Oracle 19c Database Administration

Introduction

Table Of Contents

Table Of Contents…

Objectives

- After completing this lesson, you should be able to:
  - Describe the differences in Oracle Database editions, options, and packs
  - List the database offerings in Oracle Cloud
  - Explain the sample database that will be used in the course practices

Course Objectives

- After completing this course, you should be able to:
  - Describe Oracle Database architecture
  - Explain Oracle Database Cloud Service (DBCS) architecture and features
  - Create and manage DBCS database deployments
  - Configure the database to support your applications
  - Manage database security and implement auditing
  - Implement basic backup and recovery procedures
  - Move data between databases and files
  - Employ basic monitoring procedures and manage performance

Introducing Oracle Database

- Oracle provides cloud and on-premises offerings.
- The purpose of Oracle Database is to store, organize, and retrieve data for your applications.
- You can install Oracle Database in your environment (on-premises) or use Oracle Database in Oracle's environment (cloud).

Oracle Database 19c

- First annual release of Oracle Database
- Will be released first on Oracle Cloud and Engineered Systems, with on-premises releases following
- Quarterly Release Updates (RUs) and Release Update Revisions (RURs) will be delivered

Oracle Database 19c On-Premises Editions

- Oracle Database is available in the following editions, each suitable for different development and deployment scenarios:
  - Oracle Database Personal Edition (PE)
  - Oracle Database Standard Edition 2 (SE2)
  - Oracle Database Enterprise Edition (EE)

Oracle Database Standard Edition 2

- SE2 supports Oracle Real Application Clusters (RAC).
- SE2 supports single tenant but lacks the following features, options, and tools:
  - Parallel execution
  - Oracle Data Guard
  - Enterprise Manager Cloud Control
  - Management packs

Oracle Database Options

Option	Description
Oracle Active Data Guard	Increases performance, availability, data protection, and return on investment wherever Data Guard is used for real-time data protection and availability
Oracle Advanced Analytics	Empowers data and business analysts to extract knowledge, discover new insights, and make predictions—working directly with large data volumes
Oracle Advanced Compression	Provides comprehensive data compression and Information Lifecycle Management (ILM) capabilities for all types of data
Oracle Advanced Security	Helps you protect sensitive information and comply with privacy and compliance regulations by enabling database encryption and data redaction
Oracle Database In-Memory	Enables any existing Oracle Database–compatible application to automatically and transparently take advantage of columnar in-memory processing, without additional programming or application changes
Oracle Database Vault	Enables you to control when, where, and by whom the database and application data can be accessed

Oracle Database Options

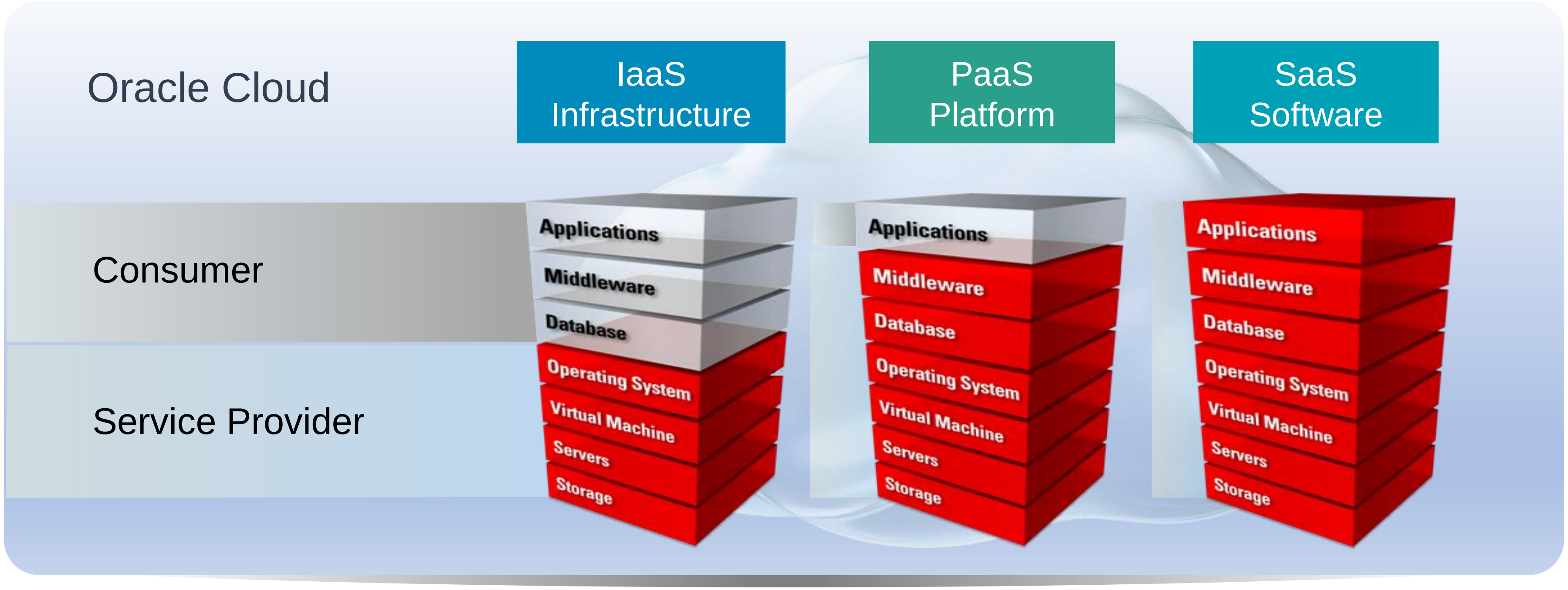
Option	Description
Oracle Label Security	Provides sophisticated and flexible security based on row labels for fine-grained access control
Oracle Multitenant	Enables an Oracle database to function as a multitenant container database (CDB) that includes one or more pluggable databases (PDBs)
Oracle On-Line Analytical Processing (OLAP)	A full-featured OLAP server embedded in Oracle Database Enterprise Edition
Oracle Partitioning	Adds significant manageability, availability, and performance capabilities to large underlying database tables and indexes
Oracle Real Application Clusters (Oracle RAC)	A database computing environment that harnesses the processing power of multiple interconnected computers using clustering technology
Oracle Real Application Testing	Comprises a suite of features that help protect database applications from the undesirable impact of routine changes
Oracle Spatial and Graph	Includes advanced features for spatial data and analysis and for physical, network, and social graph applications

Oracle Management Packs

Pack	Description
Oracle Cloud Management Pack for Oracle Database	Helps to set up a Database Cloud and operate the Database as a Service model
Oracle Data Masking and Subsetting Pack	Facilitates the creation of production-like data for nonproduction environments by replacing production data with fictitious yet realistic values
Oracle Database Lifecycle Management Pack for Oracle Database	Provides a comprehensive solution that helps database, system, and application administrators automate the processes required to manage the Oracle Database Lifecycle
Oracle Diagnostics Pack	Provides automatic performance diagnostic and advanced system monitoring functionality
Oracle Tuning Pack	Provides database administrators with expert performance management for the Oracle environment, including SQL tuning and storage optimizations

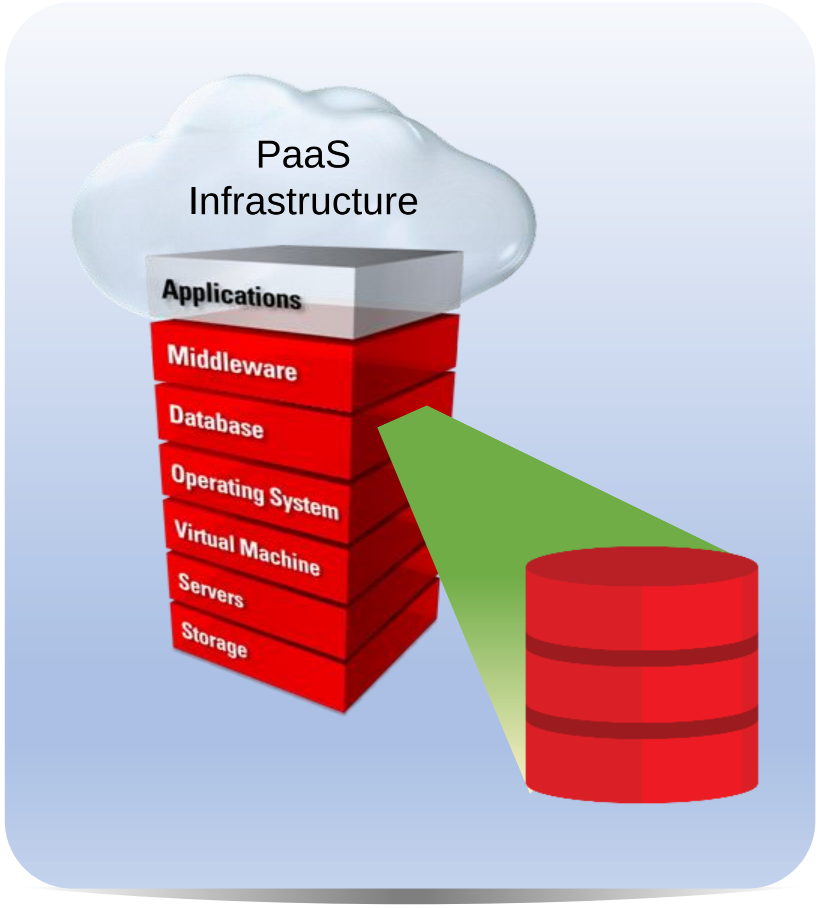
What Is Oracle Cloud?

- Oracle Cloud is an enterprise cloud for business. Oracle Cloud offers self-service business applications delivered on an integrated development and deployment platform with tools to extend services and create new services rapidly.



Oracle Database Cloud Service: Overview

- Oracle Database Cloud Service is a PaaS offering.
- With Oracle Database Cloud Service, you can:
  - Provision a full-featured dedicated Oracle database
  - Use cloud tooling to back up, patch, and manage the database
  - Avail of the complete administration privileges of the server and database to manage it as you need



Oracle Database Cloud Service Editions

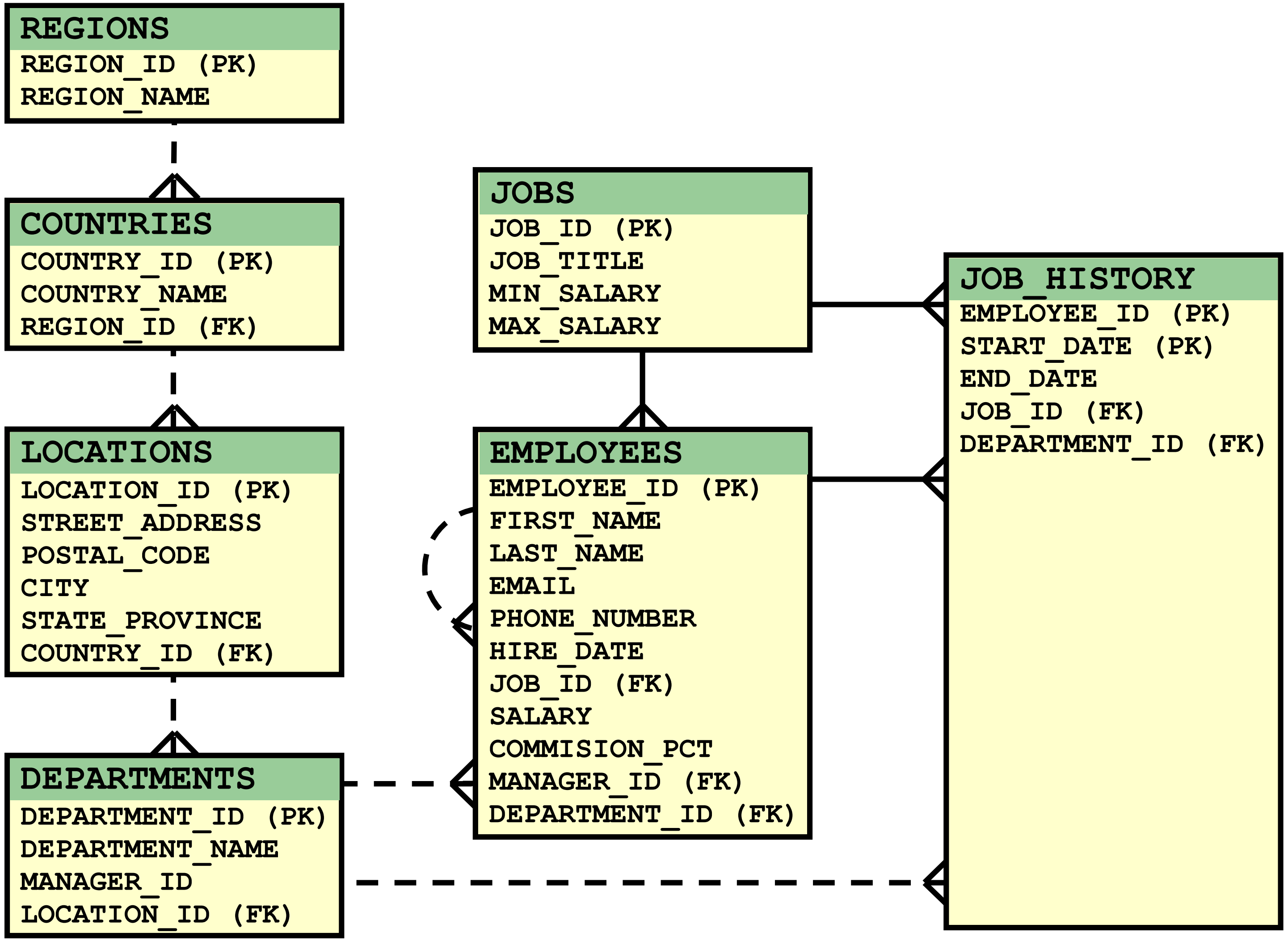
Edition	Included Options	Included Packs
Standard	None	None
Enterprise	None	None
Enterprise—High Performance	Advanced Analytics, Advanced Compression, Advanced Security, Database Vault, Label Security, Multitenant, OLAP, Partitioning, Real Application Testing, Spatial and Graph	Cloud Management for Oracle Database, Database Lifecycle Management, Data Masking and Subsetting, Diagnostics, Tuning
Enterprise—Extreme Performance	Active Data Guard, Advanced Analytics, Advanced Compression, Advanced Security, Database In-Memory, Database Vault, Label Security, Multitenant, OLAP, Partitioning, Real Application Clusters, Real Application Testing, Spatial and Graph	Cloud Management for Oracle Database, Database Lifecycle Management, Data Masking and Subsetting, Diagnostics, Tuning

Oracle SQL and PL/SQL

```
SQL> SELECT employee_id, first_name, last_name FROM employees
     WHERE employee_id=216 ORDER BY 1;
```

  - `SELECT` lists the database columns for which you want to view data.
  - `FROM` lists the tables that contain those database columns.
  - `WHERE` specifies column limits and table joins (this part essentially filters the rows of data).
  - `ORDER BY` specifies the columns by which the results are sorted.
- PL/SQL is a procedural extension to Oracle SQL.
  - It enables you to control the flow of a SQL program, use variables, and write error-handling procedures.

HR Schema



REGIONS
REGION_ID (PK)
REGION_NAME
COUNTRIES
COUNTRY_ID (PK)
COUNTRY_NAME
REGION_ID (FK)
LOCATIONS
LOCATION_ID (PK)
STREET_ADDRESS
POSTAL_CODE
CITY
STATE_PROVINCE
COUNTRY_ID (FK)
DEPARTMENTS
DEPARTMENT_ID (PK)
DEPARTMENT_NAME
MANAGER_ID
LOCATION_ID (FK)
JOBS
JOB_ID (PK)
JOB_TITLE
MIN_SALARY
MAX_SALARY
EMPLOYEES
EMPLOYEE_ID (PK)
FIRST_NAME
LAST_NAME
EMAIL
PHONE_NUMBER
HIRE_DATE
JOB_ID (FK)
SALARY
COMMISION_PCT
MANAGER_ID (FK)
DEPARTMENT_ID (FK)
JOB_HISTORY
EMPLOYEE_ID (PK)
START_DATE (PK)
END_DATE
JOB_ID (FK)
DEPARTMENT_ID (FK)

Suggested Course Schedule

Day	Lessons
1	1. Introduction 2. Oracle Database Architecture 3. Introduction to Oracle Database Cloud Service 4. Creating DBCS Database Deployments 5. Accessing an Oracle Database
2	6. Managing DBCS Database Deployments 7. Managing Database Instances 8. Understanding Oracle Net Services 9. Administering User Security

Day	Lessons
3	10. Creating PDBs 11. Creating Master Encryption Keys for PDBs 12. Creating and Managing Tablespaces 13. Managing Storage Space
4	14. Managing Undo Data 15. Moving Data 16. Backup and Recovery Concepts 17. Backup and Recovery Configuration
5	18. Creating Database Backups 19. Performing Database Recovery 20. Monitoring and Tuning Database Performance 21. SQL Tuning

Summary

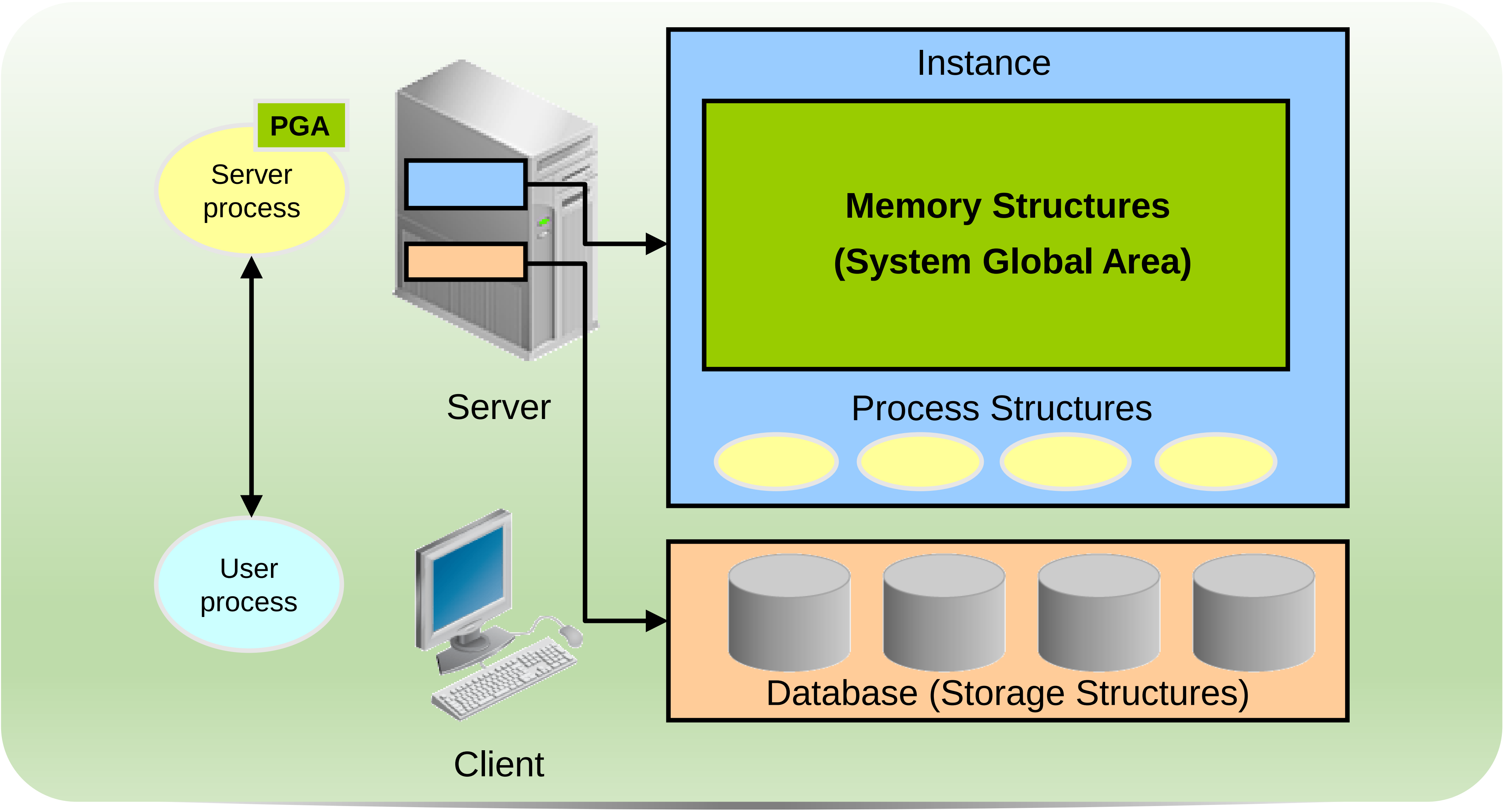
- In this lesson, you should have learned how to:
  - Describe the differences in Oracle Database editions, options, and packs
  - List the database offerings in Oracle Cloud
  - Explain the sample database that will be used in the course practices

Oracle Database Architecture

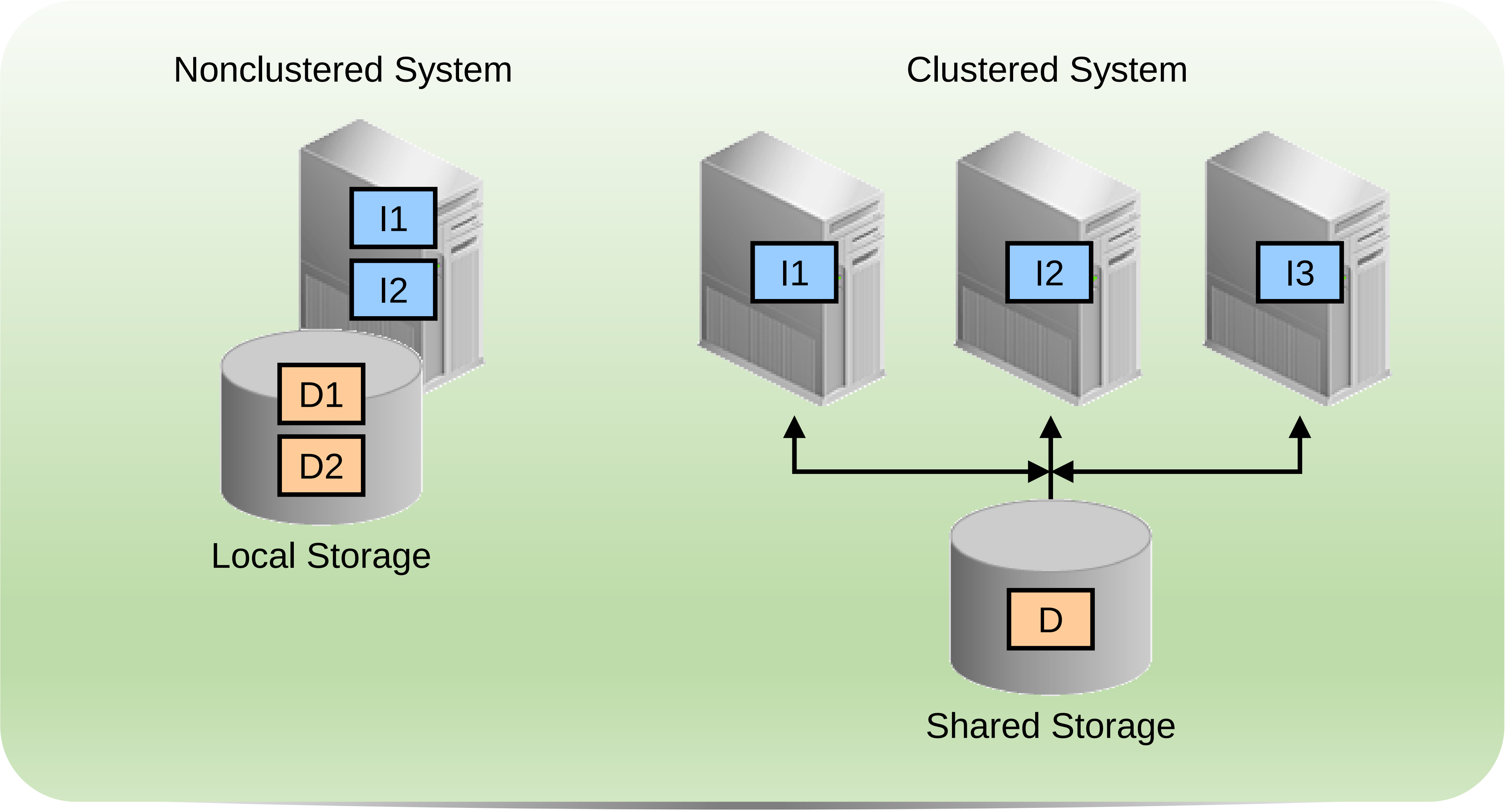
Objectives

- After completing this lesson, you should be able to:
  - List the major architectural components of Oracle Database
  - Explain memory structures
  - Describe background processes
  - Correlate logical and physical storage structures
  - Describe multitenant architecture

Oracle Database Server Architecture: Overview

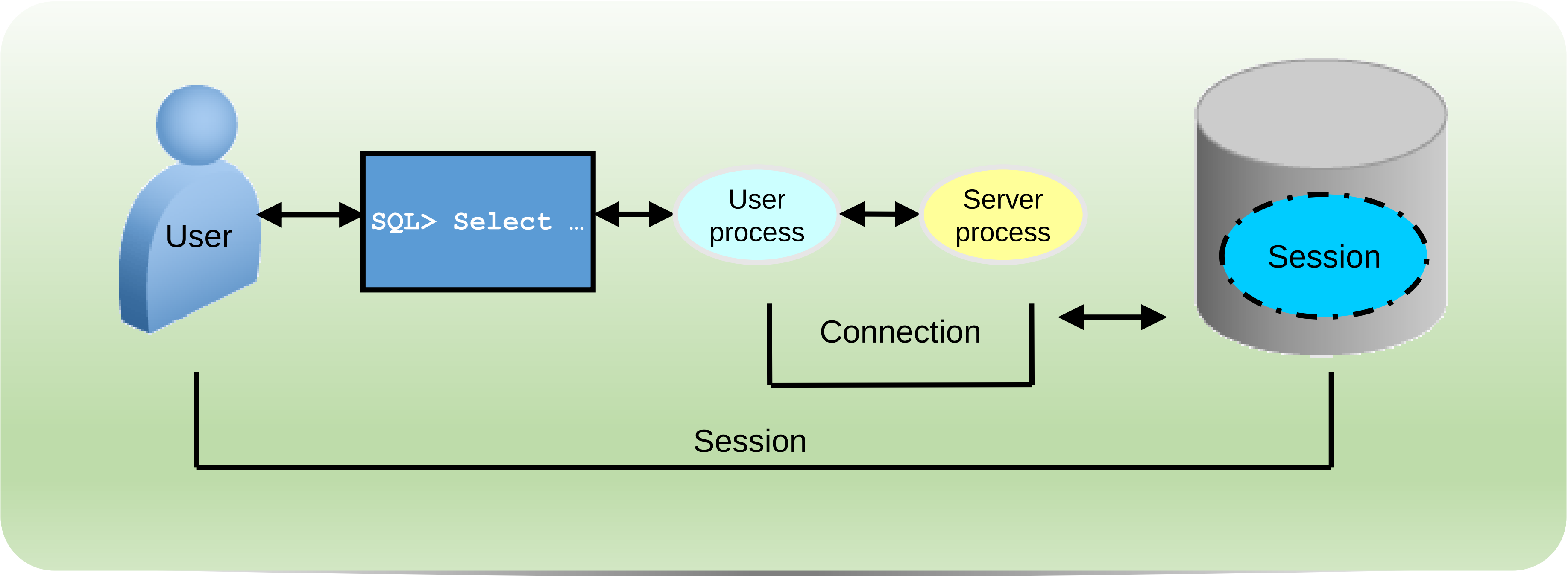


Oracle Database Instance Configurations

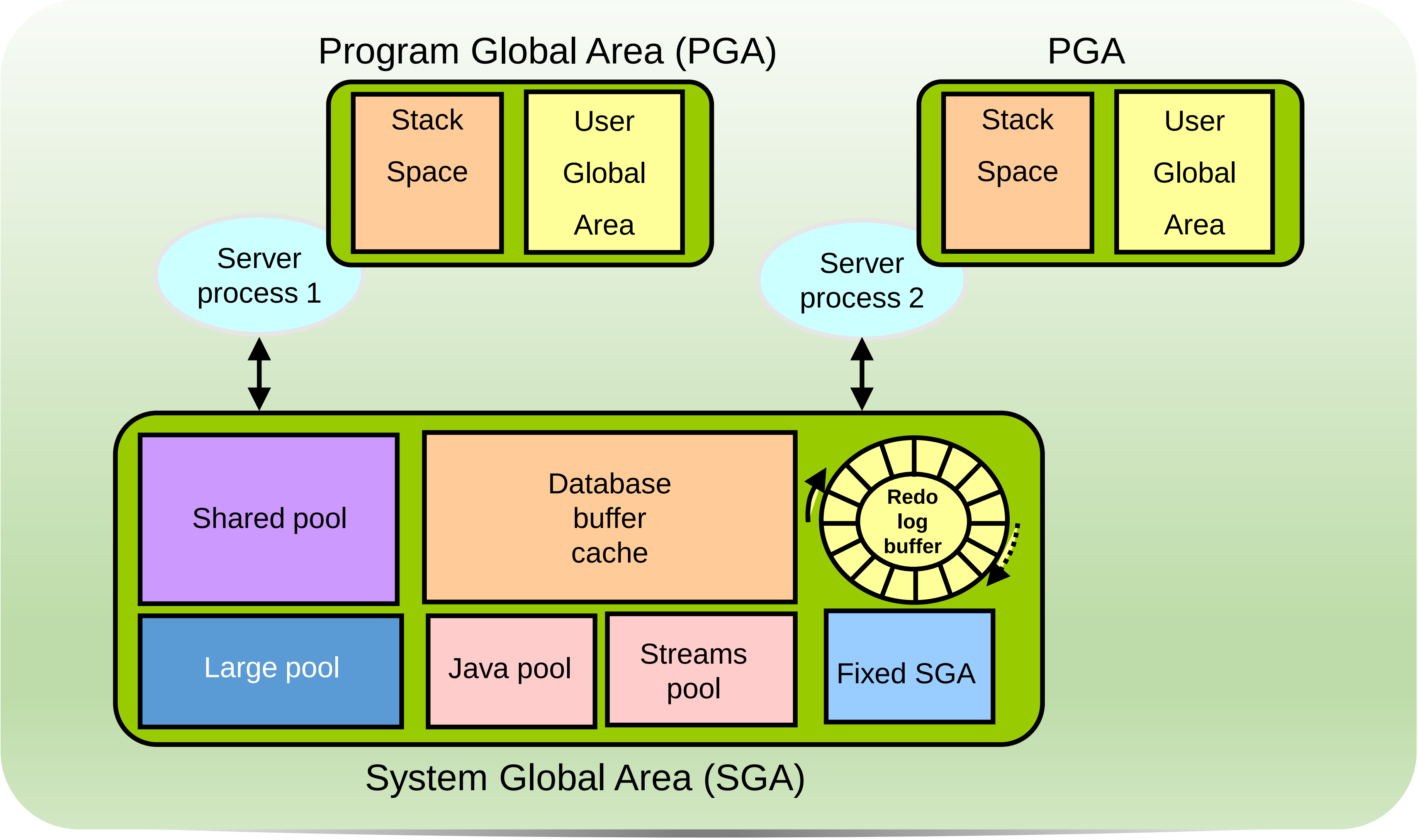


Connecting to the Database Instance

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

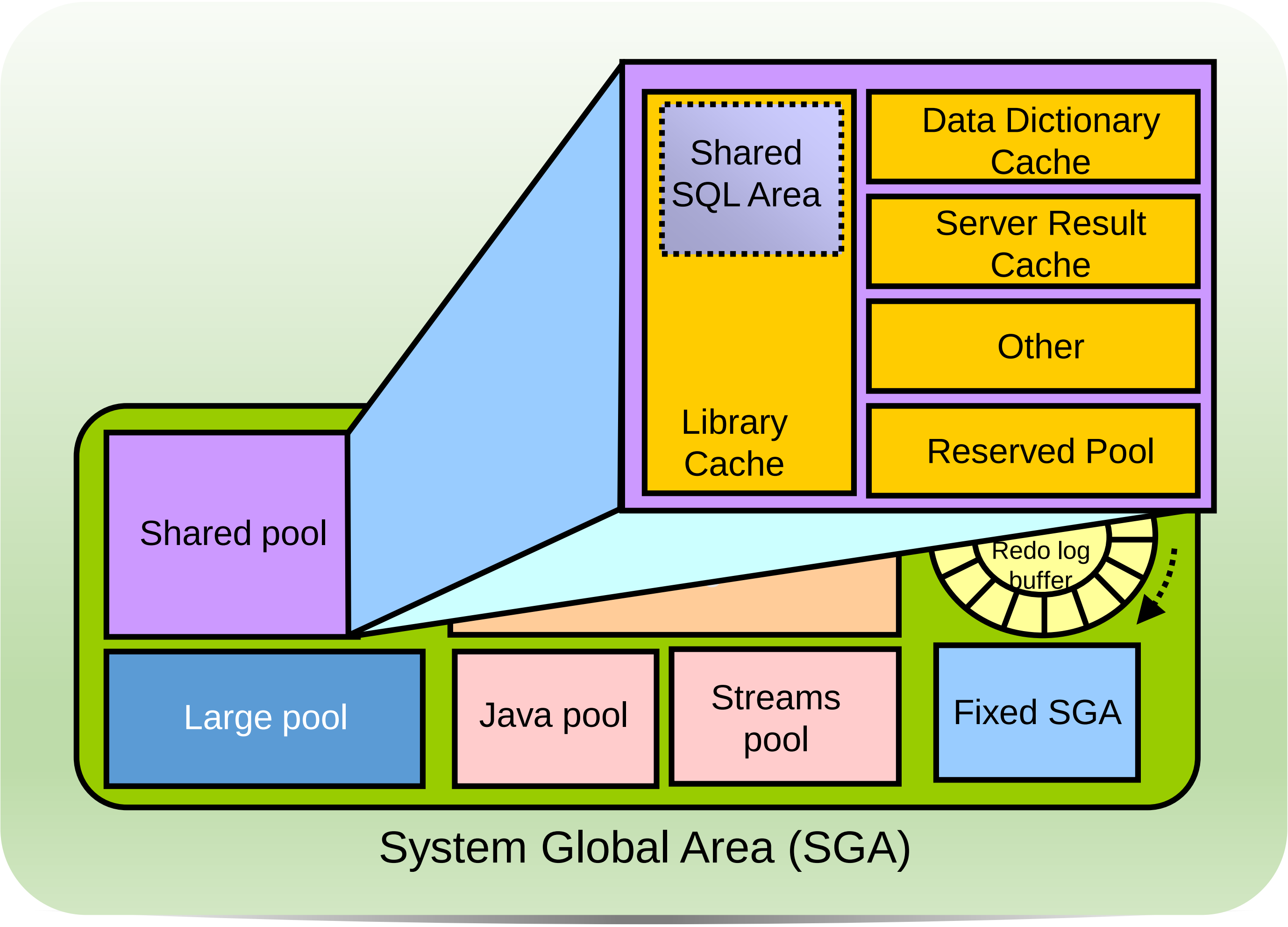


Oracle Database Memory Structures



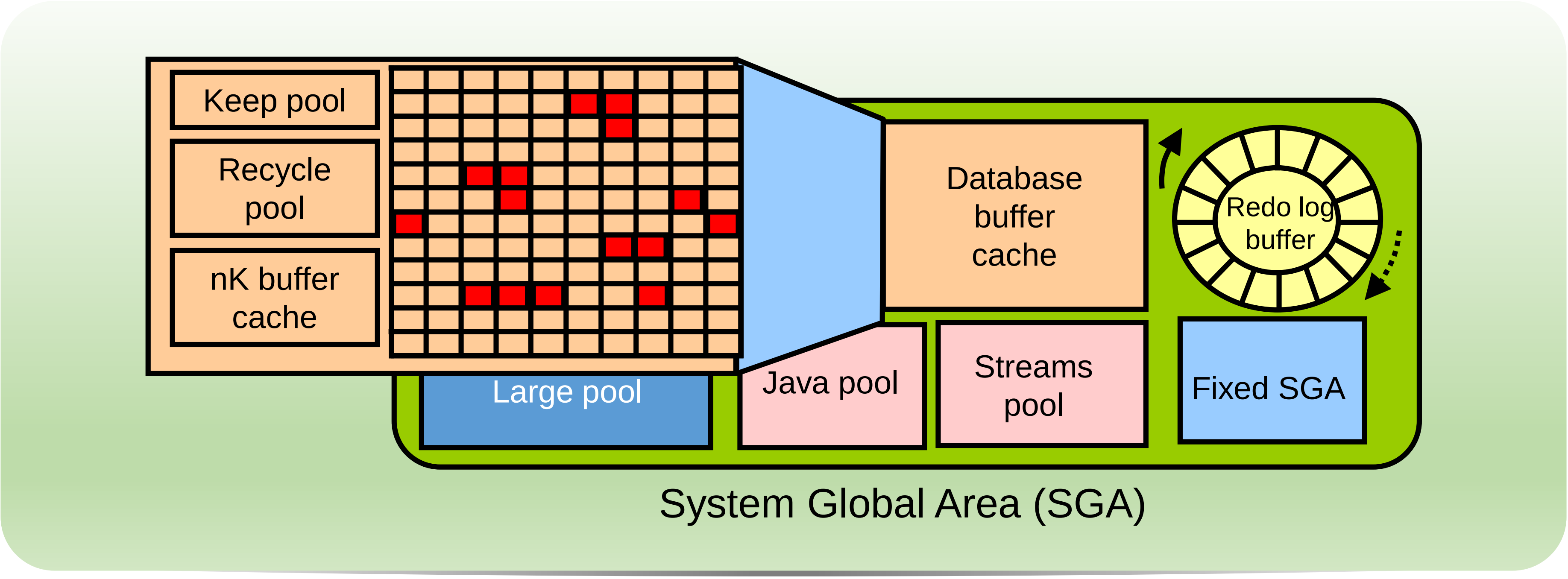
Shared Pool

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



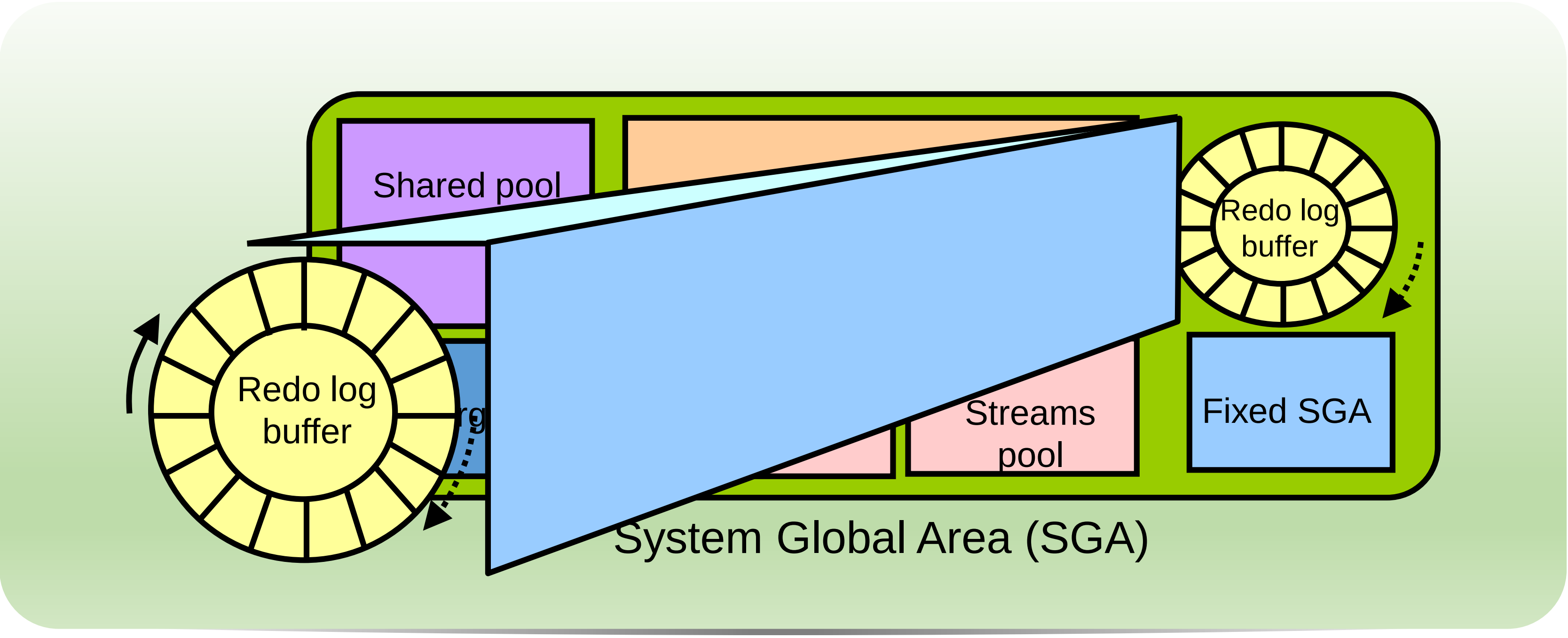
Database Buffer Cache

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



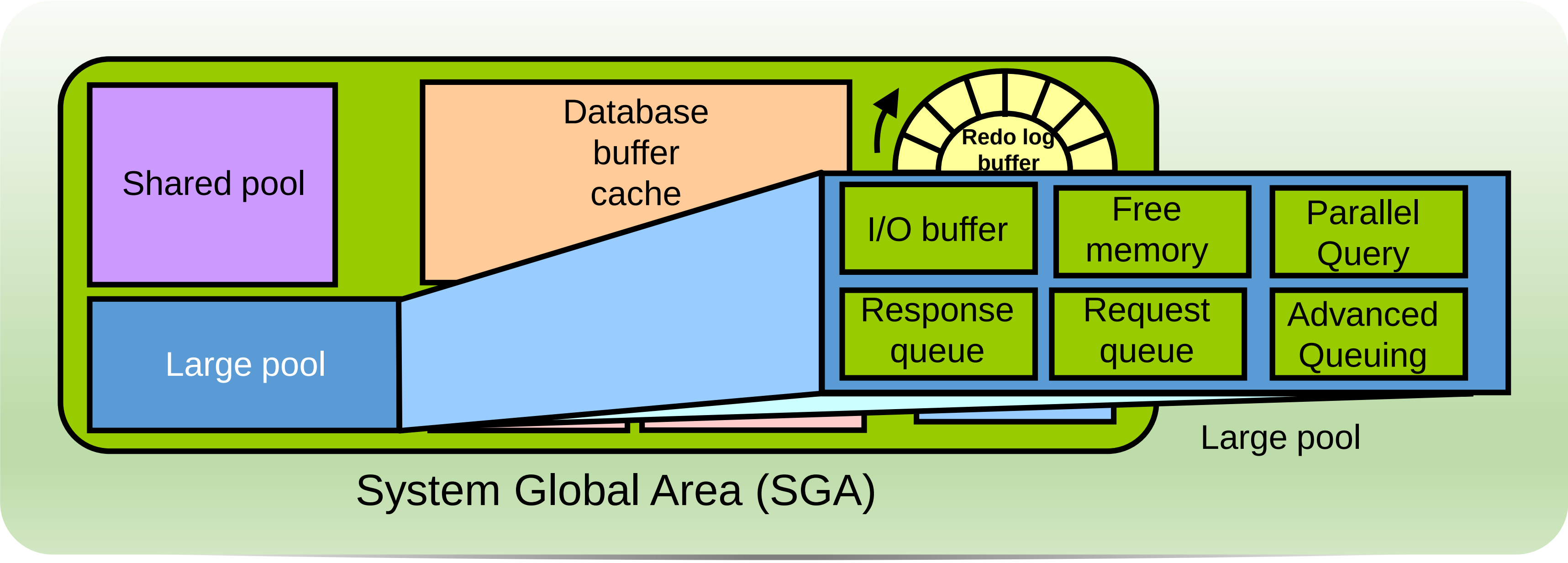
Redo Log Buffer

- 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



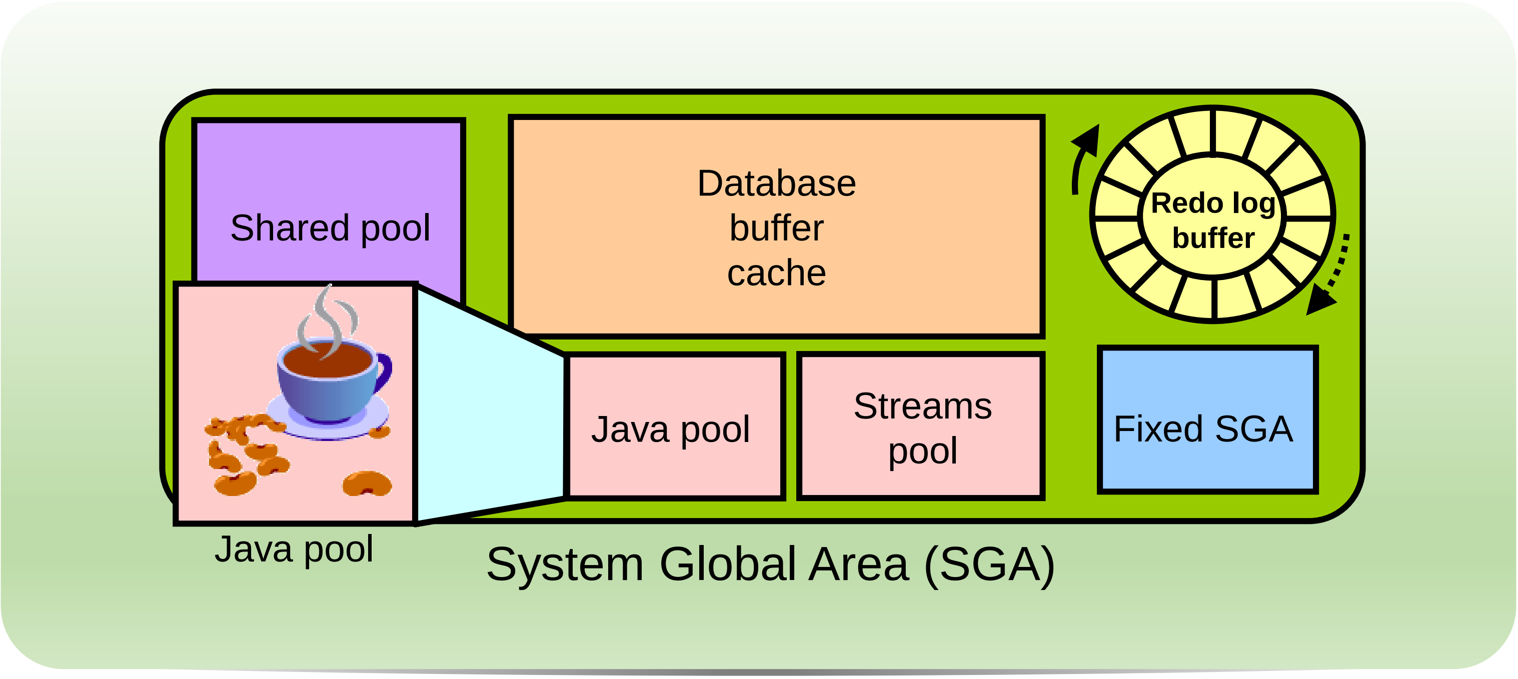
Large Pool

- 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



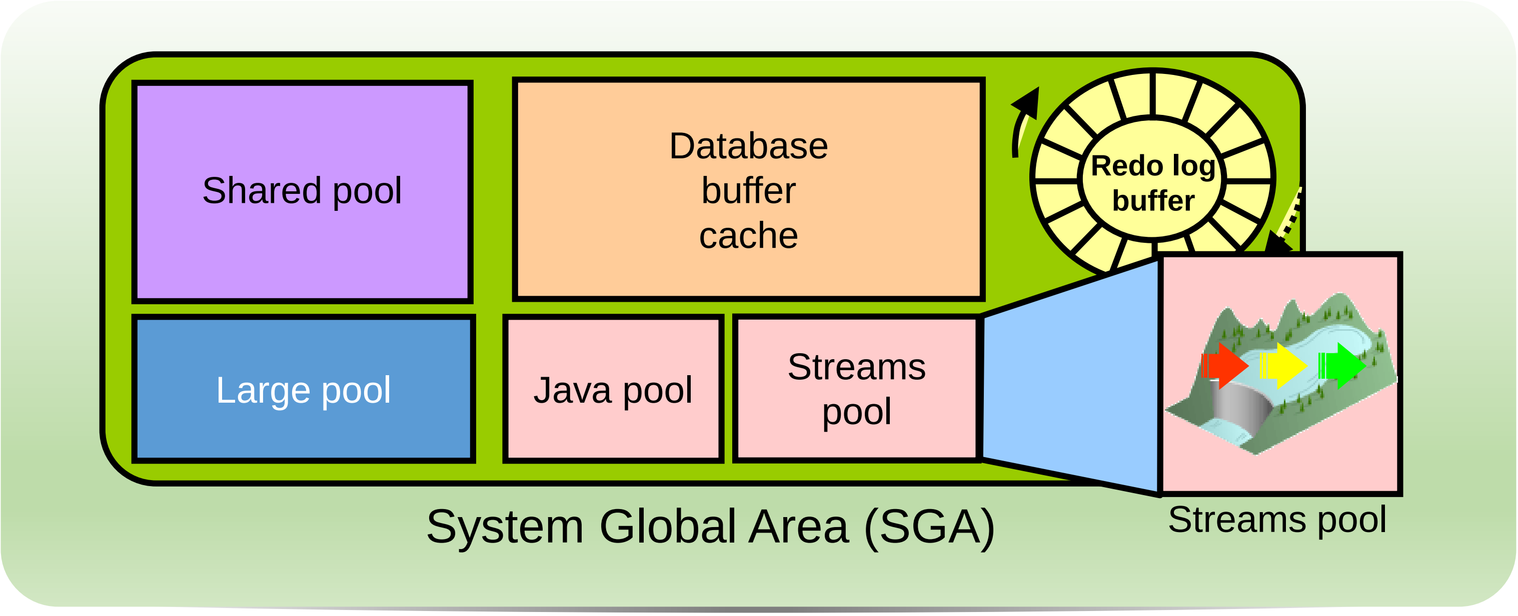
Java Pool

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

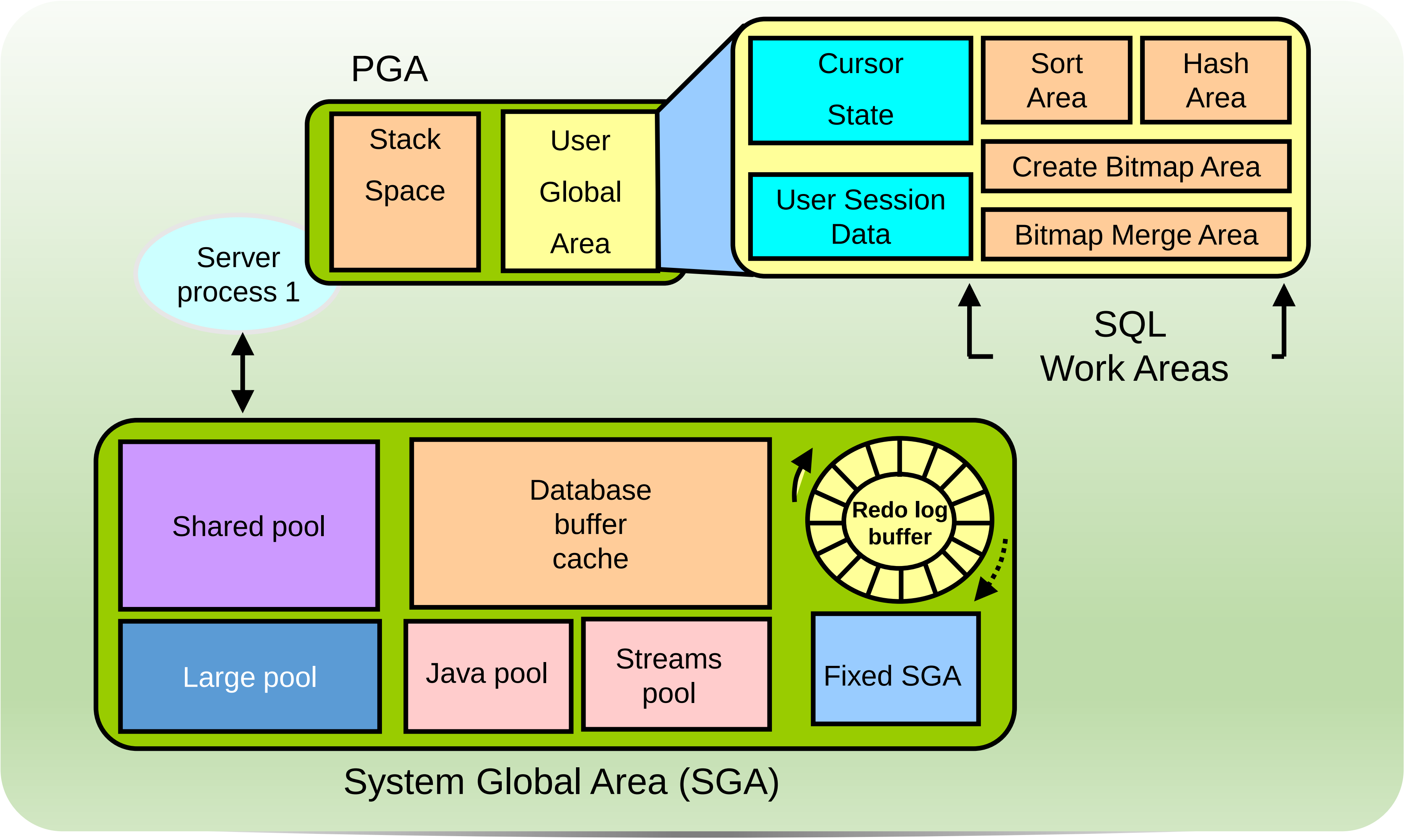


Streams Pool

- Streams pool memory is used exclusively by Oracle Streams to:
  - Store buffered queue messages
  - Provide memory for Oracle Streams processes

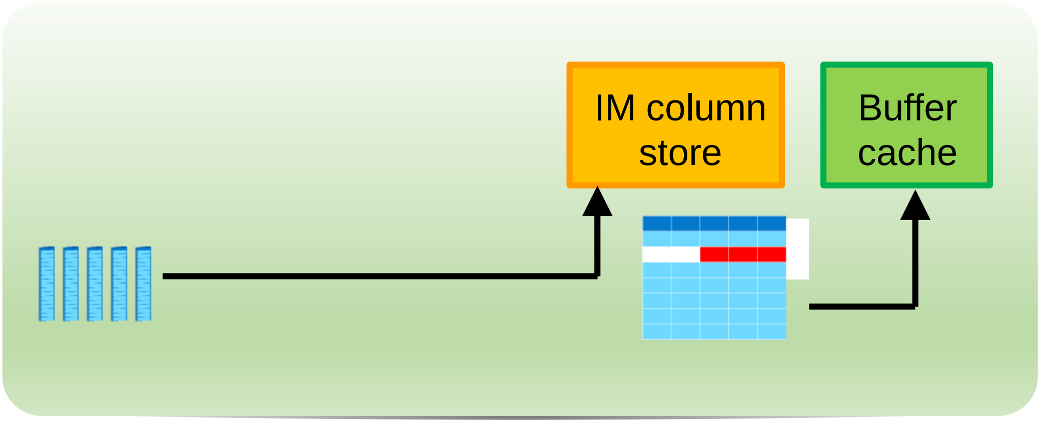


Program Global Area (PGA)



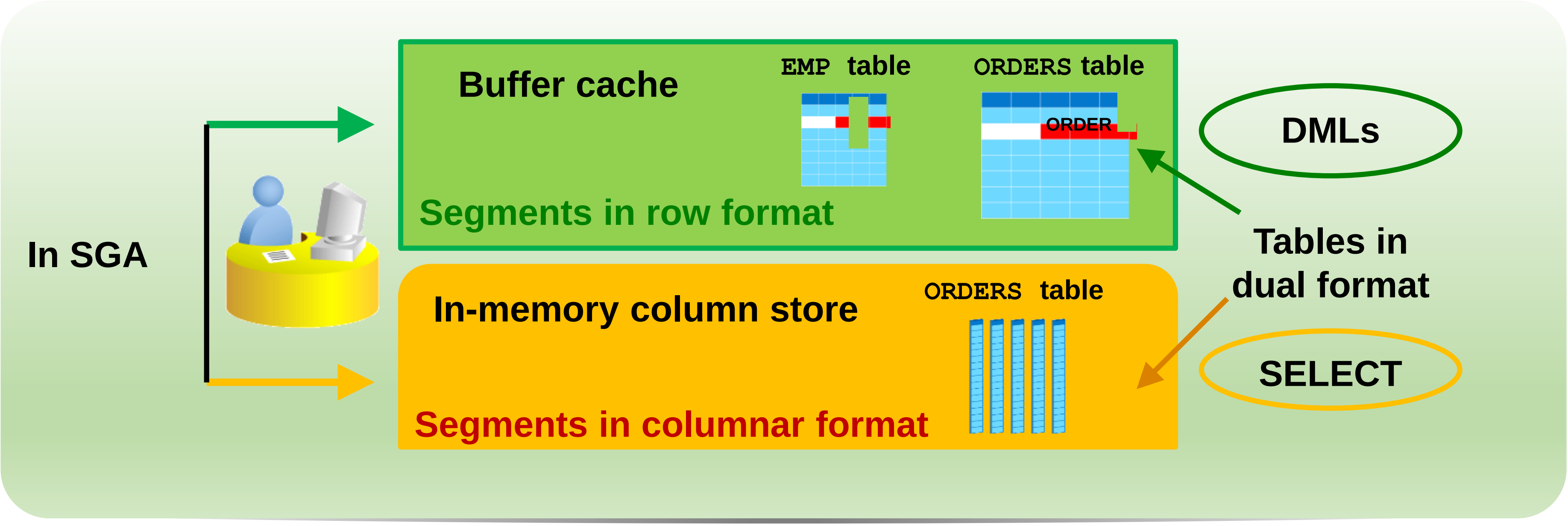
In-Memory Column Store: Introduction

- Instant query response:
  - Faster queries on very large tables on any columns (100x)
  - Use of scans, joins, and aggregates
  - Without indexes
  - Best suited for analytics: few columns, many rows
- Faster DML: Removal of most analytics indexes (3 to 4x)
- Full application transparency
- Easy setup:
  - In-memory column store configuration
  - Segment attributes



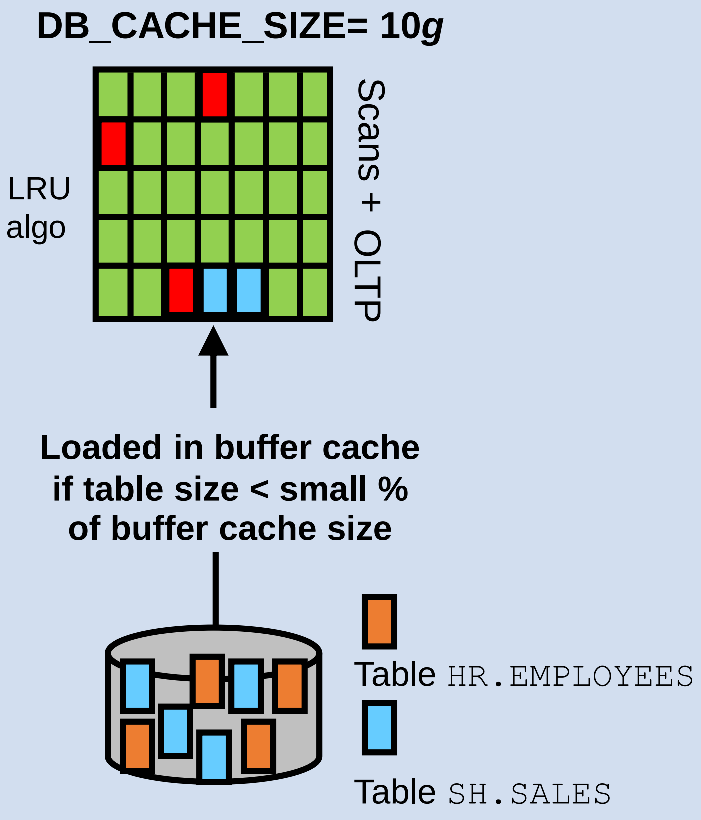
In-Memory Column Store: Overview

- A pool in the SGA: In-Memory column store
  - Segments populated into the IM column store are converted into a columnar format.
  - In-Memory segments are transactionally consistent with the buffer cache.
- Only one segment on disk and in row format



Full Database In-Memory Caching

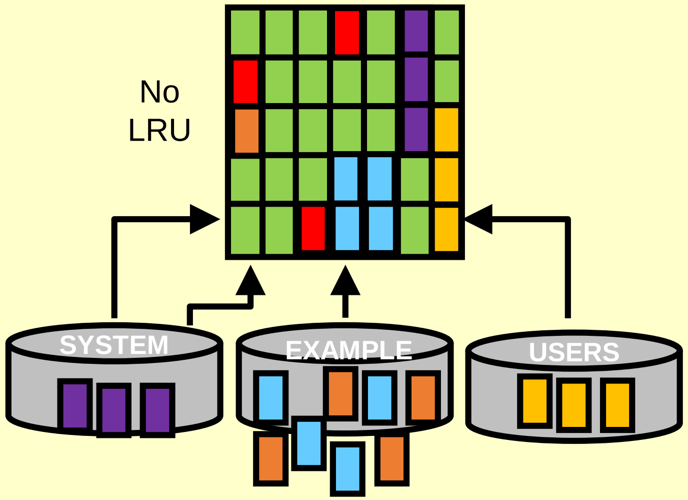
Traditional Buffer Cache Usage



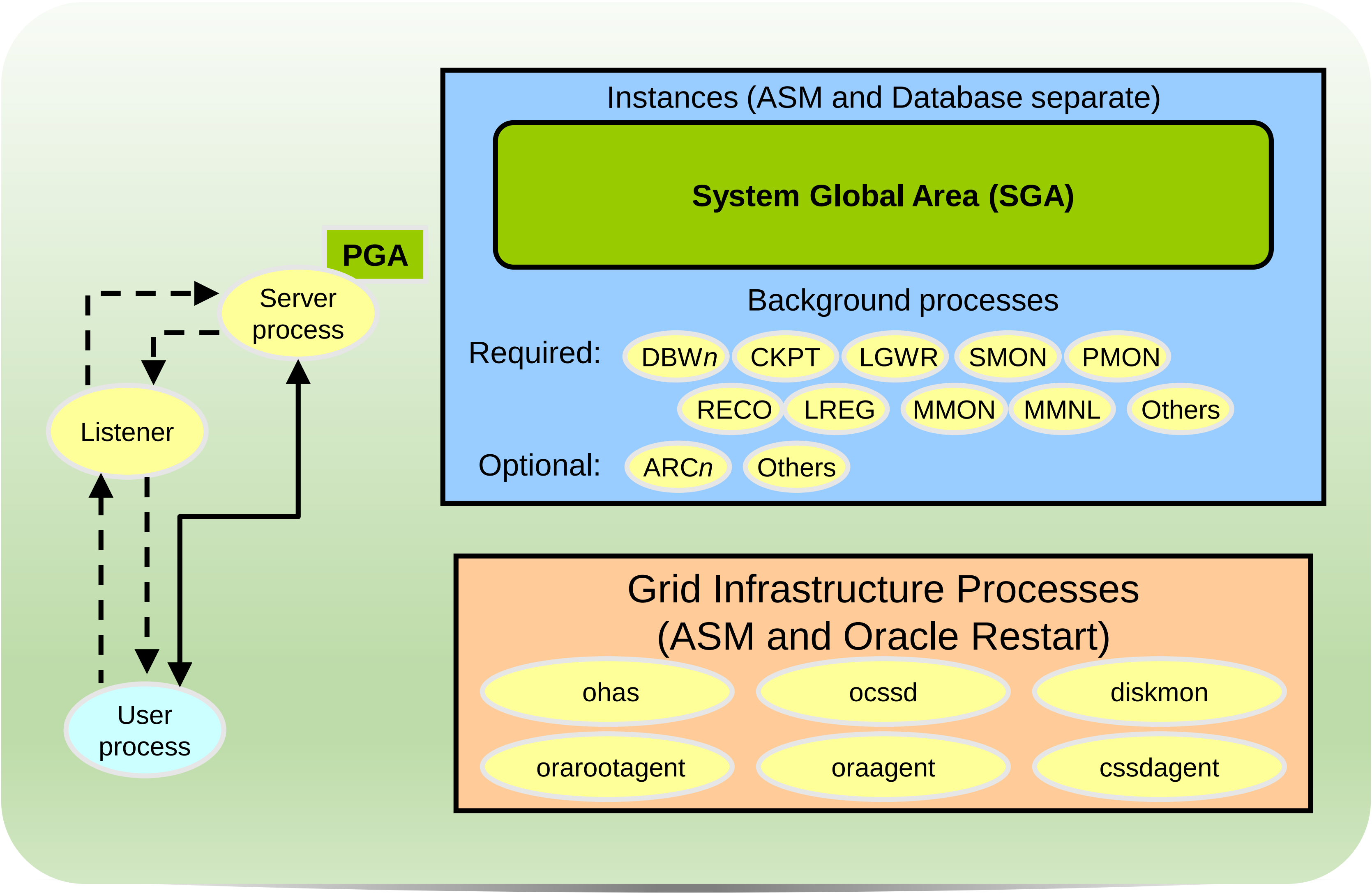
Full Database In-Memory Caching

Entire database is loaded into the buffer cache:

- Huge performance benefits
- Two modes
  - Full Database Caching
  - Force Full Database Caching

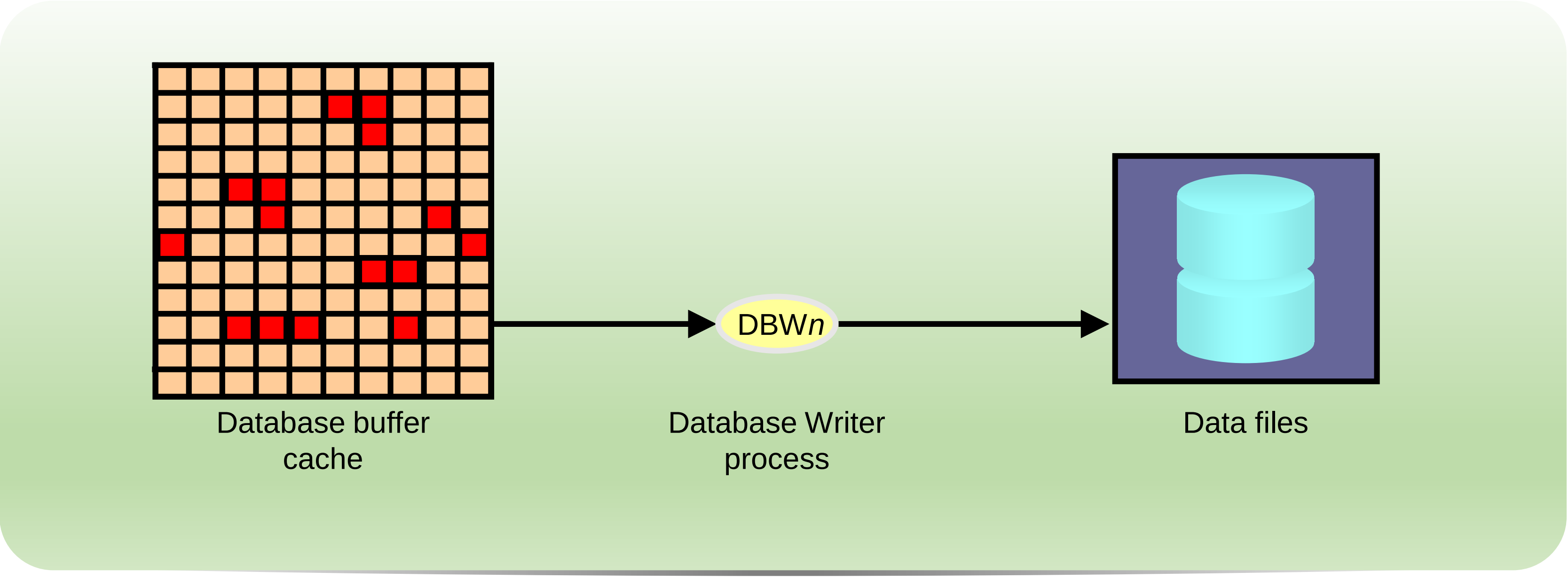


Process Structures



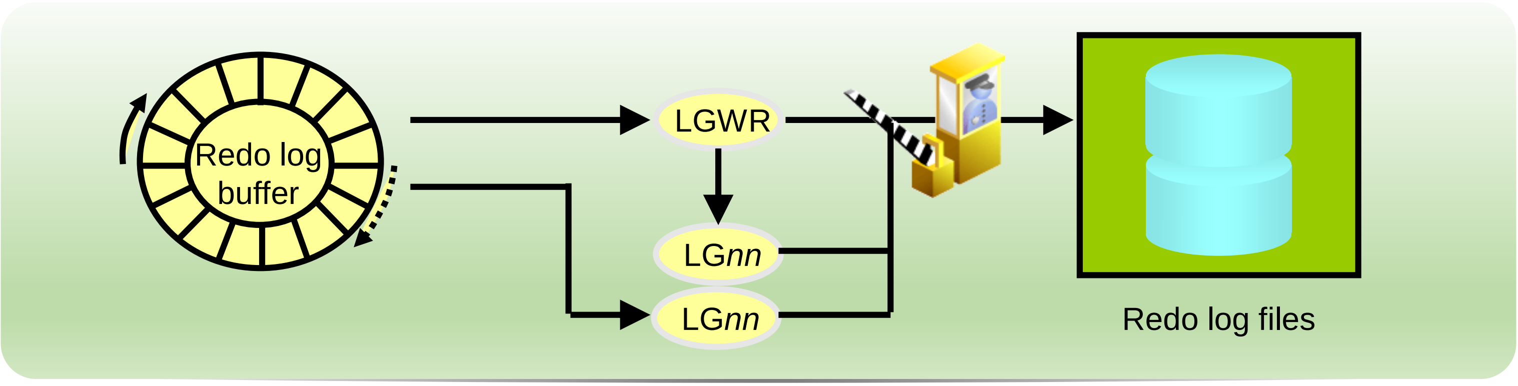
Database Writer Process (DBWn)

