Oracle Database 11*g*: Administration Workshop I

Volume I • Student Guide

D50102GC20
Edition 2.0
September 2009
D62541

ORACLE

Authors

Deirdre Matishak

Mark Fuller

Technical Contributors and Reviewers

Maria Billings

Herbert Bradbury

Yanti Chang

Timothy Chien

Andy Fotunak

Gerlinde Frenzen

Steve Friedberg

Joel Goodman

Vimala Jacob

Dominique Jeunot

Pete Jones

Fukue Kawabe

Donna Keesling

Sean Kim

Achiel Langers

Gwen Lazenby

Essi Parast

Randy Richeson

Joe Roch

Hilda Simon

Ira Singer

Jim Spiller

Supithran Thananayagam

Branislav Valny

Manju Varrier

Editors

Raj Kumar

Daniel Milne

Graphic Designer

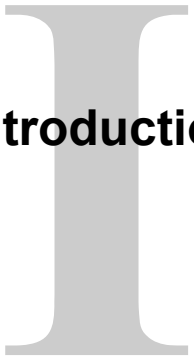
Rajiv Chandrabhanu

Publishers

Jobi Varghese

Veena Narasimhan

Contents



Introduction

Course Objectives

After completing this course, you should be able to:

- Install, create, and administer Oracle Database 11*g* Release 2
- Configure the database for an application
- Employ basic monitoring procedures
- Implement a backup and recovery strategy
- Move data between databases and files

ORACLE

Course Objectives

In this course, you install the Oracle Database 11*g* Release 2 Enterprise Edition software, create a new database, and learn how to administer the database.

You also configure the database to support an application and perform tasks such as creating users, defining storage structures, and setting up security. This course uses a fictional application. However, you perform all the core tasks that are necessary for a real application.

Database administration does not end after you configure your database. You also learn how to protect it by designing a backup and recovery strategy and how to monitor it to ensure that it operates smoothly.

Suggested Schedule

Day	Lessons	Day	Lessons
1	1. Exploring the Oracle Database Architecture 2. Preparing the Database Environment 3. Creating an Oracle Database 4. Managing Database Instances	3	9. Managing Data Concurrency 10. Managing Undo Data 11. Implementing Oracle Database Auditing
2	5. Managing ASM Instances 6. Configuring the Oracle Network Environment 7. Managing Database Storage Structures 8. Administering User Security	4	12. Database Maintenance 13. Performance Management 14. Backup and Recovery Concepts
		5	15. Performing Database Backups 16. Performing Database Recovery 17. Moving Data 18. Working with Support

ORACLE

Oracle Products and Services

- Oracle Database
- Oracle WebLogic Application Server
- Oracle Applications
- Oracle Collaboration Suite
- Oracle Developer Suite
- Oracle Services

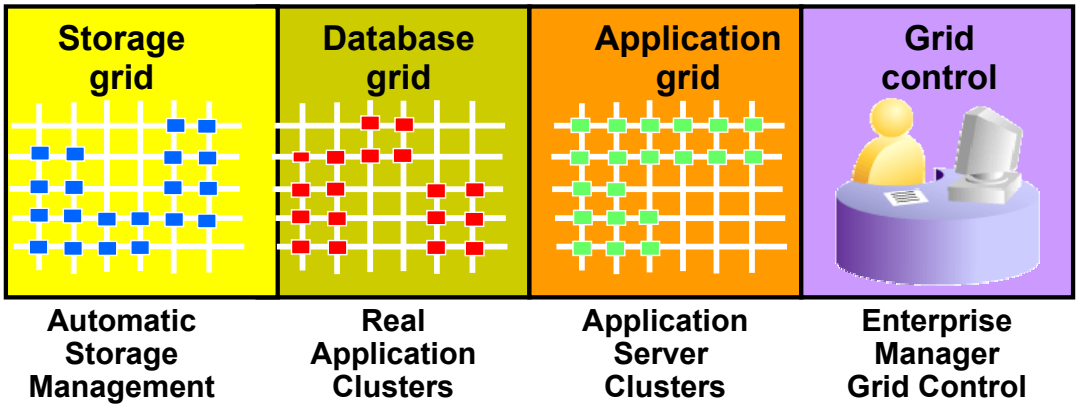
ORACLE

Oracle Products and Services

- **Oracle Database:** The Oracle database is the first database that is designed for enterprise grid computing (the most flexible and cost-effective way to manage information and applications).
- **Oracle WebLogic Application Server:** Oracle's Java 2 Platform, Enterprise Edition, certified server integrates everything that is needed to develop and deploy Web-based applications. The application server deploys e-business portals, Web services, and transactional applications such as PL/SQL, Oracle Forms, and Java EE based applications.
- **Oracle Applications:** Oracle E-Business Suite is a complete set of business applications for managing and automating processes across your organization.
- **Oracle Collaboration Suite:** Oracle Collaboration Suite is a single integrated system for all your organization's communications data: voice, email, fax, wireless, calendar information, and files.
- **Oracle Developer Suite:** Oracle Developer Suite is a complete, integrated environment that combines application development and business intelligence tools.
- **Oracle Services:** Services such as Oracle Consulting and Oracle University provide you with the necessary expertise for your Oracle projects. For links to a variety of resources, see the appendix titled "Next Steps: Continuing Your Education."

Oracle Database 11*g*: "*g*" Stands for Grid

- Open Grid Forum (OGF)
- Oracle's grid infrastructure:
  - Low cost
  - High quality of service
  - Easy to manage



ORACLE

Oracle Database 11*g*: "*g*" Stands for Grid

Open Grid Forum (OGF) is a standards body that develops standards for grid computing. It comprises a set of committees and working groups that focus on various aspects of grid computing. The committees and working groups are composed of participants from academia, the research community, and (increasingly) commercial companies. You can see the website of OGF at http://www.ogf.org.

Oracle has created the grid computing infrastructure software that balances all types of workloads across servers and enables all those servers to be managed as one complete system. Grid computing can achieve the same very high level of reliability as mainframe computing because all components are clustered. But unlike mainframes and large UNIX symmetric multiprocessing (SMP) servers, a grid can be built with open system technologies, such as Intel processors and the Linux operating system, at a very low cost.

Oracle's grid computing technology includes:

- Automatic Storage Management (ASM)
- Real Application Clusters (RAC)
- Application Server Clusters
- Enterprise Manager Grid Control

Oracle Database 11*g*: "*g*" Stands for Grid (continued)

Automatic Storage Management spreads database data across all disks, creates and maintains a storage grid, and provides the highest input/output (I/O) throughput with minimal management costs. As disks are added or dropped, ASM redistributes the data automatically. (There is no need for a logical volume manager to manage the file system.) Data availability increases with optional mirroring, and you can add or drop disks online. See the lesson titled "Managing Database Storage Structures."

Oracle's **Real Application Clusters** runs and scales all application workloads on a cluster of servers and offers the following features:

- **Integrated clusterware:** This includes functionality for cluster connectivity, messaging and locking, cluster control, and recovery. It is available on all platforms that are supported by Oracle Database 10*g* or later.
- **Automatic workload management:** Rules can be defined to automatically allocate processing resources to each service both during normal operations and in response to failures. These rules can be dynamically modified to meet the changing business needs. This dynamic resource allocation within a database grid is unique to Oracle RAC.
- **Automatic event notification to the mid-tier:** When a cluster configuration changes, the mid-tier can immediately adapt to instance failover or availability of a new instance. This enables end users to continue working in the event of instance failover without the delays typically caused by network timeouts. In the event of new instance availability, the mid-tier can immediately start load balancing connections to that instance. Oracle Database 10*g* or later Java Database Connectivity (JDBC) drivers have the "fast connection failover" functionality that can be automatically enabled to handle these events.

Oracle WebLogic Application Grid works with any application server - including Oracle WebLogic Server, IBM WebSphere Application Server, and JBoss Application Server - or in a pure grid environment without an application server. Oracle WebLogic Application Grid provides extreme and predictable application scalability and performance. With capacity on demand, Oracle WebLogic Application Grid can linearly scale out middleware infrastructure from a few to thousands of servers. Through its in-memory data grid solution, it provides fast access to frequently used data. Leveraging this grid capability, computation can be done in parallel, further improving application performance.

Enterprise Manager Grid Control manages gridwide operations that include managing the entire stack of software, provisioning users, cloning databases, and managing patches. It can monitor the performance of all applications from the point of view of your end users. Grid Control views the performance and availability of the grid infrastructure as a unified whole rather than as isolated storage units, databases, and application servers. You can group hardware nodes, databases, and application servers into single logical entities and manage a group of targets as one unit.

Note: In this course, you use Enterprise Manager Database Console to manage one database at a time.

Grid Infrastructure for Single-Instance

Grid Infrastructure for Single-Instance is introduced with Oracle Database 11*g* Release 2.

- Is installed from the clusterware media, separate from Oracle database software
- Contains Oracle Automatic Storage Management (ASM)
- Contains Oracle Restart – a high availability solution for non-clustered databases
  - Can monitor and restart the following components:
    - Database Instances
    - Oracle Net Listener
    - Database Services
    - Automatic Storage Management (ASM) Instance
    - ASM Disk Groups
    - Oracle Notification Services (ONS/eONS) for Data Guard

ORACLE

Grid infrastructure for Single-Instance

Grid Infrastructure for Single-Instance is introduced with Oracle Database 11*g* Release 2. It is installed from the clusterware media, separate from Oracle database software and now includes Oracle Automatic Storage Management and a new feature called Oracle Restart.

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

- Database Instances
- Oracle Net Listener
- Database Services
- ASM Instance
- ASM Disk Groups
- Oracle Notification Services (ONS/eONS) for Data Guard

Oracle Restart ensures that the components are started in the proper order, in accordance with component dependencies. If a component must be shut down, it ensures that the dependent components are cleanly shut down first. Oracle Restart runs out of the Oracle Grid Infrastructure home, which you install separately from Oracle database homes.

1

Exploring the Oracle Database Architecture

Objectives

After completing this lesson, you should be able to:

- List the major architectural components of Oracle Database
- Explain the memory structures
- Describe the background processes
- Correlate the logical and physical storage structures
- Describe ASM storage components

ORACLE

Objectives

This lesson provides a detailed overview of the Oracle Database architecture. You learn about the physical and logical structures and about various components.

Oracle Database

The Oracle relational database management system (RDBMS) provides an open, comprehensive, integrated approach to information management

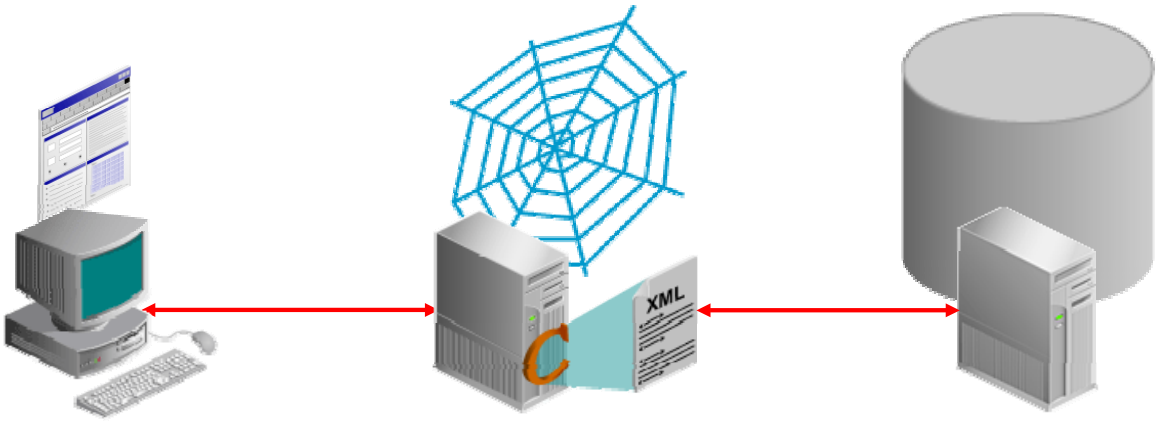
ORACLE

Oracle Database

A database is a collection of data treated as a unit. The purpose of a database is to store and retrieve related information.

The Oracle relational database management system (RDBMS) reliably manages a large amount of data in a multiuser environment so that many users can concurrently access the same data. This is accomplished while delivering high performance. At the same time, it prevents unauthorized access and provides efficient solutions for failure recovery.

Connecting to a Server



Multitier architecture shown

Connecting to a Server

A database user can connect to an Oracle server in one of three ways:

- The user logs on to the operating system running the Oracle instance and starts an application or tool that accesses the database on that system. The communication pathway is established using the interprocess communication mechanisms available on the host operating system.
- The user starts the application or tool on a local computer and connects over a network to the computer running the Oracle database. In this configuration (called *client/server*), network software is used to communicate between the user and the back-end server. The client/server architecture database system has two parts: a front end (client) and a back end (server) connected through a network. Network software is used to communicate between the user and the Oracle server.
  - The client is a database application that initiates a request for an operation to be performed on the database server. It requests, processes, and presents data managed by the server. The client workstation can be optimized for its job. For example, the client might not need large disk capacity, or it might benefit from graphic capabilities. Often, the client runs on a different computer than the database server. Many clients can simultaneously run against one server.

Connecting to a Server (continued)

- The server runs Oracle Database software and handles the functions required for concurrent, shared data access. The server receives and processes requests that originate from client applications. The computer that manages the server can be optimized for its duties. For example, the server computer can have large disk capacity and fast processors.

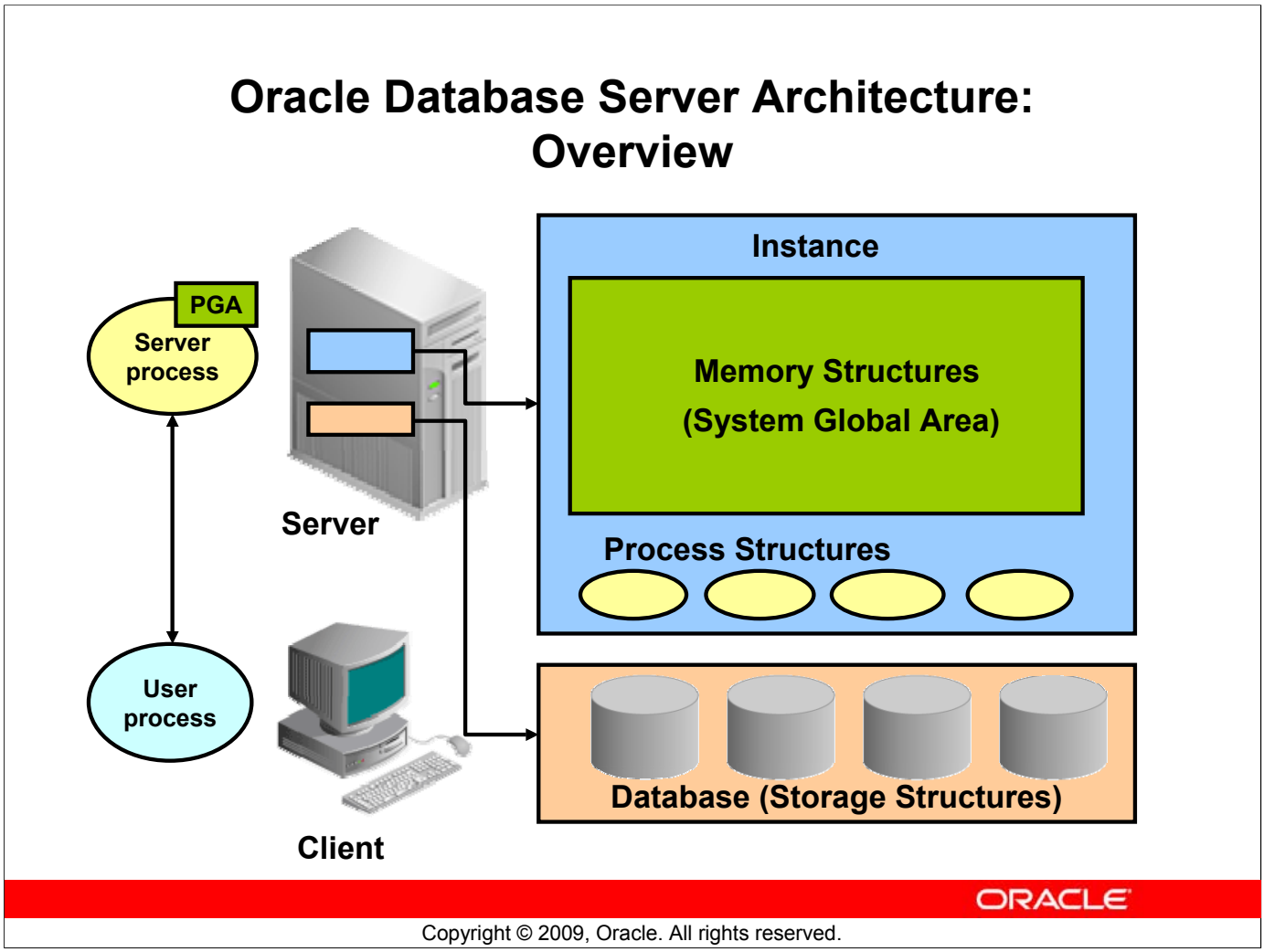
- The user accesses an application server through a tool (such as a Web browser) on the local computer (client). The application server then interacts with a back-end database server on behalf of the client.

A traditional multitier architecture has the following components:

- A client or initiator process that starts an operation
- One or more application servers that perform parts of the operation. An application server contains a large part of the application logic, provides access to the data for the client, and performs some query processing, thus removing some of the load from the database server. The application server can serve as an interface between clients and multiple database servers and can provide an additional level of security.
- An end server or database server that stores most of the data used in the operation

This architecture enables use of an application server to do the following:

- Validate the credentials of a client (such as a Web browser)
- Connect to an Oracle Database server
- Perform the requested operation on behalf of the client



Oracle Database Server Architecture

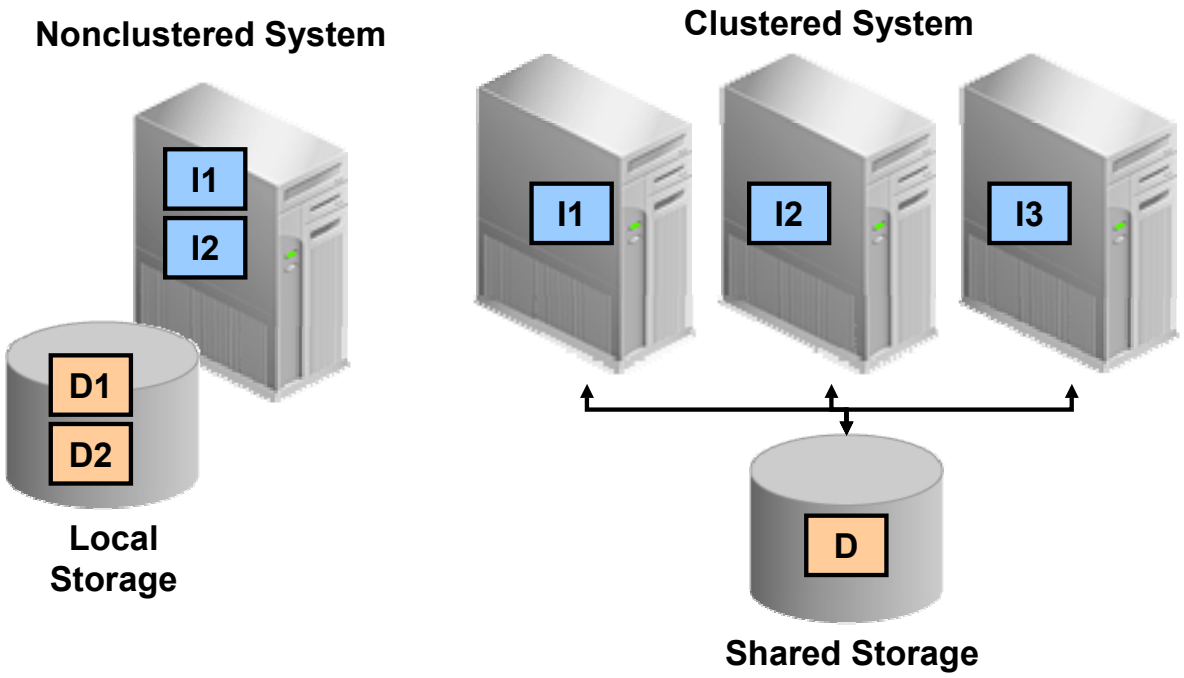
There are three major structures in Oracle Database server architecture: memory structures, process structures, and storage structures. A basic Oracle database system consists of an Oracle database and a database instance.

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

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

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

Instance: Database Configurations

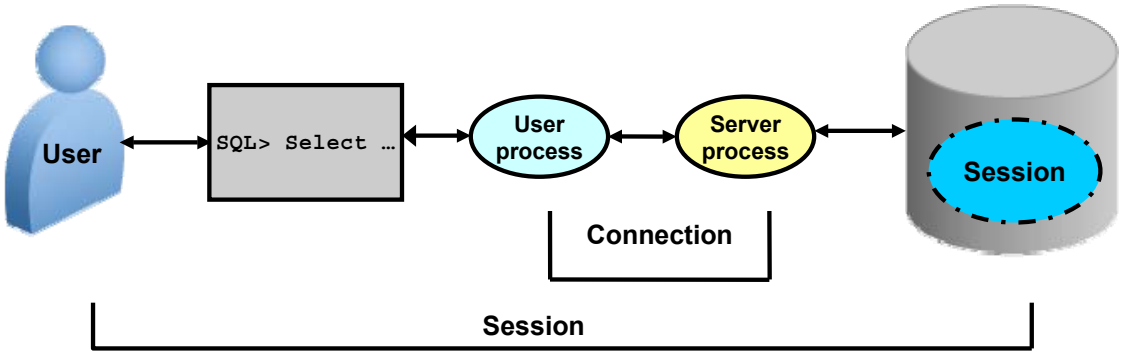


Instance: Database Configurations

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

Connecting to the Database Instance

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



ORACLE

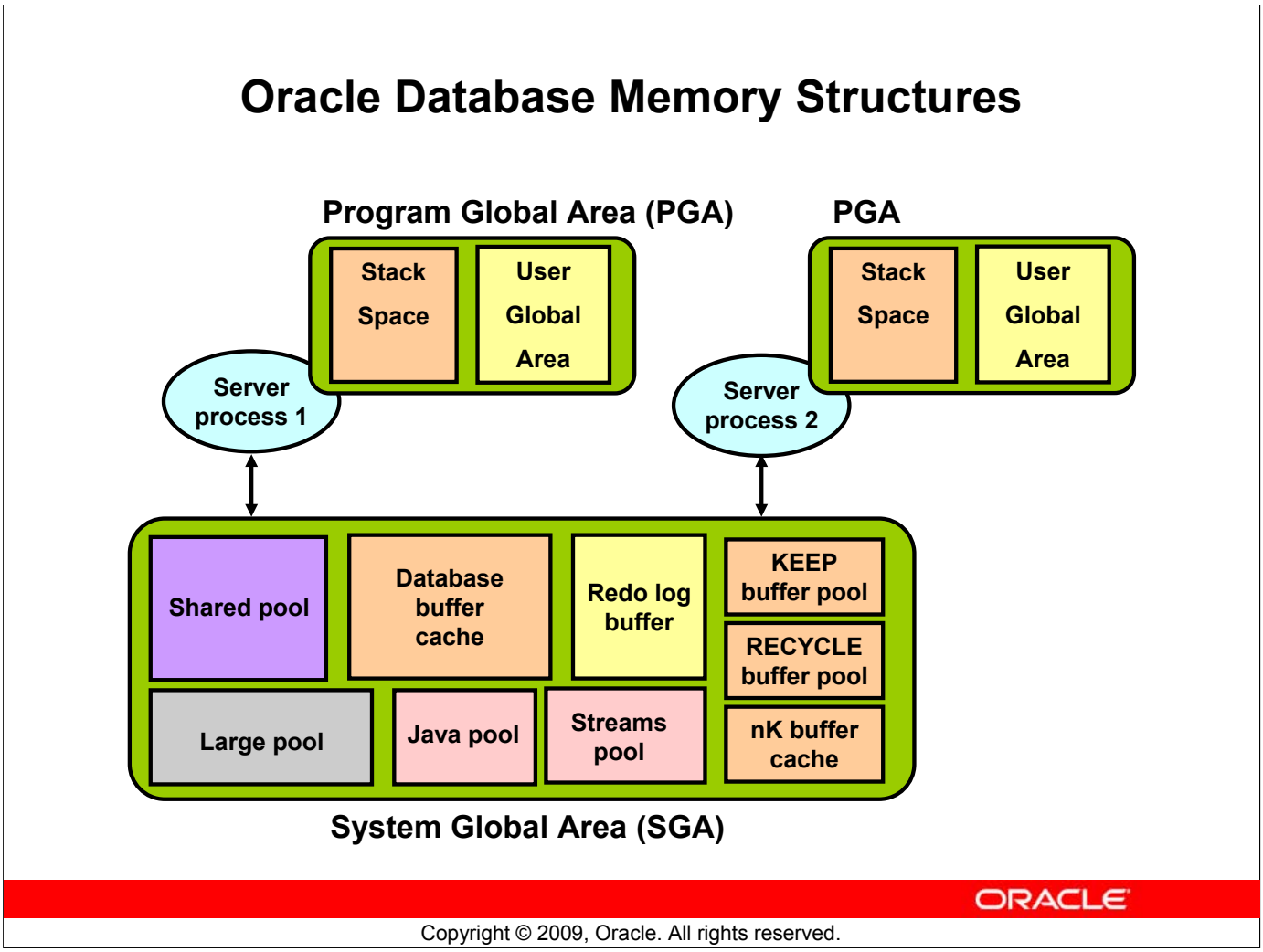
Connecting to the Database Instance

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

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

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

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



Oracle Database Memory Structures

Oracle Database creates and uses memory structures for various purposes. For example, memory stores program code being run, data that is shared among users, and private data areas for each connected user.

Two basic memory structures are associated with an instance:

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

Oracle Database Memory Structures (continued)

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

- **Shared pool:** Caches various constructs that can be shared among users
- **Database buffer cache:** Caches blocks of data retrieved from the database
- **KEEP buffer pool:** A specialized type of database buffer cache that is tuned to retain blocks of data in memory for long periods of time
- **RECYCLE buffer pool:** A specialized type of database buffer cache that is tuned to recycle or remove block from memory quickly
- **nK buffer cache:** One of several specialized database buffer caches designed to hold block sizes different than the default database block size
- **Redo log buffer:** Caches redo information (used for instance recovery) until it can be written to the physical redo log files stored on the disk
- **Large pool:** Optional area that provides large memory allocations for certain large processes, such as Oracle backup and recovery operations, and I/O server processes
- **Java pool:** Used for all session-specific Java code and data in the Java Virtual Machine (JVM)
- **Streams pool:** Used by Oracle Streams to store information required by capture and apply

When you start the instance by using Enterprise Manager or SQL*Plus, the amount of memory allocated for the SGA is displayed.

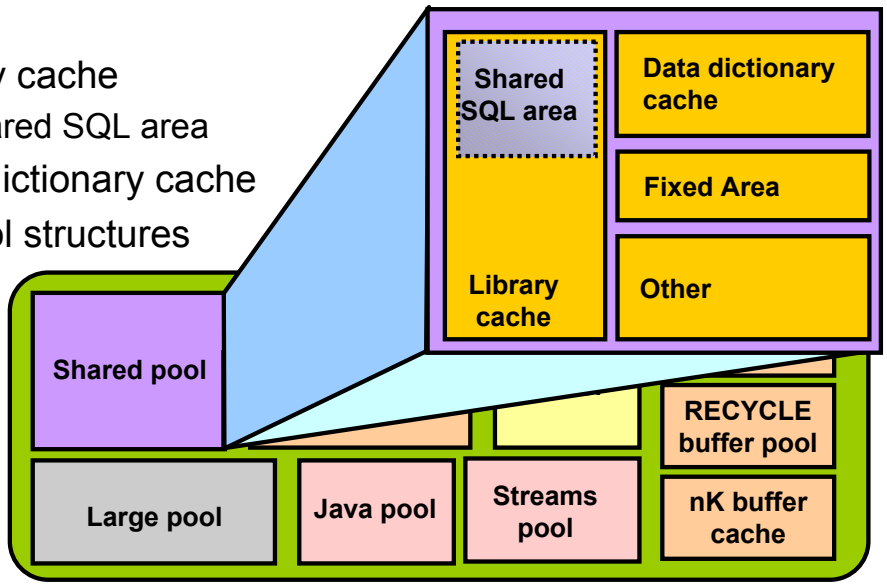
A Program Global Area (PGA) is a memory region that contains data and control information for each server process. An Oracle server process services a client's requests. Each server process has its own private PGA that is allocated when the server process is started. Access to the PGA is exclusive to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

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

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

Shared Pool

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



ORACLE

Shared Pool

The shared pool portion of the SGA contains the library cache, the data dictionary cache, the SQL query result cache, the PL/SQL function result cache, buffers for parallel execution messages, and control structures.

The *data dictionary* is a collection of database tables and views containing reference information about the database, its structures, and its users. Oracle Database accesses the data dictionary frequently during SQL statement parsing. This access is essential to the continuing operation of Oracle Database.

The data dictionary is accessed so often by Oracle Database that two special locations in memory are designated to hold dictionary data. One area is called the *data dictionary cache*, also known as the row cache because it holds data as rows instead of buffers (which hold entire blocks of data). The other area in memory to hold dictionary data is the *library cache*. All Oracle Database user processes share these two caches for access to data dictionary information.

Oracle Database represents each SQL statement that it runs with a shared SQL area (as well as a private SQL area kept in the PGA). Oracle Database recognizes when two users are executing the same SQL statement and reuses the shared SQL area for those users.

Shared Pool (continued)

A shared SQL area contains the parse tree and execution plan for a given SQL statement. Oracle Database saves memory by using one shared SQL area for SQL statements run multiple times, which often happens when many users run the same application.

When a new SQL statement is parsed, Oracle Database allocates memory from the shared pool to store in the shared SQL area. The size of this memory depends on the complexity of the statement.

Oracle Database processes PL/SQL program units (procedures, functions, packages, anonymous blocks, and database triggers) in much the same way it processes individual SQL statements. Oracle Database allocates a shared area to hold the parsed, compiled form of a program unit. Oracle Database allocates a private area to hold values specific to the session that runs the program unit, including local, global, and package variables (also known as package instantiation) and buffers for executing SQL. If more than one user runs the same program unit, then a single, shared area is used by all users, while all users maintain separate copies of their own private SQL areas, holding values specific to their own sessions.

Individual SQL statements contained in a PL/SQL program unit are processed just like other SQL statements. Despite their origins in a PL/SQL program unit, these SQL statements use a shared area to hold their parsed representations and a private area for each session that runs the statement.

The SQL query result cache and PL/SQL function result cache are new to Oracle Database 11*g*. They share the same infrastructure, appear in the same dynamic performance (`V$`) views, and are administered using the same supplied package.

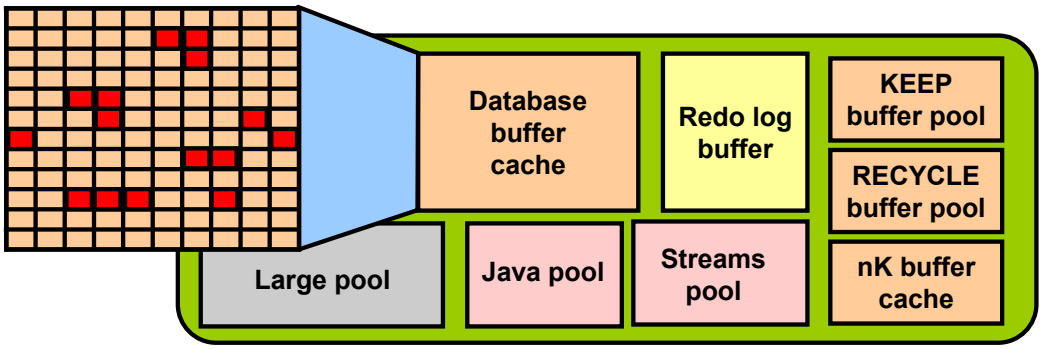
Results of queries and query fragments can be cached in memory in the SQL query result cache. The database can then use cached results to answer future executions of these queries and query fragments. Because retrieving results from the SQL query result cache is faster than rerunning a query, frequently run queries experience a significant performance improvement when their results are cached.

A PL/SQL function is sometimes used to return the result of a computation whose inputs are one or several parameterized queries issued by the function. In some cases, these queries access data that changes very infrequently compared to the frequency of calling the function. You can include syntax in the source text of a PL/SQL function to request that its results be cached in the PL/SQL function result cache and (to ensure correctness) that the cache be purged when tables in a list of tables experience DML.

The fixed area of the shared pool represents startup overhead for the SGA. It is very small in comparison to a typically sized shared pool or SGA.

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



ORACLE

Database Buffer Cache

The database buffer cache is the portion of the SGA that holds block images read from the data files or constructed dynamically to satisfy the read consistency model. All users who are concurrently connected to the instance share access to the database buffer cache.

The first time an Oracle Database user process requires a particular piece of data, it searches for the data in the database buffer cache. If the process finds the data already in the cache (a cache hit), it can read the data directly from memory. If the process cannot find the data in the cache (a cache miss), it must copy the data block from a data file on disk into a buffer in the cache before accessing the data. Accessing data through a cache hit is faster than data access through a cache miss.

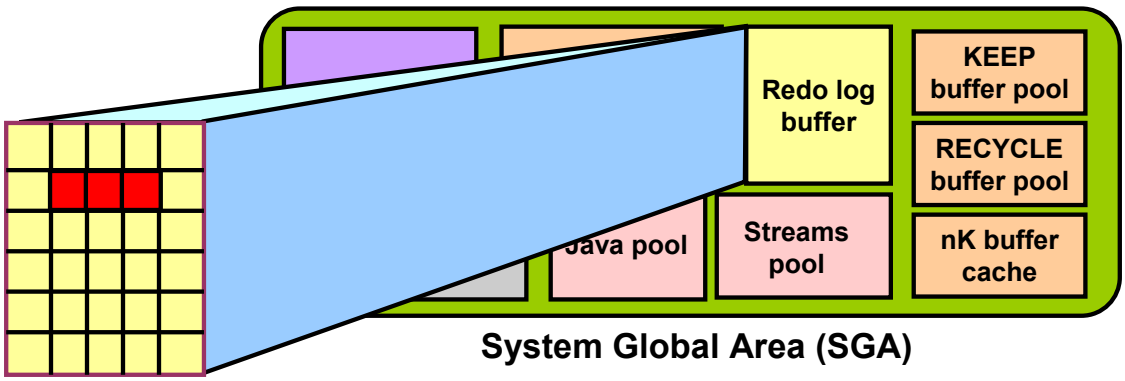
The buffers in the cache are managed by a complex algorithm that uses a combination of least recently used (LRU) lists and touch count. The LRU helps to ensure that the most recently used blocks tend to stay in memory to minimize disk access.

The KEEP buffer pool and the RECYCLE buffer pool are used for specialized buffer pool tuning. The KEEP buffer pool is designed to retain buffers in memory longer than the LRU would normally retain them. The RECYCLE buffer pool is designed to flush buffers from memory faster than the LRU would normally do so.

Additional buffer caches can be configured to hold blocks of a size that is different than the default block size.

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



Redo Log Buffer

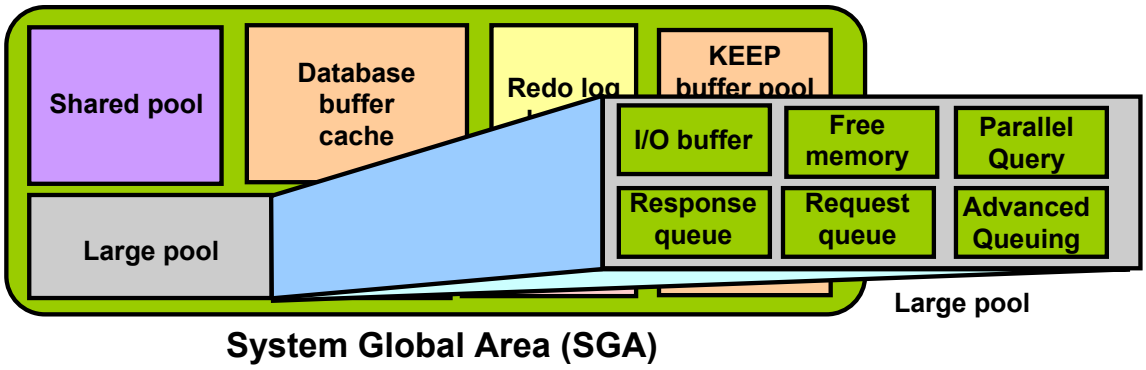
The redo log buffer is a circular buffer in the SGA that holds information about changes made to the database. This information is stored in redo entries. Redo entries contain the information necessary to reconstruct (or redo) changes that are made to the database by DML, DDL, or internal operations. Redo entries are used for database recovery if necessary.

As the server process makes changes to the buffer cache, redo entries are generated and written to the redo log buffer in the SGA. The redo entries take up continuous, sequential space in the buffer. The log writer background process writes the redo log buffer to the active redo log file (or group of files) on disk.

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



Large Pool

The database administrator can configure an optional memory area called the *large pool* to provide large memory allocations for:

- Session memory for the shared server and the Oracle XA interface (used where transactions interact with multiple databases)
- I/O server processes
- Oracle Database backup and restore operations
- Parallel Query operations
- Advanced Queuing memory table storage

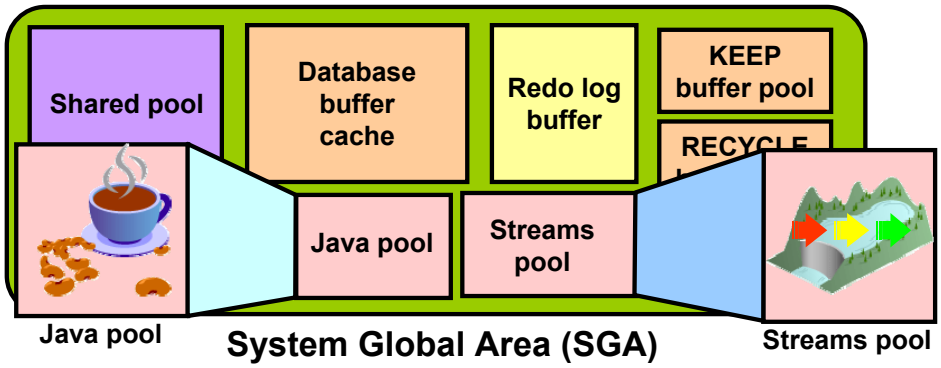
By allocating session memory from the large pool for shared server, Oracle XA, or parallel query buffers, Oracle Database can use the shared pool primarily for caching shared SQL and avoid the performance overhead that is caused by shrinking the shared SQL cache.

In addition, the memory for Oracle Database backup and restore operations, for I/O server processes, and for parallel buffers is allocated in buffers of a few hundred kilobytes. The large pool is better able to satisfy such large memory requests than the shared pool.

The large pool is not managed by a least recently used (LRU) list.

Java Pool and Streams Pool

- Java pool memory is used to store all session-specific Java code and data in the JVM.
- Streams pool memory is used exclusively by Oracle Streams to:
  - Store buffered queue messages
  - Provide memory for Oracle Streams processes



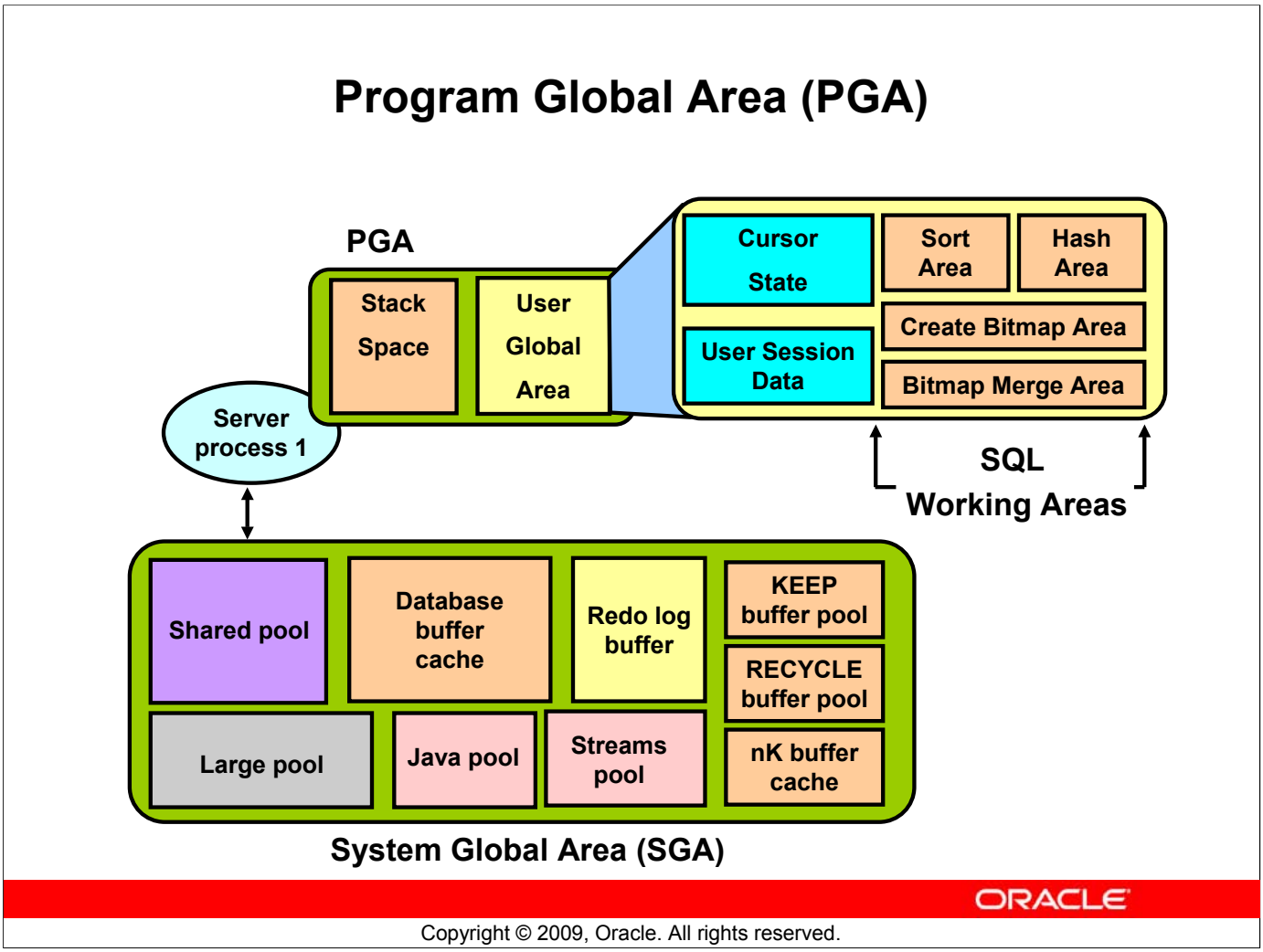
Java Pool and Streams Pool

Java pool memory is used to store all session-specific Java code and data in the JVM. Java pool memory is used in different ways, depending on the mode in which Oracle Database is running.

The Streams pool is used exclusively by Oracle Streams. The Streams pool stores buffered queue messages, and it provides memory for Oracle Streams capture processes and apply processes.

Unless you specifically configure it, the size of the Streams pool starts at zero. The pool size grows dynamically as needed when Oracle Streams is used.

Note: A detailed discussion of Java programming and Oracle Streams is beyond the scope of this class.



Program Global Area (PGA)

The Program Global Area (PGA) is a private memory region containing data and control information for a server process. Each server process has a distinct PGA. Access to it is exclusive to that the server process is read only by Oracle code acting on behalf of it. It is not available for developer's code.

Every PGA contains stack space. In a dedicated server environment, each user connecting to the database instance has a separate server process. For this type of connection, the PGA contains a subdivision of memory known as the user global area (UGA). The UGA is composed of the following:

- Cursor area for storing runtime information on cursors
- User session data storage area for control information about a session
- SQL working areas for processing SQL statements consisting of:
  - A sort area for functions that order data such as `ORDER BY` and `GROUP BY`
  - A hash area for performing hash joins of tables
  - A create bitmap area used in bitmap index creation common to data warehouses
  - A bitmap merge area used for resolving bitmap index plan execution

In a shared server environment, multiple client users share the server process. In this model, the UGA is moved into the SGA (shared pool or large pool if configured) leaving the PGA with only stack space.

Quiz

Memory region that contains data and control information for a server or background process is called:

1. Shared Pool
2. PGA
3. Buffer Cache
4. User session data

Answer: 2

Quiz

What is read into the Database Buffer Cache from the data files?

1. Rows
2. Changes
3. Blocks
4. SQL

Answer: 3

Process Architecture

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

ORACLE

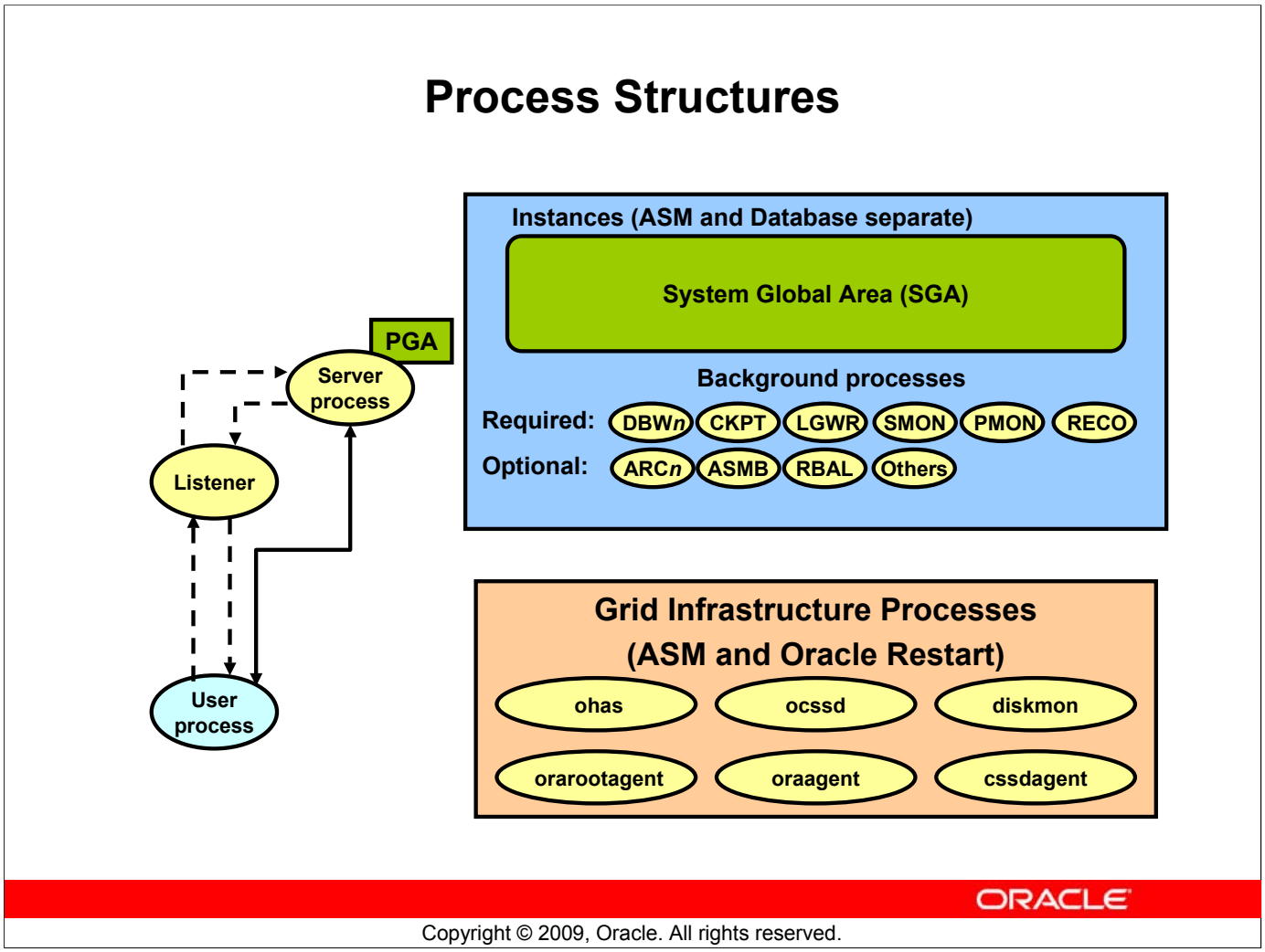
Process Architecture

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

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

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

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



Process Structures

Server Processes

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

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

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

Background Processes

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

Process Structures (continued)

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

- Database writer process (DBW*n*)
- Log writer process (LGWR)
- Checkpoint process (CKPT)
- System monitor process (SMON)
- Process monitor process (PMON)
- Recoverer process (RECO)
- Job queue coordinator (CJQ0)
- Job slave processes (J*nnn*)
- Archiver processes (ARC*n*)
- Queue monitor processes (QMN*n*)

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

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

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

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

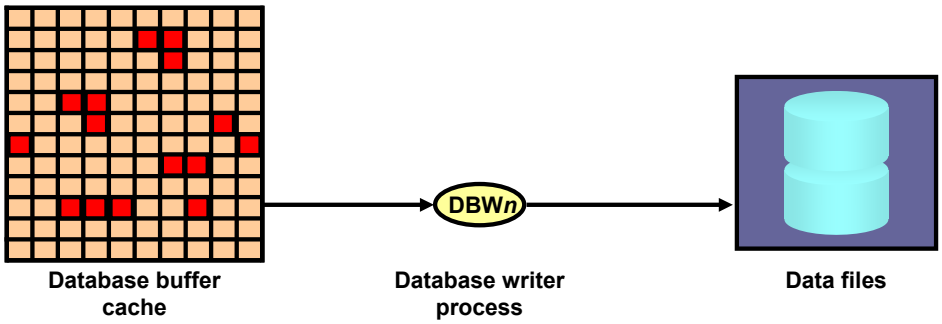
- `ohasd`: Oracle High Availability Service daemon that is responsible to starting Oracle Clusterware processes
- `ocssd`: Cluster Synchronization Service daemon
- `diskmon`: Disk Monitor daemon that is responsible for input and output fencing for HP Oracle Exadata Storage Server
- `cssdagent`: Starts,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

Note: For a more detailed list of the background processes, please consult the *Oracle Background Processes* appendix in this course or the *Oracle Database Reference* guide.

Database Writer Process (DBWn)

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

- Asynchronously while performing other processing
- To advance the checkpoint



ORACLE

Database Writer Process (DBW*n*)

The Database Writer process (DBW*n*) writes the contents of buffers to data files. The DBW*n* processes are responsible for writing modified (dirty) buffers in the database buffer cache to disk. Although one Database Writer process (DBW*0*) is adequate for most systems, you can configure additional processes (DBW*1* through DBW*9* and DBW*a* through DBW*z*) to improve write performance if your system modifies data heavily. These additional DBW*n* processes are not useful on uniprocessor systems.

When a buffer in the database buffer cache is modified, it is marked dirty and is added to the head of the checkpoint queue that is kept in SCN order. This order therefore matches the order of redo that is written to the redo logs for these changed buffers. When the number of available buffers in the buffer cache falls below an internal threshold (to the extent that server processes find it difficult to obtain available buffers), DBW*n* writes non frequently used buffers to the data files from the tail of the LRU list so that processes can replace buffers when they need them. DBW*n* also writes from the tail of the checkpoint queue to keep the checkpoint advancing.

Database Writer Process (DBW*n*) (continued)

The SGA contains a memory structure that has the redo byte address (RBA) of the position in the redo stream where recovery should begin in the case of an instance failure. This structure acts as a pointer into the redo and is written to the control file by the CKPT process once every three seconds. Because the DBW*n* writes dirty buffers in SCN order, and because the redo is in SCN order, every time DBW*n* writes dirty buffers from the LRUW list, it also advances the pointer held in the SGA memory structure so that instance recovery (if required) begins reading the redo from approximately the correct location and avoids unnecessary I/O. This is known as *incremental checkpointing*.

Note: There are other cases when DBW*n* may write (for example, when tablespaces are made read-only or are placed offline). In such cases, no incremental checkpoint occurs because dirty buffers belonging only to the corresponding data files are written to the database unrelated to the SCN order.

The LRU algorithm keeps more frequently accessed blocks in the buffer cache to minimize disk reads. A `CACHE` option can be placed on tables to help retain block even longer in memory.

The `DB_WRITER_PROCESSES` initialization parameter specifies the number of DBW*n* processes. The maximum number of DBW*n* processes is 36. If it is not specified by the user during startup, Oracle Database determines how to set `DB_WRITER_PROCESSES` based on the number of CPUs and processor groups.

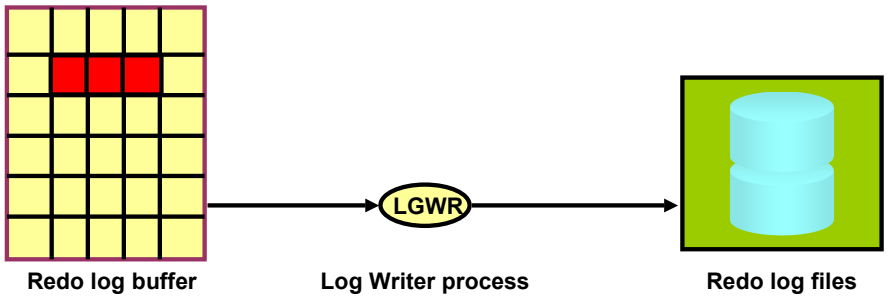
The DBW*n* process writes dirty buffers to disk under the following conditions:

- When a server process cannot find a clean reusable buffer after scanning a threshold number of buffers, it signals DBW*n* to write. DBW*n* writes dirty buffers to disk asynchronously while performing other processing.
- DBW*n* writes buffers to advance the checkpoint, which is the position in the redo thread (log) from which instance recovery begins. This log position is determined by the oldest dirty buffer in the buffer cache.

In all cases, DBW*n* performs batched (multiblock) writes to improve efficiency. The number of blocks written in a multiblock write varies by operating system.

Log Writer Process (LGWR)

- Writes the redo log buffer to a redo log file on disk
- Writes:
  - When a user process commits a transaction
  - When the redo log buffer is one-third full
  - Before a DBWn process writes modified buffers to disk
  - Every 3 seconds



Log Writer Process (LGWR)

The Log Writer process (LGWR) is responsible for redo log buffer management by writing the redo log buffer entries to a redo log file on disk. LGWR writes all redo entries that have been copied into the buffer since the last time it wrote.

The redo log buffer is a circular buffer. When LGWR writes redo entries from the redo log buffer to a redo log file, server processes can then copy new entries over the entries in the redo log buffer that have been written to disk. LGWR normally writes fast enough to ensure that space is always available in the buffer for new entries, even when access to the redo log is heavy. LGWR writes one contiguous portion of the buffer to disk.

LGWR writes:

- When a user process commits a transaction
- When the redo log buffer is one-third full
- Before a DBW*n* process writes modified buffers to disk (if necessary)
- Every three seconds

Log Writer Process (LGWR) (continued)

Before DBW*n* can write a modified buffer, all redo records that are associated with the changes to the buffer must be written to disk (the write-ahead protocol). If DBW*n* finds that some redo records have not been written, it signals LGWR to write the redo records to disk and waits for LGWR to complete writing the redo log buffer before it can write out the data buffers. LGWR writes to the current log group. If one of the files in the group is damaged or unavailable, LGWR continues writing to other files in the group and logs an error in the LGWR trace file and in the system alert log. If all files in a group are damaged, or if the group is unavailable because it has not been archived, LGWR cannot continue to function.

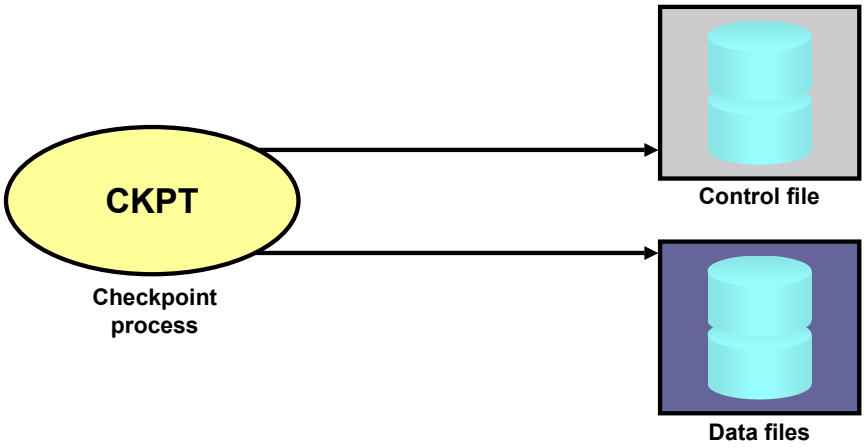
When a user issues a `COMMIT` statement, LGWR puts a commit record in the redo log buffer and writes it to disk immediately, along with the transaction's redo entries. The corresponding changes to data blocks are deferred until it is more efficient to write them. This is called a *fast commit mechanism*. The atomic write of the redo entry containing the transaction's commit record is the single event that determines whether the transaction has committed. Oracle Database returns a success code to the committing transaction, although the data buffers have not yet been written to disk.

If more buffer space is needed, LGWR sometimes writes redo log entries before a transaction is committed. These entries become permanent only if the transaction is later committed. When a user commits a transaction, the transaction is assigned a system change number (SCN), which Oracle Database records along with the transaction's redo entries in the redo log. SCNs are recorded in the redo log so that recovery operations can be synchronized in Real Application Clusters and distributed databases.

In times of high activity, LGWR can write to the redo log file by using group commits. For example, suppose that a user commits a transaction. LGWR must write the transaction's redo entries to disk. As this happens, other users issue `COMMIT` statements. However, LGWR cannot write to the redo log file to commit these transactions until it has completed its previous write operation. After the first transaction's entries are written to the redo log file, the entire list of redo entries of waiting transactions (not yet committed) can be written to disk in one operation, requiring less I/O than do transaction entries handled individually. Therefore, Oracle Database minimizes disk I/O and maximizes performance of LGWR. If requests to commit continue at a high rate, every write (by LGWR) from the redo log buffer can contain multiple commit records.

Checkpoint Process (CKPT)

- Records checkpoint information in
  - Control file
  - Each data file header



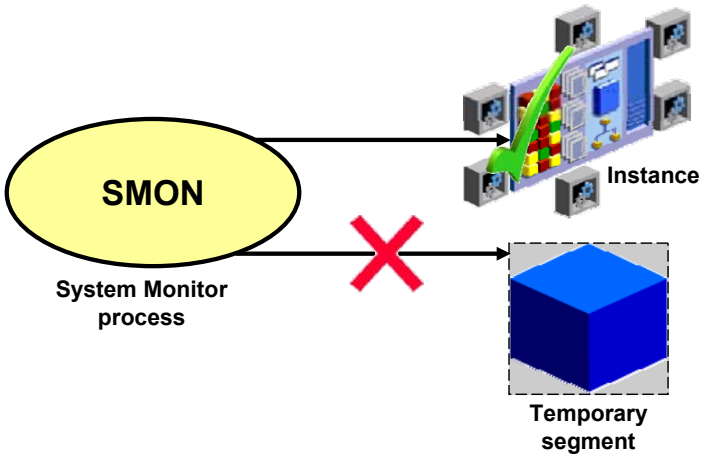
Checkpoint Process (CKPT)

A *checkpoint* is a data structure that defines a system change number (SCN) in the redo thread of a database. Checkpoints are recorded in the control file and in each data file header. They are a crucial element of recovery.

When a checkpoint occurs, Oracle Database must update the headers of all data files to record the details of the checkpoint. This is done by the CKPT process. The CKPT process does not write blocks to disk; DBW*n* always performs that work. The SCNs recorded in the file headers guarantee that all changes made to database blocks prior to that SCN have been written to disk.

System Monitor Process (SMON)

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



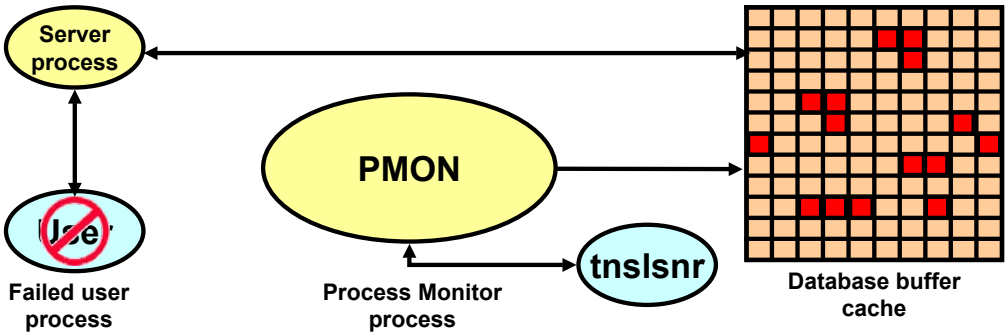
System Monitor Process (SMON)

The System Monitor process (SMON) performs recovery at instance startup if necessary. SMON is also responsible for cleaning up temporary segments that are no longer in use. If any terminated transactions were skipped during instance recovery because of file-read or offline errors, SMON recovers them when the tablespace or file is brought back online.

SMON checks regularly to see whether the process is needed. Other processes can call SMON if they detect a need for it.

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
- Dynamically registers database services with listeners



Process Monitor Process (PMON)

The Process Monitor process (PMON) performs process recovery when a user process fails. PMON is responsible for cleaning up the database buffer cache and freeing resources that the user process was using. For example, it resets the status of the active transaction table, releases locks, and removes the process ID from the list of active processes.

PMON periodically checks the status of dispatcher and server processes, and restarts any that have stopped running (but not any that Oracle Database has terminated intentionally). PMON also registers information about the instance and dispatcher processes with the network listener.

Like SMON, PMON checks regularly to see whether it is needed; it can be called if another process detects the need for it.

Recoverer Process

- 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

ORACLE

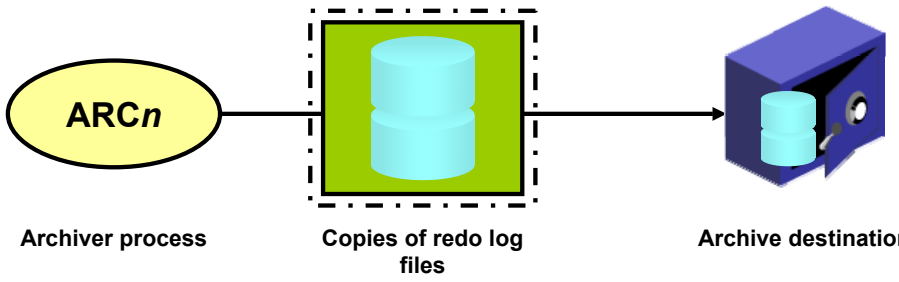
Recoverer Process (RECO)

The Recoverer process (RECO) is a background process that is used with the distributed database configuration that automatically resolves failures involving distributed transactions. The RECO process of an instance automatically connects to other databases involved in an in-doubt distributed transaction. When the RECO process reestablishes a connection between involved database servers, it automatically resolves all in-doubt transactions, removing from each database's pending transaction table any rows that correspond to the resolved in-doubt transactions.

If the RECO process fails to connect with a remote server, RECO automatically tries to connect again after a timed interval. However, RECO waits an increasing amount of time (growing exponentially) before it attempts another connection.

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



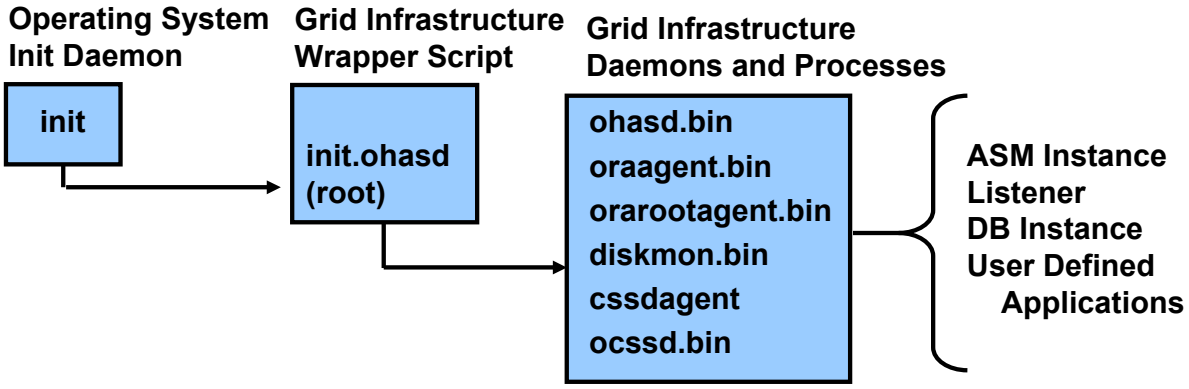
Archiver Processes (ARC*n*)

The archiver processes (ARC*n*) copy redo log files to a designated storage device after a log switch has occurred. ARC*n* processes are present only when the database is in `ARCHIVELOG` mode and automatic archiving is enabled.

If you anticipate a heavy workload for archiving (such as during bulk loading of data), you can increase the maximum number of archiver processes. There can also be multiple archive log destinations. It is recommended that there be at least one archiver process for each destination. The default is to have four archiver processes.

Process Startup Sequence

- Oracle Grid Infrastructure is started by the OS init daemon.



- Oracle Grid Infrastructure installation modifies the `/etc/inittab` file to ensure startup every time machine is started in corresponding run level.

```
# cat /etc/inittab
..
h1:35:respawn:/etc/init.d/init.ohasd run >/dev/null 2>&1 </dev/null
```

ORACLE

Process Startup Sequence

During the installation of Oracle Grid Infrastructure, entries are placed in the operating system `/etc/inittab` file to start a wrapper script. The wrapper script is responsible for setting up environment variables and then starting the Oracle Grid Infrastructure daemons and processes.

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

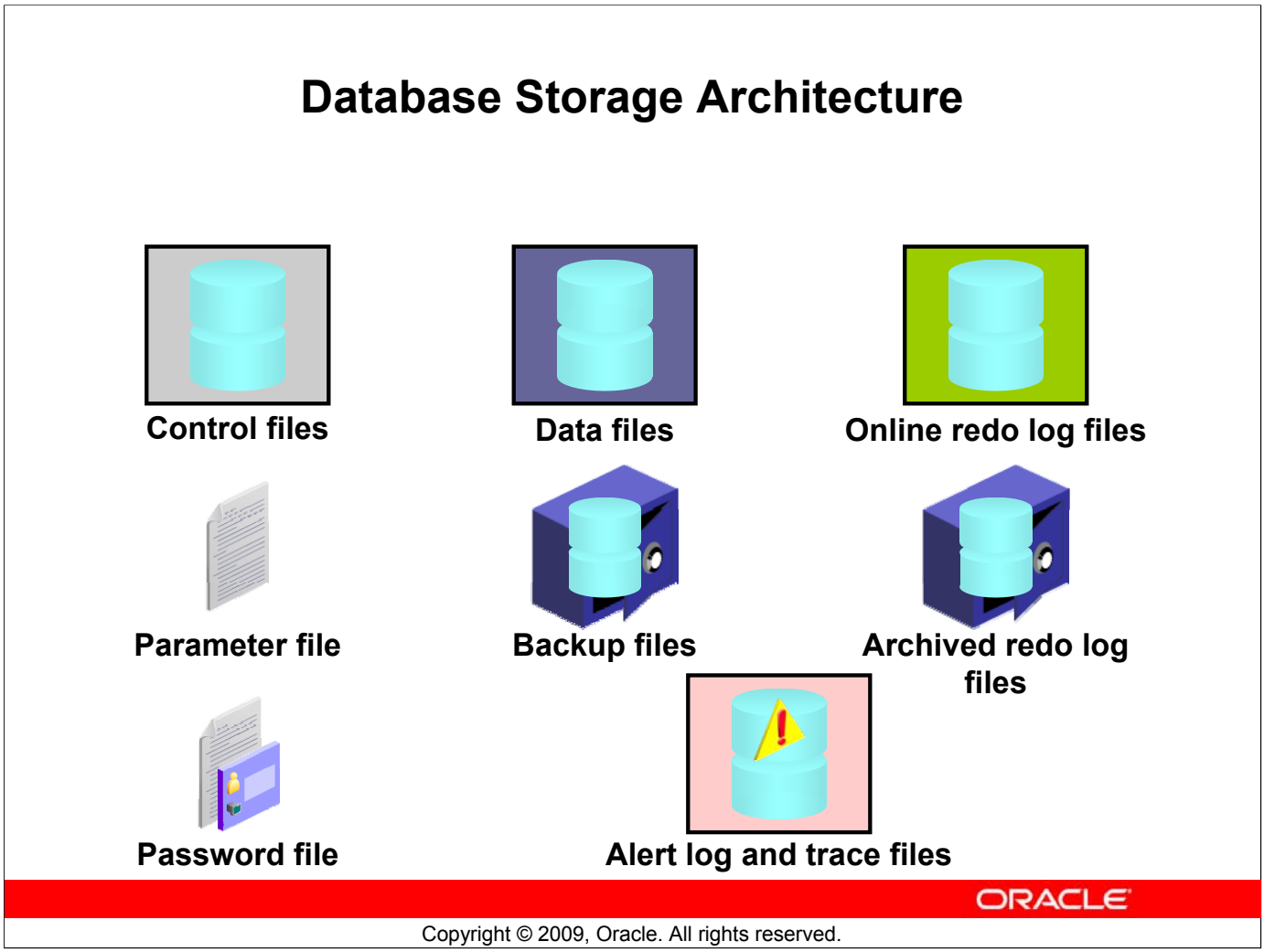
The format of the UNIX `/etc/inittab` file is as follows:

```
id : run levels : action : process with parameters
```

The wrapper script is started with the respawn action so it will be restarted whenever it is terminated.

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

Note: It is not supported to execute the wrapper script directly.



Database Storage Architecture

The files that constitute an Oracle database are organized into the following:

- **Control files:** Contain data about the database itself (that is, physical database structure information). These files are critical to the database. Without them, you cannot open data files to access the data in the database. It can also contain metadata related to backups.
- **Data files:** Contain the user or application data of the database, as well as metadata and the data dictionary
- **Online redo log files:** Allow for instance recovery of the database. If the database server crashes and does not lose any data files, the instance can recover the database with the information in these files.

