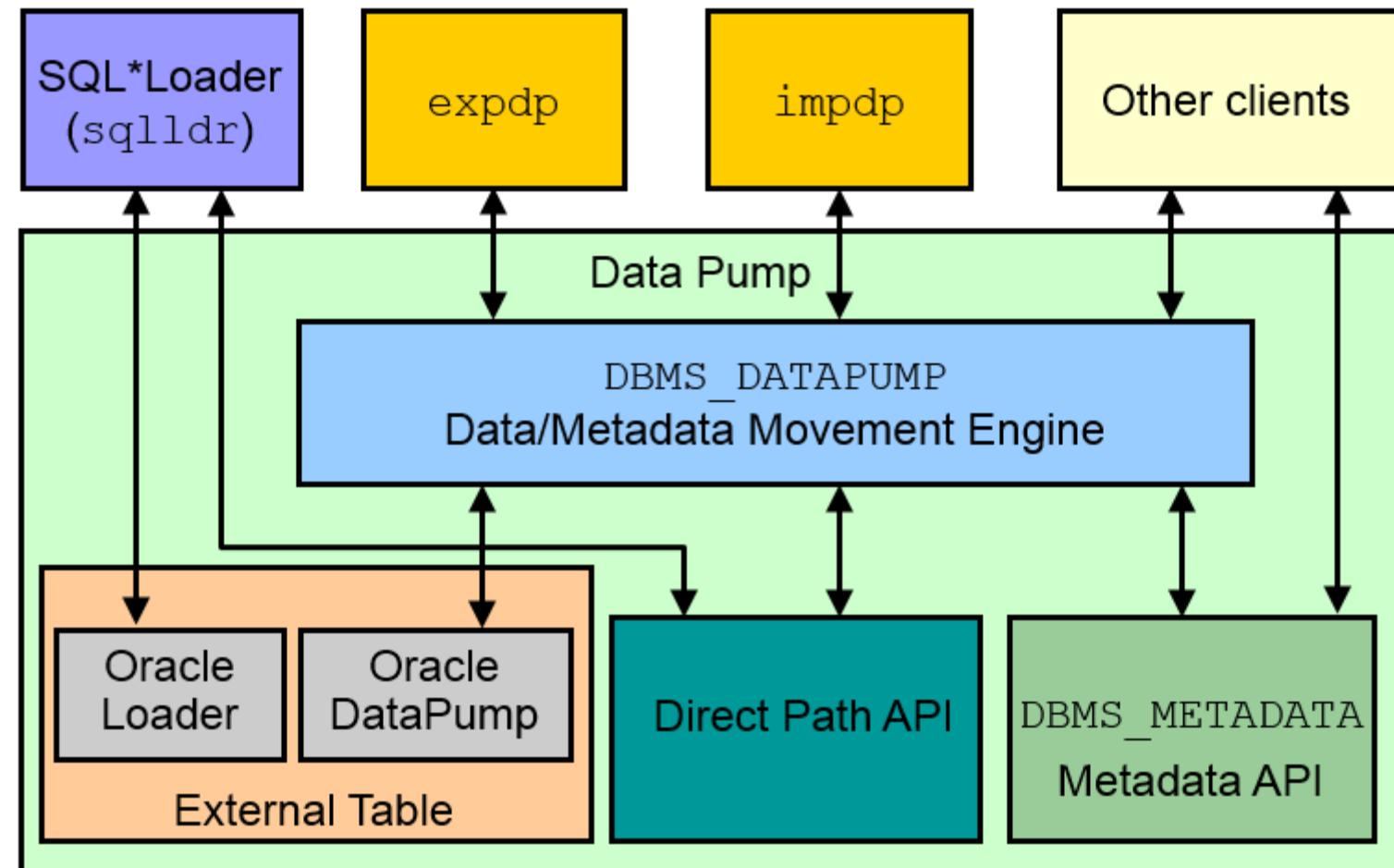
23. Introduction to Loading and Transporting Data

Objectives

After completing this lesson, you should be able to:

- Describe some ways to move data
- Explain the general architecture of Oracle Data Pump and SQL*Loader

Moving Data: General Architecture



Oracle Data Pump: Overview

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

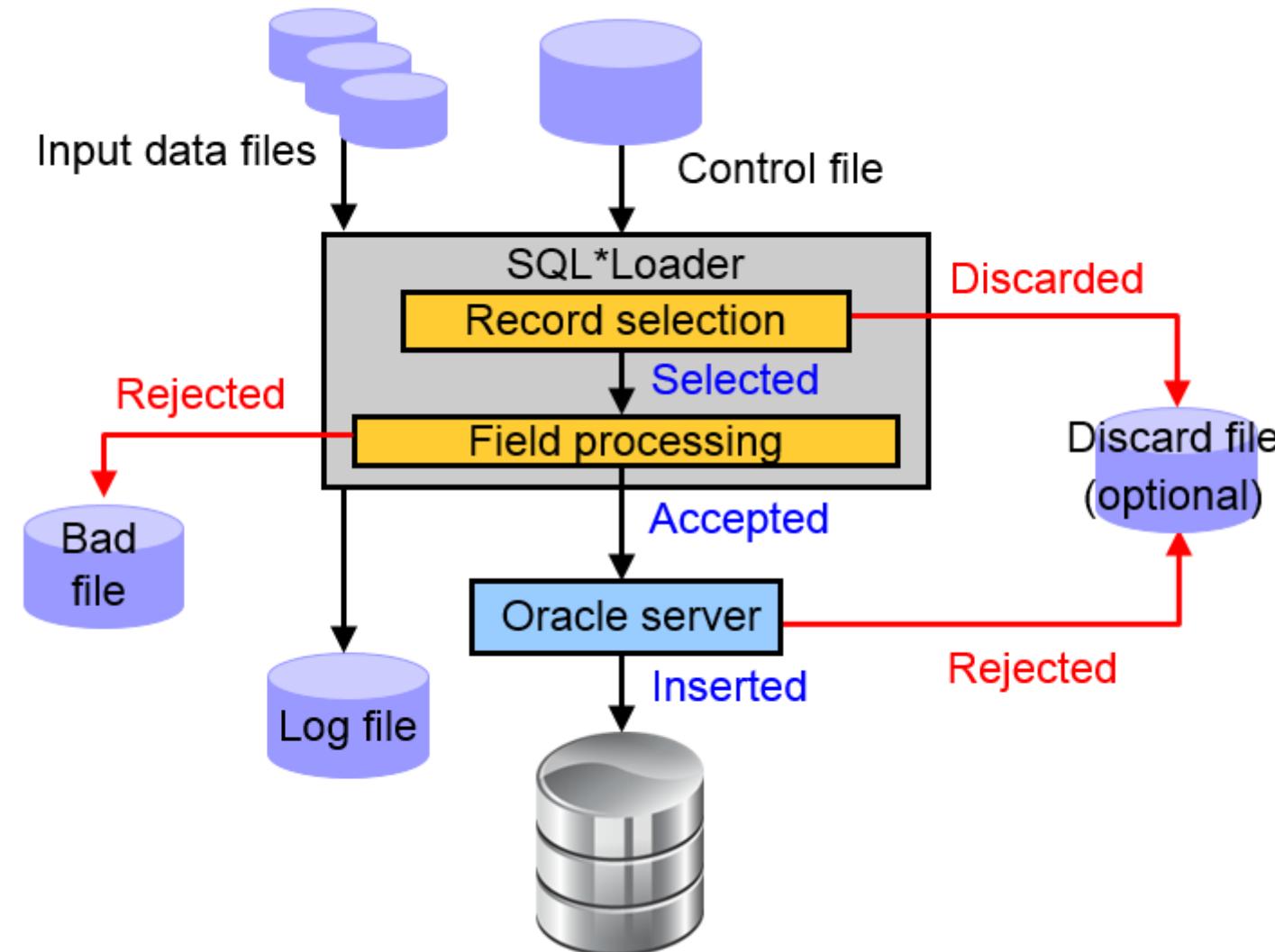
- Can be called via DBMS_DATAPUMP
- Provides the following tools:
 - expdp and impdp
 - GUI interface in Enterprise Manager Cloud Control
- Provides several data movement methods:
 - Conventional path load
 - Direct path
 - External tables
 - Transportable tablespace
 - Network link support
- Detaches from and reattaches to long-running jobs
- Restarts Data Pump jobs

Oracle Data Pump: Benefits

Data Pump offers many benefits and features, such as:

- Fine-grained object and data selection
- Explicit specification of database version
- Parallel execution
- Network mode in a distributed environment
- Remapping capabilities
- Data sampling and metadata compression
- Compression of data during a Data Pump export
- Security through encryption
- Ability to export XMLType data as CLOBs

SQL Loader: Overview



Summary

In this lesson, you should have learned how to:

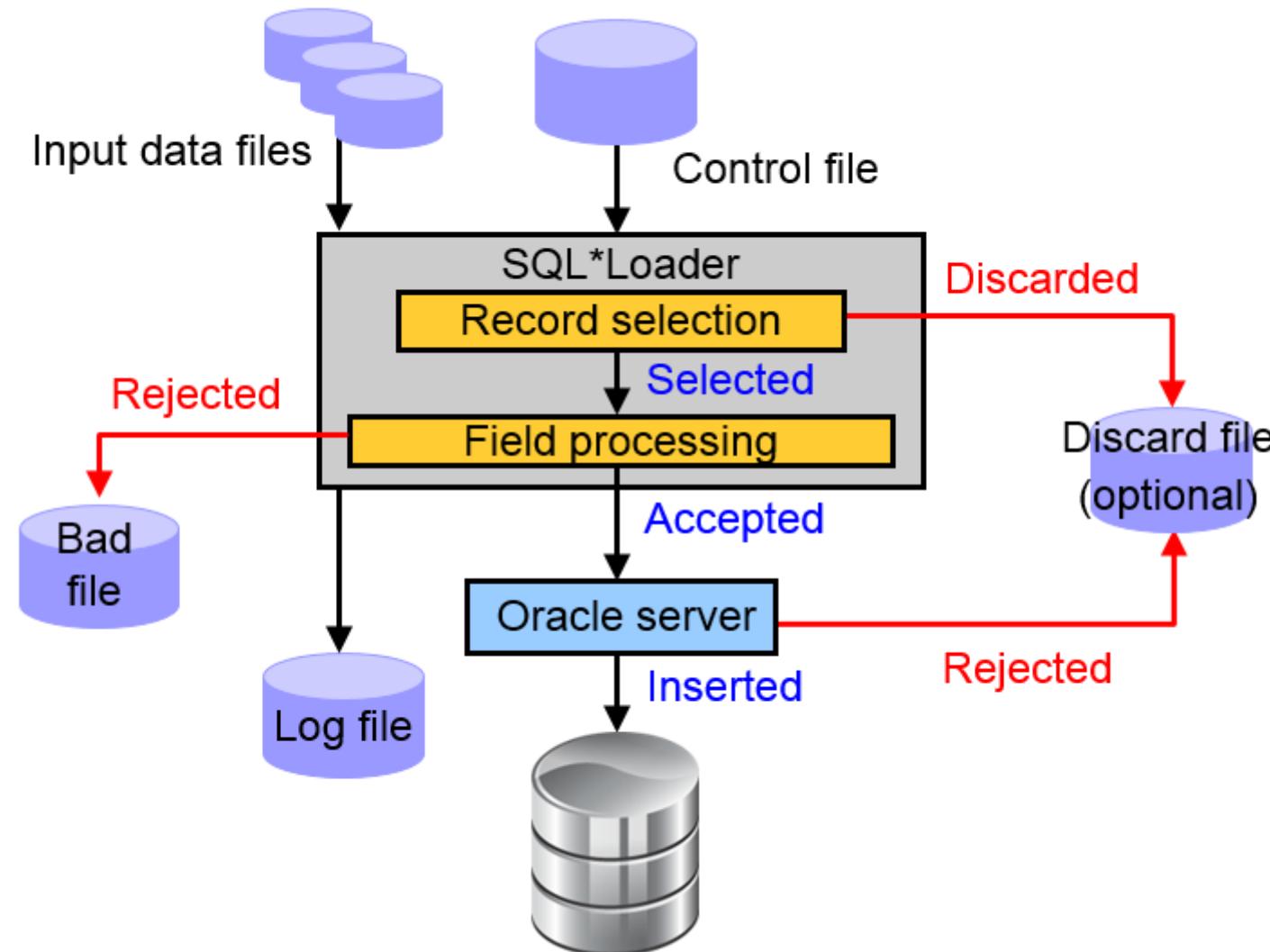
- Describe some ways to move data
- Explain the general architecture of Oracle Data Pump and SQL*Loader

24. Loading Data

Objectives

After completing this lesson, you should be able to use SQL*Loader to load data from a non-Oracle database (or user files).

SQL Loader: Review



Creating the SQL*Loader Control File

The SQL*Loader control file contains:

- Location of the data to be loaded
- Data format
- Configuration details:
 - Memory management
 - Record rejection
 - Interrupted load handling details
- Data manipulation details

```
1      -- This is a sample control file
2 LOAD DATA
3 INFILE 'SAMPLE.DAT'
4 BADFILE 'sample.bad'
5 DISCARDFILE 'sample.dsc'
6 APPEND
7 INTO TABLE emp
8 WHEN (57) = '. '
9 TRAILING NULLCOLS
10 (hiredate SYSDATE,
     deptno POSITION(1:2) INTEGER EXTERNAL(3)
     NULLIF deptno=BLANKS,
     ...
     empno POSITION(45) INTEGER EXTERNAL
     TERMINATED BY whitespace,
     ...
)
```

SQL*Loader Loading Methods

Conventional Load

Uses COMMIT

Always generates redo entries

Enforces all constraints

Fires INSERT triggers

Can load into clustered tables

Allows other users to modify tables during load operation

Maintains index entries on each insert

Direct Path Load

Uses data saves (faster operation)

Generates redo only under specific conditions

Enforces only PRIMARY KEY, UNIQUE, and NOT NULL constraints

Does not fire INSERT triggers

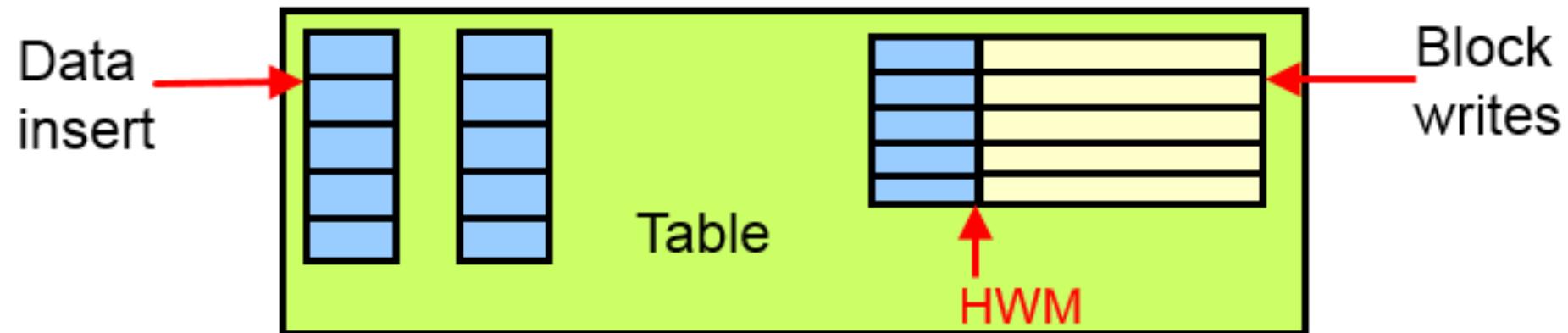
Does not load into clusters

Prevents other users from making changes to tables during load operation

Merges new index entries at the end of the load

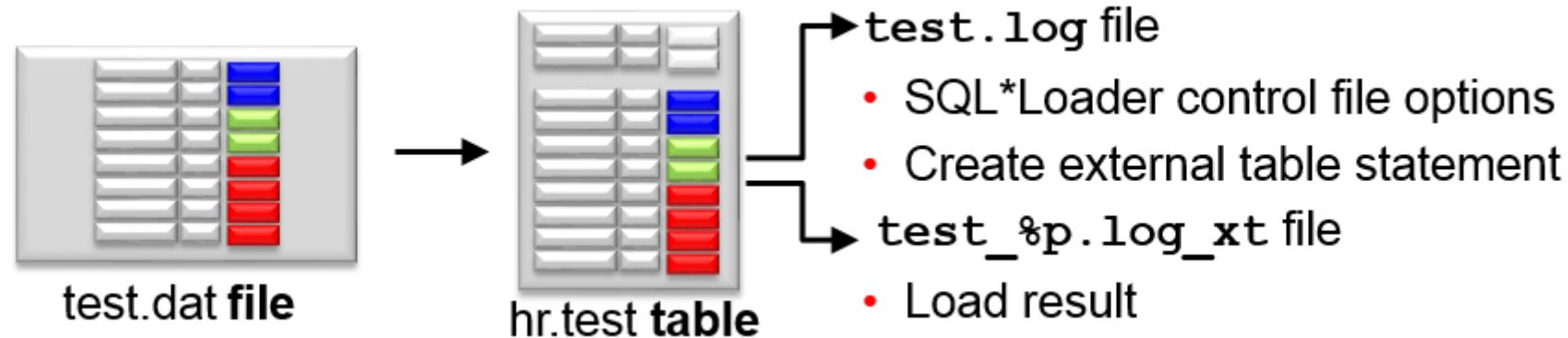
Protecting Against Data Loss

- Use data saves to protect against loss of data due to instance failure.
- Use the SQL*Loader ROWS parameter to specify when a data save should occur during a direct path load.

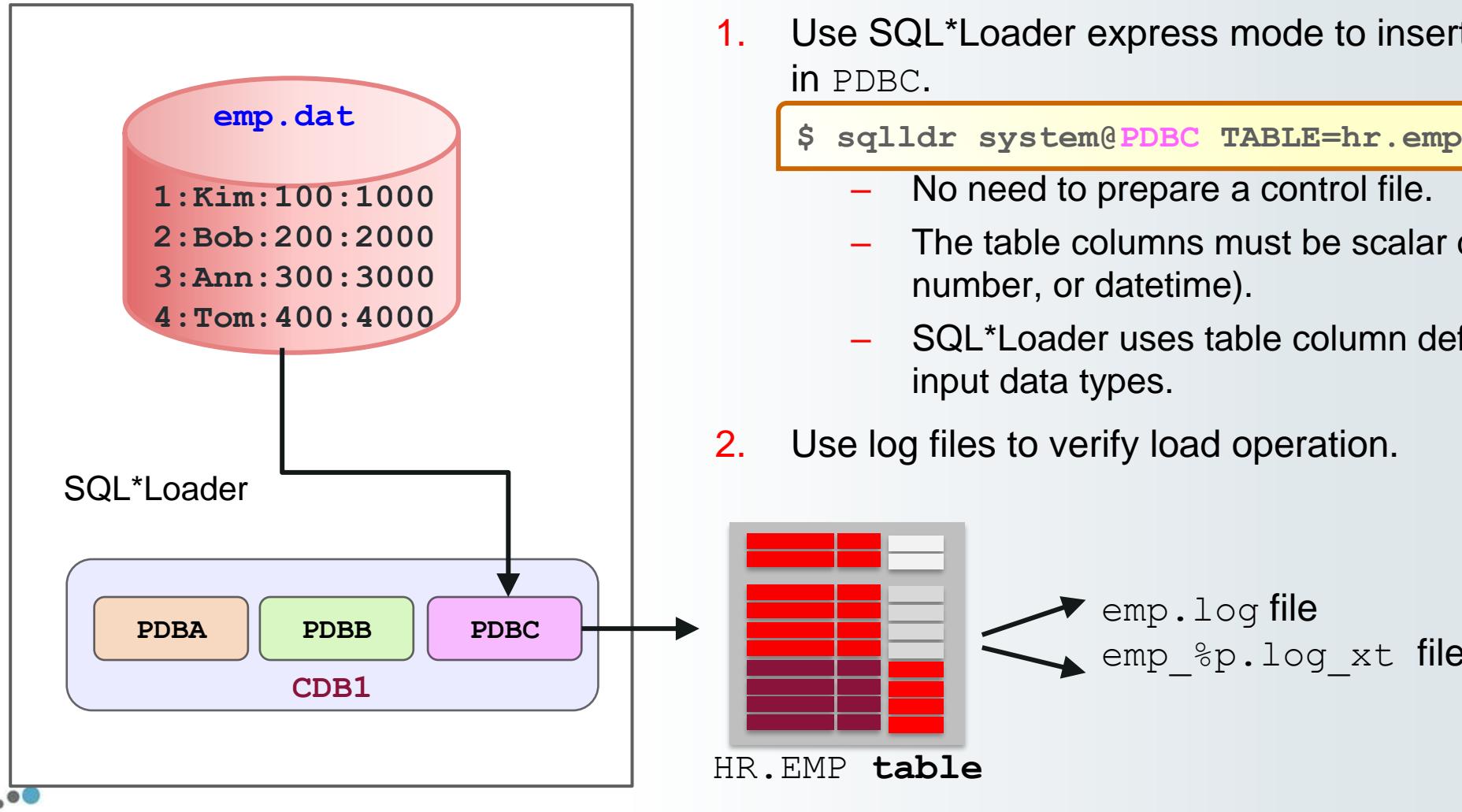


SQL*Loader Express Mode

- Specify a table name to initiate an express mode load.
- Table columns must be scalar data types (character, number, or datetime).
- A data file can contain only delimited character data.
- SQL*Loader uses table column definitions to determine input data types.
- There is no need to create a control file.



Using SQL*Loader to Load a Table in a PDB



1. Use SQL*Loader express mode to insert rows into `HR.EMP` in `PDBC`.

```
$ sqlldr system@PDBC TABLE=hr.emp
```

- No need to prepare a control file.
- The table columns must be scalar data types (character, number, or datetime).
- SQL*Loader uses table column definitions to determine input data types.

2. Use log files to verify load operation.

Summary

In this lesson, you should have learned how to use external tables to move data via platform-independent files.

Practice Overview

- Loading Data into a PDB from an External File

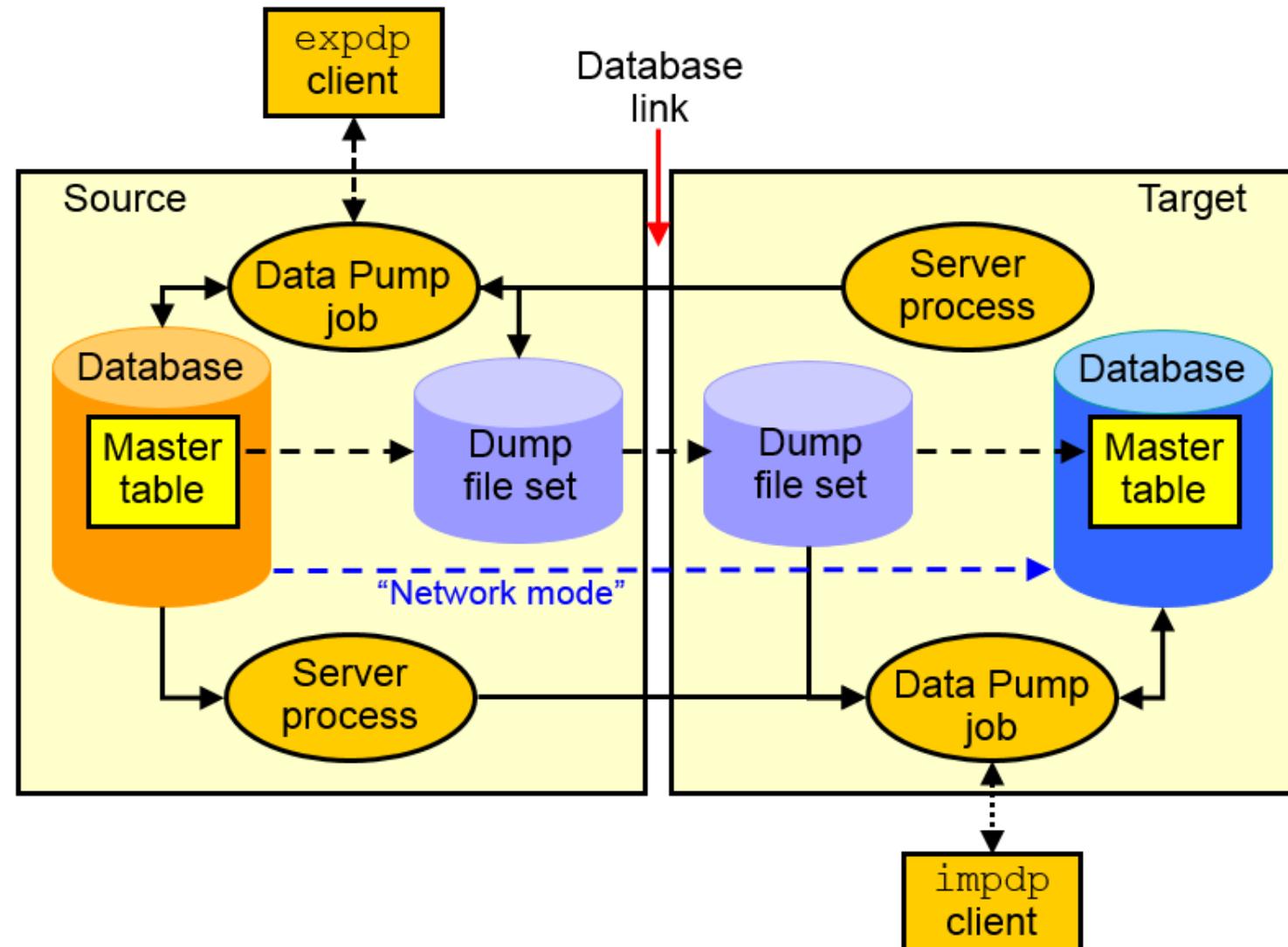
25. Transporting Data

Objectives

After completing this lesson, you should be able to:

- Explain the general architecture of Oracle Data Pump
- Use Data Pump Export and Import to move data between Oracle databases
- Transport tablespaces between databases by using image copies or backup sets
- Transport databases by using data files or backup sets

Data Pump Export and Import Clients



Data Pump Interfaces and Modes

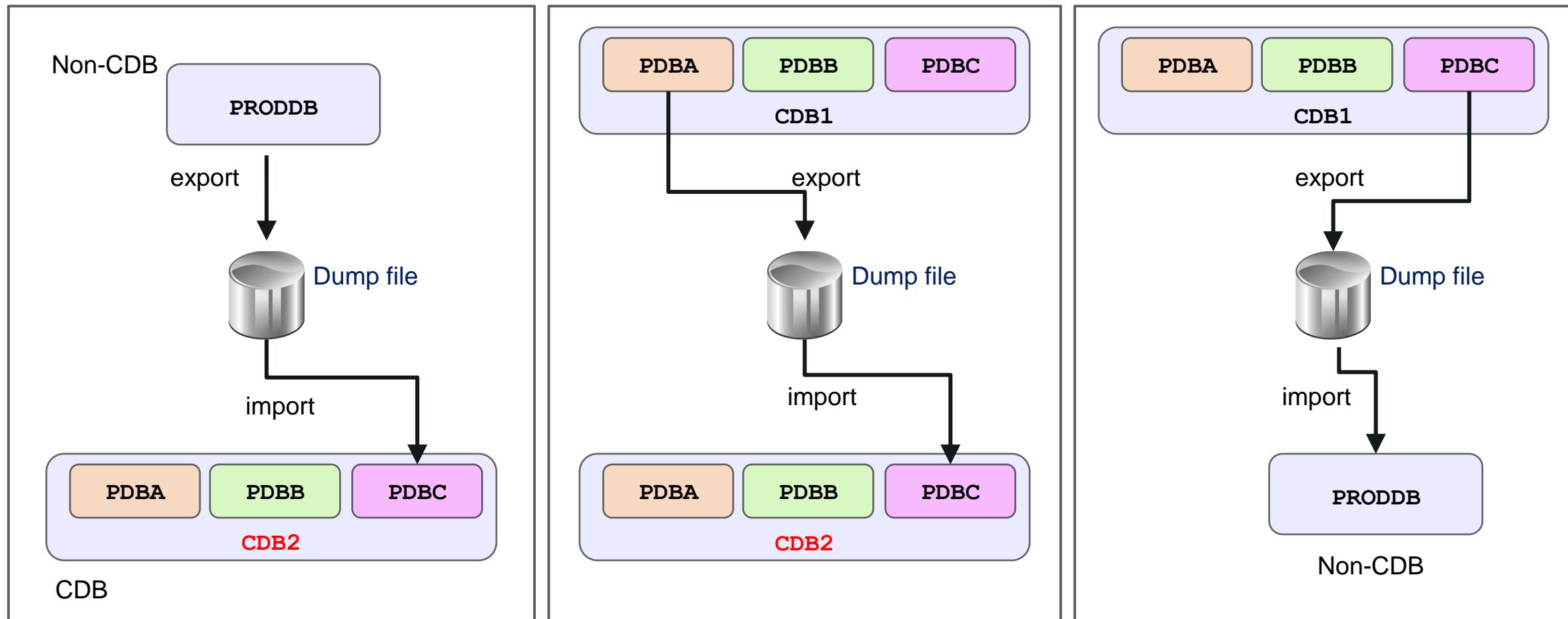
- Data Pump Export and Import interfaces:
 - Command line
 - Parameter file
 - Interactive command line
 - Enterprise Manager Cloud Control
- Data Pump Export and Import modes:
 - Full
 - Schema
 - Table
 - Tablespace
 - Transportable tablespace
 - Transportable database

Data Pump Import Transformations

You can remap:

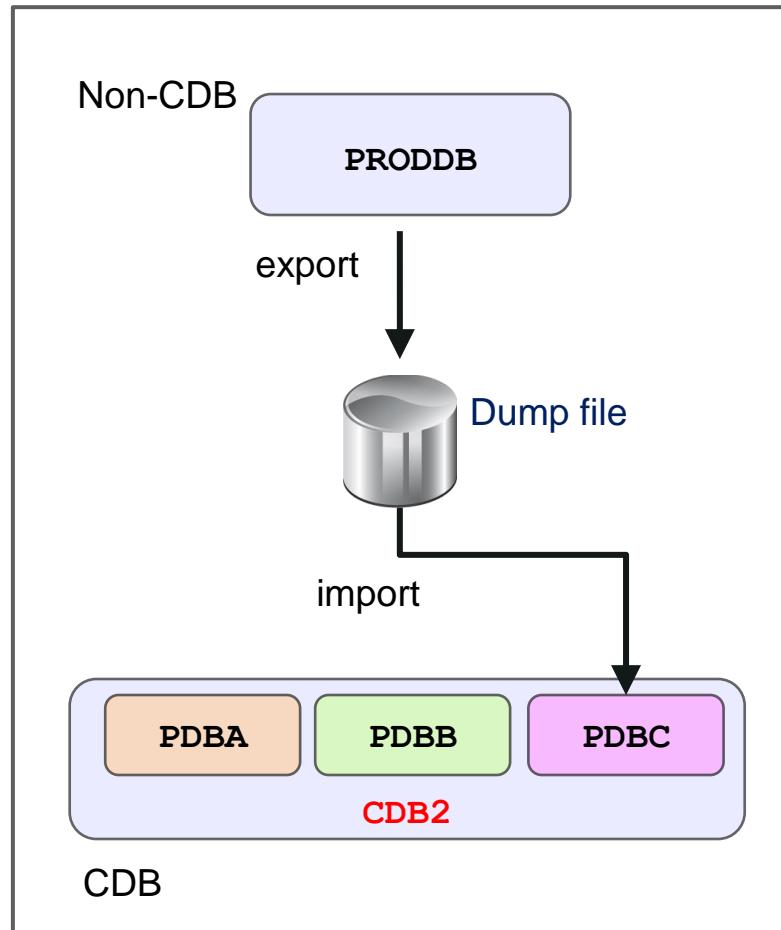
- Data files by using REMAP_DATAFILE
- Tablespaces by using REMAP_TABLESPACE
- Schemas by using REMAP_SCHEMA
- Tables by using REMAP_TABLE
- Data by using REMAP_DATA
- Directory by using REMAP_DIRECTORY

Using Oracle Data Pump with PDBs



Use the PDB service name to export from or import into a PDB.