- Writes modified (dirty) buffers in the database buffer cache to disk:
  - Asynchronously while performing other processing
  - To advance the checkpoint



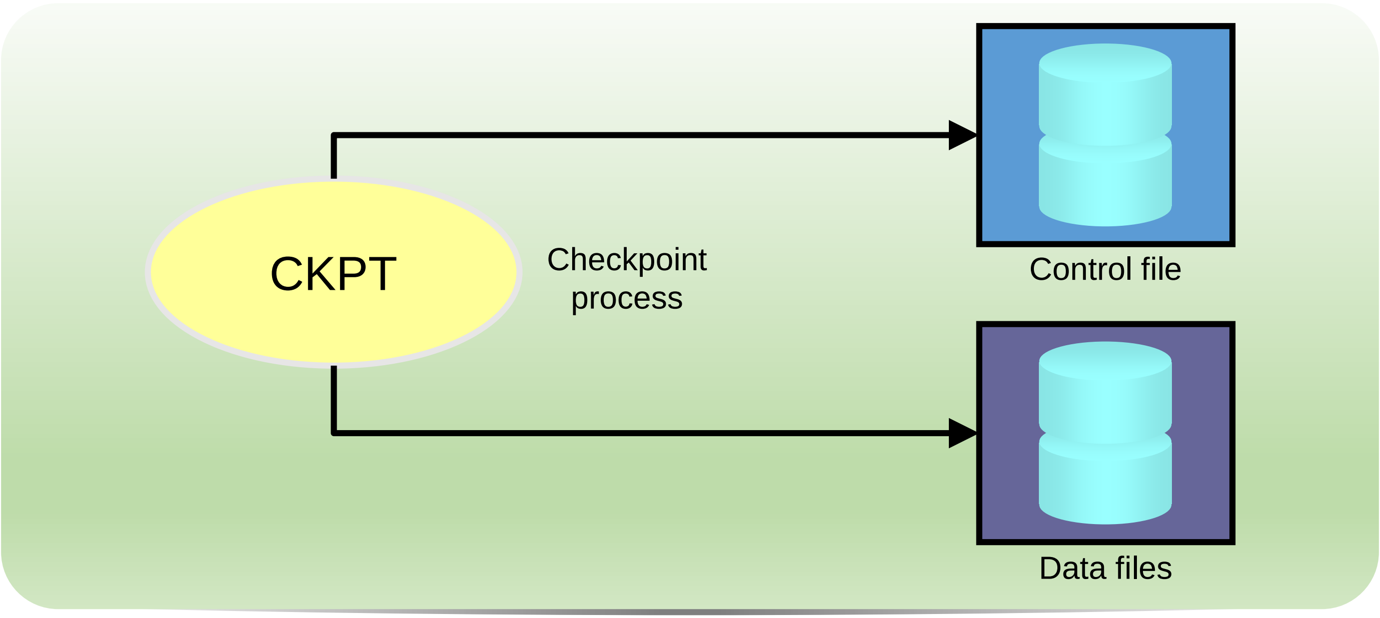
Log Writer Process (LGWR)

- 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 LG*nn* processes and ensures correct order for operations that must be ordered



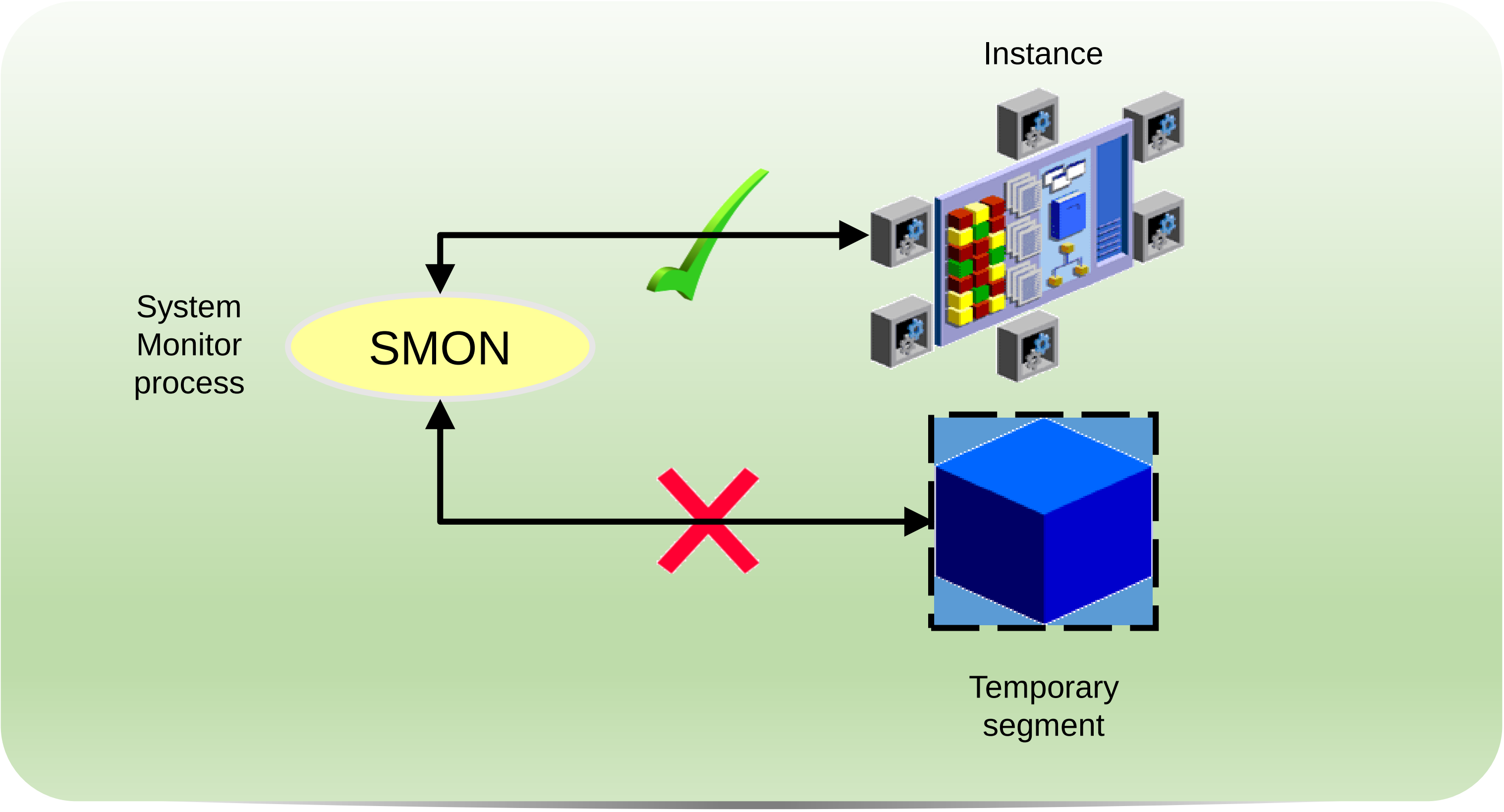
Checkpoint Process (CKPT)

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



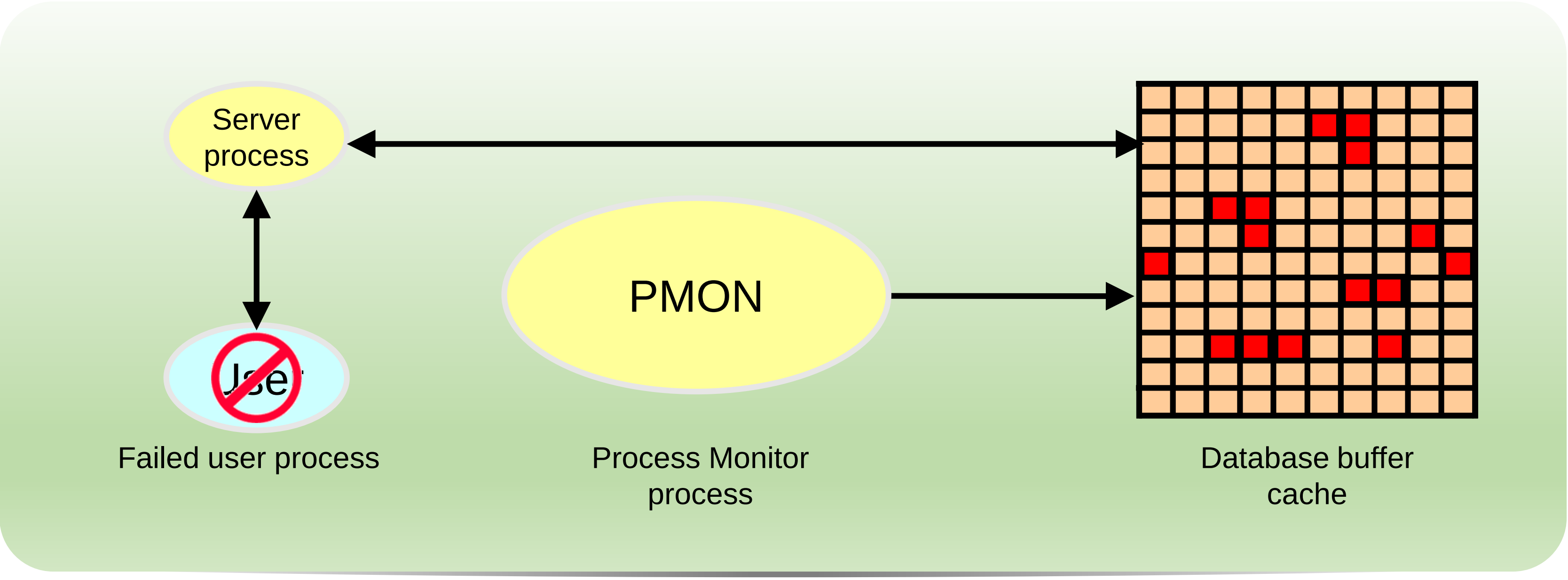
System Monitor Process (SMON)

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



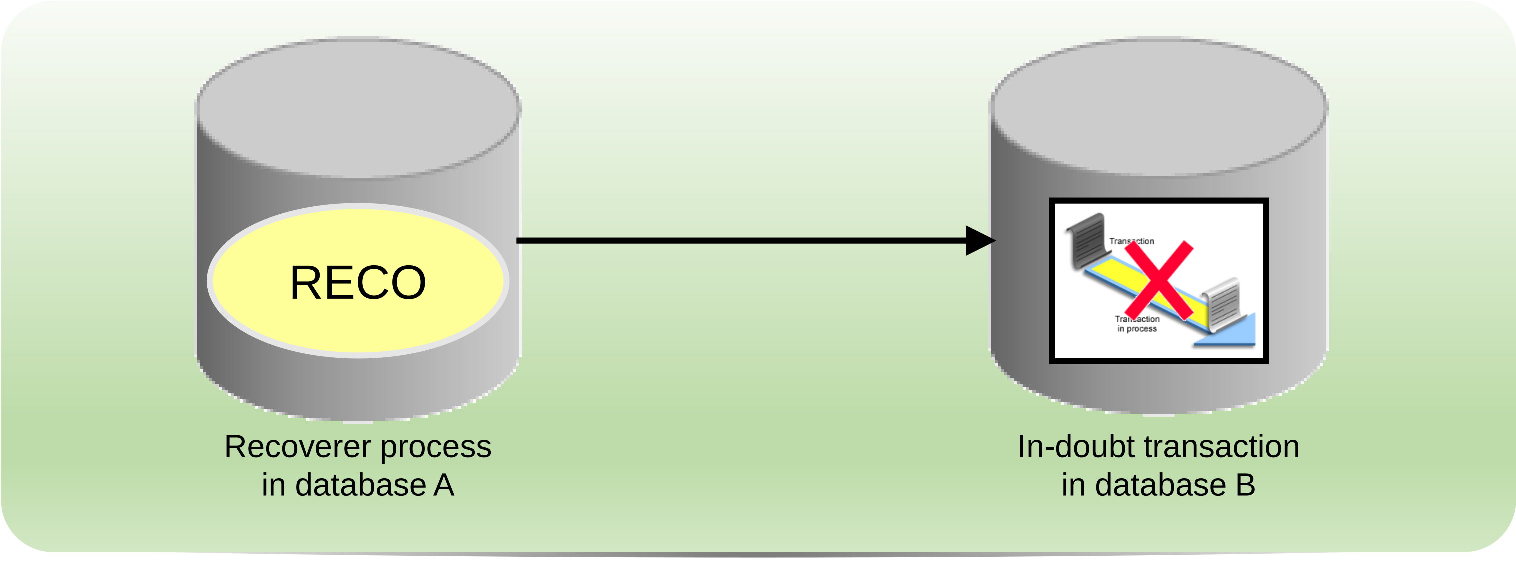
Process Monitor Process (PMON)

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



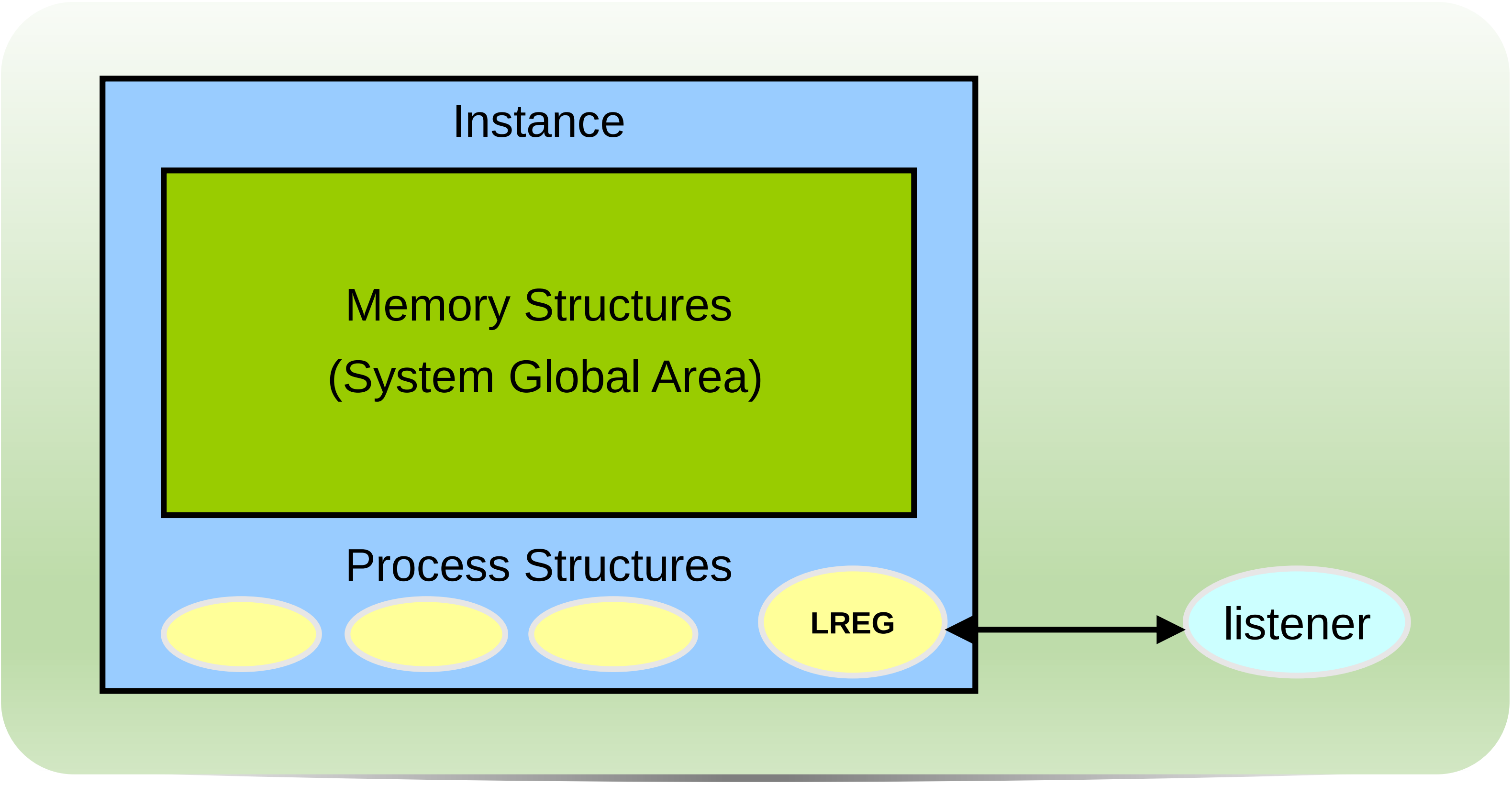
Recoverer Process (RECO)

- 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



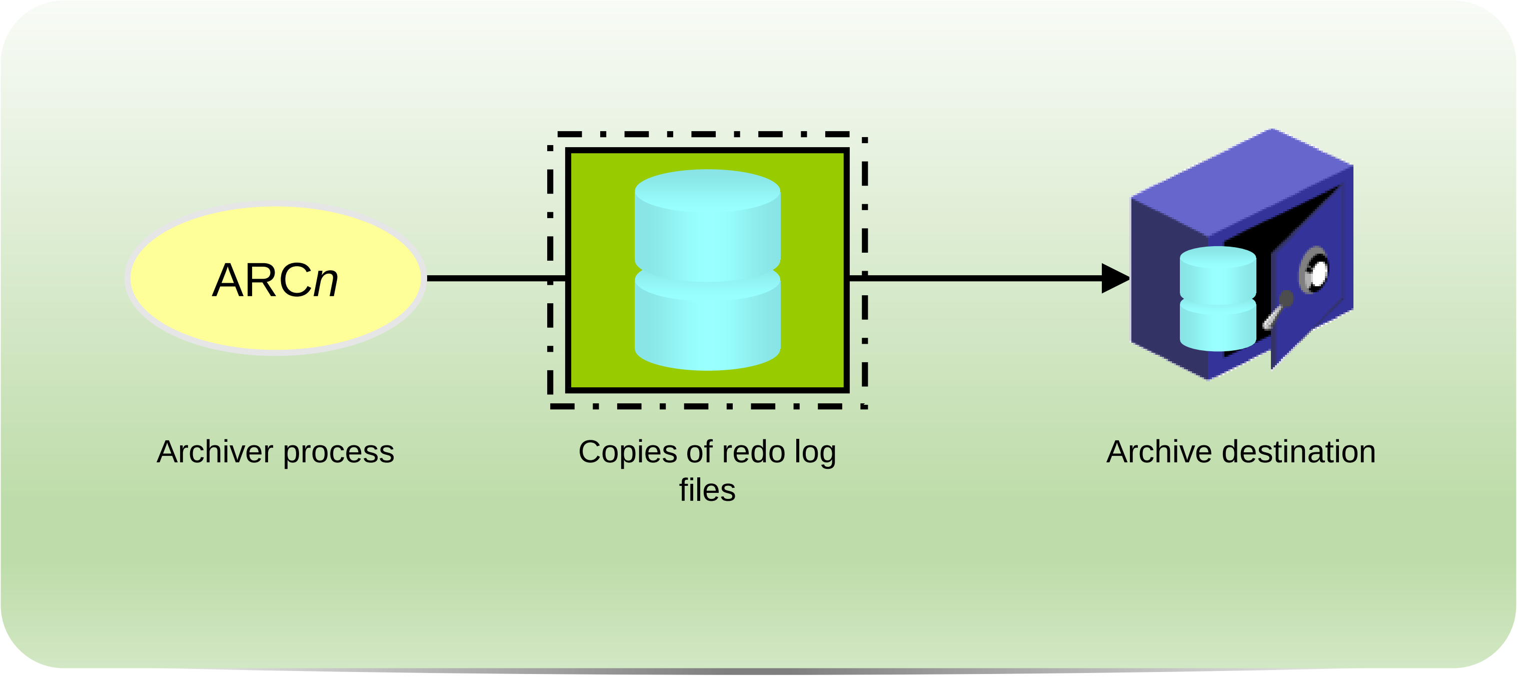
Listener Registration Process (LREG)

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

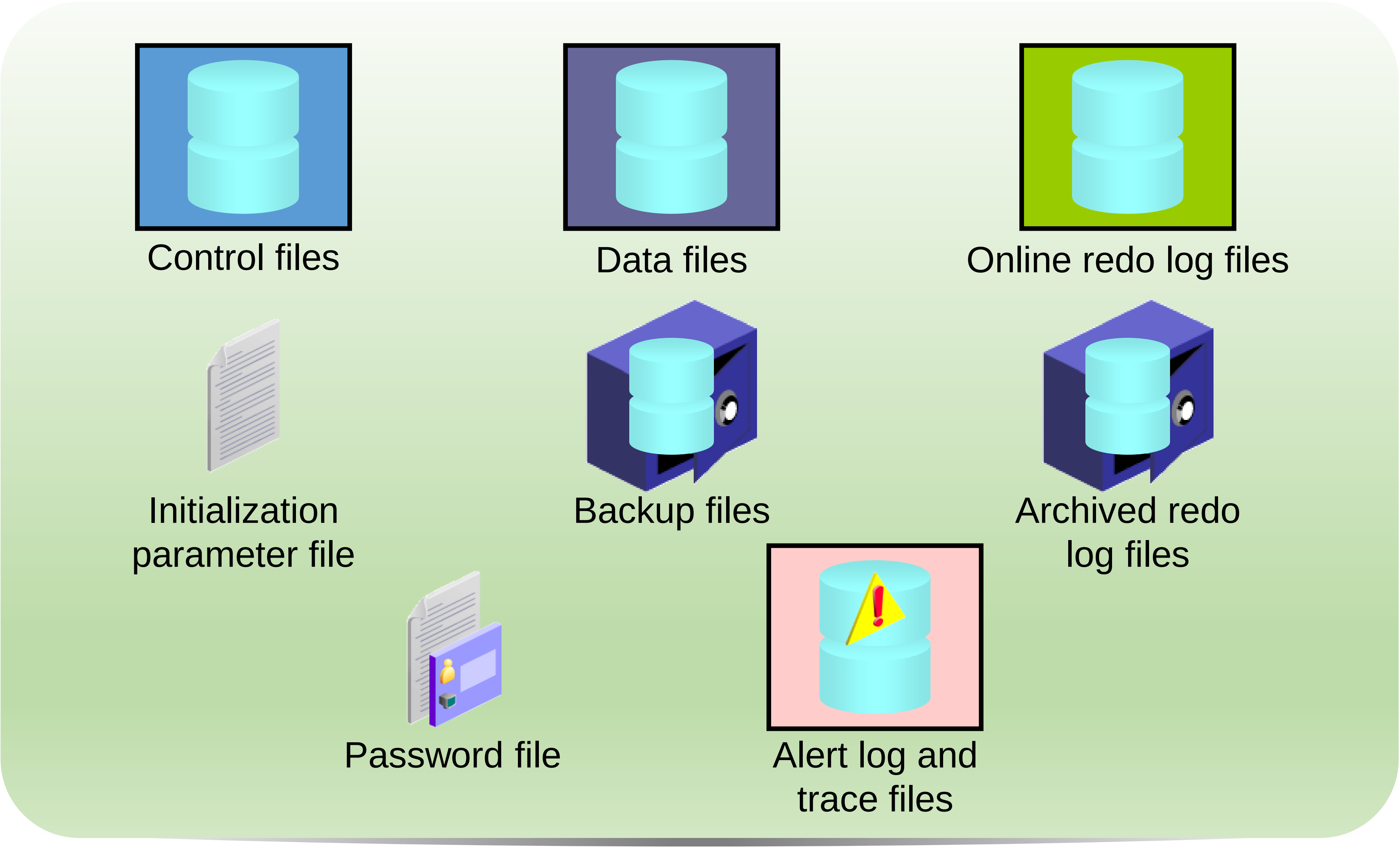


Archiver Processes (ARC*n*)

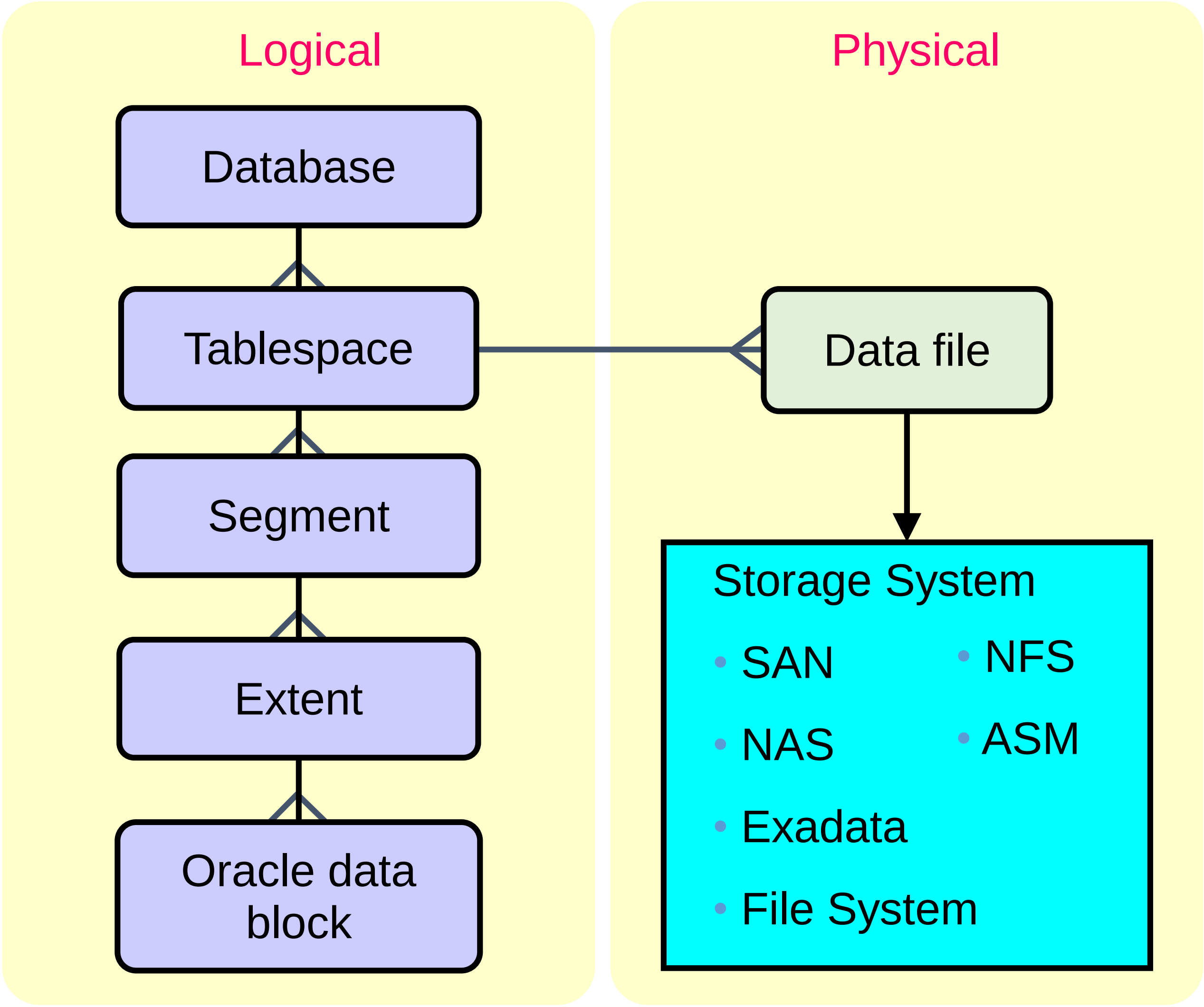
- Copy redo log files to a designated storage device after a log switch has occurred
- Can collect transaction redo data and transmit that data to standby destinations



Database Storage Architecture

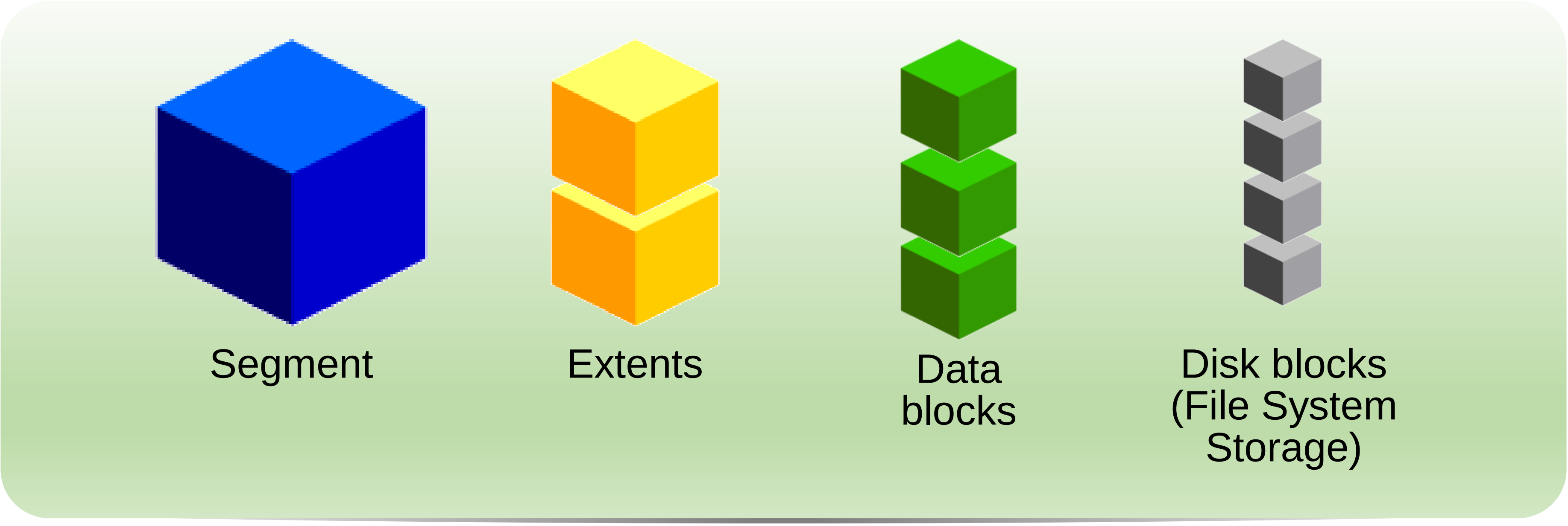


Logical and Physical Database Structures

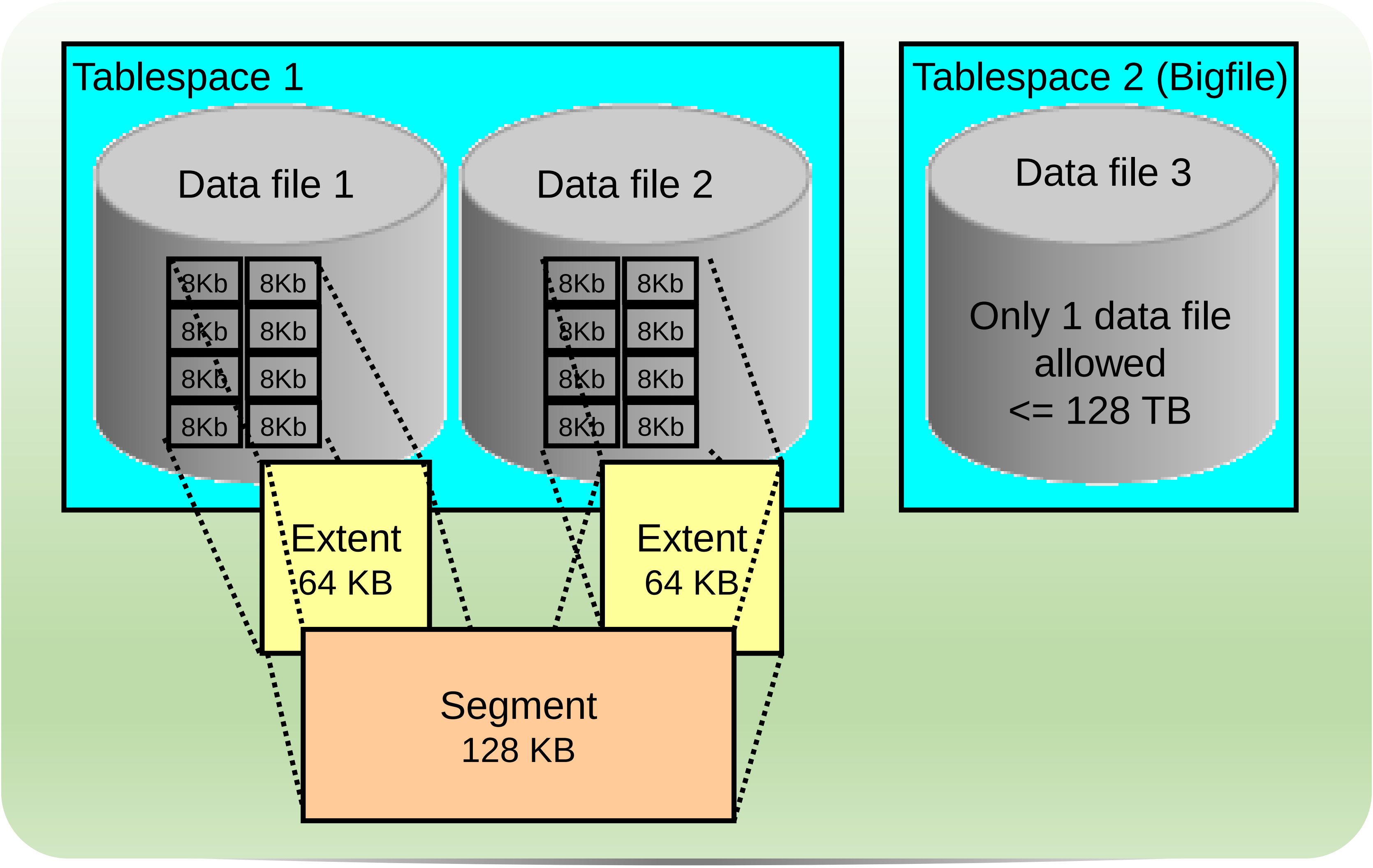


Segments, Extents, and Blocks

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



Tablespaces and Data Files



Default Tablespaces

Tablespace	Description
`SYSTEM`	The `SYSTEM` tablespace is used for core functionality.
`SYSAUX`	The `SYSAUX` tablespace is an auxiliary tablespace to the `SYSTEM` tablespace and helps reduce the load on the `SYSTEM` tablespace.
`TEMP`	The `TEMP` tablespace contains schema objects only for a session's duration.
`UNDO`	The `UNDO` tablespace stores the data needed to roll back, or undo, changes to the database.
`USERS`	The `USERS` tablespace stores user objects and data.

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.

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

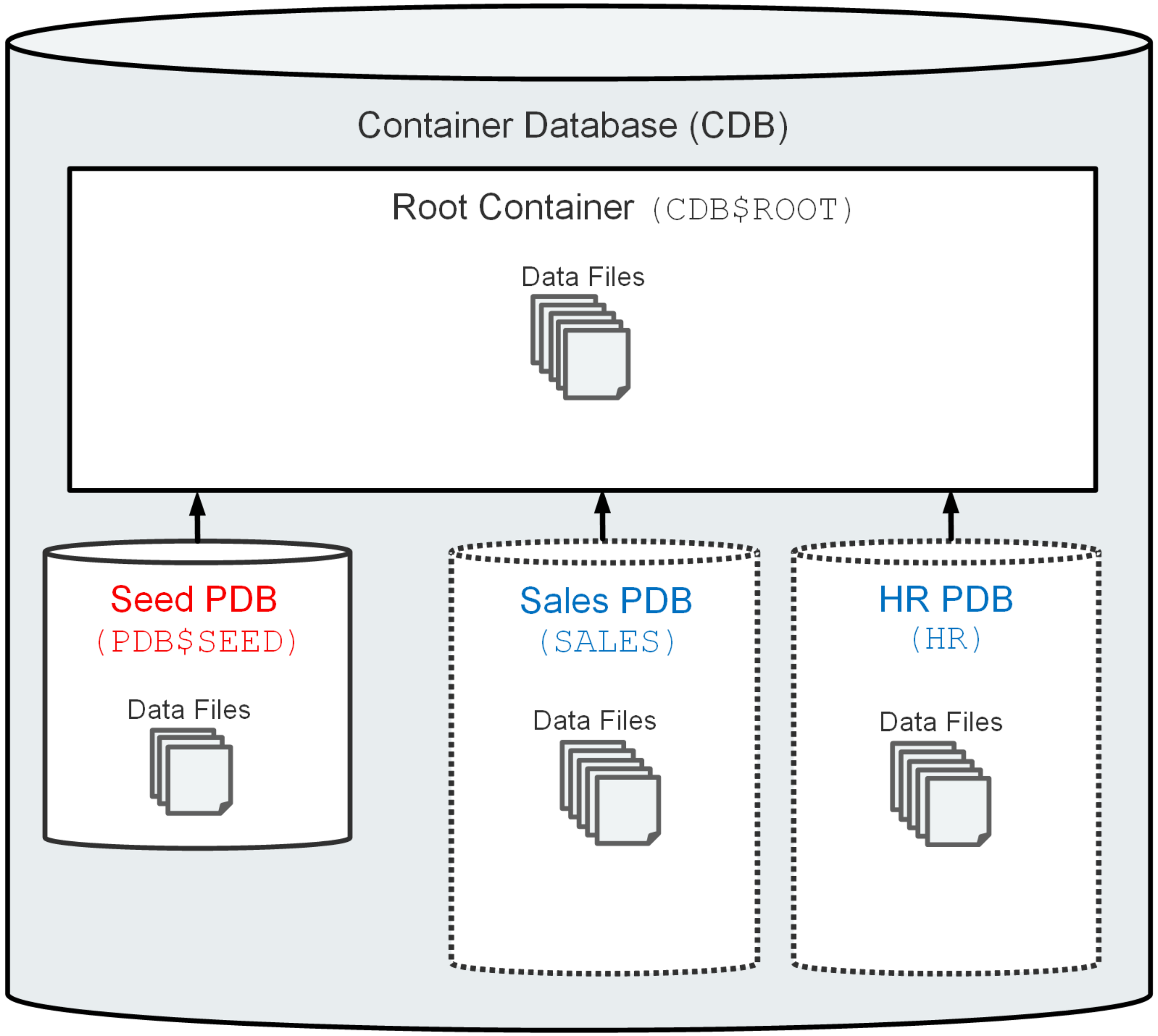
Example:

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

Oracle Container Database: Introduction

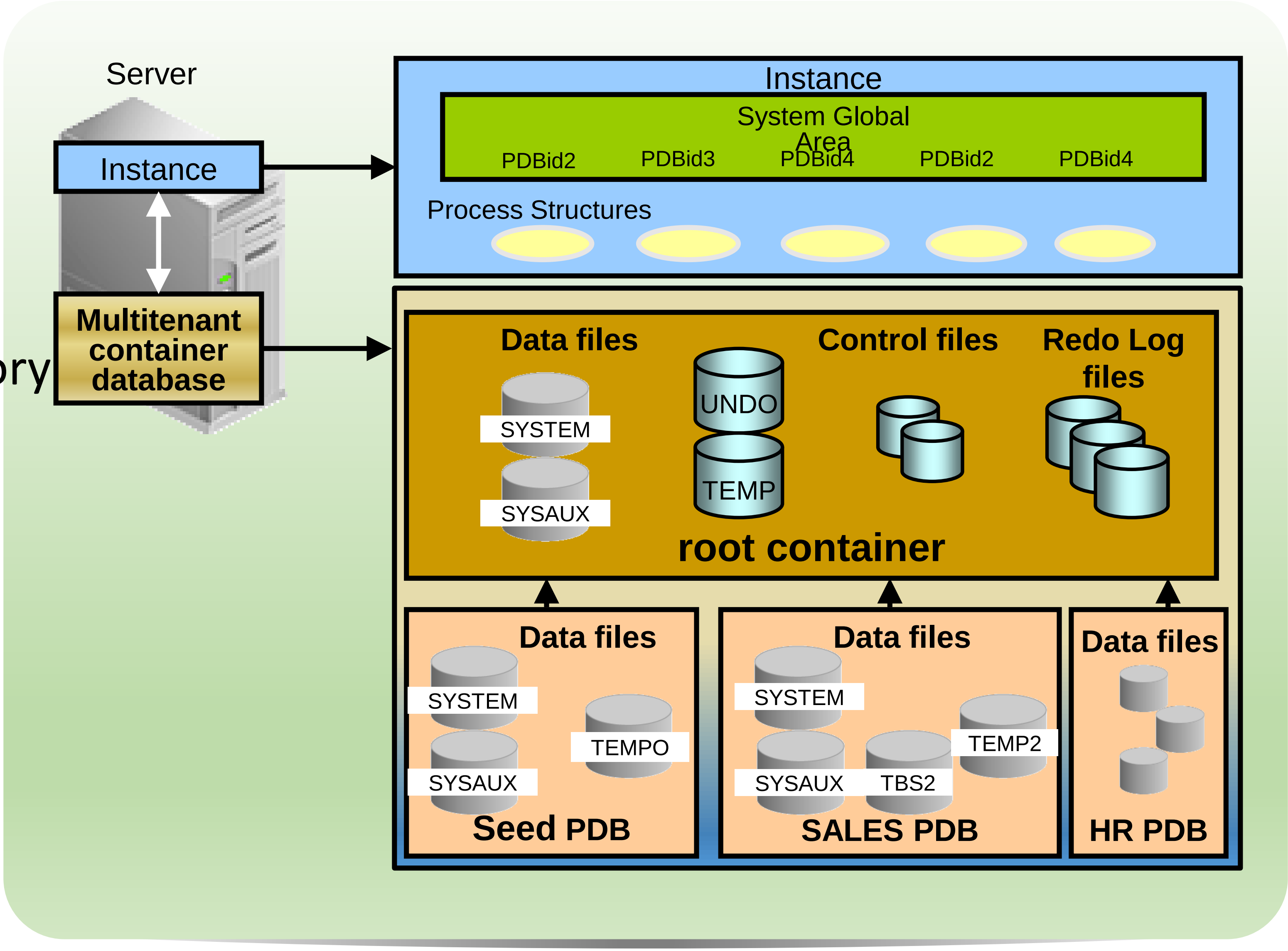
- *Pluggable database*: Is a set of database schemas that appears logically to users and applications as a separate database
- *Multitenant container database*: Has a database instance and database files at the physical level
- All pluggable databases share:
  - Background processes
  - Shared/process memory
  - Oracle metadata

Multitenant Database



Multitenant Architecture

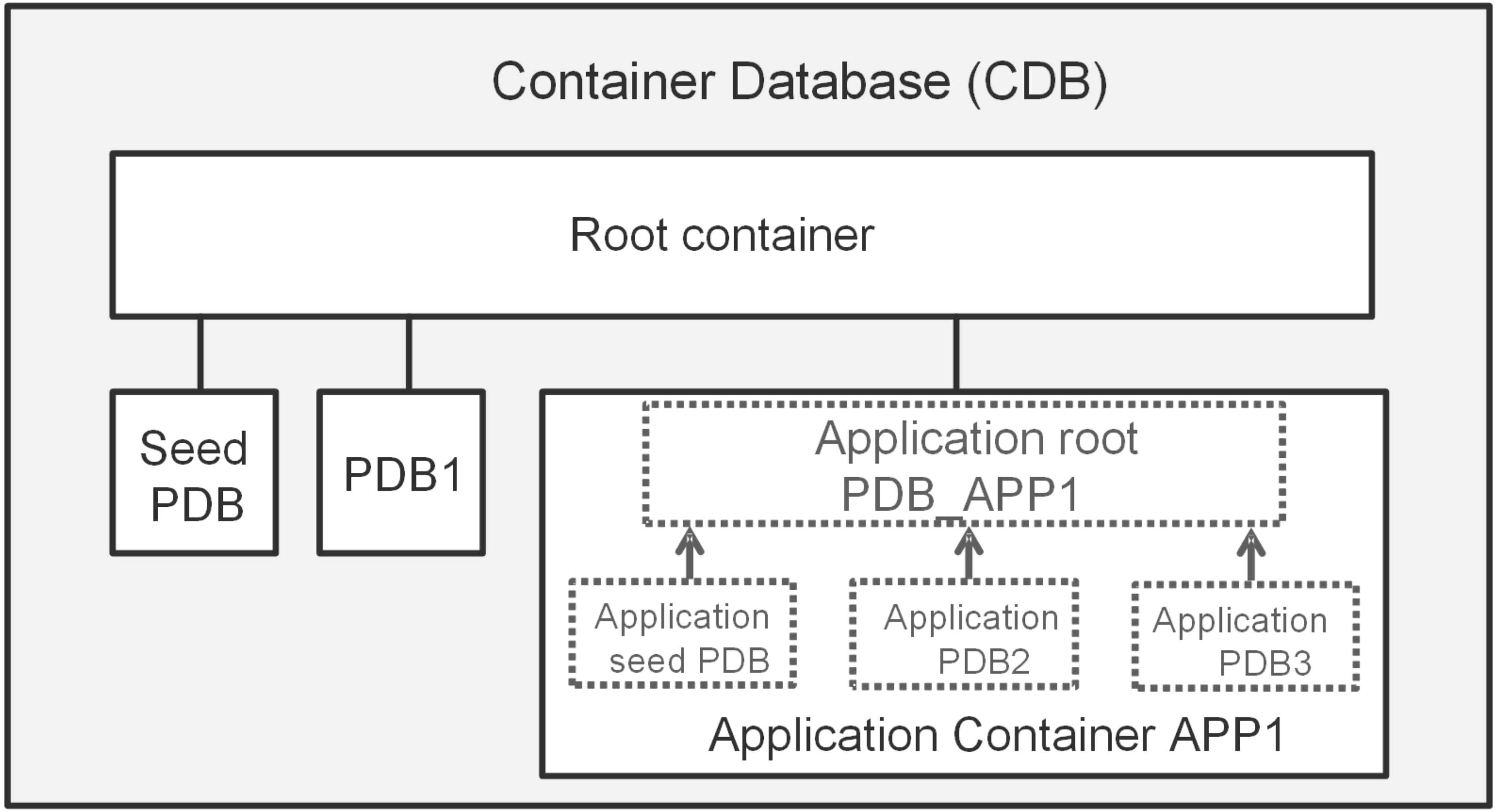
- All PDBs share:
  - Background processes
  - Shared/process memory
  - Oracle metadata
  - Redo log files
  - Control files
  - Undo tablespace



Default Tablespaces in the Multitenant Architecture

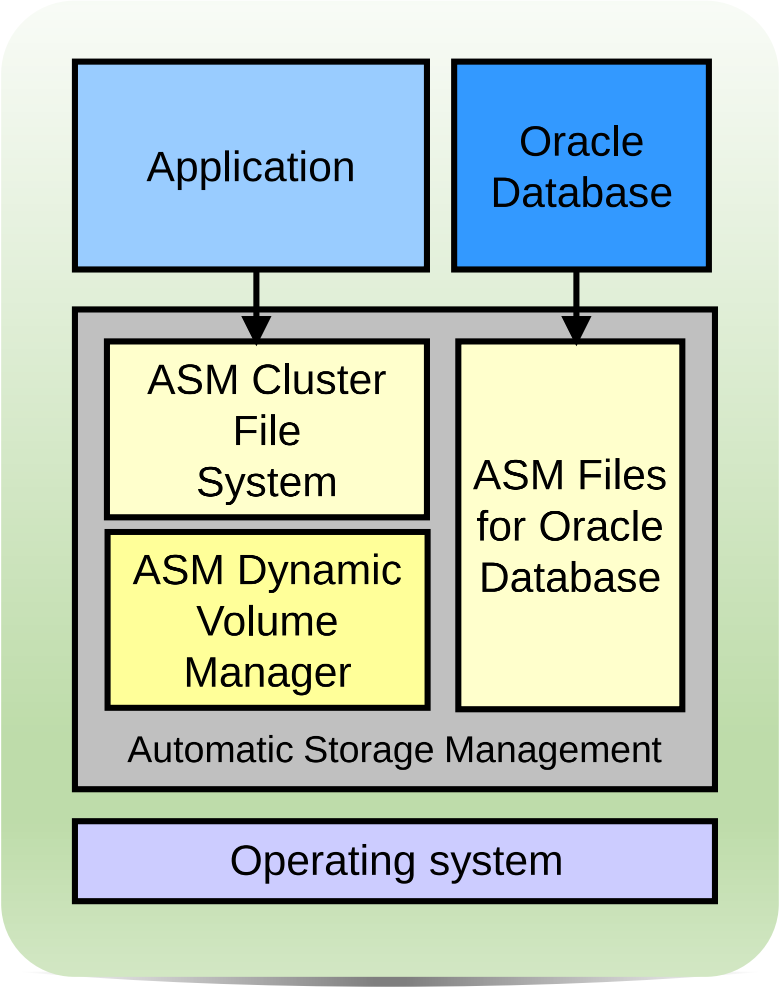
Tablespace	Description
SYSTEM	In the root container, it contains Oracle-supplied metadata. In a PDB, it contains user metadata.
SYSAUX	It exists in the root container and in each PDB.
TEMP	By default, the root container has a single default temporary tablespace that every PDB uses. You can create separate temporary tablespaces in PDBs.
UNDO	One active undo tablespace exists in the root container. It is recommended that you have a local undo tablespace in each PDB.
USERS	The root container and each PDB have a USERS tablespace.

Application Containers

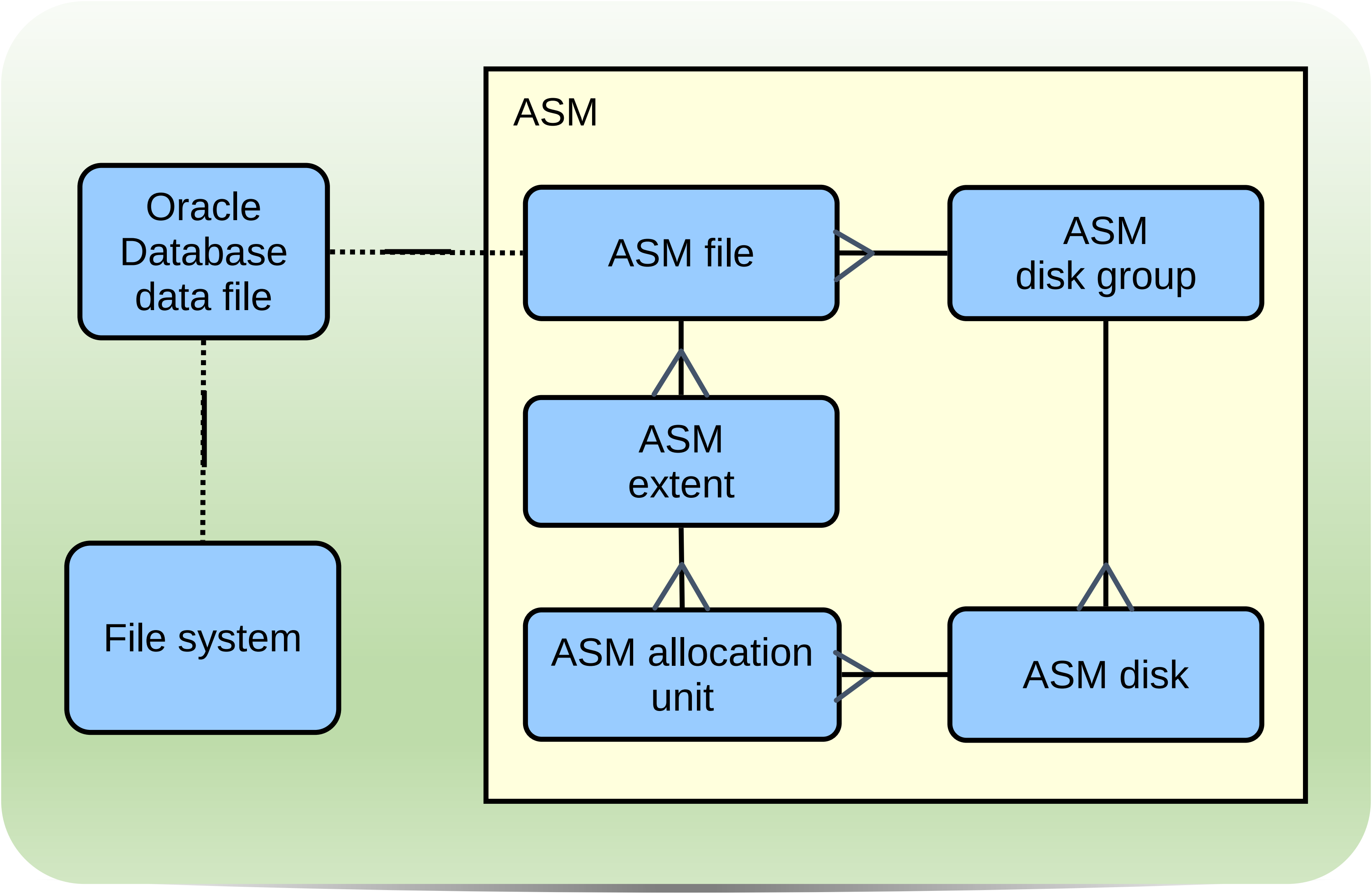


Automatic Storage Management

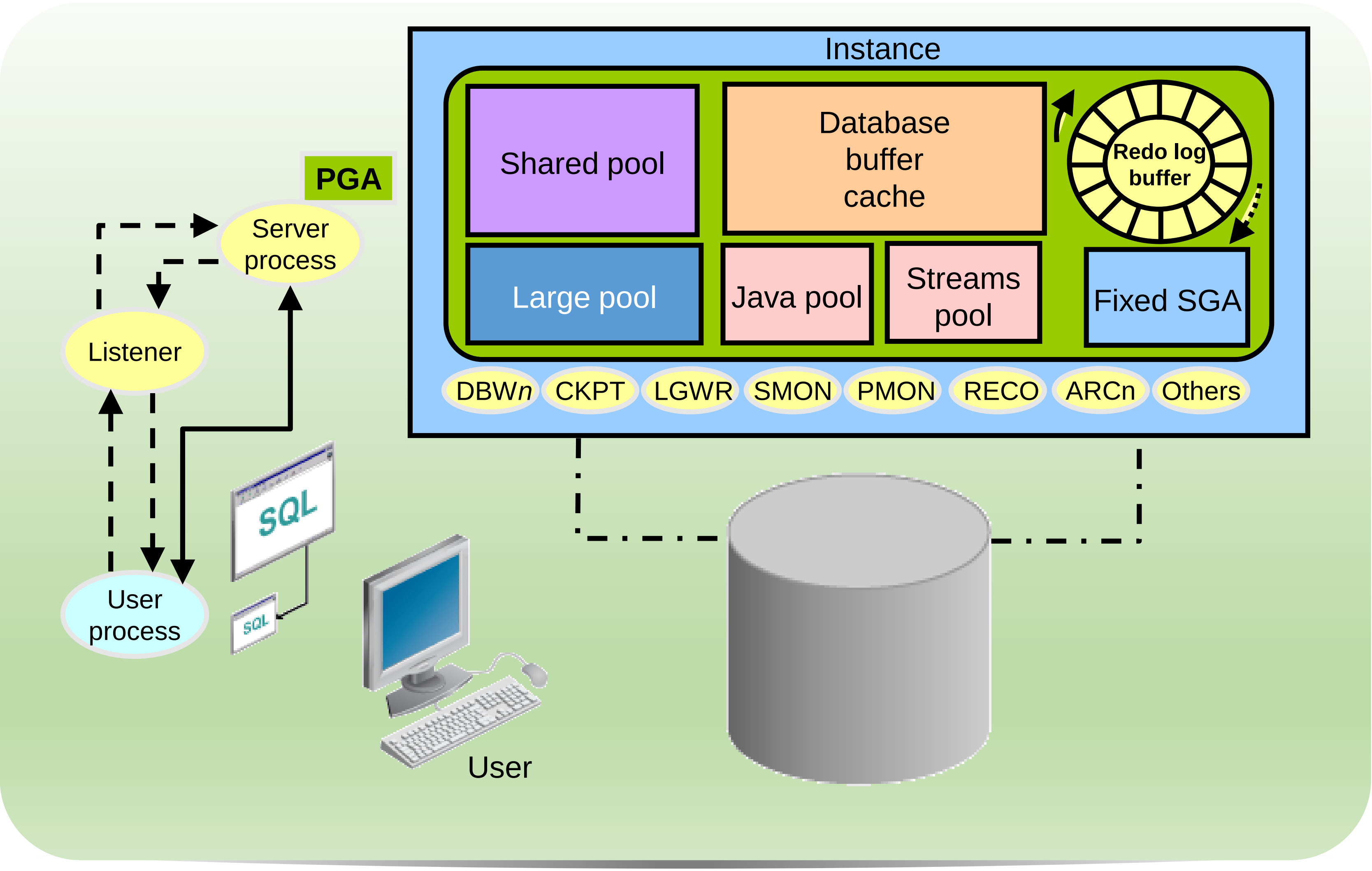
- Is a portable and high-performance cluster file system
- Manages Oracle database files
- Manages application files with ASM Cluster File System (ACFS)
- Spreads data across disks to balance load
- Mirrors data in case of failures
- Solves storage management challenges



ASM Storage Components



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



Summary

- In this lesson, you should have learned how to:
  - List the major architectural components of Oracle Database
  - Explain memory structures
  - Describe background processes
  - Correlate logical and physical storage structures
  - Describe multitenant architecture

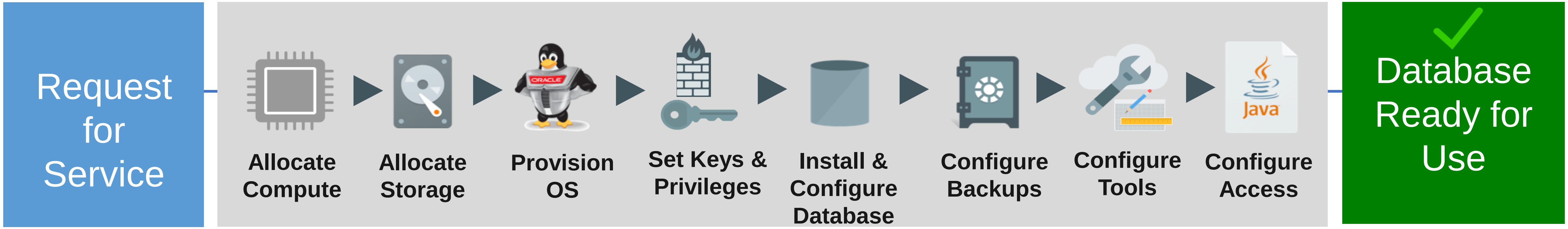
Introduction to Oracle Database Cloud Service

Creating DBCS Database Deployments

Objectives

- After completing this lesson, you should be able to:
  - Create a DBCS database deployment
  - Describe how an SSH key pair is used in authentication
  - Explain how storage volumes are allocated for a DBCS database deployment

Automated Database Provisioning

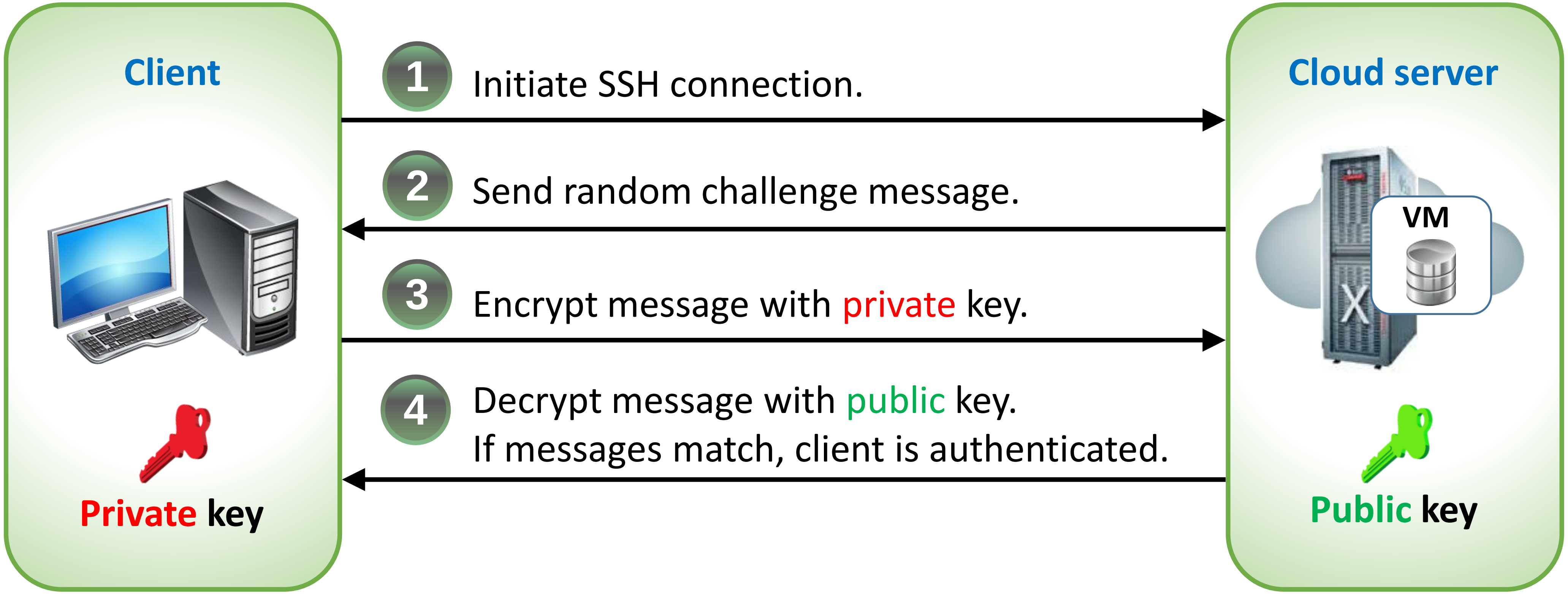


Automated provisioning based on input to the Create Oracle Database Cloud Service Instance wizard or through REST APIs

Creating a Database Deployment

- The wizard takes you through the database deployment creation, prompting for the following information:
  - Subscription: Service level, billing frequency, software release, and software edition
  - Service: Service name, compute shape, time zone, and SSH key
  - Database Configuration: Storage, database name (SID), PDB name, character set, and password for administrative accounts
  - Backup and Recovery Configuration: Backup destination and cloud storage information, if applicable
  - Additional Options: Real Application Clusters (RAC), Data Guard, and GoldenGate

How SSH Key Pairs Are Used



Creating an SSH Key Pair

- Perform one of the following:
  - Generate an SSH key pair as part of the database deployment creation.
  - Create an SSH key pair before creating your database deployment.

1

Local PC

OPTIONAL: Generate and store a private and public key pair on a local PC by using the PuTTY keygen utility or on Linux.

2

Cloud server

Create the Database Cloud Service Database deployment, developing the keys as part of the deployment creation, or provide the pre-created public key.

Storage Used for Database Files

Storage Volume	Description
`bits`	30 GB volume completely allocated to `/u01` on the compute node.
`boot`	21 GB volume allocated to the following file system mounts on the virtual machine: `/` (root), `/boot`, and swap space.
`data`	GB size equal to the value provided in the Usable Data Storage field during the database deployment creation process, with a minimum of 11 GB. This volume is completely allocated to `/u02` on the compute node.
`fra`	If backups are configured, GB size equal to 1.7 times the size of the `data` volume. If backups are not configured, GB size equal to 0.1 times the size of the `data` volume, with a minimum of 7 GB. This volume is completely allocated to `/u03` on the compute node.
`redo`	10 GB volume completely allocated to `/u04` on the compute node.