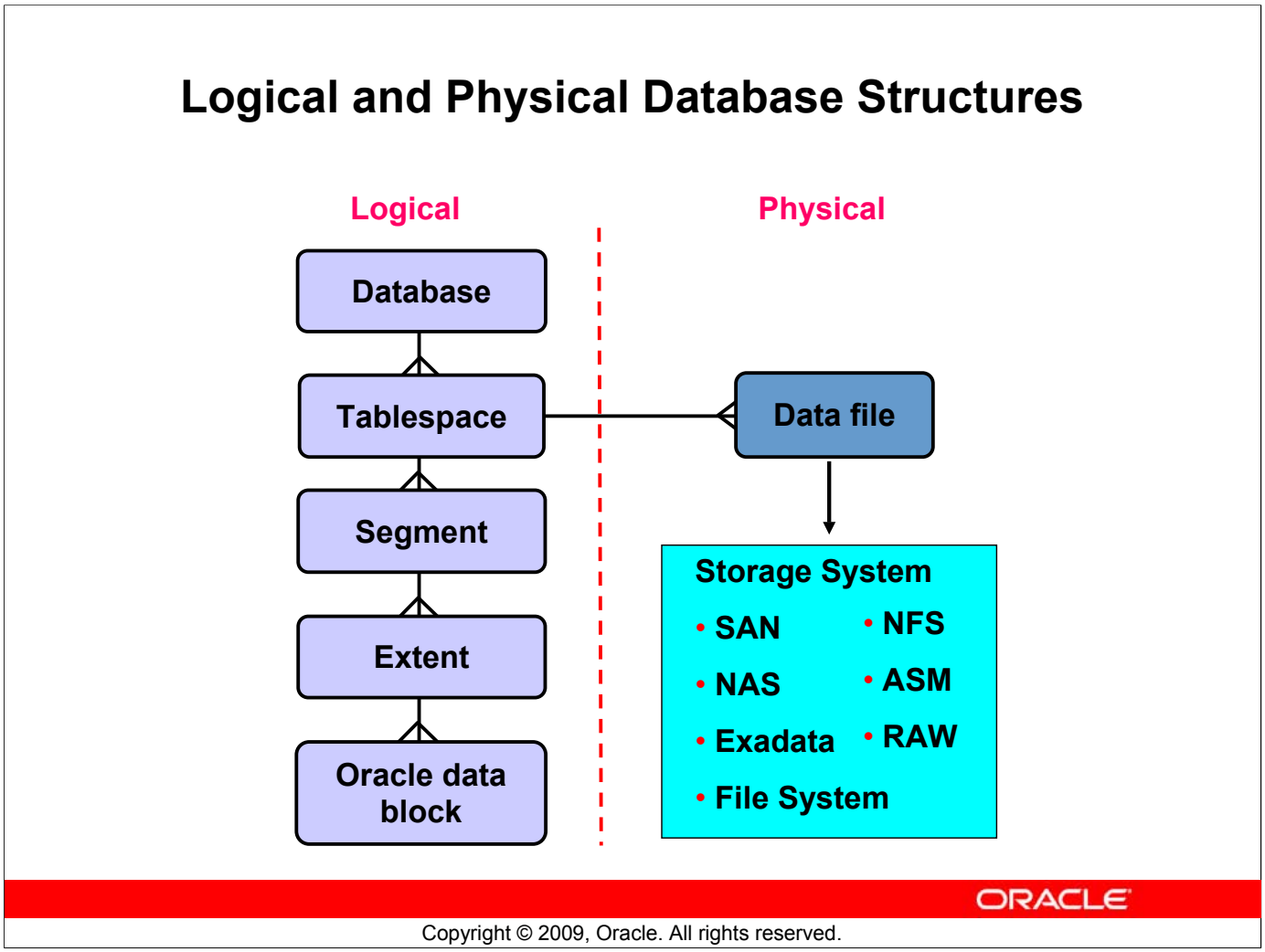
The following additional files are important to the successful running of the database:

- **Parameter file:** Is used to define how the instance is configured when it starts up
- **Password file:** Allows users using the `sysdba`, `sysoper`, and `sysasm` roles to connect remotely to the instance and perform administrative tasks
- **Backup files:** Are used for database recovery. You typically restore a backup file when a media failure or user error has damaged or deleted the original file.
- **Archived redo log files:** Contain an ongoing history of the data changes (redo) that are generated by the instance. Using these files and a backup of the database, you can recover a lost data file. That is, archive logs enable the recovery of restored data files.

Database Storage Architecture (continued)

- **Trace files:** Each server and background process can write to an associated trace file. When an internal error is detected by a process, the process dumps information about the error to its trace file. Some of the information written to a trace file is intended for the database administrator, whereas other information is for Oracle Support Services.
- **Alert log file:** These are special trace entries. The alert log of a database is a chronological log of messages and errors. Oracle recommends that you review the alert log periodically.

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



Logical and Physical Database Structures

The database has logical structures and physical structures.

Databases, Tablespaces, and Data Files

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

Tablespaces

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

Data Blocks

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

Logical and Physical Database Structures (continued)

Extents

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

Segments

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

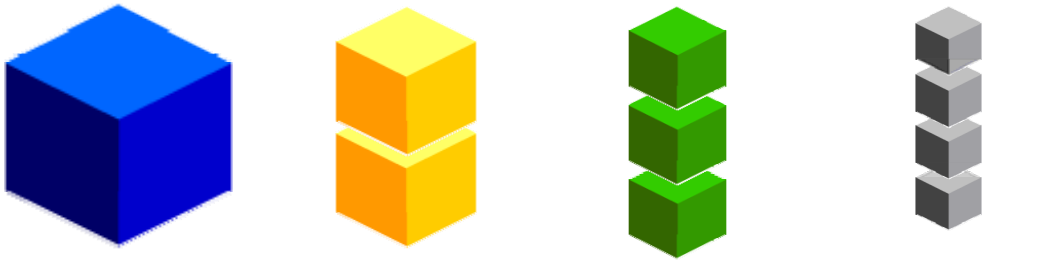
- **Data segments:** Each nonclustered, non-index-organized table has a data segment, with the exception of external tables, global temporary tables, and partitioned tables in which each table has one or more segments. All of the table's data is stored in the extents of its data segment. For a partitioned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

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

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

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.



ORACLE

Segments, Extents, and Blocks

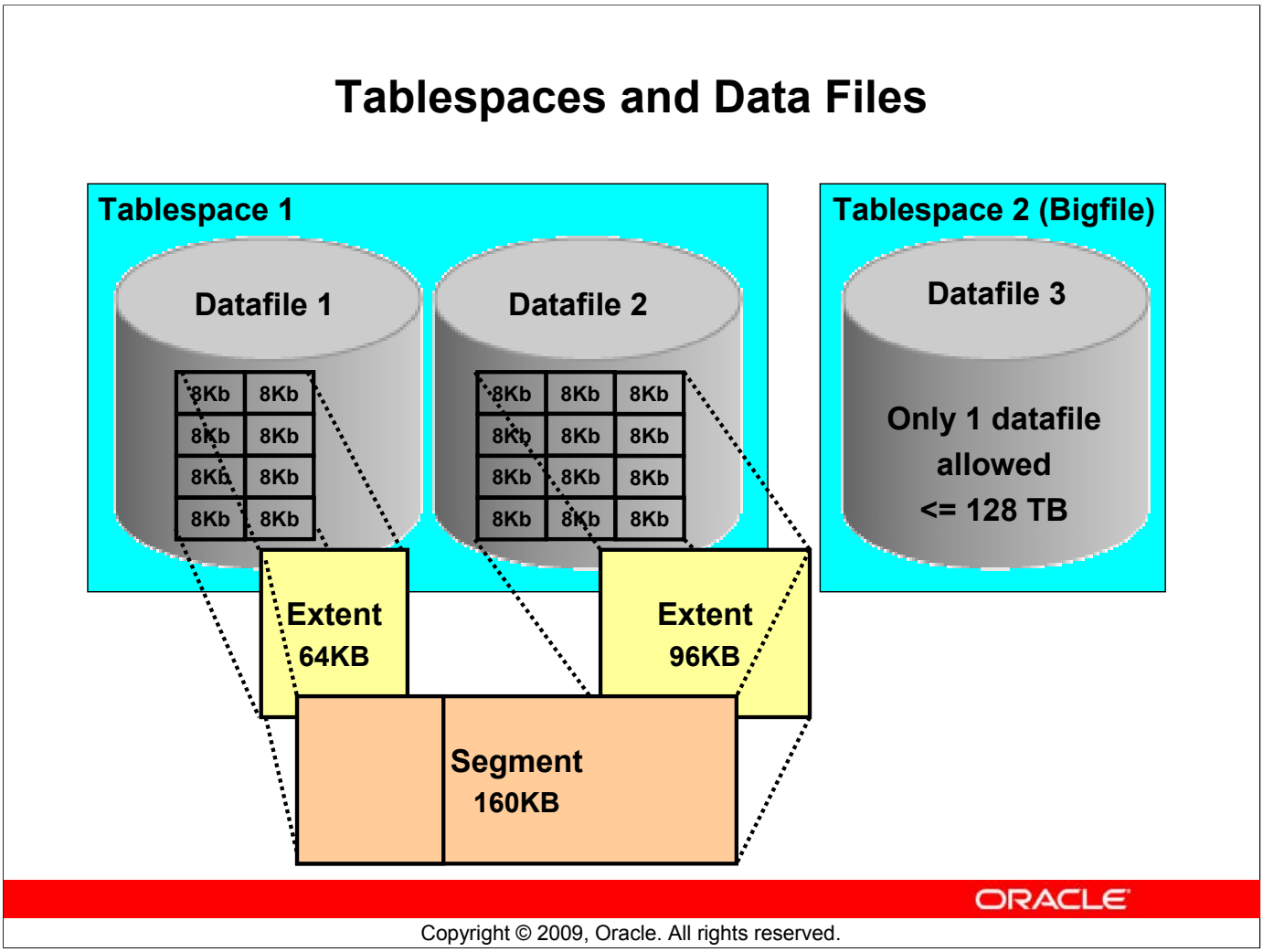
A subset of database objects such as tables and indexes are stored as segments in tablespaces. Each segment contains one or more extents. An extent consists of contiguous data blocks, which means that each extent can exist only in one data file. Data blocks are the smallest unit of I/O in the database.

When the database requests a set of data blocks from the operating system (OS), the OS maps this to an actual file system or disk block on the storage device. Because of this, you do not need to know the physical address of any of the data in your database. This also means that a data file can be striped or mirrored on several disks.

The size of the data block can be set at the time of database creation. The default size of 8 KB is adequate for most databases. If your database supports a data warehouse application that has large tables and indexes, a larger block size may be beneficial.

If your database supports a transactional application in which reads and writes are random, specifying a smaller block size may be beneficial. The maximum block size depends on your OS. The minimum Oracle block size is 2 KB; it should rarely (if ever) be used.

You can have tablespaces with a nonstandard block size. For details, see the *Oracle Database Administrator's Guide*.



Tablespaces and Data Files

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

The graphic on the slide illustrates tablespace one composed of two datafiles. A segment of 160 KB size is spanning the two datafiles, composed of two extents. The first extent of size 64 KB is in the first datafile and the second extent of size 96 KB is in the second datafile. Both extents are formed from contiguous 8Kb Oracle blocks.

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

SYSTEM and SYSAUX Tablespaces

- The SYSTEM and SYSAUX tablespaces are mandatory tablespaces that are created at the time of database creation. They must be online.
- The SYSTEM tablespace is used for core functionality (for example, data dictionary tables).
- The auxiliary SYSAUX tablespace is used for additional database components (such as the Enterprise Manager Repository).
- The SYSTEM and SYSAUX tablespaces are not recommended to be used to store application's data.

ORACLE

SYSTEM and SYSAUX Tablespaces

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

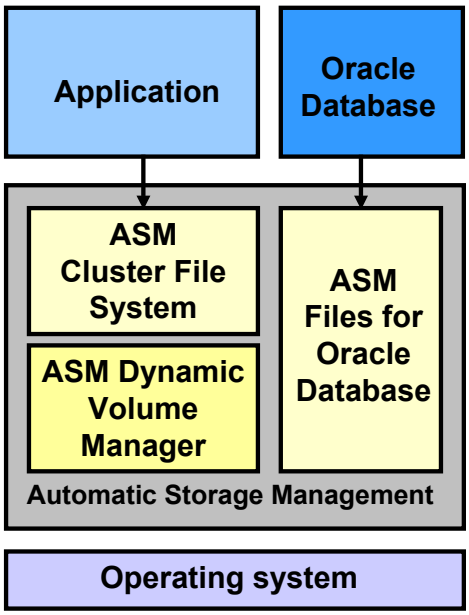
A tablespace can be online (accessible) or offline (not accessible). The SYSTEM tablespace is always online when the database is open. It stores tables that support the core functionality of the database, such as the data dictionary tables.

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

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

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



ORACLE

Automatic Storage Management

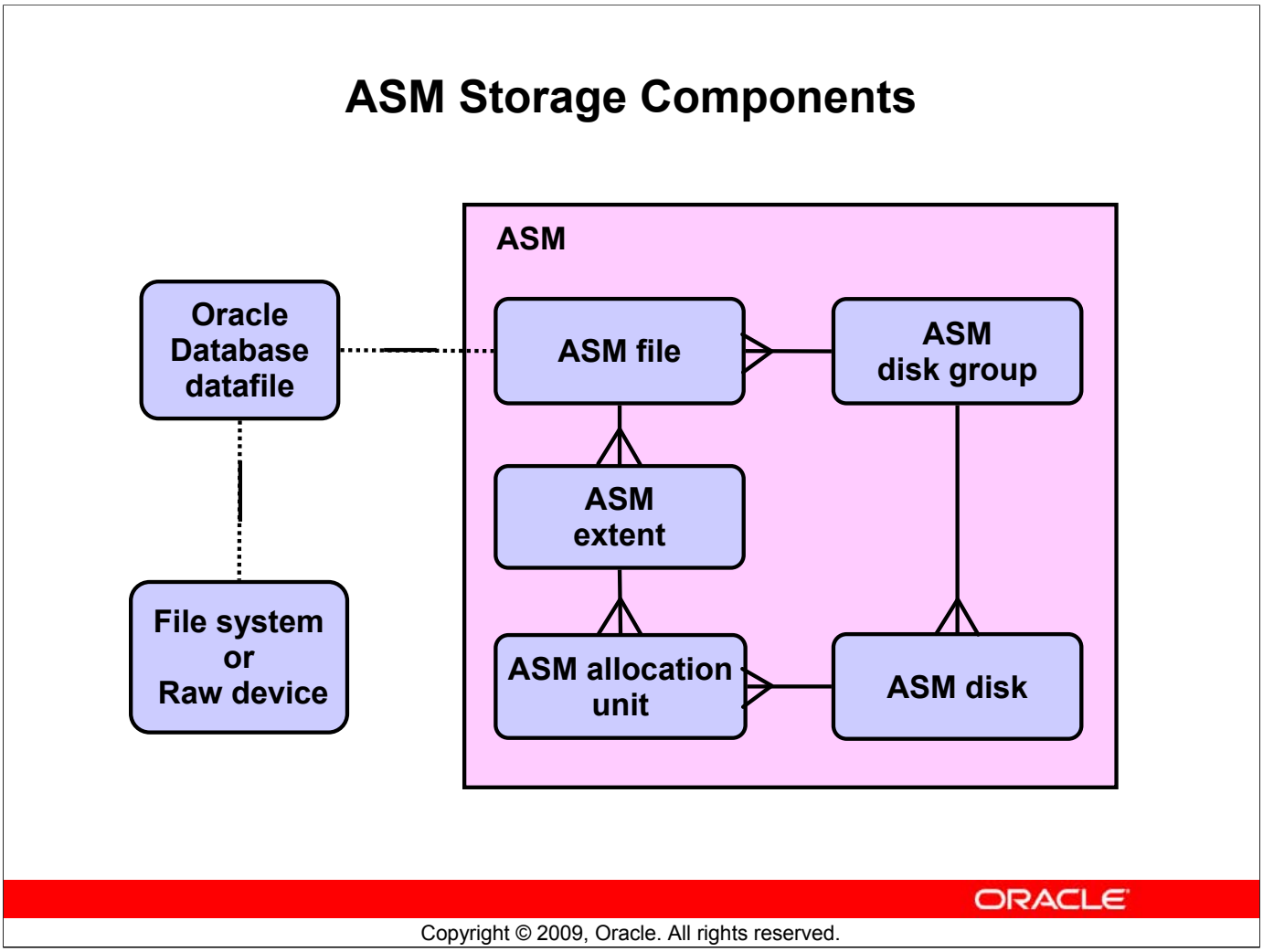
Automatic Storage Management (ASM) provides vertical integration of the file system and the volume manager for Oracle database files. ASM can provide management for single symmetric multiprocessing (SMP) machines or across multiple nodes of a cluster for Oracle Real Application Clusters (RAC) support.

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

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

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

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

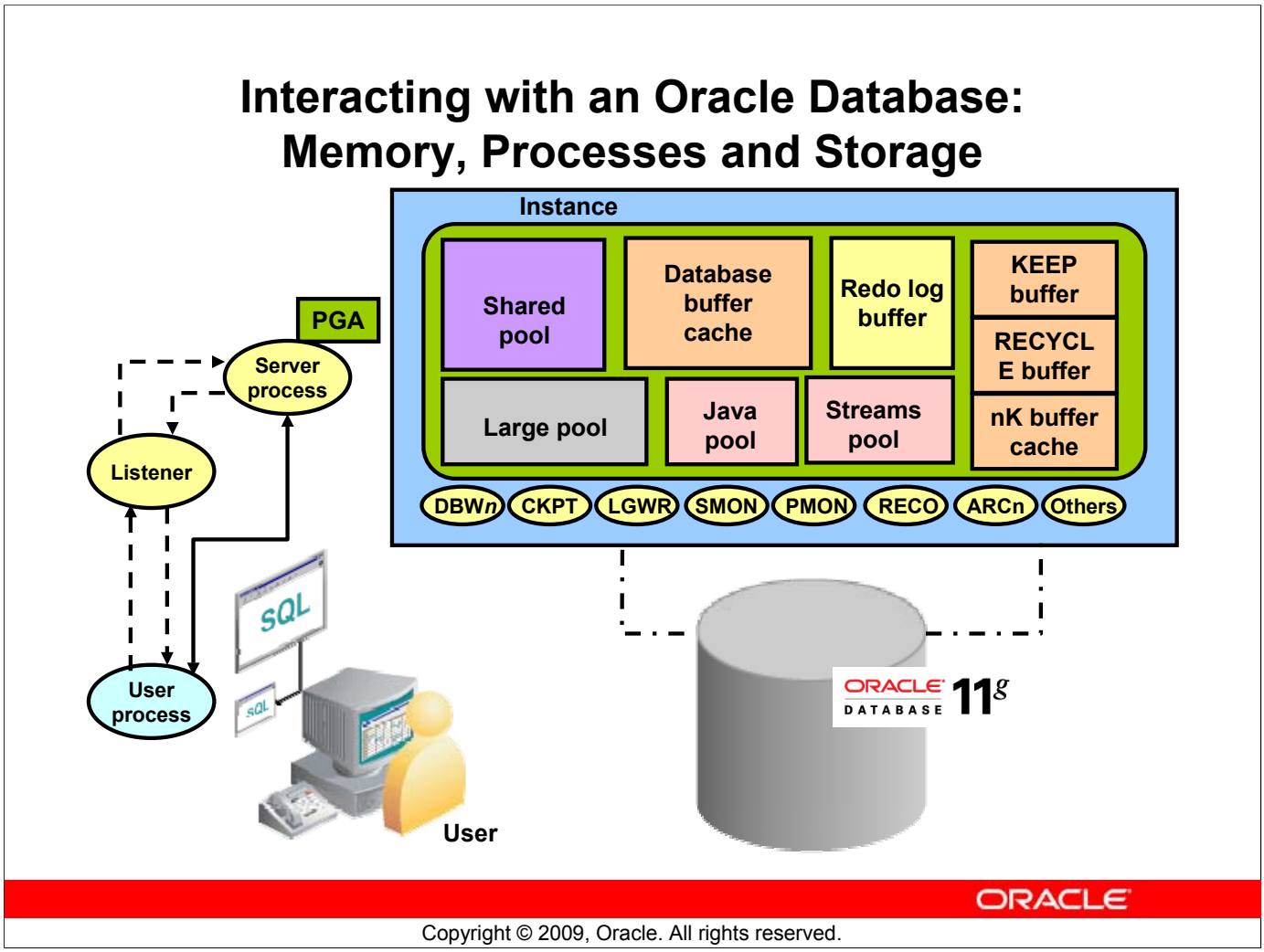


ASM Storage Components

ASM does not eliminate any existing database functionality. Existing databases are able to operate as they always have. New files may be created as ASM files, whereas existing ones are administered in the old way or can be migrated to ASM.

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

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



Interacting with an Oracle Database

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

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

Interacting with an Oracle Database (continued)

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

Quiz

The Process Monitor process (PMON):

1. Performs recovery at instance startup
2. Performs process recovery when a user process fails
3. Automatically resolves all in-doubt transactions
4. Writes the redo log buffer to a redo log file

Answer: 2

Quiz

ASM Files are accessed by which types of instances?

1. RDBMS Instances only
2. ASM Instances only
3. Both RDBMS and ASM Instances

Answer: 3

Summary

In this lesson, you should have learned how to:

- List the major architectural components of Oracle Database
- Explain the memory structures
- Describe the background processes
- Correlate the logical and physical storage structures
- Describe the ASM storage components

ORACLE

Practice 1: Overview

This is a paper practice with questions about:

- Database architecture
- Memory
- Processes
- File structures

2

Installing your Oracle Software

Objectives

After completing this lesson, you should be able to:

- Describe your role as a database administrator (DBA) and explain typical tasks and tools
- Plan an Oracle software installation
- Install Oracle Grid Infrastructure for a standalone server
- Install the Oracle database software

ORACLE

Tasks of an Oracle Database Administrator

The approach for designing, implementing, and maintaining an Oracle database involves the following tasks:

1. Evaluating the database server hardware
2. Installing the Oracle software
3. Planning the database and security strategy
4. Creating, migrating, and opening the database
5. Backing up the database
6. Enrolling system users and planning for their Oracle Network access
7. Implementing the database design
8. Recovering from database failure
9. Monitoring database performance

ORACLE

Tasks of an Oracle Database Administrator

A DBA is typically responsible for installing the Oracle software and creating the database. As a DBA, you may be responsible for creating database storage structures, such as tablespaces. In addition, you may create the schema or set of objects to hold application data.

You must ensure that the database is available for users. You accomplish this by starting up the database, backing up the database on a regular basis, and monitoring the performance of the database. These tasks should be performed within the framework of a security strategy.

As you proceed through the lessons in this course, you learn how to perform each of these tasks. You can also refer to the *Oracle Database Administrator's Guide* for additional information about each of the tasks outlined in the slide.

In this lesson, you focus on installation. For this core task, consider the following subtasks:

- Understanding how the installation fits into the overall technical architecture of an organization
- Reviewing (and updating) capacity plans
- Choosing the database software (required version and options)
- Ensuring that system requirements are met for all chosen elements

Tools for Administering an Oracle Database

- Oracle Universal Installer
- Database Configuration Assistant
- Database Upgrade Assistant
- Oracle Net Manager
- Oracle Net Configuration Assistant
- Oracle Enterprise Manager
- Server Control Utility
- SQL*Plus
- Recovery Manager
- Data Pump
- SQL*Loader

ORACLE

Tools for Administering an Oracle Database

You can use the following tools for installation and upgrade:

- **Oracle Universal Installer (OUI):** Installs your Oracle software and options; can automatically launch the Database Configuration Assistant to create a database
- **Database Configuration Assistant (DBCA):** Creates a database from Oracle-supplied templates, enabling you to copy a preconfigured seed database (Alternatively, you can create your own database and templates.)
- **Database Upgrade Assistant (DBUA):** Guides you through the upgrade of your existing database to a new Oracle release
- **Oracle Net Manager (`netmgr`):** Configures network connectivity for your Oracle databases and applications
- **Oracle Net Configuration Assistant (NetCA):** A graphical, wizard-based tool—used to configure and manage Oracle Network configurations

Tools for Administering an Oracle Database (continued)

The following tools are used to manage your Oracle instance and database:

- **Oracle Enterprise Manager (EM):** Combines a graphical console, agents, common services, and tools to provide an integrated and comprehensive system management platform for managing Oracle products. After you have installed the Oracle software, created or upgraded a database, and configured the network, you can use EM as the single interface for managing your database. In addition to providing a Web-based user interface for executing SQL commands, it interfaces with other Oracle components that are used to administer your database (for example, Recovery Manager and Scheduler).
- The main EM tools that are used to administer an Oracle database are:
  - **Enterprise Manager Database Console:** To administer one database
  - **Enterprise Manager Grid Control:** To administer many databases at the same time
- **Server Control Utility (`srvctl`):** Standard command-line interface that can be used to start and stop the database and instances, manage ASM instances, manage configuration information, and move or remove instances and services. You can also use `SRVCTL` to add services and manage configuration information.
- **SQL*Plus:** Standard command-line interface for managing your database
- **Recovery Manager (RMAN):** Oracle tool that provides a complete solution for the backup, restoration, and recovery needs of the entire database or of specific database files
- **Data Pump:** Enables the high-speed transfer of data from one database to another (For example, you may want to export a table and import it into another database.)
- **SQL*Loader:** Enables the loading of data from an external file into an Oracle database; one of several Oracle utilities that you can use to load data into database tables
- **Command-line tools:**
  - To administer Enterprise Manager:
    ```
    emctl start | status | stop dbconsole
    ```
  - To administer the listener:
    ```
    lsnrctl start | status | stop
    ```

Planning Your Installation

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

ORACLE

Planning Your Installation

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

- **What Oracle Software Are You Installing?**
  For the purposes of this course it is assumed you are installing the Oracle database. Oracle recommended best practice is to use Automatic Storage Management (ASM) as the storage technique. This involves the installation of Oracle Grid Infrastructure from the cluster software media. This installs the components required for ASM as well as Oracle Restart.
- **Does the Hardware Involved Meet the Minimum Required Specifications?**
  Identify all hardware that will be involved in the installation process and ensure that the minimum suggested specifications are met.
- **Is There a Recommended Order of Installation When Multiple Products Are Involved?**
  Whenever possible it is recommended that the Oracle Grid Infrastructure be installed before the Oracle database software. Performing the installation in this order means that a newly created database can be configured to use ASM disk groups and this database is automatically registered with Oracle Restart. If Oracle Grid Infrastructure is installed after the Oracle database then manual configuration steps are required to register that database with Oracle Restart. Migration steps are required if you want this existing database to use ASM disk groups for storage.

Planning Your Installation (continued)

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

Oracle Grid Infrastructure and Oracle Database Installation: System Requirements

- Memory requirements:
  - 1 GB for the database instance with Oracle Enterprise Manager Database Control
  - 1.5 GB for the ASM instance and Oracle Restart
- Disk space requirements:
  - 3 GB of swap space (based on 2 GB RAM)
  - 1 GB of disk space in the /tmp directory
  - 3.8 GB for the Oracle Database software
  - 4.5 GB for the Grid Infrastructure software
  - 1.7 GB for the preconfigured database (optional)
  - 3.4 GB for the fast recovery area (optional)
- Operating system (see documentation)

ORACLE

Oracle Grid Infrastructure and Oracle Database Installation: System Requirements

- A standard database installation can be completed on a computer with 1 GB of RAM and 1.5 GB of swap space or larger. A standard Oracle Grid Infrastructure for standalone server installation requires a minimum of 1.5 GB RAM as well.
- The amount of swap space required is dependent on the amount of RAM (for example, for 2 GB RAM you need 3 GB of swap space). Please see the OS specific installation guide for more details.
- Depending on the activity level of the machine on which you are installing the Oracle Database software, the standard installation can complete in 20 minutes or less.
- Some installation details:
  - Oracle Database 11*g* ships two seed database templates.
  - Duplicated files are removed.
  - Many other products and demonstrations are installable from additional CDs.

The hardware requirements listed in the slide are minimal requirements across all platforms. Your installation may have additional requirements (especially disk space).

Note: An Enterprise Edition installation type that includes a standard seed database is referred to as a “standard installation.”

Preparing the Operating System

Create the required operating system users and groups:

- Groups:
  - `oinstall`
  - `dba`
  - Optional groups (if doing separation of duty across multiple users):
    - `oper`
    - `asmdba`
    - `asmoper`
    - `asmadmin`
- Users:
  - Software owner, usually `oracle`
  - Can create multiple users for multiple product installations

ORACLE

Preparing the Operating System

There are steps that need to be performed by the systems administrator for the hardware you are using for your Oracle installation. Going into all the operating system configuration details and commands is beyond the scope of this course. As well, each operating system has specific requirements for your Oracle software. Instead we will go through some of the high-level requirements and recommend that you refer to your operating system–specific installation documentation for your Oracle installation.

One of the required steps is to create the necessary operating system groups and users. The two required operating system groups are: `oinstall` and `dba`. If you are implementing separation of duty across multiple user accounts then other groups that should be created are: `oper`, `asmdba`, `asmoper`, and `asmadmin`. You need at least one operating system user to act as the owner of your Oracle installation. In most cases the user `oracle` is configured for this purpose. If you want to have a true separation of duty then you can have separate owners for each of your Oracle products.

Setting Environment Variables

Oracle environment variables:

- ORACLE_BASE: Base of the Oracle directory structure. Recommended to set this before installation.
- ORACLE_HOME: The environment in which Oracle products run. Not required before installation if ORACLE_BASE is set.
- ORACLE_SID: Not required before installation, but useful afterwards for ease of interaction with a particular instance
- NLS_LANG: Optional environment variable that controls language, territory, and client character set settings

ORACLE

Setting Environment Variables

There are many Oracle environment variables involved in each Oracle environment. Those mentioned here are important to the successful installation and use of an Oracle database. None of these are required to be set, but you can avoid future problems by setting them.

- **ORACLE_BASE:** Specifies the base of the Oracle directory structure for Optimal Flexible Architecture (OFA) that is recommended by Oracle Support. Use is optional; if used, it can facilitate future installations and upgrades. This is a directory path, as shown in the following example:
  ```
  /u01/app/oracle
  ```
- **ORACLE_HOME:** The environment in which Oracle products run. Not required before installation if ORACLE_BASE is set. The OUI can use the ORACLE_BASE setting to determine the recommended ORACLE_HOME for your installation. Having this environment variable makes maintenance and management of Oracle software easier. This is a directory path, as shown in the following example:
  ```
  /u01/app/oracle/product/11.2.0/dbhome_1
  ```
- **ORACLE_SID:** The system identifier for an Oracle instance, such as orcl for a database or +ASM for an ASM instance. Not required before installation, but useful afterwards for ease of interaction with a particular instance.
- **NLS_LANG:** Optional environment variable that controls language, territory, and client character set settings as in the following example:
  ```
  AMERICAN_DENMARK.WE8MSWIN1252
  ```
  For more information about valid languages, territories, character sets, and language support, see the Oracle Database Globalization Support Guide.

Checking the System Requirements

- Adequate temporary space
- 64-bit versus
- 32-bit issues
- Correct operating system (OS)
- OS patch level
- System packages
- System and kernel parameters
- X Server permissions
- Sufficient swapping
- ORACLE_HOME status

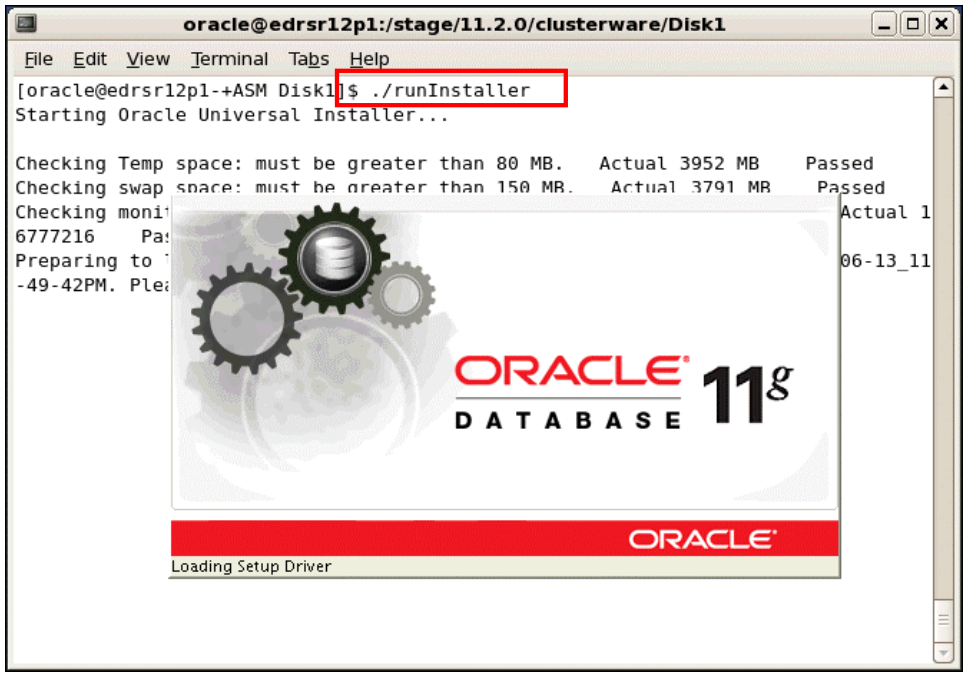
```
[oracle@edrsr12p1-+ASM Disk1]$ ./runInstaller
Starting Oracle Universal Installer...

Checking Temp space: must be greater than 80 MB.   Actual 15067 MB    Passed
Checking swap space: must be greater than 150 MB.   Actual 4000 MB    Passed
Checking monitor: must be configured to display at least 256 colors.    Actual 6
5536    Passed
Preparing to launch Oracle Universal Installer from /tmp/OraInstall2009-05-15_12
-04-10AM. Please wait ...
```


Checking the System Requirements

The Oracle Universal Installer automates most of the prerequisite checks to verify the following:

- Minimum temporary space requirements for installation and configuration are checked. Those requirements are validated during the installation process.
- 64-bit installations are prevented from being installed into Oracle homes with 32-bit software already installed (and vice versa).
- Oracle Grid Infrastructure 11*g* and Oracle Database 11*g* is certified against several versions of the Linux platform, as well as other platforms.
- All required OS patches are installed.
- All required system and kernel parameters are set correctly.
- The `DISPLAY` environment variable is set and the user has sufficient permissions to display to the specified `DISPLAY`.
- The system has a sufficient swapping set.
- The Oracle home for the new installation either is empty or is one of a handful of supported releases on top of which Oracle Database 11*g* can be installed. The installation process also verifies that those releases are registered in the Oracle inventory.

Oracle Universal Installer (OUI)



Oracle Universal Installer (OUI)

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

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

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

Example: Installation Scenario

The installation scenario being presented in this lesson is divided into two parts:

- Part One: Install Oracle Grid Infrastructure for standalone server
- Part Two: Install Oracle Database software

ORACLE

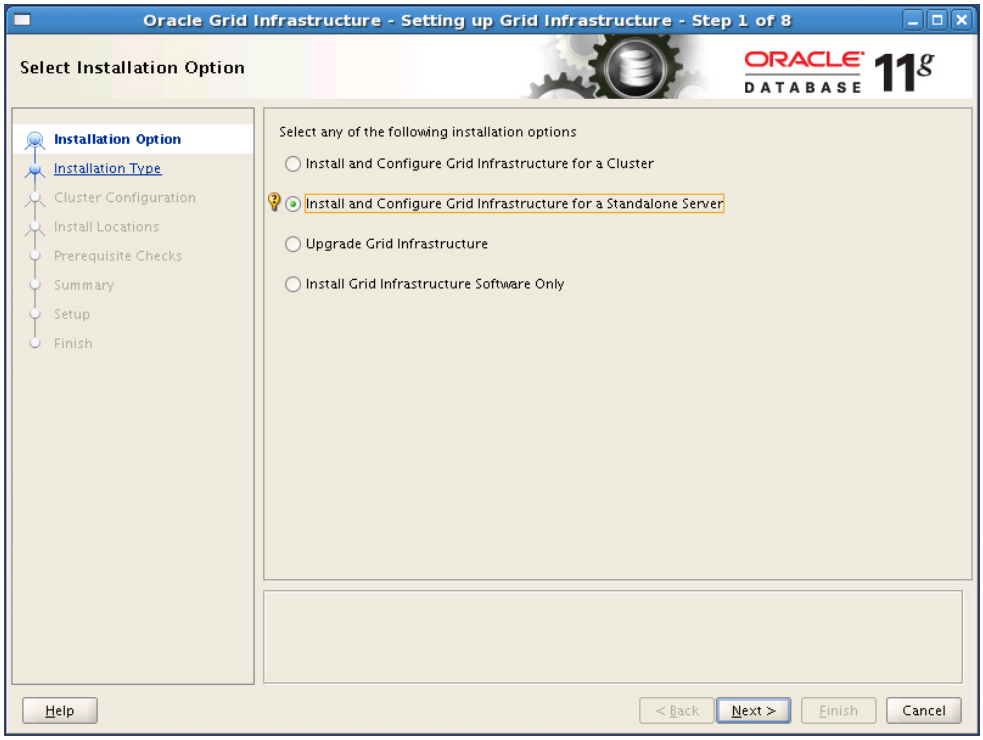
Example: Installation Scenario

The installation scenario being presented in this lesson is divided into two parts:

- **Part One:** Install Oracle Grid Infrastructure for standalone server
- **Part Two:** Install Oracle Database software

As part of the Oracle Grid Infrastructure installation, the steps to configure the ASM disk groups are presented and Oracle Restart is configured. The Oracle Grid Infrastructure is installed first so that the database created after the Oracle Database software installation will be able to use the ASM disk groups and be automatically registered with Oracle Restart.

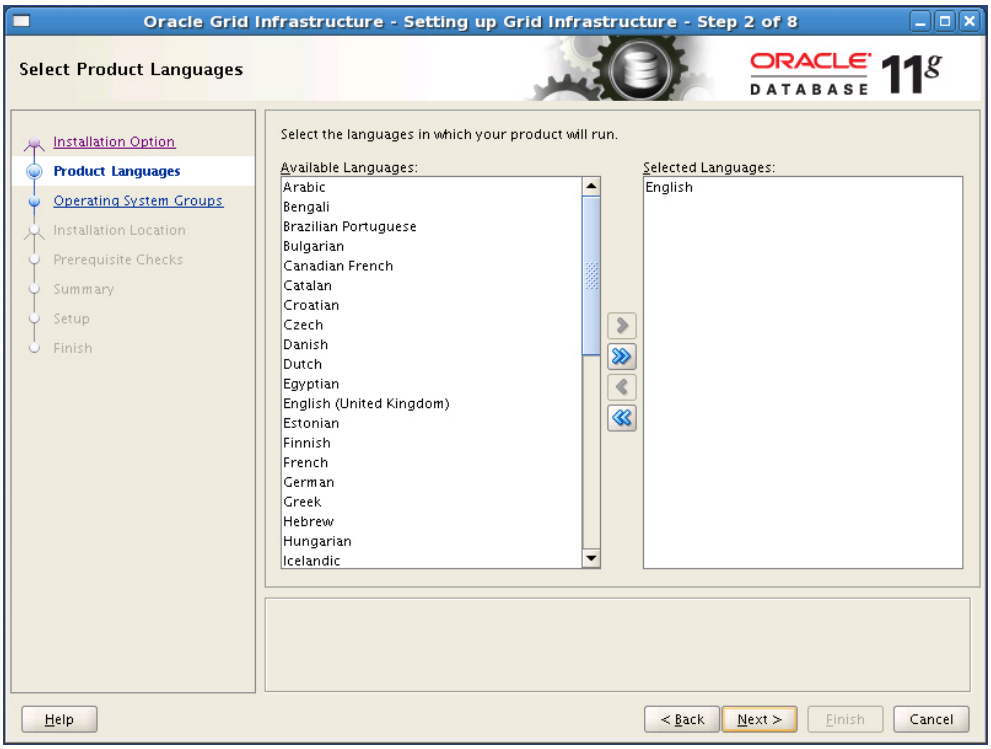
Part One: Installing the Oracle Grid Infrastructure for Standalone Server



Part One: Installing the Oracle Grid Infrastructure for Standalone Server

To install the Oracle Grid Infrastructure software by using the Oracle Universal Installer (OUI), log on to your computer as a member of the administrative group that is authorized to install the Oracle software and to create and manage the database. Insert the distribution CD for the clusterware into your CD drive, or navigate to the Oracle clusterware staging location. From that location enter `./runInstaller` to start the OUI. The Installation Option of the OUI appears. Select the Install and Configure Grid Infrastructure for a Standalone Server option and click Next.

Selecting Product Languages



ORACLE

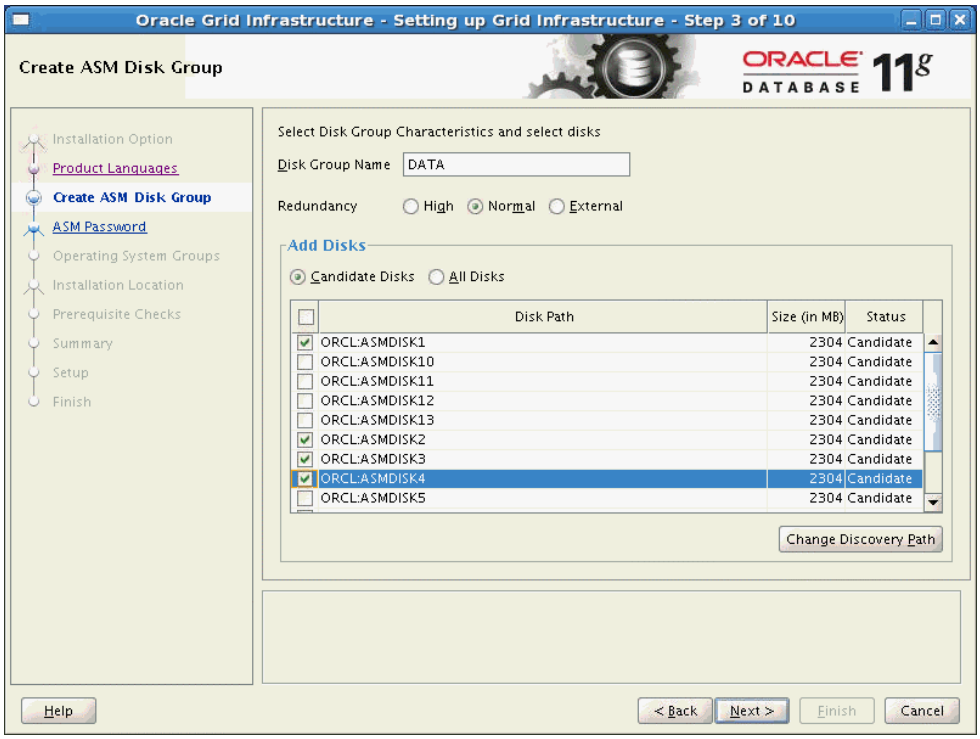
Selecting Product Languages

The "Select Product Languages" page is displayed. To add a language to the installation, click the language to highlight it and then use the right-arrow button to move it the "Selected Languages" list. Multiple languages can be selected by holding down the control key while selecting them with the mouse. Click the Next button to proceed with the installation.

The current list of languages includes Arabic, Bengali, Brazilian Portuguese, Bulgarian, Canadian French, Catalan, Croatian, Czech, Danish, Dutch, Egyptian, English (United Kingdom), Estonian, Finnish, French, German, Greek, Hebrew, Hungarian, Icelandic, Indonesian, Italian, Japanese, Korean, Latin American Spanish, Latvian, Lithuanian, Malay, Mexican Spanish, Norwegian, Polish, Portuguese, Romanian, Russian, Simplified Chinese, Slovak, Slovenian, Spanish, Swedish, Thai, Traditional Chinese, Turkish, Ukrainian, and Vietnamese.

Note: This list is subject to change with updates.

Creating an ASM Disk Group



ORACLE

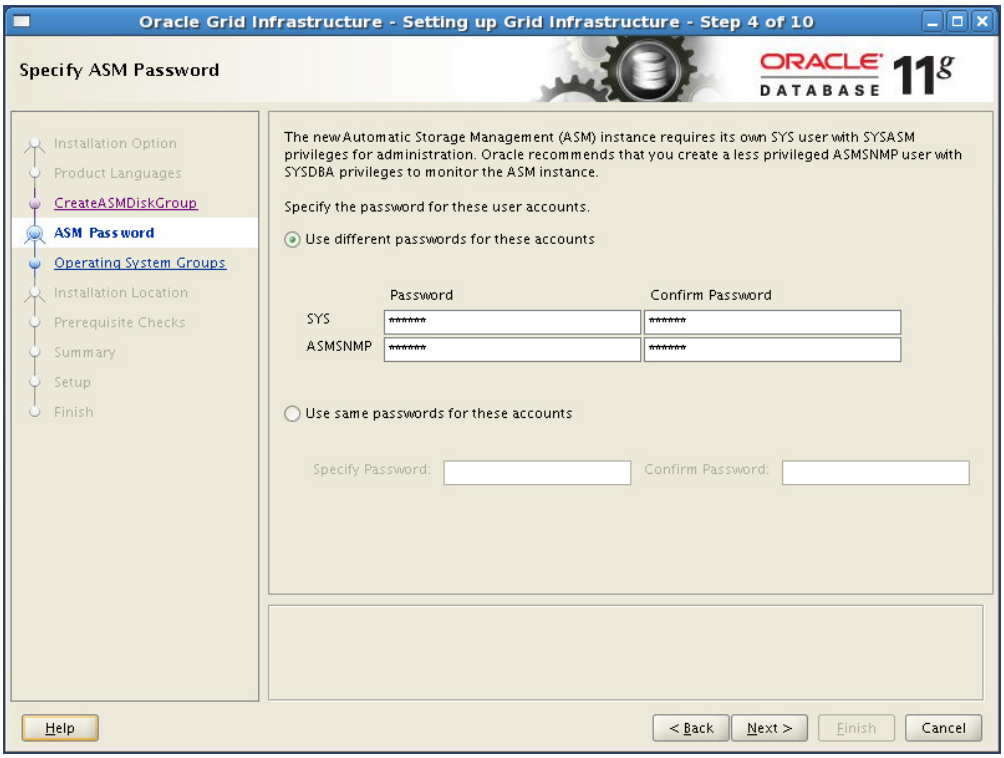
Creating an ASM Disk Group

The “Create ASM Disk Group” page is displayed. The Oracle Grid Infrastructure includes support for ASM and Oracle Restart. The OUI will not proceed unless an ASM disk group is created. Enter the name of the first ASM disk group to be created in the “Disk Group Name” field. If the “Add Disks” section is not populated with candidate disks, the click the “Change Discovery Path” button, and enter the disk discovery path to the ASM disks.

Oracle recommends that you create two disk groups for most environments. The OUI is only capable of creating a single ASM disk group at this time. You may create the recommended second ASM disk group after the installation using the ASM Configuration Assistant (`asmca`) utility or SQL*Plus. Click the Next button to proceed with the installation.

Note: Devices appear in the list for our example due to ASMLib being configured. ASMLib is only for Linux platforms. On other platforms you may need to click the Change Discovery Path button.

Defining ASM Passwords

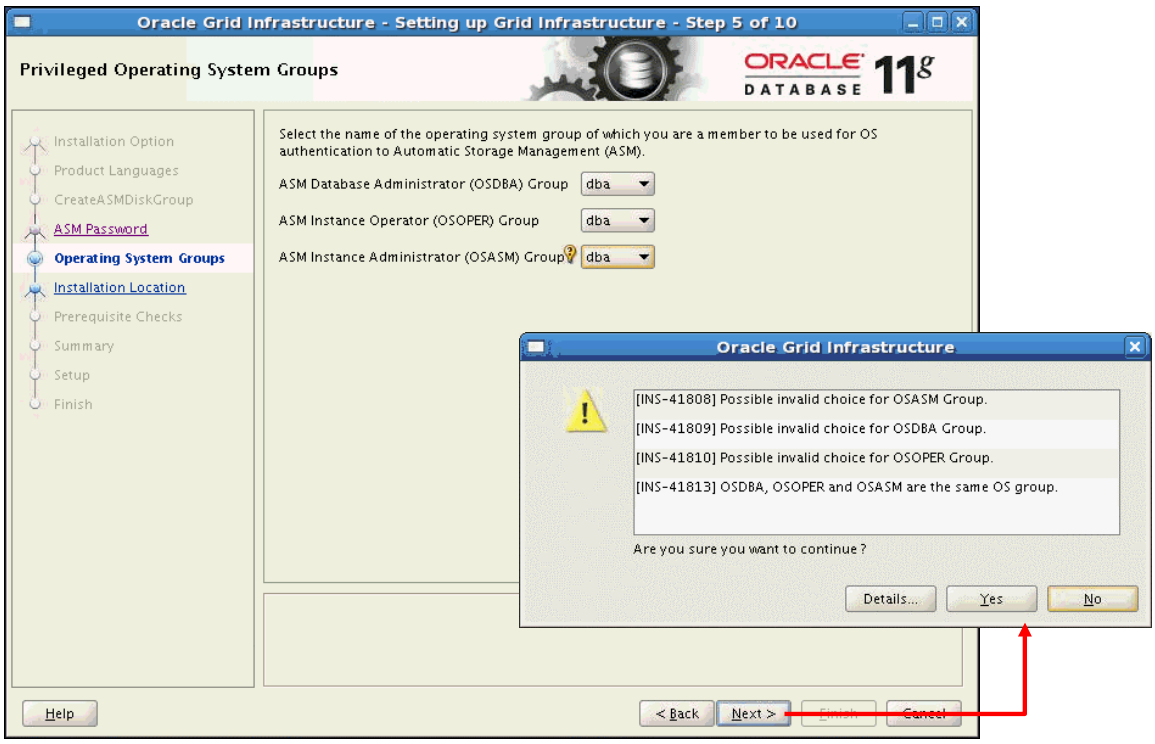


ORACLE

Defining ASM Passwords

The “Specify ASM Password” page is displayed. Passwords must be supplied for two accounts, the `SYS` account and the `ASMSNMP` account. The option to use different passwords for the accounts or the option to use the same password for the accounts is presented. The ASM instance does not contain a data dictionary like an Oracle Database, so the only authentication methods are operating system authentication and password file authentication. The `SYS` account will be added to the password file (`orapw+ASM` on Linux) and granted the `SYSDBA`, `SYSOPER`, and `SYSASM` privileges. The `ASMSNMP` account will be added to the password file and granted only the `SYSDBA` privilege. After entering appropriate passwords, click the Next button to proceed with the installation.

Defining Privileged Operating System Groups



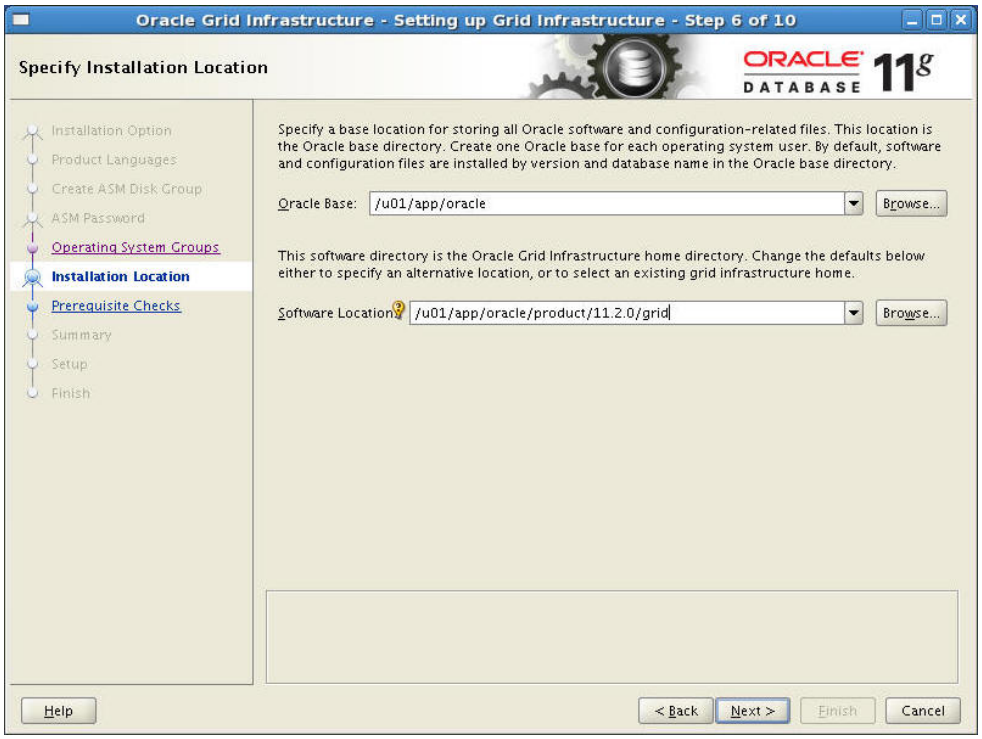
Defining Privileged Operating System Groups

The "Privileged Operating System Groups" page is displayed. The OUI utility provides suggested default values for the following groups if the current user is an operating system member of the groups:

- ASM Database Administrator (OSDBA) Group – `asmdba`
- ASM Instance Operator (OSOPER) Group – `asmoper`
- ASM Instance Administrator (OSASM) Group – `asmadmin`

Because this installation is for a standalone server, it is common practice to use the same operating system group for all three, such as `dba` (as shown in the slide). Click the Next button to proceed with the installation. Click Yes in response to the warning that appears because we are using the same operating system group for OSDBA, OSOPER, and OSASM.

Specifying Installation Location

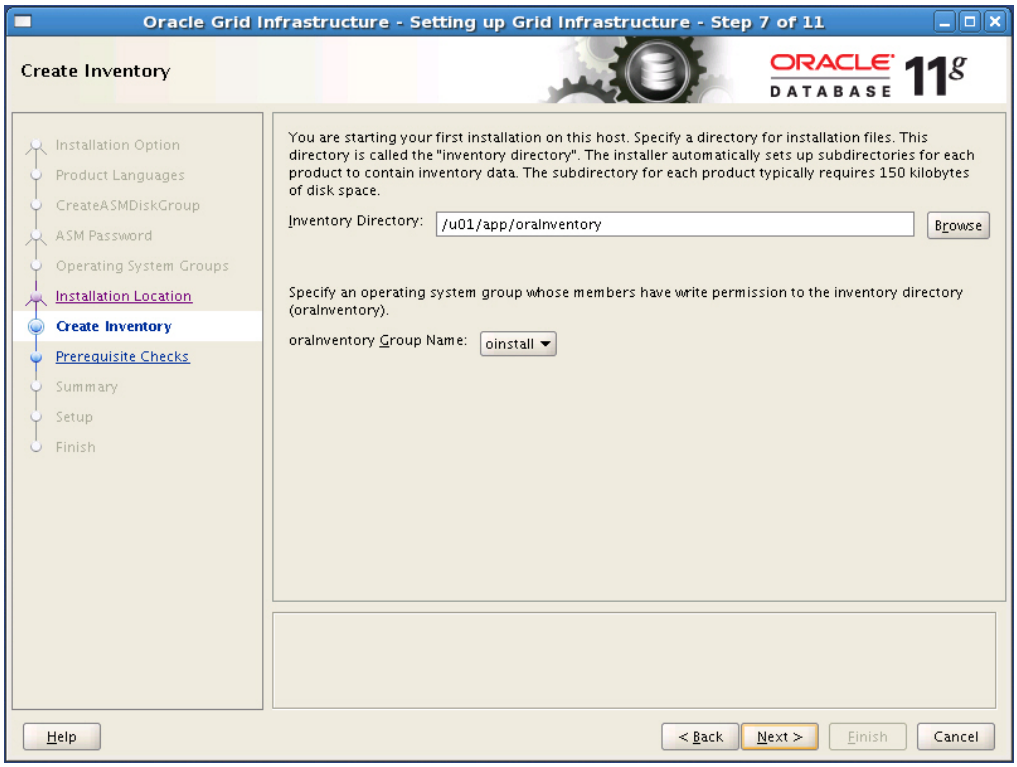


ORACLE

Specifying Installation Location

The “Specify Installation Location” page is displayed. For the “Oracle Base” field, enter the value of the `ORACLE_BASE` for the software owner. The default value is `/u01/app/oracle.` For the “Software Location” field, enter the value of the `ORACLE_HOME` for the Grid Infrastructure software. The default value is `/u01/app/oracle/product/11.2.0/grid`. Click the Next button to proceed with the installation.

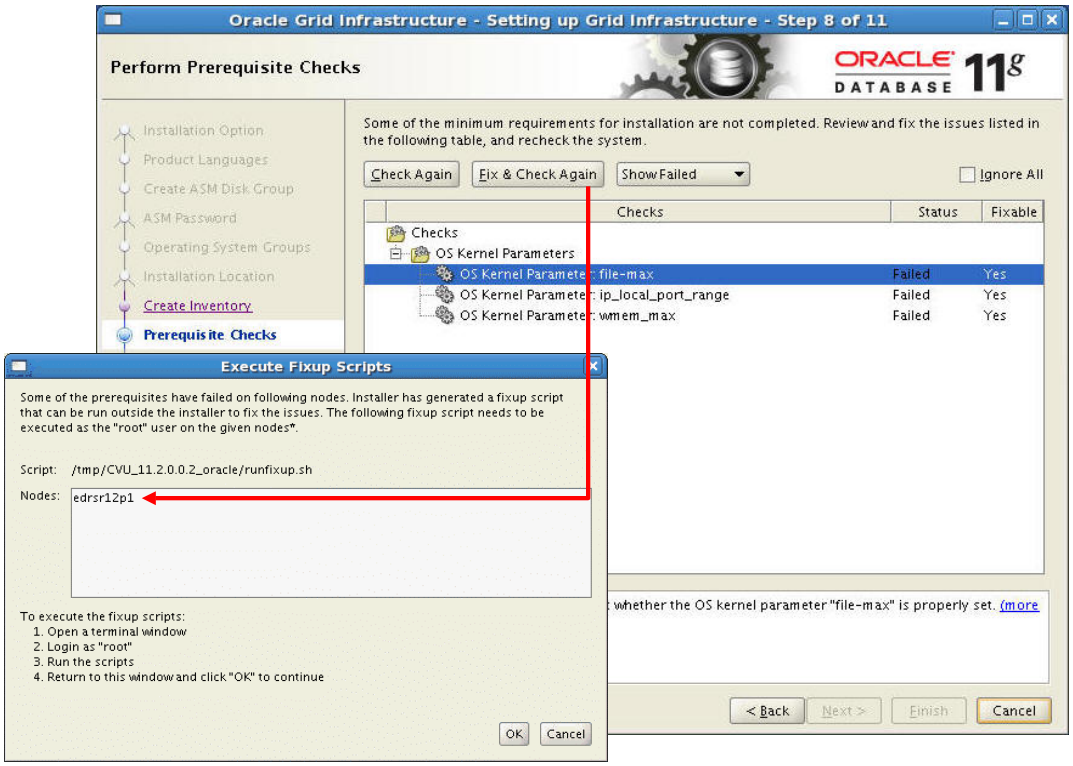
Creating Inventory



Creating Inventory

If an Oracle Central Inventory directory does not already exist on the host machine, then the "Create Inventory" page is displayed. For the "Inventory Directory" field, enter the desired location for the `oraInventory` directory. The recommended directory is `/u01/app/oraInventory`. The `oraInventory` directory should be one level higher than the `ORACLE_BASE` directory. Select the `oraInventory` group name from the selection list. The recommended name is `oinstall`. Click the Next button to proceed with the installation.

Performing Prerequisite Checks

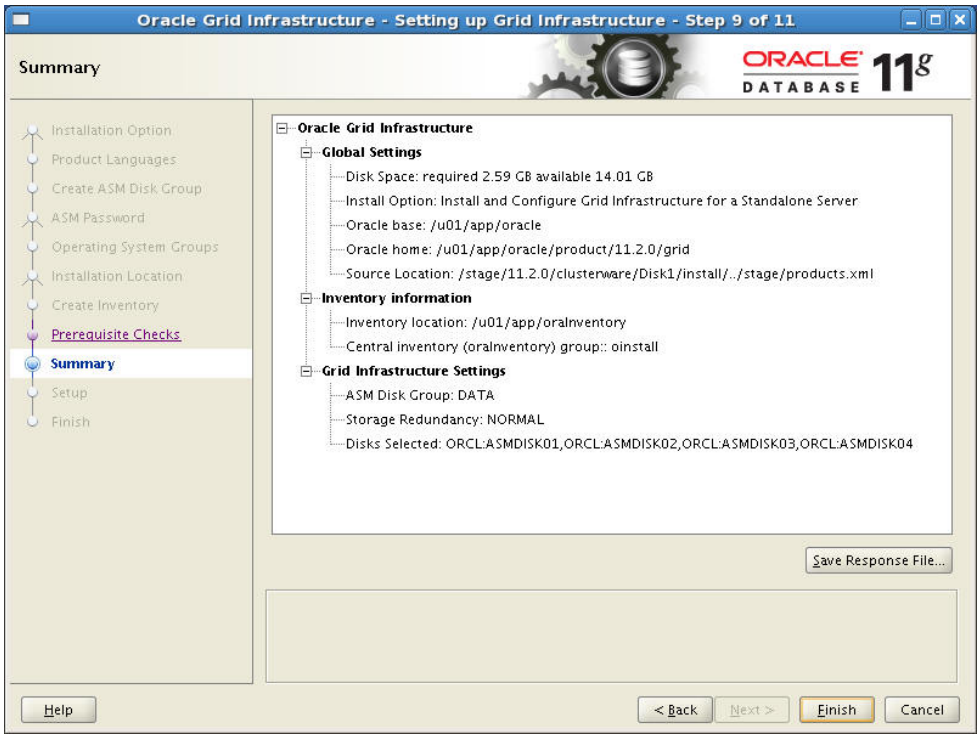


Performing Prerequisite Checks

The “Perform Prerequisite Checks” page appears. There is nothing to do on this page unless the checks fail with errors. If errors do occur, they will need to be corrected before proceeding with the installation. In the example on the slide, there are some kernel parameters that did not pass the prerequisite checks. You are instructed to run a script that will correct these kernel parameters and then run the checks again. If the error is one that is not fixable, you must either fix the problem manually or choose to Ignore All and continue on with the installation (if the failed check is not critical).

Note: Not all failures can be fixed by the `runfixup.sh` script generated by the OUI. In such cases, manual intervention to correct the problem is required before you can reattempt the installation.

Verifying Installation Summary Data

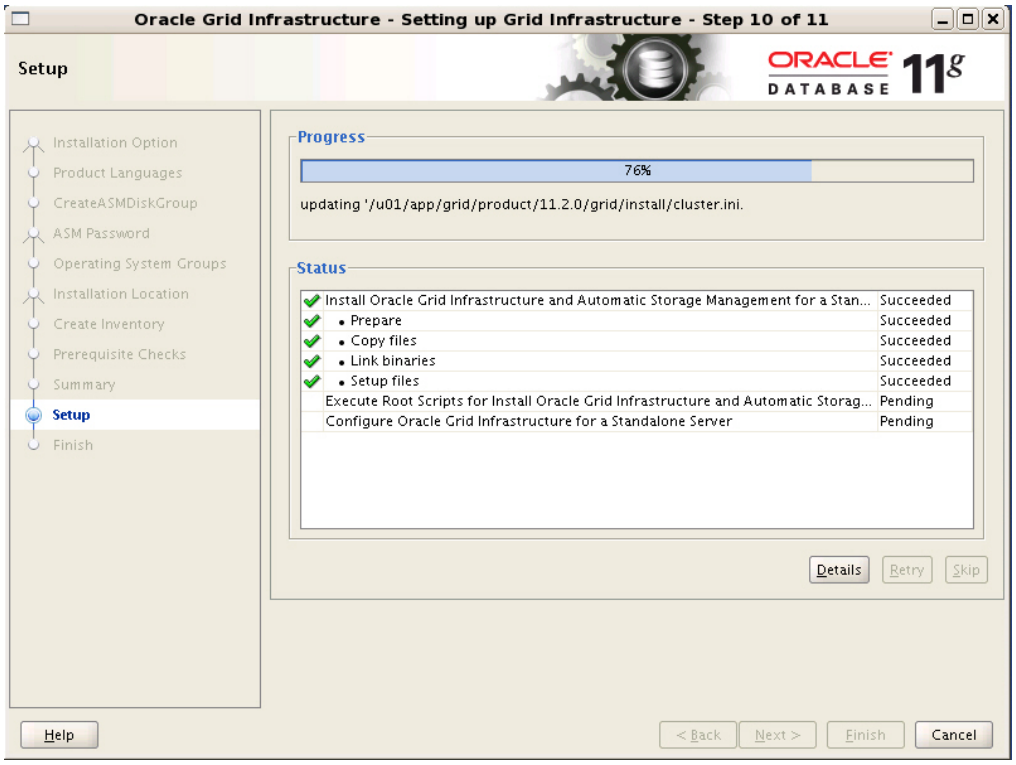


ORACLE

Verifying Installation Summary Data

The “Summary” page is displayed. Review the information presented that is related to the installation answers provided on previous pages. You are presented with the option to save the interactive installation questions and answers to a response file. A response file can be used in future installations to perform a silent installation in the same manner in which this installation is being performed. The response file is a text file and can also be edited with a text-editing tool to modify it for a different installation. Click the Finish button to proceed with the installation.

Monitoring Installation Progress

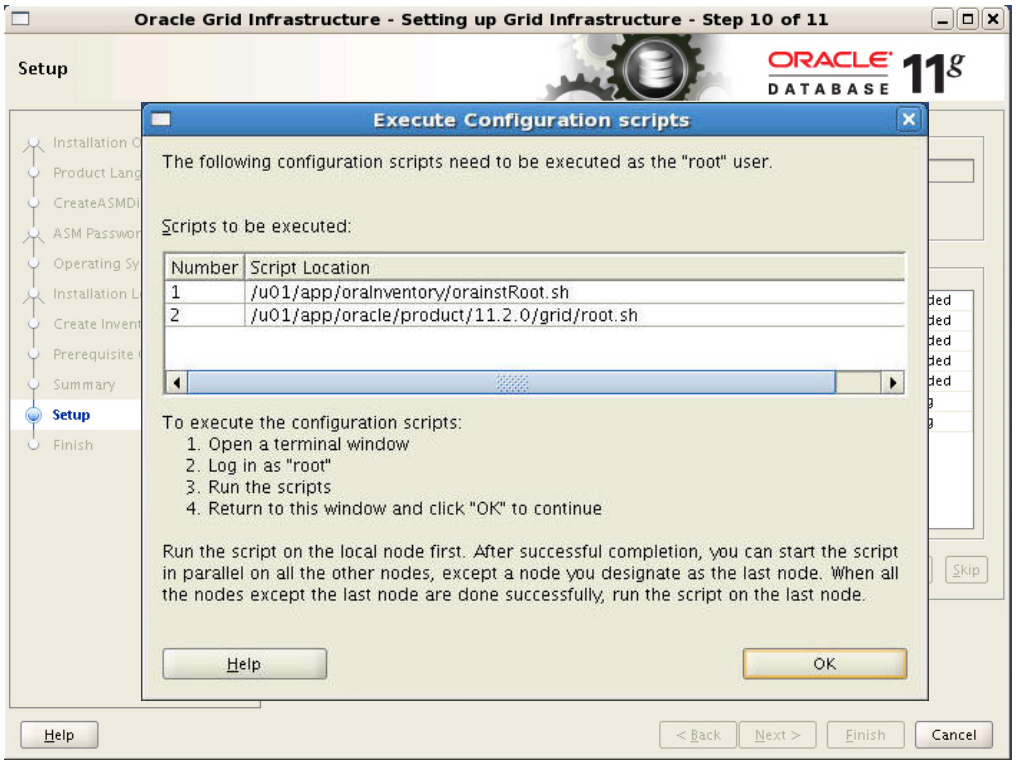


ORACLE

Monitoring Installation Progress

The “Setup” page appears, showing the progress of the installation. The progress will include preparing for installation, copying files, linking libraries, and creating setup files. There is nothing to do on this page unless the installation fails with errors. If errors do occur, they will need to be corrected before proceeding with the installation.

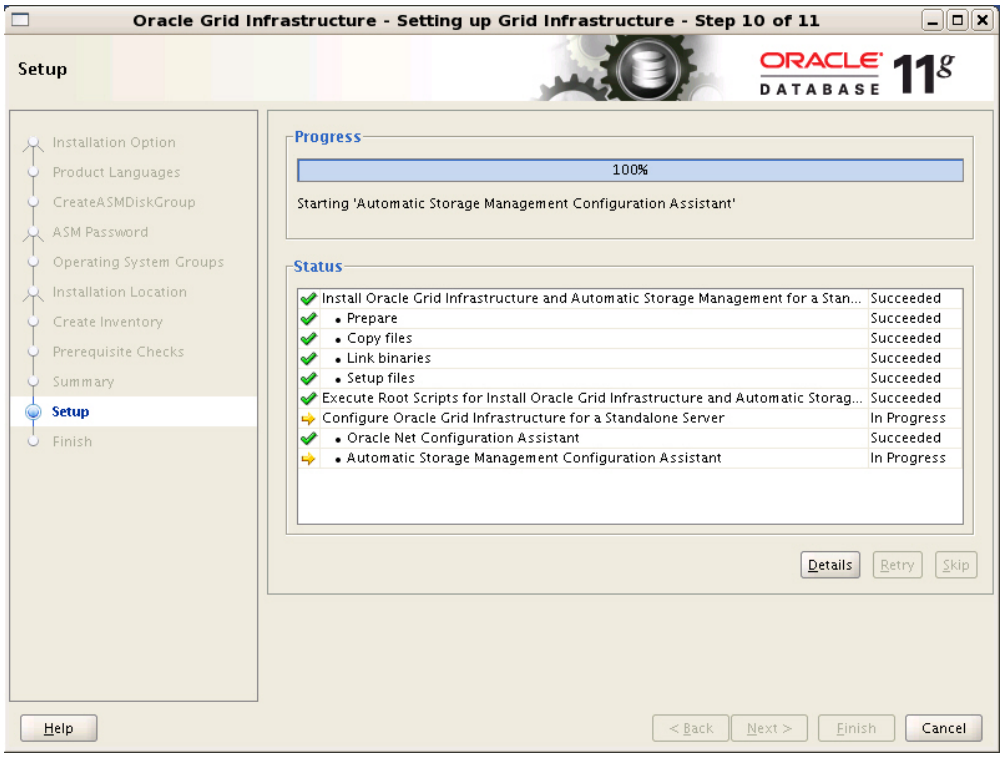
Executing `root` Configuration Scripts



Executing `root` Configuration Scripts

The “Execute Configuration Scripts” dialog page appears next, indicating configuration scripts that need to be executed as the `root` user. The `orainstRoot.sh` script is responsible for changing the permission of the Oracle Central Inventory directory by adding read and write permissions for the group, along with removing the read, write, and execute permissions for world. The `root.sh` script copies files to the `/usr/local/bin` directory, creates the `/etc/oratab` file, creates OCR keys for the `grid` user, starts the `ohasd` daemon, and modifies the `/etc/inittab` to automatically start the `ohasd` daemon when the machine is started. After you execute the `root` scripts, click the OK button to return to the "Setup" progress page to proceed with the installation.