Exporting from a Non-CDB and Importing into a PDB



1. Export PRODDB with the FULL=Y parameter:

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

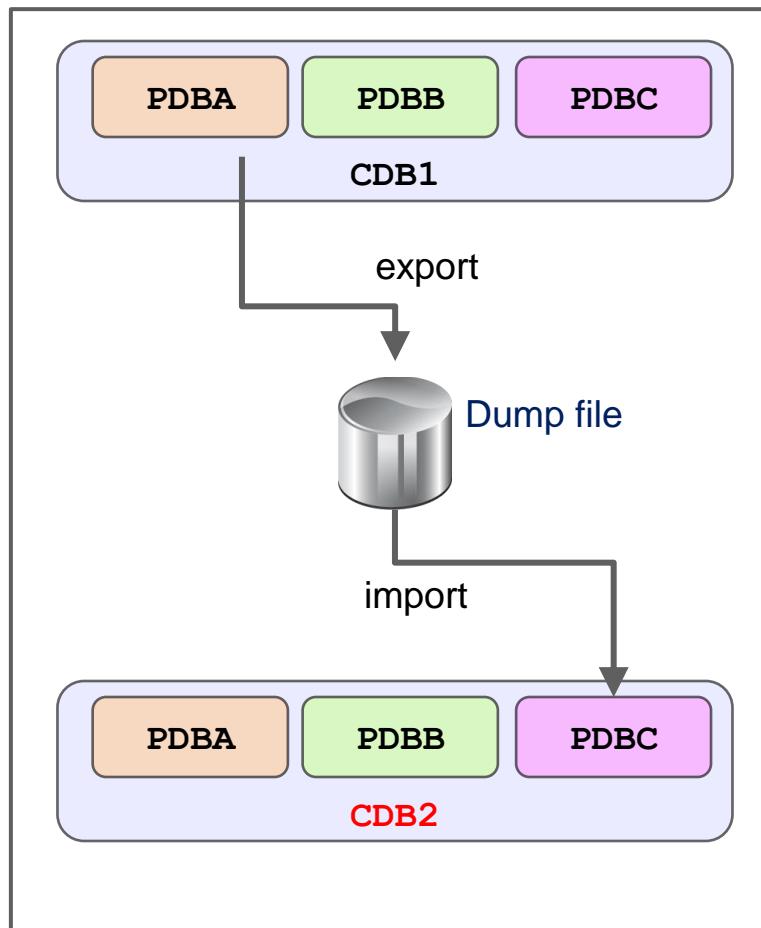
1. If PDBC does not exist in CDB2, create PDBC in CDB2:

```
SQL> CONNECT sys@CDB1
SQL> CREATE PLUGGABLE DATABASE PDBC ...;
```

1. Open PDBC.
2. Create a Data Pump directory in PDBC.
3. Copy the dump file to the Data Pump directory.
4. Create the same PRODDB tablespaces in PDBC for new local users' objects.
5. Import into PDBC with FULL=Y and REMAP parameter:

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

Exporting and Importing Between PDBs



1. Export PDBA from CDB1 with the FULL=Y parameter:

```
$ expdp system@PDBA FULL=Y ...
```

1. If PDBC does not exist in CDB2, create PDBC in CDB2:

```
SQL> CONNECT sys@CDB2  
SQL> CREATE PLUGGABLE DATABASE PDBC ...;
```

1. Open PDBC.
2. Create a Data Pump directory in PDBC.
3. Copy the dump file to the Data Pump directory.
4. Create the same PDBA tablespaces in PDBC for new local users' objects.
5. Import into PDBC of CDB2 with the FULL and REMAP parameters:

```
$ impdp system@PDBC FULL=Y REMAP_SCHEMA=c##u:lu...
```

Full Transportable Export/Import

- A full transportable export exports all objects and data necessary to create a complete copy of the database. Specify these parameter values:
 - TRANSPORTABLE=ALWAYS
 - FULL=Y

```
$ expdp user_name@pdb FULL=y DUMPFILE=expdat.dmp DIRECTORY=data_pump_dir  
TRANSPORTABLE=always
```

- A full transportable import imports a dump file only if it has been created using the transportable option during export.
 - TRANSPORT_DATAFILES
 - If the TRANSPORTABLE parameter is specified, the NETWORK_LINK parameter is required.

Full Transportable Export/Import: Example

Source Database

PRODDB



1

- Tablespaces read-only :
• APPL1
• HRTBS

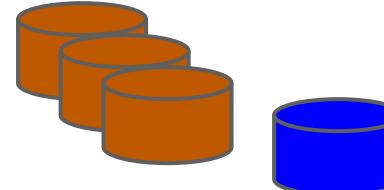
2

```
$ expdp user_name@proddb FULL=y  
DUMPFILE=expdat.dmp  
DIRECTORY=data_pump_dir  
TRANSPORTABLE=always  
LOGFILE=export.log
```

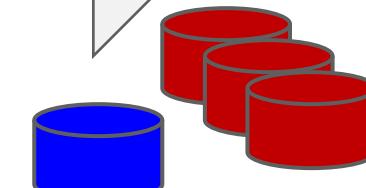
Endian conversion if necessary
RMAN CONVERT

Or → →
DBMS_FILE_TRANSFER

3



4



Data Files + dumpfile transport

5



Target Database

PDBPROD

```
$ impdp user_name@pdbprod FULL=y  
DUMPFILE=expdat.dmp  
DIRECTORY=data_pump_dir  
TRANSPORT_DATAFILES=  
'/oracle/oradata/prod/file1.dbf',  
'/oracle/oradata/prod/file2.dbf',  
'/oracle/oradata/prod/file3.dbf'  
LOGFILE=import.log
```

Transporting a Database Over the Network: Example

To transport a database over the network, perform an import using the `NETWORK_LINK` parameter.

1. Create a database link in the target to the source database.
2. Make the user-defined tablespaces in the source database read-only.
3. Transport the datafiles for all of the user-defined tablespaces from the source to the target location.
4. Perform conversion of the datafiles if necessary.
5. Import in the target database.

```
$ impdp username@dbname full=Y network_link = sourcedb  
    transportable = always  
    transport_datafiles = '/oracle/oradata/prod/sales01.dbf',  
                        '/oracle/oradata/prod/cust01.dbf'  
    logfile=import.log
```

Using RMAN to Transport Data Across Platforms

Transporting databases, datafiles, and tablespaces across platforms:

- Cross-platform transport (with different endian formats)
- Based on image copies and backup sets
- Use of inconsistent tablespace backups

Benefits:

- Reduced down time for platform migrations
- Choice of compression and multisection
- Not cataloged in control file, not used for regular restore operations

RMAN CONVERT Command

RMAN:

- Converts tablespaces, data files, or databases to the format of a destination platform
- Does not change input files
- Writes converted files to output destination
- Can convert on source or destination platform
- Assumes you initiate the data transfer

```
rman target sys@orcl
RMAN> ALTER TABLESPACE bartbs READ ONLY;

RMAN> CONVERT TABLESPACE bartbs
      TO PLATFORM 'Solaris Operating System (x86-64)'
      FORMAT '/tmp/transport/%U';
```

Transporting Data with Minimum Down Time

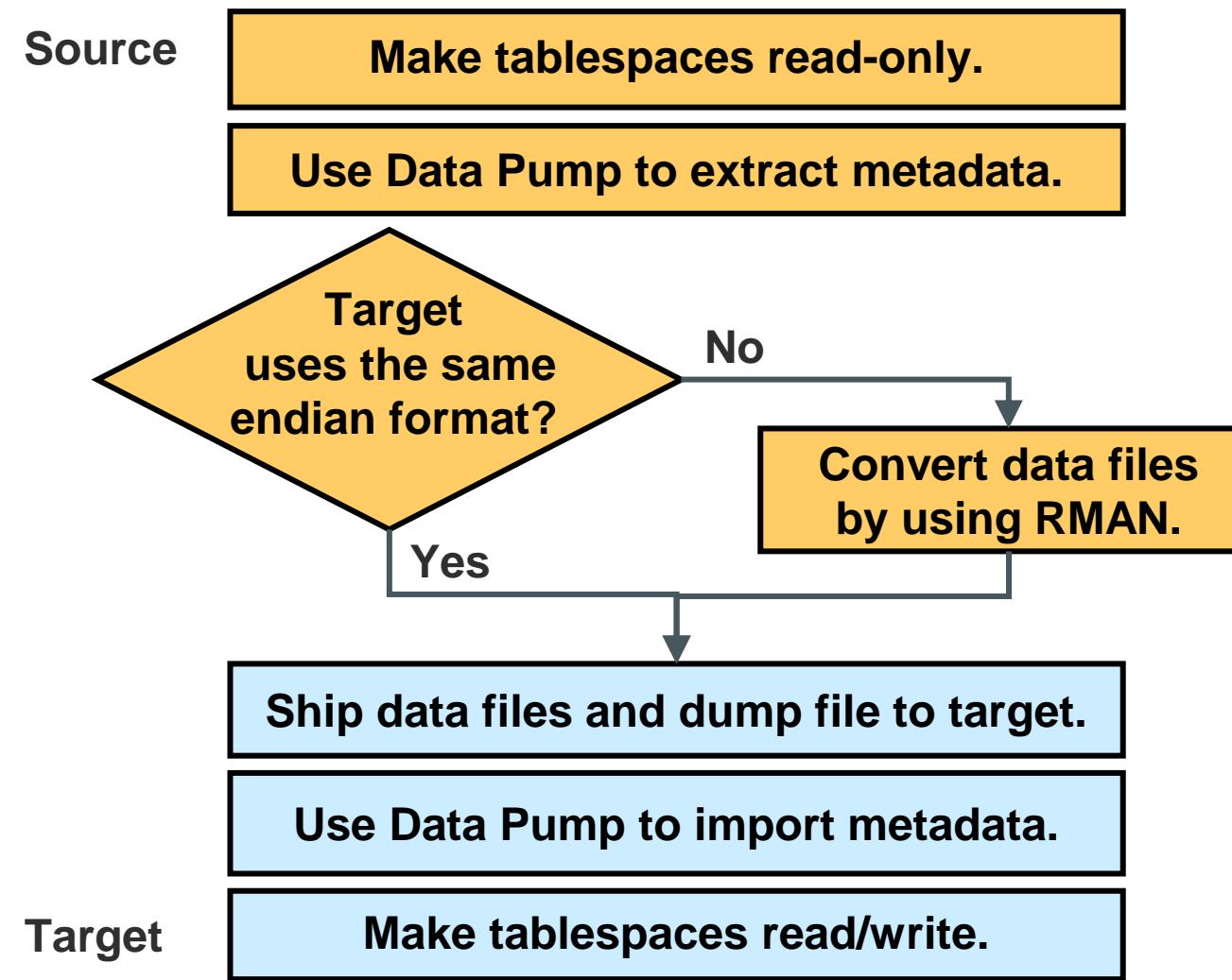
Consider the required database open mode and endian format:

- Database transport: READ ONLY, same endian format
- Tablespace transport: READ WRITE, different endian format

Example:

1. Create a database incremental level 0 backup and apply it to the destination.
2. Create incremental backups and apply them to the destination.
3. Repeat: Create and apply incremental backups.
4. Perform the final incremental backup in `READ ONLY` mode, apply it, and open both databases consistent with each other.

Transporting a Tablespace by Using Image Copies



Determining the Endian Format of a Platform

- Cross-platform transportable tablespaces:
 - Simplify moving data between data warehouse and data marts
 - Allow database migration from one platform to another
 - Allow the same character set on source and target platforms
- List of supported platforms and their endian formats:

```
SELECT * FROM V$TRANSPORTABLE_PLATFORM;
```

- Determine the endian format of source and target platforms:

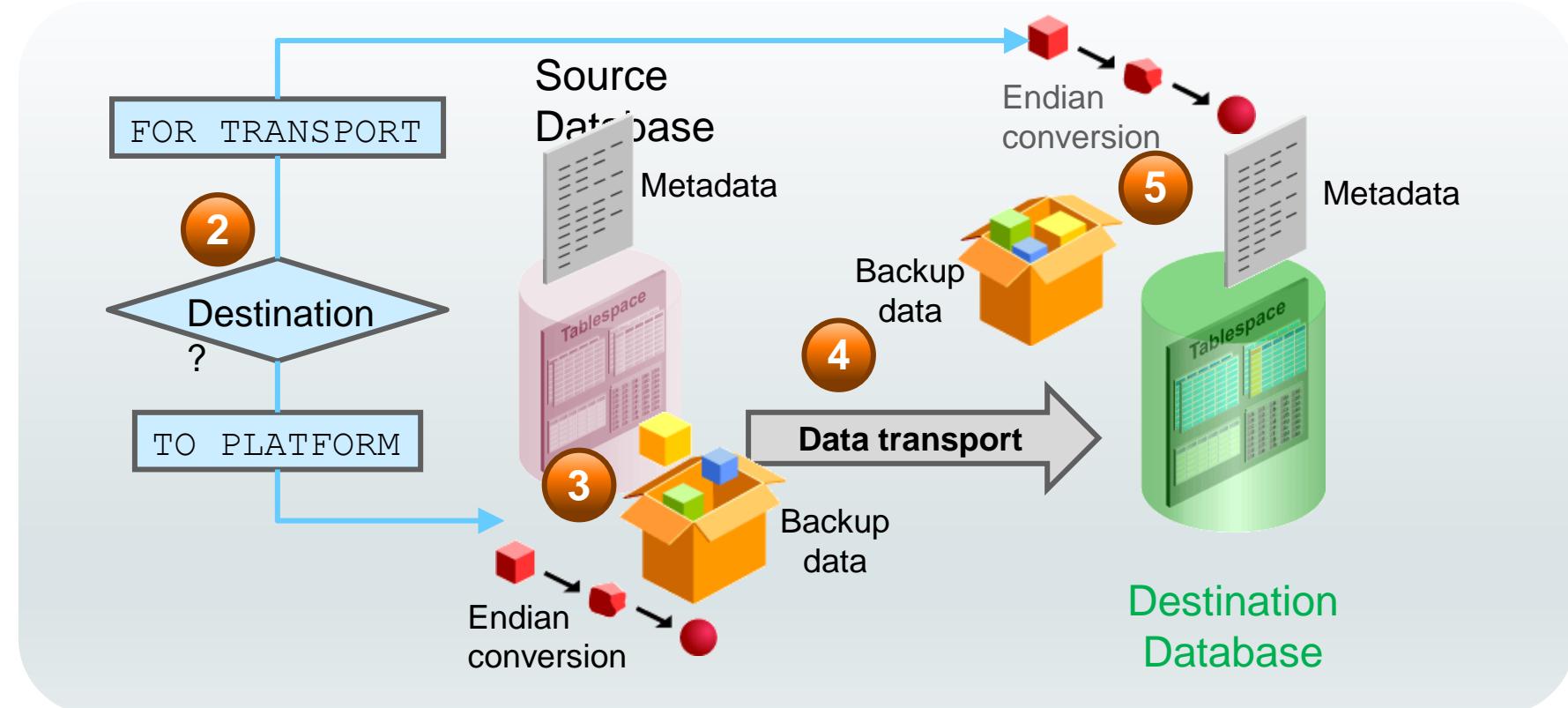
```
SELECT tp.endian_format
FROM v$transportable_platform tp, v$database sp
WHERE tp.platform_name = sp.platform_name;
```

Transporting Data with Backup Sets



Prerequisites

- COMPATIBLE=12.0 (or greater)
- READ ONLY mode for creation of cross-platform database backup
- Database in READ WRITE mode for creation of cross-platform tablespace backup



Transporting a Tablespace

Metadata



Source Database

1. Verify the prerequisites.
2. Start an RMAN session in the source database.
3. Query the exact name of the destination platform.
4. Change the tablespace to **read-only**.

```
RMAN> ALTER TABLESPACE test READ ONLY;
```

1. Perform a cross-platform transportable backup and a Data Pump export.

- Conversion on the destination host

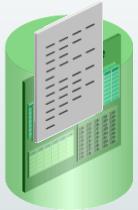
```
RMAN> BACKUP FOR TRANSPORT FORMAT '/bkp/test.bck'  
      DATAPUMP FORMAT '/bkp/test_meta.bck' TABLESPACE test;
```

- Conversion on the source host

```
RMAN> BACKUP TO PLATFORM 'HP Tru64 UNIX'  
      FORMAT '/bkp/test.bck'  
      DATAPUMP FORMAT '/bkp/test_meta.bck' TABLESPACE test;
```

Transporting a Tablespace

Metadata



Destination Database

6. Move the backup sets and the Data Pump export dump file to the destination host.
7. Connect to the destination host as TARGET.
8. Restore the cross-transportable backup and the Data Pump export.

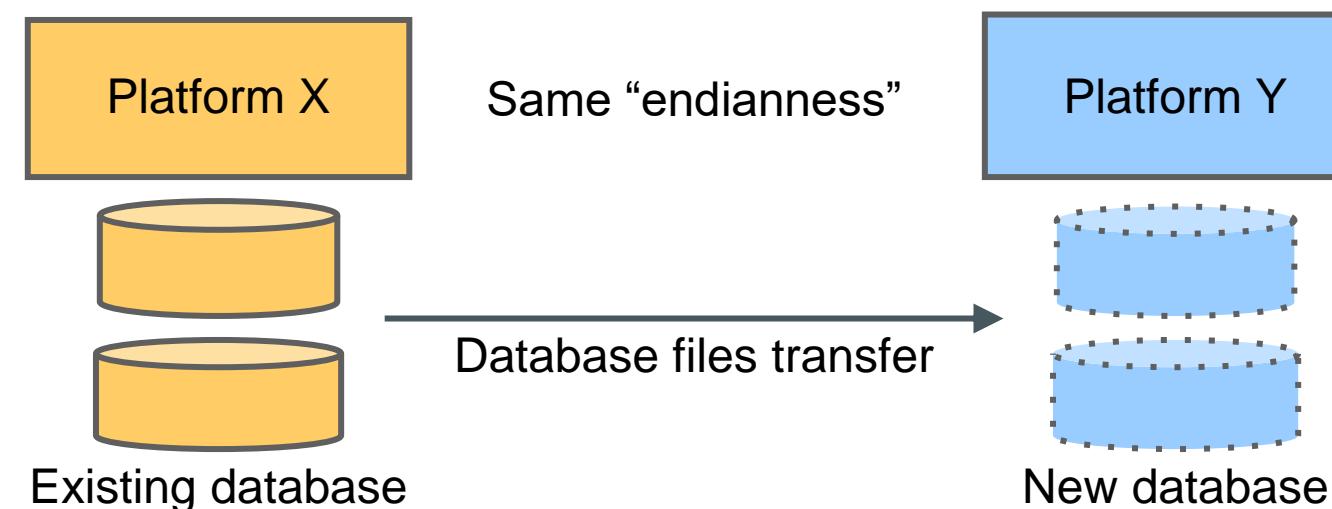
```
RMAN> RESTORE FOREIGN TABLESPACE test  
      FORMAT '/oracle/test.dbf'  
      FROM BACKUPSET '/bkp/test.bck'  
      DUMP FILE FROM BACKUPSET '/bkp/test_meta.bck' ;
```

Transporting Inconsistent Tablespaces

- Create cross-platform inconsistent incremental backups with the `ALLOW INCONSISTENT` clause.
- Restore the inconsistent cross-platform tablespace backup with the `RESTORE FOREIGN TABLESPACE` command.
- Recover restored data files copies with cross-platform incremental backups with the `RECOVER FOREIGN DATAFILECOPY` command.

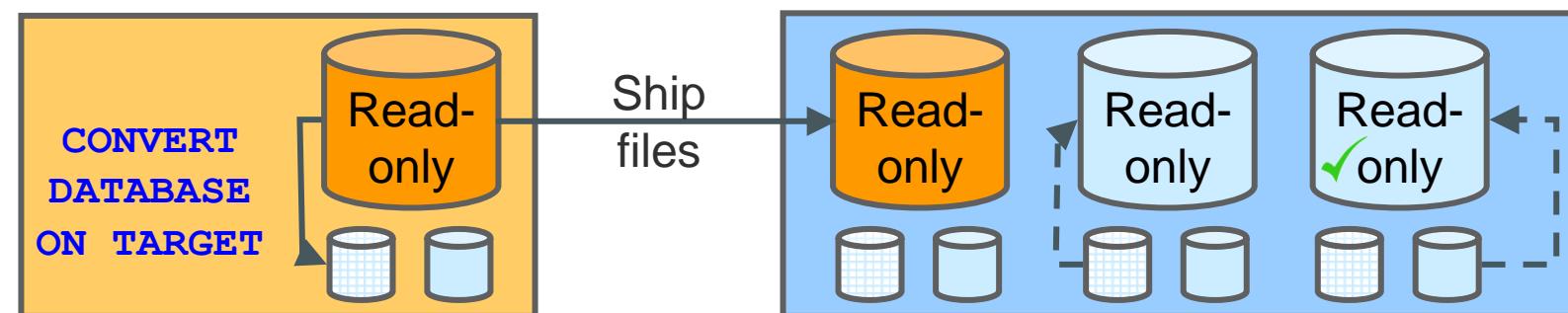
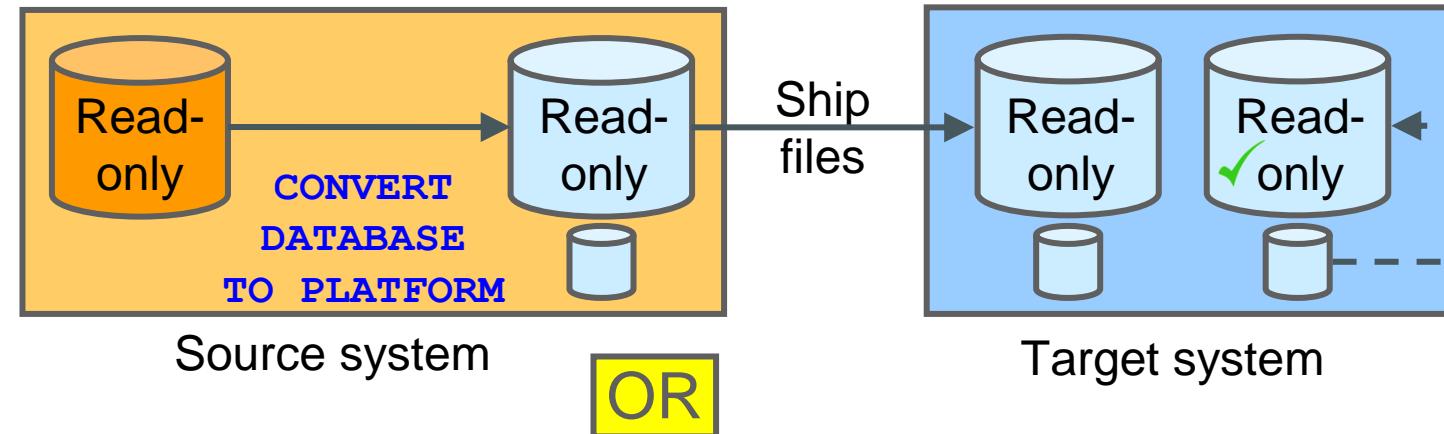
Database Transport: Data Files

- Generalize the transportable tablespace feature.
- Data can easily be distributed from a data warehousing environment to data marts, which are usually on smaller platforms.
- A database can be migrated from one platform to another very quickly.

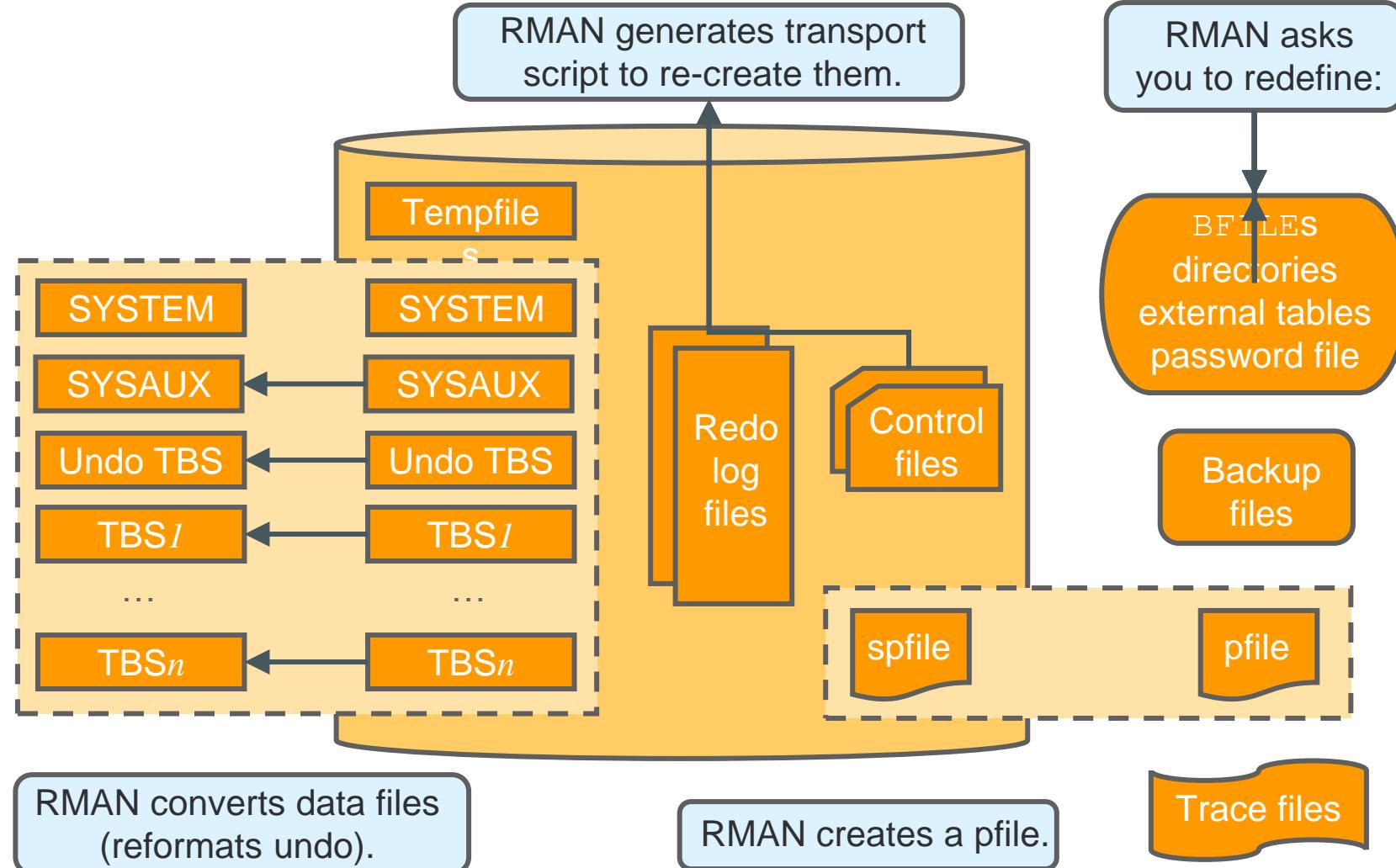


Transporting a Database

Open database in **READ ONLY** mode



Transporting a Database: Conversion



Transporting a Database: Example 1

1

```
SQL> startup mount;
SQL> alter database open read only;
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
db_ready BOOLEAN;
BEGIN
db_ready := DBMS_TDB.CHECK_DB('Microsoft Windows IA (32-bit)');
END;
/
SQL> host rman target=/
RMAN> CONVERT DATABASE TRANSPORT SCRIPT 'crdb.sql' NEW DATABASE
'newdb' TO PLATFORM 'Microsoft Windows IA (32-bit)' FORMAT '/tmp/%U';
```

Source

2

3

4 Ship data files, PFILE, and crdb.sql

5

```
$ sqlplus / as sysdba
SQL> @crdb.sql
```

Target

Transporting a Database: Example 2

```
1 $ sqlplus / as sysdba  
SQL> startup mount;  
SQL> alter database open read only;  
SQL> SET SERVEROUTPUT ON  
SQL> DECLARE  
2 db_ready BOOLEAN;  
BEGIN  
db_ready := DBMS_TDB.CHECK_DB('Microsoft Windows IA (32-bit)');  
END;  
/  
SQL> host rman target=/  
3 RMAN> CONVERT DATABASE ON TARGET PLATFORM CONVERT SCRIPT 'cnvt.sql'  
TRANSPORT SCRIPT 'crdb.sql' NEW DATABASE 'newdb' FORMAT '/tmp/%U';
```

4 Ship data files, PFILE, cnvt.sql, crdb.sql

```
5 $ sqlplus / as sysdba  
SQL> host rman target=/  
RMAN> @cnvt.sql  
6 RMAN> exit;  
SQL> @crdb.sql
```

Transporting a Database: Considerations

- Create the password file on the target platform.
- Transport the BFILEs used in the source database.
- The generated PFILE and transport script use Oracle Managed Files (OMF).
- Use DBNEWID to change the DBID.