Summary

- In this lesson, you should have learned how to:
  - Create a DBCS database deployment
  - Describe how an SSH key pair is used in authentication
  - Explain how storage volumes are allocated for a DBCS database deployment

Practice 4: Overview

- 4-1: Creating a Database Deployment

Objectives

- After completing this lesson, you should be able to:
  - Explain the Oracle Cloud and Database Cloud offerings
  - Describe the architecture and features of Oracle Database Cloud Service

Oracle Cloud: Overview

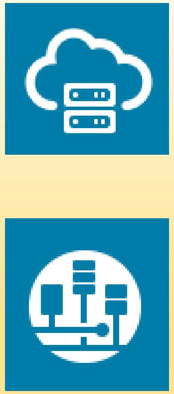
- **Oracle Cloud Software as a Service (SaaS)**
  - Provides best-of-breed cloud software in a complete, secure, and connected cloud suite
  - Comes embedded with modern best practice processes and built-in social, mobile, and analytic capabilities
- **Oracle Cloud Platform as a Service (PaaS)**
  - Enables enterprise IT and independent software vendor (ISV) developers to rapidly build and deploy rich applications
  - Uses an enterprise-grade cloud platform based on the industry's best database and application server
- **Oracle Cloud Infrastructure as a Service (IaaS)**
  - Is a set of core infrastructure capabilities such as elastic compute and storage
  - Provides customers with the ability to run any workload in the cloud

Database Cloud Service Offerings

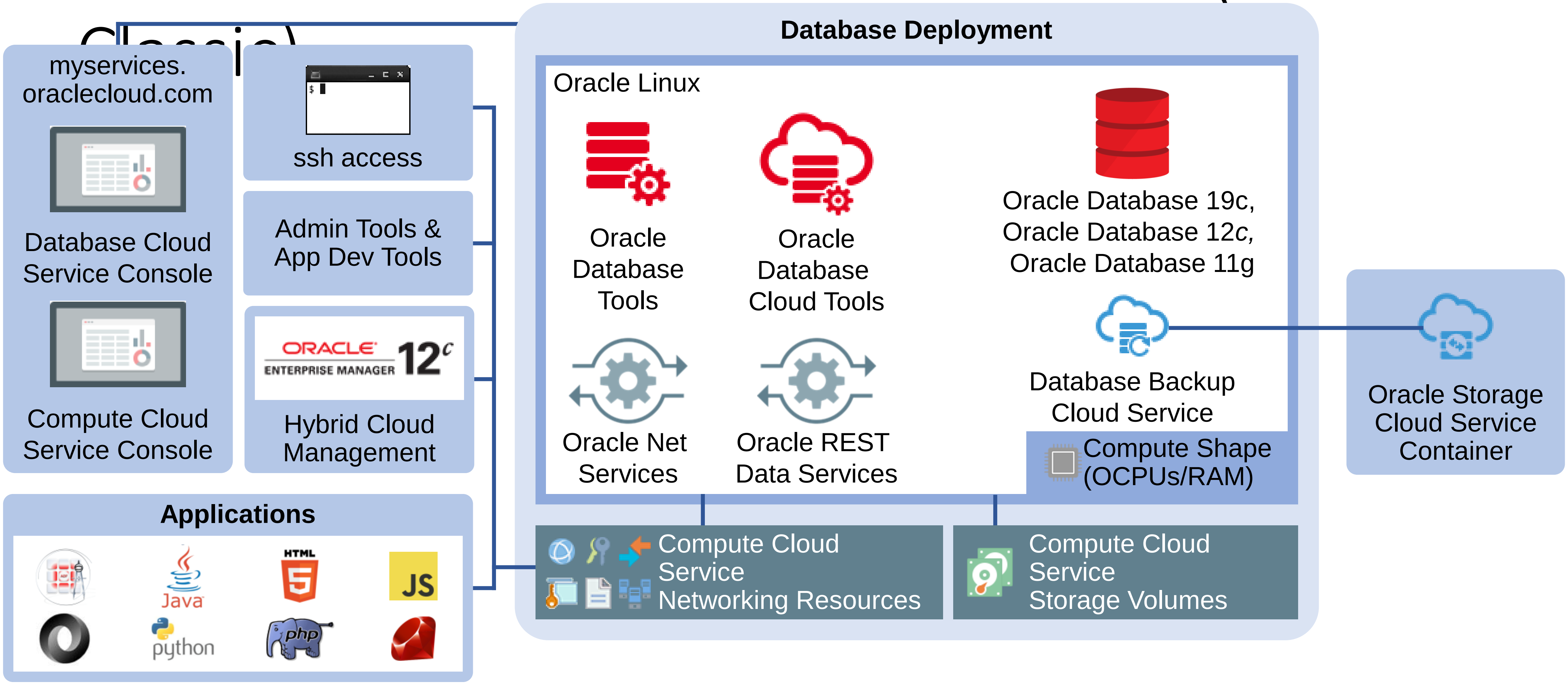
- Oracle Database Cloud Service
  - Offers elastic database services for application development, testing, and production deployment
  - Has an easy-to-use web console user interface and RESTful API to provision and administer Oracle Database on Oracle Compute Cloud
- Oracle Database Exadata Cloud Service
  - Includes all the benefits of Exadata performance
  - Offers 100% compatibility with existing business-critical applications and databases that are on premises
- Oracle Database Exadata Express Cloud Service—Managed
  - Provides a full Oracle Database experience for small- and medium-sized databases
  - Has service that is fully managed by Oracle

Infrastructure for Oracle Database Cloud Service

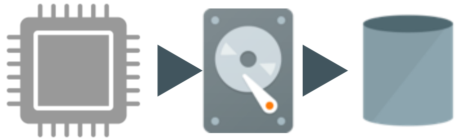
- Oracle Cloud Infrastructure Classic (OCI Classic)
  - Oracle's first-generation cloud infrastructure
  - Includes Compute Classic and Storage Classic
- Oracle Cloud Infrastructure (OCI)
  - Oracle's second-generation cloud infrastructure
  - Includes Compute and Storage



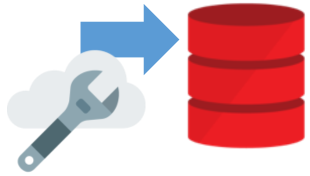
Database Cloud Service Architecture (OCI Classic)



Features and Tooling



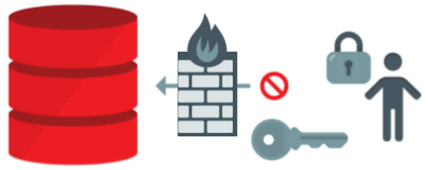
Simple Provisioning



Automated Patching

Automated Backups

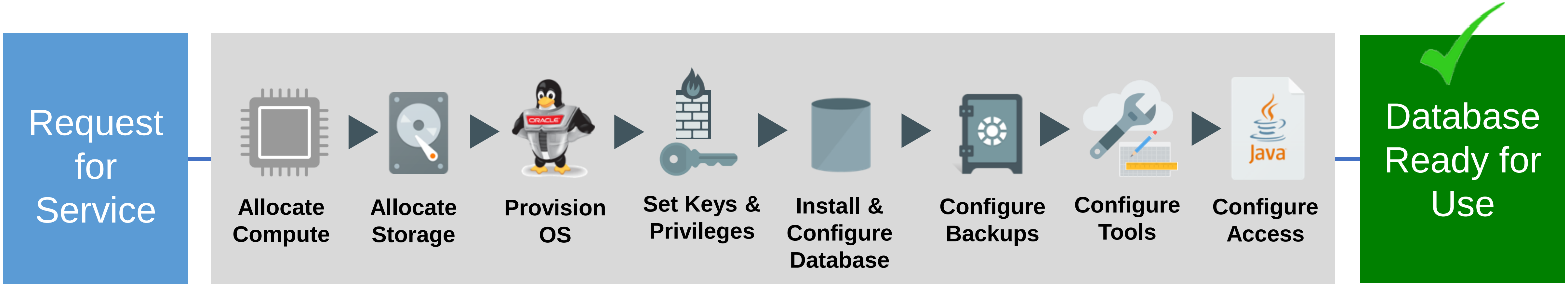
Any Language



Advanced Security

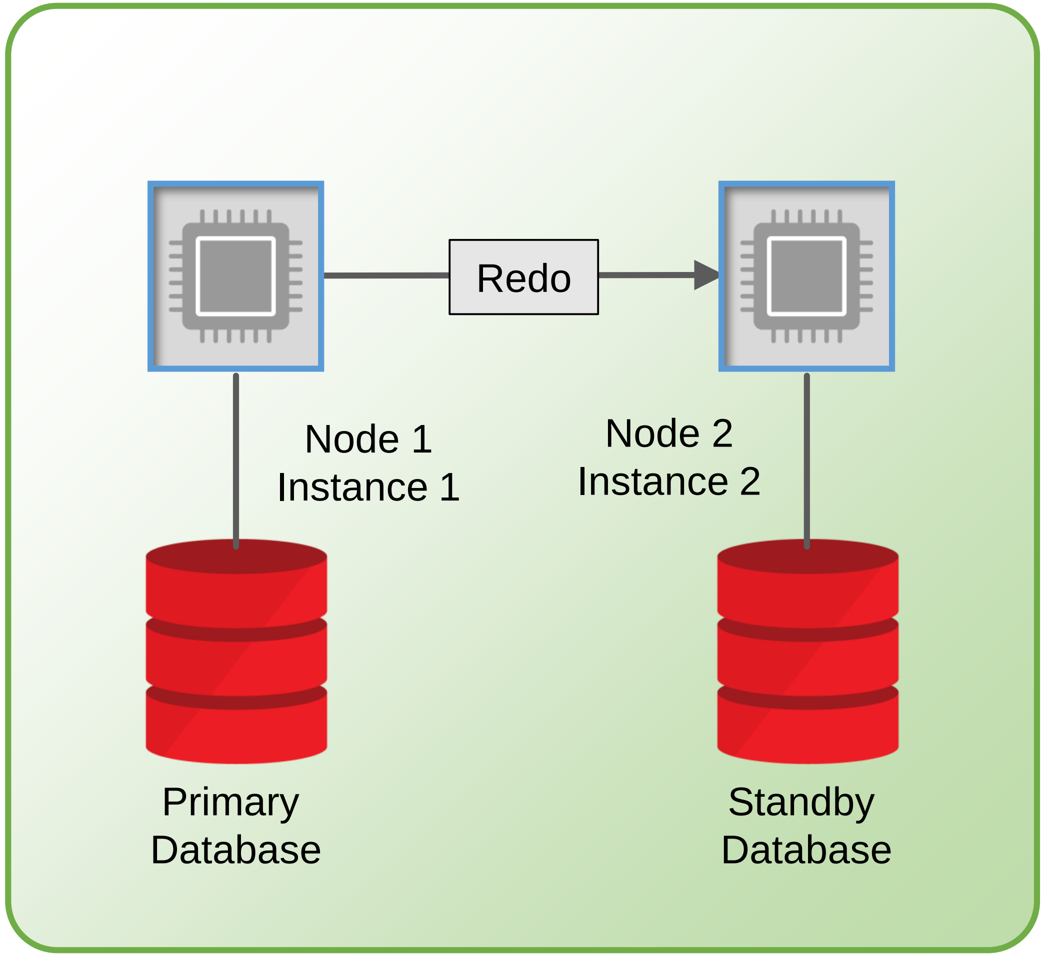
Local and Remote Management

Automated Database Provisioning

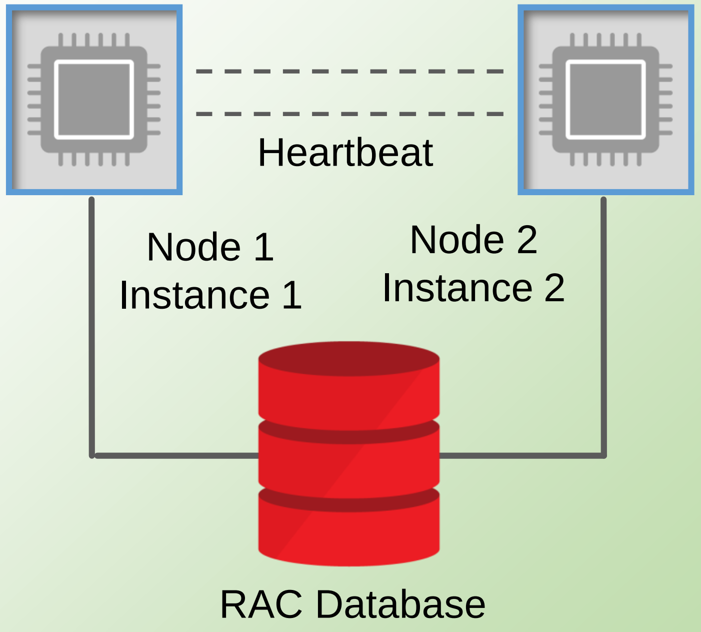


Automated provisioning based on input to the Create Oracle Database Cloud Service Instance wizard or through REST APIs

Additional Database Configuration Options



- Oracle Real Application Clusters (Oracle RAC)



Summary

- In this lesson, you should have learned how to:
  - Explain the Oracle Cloud and Database Cloud offerings
  - Describe the architecture and features of Oracle Database Cloud Service

Practice 3: Overview

- 3-1: Exploring Oracle Cloud

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
  - Describe the Oracle-supplied user accounts for Oracle Database
  - Query the data dictionary in 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.
- Finding the right tool for the job often comes down to preference (for example, whether you prefer to work with code or use a graphical user interface).
- 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
  - 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:

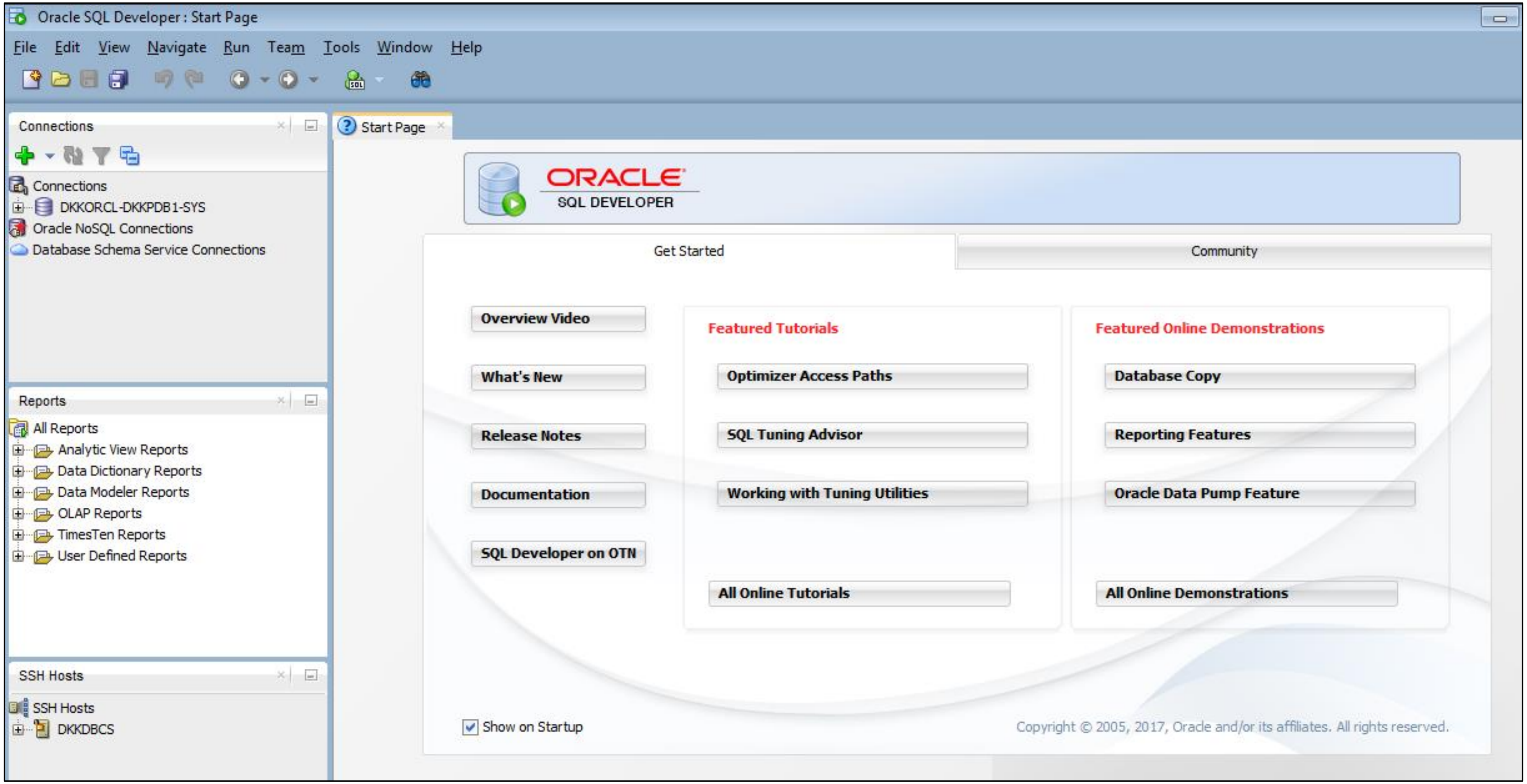
```
$ sqlplus hr/hr@HRPDB @script.sql
```

Username and password

Connect identifier

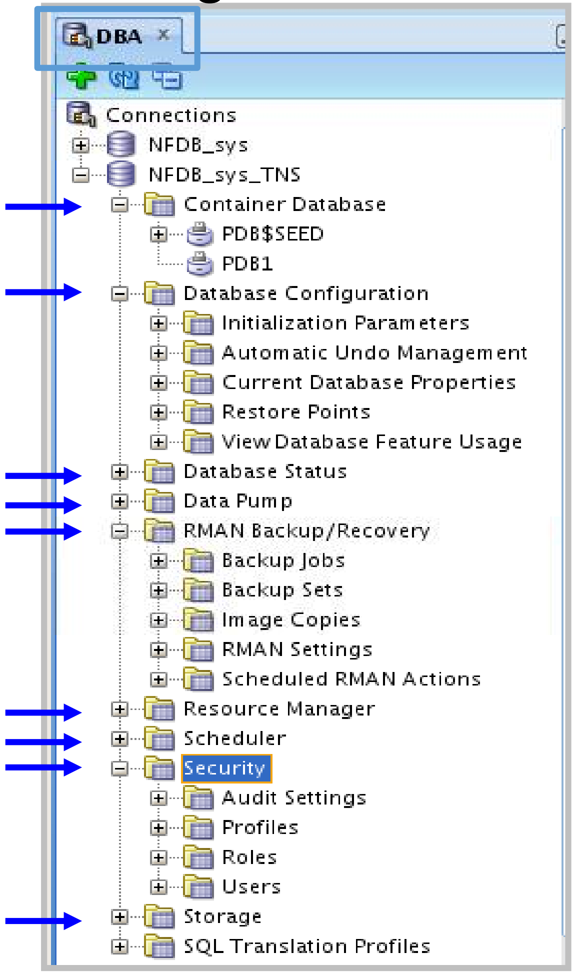
File name

Oracle SQL Developer

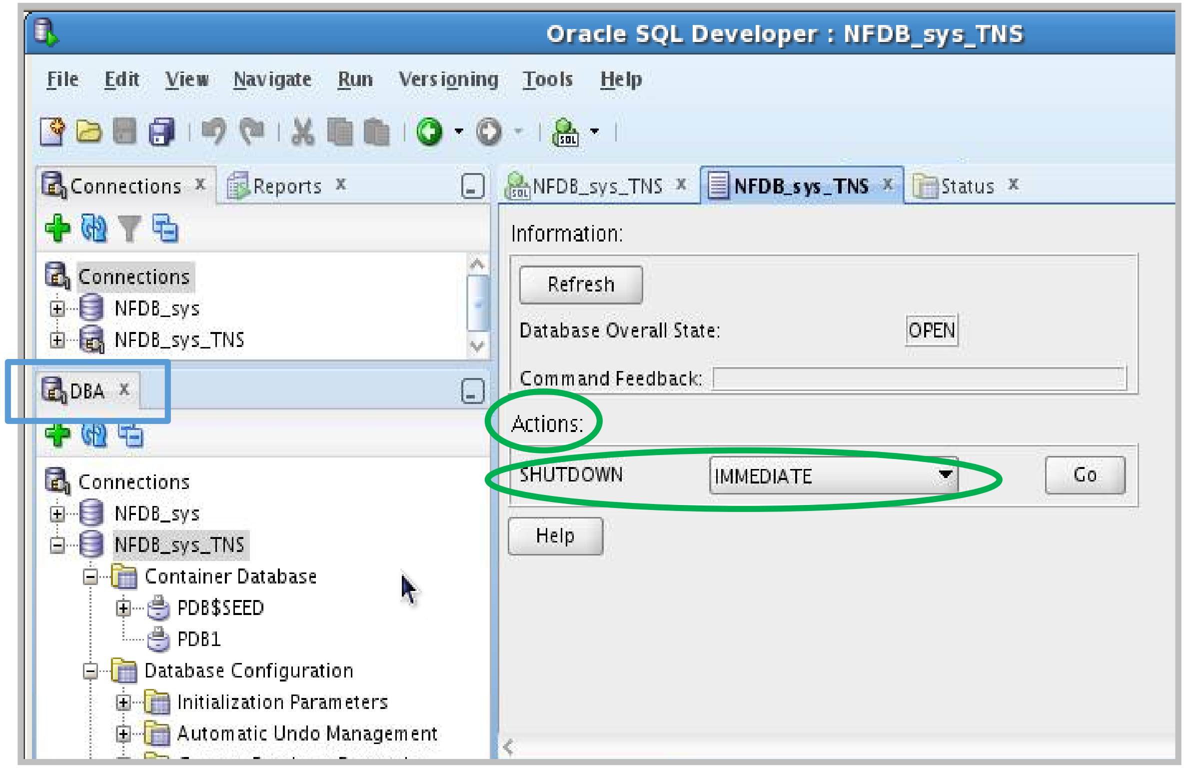


Oracle SQL Developer: DBA Actions

Using DBA features through DBA navigator



Performing DBA actions



SQL Developer Command Line (SQLcl)

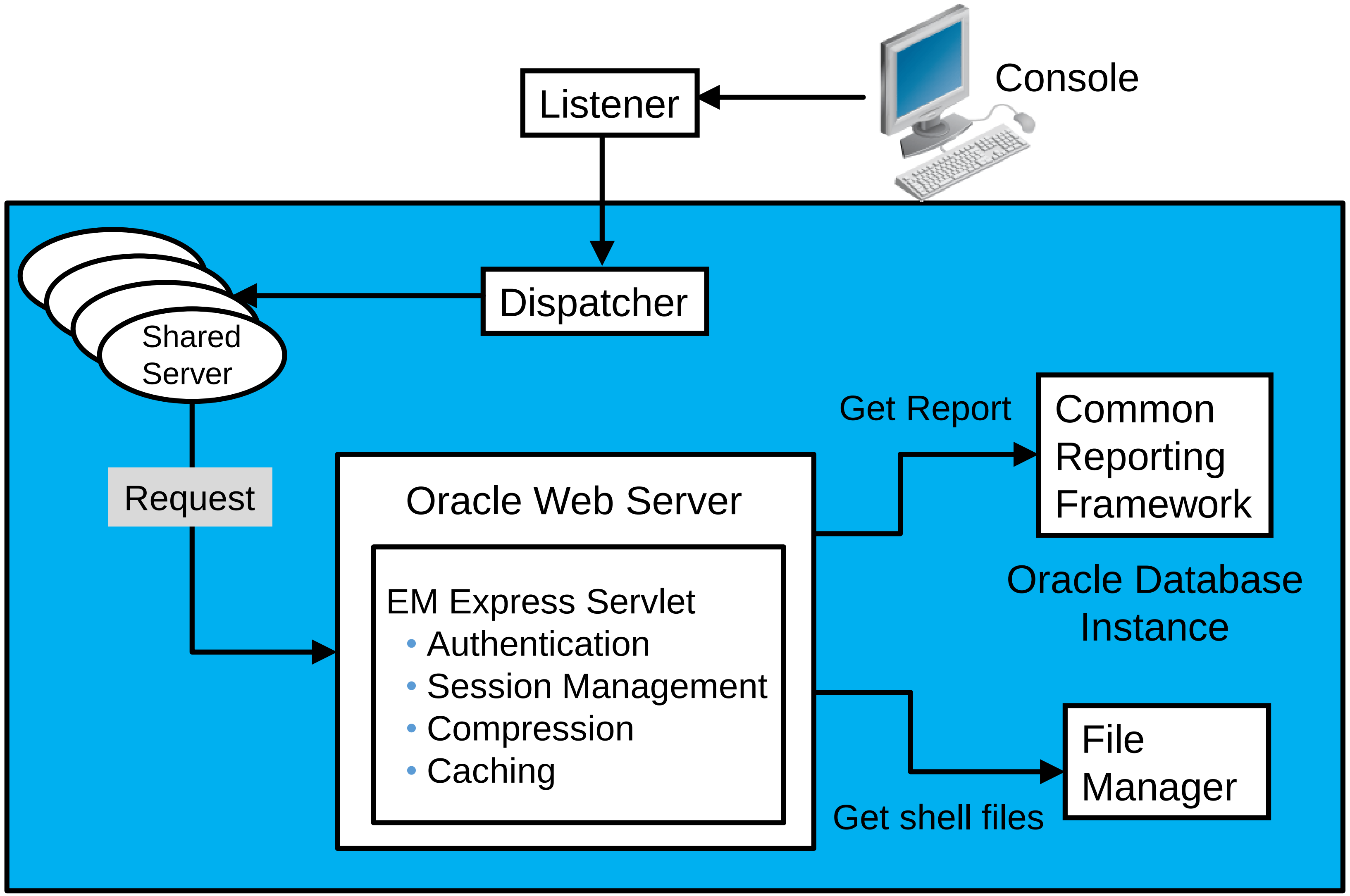
- SQLcl:
  - Is a tool that can access Oracle databases
  - Is a cross between SQL*Plus and SQL Developer
  - Incorporates Linux-like and IDE features, and color
- Example query with formatting to fit the screen:

```
SQL> SELECT first_name, last_name, salary, hire_date FROM hr.employees;

FIRST_NAME    LAST_NAME    SALARY    HIRE_DATE
Steven        King         24000     17-JUN-03
Neena         Kochhar      17000     21-SEP-05
...
```

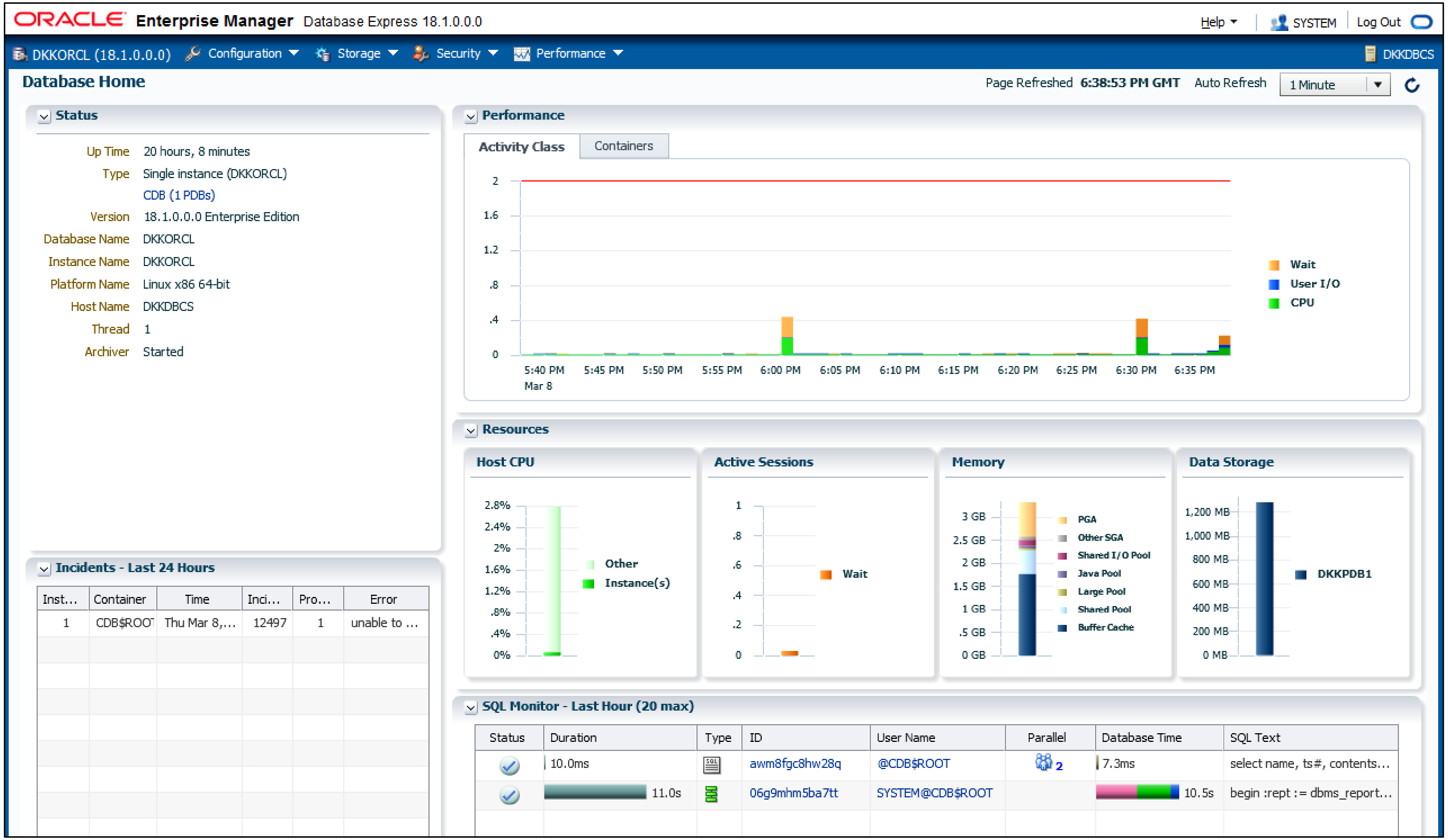
Database Configuration Assistant (DBCA)

- 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



Using the Database Home Page

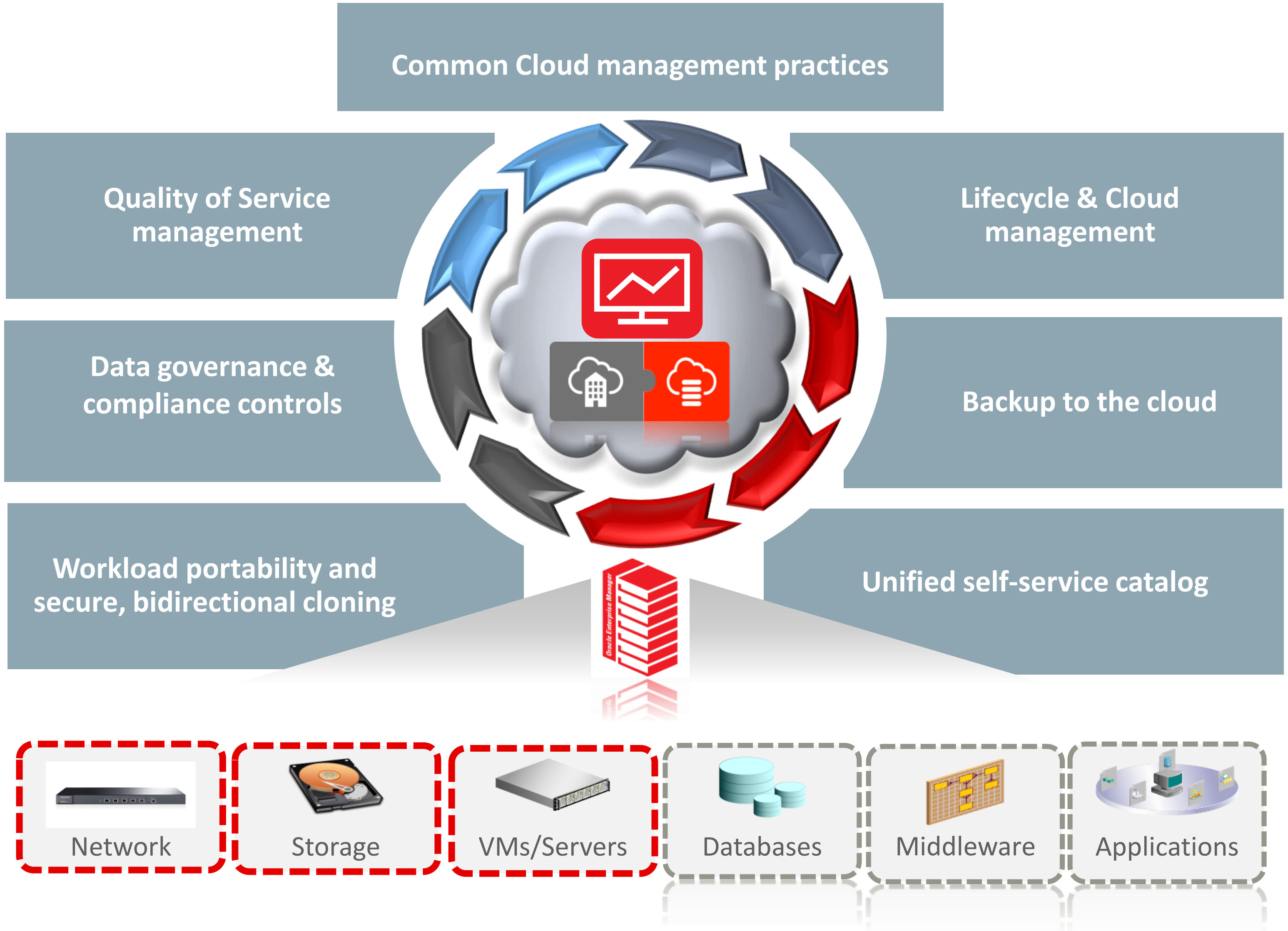


Enterprise Manager Cloud Control 13*c* Features

- Enterprise-Ready Framework
- Unified Cloud Management: Hybrid Cloud
- Chargeback and Capacity Planning
- Exadata and Exalogic Management
- Configuration Management
- Provisioning and Patching
- Application Quality Management
- Database Management
- Middleware Management
- Applications Management

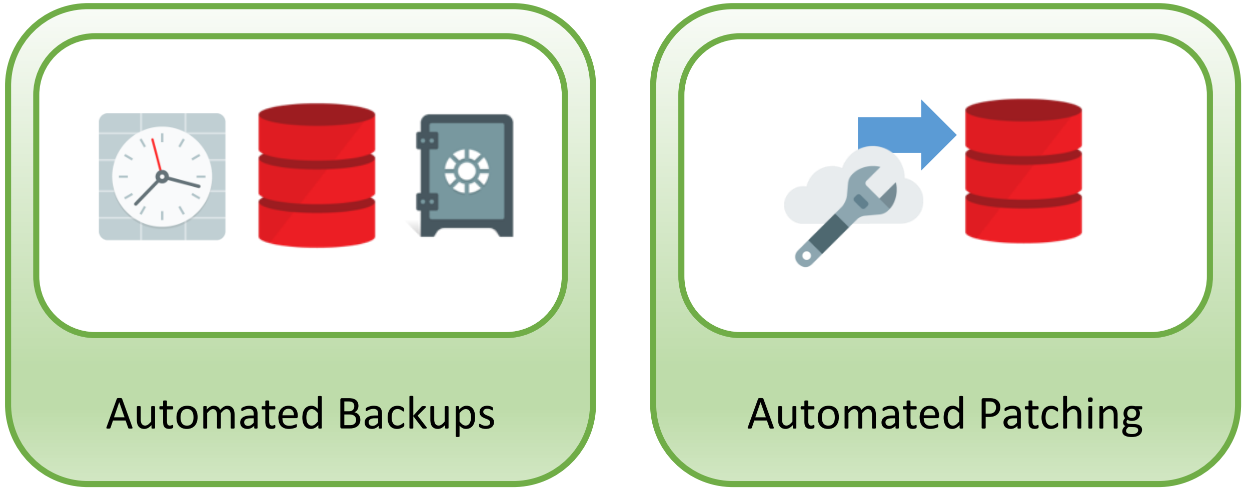
ORACLE ENTERPRISE MANAGER 13c

Single Pane of Glass for Enterprise Management



Oracle Database Cloud Service Tools

- Oracle Database Cloud Service includes:
  - Oracle Database tools such as SQL*Plus and Enterprise Manager Database Express
  - Cloud tooling that simplifies backup, recovery, patching, and upgrade operations



Cloud Tooling

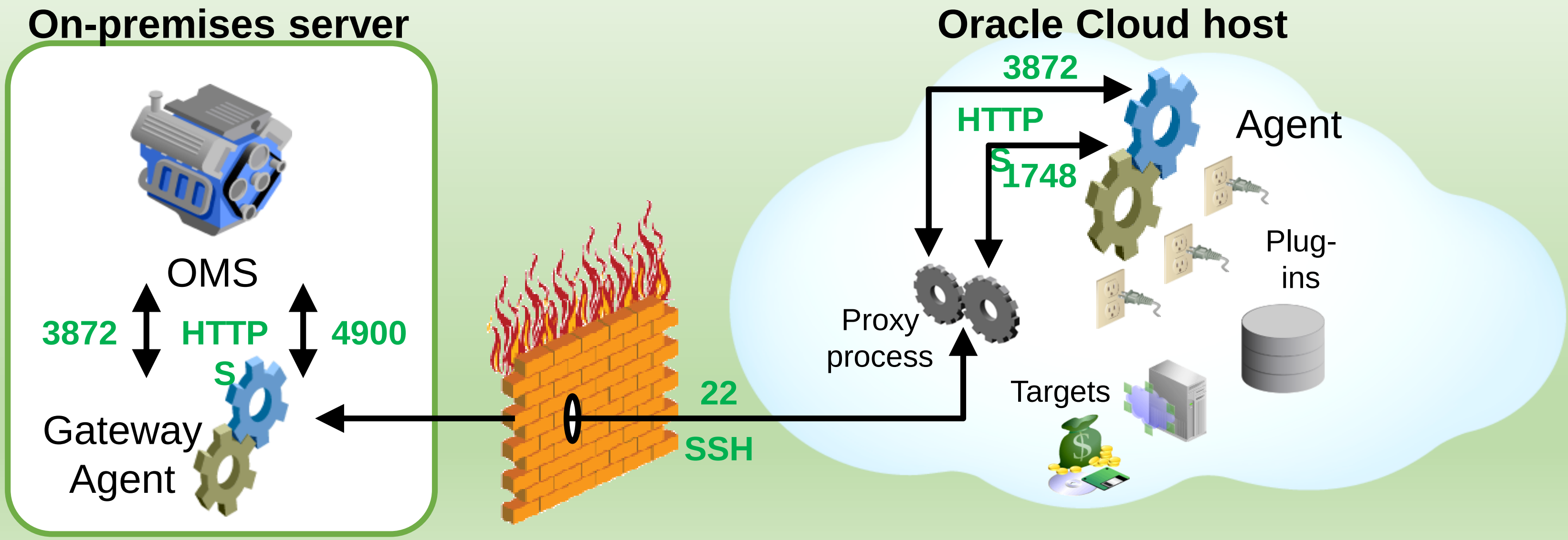
Feature	Description	Tools
Simple Automated Backups	Perform on-demand backups Change automatic backup configuration	`bkup_api` utility `raccli` utility
Simple Automated Recovery	Restore from backups and recover the database	`orec` subcommand of the `dbaascli` utility `raccli` utility
Simple Automated Patching	Apply patches	`dbpatchm` subcommand of the `dbaascli` utility `dbpatchmdg` utility `raccli` utility

Accessing Tools and Features from the DBCS Console

- From the database deployment menu, you can select the following to access consoles and perform other actions:
  - Application Express Console
  - DBaaS Monitor Console
  - EM Console
  - SSH Access
  - Access Rules
  - Delete

Using Enterprise Manager Cloud Control

- You can monitor and manage DBCS with on-premises Enterprise Manager Cloud Control.



See [Using Oracle Enterprise Manager Cloud Control with Database Cloud Service]() in *Using Oracle Database Cloud Service*

Summary

- In this lesson, you should have learned how to:
  - Connect to an Oracle Database
  - Describe the tools used to access an Oracle Database
  - Describe the Oracle-supplied user accounts for Oracle Database
  - Query the data dictionary in an Oracle Database

Practice 5: Overview

- 5-1: Viewing Database Deployment Information and Menus
- 5-2: Connecting to the Database Deployment Compute Node
- 5-3: Exploring a CDB by Using SQL*Plus
- 5-4: Exploring a PDB by Using SQL*Plus
- 5-5: Installing the HR Sample Schema
- 5-6: Copying Course Practice Files

Managing DBCS Database Deployments

Objectives

- After completing this lesson, you should be able to:
  - Manage the compute node associated with a database deployment
  - Manage network access to a database deployment
  - Scale the compute shape and storage
  - Patch the database deployment

Managing the Compute Node

Action	Result of the Action
Start	An Oracle Compute Cloud Service instance is allocated, resources are attached, and it is started.
Stop	An Oracle Compute Cloud Service instance is stopped. Only actions after stop are start and delete.
Restart	An Oracle Compute Cloud Service instance is stopped and immediately started again.

Managing Network Access to DBCS (OCI Classic)

- By default, network access to the compute node is provided by Secure Shell (SSH) connections on port 22.
- To access network protocols and services by using a port other than port 22:
  - Use the Oracle Compute Cloud Service console or the Oracle Database Cloud Service console to enable an access rule
  - Create an SSH tunnel to the port

Enabling Access to a Compute Node Port (OCI Classic)

- Database Cloud Service relies on Oracle Compute Cloud Service to provide secure network access to database deployments.
- The Oracle Compute Cloud Service security rules that are created, but not enabled, are as follows:
  - ora_p2_dbconsole: For port 1158, used by Enterprise Manager 11*g* Database Control
  - ora_p2_dbexpress: For port 5500, used by Enterprise Manager Database Express 12c
  - ora_p2_dblistener: For port 1521, used by SQL*Net
  - ora_p2_http: For port 80, used for HTTP connections
  - ora_p2_httpssl: For port 443, used for HTTPS connections including Oracle REST Data Services, Oracle Application Express, and Oracle DBaaS Monitor
- You enable the security rules through the Database Cloud Service console.
- Enabling one of the predefined security rules opens the given port to the public Internet.

Scaling a Database Deployment

Action	Description
Scale Up	Select a new compute shape. Add raw storage to the database deployment.
Scale Down	Select a new compute shape.

Patching DBCS

- Patch management tasks:
  - Check prerequisites before applying a patch.
  - Apply a patch.
  - Roll back a patch or a failed patch.
- Tools and utilities:
  - Oracle Database Cloud Service console for single-instance databases
  - `dbpatchm` subcommand of the `dbaascli` utility for single-instance databases
  - `raccli` utility for Oracle RAC databases
  - `dbpatchmdg` utility for Oracle Data Guard configurations

Using the DBCS Console to Manage Patches

- Use the menu on the Patching tab to:
  - Check the prerequisites of a patch before you apply it to a database deployment on Database Cloud Service
  - Apply a patch to a database deployment on Database Cloud Service

Using the `dbaascli` Utility to Manage Patches

- Checking prerequisites:

```
# dbaascli dbpatchm --run -prereq
```

- Applying a patch:

```
# dbaascli dbpatchm --run -apply
```

- Rolling back a patch:

```
# dbaascli dbpatchm --run -rollback
```

Summary

- In this lesson, you should have learned how to:
  - Manage the compute node associated with a database deployment
  - Manage network access to a database deployment
  - Scale the compute shape and storage
  - Patch the database deployment

Practice 6: Overview

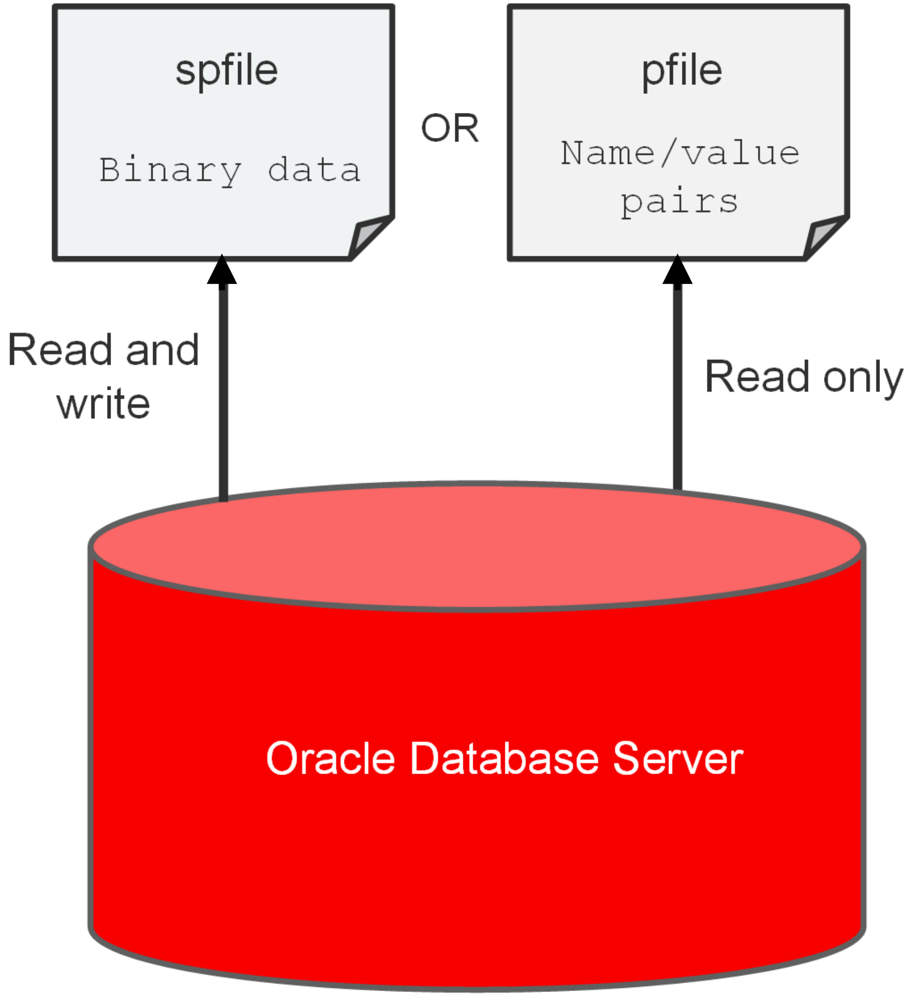
- 6-1: Accessing Enterprise Manager Database Express
- 6-2: Exploring a CDB and PDB by Using Enterprise Manager Database Express

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
  - Start up and shut down Oracle databases
  - Open and close PDBs
  - 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 EM Express 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.

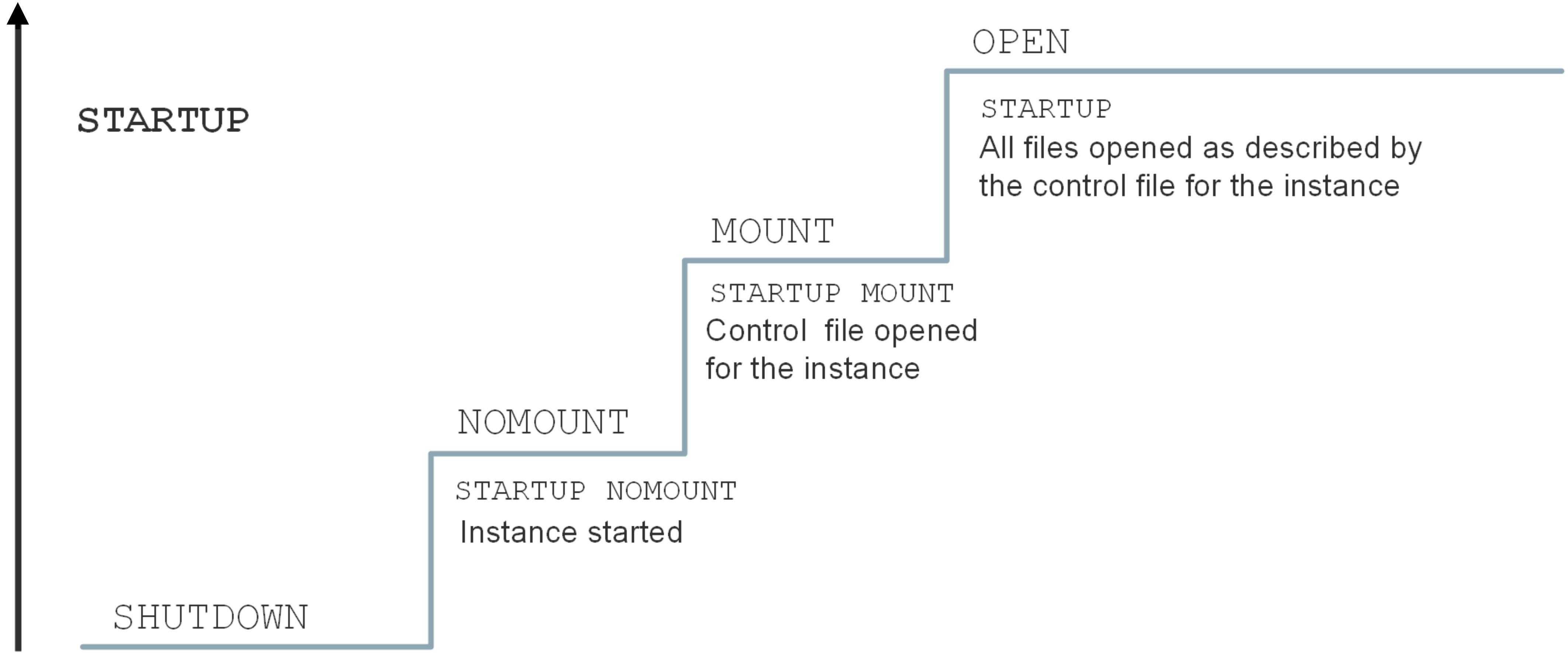
Viewing Initialization Parameters

- 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`

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`, `TRANSACTIONAL`, and `NORMAL`.

	ABORT	**IMMEDIATE**	**TRANSACTIONAL**	**NORMAL**
Allows new connections	No	No	No	No
Waits until current sessions end	No	No	No	Yes
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

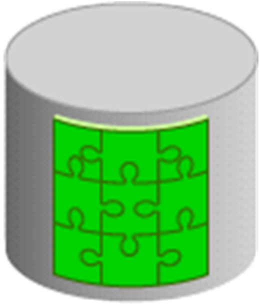
or

SHUTDOWN IMMEDIATE

On the way up:

- No instance recovery

Consistent database

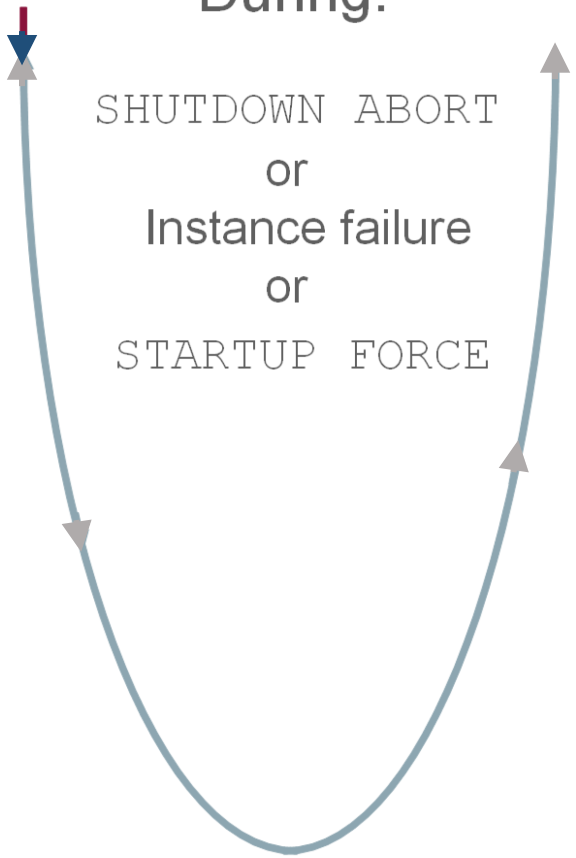


Comparing SHUTDOWN Modes

On the way down:

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

During:



Inconsistent database

On the way up:

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

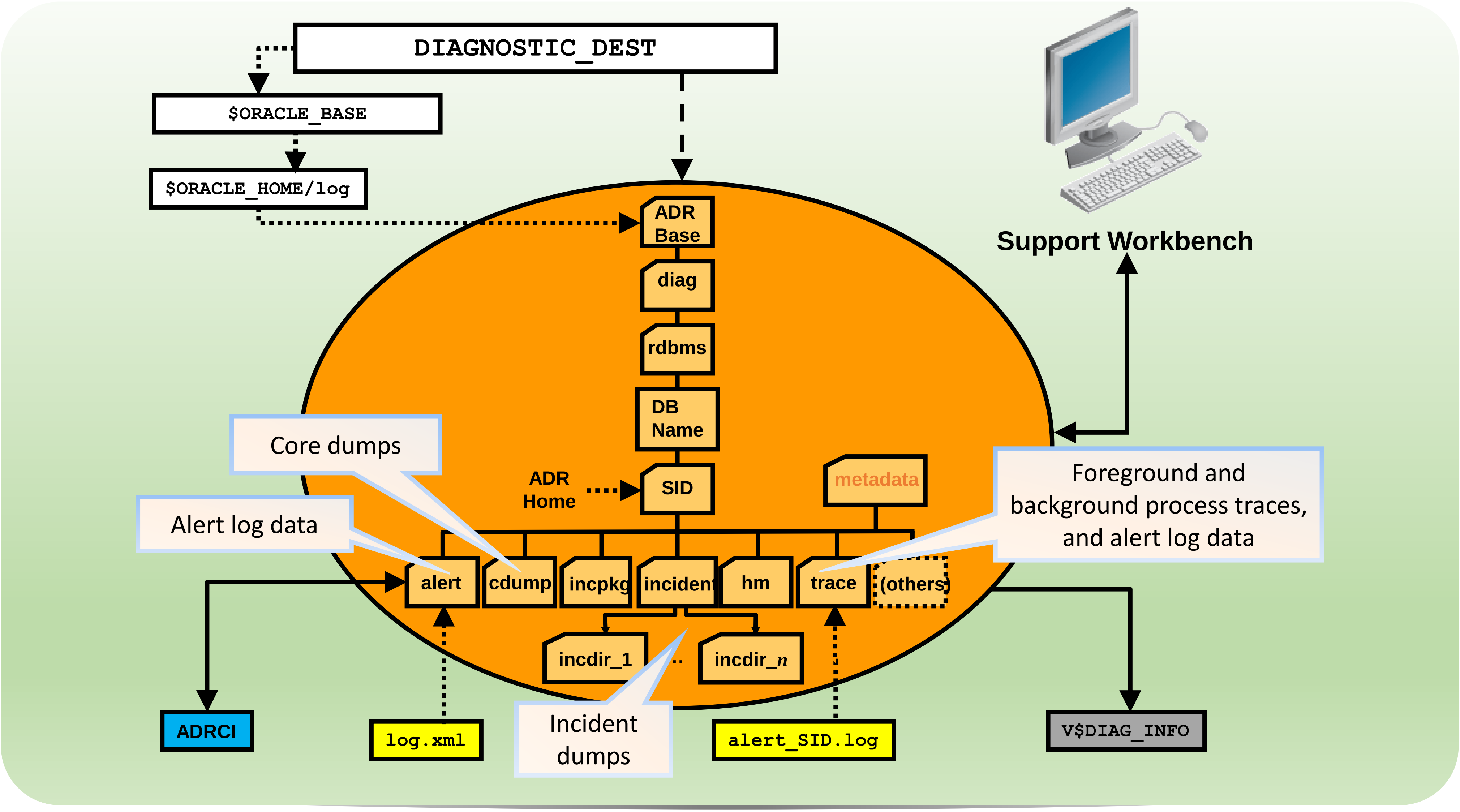
Opening and Closing PDBs

- 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.

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 containing 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