Executing Configuration Assistants

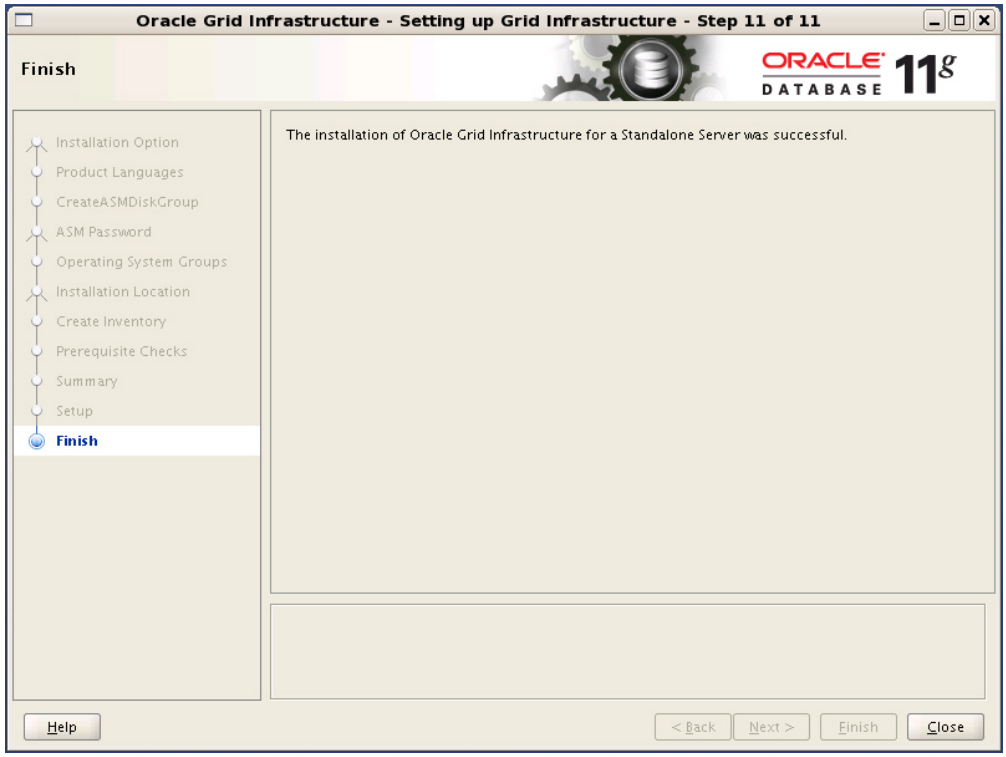


ORACLE

Executing Configuration Assistants

The Oracle Net Configuration Assistant (`netca`) is invoked with a silent install method to create the network configuration files, followed by the Automatic Storage Management Configuration Assistant (`asmca`) to create the ASM disk group and register ASM components with Oracle Restart.

Finishing Installation

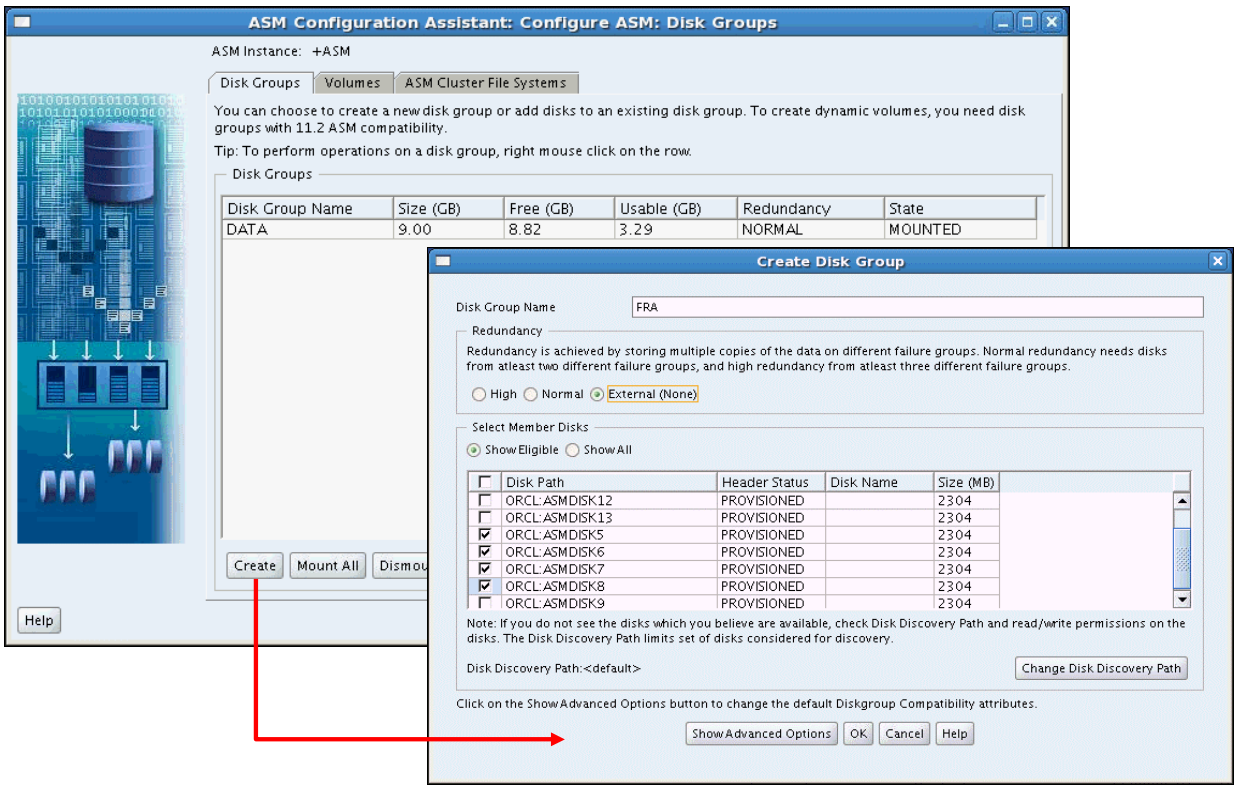


ORACLE

Finishing Installation

The “Finish” page is displayed at the end of the installation. Click the Close button to exit the OUI utility.

Configuring the FRA Disk Group



Configuring the FRA Disk Group

Because you can only configure one disk group during the installation of Oracle Grid Infrastructure, the creation of further disk groups must be done manually. In our scenario, we want an FRA disk group that we can use for the fast recovery area for our database. The ASM Configuration Assistant (`asmca`) utility provides an intuitive GUI interface allowing you to easily create new or remove existing ASM disk groups.

Quiz

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

1. True
2. False

Answer: 2

Quiz Comments

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

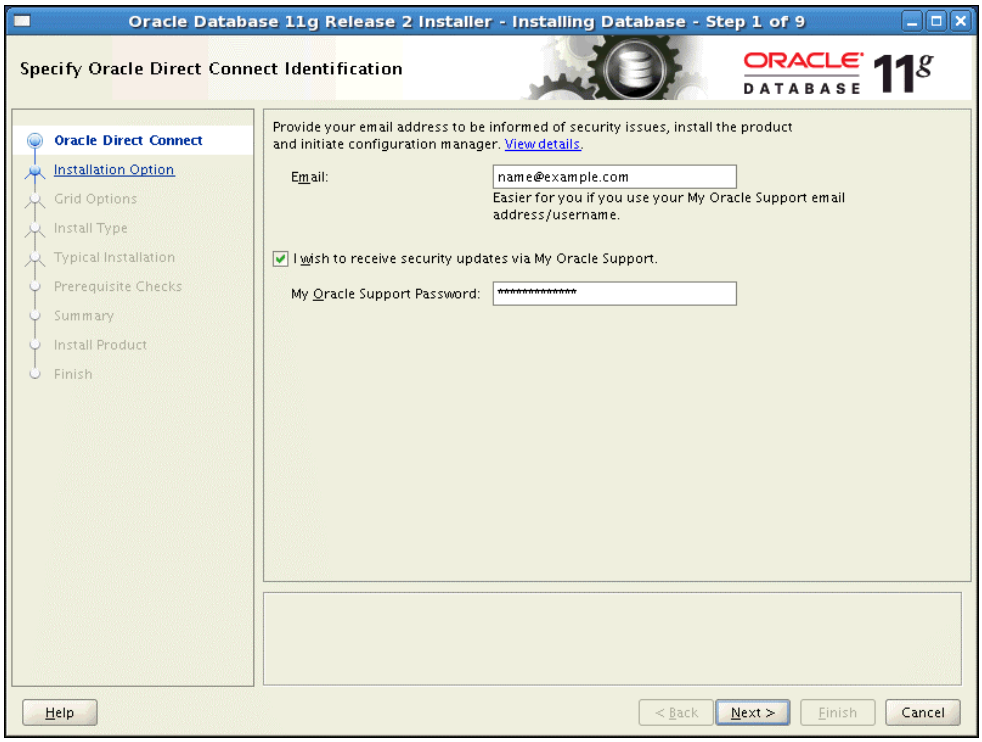
Quiz

During Grid infrastructure setup it is a possible to:

1. Specify exact location of datafiles for ASM instance
2. Create only one DISKGROUP
3. Specify size of SGA for ASM instance
4. Create several DISKGROUPS

Answer: 2

Part Two: Installing the Oracle Database Software

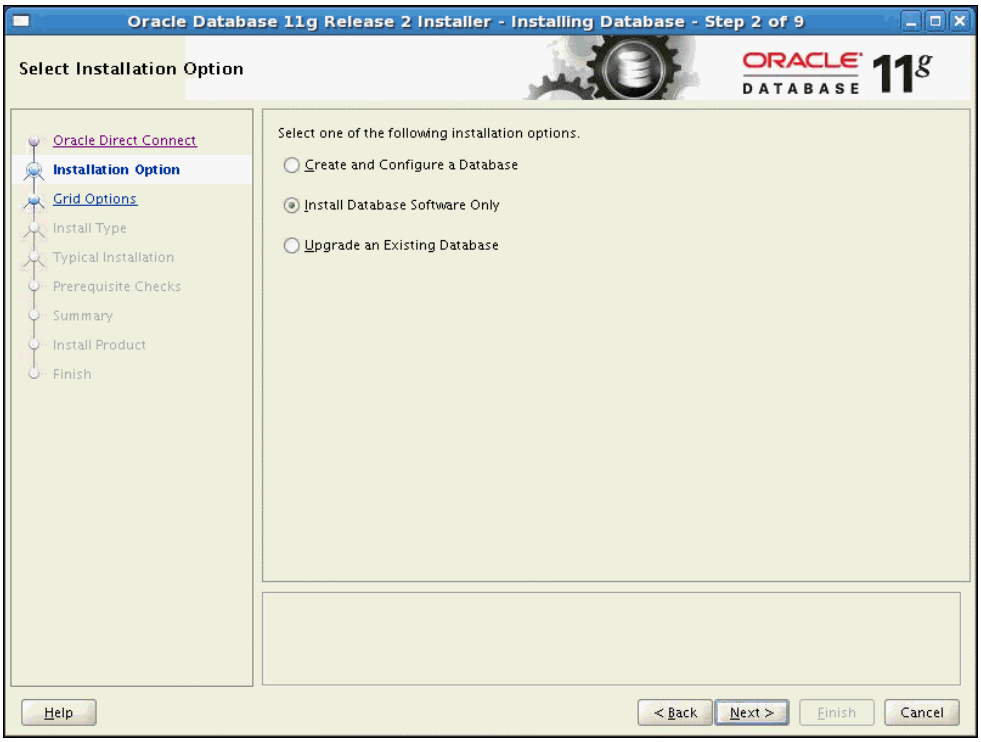


ORACLE

Part Two: Installing the Oracle Database Software

Now let's look at the installation steps for the Oracle Database software. As before, log on to your computer as a member of the administrative group that is authorized to install the Oracle software and to create and manage the database. Insert the distribution CD for the database into your CD drive, or navigate to the Oracle database staging location. From that location, enter `./runInstaller` to start the Oracle Universal Installer (OUI). If desired, enter the email address where you want to be informed of security issues. If you want to receive security updates via My Oracle Support then also include your My Oracle Support password. If you do not enter your email address, then a warning message is displayed asking if you are sure you want to remain uninformed of critical issues in your configuration. Click Yes in response to this warning to continue with the installation.

Choosing the Type of Installation



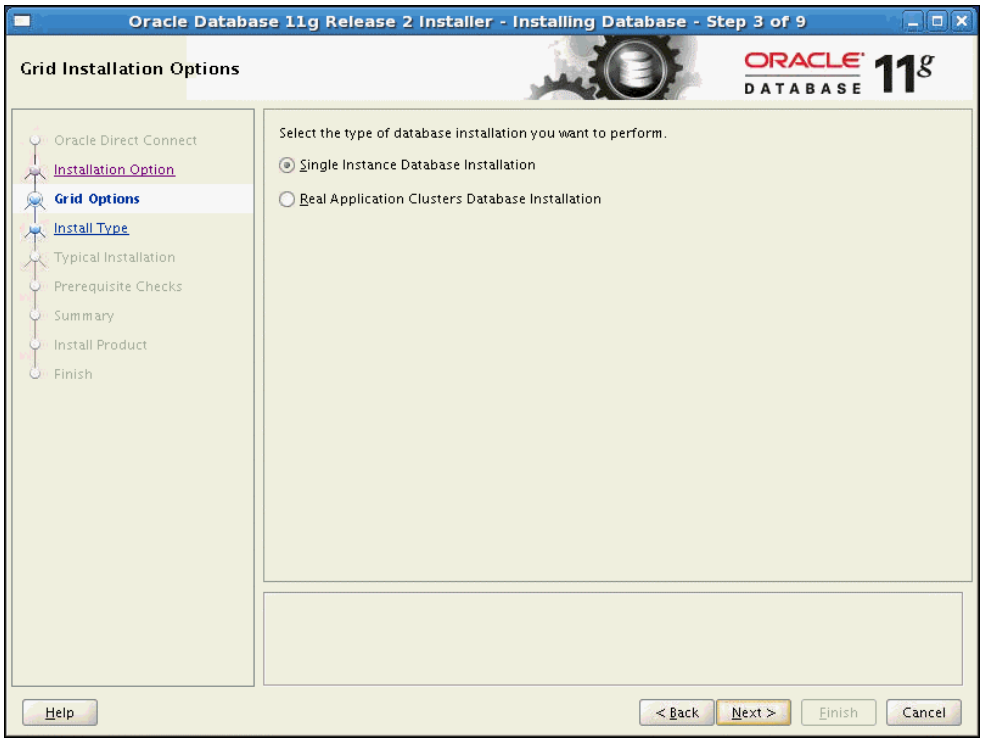
ORACLE

Choosing the Type of Installation

The Select Installation Option page is displayed Choose the type of installation you want to perform and click Next:

- **Create and Configure a Database:** This option creates a database after the product is installed.
- **Install Database Software Only:** This option only installs the Oracle Database binaries.
- **Upgrade an Existing Database:** This option is used to upgrade a database of an earlier release.

Choosing Grid Installation Options

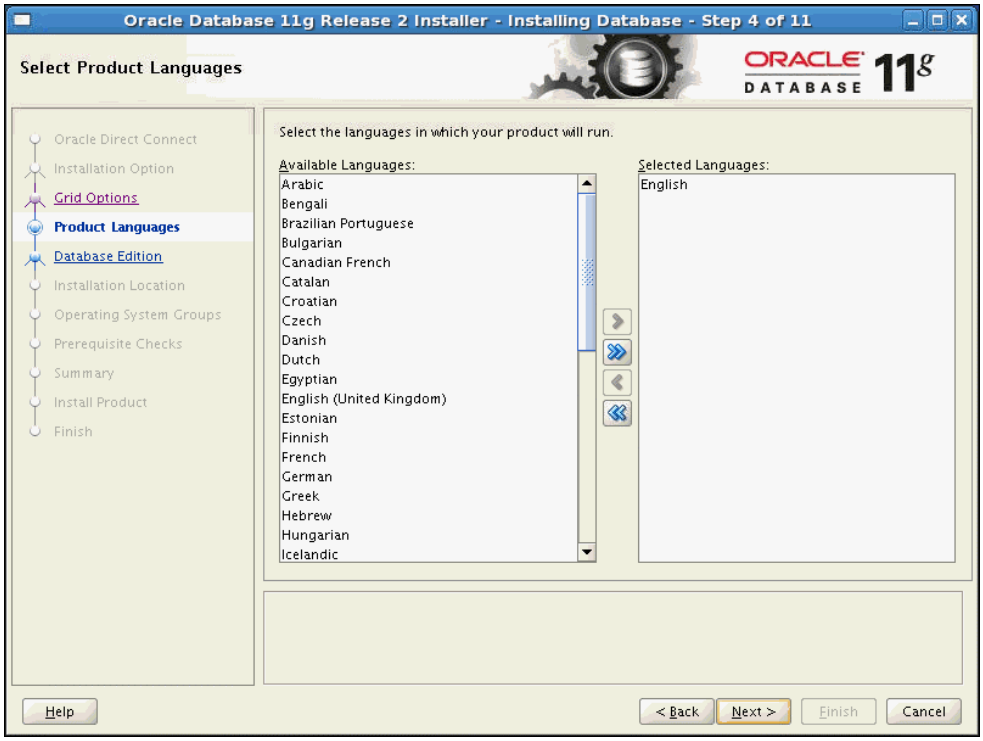


ORACLE

Choosing Grid Installation Options

The Grid Installation Option page is displayed Choose whether this is to be a single instance database installation or an Oracle RAC database installation on a cluster. Click Next to continue.

Choosing Language Settings

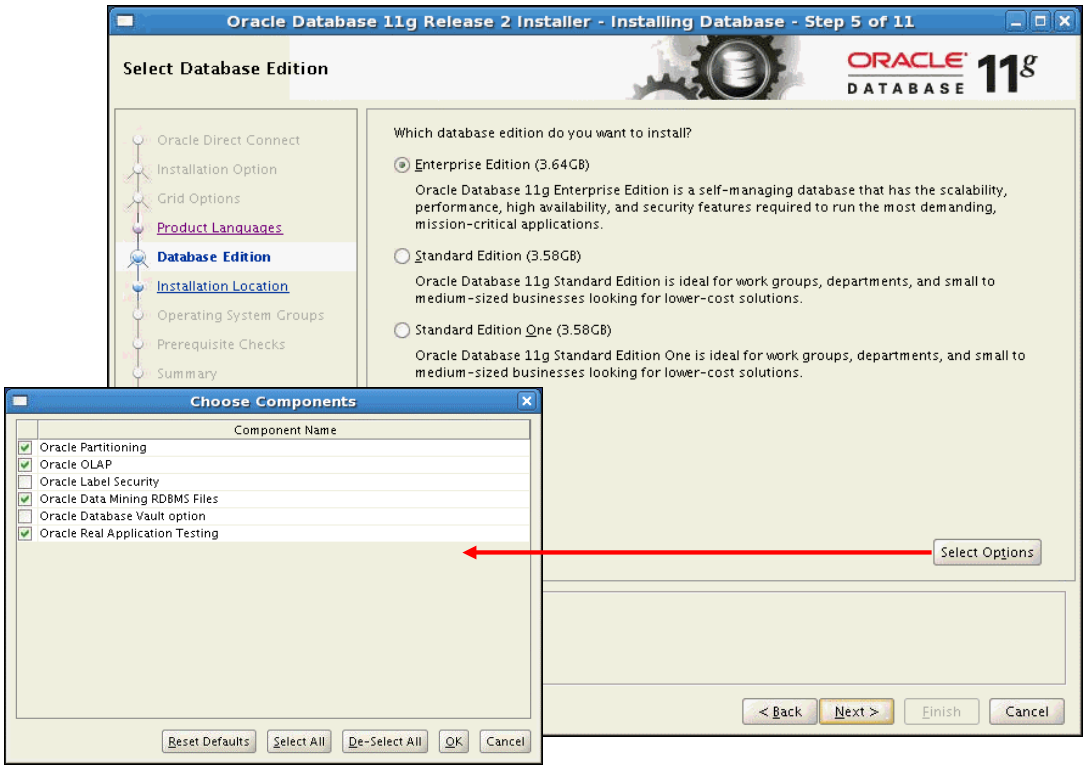


ORACLE

Choosing Language Settings

The Select Product Languages page is next. Here you select all the languages in which your product will run. English is selected by default and cannot be removed from the Selected Languages list. Click Next to continue.

Choosing the Database Edition



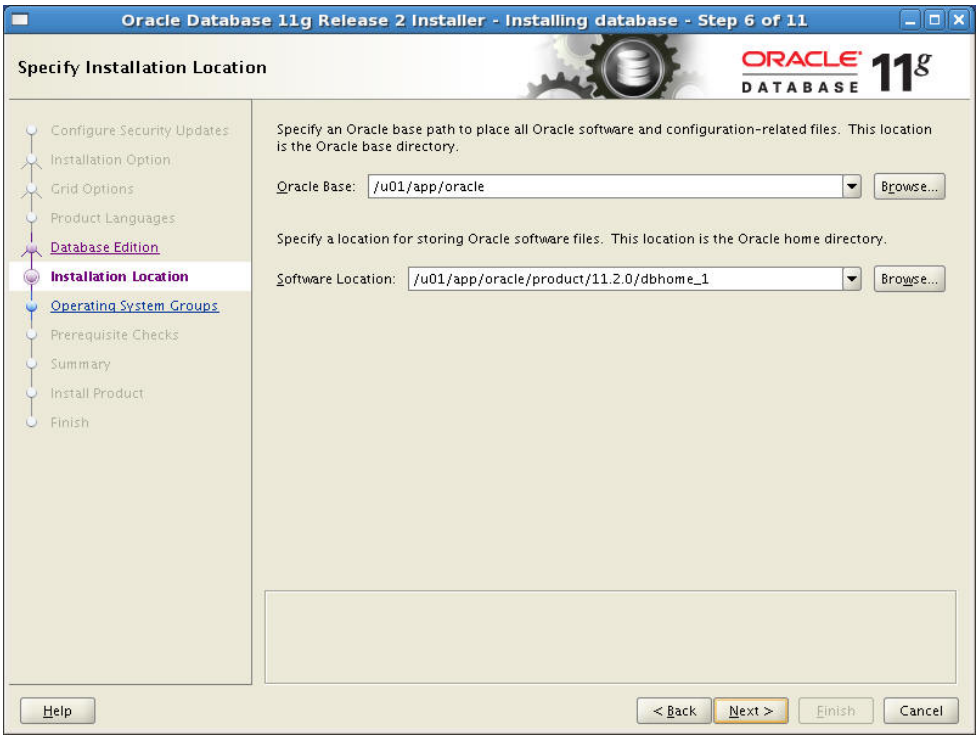
Choosing the Database Edition

The Select Database Edition page is displayed Choose from one of the following three editions:

- **Enterprise Edition:** This edition delivers a self-managing database that provides performance, scalability, security and reliability on a choice of clustered or single servers. It provides comprehensive features to easily manage the most demanding transaction processing, business intelligence, and content management applications.
- **Standard Edition:** This edition provides a full-featured database for servers with up to four sockets. It includes Oracle Real Application Clusters for higher availability, provides enterprise-class performance and security, is simple to manage, and can easily scale as demand increases. It is also upwardly compatible with Enterprise edition.
- **Standard Edition One:** This edition provides a full-featured database for servers with up to two sockets. It provides enterprise-class performance, security, and manageability that can easily scale as demand increases. It is also upwardly compatible with other database editions.

Click the Select Options button to further customize what components are installed. Click Next when finished with all selections on the Select Database Edition page.

Specifying Installation Location



ORACLE

Specifying Installation Location

The Installation Location page is displayed next. A suggested Oracle base path appears by default. You can change the path based on your requirement. In the Software Location section, you can accept the default values or enter the Oracle home name and directory path in which you want to install Oracle components. The directory path should not contain spaces. Click Next to continue the installation process.

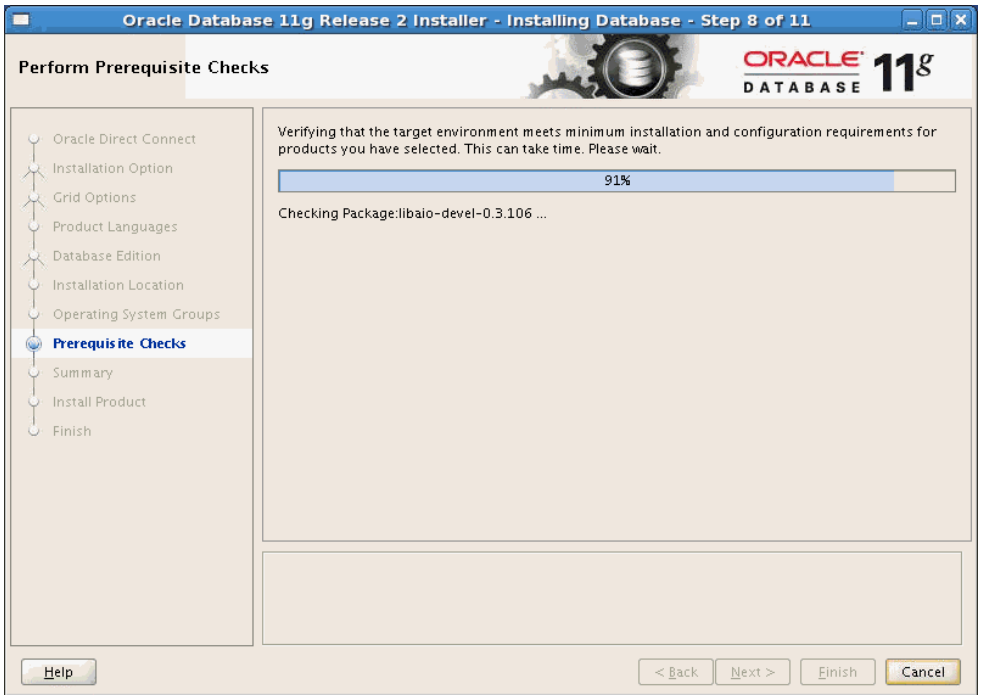
Choosing Operating System Groups

ORACLE

Choosing Operating System Groups

The Privileged Operating System Groups page is displayed. Choose the appropriate operating system group for the `OSDBA` and `OSOPER` privileges. By default, the `dba` is specified for `OSDBA` and `oper` is specified for `OSOPER`. In our example, we are using `dba` for both because we are doing a stand-alone installation that does not implement separation of duty. Click Next to continue.

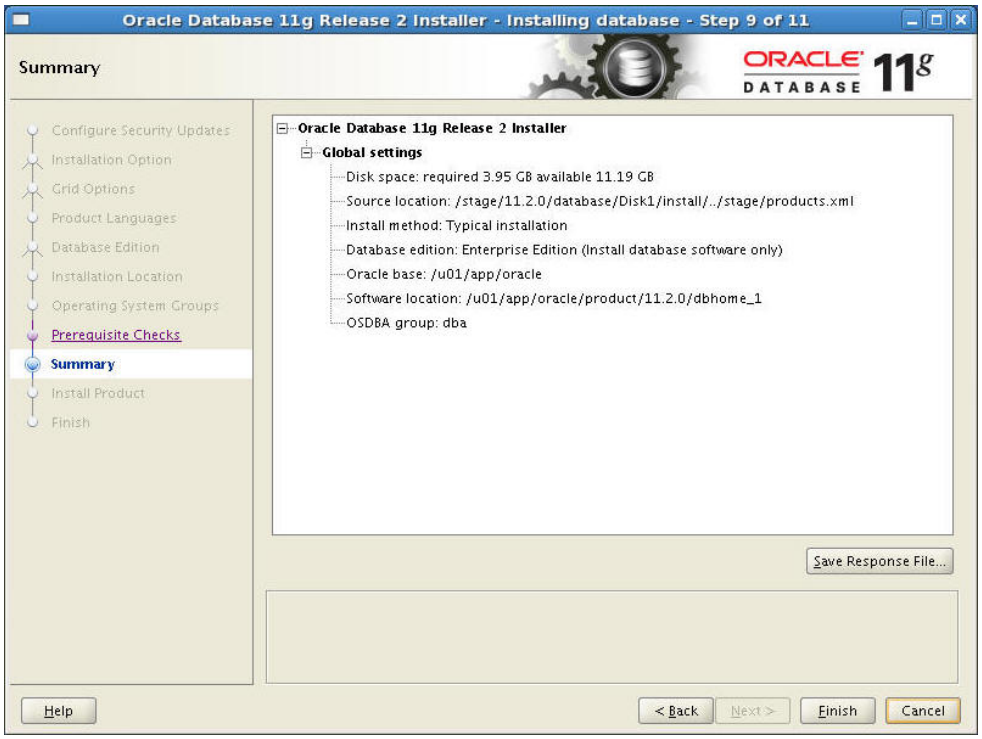
Performing Prerequisite Checks



Performing Prerequisite Checks

The Perform Prerequisite Checks page is displayed. The OUI goes through various prerequisite checks. Once 100% of the checks have been performed, the OUI returns back information if any of the tests fail. There is nothing to do on this page unless the checks fail with errors. If any prerequisite checks fail you are presented with a page where you can click Fix & Check Again at which point the OUI will generate a script to fix any issues it can. If the error is one that is not fixable by the OUI, then you must either fix the problem manually or choose to Ignore All and continue on with the installation (if the failed check is not critical).

Installation Summary Page

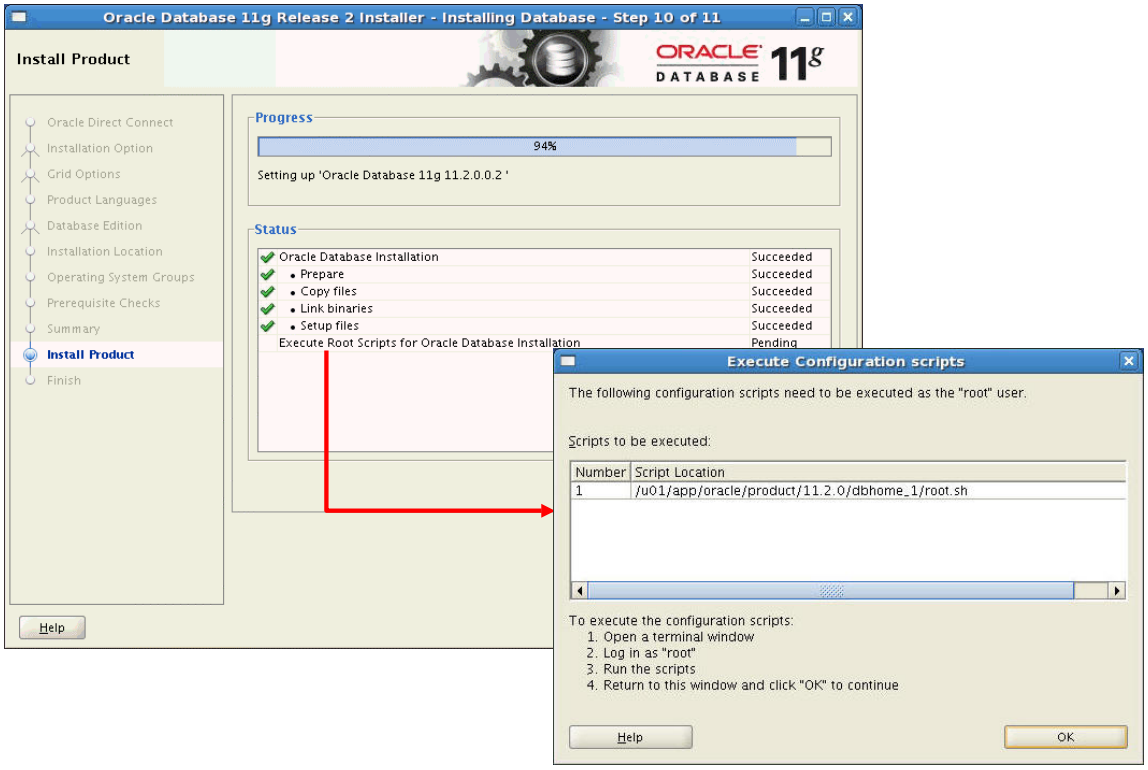


ORACLE

Installation Summary Page

The "Summary" page is displayed. Review the information presented that is related to the installation answers provided on previous pages. You are presented with the option to save the interactive installation questions and answers to a response file. A response file can be used in future installations to perform a silent installation in the same manner in which this installation is being performed. The response file is a text file and can also be edited with a text-editing tool to modify it for a different installation. Click the Finish button to proceed with the installation.

The Install Product Page



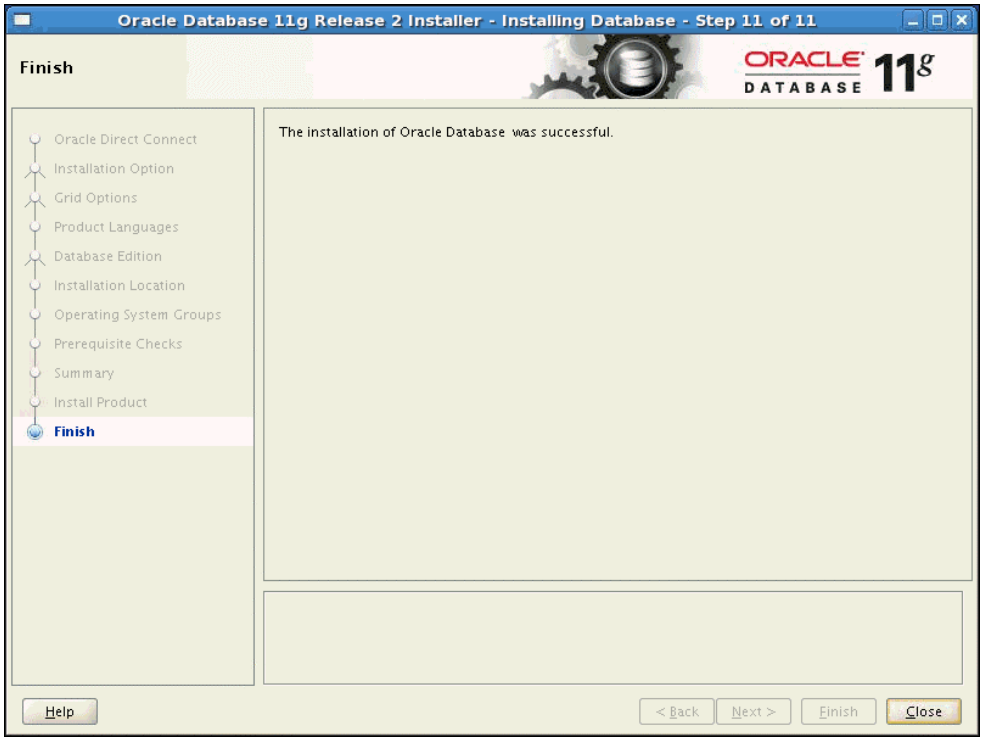
The Install Product Page

Your installation process pauses at this point, requiring you to execute an additional configuration script as the `root` user. In a separate terminal window, enter:

```
$ su
# password: oracle <root password, does not appear in the window>
# /u01/app/oracle/product/11.2.0/dbhome_1/root.sh
```

Accept the default for the local `bin` directory during a Linux or UNIX installation. When the scripts are finished, exit from the `root` account and close the window, and then click OK in the Execute Configuration scripts dialog box to allow the installation to complete.

Installation Finished

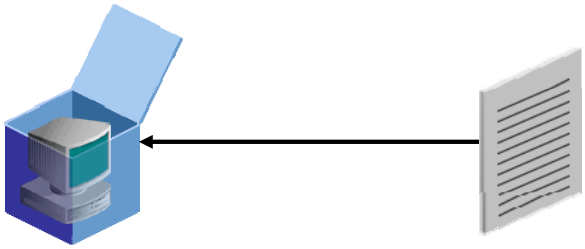


ORACLE

Installation Finished

After all the installation steps are completed, the “Finish” page is displayed. Click the Close button to exit the OUI utility.

Installation Option: Silent Mode



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

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

ORACLE

Installation Option: Silent Mode

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

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

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

Quiz

A response file is:

1. A binary file that can be edited using a binary editing program
2. A binary file that can be created by the installer program
3. A text file that cannot be edited, but can be created by the installer program
4. A text file that can be edited with a text editor

Answer: 4

Quiz

During Database software installation you can specify groups for:

1. the `osoper` group
2. the `osasm` group
3. the `osdba` group
4. the `osadmin` group

Answers: 1, 3

Summary

In this lesson, you should have learned how to:

- Describe your role as a database administrator (DBA) and explain typical tasks and tools
- Plan an Oracle software installation
- Install Oracle Grid Infrastructure for a standalone server
- Install the Oracle database software

ORACLE

Practice 2 Overview: Preparing the Database Environment

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

Note: Completing this practice is critical for all subsequent practices.

3

Creating an Oracle Database Using DBCA

Objectives

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

- Create a database by using the Database Configuration Assistant (DBCA)
- Generate database creation scripts with the DBCA
- Manage database design templates with the DBCA
- Perform additional tasks with the DBCA

ORACLE

Planning the Database

As a DBA, you must plan:

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

ORACLE

Planning the Database

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

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

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

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

Databases: Examples

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

ORACLE

Databases: Examples

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

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

The information on this page and the previous one present considerations that you will encounter as a DBA. This course (in its entirety) is designed to help you address them.

Choosing the Appropriate Character Set

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

ORACLE

Choosing the Appropriate Character Set

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

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

Choosing the Appropriate Character Set (continued)

Single-Byte Character Sets

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

Examples of Single-Byte Schemes

7-bit character set:

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

8-bit character set:

- International Organization for Standards (ISO) 8859-1 West European (`WE8ISO8859P1`)
- DEC 8-bit West European (`WE8DEC`)
- Extended Binary Coded Decimal Interchange Code (EBCDIC) Code Page 1144 8-bit Italian (`I8EBCDIC1144`)

Multibyte Character Sets

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

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

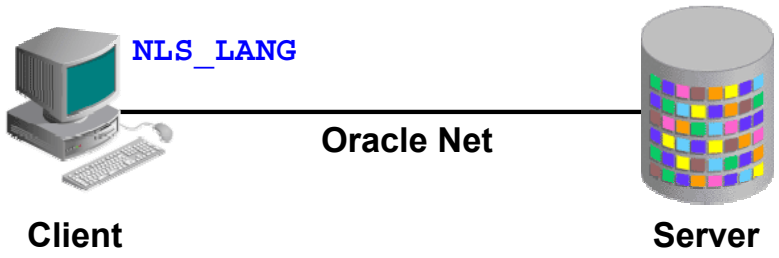
The Unicode standard has been adopted by many software and hardware vendors. Many operating systems and browsers now support Unicode. Unicode is required by standards such as XML, Java, JavaScript, LDAP, and WML. It is also synchronized with the ISO/IEC 10646 standard.

Examples of Varying-Width Multibyte Schemes

- Shift-JIS 16-bit Japanese (`JA16SJIS`)
- MS Windows Code Page 950 with Hong Kong Supplementary Character Set HKSCS-2001 (`ZHT16HKSCS`)
- Unicode 4.0 UTF-8 Universal character set (`AL32UTF8`) - a variable-width type of encoding and also a strict superset of ASCII.
- Unicode (AL16UTF16 – a 16-bit encoding of Unicode that is used by both Microsoft Windows 2000 and Windows XP.

How Are Character Sets Used?

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



ORACLE

How Are Character Sets Used?

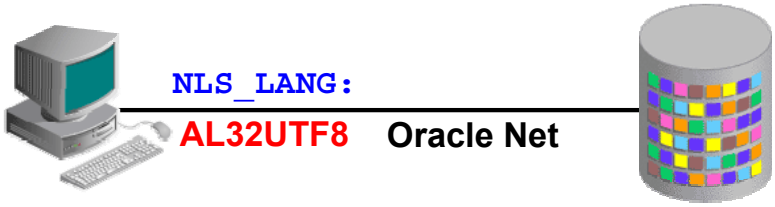
The NLS_LANG parameter defines a client terminal's character-encoding scheme. Different clients can use different encoding schemes. Data passed between the client and the server is converted automatically between the two encoding schemes. The database's encoding scheme should be a superset, or equivalent,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

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

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

Problems to Avoid

Example:



Client
Windows English
Code page: WE8MSWIN1252

Server
Database character set:
AL32UTF8

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

Issue: Invalid data are entered into the database.

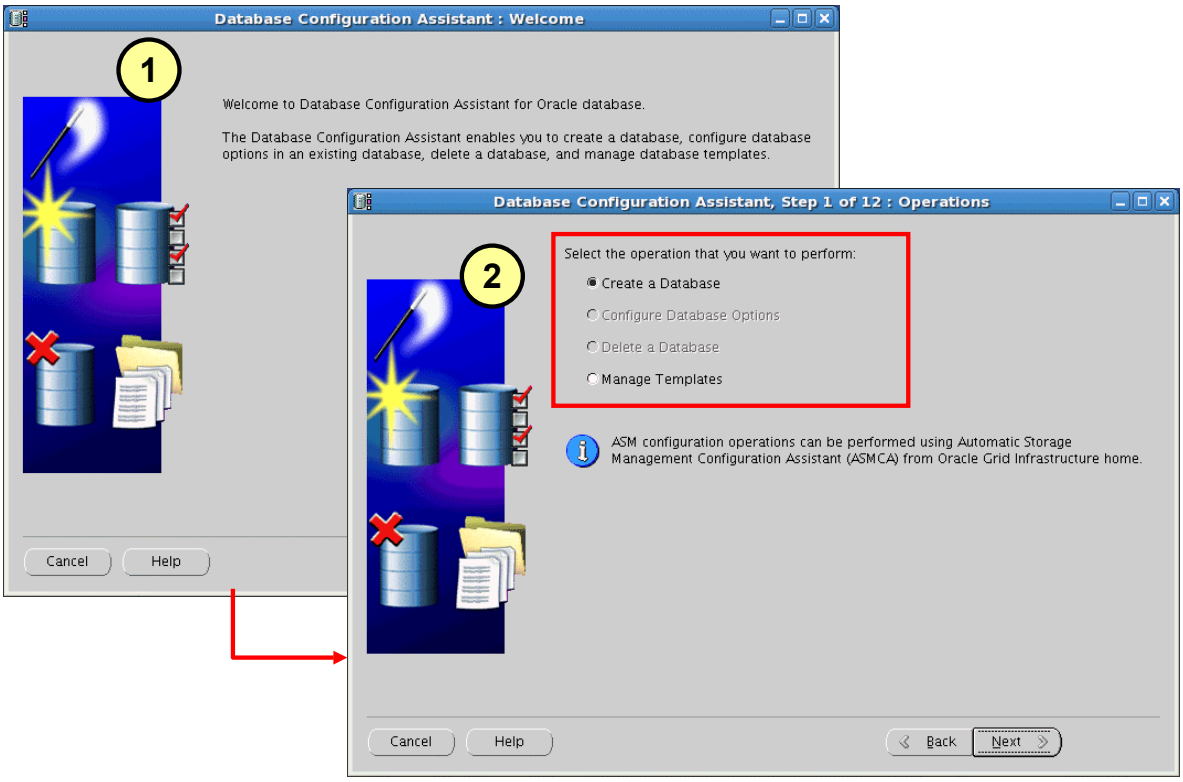
Problems to Avoid

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

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

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

Database Configuration Assistant (DBCA)



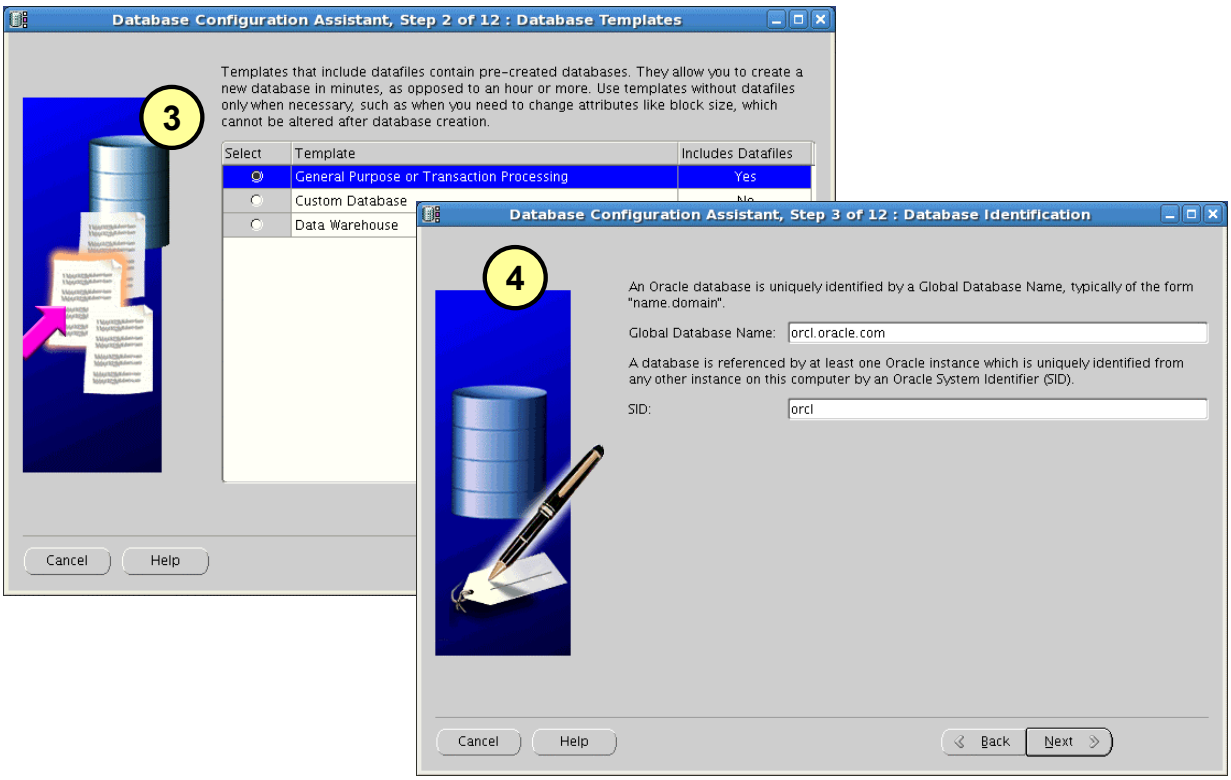
Database Configuration Assistant (DBCA)

The Database Configuration Assistant (DBCA) offers several choices to assist you with various operations. You can use the DBCA to create, change the configuration of, or delete a database. You can also create a database from a list of predefined templates or use an existing database as a sample to create a new database or template.

The DBCA provides several options to allow you to create a database to meet your needs. The DBCA provides a series of pages where you enter configuration information. On most pages, the DBCA provides default settings that you can accept if they apply. The steps involved in creating a database using the DBCA are as follows:

1. Log on to your computer as a member of the OS DBA group that is authorized to install the Oracle software. If required, set environment variables and enter `dbca` to invoke the DBCA. The main welcome page appears and you click Next to continue.
2. Choose the action you want to perform. In this case select Create a Database and click Next to begin that action.

Using the DBCA to Create a Database

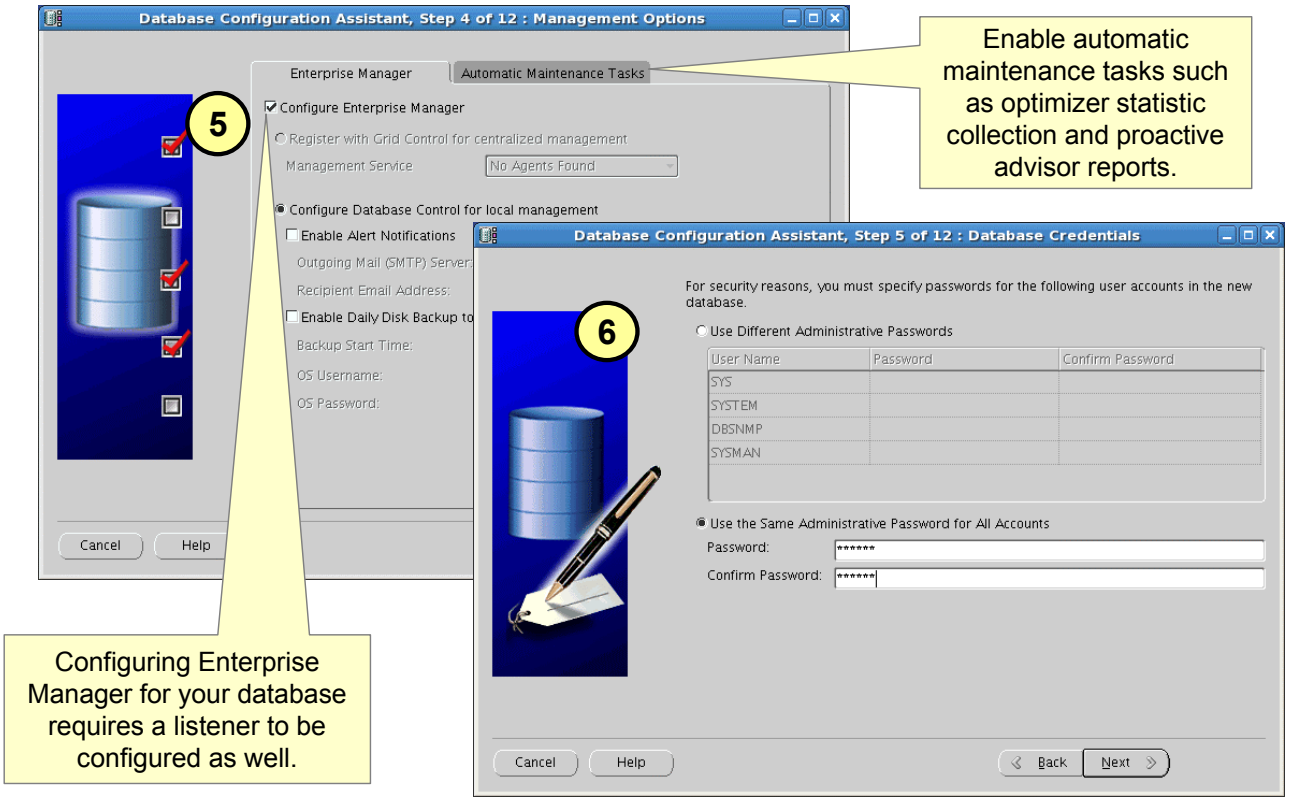


ORACLE

Using the DBCA to Create a Database

3. **Database Templates:** Select the type of database template to be used in creating the database. There are three database templates (General Purpose or Transaction Processing, Custom Database, and Data Warehouse) that copy a preconfigured database, including the data files. These files include control files, redo log files, and data files for various included tablespaces. Click Show Details to see the configuration for each type of database.
   For more complex environments, you may want to select the Custom Database option.
4. **Database Identification:** Enter the Global Database Name in the form `database_name.domain_name`, and the system identifier (SID). The SID defaults to the database name and uniquely identifies the instance associated with the database.

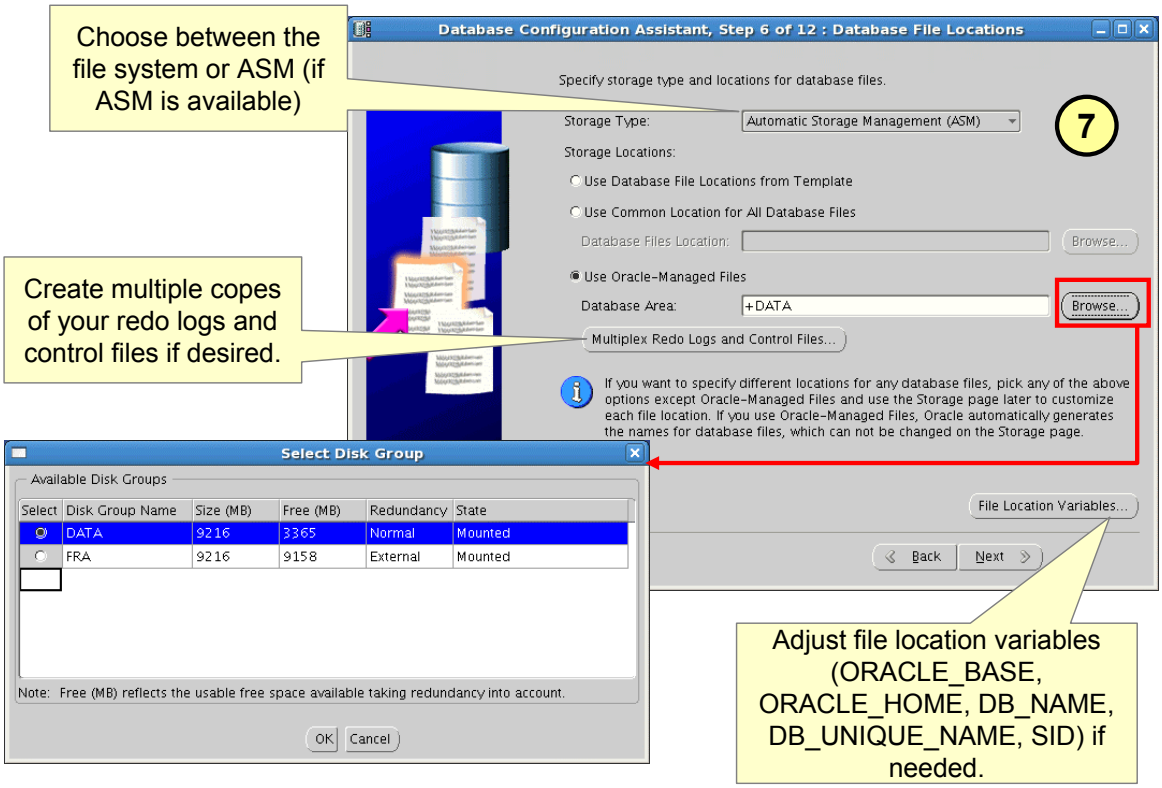
Using the DBCA to Create a Database



Using the DBCA to Create a Database (continued)

5. **Management Options:** Use this page to set up your database so that it can be managed with Oracle Enterprise Manager. Select the default: "Configure the Database with Enterprise Manager." Optionally, this page allows you to configure alert notifications and daily disk backup area settings. The Register with Grid Control option is grayed-out if DBCA cannot detect an Enterprise Manager Grid Control agent to communicate with. **Note:** Configuring Enterprise Manager for your database requires a listener to be configured as well (in our classroom the listener was configured during the installation of the Oracle Grid Infrastructure).
6. **Database Credentials:** Use this page to specify the passwords for the administrative accounts, such as `SYS` and `SYSTEM`. In class, use `oracle_4U` as the password for all administrative accounts.

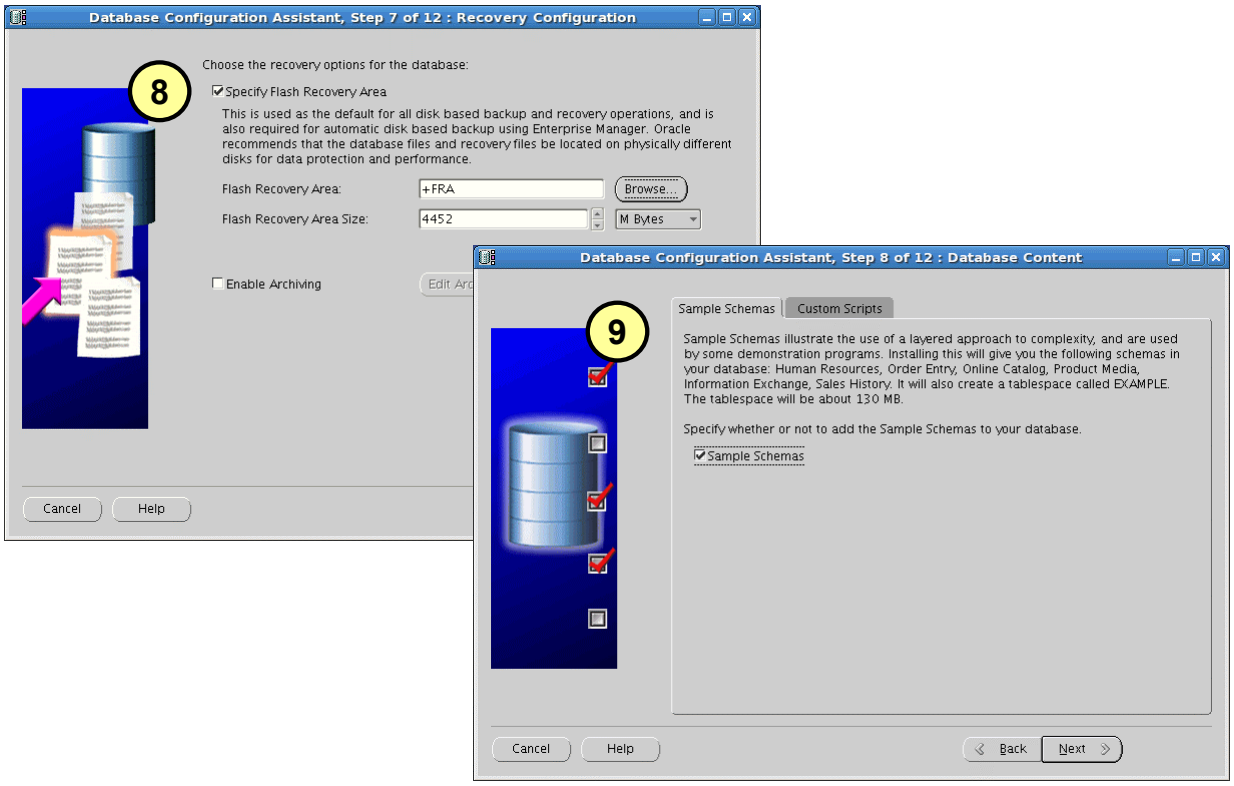
Using the DBCA to Create a Database



Using the DBCA to Create a Database (continued)

7. **Database File Locations:** Specify the type of storage mechanism (such as Automatic Storage Management (ASM) or File System) that you want your database to use. For Storage Location, choose according to your needs. In our example ASM is the storage mechanism so Oracle-Managed Files (OMF) is used. The Browse button allows you to view the available disk groups and choose the most appropriate disk group (`+DATA` is chosen in the example on the slide). You will be asked to provide the ASMSNMP password specific to ASM after you select the disk group. Oracle Managed Files (OMF) can be used with file system storage as well, eliminating the need for you to directly manage the operating system files in an Oracle database. If desired you can create additional copies of the redo logs and controls files by clicking the Multiplex Redo Logs and Control Files Button. A button is also provided to allow you to adjust the file location variables: `ORACLE_BASE`, `ORACLE_HOME`, `DB_NAME`, `DB_UNIQUE_NAME`, `SID`.

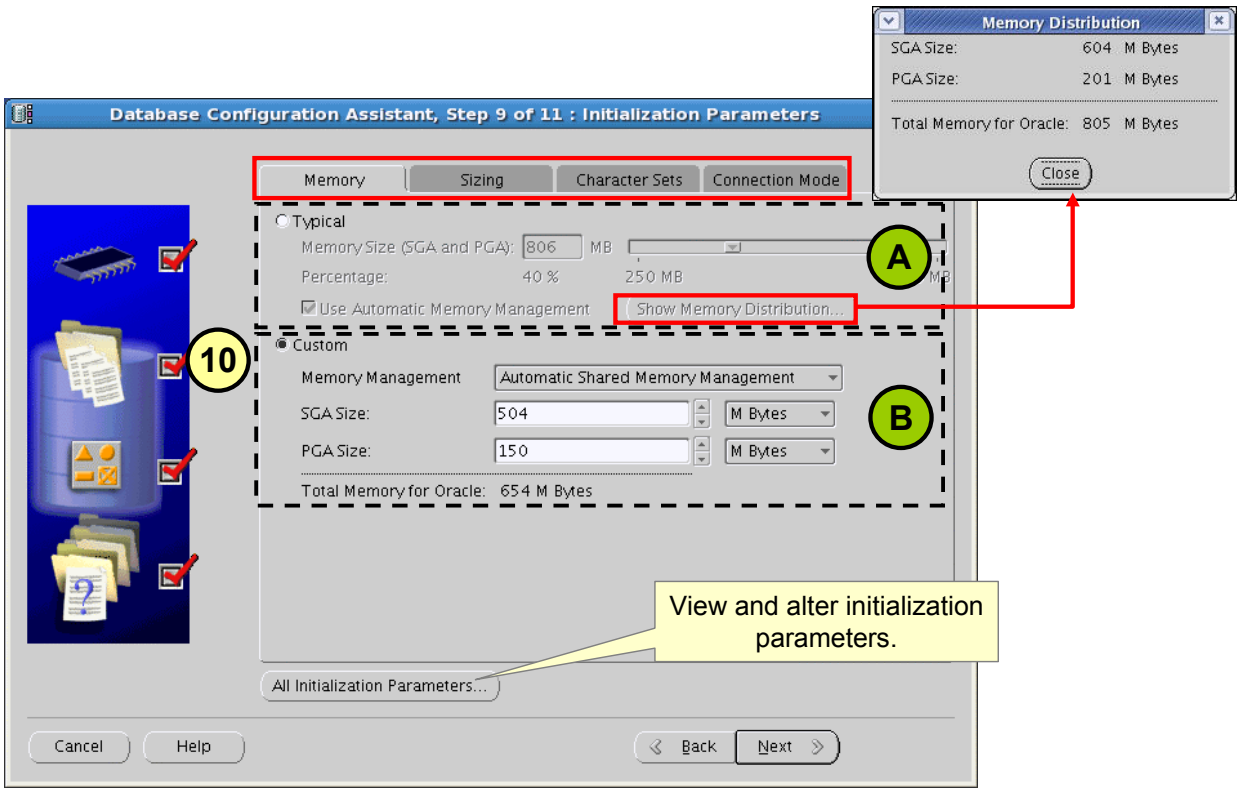
Using the DBCA to Create a Database



Using the DBCA to Create a Database (continued)

8. **Recovery Configuration:** Choose the recovery options for you database. To configure the fast recovery area check the Specify Flash Recovery Area option and then specify the location and size of the fast recovery area. In the example shown on the location of the fast recovery area is set to use the +FRA ASM disk group and the size is 4452 MB. It is recommended that the size of your fast recovery area be larger than the size of your database. Archiving and archiving related parameters can also be configured from this screen
   Note: The flash recovery area has been renamed to fast recovery area but you may still see references to flash recovery area within some Oracle products at this time.
9. **Database Content:** This pages provide options for selecting components (such as Sample Schemas) and a tab where you can specify any custom scripts that should be run after the database has been created.

Using the DBCA to Create a Database



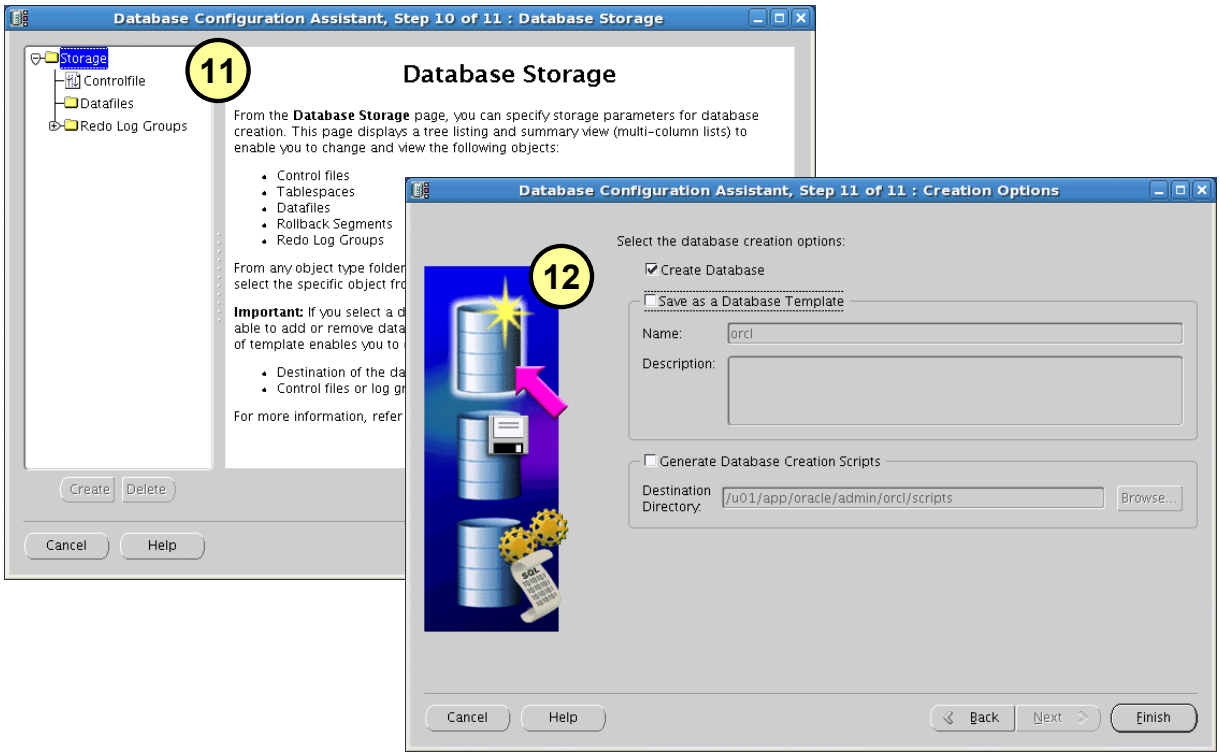
Using the DBCA to Create a Database (continued)

10. **Initialization Parameters:** The tabs on this page provide access to pages that enable you to change default database settings:
    - **Memory:** Use this page to set the initialization parameters that control memory usage. Use either (A) Typical or (B) Custom memory allocation.
    - **Sizing:** To specify block size, enter the size in bytes or accept the default.
    - **Character Sets:** Use this page to specify the character sets for your database.
    - **Best Practice Tip:** Oracle Corporation recommends (whenever possible) that you use Unicode for a database character set because it provides optimal flexibility for supporting Web technologies as well as many spoken languages.
    - **Connection Mode:** Select Dedicated or Shared Server Mode. For more details, see the lesson titled "Configuring the Oracle Network Environment."

- View and modify any initialization parameters by clicking the All Initialization Parameters button.

Note: Several initialization parameters are set for the lifetime of a database, such as the `DB_BLOCK_SIZE` parameter.

Using the DBCA to Create a Database

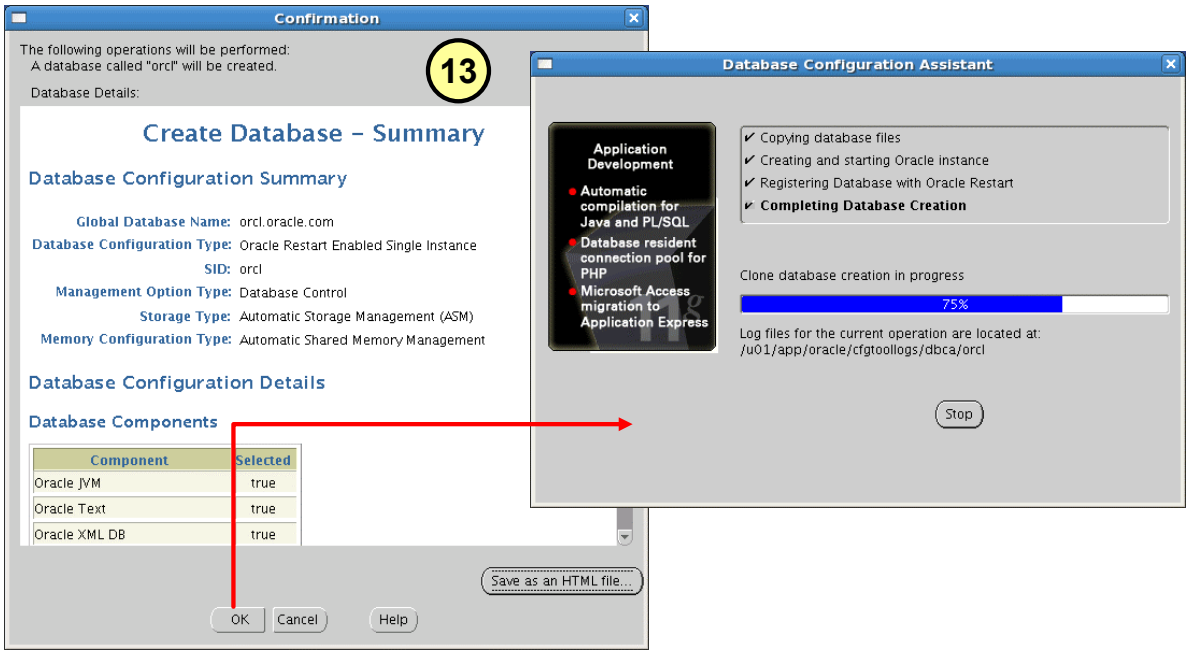


ORACLE

Using the DBCA to Create a Database (continued)

11. **Database Storage:** Review and modify, if necessary, the current database storage settings. If you selected one of the preconfigured templates for a database, you cannot add or remove control files or data files.
12. **Create Options:** You have the following options: create your database at this time, save the database definition as a template, and generate scripts. If you choose all options and click Finish, the DBCA first saves the database template, then generates the scripts into your destination directory, and finally creates your database.

Create Database Summary



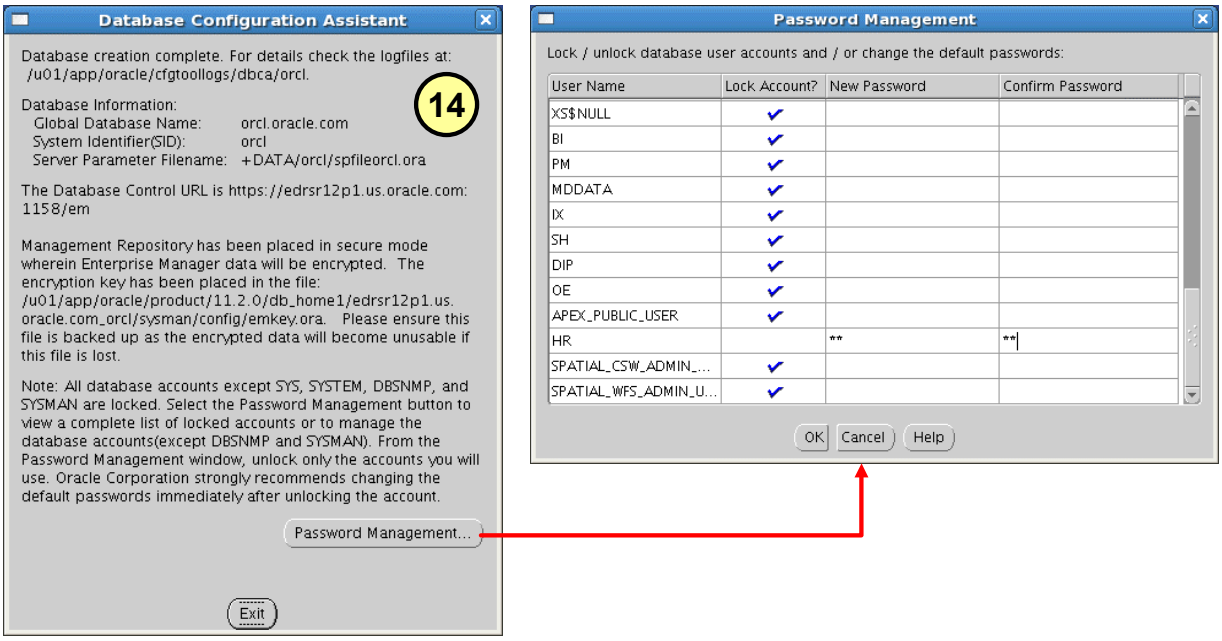
ORACLE

Create Database Summary

13. A Confirmation page appears after you click Finish allowing you to review all the configuration of the database prior to creation. Click OK to dismiss the Confirmation page and begin creation of the database.