Transporting a Database with Backup Sets

Metadata



Source Database

1. Verify the prerequisites:
 - COMPATIBLE: Greater or equal 12.0
 - OPEN_MODE: Read only

2. Start an RMAN session to connect to the source database.

```
RMAN> CONNECT TARGET sys/p@orcl
```

1. Query the exact name of the destination platform in V\$TRANSPORTABLE_PLATFORM.
2. Back up the source database:

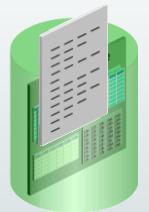
```
RMAN> BACKUP TO PLATFORM='HP Tru64 UNIX'  
FORMAT '/bkp_dir/trans_U%' DATABASE;
```

Or:

```
RMAN> BACKUP FOR TRANSPORT FORMAT '/bkp_dir/trans_U%'  
DATABASE;
```

Transporting a Database with Backup Sets

Metadata



Destination Database

5. Disconnect from the source database.
6. Move the backup sets and the Data Pump export dump file to the destination host.
7. Connect to the destination host as TARGET.

```
RMAN> CONNECT TARGET sys/p@orc12
```

7. Restore the full backup set with the RESTORE command.

```
RMAN> RESTORE FOREIGN DATABASE TO NEW  
      FROM BACKUPSET '/bkp_dir/trans_U%';
```

Summary

In this lesson, you should have learned how to:

- Explain the general architecture of Oracle Data Pump
- Use Data Pump Export and Import to move data between Oracle databases
- Transport tablespaces between databases by using image copies or backup sets
- Transport databases by using data files or backup sets

Practice Overview

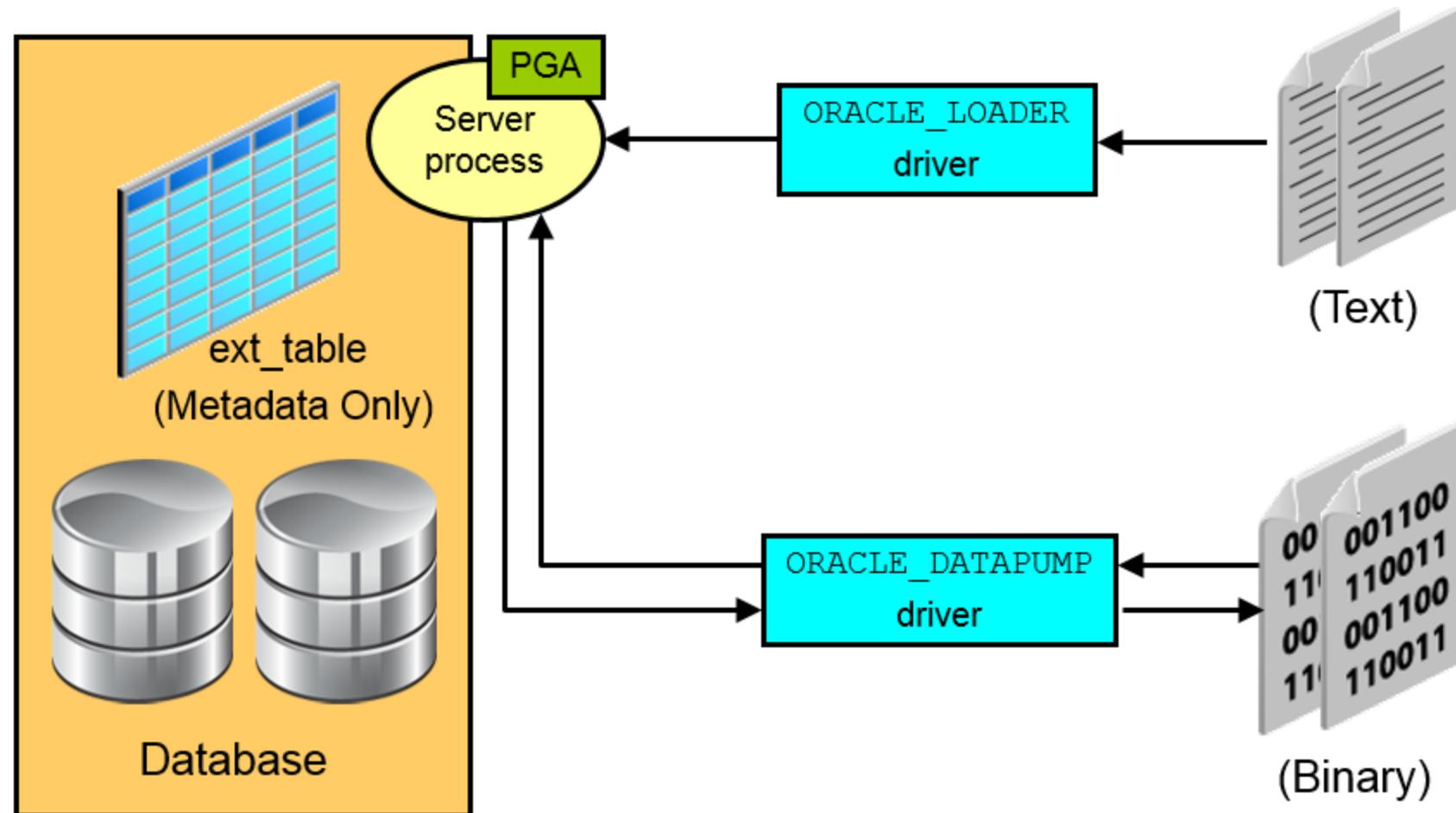
- Moving Data from One PDB to Another PDB
- Transporting a Tablespace

26. Using External Tables to Load and Transport Data

Objectives

After completing this lesson, you should be able to use external tables to move data via platform-independent files.

External Tables



External Tables: Benefits

- Data can be used directly from the external file or loaded into another database.
- External data can be queried and joined directly in parallel with tables residing in the database, without requiring it to be loaded first.
- The results of a complex query can be unloaded to an external file.
- You can combine generated files from different sources for loading purposes.

ORACLE_LOADER Access Driver

```
CREATE TABLE extab_employees
  (employee_id          NUMBER(4),
   first_name           VARCHAR2(20),
   last_name            VARCHAR2(25),
   hire_date             DATE)
  ORGANIZATION EXTERNAL
    (TYPE ORACLE_LOADER
     DEFAULT DIRECTORY extab_dat_dir
     ACCESS PARAMETERS
       (records delimited by newline
        badfile extab_bad_dir:'empxt%a_%p.bad'
        logfile extab_log_dir:'empxt%a_%p.log'
        fields terminated by ','
        missing field values are null
        (employee_id, first_name, last_name,
         hire_date char date_format date mask "dd-mon-yyyy"))
     LOCATION ('empxt1.dat', 'empxt2.dat'))
PARALLEL REJECT LIMIT UNLIMITED;
```

ORACLE_DATAPUMP Access Driver

```
CREATE TABLE ext_emp_query_results
(first_name, last_name, department_name)
ORGANIZATION EXTERNAL
(
  TYPE ORACLE_DATAPUMP
  DEFAULT DIRECTORY ext_dir
  LOCATION ('emp1.exp','emp2.exp','emp3.exp')
)
PARALLEL
AS
SELECT e.first_name,e.last_name,d.department_name
FROM employees e, departments d
WHERE e.department_id = d.department_id AND
      d.department_name in ('Marketing', 'Purchasing');
```

External Tables

- Querying an external table:

```
SELECT * FROM extab_employees;
```

- Querying and joining an external table with an internal table:

```
SELECT e.employee_id, e.first_name, e.last_name, d.department_name  
FROM departments d, extab_employees e  
WHERE d.department_id = e.department_id;
```

- Appending data from an external table to an internal table:

```
INSERT /*+ APPEND */ INTO hr.employees  
SELECT * FROM extab_employees;
```

Viewing Information About External Tables

Dictionary Views	Description
[DBA ALL USER] _EXTERNAL_TABLES	Specific attributes
[DBA ALL USER] _EXTERNAL_LOCATIONS	Data sources
[DBA ALL USER] _TABLES	All tables
[DBA ALL USER] _TAB_COLUMNS	Columns of tables
[DBA ALL] _DIRECTORIES	Directory objects

Summary

In this lesson, you should have learned how to use external tables to move data via platform-independent files.

Practice Overview

- Querying External Tables
- Unloading External Tables

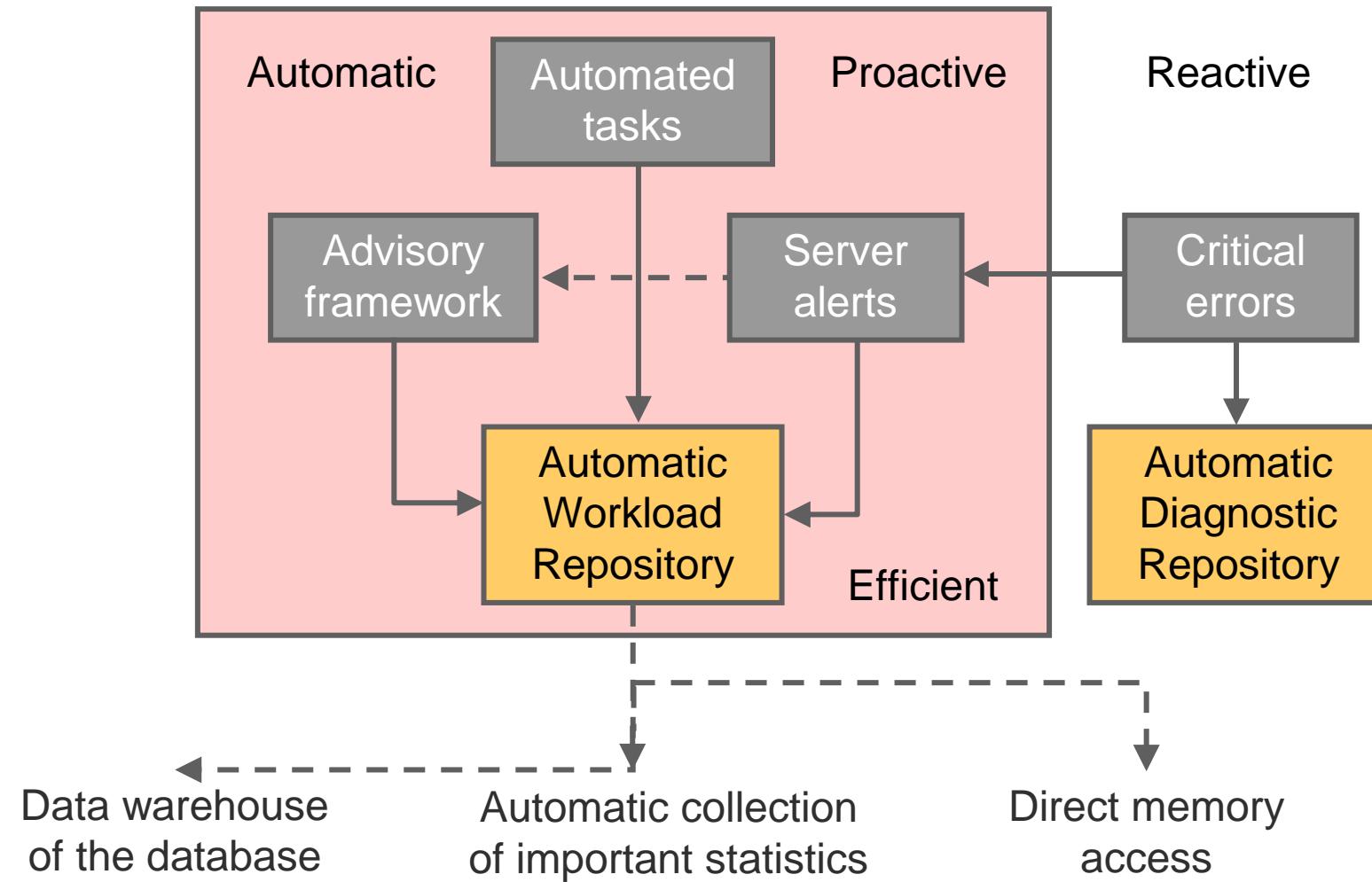
27. Automated Maintenance Tasks: Overview

Objectives

After completing this lesson, you should be able to:

- Describe Oracle Database's proactive database maintenance infrastructure
- Discuss automated maintenance tasks
- Explain maintenance windows

Proactive Database Maintenance Infrastructure



Automated Maintenance Tasks: Components

- Automated maintenance tasks: Tasks that are started automatically at regular intervals (maintenance windows) to perform maintenance operations on the database
- Maintenance windows: Predefined time intervals that are intended to occur during a period of low system load
- Oracle Scheduler
 - An enterprise job scheduler
 - A job is created for each maintenance task scheduled to run in a maintenance window when it opens.
- Oracle Database Resource Manager
 - Enables you to manage resource allocation for a database
 - Predefined maintenance windows use the `DEFAULT_MAINTENANCE_PLAN` resource plan by default.

Predefined Automated Maintenance Tasks

Task	Description
Automatic Optimizer Statistics Collection	Collects statistics for all schema objects which have no statistics or only stale statistics
Optimizer Statistics Advisor	Analyzes how statistics are being gathered and suggests changes
Automatic Segment Advisor	Identifies segments that have reclaimable space and makes defragmenting recommendations
Automatic SQL Tuning Advisor	Examines the performance of high-load SQL statements and makes tuning recommendations
SQL Plan Management (SPM) Evolve Advisor	Evolves plans that have recently been added to the SQL plan baseline

Maintenance Windows

Maintenance windows are:

- Contiguous time intervals during which automated maintenance tasks are run
- Oracle Scheduler windows that belong to the window group named `MAINTENANCE_WINDOW_GROUP`

Predefined Maintenance Windows

Task	Description
MONDAY_WINDOW	Starts at 10 PM on Monday and ends at 2 AM
TUESDAY_WINDOW	Starts at 10 PM on Tuesday and ends at 2 AM
WEDNESDAY_WINDOW	Starts at 10 PM on Wednesday and ends at 2 AM
THURSDAY_WINDOW	Starts at 10 PM on Thursday and ends at 2 AM
FRIDAY_WINDOW	Starts at 10 PM on Friday and ends at 2 AM
SATURDAY_WINDOW	Starts at 6 AM on Saturday and is 20 hours long
SUNDAY_WINDOW	Starts at 6 AM on Sunday and is 20 hours long

Viewing Maintenance Window Details

Maintenance Window Group

10 PM – 2 AM Mon to Fri

6 AM – 2 AM Sat to Sun

The screenshot shows the Oracle Enterprise Manager Cloud Control 13c interface. The top navigation bar includes links for Oracle Database, Performance, Availability, Security, Schema, Administration, and Scheduler Windows. The Scheduler Windows page displays a list of system windows with the following details:

Select	Name	Resource Plan	Enabled	Next Open Date	End Date	Duration (min)	Active	Description
<input checked="" type="radio"/>	WEEKNIGHT_WINDOW			Feb 7, 2018 10:00:00 PM		480	FALSE	Weeknight window - for compatibility only
<input checked="" type="radio"/>	WEEKEND_WINDOW			Feb 10, 2018 12:00:00 AM		2880	FALSE	Weekend window - for compatibility only
<input checked="" type="radio"/>	FRIDAY_WINDOW	DEFAULT_MAINTENANCE_PLAN	✓	Sep 21, 2018 10:00:00 PM		240	FALSE	Friday window for maintenance tasks
<input checked="" type="radio"/>	SATURDAY_WINDOW	DEFAULT_MAINTENANCE_PLAN	✓	Sep 22, 2018 6:00:00 AM		1200	FALSE	Saturday window for maintenance tasks
<input checked="" type="radio"/>	SUNDAY_WINDOW	DEFAULT_MAINTENANCE_PLAN	✓	Sep 23, 2018 6:00:00 AM		1200	FALSE	Sunday window for maintenance tasks
<input checked="" type="radio"/>	MONDAY_WINDOW	DEFAULT_MAINTENANCE_PLAN	✓	Sep 24, 2018 10:00:00 PM		240	FALSE	Monday window for maintenance tasks
<input checked="" type="radio"/>	TUESDAY_WINDOW	DEFAULT_MAINTENANCE_PLAN	✓	Sep 25, 2018 10:00:00 PM		240	FALSE	Tuesday window for maintenance tasks
<input checked="" type="radio"/>	WEDNESDAY_WINDOW	DEFAULT_MAINTENANCE_PLAN	✓	Sep 26, 2018 10:00:00 PM		240	FALSE	Wednesday window for maintenance tasks
<input checked="" type="radio"/>	THURSDAY_WINDOW	DEFAULT_MAINTENANCE_PLAN	✓	Sep 27, 2018 10:00:00 PM		240	FALSE	Thursday window for maintenance tasks

Automated Maintenance Tasks

Autotask maintenance process:

1. The maintenance window opens.
2. The Autotask background process schedules jobs.
3. Oracle Scheduler initiates jobs.
4. Oracle Resource Manager limits the impact of Autotask jobs.

Summary

In this lesson, you should have learned how to:

- Describe Oracle Database's proactive database maintenance infrastructure
- Discuss automated maintenance tasks
- Explain maintenance windows

28. Automated Maintenance Tasks: Managing Tasks and Windows

Objectives

After completing this lesson, you should be able to:

- Enable and disable maintenance tasks
- Create, modify, and remove maintenance windows
- Reduce or increase resource allocation to automated maintenance tasks

Configuring Automated Maintenance Tasks

You can perform the following configuration tasks:

- Adjust the duration and start time of the maintenance window.
- Control the resource plan that allocates resources to automated maintenance tasks during each window.
- Enable or disable individual tasks in some or all maintenance windows.



Enabling and Disabling Maintenance Tasks

- Enable or disable maintenance tasks for all maintenance windows:
 - Use the ENABLE and DISABLE procedures of the DBMS_AUTO_TASK_ADMIN package with the WINDOW_NAME argument set to NULL.
 - Use the ENABLE and DISABLE procedures with no arguments to enable or disable all automated maintenance tasks for all windows.
- Enable or disable maintenance tasks for specific maintenance windows:
 - Use the ENABLE and DISABLE procedures of the DBMS_AUTO_TASK_ADMIN package with the WINDOW_NAME argument set to a window name.



Creating and Managing Maintenance Windows

- To create a new maintenance window:
 1. Use the DBMS_SCHEDULER.CREATE_WINDOW procedure to create the window.
 2. Use the DBMS_SCHEDULER.ADD_GROUP_MEMBER procedure to add the new window to the MAINTENANCE_WINDOW_GROUP window group.
- To change the attributes of a maintenance window:
 1. Use the DBMS_SCHEDULER.DISABLE procedure to disable the window.
 2. Use the DBMS_SCHEDULER.SET_ATTRIBUTE procedure to modify the window attributes.
 3. Use the DBMS_SCHEDULER.ENABLE procedure to re-enable the window.
- To remove a maintenance window, use the DBMS_SCHEDULER.REMOVE_GROUP_MEMBER procedure.

Resource Allocations for Automated Maintenance Tasks

- Automated maintenance tasks run under the `ORA$AUTOTASK` subplan of the `DEFAULT_MAINTENANCE_PLAN` resource plan.
- Any resource allocation that is unused by sessions in `SYS_GROUP` is shared by sessions belonging to `OTHER_GROUPS` and `ORA$AUTOTASK` in the percentages shown below:

Consumer Group/subplan	Level 1	Maximum Utilization Limit
<code>ORA\$AUTOTASK</code>	5%	90
<code>OTHER_GROUPS</code>	20%	-
<code>SYS_GROUP</code>	75%	-

- `ORA$AUTOTASK` cannot be allocated more than 90% of the CPU resources.

Changing Resource Allocations for Maintenance Tasks

- Change the percentage of resources allocated to the `ORA$AUTOTASK` subplan in the resource plan for the window of interest.
- Adjust the resource allocation for one or more subplans or consumer groups in the window's resource plan so that the resource allocation at the top level of the plan adds up to 100%.

Summary

In this lesson, you should have learned how to:

- Enable and disable maintenance tasks
- Create, modify, and remove maintenance windows
- Reduce or increase resource allocation to automated maintenance tasks

Practice Overview

- Enabling and Disabling Automated Maintenance Tasks
- Modifying the Duration of a Maintenance Window

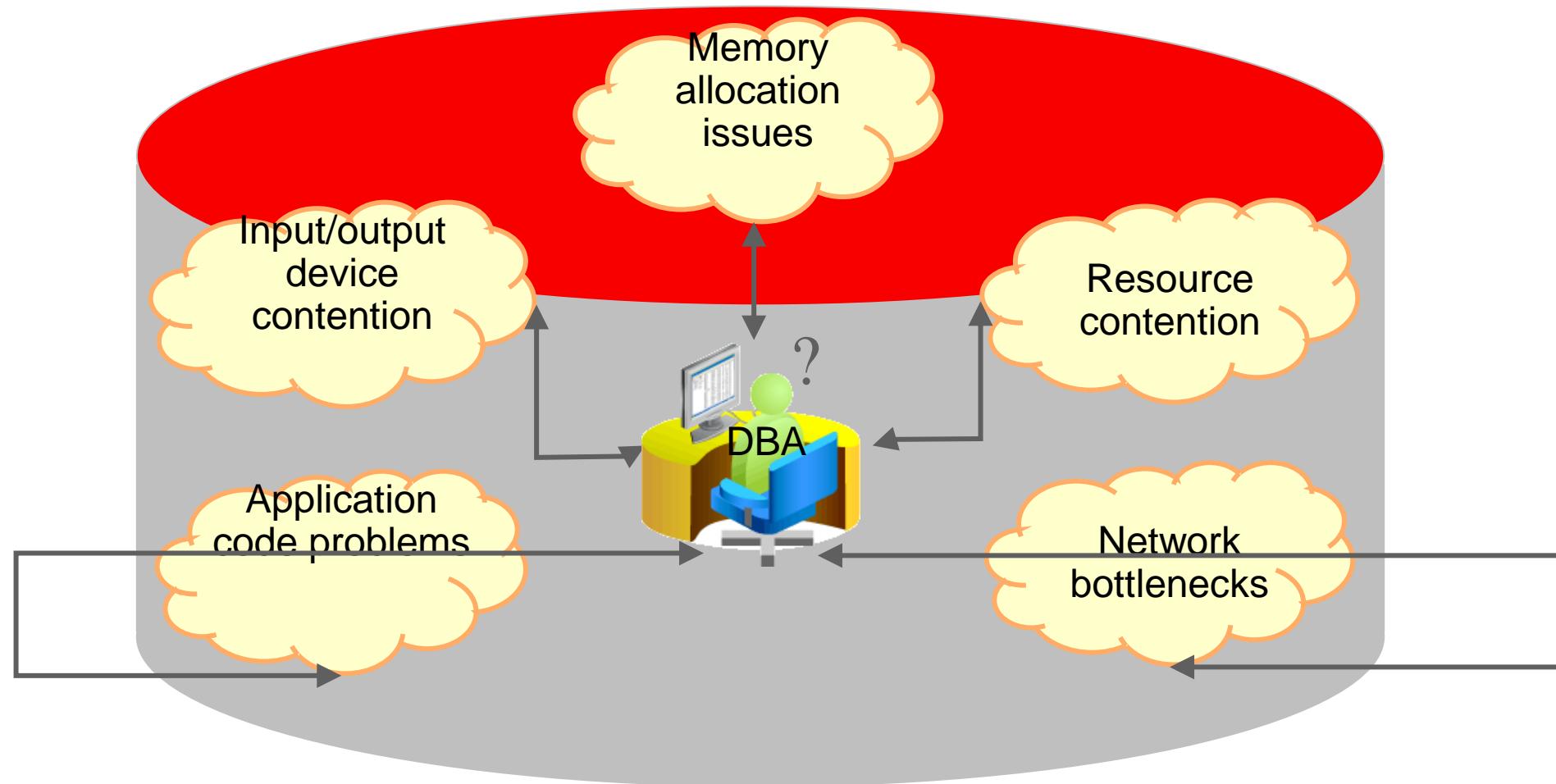
29. Database Monitoring and Tuning Performance Overview

Objectives

After completing this lesson, you should be able to describe:

- The activities that you perform to manage database performance
- The Oracle performance tuning methodology

Performance Management Activities



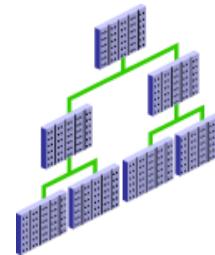
Performance Planning Considerations



System Architecture
Investment



Workload
Testing



Application Design
Principles

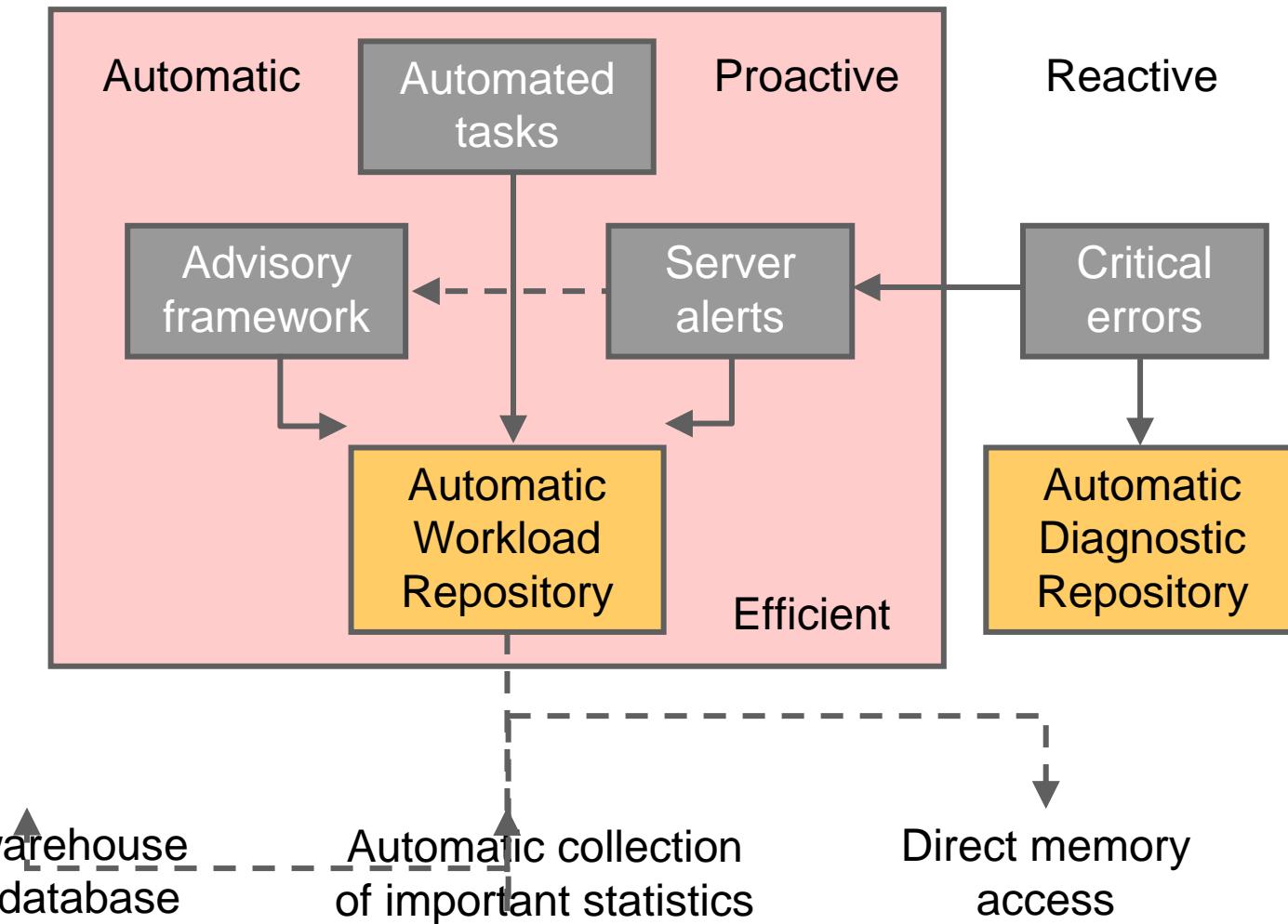


Scalability



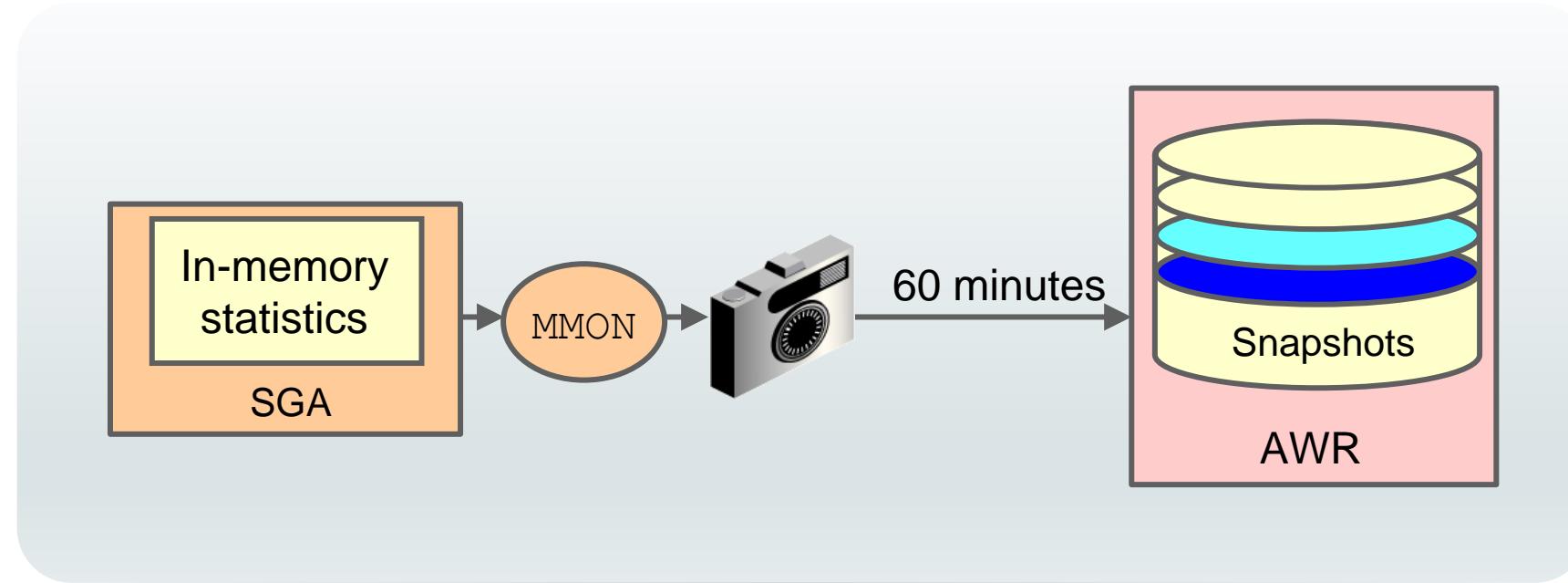
New Application
Deployment

Database Maintenance



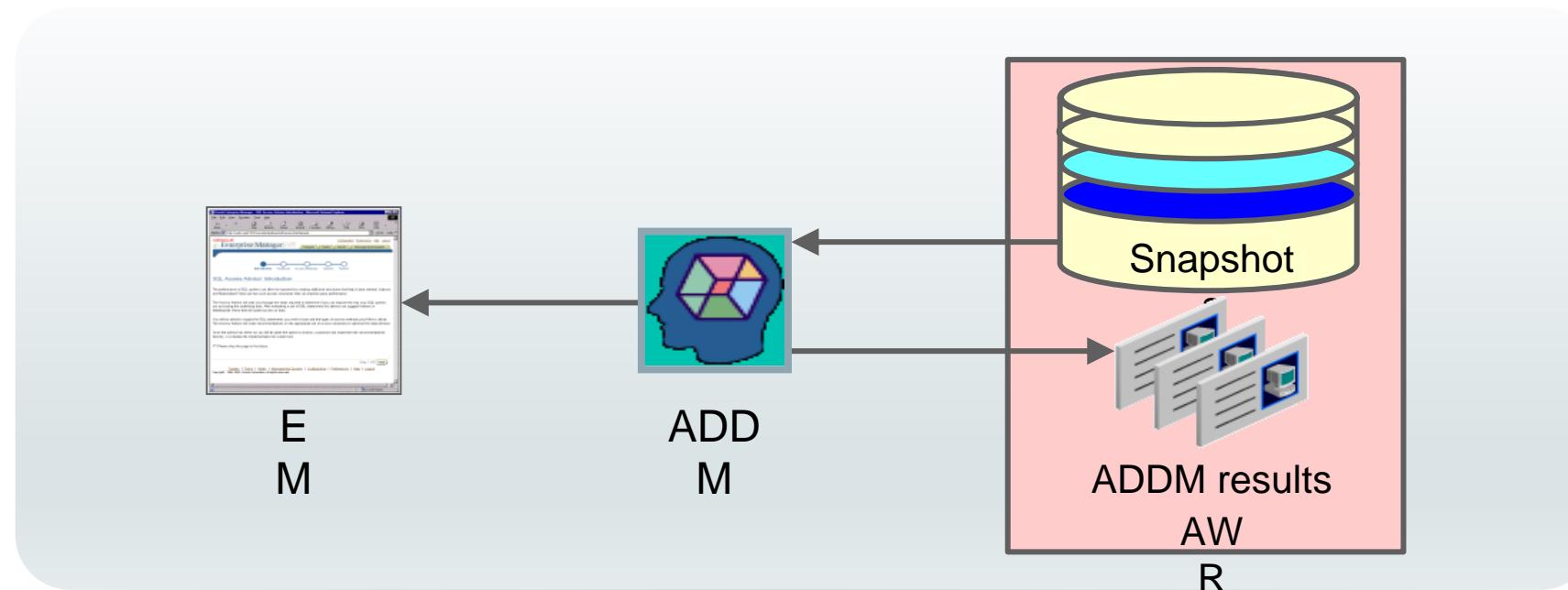
Automatic Workload Repository (AWR)