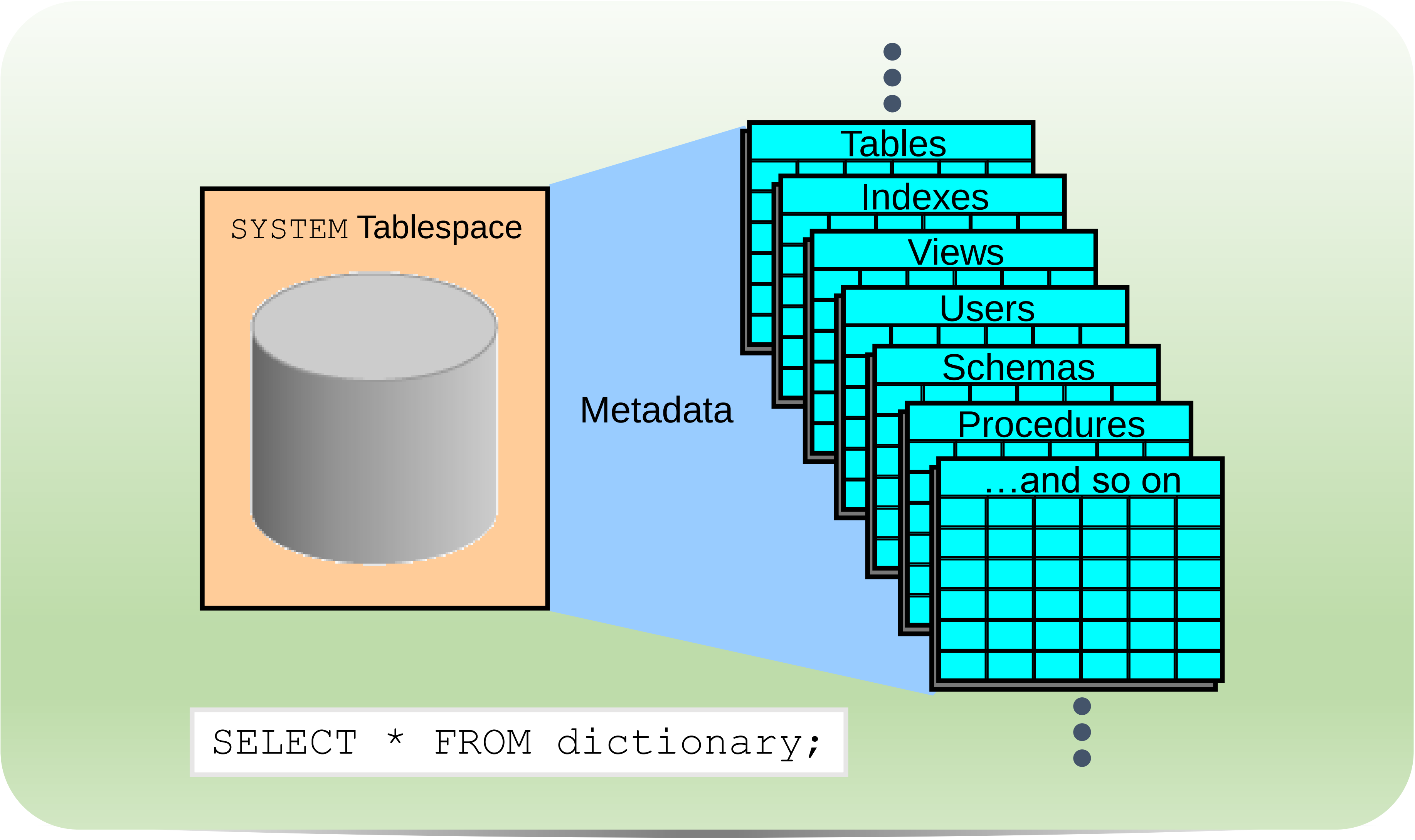
- Dynamic performance views provide access to information about the changing states of instance memory structures:
  - Sessions, file states, and locks
  - Progress of jobs and tasks
  - Backup status, memory usage, and allocation
  - System and session parameters
  - SQL execution
  - Statistics and 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;
```

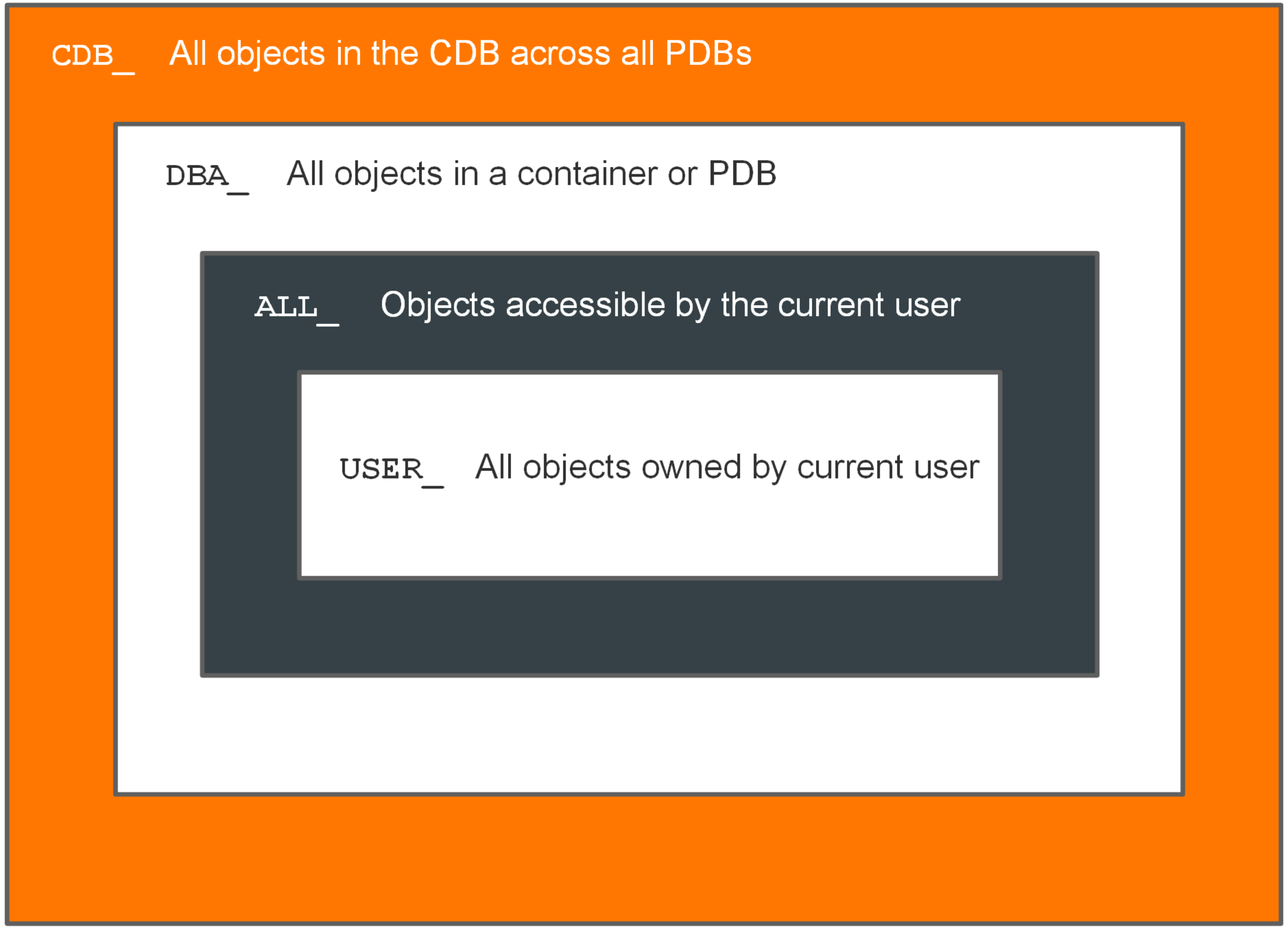
Considerations for Dynamic Performance Views

- 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



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
  - Start up and shut down Oracle databases
  - Open and close PDBs
  - Work with the Automatic Diagnostic Repository (ADR)
  - Query dynamic performance views

Practice 7: Overview

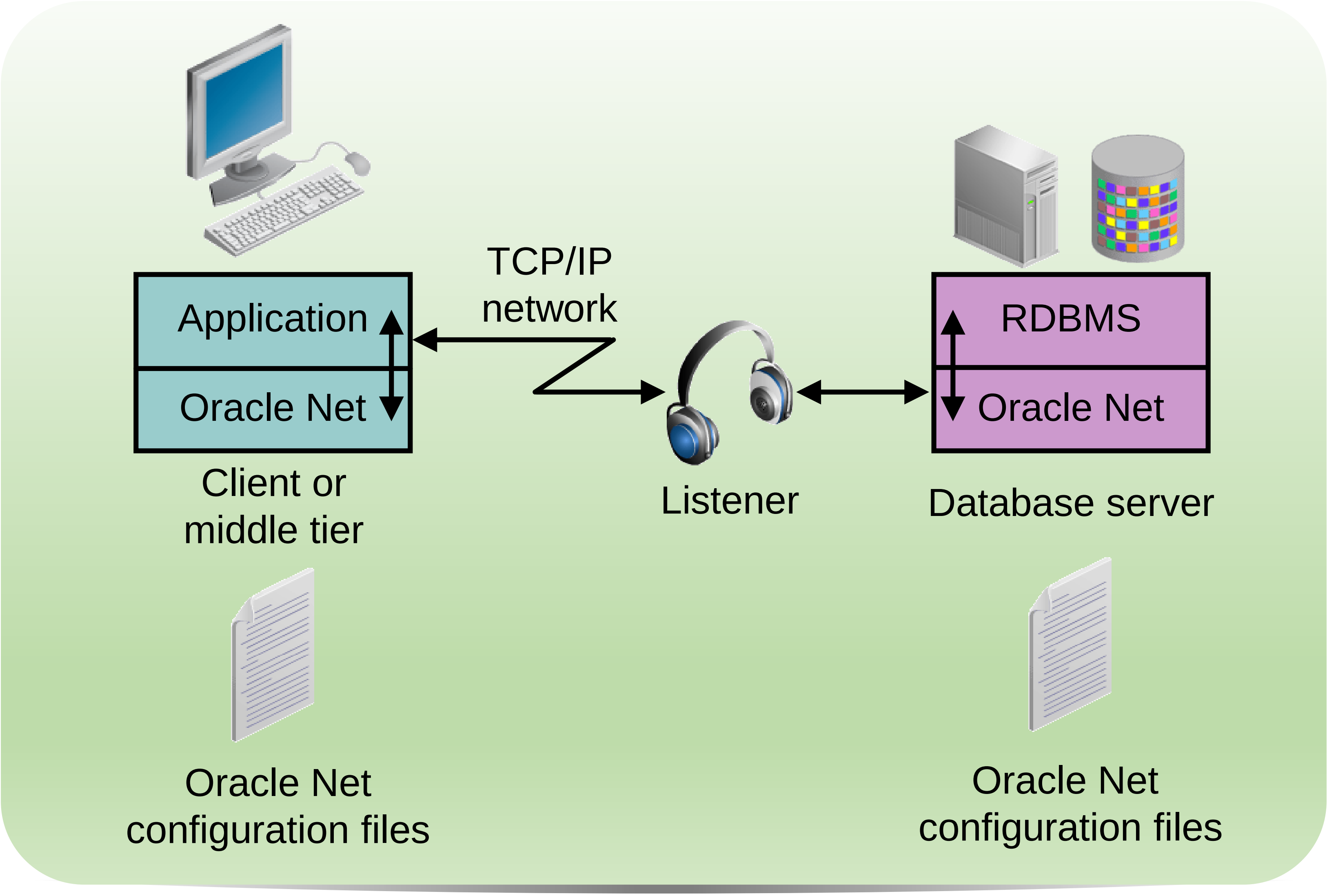
- 7-1: Investigating Initialization Parameter Files
- 7-2: Viewing Initialization Parameters by Using SQL*Plus
- 7-3: Modifying Initialization Parameters by Using SQL*Plus
- 7-4: Modifying an Initialization Parameter by Using Enterprise Manager Database Express
- 7-5: Shutting Down and Starting Up the Oracle Database
- 7-6: Viewing Diagnostic Information

Oracle Net Services

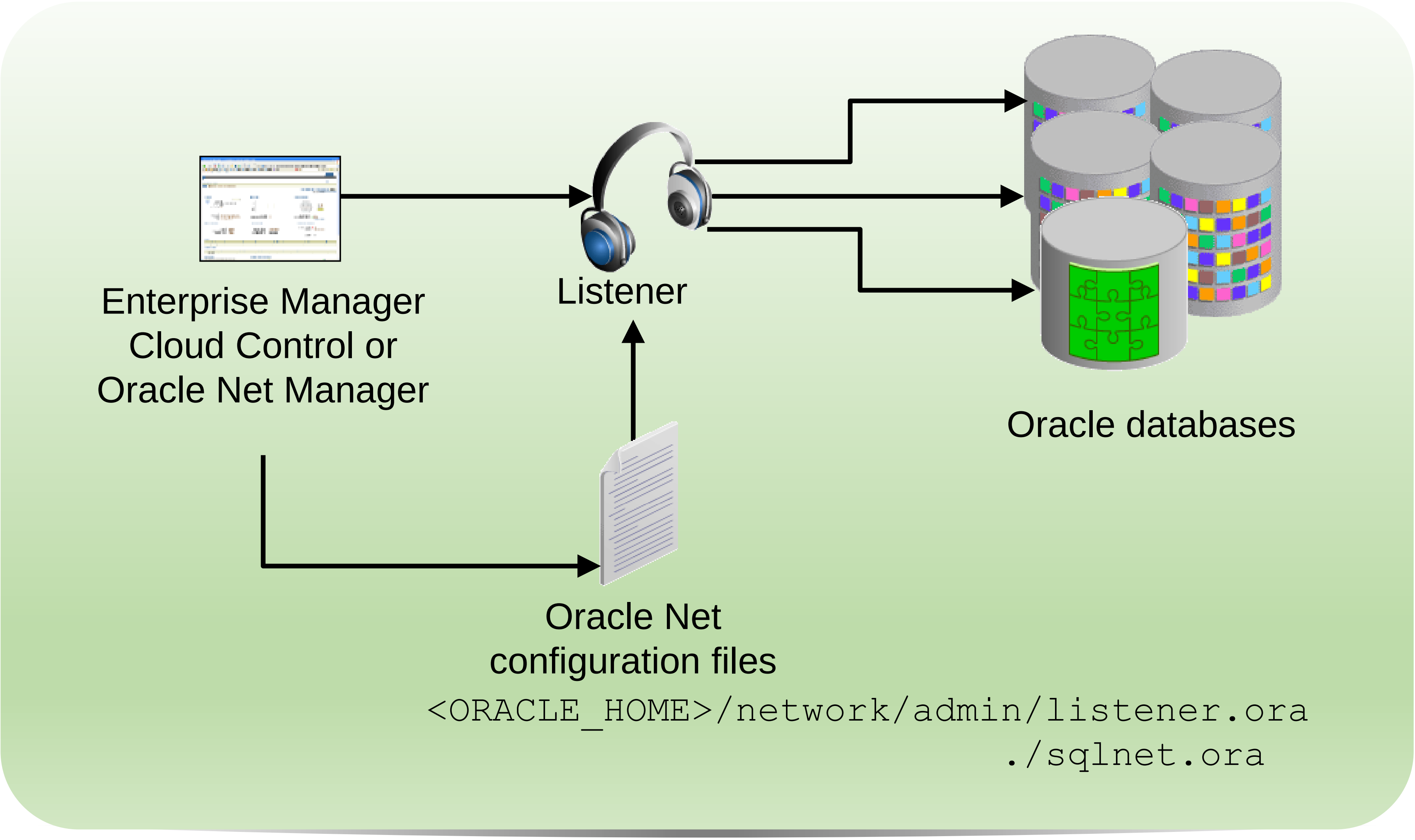
Objectives

- After completing this lesson, you should be able to:
  - Describe Oracle Net Services
  - Explain how listeners work
  - Configure listeners for dynamic or static service registration
  - Configure local naming for database connections
  - Test Oracle Net connectivity with `tnsping`
  - Configure communication between databases by creating database links
  - Explain the difference between dedicated and shared server configurations

Oracle Net Services: Overview



Oracle Net Listener: Overview



The Default Listener

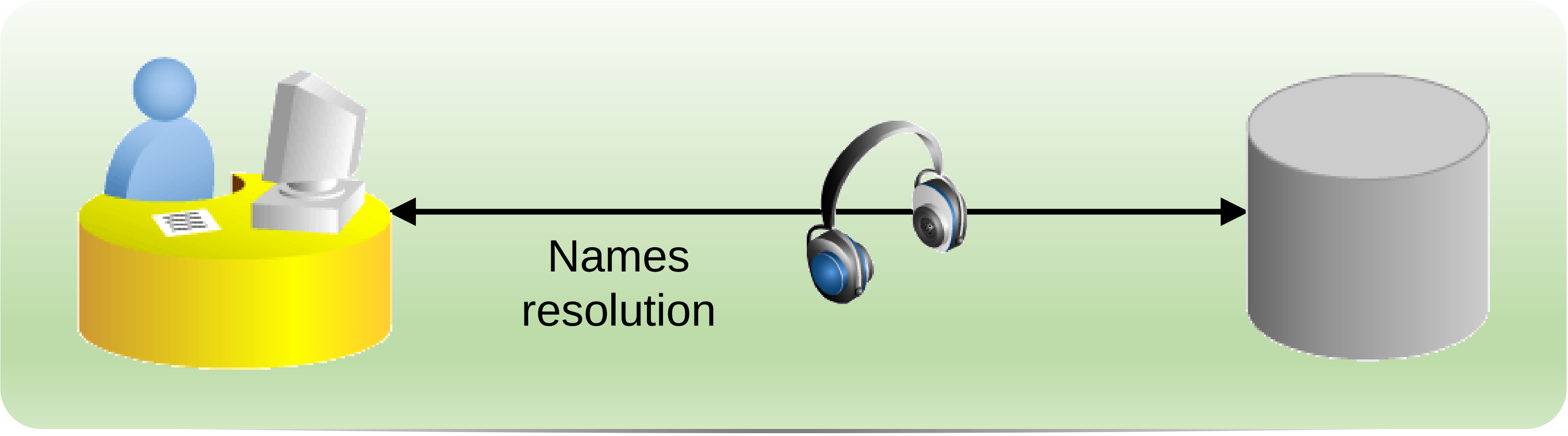
- During an Oracle Database installation, Oracle Universal Installer launches Oracle Net Configuration Assistant and creates a local listener named `LISTENER`.
- `LISTENER` is automatically populated with available database services through a feature called dynamic service registration.
- `LISTENER` listens on the following TCP/IP protocol address:

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

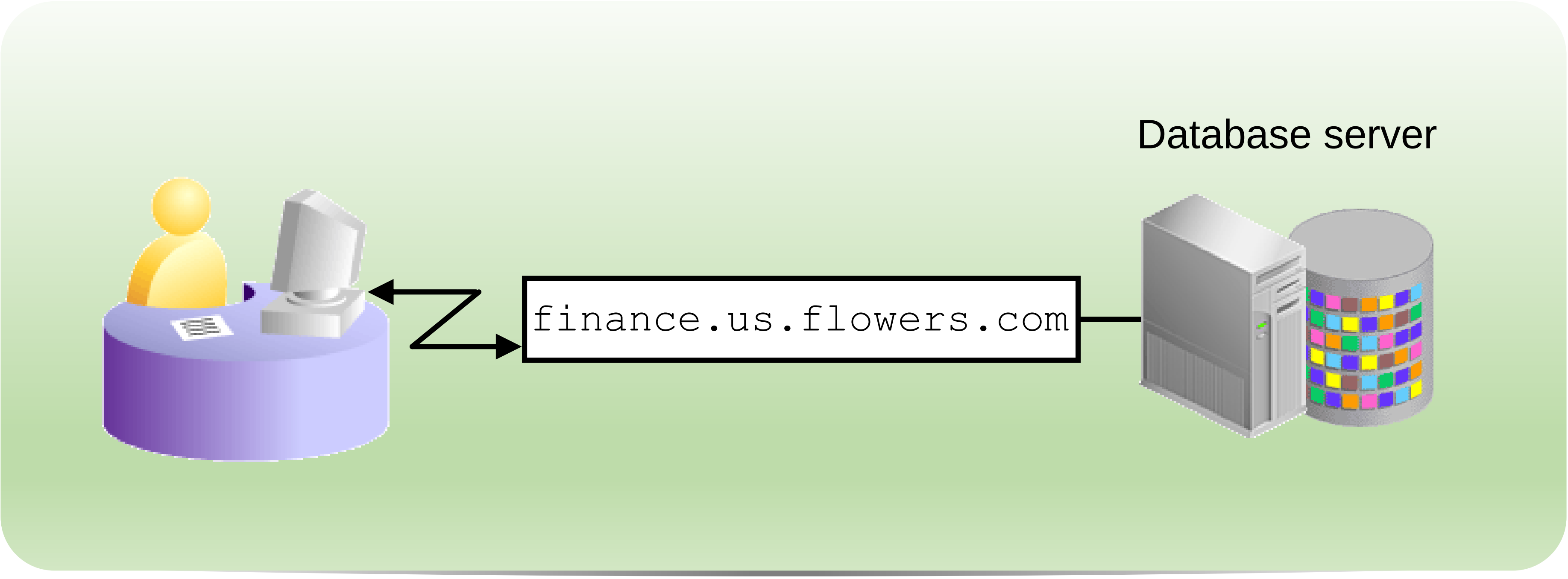
- Without any configuration, you can access your database instance immediately through `LISTENER`.
- If the listener name is `LISTENER` and it cannot be resolved, a protocol address of TCP/IP and a port number of 1521 is assumed.

Establishing Oracle Network Connections

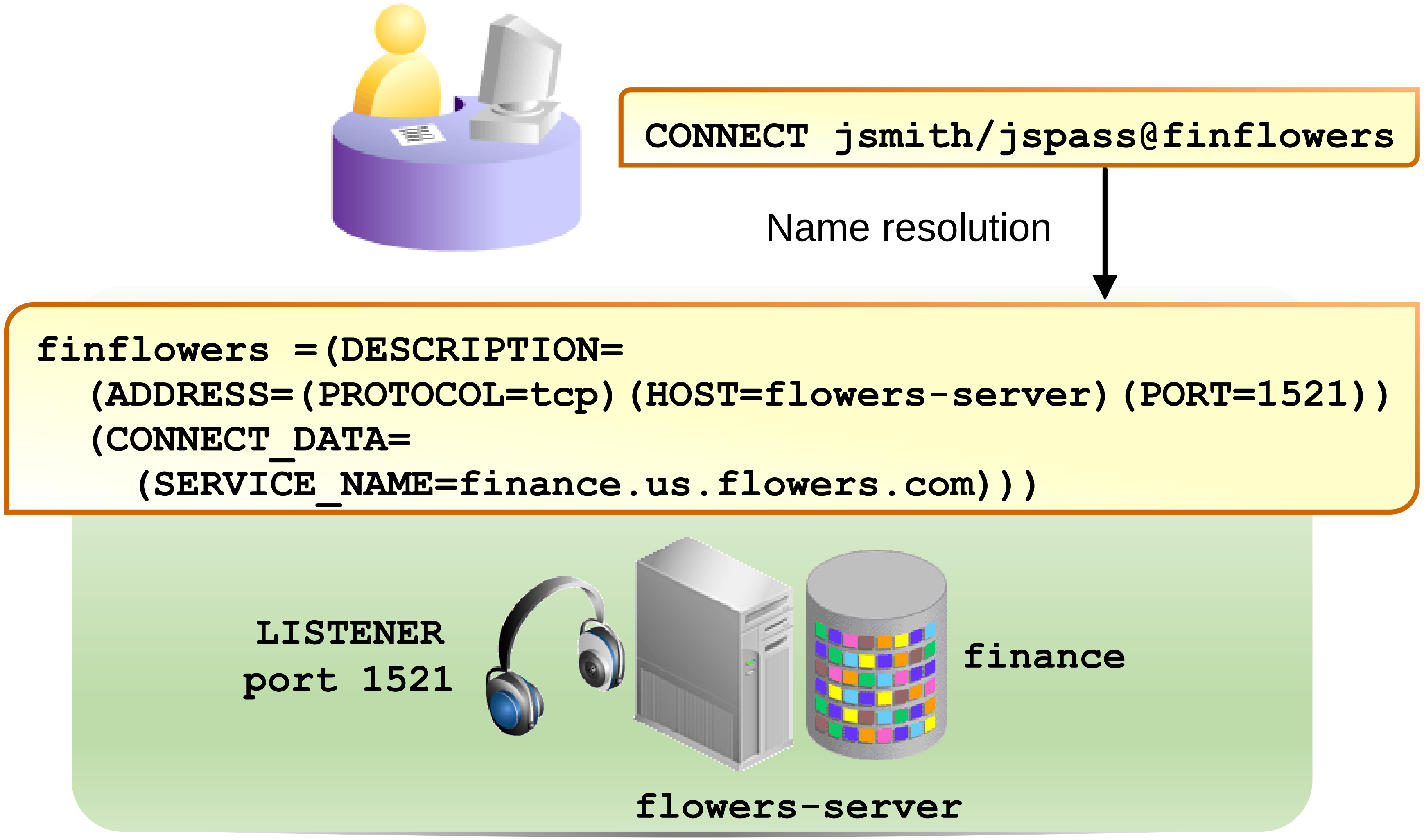
- To make a client or middle-tier connection, Oracle Net requires the client to know the:
  - Host where the listener is running
  - Port that the listener is monitoring
  - Protocol that the listener is using
  - Name of the service that the listener is handling



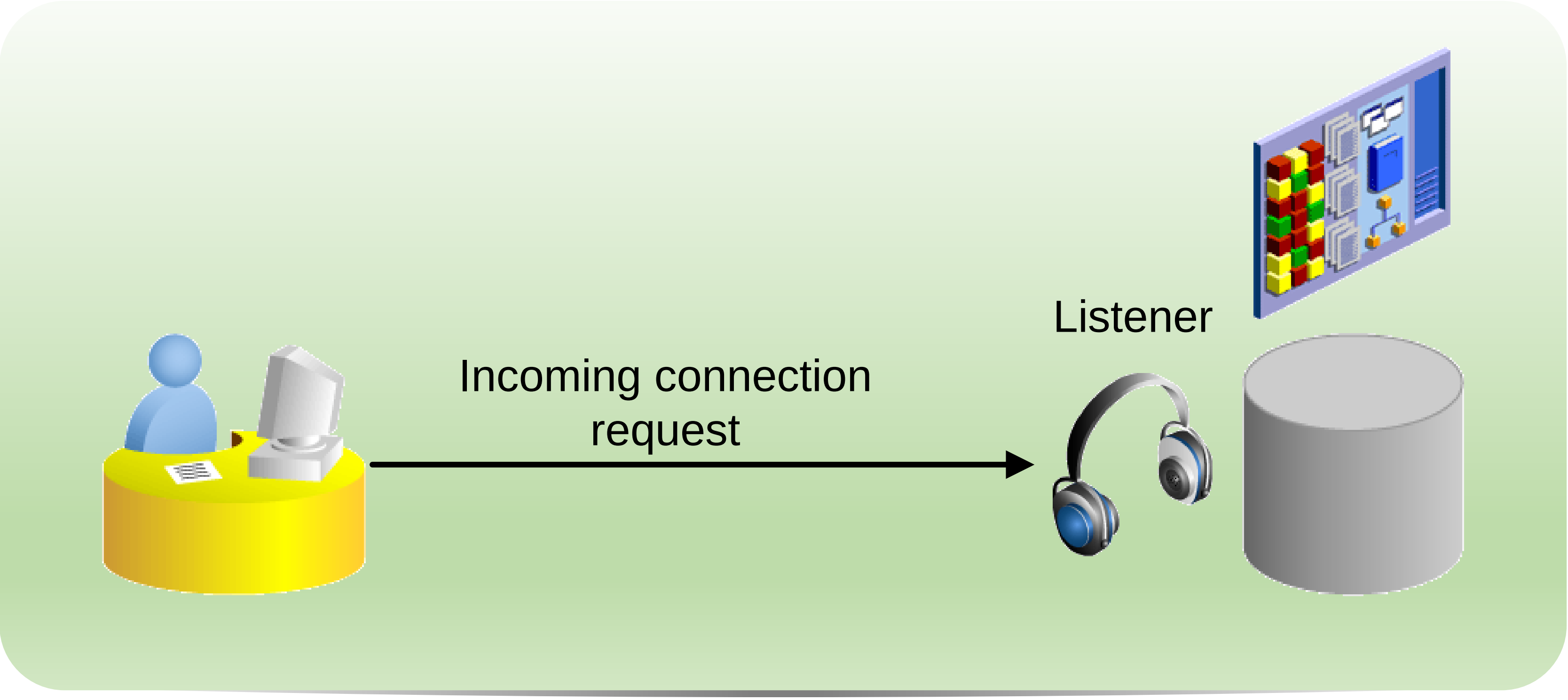
Connecting to an Oracle Database



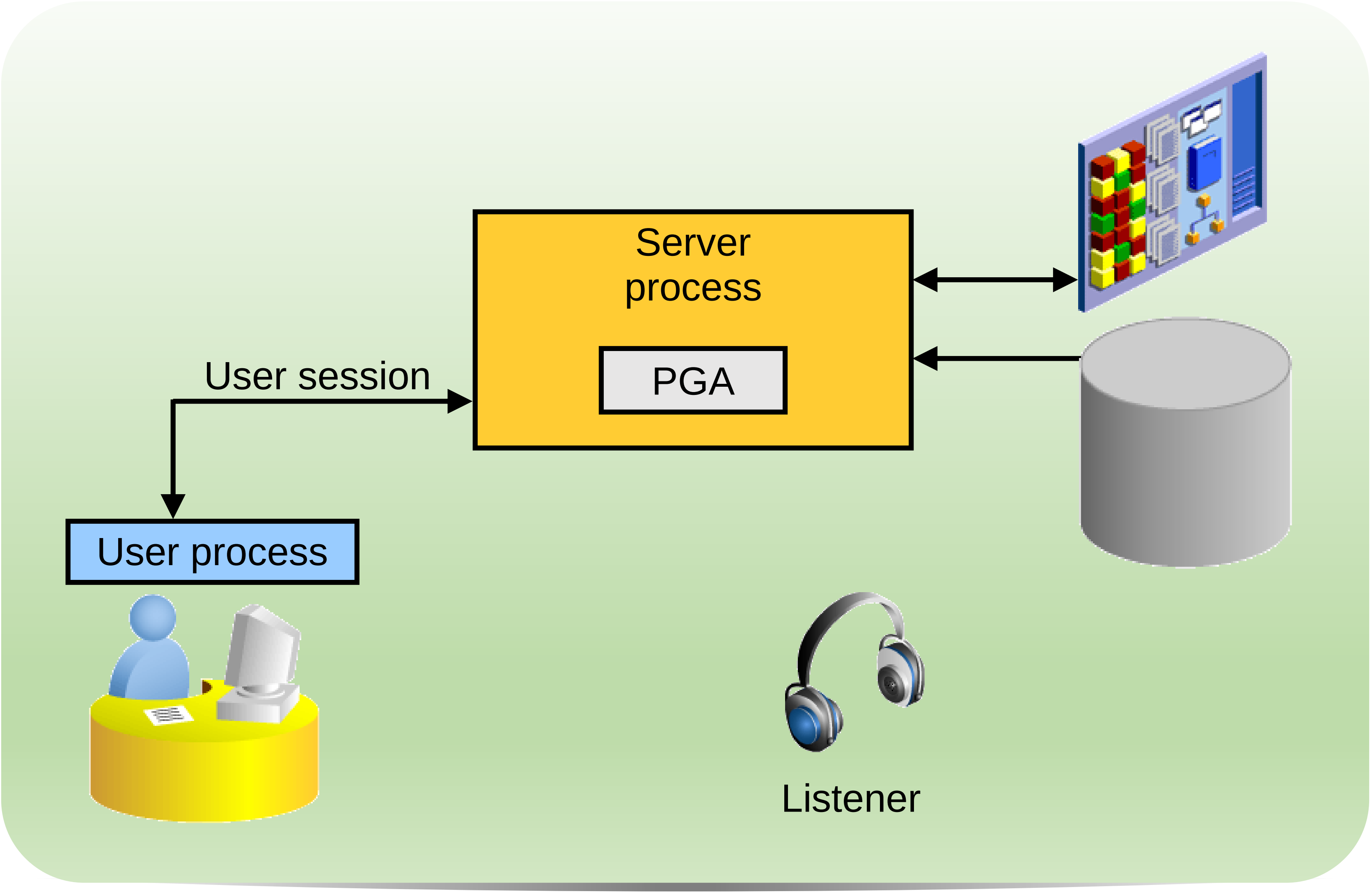
Name Resolution



Establishing a Connection



User Sessions



Configuring Dynamic Service Registration

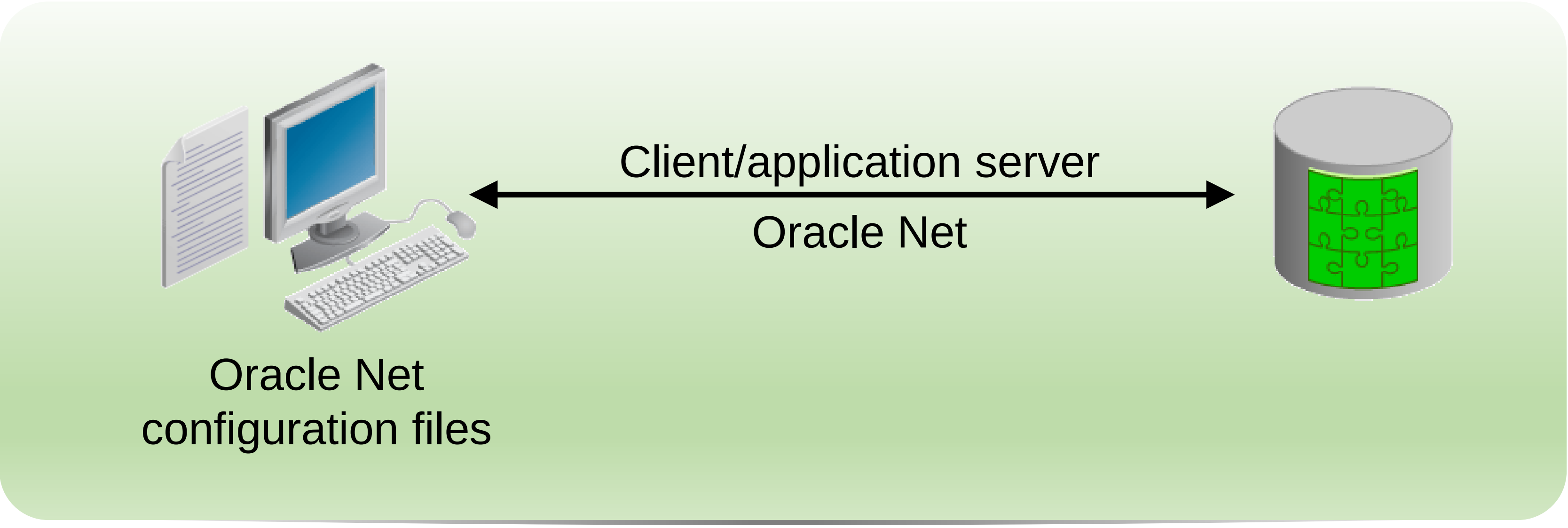
- By default, an Oracle database is configured to use dynamic service registration (service registration), which allows the Oracle database to identify its available services to listeners automatically.
- The LREG process polls the listeners to see if they're running and, if so, registers database service 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.

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 Configuration utility shows the services status as `UNKNOWN`.
- You can have both static listeners and dynamic listeners configured at the same time.
- 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.

Naming Methods

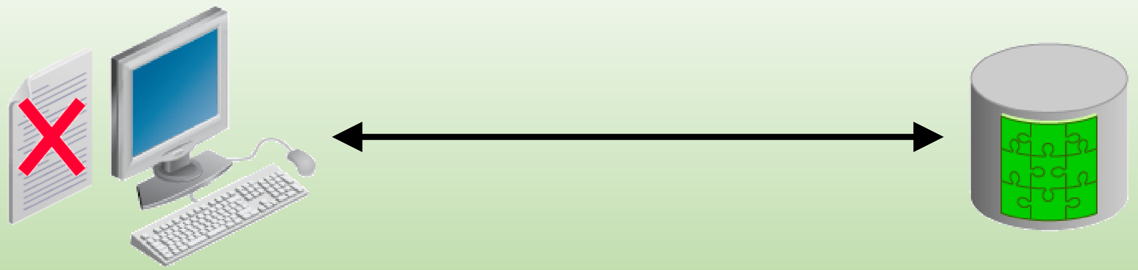
- Oracle Net supports several methods of resolving connection information:
  - Easy connect naming: Uses a TCP/IP connect string
  - Local naming: Uses a local configuration file
  - Directory naming: Uses a centralized LDAP-compliant directory server



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

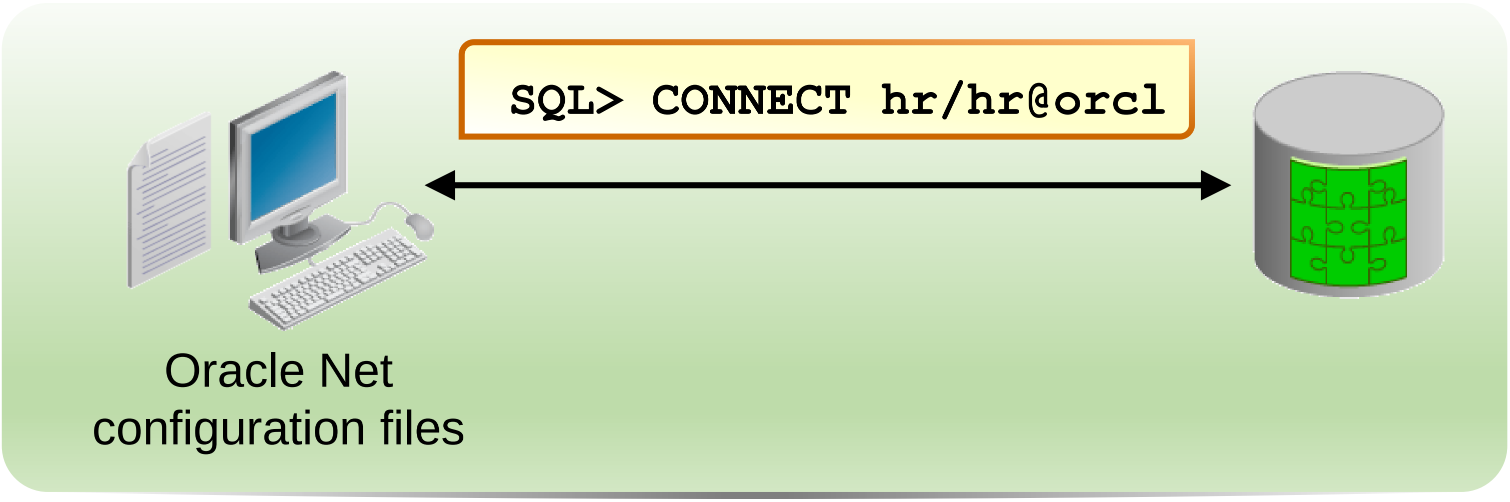
```
SQL> CONNECT hr/hr@db.us.oracle.com:1521/dba11g
```



No Oracle Net configuration files

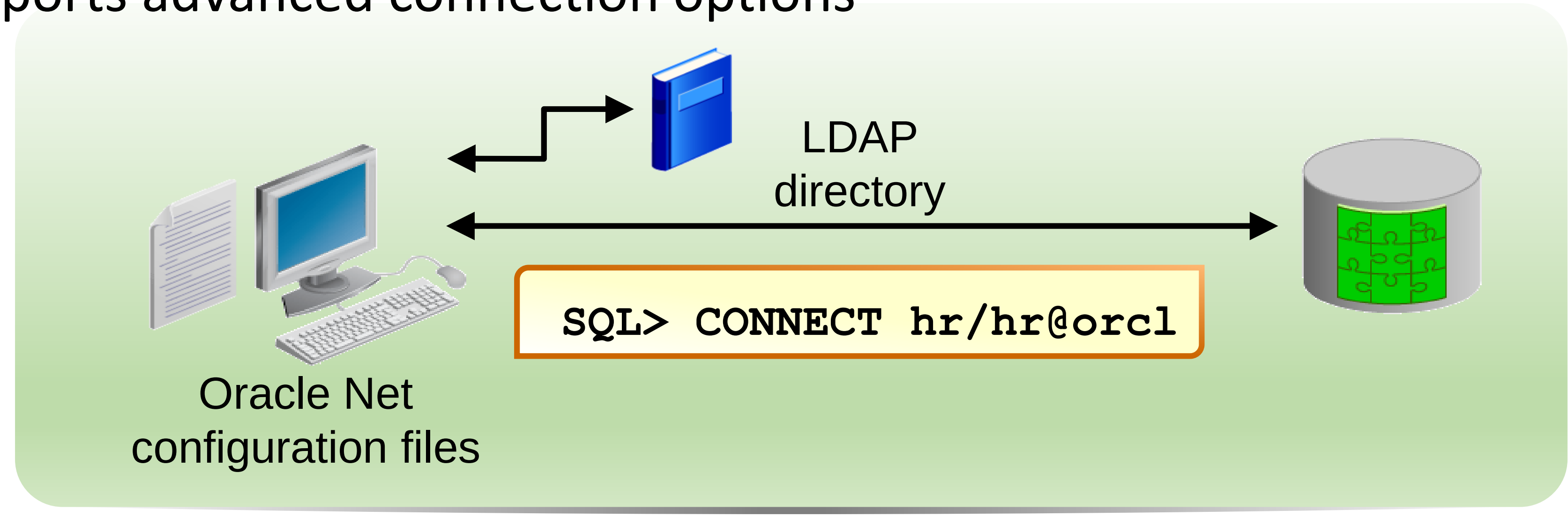
Local Naming

- 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



Tools for Configuring and Managing Oracle Net Services

- Enterprise Manager Net Services Administration page
- Oracle Net Manager
- Oracle Net Configuration Assistant
- Listener Control Utility

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`

Advanced Connection Options

- When a database service is accessible by multiple listener protocol addresses, you can specify the order in which the addresses are to be used.
- Oracle Net supports the following advanced connection options with local and directory naming:
  - Connect-time failover
  - Load balancing
  - Source routing

Testing Oracle Net Connectivity with `tnsping`

- The `tnsping` utility tests Oracle Net service aliases.
- It validates connectivity between a client and the Oracle Net Listener.
  - It validates that the host name, port, and protocol reach a listener.
  - It does not check whether the listener handles the service name.
  - It does not verify that the requested service is available.
- The `tnsping` utility also reveals the location of the configuration files. In a system with multiple `ORACLE_HOME` locations, this can be helpful.
- Examples:
  - `tnsping` supports Easy Connect names resolution:

    ```
    $tnsping host01.example.com:1521/orcl
    ```

  - `tnsping` also supports local and directory naming:

    ```
    $ tnsping orcl
    ```

Configuring Communication Between Database Instances

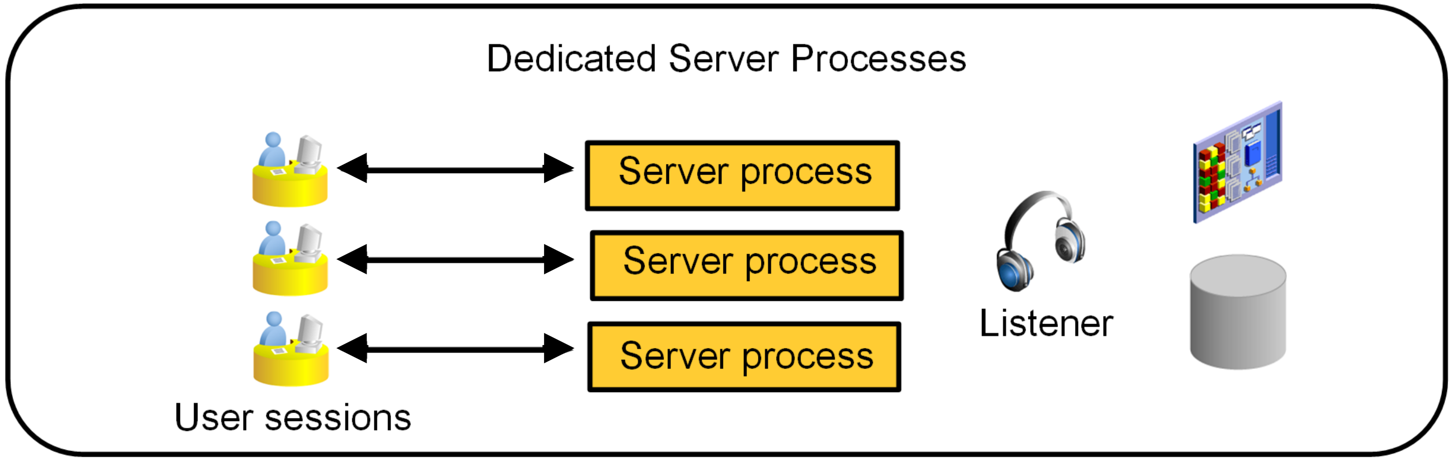
- You must configure network connectivity (for example, `tnsnames.ora`) and a database link for database instance to database instance communication.
- A database link is a schema object that enables you to access objects on a different database.
- SQL command to create a fixed user, private database link:

```
CREATE DATABASE LINK <database_link_name>
CONNECT TO <user> IDENTIFIED BY <pwd>
USING '<connect_string_for_remote_db>'
```

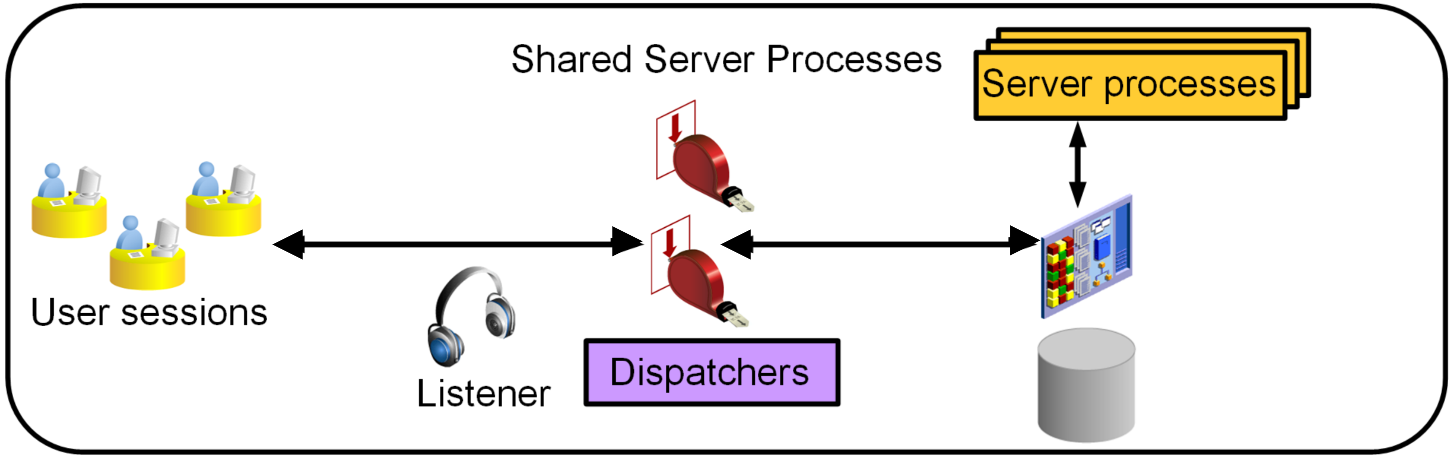
- SQL command to query a table by using the database link:

```
SELECT * FROM employees@<database_link_name>
```

Comparing Dedicated and Shared Server Configurations



One server process for each client



A small pool of server processes can serve a large number of clients

Summary

- In this lesson, you should have learned how to:
  - Describe Oracle Net Services
  - Explain how listeners work
  - Configure listeners for dynamic or static service registration
  - Configure local naming for database connections
  - Test Oracle Net connectivity with `tnsping`
  - Configure communication between databases by creating database links
  - Explain the difference between dedicated and shared server configurations

Practice 8: Overview

- 8-1: Exploring the Default Listener
- 8-2: Creating a Static Listener for a PDB
- 8-3: Verifying the Net Service Name for MYPDB1

Administering User Security

Objectives

- After completing this lesson, you should be able to:
  - Create database users
  - Grant privileges to database users
  - Create and grant roles to users or other roles
  - Revoke privileges and roles from users and other roles
  - Create and assign profiles to users
  - Explain the various authentication options for users
  - Assign quota to users
  - Apply the principle of least privilege

Oracle Cloud User Roles and Privileges

ACCOUNT ADMINISTRATORS
Manage the service account
- **Request trial subscriptions**
- **Activate services**

SERVICE ADMINISTRATORS
Manage service operations
- **Administer cloud services**
- **Assign roles to users**
- **Provide sign-in credentials and URLs to users**

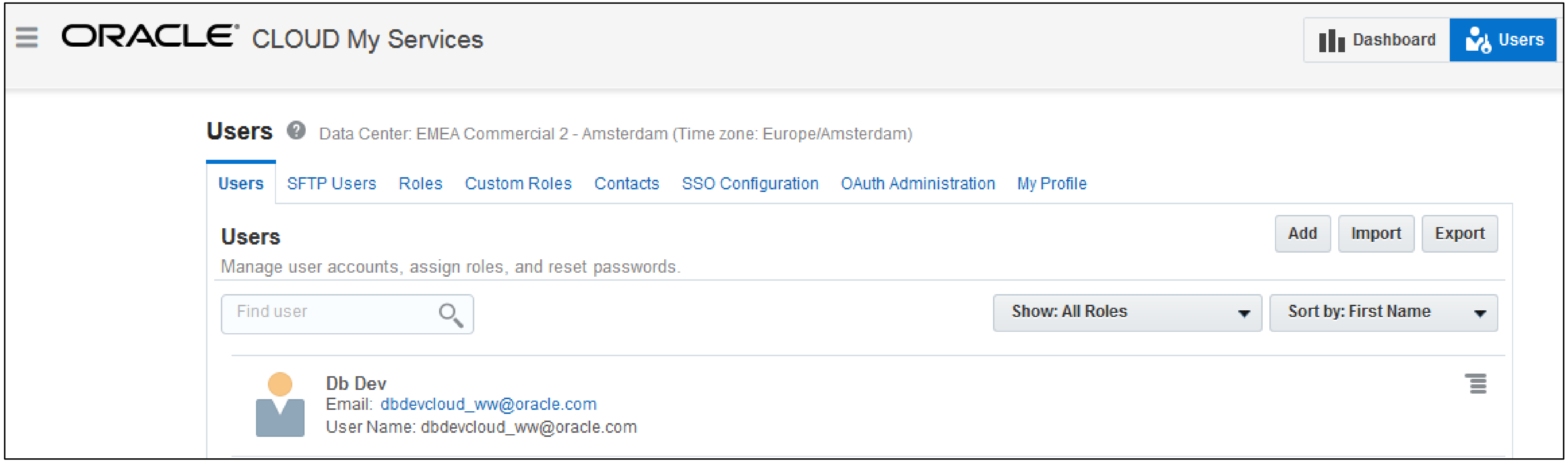
IDENTITY DOMAIN ADMINISTRATORS
Manage users and roles
- **Create user accounts**
- **Assign roles**
- **Reset passwords**

USERS IN A CLOUD SERVICE

See Oracle Cloud User Roles and Privileges in *Getting Started with Oracle Cloud* for additional information.

Administering Oracle Cloud Users, Roles, and Privileges

- Administer Cloud Services users by accessing the Users page.
- Cloud Services users are different from Oracle Database users.



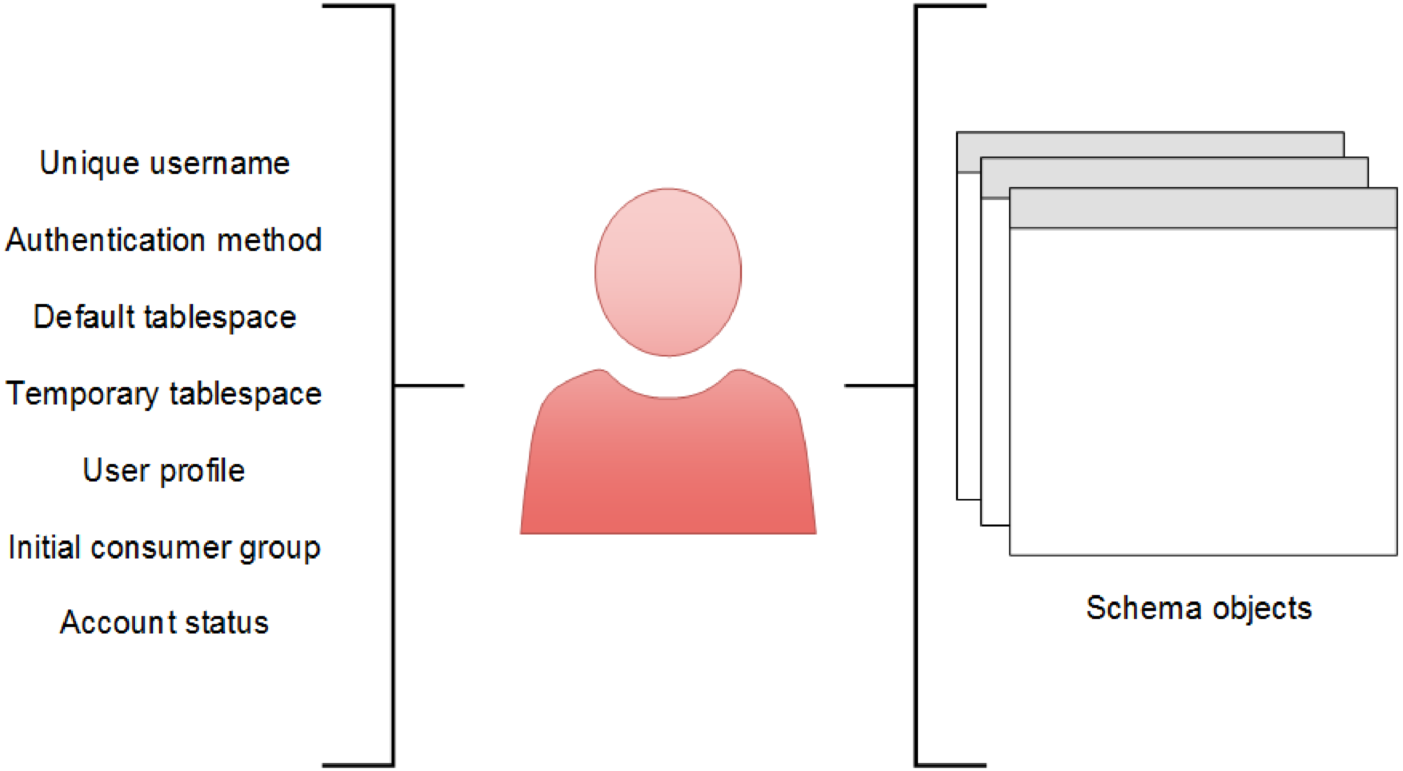
See Creating a User and Assigning a Role in *Getting Started with Oracle Cloud* for details.

Managing Oracle Cloud Compute Node Users

- When a database deployment is created, three Linux users are created.

OS User	Authorization
`opc`	Authorized to log in to the compute node Authorized to run `root` commands Can use `sudo -s`
`oracle`	Authorized to log in to the compute node Not authorized to run `root` commands
`root`	Not authorized to log in to the compute node

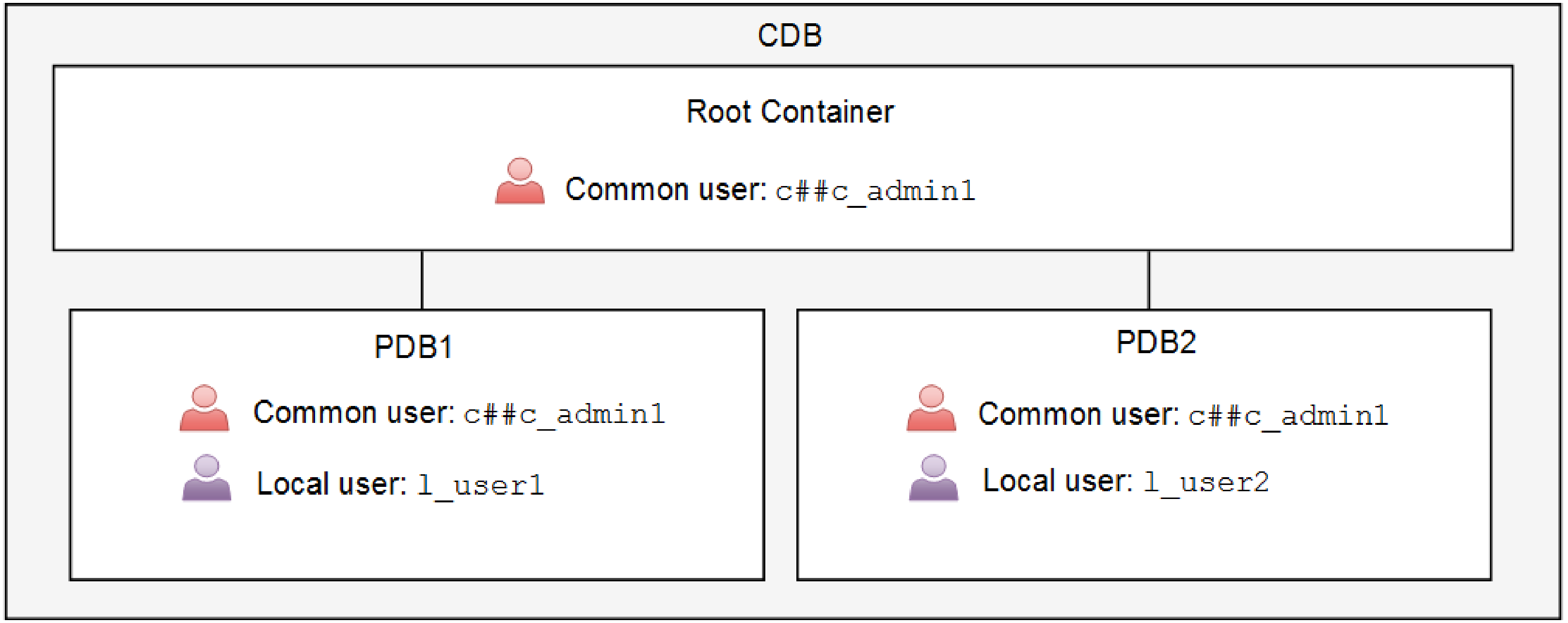
Database User Accounts



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



Schema-Only Account

Ensure that a user cannot log in to the instance:

- Enforce data access through the application.
- Secure schema objects.
  - Prevent objects from being dropped by the connected schema.
- Use the `NO AUTHENTICATION` clause.
  – Can be replaced by `IDENTIFIED BY VALUES`
- A schema-only account cannot be:
  - Granted system administrative privileges
  - Used in database links

```
DBA_USERS
  AUTHENTICATION_TYPE = NONE | PASSWORD
```

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).

Password Authentication

- Password authentication is also referred to as "authentication" by the Oracle Database server.
- 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.
  - If you decide on expiring user passwords, make sure that users have the ability to change the password. Some applications do not have this functionality.
  - 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.

Password File Authentication

- You can use password file authentication for an Oracle database instance and for an Oracle Automatic Storage Management (Oracle ASM) instance.
- If authentication succeeds, the connection is logged with the `SYS` user.
- A password file stores database usernames and case-sensitive passwords for administrator users (common and local administrators).
- DBCA creates a password file during installation.
- To prepare for password file authentication, you must:
  - Create the password file.
  - Set the `REMOTE_LOGIN_PASSWORDFILE` initialization parameter.
  - Grant system administrative privileges (for example, `GRANT SYSDBA TO mydba`).
- Use the `CONNECT` command in SQL*Plus to connect. For example:

```
SQL> CONNECT mydba AS SYSDBA
```

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
```

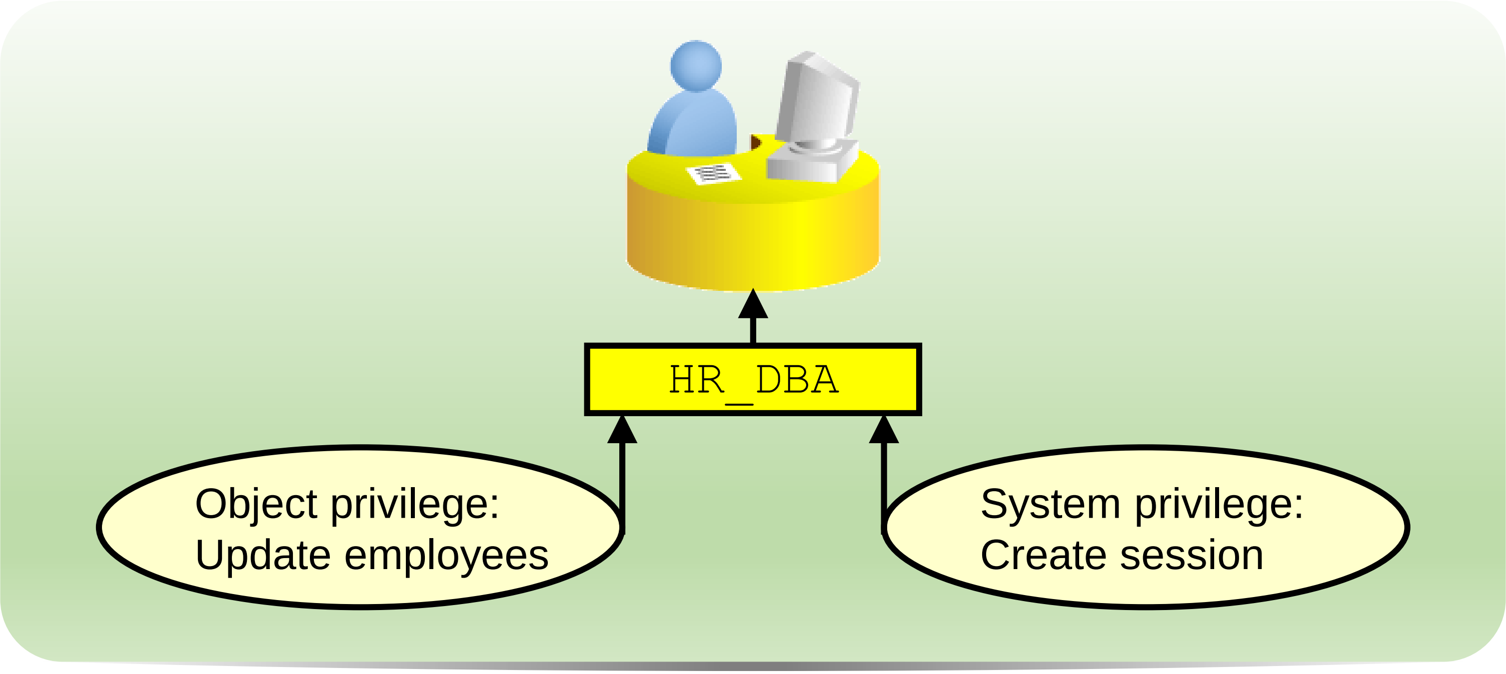
- If you are not a member of one of these OS groups, you will not be able to connect as an administrative user via OS authentication.

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`

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, 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

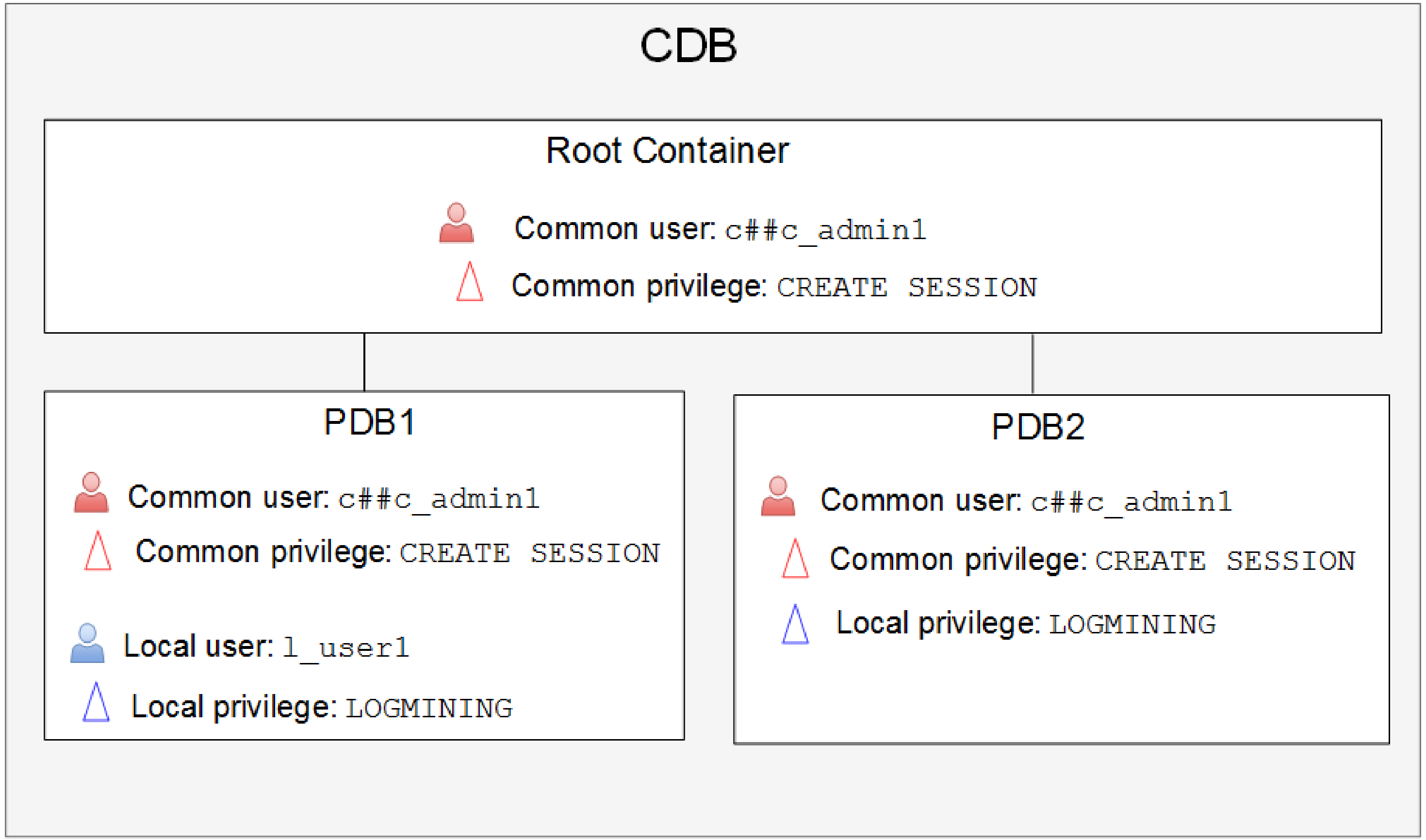
Privilege	Description
SYSDBA	Perform all administrative tasks in the database, including create and drop a database, open and mount a database, start up and shut down an Oracle database, create an SPFILE, put a database in or remove a database from ARCHIVELOG mode, perform incomplete recovery operations, 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

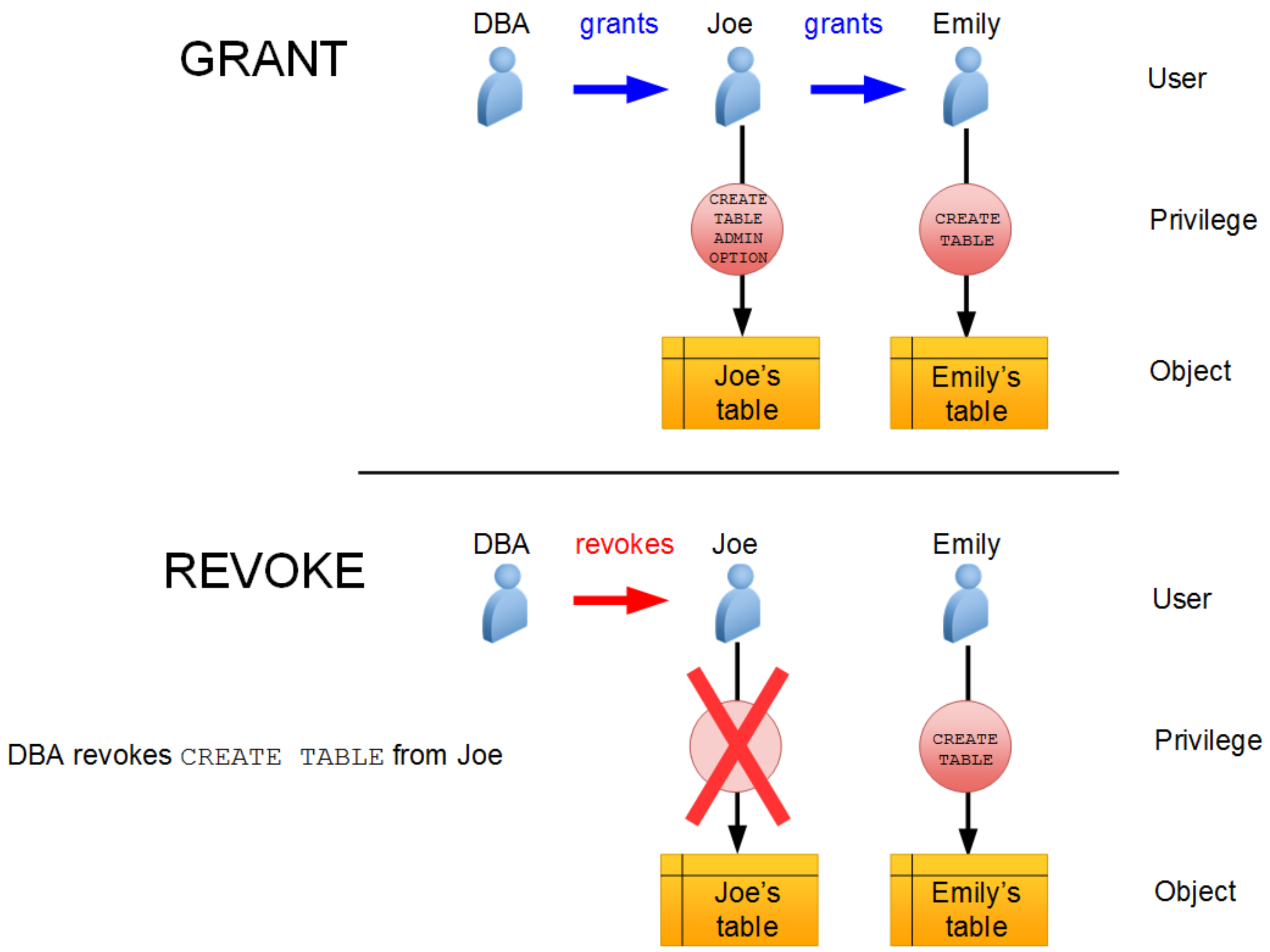
- Object privileges allow a user to perform a particular action on a specific object, such as a table, view, sequence, procedure, function, or package.
- Without specific permission, users can access only their own objects.
- Object privileges can be granted by the owner of an object, by the administrator, or by 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]
```

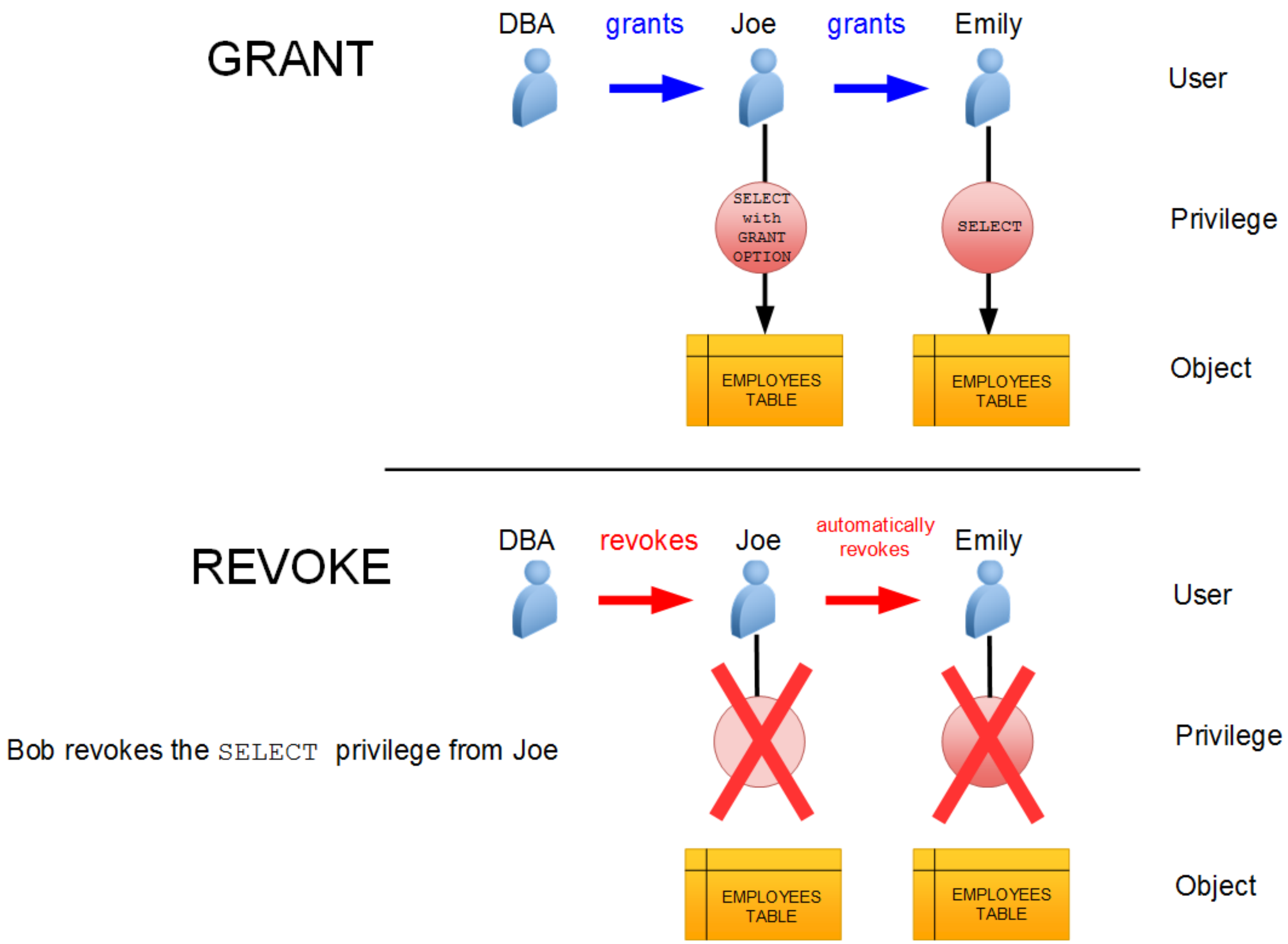
Granting Privileges in a Multitenant Environment



Granting and Revoking System Privileges



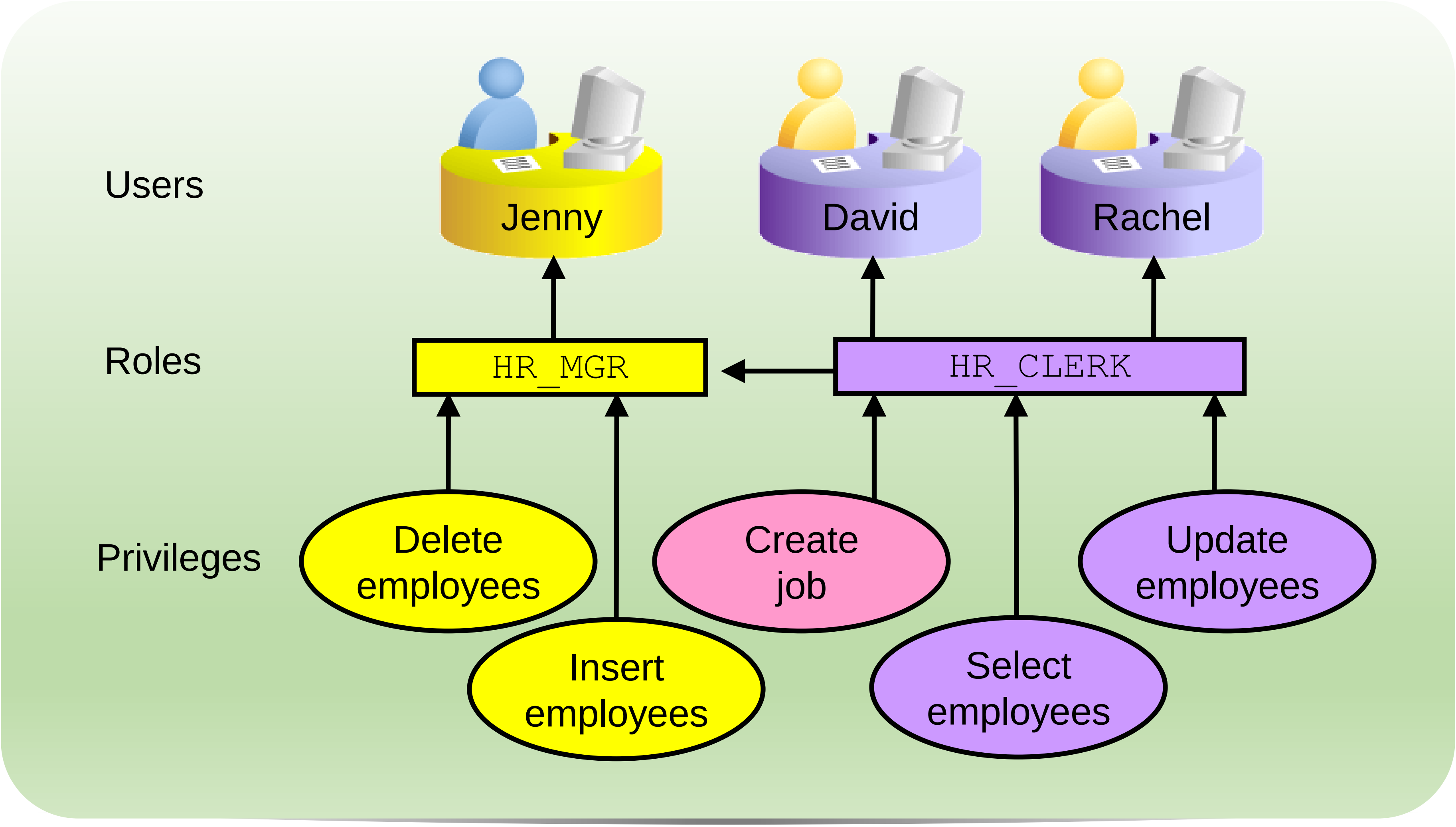
Granting and Revoking Object Privileges



Using Roles to Manage Privileges

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

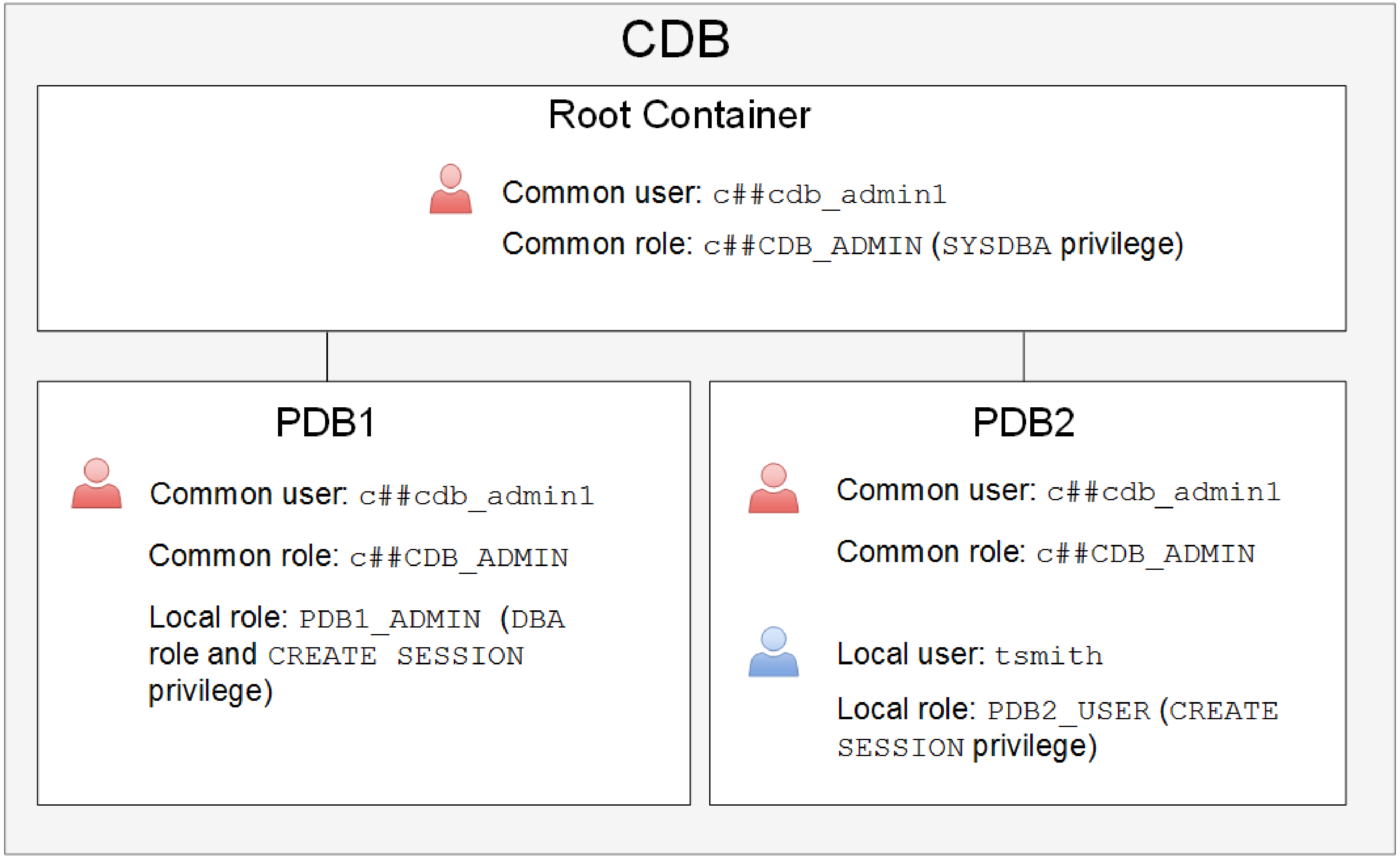
Assigning Privileges to Roles and Assigning Roles to Users



Oracle-Supplied Roles

Account	Description
DBA	Includes most system privileges and several other roles. Do not grant this role to nonadministrators. Users with this role can connect to the CDB or PDB only when it is open.
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

Creating and Granting Roles



Assigning Roles

- To assign (grant) a role to a user or another role by using SQL*Plus, use the `GRANT` command.
- There are two ways to grant a role in a multitenant architecture:
  - 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>;
```

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

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` before profiles can impose resource limitations.

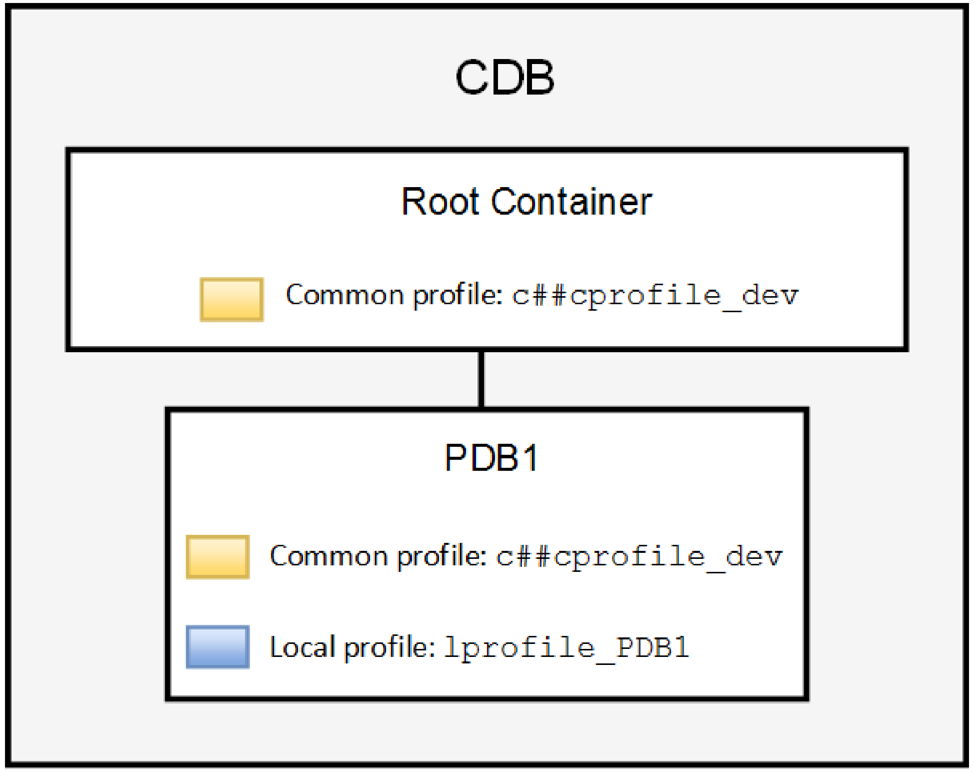
Creating Profiles in a Multitenant Architecture

- Common profile:

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

- Local profile:

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



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 that you can create by executing the `utlpwdmg.sql` script:
  - `ORA12c_VERIFY_FUNCTION`
  - `ORA12c_STRONG_VERIFY_FUNCTION`
  - `VERIFY_FUNCTION_11g`
- These functions must be owned by the `SYS` user.
- Password complexity checking is not enforced for the `SYS` user.