    Note: You may want to save your database definition as an HTML file for easy reference.

Password Management



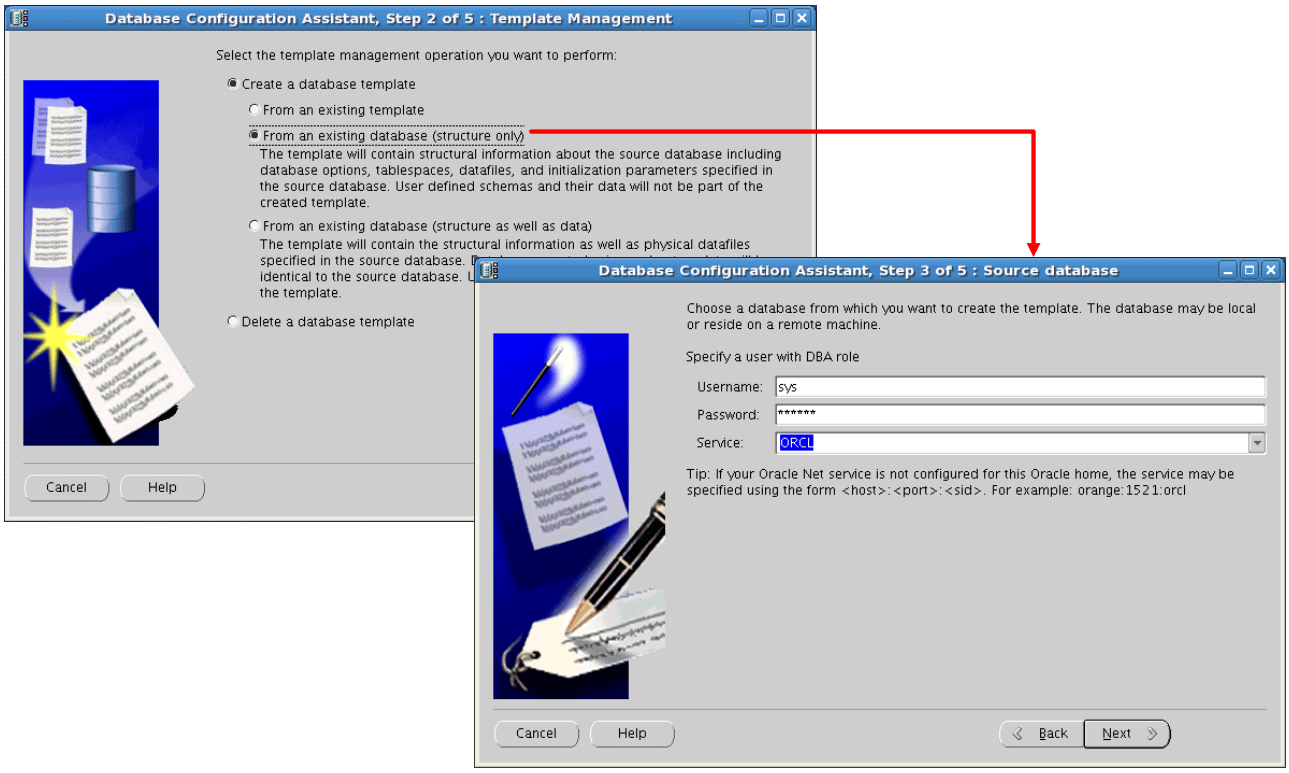
ORACLE

Password Management

14. After the DBCA finishes, note the following information for future reference:
    - Location of installation log files
    - Global database name
    - System identifier (SID)
    - Server parameter file name and location
    - Enterprise Manager URL

    Click Password Management to unlock database accounts that you plan to use. Provide a password when you unlock an account. Any accounts not unlocked at this time can be unlocked later as required.

Creating a Database Design Template



ORACLE

Creating a Database Design Template

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

There are three ways to create a template:

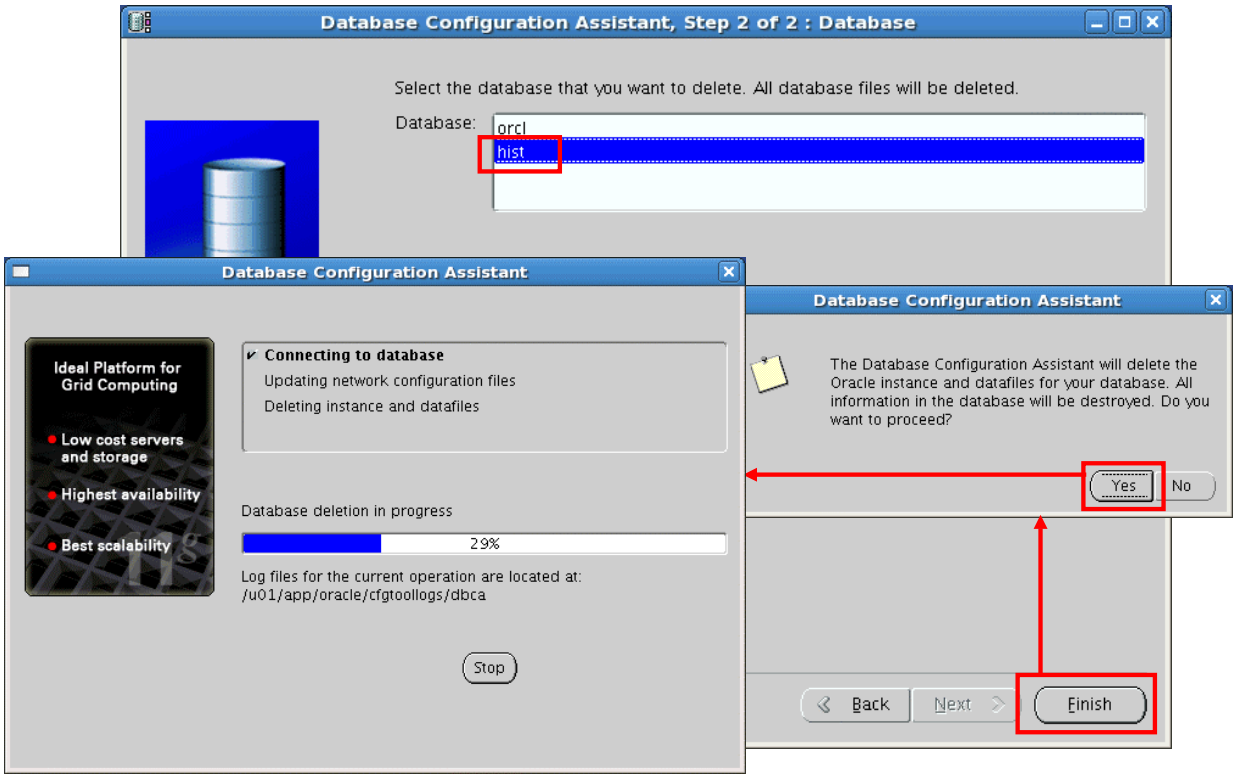
- From an existing template
- From an existing database (structure only)
- From an existing database (structure as well as data)

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

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

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

Using the DBCA to Delete a Database



Using the DBCA to Delete a Database

Start the DBCA by entering `dbca` in a terminal window, and click Next on the Welcome page. To delete the database, perform the following steps:

1. On the Operations page, select Delete a Database. Then click Next.
2. Select the database that you want to delete (in this example, `hist`), and click Finish.
3. Click Yes to confirm your deletion.
4. When the deletion is completed, you will be asked whether you want to perform another operation. Answer accordingly.

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

Using the DBCA to Delete a Database (continued)

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

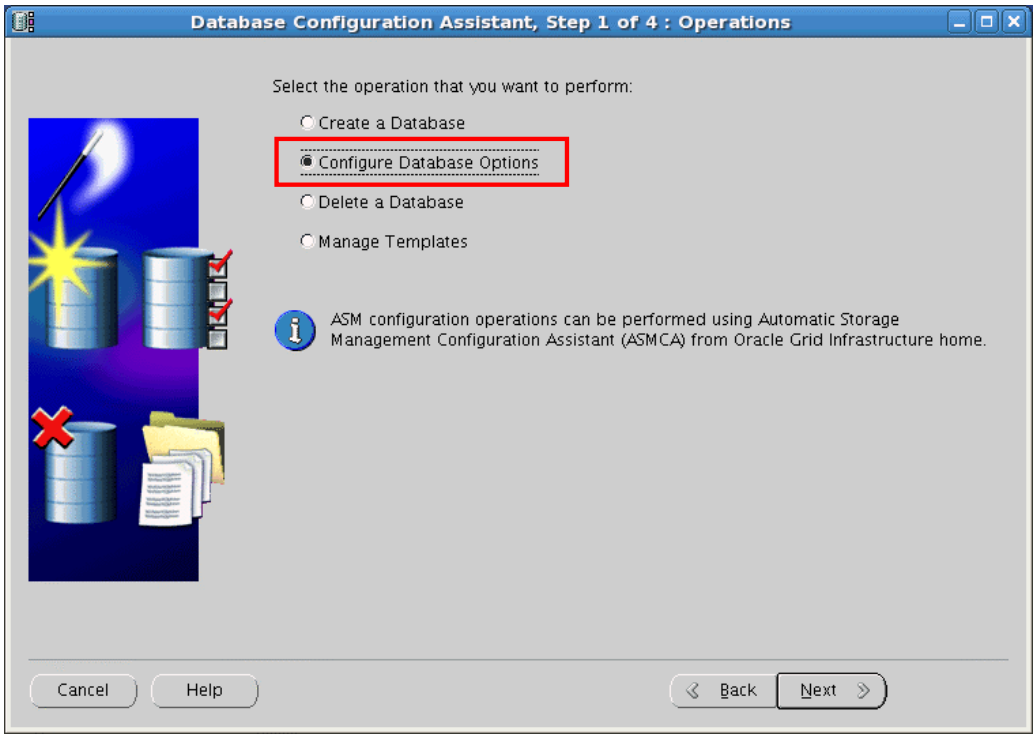
- The database must be mounted and closed.
- The database must be mounted exclusively (not in shared mode).
- The database must have been started in `RESTRICT` mode.

An example of these statements are:

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

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

Using the DBCA for Additional Tasks



ORACLE

Using the DBCA for Additional Tasks

You can use the DBCA to configure Database Options (such as Oracle Label Security or Oracle Database Vault). Not all options are installed by default during database software installation so these options will have to be installed prior to attempting to configure a database to use the options.

Note

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

Quiz

The parameter `DB_BLOCK_SIZE` is set for the lifetime of a database and cannot be changed.

1. True
2. False

ORACLE

Answer: 1

Quiz

In order to drop a database using the `DROP DATABASE` command, the database must be:

1. Open and in `RESTRICT` mode
2. Mounted exclusively in `RESTRICT` mode
3. Shut down with the immediate option

Answer: 2

Summary

In this lesson, you should have learned how to:

- Create a database by using the Database Configuration Assistant (DBCA)
- Generate database creation scripts with the DBCA
- Manage database design templates with the DBCA
- Perform additional tasks with the DBCA

ORACLE

Practice 3 Overview: Using the DBCA

This practice covers the following topics:

- Creating the ORCL database by using the DBCA
- Unlocking the HR schema

Note: Completing database creation and unlocking the HR schema are critical for all following practices.

- Creating the ORCL database design template by using the DBCA
- Creating database creation scripts by using the DBCA

ORACLE

4

Managing the Database Instance

Objectives

After completing this lesson, you should be able to:

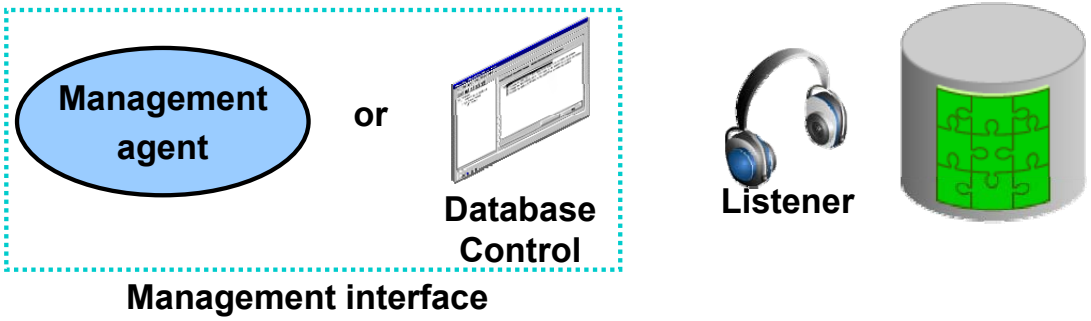
- Start and stop the Oracle database and components
- Use Oracle Enterprise Manager
- Access a database with SQL*Plus
- Modify database initialization parameters
- Describe the stages of database startup
- Describe database shutdown options
- View the alert log
- Access dynamic performance views

ORACLE

Management Framework

Oracle Database 11*g* Release 2 management framework components:

- Database instance
- Listener
- Management interface:
  - Database Control
  - Management agent (when using Grid Control)



ORACLE

Management Framework

There are three major components of the Oracle database management framework:

- The database instance that is being managed
- A listener that allows connections to the database
- The management interface. This may be either a management agent running on the node where the database server runs (which connects it to Oracle Enterprise Manager Grid Control) or the stand-alone Oracle Enterprise Manager Database Control. This is also referred to as the *Database Console*.

Each of these components must be started before you can use the services of the component and must be shut down cleanly when shutting down the server hosting the Oracle database.

Starting and Stopping Database Control

```
$ . oraenv
ORACLE_SID = [orcl] ? orcl
The Oracle base for ORACLE_HOME=/u01/app/oracle/product/11.2.0/db_home1
is /u01/app/oracle
$ emctl start dbconsole
Oracle Enterprise Manager 11g Database Control Release 11.2.0.1.0 Copy-
right (c) 1996, 2009 Oracle Corporation.  All rights reserved. http://
host01.example.com:1158/em/console/aboutApplication
Starting Oracle Enterprise Manager 11g Database Control . ......started.
------------------------------------------------------------------
Logs are generated in directory /u01/app/oracle/product/11.2.0/db_home1/
host01.example.com_orcl/sysman/ log
```

```
$ emctl stop dbconsole
Oracle Enterprise Manager 11g Database Control Release 11.2.0.1.0
Copyright (c) 1996, 2009 Oracle Corporation.  All rights reserved.
https://host01.example.com:1158/em/console/aboutApplication
Stopping Oracle Enterprise Manager 11g Database Control ...
 ...  Stopped.
```

ORACLE

Starting and Stopping Database Control

Oracle Database provides *Database Control*, which is a stand-alone management console for databases that are not connected to the Grid Control framework. Each database that is managed with Database Control has a separate Database Control installation; from any one Database Control, you can manage only one database. Before using Database Control, ensure that a `dbconsole` process is started.

Command to start the `dbconsole` process:

```
emctl start dbconsole
```

Command to stop the `dbconsole` process:

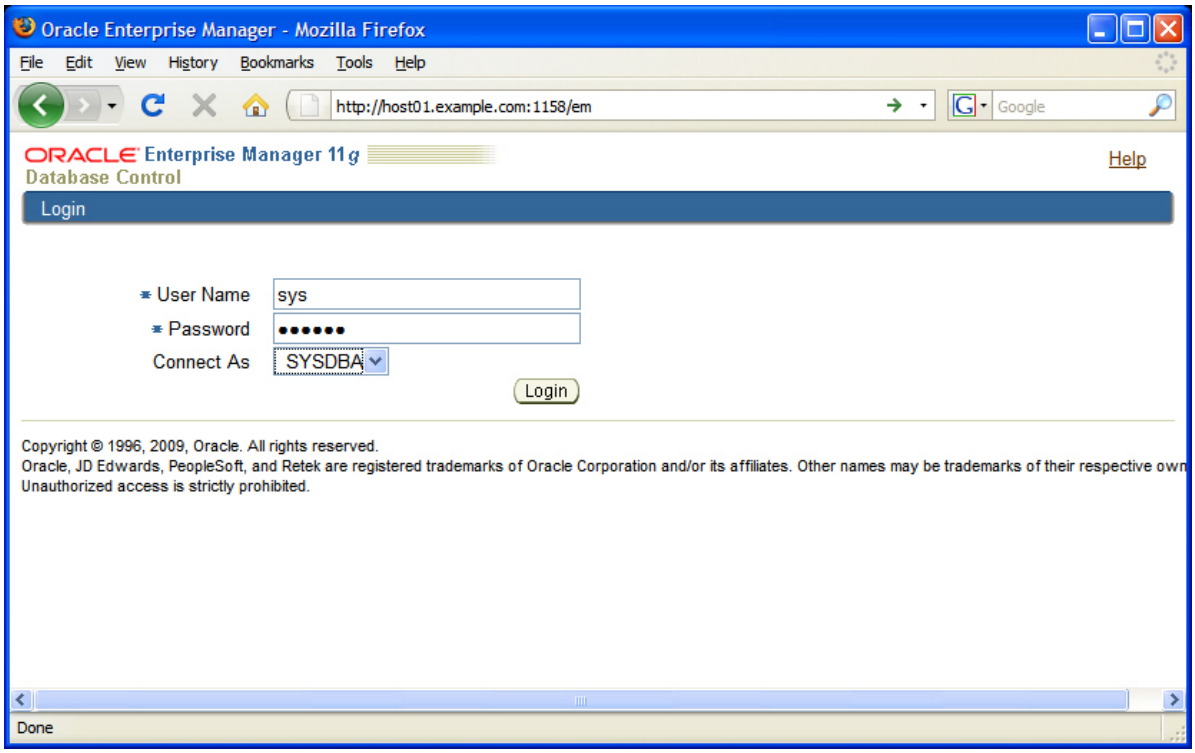
```
emctl stop dbconsole
```

Command to view the status of the `dbconsole` process:

```
emctl status dbconsole
```

Note: You may need to navigate to your `$ORACLE_HOME/bin` directory if this directory is not in your operating system (OS) path. There are two `$ORACLE_HOME` locations if Grid Infrastructure is installed and both contain the `emctl` utility. The `emctl` utility should always be invoked using the Oracle database `$ORACLE_HOME` and not the Grid Infrastructure `$ORACLE_HOME`. Database Control uses a server-side agent process. This agent process automatically starts and stops when the `dbconsole` process is started or stopped.

Oracle Enterprise Manager



ORACLE

Oracle Enterprise Manager

When you install the Oracle Database software, the Oracle Universal Installer (OUI) also installs Oracle Enterprise Manager (Enterprise Manager). Its Web-based Database Control serves as the primary tool for managing your Oracle database. Enterprise Manager provides a graphical interface for doing almost any task that you need to do as a database administrator (DBA). Viewing alert summaries and performance graphs, creating and modifying objects, and performing backup and recovery are some of the things that you can do with Enterprise Manager. In most cases, you can click links in Enterprise Manager to find more specific information about the contents of a page.

Note: In Oracle Database 11*g* Release 2, the URL to access Enterprise Manager uses HTTPS (instead of HTTP) as the protocol to enable a secure connection. To reach the Enterprise Manager `dbconsole`, you must therefore enter a URL in the following format:

```
https://machine_name:port/em
```

For the first database that you create on a machine, the default port number for accessing Enterprise Manager Database Control is 1158. It is possible to have different numbers, especially if there are multiple databases on the same host. To determine the port number, check the `portlist.ini` file. Ports for some Oracle Database applications are listed in the `portlist.ini` file, which is located in the `$ORACLE_HOME/install` directory.

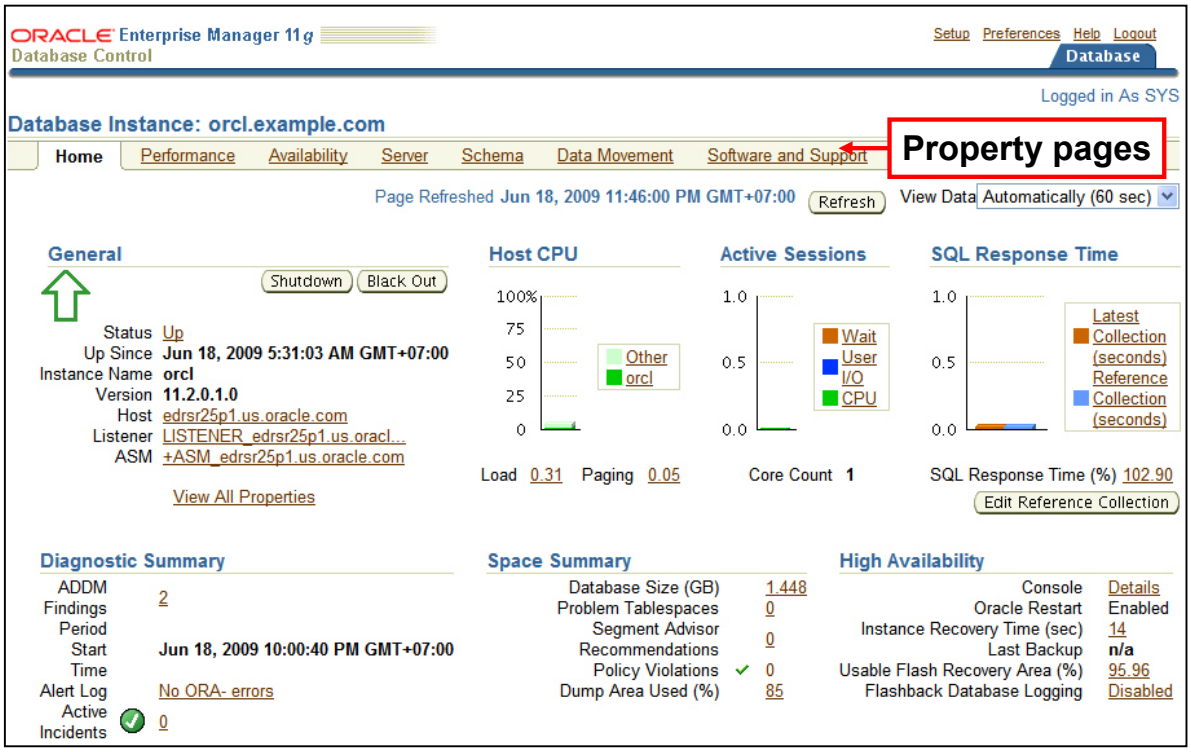
Oracle Enterprise Manager (continued)

When you enter the URL for Enterprise Manager, the content that is displayed depends on the status of the database:

- If the database is *up*, Enterprise Manager displays the Database Control Login page. Log in to the database by using a username that is authorized to access Database Control. Initially, this is `SYS`, `SYSMAN`, or `SYSTEM`. Use the password that you specified for the account during database installation. For the Connect As option, select either `Normal` or `SYSDBA` to log in to the database with special database administration privileges.
- If the database is *down***,** Enterprise Manager displays the "Startup/Shutdown and Perform Recovery" page. If this is the case, click the Startup/Shutdown button. You are then prompted for the host and target database login usernames and passwords, which you must enter.

Note: If you have trouble starting Enterprise Manager, ensure that a listener is started.

Database Home Page



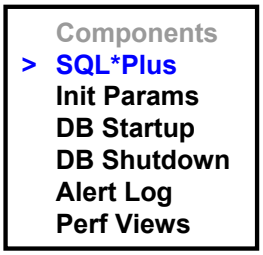
Database Home Page

The Database Home page displays the current state of the database by displaying a series of metrics that portray the overall health of the database. With the property pages (also referred to as *tabs*), you can access the Performance, Availability, Server, Schema, Data Movement, and Software and Support pages for managing your database.

You can view the following performance and status information about your database instance on the Database Home page:

- Instance name, database version, Oracle home location, media-recovery options, and other pertinent instance data
- Current instance availability
- Outstanding alerts
- Session-related and SQL-related performance information
- Key space-usage metrics
- Drill-down links (for example, `LISTENER_<host_name>`) to provide increasing levels of detail

Other Oracle Tools



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

ORACLE

Other Oracle Tools

In addition to Enterprise Manager, you can use SQL*Plus and SQL Developer to issue SQL statements. These tools enable you to perform many of the database management operations as well as to select, insert, update, or delete data in the database.

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

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

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

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

Note: This course uses Enterprise Manager and SQL*Plus.

Using SQL*Plus

SQL*Plus is:

- A command-line tool
- Used interactively or in batch mode

```
$ sqlplus hr

SQL*Plus: Release 11.2.0.1.0 - Production on Thu Jun 18 05:04:49 2009
Copyright (c) 1982, 2009, Oracle.  All rights reserved.
Enter Password: *******

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - Production
With the Partitioning, Automatic Storage Management, OLAP, Data Mining
and Real Application Testing options

SQL> select last_name from employees;
LAST_NAME
-------------------------
Abel
Ande
…
```

ORACLE

Using SQL*Plus

You can use the command-line interface of SQL*Plus to execute SQL*Plus, SQL, and PL/SQL commands to:

- Enter, edit, run, store, retrieve, and save SQL commands and PL/SQL blocks
- Format, calculate, store, and print query results
- List column definitions for any table
- Send messages to and accept responses from an end user
- Perform database administration

To start SQL*Plus:

1. Open a terminal window.
2. At the command-line prompt, enter the SQL*Plus command in the following form:

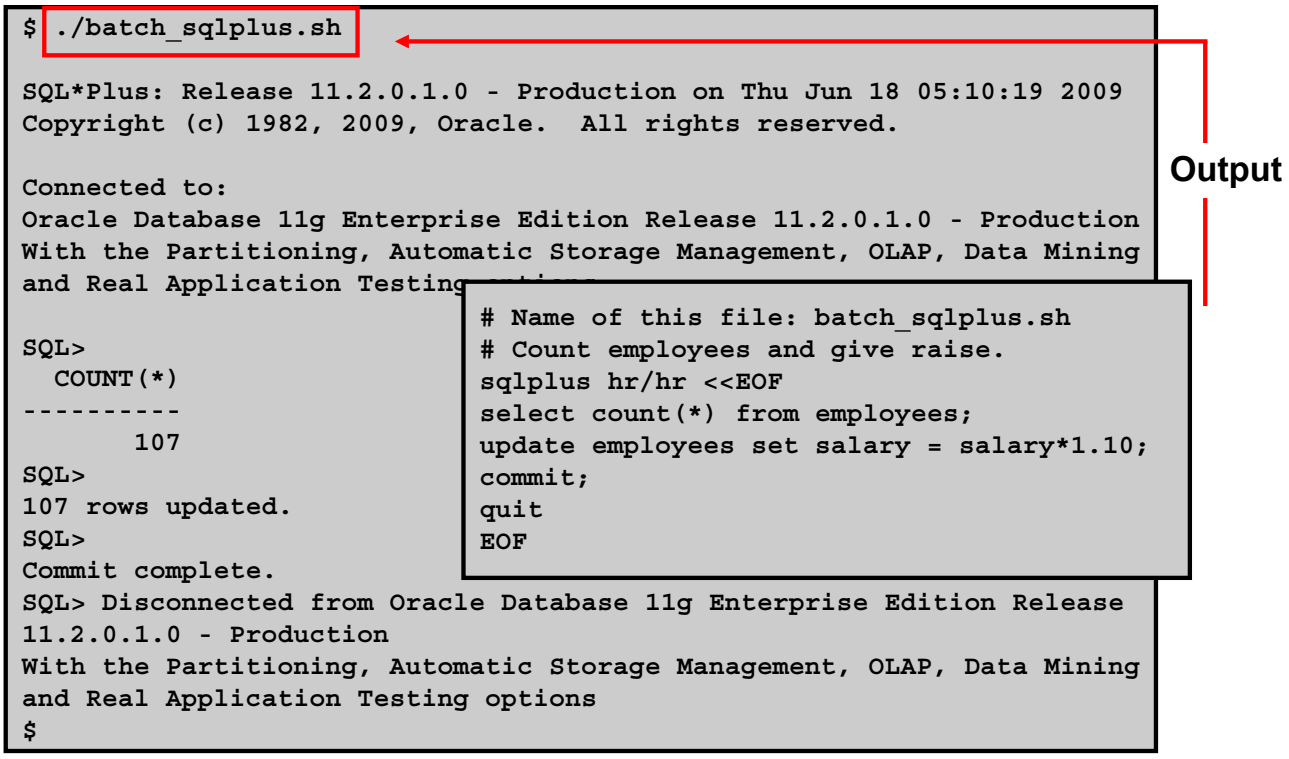
   ```
   $ sqlplus <userid>/<pwd> or /nolog
   ```

3. If you use the NOLOG option, you must enter CONNECT followed by the username you want to connect as.

   ```
   SQL> connect <username>
   ```

4. When prompted, enter the user's password. SQL*Plus starts and connects to the default database.

Calling SQL*Plus from a Shell Script

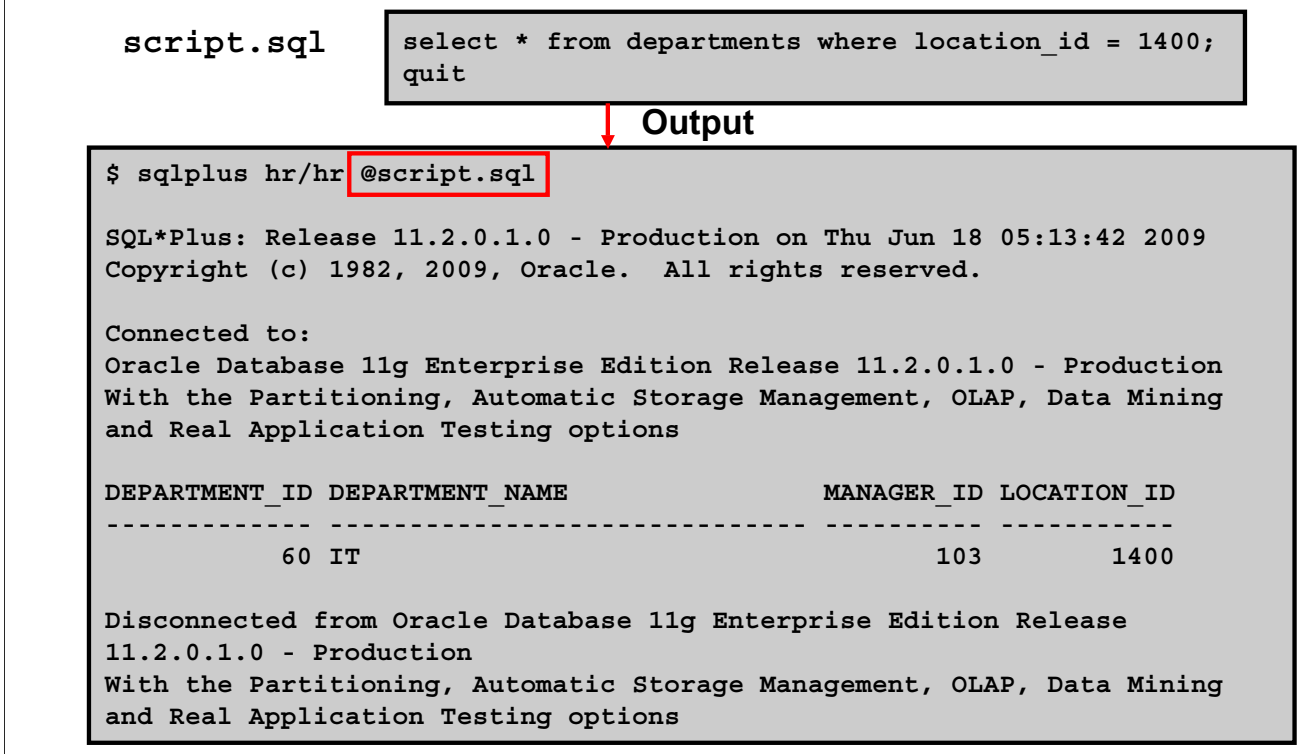


Calling SQL*Plus from a Shell Script

You can call SQL*Plus from a shell script or `BAT` file by invoking `sqlplus` and using the operating system scripting syntax for passing parameters.

In this example, the `SELECT`, `UPATE` and `COMMIT` statements are executed before SQL*Plus returns control to the operating system.

Calling a SQL Script from SQL*Plus



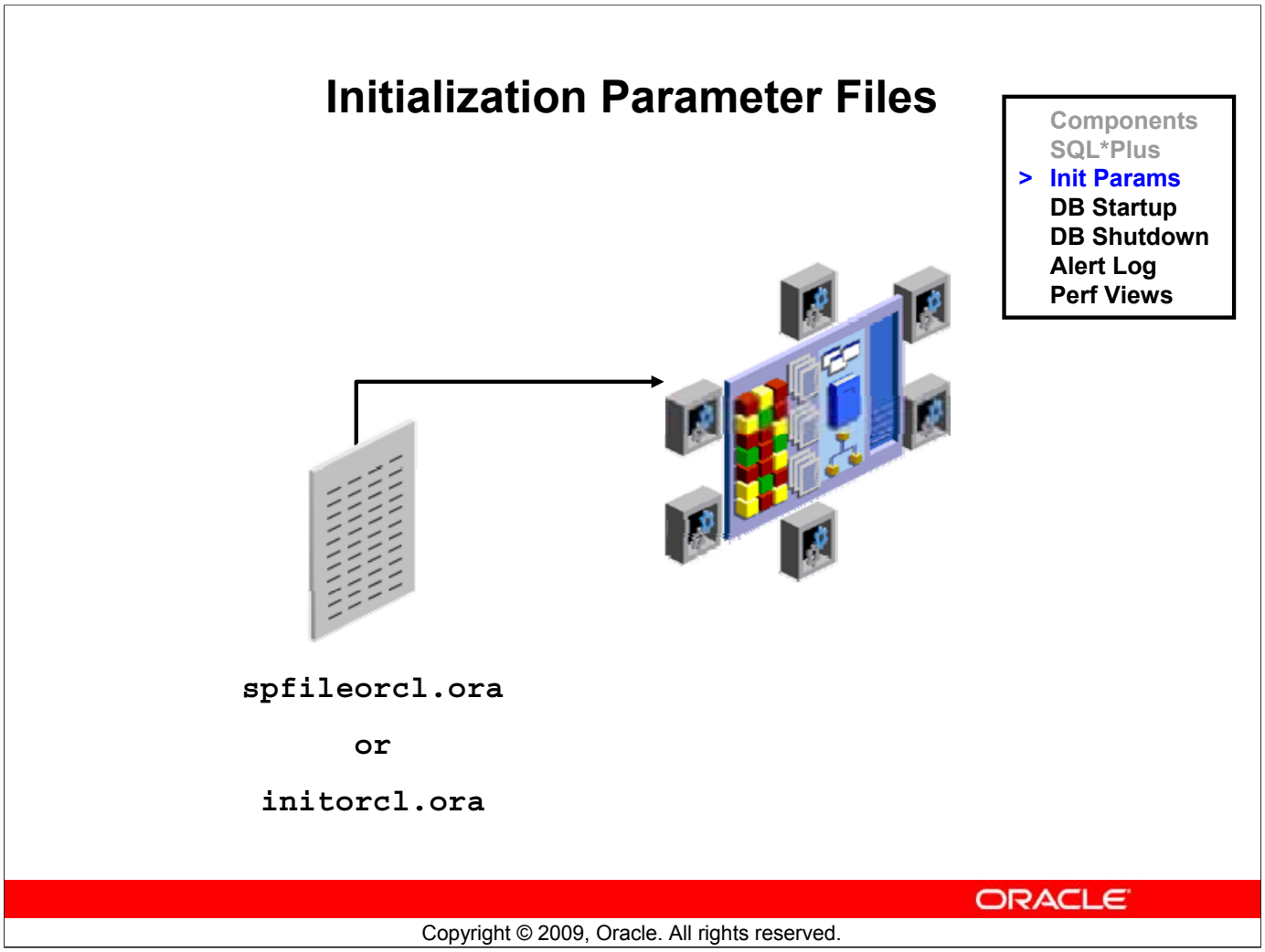
Calling a SQL Script from SQL*Plus

You can call an existing SQL script file from within SQL*Plus. This can be done at the command line when first invoking SQL*Plus, as shown in the slide. It can also be done from inside a SQL*Plus session simply by using the "@" operator. For example, this runs the script from within an already established SQL*Plus session:

```
SQL> @script.sql
```

Note: The default file extension for script files is `.sql`. When a script is saved from SQL*Plus by using the save command, this extension is automatically supplied. Scripts with this extension can be executed even without supplying the extension at execution time, as in the following example:

```
SQL> @script
```



Initialization Parameter Files

When you start the instance, an initialization parameter file is read. There are two types of parameter files.

- **Server parameter file (SPFILE):** This is the preferred type of initialization parameter file. It is a binary file that can be written to and read by the database server and *must not be edited manually*. It resides on the server on which the Oracle instance is executing; it is persistent across shutdown and startup. The default name of this file, which is automatically sought at startup, is `spfile<SID>.ora`.
- **Text initialization parameter file:** This type of initialization parameter file can be read by the database server, but it is not written to by the server. The initialization parameter settings must be set and changed manually by using a text editor so that they are persistent across shutdown and startup. The default name of this file (which is automatically sought at startup if an SPFILE is not found) is `init<SID>.ora`.

It is recommended that you create an SPFILE as a dynamic way to maintain initialization parameters.

Note: The Oracle database searches the `$ORACLE_HOME/dbs` directory on Linux for the initialization files. With ASM, the SPFILE is often located into an ASM disk group. In this case, an `init<SID>.ora` file should exist in the `$ORACLE_HOME/dbs` directory that identifies the SPFILE location.

Initialization Parameter Files (continued)

Types of Values for Initialization Parameters

The Oracle database server has the following types of values for initialization parameters:

- Boolean
- String
- Integer
- Parameter file
- Reserved
- Big Integer

Derived Parameter Values

Some initialization parameters are derived, meaning that their values are calculated from the values of other parameters. Normally, you should not alter values for derived parameters. But if you do, the value that you specify overrides the calculated value.

For example, the default value of the `SESSIONS` parameter is derived from the value of the `PROCESSES` parameter. If the value of `PROCESSES` changes, the default value of `SESSIONS` changes as well unless you override it with a specified value.

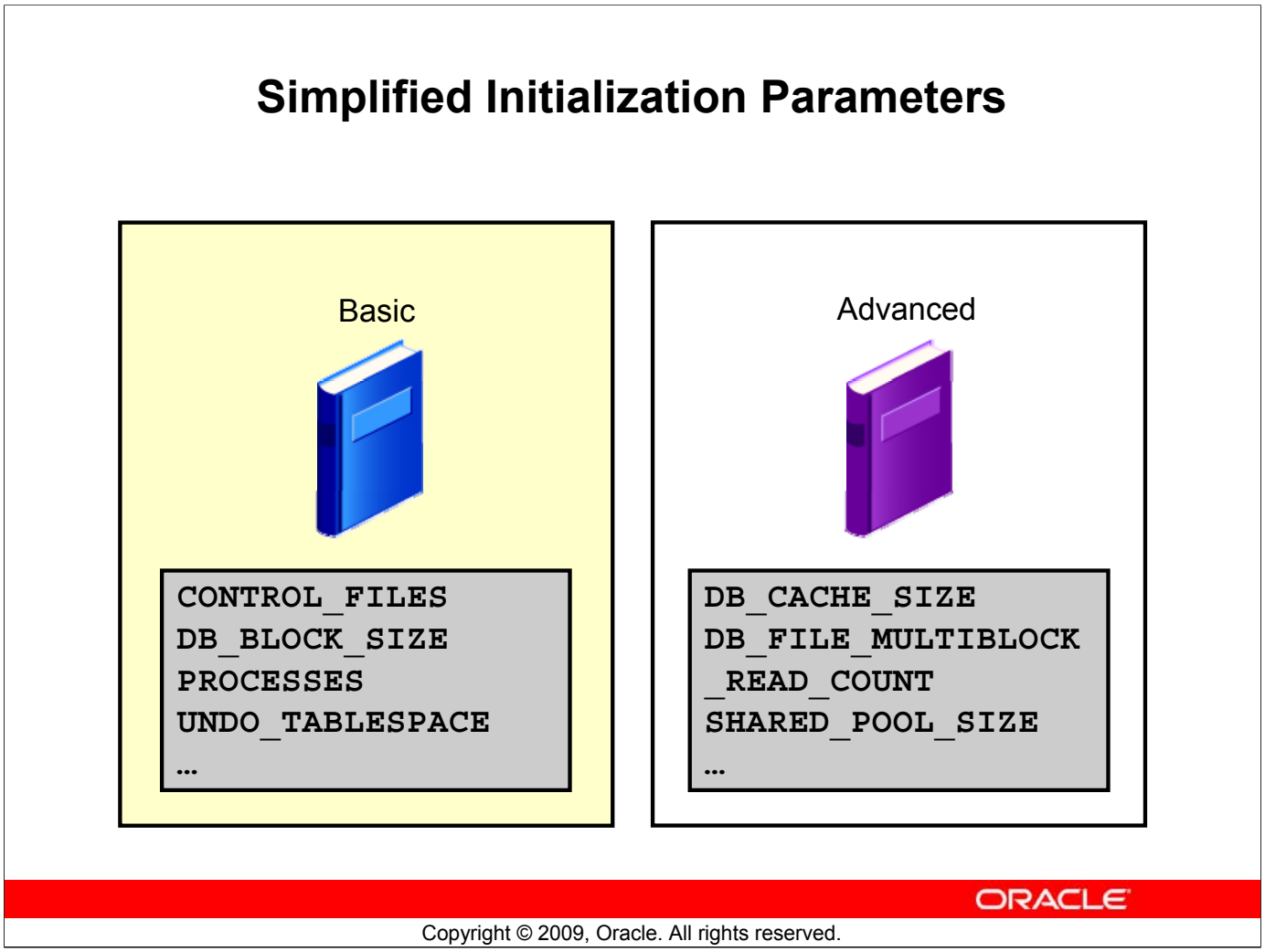
Operating System–Dependent Parameter Values

The valid values or value ranges of some initialization parameters depend on the host operating system. For example, the `DB_FILE_MULTIBLOCK_READ_COUNT` parameter specifies the maximum number of blocks that are read in one I/O operation during a sequential scan; this parameter is platform dependent. The size of those blocks, which is set by `DB_BLOCK_SIZE`, has a default value that depends on the operating system.

Setting Parameter Values

Initialization parameters offer the most potential for improving system performance. Some parameters set capacity limits but do not affect performance. For example, when the value of `OPEN_CURSORS` is `10`, a user process attempting to open its eleventh cursor receives an error. Other parameters affect performance but do not impose absolute limits. For example, reducing the value of `OPEN_CURSORS` does not prevent work even though it may slow the performance.

Increasing the values of parameters may improve your system's performance, but increasing most parameters also increases the system global area (SGA) size. A larger SGA can improve database performance up to a point. In virtual memory operating systems, an SGA that is too large can degrade performance if it is swapped in and out of memory. Operating system parameters that control virtual memory working areas should be set with the SGA size in mind. The operating system configuration can also limit the maximum size of the SGA.



Simplified Initialization Parameters

Initialization parameters are of two types: basic and advanced.

In the majority of cases, it is necessary to set and tune only the 30 basic parameters to get reasonable performance from the database. In rare situations, modification of the advanced parameters may be needed to achieve optimal performance. There are about 314 advanced parameters.

A basic parameter is defined as one that you are likely to set to keep your database running with good performance. All other parameters are considered to be advanced.

Examples of basic parameters:

- Determining the global database name: `DB_NAME` and `DB_DOMAIN`
- Specifying a fast recovery area and size: `DB_RECOVERY_FILE_DEST` and `DB_RECOVERY_FILE_DEST_SIZE`
- Specifying the total size of all SGA components: `SGA_TARGET`
- Specifying the method of undo space management tablespace: `UNDO_TABLESPACE`
- `COMPATIBLE` initialization parameter and irreversible compatibility

Note: Some of the initialization parameters are listed on the following pages. For a complete list, see the *Oracle Database Reference*.

Initialization Parameters: Examples

Parameter	Specifies
CONTROL_FILES	One or more control file names
DB_FILES	Maximum number of database files
PROCESSES	Maximum number of OS user processes that can simultaneously connect
DB_BLOCK_SIZE	Standard database block size used by all tablespaces
DB_CACHE_SIZE	Size of the standard block buffer cache

ORACLE

Initialization Parameters: Examples

CONTROL_FILES parameter: Specifies one or more control file names. Oracle strongly recommends that you multiplex and mirror control files. Range of values: from one to eight file names (with path names). Default value: OS dependent.

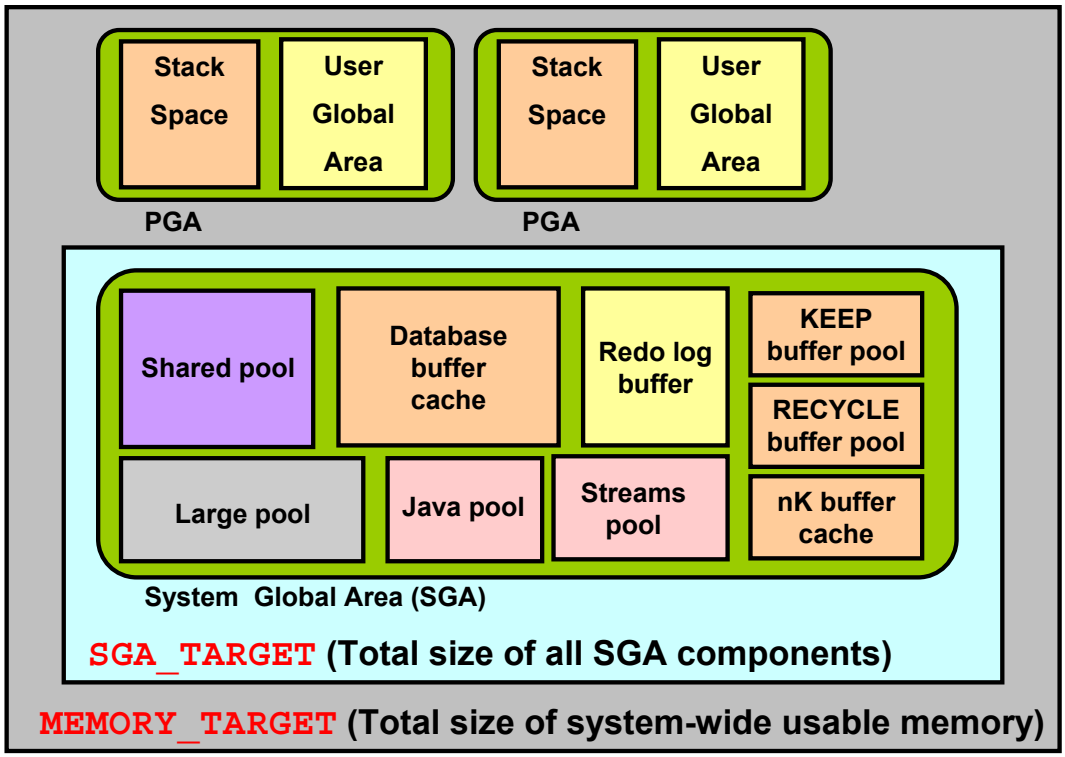
DB_FILES parameter: Specifies the maximum number of database files that can be opened for this database. Range of values: OS dependent. Default value: 200.

PROCESSES parameter: Specifies the maximum number of OS user processes that can simultaneously connect to an Oracle server. This value should allow for all background processes and user processes. Range of values: from 6 to an OS-dependent value. Default value: 100.

DB_BLOCK_SIZE parameter: Specifies the size (in bytes) of an Oracle database block. This value is set at database creation and cannot be subsequently changed. This specifies the standard block size for the database. All tablespaces will use this size by default. Range of values: 2048 to 32768 (OS dependent). Default value: 8192 .

DB_CACHE_SIZE parameter: Specifies the size of the standard block buffer cache. Range of values: at least 16 MB. Default value: 0 if SGA_TARGET is set, otherwise the larger of 48 MB or (4 MB*cpu_count).

Initialization Parameters: Examples



ORACLE

Initialization Parameters: Examples (continued)

SGA_TARGET specifies the total size of all SGA components. If SGA_TARGET is specified, the following memory pools are automatically sized:

- Buffer cache (DB_CACHE_SIZE)
- Shared pool (SHARED_POOL_SIZE)
- Large pool (LARGE_POOL_SIZE)
- Java pool (JAVA_POOL_SIZE)
- Streams pool (STREAMS_POOL_SIZE)

If these automatically tuned memory pools are set to nonzero values, the values are used as minimum levels by Automatic Shared Memory Management (ASMM). You set minimum values if an application component needs a minimum amount of memory to function properly.

The following pools are manually sized components and are not affected by ASMM:

- Log buffer
- Other buffer caches (such as KEEP and RECYCLE) and other block sizes
- Fixed SGA and other internal allocations

The memory allocated to these pools is deducted from the total available memory for SGA_TARGET when ASMM is enabled.

Note: The MMON process computes the values of the automatically tuned memory pools to support ASMM.

Initialization Parameters: Examples (continued)

MEMORY_TARGET specifies the Oracle systemwide usable memory. The database tunes memory to the MEMORY_TARGET value, reducing or enlarging the SGA and PGA as needed.

In a text-based initialization parameter file, if you omit MEMORY_MAX_TARGET and include a value for MEMORY_TARGET, the database automatically sets MEMORY_MAX_TARGET to the value of MEMORY_TARGET. If you omit the line for MEMORY_TARGET and include a value for MEMORY_MAX_TARGET, the MEMORY_TARGET parameter defaults to zero. After startup, you can then dynamically change MEMORY_TARGET to a nonzero value if it does not exceed the value of MEMORY_MAX_TARGET. The MEMORY_TARGET parameter is modifiable with the ALTER SYSTEM command. Values range from 152 MB to MEMORY_MAX_TARGET.

Initialization Parameters: Examples

Parameter	Specifies
PGA_AGGREGATE_TARGET	Amount of PGA memory allocated to all server processes
SHARED_POOL_SIZE	Size of shared pool (in bytes)
UNDO_MANAGEMENT	Undo space management mode to be used

ORACLE

Initialization Parameters: Examples (continued)

PGA_AGGREGATE_TARGET parameter: Specifies the amount of Program Global Area (PGA) memory allocated to all server processes attached to the instance. This memory does not reside in the System Global Area (SGA). The database uses this parameter as a target amount of PGA memory to use. When setting this parameter, subtract the SGA from the total memory on the system that is available to the Oracle instance. The range of values comprises integers plus the letters *K*, *M*, or *G* (to specify this limit in kilobytes, megabytes, or gigabytes). The minimum value is 10 MB and the maximum value is (4096 GB – 1). The default is 10 MB or 20% of the size of the SGA, whichever is greater.

SHARED_POOL_SIZE parameter: Specifies the size of the shared pool in bytes. The shared pool contains objects such as shared cursors, stored procedures, control structures, and parallel execution message buffers. Range of values: OS dependent. Default value: 0 if SGA_TARGET is set, otherwise 128 MB if 64 bit; 48 MB if 32 bit.

UNDO_MANAGEMENT parameter: Specifies the undo space management mode that the system should use. When set to AUTO, the instance is started in Automatic Undo Management (AUM) mode. Otherwise, it is started in Rollback Undo (RBU) mode. In RBU mode, undo space is allocated externally as rollback segments. In AUM mode, undo space is allocated externally as undo tablespaces. Range of values: AUTO or MANUAL. If the UNDO_MANAGEMENT parameter is omitted when the first instance is started, the default value AUTO is used.

Using SQL*Plus to View Parameters

```
 SQL> SELECT name , value FROM V$PARAMETER;
NAME                    VALUE
------------           ----------
lock_name_space         2
processes               150
sessions                247
timed_statistics        TRUE
timed_os_statistics     0
…

SQL>SHOW PARAMETER SHARED_POOL_SIZE
NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------------
shared_pool_size                     big integer 0

SQL> show parameter para
NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------------------
fast_start_parallel_rollback         string      LOW
parallel_adaptive_multi_user         boolean     TRUE
parallel_automatic_tuning            boolean     FALSE
parallel_execution_message_size      integer     16384
parallel_instance_group              string
…
```

ORACLE

Using SQL*Plus to View Parameters

The slide shows examples of using SQL*Plus to view parameters. You can query the data V$PARAMETER dictionary view to find the values of the various parameters. V$PARAMETER displays the current parameter values in the current session. You can also use the SHOW PARAMETER command with any string to view parameters that contain that string.

The query in the following example is requesting the name and values of the parameters. Use a WHERE clause to specify specific parameter names:

```
SQL> SELECT name, value FROM V$PARAMETER WHERE name LIKE
'%pool%';
NAME                          VALUE
----------------------------- ----------
shared_pool_size              0
large_pool_size               0
java_pool_size                0
streams_pool_size             0
shared_pool_reserved_size     8808038
buffer_pool_keep
…

9 rows selected.
```

Using SQL*Plus to View Parameters (continued)

Description of the view:

```
SQL> desc V$parameter
 Name                                      Null?  Type
 ----------------------------------------- ------- ------------
 NUM                                               NUMBER
 NAME                                              VARCHAR2(80)
 TYPE                                              NUMBER
 VALUE                                             VARCHAR2(4000)
 DISPLAY_VALUE                                     VARCHAR2(4000)
 ISDEFAULT                                         VARCHAR2(9)
 ISSES_MODIFIABLE                                  VARCHAR2(5)
 ISSYS_MODIFIABLE                                  VARCHAR2(9)
 ISINSTANCE_MODIFIABLE                             VARCHAR2(5)
 ISMODIFIED                                        VARCHAR2(10)
 ISADJUSTED                                        VARCHAR2(5)
 ISDEPRECATED                                      VARCHAR2(5)
 ISBASIC                                           VARCHAR2(5)
 DESCRIPTION                                       VARCHAR2(255)
 UPDATE_COMMENT                                    VARCHAR2(255)
 HASH                                              NUMBER
```

The second example shows the use of the SQL*Plus `SHOW PARAMETER` command to view parameter settings. You can also use this command to find all parameters that contain a text string. For example, you can find all parameter names that include the string db by using the following command:

```
SQL> show parameter db
NAME                                 TYPE        VALUE
------------------------------------ ----------- ----------
…
db_8k_cache_size                     big integer 0
db_block_buffers                     integer     0
db_block_checking                    string      FALSE
db_block_checksum                    string      TYPICAL
db_block_size                        integer     8192
db_cache_advice                      string      ON
db_cache_size                        big integer 0
…
```

Other Views Containing Information About Parameters

- `V$SPPARAMETER`: Displays information about the contents of the server parameter file. If a server parameter file was not used to start the instance, each row of the view will contain `FALSE` in the `ISSPECIFIED` column.
- `V$PARAMETER2`: Displays information about the initialization parameters that are currently in effect for the session, with each parameter value appearing as a row in the view. A new session inherits parameter values from the instance-wide values displayed in the `V$SYSTEM_PARAMETER2` view.
- `V$SYSTEM_PARAMETER`: Displays information about the initialization parameters that are currently in effect for the instance.

Changing Initialization Parameter Values

- Static parameters:
  - Can be changed only in the parameter file
  - Require restarting the instance before taking effect
  - Account for about 110 parameters
- Dynamic parameters:
  - Can be changed while database is online
  - Can be altered at:
    - Session level
    - System level
  - Are valid for duration of session or based on `SCOPE` setting
  - Are changed by using `ALTER SESSION` and `ALTER SYSTEM` commands
  - Account for about 234 parameters

ORACLE

Changing Initialization Parameter Values

There are two types of initialization parameters.

Static parameters: Affect the instance or entire database and can be modified only by changing the contents of the `init.ora` or the SPFILE. Static parameters require the database to be shut down and restarted to take effect. They cannot be changed for the current instance.

Dynamic parameters: Can be changed while database is online. There are two types:

- *Session-level parameters* affect only a user session. Examples include national language support (NLS) parameters that can be used to specify national language settings for sorts, date parameters, and so on. You can use these in a given session; they expire when the session ends.
- *System-level parameters* affect the entire database and all sessions. Examples include modifying the SGA_TARGET value and setting archive log destinations. These parameters stay in effect based on the `SCOPE` specification. To make them permanent, you have to add these parameter settings to the SPFILE by specifying the SCOPE=both option or manually editing the PFILE.

Dynamic parameters can be changed by using the `ALTER SESSION` and `ALTER SYSTEM` commands.

Changing Initialization Parameter Values (continued)

Use the SET clause of the ALTER SYSTEM statement to set or change initialization parameter values. The optional SCOPE clause specifies the scope of a change as follows:

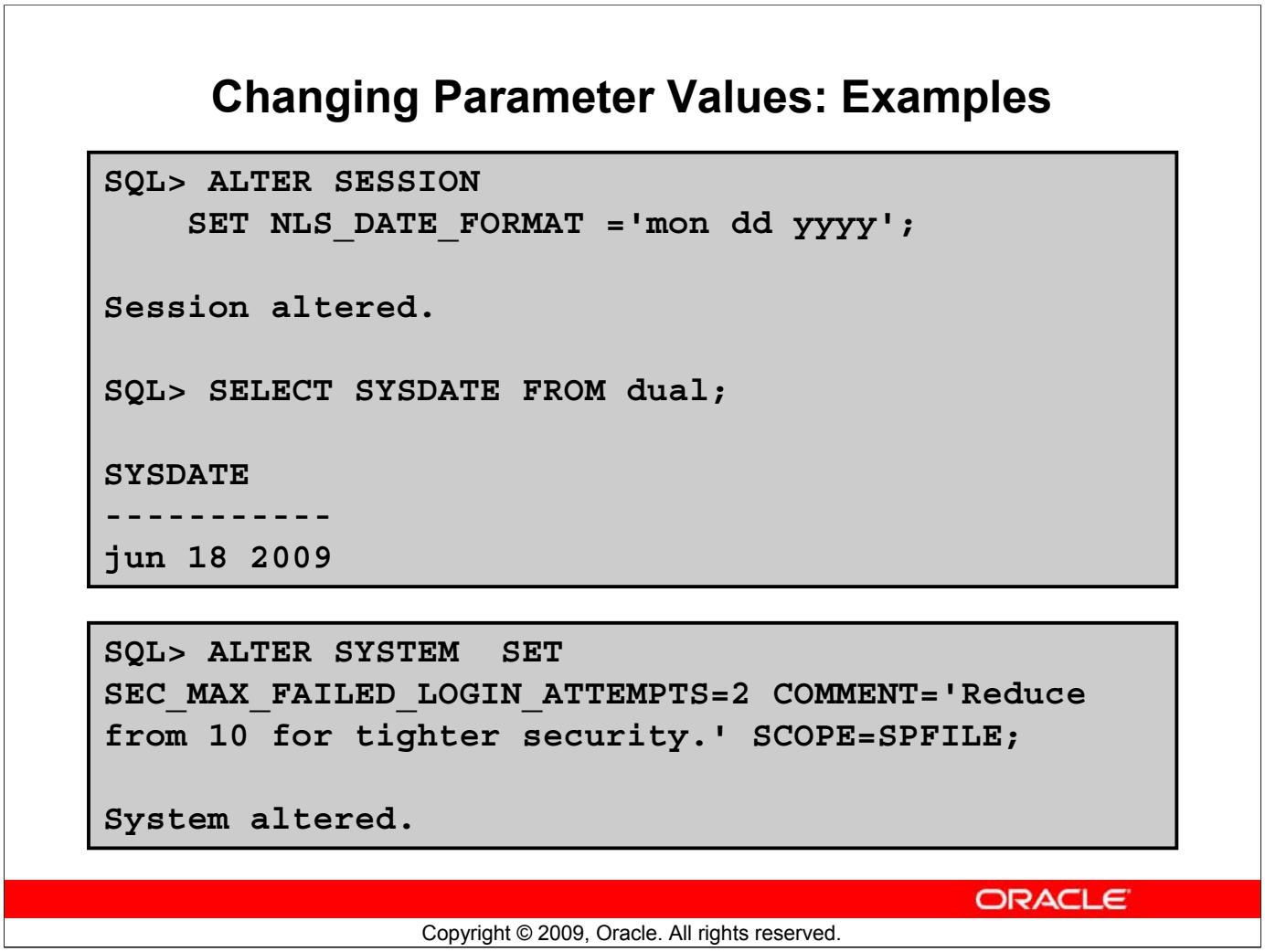
- **SCOPE=SPFILE:** The change is applied in the server parameter file only. No change is made to the current instance. For both dynamic and static parameters, the change is effective at the next startup and is persistent. This is the only SCOPE specification allowed for static parameters.
- **SCOPE=MEMORY:** The change is applied in memory only. The change is made to the current instance and is effective immediately. For dynamic parameters, the effect is immediate but not persistent because the server parameter file is not updated. For static parameters, this specification is not allowed.
- **SCOPE=BOTH:** The change is applied in both the server parameter file and memory. The change is made to the current instance and is effective immediately. For dynamic parameters, the effect is persistent because the server parameter file is updated. For static parameters, this specification is not allowed.

It is an error to specify SCOPE=SPFILE or SCOPE=BOTH if the instance did not start up with a server parameter file. The default is SCOPE=BOTH if a server parameter file was used to start up the instance, and the default is MEMORY if a text initialization parameter file was used to start up the instance.

For some dynamic parameters, you can also specify the DEFERRED keyword. When it is specified, the change is effective only for future sessions. This is only valid for the following parameters:

- backup_tape_io_slaves
- recyclebin
- audit_file_dest
- object_cache_optimal_size
- object_cache_max_size_percent
- sort_area_size
- sort_area_retained_size
- olap_page_pool_size

When you specify SCOPE as SPFILE or as BOTH, an optional COMMENT clause lets you associate a text string with the parameter update. The comment is written to the server parameter file.



Changing Parameter Values: Examples

The first statement in the slide is an example of changing a session-level parameter. The user is setting the session date format to be `mon dd yyyy`. As a result, any queries on the date will display dates in that format. Session-level parameters can also be set in applications by using PL/SQL.

The second statement changes the maximum number of failed login attempts before the connection is dropped. It includes a comment and explicitly states that the change is to be made only in the server parameter file. After the specified number of failure attempts, the connection is automatically dropped by the server process. This is not a dynamic parameter and the Oracle database instance will need to be restarted before the change can take effect.

Quiz

Enterprise Manager Database Control can be used to manage many databases concurrently.

1. True
2. False

Answer: 2

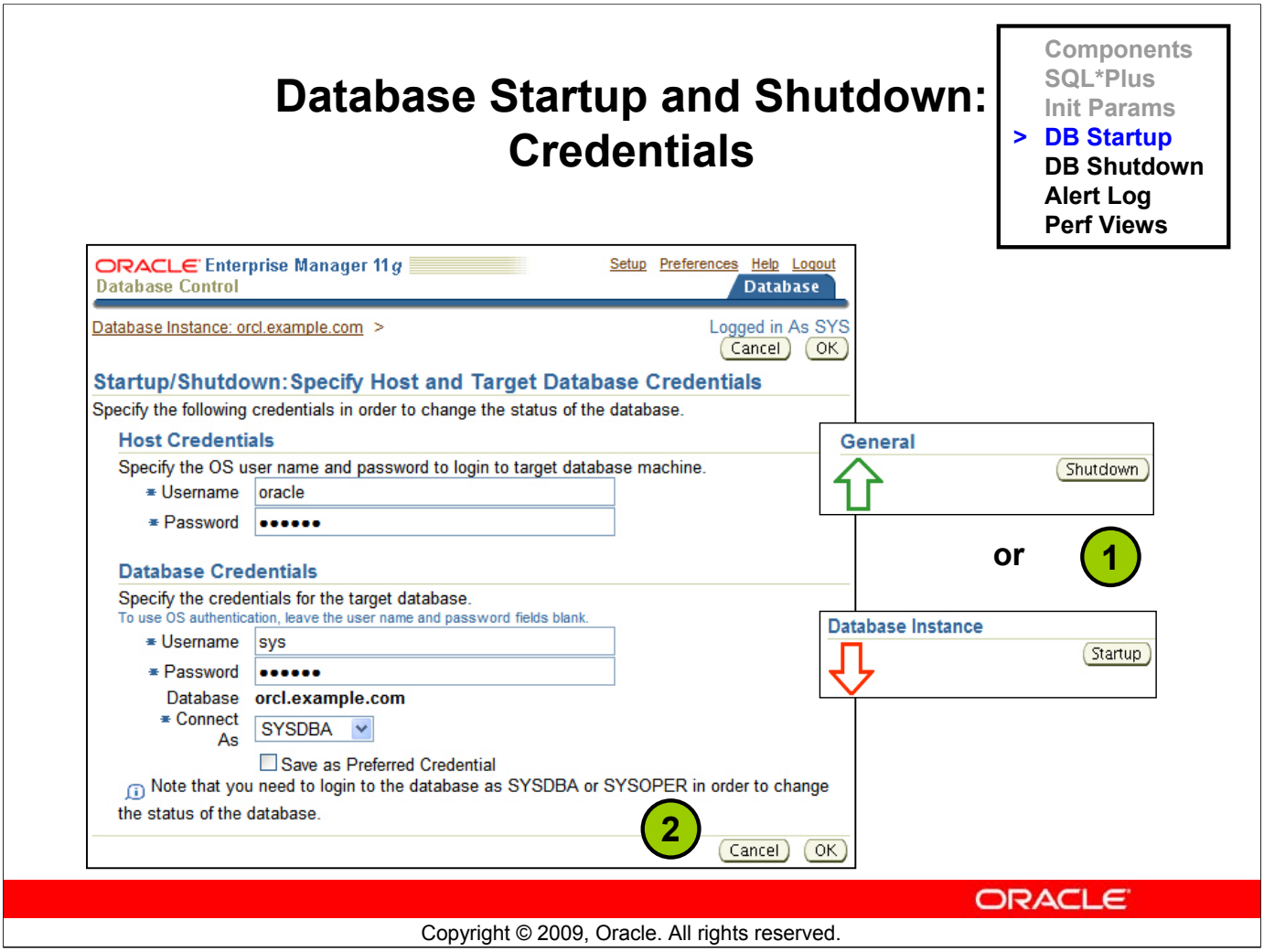
Quiz

The majority of the database parameters are dynamic and can be changed without having to shut down the database instance.

1. True
2. False

Answer: 1

Database Startup and Shutdown: Credentials



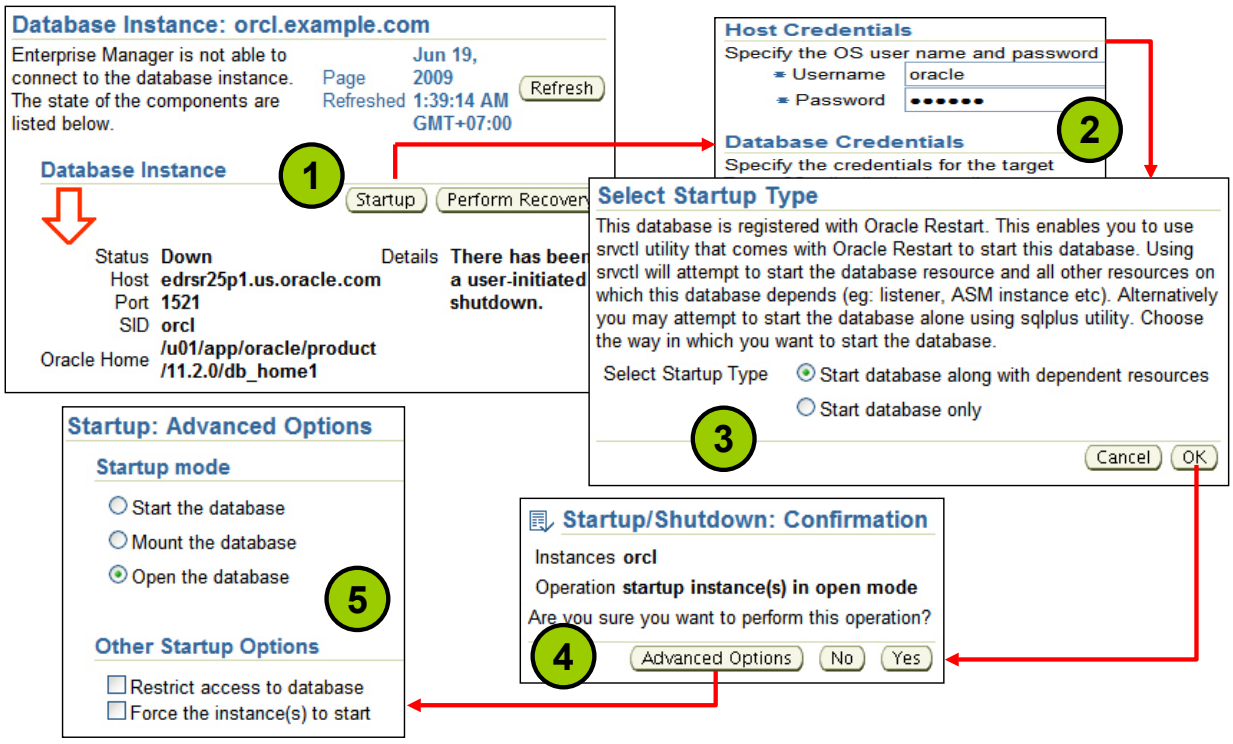
Database Startup and Shutdown: Credentials

When you click either Startup or Shutdown, you are prompted for credentials that are used for both logging on to the host (the computer on which the database resides) and logging in to the database itself. You must enter a database account that has the `SYSDBA` privilege. Enter the credentials.

After the credentials information, you will then be prompted for the method of startup or shutdown. You can then click Advanced Options to change any startup options or shutdown mode as needed. You can also click Show SQL to see the SQL statements that are used for startup or shutdown.

Note: The default option to shut down by using Enterprise Manager is `IMMEDIATE`. The default when issuing the `SHUTDOWN` command from SQL*Plus is `NORMAL`.