- Built-in repository of performance information
- Snapshots of database metrics taken every 60 minutes and retained for eight days
- Foundation for all self-management functions



Automatic Database Diagnostic Monitor (ADDM)

- Runs after each AWR snapshot
- Monitors the instance, detects bottlenecks
- Stores results in the AWR



Configuring Automatic ADDM Analysis at the PDB Level

1. Enable PDB AWR snapshot creation on the CDB root and on each PDB:

```
SQL> ALTER SYSTEM SET awr_pdb_autoflush_enabled = TRUE;
```

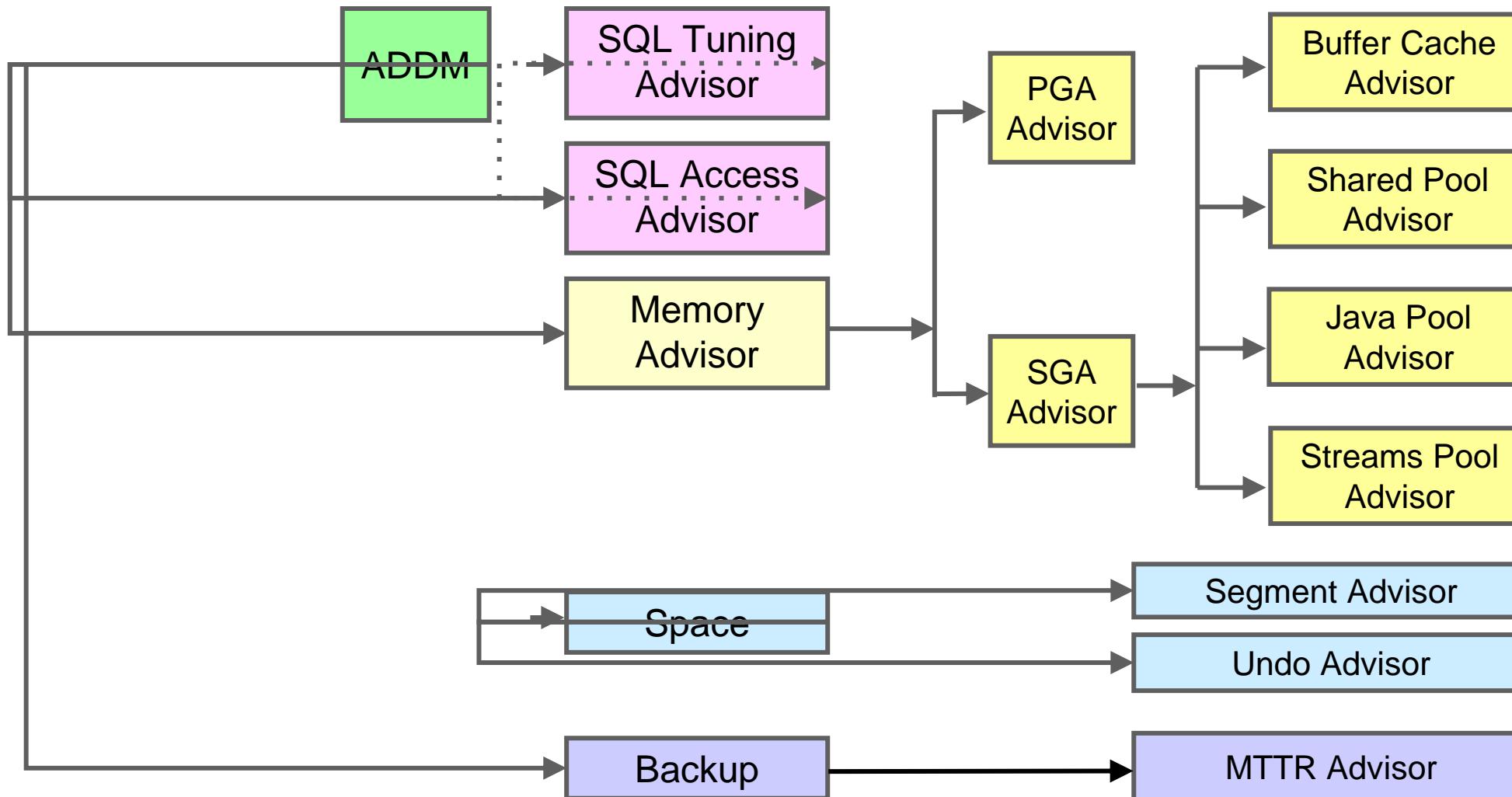
1. Set the AWR snapshot interval to greater than 0 at the PDB level:

```
SQL> CONNECT sys@PDB1 AS SYSDBA
SQL> EXEC dbms_workload_repository.modify_snapshot_settings(interval => 60)
```

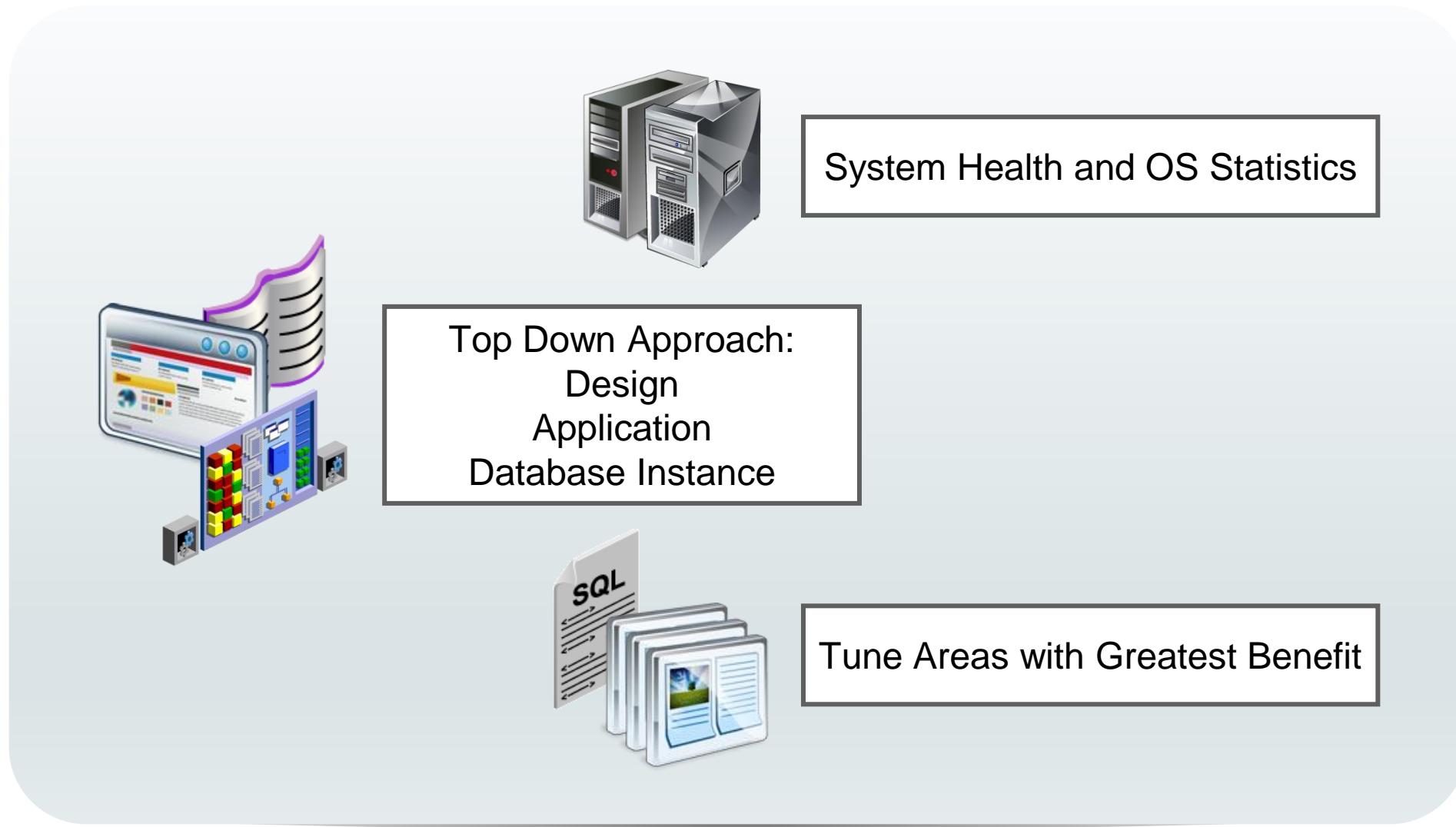
1. Execute the ADDM task (manually when required):

```
SQL> CONNECT sys@PDB1 AS SYSDBA
SQL> EXEC DBMS_ADDM.ANALYZE_DB(:tname, begin_snapshot =>1, end_snapshot =>2)
```

Advisory Framework



Performance Tuning Methodology



Summary

In this lesson, you should have learned how to describe:

- The activities that you perform to manage database performance
- The Oracle performance tuning methodology

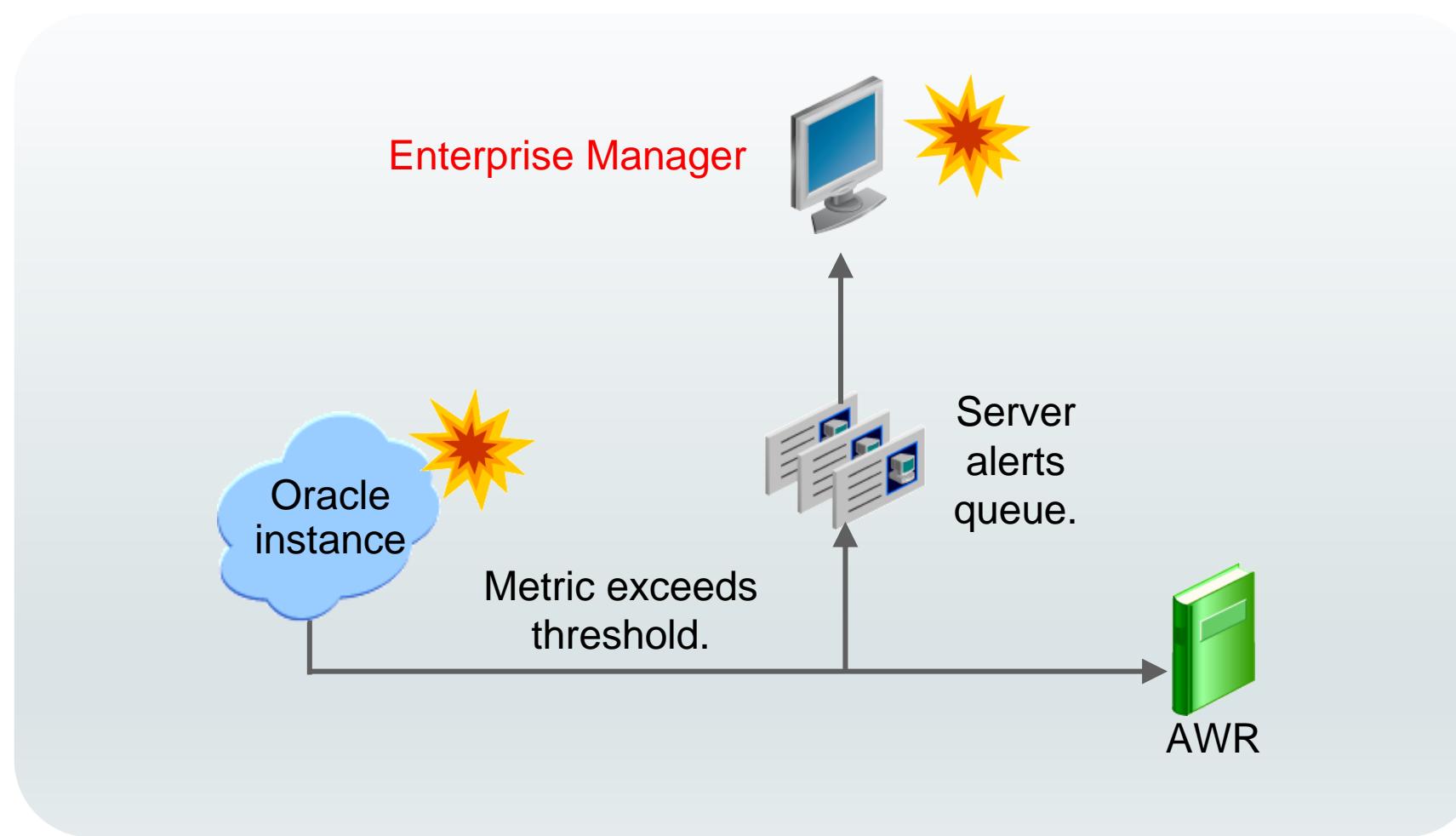
30. Monitoring Database Performance

Objectives

After completing this lesson, you should be able to:

- Use performance views and tools to monitor database instance performance
- Describe the server statistics and metrics that are collected by the Oracle Database server

Server-Generated Alerts



Setting Metric Thresholds

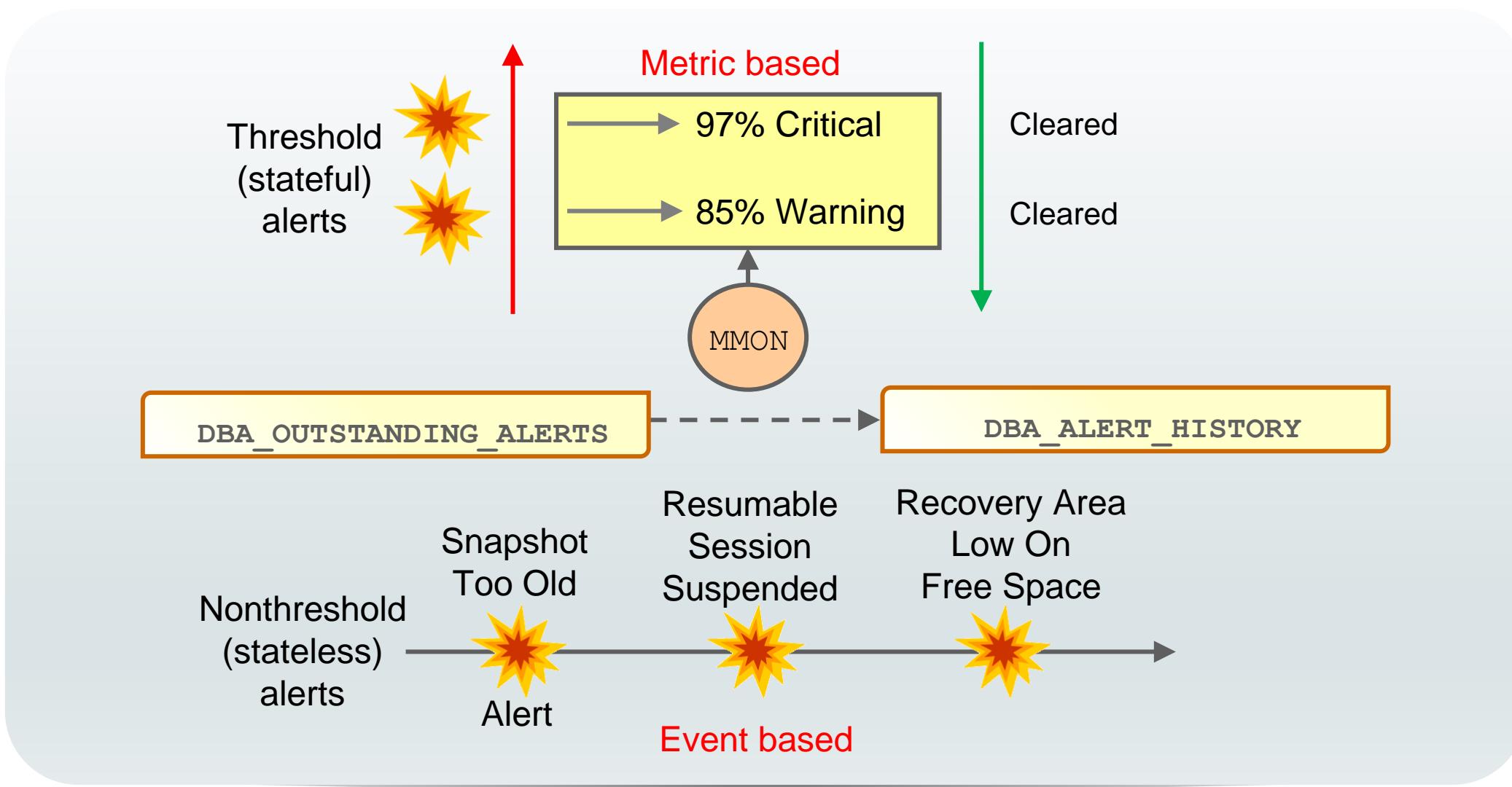
View and change threshold settings for the server alert metrics by using:

- The `GET_THRESHOLD` and `SET_THRESHOLD` procedures of the `DBMS_SERVER_ALERT` PL/SQL package
- The Metric and Collection Settings page in Enterprise Manager Cloud Control

Reacting to Alerts

- If necessary, you should gather more input (for example, by running ADDM or another advisor).
- Investigate critical errors.
- Take corrective measures.
- Acknowledge alerts that are not automatically cleared.

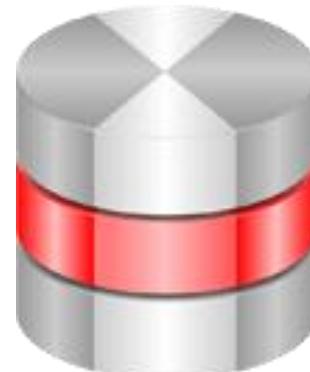
Alert Types and Clearing Alerts



Database Server Statistics and Metrics

Cumulative statistics:

- Wait events with time information
- Time model



Metrics: Statistic rates

Sampled statistics:

- Active session history
- Statistics by session, SQL, and service
- Other dimensions



Performance Monitoring

- Enterprise Manager Database Express
- Enterprise Manager Cloud Control
- Performance views

Instance/Database

V\$DATABASE
V\$INSTANCE
V\$PARAMETER
V\$SPPARAMETER
V\$SYSTEM_PARAMETER
V\$PROCESS
V\$BGPROCESS
V\$PX_PROCESS_SYSSTAT
V\$SYSTEM_EVENT

Disk

V\$DATAFILE
V\$FILESTAT
V\$LOG
V\$LOG_HISTORY
V\$DBFILE
V\$TEMPFILE
V\$TEMPSEG_USAGE
V\$SEGMENT_STATISTICS

Memory

V\$BUFFER_POOL_STATISTIC
S
V\$LIBRARYCACHE
V\$SGAINFO
V\$PGASTAT

Contention

V\$LOCK
V\$UNDOSTAT
V\$WAITSTAT
V\$LATCH

Viewing Statistics Information

V\$SYSSTAT

- STATISTIC#
- NAME
- CLASS
- VALUE
- STAT_ID

V\$SYSTEM_WAIT_CLAS S

- WAIT_CLASS_ID
- WAIT_CLASS#
- WAIT_CLASS
- TOTAL_WAITS
- TIME_WAITED

V\$SGASTA T

- POOL
- NAME
- BYTES

V\$EVENT_NAME

- EVENT_NUMBER
- EVENT_ID
- NAME
- PARAMETER1
- PARAMETER2
- PARAMETER3
- WAIT_CLASS

V\$SYSTEM_EVENT

- EVENT
- TOTAL_WAITS
- TOTAL_TIMEOUTS
- TIME_WAITED
- AVERAGE_WAIT
- TIME_WAITED_MICRO



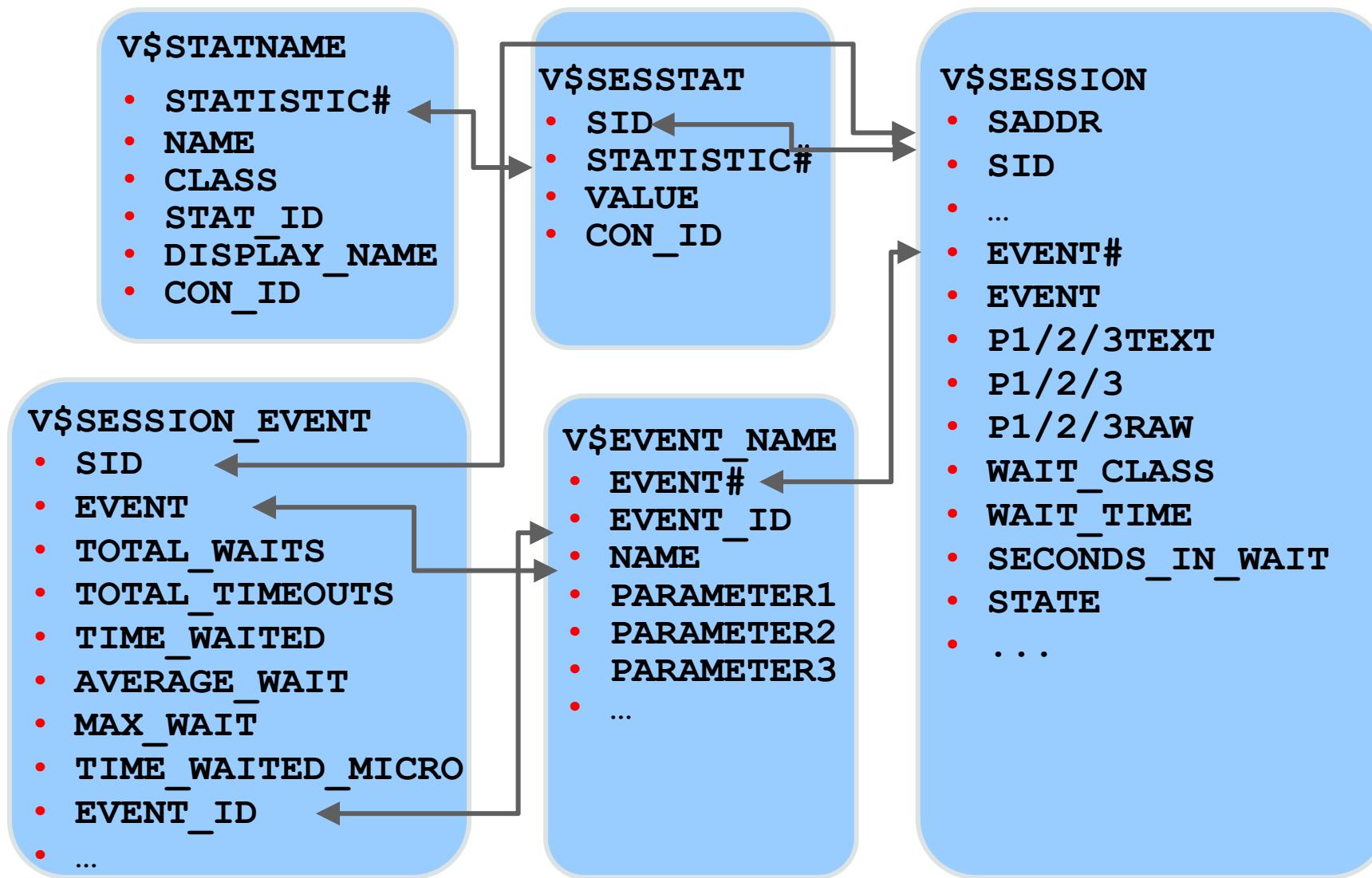
Monitoring Wait Events



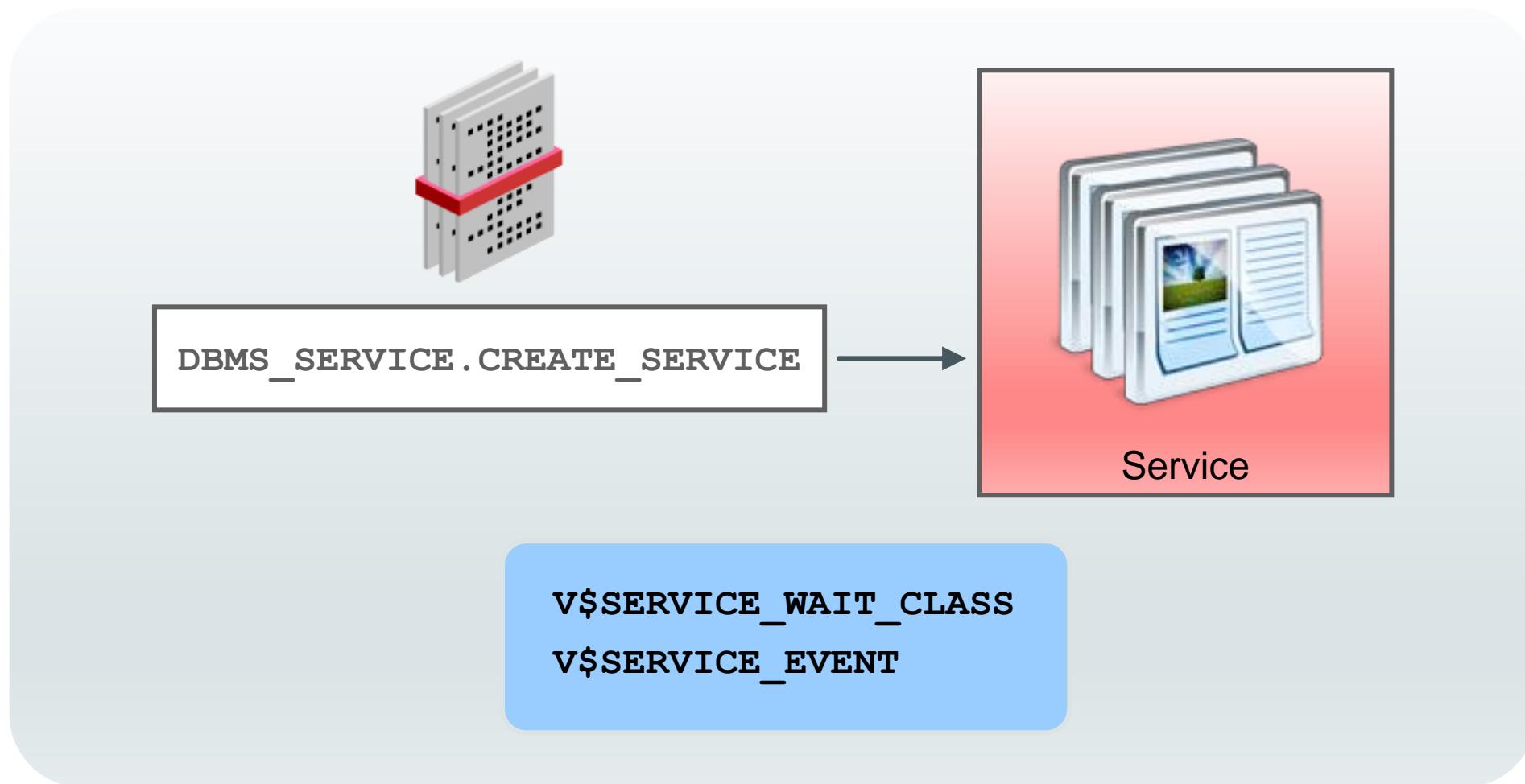
Wait events: Statistics indicating the server process had to wait for an event to complete

V\$EVENT_NAME

Monitoring Sessions



Monitoring Services



Summary

In this lesson, you should have learned how to:

- Use performance views and tools to monitor database instance performance
- Describe statistics and metrics that are collected by the Oracle Database server