Assigning Profiles

- There are two ways to assign a profile:
  - 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>;
```

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 he drops objects (with the `PURGE` clause) or purges his objects in the recycle bin.

Assigning Quotas

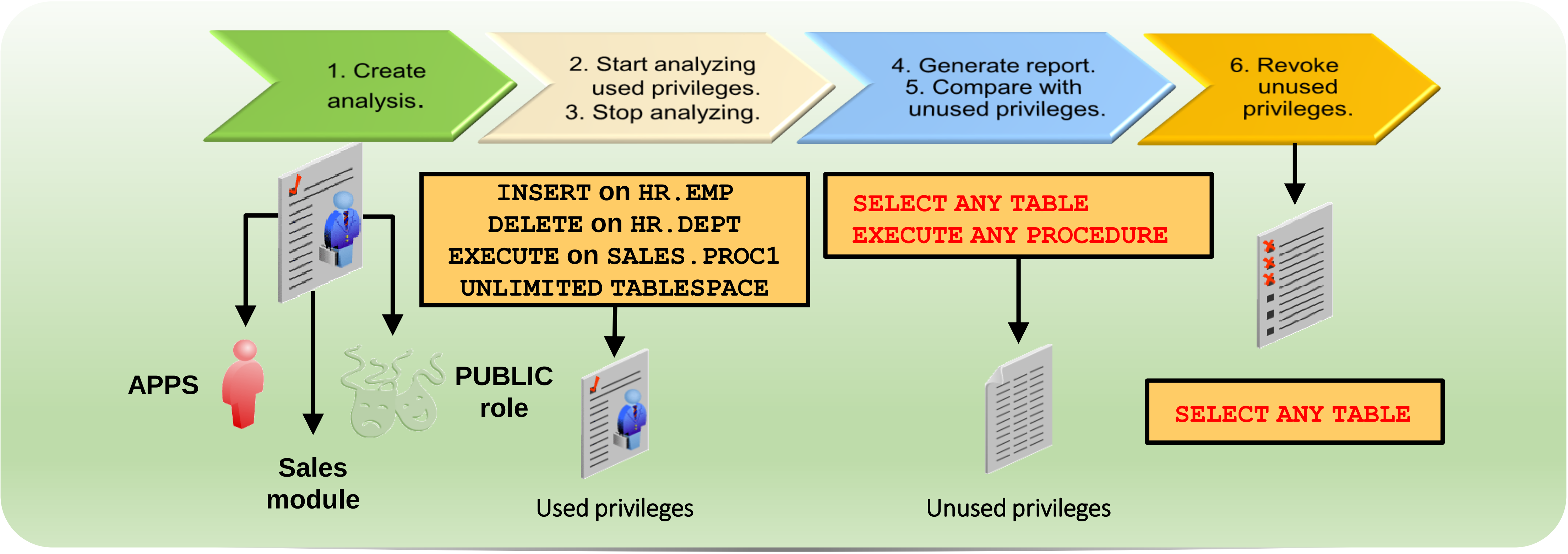
- You have three options for providing quota for a user on a tablespace:
  - `UNLIMITED`
  - Value
  - `UNLIMITED TABLESPACE` system privilege

Applying the Principle of Least Privilege

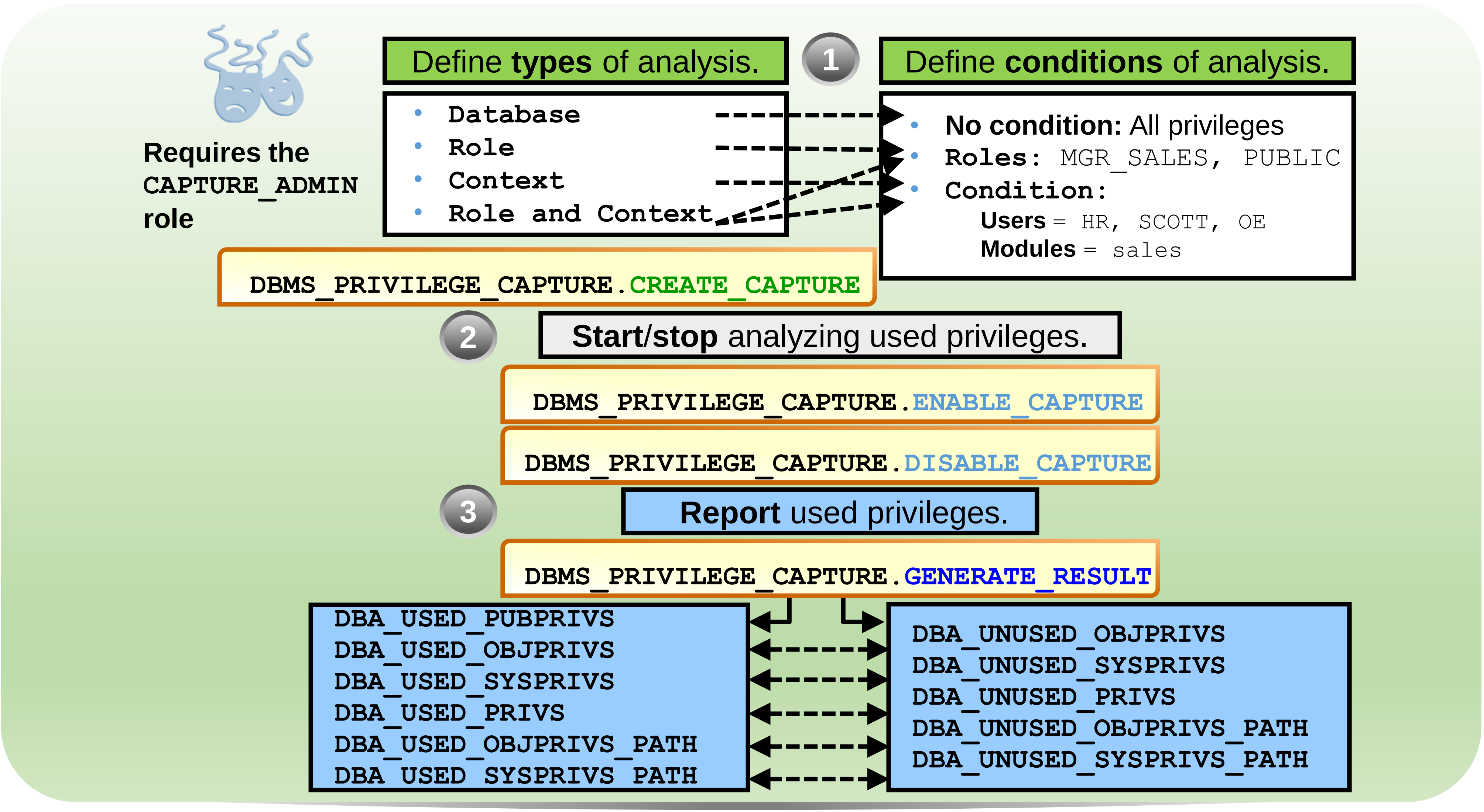
- The principle of least privilege means that a user must be given only those privileges that are required to efficiently complete a task.
- This reduces the chances of users modifying or viewing data (either accidentally or maliciously) that they do not have the privilege to modify or view.
- Ways to apply the principle of least privilege:
  - Protect the data dictionary
  - Revoke unnecessary privileges from `PUBLIC`
  - Use access control lists (ACLs) to control network access
  - Restrict access to OS directories
  - Limit users with administrative privileges
  - Restrict remote database authentication
  - Enable unified auditing

Privilege Analysis

- Analyze used privileges to revoke unnecessary privileges.
- Use the `DBMS_PRIVILEGE_CAPTURE` package.



Privilege Analysis Flow



Summary

- In this lesson, you should have learned how to:
  - Create database users
  - Grant privileges to database users
  - Create and grant roles to users or other roles
  - Revoke privileges and roles from users and other roles
  - Create and assign profiles to users
  - Explain the various authentication options for users
  - Assign quota to users
  - Apply the principle of least privilege

Practice 9: Overview

- 9-1: Creating Common and Local Users
- 9-2: Creating a Local User for an Application
- 9-3: Granting a Local Role (DBA) to PDBADMIN
- 9-4: Using EM Express to Create a Local Profile
- 9-5: Using EM Express to Create Local Roles
- 9-6: Using EM Express to Create Local Users
- 9-7: Configuring a Default Role for a User
- 9-8: Exploring OS and Password File Authentication

Creating PDBs

Objectives

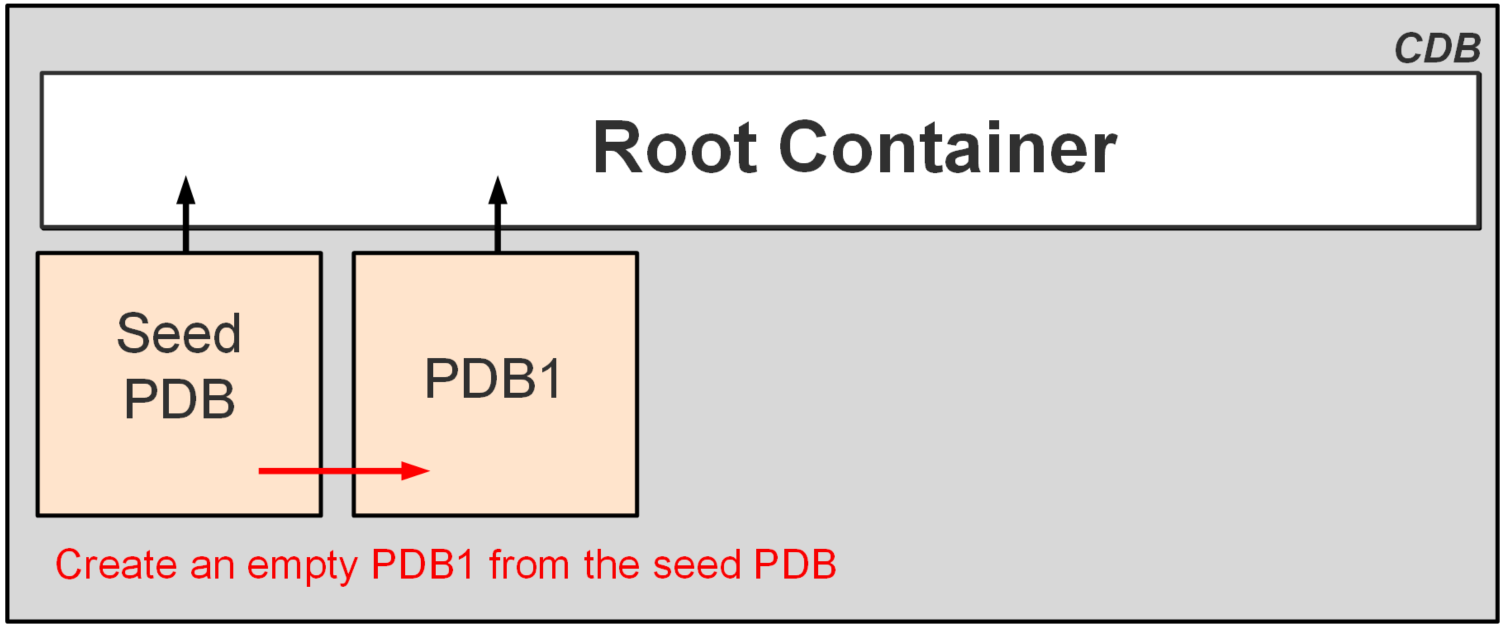
- After completing this lesson, you should be able to:
  - Describe the methods and tools used to create PDBs
  - Create PDBs from seed by using SQL*Plus
  - Clone PDBs by using SQL*Plus
  - Unplug and plug in PDBs by using SQL*Plus
  - Drop PDBs by using SQL*Plus

Methods and Tools to Create PDBs

- Methods to create PDBs:
  - Create a PDB by using the seed
  - Create a PDB from a non-CDB
  - Clone an existing PDB or non-CDB
  - Plug an unplugged PDB into a different CDB
  - Relocate a PDB to a different CDB
  - Create a PDB as a proxy PDB
- Tools to create PDBs:
  - SQL*Plus
  - SQL Developer
  - Enterprise Manager Cloud Control
  - DBCA—Create a PDB from seed or by using the unplug/plug method.

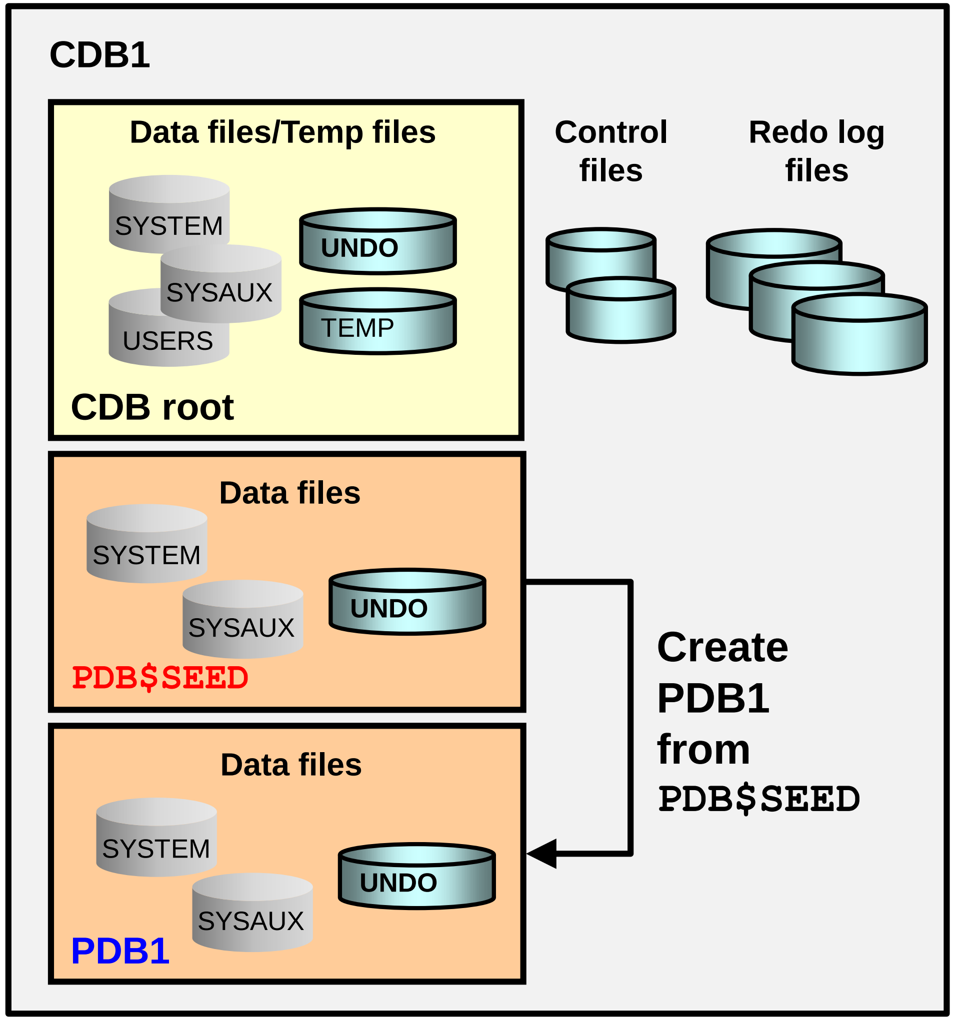
Creating PDBs from Seed

- You can create a new empty PDB by using the seed PDB as a template.
- Every CDB has a seed PDB.
- To create a PDB from seed with SQL*Plus, use the `CREATE PLUGGABLE DATABASE` statement.



Creating a New PDB from PDB$SEED

- Copies the data files from `PDB$SEED` data files
- Creates the `SYSTEM`, `SYSAUX`, and `UNDO` tablespaces
- 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



Examples: Creating a PDB from Seed

- Create a new PDB from the seed by using `FILE_NAME_CONVERT`:

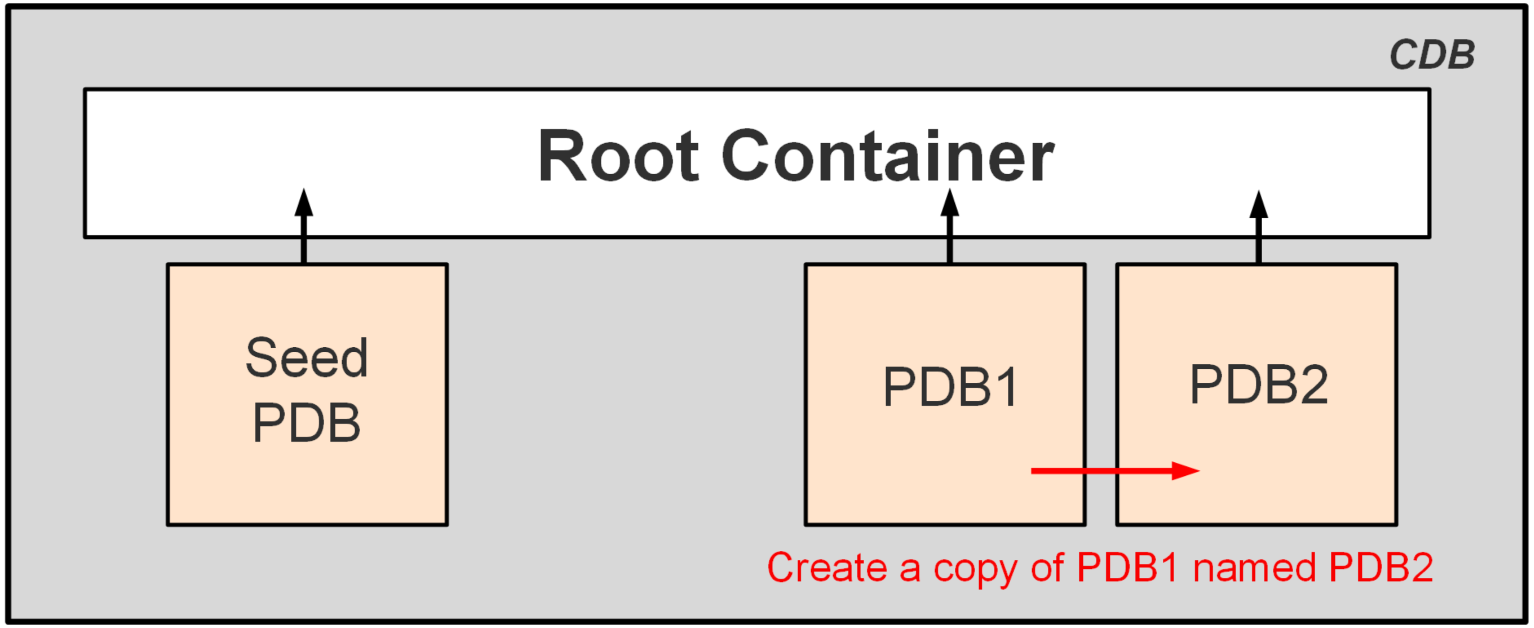
```
SQL> CREATE PLUGGABLE DATABASE pdb1
     ADMIN USER admin1 IDENTIFIED BY p1 ROLES=(CONNECT)
     FILE_NAME_CONVERT = ('PDB$SEEDdir', 'PDB1dir');
```

- Set the `PDB_FILE_NAME_CONVERT` initialization parameter and then create the PDB:

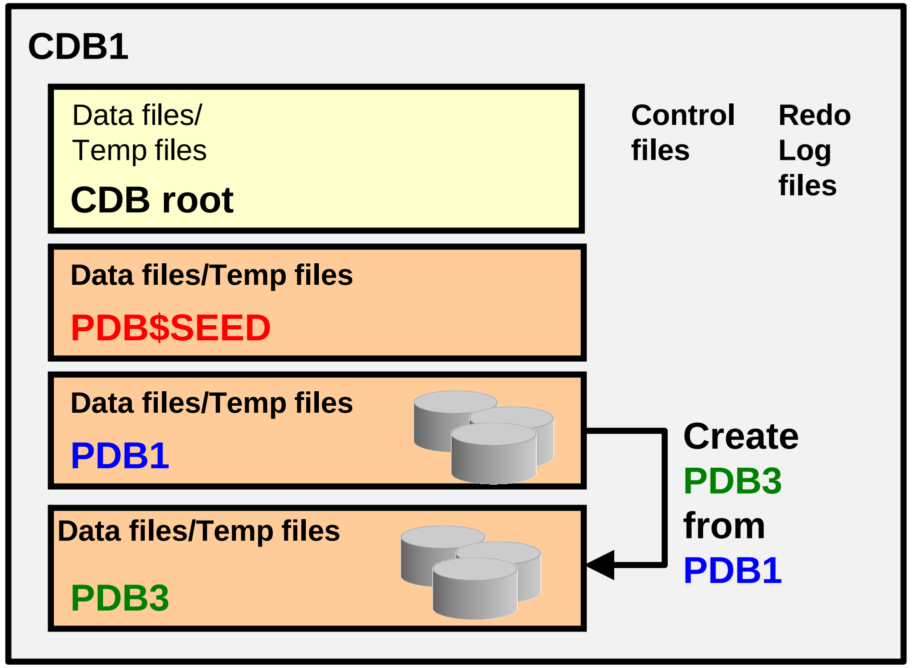
```
SQL> CREATE PLUGGABLE DATABASE pdb1
     ADMIN USER pdb1_admin IDENTIFIED BY p1 ROLES=(CONNECT);
```

Cloning PDBs

- Cloning is copying a source PDB from a CDB and plugging the copy into the same CDB or another CDB.
- Example: PDB1 is cloned as PDB2 in the same CDB. The seed PDB, while present in the CDB, is not used.



Cloning Regular PDBs



PDB3 owns:

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

1. Define how Oracle will find the location of the data files:
   - In `init.ora`, set `DB_CREATE_FILE_DEST='PDB3dir'`
   - In `init.ora`, set `PDB_FILE_NAME_CONVERT='PDB1dir', 'PDB3dir'`
   - Using the `CREATE_FILE_DEST='PDB3dir'` clause
2. Connect to the CDB root to close **PDB1**.
3. Clone **PDB3** from **PDB1**.
4.

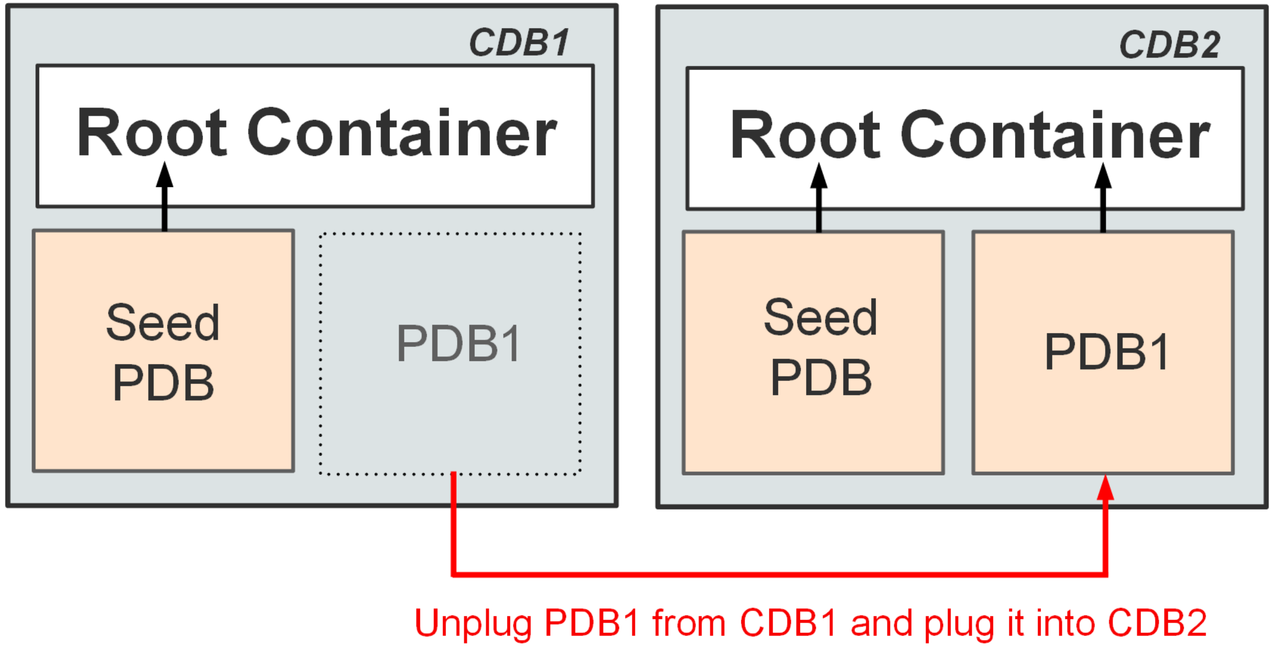
```
SQL> CREATE PLUGGABLE DATABASE pdb3 FROM pdb1
     CREATE_FILE_DEST = 'PDB3dir';
```

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

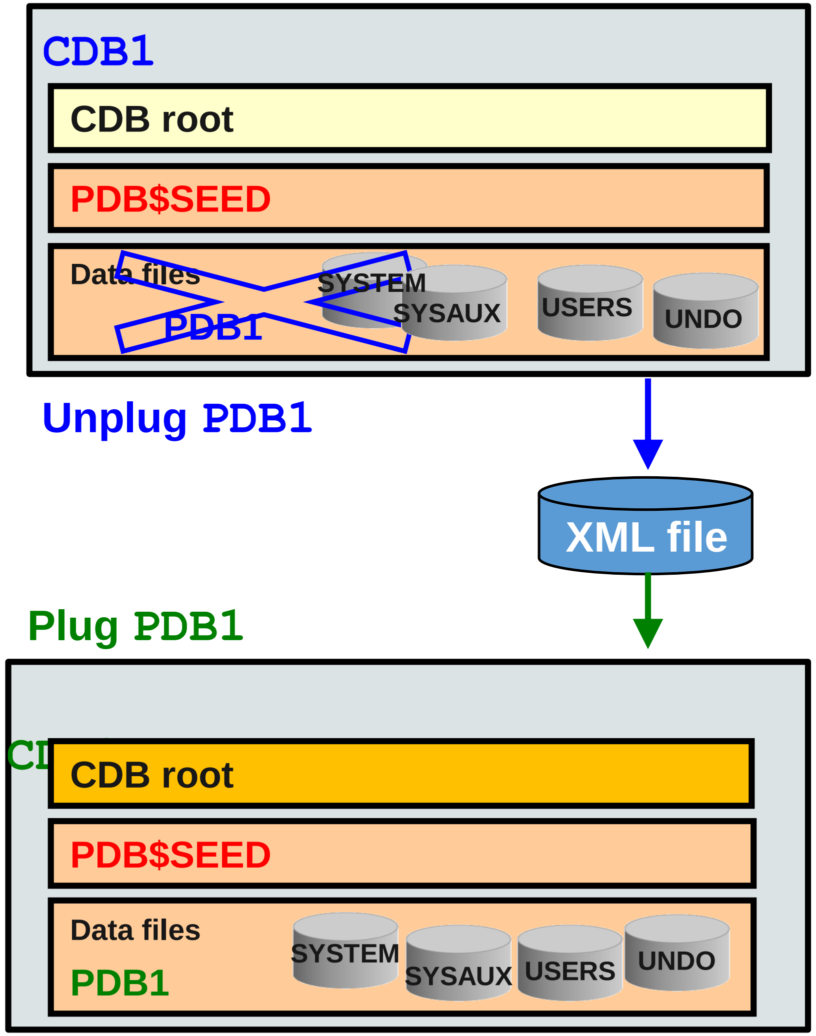
Note: Cloning metadata only with `NO DATA`

Unplugging and Plugging in PDBs

- Unplugging a PDB is disassociating the PDB from its CDB.
- Plugging in a PDB is associating a PDB with a CDB.
- You can plug a PDB into the same or another CDB.
- Example: PDB1 is unplugged from CDB1 and plugged into CDB2.



Plugging an Unplugged Regular PDB into a 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**:

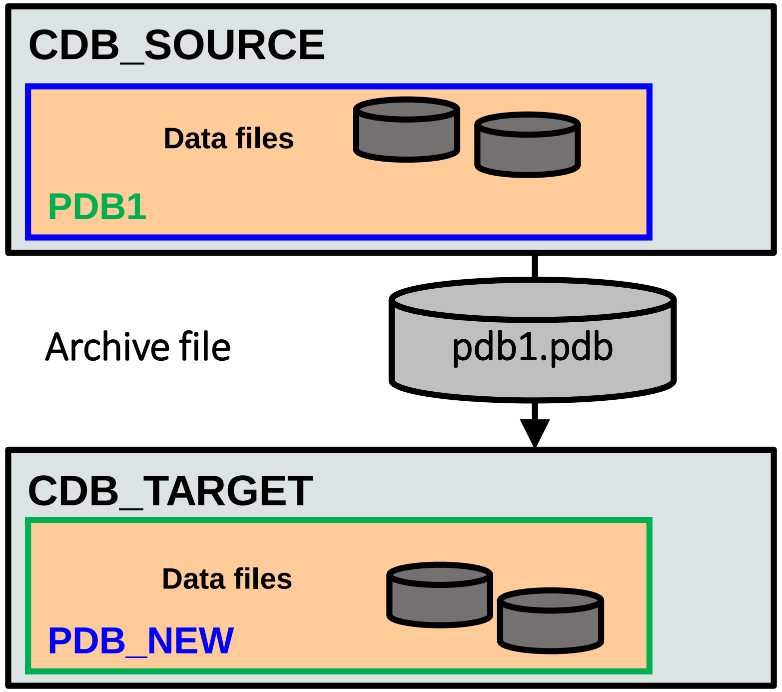
1. Connect to **CDB2** as a common user.
2. Use the `DBMS_PDB` package to check the compatibility of **PDB1** with **CDB2**.

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

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

Plugging Using an Archive File

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




```
SQL> ALTER PLUGGABLE DATABASE pdb1
        UNPLUG INTO '/tmp/pdb1.pdb';
```

```
SQL> CREATE PLUGGABLE DATABASE pdb_new
        USING '/tmp/pdb1.pdb';
```

Dropping PDBs

- When you drop a PDB, you remove all references to it and its data files in the control file of the CDB.
- Archived logs and backups associated with the dropped PDB are not deleted in case you later want to recover the PDB.
- You can also use Oracle Recovery Manager (RMAN) to delete archived logs and backups.
- You use the `DROP PLUGGABLE DATABASE` statement to drop a pluggable database (PDB).
  - Example:

```
SQL> DROP PLUGGABLE DATABASE SALESPDB INCLUDING DATAFILES;
```

Summary

- In this lesson, you should have learned how to:
  - Describe the methods and tools used to create PDBs
  - Create PDBs from seed by using SQL*Plus
  - Clone PDBs by using SQL*Plus
  - Unplug and plug in PDBs by using SQL*Plus
  - Drop PDBs by using SQL*Plus

Practice 10: Overview

- 10-1: Creating a PDB from Seed
- 10-2: Cloning a PDB
- 10-3: Unplugging and Plugging in a PDB
- 10-4: Dropping a PDB

Creating Master Encryption Keys for PDBs

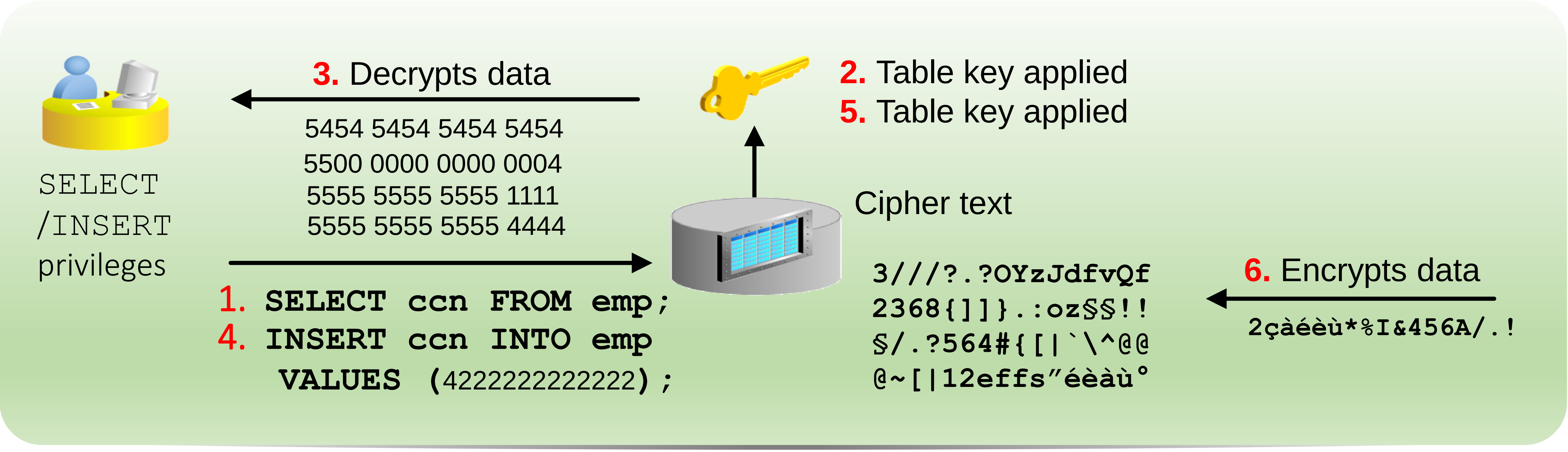
Objectives

- After completing this lesson, you should be able to:
  - Describe the implementation of master encryption keys for PDBs
  - Create and activate a master encryption key for a new PDB

Encryption in Database Cloud Service

- Database Cloud Service databases include a key management framework that stores and manages keys and credentials used to encrypt data in the database data files and in backups.
- The key management framework includes:
  - The keystore (referred to as a wallet in Oracle Database 11*g* and previous releases) to securely store Transparent Data Encryption (TDE) master encryption keys
  - The management framework to securely and efficiently manage the keystore and key operations for various database components

Transparent Data Encryption (TDE): Overview

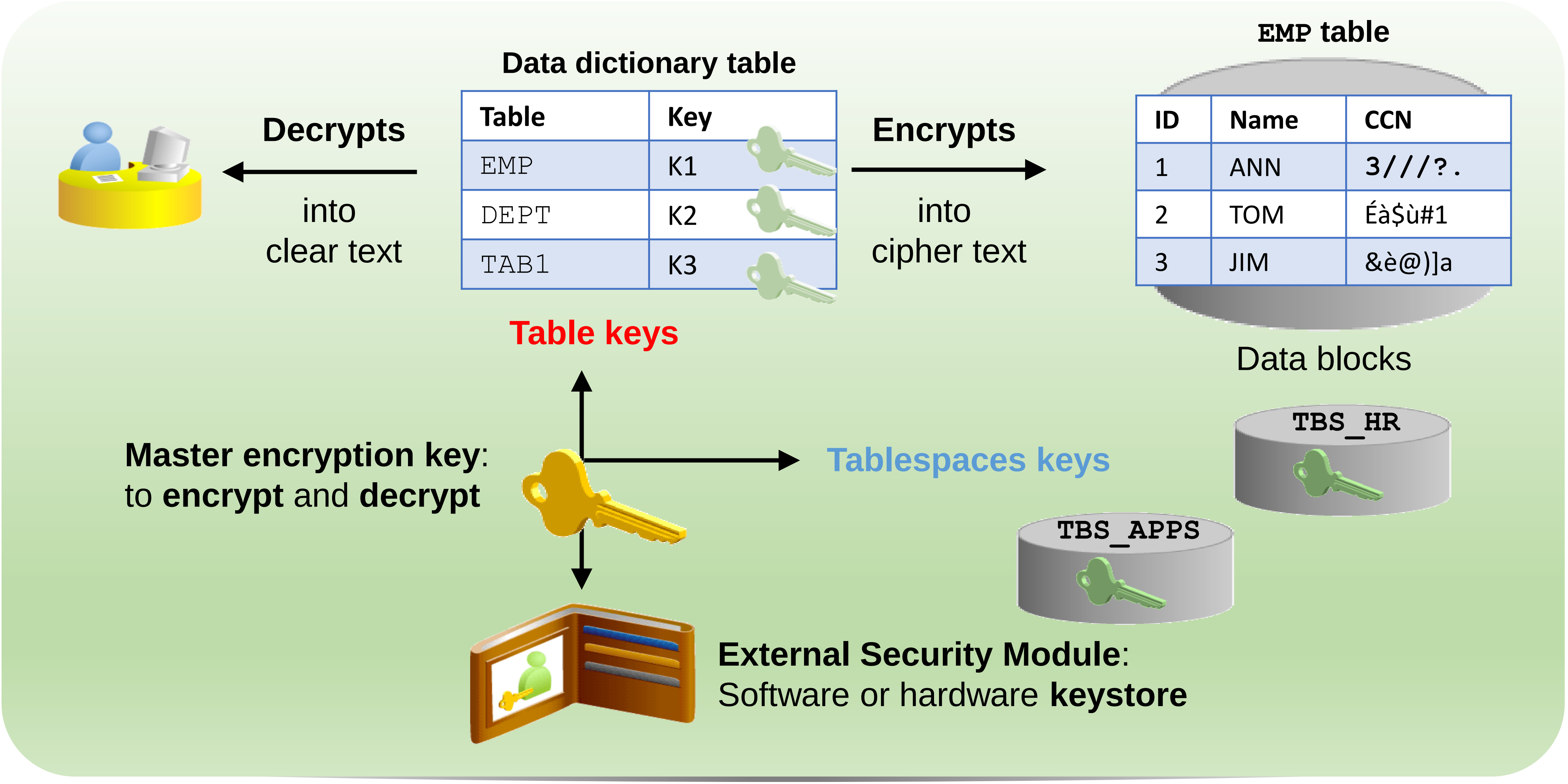


- Encrypts data in data files, redo log files, archived redo log files, and backup files
- Encrypts data in memory (only for column encryption)
- Manages keys automatically
- Does not require application changes

Components of TDE

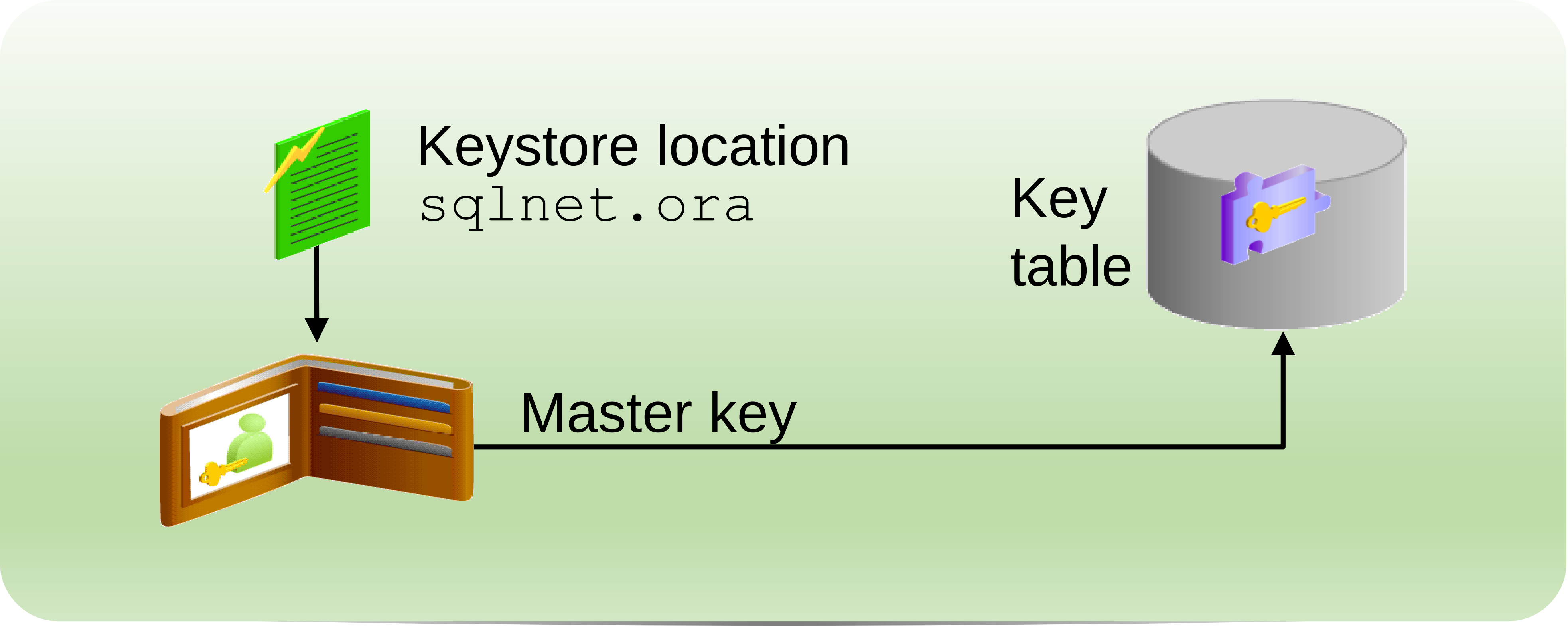
- Key architecture
  - Two-tier architecture: A unified master encryption key stored in an external security module is used to encrypt the table key or tablespace key.
  - Low overhead re-key operation: Some security regulations require periodical changes of encryption keys.
- External security module
  - Software keystore
  - Hardware keystore (HSM)
- Algorithm support

Using TDE


Data dictionary table
Table | Key
EMP | K1
DEPT | K2
TAB1 | K3
Decrypts
into
clear text
Encrypts
into
cipher text
EMP table
ID | Name | CCN
1 | ANN | 3///?.
2 | TOM | Éà$ù#1
3 | JIM | &è@)]a
Data blocks
Table keys
Master encryption key:
to encrypt and decrypt
Tablespaces keys
TBS_HR
TBS_APPS
External Security Module:
Software or hardware keystore

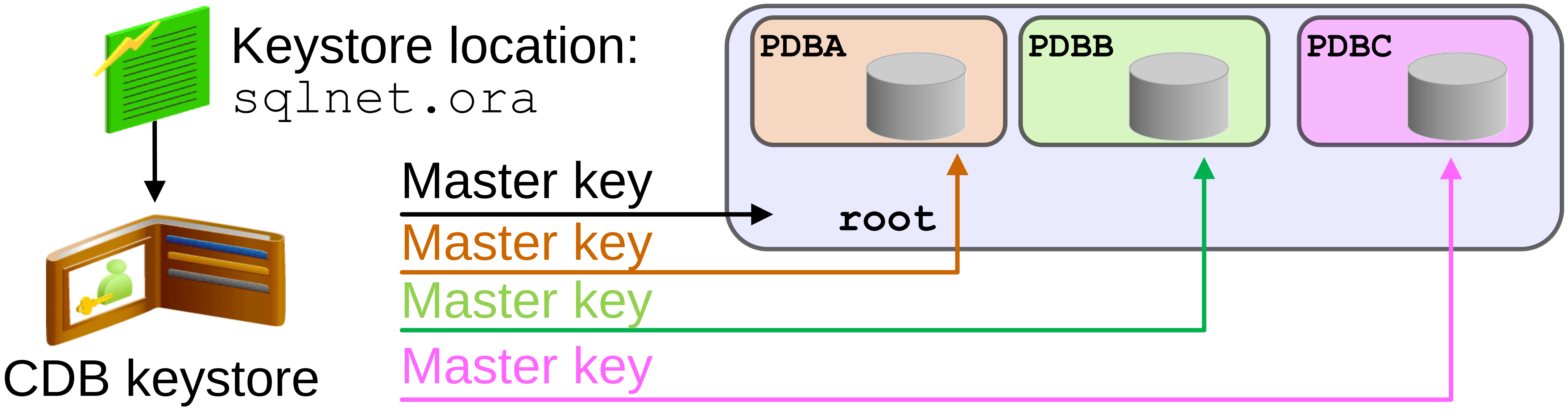
Defining the Keystore Location

- The location of the keystore file is specified in an entry in the `$ORACLE_HOME/network/admin/sqlnet.ora` file.



CDB and PDB Master Encryption Keys

- There is one master encryption key per PDB to encrypt PDB data.
- The master key must be transported from the source database keystore to the target database keystore when a PDB is moved from one host to another.
- After creating or plugging in a new PDB, you must create and activate a master encryption key for the PDB.



Do You Need to Create and Activate a Master Encryption Key?

1. Set the container to the PDB.

2. Query the `STATUS` column in `V$ENCRYPTION_WALLET`.

```
SQL> ALTER SESSION SET CONTAINER = pdb;
```

3. If `STATUS` contains a value of `OPEN_NO_MASTER_KEY`, create and activate the master encryption key (shown in the next slide).

```
SQL> SELECT wrl_parameter, status, wallet_type
  2  FROM v$encryption_wallet;
```

Creating and Activating a Master Encryption Key

1. Close the auto-login keystore in the root container and then reopen it as a password keystore.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE close;
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE open
  2  IDENTIFIED BY keystore-password CONTAINER = all;
```

2. Set the container to the PDB. Create and activate a master encryption key in the PDB.

```
SQL> ADMINISTER KEY MANAGEMENT SET KEY USING TAG 'tag'
  2  IDENTIFIED BY keystore-password
  3  WITH BACKUP USING 'backup_identifier';
```

3. Query `V$ENCRYPTION_WALLET` again, verifying that `STATUS` is `OPEN`.

```
SQL> SELECT wrl_parameter, status, wallet_type
  2  FROM v$encryption_wallet;
```

Summary

- In this lesson, you should have learned how to:
  - Describe the implementation of master encryption keys for PDBs
  - Create and activate a master encryption key for a new PDB

Practice 11: Overview

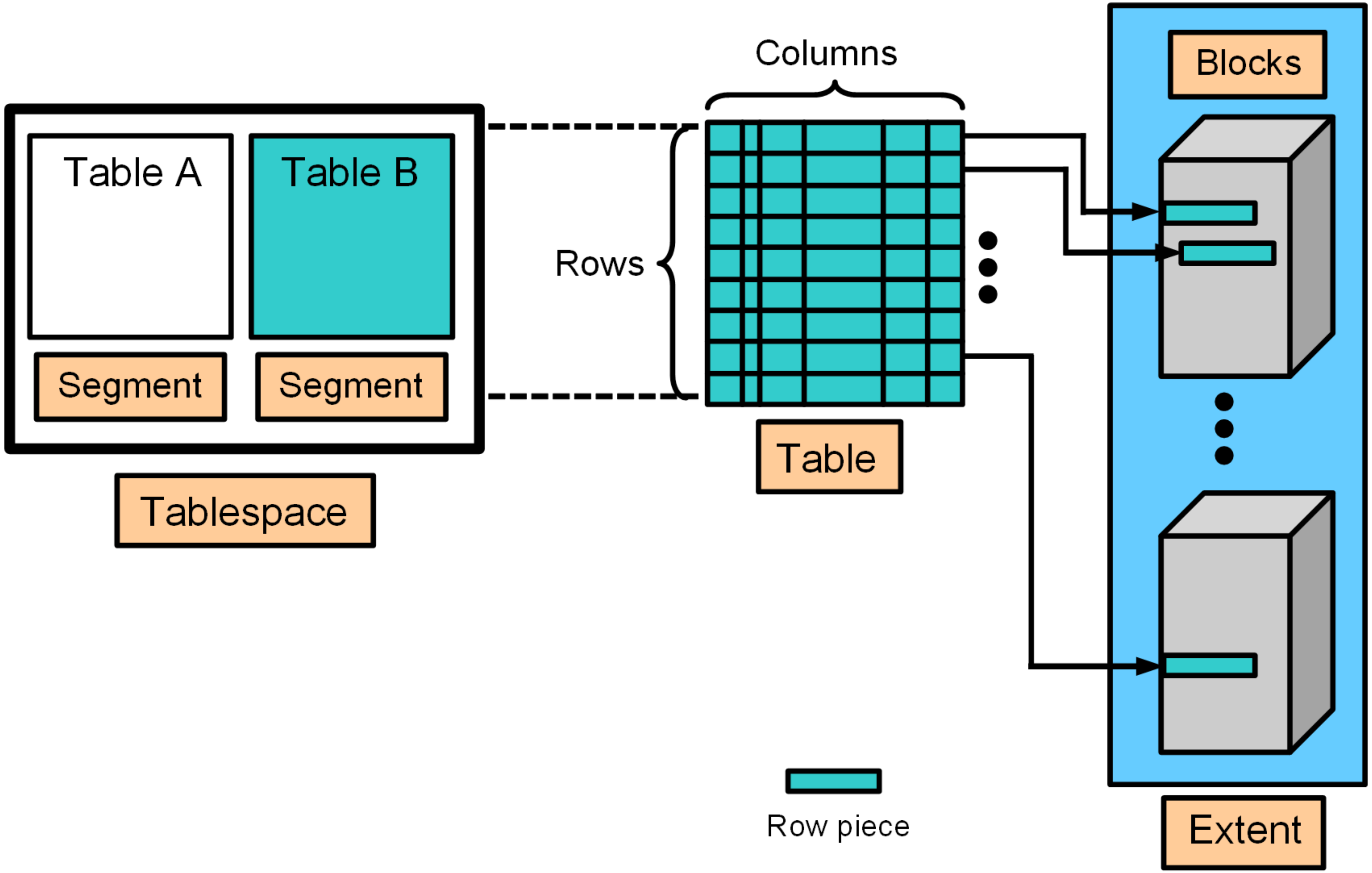
- 11-1: Creating and Activating an Encryption Key
- 11-2: Creating and Activating the Encryption Key for PDB2

Creating and Managing Tablespaces

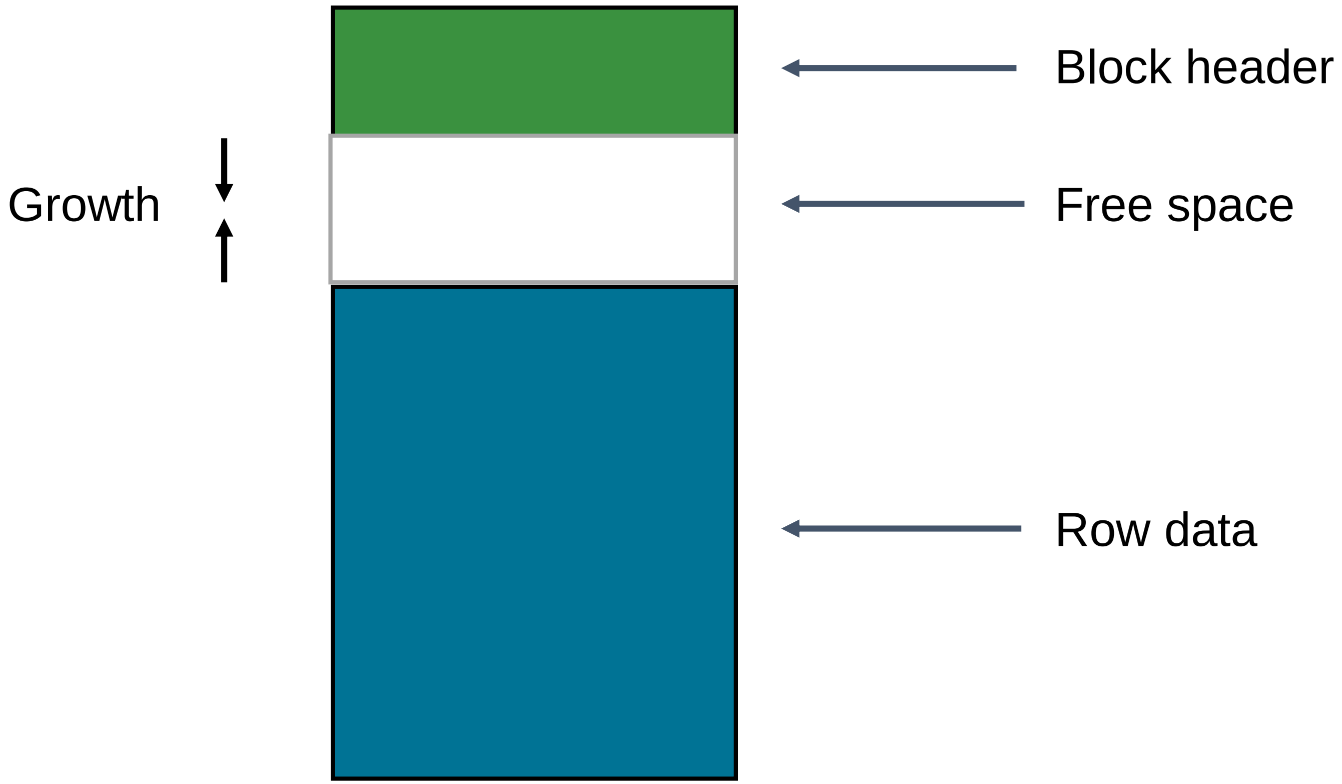
Objectives

- After completing this lesson, you should be able to:
  - Explain how table data is stored in the database
  - Use SQL*Plus to:
    - Create and drop tablespaces
    - Alter tablespaces
    - View tablespace information
  - Implement Oracle Managed Files (OMF)
  - Use SQL*Plus to move and rename online data files
  - Implement tablespace encryption

How Table Data Is Stored



Database Block Content



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 a graphical interface, such as EM Express.
- 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 if you are running your database in automatic undo management mode. Oracle strongly recommends that you use automatic undo management mode rather than using rollback segments for undo.
  - Temporary tablespace: Contains schema objects only for the duration of a session. Objects in temporary tablespaces are stored in temp files.

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;
```

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

Review: Implementing Oracle Managed Files (OMF)

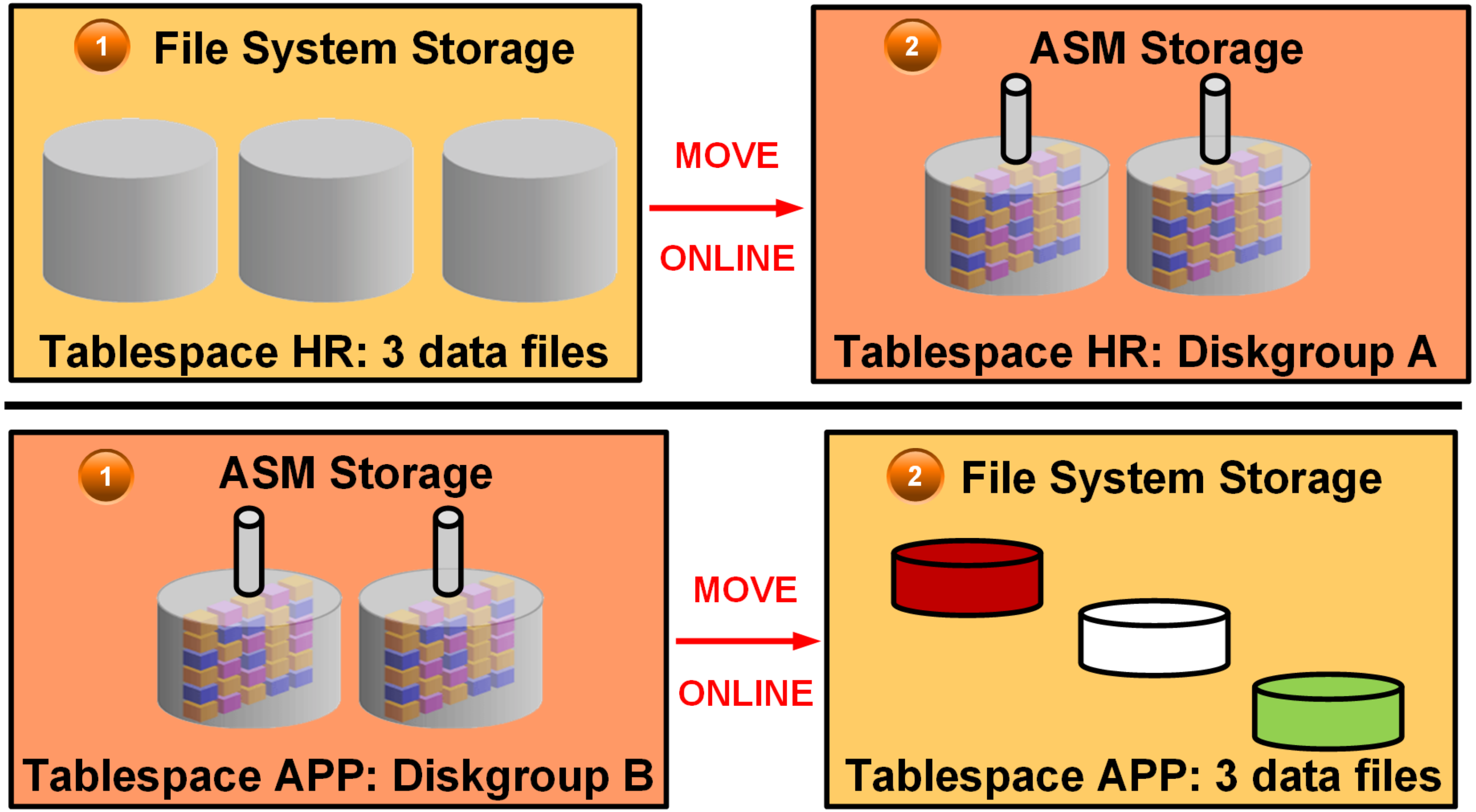
- Specify file operations in terms of database objects rather than file names.

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;
```

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';
```

Tablespace Encryption by Default in DBCS

- In Oracle Database Cloud Service, user-created tablespaces are encrypted by default.
- Tablespaces created when the database is first created (in the root container, PDB seed, and `PDB1`) are NOT encrypted.
- The default encryption algorithm is AES128.
- The underlying architecture supporting this feature is Transparent Data Encryption (TDE).

Controlling Tablespace Encryption by Default

Parameter Value	Description
ALWAYS	Any tablespace created will be transparently encrypted with the AES128 algorithm unless a different algorithm is specified in the ENCRYPTION clause.
CLOUD_ONLY (Default value)	Tablespaces created in a Database Cloud Service database will be transparently encrypted with the AES128 algorithm unless a different algorithm is specified in the ENCRYPTION clause. For non-Database Cloud Service databases, tablespaces will only be encrypted if the ENCRYPTION clause is specified.
DDL	Tablespaces are not transparently encrypted and are only encrypted if the ENCRYPTION clause is specified.

Managing the Software Keystore and Master Encryption Key

- When the Database Cloud Service instance is created, a local auto-login software keystore is also created.
- The keystore is local to the compute node and is protected by a system-generated password.
- The auto-login software keystore is automatically opened when accessed.
- Use the `dbaascli` utility to change (rotate) the master encryption key.

```
DBAAS> tde rotate masterkey
Executing command tde rotate masterkey
Enter keystore password:
Successfully rotated TDE masterkey
```

Creating an Encrypted Tablespace by Using a Nondefault Algorithm

- Use the ENCRYPTION USING clause to specify the algorithm:

```
SQL> CREATE TABLESPACE encrypt_ts
     DATAFILE '$ORACLE_HOME/dbs/encrypt.dat' SIZE 100M
     ENCRYPTION USING '3DES168'
     DEFAULT STORAGE (ENCRYPT);
```

- Restrictions:
  - Temporary and undo tablespaces cannot be encrypted.
  - BFILE data type and external tables are not encrypted.
  - The key for an encrypted tablespace cannot be changed.
  - The SYSTEM tablespace cannot be encrypted.

Summary

- In this lesson, you should have learned how to:
  - Explain how table data is stored in the database
  - Use SQL*Plus to:
    - Create and drop tablespaces
    - Alter tablespaces
    - View tablespace information
  - Implement Oracle Managed Files (OMF)
  - Use SQL*Plus to move and rename online data files
  - Implement tablespace encryption

Practice 12: Overview

- 12-1: Viewing Tablespace Information
- 12-2: Creating a Tablespace
- 12-3: Creating a Tablespace that is Encrypted by Default

Managing Storage Space

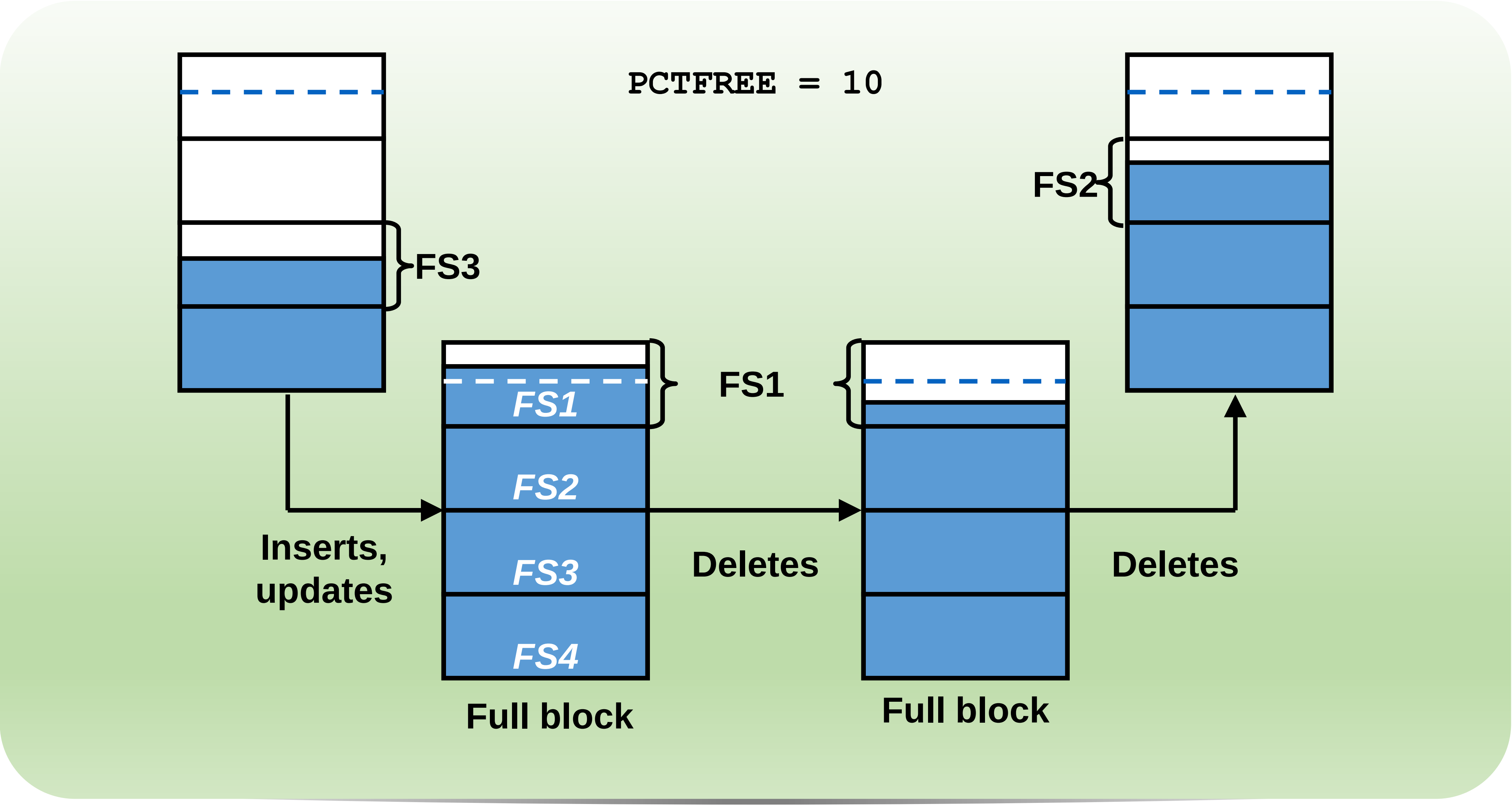
Objectives

- After completing this lesson, you should be able to:
  - Describe how the Oracle Database server automatically manages space
  - Save space by using compression
  - Proactively monitor and manage tablespace space usage
  - Describe segment creation in the Oracle database
  - Control deferred segment creation
  - Reclaim wasted space from tables and indexes by using the segment shrink functionality
  - Manage resumable space allocation

Space Management Features

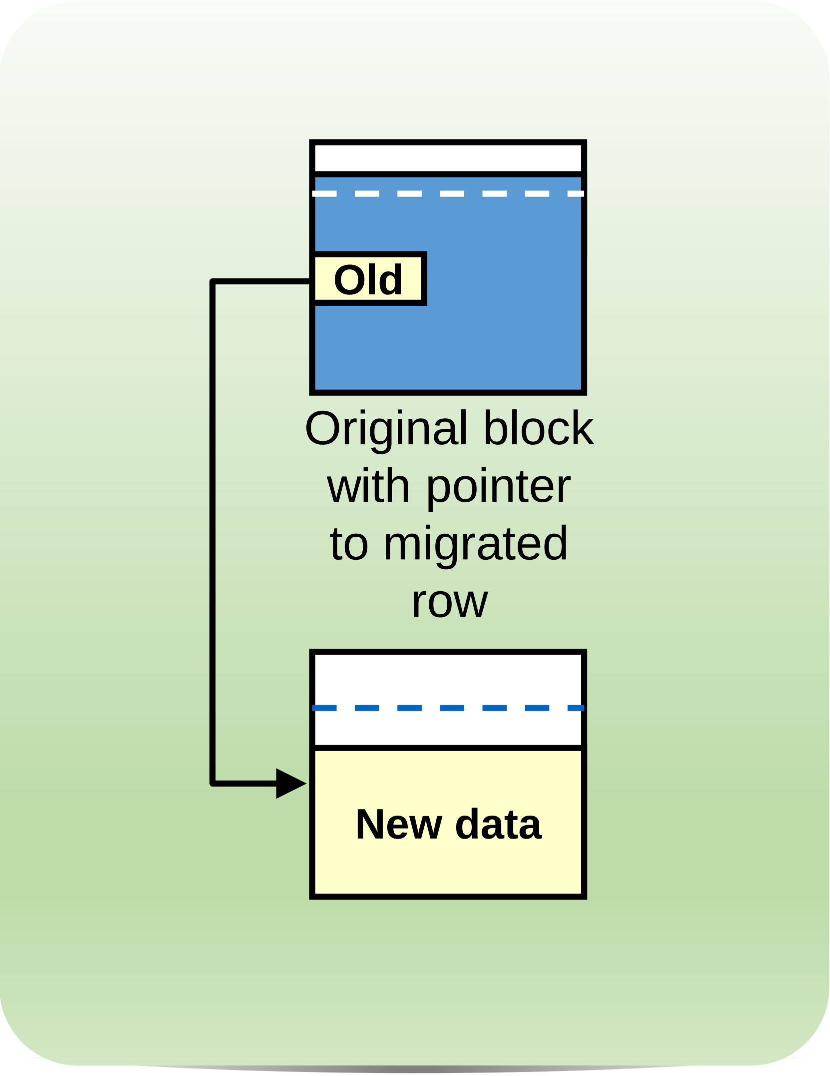
- Oracle Managed Files (OMF)
- Free-space management with bitmaps (“locally managed”) and automatic data file extension
- Proactive space management (default thresholds and server-generated alerts)
- Space reclamation (shrinking segments, online table redefinition)
- Capacity planning (growth reports)

Block Space Management

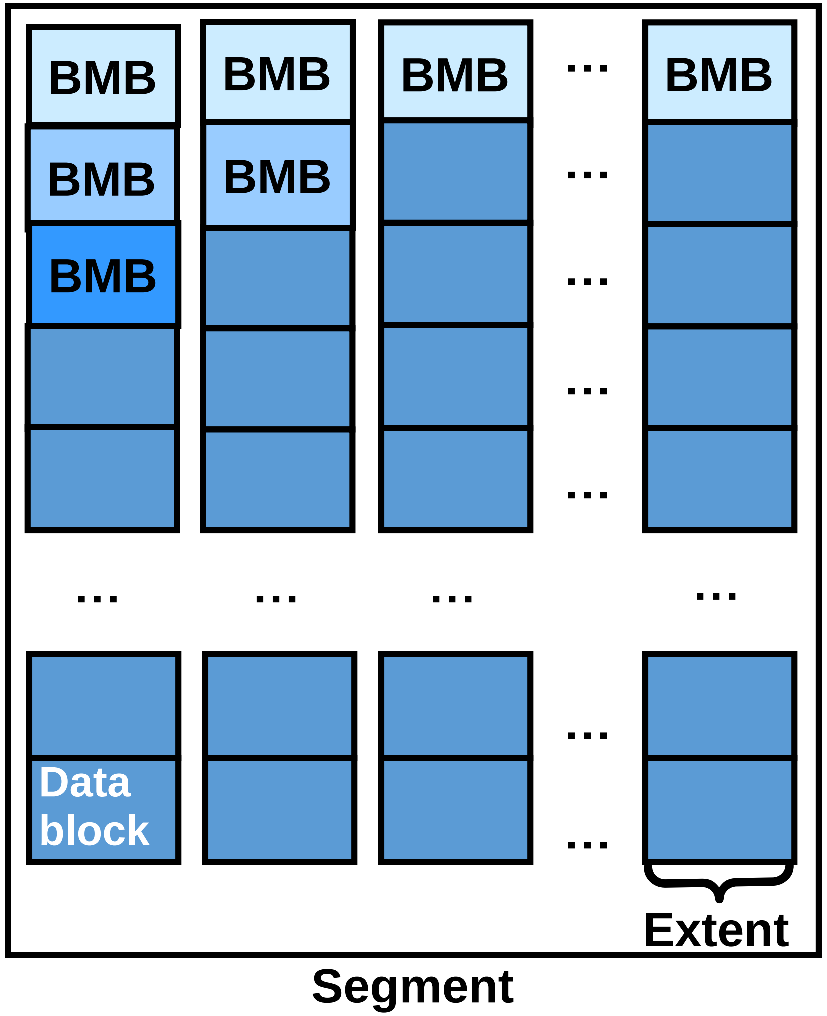


Row Chaining and Migration

- 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)

Types of Segments

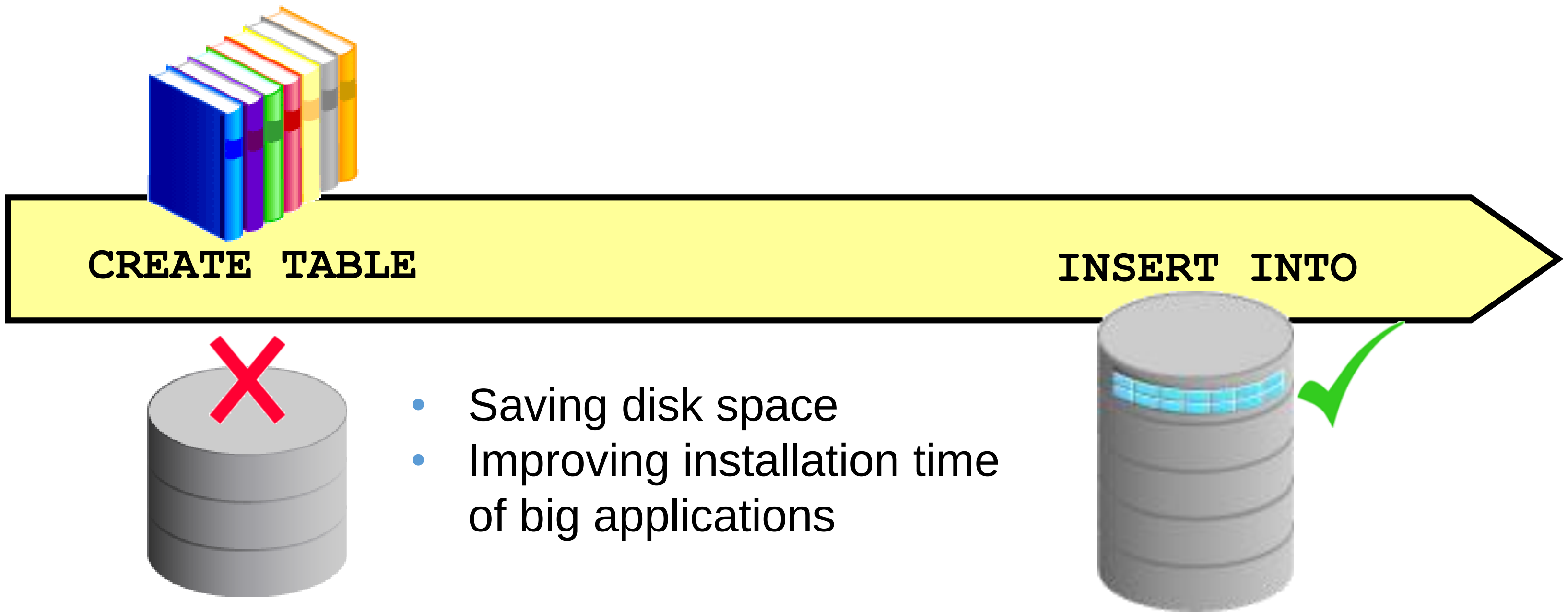
- A segment is a set of extents allocated for a certain logical structure.
- The different types of segments include:
  - Table and cluster
  - Index
  - Undo
  - Temporary
- Segments are dynamically allocated by the Oracle Database server.

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

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



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;
```

Restrictions and Exceptions

- Segment creation on demand is not for the following:
  - IOTs, clustered tables, or other special tables
  - Tables in dictionary-managed tablespaces

Space-Saving Features

- No segments for unusable indexes and index partitions
- Creating an index without a segment:

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

- Removing any allocated space for an index:

```
ALTER INDEX test_i UNUSABLE
```

- Creating the segment for an index:

```
ALTER INDEX test_i REBUILD
```

Private Temporary Tables

USER_PRIVATE_TEMP_TABLES

- Private Temporary Tables (PTTs) exist only for the session that creates them.
  - You can create a PTT with 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 PTT with the same name but different shape.