Starting Up an Oracle Database Instance

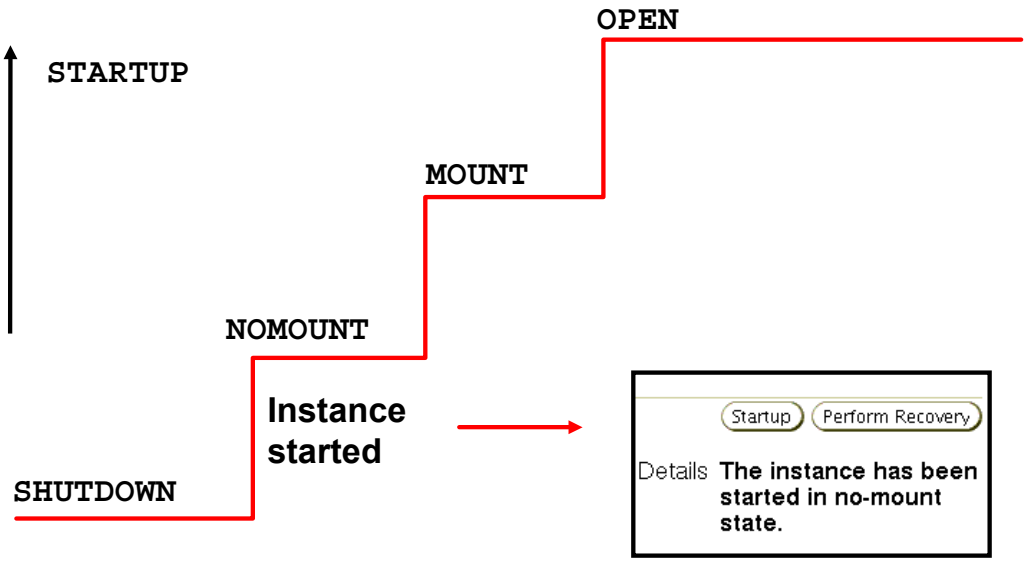


ORACLE

Starting Up an Oracle Database Instance

If the database is currently not started when you go to the Enterprise Manager Database Control page, click Startup. Then enter the host credentials and, optionally, choose the startup mode. If the Oracle database has been registered with Oracle Restart, a separate dialog will prompt you with the choice of using the Server Control (`SRVCTL`) utility or SQL*Plus to start the database instance. The `SRVCTL` utility is the recommended utility when using Oracle Restart because it has the ability to start up dependent resources that may be needed.

Starting Up an Oracle Database Instance: NOMOUNT



ORACLE

Starting Up an Oracle Database Instance: NOMOUNT

When starting the database instance, select the state in which it starts. The following scenarios describe different stages of starting up an instance.

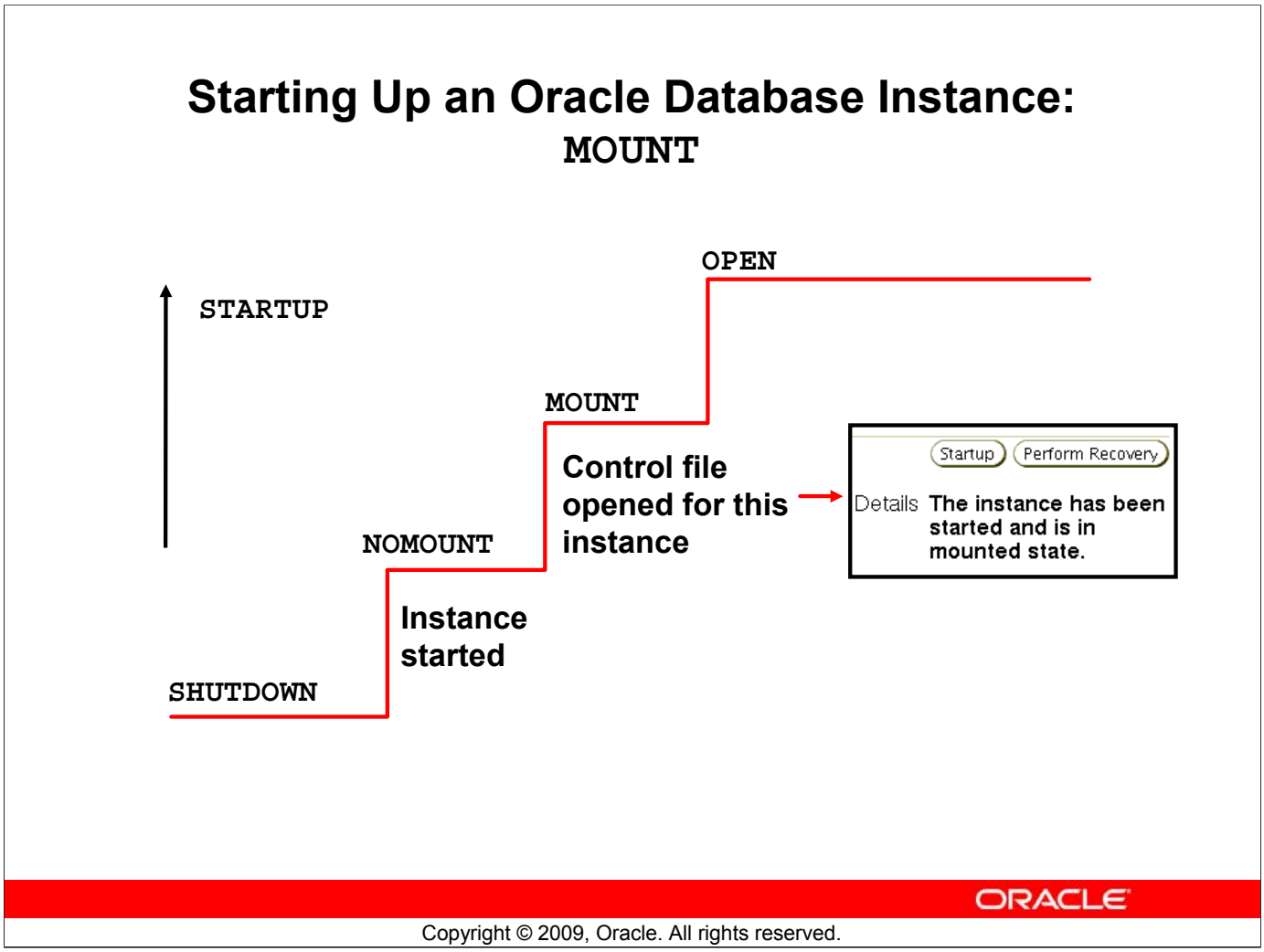
An instance is typically started only in NOMOUNT mode during database creation, during re-creation of control files, or during certain backup and recovery scenarios.

Starting an instance includes the following tasks:

- Searching $ORACLE_HOME/dbs for a file of a particular name in this sequence:
  1. Search for spfile<SID>.ora.
  2. If spfile<SID>.ora is not found, search for spfile.ora.
  3. If spfile.ora is not found, search for init<SID>.ora.

     This is the file that contains initialization parameters for the instance. Specifying the PFILE parameter with STARTUP overrides the default behavior.
- Allocating the SGA
- Starting the background processes
- Opening the alert_<SID>.log file and the trace files

Note: SID is the system ID, which identifies the instance name (for example, ORCL).



Starting Up an Oracle Database Instance: MOUNT

Mounting a database includes the following tasks:

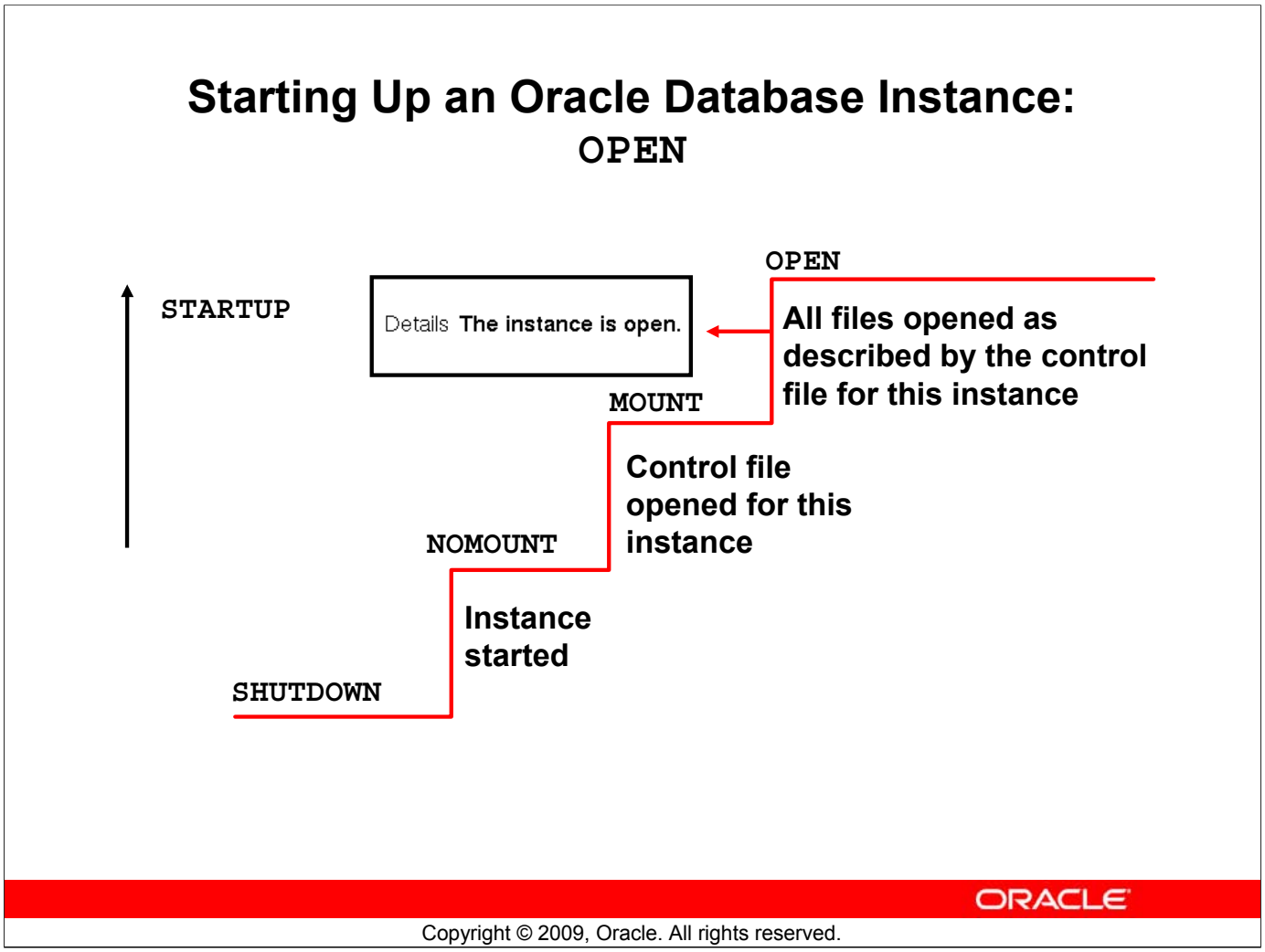
- Associating a database with a previously started instance
- Locating and opening all of the control files specified in the parameter file
- Reading the control files to obtain the names and statuses of the data files and online redo log files (However, no checks are performed to verify the existence of the data files and online redo log files at this time.)

To perform specific maintenance operations, start an instance and mount a database, but do not open the database.

For example, the database must be mounted but must not be opened during the following tasks:

- Renaming data files (Data files for an offline tablespace can be renamed when the database is open.)
- Enabling and disabling online redo log file archiving options
- Performing full database recovery

Note: A database may be left in `MOUNT` mode even though an `OPEN` request has been made. This may be because the database needs to be recovered in some way. If recovery is performed while in the `MOUNT` state, the redo logs are open for reads and the data files are open as well to read the blocks needing recovery and to write blocks if required during recovery.



Starting Up an Oracle Database Instance: OPEN

A normal database operation means that an instance is started and the database is mounted and opened. With a normal database operation, any valid user can connect to the database and perform typical data access operations.

Opening the database includes the following tasks:

- Opening the data files
- Opening the online redo log files

If any of the data files or online redo log files are not present when you attempt to open the database, the Oracle server returns an error.

During this final stage, the Oracle server verifies that all data files and online redo log files can be opened, and checks the consistency of the database. If necessary, the System Monitor (SMON) background process initiates instance recovery.

You can start up a database instance in restricted mode so that it is available to users with administrative privileges only. To start an instance in restricted mode, select the "Restrict access to database" option on the Advanced Startup Options page.

Startup Options: Examples

- Using the `sqlplus` utility:

```
SQL> startup                                    1
```

```
SQL> startup nomount                            2
```

```
SQL> alter database mount;                      3
```

```
SQL> alter database open;                       4
```

- Using the `srvctl` utility with Oracle Restart

```
$ srvctl start database -d orcl -o mount
```

ORACLE

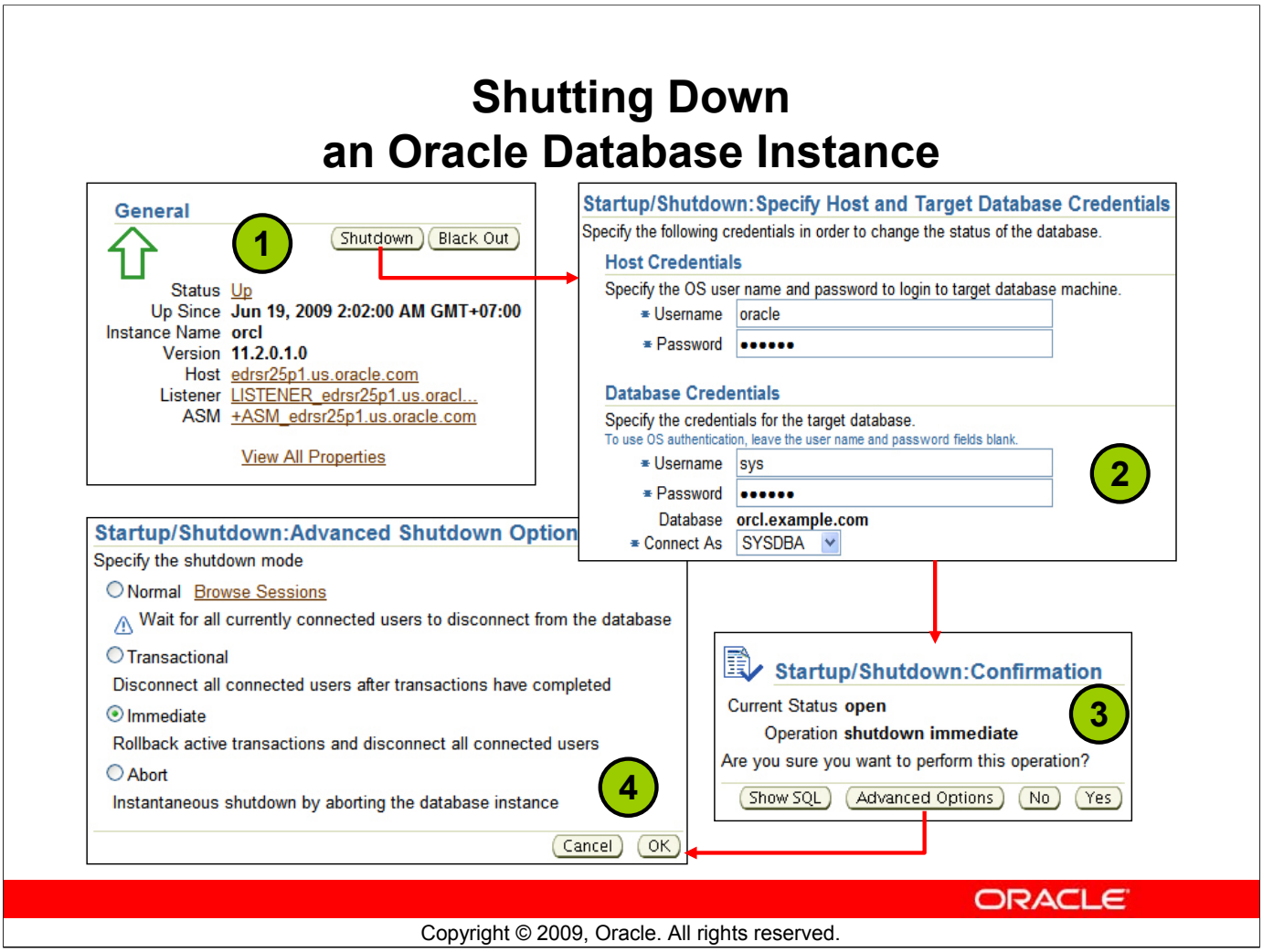
Startup Options: Examples

The slide shows the SQL*Plus syntax to start up the database.

1. This command starts the instance, associates the database files to it, and mounts and opens the database
2. This command starts the instance and the database is not mounted
3. This command mounts a database from the `NOMOUNT` state.
4. This command opens the database from the `MOUNT` state.

When the database is enabled with Oracle Restart, the `srvctl` utility can be used to start the database instance. The `srvctl` utility has the advantage that it can also start all required dependent resources such as the ASM instance, ASM disk groups, and listener.

Note: The `srvctl` utility is located in both the `$ORACLE_HOME/bin` directory for the Grid infrastructure software and the `$ORACLE_HOME/bin` directory for the Oracle database software. You should use the `srvctl` utility from the Oracle database software when starting the Oracle database. You should use the `srvctl` utility from the Grid Infrastructure software when starting the ASM Instance or the listener.



Shutting Down an Oracle Database Instance

If the instance is already started when you go to the Enterprise Manager Database Control page, click the Shutdown button to shut down the instance. You will be prompted to verify or enter the host and database credentials. Click OK to receive the Startup/Shutdown confirmation dialog. If you then click the Advanced Options button, you can select the mode of the shutdown: `NORMAL`, `TRANSACTIONAL`, `IMMEDIATE`, or `ABORT`.

Shutdown Modes

Shutdown Modes	A	I	T	N
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

Shutdown modes:

- A = ABORT
- I = IMMEDIATE
- T = TRANSACTIONAL
- N = NORMAL

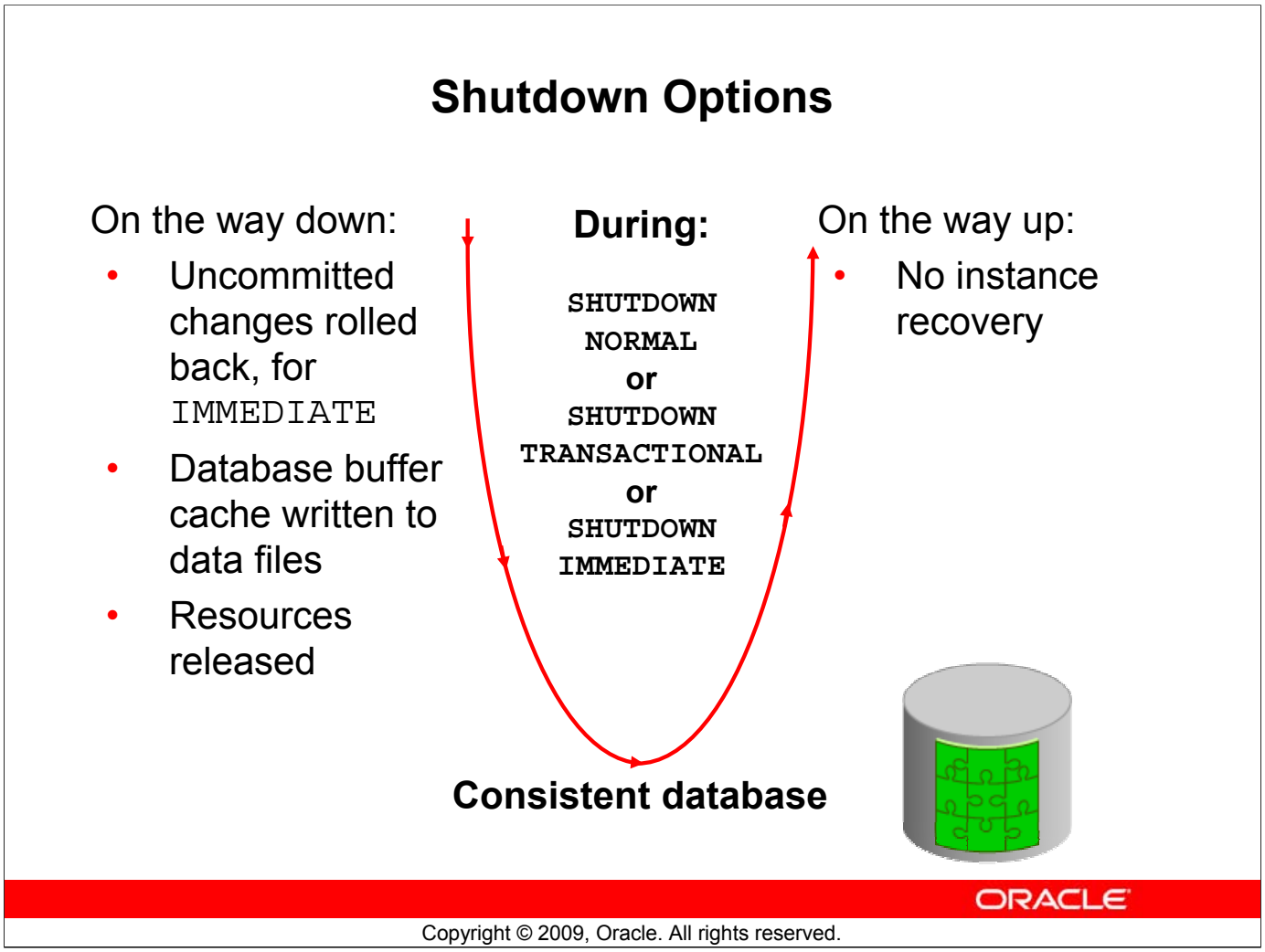
ORACLE

Shutdown Modes

Shutdown modes are progressively more accommodating of current activity in this order:

- **ABORT:** Performs the least amount of work before shutting down. Because this mode requires recovery before startup, use it only when necessary. It is typically used when no other form of shutdown works, when there are problems when starting the instance, or when you need to shut down immediately because of an impending situation (such as notice of a power outage within seconds).
- **IMMEDIATE:** Is the most typically used option. Uncommitted transactions are rolled back.
- **TRANSACTIONAL:** Allows existing transactions to finish, but not starting new transactions
- **NORMAL:** Waits for sessions to disconnect

If you consider the amount of time that it takes to perform the shutdown, you find that ABORT is the fastest and NORMAL is the slowest. NORMAL and TRANSACTIONAL can take a long time depending on the number of sessions and transactions.



Shutdown Options

SHUTDOWN NORMAL

NORMAL is the default shutdown mode if no mode is specified. A normal database shutdown proceeds with the following conditions:

- No new connections can be made.
- The Oracle server waits for all users to disconnect before completing the shutdown.
- Database and redo buffers are written to disk.
- Background processes are terminated and the SGA is removed from memory.
- The Oracle server closes and dismounts the database before shutting down the instance.
- The next startup does not require an instance recovery.

SHUTDOWN TRANSACTIONAL

A shutdown in TRANSACTIONAL mode prevents clients from losing data, including results from their current activity. A transactional database shutdown proceeds with the following conditions:

- No client can start a new transaction on this particular instance.
- A client is disconnected when the client ends the transaction that is in progress.
- When all transactions have been completed, a shutdown occurs immediately.
- The next startup does not require an instance recovery.

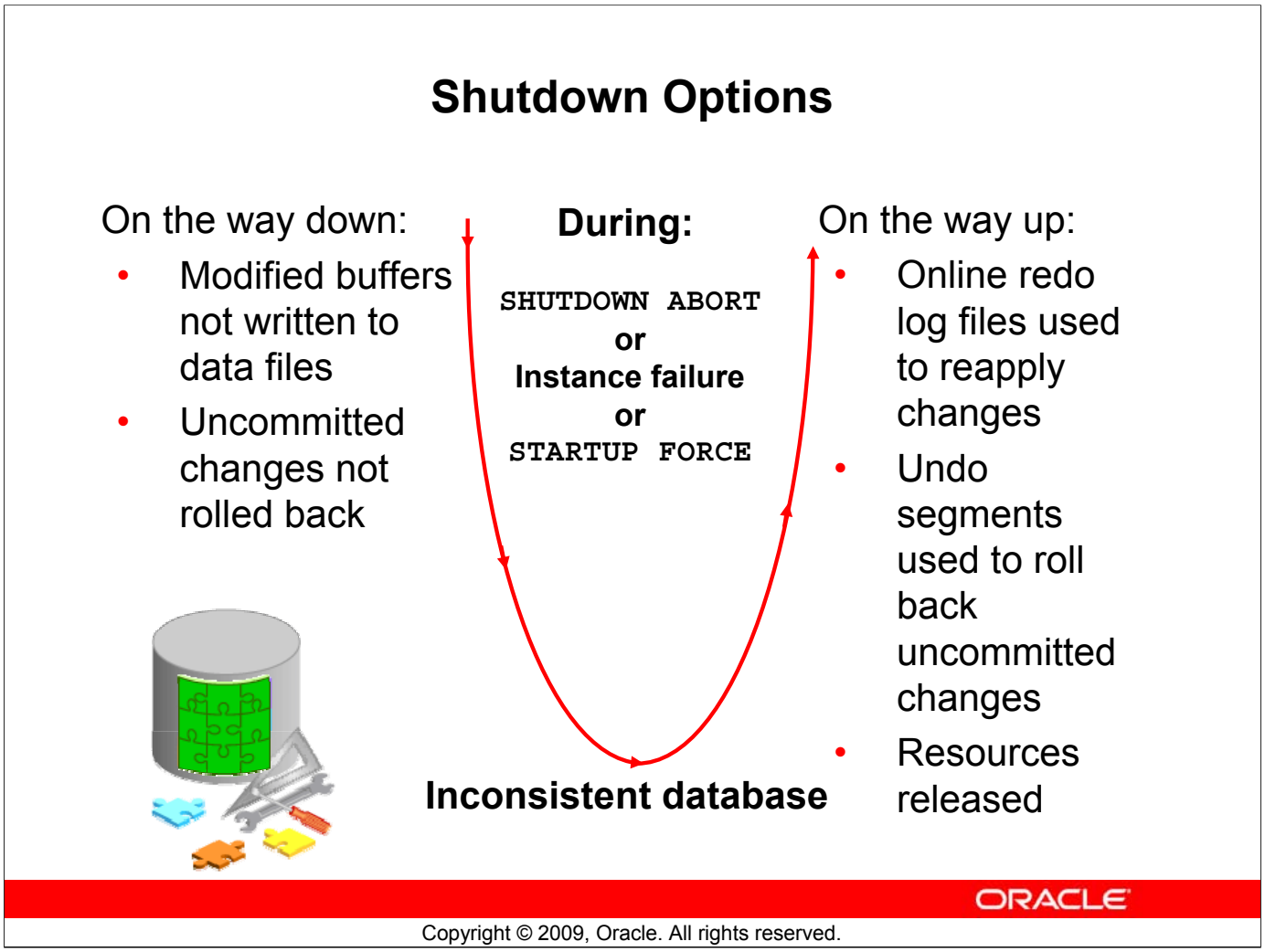
Shutdown Options (continued)

SHUTDOWN IMMEDIATE

A shutdown in IMMEDIATE mode proceeds with the following conditions:

- Current SQL statements being processed by the Oracle database are not completed.
- The Oracle server does not wait for the users who are currently connected to the database to disconnect.
- The Oracle server rolls back active transactions and disconnects all connected users.
- The Oracle server closes and dismounts the database before shutting down the instance.
- The next startup does not require an instance recovery.

Note: IMMEDIATE is the default shutdown mode when using Enterprise Manager.



Shutdown Options (continued)

SHUTDOWN ABORT

If shutdown in NORMAL, TRANSACTIONAL, and IMMEDIATE modes does not work, you can abort the current database instance. Aborting an instance proceeds with the following conditions:

- Current SQL statements being processed by the Oracle server are immediately terminated.
- The Oracle server does not wait for users who are currently connected to the database to disconnect.
- Database and redo buffers are not written to disk.
- Uncommitted transactions are not rolled back.
- The instance is terminated without closing the files.
- The database is not closed or dismounted.
- The next startup requires instance recovery, which occurs automatically.

Note: It is not advisable to back up a database that is in an inconsistent state.

Shutdown Options: Examples

- Using SQL*Plus:

```
SQL> shutdown                          1

SQL> shutdown transactional            2

SQL> shutdown immediate                3

SQL> shutdown abort                    4
```

- Using the SRVCTL utility with Oracle Restart

```
$ srvctl stop database -d orcl -o abort
```

ORACLE

Shutdown Options: Examples

The slide shows examples using both SQL*Plus and the SRVCTL utility to shut down the database.

1. This command initiates a normal shutdown. The database will not shutdown until all users have logged out.
2. This command initiates a transactional shutdown. The database will not shutdown until all existing transactions completed.
3. This command initiates an immediate shutdown. The uncommitted transactions will be rolled back
4. This command initiates a shutdown abort.

When the database is enabled with Oracle Restart, the SRVCTL utility can be used to shutdown the database instance.

Note: The SRVCTL utility is located in both the `$ORACLE_HOME/bin` directory for the Grid infrastructure software and the `$ORACLE_HOME/bin` directory for the Oracle database software. You should use the SRVCTL utility from the Oracle database software when starting the Oracle database. You should use the SRVCTL utility from the Grid Infrastructure software when starting the ASM Instance or the listener.

Viewing the Alert Log

Components
SQL*Plus
Init Params
DB Startup
DB Shutdown
> Alert Log
Perf Views

Database Home page > Related Links region > Alert Log Content

View Entries Last 50 Go Search

Previous 1-25 of 50 Next 25

Timestamp	Type	Level	Incident ID	Group	Message ID	Message Text
Jun 19, 2009 10:00:16 PM GMT+07:00	NOTIFICATION	16		sqltune	kesaiTuneSqlDrv:5067:3456118459	End automatic SQL Tuning Advisor run for special tuning task "SYS_AUTO_SQL_TUNING_TASK"
Jun 19, 2009 10:00:03 PM GMT+07:00	NOTIFICATION	16		sqltune	kesaiTuneSqlDrv:4555:2579917519	Begin automatic SQL Tuning Advisor run for special tuning task "SYS_AUTO_SQL_TUNING_TASK"
Jun 19, 2009 10:00:00 PM GMT+07:00	NOTIFICATION	16		process start	ksbrdp:3833:3697353022	VKRM started with pid=24, OS id=7929
Jun 19, 2009 10:00:00 PM GMT+07:00	NOTIFICATION	16		process start	ksbs1p_real:2253:2371767696	Starting background process VKRM
Jun 19, 2009 2:07:22 AM GMT+07:00	NOTIFICATION	16		process start	ksbrdp:3833:3697353022	SMCO started with pid=23, OS id=30582
Jun 19, 2009 2:07:22 AM GMT+07:00	NOTIFICATION	16		process start	ksbs1p_real:2253:2371767696	Starting background process SMCO
Jun 19, 2009 2:02:26 AM GMT+07:00	NOTIFICATION	16		process start	ksbrdp:3833:3697353022	CJQ0 started with pid=33, OS id=29846

ORACLE

Viewing the Alert Log

Each database has an `alert_<sid>.log` file. The file is on the server with the database and is stored in `$ORACLE_BASE/diag/rdbms/<db_name>/<SID>/trace` by default if `$ORACLE_BASE` is set.

The alert file of a database is a chronological log of messages such as the following:

- 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

Oracle Database uses the alert log to keep a record of these events as an alternative to displaying the information on an operator's console. (Many systems also display this information on the console.) If an administrative operation is successful, a message is written in the alert log as "completed" along with a time stamp.

Viewing the Alert Log (continued)

Enterprise Manager monitors the alert log file and notifies you of critical errors. You can also view the log to see noncritical error and information messages. Because the file can grow to an unmanageable size, you can periodically back up the alert file and delete the current alert file. When the database attempts to write to the alert file again, it re-creates a new one.

Note: There is an XML version of the alert log in the `$ORACLE_BASE/diag/rdbms/<db_name>/<SID>/alert` directory.

To determine the location of the alert log with SQL*Plus:

- Connect to the database with SQL*Plus (or another query tool such as SQL Developer).
- Query the `V$DIAG_INFO` view.

To view the text-only alert log without the XML tags:

- In the `V$DIAG_INFO` query results, note the path that corresponds to the Diag Trace entry. Change directory to that path.
- Open the `alert_SID.log` file with a text editor.

To view the XML-formatted alert log:

- In the `V$DIAG_INFO` query results, note the path that corresponds to the Diag Alert entry. Change directory to that path.
- Open the `log.xml` file with a text editor.

Using Trace Files

- Each server and background process can write to an associated trace file.
- Error information is written to the corresponding trace file.
- Automatic diagnostic repository (ADR)
  - Is a systemwide central tracing and logging repository
  - Stores database diagnostic data such as:
    - Traces
    - Alert log
    - Health monitor reports

ORACLE

Using Trace Files

Each server and background process can write to an associated trace file. When a process detects an internal error, it dumps information about the error to its trace file. If an internal error occurs and information is written to a trace file, the administrator should contact Oracle Support Services.

All file names of trace files associated with a background process contain the name of the process that generated the trace file. The one exception to this is trace files that are generated by job queue processes (`Jnnn`).

Additional information in trace files can provide guidance for tuning applications or an instance. Background processes always write this information to a trace file when appropriate.

Beginning with Oracle Database 11*g*, an advanced fault diagnosability infrastructure is included for preventing, detecting, diagnosing, and resolving problems. In particular, problems that are targeted include critical errors such as those caused by database code bugs, metadata corruption, and customer data corruption.

Using Trace Files (continued)

When a critical error occurs, an incident number is assigned to it; diagnostic data for the error (such as trace files) is immediately captured and tagged with this number. The data is then stored in the automatic diagnostic repository (ADR)—a file-based repository outside the database—where it can later be retrieved by incident number and analyzed.

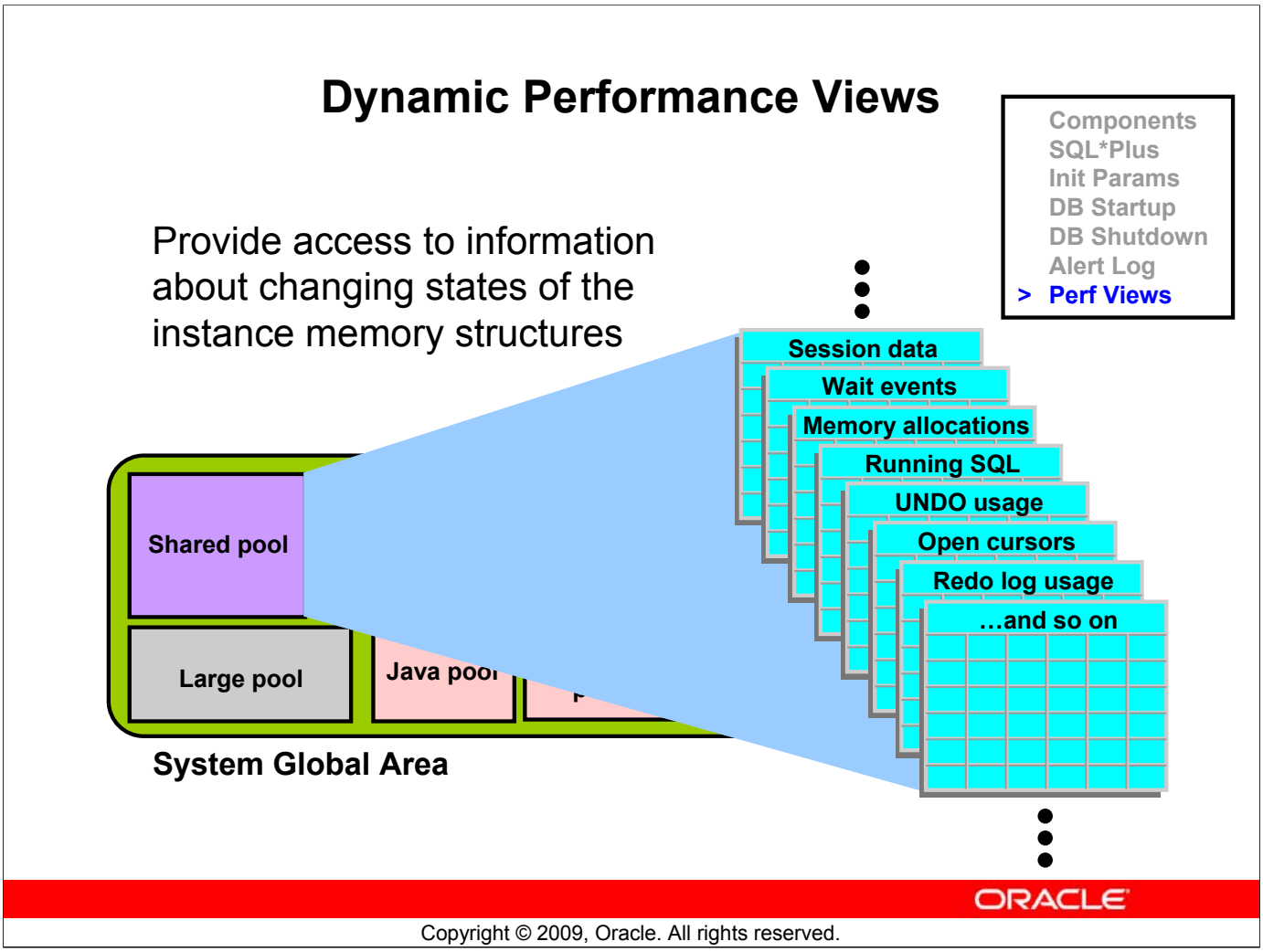
The ADR is a systemwide tracing and logging central repository for database diagnostic data such as traces, the alert log, health monitor reports, and more.

The ADR root directory is known as *ADR base*. Its location is set by the `DIAGNOSTIC_DEST` initialization parameter. If this parameter is omitted or left null, the database sets `DIAGNOSTIC_DEST` upon startup as follows:

- If the `ORACLE_BASE` environment variable is set, `DIAGNOSTIC_DEST` is set to the directory designated by `ORACLE_BASE`.
- If the `ORACLE_BASE` environment variable is not set, `DIAGNOSTIC_DEST` is set to `ORACLE_HOME/log`.

The location of an ADR home is given by the following path, which starts at the ADR base directory:

```
./diag/product_type/db_id/instance_id
```



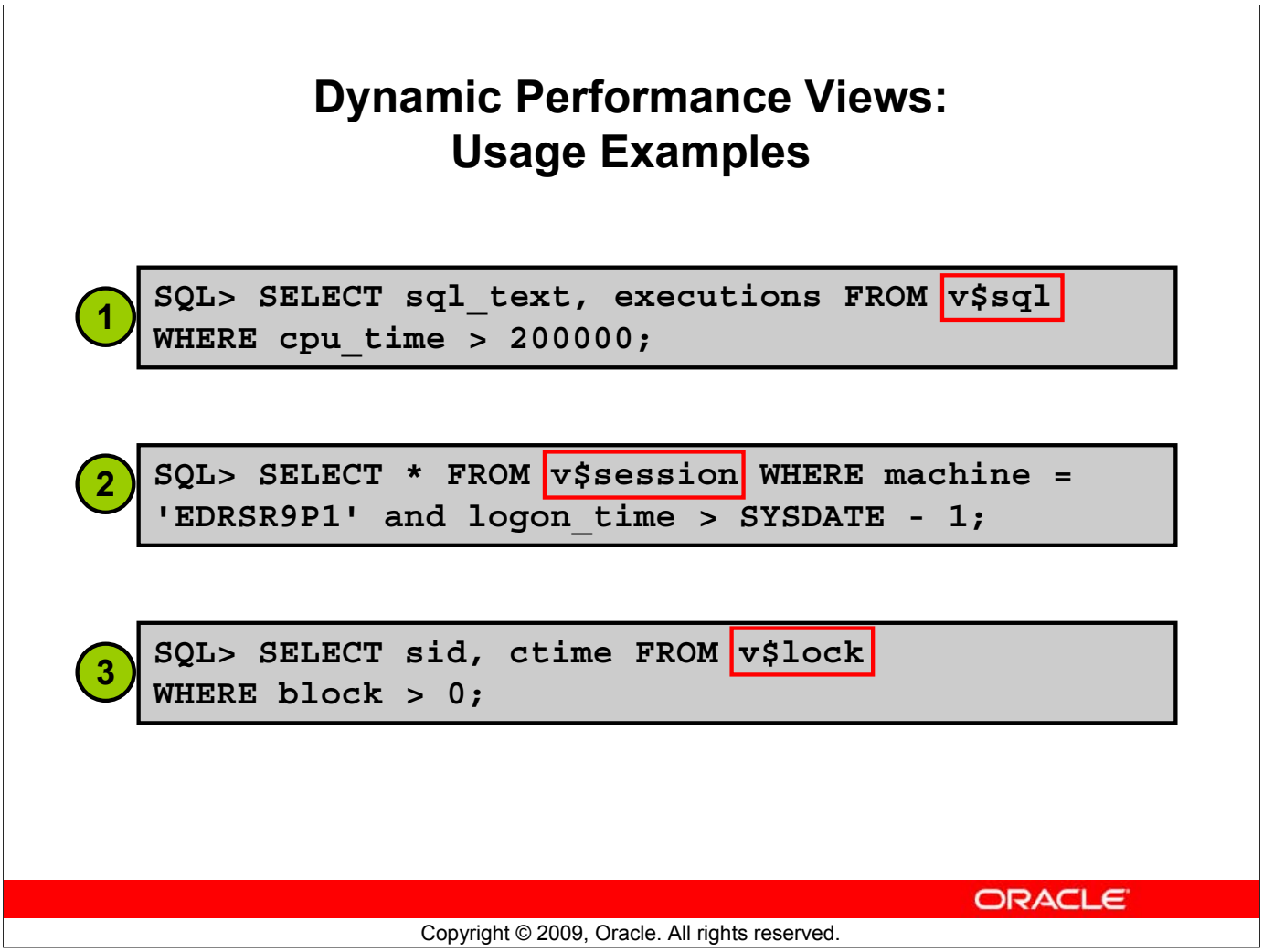
Dynamic Performance Views

The Oracle database also maintains a more dynamic set of data about the operation and performance of the database instance. These dynamic performance views are based on virtual tables that are built from memory structures inside the database server. That is, they are not conventional tables that reside in a database. This is the reason why some of them are available before a database is mounted or open.

Dynamic performance views include information about:

- Sessions
- File states
- Progress of jobs and tasks
- Locks
- Backup status
- Memory usage and allocation
- System and session parameters
- SQL execution
- Statistics and metrics

Note: The `DICT` and `DICT_COLUMNS` views also contain the names of these dynamic performance views. Dynamic performance views start with the prefix 'v$' and there are over 590 of them.



Dynamic Performance Views: Usage Examples

A frequent user of these views is Enterprise Manager, but users can also query these views as needed. The three examples shown in the slide answer the following questions:

1. For which SQL statements (and their associated numbers of executions) is the CPU time consumed greater than 200,000 microseconds?
2. Which current sessions are logged in from the `EDRSR9P1` computer on the last day?
3. What are the session IDs of those sessions that are currently holding a lock that is blocking another user, and how long have those locks been held?

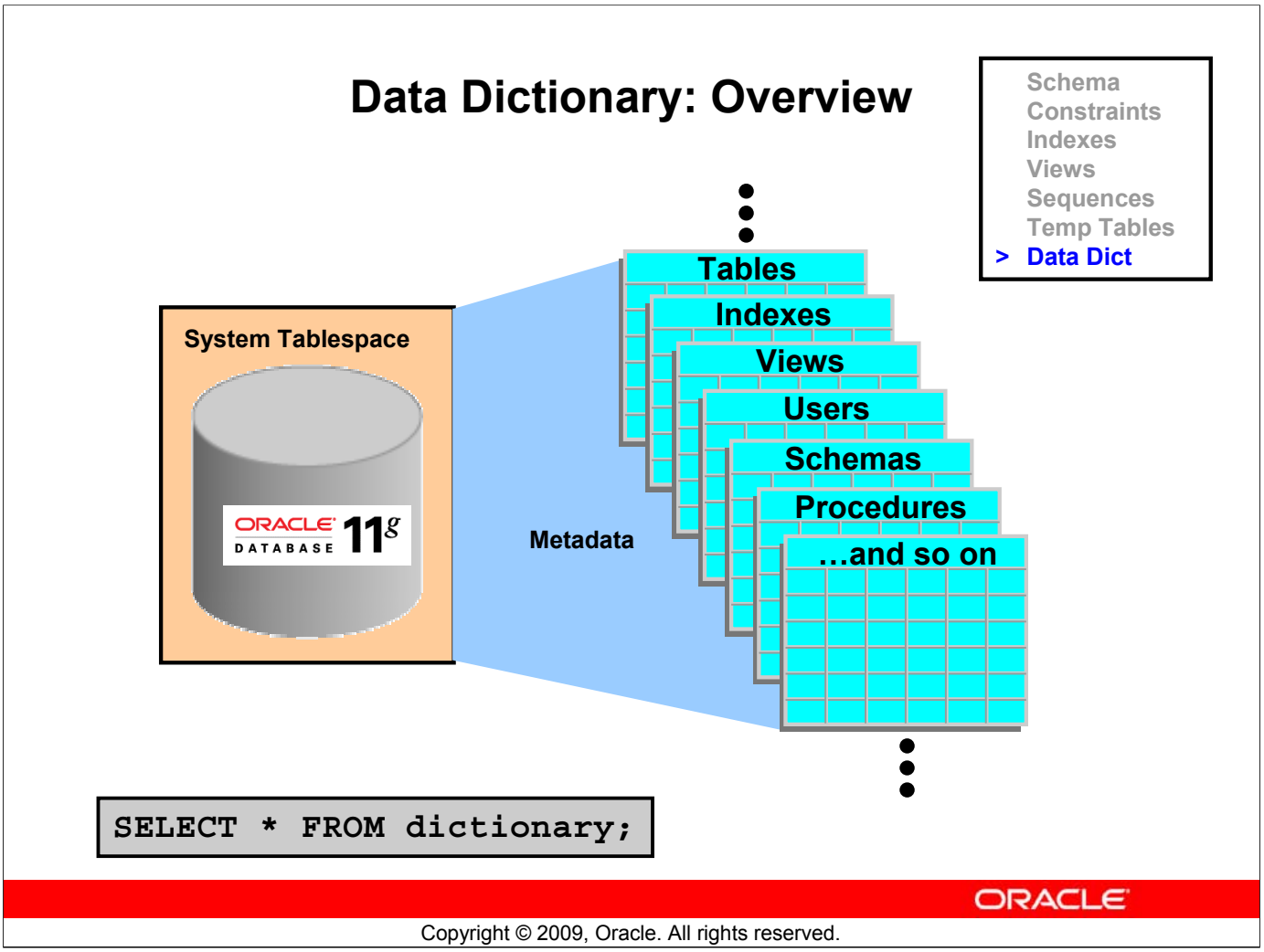
Dynamic Performance Views: Considerations

- These views are owned by the SYS user.
- Different views are available at different times:
  - The instance has been started.
  - The database is mounted.
  - The database is 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.

Dynamic Performance Views: Considerations

Some dynamic views contain data that is not applicable to all states of an instance or database. For example, if an instance has just been started but no database is mounted, you can query V$BGPROCESS to see the list of background processes that are running. But you cannot query V$DATAFILE to see the status of database data files because it is the mounting of a database that reads the control file to find out about the data files associated with a database.

Some V$ views contain information that is similar to information in the corresponding DBA_ views. For example, V$DATAFILE is similar to DBA_DATA_FILES. Note also that V$ view names are generally singular and DBA_ view names are plural.



Data Dictionary: Overview

The Oracle data dictionary is the metadata of the database and contains the names and attributes of all objects in the database. The creation or modification of any object causes an update to the data dictionary that reflects those changes. This information is stored in the base tables that are maintained by the Oracle database, but you access these tables by using predefined views rather than reading the tables directly.

The data dictionary:

- Is used by the Oracle database server to find information about users, objects, constraints, and storage
- Is maintained by the Oracle database server as object structures or definitions are modified
- Is available for use by any user to query information about the database
- Is owned by the `SYS` user
- Should never be modified directly using SQL

Note: The `DICTIONARY` data dictionary view (or the `DICT` synonym for this) contains the names and descriptions of data dictionary tables and views. Use the `DICT_COLUMNS` view to see the view columns and their definitions. For complete definitions of each view, see the *Oracle Database Reference*. There are over 1000 views that reference hundreds of base tables.

Data Dictionary Views

	Who Can Query	Contents	Subset of	Notes
DBA_	DBA	Everything	N/A	May have additional columns meant for DBA use only
ALL_	Everyone	Everything that the user has privileges to see	DBA_ views	Includes user's own objects and other objects the user has been granted privileges to see
USER_	Everyone	Everything that the user owns	ALL_ views	Is usually the same as ALL_ except for the missing OWNER column (Some views have abbreviated names as PUBLIC synonyms.)

ORACLE

Data Dictionary Views

The view prefixes indicate the data (and how much of that data) a given user can see.

The global view of everything is accessed only by users with DBA privileges, using the DBA_ prefix.

The next level of privilege is at the ALL_ prefix level, which represents all objects that the querying user is privileged to see, whether the user owns them or not. For example, if USER_A has been granted access to a table owned by USER_B, then USER_A sees that table listed in any ALL_ view dealing with table names.

The USER_ prefix represents the smallest scope of visibility. This type of view shows only those objects that the querying user owns (that is, those that are present in the user's own schema).

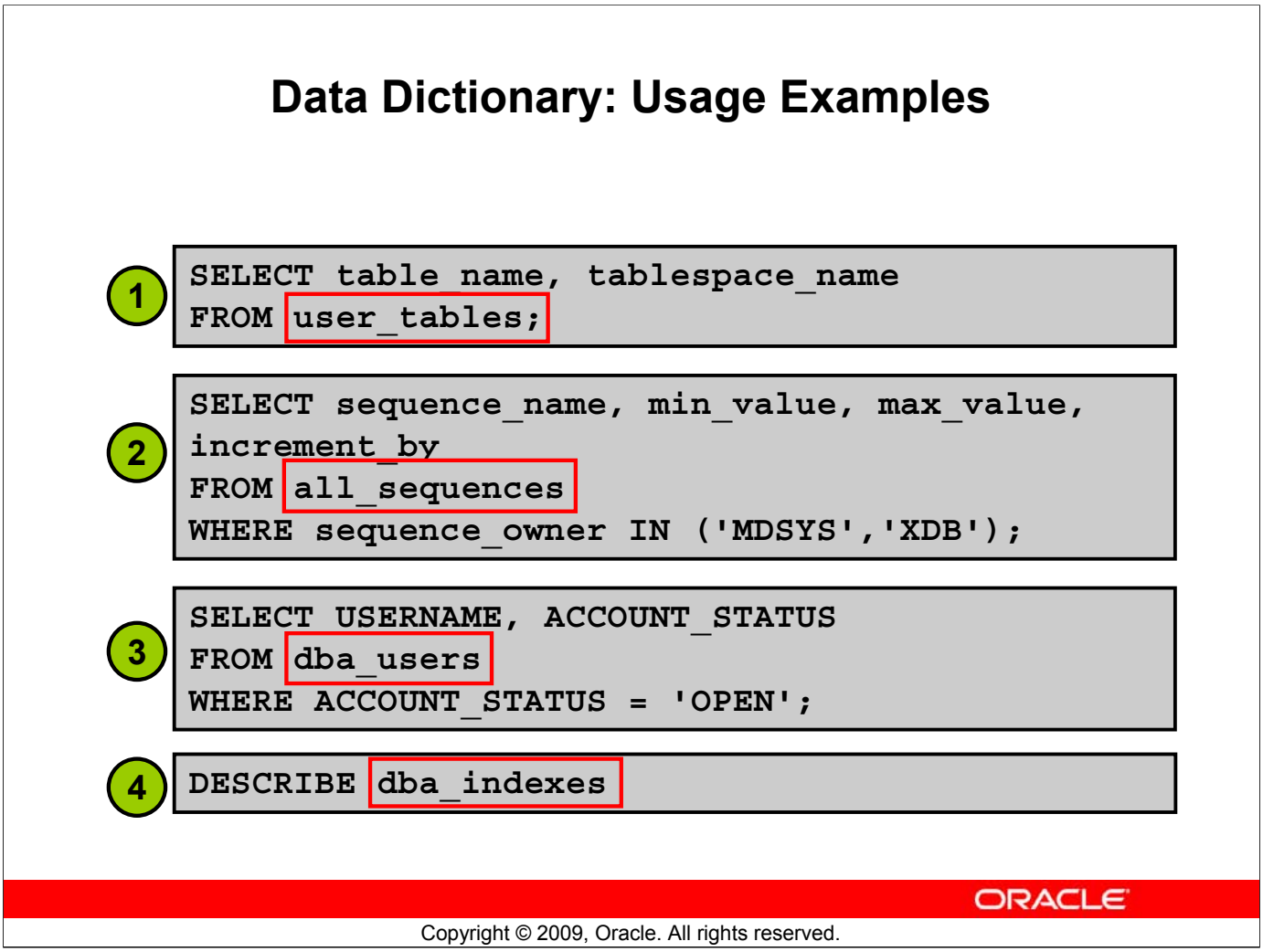
Generally, each view set is a subset of the higher-privileged view set, row-wise and column-wise. Not all views in a given view set have a corresponding view in the other view sets. This is dependent on the nature of the information in the view. For example, there is a DBA_LOCK view, but there is no ALL_LOCK view. This is because only a DBA would have interest in data about locks. Be sure to choose the appropriate view set to meet the need that you have. If you have the privilege to access the DBA views, you still may want to query only the USER version of the view because the results show information on objects that you own and you may not want other objects to be added to your result set.

Data Dictionary Views (continued)

The DBA_ views can only be queried by users with the SYSDBA or SELECT ANY DICTIONARY privilege.

Not all dictionary views start with the prefix DBA_, ALL_, and USER_. The following views or synonyms to views are exceptions to this:

- AUDIT_ACTIONS
- CAT
- CHANGE_PROPAGATIONS
- CHANGE_PROPAGATION_SETS
- CHANGE_SETS
- CHANGE_SOURCES
- CHANGE_TABLES
- CLIENT_RESULT_CACHE_STATS$
- CLU
- COLS
- COLUMN_PRIVILEGES
- DATABASE_COMPATIBLE_LEVEL
- DBMS_ALERT_INFO
- DBMS_LOCK_ALLOCATED
- DICT
- DICTIONARY
- DICT_COLUMNS
- DUAL
- GLOBAL_NAME
- IND
- INDEX_HISTOGRAM
- INDEX_STATS
- LOGSTDBY_UNSUPPORTED_TABLES
- NLS_DATABASE_PARAMETERS
- NLS_INSTANCE_PARAMETERS
- NLS_SESSION_PARAMETERS
- OBJ
- RECYCLEBIN
- RESOURCE_COST
- ROLE_ROLE_PRIVS
- ROLE_SYS_PRIVS
- ROLE_TAB_PRIVS
- SEQ
- SESSION_PRIVS
- SESSION_ROLES
- SM$VERSION
- SYN
- TABLE_PRIVILEGES
- TABS



Data Dictionary: Usage Examples

The example queries in the slide answer the following questions:

1. What are the names of the tables (along with the name of the tablespace where they reside) that have been created in your schema?
2. What is the significant information about the sequences in the database that you have access to?
3. What users in this database are currently able to log in?
4. What are the columns of the `DBA_INDEXES` view? This shows you what information you can view about all the indexes in the database. The following is a partial output of this command:

```
SQL> DESCRIBE dba_indexes
 Name              Null?    Type
 ----------------- -------- -------------
 OWNER             NOT NULL VARCHAR2(30)
 INDEX_NAME        NOT NULL VARCHAR2(30)
 INDEX_TYPE                 VARCHAR2(27)
 TABLE_OWNER       NOT NULL VARCHAR2(30)
 TABLE_NAME        NOT NULL VARCHAR2(30)
```

Quiz

When using Oracle Restart, the server control utility (`srvctl`) must be used instead of SQL*Plus to start and stop a database instance.

1. True
2. False

Answer: 2

Quiz

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

1. USER_TABLES
2. ALL_TABLES
3. DBA_TABLES
4. ANY_TABLES

ORACLE

Answer: 3

Summary

In this lesson, you should have learned how to:

- Start and stop the Oracle database and components
- Use Oracle Enterprise Manager
- Access a database with SQL*Plus
- Modify database initialization parameters
- Describe the stages of database startup
- Describe database shutdown options
- View the alert log
- Access dynamic performance views

ORACLE

Practice 4 Overview: Managing the Oracle Instance

This practice covers the following topics:

- Navigating in Enterprise Manager
- Viewing and modifying initialization parameters
- Stopping and starting the database instance
- Viewing the alert log
- Connecting to the database by using SQL*Plus

ORACLE

5

Managing the ASM Instance

Objectives

After completing this lesson, you should be able to:

- Describe the benefits of using ASM
- Manage the ASM instance
- Create and drop ASM disk groups
- Extend ASM disk groups
- Retrieve ASM metadata by using various utilities

ORACLE

Objectives

This lesson provides a more detailed look at the ASM instance and how to manage it with various utilities.

ASM Benefits for Administrators

ASM eliminates:

- I/O performance tuning
- Data file movements and reorganizations
- File name management
- Logical volume management
- File system management
- Cluster file system management
- Raw device management

ASM significantly reduces:

- Logical Unit Number (LUN) management
  - Fewer, larger LUNs
- Database administrator dependence on system administrator
- Likelihood of errors associated with manual maintenance tasks

ORACLE

ASM Benefits for Administrators

ASM eliminates the need for many tasks that are required in non-ASM storage environments. There include:

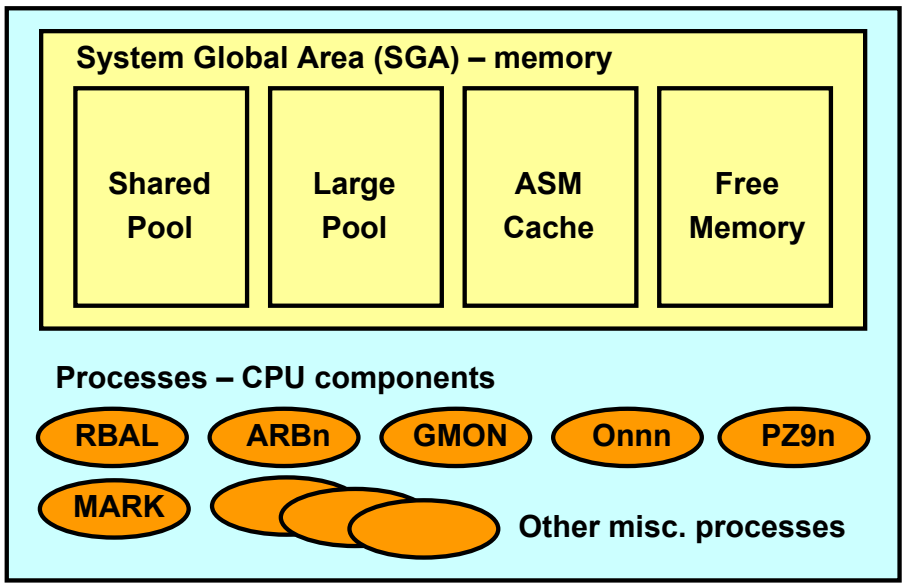
- **I/O performance tuning:** ASM's stripe-and-mirror-everything policy coupled with automatic rebalancing operations means that I/O performance tuning aimed at balancing disk utilization and eliminating disk hot-spots is not required.
- **Data file movements and reorganizations:** Juggling the placement of data files to satisfy performance requirements and space constraints is not required.
- **File name management:** You no longer need to define and enforce a file-naming policy.
- **Logical volume, file system, cluster file system and raw device management:** You no longer require these storage elements.

ASM delivers further benefits by reducing effort in these important areas:

- Logical Unit Number (LUN) management effort is reduced because ASM will typically require fewer, larger LUNs.
- The dependence that often exists between a database administrator and a system administrator is greatly reduced. For example, system administrator involvement is not required to add a new data file or move disk resources from one disk group to another.
- The likelihood of errors associated with manual maintenance tasks is greatly reduced. For example, using a conventional file system a newly created data file may accidentally break a file-naming convention that results in it not being backed up with the rest of the database.

ASM Instance

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



ASM Instance

ORACLE

ASM Instance

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

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

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

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

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

ASM Components: ASM Instance (continued)

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

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

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

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

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

ASM Components: ASM Instance—Primary Processes

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

Process	Description
RBAL	Opens all device files as part of discovery and coordinates the rebalance activity
ARB*n*	One or more slave processes that do the rebalance activity
GMON	Responsible for managing the disk-level activities such as drop or offline and advancing the ASM disk group compatibility
MARK	Marks ASM allocation units as stale when needed
Onnn	One or more ASM slave processes forming a pool of connections to the ASM instance for exchanging messages
PZ9n	One or more parallel slave processes used in fetching data on clustered ASM installation from GV$ views

ORACLE

ASM Components: ASM Instance—Primary Processes

The ASM instance uses dedicated background processes for much of its functionality. The RBAL process coordinates rebalance activity for disk groups in an Automatic Storage Management instance. It performs a global open on Automatic Storage Management disks. The ARB*n* processes perform the actual rebalance data extent movements in an Automatic Storage Management instance. There can be many of these at a time, called ARB0, ARB1, and so on. The GMON process maintains disk membership in ASM disk groups. The MARK process marks ASM allocation units as stale following a missed write to an offline disk. The Onnn processes represent the server side of a client/server connection. These processes will appear the moment the instance is started, and will disappear after that. They form a pool of connections to the ASM instance for exchanging messages and only appear when needed. The PZ9n processes represent one or more parallel slave processes that are used in fetching data when ASM is running in a clustered configuration on more than one machine concurrently.

ASM Instance Initialization Parameters

The ASM instance uses a small subset of the parameters that an Oracle Database instance uses.

```
INSTANCE_TYPE = ASM
ASM_POWER_LIMIT = 1
ASM_DISKSTRING = '/dev/sda1','/dev/sdb*'
ASM_DISKGROUPS = DATA2, FRA
ASM_PREFERRED_READ_FAILURE_GROUPS = DATA.FailGroup2
DIAGNOSTIC_DEST = /u01/app/oracle
LARGE_POOL_SIZE = 12M
REMOTE_LOGIN_PASSWORDFILE = EXCLUSIVE
```

ORACLE

ASM Instance Initialization Parameters

An ASM instance is controlled by a parameter file in the same way as a regular database instance. Parameters commonly set there include:

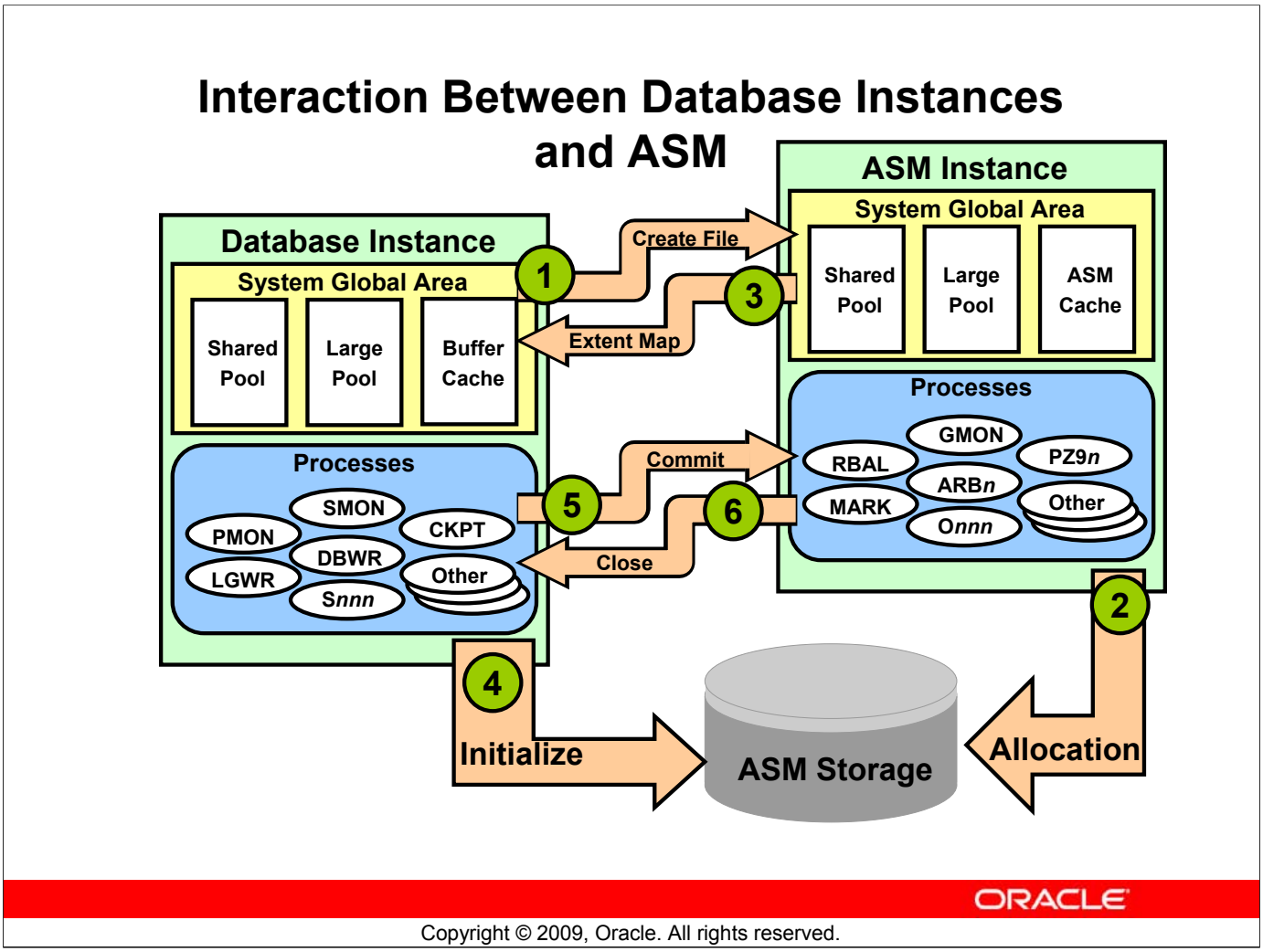
- `INSTANCE_TYPE` should be set to ASM for ASM instances. This is the only parameter that must be defined. For database instances, this is set to the value `RDBMS`.
- `ASM_POWER_LIMIT` controls the speed for a rebalance operation. Values range from 1 through 11, with 11 being the fastest. If omitted, this value defaults to 1.
- `ASM_DISKSTRING` is an operating system–dependent value used by ASM to limit the set of disks considered for discovery. The default value is the null string, and this will be sufficient in most cases. A more restrictive value as shown above may reduce the time required for ASM to perform discovery, and thus improve disk group mount times.
- `ASM_DISKGROUPS` is the list of names of disk groups to be mounted by an ASM instance at startup, or when the `ALTER DISKGROUP ALL MOUNT` command is used. Oracle Restart can mount disk groups if they are listed as dependencies even if they are not listed with the `ASM_DISKGROUPS` parameter. This parameter has no default value.
- `ASM_PREFERRED_READ_FAILURE_GROUPS` specifies the failure groups that contain preferred read disk. This is useful in extended or stretched cluster databases that have mirrored copies of data with one of the copies in close proximity to the server.

ASM Instance Initialization Parameters (continued)

- DIAGNOSTIC_DEST specifies the location of the Automatic Diagnostic Repository (ADR) home. Trace files, alert logs, core files, and incident files can be found under this directory. The default value of this parameter is derived from the value of ORACLE_BASE.
- LARGE_POOL_SIZE specifies (in bytes) the size of the large pool allocation heap. The large pool allocation heap is used in shared server systems for session memory, by parallel execution for message buffers, and by backup processes for disk I/O buffers. The ASM instance makes use of automatic memory management, so this parameter serves as a minimum size that the large pool can be lowered to.
- REMOTE_LOGIN_PASSWORDFILE specifies whether the Oracle software checks for a password file. The default value is EXCLUSIVE.

The eight parameters listed above are the only nondefault parameters created for an ASM instance. The ASM instance differs from a database instance in that not all database parameters are valid for an ASM instance. Approximately 74 of the 344 total database instance parameters can be used with an ASM instance. The remaining parameters not listed on the slide can be set as needed, although default values should be sufficient for most installations.

Note: Automatic memory management is enabled by default on ASM instances, even when the MEMORY_TARGET parameter is not explicitly set. This is the only parameter that you need to set for complete ASM memory management. Oracle Corporation strongly recommends that you use automatic memory management for ASM.



Interaction Between Database Instances and ASM

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

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

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

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

ASM Instance: Dynamic Performance Views

The ASM instance hosts memory-based metadata tables presented as dynamic performance views.

- Accessed by ASM utilities to retrieve metadata-only information using the SQL language
- Contains many dedicated ASM-related views such as:

V$ASM_ALIAS	V$ASM_ATTRIBUTE	V$ASM_CLIENT
V$ASM_DISK	V$ASM_DISK_IOSTAT	V$ASM_DISK_STAT
V$ASM_DISKGROUP	V$ASM_DISKGROUP_STAT	V$ASM_FILE
V$ASM_OPERATION	V$ASM_TEMPLATE	

ORACLE

ASM Instance: Dynamic Performance Views

One of the major functions of any instance is the storage of memory-based metadata tables. These tables start with the prefix `X$` and are generally not documented. A series of dynamic performance views that start with the prefix `V$` are used to display a customized presentation of the data contained in the `X$` memory tables. The information is presented in a read-only fashion available only to administrators with privileges. The information is retrieved from ASM using the SQL language. The above slide lists the most common dynamic performance views that contain ASM-related metadata. There are several hundred additional dynamic performance views, but most of these views will be empty because they require a database instance to mount a database control file. ASM instances do not mount database control files. For a complete list of the dynamic performance views, see the *Oracle Database Reference 11g Release 2 (11.2)* documentation manual.

ASM System Privileges

- An ASM instance does not have a data dictionary, so the only way to connect to ASM is by using these system privileges.

ASM Privilege	Privilege Group (Suggested)	Privilege
SYSASM	OSASM (asmadmin)	Full administrative privilege
SYSDBA	OSDBA (asmdba)	Access to data stored on ASM, and SYSASM in the current release
SYSOPER	OSOPER (asmoper)	Limited privileges to start and stop the ASM instance along with a set of nondestructive ALTER DISKGROUP commands

- The SYS user is automatically created with the SYSASM privilege.