Practice Overview

- Using Enterprise Manager Database Express to Manage Performance

31. Database Processes

Objectives

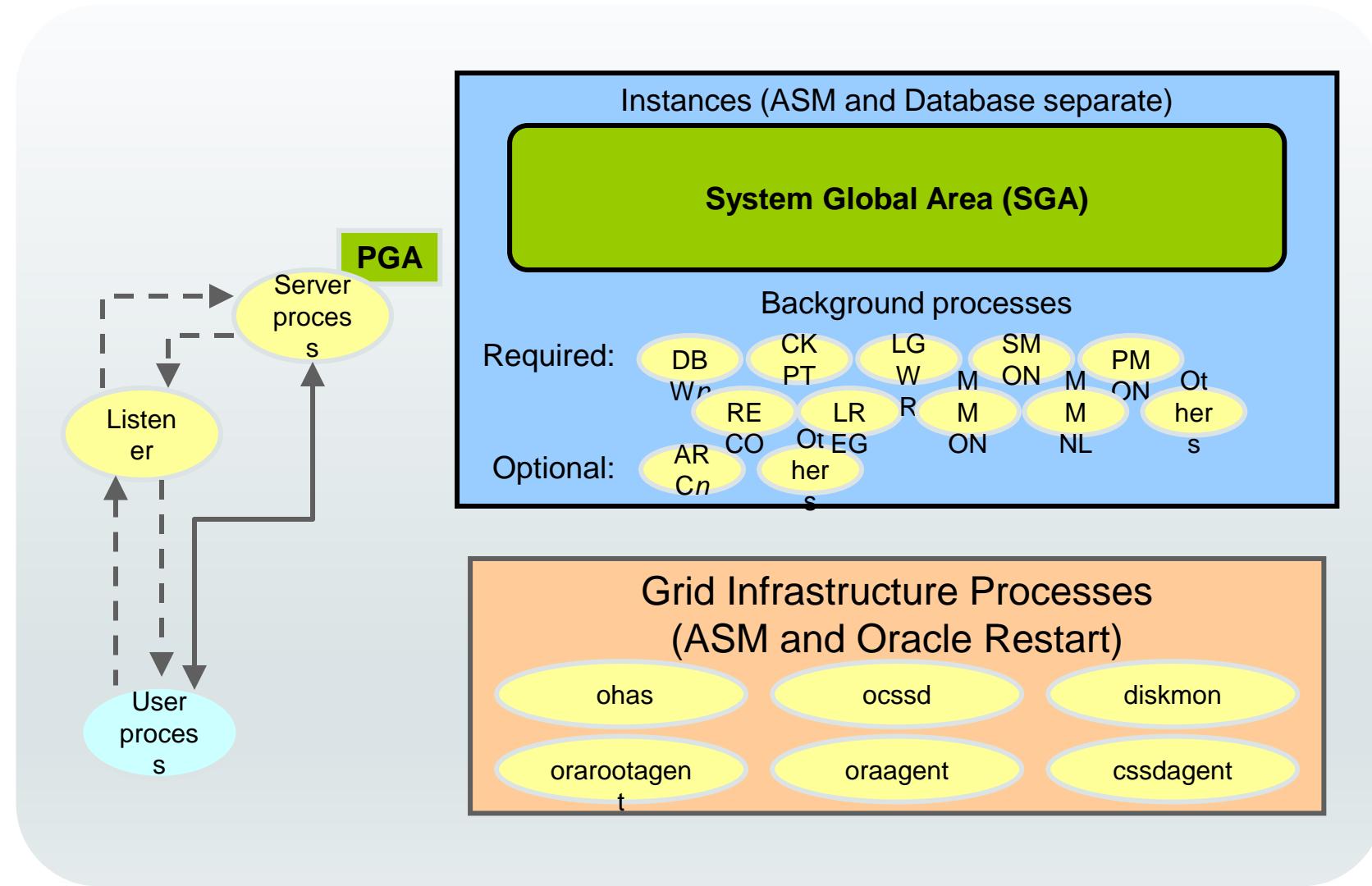
After completing this lesson, you should be able to

- Describe background processes

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

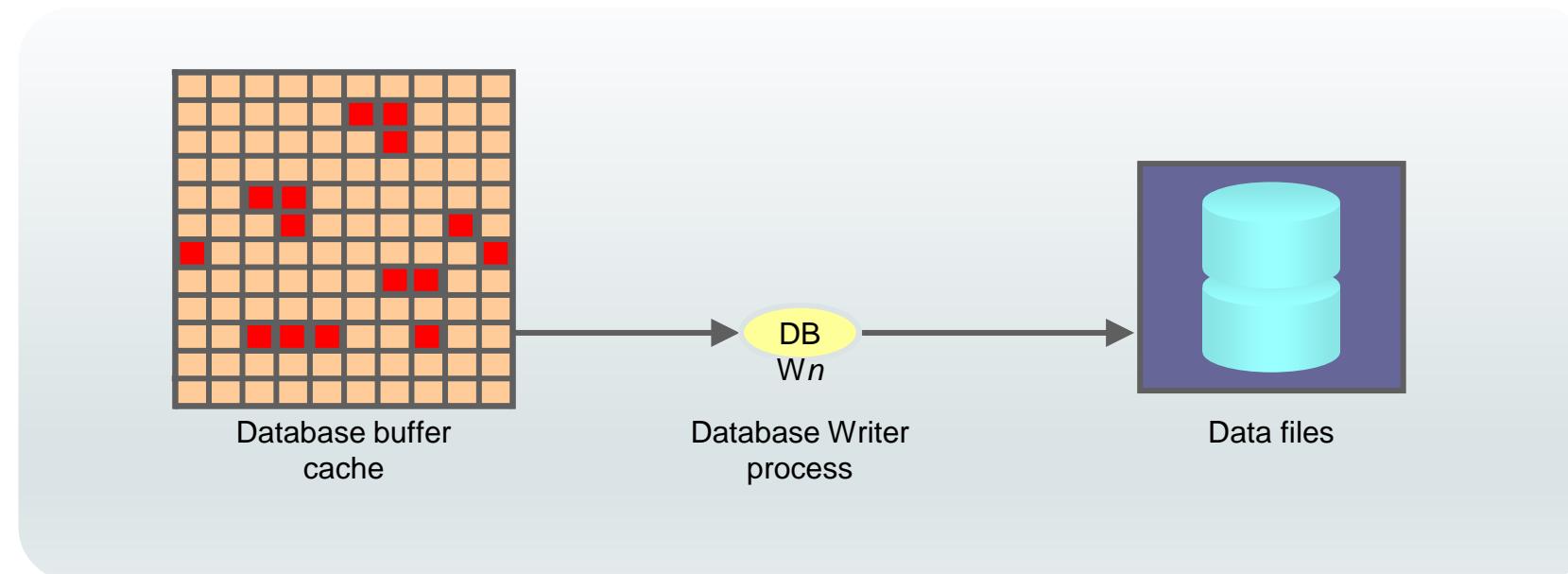
Process Structures



Database Writer Process (DBW n & BW nn)

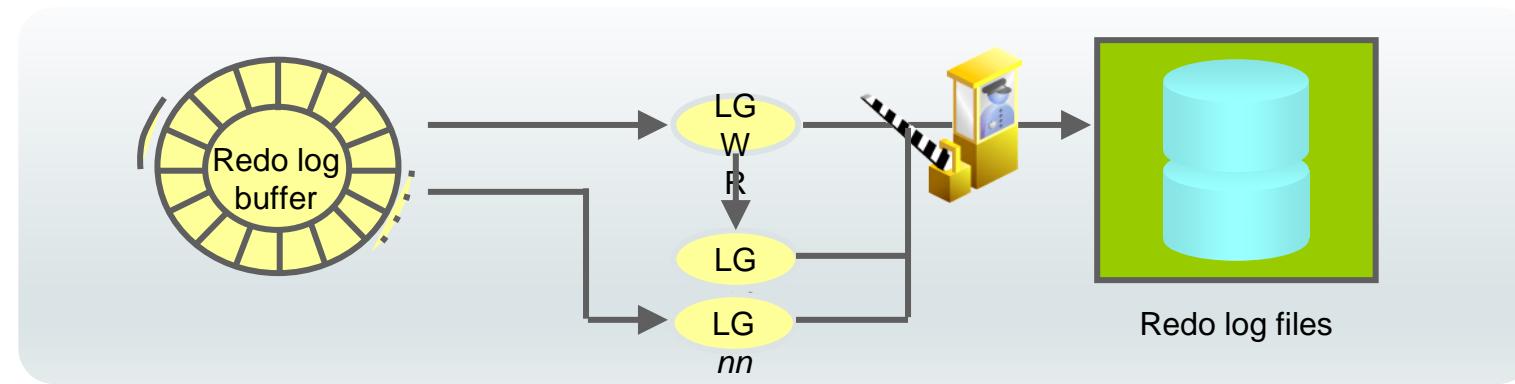
Writes modified (dirty) buffers in the database buffer cache to disk:

- Asynchronously while performing other processing
- To advance the checkpoint



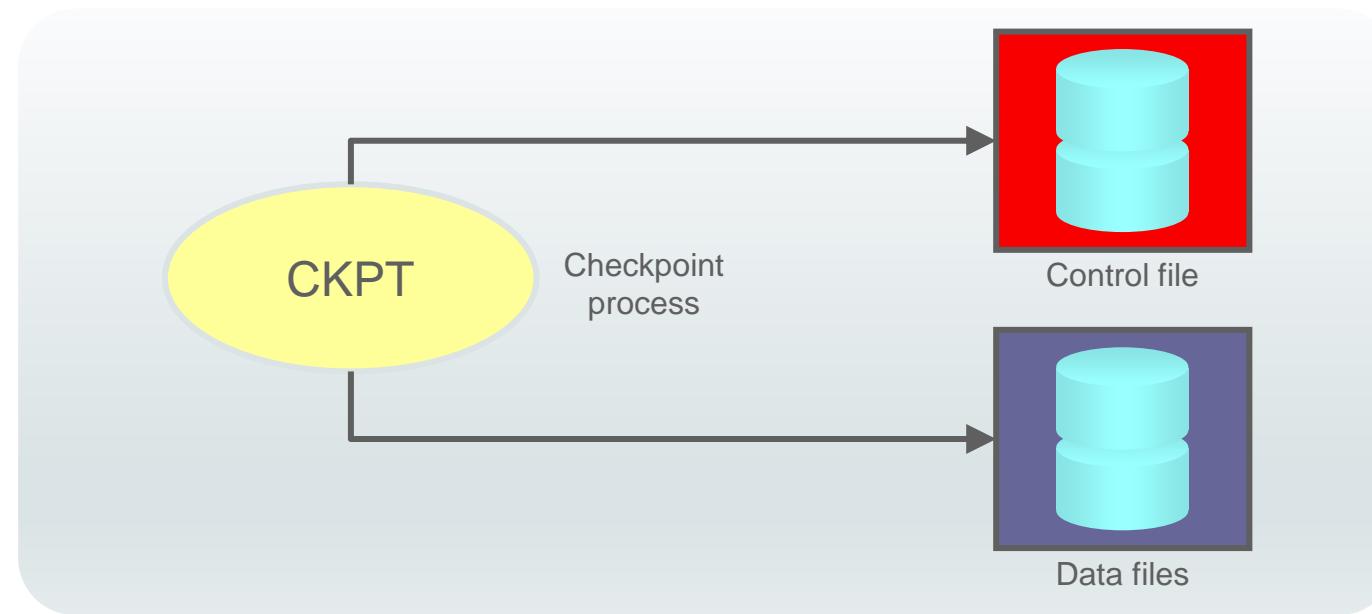
Log Writer Process (LGWR & LGnn)

- Writes the redo log buffer to a redo log file on disk:
 - When a user process commits a transaction
 - When an online redo log switch occurs
 - When the redo log buffer is one-third full or contains 1 MB of buffered data
 - Before a DBW n process writes modified buffers to disk
 - When three seconds have passed since the last write
- Serves as coordinator of LGnn processes and ensures correct order for operations that must be ordered



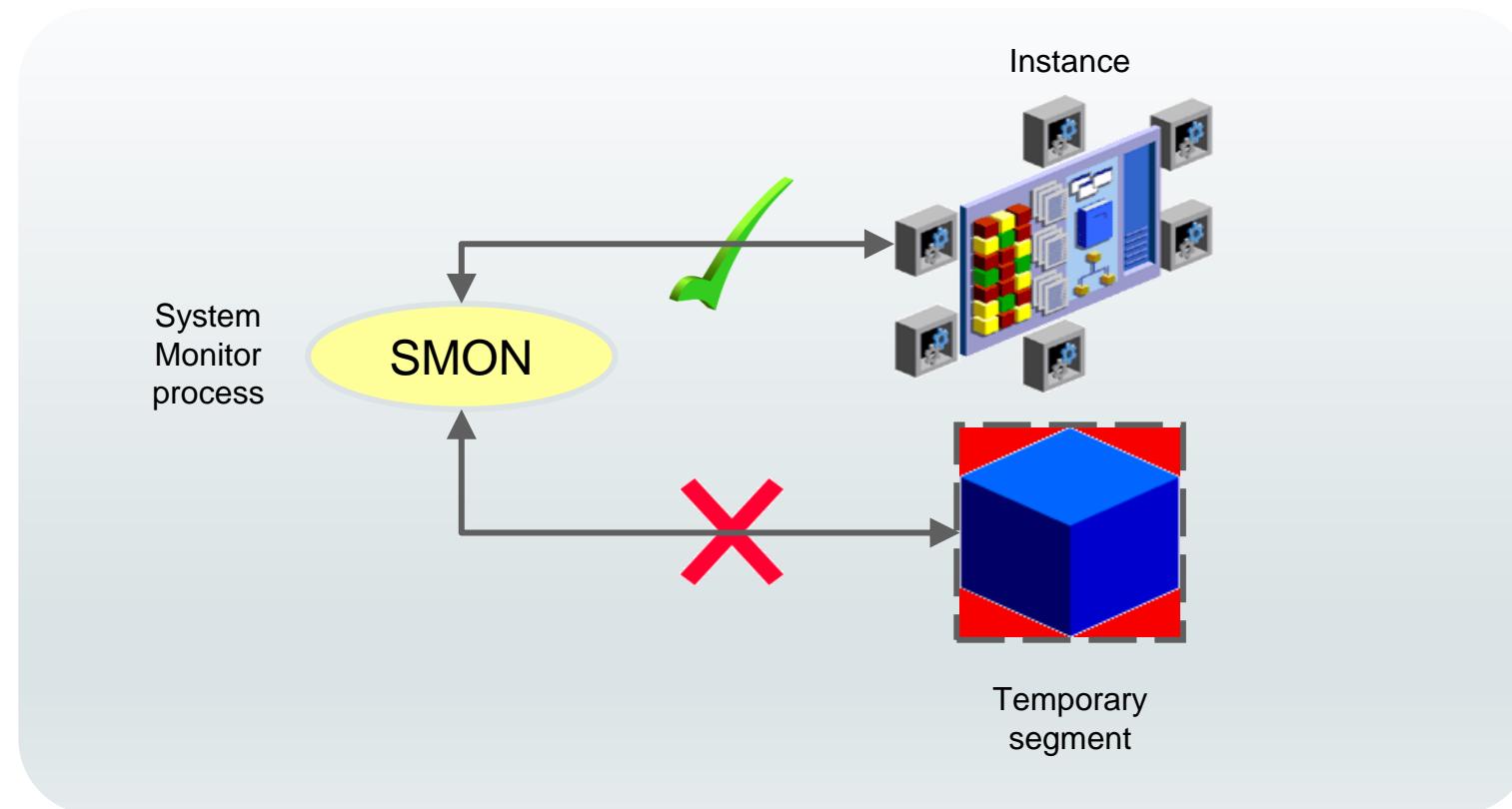
Checkpoint Process (CKPT)

- Records checkpoint information in:
 - The Control file
 - Each data file header
- Signals DBW n to write blocks to disk



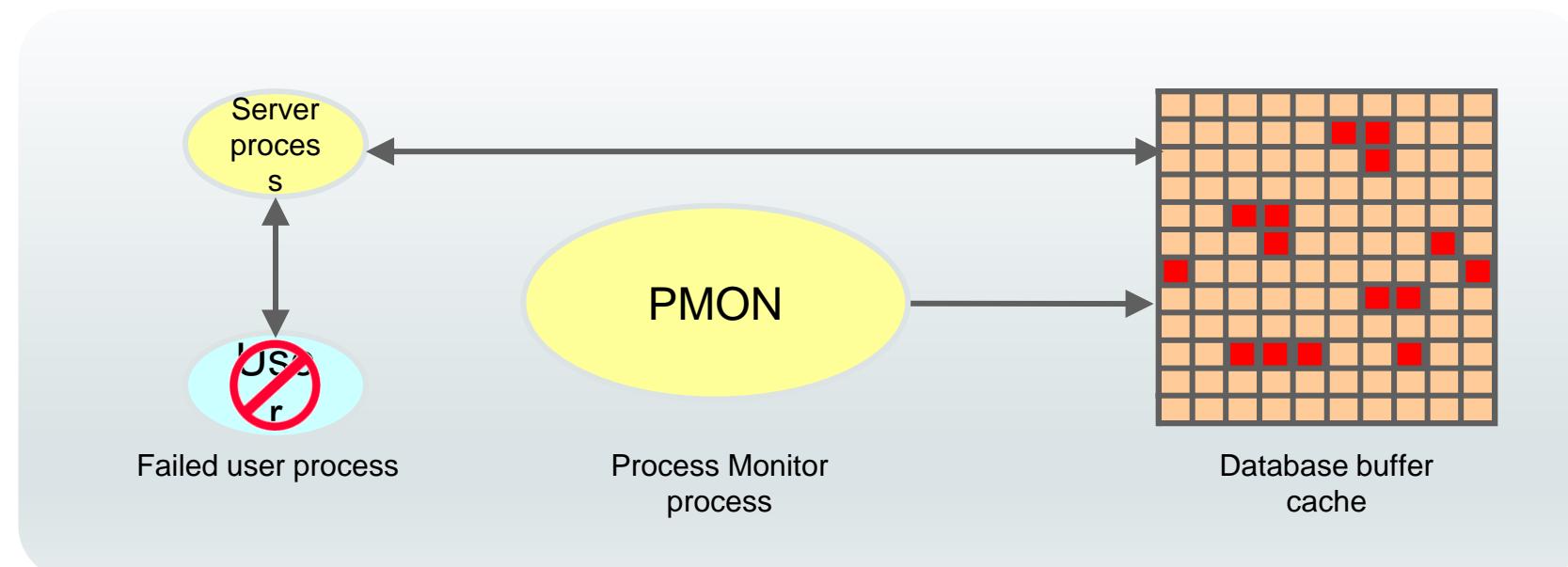
System Monitor Process (SMON)

- Performs recovery at instance startup
- Cleans up unused temporary segments



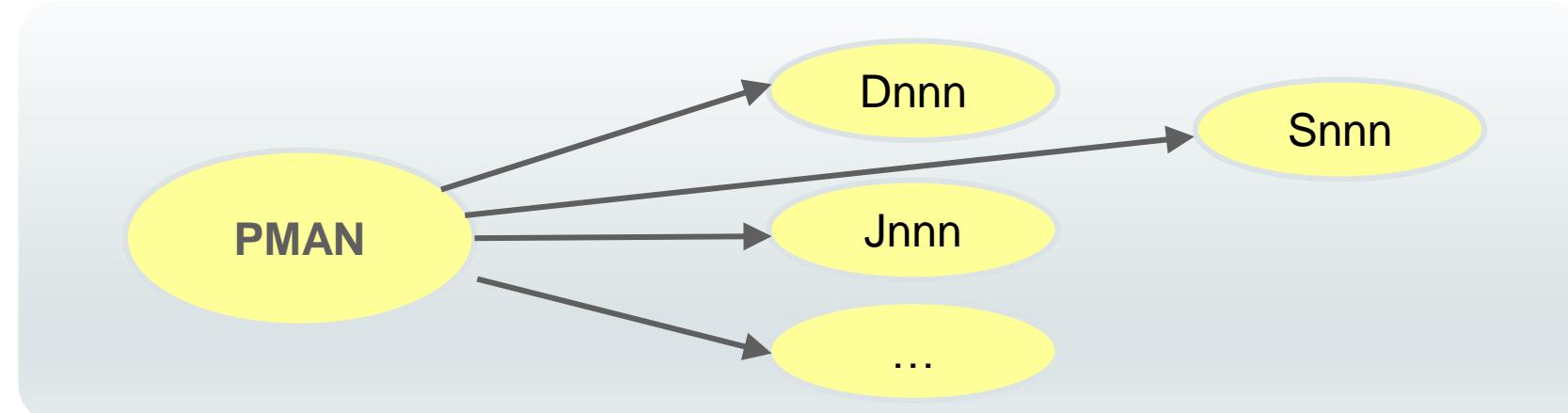
Process Monitor Process (PMON)

- Performs process recovery when a user process fails
 - Cleans up the database buffer cache
 - Freed resources that are used by the user process
- Monitors sessions for idle session timeout



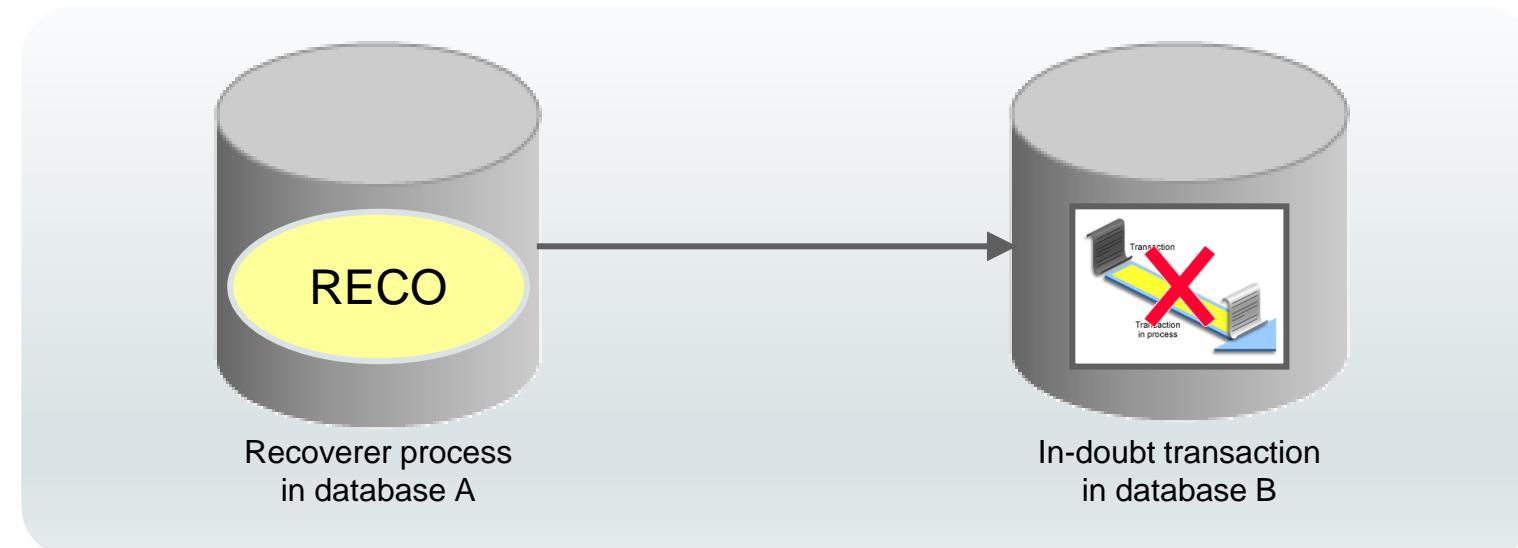
Process Manager (PMAN)

- PMAN monitors, spawns, and stops the following types of processes as needed:
 - Dispatcher and shared server processes
 - Connection broker and pooled server processes for database resident connection pools
 - Job queue processes
 - Restartable background processes



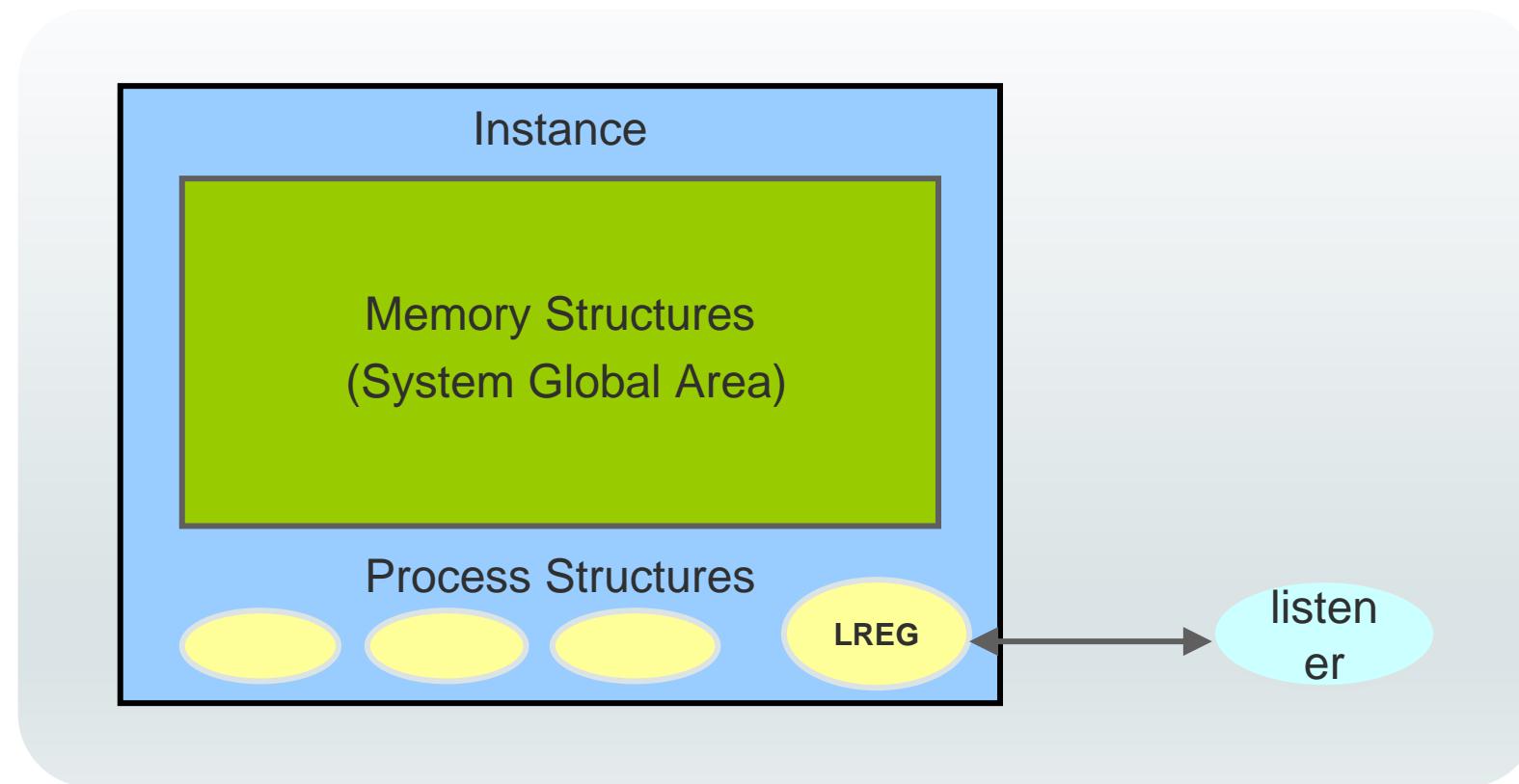
Recoverer Process (RECO)

- Used with the distributed database configuration
- Automatically connects to other databases involved in in-doubt distributed transactions
- Automatically resolves all in-doubt transactions
- Removes any rows that correspond to in-doubt transactions



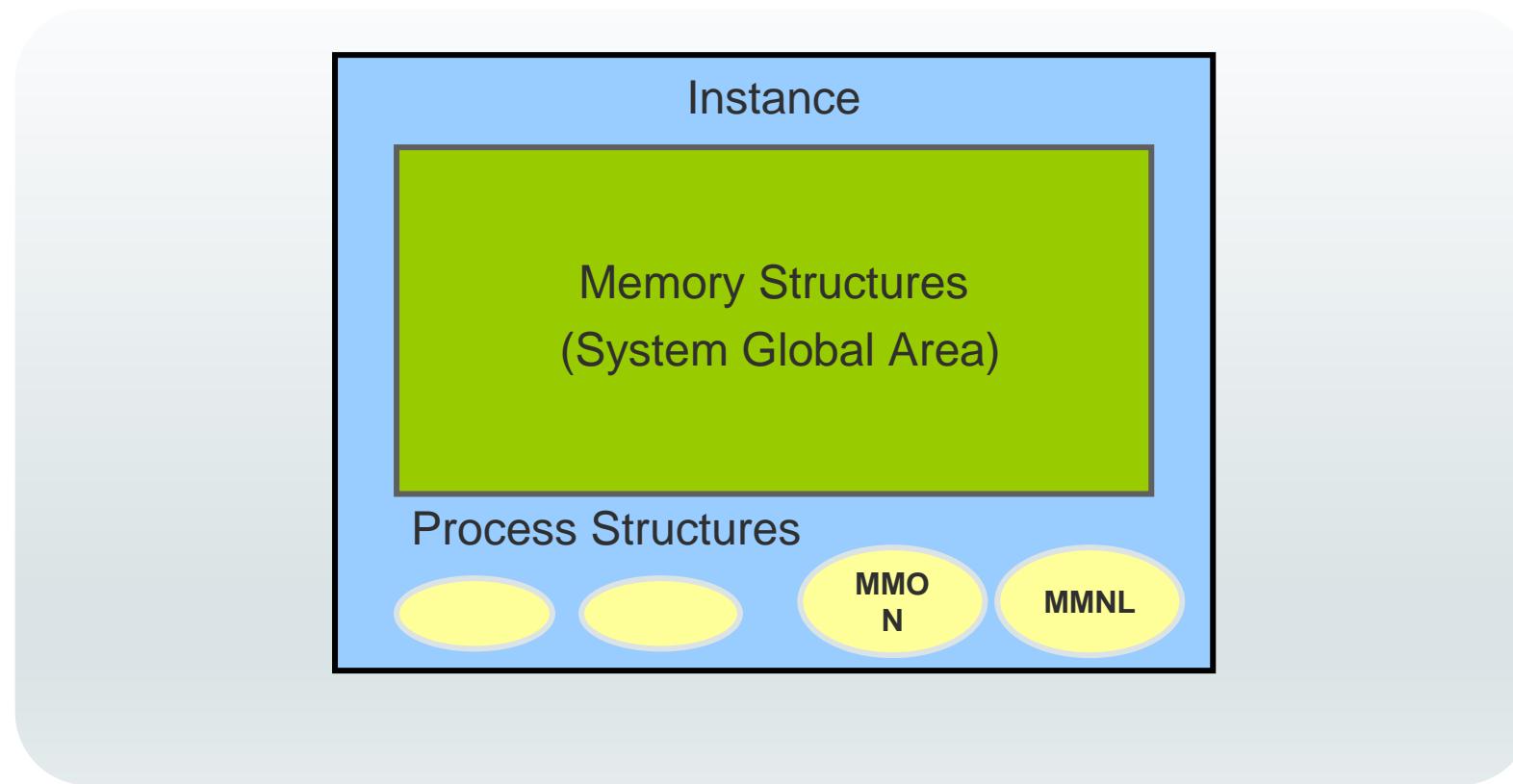
Listener Registration Process (LREG)

Registers information about the database instance and dispatcher processes with Oracle Net Listener



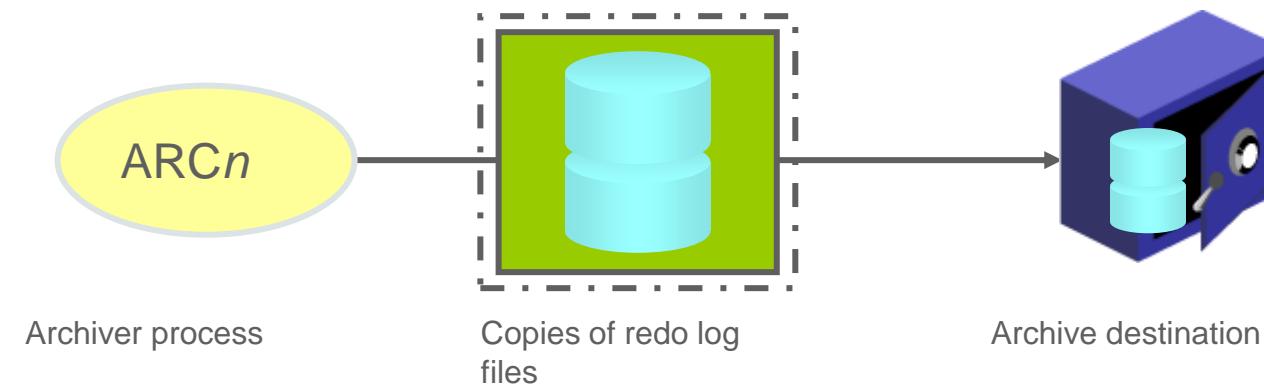
Manageability Monitor Process (MMON)

Performs or schedules many manageability task

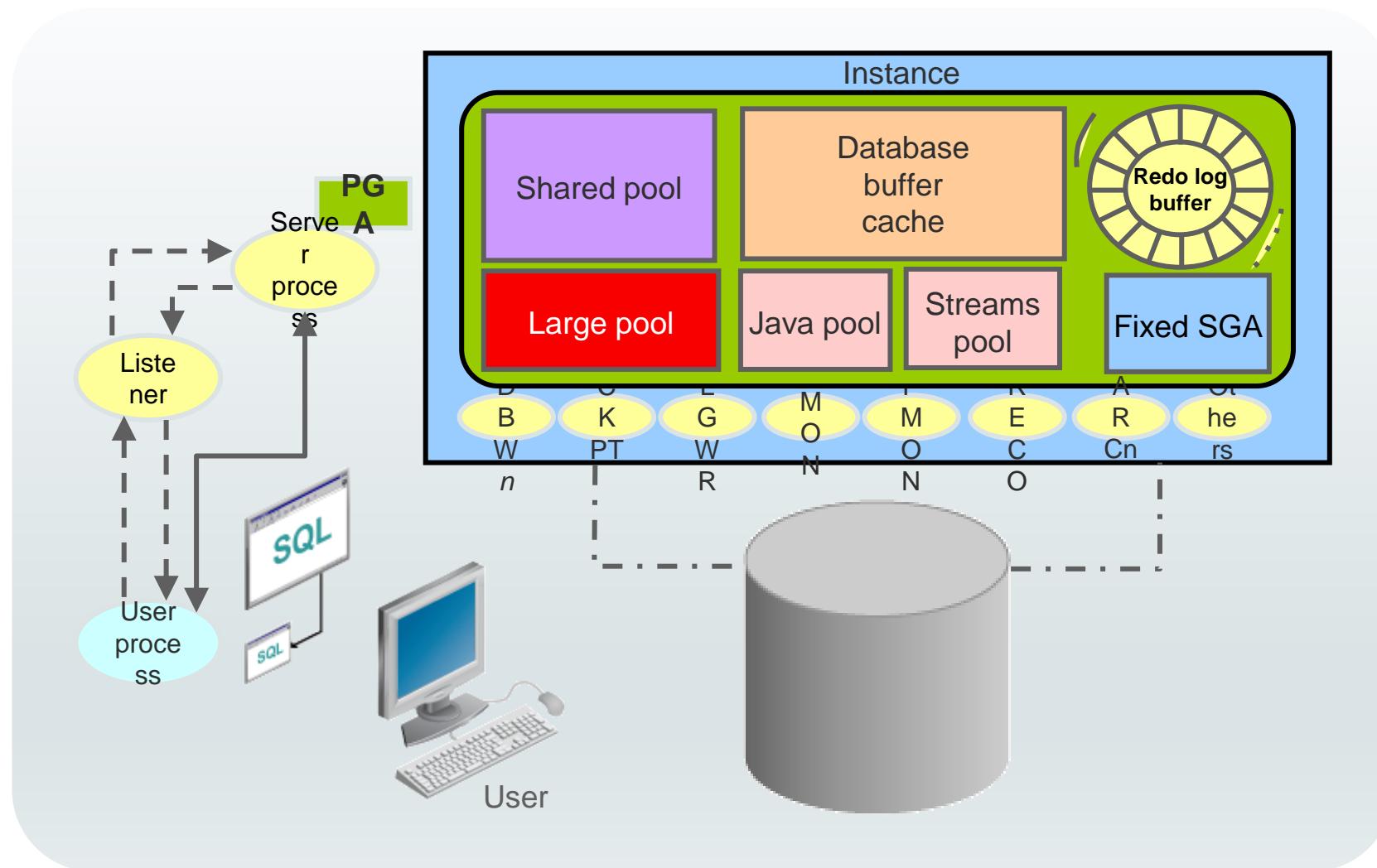


Archiver Processes (ARCn)

- Copy redo log files to a designated storage device after a log switch has occurred
- Can collect transaction redo data and transmit that data to standby destinations



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



Summary

In this lesson, you should have learned how to configure and monitor memory components for optimal performance.