  - PTT 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;
```

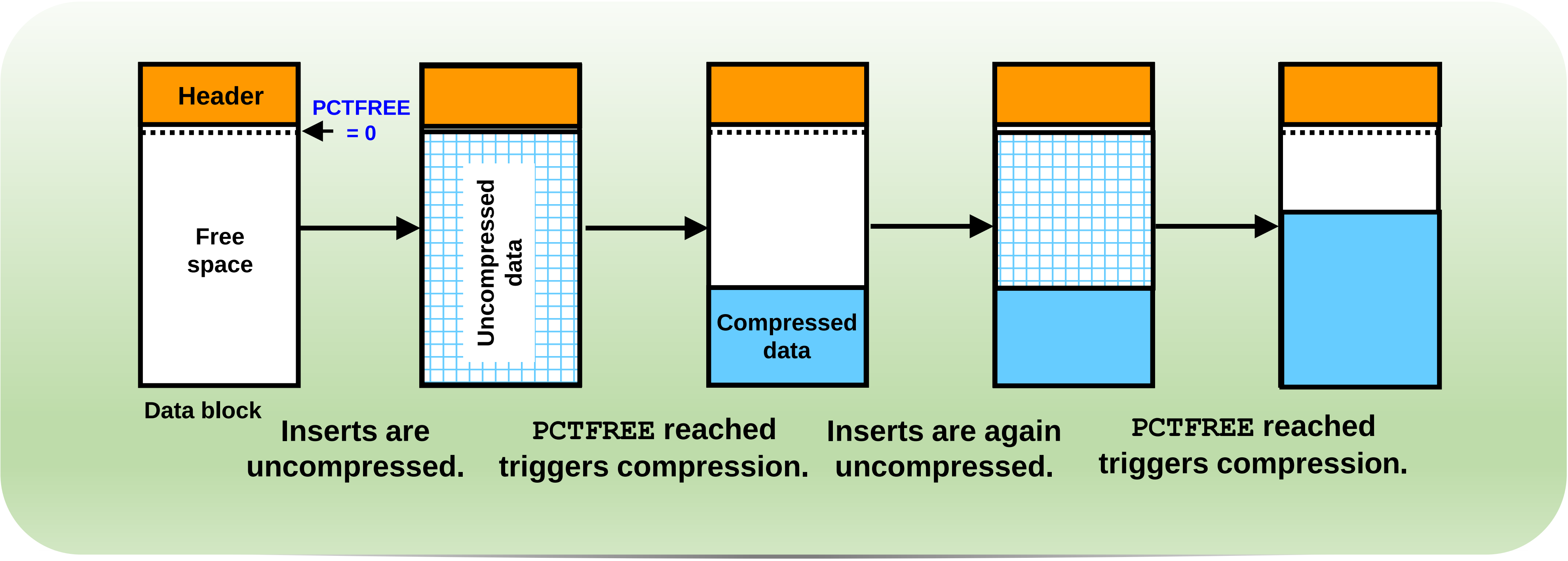
Table Compression: Overview

- 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 [BASIC]`	DSS
Advanced row compression	High	Minimal	`ROW STORE COMPRESS ADVANCED`	OLTP, DSS

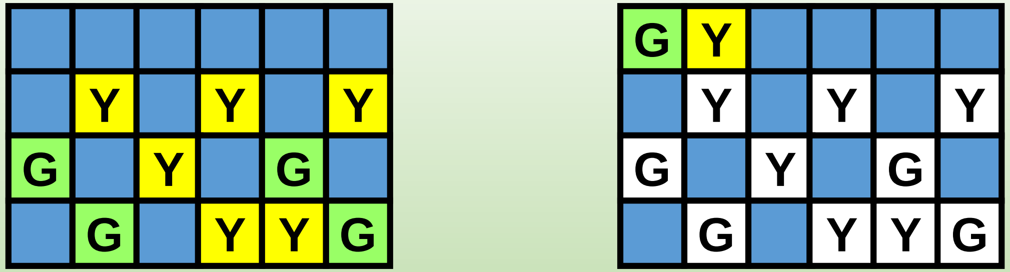
Compression for Direct-Path Insert Operations

- 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



Uncompressed block

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

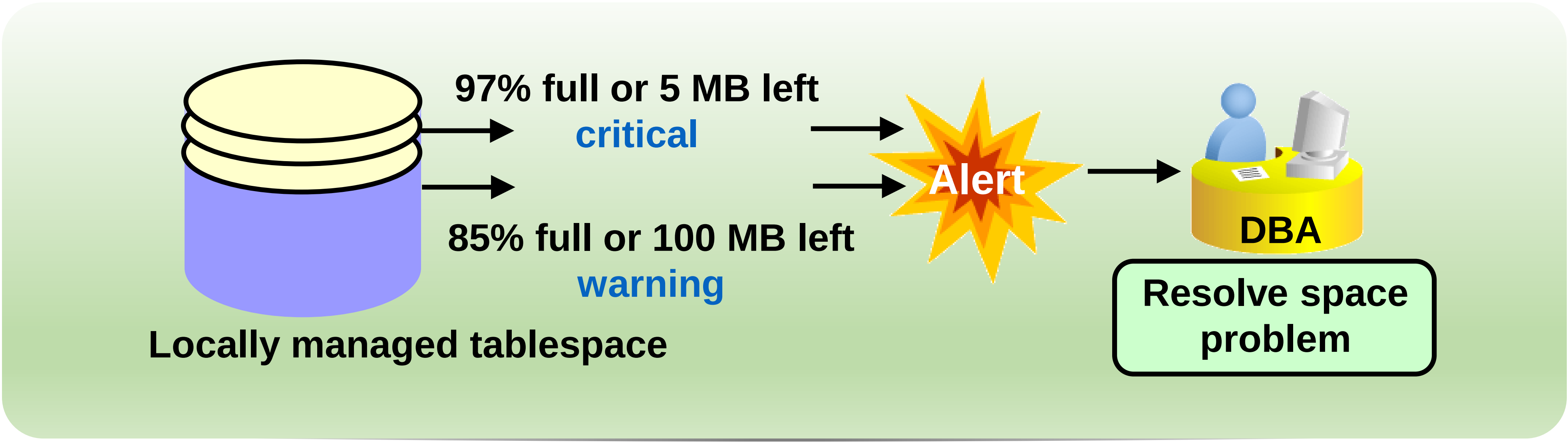
Specifying Table Compression

- 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 Compression Advisor

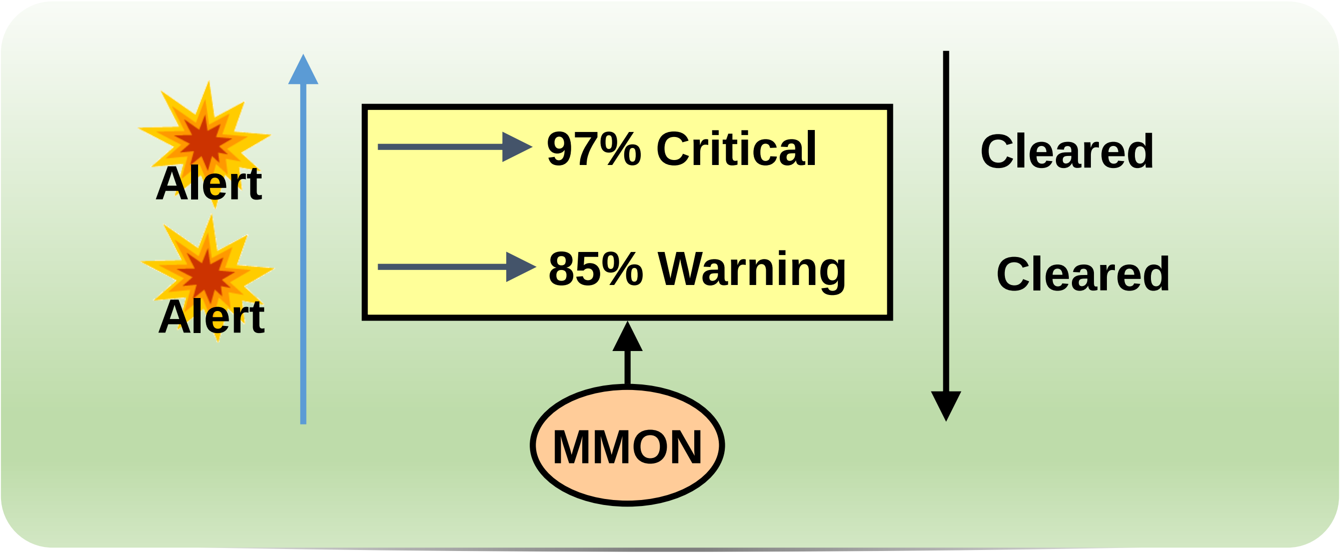
- Analyzes objects to give an estimate of space savings for different compression methods
- Helps in deciding the correct compression level for an application
- Recommends various strategies for compression
  - Picks the right compression algorithm for a particular data set
  - Sorts on a particular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

Monitoring Tablespace Space Usage

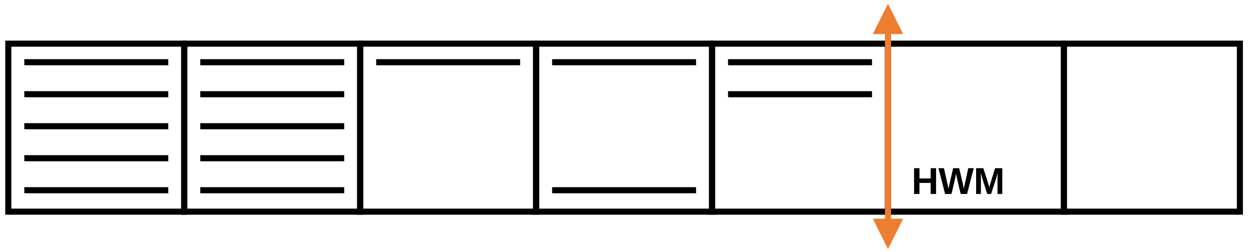


- Read-only and offline tablespaces: Do not set up alerts.
- Temporary tablespace: Threshold corresponds to space currently used by sessions.
- Undo tablespace: Threshold corresponds to space used by active and unexpired extents.
- Auto-extensible files: Threshold is based on the maximum file size.

Reclaiming Space by Shrinking Segments

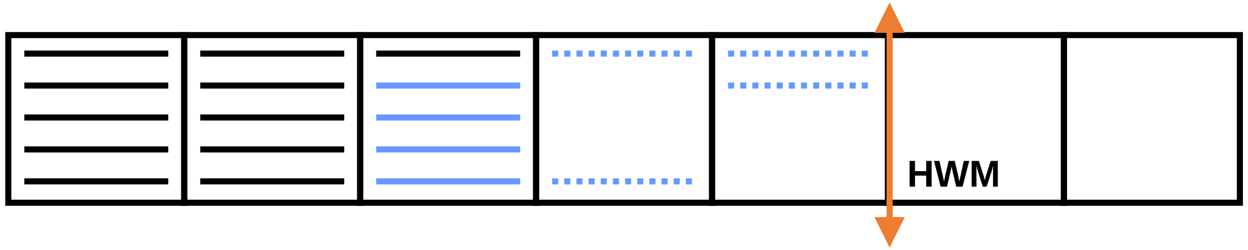
- Shrink is an online and in-place operation.
- It is applicable only to segments residing in ASSM tablespaces.
- Candidate segment types:
  - Heap-organized tables and index-organized tables
  - Indexes
  - Partitions and subpartitions
  - Materialized views and materialized view logs

Shrinking Segments



1

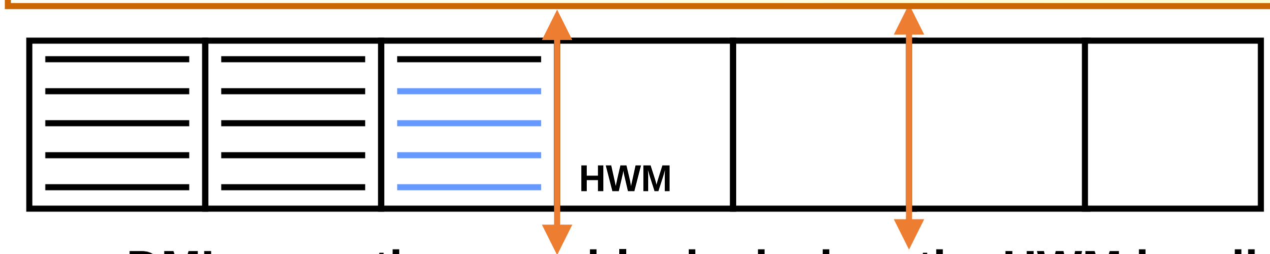
```
ALTER TABLE employees SHRINK SPACE COMPACT;
```



DML operations and queries can be issued during compaction.

2

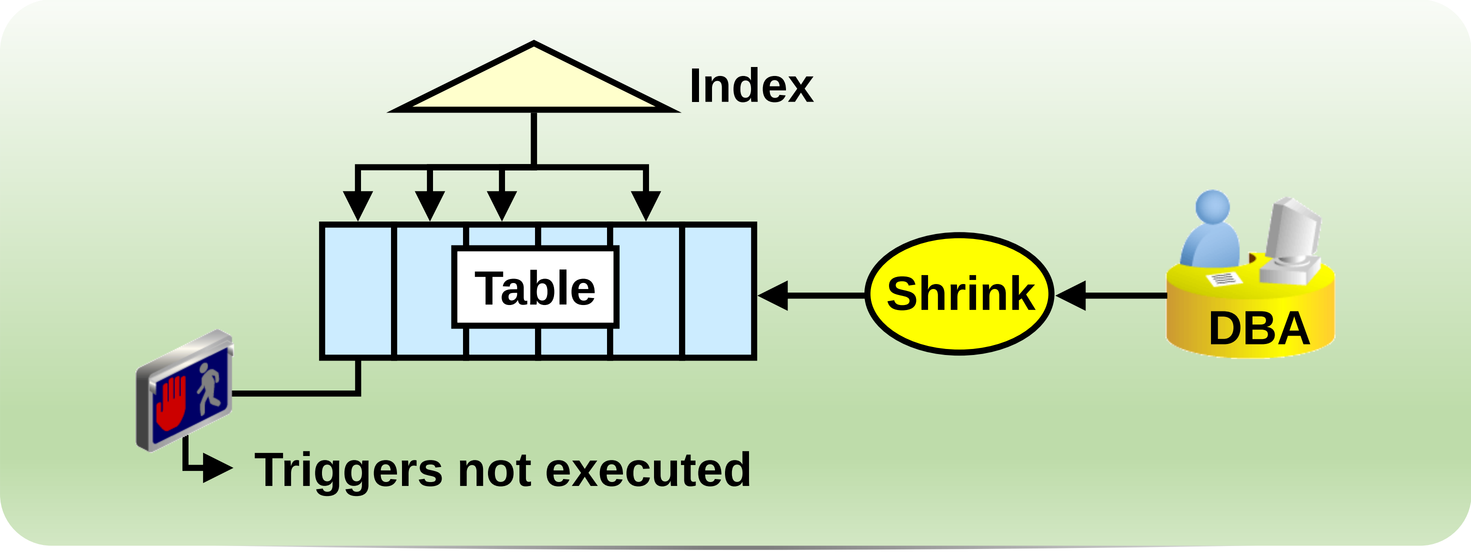
```
ALTER TABLE employees SHRINK SPACE;
```



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



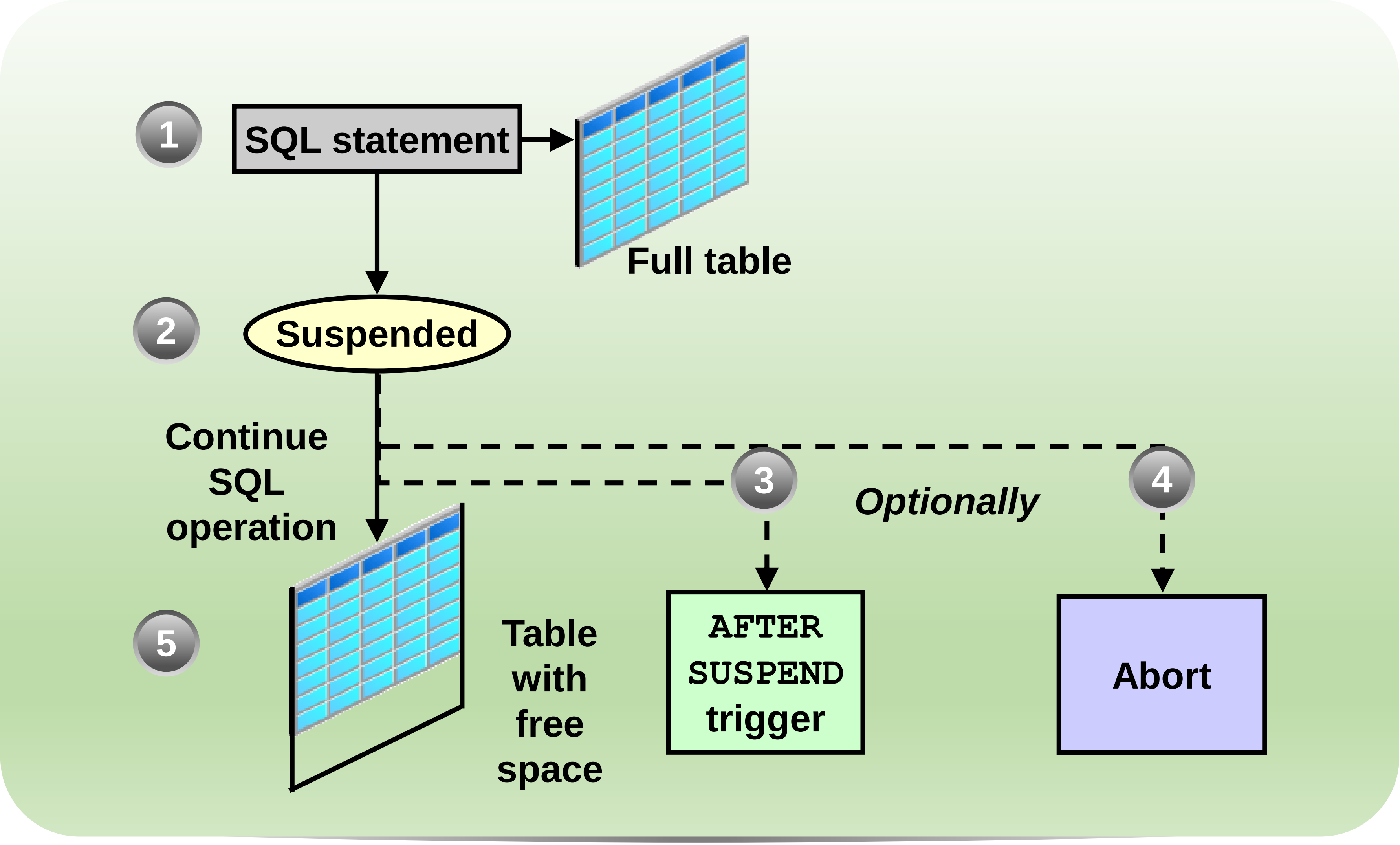
Managing Resumable Space Allocation

- 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 how the Oracle Database server automatically manages space
  - Save space by using compression
  - Proactively monitor and manage tablespace space usage
  - Describe segment creation in the Oracle database
  - Control deferred segment creation
  - Reclaim wasted space from tables and indexes by using the segment shrink functionality
  - Manage resumable space allocation

Practice 13: Overview

- 13-1: Managing Tablespace Space
- 13-2: Using Compression
- 13-3: Handling Resumable Space Allocation

Managing Undo Data

Objectives

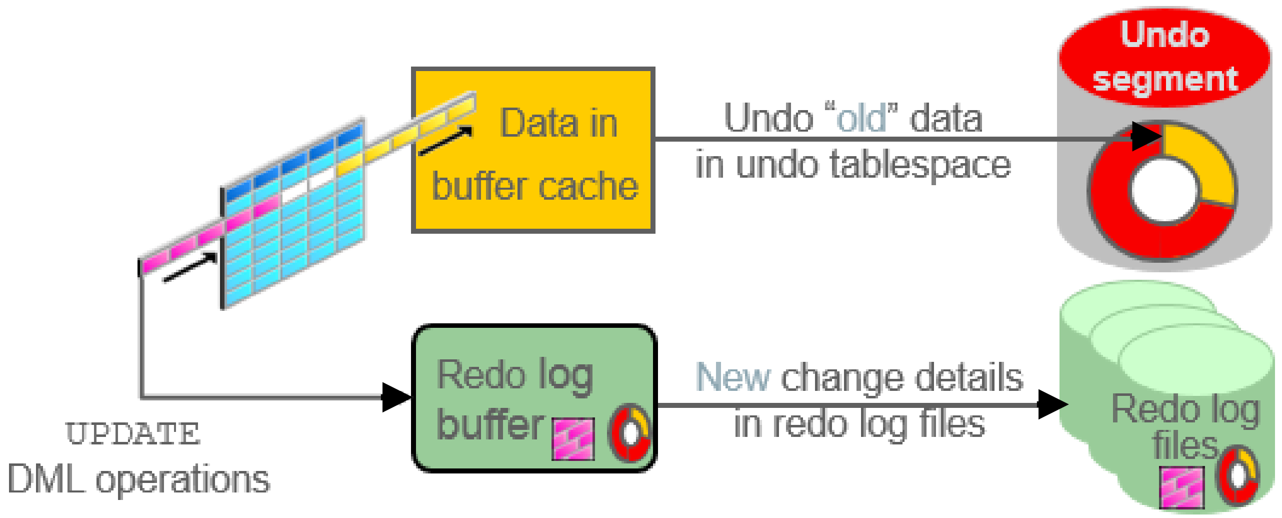
- After completing this lesson, you should be able to:
  - Explain DML and undo data generation
  - Monitor and administer undo data
  - Describe the difference between undo data and redo data
  - Configure undo retention
  - Guarantee undo retention
  - Enable temporary undo
  - Use the Undo Advisor

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.

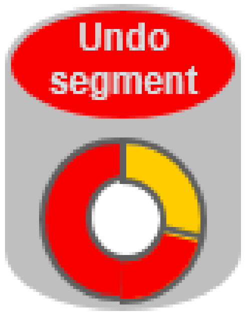


Storing Undo Information

- 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

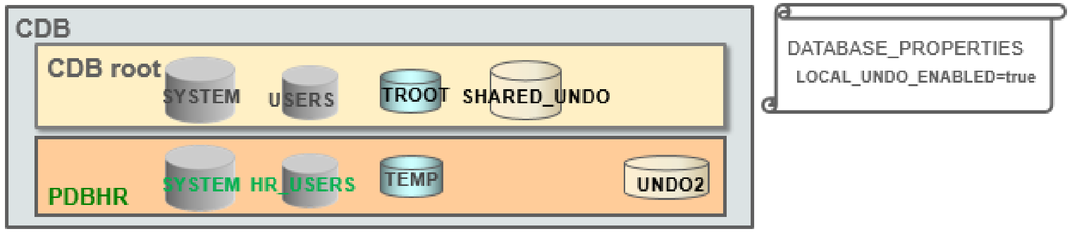


Managing Undo

- 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

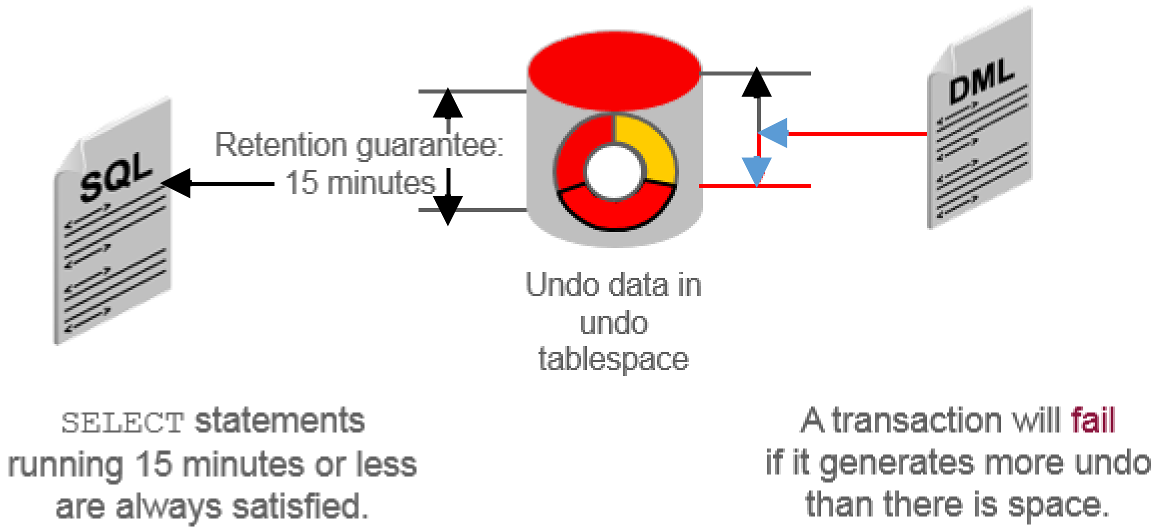
- `UNDO_RETENTION` specifies (in seconds) how long already committed undo information is to be retained.
- Set this parameter when:
  - The undo tablespace has the `AUTOEXTEND` option enabled
  - You want to set undo retention for `LOBs`
  - You want to guarantee retention

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;
```

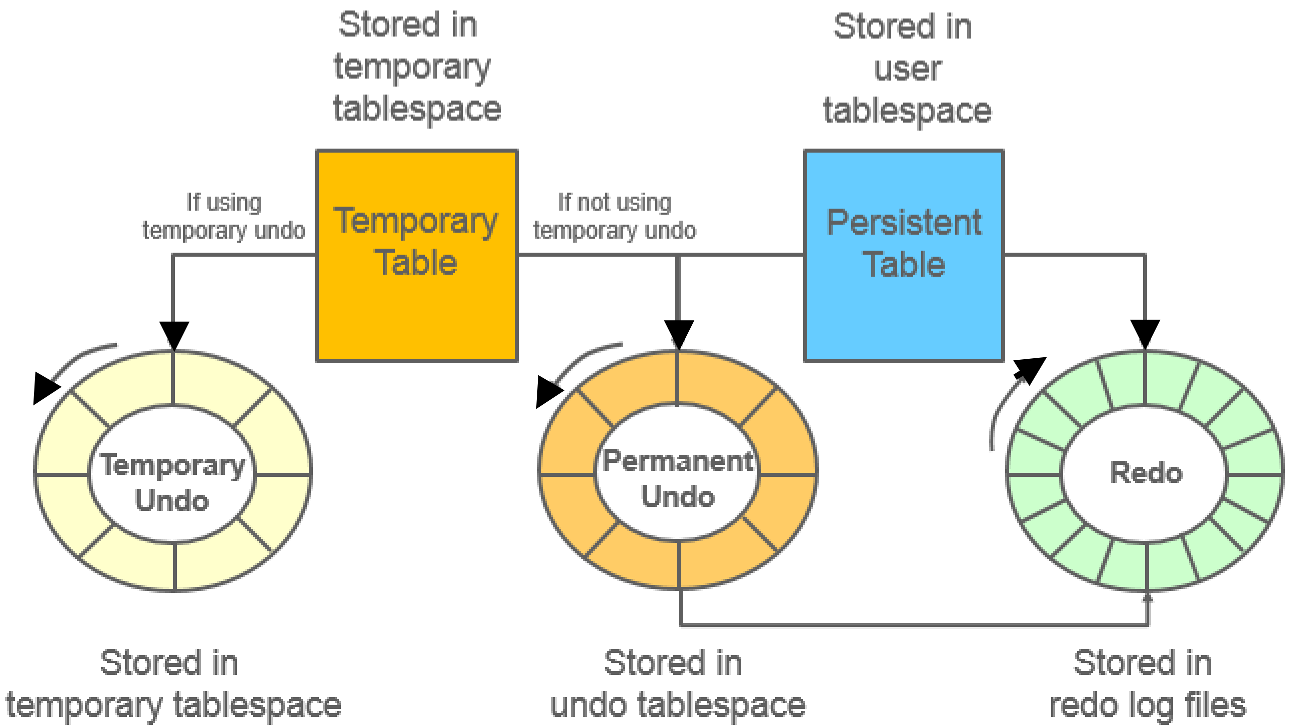


This example is based on an `UNDO_RETENTION` setting of 900 seconds (15 minutes).

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:

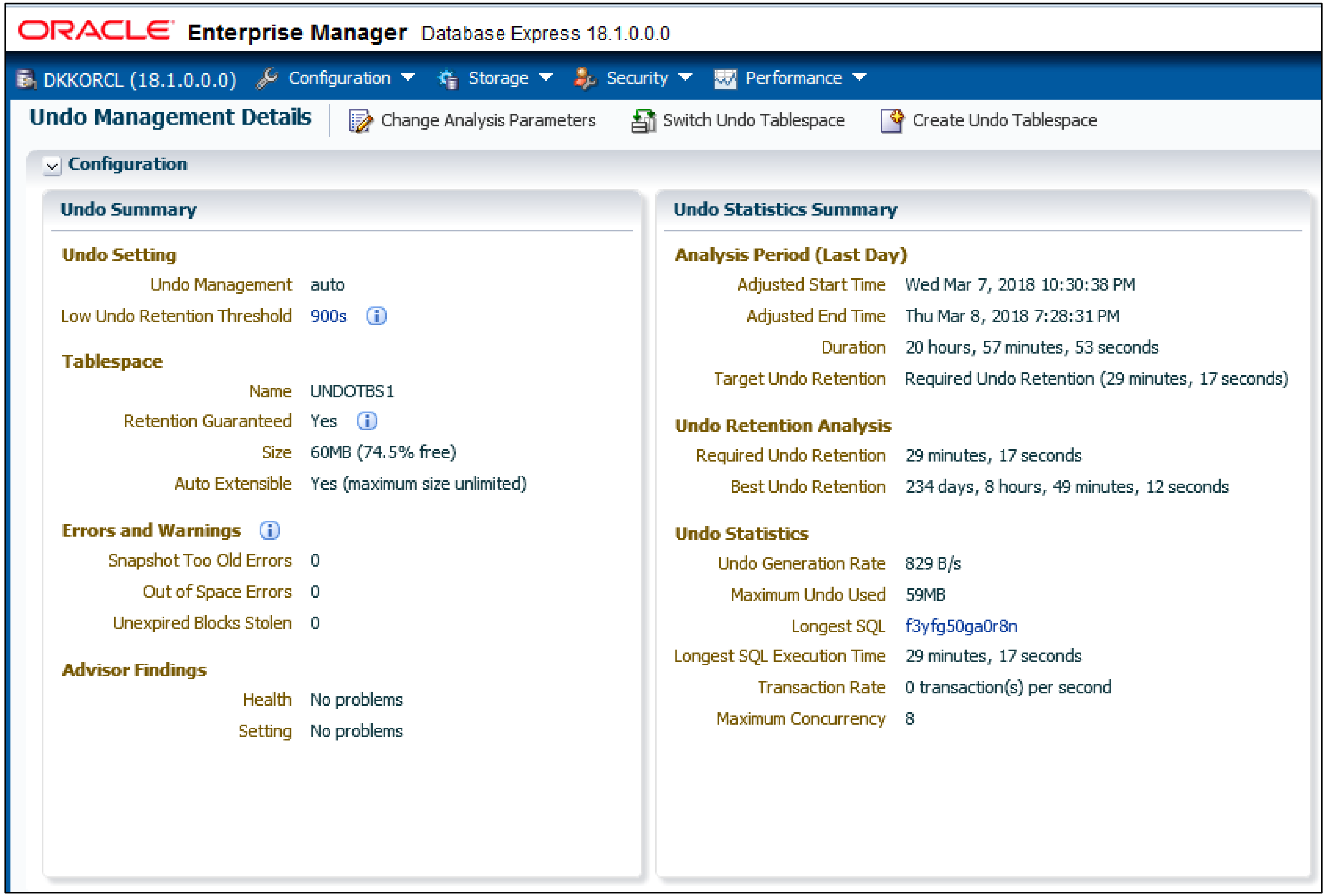
```
SQL> ALTER SYSTEM SET temp_undo_enabled = true;
```

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

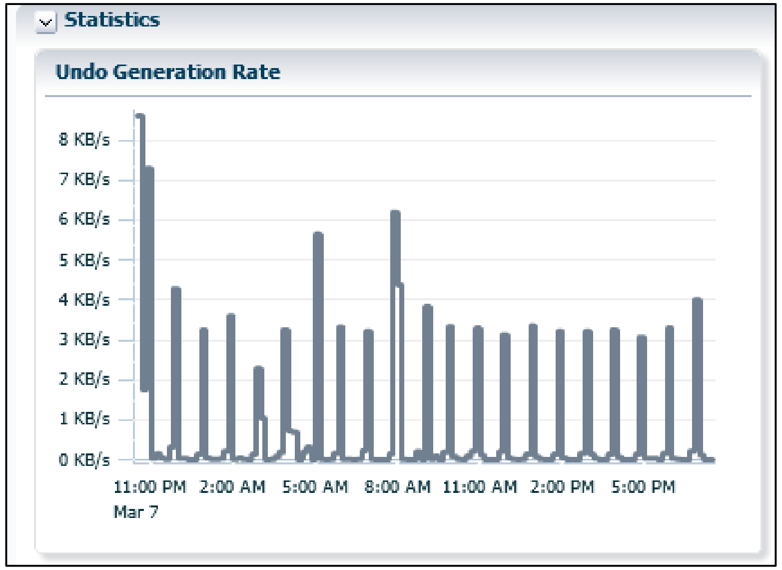
Monitoring Temporary Undo

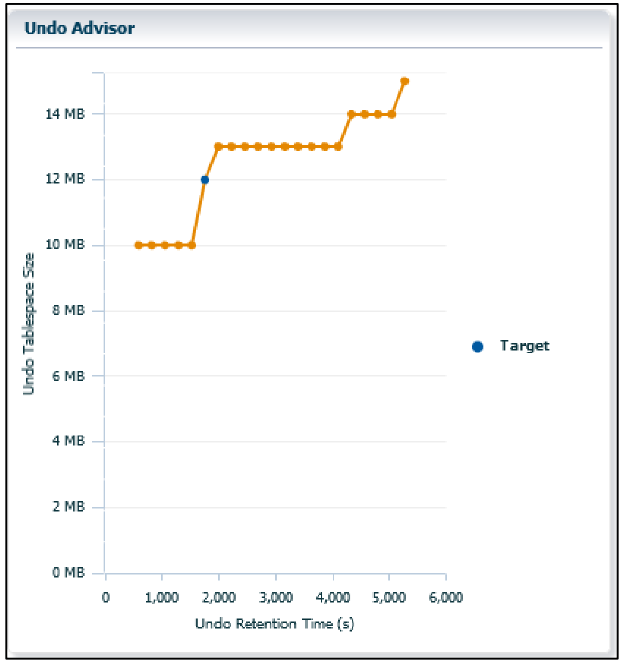
```
SQL> SELECT to_char(BEGIN_TIME,'dd/mm/yy hh24:mi:ss') "BEGIN TIME",
  2  txncount "TXNCNT", maxconcurrency, undoblkcnt, uscount "USCNT",
  3  nospaceerrcnt "NOSPEERRCNT"
  4  FROM  v$tempundostat;

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>
```

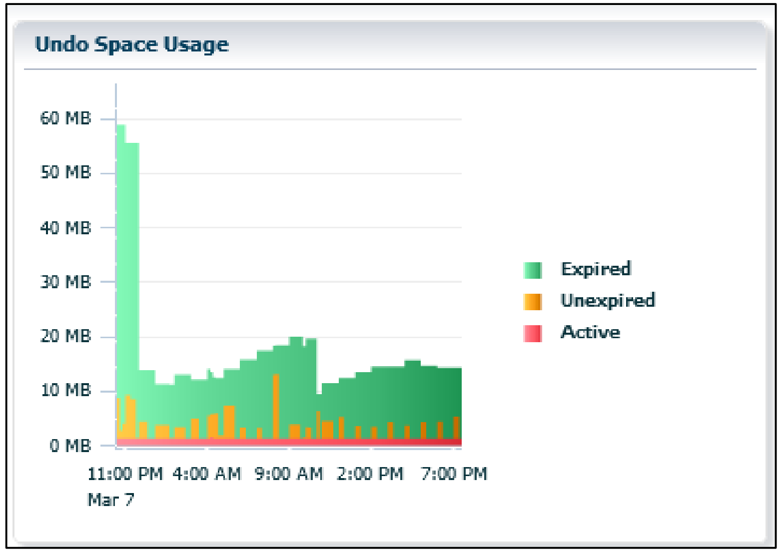
Viewing Undo Information

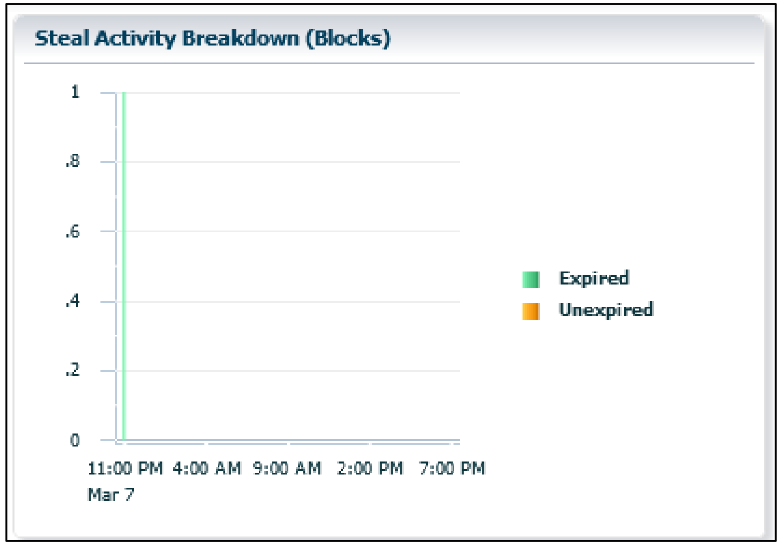


Viewing Undo Activity









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
  - Use the Undo Advisor

Practice 14: Overview

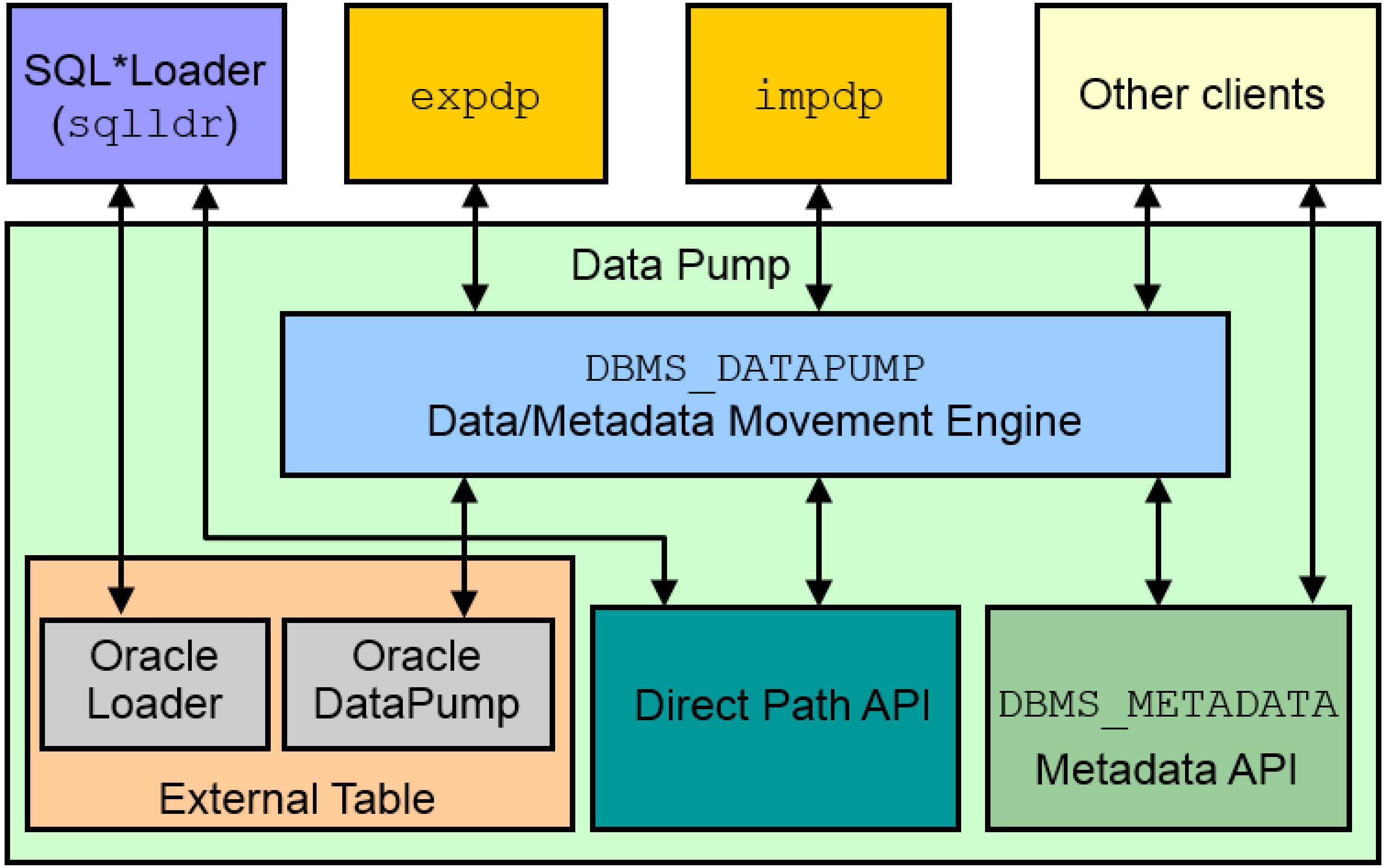
- 14-1: Managing Undo Data

Moving Data

Objectives

- After completing this lesson, you should be able to:
  - Describe ways to move data
  - Explain the general architecture of Oracle Data Pump
  - Use Data Pump Export and Import to move data between Oracle databases
  - Use SQL*Loader to load data from a non-Oracle database (or user files)
  - Use external tables to move data via platform-independent files
  - Describe methods that can be used to migrate databases to Oracle Database Cloud Service

Moving Data: General Architecture



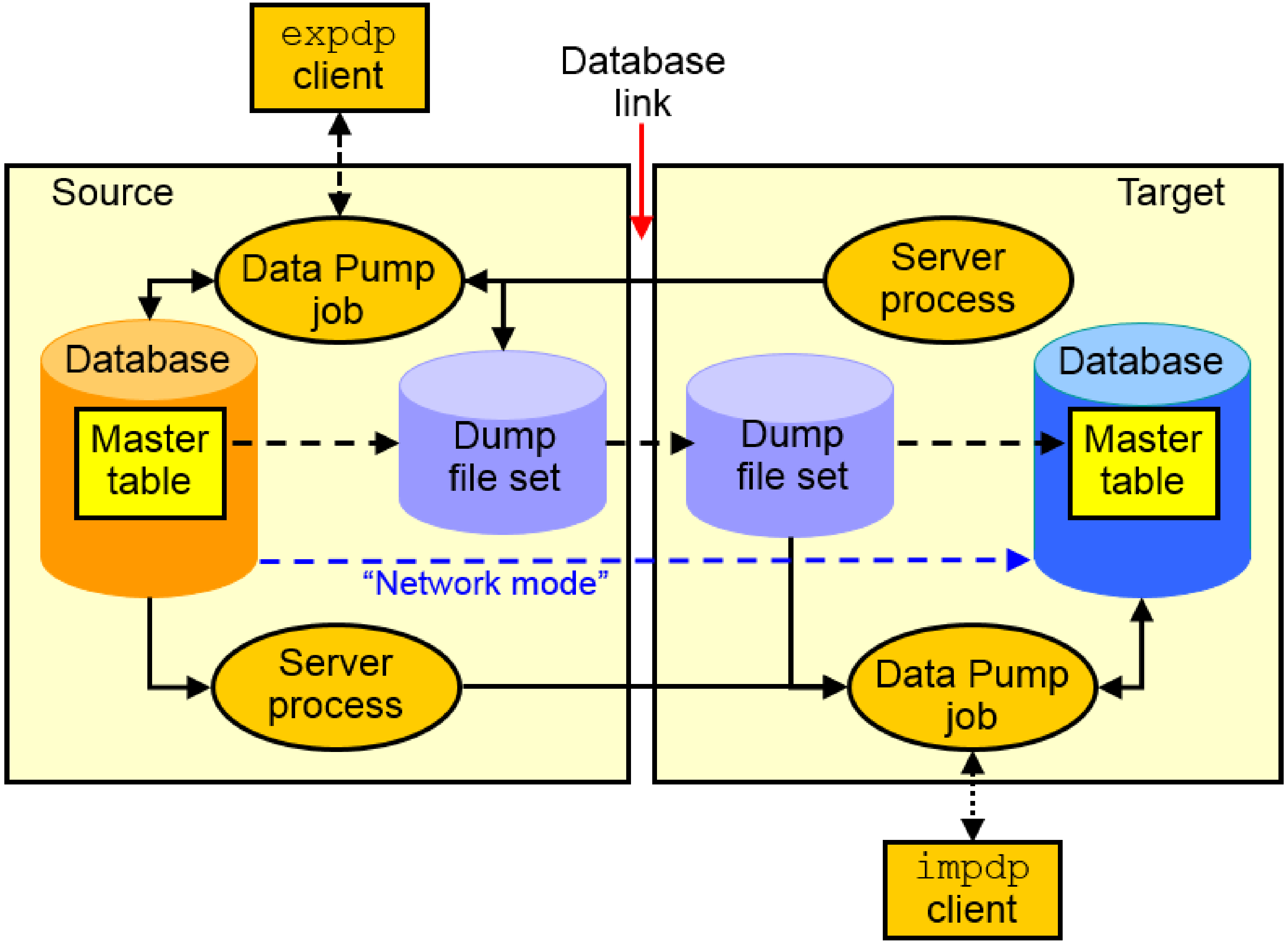
Oracle Data Pump: Overview

- As a server-based facility for high-speed data and metadata movement, Oracle Data Pump:
  - Is callable via `DBMS_DATAPUMP`
  - Provides the following tools:
    - `expdp` 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`

Data Pump Export and Import Clients



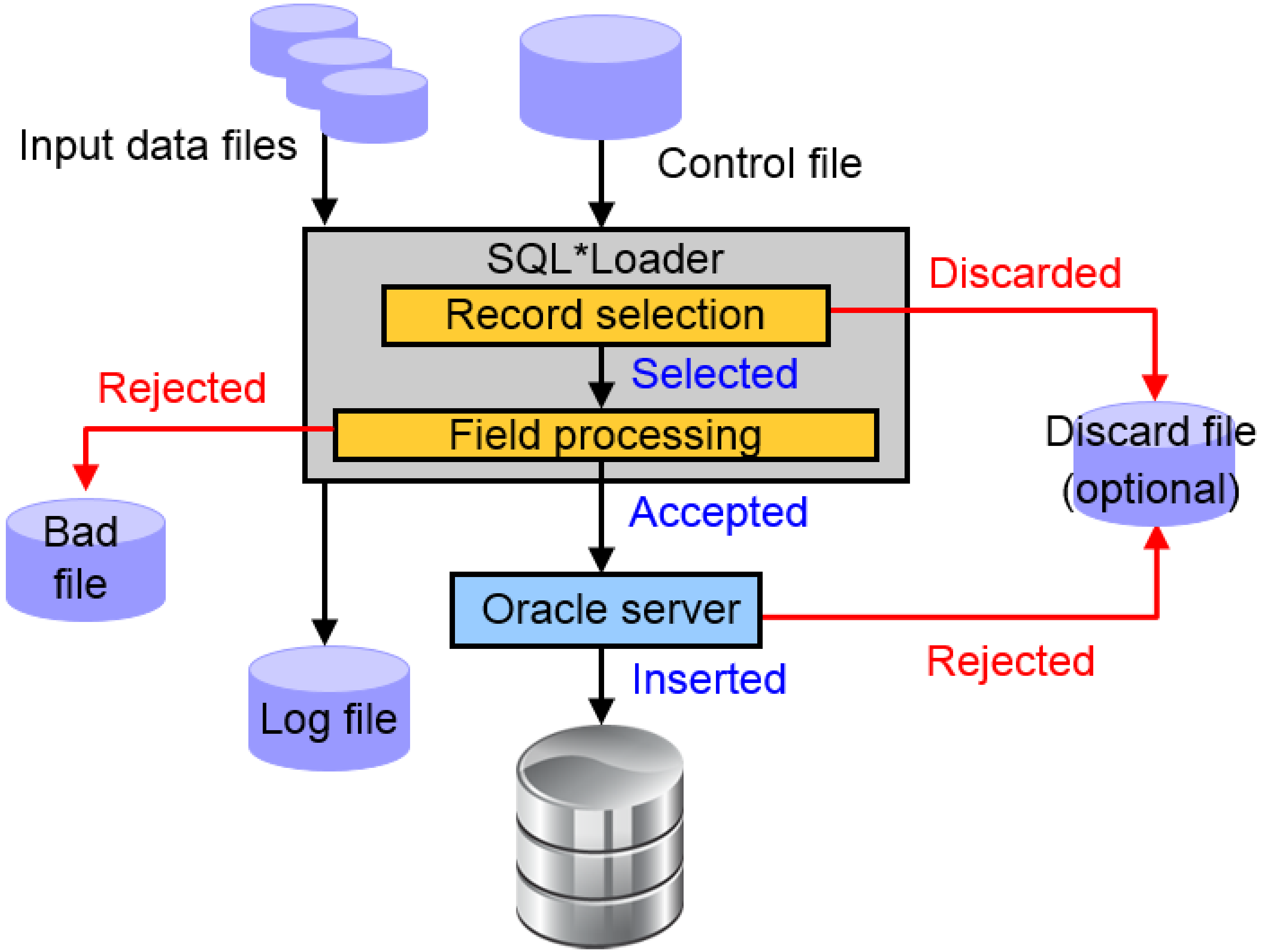
Data Pump Interfaces and Modes

- 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`

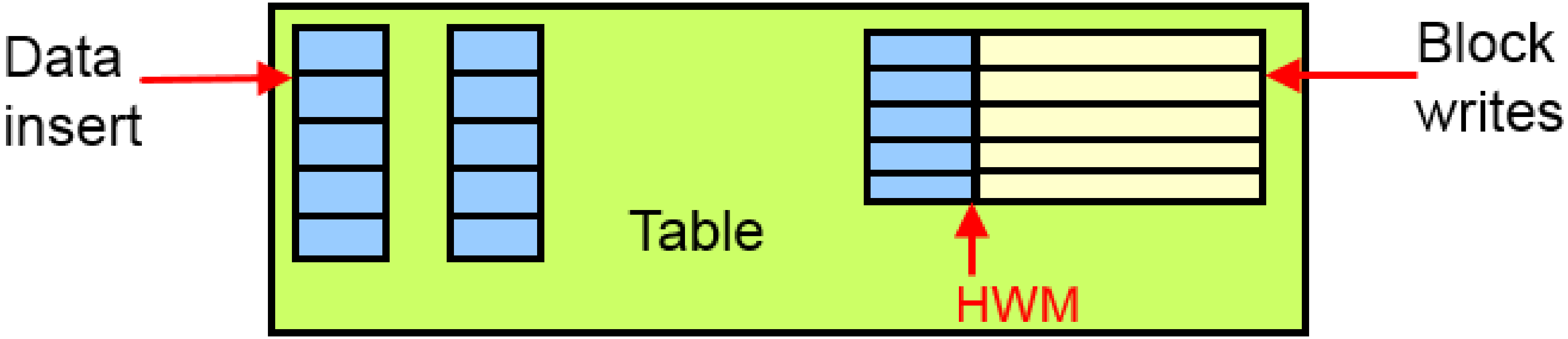
SQL Loader: Overview



Comparing Loading Methods

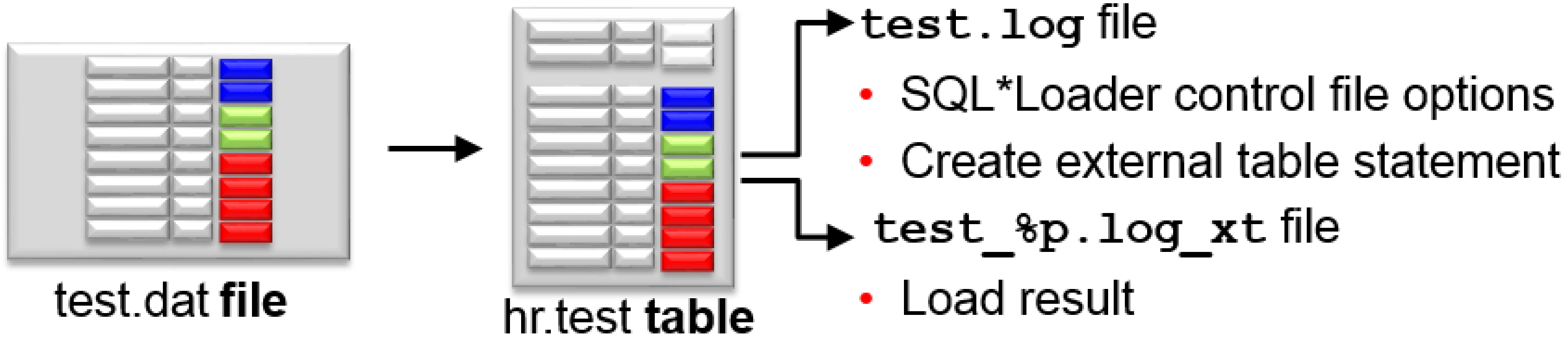
Conventional Load	Direct Path Load
Uses `COMMIT`	Uses data saves (faster operation)
Always generates redo entries	Generates redo only under specific conditions
Enforces all constraints	Enforces only `PRIMARY KEY`, `UNIQUE`, and `NOT NULL` constraints
Fires `INSERT` triggers	Does not fire `INSERT` triggers
Can load into clustered tables	Does not load into clusters
Allows other users to modify tables during load operation	Prevents other users from making changes to tables during load operation
Maintains index entries on each insert	Merges new index entries at the end of the load

Data Save Feature

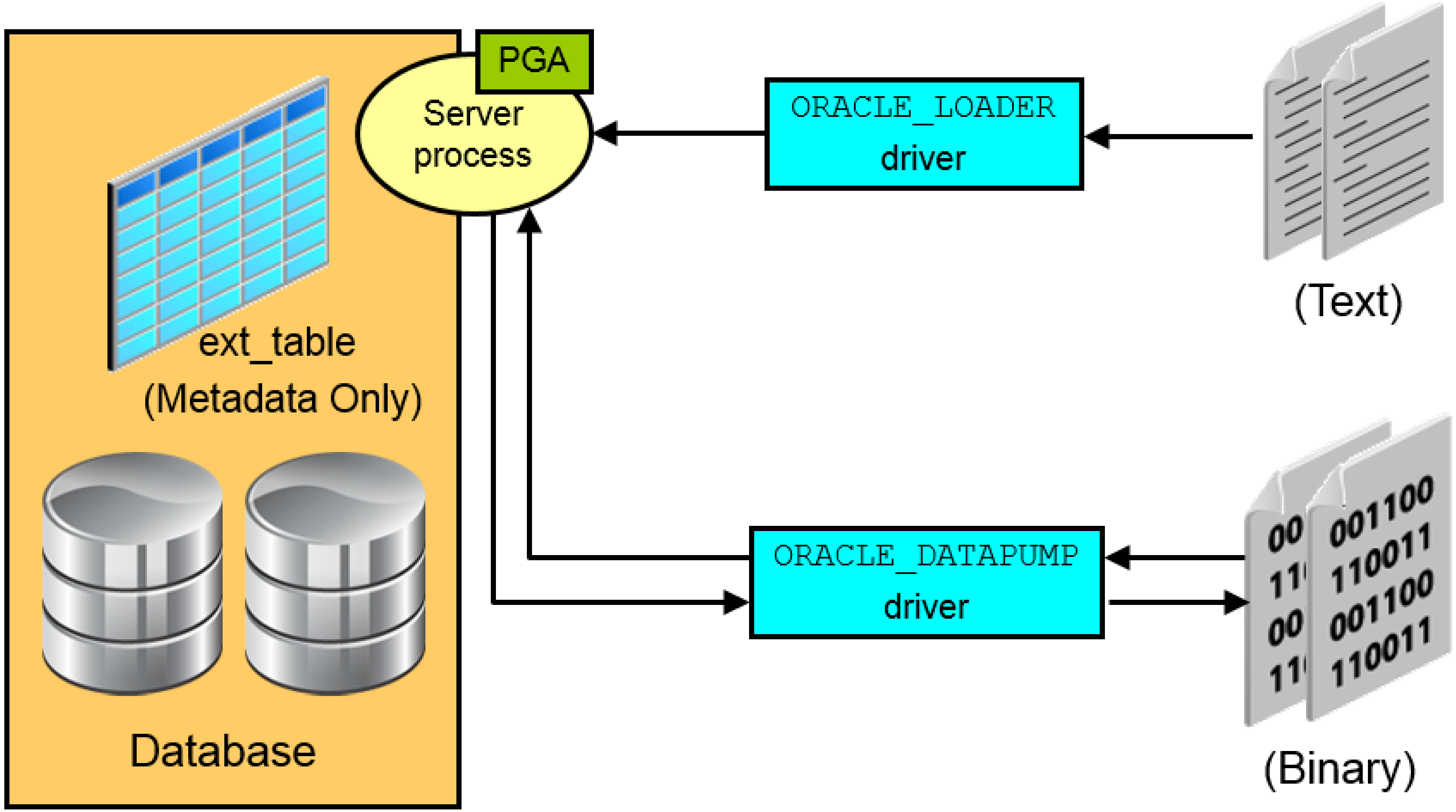


Express Mode

- 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.



External Tables



External Table 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.

Migrating to Oracle Database Cloud Service: Considerations

- Some of the characteristics and factors to consider when choosing a migration method are:
  - On-premises database version
  - Oracle Database Cloud database version
  - On-premises host operating system and version
  - On-premises database character set
  - Quantity of data, including indexes
  - Data types used in the on-premises database
  - Storage for data staging
  - Acceptable length of system outage
  - Network bandwidth

Migrating to Oracle Database Cloud Service: Information Gathering

- To determine which migration methods are applicable to your migration scenario, gather the following information:
  - Database version of your on-premises database
  - For on-premises Oracle Database 12c databases, the architecture of the database (multitenant or non-CDB)
  - Endian format (byte ordering) of your on-premises database's host platform
  - Database character set of your on-premises database and your Database Cloud Service database
  - Database version of your Database Cloud Service database

See Choosing a Migration Method in *Using Oracle Database Cloud Service* for additional information.

Applicable Migration Methods

Method	On-premises 11g database to Cloud 11g database	On-premises 11g database to Cloud 12c PDB	On-premises 12c non-CDB to Cloud 12c PDB	On-premises 12c PDB to Cloud 12c PDB
Data Pump Conventional Export/Import	Y	Y	Y	Y
Data Pump Transportable Tablespace	Y	Y	Y	Y
Data Pump Full Transportable	N	Y	Y	Y
RMAN Transportable Tablespace with Data Pump	Y	Y	Y	Y
RMAN CONVERT Transportable Tablespace with Data Pump	Y	Y	Y	Y
RMAN Cross-Platform Transportable Tablespace Backup Sets	N	N	Y	Y
RMAN Cross-Platform Transportable PDB	N	N	N	Y

Applicable Migration Methods

Method	On-premises 11g database to Cloud 11g database	On-premises 11g database to Cloud 12c PDB	On-premises 12c non-CDB to Cloud 12c PDB	On-premises 12c PDB to Cloud 12c PDB
Unplugging/Plugging	N	N	Y	Y
Remote Cloning	N	N	Y	Y
SQL Developer and SQL*Loader to Migrate Selected Objects	N	N	Y	Y
SQL Developer and `INSERT` Statements to Migrate Selected Objects	N	N	Y	Y

Summary

- In this lesson, you should have learned how to:
  - Describe ways to move data
  - Explain the general architecture of Oracle Data Pump
  - Use Data Pump Export and Import to move data between Oracle databases
  - Use SQL*Loader to load data from a non-Oracle database (or user files)
  - Use external tables to move data via platform-independent files
  - Describe methods that can be used to migrate databases to Oracle Database Cloud Service

Practice 15: Overview

- 15-1: Moving Data from One PDB to Another
- 15-2: Loading Data into a PDB from an External File

Backup and Recovery Concepts

Objectives

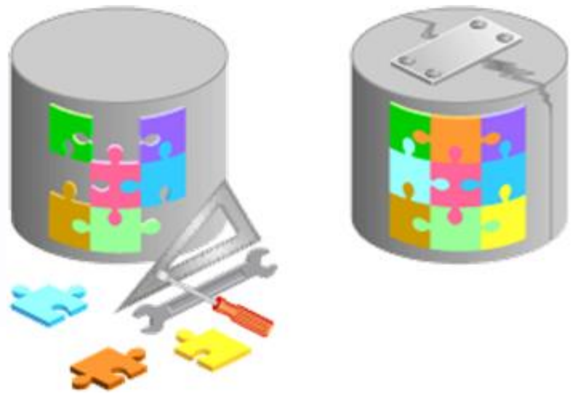
- After completing this lesson, you should be able to:
  - Identify DBA responsibilities regarding database backup and recovery
  - Identify the types of failure that can occur in an Oracle database
  - Describe instance recovery
  - Describe complete and incomplete recovery

DBA Responsibilities

- Protect the database from failure wherever possible.
- Increase the mean time between failures (MTBF).
- Protect critical components by using redundancy.
- Decrease the mean time to recover (MTTR).
- Minimize the loss of data.

Categories of Failure

- Failures can generally be divided into the following categories:
  - Statement failure
  - User process failure
  - Network failure
  - User error
  - Instance failure
  - Media failure



Statement Failure

Typical Problems	Possible Solutions
Attempts to enter invalid data into a table	Work with users to validate and correct data.
Attempts to perform operations with insufficient privileges	Provide the appropriate object or system privileges.
Attempts to allocate space that fails	Enable resumable space allocation. Increase owner quota. Add space to the tablespace.
Logic errors in applications	Work with developers to correct program errors.

User Process Failure

Typical Problems	Possible Solutions
A user performs an abnormal disconnect.	A DBA's action is not usually needed to resolve user process failures. Instance background processes roll back uncommitted changes and release locks. The DBA should watch for trends.
A user's session is abnormally terminated.	
A user experiences a program error that terminates the session.	

Network Failure

Typical Problems	Possible Solutions
Listener fails	Configure a backup listener and connect-time failover.
Network interface card (NIC) fails	Configure multiple network cards.
Network connection fails.	Configure a backup network connection.

User Error

Typical Problems	Possible Solutions
User inadvertently deletes or modifies data	Roll back a transaction and dependent transactions or rewind the table
User drops a table	Recover the table from recycle bin Recover the table from a backup

Use Oracle LogMiner to query your online redo logs and archived redo logs through an Enterprise Manager or SQL interface.

Instance Failure

Typical Causes	Possible Solutions
Power outage	Restart the instance by using the `STARTUP` command. Recovering from instance failure is automatic, including rolling forward changes in the redo logs and then rolling back any uncommitted transactions.
Hardware failure	
Failure of one of the critical background processes	
Emergency shutdown procedures	Investigate the causes of failure by using the alert log, trace files, and Enterprise Manager.

Media Failure

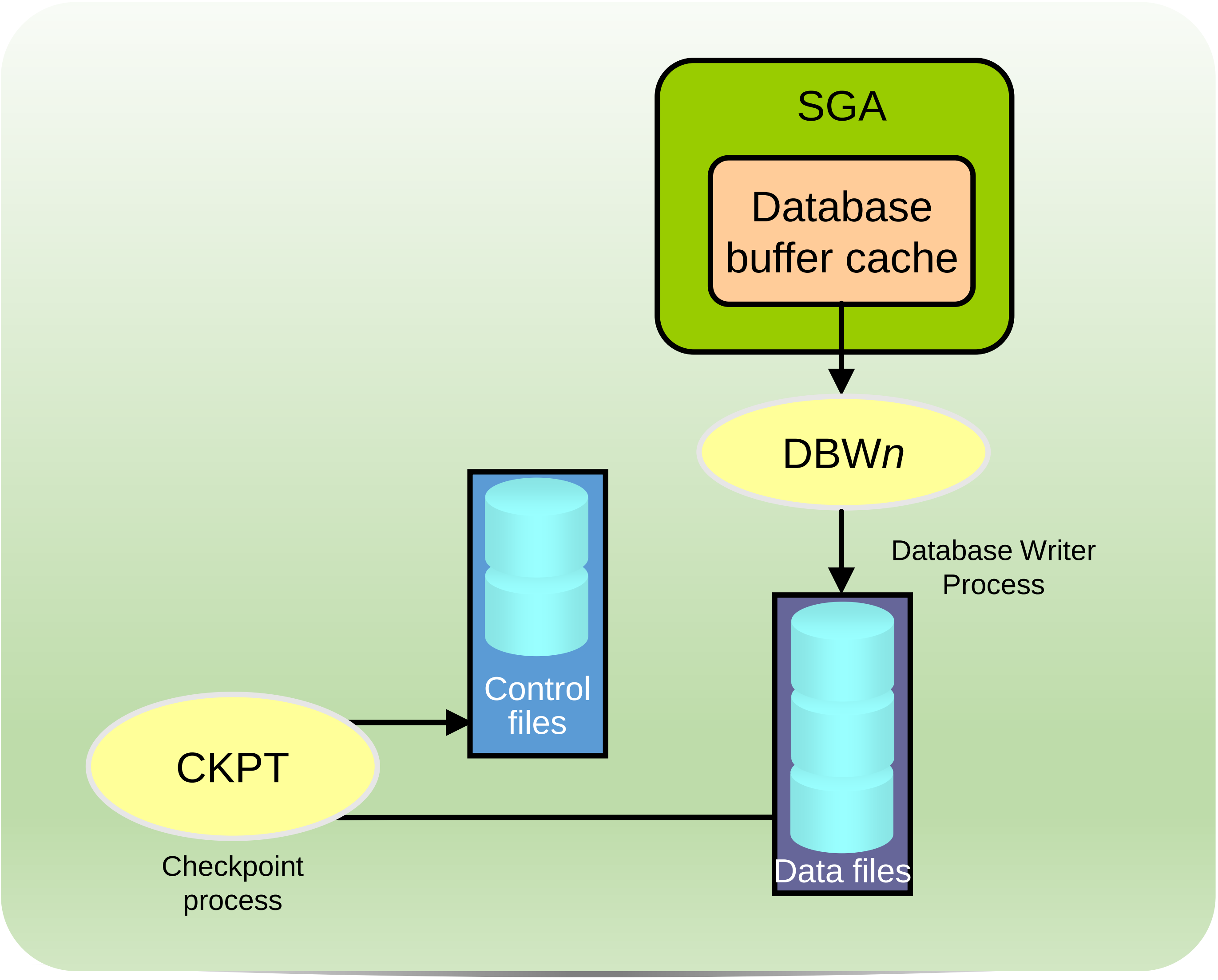
Typical Causes	Possible Solutions
Failure of a disk drive	Restore the affected file from backup.
Failure of a disk controller	Inform the database about a new file location (if necessary).
Deletion or corruption of a file needed for a database operation	Recover the file by applying redo information (if necessary).

Understanding Instance Recovery

- You can understand instance recovery by becoming familiar with these concepts and procedures:
  - The checkpoint (CKPT) process
  - Redo log files and the Log Writer (LGWR) process
  - Automatic instance or crash recovery
  - Phases of instance recovery
  - Tuning instance recovery
  - Using the MTTR Advisor

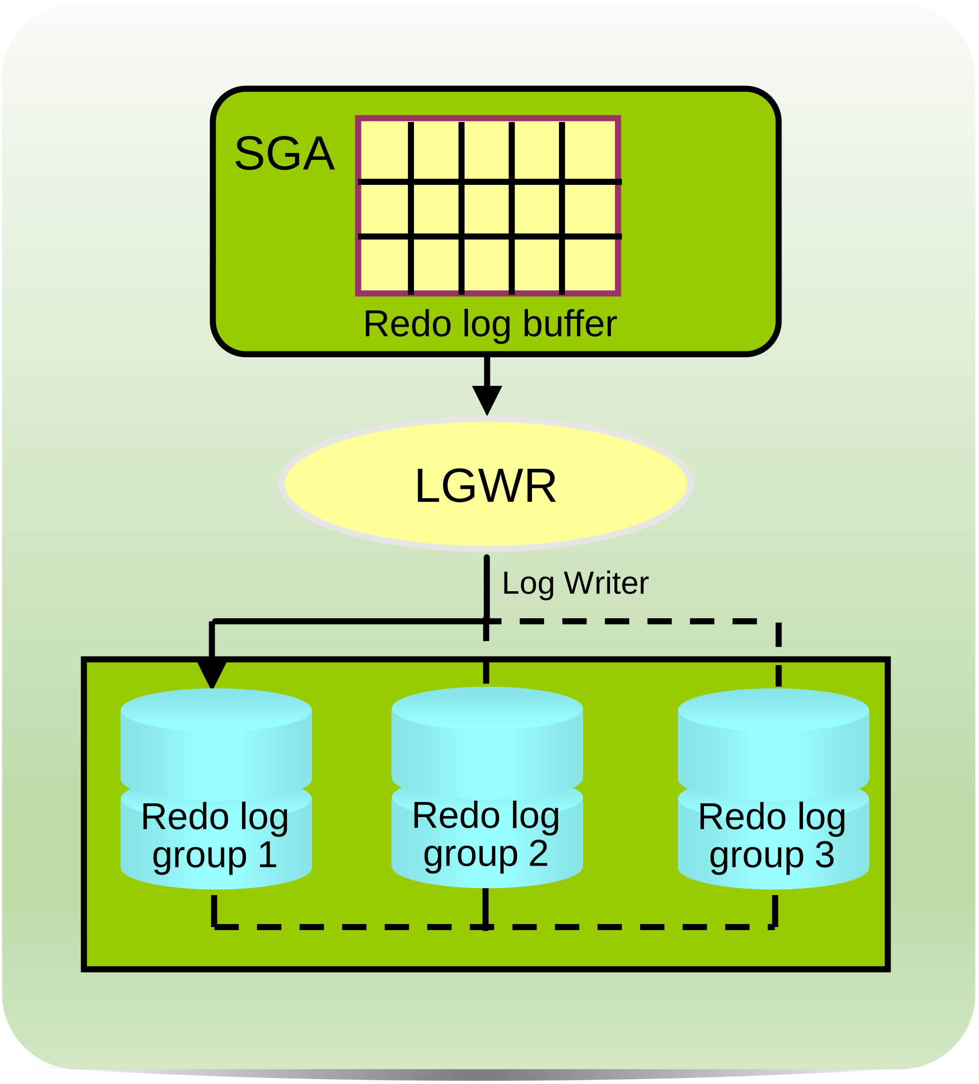
The Checkpoint (CKPT) Process

- CKPT is responsible for:
  - Updating data file headers with checkpoint information
  - Updating control files with checkpoint information
  - Signaling DBW*n* at full checkpoints



Redo Log Files and the Log Writer (LGWR) Process

- Redo log files:
  - Record changes to the database
  - Should be multiplexed to protect against loss
- Log Writer (LGWR) writes:
  - At commit
  - When one-third full
  - Every three seconds
  - Before DBWn writes
  - Before clean shutdowns

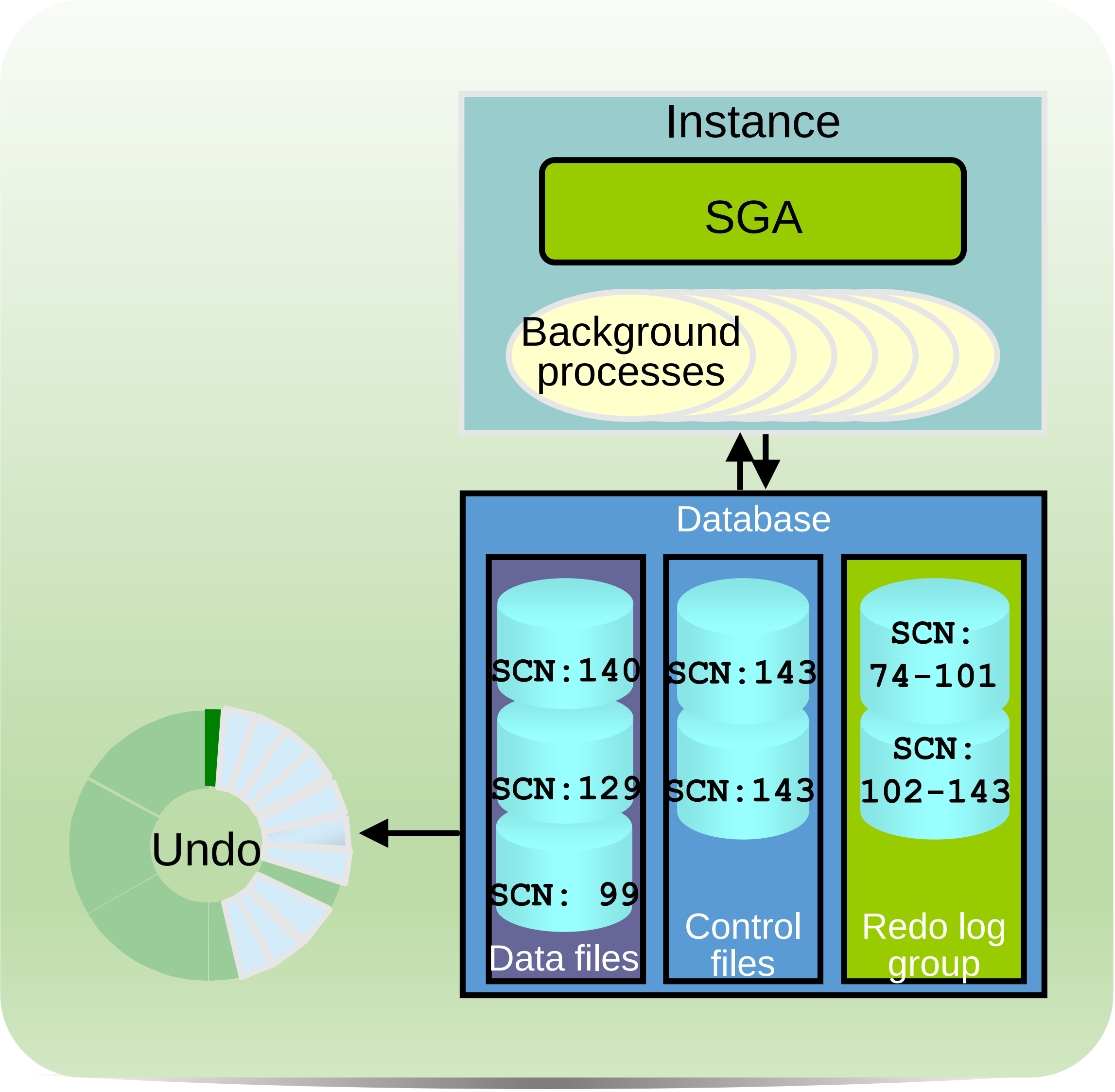


Automatic Instance Recovery or Crash Recovery

- Automatic instance or crash recovery:
  - Is caused by attempts to open a database whose files are not synchronized on shutdown
  - Uses information stored in redo log groups to synchronize files
  - Involves two distinct operations:
    - Rolling forward: Redo log changes (both committed and uncommitted) are applied to data files.
    - Rolling back: Changes that are made but not committed are returned to their original state.