ORACLE

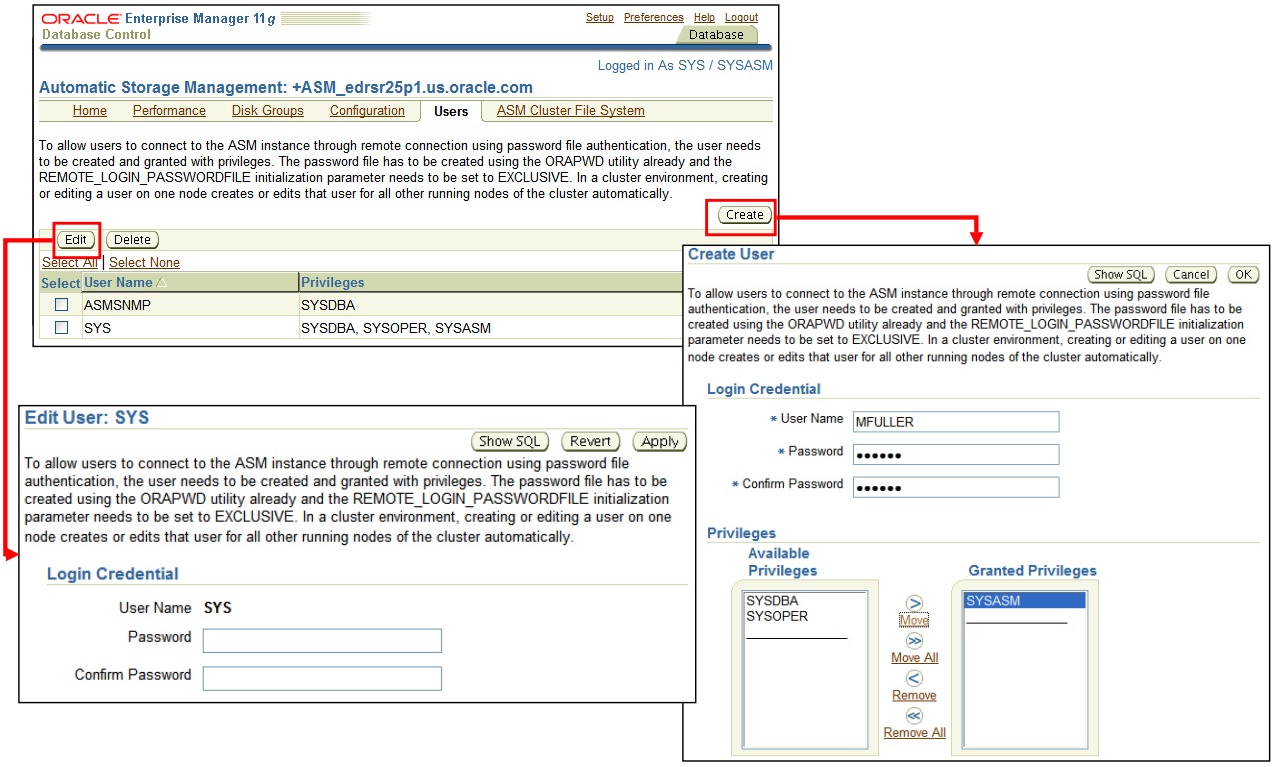
ASM System Privileges

An ASM instance does not have a data dictionary, so the only way to connect to an ASM instance is by using one of three system privileges, SYSASM, SYSDBA, or SYSOPER. The following list introduces these ASM system privileges.

- **SYSASM:** This privilege provides full administrative privilege for the ASM instance.
- **SYSDBA:** This privilege grants access to data stored on ASM, and in the current release, grants the SYSASM administrative privileges.
- **SYSOPER:** This privilege grants the ability to start and stop ASM instances along with a set of nondestructive ALTER DISKGROUP commands. Other commands, such as CREATE DISKGROUP, are not allowed.

When ASM is installed, operating system groups are used to authenticate the SYSASM, SYSDBA, and SYSOPER privileges. These groups are referred to as the OSASM, OSDBA, and OSOPER groups, respectively, by the Oracle Universal Installer (OUI) utility, that is, the OSASM group is the operating system group that is granted the SYSASM privilege. The suggested names of the OSASM, OSDBA, and OSOPER group values are asmadmin, asmdba, and asmoper, respectively. Therefore, SYSASM is the name used by the database, OSASM is the name used by the OUI utility, and asmadmin is the name used by the operating system. All refer to the same group of users. When an ASM instance is first created, sys and asmsnmp are the only ASM users that are defined.

Using Enterprise Manager to Manage ASM Users



Using Enterprise Manager to Manage ASM Users

Enterprise Manager allows you to manage the users who access the ASM instance through remote connection (using password file authentication). These users are reserved exclusively for the ASM instance.

You have this functionality only when you are connected as the `SYSASM` user. It is hidden if you connect as `SYSDBA` or `SYSOPER` users.

- When you click the Create button, the Create User page is displayed.
- When you click the Edit button the Edit User page is displayed.
- By clicking the Delete button, you can delete the created users.

Note: To log in to ASM with the `SYSASM` role, click the Preferences link at the top of the page, followed by the Preferred Credentials link. You will then see ASM in the list of target types. Click the Set Credentials icon beside the ASM target type to define an account and password that has the `SYSASM` privilege. You may need to log out of Database Control and log in before the change takes effect.

Starting and Stopping ASM Instances Using SQL*Plus

Using SQL*Plus to start and stop ASM instances is similar to the way in which you start and stop database instances.

```
$ . oraenv
ORACLE_SID = [orcl] ? +ASM
The Oracle base for ORACLE_HOME=/u01/app/oracle/product/11.2.0/grid is
/u01/app/oracle
$ sqlplus / AS SYSASM
SQL*Plus: Release 11.2.0.1.0 - Production on Wed Jul 8 20:46:46 2009
Copyright (c) 1982, 2009, Oracle.  All rights reserved.
Connected to an idle instance.
SQL> startup
ASM instance started

Total System Global Area  284565504 bytes
Fixed Size                  1336028 bytes
Variable Size             258063652 bytes
ASM Cache                  25165824 bytes
ASM diskgroups mounted
ASM diskgroups volume enabled
SQL> shutdown abort
```


Starting and Stopping ASM Instances Using SQL*Plus

With SQL*Plus you start an ASM instance by using the `STARTUP` command similarly to the way in which you start an Oracle Database instance. When starting an ASM instance, note the following:

- To connect to an ASM instance with SQL*Plus, set the `ORACLE_SID` environment variable to the ASM SID. The default ASM SID for a single instance database is `+ASM`, and the default SID for ASM for an Oracle RAC node is `+ASM`*`node_number`* where *`node_number`* is the number of the node. The `oraenv` script will set the `ORACLE_BASE`, `ORACLE_SID`, `ORACLE_HOME`, and `PATH` variables.
- The initialization parameter file must contain the following entry:

  ```
  INSTANCE_TYPE = ASM
  ```

  This parameter indicates that an ASM instance, not a database instance, is starting.
- When you run the `STARTUP` command, rather than trying to mount and open a database, this command attempts to mount the disk groups specified by the initialization parameter `ASM_DISKGROUPS`. If you have not entered a value for `ASM_DISKGROUPS`, you can later mount disk groups with the `ALTER DISKGROUP...MOUNT` command.

Starting and Stopping ASM Instances Using SQL*Plus (continued)

The following list describes the STARTUP command parameters relevant to ASM.

- **FORCE:** Issues a SHUTDOWN ABORT to the ASM instance before restarting it
- **MOUNT or OPEN:** Mounts the disk groups specified in the ASM_DISKGROUPS initialization parameter. This is the default if no command parameter is specified.
- **NOMOUNT:** Starts up the ASM instance without mounting any disk groups
- **RESTRICT:** Starts up an instance in restricted mode. The RESTRICT clause can be used in combination with the MOUNT, NOMOUNT, and OPEN clauses.
  In restricted mode, database instances cannot use the disk groups. That is, databases cannot open files that are in that disk group. Also, if a disk group is mounted by an instance in restricted mode, then that disk group cannot be mounted by any other instance in the cluster. Restricted mode enables you to perform maintenance tasks on a disk group without interference from clients. Rebalance operations that occur while a disk group is in restricted mode eliminate the lock and unlock extent map messaging that occurs between ASM instances in a clustered environment. This improves the overall rebalance throughput. At the end of a maintenance period, you must explicitly dismount the disk group and remount it in normal mode.

The ASM shutdown process is initiated when you run the SHUTDOWN command in SQL*Plus. Before you run this command, ensure that the ORACLE_SID and ORACLE_HOME environment variables are set so that you can connect to the ASM instance.

Oracle strongly recommends that you shut down all database instances that use the ASM instance before attempting to shut down the ASM instance.

The following list describes the SHUTDOWN command parameters relevant to ASM.

- **NORMAL:** ASM waits for any in-progress SQL to complete before dismounting all of the disk groups and shutting down the ASM instance. Before the instance is shut down, ASM waits for all of the currently connected users to disconnect from the instance. If any database instances are connected to the ASM instance, then the SHUTDOWN command returns an error and leaves the ASM instance running. NORMAL is the default shutdown mode.
- **IMMEDIATE or TRANSACTIONAL:** ASM waits for any in-progress SQL to complete before dismounting all of the disk groups and shutting down the ASM instance. ASM does not wait for users currently connected to the instance to disconnect. If any database instances are connected to the ASM instance, then the SHUTDOWN command returns an error and leaves the ASM instance running.
- **ABORT:** The ASM instance immediately shuts down without the orderly dismount of disk groups. This causes recovery to occur upon the next ASM startup. If any database instance is connected to the ASM instance, then the database instance aborts.

Note: The NORMAL, IMMEDIATE, and TRANSACTIONAL forms of shutdown do not apply when there are connected RDBMS instances. The following error will be returned:

```
ORA-15097: cannot SHUTDOWN ASM instance with connect
RDBMS instance
```

Starting and Stopping ASM Instances Using `srvctl`

The Server Control utility (`srvctl`) can be used to start and stop ASM instances.

```
$ . oraenv
ORACLE_SID = [orcl] ? +ASM
The Oracle base for
    ORACLE_HOME=/u01/app/oracle/product/11.2.0/grid is
    /u01/app/oracle
$ srvctl start asm -o mount
$ srvctl stop asm -f
```

The Sever Control utility (`srvctl`) can be used to check the status of ASM instances.

```
$ srvctl status asm
ASM is running on edrsr25p1
```

ORACLE

Starting and Stopping ASM Instances Using `srvctl`

The Server Control utility (`srvctl`) can be used to start and stop ASM instances along with other resources managed by the Grid Infrastructure. The `srvctl` utility can be found under both the `ORACLE_HOME/bin` location for Grid infrastructure and the `ORACLE_HOME/bin` location of the database installation. You should use the `srvctl` utility located in the Grid Infrastructure `ORACLE_HOME` when managing ASM, listeners, or Oracle restart. The `srvctl` utility can be used to control ASM in the following ways:

- Start an ASM instance.
  `srvctl start asm [-o <start_option>]`
  `<start_option>` is one of the valid instance startup options (`FORCE`, `MOUNT`, `OPEN`, `NOMOUNT` or `RESTRICT`) (optional)
- Stop an ASM instance.
  `srvctl stop asm [-o <stop_option>] -f`
  `<stop_option>` is one of the valid instance shutdown options (`NORMAL`, `IMMEDIATE`, `TRANSACTIONAL` or `ABORT`) (optional), and –f is to force
- Report the status of an ASM instance.
  `srvctl status asm`

Starting and Stopping ASM Instances Using `asmcmd`

The `asmcmd` utility provides a command-line interface to ASM without using the SQL language.

```
$ . oraenv
ORACLE_SID = [orcl] ? +ASM
The Oracle base for ORACLE_HOME=/u01/app/oracle/product/11.2.0/grid is
/u01/app/oracle
$ asmcmd
Connected to an idle instance.
ASMCMD> startup
ASM instance started

Total System Global Area  284565504 bytes
Fixed Size                  1336028 bytes
Variable Size             258063652 bytes
ASM Cache                  25165824 bytes
ASM diskgroups mounted
ASM diskgroups volume enabled
ASMCMD> shutdown --abort
ASM instance shut down
Connected to an idle instance.
```


Starting and Stopping ASM Instances Using `asmcmd`

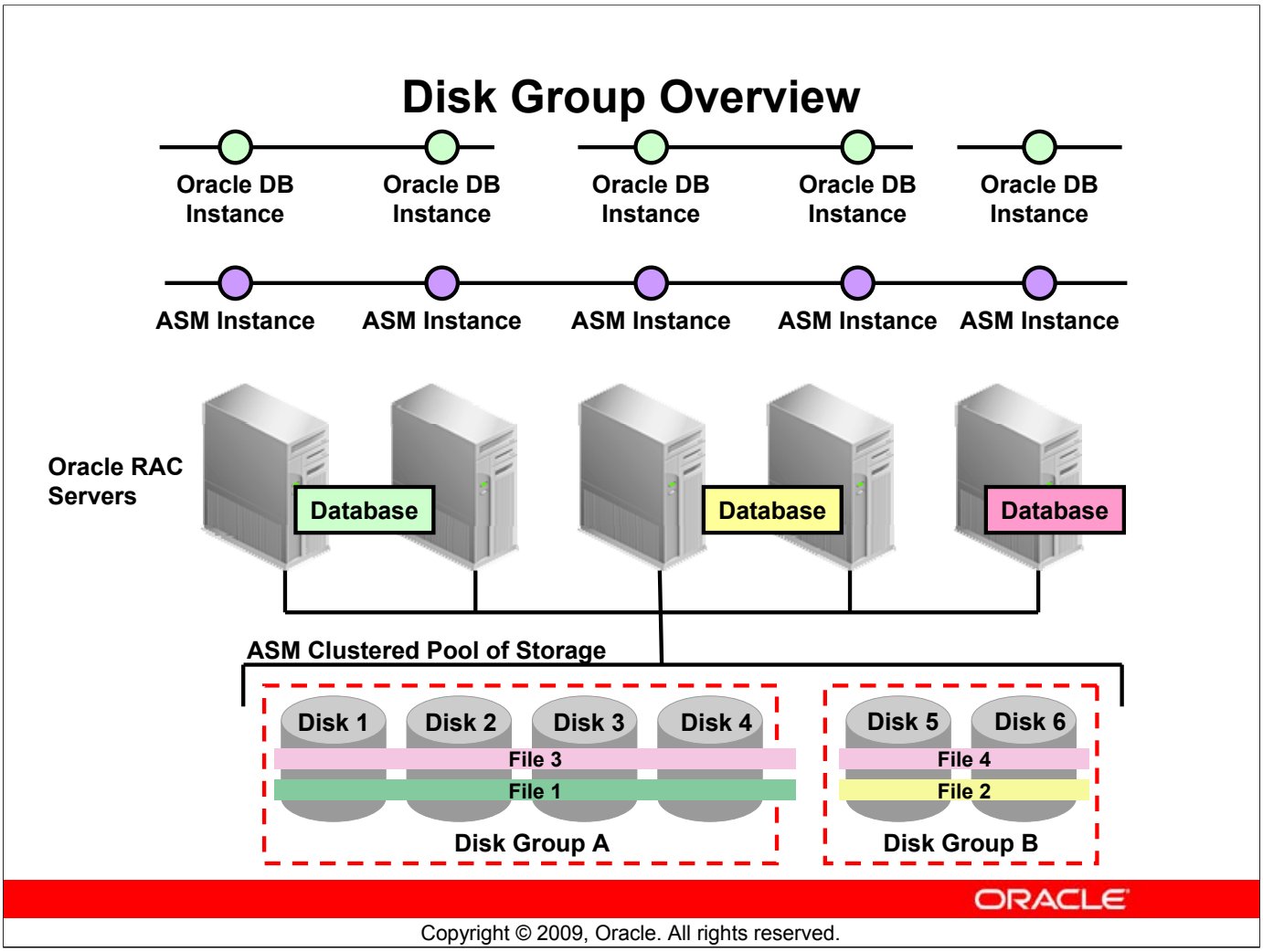
ASM metadata for administration is found inside the dynamic performance views that are contained within the ASM instance. These views are usually accessed with an ASM utility using the SQL language. The requirement for knowledge of the SQL language increases the learning curve for mastering ASM, and SQL language training is not a normal requirement in the learning path of a systems administrator or storage administrator. The `asmcmd` utility provides a pseudo-shell-like environment that accepts UNIX-style syntax for common ASM administration tasks. It can be used to manage Oracle ASM instances, disk groups, file access control for disk groups, files and directories within disk groups, templates for disk groups, and volumes.

You can use the `asmcmd` utility to start up and shut down your ASM instance. The supported startup options include:

- `--nomount` (To start the ASM instance without mounting disk groups)
- `--restrict` (To start the ASM instance and restrict database usage)
- `--pfile <pfile.ora>` (To start the ASM instance with a custom pfile)

The supported shutdown options include:

- `--immediate` (Performs shutdown immediately)
- `--abort` (Aborts all existing operations)



Disk Group Overview

A disk group is a logical grouping of one or more disks that ASM manages as a collection. Each disk group contains the metadata associated with itself. You can think of an ASM disk group as conceptually similar to a logical volume in a typical storage area network.

Files are allocated from the space inside a disk group. The content of files that are stored in a disk group are evenly distributed, or striped, across the disks in the disk group to eliminate hot spots and to provide uniform performance across the disks. Each ASM file is completely contained within a single disk group. However, a disk group can contain files belonging to several databases and a single database can use different files from multiple disk groups.

A key attribute of a disk group is its redundancy setting. There are three possible disk group redundancy settings:

- External redundancy, where ASM does not provide any mirroring and the disks are assumed to be highly reliable.
- Normal redundancy, where ASM supports 2-way mirroring by default to assure data integrity for less reliable storage.
- High redundancy, where ASM supports 3-way mirroring by default for even greater assurance of data integrity.

ASM supports the creation of up to 63 disk groups; however, for most installations you rarely need more than a few.

ASM Disks

ASM disks:

- Are the storage devices provisioned to ASM disk groups
- Are accessed through normal O/S interfaces
- Must be read and write accessible by the ASM owner
- Must be accessible by all nodes in a cluster
- May have different O/S names or paths on different nodes
- May be:
  - An entire physical disk or partition of a physical disk
  - A disk or partition from a storage array
  - Logical volumes (LV) or logical units (LUN)
  - Network-attached files (NFS)

ORACLE

ASM Disks

ASM disk groups contain one or more ASM disks.

ASM disks must be read and write accessible by the ASM owner from all nodes in the cluster.

An ASM disk can be a partition on a disk, but Oracle strongly advises against sharing the physical disks used by ASM with other applications. This is because the I/O performance of the ASM disks within a disk group should be similar and consistent for the ASM stripe and mirror policy to work best. Sharing devices between ASM and other applications makes it difficult to assure similar and consistent disk performance.

It is not necessary for the O/S device names of ASM disks on separate nodes in a cluster to be the same. ASM identifies members of a disk group by reading the headers on ASM disks.

Generally, ASM disks are raw LUNs from a storage array presented to ASM. In addition, ASM disks can also be files on a remote NFS filer.

Allocation Units

ASM disks are divided into allocation units (AU):

- AU size is configurable at disk group creation.
- Default AU size is 1 MB:
  - Small enough to be cached by database and large enough for efficient sequential access
- Allowable AU sizes:
  - 1, 2, 4, 8, 16, 32, or 64 MB
  - Large AUs may be useful in very large database (VLDB) scenarios or when using specialized storage hardware

ORACLE

Allocation Units

Within an ASM disk, space is divided into allocation units (AU). The default AU size is one megabyte, which is small enough to not become a hot spot, but large enough to provide efficient sequential access. You can set the AU size when you create a disk group. You cannot change the AU size for a disk group. Larger AU sizes may be useful in very large database (VLDB) scenarios or when using specialized storage hardware. If an AU is constantly accessed, then the database kernel caches it for more efficient access.

ASM Files

ASM files:

- Are a collection of ASM extents composed of AUs
  - Variable sized extents support large files
- Appear as normal files to the database kernel
- Have file names that start with '+'
  - For example, `+DATA/orcl/datafile/system.256.689832921`
- May be associated with an optional alias file name
  - For example, `+DATA/dbfiles/mydb/system01.dbf`
- Are evenly distributed across disks in a disk group
- Are mirrored according to the policies defined in the disk group

ORACLE

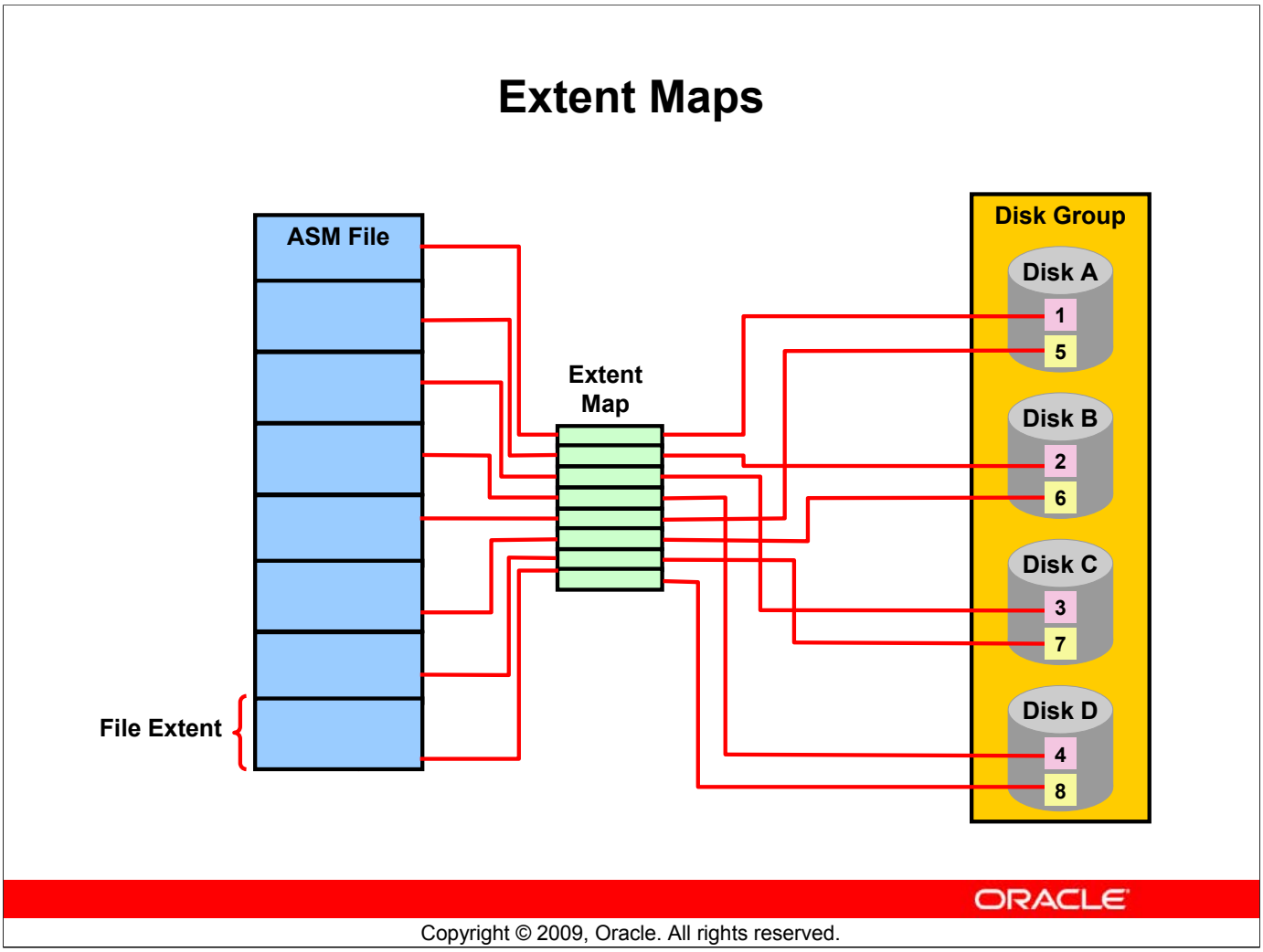
ASM Files

ASM exposes a set of files for use by clients of ASM. An ASM file is composed of a set of allocation units and appears as a normal file to the database kernel.

Each ASM file has a unique system-generated name. The slide shows an example of a fully-qualified ASM file name. The fully qualified ASM file name represents a hierarchy beginning with a concatenation of a plus sign with the disk group name. Following the disk group name is the database name and file type. The final element is composed of a tag name, file number, and incarnation number. An alias can optionally be created to give administrators a more user-friendly means of referring to an ASM file.

Files are evenly spread across the ASM disks in a disk group using the policy of stripe and mirror everything (SAME).

ASM natively supports most database-related file types, such as data files, log files, control files, RMAN backups, and others. Prior to Oracle Database 11*g* Release 2, ASM only supported Oracle database–related files and could not be used to store and manage ASCII trace files and alert logs, Oracle binaries, the Oracle Cluster Registry (OCR), and the cluster-voting disk. Oracle Database 11*g* Release 2 removes this restriction by providing the means to run a general purpose file system over ASM.



Extent Maps

ASM keeps track of a file's layout with metadata called an extent map. An extent map is a table that maps data extents in a file to allocation units on disk.

The relationship between file extents and allocation units is as follows. An extent contains:

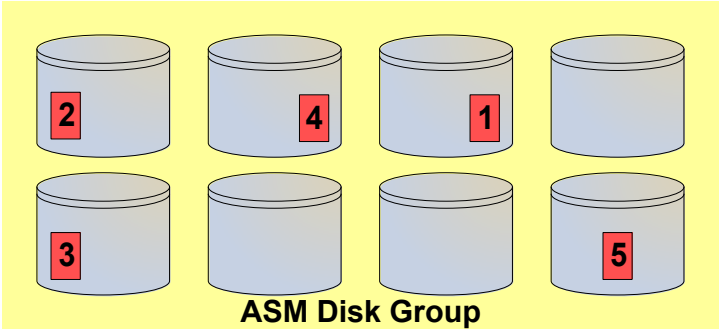
- One AU for the first 20,000 extents (0–19999)
- 4 AUs for the next 20,000 extents (20000–39999)
- 16 AUs for extents above 40,000

Variable-sized extents, coupled with large allocation units, can be used to accommodate very large ASM files.

Striping Granularity

ASM separates striping for load balance and striping for latency:

- Coarse-grain striping concatenates allocation units for load balancing.
  - For example:



ORACLE

Striping Granularity

In ASM, striping has two primary purposes:

- Load balance I/O across all the disks in a disk group.
- Improve I/O latency.

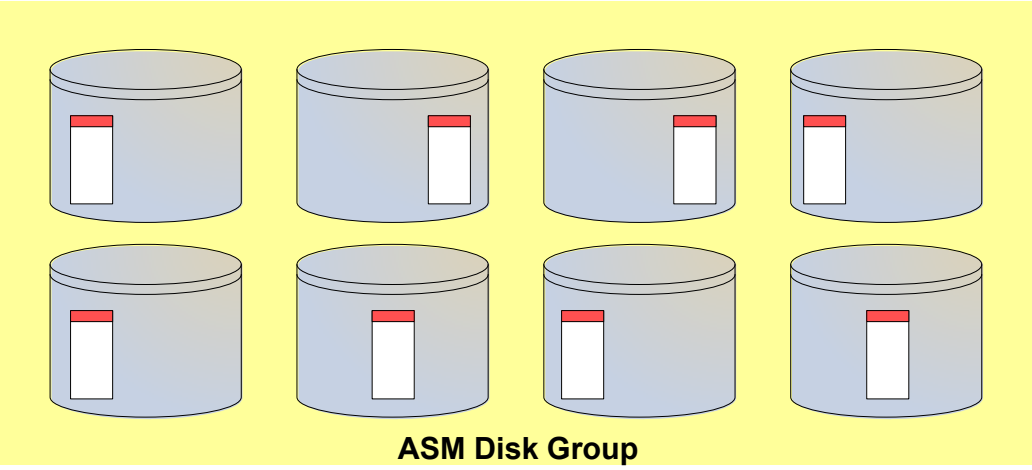
Coarse-grain striping spreads allocation units across the disks in a disk group. This is what provides load balancing for disk groups. When a file is allocated, ASM spreads allocation units evenly across all of the disks. Sometimes the distribution cannot be perfectly even, but over time it will tend to be nearly equal. The above diagram shows a file with five allocation units striped across five disks in an external redundancy disk group containing eight disks in total.

For the first 20,000 extents, the extent size is equal to the AU size. After 20,000 extents and up to 40,000 extents, the extent sets are always allocated 8 at a time with the extent size equal to 4*AU size. If the AU size is 1 MB, this means the ASM file will grow 64 MB at a time (8 * 4 * 1 MB). If the file is coarse-grained striped then it is striped across the 8 extent sets with stripes of 1 AU. Striping is always done at the AU level, not at the extent level. Thus every AU of a coarse-grained file is on a different disk than the previous AU of that file no matter how large the file. After 40,000 extents, the extents are still allocated 8 at a time, but with an extent size equal to 16*AU size.

Fine-Grained Striping

Fine-grain striping puts 128 KB stripe units across groups of allocation units to improve latency.

- Disk group with 8 disks and external redundancy
- Default AU size of 1 MB in use
- First 1 MB extent written as 128 KB stripes across 8 AUs



Fine-Grained Striping

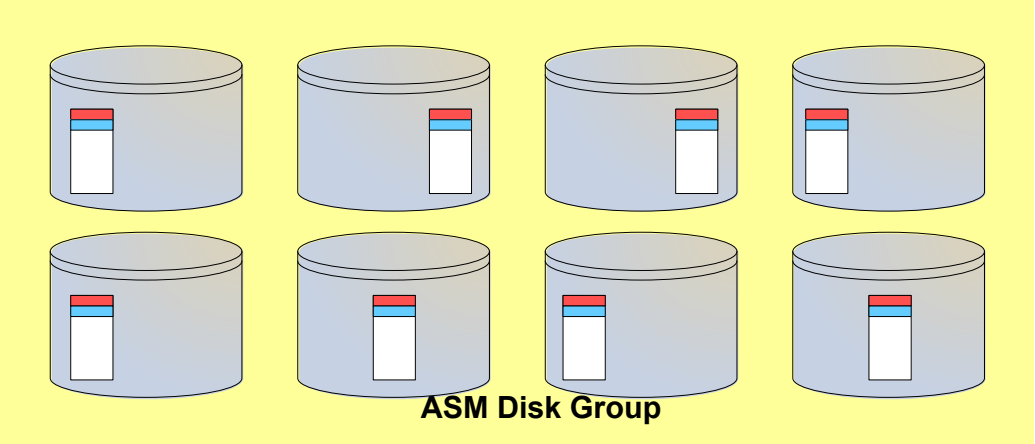
Fine-grain striping splits data extents into 128 KB chunks and it is provided to improve latency for certain types of files by spreading the load for each extent across a number of disks. Fine grain striping is used by default for control files and online redo log files.

The diagram on this page shows how fine grain striping works. In this example, the first 1 MB extent of a new file ends up occupying the first 128 KB of 8 different allocation units spread across the eight disks in the disk group. Consequently, a one-megabyte read or write is spread across eight disks instead of one.

Fine-Grained Striping

Example:

- Disk group with 8 disks and external redundancy
- Default AU size of 1 MB in use
- Next 1 MB extent written as 128 KB stripes across the same 8 allocation units until they are full



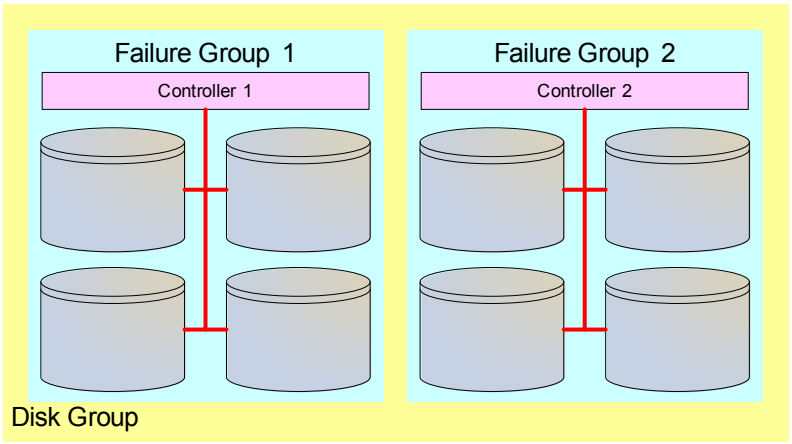
ORACLE

Fine-Grained Striping (continued)

Continuing the previous example, the next one megabyte extent of space is spread across the second 128 KB of each of the same allocation units. This pattern continues until the first set of allocation units is filled and another set is allocated.

ASM Failure Groups

- A set of disks sharing a common resource whose failure needs to be tolerated
- Mirrored extent copies stored in separate failure groups
- Storage hardware dictates failure group boundaries
  - Example based on isolating disk controllers:



ORACLE

ASM Failure Groups

Within a disk group, disks may be collected into failure groups. Failure groups are the way a storage or database administrator specifies the hardware boundaries that ASM mirroring operates across.

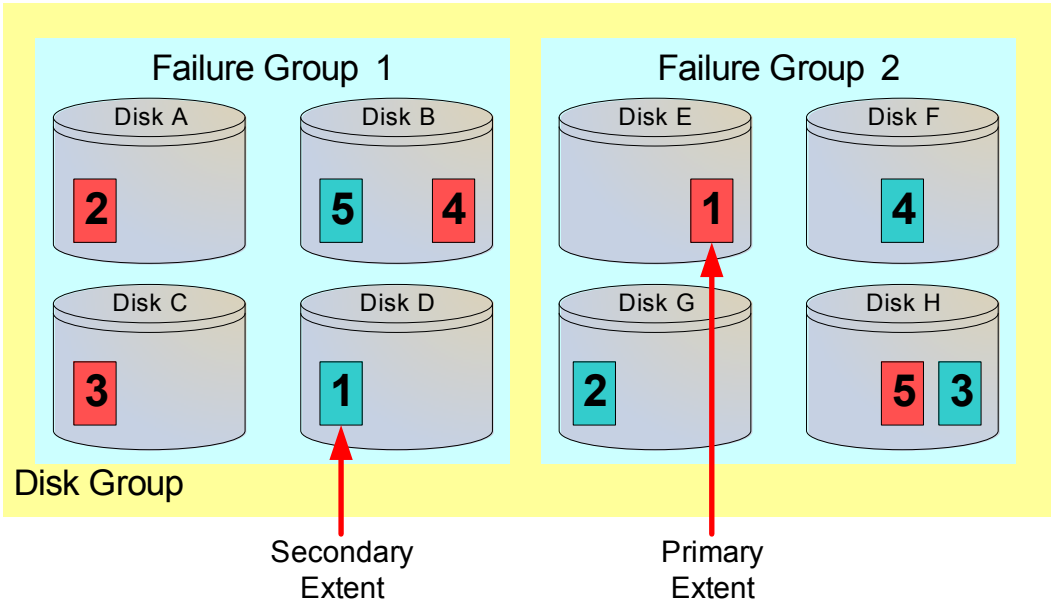
For example, all the disks attached to a single disk controller could be specified to be within a common failure group. This would lead to file extents being mirrored on disks connected to separate controllers. Additionally, an administrator can configure ASM to choose a default failure group policy. The default policy is that each individual disk is in its own failure group.

You can group disks into failure groups using whatever criteria you need. Failure groups can be used to protect from the failure of individual disks, disk controllers, I/O network components, and even entire storage systems. Typically, an administrator would analyze their storage environment and would organize failure groups to mitigate specific failure scenarios.

It is up to the database or storage administrator to determine what is the best failure group configuration for his or her installation.

Stripe and Mirror Example

Normal redundancy disk group with eight disks in total, spread across two failure groups.



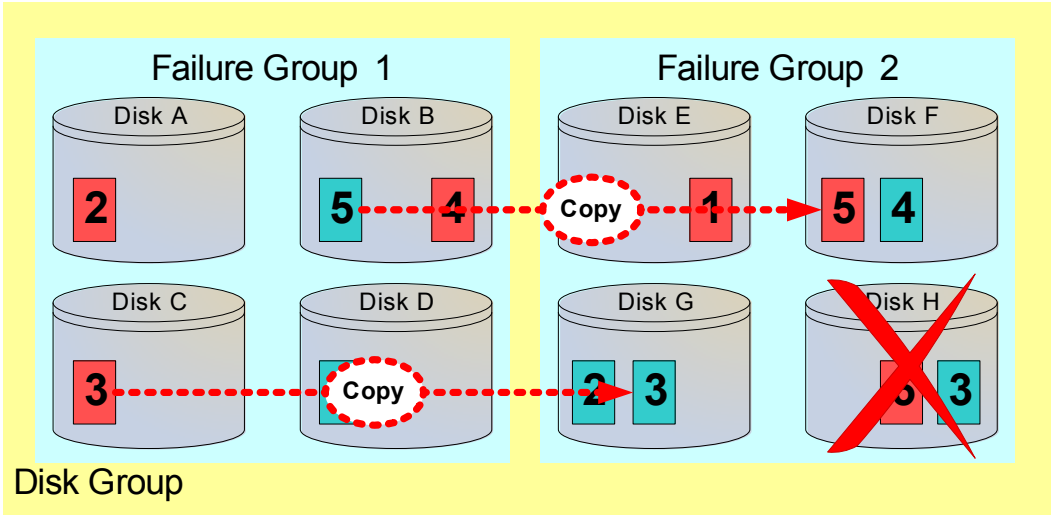
Stripe and Mirror Example

The diagram on this page illustrates an example showing striping and mirroring in a normal redundancy disk group. The red blocks represent a file with five extents being striped across five of the eight disks in the disk group. The blue blocks represent mirrored copies of the file's extents. Note that regardless of the distribution of extents across the various disks and failure groups, each extent has exactly one copy in each failure group.

When a file is allocated, the primary extents are allocated for performance and the secondary copies are allocated for integrity. For this reason all database reads are made against the primary extents by default.

Failure Example

If disk H fails, then the extents it contained are re-created on surviving disks from surviving mirrors.



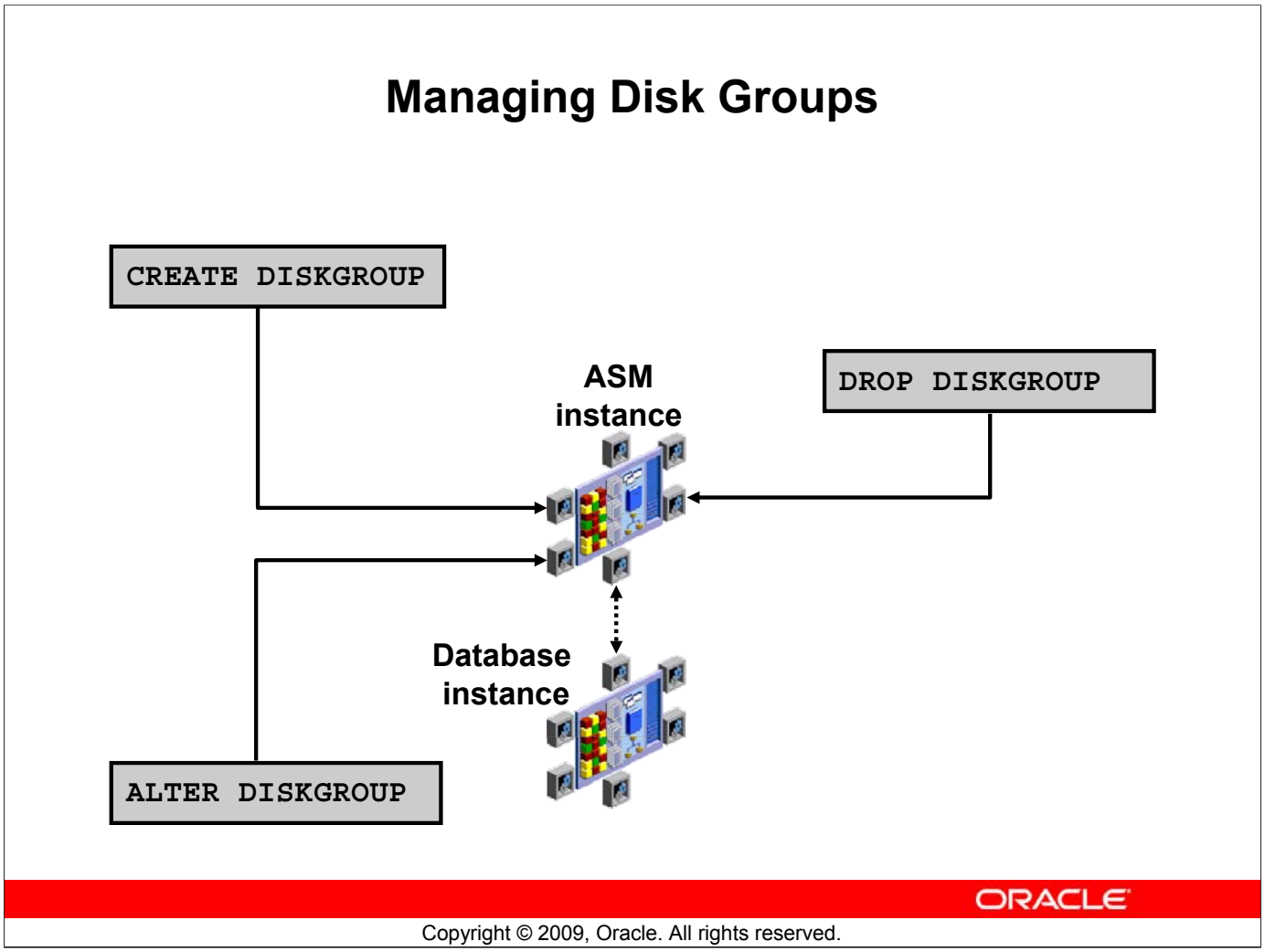
Failure Example

Extending the example on the previous page, imagine that disk H fails and the data it contains is no longer accessible. This failure requires that all the extents on the failed disk need to be recovered and copied to another disk.

Extents three and five are copied from the surviving copy to a free region on another disk in the same failure group. In this example, extent five is copied from disk B to Disk F and extent three is copied from Disk C to Disk G.

The last step after a disk fails is that ASM drops the failed disk from the disk group.

The removal of a disk triggers essentially the same process; however, in that case the extents on the disk being removed are first copied to an available alternative location.



Managing Disk Groups

The main goal of an ASM instance is to manage disk groups and protect their data. ASM instances also communicate file layout to database instances. In this way, database instances can directly access files stored in disk groups.

There are several disk group administrative commands. They all require the `SYSASM` or `SYSDBA` privilege and must be issued from an ASM instance.

You can add new disk groups. You can also modify existing disk groups to add new disks, remove existing ones, and perform many other operations. You can remove existing disk groups.

Creating and Dropping Disk Groups Using SQL*Plus

```
$ . oraenv
ORACLE_SID = [orcl] ? +ASM
The Oracle base for ORACLE_HOME=/u01/app/oracle/product/11.2.0/grid is
/u01/app/oracle
$ sqlplus / AS SYSASM
SQL*Plus: Release 11.2.0.1.0 - Production on Wed Jul 8 20:46:46 2009
Copyright (c) 1982, 2009, Oracle.  All rights reserved.
..
SQL> CREATE DISKGROUP dgroupA NORMAL REDUNDANCY
FAILGROUP controller1 DISK
   '/devices/A1' NAME diskA1 SIZE 120G FORCE,
   '/devices/A2',
FAILGROUP controller2 DISK
   '/devices/B1',
   '/devices/B2';
```

```
SQL> DROP DISKGROUP dgroupA INCLUDING CONTENTS;
```

ORACLE

Creating and Dropping Disk Groups

Assume that ASM disk discovery identified the following disks in the `/dev` directory: `A1`, `A2`, `B1`, and `B2`. Also, assume that disks `A1` and `A2` are on a separate disk controllers from disks `B1` and `B2`. The first example in the slide illustrates how to configure a disk group called `DGROUPA` with two failure groups: `CONTROLLER1` and `CONTROLLER2`.

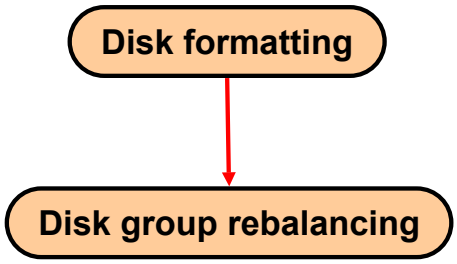
The example also uses the default redundancy attribute, `NORMAL REDUNDANCY`, for the disk group. You can optionally provide a disk name and size for the disk. If you do not supply this information, ASM creates a default name and attempts to determine the size of the disk. If the size cannot be determined, an error is returned. `FORCE` indicates that a specified disk should be added to the specified disk group even though the disk is already formatted as a member of an ASM disk group. Using the `FORCE` option for a disk that is not formatted as a member of an ASM disk group returns an error.

As shown by the second statement in the slide, you can delete a disk group along with all its files. To avoid accidental deletions, the `INCLUDING CONTENTS` option must be specified if the disk group still contains any files besides internal ASM metadata. The disk group must be mounted in order for it to be dropped. After ensuring that none of the disk group files are open, the group and all its drives are removed from the disk group. Then the header of each disk is overwritten to eliminate the ASM formatting information.

Adding Disks to Disk Groups

```
ALTER DISKGROUP dgroupA ADD DISK
  '/dev/sde1' NAME A5,
  '/dev/sdf1' NAME A6,
  '/dev/sdg1' NAME A7,
  '/dev/sdh1' NAME A8;
```

```
ALTER DISKGROUP dgroupA ADD DISK '/devices/A*';
```



ORACLE

Adding Disks to Disk Groups

This example shows how to add disks to a disk group. You execute an `ALTER DISKGROUP ADD DISK` command to add the disks. The first statement adds four new disks to the `DGROUPA` disk group.

The second statement demonstrates the interactions of discovery strings. Consider the following configuration:
`/devices/A1` is a member of disk group `DGROUPA`.
`/devices/A2` is a member of disk group `DGROUPA`.
`/devices/A3` is a member of disk group `DGROUPA`.
`/devices/A4` is a candidate disk.
The second command adds `A4` to the `DGROUPA` disk group. It ignores the other disks, even though they match the discovery string, because they are already part of the `DGROUPA` disk group. The diagram shows that, when you add a disk to a disk group, the ASM instance ensures that the disk is addressable and usable. The disk is then formatted and rebalanced. The rebalance process is time consuming because it moves extents from all files onto the new disk.

Note: Rebalancing does not block any database operations. The main impact that a rebalance process has is on the I/O load on the system. The higher the power of the rebalance, the more I/O load it puts on the system. Thus less I/O bandwidth is available for database I/Os.

Miscellaneous ALTER Commands

Remove a disk from dgroupA:

```
ALTER DISKGROUP dgroupA DROP DISK A5;
```

Add and drop a disk in a single command:

```
ALTER DISKGROUP dgroupA
 DROP DISK A6
 ADD FAILGROUP controller3
        DISK '/dev/sdi1' NAME A9;
```

Cancel a disk drop operation:

```
ALTER DISKGROUP dgroupA UNDROP DISKS;
```

ORACLE

Miscellaneous ALTER Commands

The first statement in the slide shows how to remove one of the disks from the DGROUPA disk group. The second statement shows how you can add and drop a disk in a single command. The big advantage in this case is that rebalancing is not started until the command completes. The third statement shows how to cancel a disk drop operation. The UNDROP command operates only on pending drops of disks; it has no effect on drops that have completed.

The following statement rebalances the DGROUPB disk group, if necessary:

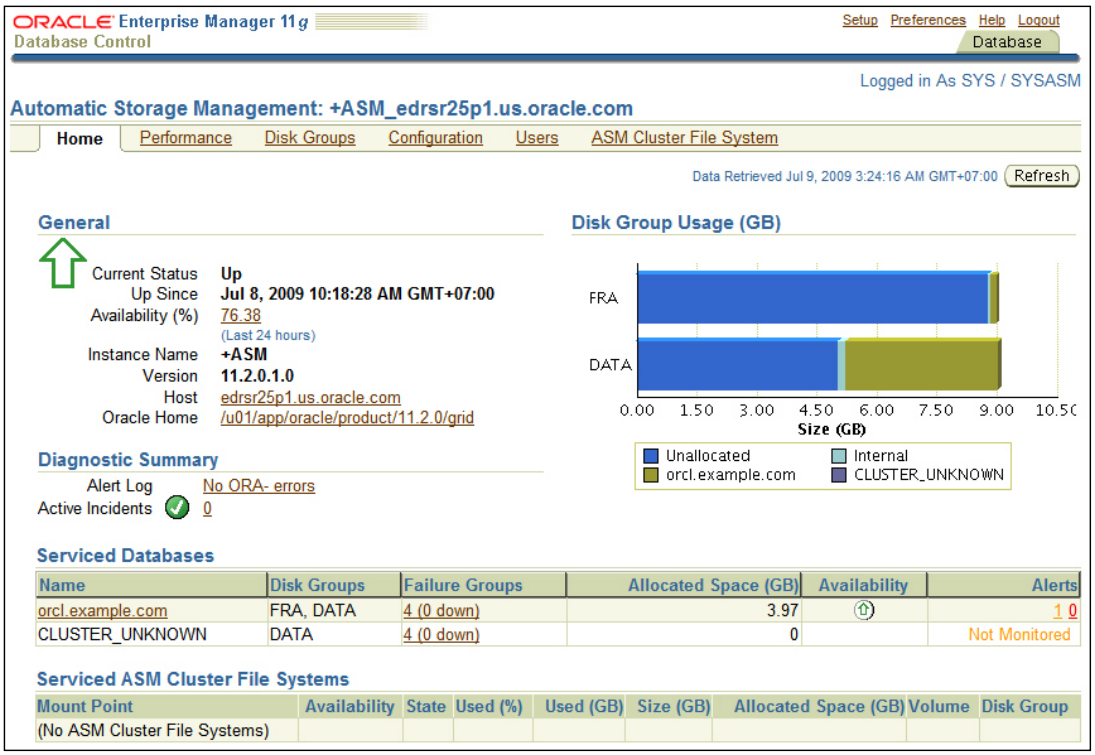
```
ALTER DISKGROUP dgroupB REBALANCE POWER 5;
```

This command is generally not necessary because it is automatically done as disks are added, dropped, or resized. However, it is useful if you want to use the POWER clause to override the default speed defined by the initialization parameter ASM_POWER_LIMIT. You can change the power level of an ongoing rebalance operation by reentering the command with a new level. A power level of zero causes rebalancing to halt until the command is either implicitly or explicitly reinvoked. The following statement dismounts DGROUPA:

```
ALTER DISKGROUP dgroupA DISMOUNT;
```

The MOUNT and DISMOUNT options allow you to make one or more disk groups available or unavailable to the database instances. The ability to manually unmount and mount is useful in a clustered ASM environment supporting a single instance, when that instance is failed over to a different node.

ASM Management Using Enterprise Manager



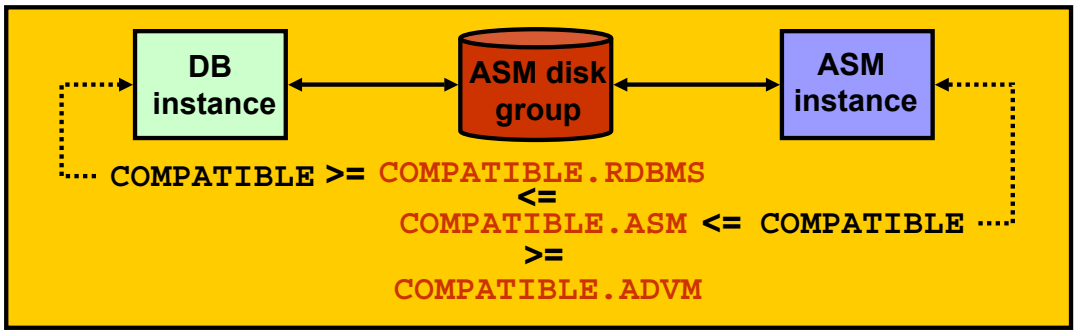
ORACLE

ASM Management Using Enterprise Manager

Oracle Enterprise Manager (EM) is Oracle's family of management tools spanning database, middleware, applications, networks, IT infrastructure and more. Enterprise Manager delivers a browser-based environment that provides a point-and click alternative for common ASM administration tasks.

ASM Disk Group Compatibility

- Compatibility of each disk group is separately controllable:
  - ASM compatibility controls ASM metadata on-disk structure.
  - RDBMS compatibility controls minimum consumer client level.
  - ADVM compatibility determines whether a disk group can contain Oracle ASM volumes.
- Setting disk group compatibility is irreversible.



ASM Disk Group Compatibility

There are three kinds of compatibility applicable to ASM disk groups: dealing with the persistent data structures that describe a disk group, the capabilities of the clients (consumers of disk groups), and the ability to contain volumes in a disk group. These attributes are called *ASM compatibility* , *RDBMS compatibility*, and *ADVM compatibility*, respectively. The compatibility of each disk group is independently controllable. This is required to enable heterogeneous environments with disk groups from both Oracle Database 10*g* and Oracle Database 11*g*. These three compatibility settings are attributes of each ASM disk group:

- RDBMS compatibility refers to the minimum compatible version of the RDBMS instance that would allow the instance to mount the disk group. This compatibility dictates the format of messages that are exchanged between the ASM and database (RDBMS) instances. An ASM instance has the capability to support different RDBMS clients running at different compatibility settings. The database-compatible version setting of each instance must be greater than or equal to the RDBMS compatibility of all disk groups used by that database. Database instances are typically run from a different Oracle home than the ASM instance. This implies that the database instance may be running a different software version than the ASM instance. When a database instance first connects to an ASM instance, it negotiates the highest version that they both can support.

ASM Disk Group Compatibility (continued)

The compatibility parameter setting of the database, software version of the database, and the RDBMS compatibility setting of a disk group determine whether a database instance can mount a given disk group.

- ASM compatibility refers to the persistent compatibility setting controlling the format of data structures for ASM metadata on disk. The ASM compatibility level of a disk group must always be greater than or equal to the RDBMS compatibility level of the same disk group. ASM compatibility is concerned only with the format of the ASM metadata. The format of the file contents is up to the database instance. For example, the ASM compatibility of a disk group can be set to 11.0 whereas its RDBMS compatibility could be 10.1. This implies that the disk group can be managed only by ASM software whose software version is 11.0 or higher, whereas any database client whose software version is higher than or equal to 10.1 can use that disk group.
- The ADVM compatibility attribute determines whether the disk group can contain Oracle ASM volumes in the disk group. The value must be set to 11.2 or higher. Before setting this attribute, the `COMPATIBLE.ASM` value must be 11.2 or higher. Also, the ADVM volume drivers must be loaded.

The compatibility of a disk group needs to be advanced only when there is a change to either persistent disk structures or protocol messaging. However, advancing disk group compatibility is an irreversible operation. You can set the disk group compatibility by using either the `CREATE DISKGROUP` or `ALTER DISKGROUP` commands.

Note: In addition to the disk group compatibilities, the *compatible* parameter (database *compatible version*) determines the features that are enabled; it applies to the database or ASM instance depending on the *instance_type* parameter. For example, setting it to 10.1 would preclude use of any features introduced in Oracle Database 11*g* (disk online/offline, variable extents, and so on).

ASM Disk Group Attributes

Name	Property	Values	Description
au_size	Create, Alter	1|2|4|8|16|32|64MB	Size of allocation units in the disk group
compatible.rdbms	Create, Alter	Valid database version	Format of messages exchanged between DB and ASM
compatible.asm	Create, Alter	Valid ASM instance version	Format of ASM metadata structures on disk
compatible.advm	Create, Alter	Valid ASM instance version	Allows Oracle ASM volumes in disk group
disk_repair_time	Create, Alter	0 M to 2^{32} D	Length of time before removing a disk once OFFLINE
template.*tname*. redundancy	Alter	UNPROTECT|MIRROR|HIGH	Redundancy of specified template
template.*tname*. stripe	Alter	COARSE|FINE	Striping attribute of specified template

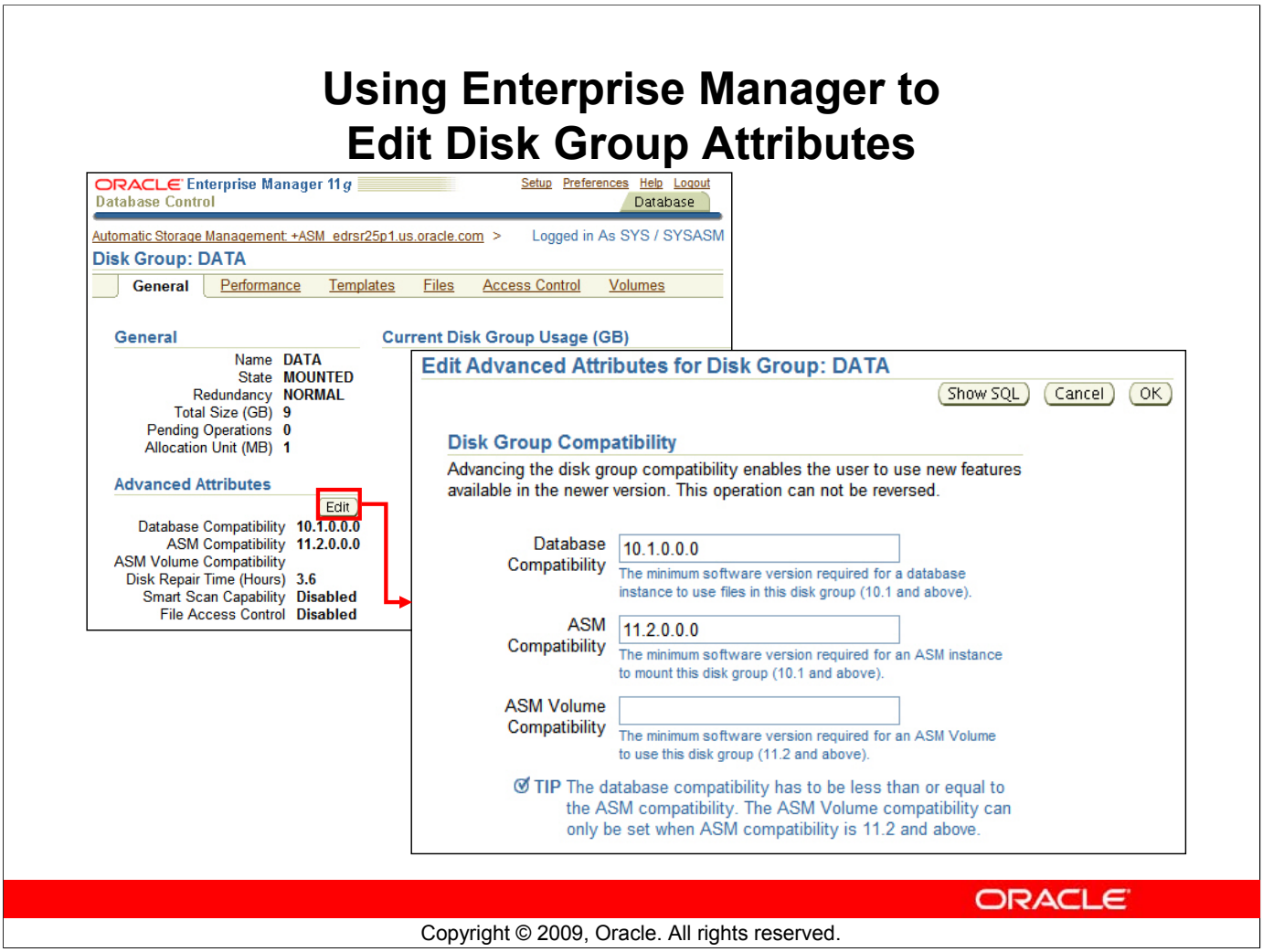
```
CREATE DISKGROUP DATA2 NORMAL REDUNDANCY
DISK '/dev/sda1','/dev/sdb1'
ATTRIBUTE 'compatible.asm'='11.2';
```


ASM Disk Group Attributes

Whenever you create or alter an ASM disk group, you have the ability to change its attributes using the new `ATTRIBUTE` clause of the `CREATE DISKGROUP` and `ALTER DISKGROUP` commands. These attributes are briefly summarized in the table given in the slide:

- ASM enables the use of different allocation unit (AU) sizes that you specify when you create a disk group. The AU can be 1, 2, 4, 8, 16, 32, or 64 MB in size.
- RDBMS compatibility: See the "ASM Disk Group Compatibility" section for more information.
- ASM compatibility: See the "ASM Disk Group Compatibility" section for more information.
- You can specify the `DISK_REPAIR_TIME` in units of minute (M), hour (H), or day (D). If you omit the unit, then the default is H. If you omit this attribute, then the default is 3.6H. You can override this attribute with an `ALTER DISKGROUP ... DISK OFFLINE` statement.
- You can also specify the redundancy attribute of the specified template.
- You can also specify the striping attribute of the specified template.

Note: For each defined disk group, you can look at all defined attributes through the `V$ASM_ATTRIBUTE` fixed view.



Using Enterprise Manager to Edit Disk Group Attributes

Enterprise Manager provides a simple way to store and retrieve environment settings related to disk groups.

You can set the compatible attributes from both the Create Disk Group page and the Edit Disk Group Advanced Attributes page. The `disk_repair_time` attribute is added to only edit the disk group advanced attributes page.

Note: For pre-11*g* ASM instances, the default ASM compatibility and the client compatibility are both 10.1. For 11*g* ASM instances, the default ASM compatibility is 11.2 and the database compatibility is 10.1.

Retrieving ASM Metadata

- Using SQL*Plus:

```
SQL> SELECT f.type, f.redundancy, f.striped, f.modification_date,
a.system_created, a.name FROM v$asm_alias a, v$asm_file f WHERE
a.file_number = f.file_number and a.group_number = f.group_number
and type='DATAFILE';
TYPE       REDUND   STRIPE   MODIFICAT   S   NAME
---------  -------  -------  ----------  -   ---------------------
DATAFILE   MIRROR   COARSE   08-JUL-09   Y   SYSTEM.256.689832921
DATAFILE   MIRROR   COARSE   08-JUL-09   Y   SYSAUX.257.689832923
..
```

- Using `asmcmd`:

```
ASMCMD> ls -l +DATA/orcl/datafile
Type       Redund   Striped   Time              Sys   Name
DATAFILE   MIRROR   COARSE    JUL 08 21:00:00   Y SYSTEM.256.689832921
DATAFILE   MIRROR   COARSE    JUL 08 21:00:00   Y SYSAUX.257.689832923
..
```

ORACLE

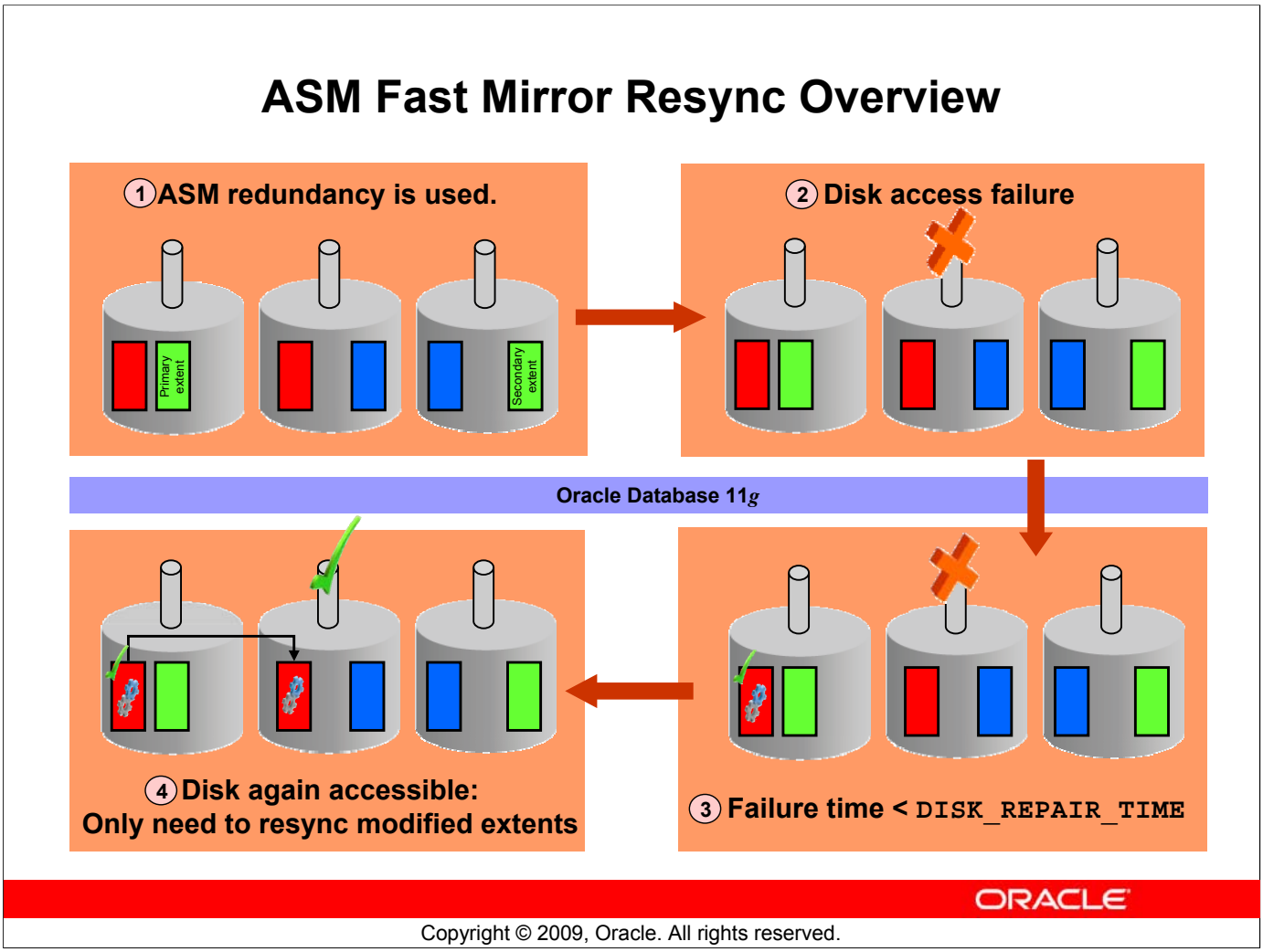
Retrieving ASM Metadata

The ASM instance hosts memory-based metadata tables presented as dynamic performance views. This data can be queried using SQL*Plus, `asmcmd`, or Enterprise Manager.

Using SQL*Plus requires knowledge of the SQL language and may involve joining multiple dynamic performance views to retrieve relevant information. This first example on the slide shows a join between `v$asm_file` and `v$asm_alias` to display metadata regarding the datafiles of a database. If this query is performed against the ASM instance, it could retrieve datafiles from multiple databases based on the way the syntax is designed. You would need to use additional filter conditions to restrict the output to a single database.

The `asmcmd` utility has the advantage of being able to connect to the ASM instance and retrieve metadata without knowledge of the SQL language. It used a style similar to UNIX notation. The second example on this slide uses `asmcmd` to retrieve the same metadata as the SQL example did. Another advantage of this example is that the output is restricted to the datafiles of a single database because the path that is being listed contains the database name of `orcl`, and the file type of `datafile`. Therefore, what appears as directories in `asmcmd` would require SQL filter conditions using the `WHERE` clause to give the same result.

Note: Enterprise Manager Database Control is able to display most of the ASM metadata by simply navigating among the various ASM Web pages.



ASM Fast Mirror Resync Overview

ASM fast mirror resync significantly reduces the time required to resynchronize a transient failure of a disk. When a disk goes offline following a transient failure, ASM tracks the extents that are modified during the outage. When the transient failure is repaired, ASM can quickly resynchronize only the ASM disk extents that have been affected during the outage.

This feature assumes that the content of the affected ASM disks has not been damaged or modified.

When an ASM disk path fails, the ASM disk is taken offline but not dropped if you have set the `DISK_REPAIR_TIME` attribute for the corresponding disk group. The setting for this attribute determines the duration of a disk outage that ASM tolerates while still being able to resynchronize after you complete the repair.

Note: The tracking mechanism uses one bit for each modified allocation unit. This ensures that the tracking mechanism is very efficient.

Quiz

Which parameter is required for an ASM instance?

1. INSTANCE_TYPE
2. ASM_DISKGROUPS
3. LARGE_POOL_SIZE
4. None of the above

Answer: 1

Quiz

Fine-grain striping, by default, is used for _____ and _____.