Practice Overview

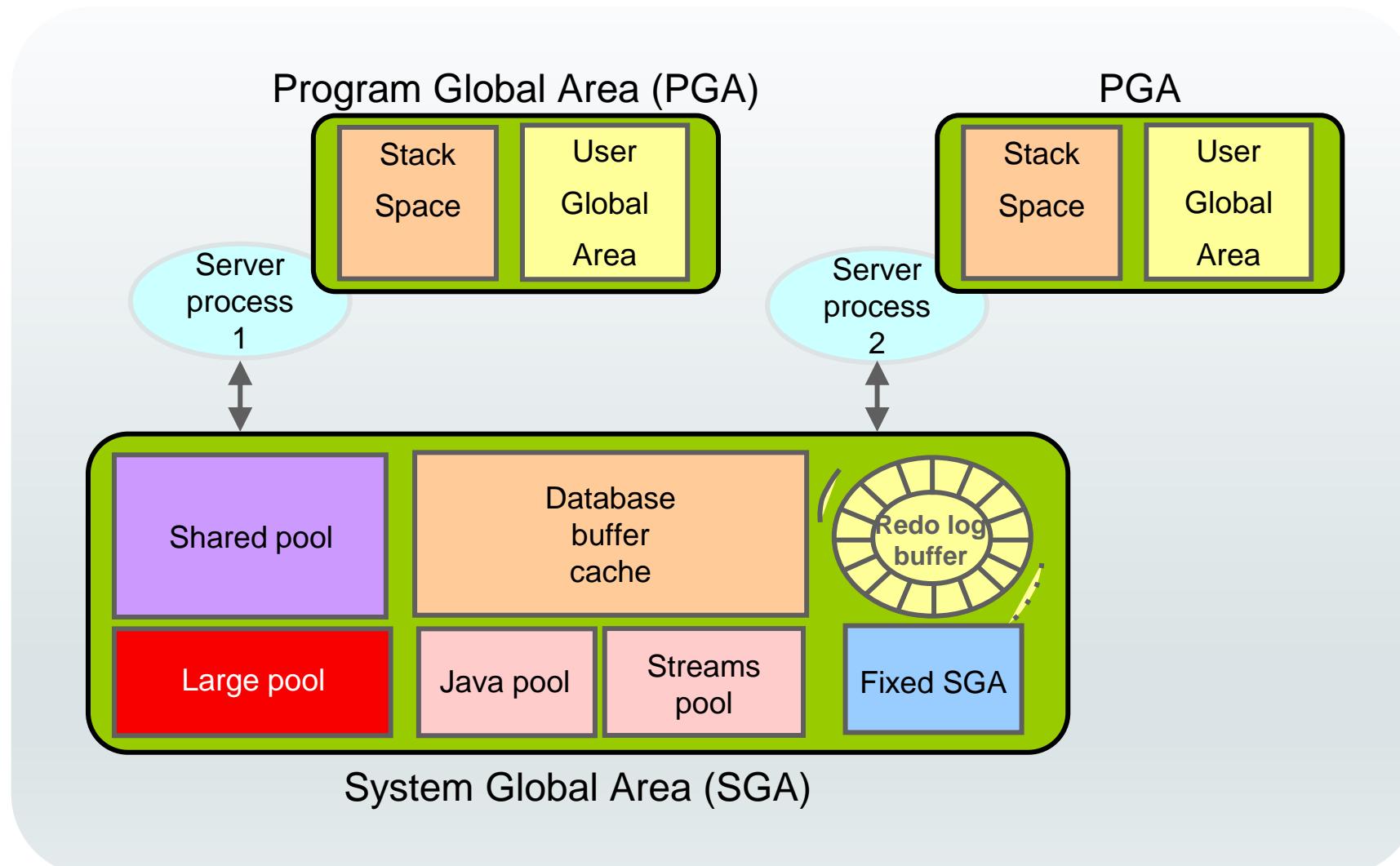
- Examining the Database Background Processes
- Identifying the Database Foreground Server Processes

32. Managing Memory

Objectives

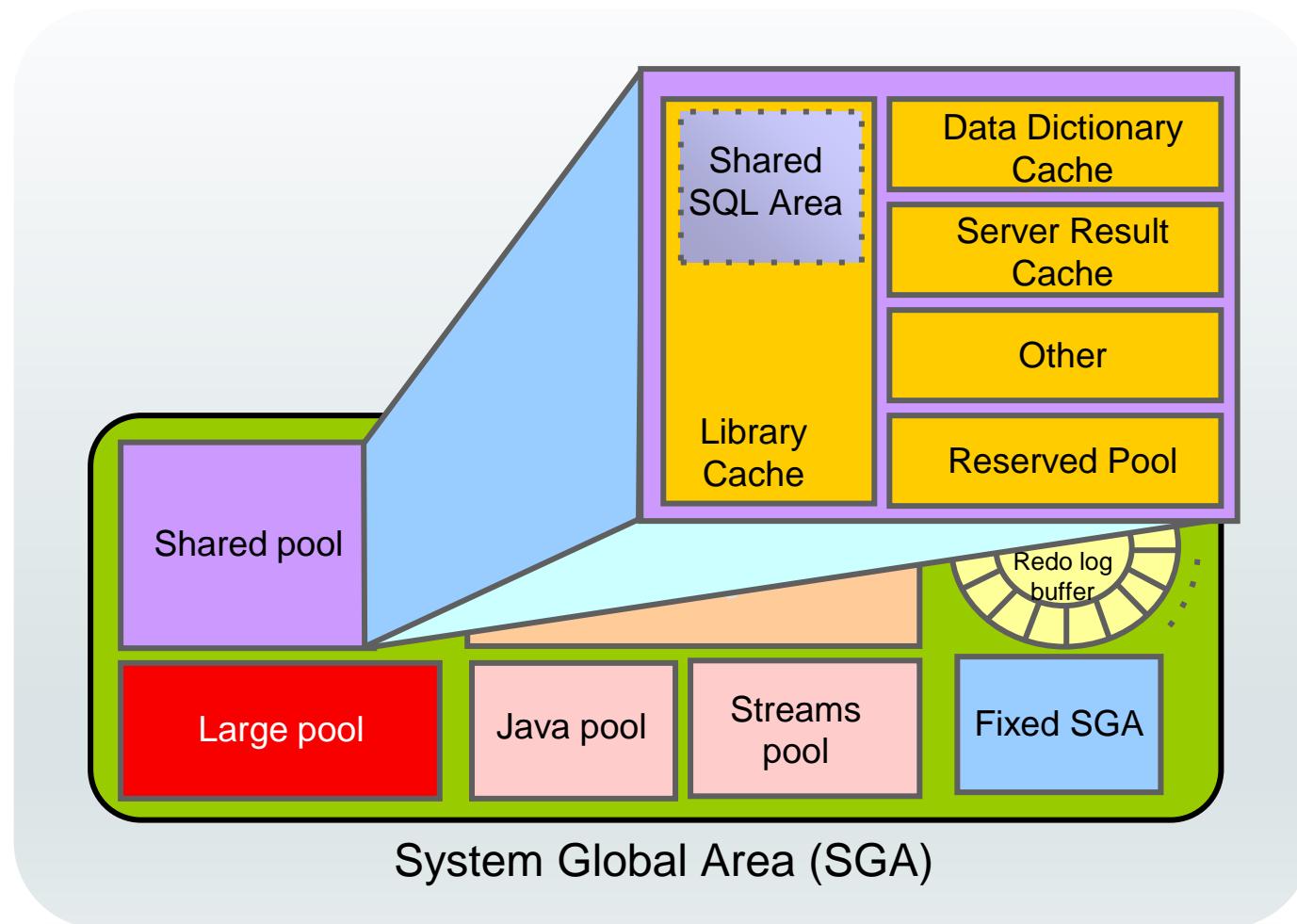
After completing this lesson, you should be able to configure and monitor memory components for optimal performance.

Managing Memory Components



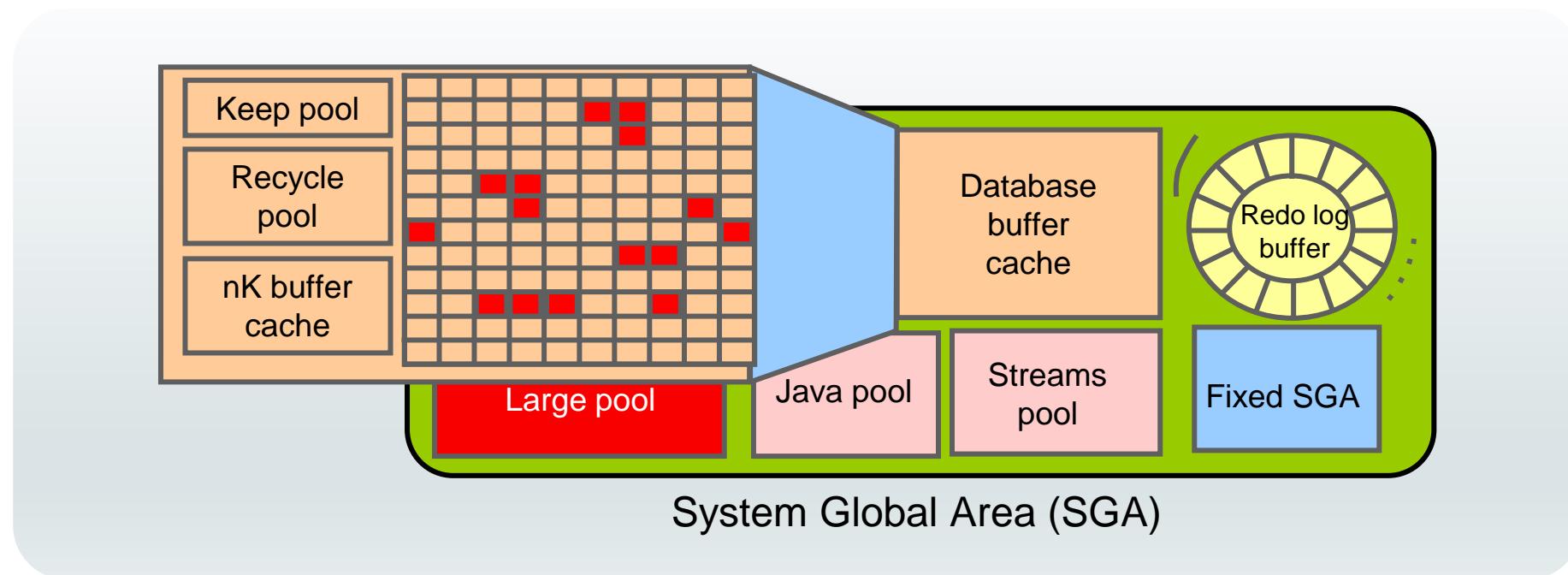
Shared Pool

- Is a portion of the SGA
- Contains:
 - Library cache
 - Shared SQL area
 - Data dictionary cache
 - Server result cache



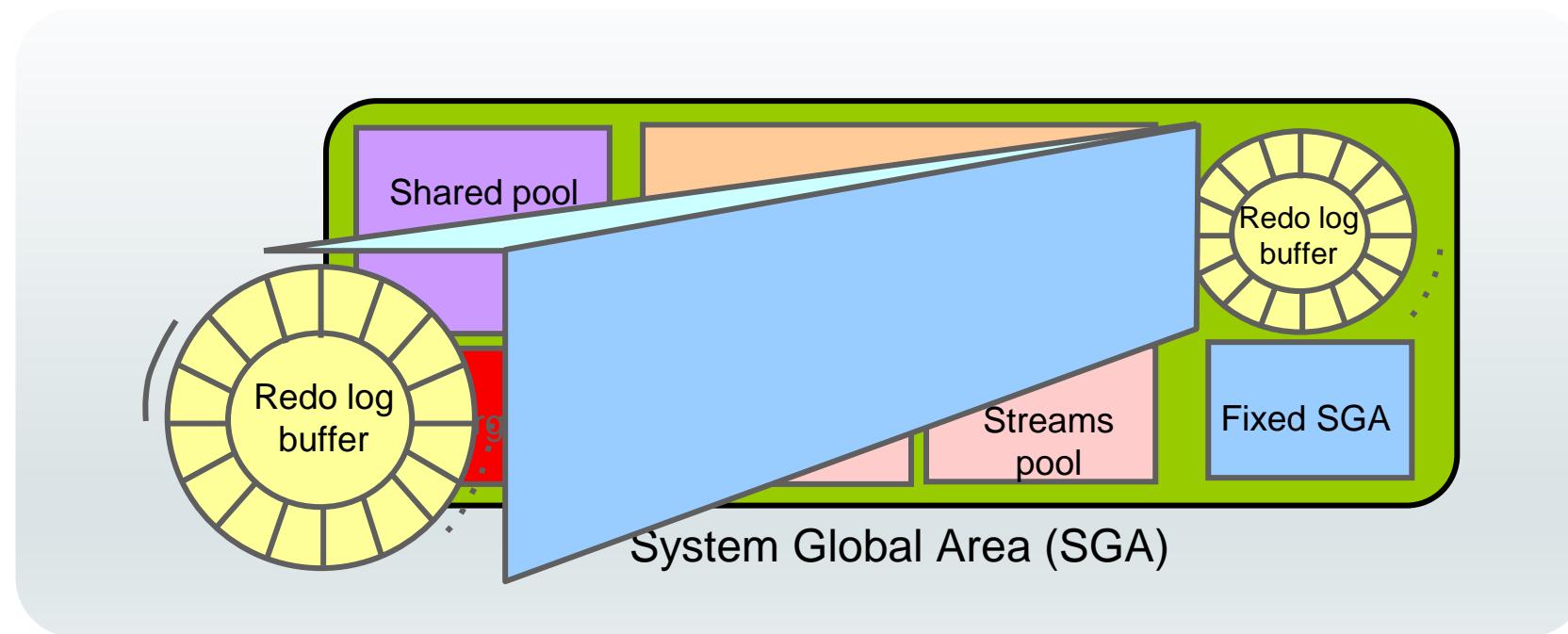
Database Buffer Cache

- Is part of the SGA
- Holds copies of data blocks that are read from data files
- Is shared by all concurrent users



Redo Log Buffer

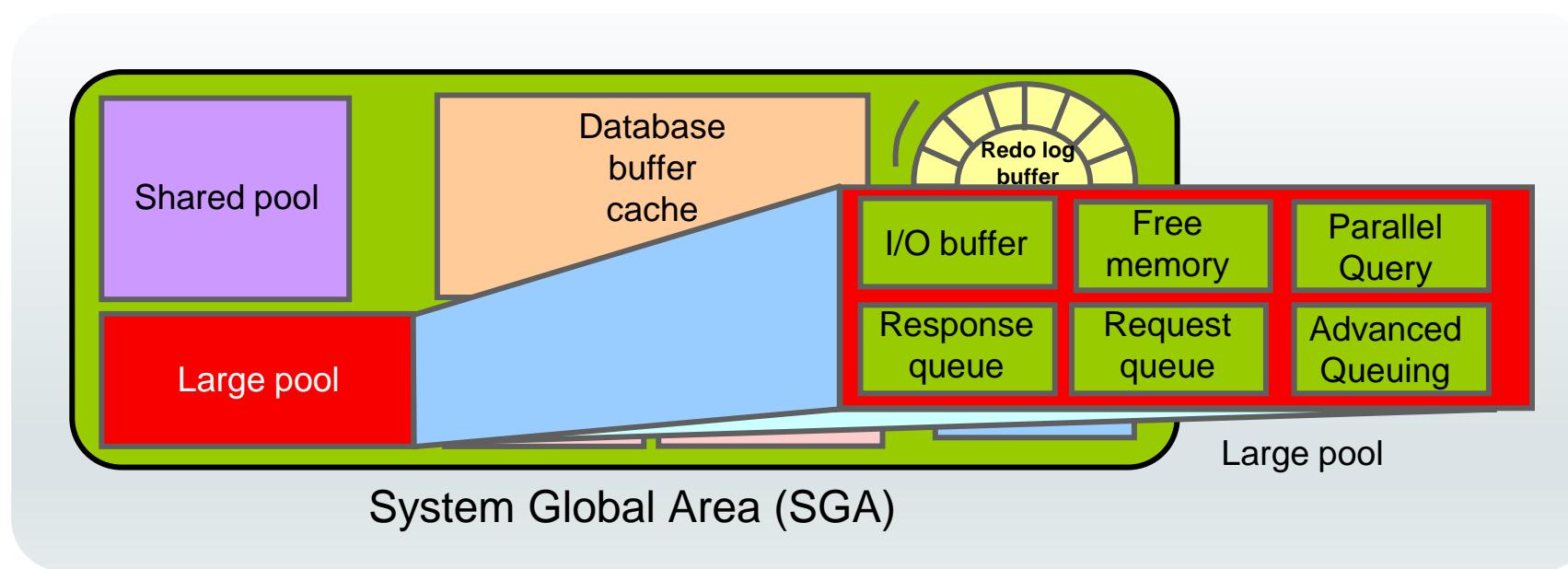
- Is a circular buffer in the SGA
- Holds information about changes made to the database
- Contains redo entries that have the information to redo changes made by operations such as DML and DDL



Large Pool

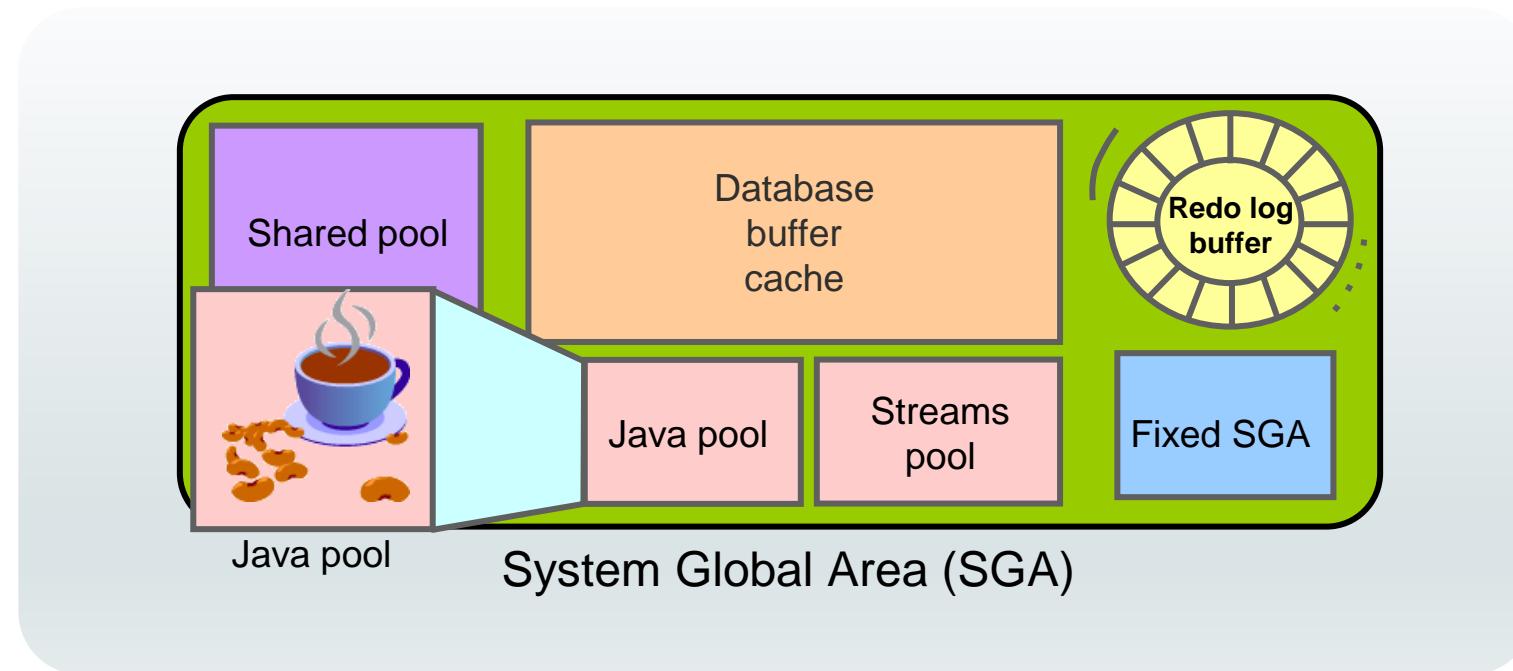
Provides large memory allocations for:

- Session memory for the shared server and the Oracle XA interface
- I/O server processes
- Oracle Database backup and restore operations



Java Pool

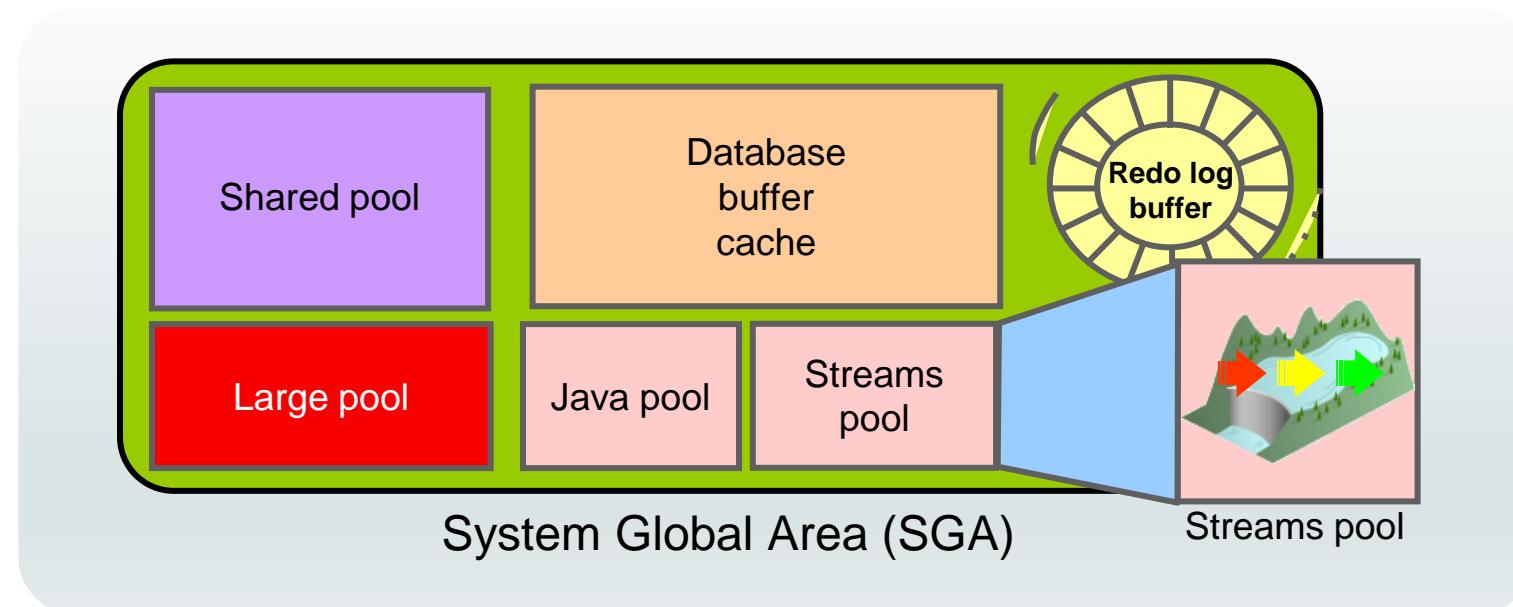
Java pool memory is used to store all session-specific Java code and data in the JVM.