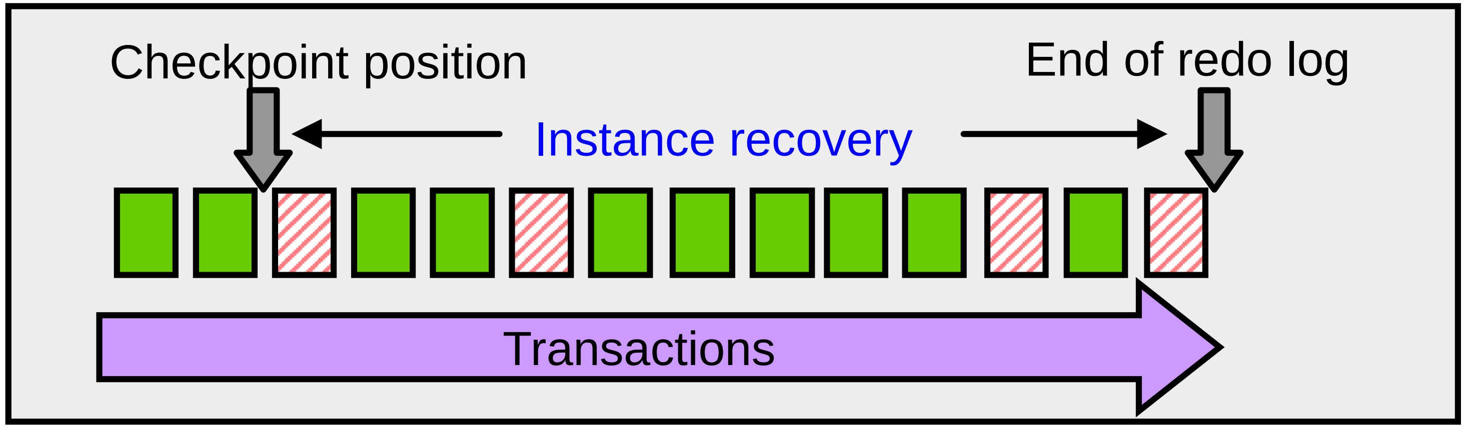
Phases of Instance Recovery

1. Instance startup (data files are out of sync)
2. Roll forward (redo)
3. Committed and uncommitted data in files
4. Database opened
5. Roll back (undo)
6. Committed data in files



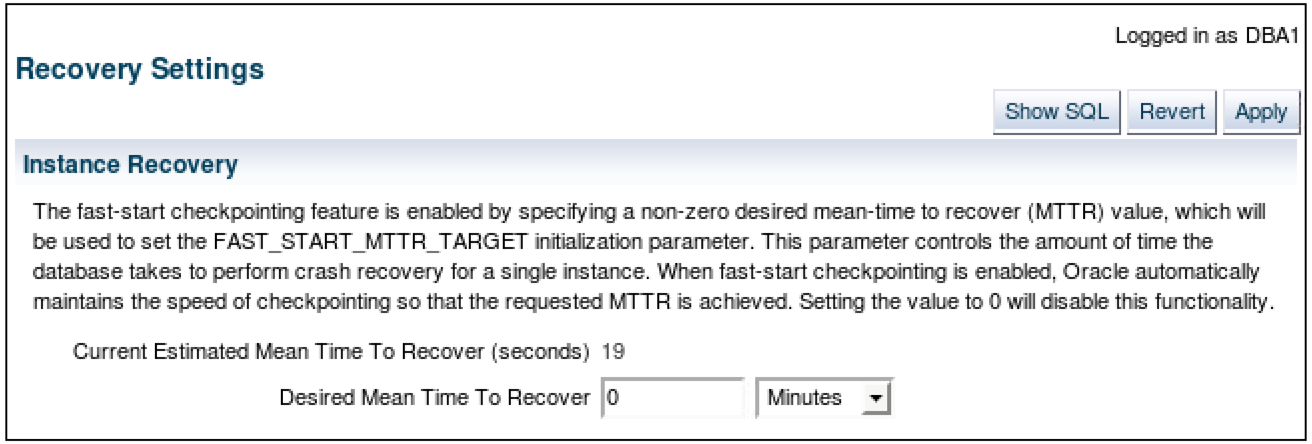
Tuning Instance Recovery

- During instance recovery, the transactions between the checkpoint position and the end of the redo log must be applied to data files.
- You tune instance recovery by controlling the difference between the checkpoint position and the end of the redo log.



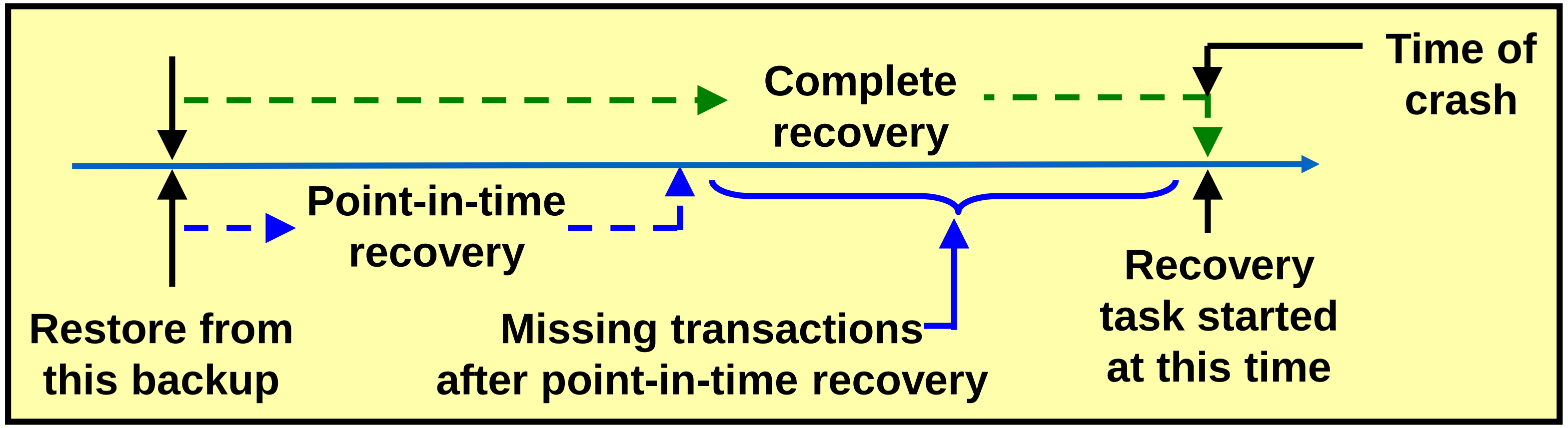
Using the MTTR Advisor

- Specify the desired time in seconds or minutes.
- The default value is 0 (disabled).
- The maximum value is 3,600 seconds (one hour).

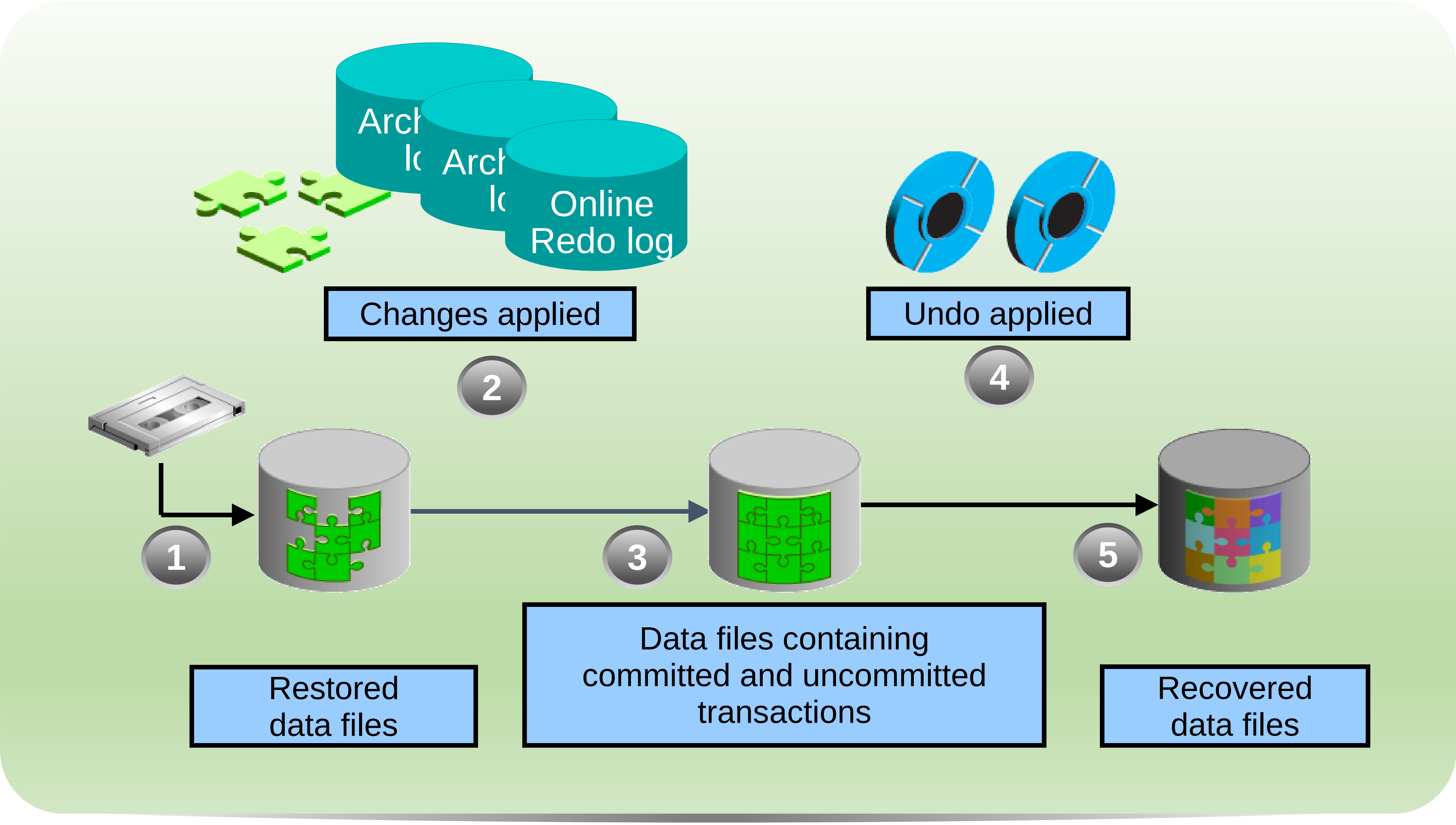


Comparing Complete and Incomplete Recovery

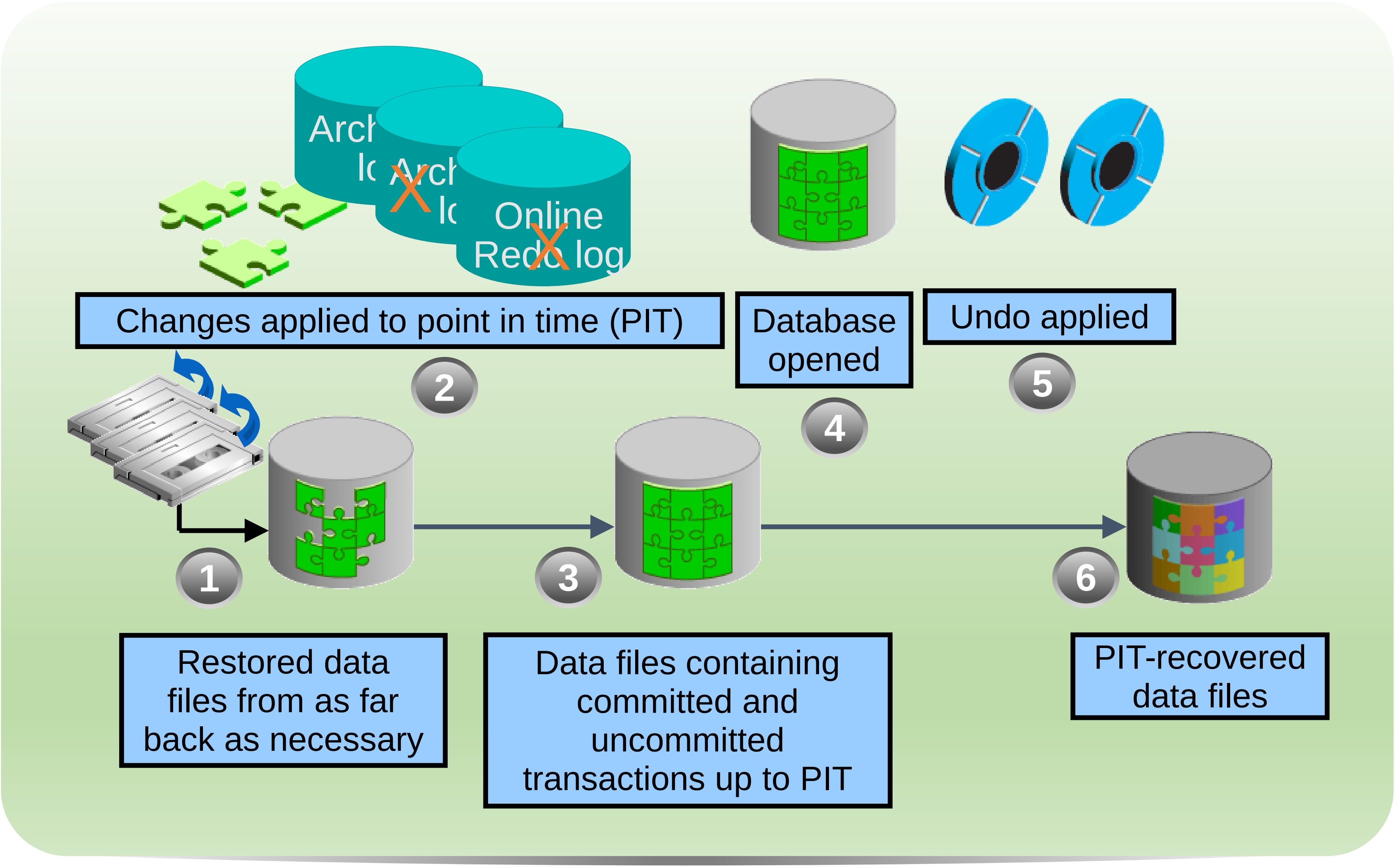
- Recovery can have two kinds of scope:
  - Complete recovery: Brings the database or tablespace up to the present, including all committed data changes made to the point in time when the recovery was requested
  - Incomplete or point-in-time recovery (PITR): Brings the database or tablespace up to a specified point in time in the past, before the recovery operation was requested



The Complete Recovery Process



The Point-in-Time Recovery Process



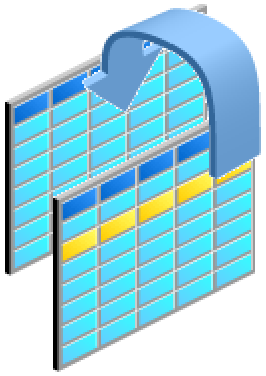
Oracle Data Protection Solutions

Backup and Recovery Objective	Recovery Time Objective (RTO)	Oracle Solution
Physical data protection	Hours/Days	Recovery Manager Oracle Secure Backup
Logical data protection	Minutes/Hours	Flashback Technologies
Recovery analysis	Minimize time for problem identification and recovery planning	Data Recovery Advisor

Disaster Recovery Objective	Recovery Time Objective (RTO)	Oracle Solution
Physical data protection	Seconds/Minutes	Data Guard Active Data Guard

Flashback Technology

- Use Flashback technology for:
  - Viewing past states of data
  - Winding data back and forth in time
  - Assisting users in error analysis and recovery



For error analysis:	For error recovery:
Oracle Flashback Query	Oracle Flashback Transaction Backout
Oracle Flashback Versions Query	Oracle Flashback Table
Oracle Flashback Transaction Query	Oracle Flashback Drop
	Oracle Flashback Database

Summary

- In this lesson, you should have learned how to:
  - Identify DBA responsibilities regarding database backup and recovery
  - Identify the types of failure that can occur in an Oracle database
  - Describe instance recovery
  - Describe complete and incomplete recovery

Backup and Recovery Configuration

Objectives

- After completing this lesson, you should be able to:
  - Configure the fast recovery area
  - Multiplex the control file
  - Multiplex redo log files
  - Configure ARCHIVELOG mode

Configuring for Recoverability

- Configure your database for maximum recoverability by:
  - Scheduling regular backups
  - Multiplexing control files
  - Multiplexing redo log groups
  - Retaining archived copies of redo logs

Configuring the Fast Recovery Area

- Fast recovery area:
  - Strongly recommended for simplified backup storage management
  - Storage space (separate from working database files)
  - Location specified by the `DB_RECOVERY_FILE_DEST` parameter
  - Size specified by the `DB_RECOVERY_FILE_DEST_SIZE` parameter
  - Large enough for backups, archived logs, flashback logs, multiplexed control files, and multiplexed redo logs
  - Automatically managed according to your retention policy
- Configuration of the fast recovery area includes specifying the location, size, and retention policy.

```
ALTER SYSTEM SET db_recovery_file_dest = directory | disk group
ALTER SYSTEM SET db_recovery_file_destsize = integer [K | M | G]
```

Monitoring the Fast Recovery Area

- Monitor the fast recovery area to ensure that it does not reach its capacity.
- The instance will pause if there isn't enough space in the fast recovery area to create an archived log.
- Query the `V$RECOVERY_FILE_DEST` view to determine the current location, disk quota, space in use, space reclaimable by deleting files, and total number of files in the fast recovery area.
- Query the `V$RECOVERY_AREA_USAGE` view to determine the percentage of the total disk quota used by different types of files.
- You can also use GUI tools such as Enterprise Manager Cloud Control to monitor the space usage.

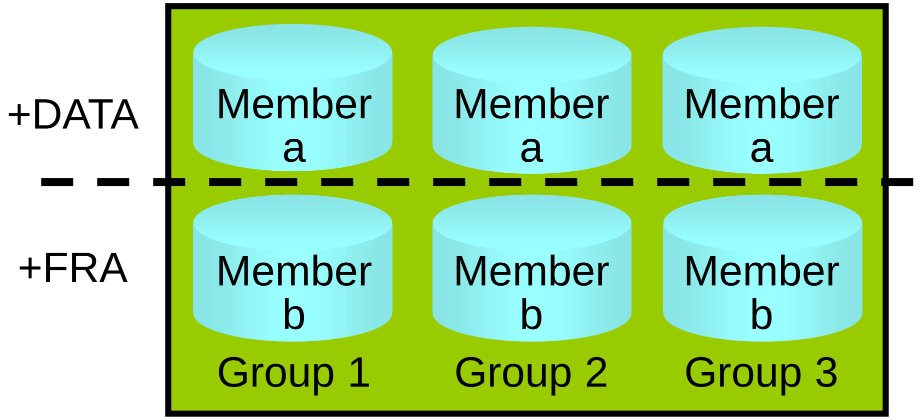
Multiplexing Control Files

- To protect against database failure, your database should have multiple copies of the control file.

	ASM Storage	**File System Storage**
Best Practice	One copy on each disk group (such as +DATA and +FRA)	At least two copies, each on a separate disk (at least one on a separate disk controller)
Steps to create additional control files	No additional control file copies required	1. Alter the SPFILE with the `ALTER SYSTEM SET control_files` command. 2. Shut down the database. 3. Copy the control file to a new location. 4. Open the database and verify the addition of the new control file.

Redo Log Files

- Multiplex redo log groups to protect against media failure and loss of data. This increases database I/O. It is suggested that redo log groups have:
  - At least two members (files) per group
  - Each member:
    - On a separate disk or controller if using file system storage
    - In a separate disk group (such as +DATA and +FRA) if using ASM



Note: Multiplexing redo logs may impact overall database performance.

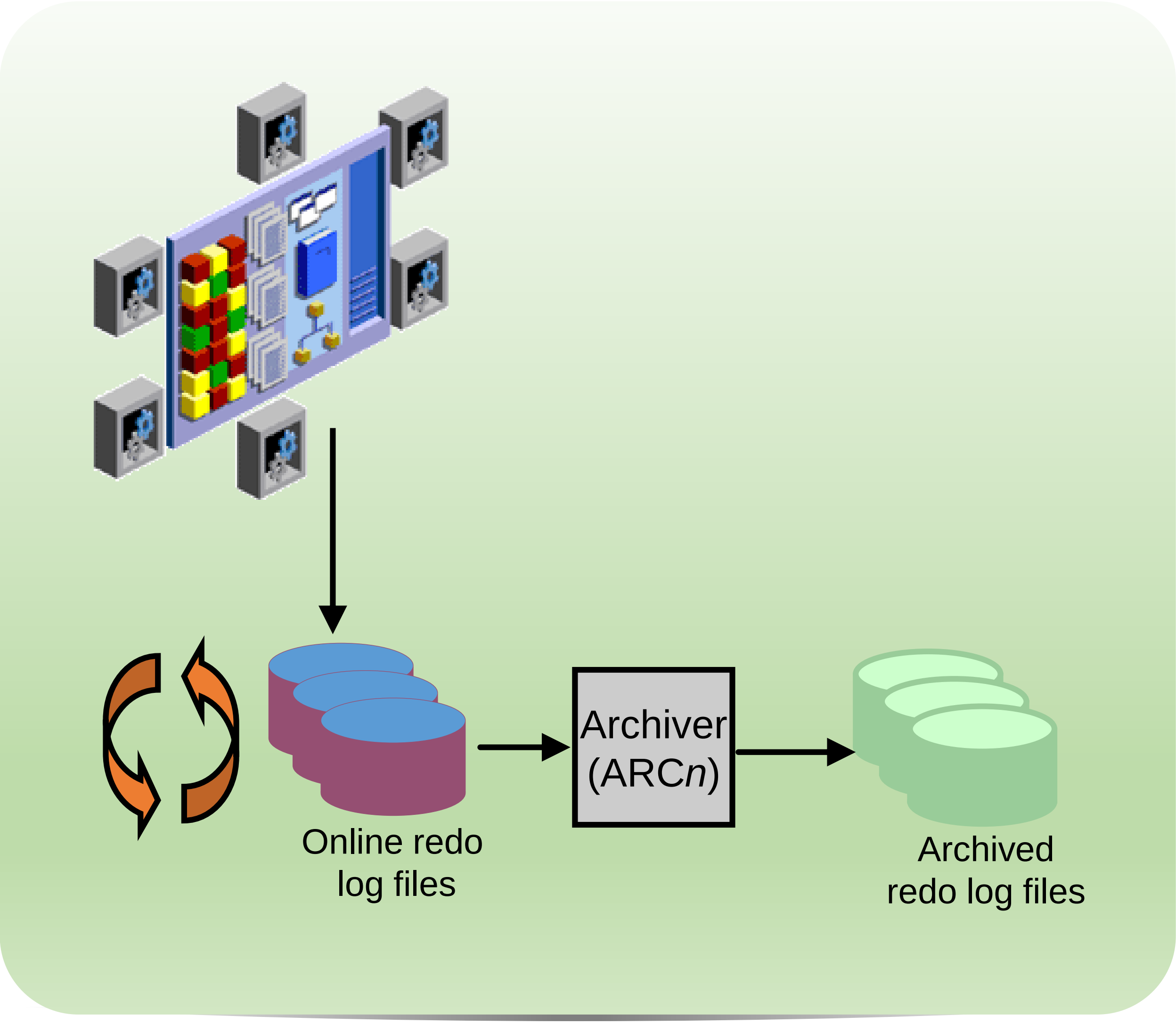
Multiplexing the Redo Log

- Add a member to an existing log group:
  - Navigate to the Redo Log Groups page in Enterprise Manager Database Express
  - Use the `ALTER DATABASE` command

```
SQL> ALTER DATABASE
  2   ADD LOGFILE MEMBER '/u01/app/oracle/oradata/orcl/redo1a.log'
  3   TO GROUP 1;
```

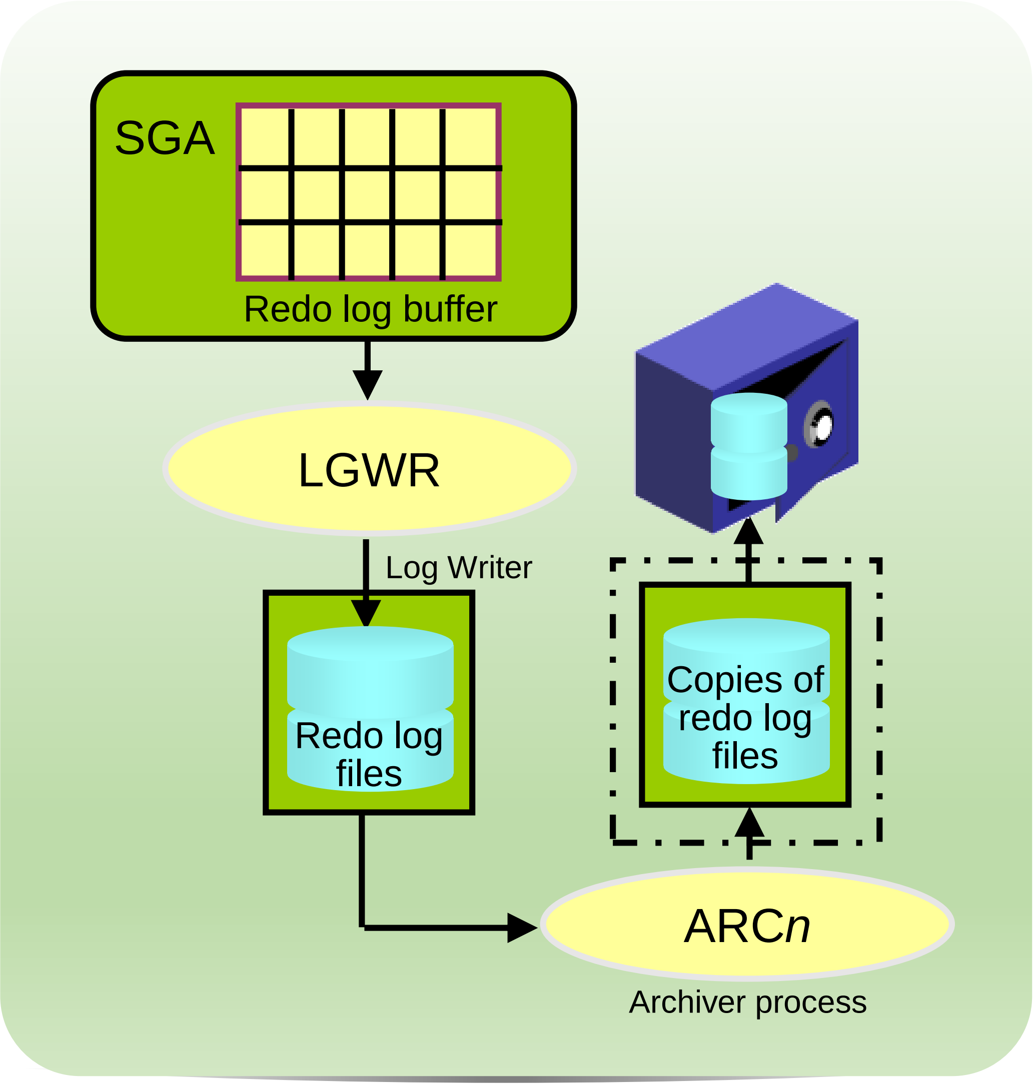
Creating Archived Redo Log Files

- To preserve redo information, create archived copies of redo log files by performing the following steps:
  1. Specify the archived redo log file-naming convention.
  2. Specify one or more archived redo log file locations.
  3. Place the database in `ARCHIVELOG` mode.



Archiver (ARC*n*) Process

- Archiver (ARC*n*):
  - Automatically archives online redo log files when the database is in `ARCHIVELOG` mode
  - Preserves a record of all changes made to the database



Archived Redo Log Files: Naming and Destinations

- Use the `LOG_ARCHIVE_DEST` initialization parameter to specify a single destination.
- Use the `LOG_ARCHIVE_DEST_n` initialization parameters to archive to two or more locations.
- If you are using file system storage, it is recommended that you add multiple locations across different disks.
- If the fast recovery area is enabled, `USE_DB_RECOVERY_FILE_DEST` is specified by default as an archived redo log file destination.

Configuring ARCHIVELOG Mode

- You can use SQL commands as follows:
  1. Shut down the database instance if it is open.
  2. Mount the database.
  3. Issue the ALTER DATABASE ARCHIVELOG command.

```
SQL> shutdown immediate
SQL> startup mount
SQL> alter database archivelog;
SQL> alter database open;
```

- You can also use Enterprise Manager Cloud Control to place the database in ARCHIVELOG mode.

Summary

- In this lesson, you should have learned how to:
  - Configure the Fast Recovery Area
  - Multiplex the control file
  - Multiplex redo log files
  - Configure ARCHIVELOG mode

Practice 17: Overview

- 17-1: Verifying that the Control File is Multiplexed
- 17-2: Checking Storage Availability
- 17-3: Configuring the Size of the Fast Recovery Area
- 17-4: Verifying that the Redo Log File is Multiplexed
- 17-5: Verifying that ARCHIVELOG Mode is Configured

Creating Database Backups

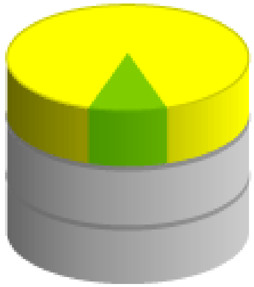
Objectives

- After completing this lesson, you should be able to:
  - Create consistent database backups
  - Back up your database without shutting it down
  - Create incremental backups
  - Modify the DBCS default backup configuration
  - Create DBCS backups

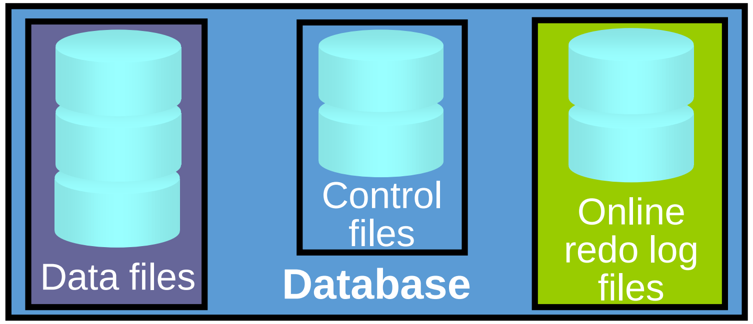
Understanding Types of Backups

- You can understand different types of backups by becoming familiar with these concepts:
  - Backup terminology
  - Types of backups
  - RMAN backup types

Backup Terminology



- *Backup strategy* may include:
  - The entire database (whole)
  - A portion of the database (partial)
- *Backup type* may indicate inclusion of:
  - All data blocks within your chosen files (full)
  - Only information that has changed since a previous backup (incremental)
    - Cumulative (changes since last level 0)
    - Differential (changes since last incremental)
- *Backup mode* may be:
  - Offline (consistent, cold)
  - Online (inconsistent, hot)



Understanding Types of Backups

- Backups may be stored as:
  - Image copies
  - Backup sets

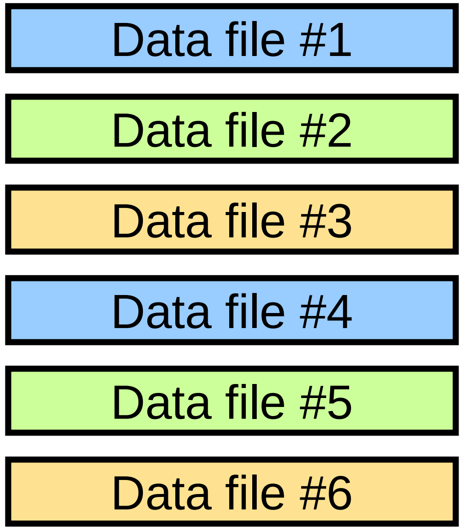
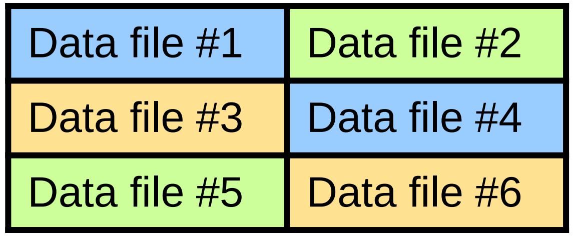


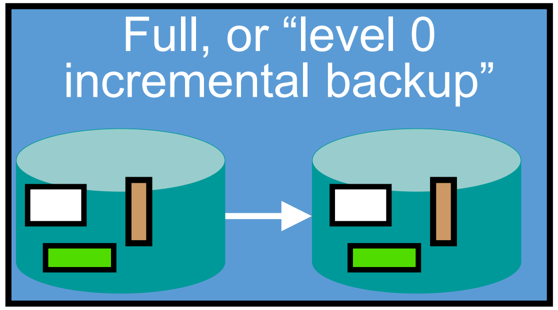
Image copies
(Duplicate data and log files in OS format)

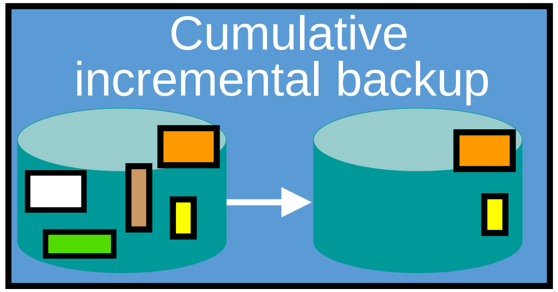


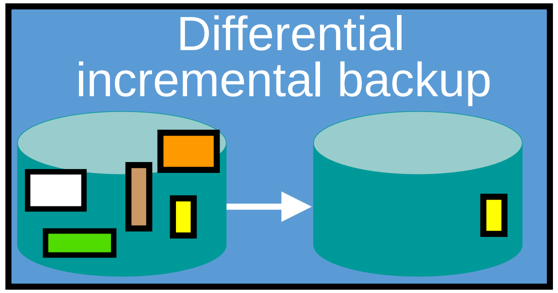
Backup set
(Binary, compressed files in Oracle proprietary format)

RMAN Backup Types

- A *full backup* contains all used data file blocks.
- A *level 0 incremental backup* is equivalent to a full backup that has been marked as level 0.
- A *cumulative level 1 incremental backup* contains only blocks modified since the last level 0 incremental backup.
- A *differential level 1 incremental backup* contains only blocks modified since the last incremental backup.







Using Recovery Manager (RMAN)

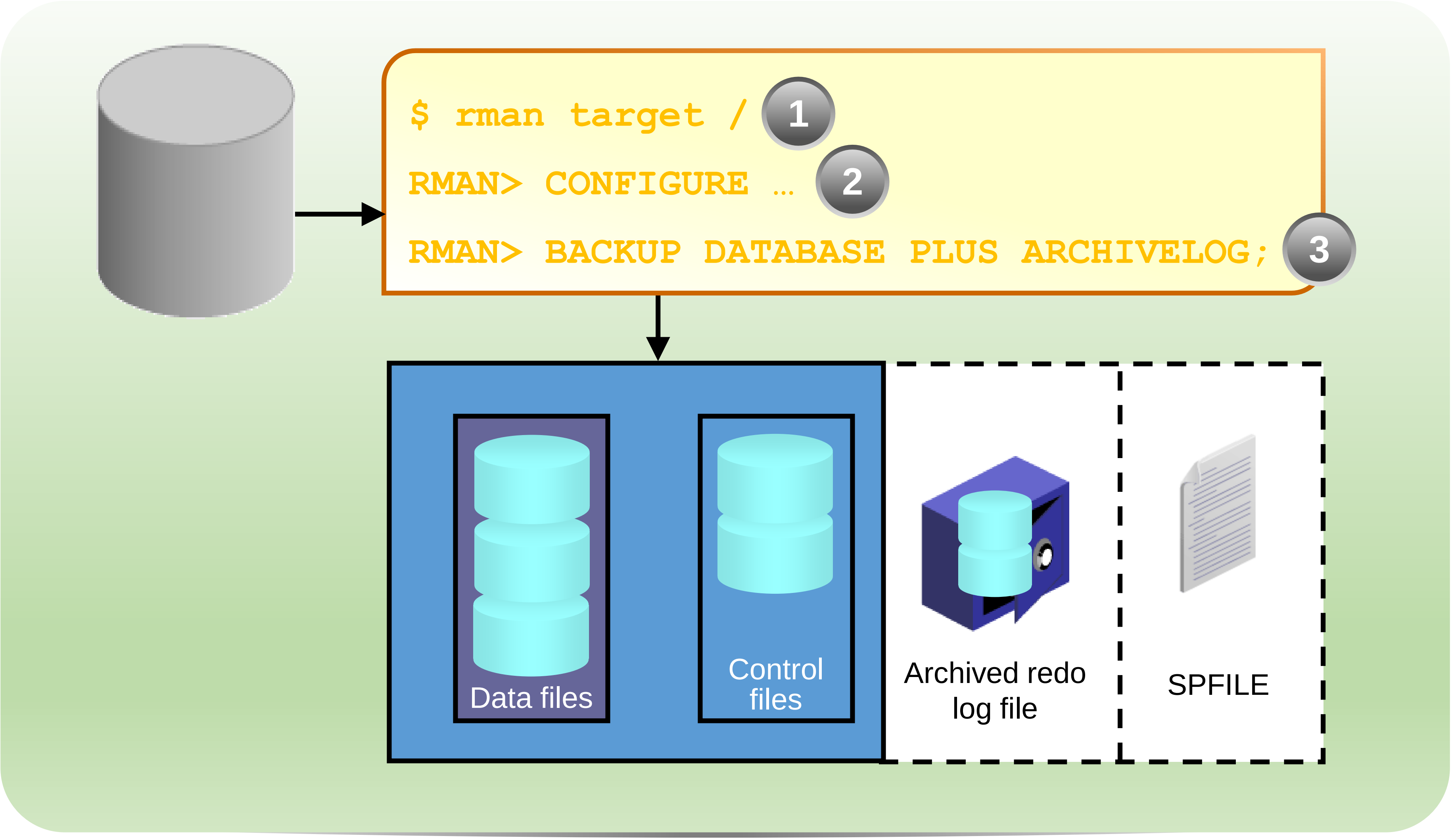
- Provides a powerful control and scripting language
- Includes a published API that enables the interface with the most popular backup software
- Backs up data, control, the archived redo log, and server parameter files
- Backs up files to disk or tape
- Is integrated with Enterprise Manager Cloud Control
- Is used by Oracle Database Cloud Service, Oracle Database Backup Service, and other Oracle Cloud services for automated backups
- Can be used with Oracle Database Cloud Service if no backup configuration was selected when the database deployment was created

Backing Up the Control File to a Trace File

- Control files can be backed up to a trace file, generating a SQL command to re-create the control file.
- Control file trace backups may be used to recover from the loss of all control files.

```
ALTER DATABASE BACKUP CONTROLFILE TO TRACE
```

Using RMAN Commands to Create Backups



Backing Up Databases on DBCS

- Database Cloud Service provides a backup feature that backs up:
  - The database
  - Database configuration files
  - Grid Infrastructure configuration files (on deployments hosting an Oracle RAC database)
  - System and cloud tooling files
- The backup feature relies on the following, which are installed in the database deployment:
  - System utilities
  - Oracle Database utilities
  - Oracle Database Backup Cloud Service

Backup Destination Choices

Backup Destination	Description	Retention
Both Cloud Storage and Local Storage	Backups are created automatically and stored both on local compute node storage and on an Oracle Storage Cloud Service container.	30 days 7 most recent days' backups available on local storage
Cloud Storage Only	Backups are created automatically and stored only on an Oracle Storage Cloud Service container.	30 days
None	No backups are created.	

Backup Configuration

- Full (level 0) backup of the database followed by rolling incremental (level 1) backups on a seven-day cycle
- Full backup of selected database configuration files
- Full backup of selected system files
- Automatic backups daily at a time between 11 PM (23:00) and 3 AM (03:00), with the specific time set during database deployment creation
- Encryption:
  - Both Cloud Storage and Local Storage: All backups to cloud storage are encrypted; backups of Enterprise Edition databases to local storage are encrypted; backups of Standard Edition databases to local storage are not encrypted.
  - Cloud Storage Only: All backups to cloud storage are encrypted.

Creating an On-Demand Backup

- Click “Backup Now” on the Oracle Database Cloud Service Instance Administration page.
- Command-line interfaces for backups:
  - For single-instance databases, use the `bkup_api` utility.
  - For RAC databases, use the `raccli` utility.

Customizing the Backup Configuration

Customization	Description	Utility to Use or File to Edit
Backup Configuration Settings	Persistent configuration settings	Recovery Manager (RMAN)
System Files	System files and directories	`/home/oracle/bkup/oscfg.spec` file
Database Configuration Files	Wallet, initialization parameter, network configuration files	`/home/oracle/bkup/dbcfg.spec` file
Retention Period	Backup retention period (days)	`bkup_api` utility
Cycle Period	Backup cycle period (days)	`bkup_api` utility
Frequency	Time that `bkup_api` is run	`/etc/crontab` file

Summary

- In this lesson, you should have learned how to:
  - Create consistent database backups
  - Back up your database without shutting it down
  - Create incremental backups
  - Modify the DBCS default backup configuration
  - Create DBCS backups

Practice 18: Overview

- 18-1: Backing Up the Control File
- 18-2: Verifying Automatic Backups of the Control File and SPFILE
- 18-3: Checking Storage Availability
- 18-4: Creating a Whole Database Backup
- 18-5: Creating a Partial Database Backup

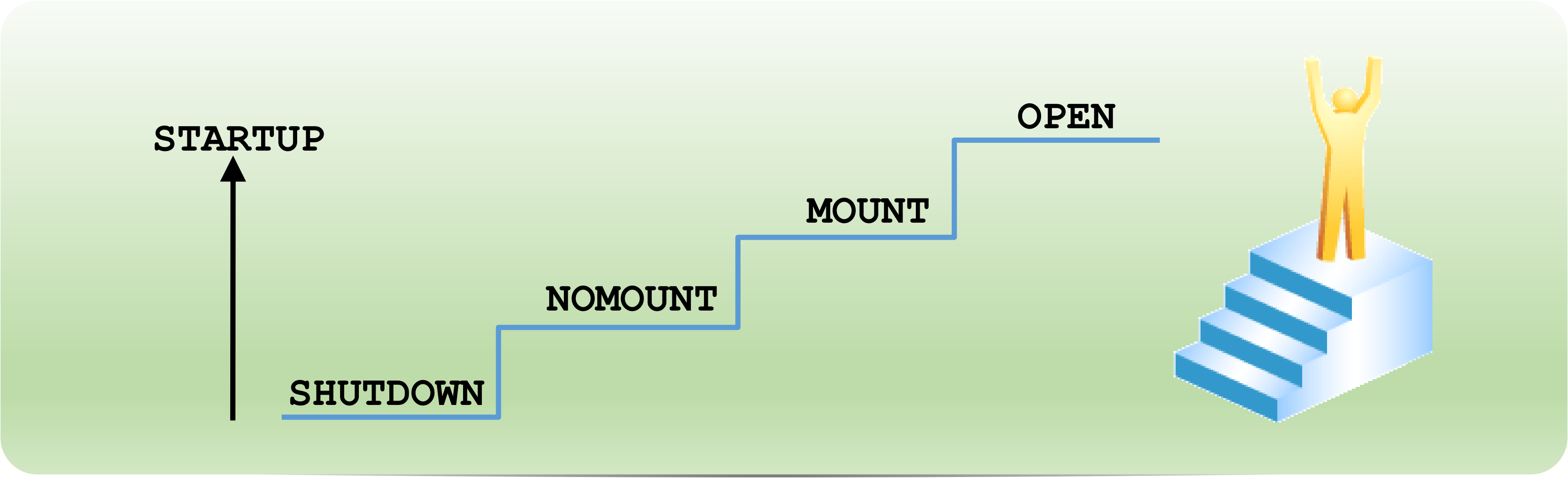
Performing Database Recovery

Objectives

- After completing this lesson, you should be able to:
  - Determine the need for performing recovery
  - Describe and use available options, such as Recovery Manager (RMAN) and the Data Recovery Advisor
  - Recover data files
  - Restore and recover a DBCS database

Opening a Database

- To open a database:
  - All control files must be present and synchronized
  - All online data files must be present and synchronized
  - At least one member of each redo log group must be present



Keeping a Database Open

- After the database is open, it fails in case of the loss of:
  - Any control file
  - A data file belonging to the system or undo tablespaces
  - An entire redo log group (as long as at least one member of the group is available, the instance remains open)

Data Recovery Advisor

- Fast detection, analysis, and repair of failures
- Handling of down time and run time failures
- Minimizing disruptions for users
- User interfaces:
  - Enterprise Manager Cloud Control
  - RMAN command line

Loss of a Control File

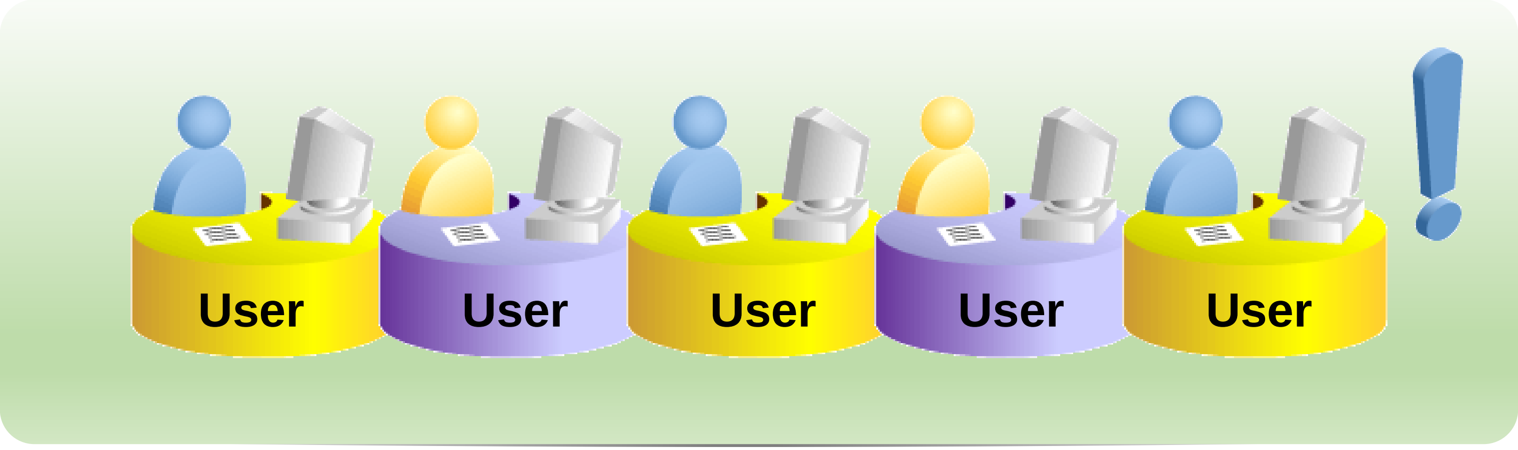
- If a control file is lost or corrupted, the instance normally aborts.
- If control files are stored as regular file system files:
  - Shut down the database
  - Copy an existing control file to replace the lost control file
- After the control file is successfully restored, open the database.

Loss of a Redo Log File

- If a member of a redo log file group is lost and if the group still has at least one member, note the following results:
  - Normal operation of the instance is not affected.
  - You receive a message in the alert log notifying you that a member cannot be found.
  - You can restore the missing log file by dropping the lost redo log member and adding a new member.
  - If the group with the missing log file has been archived, you can clear the log group to re-create the missing file.

Loss of a Data File in NOARCHIVELOG Mode

- If the database is in NOARCHIVELOG mode and if any data file is lost, perform the following tasks:
  1. Shut down the instance if it is not already down.
  2. Restore the entire database, including all data and control files, from the backup.
  3. Open the database.
  4. Have users re-enter all the changes that were made since the last backup.



Loss of a Noncritical Data File in `ARCHIVELOG` Mode

- If a data file is lost or corrupted, and if that file does not belong to the `SYSTEM` or `UNDO` tablespace, you restore and recover the missing data file.
- The other database files are available for users.

Loss of a System-Critical Data File in ARCHIVELOG Mode

- If a data file is lost or corrupted, and if that file belongs to the SYSTEM or UNDO tablespace, perform the following tasks:
  1. The instance may or may not shut down automatically. If it does not, use SHUTDOWN ABORT to bring the instance down.
  2. Mount the database.
  3. Restore and recover the missing data file.
  4. Open the database.

DBCS: Performing Recovery by Using the Console

- Restore from the most recent backup and perform complete recovery.
- Restore from a backup and recover to a specific data and time.
- Restore from a backup and recover to a specific system change number (SCN).

DBCS: Performing Recovery by Using the `dbaascli` Utility

- Use the `orec` subcommand of the `dbaascli` utility to restore and recover the database:
  - Restoring from the most recent backup and performing complete recovery

```
# dbaascli orec --args -latest
```

  - Restoring from a specific backup and performing point-in-time recovery

```
# dbaascli orec --args -pitr backup-tag
```

  - Restoring from the most recent backup and performing recovery through the specified system change number (SCN)

```
# dbaascli orec --args -scn SCN
```

  - Restoring from a specific long-term backup and performing point-in-time recovery

```
# dbaascli orec --args -keep -tag backup-tag
```

Summary

- In this lesson, you should have learned how to:
  - Determine the need for performing recovery
  - Describe and use available options, such as Recovery Manager (RMAN) and the Data Recovery Advisor
  - Recover data files
  - Restore and recover a DBCS database

Practice 19: Overview

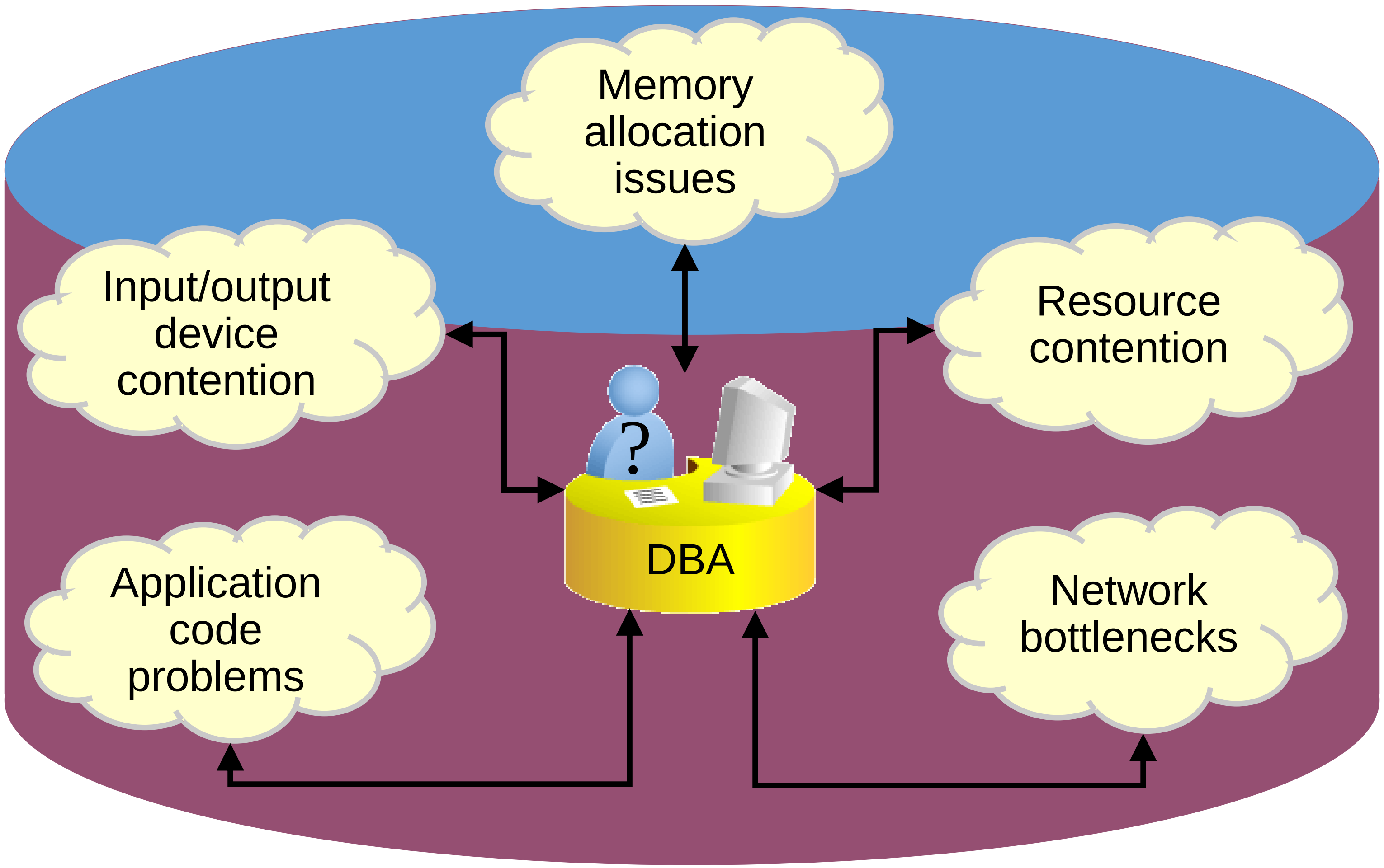
- 19-1: Recovering from an Essential Data File Loss
- 19-2: Recovering from an Application Data File Loss

Monitoring and Tuning Database Performance

Objectives

- After completing this lesson, you should be able to:
  - Describe the activities that you perform to manage database performance
  - Use Enterprise Manager Database Express and performance views to monitor database instance performance
  - Describe the Oracle performance tuning methodology
  - Describe the server statistics and metrics that are collected by the Oracle Database server
  - Configure and monitor memory components for optimal performance

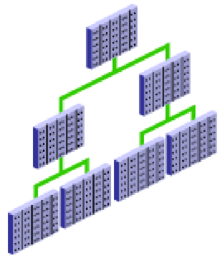
Performance Management Activities



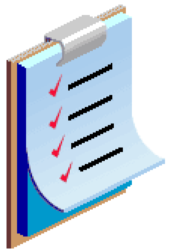
Performance Planning Considerations

System Architecture Investment

Scalability



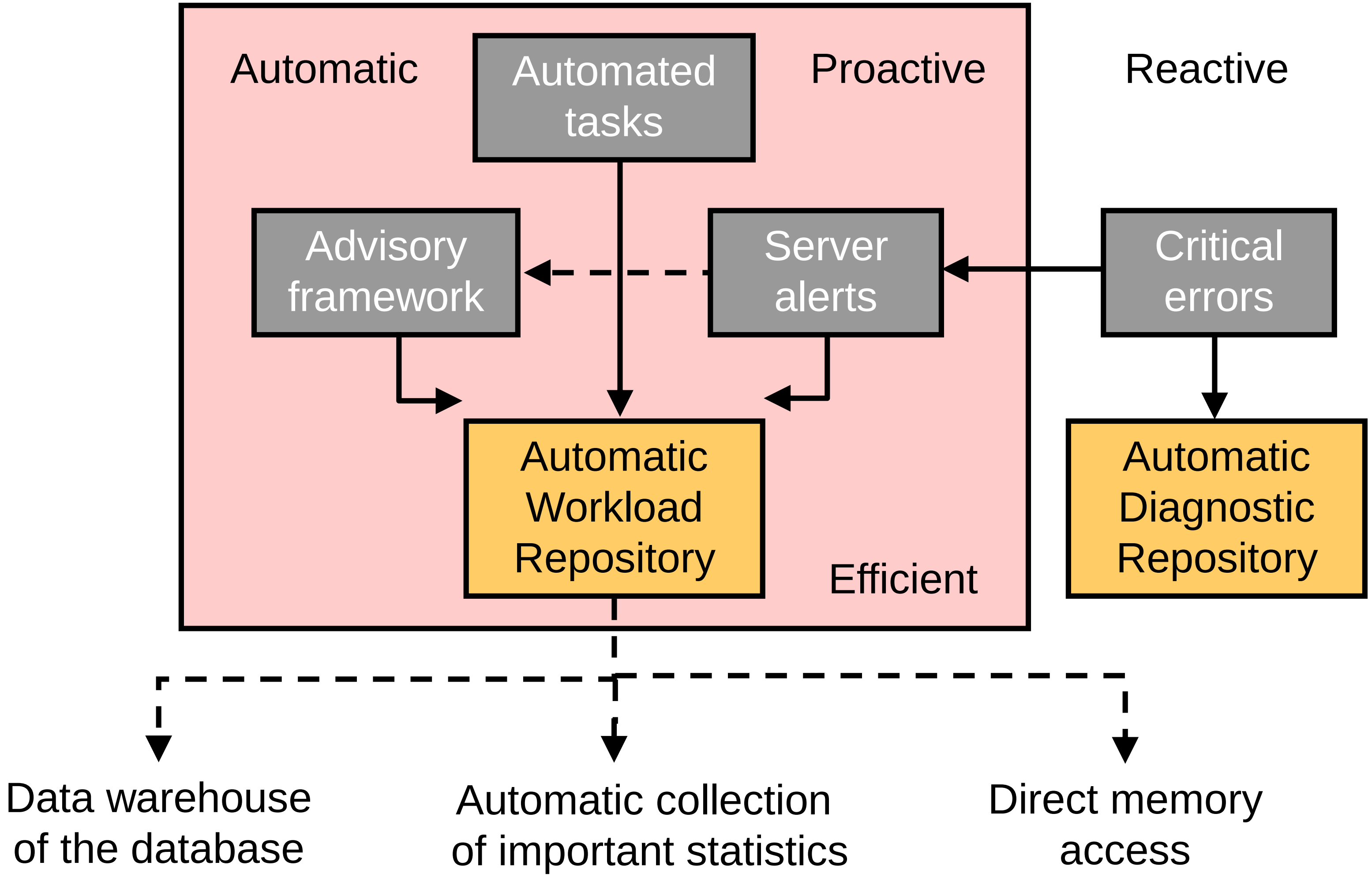
Application Design Principles



Workload Testing

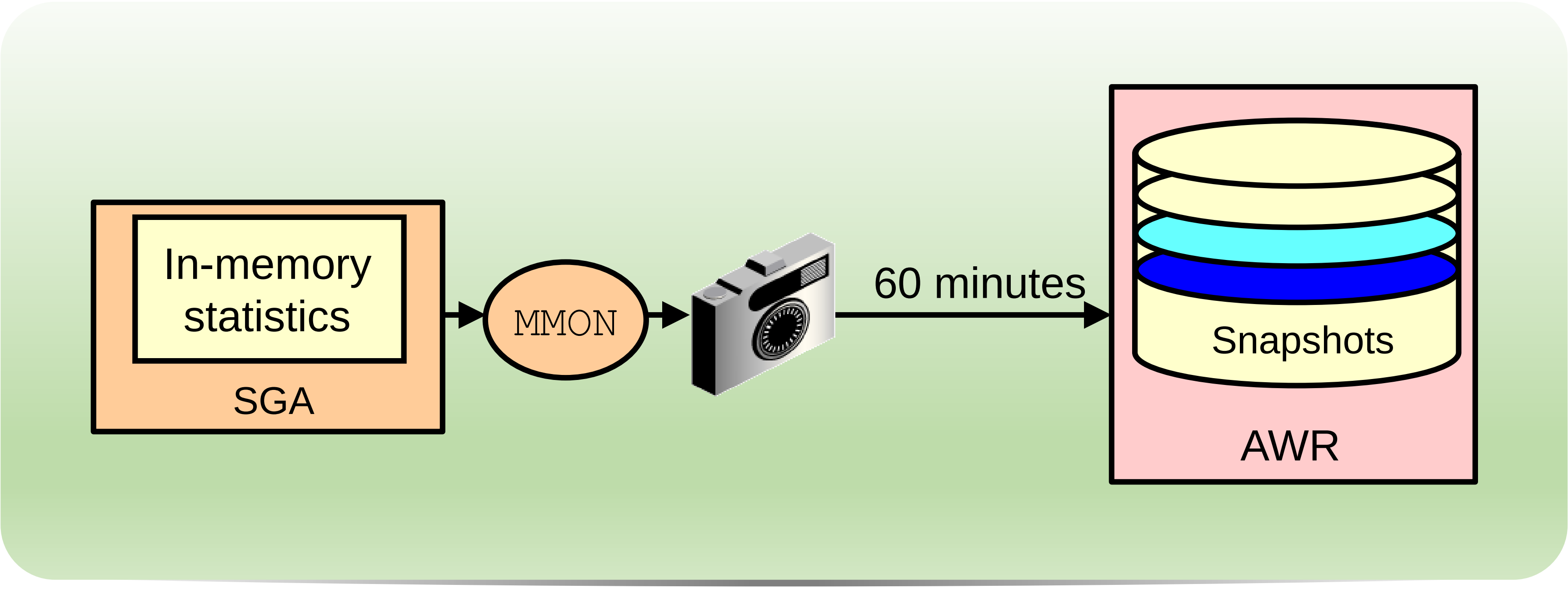
New Application Deployment

Database Maintenance



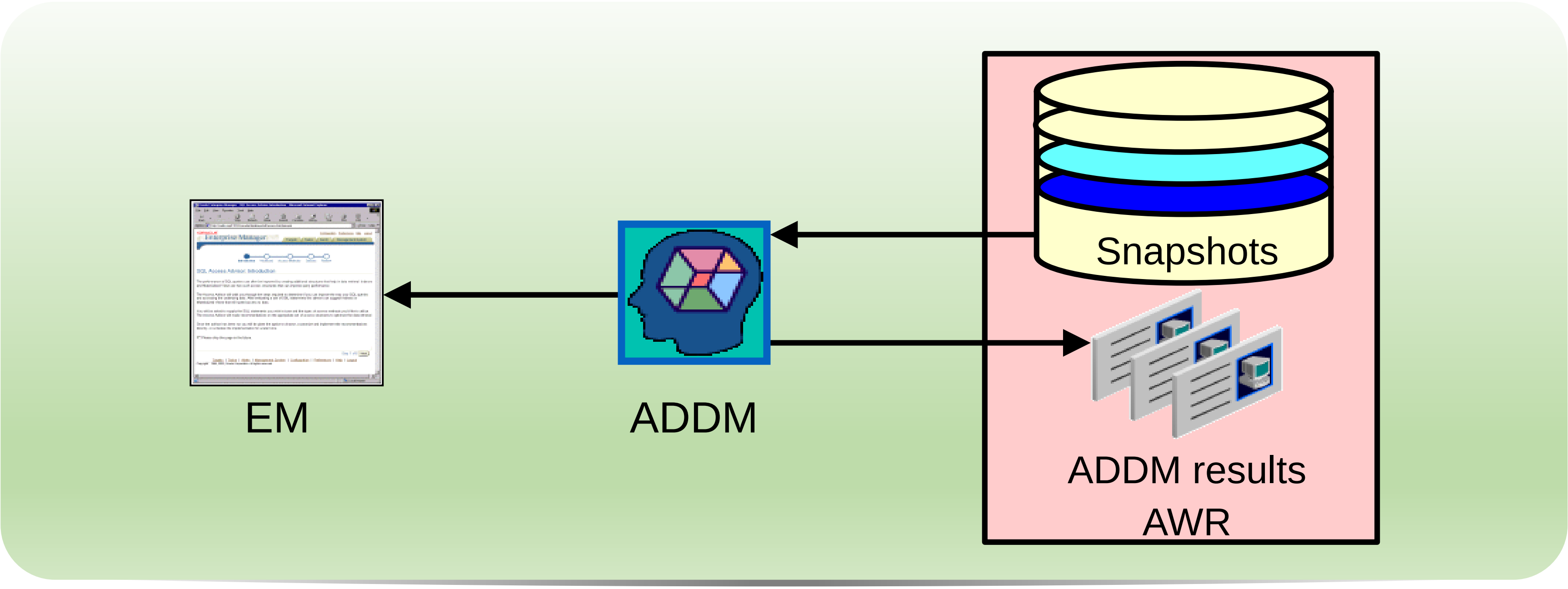
Automatic Workload Repository (AWR)

- 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

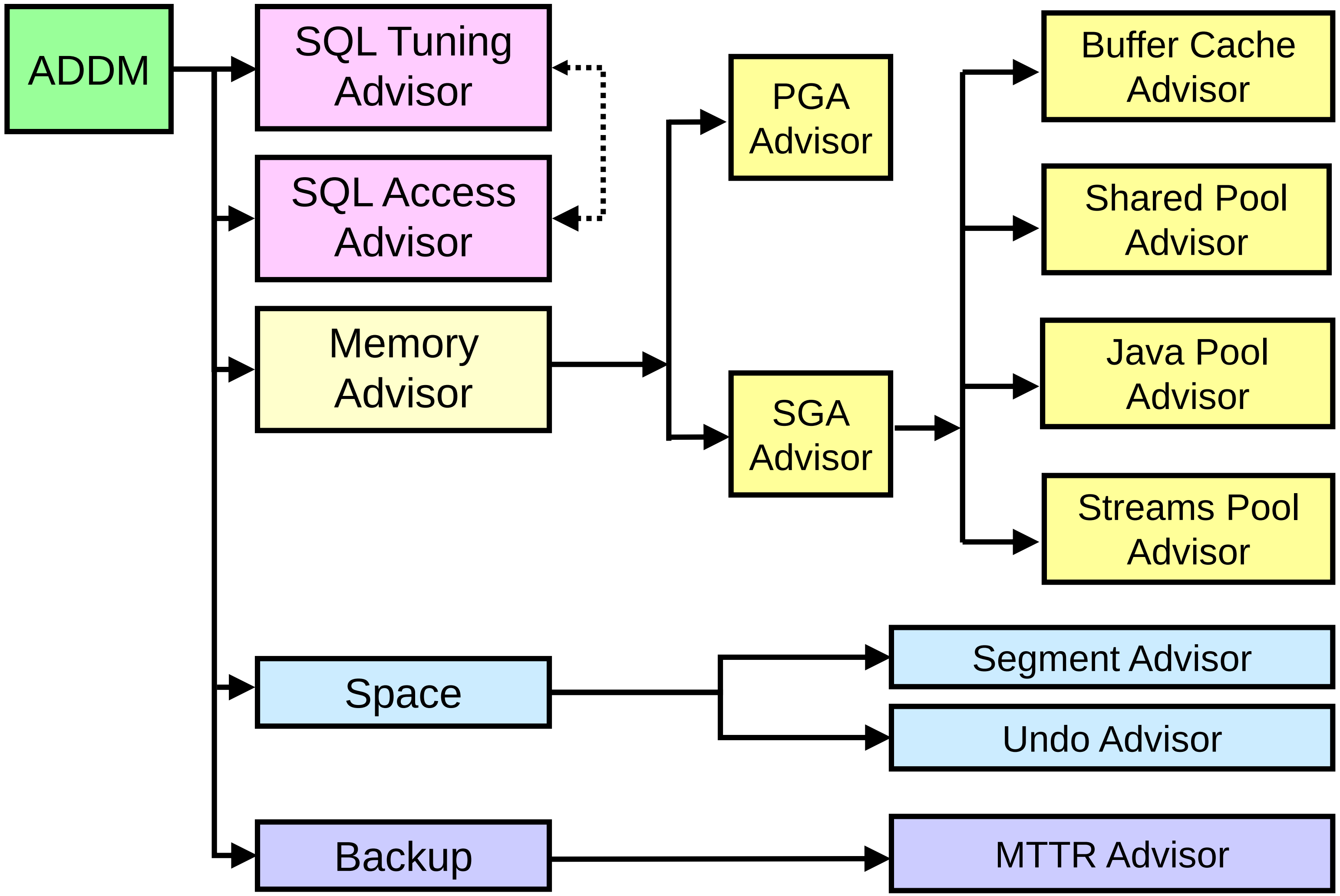


Automatic Database Diagnostic Monitor (ADDM)