1. Data files
2. Control files
3. Temp files
4. Online redo logs
5. SPFILE

ORACLE

Answers: 2, 4

Summary

In this lesson, you should have learned how to:

- Manage the ASM instance by using SQL*plus, `asmcmd`, and Enterprise Manager
- Create and drop ASM disk groups
- Specify ASM compatibility attributes
- Extend ASM disk groups
- Compare methods of retrieving ASM metadata

ORACLE

Practice 5 Overview: Managing the ASM Instance

This practice covers the following topics:

- Creating an ASM disk group with `asmcmd`
- Dropping an ASM disk group with EM
- Viewing ASM metadata

ORACLE

6

Configuring the Oracle Network Environment

Objectives

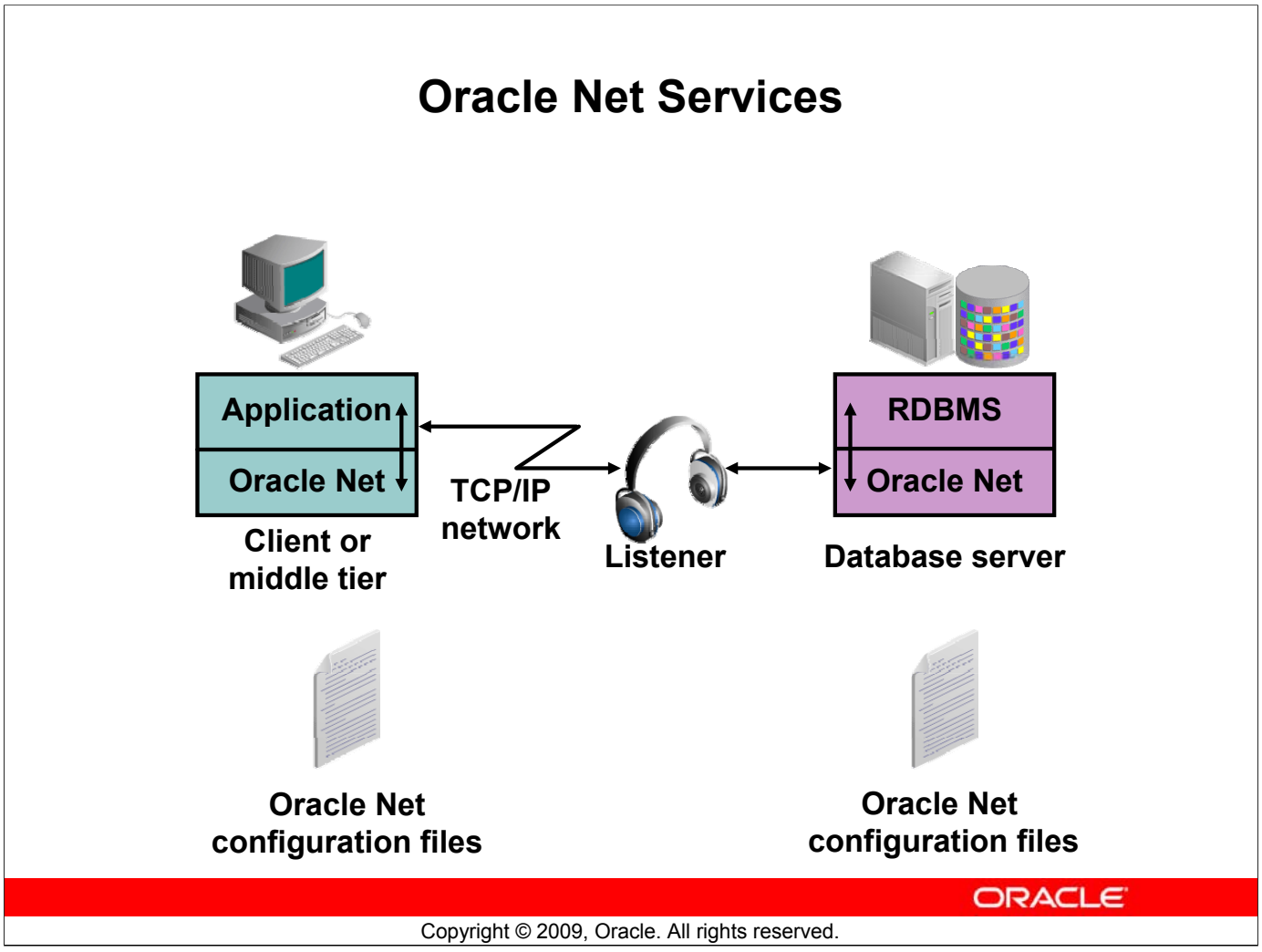
After completing this lesson, you should be able to:

- Use Enterprise Manager to:
  - Create additional listeners
  - Create Oracle Net Service aliases
  - Configure connect-time failover
  - Control the Oracle Net Listener
- Use `tnsping` to test Oracle Net connectivity
- Identify when to use shared servers and when to use dedicated servers

ORACLE

Resources

- *Oracle Database Net Services Administrator's Guide*
- *Oracle Database Net Services Reference*



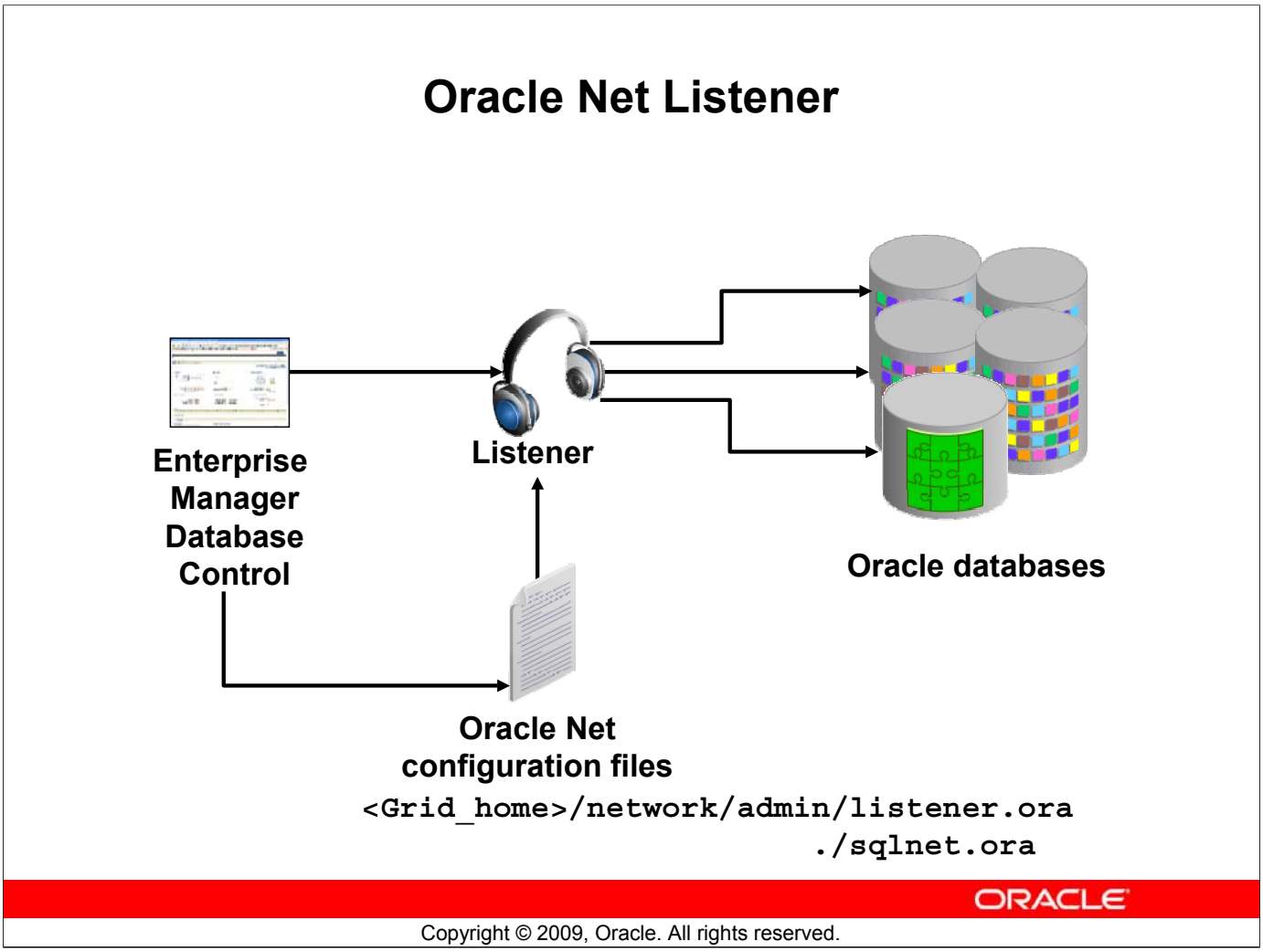
Oracle Net Services

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

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

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

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



Oracle Net Listener

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

Enterprise Manager is one of the ways to access the listener. You can control the configuration of the actual listener as well as general parameters such as password protection and log file locations.

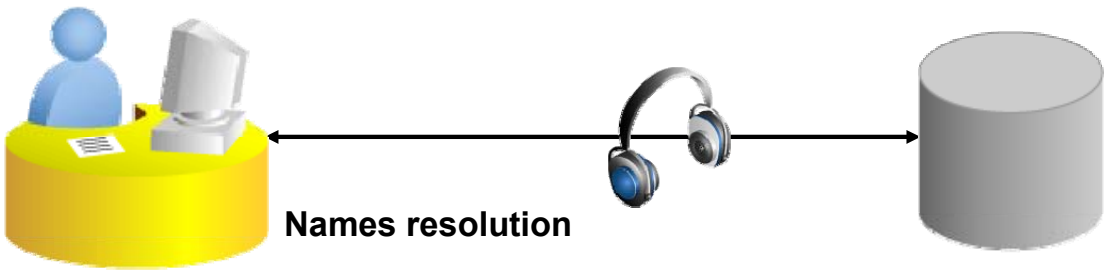
Advanced administrators can also configure Oracle Net Services by manually editing the configuration files, if necessary, with a standard operating system (OS) text editor such as `vi` or `gedit`.

Note: When the Grid Infrastructure for Standalone Server is installed, the Oracle Net Listener is started from its software installation directory, known as `<Grid_home>`. A listener is required to run from this software installation to provide connection capabilities to the ASM instance. This is also used by default to listen for all database instances that are installed on the same server.

Establishing Net 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

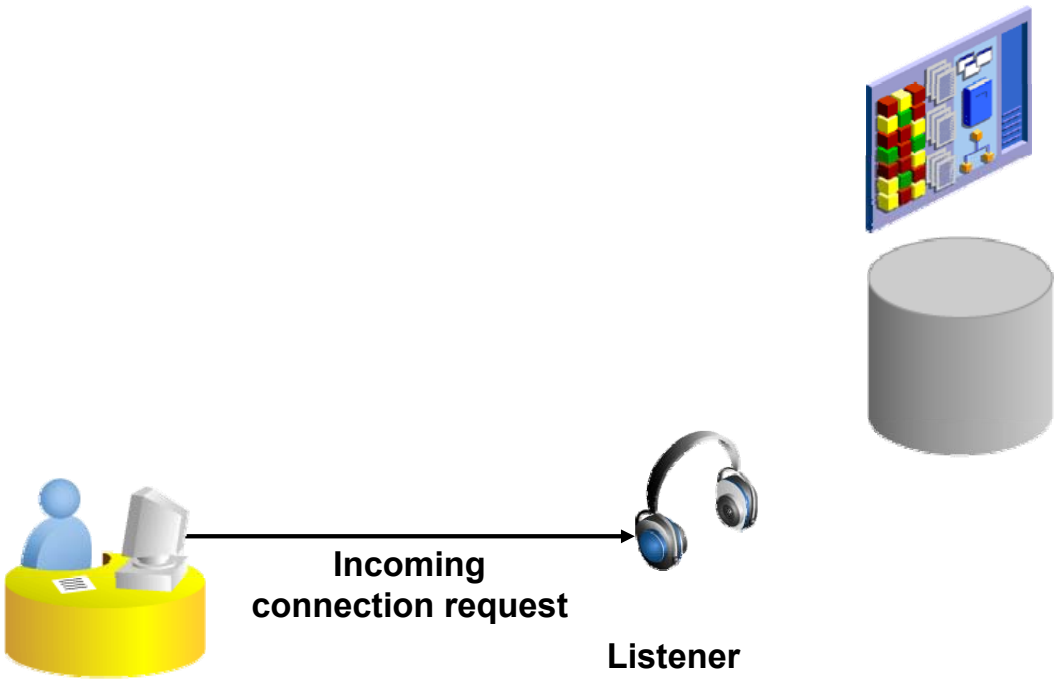


Establishing Net Connections

For an application to connect to a service through an Oracle Net Listener, the application must have information about that service, including the address or host where the listener resides, the protocol that the listener accepts, and the port that the listener monitors. After the listener is located, the final piece of information that the application needs is the name of the service to which it wants to connect.

Oracle Net names resolution is the process of determining this connection information.

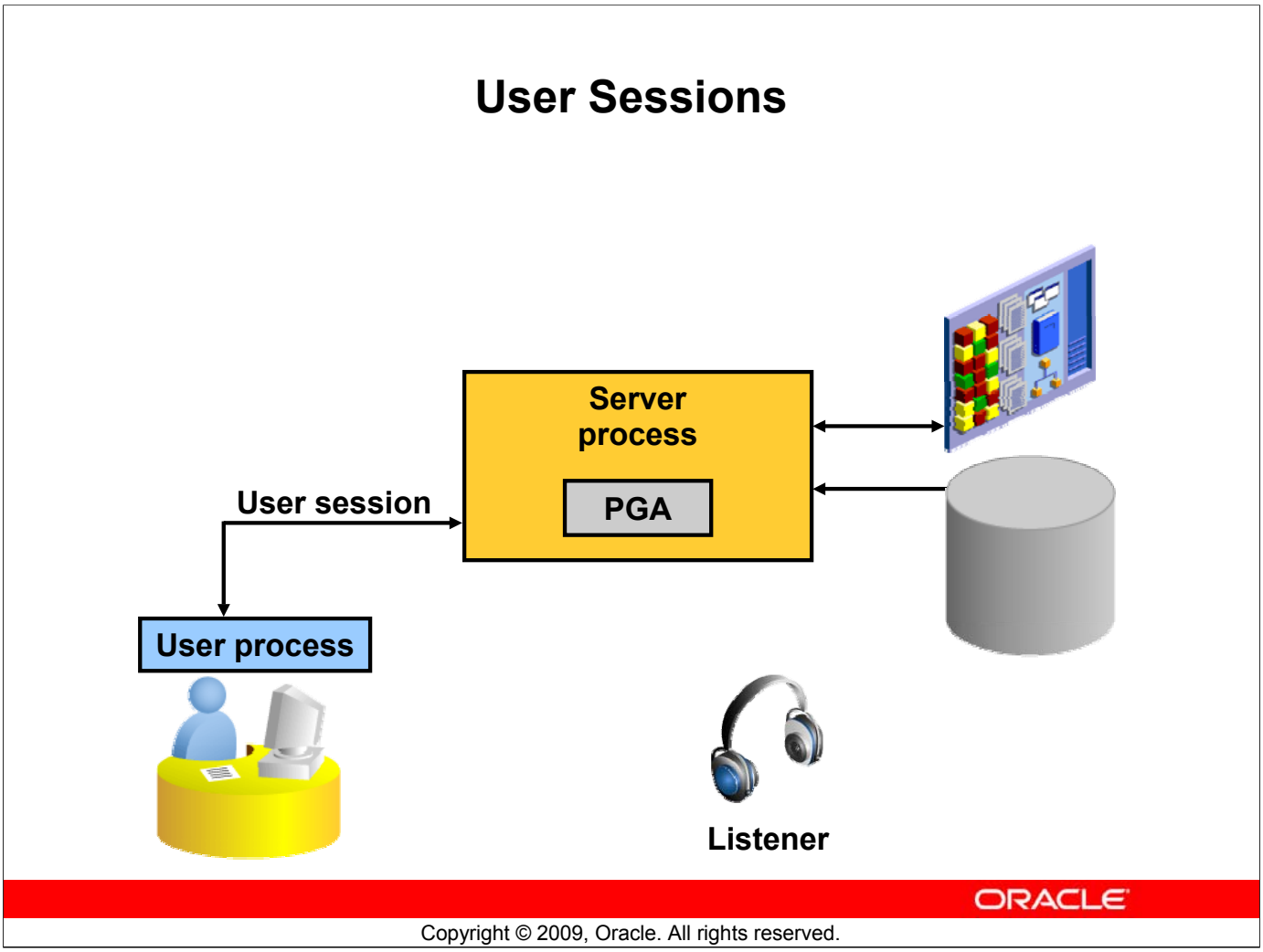
Establishing a Connection



Establishing a Connection

After Oracle Net names resolution is complete, a connection request is passed from the user or middle-tier application (hereafter referred to as the *user process*) to the listener. The listener receives a `CONNECT` packet and checks whether that `CONNECT` packet is requesting a valid Oracle Net service name.

If the service name is not requested (as in the case of a `tnsping` request), the listener acknowledges the connect request and does nothing else. If an invalid service name is requested, the listener transmits an error code to the user process.



User Sessions

If the CONNECT packet requests a valid service name, the listener spawns a new process to deal with the connection. This new process is known as the *server process*. The listener connects to the process and passes the initialization information, including the address information for the user process. At this point, the listener no longer deals with the connection and all work is passed to the server process.

The server process checks the user's authentication credentials (usually a password), and if the credentials are valid, a user session is created.

Dedicated server process: With the session established, the server process now acts as the user's agent on the server. The server process is responsible for:

- Parsing and running any SQL statements issued through the application
- Checking the database buffer cache for data blocks required to perform SQL statements
- Reading necessary data blocks from data files on the disk into the database buffer cache portion of the System Global Area (SGA), if the blocks are not already present in the SGA
- Managing all sorting activity. The Sort Area is a memory area that is used to work with sorting; it is contained in a portion of memory that is associated with the Program Global Area (PGA).
- Returning results to the user process in such a way that the application can process the information
- Reading auditing options and reporting user processes to the audit destination

Tools for Configuring and Managing the Oracle Network

- Enterprise Manager Net Services Administration page
- Oracle Net Manager
- Oracle Net Configuration Assistant
- Command line

ORACLE

Tools for Configuring and Managing the Oracle Network

Use the following tools and applications to manage your Oracle Network configuration:

- **Enterprise Manager:** Provides an integrated environment for configuring and managing Oracle Net Services. Use Enterprise Manager to configure Oracle Net Services for any Oracle home across multiple file systems and to administer listeners.
- **Oracle Net Manager:** Provides a graphical user interface (GUI) through which you can configure Oracle Net Services for an Oracle home on a local client or a server host. Oracle Net Manager enables you to configure Oracle Net Services for an Oracle home on a local client or server host. You can use Oracle Net Manager to configure the following network components:
  - **Naming:** Define simple names, connect identifiers, and map them to connect descriptors to identify the network location and identification of a service. Oracle Net Manager supports configuration of connect descriptors in local `tnsnames.ora` files or a centralized directory service.
  - **Naming methods:** Configure the different ways in which connect identifiers are resolved into connect descriptors
  - **Profiles:** Configure preferences for enabling and configuring Oracle Net features on the client or server
  - **Listeners:** Create and configure listeners to receive client connections

Tools for Configuring and Managing the Oracle Network (continued)

- **Oracle Net Configuration Assistant:** Is launched by Oracle Universal Installer when you install the Oracle software. The Oracle Net Configuration Assistant enables you to configure the listening protocol address and service information for an Oracle database. During a typical database installation, Oracle Net Configuration Assistant automatically configures a listener called `LISTENER` that has a TCP/IP listening protocol address for the database. If you do a custom installation, Oracle Net Configuration Assistant prompts you to configure a listener name and protocol address of your choice. Use the Oracle Net Configuration Assistant for initial network configuration after database installation. Thereafter, you can use the Oracle Enterprise Manager and Oracle Net Manager to configure and administer your networks.
- **Command line:** Is used to start, stop, and view the status of the listener process. It is an OS user (in this course, `oracle`) that starts or stops the listener. If the listener is not started, you cannot use Enterprise Manager.

Listener Control Utility

Oracle Net listeners can be controlled with the `lsnrctl` command-line utility (or from EM).

```
$ . oraenv
ORACLE_SID = [orcl] ? +ASM
$ lsnrctl
LSNRCTL for Linux: Version 11.2.0.1.0 - Production on 30-JUN-2009 00:47:01
Copyright (c) 1991, 2009, Oracle.  All rights reserved.
Welcome to LSNRCTL, type "help" for information.

LSNRCTL> help
The following operations are available
An asterisk (*) denotes a modifier or extended command:
start               stop                status
services            version             reload
save_config         trace               spawn
change_password     quit                exit
set*                show*
```


Listener Control Utility

When an instance starts, a listener process establishes a communication pathway to the Oracle database. The listener is then able to accept database connection requests.

The listener control utility enables you to control the listener. With `lsnrctl`, you can:

- Start the listener
- Stop the listener
- Check the status of the listener
- Reinitialize the listener from the configuration file parameters
- Dynamically configure many listeners
- Change the listener password

The basic command syntax for this utility is:

```
LSNRCTL> command [listener_name]
```

When the `lsnrctl` command is issued, the command acts on the default listener (named `LISTENER`) unless a different listener name is specified or the `SET CURRENT_LISTENER` command is executed. If the listener name is `LISTENER`, the *`listener_name`* argument can be omitted. The valid commands for `lsnrctl` are shown in the slide.

Note: The lsnrctl utility is located in both the Grid Infrastructure home and the Oracle database home. It is important to set the environment variables to the appropriate home before using it.

Listener Control Utility Syntax

Commands from the listener control utility can be issued from the command line or from the LSNRCTL prompt.

- Command-line syntax:

```
$ lsnrctl <command name>
$ lsnrctl start
$ lsnrctl status
```

- Prompt syntax:

```
LSNRCTL> <command name>
LSNRCTL> start
LSNRCTL> status
```

Listener Control Utility Syntax

The `lsnrctl` commands can be issued from within the utility (prompt syntax) or from the command line. The following two commands have the same effect but use command-line syntax and prompt syntax, respectively:

Command-line syntax:

```
$ lsnrctl start
```

Prompt syntax:

```
$ lsnrctl
LSNRCTL for Linux: Version 11.2.0.1.0 - Production on 30-JUN-
2009 01:00:01
Copyright (c) 1991, 2009, Oracle.  All rights reserved.
Welcome to LSNRCTL, type "help" for information.
LSNRCTL> start
```

The command-line syntax is typically used to execute an individual command or scripted commands. If you plan to execute several consecutive `lsnrctl` commands, the prompt syntax is more efficient. Note that the `listener_name` argument is omitted, and the stop command would thus affect the listener named `LISTENER`. Prompt syntax must be used if your listener is password protected.

Listener Control Utility Syntax (continued)

Remember that if your listener is named something other than LISTENER, you must either include the listener name with the command or use the SET CURRENT_LISTENER command. Suppose that your listener is named custom_lis. Here are two examples of stopping a listener named custom_lis by using prompt syntax:

```
LSNRCTL> stop custom_lis
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host01)(PORT=5521)))
The command completed successfully
```

This produces the same results as the following:

```
LSNRCTL> set cur custom_lis
Current Listener is custom_lis
LSNRCTL> stop
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host01)(PORT=5521)))
The command completed successfully
```

Note: In the preceding syntax, current_listener has been abbreviated to cur.

Using command-line syntax produces the same results:

```
$ lsnrctl stop custom_lis
LSNRCTL for Linux: Version 11.2.0.1.0 - Production on 30-JUN-
2009 01:01:53
Copyright (c) 1991, 2009, Oracle.  All rights reserved.
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=host01)(PORT=5521)))
The command completed successfully
```

Using SRVCTL to Start and Stop the Listener

If Oracle Restart is configured to monitor your listener, you should use SRVCTL to manage that listener.

- Example syntax:

```
$ srvctl -h
$ srvctl start listener
$ srvctl stop listener
$ srvctl start listener -l mylistener
$ srvctl status listener
```

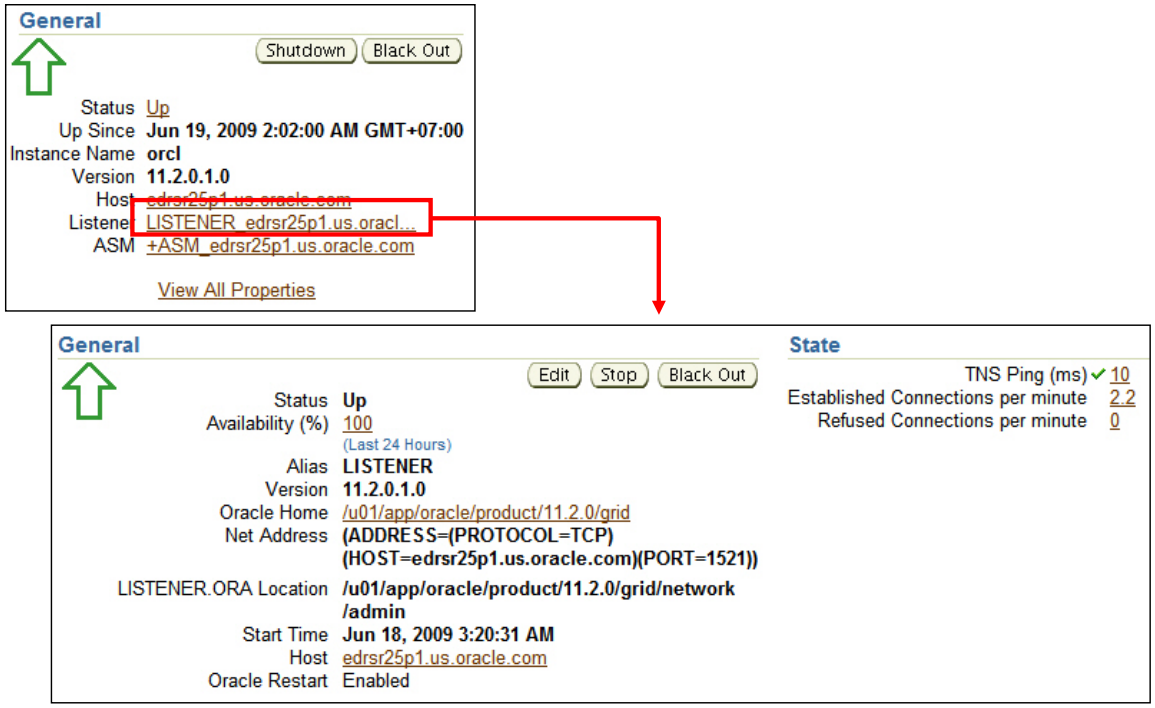
ORACLE

Using SRVCTL to Start and Stop the Listener

You can start, stop, and view the status any listener that is managed by Oracle Restart with SRVCTL. Examples include:

- To display help with the commands available in SRVCTL: `srvctl -h`
- To start the default listener: `srvctl start listener`
- To stop the default listener: `srvctl stop listener`
- To start a listener named mylistener: `srvctl start listener -l mylistener`
- To display the status of the default listener: `srvctl status listener`

Listener Home Page



Listener Home Page

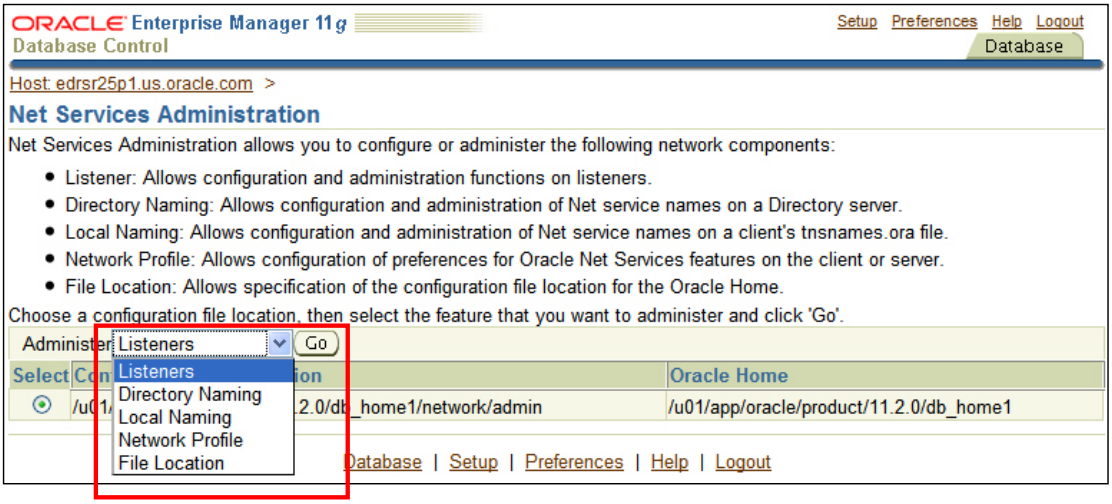
Click the Listener link on the Enterprise Manager Database Home page to access the Listener Home page.

On this page, you can see:

- Listener status and availability in the last 24 hours
- Listener version and Oracle home
- First listening address for the listener
- Location of the configuration files that are used to start the listener
- Listener start time and host information
- Oracle Restart status

To start the listener, go to the Database Home page and click the listener name to open the Listener Home page. Click Stop to stop the listener if it is running, or click Start to start the listener if it is not running. Log on to the host as the OS user who can start and stop the listener.

Net Services Administration Page



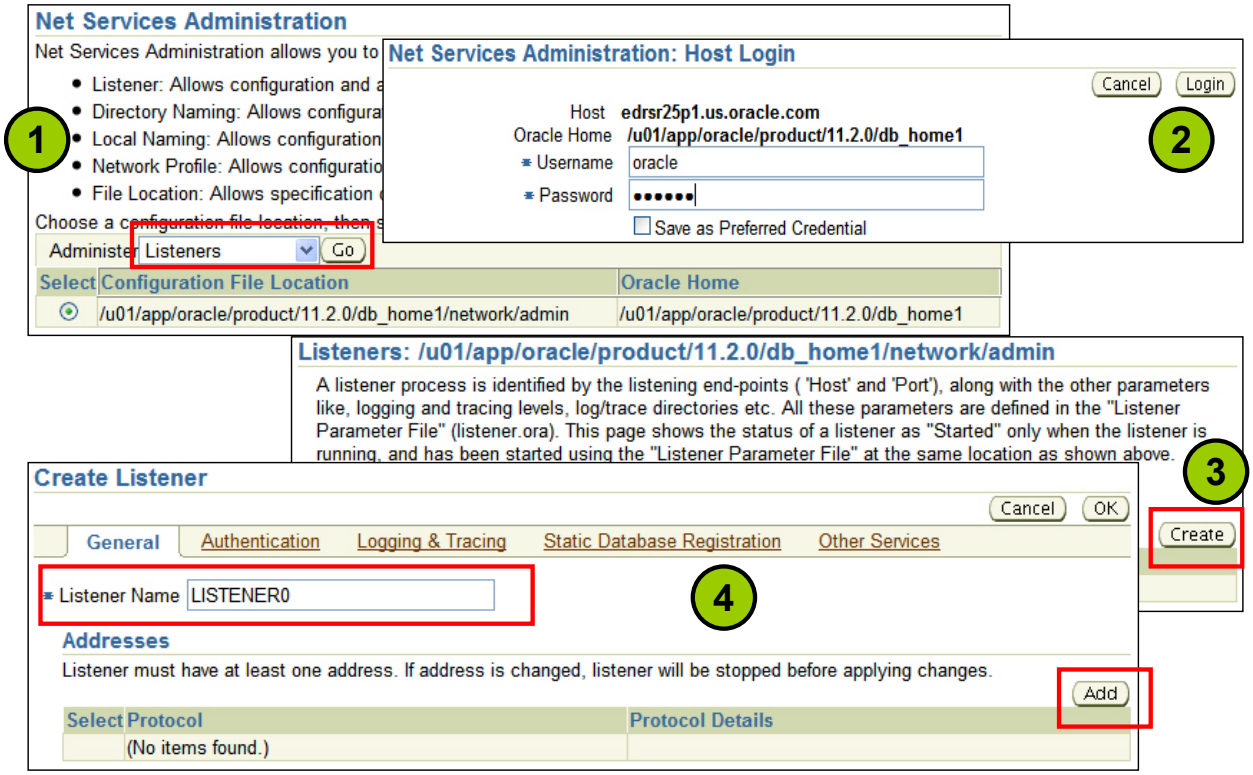
ORACLE

Net Services Administration Page

The Net Services Administration page enables you to configure Oracle Net Services for any Oracle home across multiple file systems. It also provides common administration functions for listeners. You can use Net Services Administration to configure and administer the following:

- **Listeners:** Add, remove, start, and stop a listener, as well as change its tracing and logging characteristics. You can also look at a listener's control status report.
- **Directory naming:** Define simple names and connect identifiers, and map them to connect descriptors to identify the network location and identification of a service. Save database services, Net Services, and Net Service aliases in a centralized directory service.
- **Local naming:** Save Net Service names in the `tnsnames.ora` file.
- **Profiles:** Configure the `sqlnet.ora` parameters.
- **File location:** Change the location of the configuration files of Net Services.

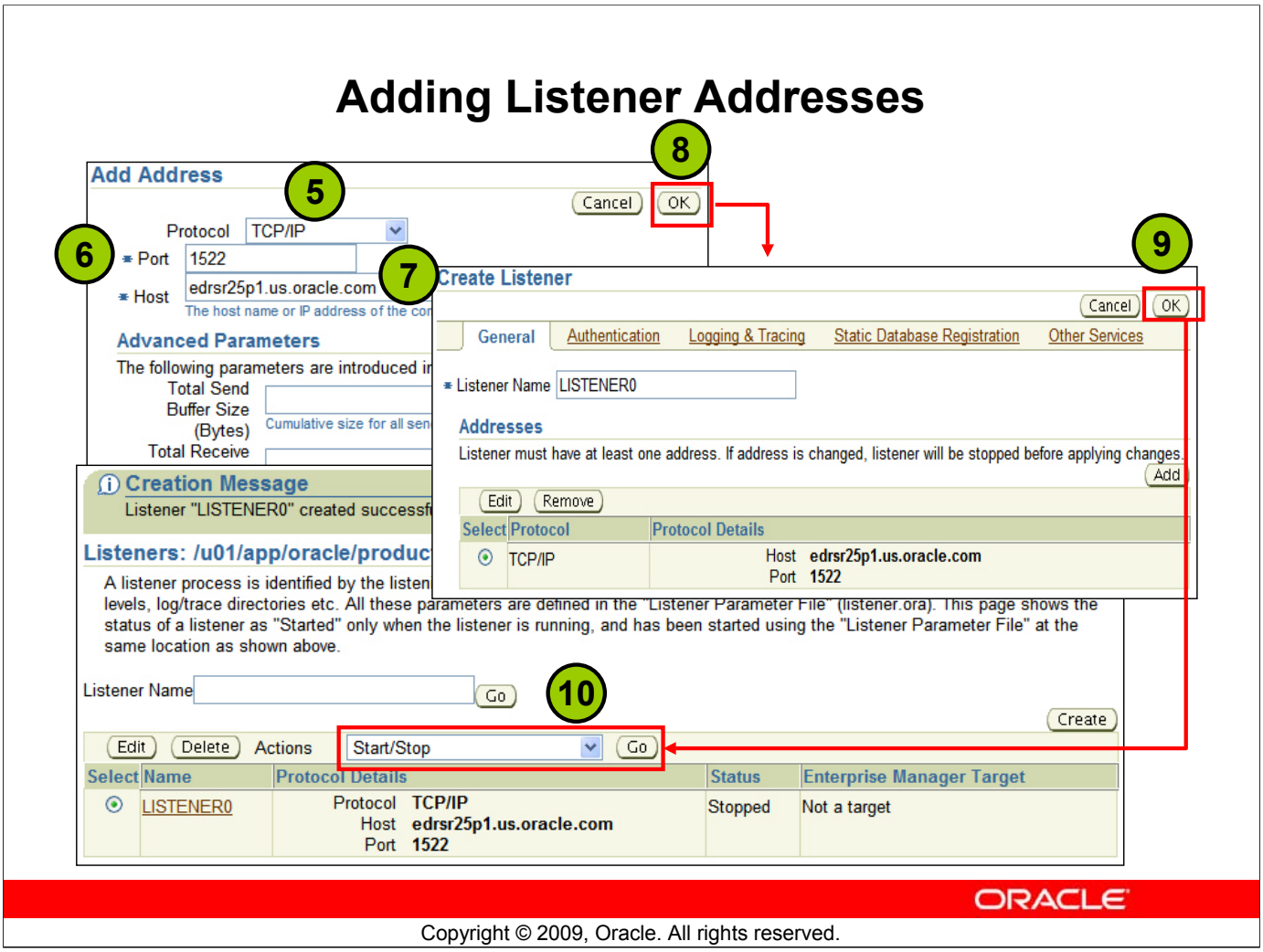
Creating a Listener



Creating a Listener

To create an Oracle Net Listener, click Net Services Administration in the Related Links region of the Listener properties page. Then perform the following steps:

1. Select Listeners from the Administer drop-down list, and click Go.
2. If you have not entered preferred credentials for the host, the Host Login page appears. Enter the username and password, and then click Login.
3. Click Create.
4. Enter a listener name. The name must be unique for this server. Click Add to add a listener address. Each listener must have at least one listener address.

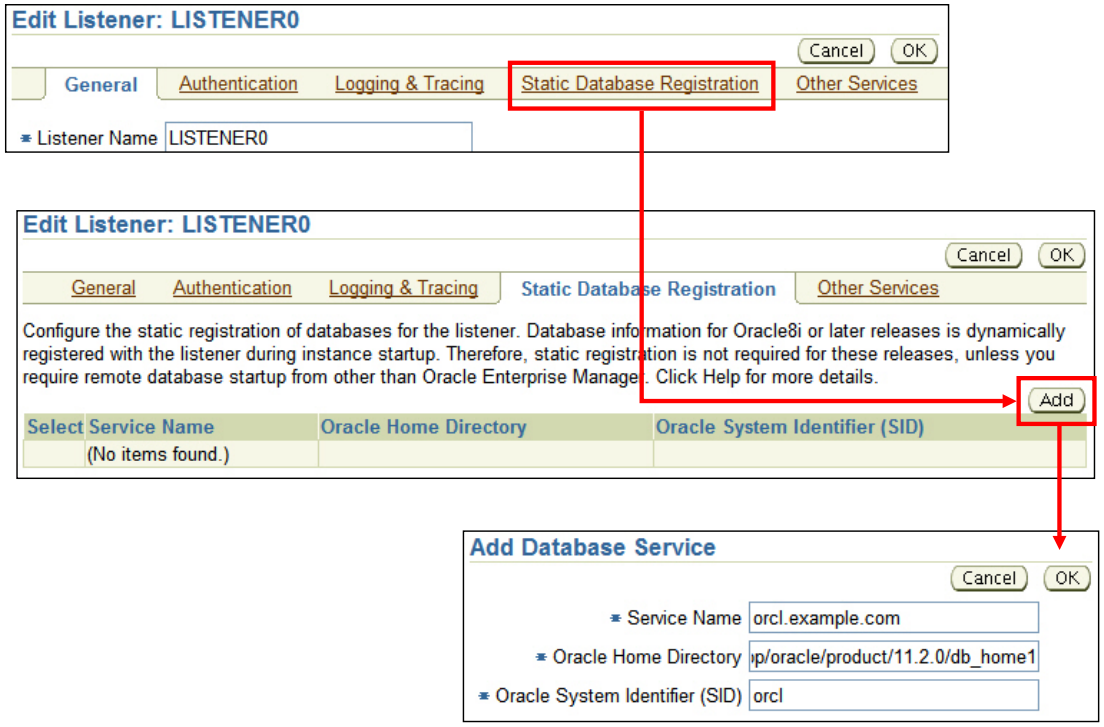


Adding Listener Addresses

The workflow to create a listener continues with the creation of the listener address:

5. Select the network protocol. TCP/IP is the most commonly used and is the default. Other choices are Internal Process Communication (IPC)—normally used for connecting to local applications (resident on the database server)—or external code libraries (EXTPROC), and TCP/IP with SSL.
   Note: The EXTPROC protocol is configured by using the Other Services tab.
6. Enter the port that you want the listener to monitor. Oracle Net's default port is 1521. If you choose to use a port other than 1521, additional configuration is required for the listener or for the instance.
7. Enter the name or IP address of the server that the listener will run on.
8. All other configuration steps are optional for the listener. Click OK to save the address. The only required configuration is the listening address and name.
9. On the Create Listener page, review the information about the address you just created and click OK to save your changes.
10. To start the new listener, select Start/Stop from the Actions drop-down list and then click Go.

Database Service Registration



Database Service Registration

For a listener to forward client connections to an instance, the listener must know the name of the instance and where the instance's ORACLE_HOME is located. The listener can find this information in two ways:

- **Dynamic service registration:** Oracle8*i* and later instances automatically register with the default listener on database startup. No additional listener configuration is required for the default listener.
- **Static service registration:** Earlier releases of the Oracle database do not automatically register with the listener and, therefore, require that the listener configuration file contain a list of all database services that the listener will serve. You may still choose to use static service registration with newer releases if:
  - Your listener is not on the default port of 1521, and you do not want to configure your instance to register with a nondefault port
  - Your application requires static service registration

To add a static database service, click Static Database Registration on the Edit Listener page, and then click the Add button. Enter the service name (same as the global database name *<DB_NAME>.<DB_DOMAIN>*), ORACLE_HOME path, and SID (same as the instance name). Click OK. For the changes to take effect, you must reload (use the RELOAD command) or restart your listener.

Database Service Registration (continued)

Service Names

The SERVICE_NAMES initialization parameter specifies one or more names by which clients can connect to the instance. The instance registers its service names with the listener. When a client requests a service, the listener determines which instances offer the requested service and routes the client to the appropriate instance.

You can specify multiple service names to distinguish among different uses of the same database, as in this example:

```
SERVICE_NAMES = sales.example.com, eurosales.example.com
```

You can also use service names to identify a single service that is available from two different databases through the use of replication.

If you do not qualify the names in this parameter with a domain, Oracle qualifies them with the value of the DB_DOMAIN parameter. If DB_DOMAIN is not specified, no domain will be applied to the nonqualified SERVICE_NAMES values.

While processing a client connection request, the listener tries to match the value of this parameter with the value of the SERVICE_NAME parameter in the client connect descriptor. If the client connect descriptor uses the SID parameter, the listener does not attempt to map the values. The value for this parameter is typically obtained from the combination of the DB_NAME and DB_DOMAIN parameters (DB_NAME.DB_DOMAIN) in the initialization parameter file, but the value can also contain any valid name used by clients to identify the service.

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
- External naming: Uses a supported non-Oracle naming service

Naming Methods

Oracle Net provides support for the following naming methods:

- **Easy connect naming:** The easy connect naming method enables clients to connect to an Oracle database server by using a TCP/IP connect string consisting of a host name and optional port and service name as follows:

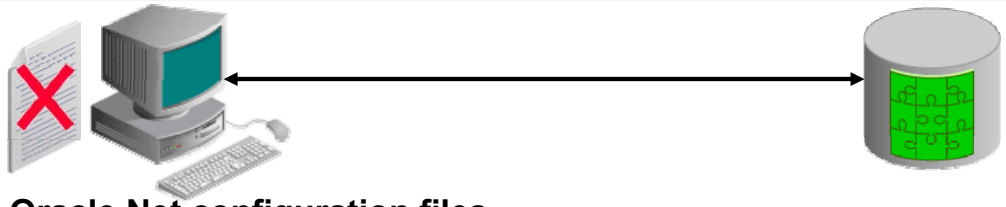
  ```
  CONNECT username/password@host[:port][/service_name]
  ```

  The easy connect naming method requires no configuration.
- **Local naming:** The local naming method stores connect descriptors (identified by their net service name) in a local configuration file named `tnsnames.ora` on the client.
- **Directory naming:** To access a database service, the directory naming method stores connect identifiers in a centralized directory server that is compliant with the Lightweight Directory Access Protocol (LDAP).
- **External naming:** The external naming method stores net service names in a supported non-Oracle naming service. Supported third-party services include:
  - Network Information Service (NIS) External Naming
  - Distributed Computing Environment (DCE) Cell Directory Services (CDS)

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

ORACLE

Easy Connect

With Easy Connect, you supply all information that is required for the Oracle Net connection as part of the connect string. Easy Connect connection strings take the following form:

```
<username>/<password>@<hostname>:<listener port>/<service name>
```

The listener port and service name are optional. If the listener port is not provided, Oracle Net assumes that the default port of 1521 is being used. If the service name is not provided, Oracle Net assumes that the database service name and host name provided in the connect string are identical.

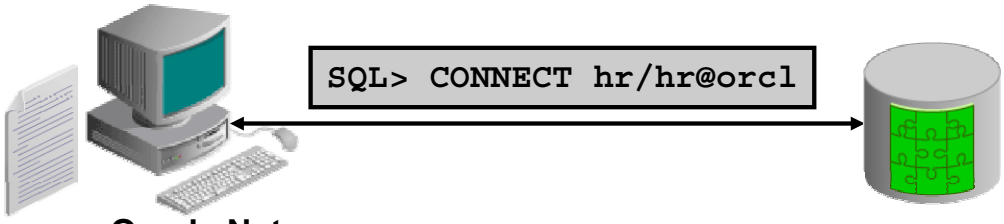
Assuming that the listener uses TCP to listen on port 1521 and the `SERVICE_NAMES=db` and `DB_DOMAIN=us.oracle.com` instance parameters, the connect string shown in the slide can be shortened:

```
SQL> connect hr/hr@db.us.oracle.com
```

Note: The `SERVICE_NAMES` initialization parameter can accept multiple comma-separated values. Only one of those values must be `db` for this scenario to work.

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



Oracle Net configuration files

ORACLE

Local Naming

With local naming, the user supplies an alias for the Oracle Net service. Oracle Net checks the alias against a local list of known services and, if it finds a match, converts the alias into host, protocol, port, and service name.

One advantage of local naming is that the database users need to remember only a short alias rather than the long connect string required by Easy Connect.

The local list of known services is stored in the following text configuration file:

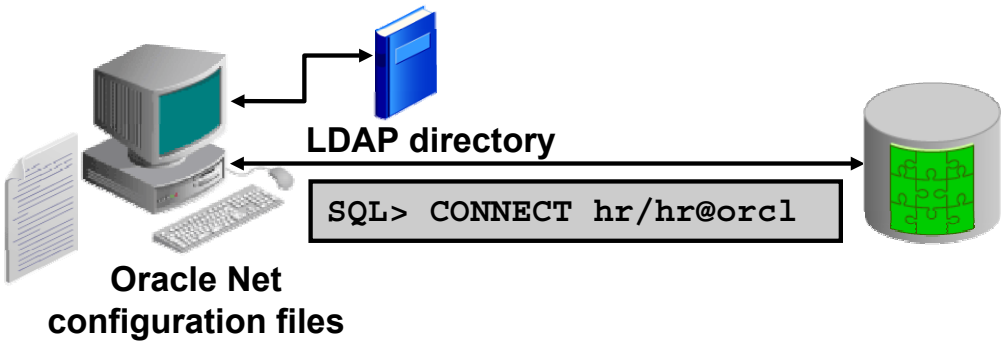
```
<oracle_home>/network/admin/tnsnames.ora
```

This is the default location of the `tnsnames.ora` file, but the file can be located elsewhere using the `TNS_ADMIN` environment variable.

Local naming is appropriate for organizations in which Oracle Net service configurations do not change often.

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



ORACLE

Directory Naming

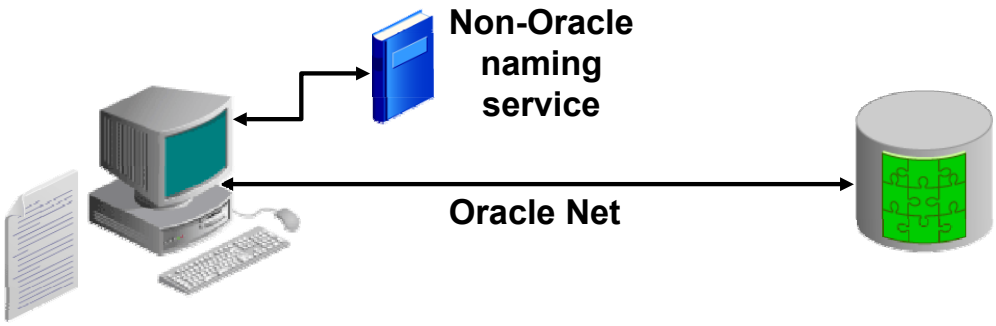
With directory naming, the user supplies an alias for the Oracle Net service. Oracle Net checks the alias against an external list of known services and, if it finds a match, converts the alias into host, protocol, port, and service name. Like local naming, database users need to remember only a short alias.

One advantage of directory naming is that the service name is available for users to connect with as soon as a new service name is added to the LDAP directory. With local naming, the database administrator (DBA) must first distribute updated `tnsnames.ora` files containing the changed service name information before users can connect to new or modified services.

Directory naming is appropriate for organizations in which Oracle Net service configurations change frequently.

External Naming Method

- Uses a supported non-Oracle naming service
- Includes:
  - Network Information Service (NIS) External Naming
  - Distributed Computing Environment (DCE) Cell Directory Services (CDS)



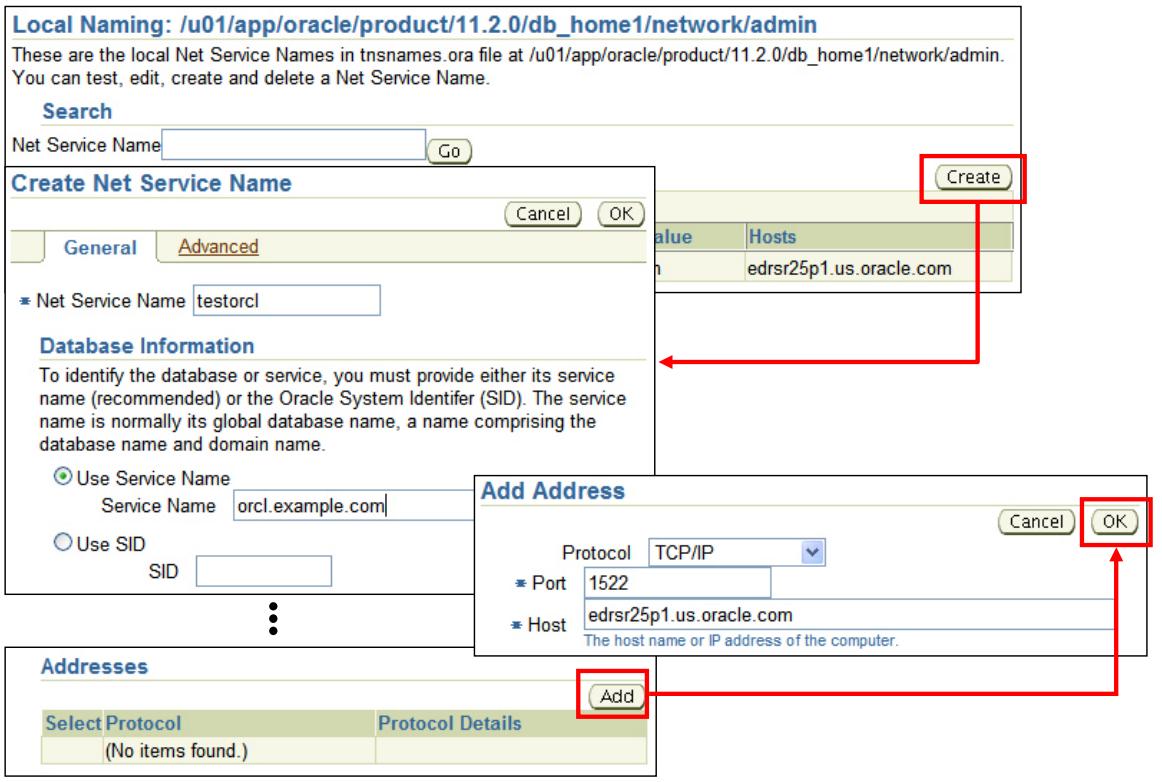
External Naming Method

The external naming method stores Net Service names in a supported non-Oracle naming service. Supported third-party services include:

- Network Information Service (NIS) External Naming
- Distributed Computing Environment (DCE) Cell Directory Services (CDS)

Conceptually, external naming is similar to directory naming.

Configuring Service Aliases



Configuring Service Aliases

To create a local Oracle Net Service alias, select Local Naming from the Administer drop-down list on the Net Services Administration page and click Go. Then click Create.

You can configure service aliases for directory naming by selecting Directory Naming instead of Local Naming.

Note: If directory naming has not already been configured, you cannot select the Directory Naming option. Directory naming is discussed in the *Oracle Enterprise Identity Management* course as well as in the *Oracle Advanced Security Administration* manual.

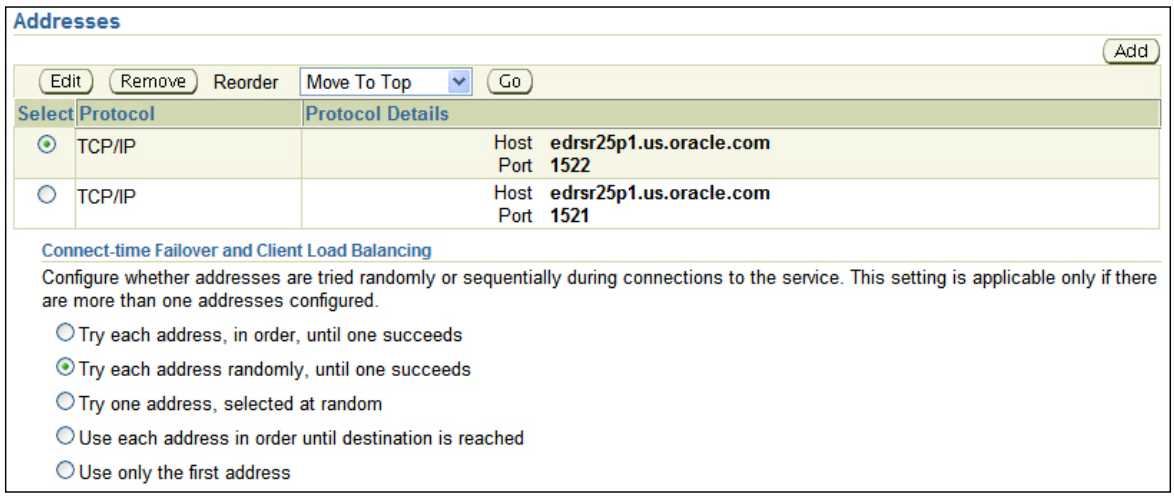
On the Create Net Service Name page, enter a unique name in the Net Service Name field. (This is the name that users enter when they want to use this alias.) Enter the service name or system identifier (SID) of the database that you want to connect to, and click the Add button to enter the address for the service name.

For the address, enter the protocol, port, and host used by the listener for the service to which you want to connect.

Advanced Connection Options

Oracle Net supports the following advanced connection options with local and directory naming:

- Connect-time failover
- Load balancing
- Source routing



ORACLE

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. The addresses can be chosen randomly or tried sequentially. In cases in which more than one listener is available, such as Oracle Real Application Clusters (RAC) configurations, Oracle Net can take advantage of listener failover and load balancing as well as Oracle Connection Manager source routing.

With *connect-time failover* enabled, the alias has two or more listener addresses listed. If the first address is not available, the second is tried. Oracle Net keeps trying addresses in the listed order until it reaches a listener that is functioning or until all addresses have been tried and failed. Transparent Application Failover (TAF) is a client-side feature that allows clients to reconnect to surviving databases in the event of a database instance failure. Notifications are used by the server to trigger TAF callbacks on the client side.

With *load balancing* enabled, Oracle Net picks an address at random from the list of addresses. The run-time connection load-balancing feature improves connection performance by balancing the number of active connections among multiple dispatchers. In a RAC environment, connection pool load balancing also has the capability to balance the number of active connections among multiple instances.

Advanced Connection Options (continued)

Source routing is used with Oracle Connection Manager, which serves as a proxy server for Oracle Net traffic, enabling Oracle Net traffic to be routed securely through a firewall. Oracle Net treats the addresses as a list of relays, connecting to the first address and then requesting to be passed from the first to the second until the destination is reached. It differs from failover or load balancing in that all addresses are used each time a connection is made.

Option	**Advanced Functionality**
Try each address in sequence until one succeeds.	Failover
Try each address randomly until one succeeds.	Failover Load balancing
Try one address selected at random.	Load balancing
Use each address in sequence until the destination is reached.	Source routing
Use only the first address.	None

Testing Oracle Net Connectivity

The `tnsping` utility that tests Oracle Net service aliases:

- Ensures connectivity between the client and the Oracle Net Listener
- Does not verify that the requested service is available
- Supports Easy Connect Names Resolution:

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

- Supports local and directory naming:

```
tnsping orcl
```

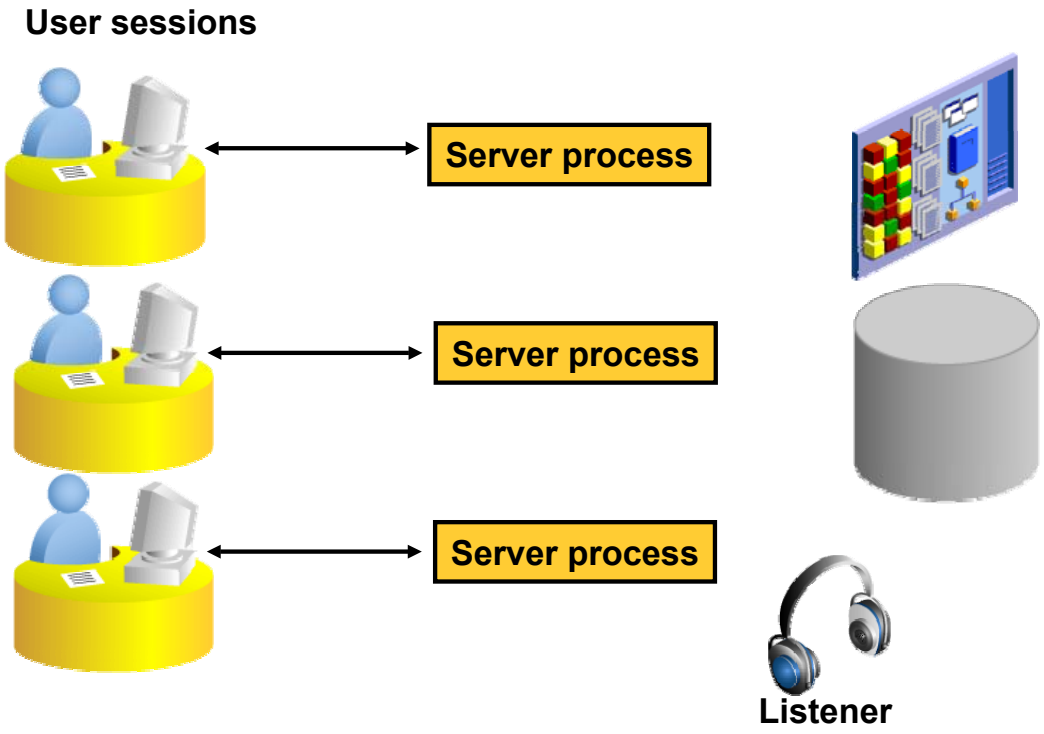
ORACLE

Testing Oracle Net Connectivity

`tnsping` is the Oracle Net equivalent of the TCP/IP ping utility. It offers a quick test to verify that the network path to a destination is good. For example, enter `tnsping orcl` in a command-line window.

The utility validates that the host name, port, and protocol reach a listener. It does not actually check whether the listener handles the service name. The `tnsping` utility also reveals the location of the configuration files. In a system with multiple `ORACLE_HOME` locations, this can be helpful.

User Sessions: Dedicated Server Process

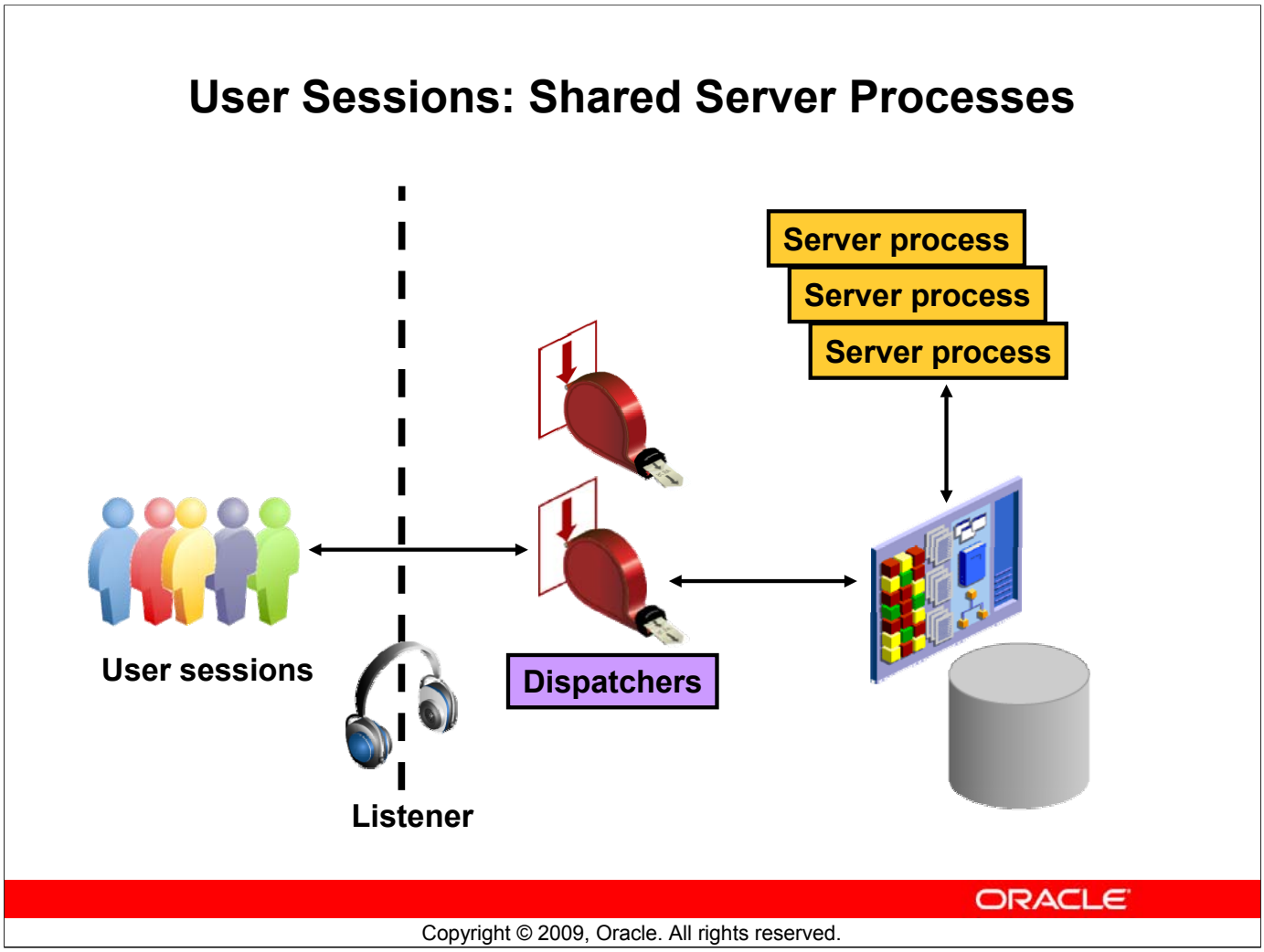


User Sessions: Dedicated Server Process

With dedicated server processes, there is a one-to-one ratio of server processes to user processes. Each server process uses system resources, including CPU cycles and memory.

In a heavily loaded system, the memory and CPU resources that are used by dedicated server processes can be prohibitive and can negatively affect the system's scalability. If your system is being negatively affected by the resource demands of the dedicated server architecture, you have the following options:

- Increasing system resources by adding more memory and additional CPU capability
- Using the Oracle Shared Server Process architecture



User Sessions: Shared Server Processes

Each service that participates in the shared server process architecture has at least one dispatcher process (and usually more). When a connection request arrives, the listener does not spawn a dedicated server process. Instead, the listener maintains a list of dispatchers that are available for each service name, along with the connection load (number of concurrent connections) for each dispatcher.

Connection requests are routed to the lightest loaded dispatcher that is servicing a given service name. Users remain connected to the same dispatcher for the duration of a session.

Unlike dedicated server processes, a single dispatcher can manage hundreds of user sessions.

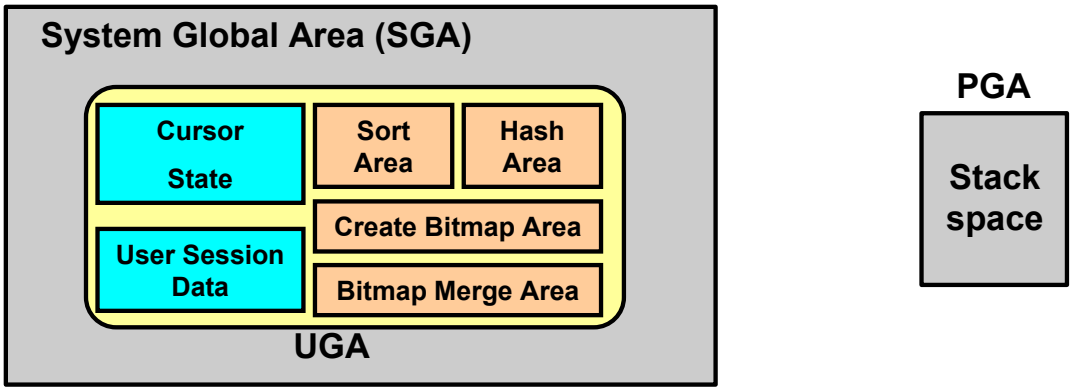
Dispatchers do not actually handle the work of user requests. Instead, they pass user requests to a common queue located in the shared pool portion of the SGA.

Shared server processes take over most of the work of dedicated server processes, pulling requests from the queue and processing them until they are complete.

Because a single user session may have requests processed by multiple shared server processes, most of the memory structures that are usually stored in the PGA must be in a shared memory location (by default, in the shared pool). However, if the large pool is configured or if `SGA_TARGET` is set for automatic memory management, these memory structures are stored in the large pool portion of the SGA.

SGA and PGA

Oracle Shared Server: User session data is held in the SGA.



Remember to consider shared server memory requirements when sizing the SGA.

ORACLE

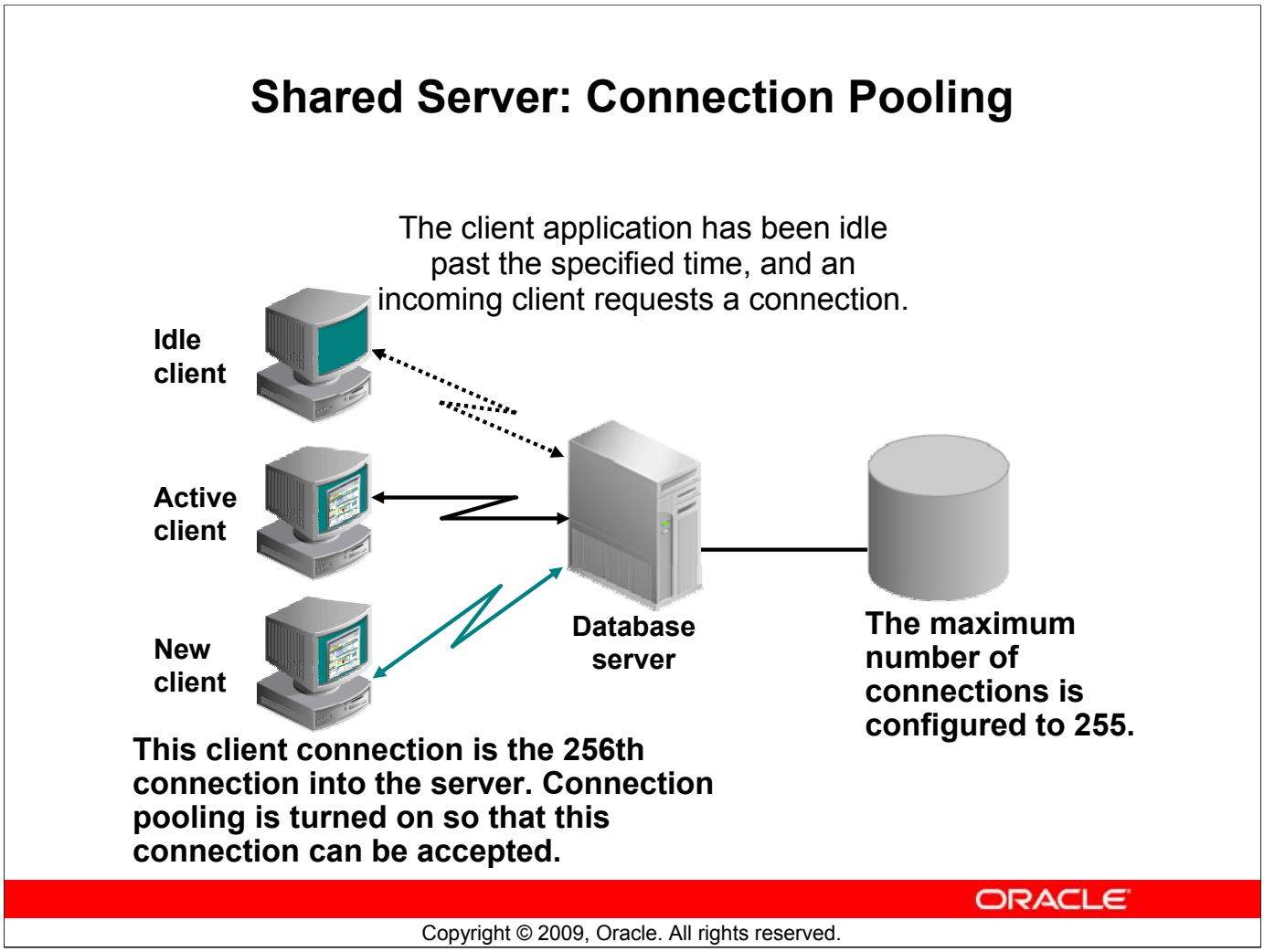
SGA and PGA

The contents of the SGA and PGA differ when dedicated servers or shared servers are used:

- Text and parsed forms of all SQL statements are stored in the SGA.
- The cursor state contains run-time memory values for the SQL statement, such as rows retrieved.
- User-session data includes security and resource usage information.
- The stack space contains local variables for the process.

Technical Note

The change in the SGA and PGA is transparent to the user; however, if you are supporting multiple users, you need to increase the `LARGE_POOL_SIZE` initialization parameter. Each shared server process must access the data spaces of all sessions so that any server can handle requests from any session. Space is allocated in the SGA for each session's data space. You limit the amount of space that a session can allocate by setting the `PRIVATE_SGA` resource limit in the Database Services region of the General page of the user's profile.



Shared Server: Connection Pooling

The connection-pooling feature enables the database server to time out an idle session and use the connection to service an active session. The idle logical session remains open, and the physical connection is automatically reestablished when the next request comes from that session. Therefore, Web applications can allow larger numbers of concurrent users to be accommodated with existing hardware. Connection pooling is configurable through the shared server.

In this example, the Oracle database server has been configured with 255 connections. One of the clients has been idle past the specified time. Connection pooling makes this connection available to an incoming client connection, which is the 256th connection. When the idle client has more work to do, the connection is reestablished for that client with another client's idle connection.

When Not to Use a Shared Server

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

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

Dispatcher

Dedicated server process

ORACLE

When Not to Use a Shared Server

The Oracle Shared Server architecture is an efficient process and memory use model, but it is not appropriate for all connections. Because of the common request queue and the fact that many users may share a dispatcher response queue, shared servers do not perform well with operations that must deal with large sets of data, such as warehouse queries or batch processing.

Backup and recovery sessions that use Oracle Recovery Manager (discussed in later lessons) also deal with very large data sets and must make use of dedicated connections.

Many administration tasks must not (and cannot) be performed by using shared server connections. These include starting up and shutting down the instance, creating tablespaces and data files, maintaining indexes and tables, analyzing statistics, and many other tasks that are commonly performed by the DBA. All DBA sessions must choose dedicated servers.

Configuring Communication Between Databases

- Sending data or messages between sites requires network configuration on both sites.
- You must configure the following:
  - Network connectivity (for example, TNSNAMES.ora)
  - Database links

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

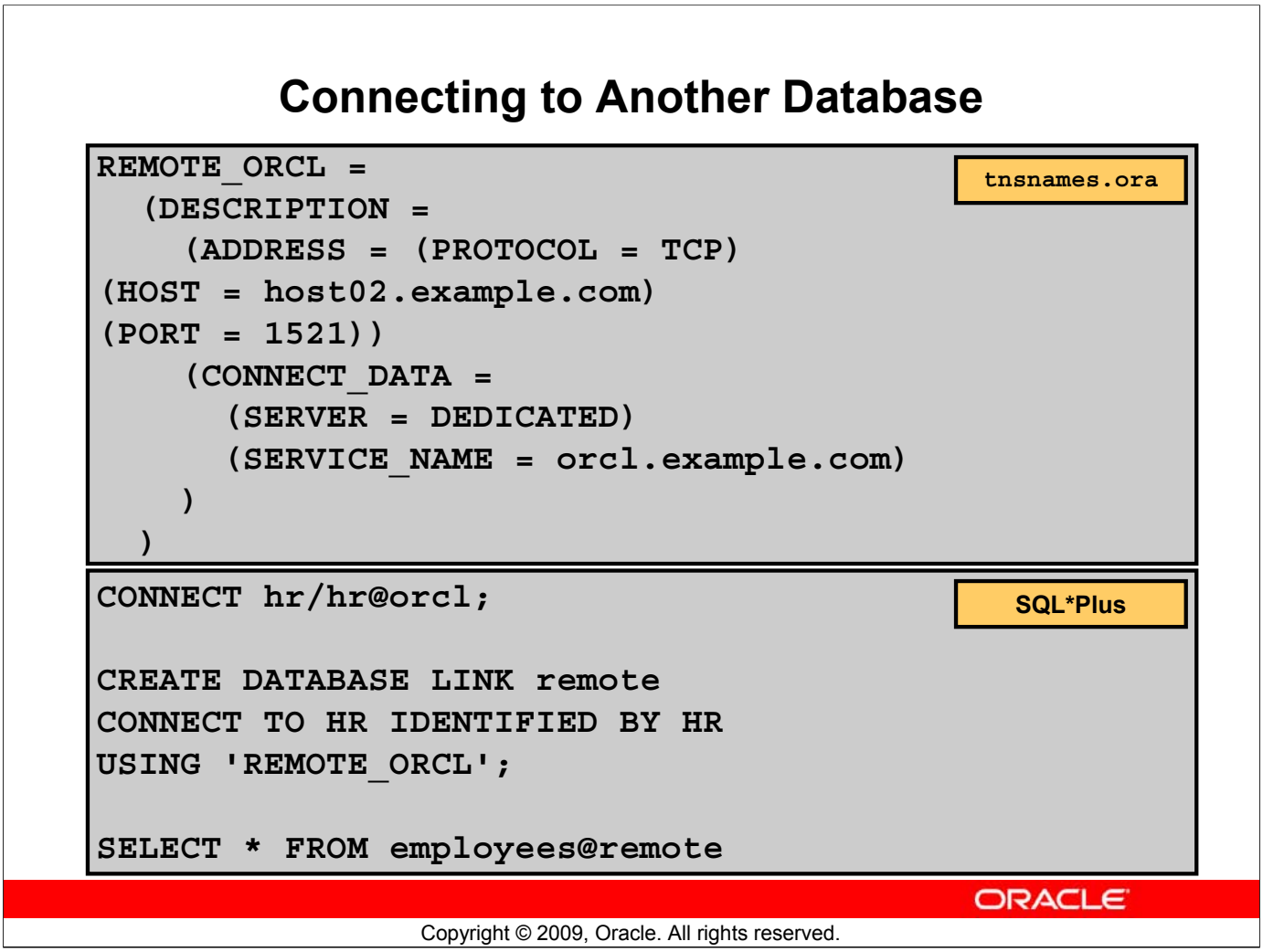

Configuring Communication Between Databases

A database link is a schema object in one database that enables you to access objects on another database. The other database need not be an Oracle database system. However, to access non-Oracle systems, you must use Oracle Heterogeneous Services.