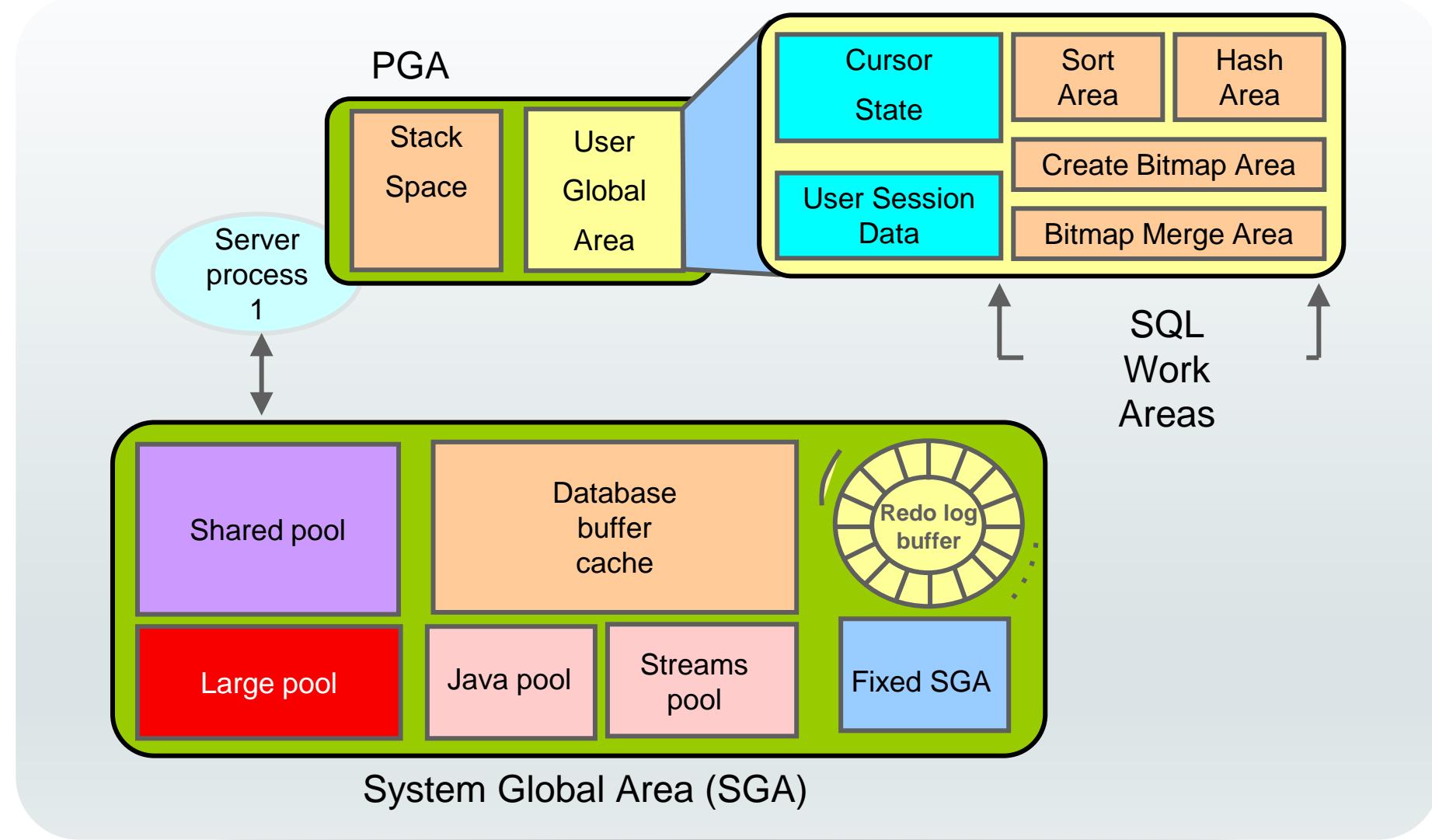
Streams Pool

Streams pool memory is used exclusively by Oracle Streams to:

- Store buffered queue messages
- Provide memory for Oracle Streams processes



Program Global Area (PGA)



Managing Memory Components

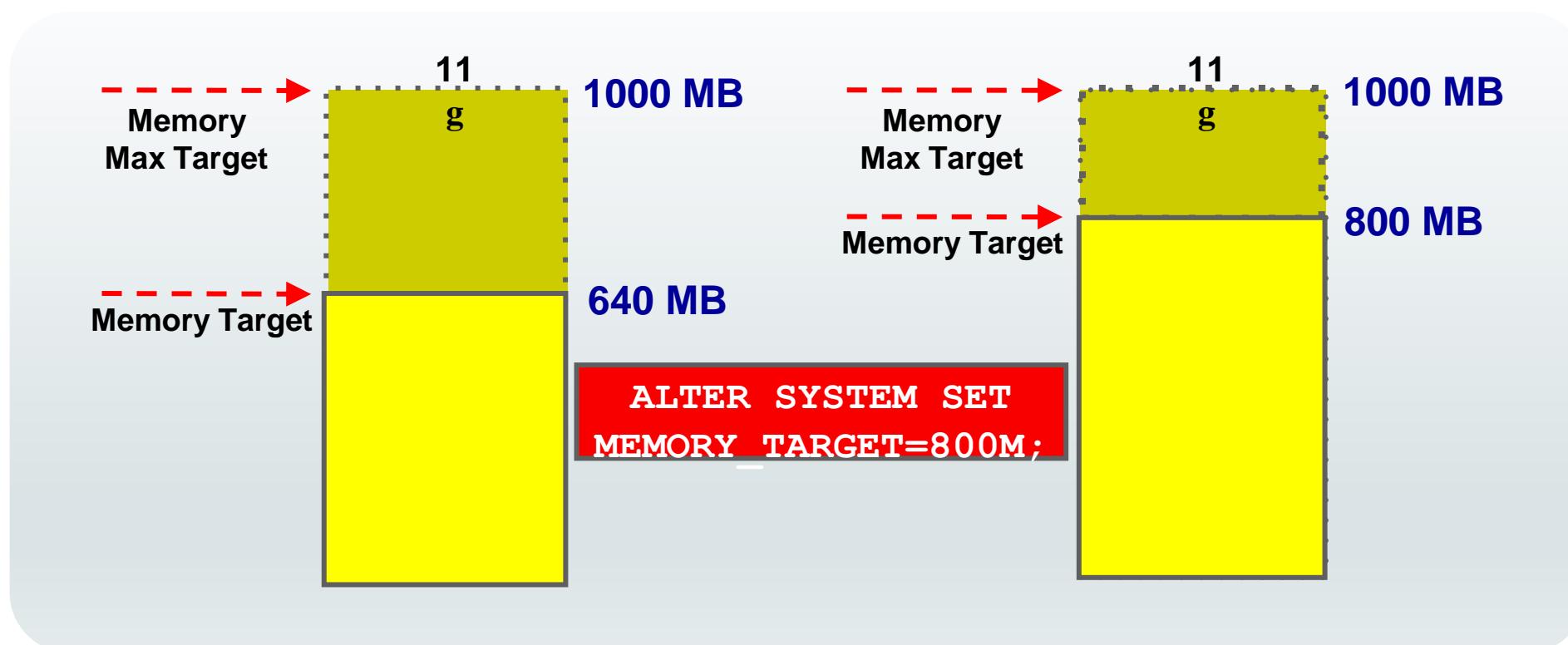
- Automatic Memory Management (AMM) enables you to specify total memory allocated to instance (including both SGA and PGA)
- Automatic Shared Memory Management (ASMM):
 - Enables you to specify total SGA memory through one initialization parameter
 - Enables the Oracle server to manage the amount of memory allocated to the shared pool, Java pool, buffer cache, streams pool, and large pool
- Manually setting shared memory management:
 - Sizes the components through multiple individual initialization parameters
 - Uses the appropriate Memory Advisor to make recommendations

Efficient Memory Usage: Guidelines

- Fit the SGA into physical memory.
- Use the Memory Advisors.
- Tune for the most efficient use of memory
 - Reduce overall physical I/O
 - Reduce the total memory needs

Automatic Memory Management

With Automatic Memory Management, the database server can size the SGA and PGA automatically according to your workload.



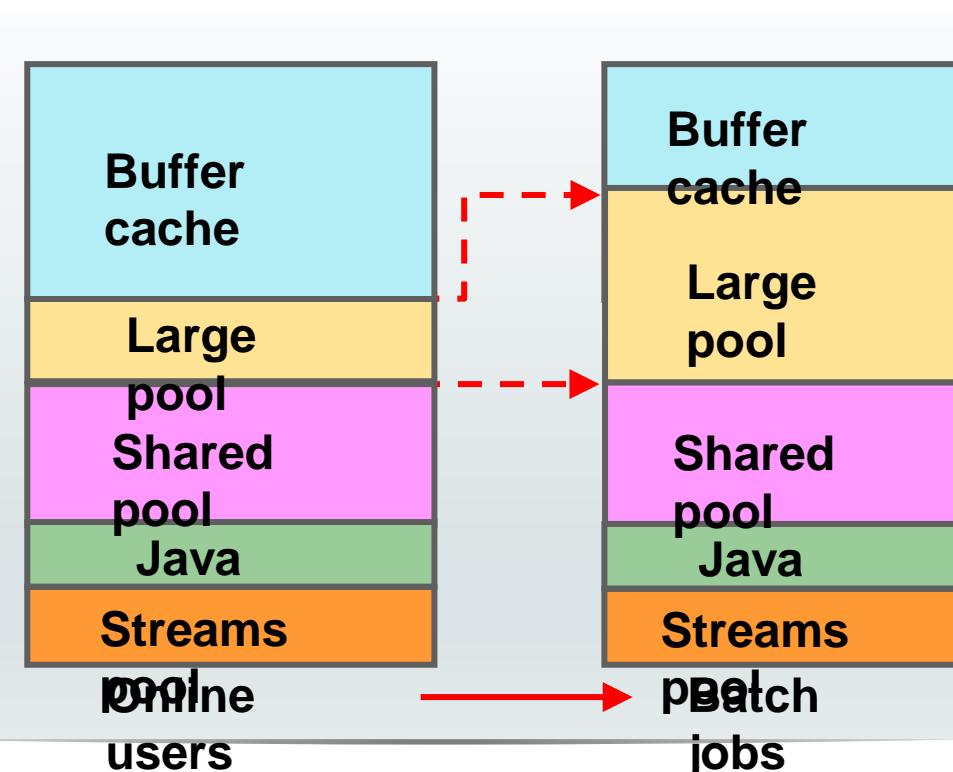
Monitoring Automatic Memory Management

View	Description
V\$MEMORY_DYNAMIC_COMPONENTS	Shows the current sizes of all dynamically tuned memory components
V\$MEMORY_RESIZE_OPS	Shows a circular history buffer of the last 800 memory resize requests
V\$MEMORY_TARGET_ADVICE	Provides tuning advice for the MEMORY_TARGET initialization parameter

Automatic Shared Memory Management

- Automatically adapts to workload changes
- Maximizes memory utilization
- Helps eliminate out-of-memory errors

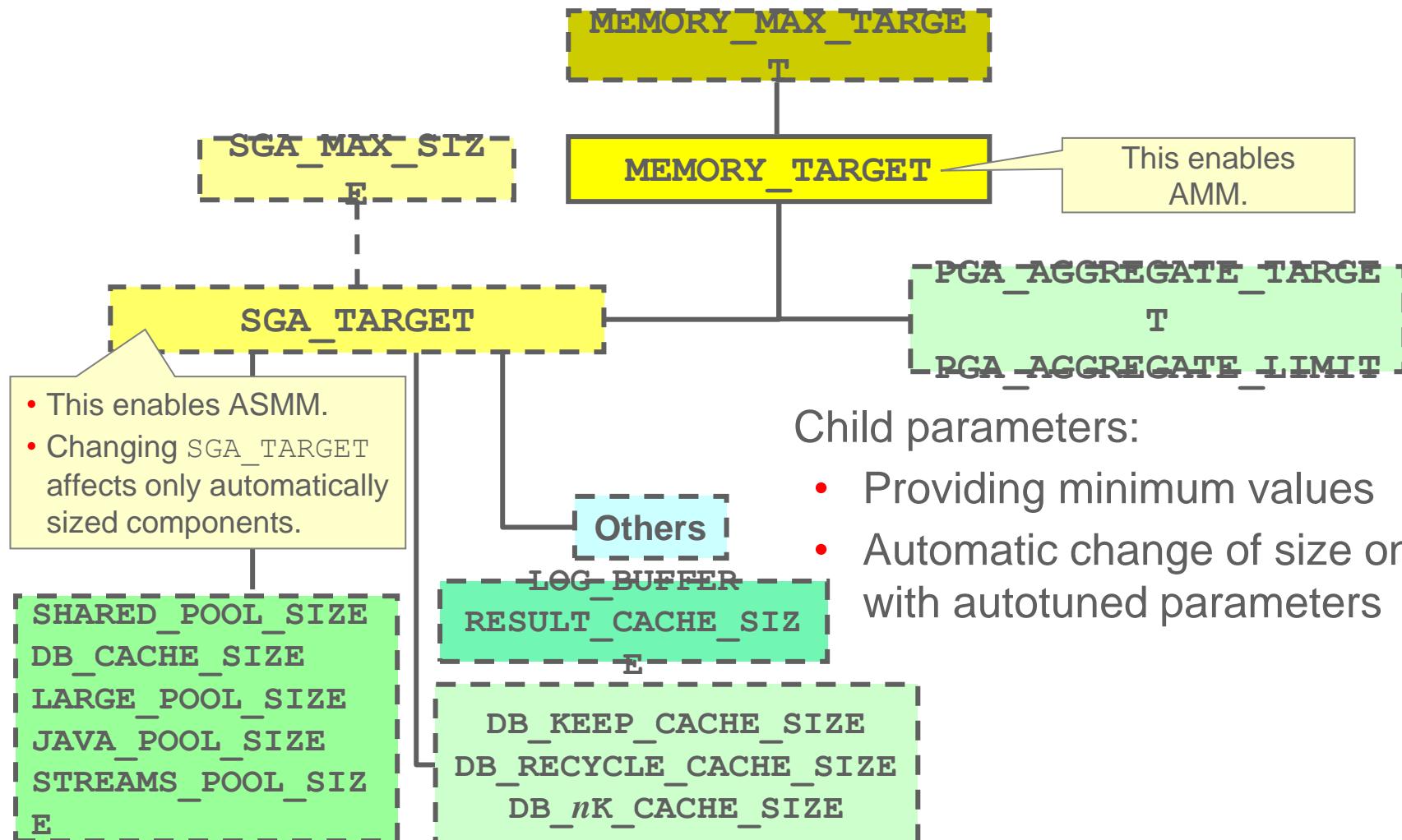
Example:



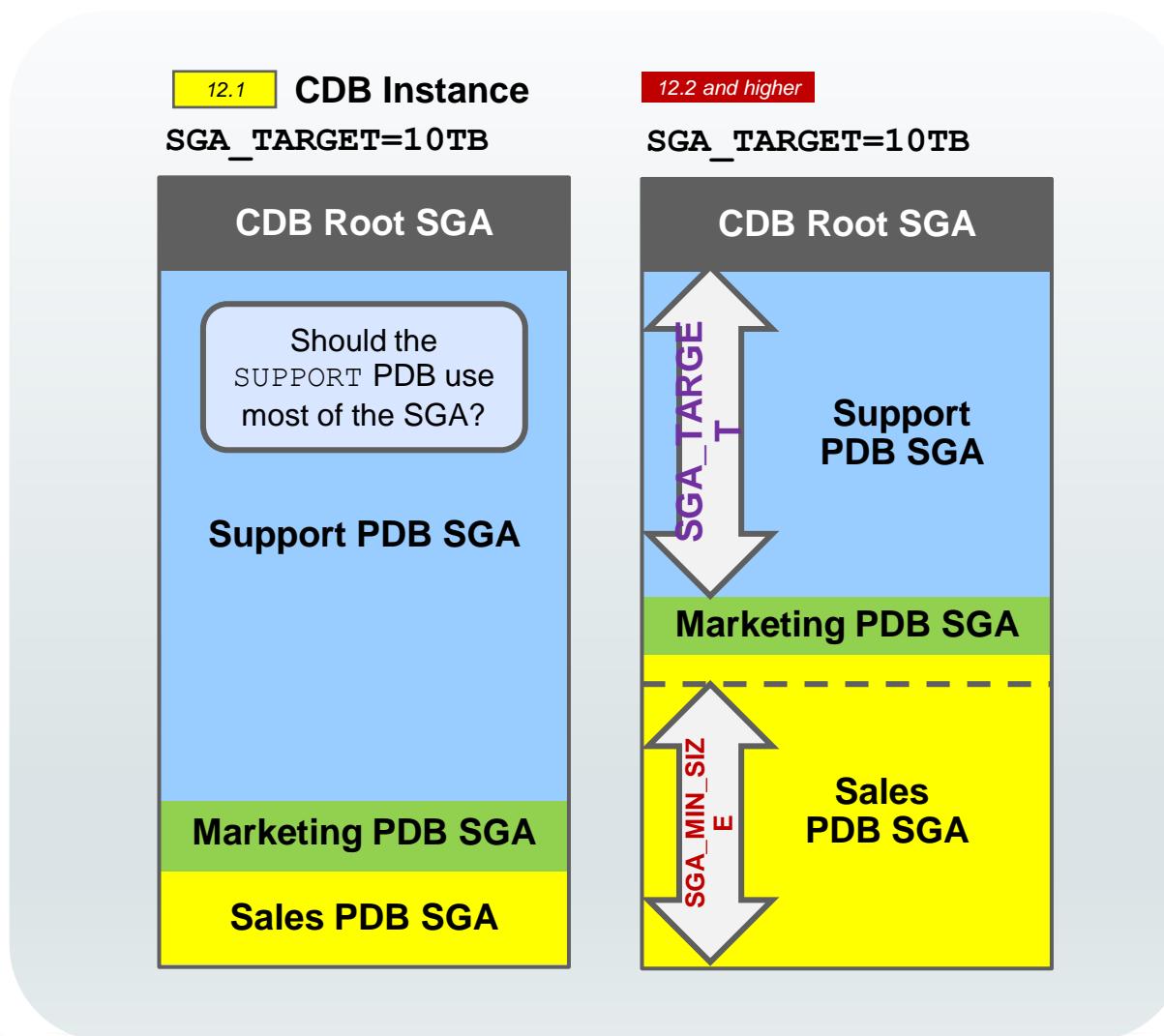
Understanding Automatic Shared Memory Management

- ASMM is based on workload information that MMON captures in the background.
- MMON uses memory advisors.
- Memory is moved to where it is needed the most by MMAN.
- If an SPFILE is used (which is recommended):
 - Component sizes are saved across shutdowns
 - Saved values are used to bootstrap component sizes
 - There is no need to relearn optimal values

Oracle Database Memory Parameters

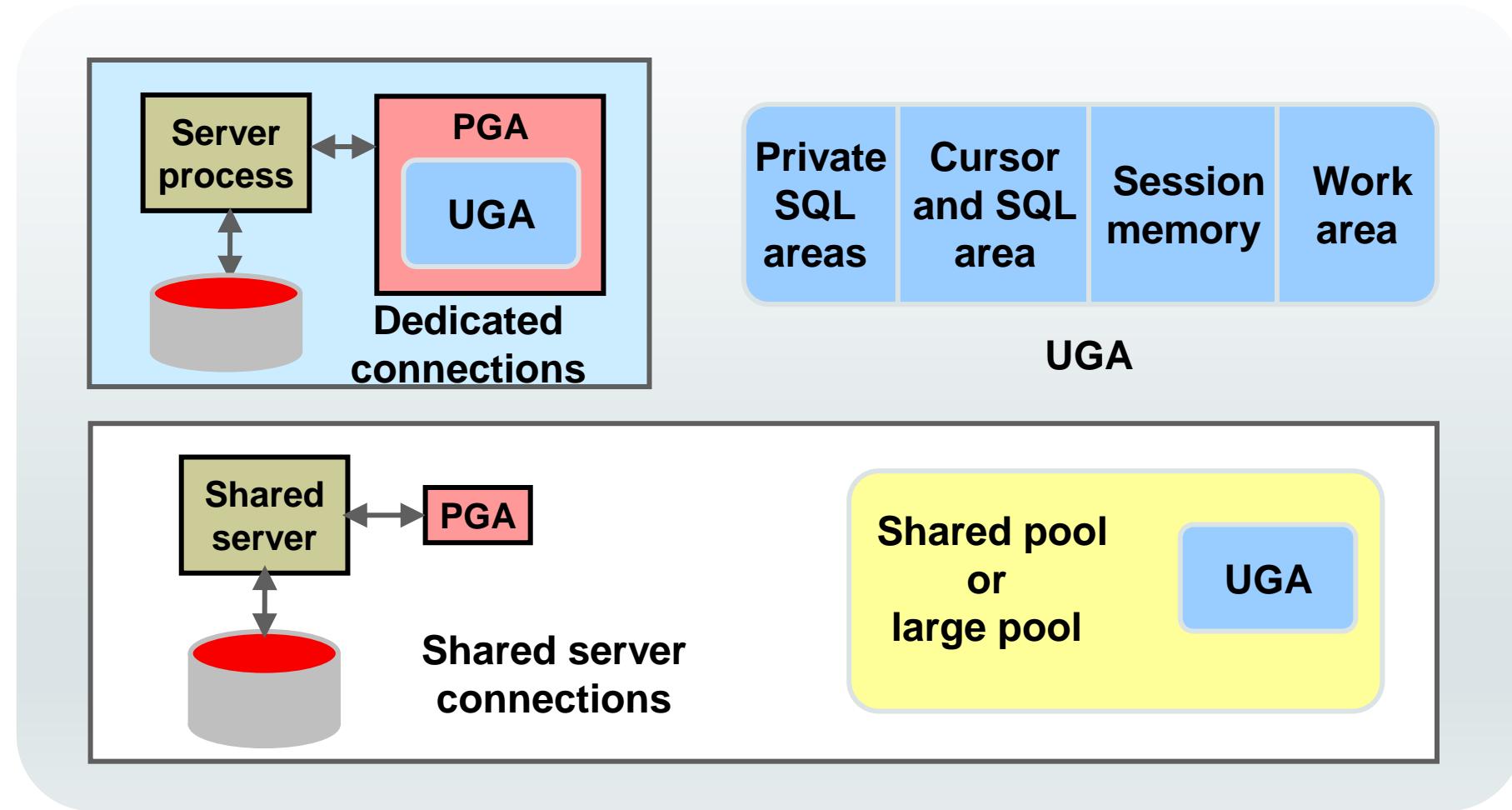


Managing the SGA for PDBs



- **SGA_TARGET** set at PDB level enforces a hard limit for the PDB's SGA.
- **SGA_TARGET** at PDB level provides more SGA for other containers.
- **SGA_MIN_SIZE** set for a PDB guarantees SGA space for the PDB.
- Parameters at PDB level:
 - **DB_CACHE_SIZE**
 - **SHARED_POOL_SIZE**
- PDB minimums cannot be > 50% of memory

Managing the Program Global Area (PGA)



Automatic PGA memory management is enabled by default.

Managing the PGA for PDBs

Instance PGA_AGGREGATE_LIMIT

- No more PGA can be allocated.
- Calls or sessions of the largest PGA users are terminated.

Instance PGA_AGGREGATE_TARGET

- All sessions must use TEMP rather than PGA.

PDB PGA_AGGREGATE_LIMIT

PDB PGA_AGGREGATE_TARGET

- These parameters set the same behavior at the PDB level.

CDB Instance

PGA_AGGREGATE_LIMIT=1TB

PGA_AGGREGATE_TARGET=500GB



CDB Root PGA

Support PDB SGA

PGA_AGGREGATE_LIMIT=300M

PGA_AGGREGATE_TARGET=150M

Sales PDB SGA

PGA_AGGREGATE_LIMIT=200M

PGA_AGGREGATE_TARGET=100M

Summary

In this lesson, you should have learned how to configure and monitor memory components for optimal performance.

Practice Overview

- Viewing Memory Configurations

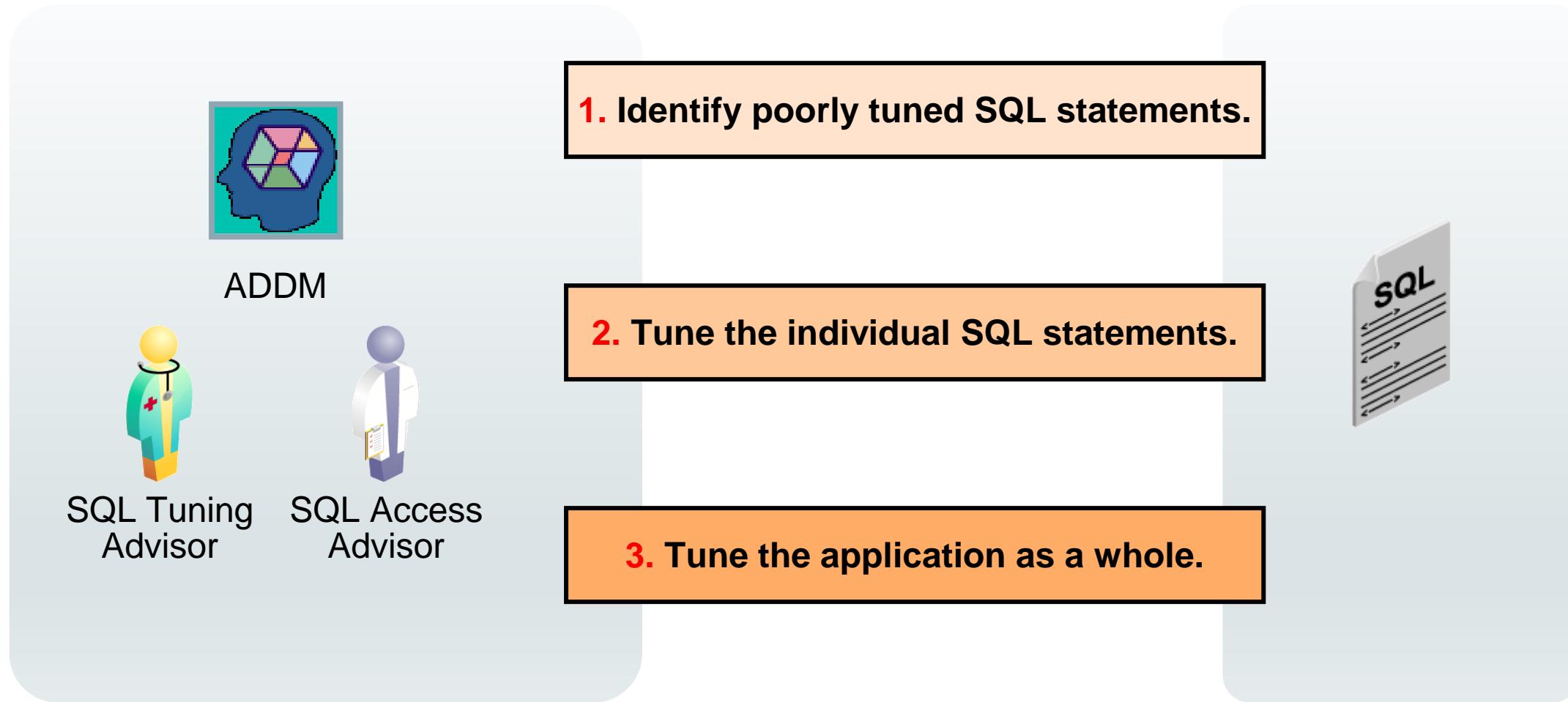
33. Analyzing SQL and Optimizing Access Paths

Objectives

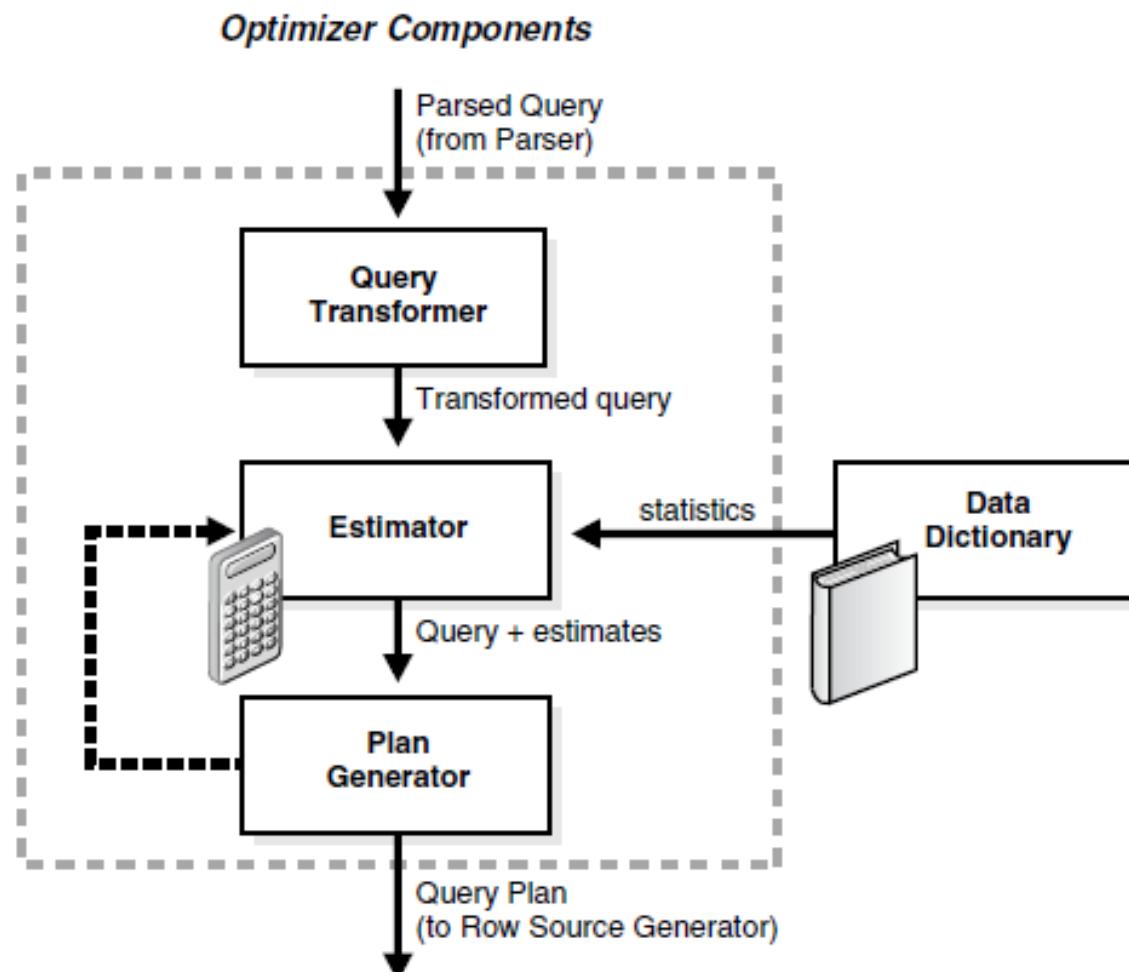
After completing this lesson, you should be able to:

- Describe the SQL tuning methodology
- Manage optimizer statistics
- Use SQL Tuning Advisor to identify and tune SQL statements that are using the most resources
- Use SQL Access Advisor to tune a workload