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



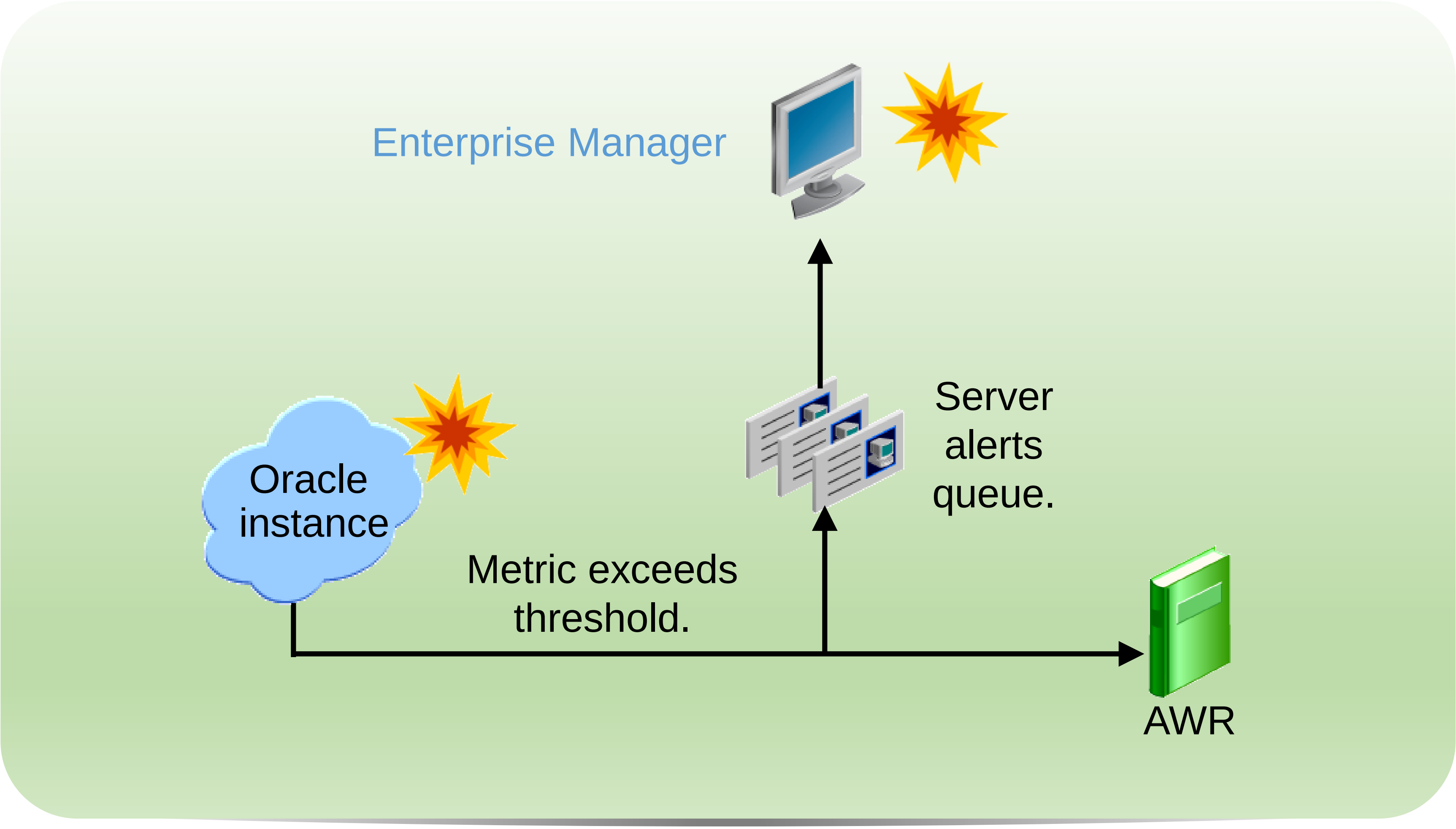
Advisory Framework



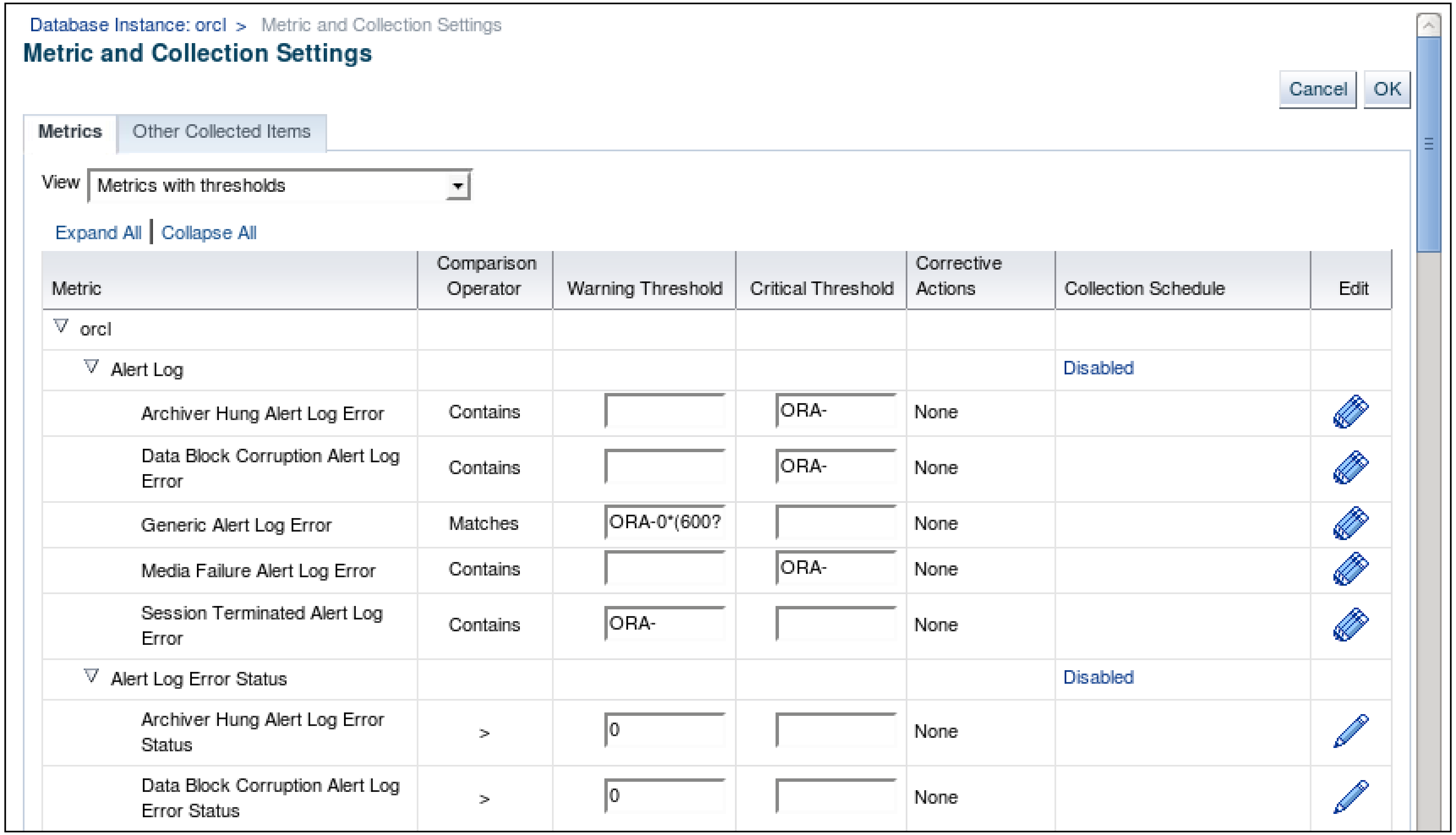
Automated Maintenance Tasks

- Autotask maintenance process:
  1. Maintenance window opens.
  2. Autotask background process schedules jobs.
  3. Scheduler initiates jobs.
  4. Resource Manager limits the impact of Autotask jobs.
- Default Autotask maintenance jobs:
  - Gathering optimizer statistics
  - Automatic Segment Advisor
  - Automatic SQL Advisor

Server-Generated Alerts



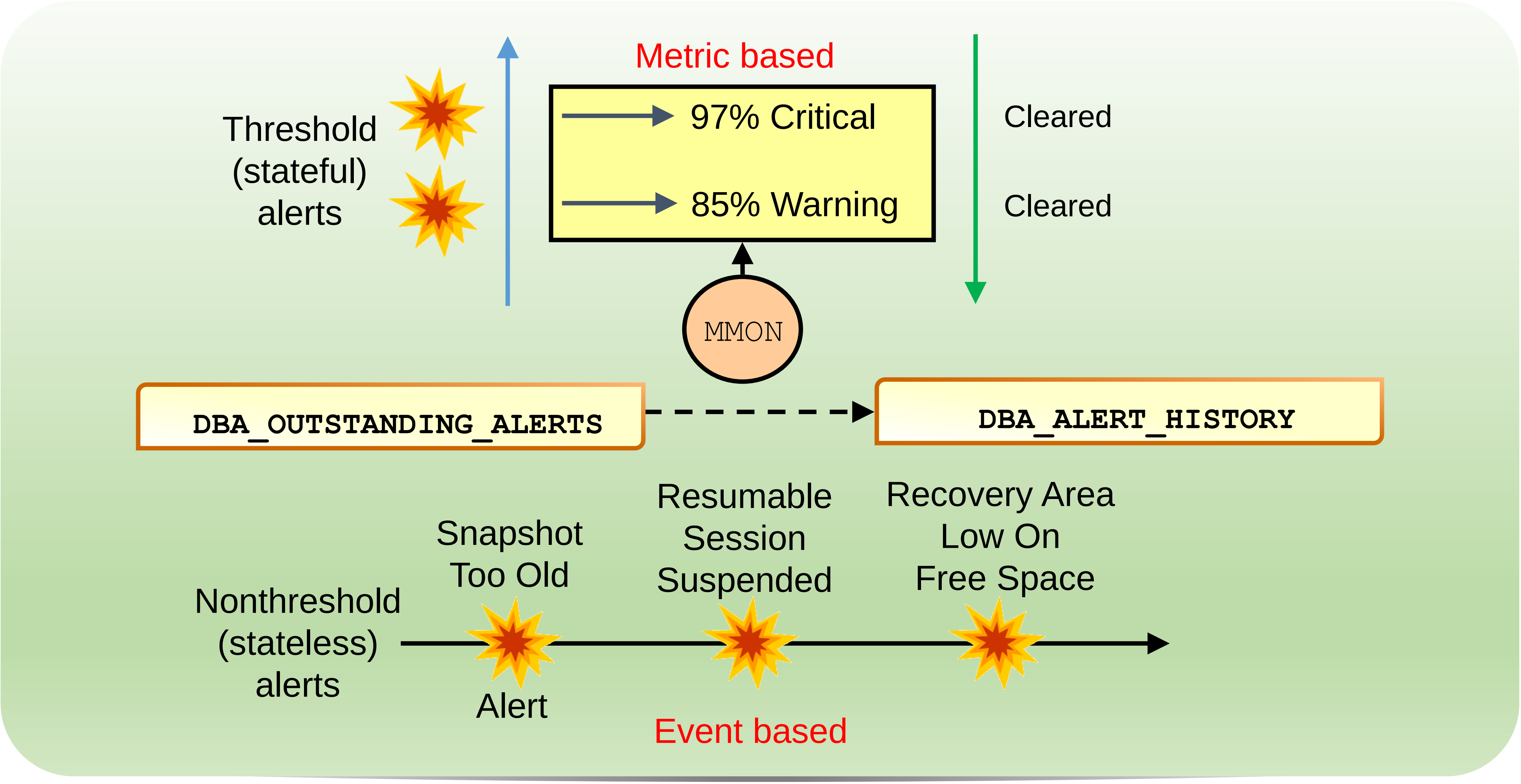
Setting Metric Thresholds



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$BUFFER_POOL_STATISTICS
V$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_CLASS
- WAIT_CLASS_ID
- WAIT_CLASS#
- WAIT_CLASS
- TOTAL_WAITS
- TIME_WAITED

V$SGASTAT
- POOL
- NAME
- BYTES

V$EVENT_NAME
- EVENT_NUMBER
- EVENT_ID
- NAME
- PARAMETER1
- PARAMETER2
- PARAMETER3
- WAIT_CLASS

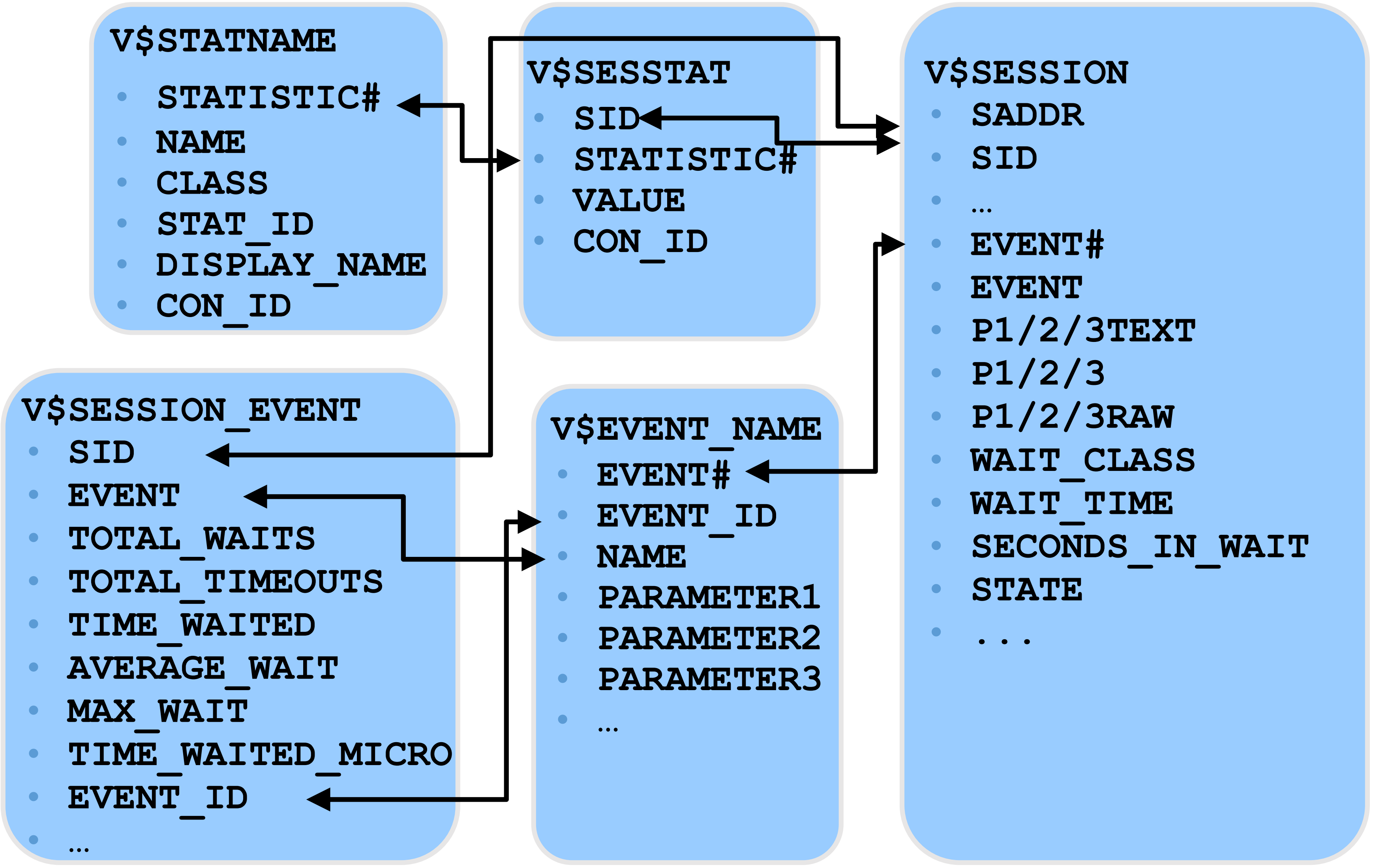
V$SYSTEM_EVENT
- EVENT
- TOTAL_WAITS
- TOTAL_TIMEOUTS
- TIME_WAITED
- AVERAGE_WAIT
- TIME_WAITED_MICRO

Monitoring Wait Events

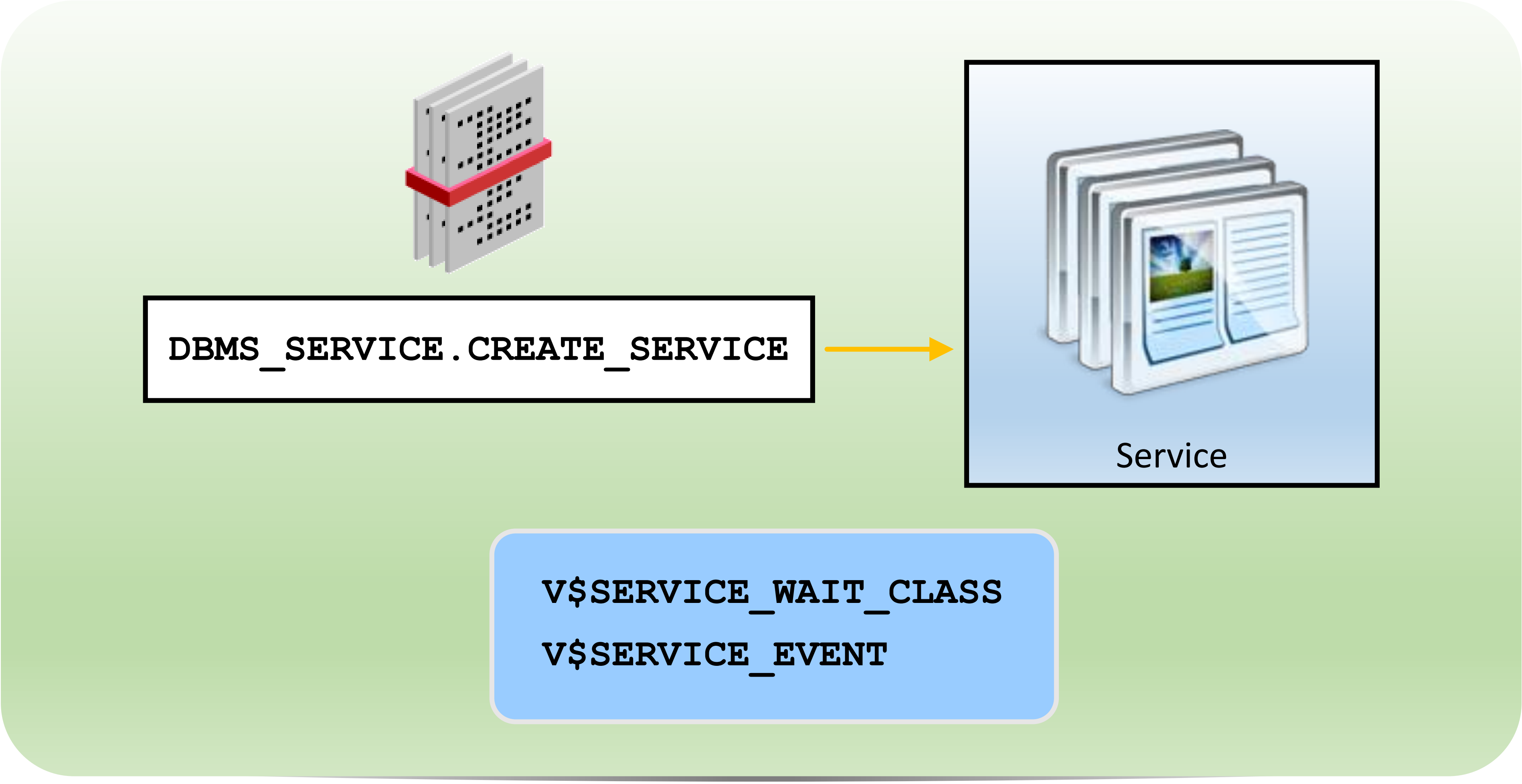
Wait events: Statistics indicating the server process had to wait for an event to complete

V$EVENT_NAME

Monitoring Sessions



Monitoring Services



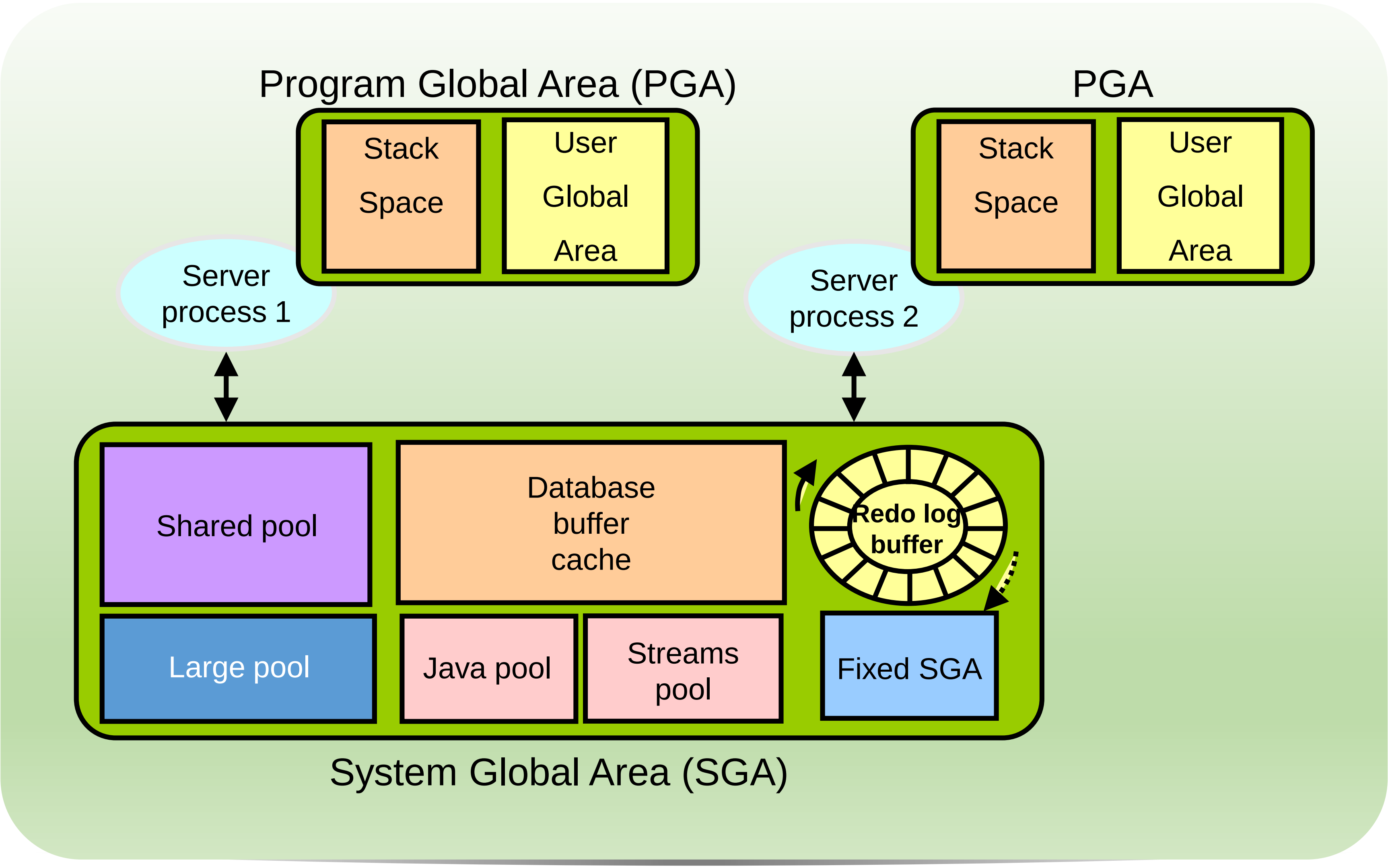
Performance Tuning Methodology

System Health and OS Statistics

Top Down Approach:
Design
Application
Database Instance

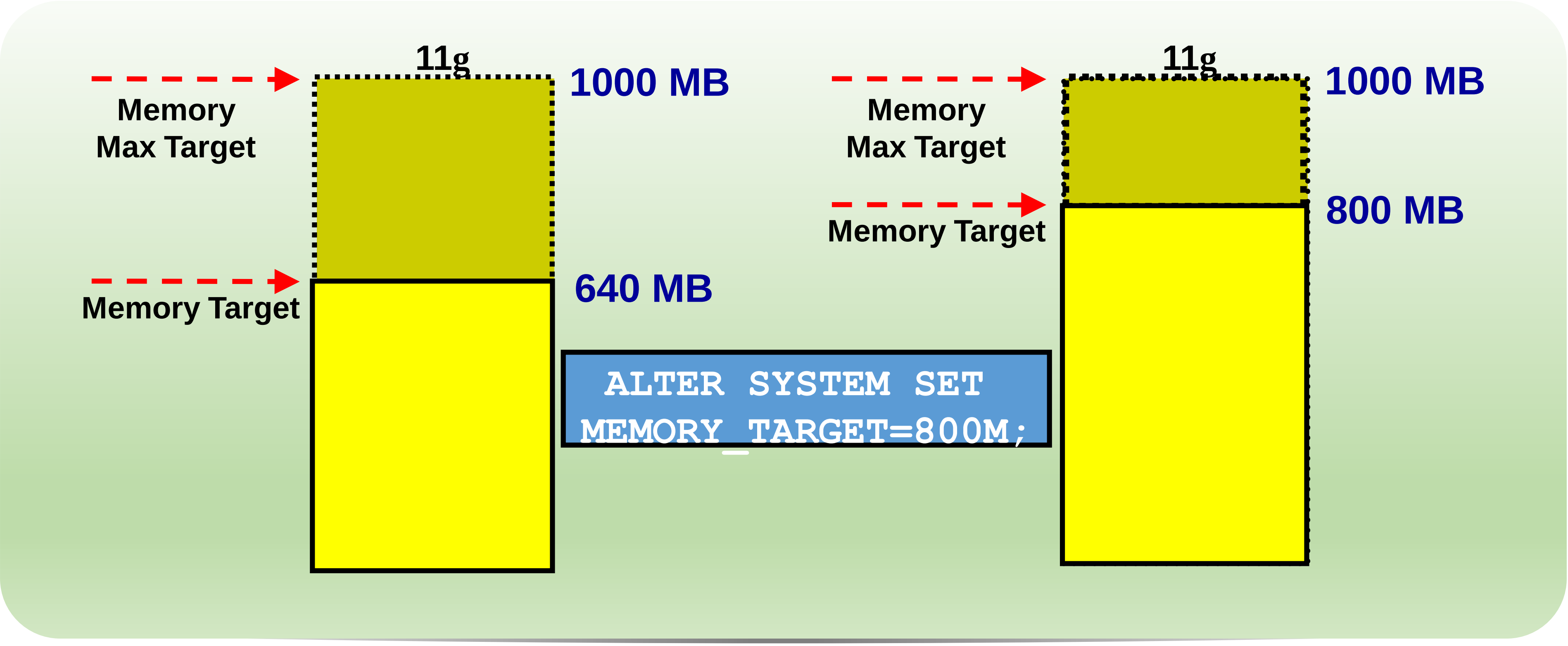
Tune Areas with Greatest Benefit

Managing Memory Components



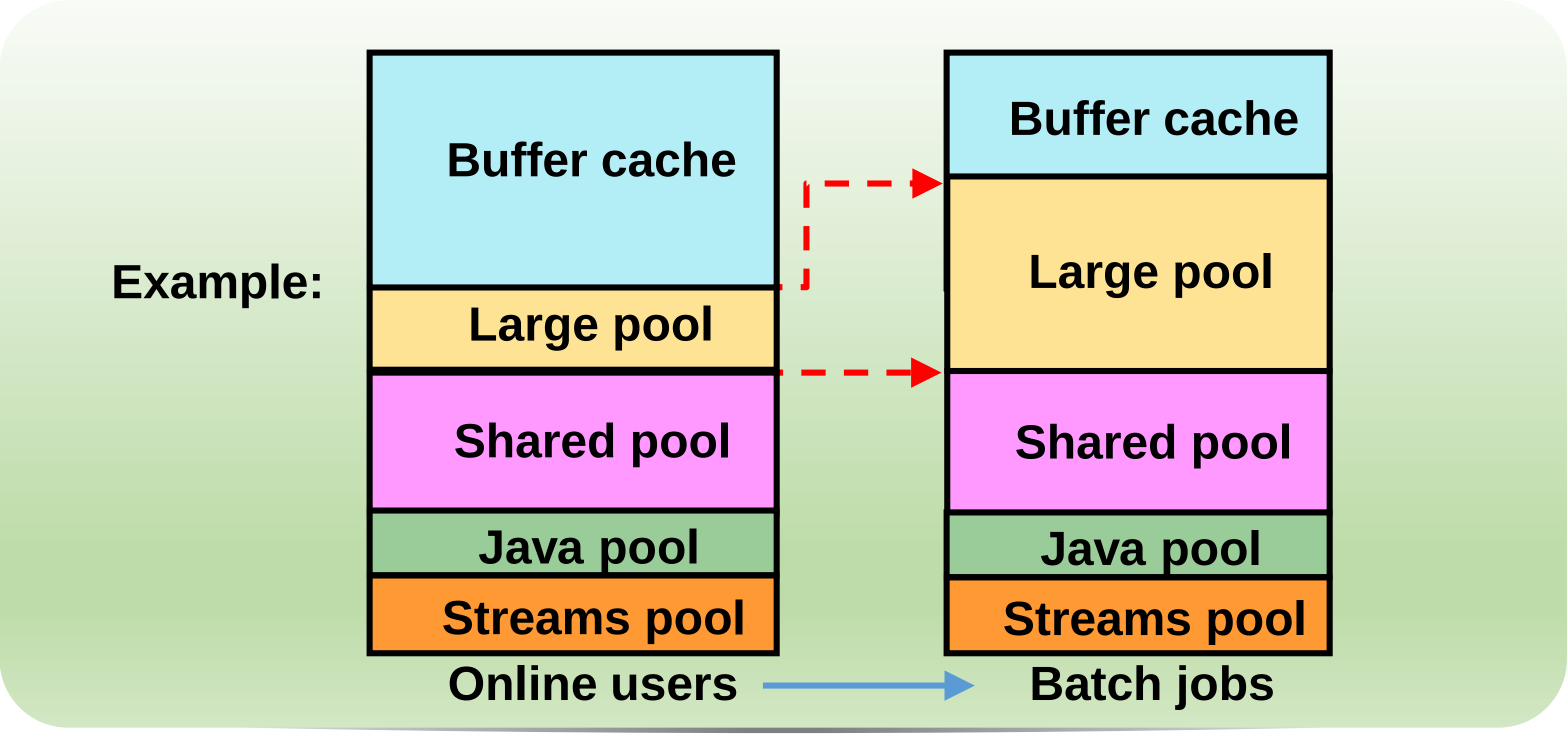
Automatic Memory Management

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

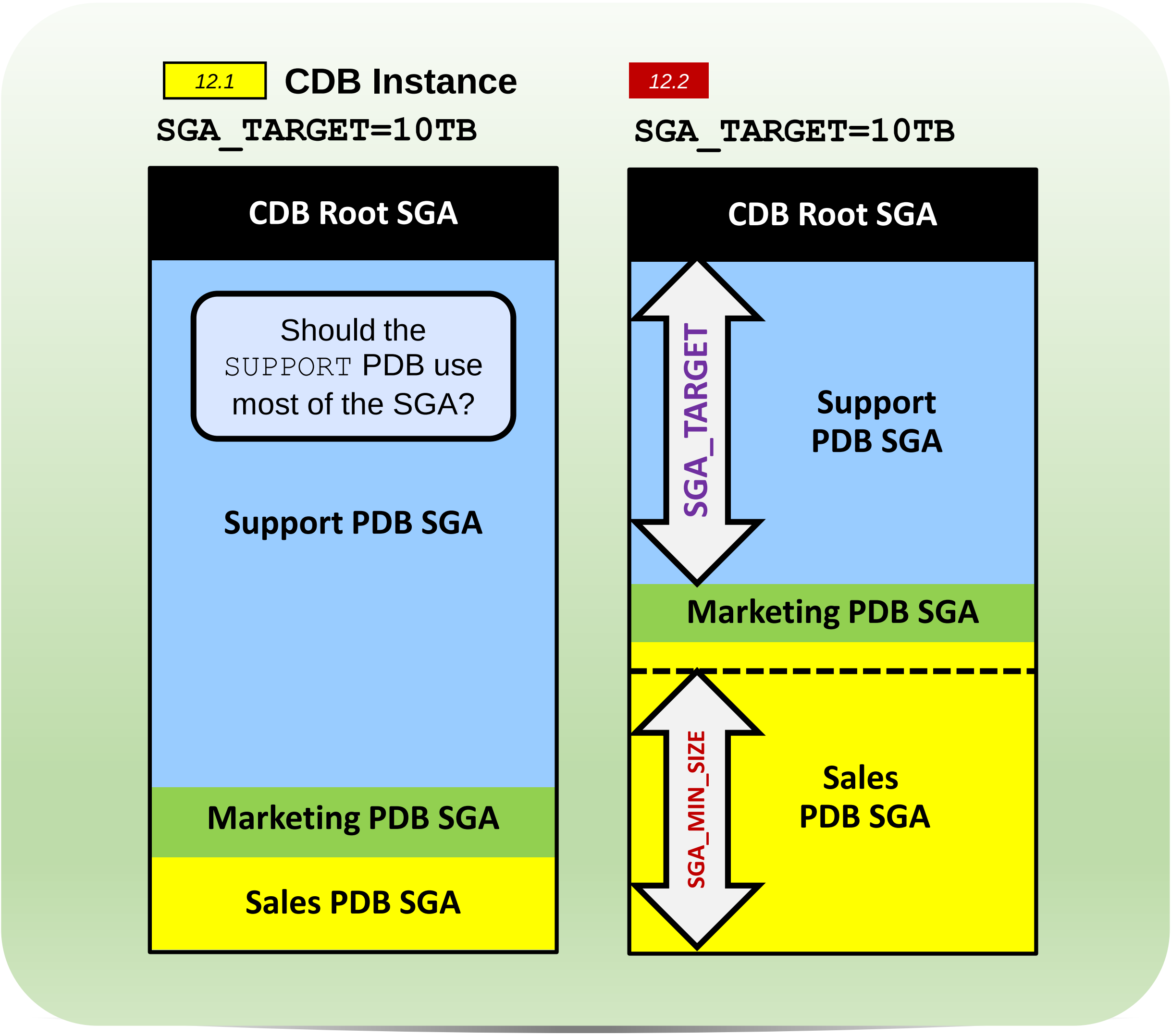


Automatic Shared Memory Management

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

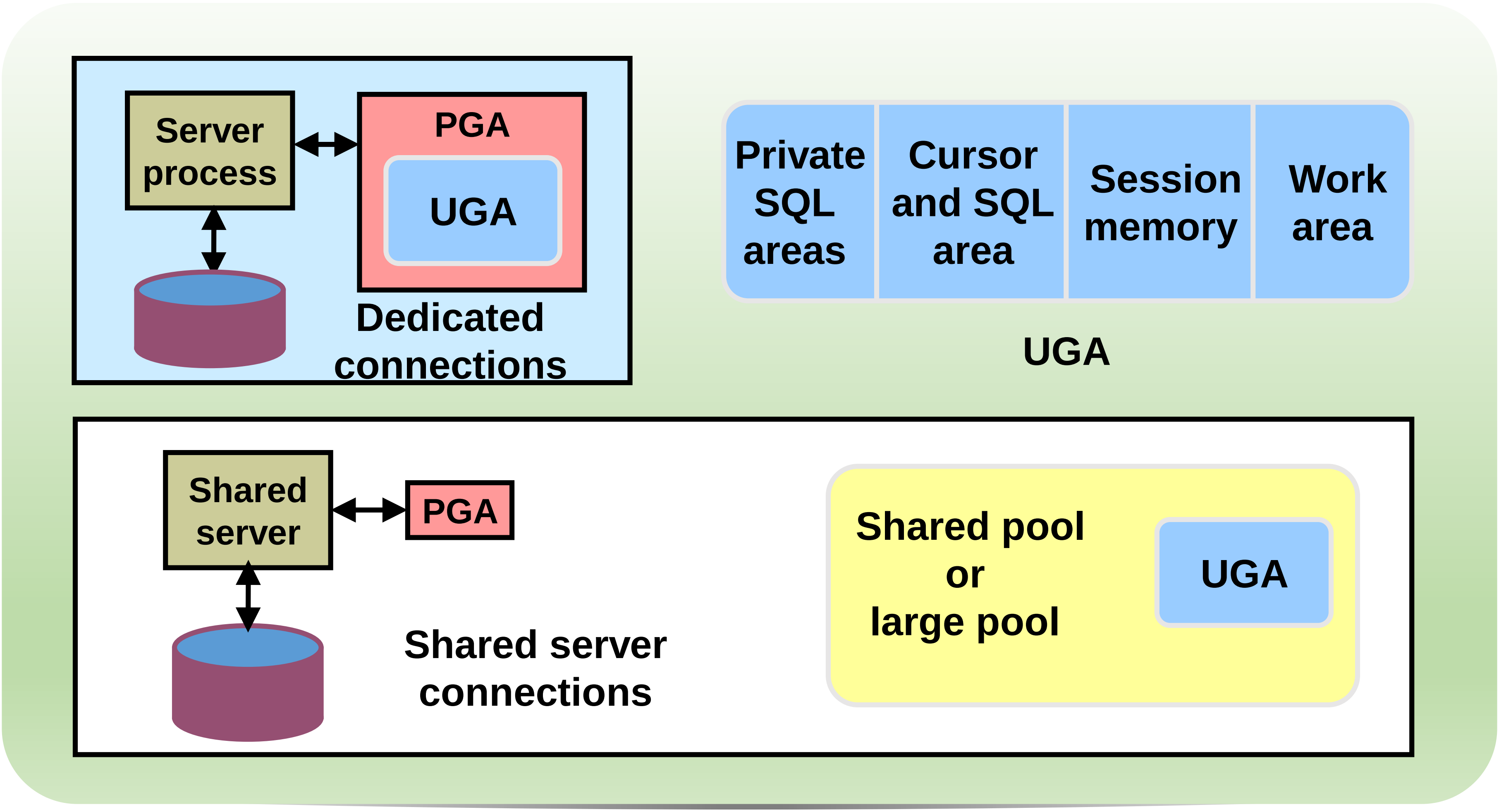


Managing the SGA for PDBs



- **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.

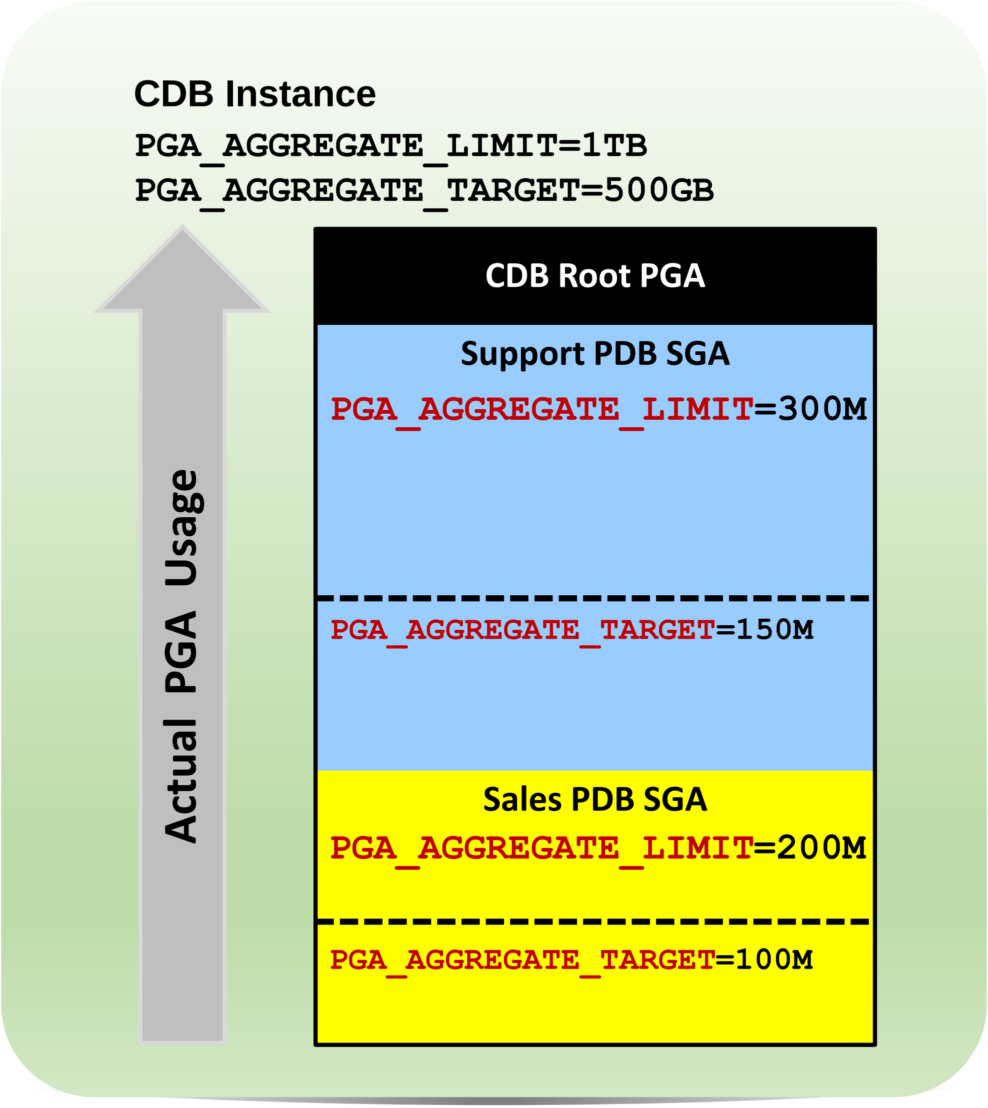
Managing the PGA for PDBs

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.



Summary

- In this lesson, you should have learned how to:
  - Describe the activities that you perform to manage database performance
  - Use performance views and tools to monitor database instance performance
  - Describe the Oracle performance tuning methodology
  - Describe statistics and metrics that are collected by the Oracle Database server
  - Configure and monitor memory components for optimal performance

Practice 20: Overview

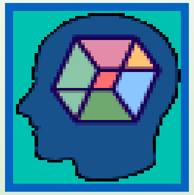
- 20-1: Managing Performance
- 20-2: Resolving Lock Conflicts

Tuning SQL

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



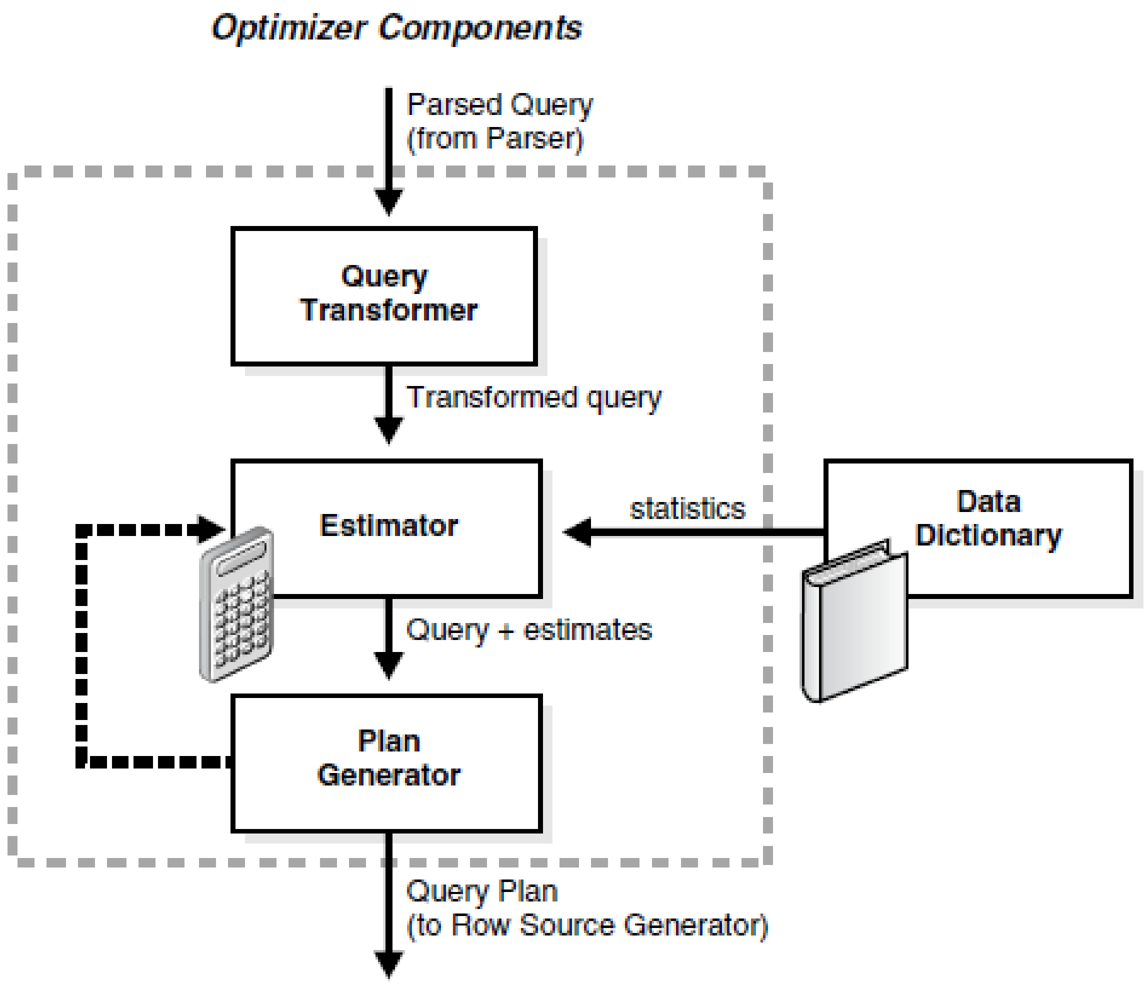
ADDM

SQL Tuning Advisor

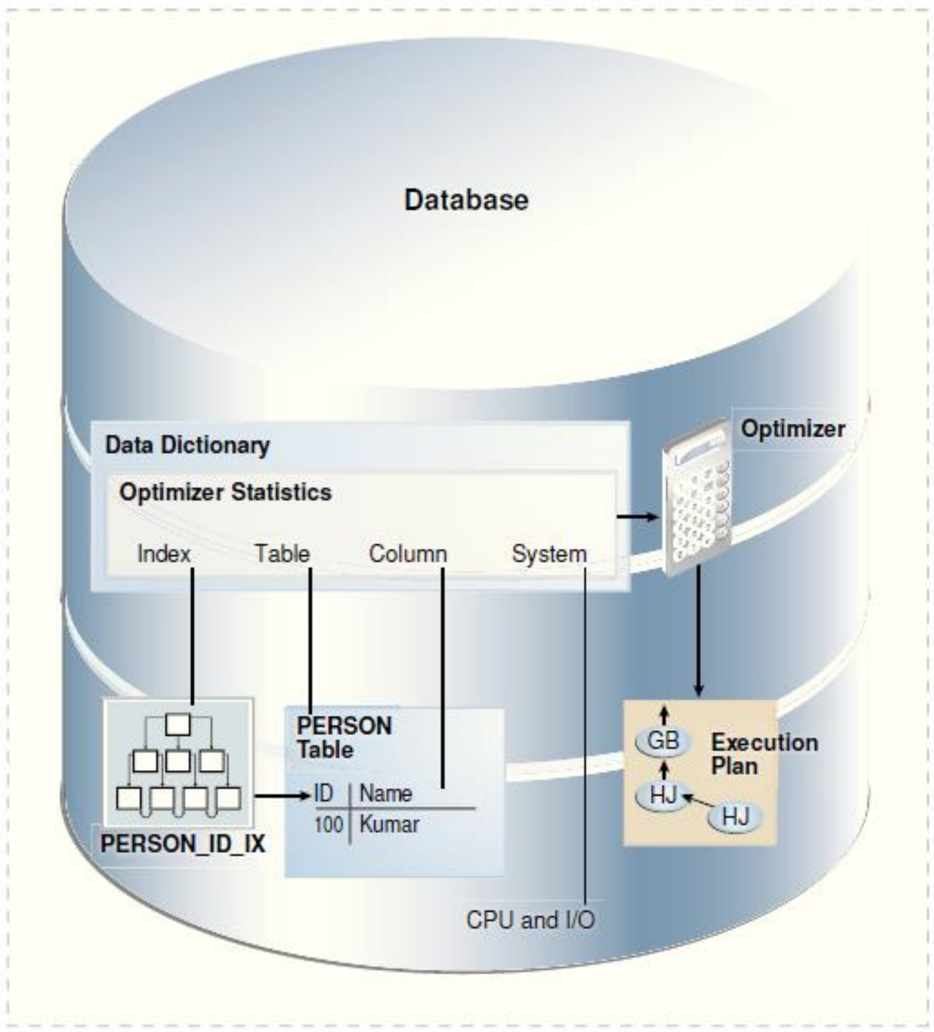
SQL Access Advisor

1. Identify poorly tuned SQL statements.
2. Tune the individual SQL statements.
3. Tune the application as a whole.

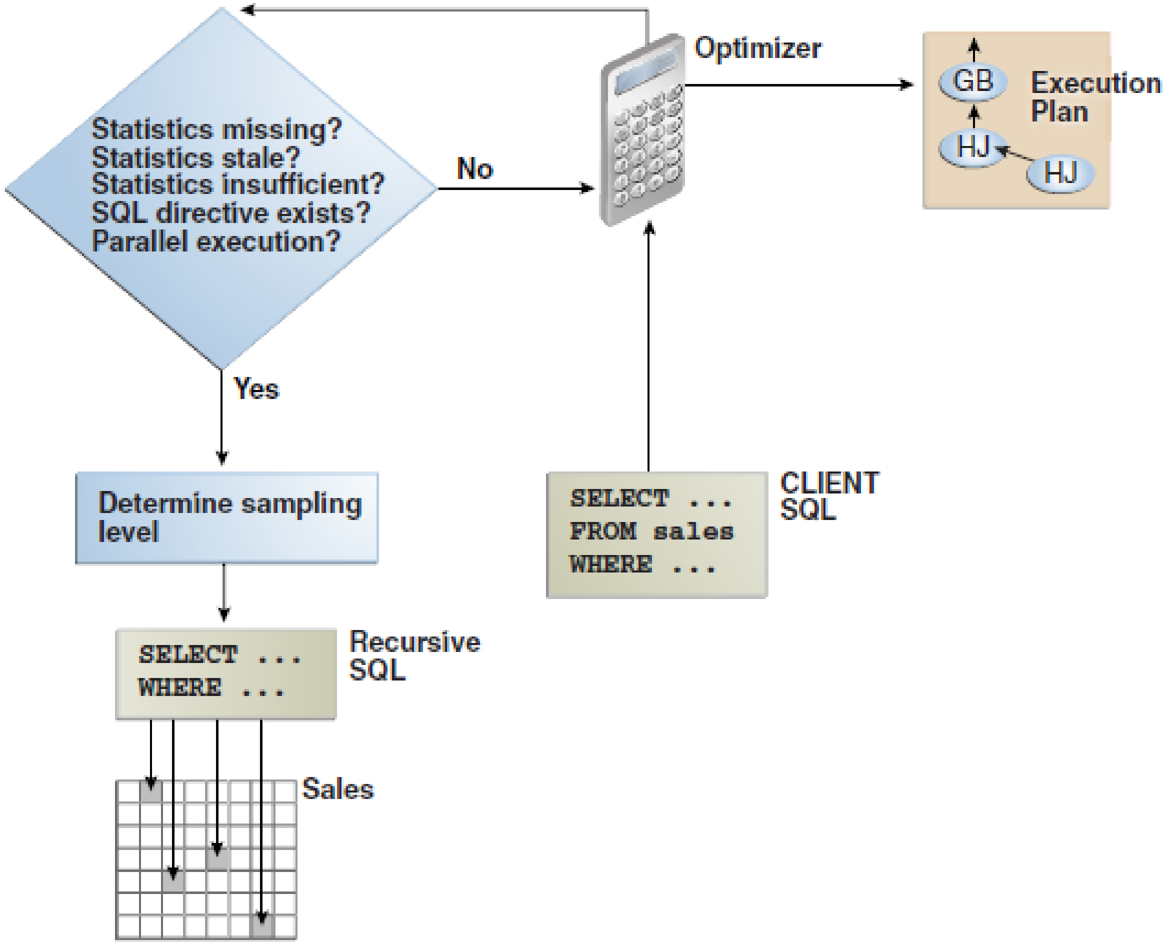
Oracle Optimizer



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

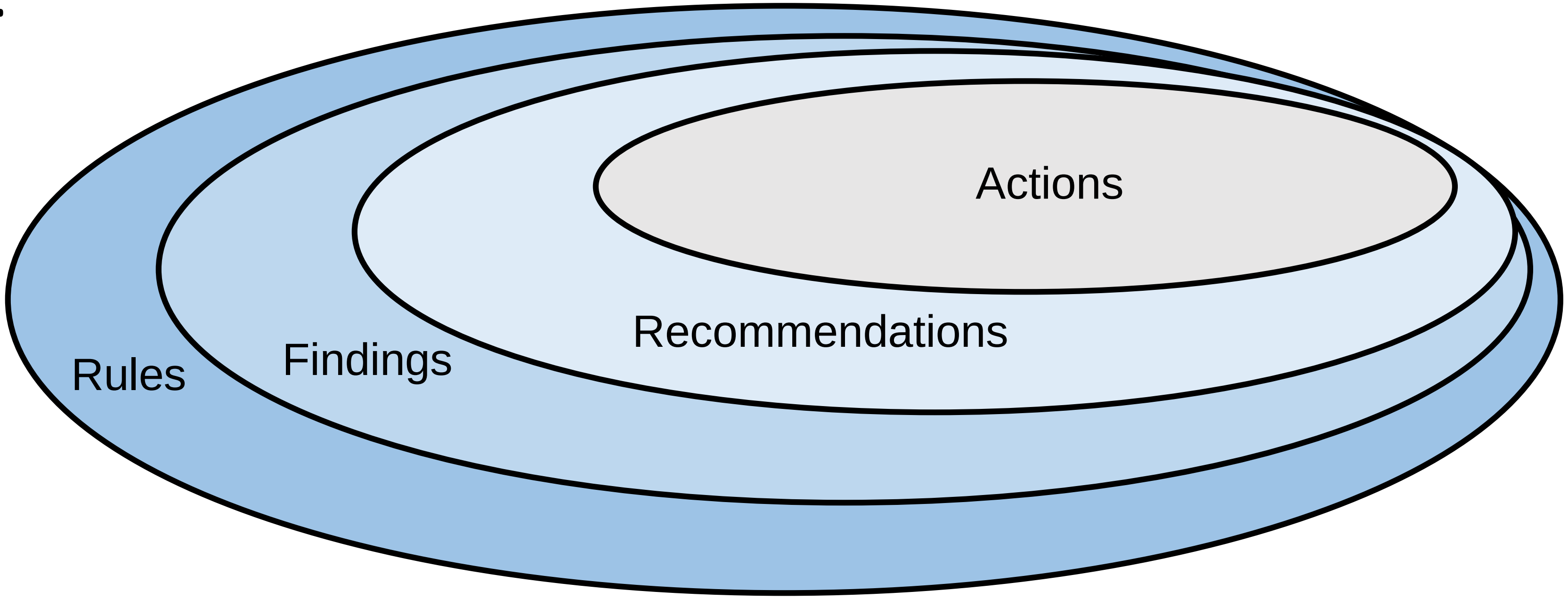
V$STATS_ADVISOR_RULES
NAME = *UseAutoJob*
RULE_TYPE = *SYSTEM*

V$STATS_ADVISOR_RULES
NAME = *UseGatherSchemaStats*
RULE_TYPE = *OPERATION*

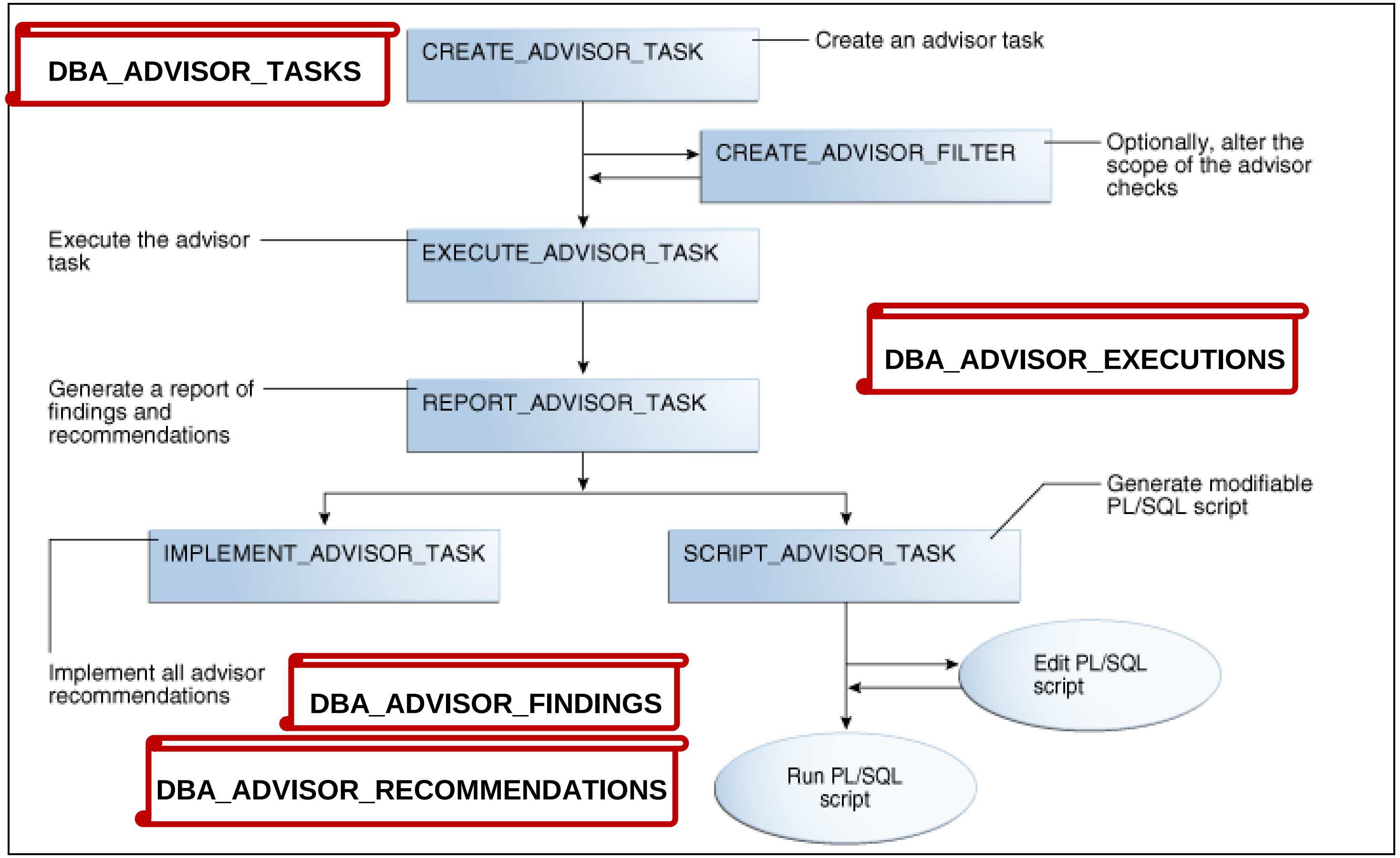
V$STATS_ADVISOR_RULES
NAME = *AvoidStaleStats*
RULE_TYPE = *OBJECT*

Optimizer Statistics Advisor Report

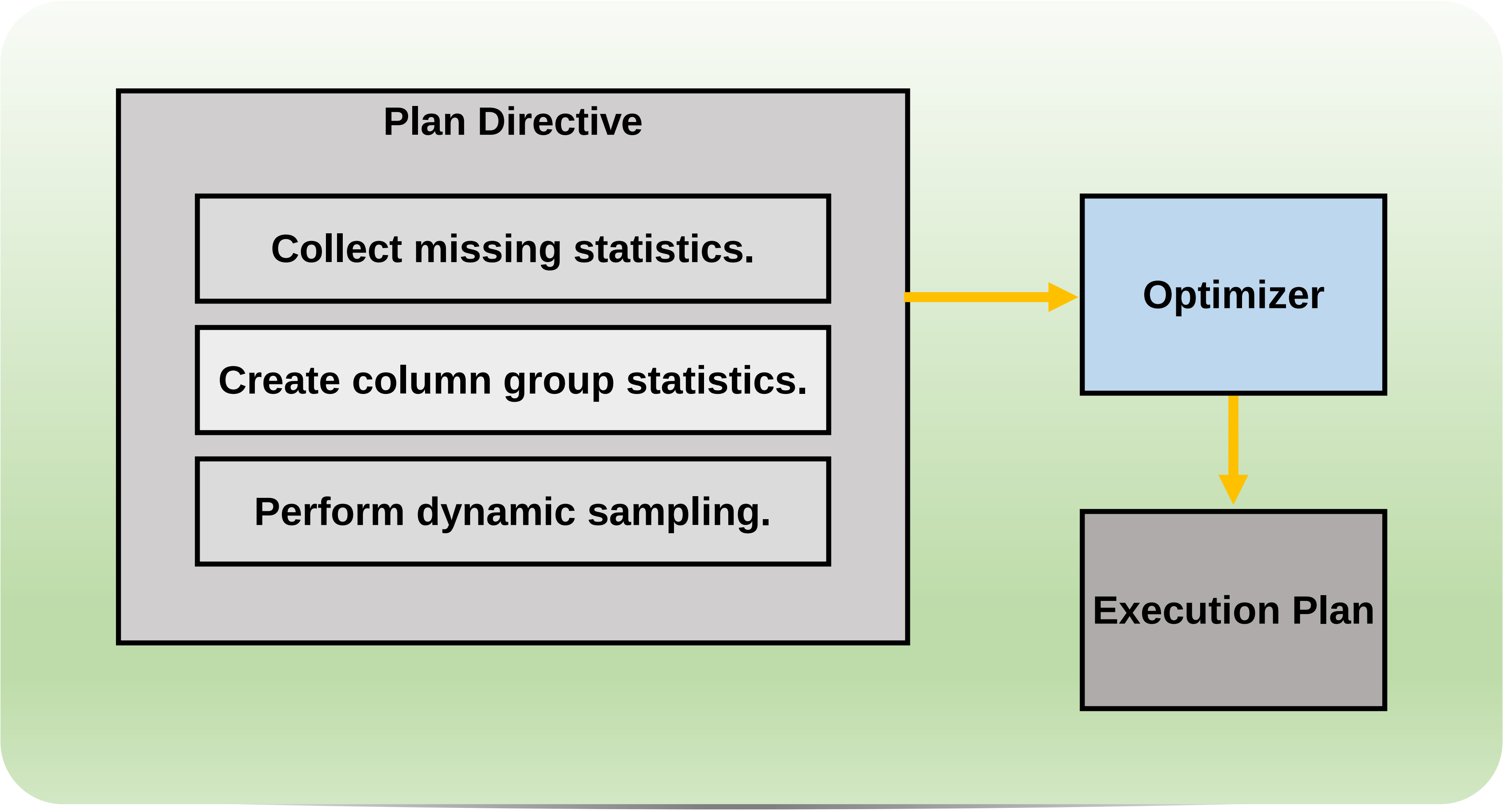
- Report sections:
  - Header
  - Summary
  - Errors
  - Findings



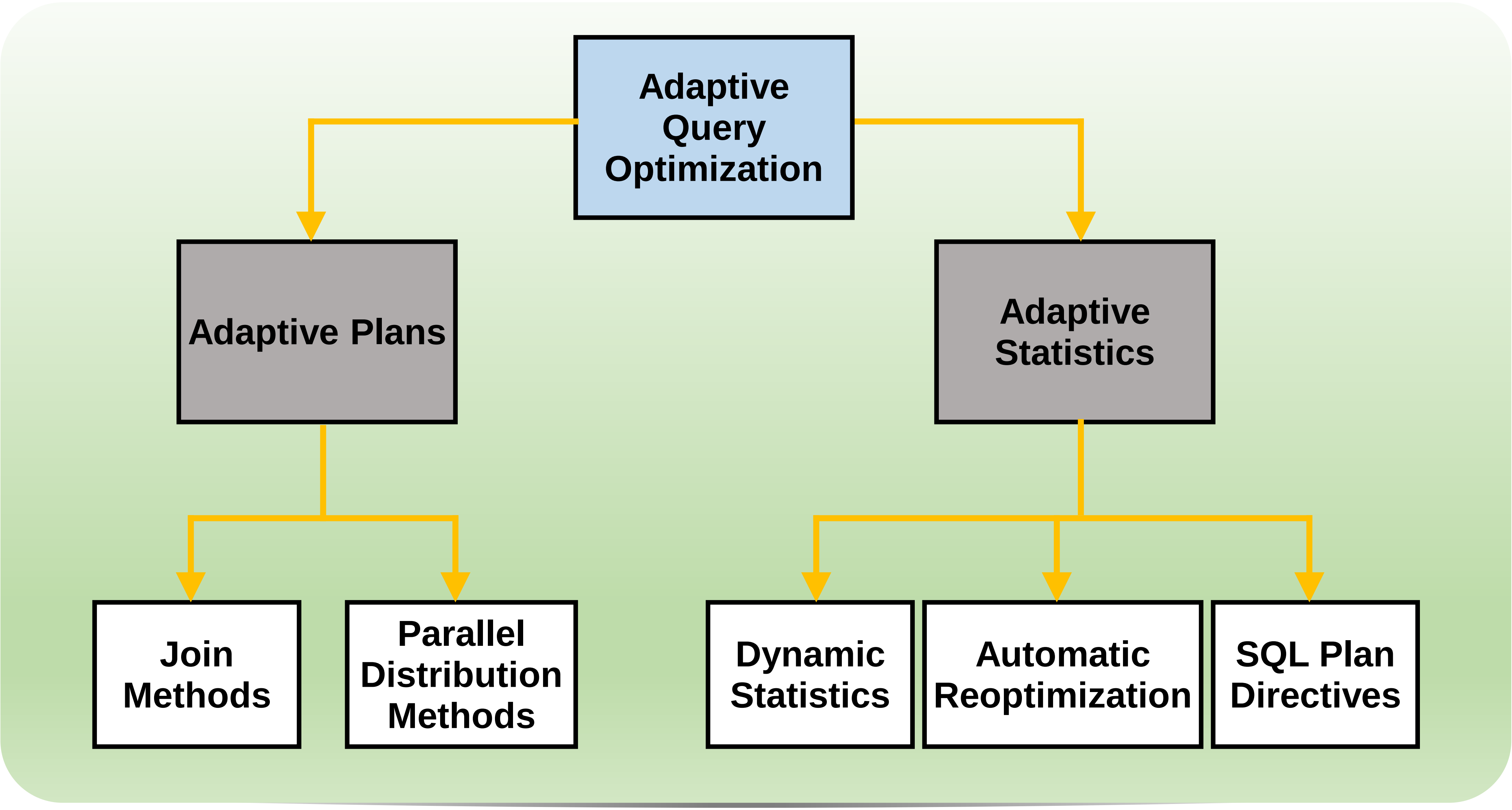
Executing Optimizer Statistics Advisor Tasks



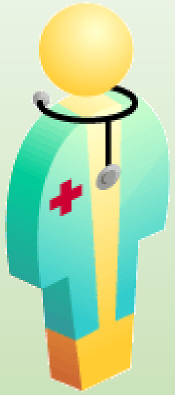
SQL Plan Directives



Adaptive Execution Plans



SQL Tuning Advisor: Overview

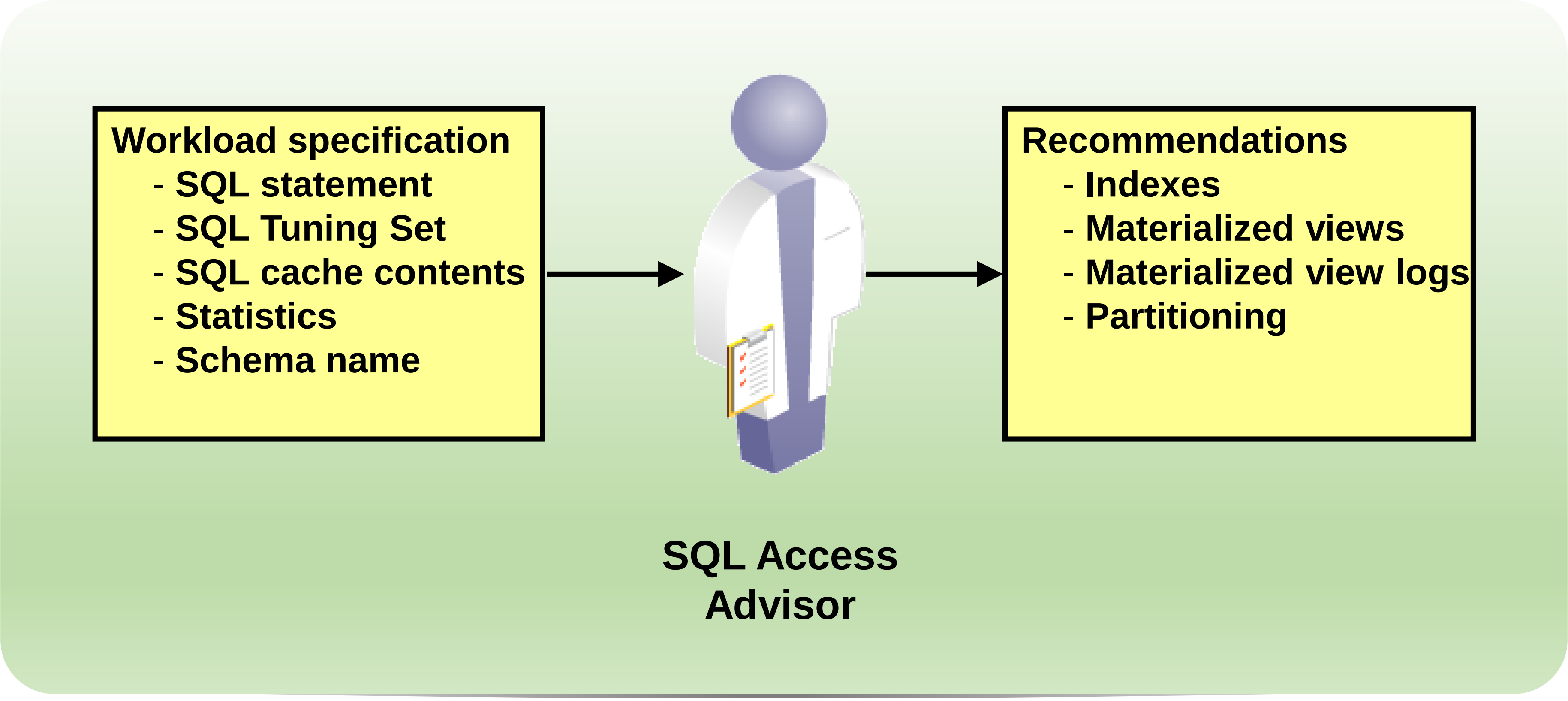


SQL Tuning Advisor

Comprehensive SQL tuning

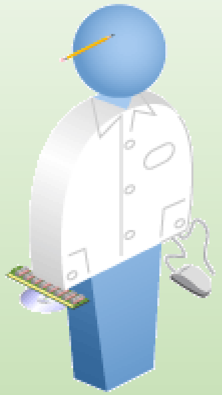
- Detect stale or missing statistics
- Tune SQL plan (SQL profile)
- Add missing index
- Restructure SQL

SQL Access Advisor: Overview



SQL Performance Analyzer: Overview

Predicts the impact of system changes



SQL Performance Analyzer

- Builds different versions of SQL workload performance
- Executes SQL serially
- Analyzes performance differences
- Offers fine-grained performance analysis on individual SQL

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 21: Overview

- 21-1: Using SQL Tuning Advisor
- 21-2: Using Optimizer Statistics Advisor