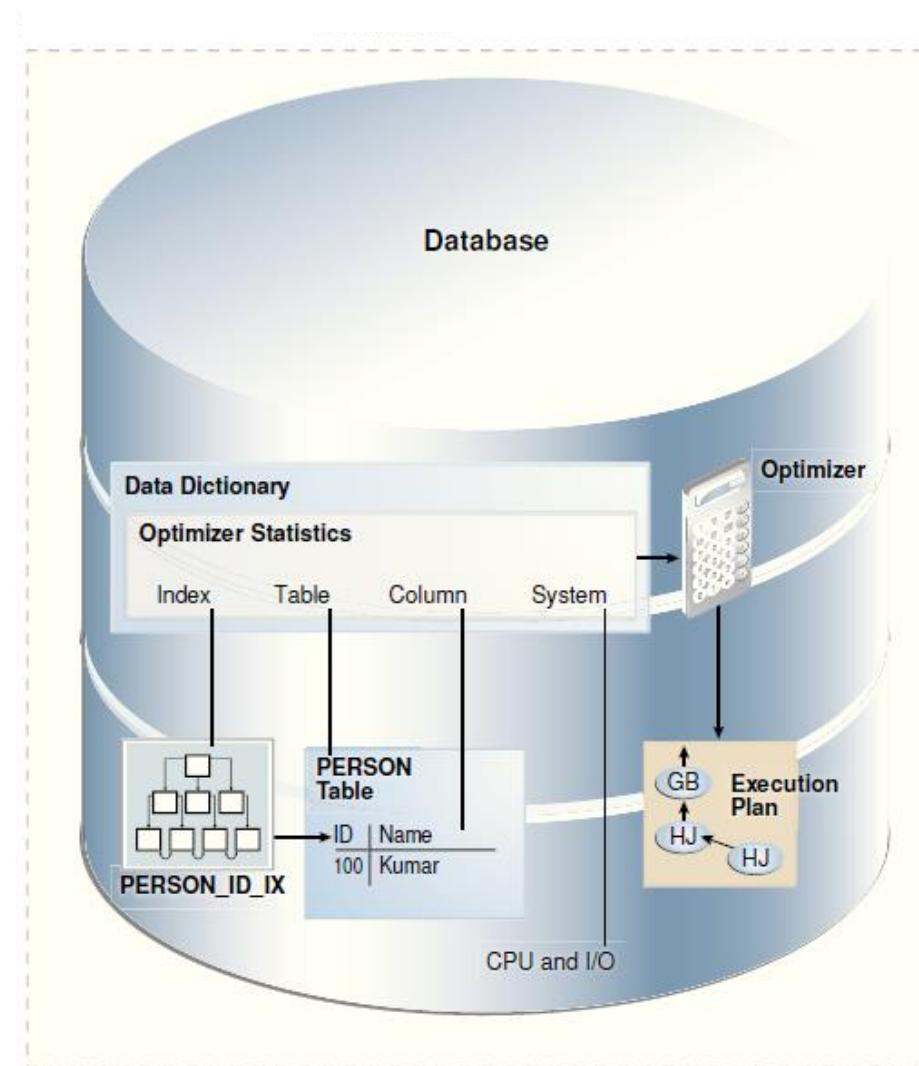
SQL Tuning Process



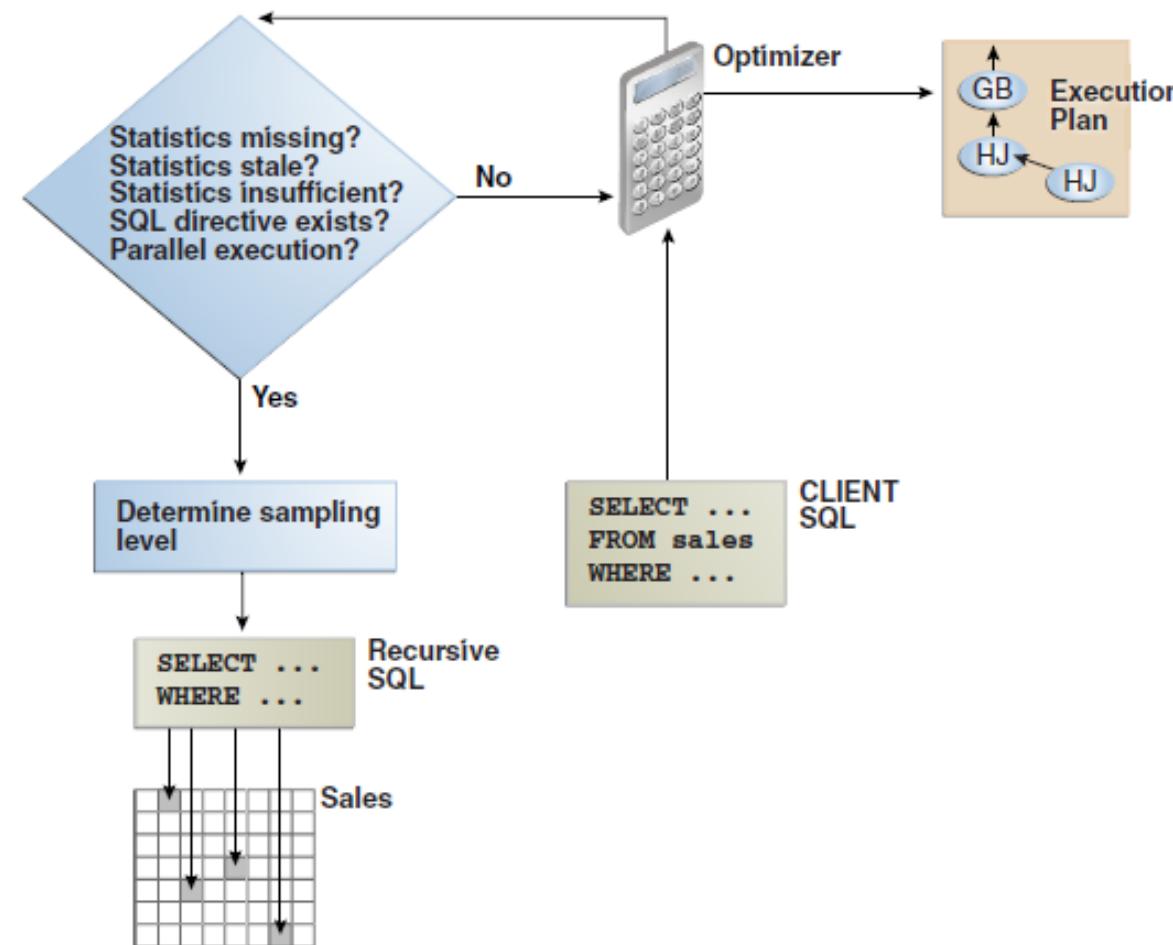
Oracle Optimizer



Optimizer Statistics



Optimizer Statistics Collection



Setting Optimizer Statistics Preferences

DBMS_STATS.GATHER_*_STATS procedures: Gather statistics for an entire database or for individual objects using default values

Use the SET_*_PREFS procedures to create preference values for any object that is not owned by SYS or SYSTEM

Query DBA_TAB_STAT_PREFS to view object-level preferences

Execute the DBMS_STATS.GET_PREFS procedure for each preference to see the global preferences

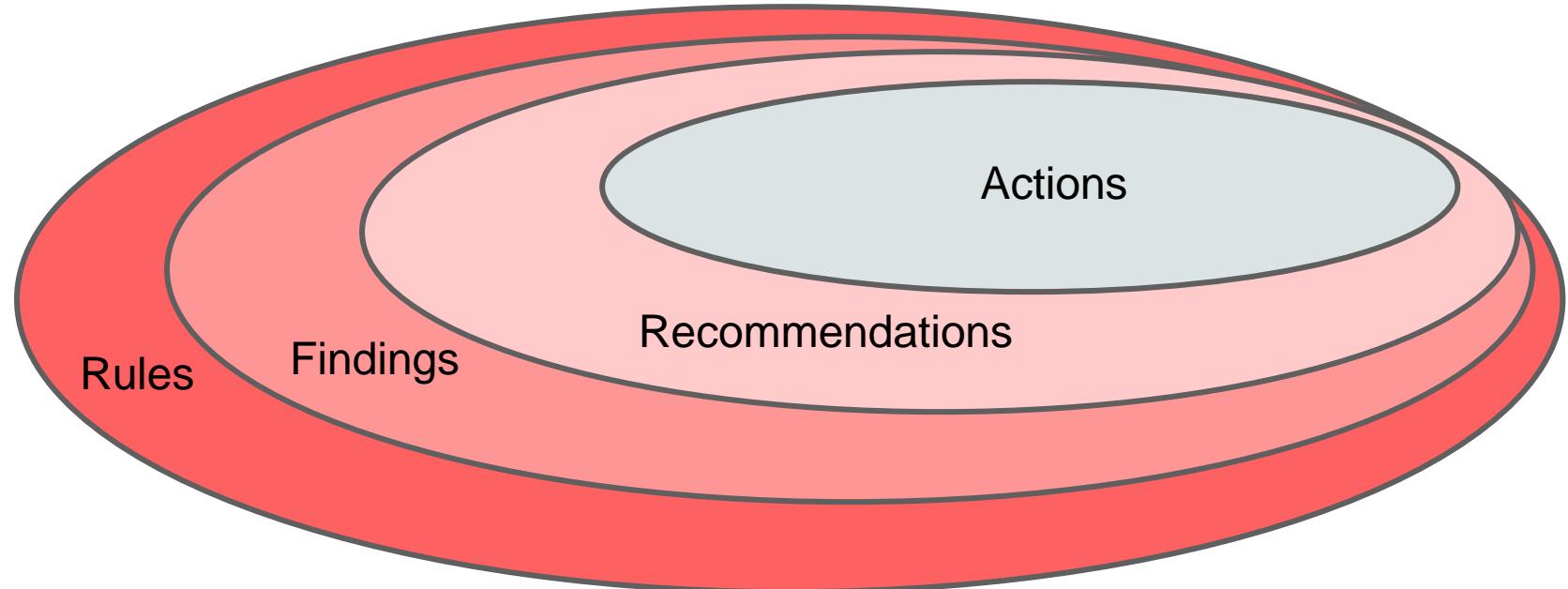
Optimizer Statistics Advisor

- If best practices change in a new release, Optimizer Statistics Advisor encodes these practices in its **rules**.
- The advisor always provides the most up-to-date recommendations.
- Track and analyze how statistics are collected.
 - Class of findings: System, Operations, Objects
- Scope of findings
 - Problems with gathering of statistics
 - Status of automatic statistic gathering jobs
 - Quality of current statistics
- Suggestion for changes to the statistics collection

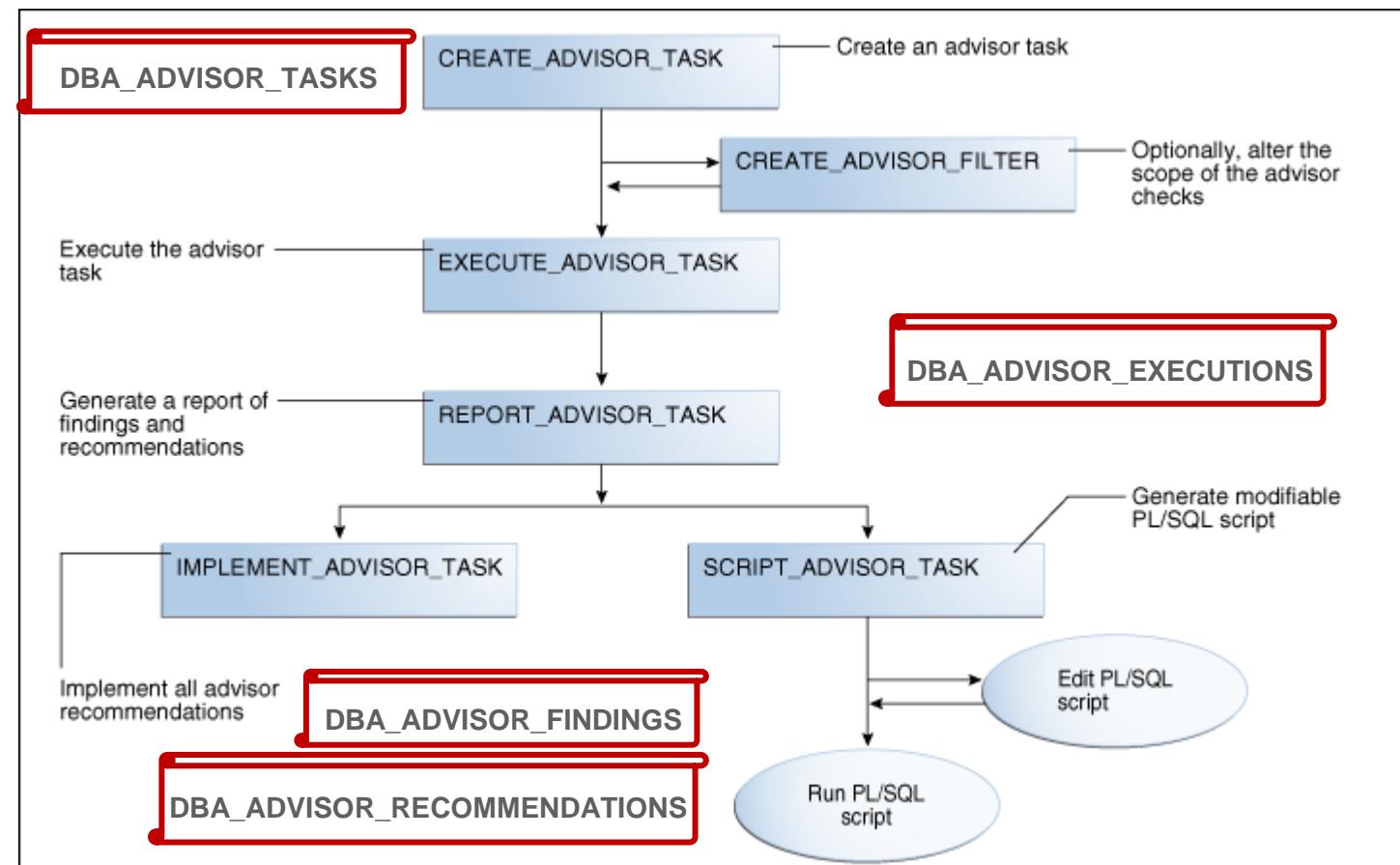
Optimizer Statistics Advisor Report

Report sections:

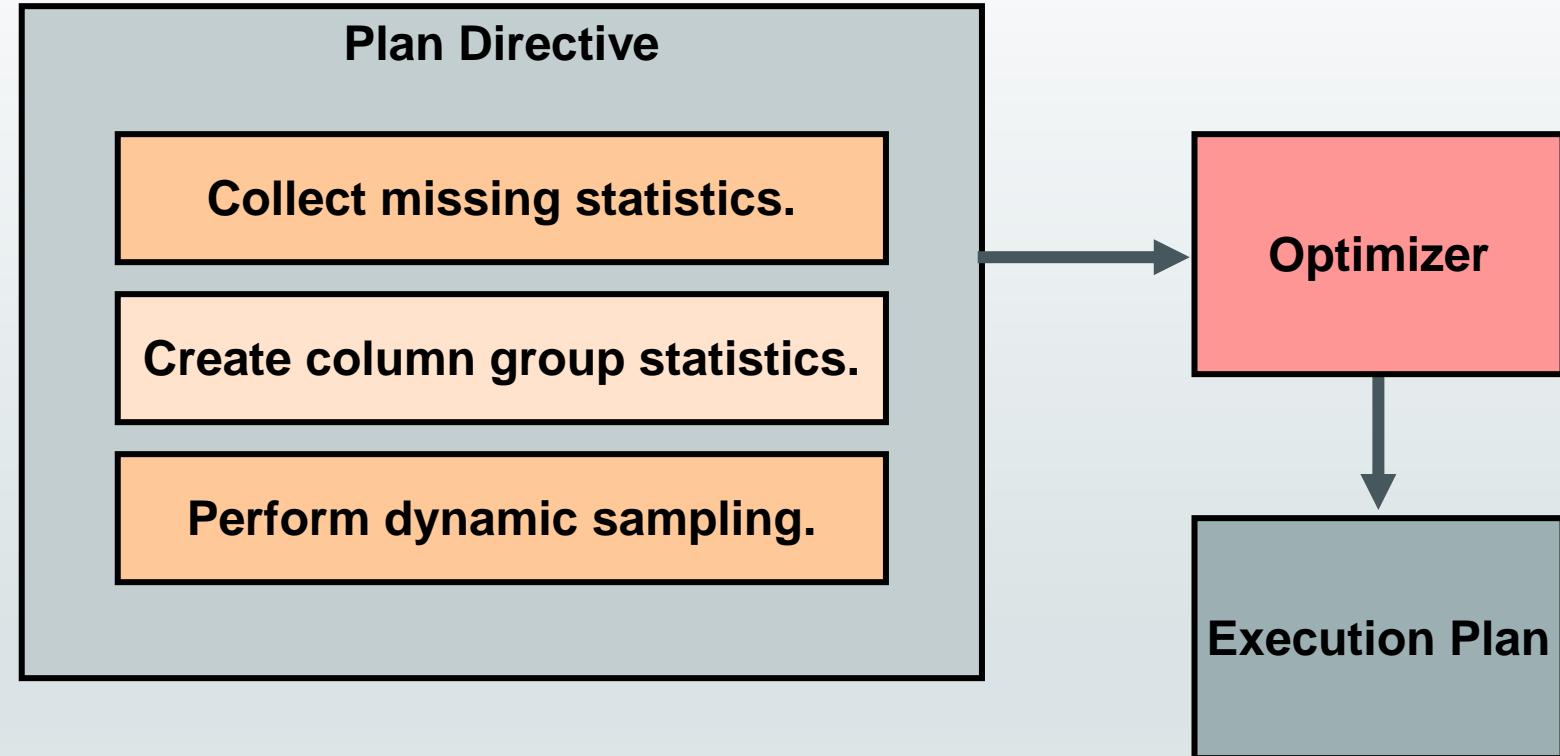
- Header
- Summary
- Errors
- Findings



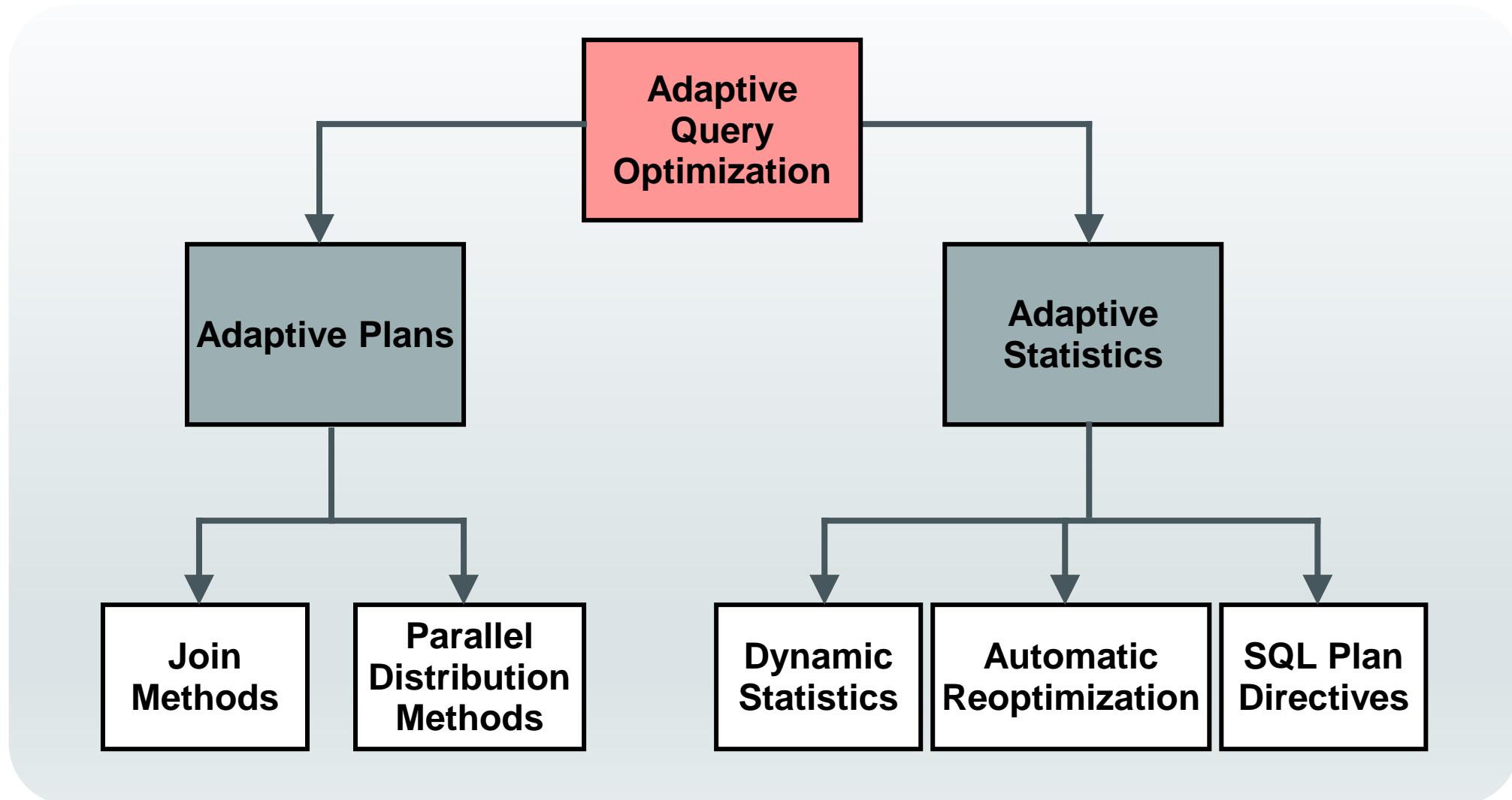
Executing Optimizer Statistics Advisor Tasks



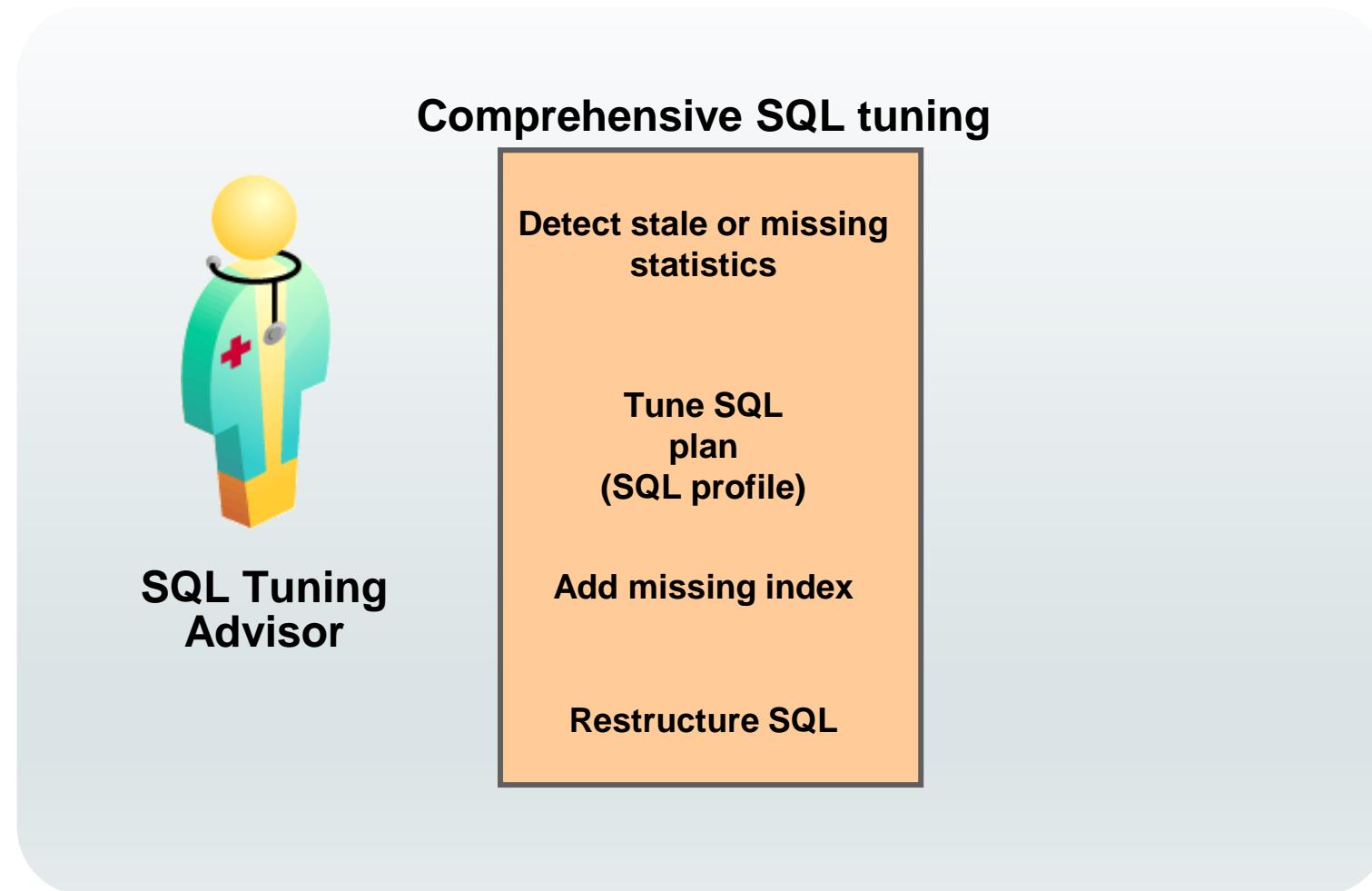
SQL Plan Directives



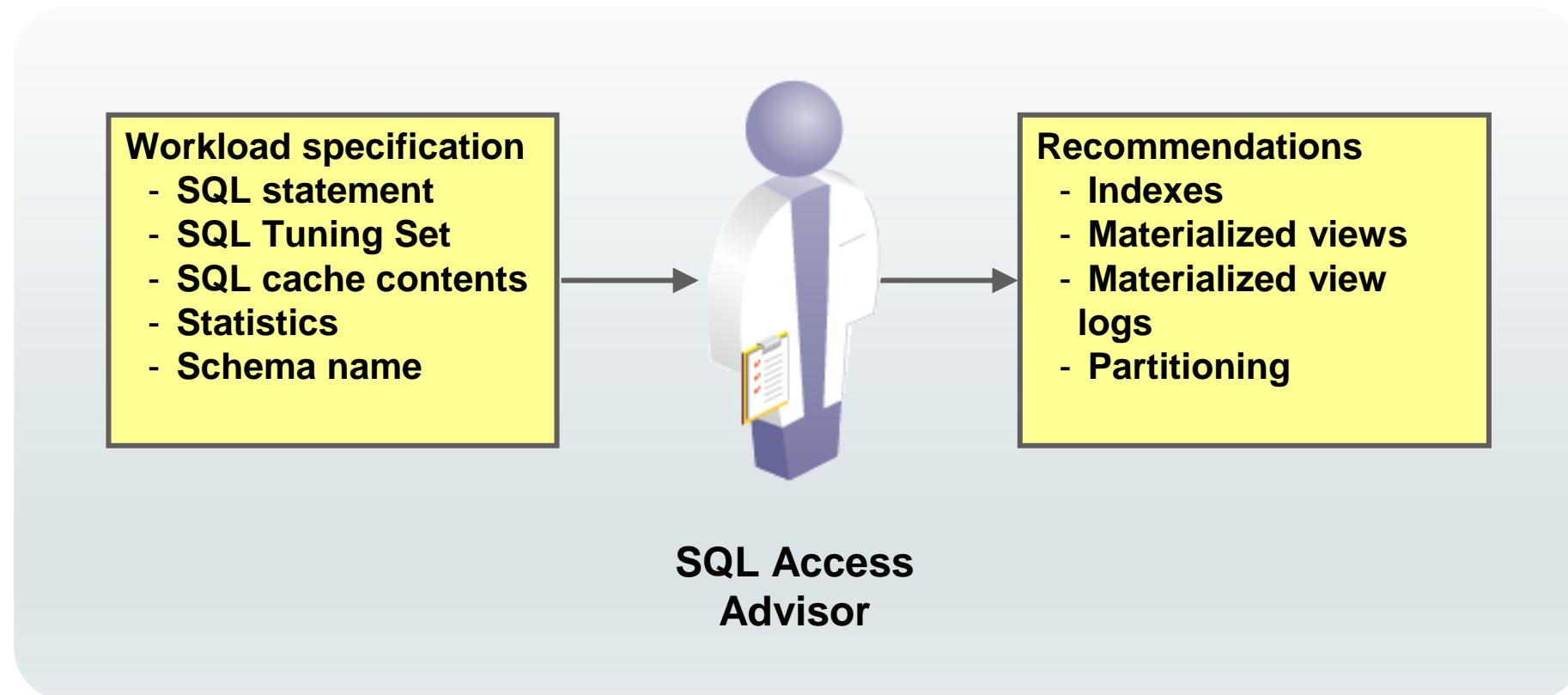
Adaptive Execution Plans



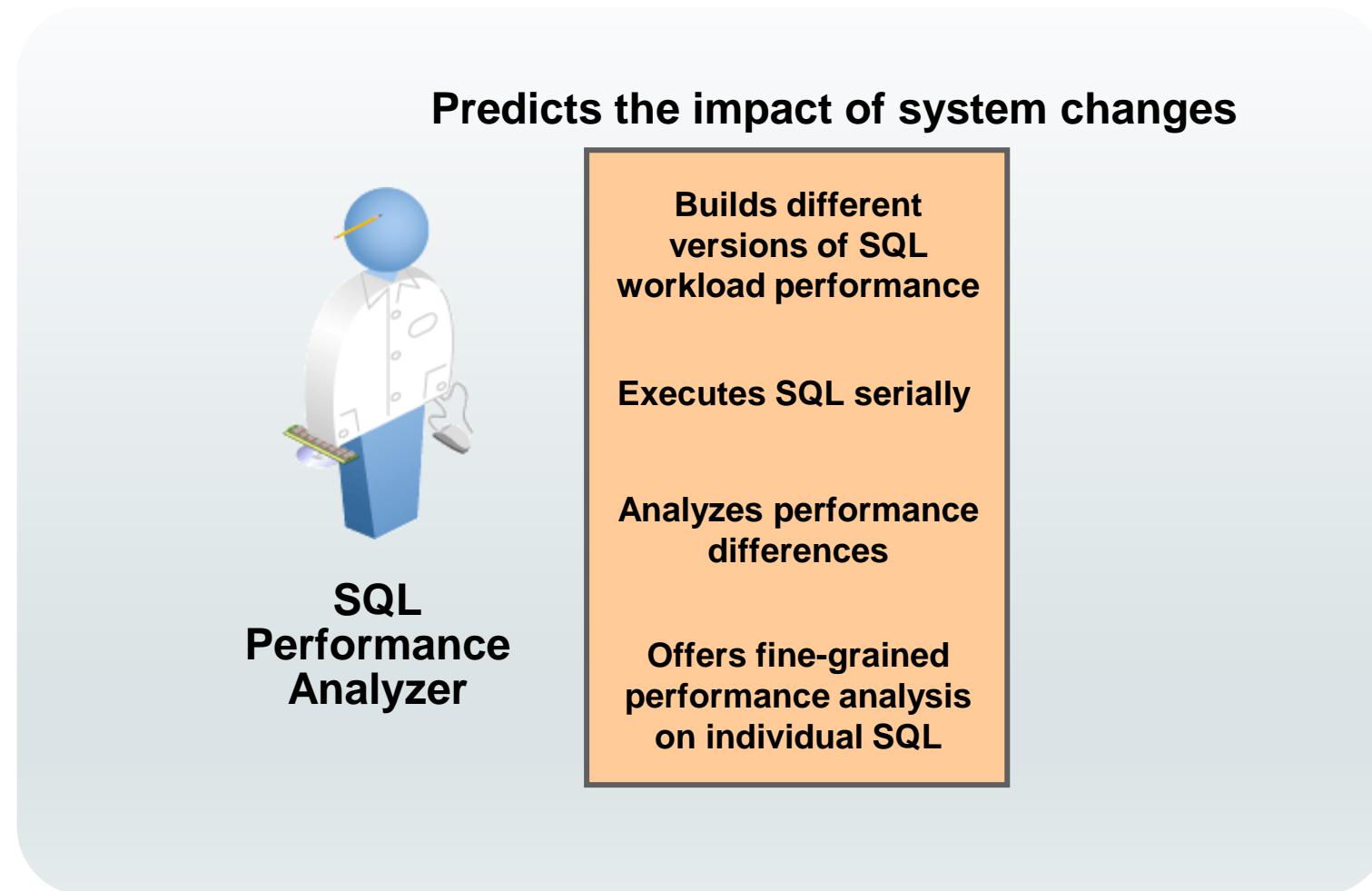
SQL Tuning Advisor: Overview



SQL Access Advisor: Overview



SQL Performance Analyzer: Overview



Managing Automated Tuning Tasks

- Use the DBMS_AUTO_TASK_ADMIN.ENABLE procedure to manage automatic space and performance tuning tasks
- Set the CLIENT_NAME parameter to the following values based on the task:
 - Automatic statistics collection: auto optimizer stats collection
 - Automatic SQL Tuning task: sql tuning advisor
 - Segment shrink: auto space advisor

```
BEGIN  
    DBMS_AUTO_TASK_ADMIN.ENABLE (  
        client_name => 'auto optimizer stats collection' ,  
        operation     => NULL , window_name => NULL) ;  
END ;
```

Summary

In this lesson, you should have learned how to:

- Describe the SQL tuning methodology
- Manage optimizer statistics
- Use SQL Tuning Advisor to identify and tune SQL statements that are using the most resources
- Use SQL Access Advisor to tune a workload

Practice Overview

- Using the SQL Tuning Advisor
- Using the Optimizer Statistics Advisor