To create a private database link, you must have the `CREATE DATABASE LINK` system privilege. To create a public database link, you must have the `CREATE PUBLIC DATABASE LINK` system privilege. You must also have the `CREATE SESSION` system privilege on the remote Oracle database.

When an application uses a database link to access a remote database, Oracle Database establishes a database session in the remote database on behalf of the local request. The `CONNECT TO` clause that is used in creating a database link determines how the connection is established on the remote database. You can create fixed user, current user, and connected user database links. Current user links are available only through the Oracle Advanced Security option. The example in the slide shows the syntax to create a fixed user database link.

After you create a database link, you can use it to refer to tables and views on the other database. In SQL statements, you can refer to a table or view on the other database by appending `@dblink` to the table or view name. You can query a table or view on the other database or use any `INSERT`, `UPDATE`, `DELETE`, or `LOCK TABLE` statement for the table.



Connecting to Another Database

The slide shows the `tnsnames.ora` entry that is needed before creating a database link. The example shows a fixed user database link called `REMOTE` that is connecting to the user `HR` by using the connect string `REMOTE_ORCL`. After you create a database link, you can use it to refer to tables and views on the other database.

The description of the view is as follows:

```
SQL> DESC DBA_DB_LINKS
 Name                      Null?    Type
 ------------------------- -------- ----------------------------
 OWNER                     NOT NULL VARCHAR2(30)
 DB_LINK                   NOT NULL VARCHAR2(128)
 USERNAME                  VARCHAR2(30)
 HOST                      VARCHAR2(2000)
 CREATED                   NOT NULL DATE

SQL> select owner, db_link, username from dba_db_links;

OWNER        DB_LINK                   USERNAME
------------ ------------------------- ---------
HR           REMOTE.EXAMPLE.COM        HR
```

Quiz

Which configuration files are used to configure the listener?

1. listener.ora
2. listener.conf
3. tnsnames.ora
4. tnsnames.conf
5. sqlnet.ora
6. sqlnet.conf

Answers: 1, 5

Quiz

When using the shared server process architecture, the PGA is relocated into the SGA.

1. True
2. False

Answer: 2

Summary

In this lesson, you should have learned how to:

- Use Enterprise Manager to:
  - Create additional listeners
  - Create Oracle Net Service aliases
  - Configure connect-time failover
  - Control the Oracle Net Listener
- Use `tnsping` to test Oracle Net connectivity
- Identify when to use shared servers and when to use dedicated servers

ORACLE

Practice 6 Overview: Working with Oracle Network Components

This practice covers the following topics:

- Configuring local Names Resolution to connect to another database
- Creating a second listener for connect-time failover

ORACLE

7

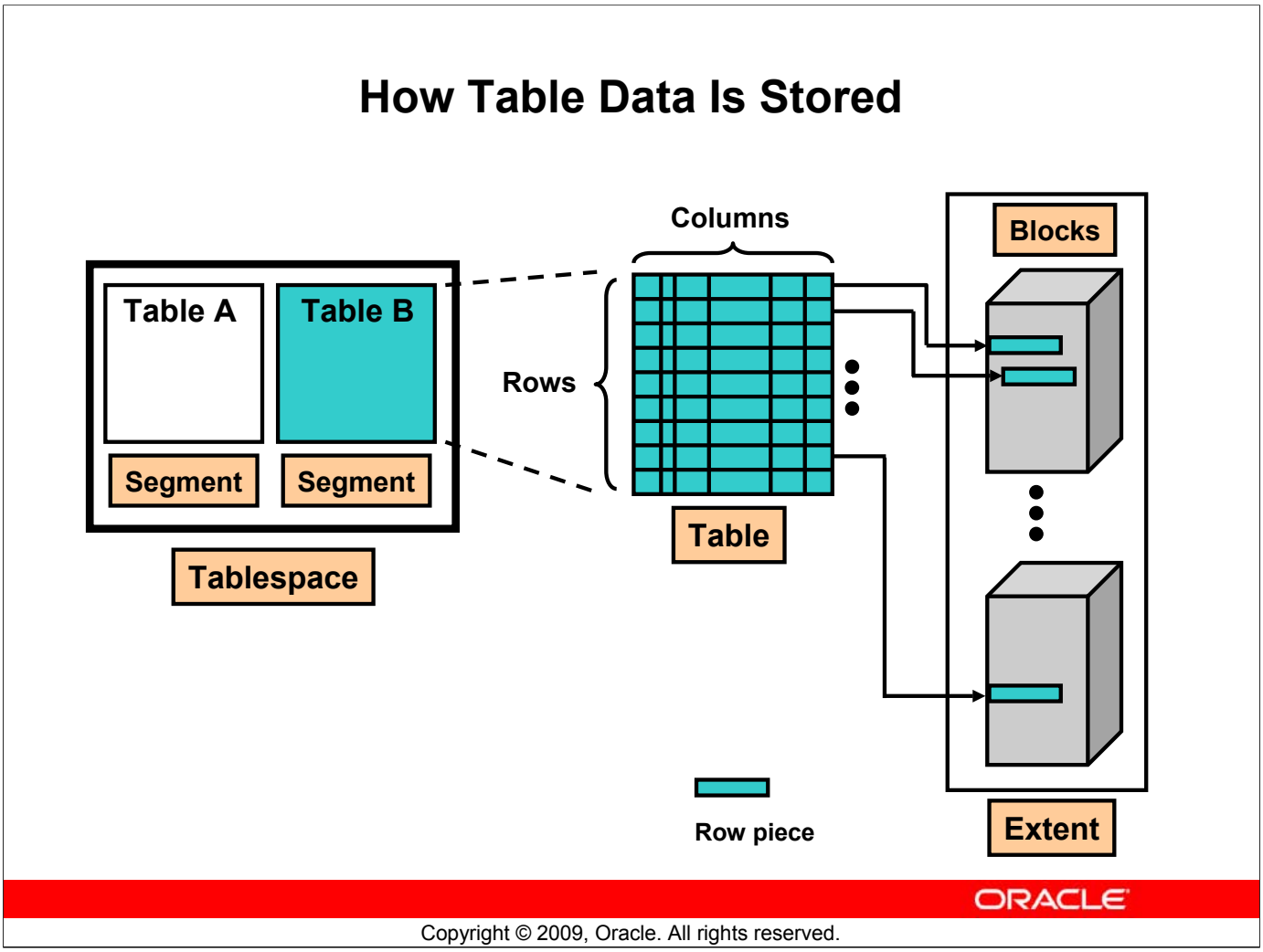
Managing Database Storage Structures

Objectives

After completing this lesson, you should be able to:

- Describe the storage of table row data in blocks
- Create and manage tablespaces
- Obtain tablespace information

ORACLE

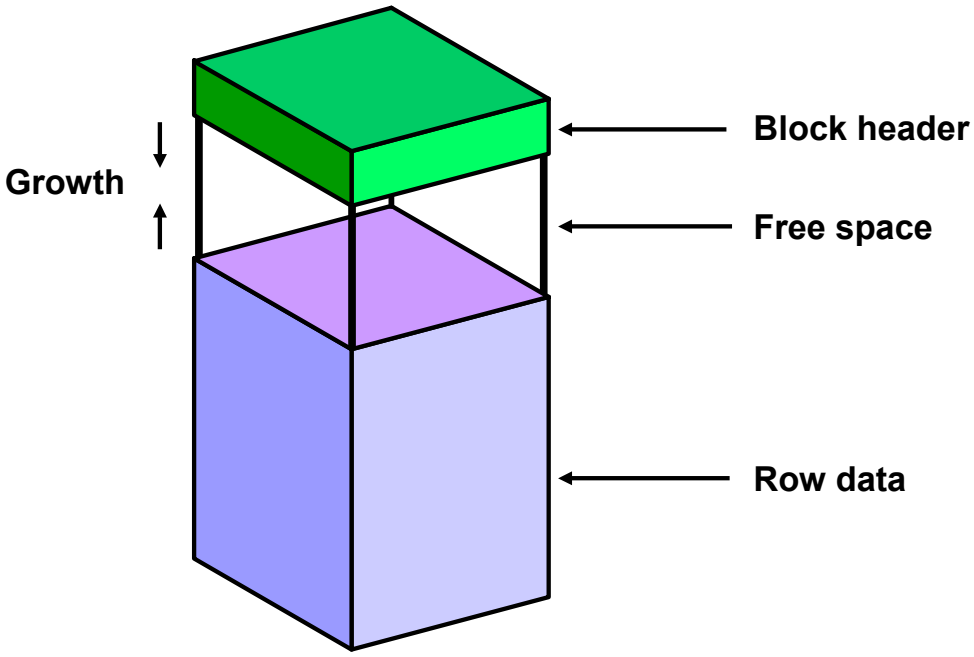


How Table Data Is Stored

When a table is created, a segment is created to hold its data. A tablespace contains a collection of segments.

Logically, a table contains rows of column values. A row is ultimately stored in a database block in the form of a row piece. It is called a *row piece* because, under some circumstances, the entire row may not be stored in one place. This happens when an inserted row is too large to fit into a single block (chained row) or when an update causes an existing row to outgrow the available free space of the current block (migrated row). Row pieces are also used when a table has more than 255 columns. In this case the pieces may be in the same block (intra-block chaining) or across multiple blocks.

Database Block: Contents

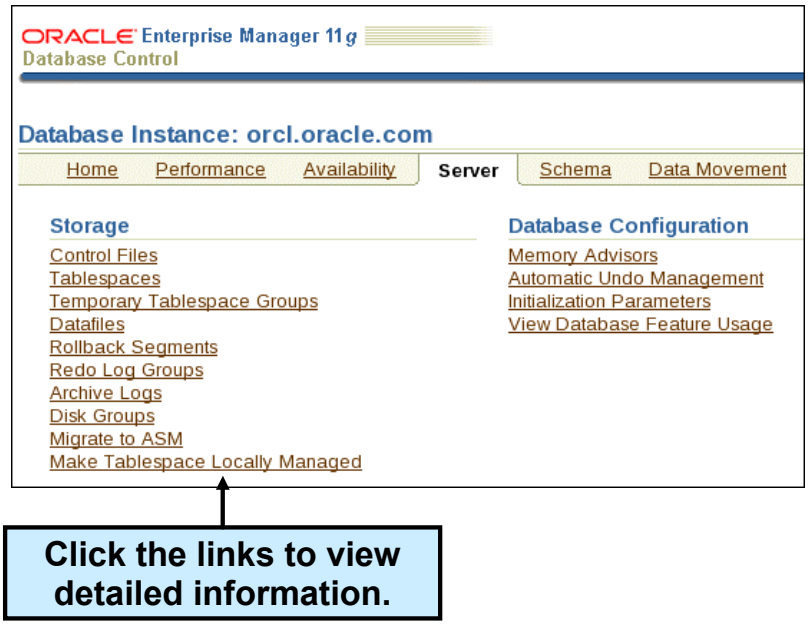


Database Block: Contents

- **Block header:** The block header contains the segment type (such as table or index), data block address, table directory, row directory, and transaction slots of approximately 23 bytes each, which are used when modifications are made to rows in the block. The block header grows downward from the top.
- **Row data:** This is the actual data for the rows in the block. Row data space grows upward from the bottom.
- **Free space:** Free space is in the middle of the block, enabling the header and the row data space to grow when necessary. Row data takes up free space as new rows are inserted or as columns of existing rows are updated with larger values.
  Examples of events that cause header growth:
  - Row directories that need more row entries
  - More transaction slots required than initially configured

  Initially, the free space in a block is contiguous. However, deletions and updates may fragment the free space in the block. The free space in the block is coalesced by the Oracle server when necessary.

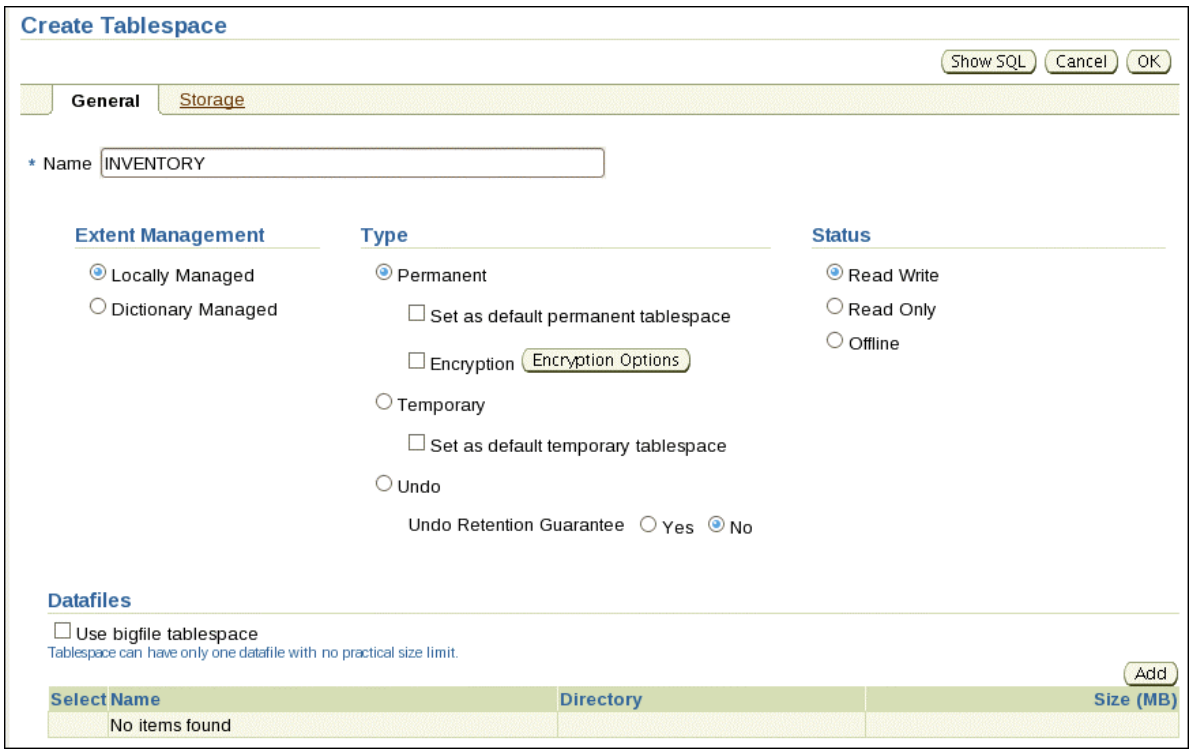
Exploring the Storage Structure



Exploring the Storage Structure

Logical data structures are stored in the physical files of the database. You can easily view the logical structures of your database through Enterprise Manager (EM). Detailed information about each structure can be obtained by clicking the links in the Storage region of the Server page.

Creating a New Tablespace

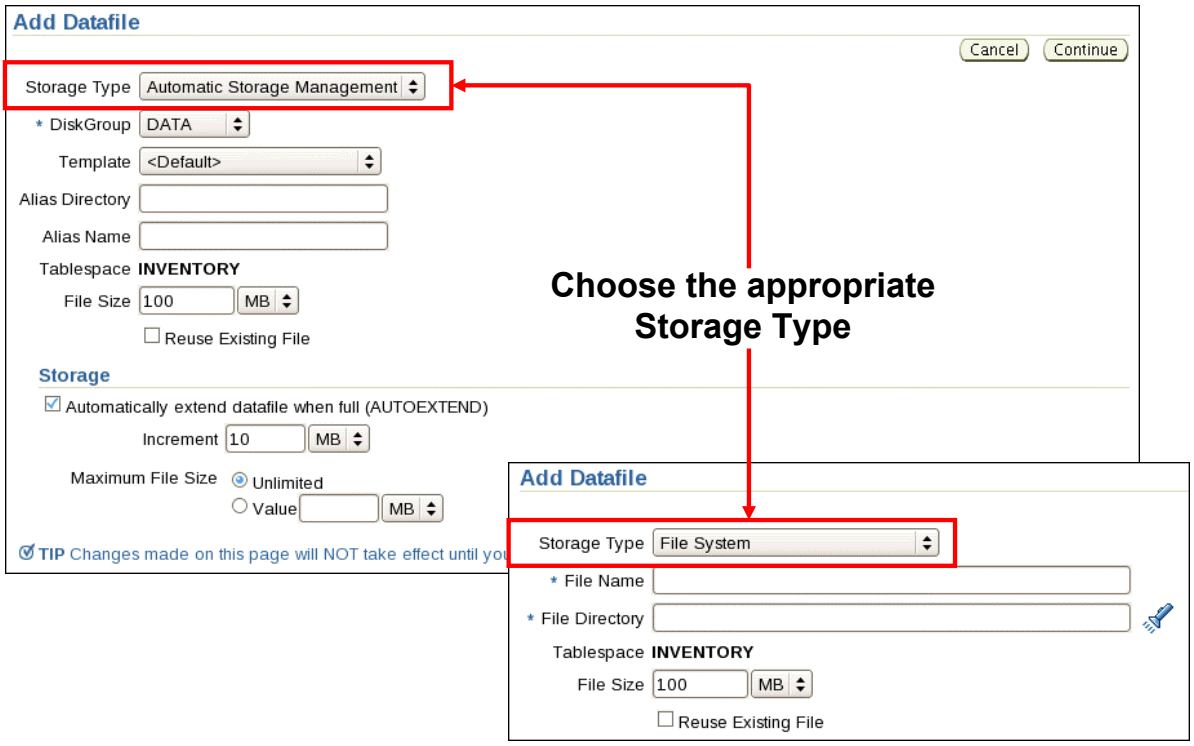


ORACLE

Creating a New Tablespace

1. Click the Server tab, and then click Tablespaces under the Storage heading.
2. Click Create.
   Note: If you want to create a tablespace that is like an existing tablespace, select an existing tablespace and then select Create Like from the Actions menu. Click Go.
   The Create Tablespace page appears.
3. Enter a name for the tablespace.
4. Under the Extent Management heading, select Locally Managed.
   The extents of a locally managed tablespace are managed efficiently in the tablespace by the Oracle database server. For a dictionary-managed tablespace, you must manage extents more actively, and data dictionary access is required for tracking them. Dictionary-managed option is provided only for backward compatibility; Oracle does not recommend their use.
5. Under the Type heading, select Permanent.
   Permanent tablespaces store permanent database objects that are created by the system or users.
6. Under the Status heading, select Read Write.
   Read Write status means that users can read and write to the tablespace after it is created. This is the default.
7. In the Datafiles region of the page, click Add to add data files to the tablespace.

Creating a New Tablespace



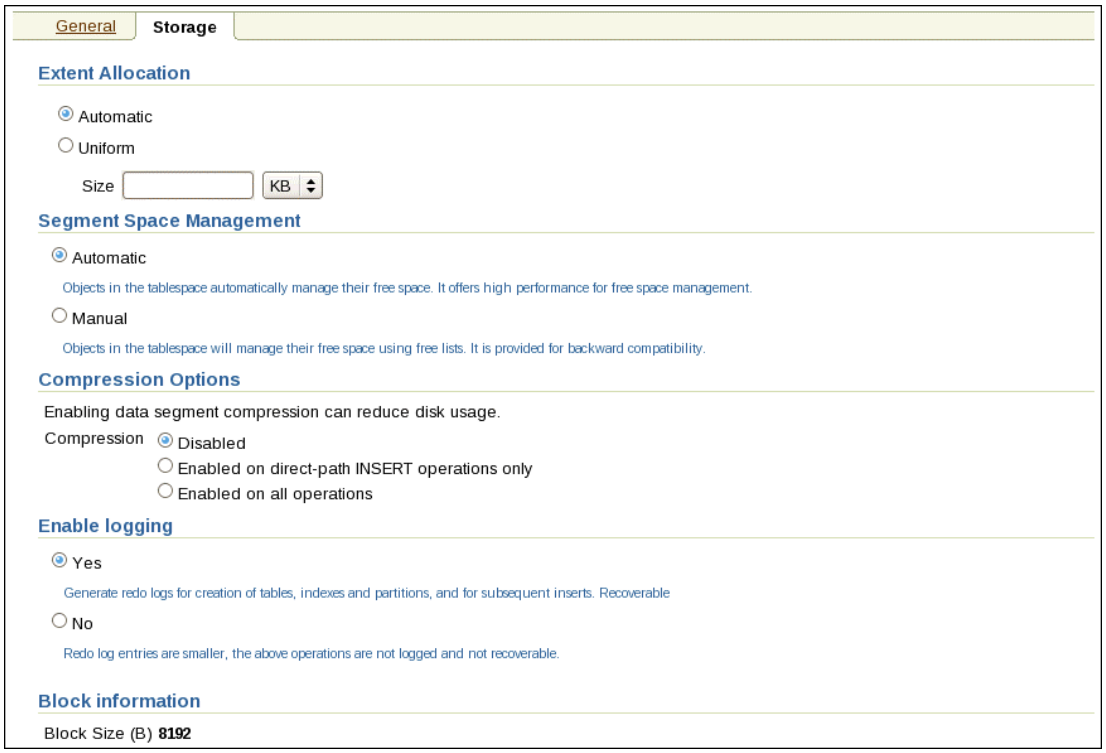
Creating a New Tablespace (continued)

A tablespace must have at least one file. Choose the appropriate storage type, depending on your environment. Bigfile tablespaces are used with extremely large databases, in which ASM or other logical volume managers support the striping or redundant array of independent disks (RAID) and dynamically extensible logical volumes.

8. On the Add Datafiles page select the desired storage type and enter the required information. For ASM choose the desired disk group. For File System enter a file name and file directory for the datafile.
9. Enter the desired file size.
10. In the Storage region, select "Automatically extend datafile when full (AUTOEXTEND)" and then specify an amount in the Increment field. This causes the data file to extend automatically each time it runs out of space. It is limited, of course, by the physical media on which it resides. Leave Maximum File Size as Unlimited or enter a maximum size. Click Continue to return to the Create Tablespace page.
12. Back on the Create Tablespace page you can click the Storage tab to make modifications to the storage options for this tablespace if desired. In most cases, you can just accept all the defaults on the Storage page. Click OK to create the tablespace.

Note: These steps show you how to quickly create a tablespace for most situations. You may need to change some options depending on your particular requirements.

Storage for Tablespaces



ORACLE

Storage for Tablespaces

Extent Allocation: The extents in a locally managed tablespace can be allocated in one of these two ways:

- **Automatic:** Also called *autoallocate*, it specifies that the sizes of the extents in the tablespace are system managed. You cannot specify Automatic for a temporary tablespace.
- **Uniform:** It specifies that the tablespace is managed with uniform extents of a size that you specify. The default size is 1 MB. All extents of temporary tablespaces are uniform. You cannot specify Uniform for an undo tablespace.

Segment Space Management: Segment space management in a locally managed tablespace can be specified as:

- **Automatic:** The Oracle database uses bitmaps to manage the free space in segments. The bitmap describes the status of each data block in a segment with respect to the amount of space in the block that is available for inserting rows. As more or less space becomes available in a data block, the new state is reflected in the bitmap. With bitmaps, the Oracle database manages free space more automatically. As a result, this form of space management is called Automatic Segment Space Management (ASSM).

Storage for Tablespaces (continued)

- **Manual:** This specifies that you want to use free lists for managing free space in segments. Free lists are lists of data blocks that have space available for inserting rows. This form of managing space in segments is called *manual segment space management* because of the need to specify and tune the `PCTUSED`, `FREELISTS`, and `FREELIST GROUPS` storage parameters for schema objects created in the tablespace. This is supported for backward compatibility; it is recommended that you use ASSM.

Compression Options: Data segment compression is disabled by default. Enabling data segment compression can save disk space usage, reduce memory use in the buffer cache, and speed up query execution during reads. There is, however, a cost in CPU overhead for data loading and DML. It is especially useful in online analytical processing (OLAP) systems, where there are lengthy read-only operations, but can also be used in online transaction processing (OLTP) systems.

For more details about when to use compression clause, see the *Oracle Database Administrator's Guide*.

Logging: The logging clause sets the default logging value for any segment created in the tablespace. Changes made to objects in the tablespace are written to the redo log. If logging is not enabled, any direct loads using SQL*Loader and direct load `INSERT` operations are not written to the redo log, and the objects are thus unrecoverable in the event of data loss. When an object is created without logging enabled, you must back up those objects if you want them to be recoverable. Choosing not to enable logging can have a significant impact on the ability to recover objects in the future. Use with caution. For more details about the logging clause, see the *Oracle Database SQL Reference*.

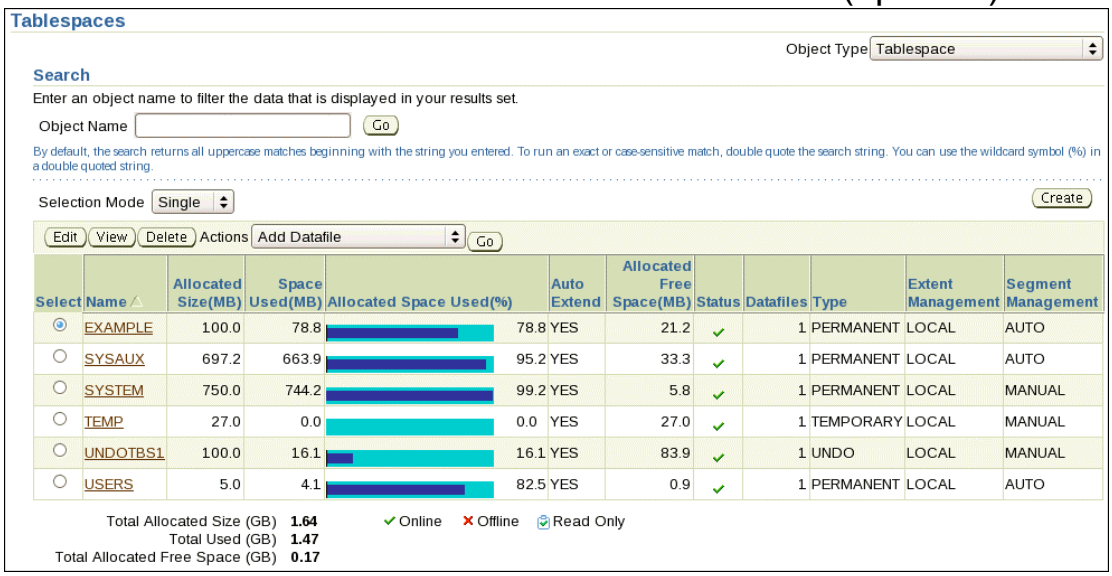
Note: If `FORCE LOGGING` mode is in effect for the database, it takes precedence over the tablespace logging setting. The database can be put into `FORCE LOGGING` mode at the time of database creation or after database creation using the `alter database force logging` command.

Block Information: This region shows the block size that is used for the tablespace being created. It is displayed here as a read-only value. If you set any of the alternate block size initialization parameters (`DB_nK_CACHE_SIZE`), those other values would be listed here as an option.

For more information about defining other block sizes, see the *Oracle Database Administrator's Guide*.

Tablespaces in the Preconfigured Database

- SYSTEM
- SYSAUX
- TEMP
- UNDOTBS1
- USERS
- EXAMPLE (optional)



Tablespaces in the Preconfigured Database

The following tablespaces are created in the preconfigured database in this course:

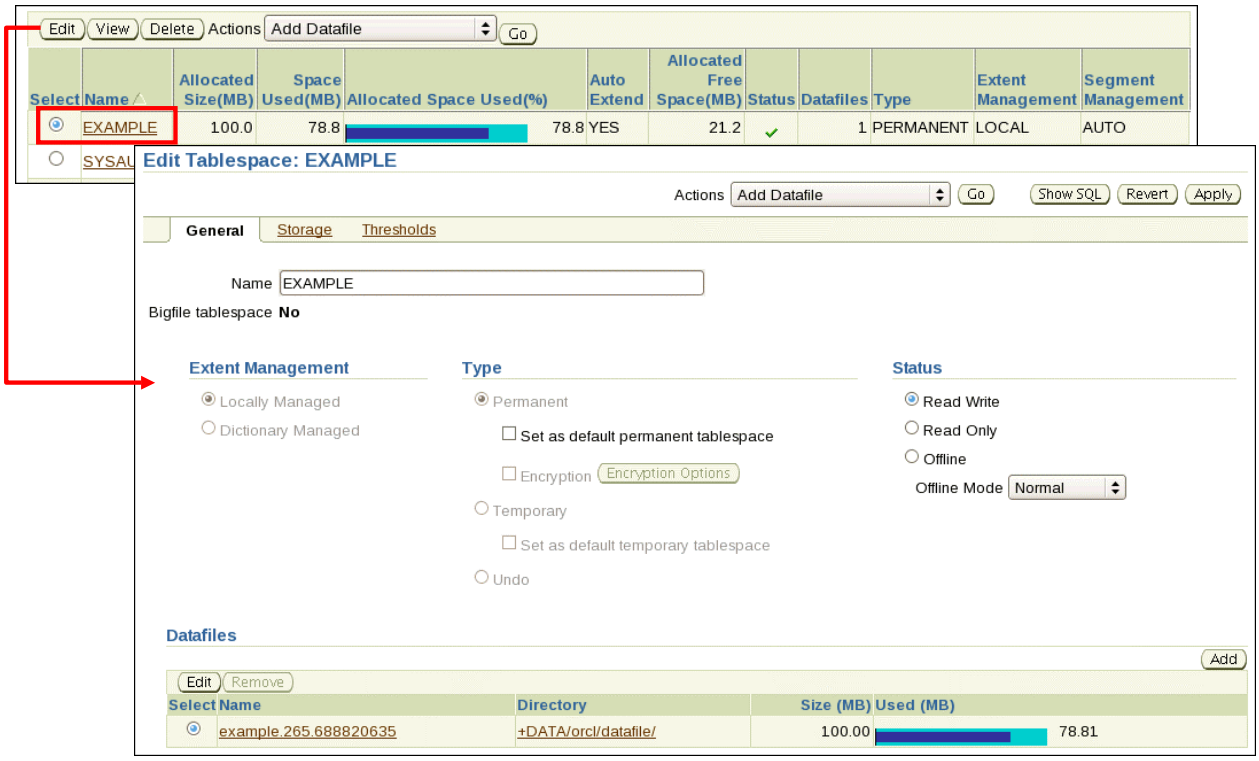
- **SYSTEM:** The SYSTEM tablespace is used by the Oracle server to manage the database. It contains the data dictionary and tables that contain administrative information about the database. These are all contained in the SYS schema and can be accessed only by the SYS user or other administrative users with the required privilege.
- **SYSAUX:** This is an auxiliary tablespace to the SYSTEM tablespace. Some components and products that used the SYSTEM tablespace or their own tablespaces in earlier releases of Oracle Database now use the SYSAUX tablespace. Every Oracle Database 10*g* (or later release) database must have a SYSAUX tablespace.
  In Enterprise Manager (EM), you can see a pie chart of the contents of this tablespace. To do this, click Tablespaces on the Administration page. Select SYSAUX and click Edit. Then click the Occupants tab. After creation, you can monitor the space usage of each occupant in the SYSAUX tablespace by using EM. If you detect that a component is taking too much space in the SYSAUX tablespace, or if you anticipate that it will, you can move the occupant into a different tablespace by selecting one of the occupants and clicking Change Tablespace.

Tablespaces in the Preconfigured Database (continued)

- **TEMP:** Your temporary tablespace is used when you execute a SQL statement that requires the creation of temporary segments (such as a large sort or the creation of an index). Just as each user is assigned a default tablespace for storing created data objects, each user is assigned a temporary tablespace. The best practice is to define a default temporary tablespace for the database, which is assigned to all newly created users unless otherwise specified. In the preconfigured database, the `TEMP` tablespace is specified as the default temporary tablespace. This means that if no temporary tablespace is specified when the user account is created, Oracle Database assigns this tablespace to the user.
- **UNDOTBS1:** This is the undo tablespace used by the database server to store undo information. If a database uses Automatic Undo Management, then it can only use a single undo tablespace at any given time. This tablespace is created at database creation time.
- **USERS:** This tablespace is used to store user objects and data. If no default tablespace is specified when a user is created then the `USERS` tablespace is the default tablespace for all objects created by that user. For the `SYS` and `SYSTEM` users, the default permanent tablespace is `SYSTEM`.
- **EXAMPLE:** This tablespace contains the sample schemas that can be installed when you create the database. The sample schemas provide a common platform for examples. Oracle documentation and courseware contain examples based on the sample schemas.

Note: To simplify administration, it is common to have a tablespace for indexes alone.

Altering a Tablespace



Altering a Tablespace

After you create a tablespace, you can alter it in several ways as the needs of your system change.

Renaming: Enter a new name for the tablespace and click Apply.

Changing the status: A tablespace can be in one of three different statuses or states. Any of the following three states may not be available because their availability depends on the type of tablespace.

- **Read Write:** The tablespace is online and can be read from and written to.
- **Read Only:** Specify read-only to place the tablespace in transition read-only mode. In this state, existing transactions can be completed (committed or rolled back), but no further data manipulation language (DML) operations are allowed on objects in the tablespace. The tablespace is online while in the read-only state. You cannot make the `SYSTEM` or `SYSAUX` tablespaces read-only.

  Note: The undo and temporary tablespaces cannot be made read-only.

Altering a Tablespace (continued)

- **Offline:** You can take an online tablespace offline so that this portion of the database is temporarily unavailable for general use. The rest of the database is open and available for users to access data. When you take it offline, you can use the following options:
  - **Normal:** A tablespace can be taken offline normally if no error conditions exist for any of the data files of the tablespace. Oracle Database ensures that all data is written to disk by taking a checkpoint for all data files of the tablespace as it takes them offline.
  - **Temporary:** A tablespace can be taken offline temporarily even if there are error conditions for one or more files of the tablespace. Oracle Database takes the data files (which are not already offline) offline, performing checkpointing on them as it does so. If no files are offline, but you use the Temporary clause, media recovery is not required to bring the tablespace back online. However, if one or more files of the tablespace are offline because of write errors, and you take the tablespace offline temporarily, the tablespace requires recovery before you can bring it back online.
  - **Immediate:** A tablespace can be taken offline immediately without Oracle Database taking a checkpoint on any of the data files. When you specify Immediate, media recovery for the tablespace is required before the tablespace can be brought online. You cannot take a tablespace offline immediately if the database is running in `NOARCHIVELOG` mode.
  - **For Recover:** The `FOR RECOVER` setting has been deprecated. The syntax is supported for backward compatibility.

Note: System tablespaces may not be taken offline.

Changing the size: You can add space to an existing tablespace by either adding data files to the tablespace or changing the size of an existing data file.

- To add a new data file to the tablespace, click Add. Then enter the information about the data file on the Add Datafile page.
  Note: You cannot add additional data files to bigfile tablespaces.
- To change the size of an existing data file, select the data file in the Datafiles region of the Edit Tablespace page by clicking the name of the data file, or select the data file and click Edit. Then, on the Edit Datafile page, you can change the size of the data file. You can make the tablespace either larger or smaller. However, you cannot make a data file smaller than the used space in the file; if you try to do so, you get the following error:

```
ORA-03297: file contains used data beyond requested RESIZE
value
```

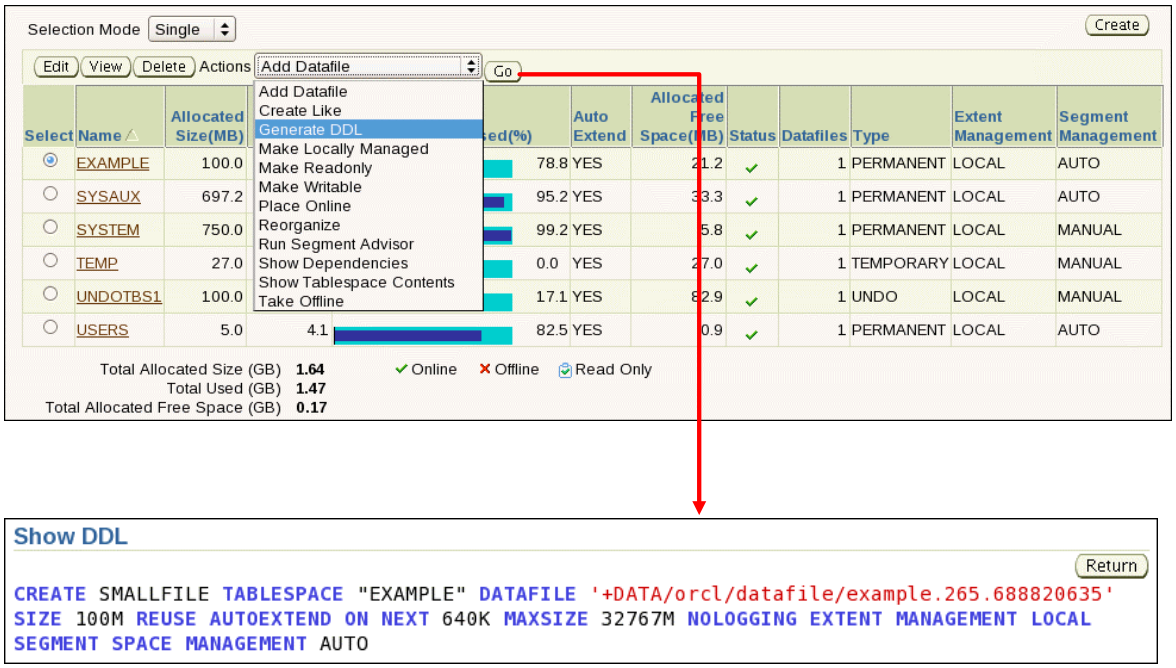
Storage options: Click Storage to change the logging behavior of the tablespace.

Thresholds: Click Thresholds to change the point at which a warning or critical level of space usage is reached on the tablespace. You have three options:

- **Use Database Default Thresholds:** This uses preset defaults, and you have the option of setting these defaults.
- **Specify Thresholds:** This enables you to set thresholds for this particular tablespace.
- **Disable Thresholds:** This turns off space usage alerts for this tablespace.

Note: Space utilization is checked only every 10 minutes by default, so it may take a few minutes for a threshold alert to register.

Actions with Tablespaces



Actions with Tablespaces

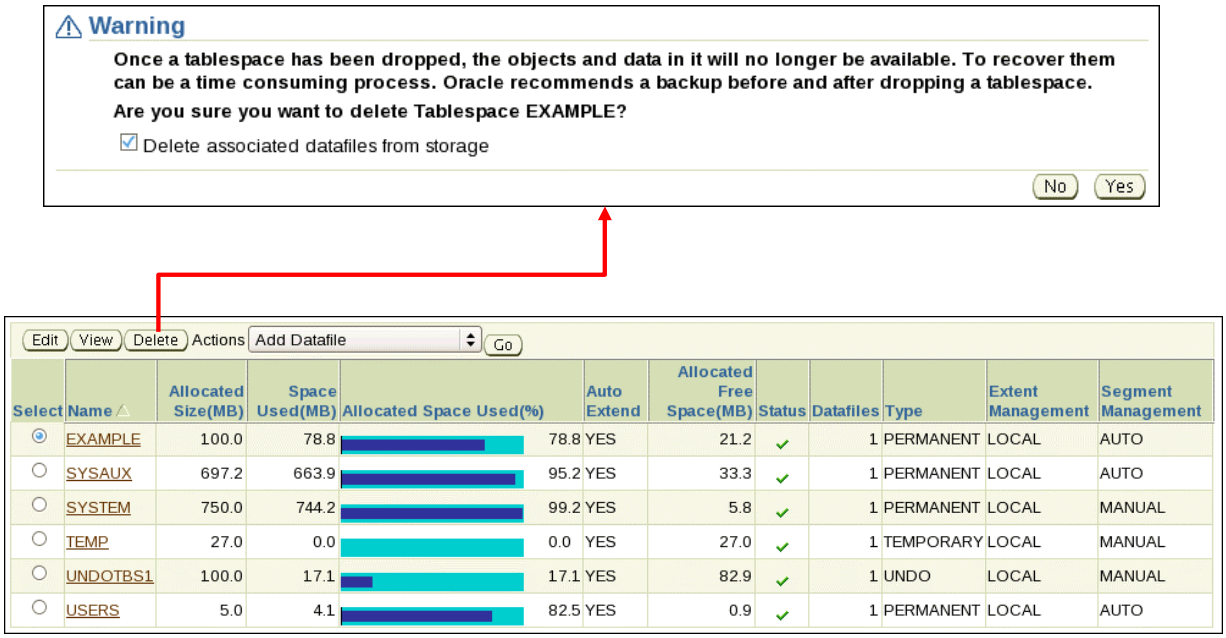
Using the Actions menu, you can perform a variety of tasks with your tablespaces. Select a tablespace and then select the action that you want to perform.

- **Add Datafile:** Adds a data file to the tablespace, which makes the tablespace larger
- **Create Like:** Creates another tablespace by using the tablespace as a template
- **Generate DDL:** Generates the data definition language (DDL) statement that creates the tablespace. This can then be copied and pasted into a text file for use as a script or for documentation purposes.
- **Make Locally Managed:** Converts the tablespace to locally managed if the tablespace is currently dictionary managed. This conversion is one-way only; you cannot convert the tablespace back to dictionary managed. You can use the PL/SQL package `DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_FROM_LOCAL` to convert to dictionary managed if needed.
- **Make Readonly:** Stops all writes to the tablespace. Current transactions are allowed to complete, but no new DML or other write activities are allowed to start on the tablespace. This appears only if the tablespace is currently not read-only.
- **Make Writable:** Allows DML and other write activities to be initiated on objects in the tablespace. This appears only if the tablespace is currently not writable.

Actions with Tablespaces (continued)

- **Place Online:** Brings a currently offline tablespace online
- **Reorganize:** Starts the Reorganization Wizard, which you can use to move objects around in the tablespace to reclaim space that otherwise may not be used. This is a task that should be performed during off-peak usage of the objects in the tablespace.
- **Run Segment Advisor:** Starts the Segment Advisor, which you can use to determine whether an object has space available for reclamation on the basis of the level of space fragmentation in the object. At the tablespace level, advice is generated for every segment in the tablespace.
- **Show Dependencies:** Shows objects on which this tablespace depends, or objects that depend on this tablespace
- **Show Tablespace Contents:** Shows information about all the segments in the tablespace, including a graphical map of all of the extents
- **Take Offline:** Makes a currently online tablespace unavailable. The tablespace is not deleted or dropped; it is just unavailable.

Dropping Tablespaces



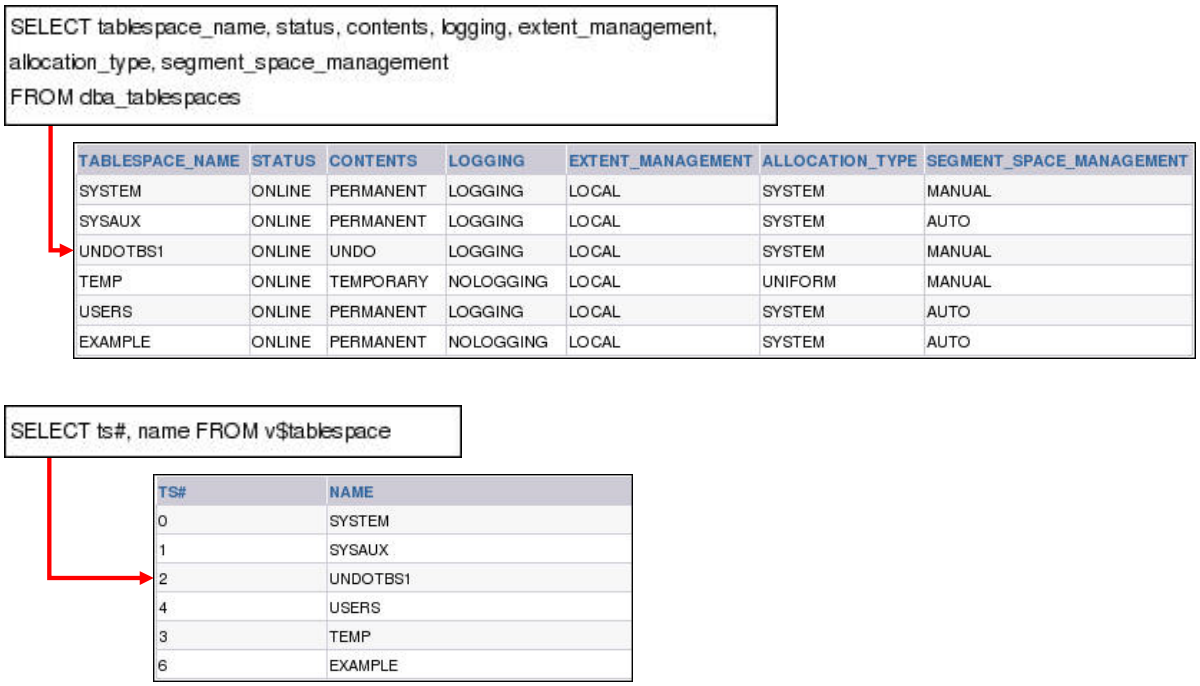
Dropping Tablespaces

You can drop a tablespace and its contents (the segments contained in the tablespace) from the database if the tablespace and its contents are no longer required. You must have the `DROP TABLESPACE` system privilege to drop a tablespace.

When you drop a tablespace, the file pointers in the control file of the associated database are removed. If you are using Oracle-managed files (OMF), the underlying operating system files are also removed. Otherwise, without OMF, you can optionally direct the Oracle server to delete the operating system files (data files) that constitute the dropped tablespace. If you do not direct the Oracle server to delete the data files at the same time that it deletes the tablespace, you must later use the appropriate commands of your operating system if you want them to be deleted.

You cannot drop a tablespace that contains active segments. For example, if a table in the tablespace is currently being used or if the tablespace contains undo data that is needed to roll back uncommitted transactions, you cannot drop the tablespace. The tablespace can be online or offline, but it is best to take the tablespace offline before dropping it.

Viewing Tablespace Information



Viewing Tablespace Information

Click View to see information about the selected tablespace. On the View Tablespace page, you can also click Edit to alter the tablespace.

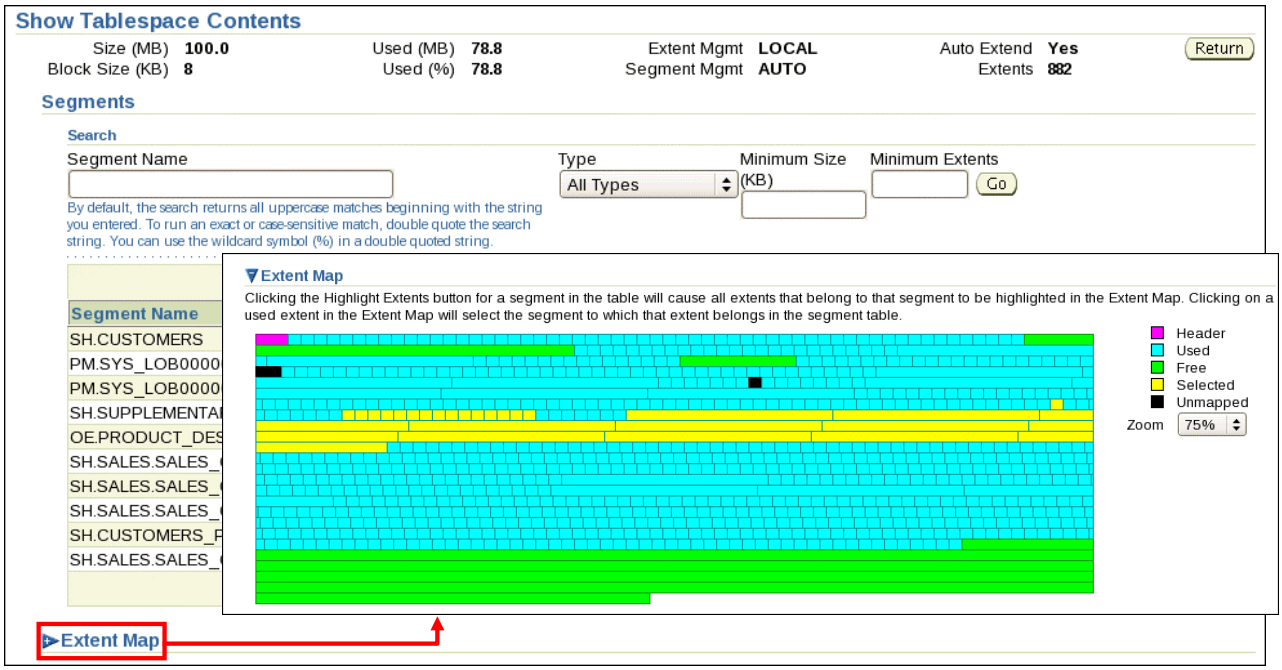
Tablespace and data file information can also be obtained by querying the following:

- **Tablespace information:**
  - `DBA_TABLESPACES`
  - `V$TABLESPACE`
- **Data file information:**
  - `DBA_DATA_FILES`
  - `V$DATAFILE`

Note: The `V$DBFILE` view displays all data files in the database. This view is retained for historical compatibility. Use of `V$DATAFILE` is recommended instead.

- **Temp file information:**
  - `DBA_TEMP_FILES`
  - `V$TEMPFILE`

Viewing Tablespace Contents



Viewing Tablespace Contents

From either the main Tablespaces page or the page for a particular tablespace, select Show Tablespace Contents from the Actions drop-down list and click Go. On the Show Tablespace Contents page, detailed information about the tablespace is displayed, including a list of the segments in the tablespace, the type of each segment, the segment size, and the number of extents in each segment. Any of these four values can be used to sort the list by clicking the column header, or to filter the list by entering values in the Search region. For a dictionary-managed tablespace, additional columns are displayed:

- Max Extents
- Next
- Percent Increase

To see a list of extents, click the link in the Extents column.

To view extents in a graphical way, expand the “Extent map” and move the cursor over individual extents. The following information is displayed:

- Name of the segment to which the extent belongs
- Extent ID
- Block ID
- Extent size in blocks
- Data file in which the extent is stored

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	Default location for the fast recovery area

Example:

```
SQL> ALTER SYSTEM SET DB_CREATE_FILE_DEST = '+DATA';
SQL> CREATE TABLESPACE tbs_1;
```

ORACLE

Oracle-Managed Files (OMF)

Oracle-managed files eliminates the need for you to directly manage the operating system files in an Oracle database. You specify operations in terms of database objects rather than file names. The database internally uses standard file system interfaces to create and delete files as needed for the following database structures:

- Tablespaces
- Redo log files
- Control files
- Archived logs
- Block change tracking files
- Flashback logs
- RMAN backups

A database can have a mixture of Oracle-managed and unmanaged files. The file system directory specified by either of these parameters must already exist; the database does not create it. The directory must also have permissions for the database to create the files in it.

The example shows that after `DB_CREATE_FILE_DEST` is set, the `DATAFILE` clause can be omitted from a `CREATE TABLESPACE` statement. The data file is created in the location specified by `DB_CREATE_FILE_DEST`. When you create a tablespace as shown, default values are assigned to all parameters.

Oracle-Managed Files (OMF) (continued)

Oracle-managed files have a specific naming format. For example, on Linux and Unix based systems the following format is used:

```
<destination_prefix>/o1_mf_%t_%u_.dbf
```

Do not rename an Oracle-managed file. The database identifies an Oracle-managed file based on its name. If you rename the file, the database is no longer able to recognize it as an Oracle-managed file and will not manage the file accordingly.

The following example sets the default location for datafile creations to `/u01/oradata` and then creates a tablespace `tbs_1` with a datafile in that location.

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

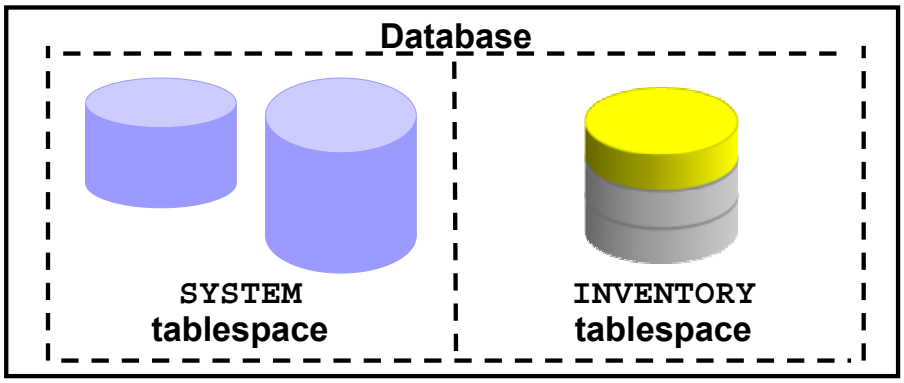
By default, Oracle-managed data files, including those for the `SYSTEM` and `SYSAUX` tablespaces, are 100MB and autoextensible.

Note: By default, ASM uses OMF files, but if you specify an alias name for an ASM data file at tablespace creation time or when adding an ASM data file to an existing tablespace, then that file will not be OMF.

Enlarging the Database

You can enlarge the database in the following ways:

- Creating a new tablespace
- Adding a data file to an existing smallfile tablespace
- Increasing the size of a data file
- Providing for the dynamic growth of a data file



ORACLE

Enlarging the Database

These activities can be performed with Enterprise Manager or with SQL statements. The size of the database can be described as the sum of all of its tablespaces.

Quiz

A database can have a mixture of Oracle-managed and unmanaged files.

1. True
2. False

Answer: 1

Quiz

Bigfile Tablespaces must have 1 file of at least 100 MB.

1. True
2. False

Answer: 2

Summary

In this lesson, you should have learned how to:

- Describe the storage of table row data in blocks
- Create and manage tablespaces
- Obtain tablespace information

ORACLE

Practice 7 Overview: Managing Database Storage Structures

This practice covers the following topics:

- Creating tablespaces
- Gathering information about tablespaces

ORACLE

8

Administering User Security

Objectives

After completing this lesson, you should be able to:

- Create and manage database user accounts:
  - Authenticate users
  - Assign default storage areas (tablespaces)
- Grant and revoke privileges
- Create and manage roles
- Create and manage profiles:
  - Implement standard password security features
  - Control resource usage by users

ORACLE

Objectives

The following terms relate to administering database users and assist you in understanding the objectives:

- A *database user account* is a way to organize the ownership of and access to database objects.
- A *password* is an authentication by the Oracle database.
- A *privilege* is a right to execute a particular type of SQL statement or to access another user's object.
- A *role* is a named group of related privileges that are granted to users or to other roles.
- *Profiles* impose a named set of resource limits on database usage and instance resources and manage account status and password management rules.
- A *quota* is a space allowance in a given tablespace. This is one of the ways by which you can control resource usage by users.

Database User Accounts

Each database user account has:

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

A schema:

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

ORACLE

Database User Accounts

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

- **A unique username:** Usernames cannot exceed 30 bytes, cannot contain special characters, and must start with a letter.
- **An authentication method:** The most common authentication method is a password, but Oracle Database 11*g* supports password, global, and external authentication methods (such as biometric, certificate, and token authentication).
- **A default tablespace:** This is a place where a user creates objects if the user does not specify some other tablespace. Note that having a default tablespace does not imply that the user has the *privilege* of creating objects in that tablespace, nor does the user have a *quota* of space in that tablespace in which to create objects. Both of these are granted separately.
- **A temporary tablespace:** This is a place where temporary objects, such as sorts and temporary tables, are created on behalf of the user by the instance. No quota is applied to temporary tablespaces.

Database User Accounts (continued)

- **A user profile:** This is a set of resource and password restrictions assigned to the user.
- **An initial consumer group:** This is used by the Resource Manager.
- **An account status:** Users can access only "open" accounts. The `account_status` may be in various combinations of "locked" and "expired."

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

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

Predefined Administrative Accounts

- SYS account:
  - Is granted the DBA role, as well as several other roles.
  - Has all privileges with ADMIN OPTION
  - Is required for startup, shutdown, and some maintenance commands
  - Owns the data dictionary and the Automatic Workload Repository (AWR)
- SYSTEM account is granted the DBA, MGMT_USER, and AQ_ADMINISTRATOR_ROLE roles.
- DBSNMP account is granted the OEM_MONITOR role.
- SYSMAN account is granted the MGMT_USER, RESOURCE and SELECT_CATALOG_ROLE roles.
- These accounts are not used for routine operations.

ORACLE

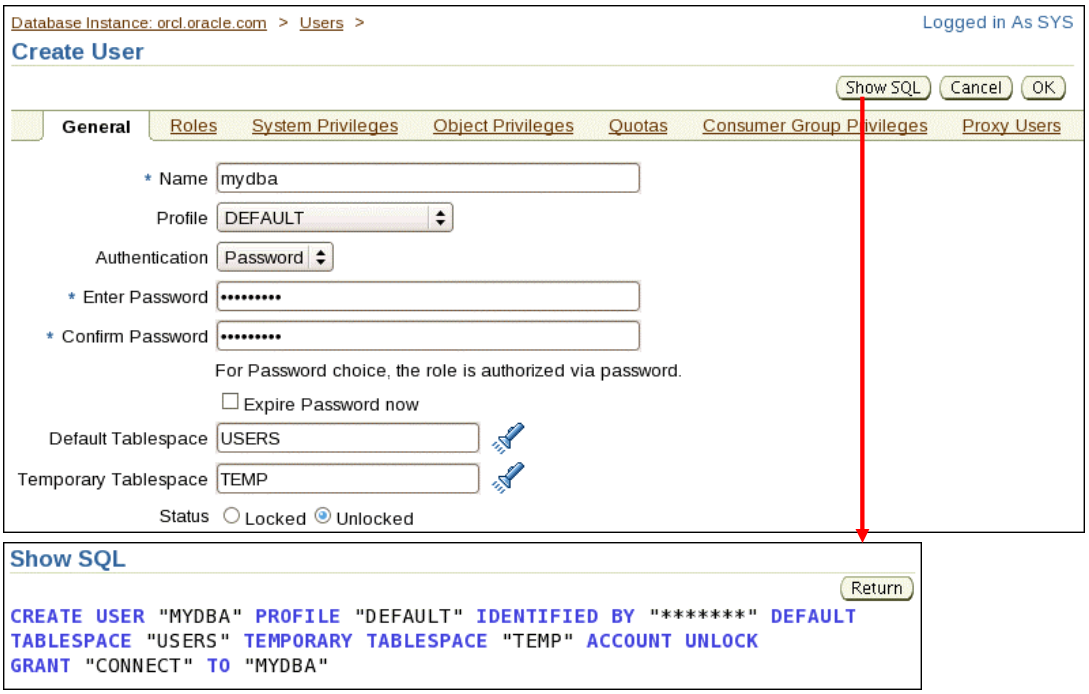
Predefined Administrative Accounts

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

The DBSNMP account is used by the management agent of Enterprise Manager to monitor and manage the database. The SYSMAN account is used to perform Oracle Enterprise Manager administration tasks. Neither DBSNMP nor SYSMAN have the SYSDBA privilege.

Best practice tip: Applying the principle of least privilege, these accounts are not used for routine operations. Users who need DBA privileges have separate accounts with the required privileges granted to them. For example, Jim has a low-privilege account called jim and a privileged account called jim_dba. This method allows the principle of least privilege to be applied, eliminates the need for account sharing, and allows individual actions to be audited.

Creating a User



Select Server > Users, and then click the Create button.

ORACLE

Creating a User

On the Users page of Enterprise Manager, you manage the database users who are allowed to access the current database. You use this page to create, delete, and modify the settings of a user.

To create a database user:

1. In Enterprise Manager Database Control, click the Server tab and then click Users in the Security section.
2. Click the Create button.

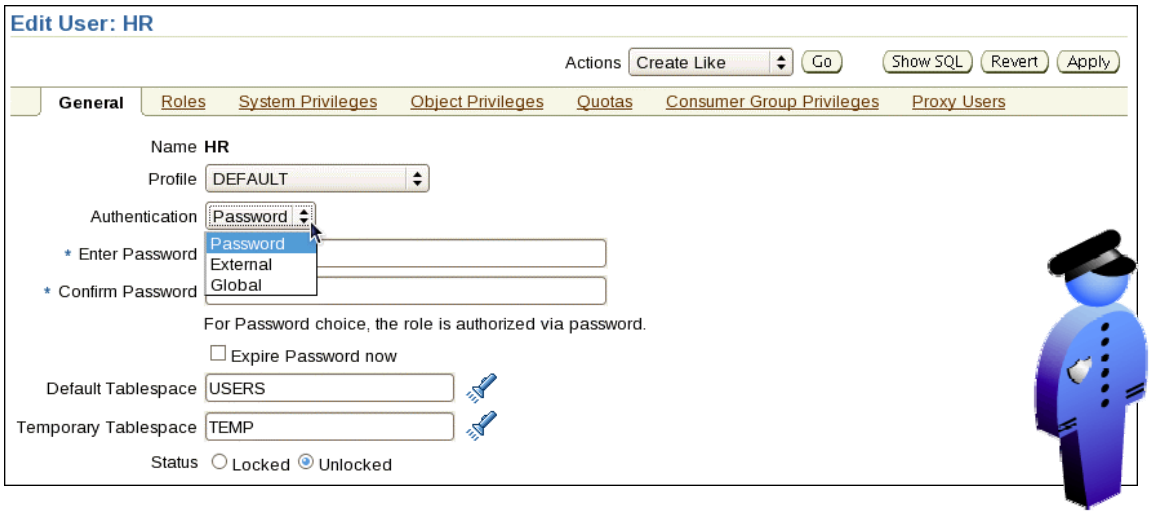
Provide the required information. Mandatory items (such as Name) are marked with an asterisk (*). The name specified must follow the same rules as those used for creating database objects. The pages that follow in this lesson give you more information about authentication. Profiles are covered later in this lesson.

Assign a default tablespace and a temporary tablespace to each user. If users do not specify a tablespace when creating an object, the object will be created in the default tablespace assigned to the object owner. This enables you to control where their objects are created. If you do not choose a default tablespace, the system-defined default permanent tablespace is used. The case is similar for the temporary tablespace: if you do not specify a tablespace, the system-defined temporary tablespace is used.

Note: Click Show SQL to see the supporting SQL syntax. For full SQL syntax for creating users, please see the *Oracle® Database SQL Language Reference* manual.

Authenticating Users

- Password
- External
- Global



Authenticating Users

Authentication means verifying the identity of someone or something (a user, device, or other entity) that wants to use data, resources, or applications. Validating that identity establishes a trust relationship for further interactions. Authentication also enables accountability by making it possible to link access and actions to specific identities. After authentication, authorization processes can allow or limit the levels of access and action that are permitted to that entity.

When you create a user, you must decide on the authentication technique to use, which can be modified later.

Password: This is also referred to as authentication by the Oracle database. 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 11*g* are case-sensitive by default. These passwords may also contain multibyte characters and are limited to 30 bytes. Each password created in a database that is upgraded to Oracle Database 11*g* remains case-insensitive until the password is changed.

Passwords are always automatically and transparently encrypted using the Advanced Encryption Standard (AES) algorithm during network (client/server and server/server) connections before sending them across the network.

Authenticating Users (continued)

External: This is authentication by a method outside the database (operating system, Kerberos, or Radius). The Advanced Security Option is required for Kerberos or Radius. Users can connect to the Oracle database without specifying a username or password. The Advanced Security Option (which is a strong authentication) allows users to be identified through the use of biometrics, x509 certificates, and token devices. With external authentication, your database relies on the underlying operating system, network authentication service, or external authentication service to restrict access to database accounts. A database password is not used for this type of login. If your operating system or network service permits, you can have it authenticate users. If you use operating system authentication, set the `OS_AUTHENT_PREFIX` initialization parameter and use this prefix in Oracle usernames. The `OS_AUTHENT_PREFIX` parameter defines a prefix that the Oracle database adds to the beginning of each user's operating system account name. The default value of this parameter is `OPS$` for backward compatibility with the previous versions of the Oracle software. The Oracle database compares the prefixed username with the Oracle usernames in the database when a user attempts to connect. For example, suppose that `OS_AUTHENT_PREFIX` is set as follows:

```
OS_AUTHENT_PREFIX=OPS$
```

If a user with an operating system account named `tsmith` needs to connect to an Oracle database and be authenticated by the operating system, the Oracle database checks whether there is a corresponding database user `OPS$tsmith` and, if so, allows the user to connect. All references to a user who is authenticated by the operating system must include the prefix, as seen in `OPS$tsmith`.

Note: The text of the `OS_AUTHENT_PREFIX` initialization parameter is case-sensitive on some operating systems. See the Oracle documentation that is specific to your operating system for more information about this initialization parameter.

Global: With the Oracle Advanced Security option, global authentication enables users to be identified through the use of Oracle Internet Directory.

For more information about advanced authentication methods, see the *Oracle Database Security* course.

Administrator Authentication

Operating system security:

- DBAs must have the OS privileges to create and delete files.
- Typical database users should not have the OS privileges to create or delete database files.

Administrator security:

- For SYSDBA, SYSOPER, and SYSASM connections:
  - DBA user by name is audited for password file and strong authentication methods
  - OS account name is audited for OS authentication
  - OS authentication takes precedence over password file authentication for privileged users
  - Password file uses case-sensitive passwords

ORACLE

Administrator Authentication

Operating system security: In UNIX and Linux, DBAs by default belong to the `oinstall` OS group, which has the required privileges to create and delete database files.

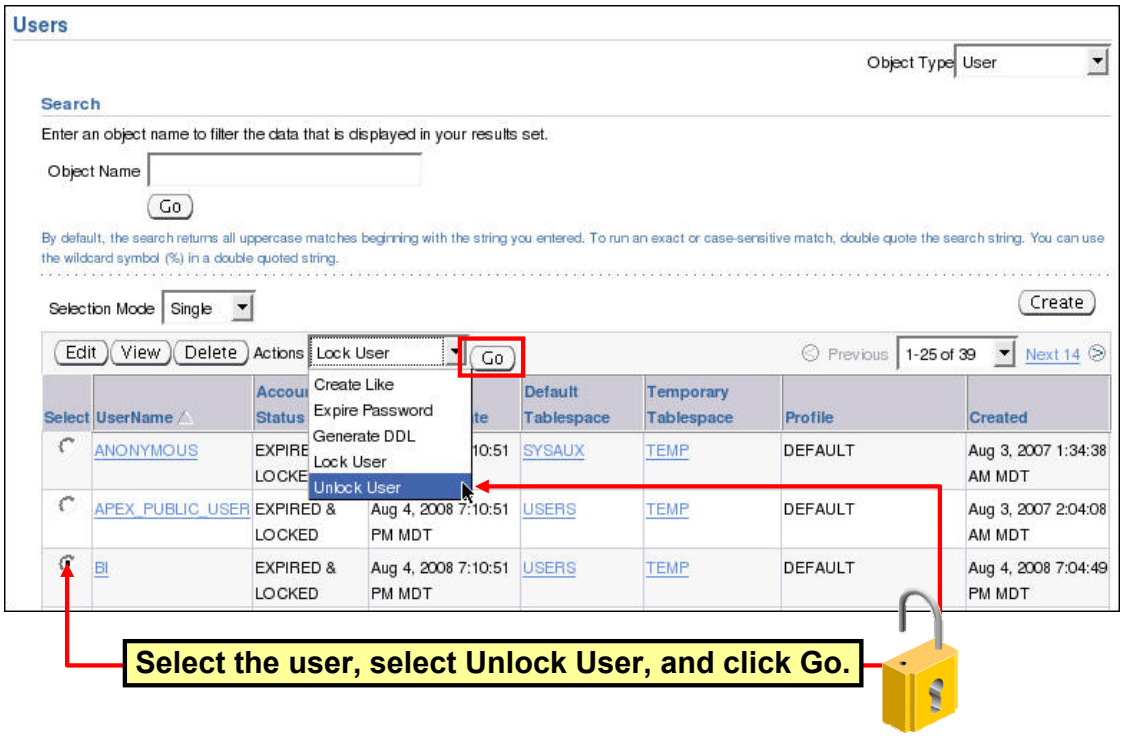
Administrator security: Connections for the privileged users `SYSDBA`, `SYSOPER`, and `SYSASM` are authorized only after verification with the password file or with the OS privileges and permissions. If OS authentication is used, the database does *not* use the supplied username and password. OS authentication is used if there is no password file, if the supplied username or password is not in that file, or if no username and password are supplied. The password file in Oracle Database 11*g* uses case-sensitive passwords by default.

However, if authentication succeeds by means of the password file, the connection is logged with the username. If authentication succeeds by means of the operating system, it is a `CONNECT / ` connection that does not record the specific user.

Note: If you are a member of the `OSDBA` or `OSOPER` group for the operating system and you connect as `SYSDBA` or `SYSOPER`, you will be connected with the associated administrative privileges regardless of the username and password that you specify. For `SYSASM` you must not specify any username or password (for example, `sqlplus / as SYSASM`).

In Oracle Database 11*g*, a privileged user may use strong authentication methods: Kerberos, SSL, or directory authentication if the Advanced Security Option is licensed.

Unlocking a User Account and Resetting the Password



Unlocking a User Account and Resetting the Password

During installation and database creation, you can unlock and reset many of the Oracle-supplied database user accounts. If you did not choose to unlock the user accounts at that time, you can unlock the user by selecting the user on the Users page, selecting Unlock User from the Actions list, and clicking Go. This does not change the password in any way. If the password was expired at the time that you unlocked the user, it will remain expired until you edit the user and change the password.

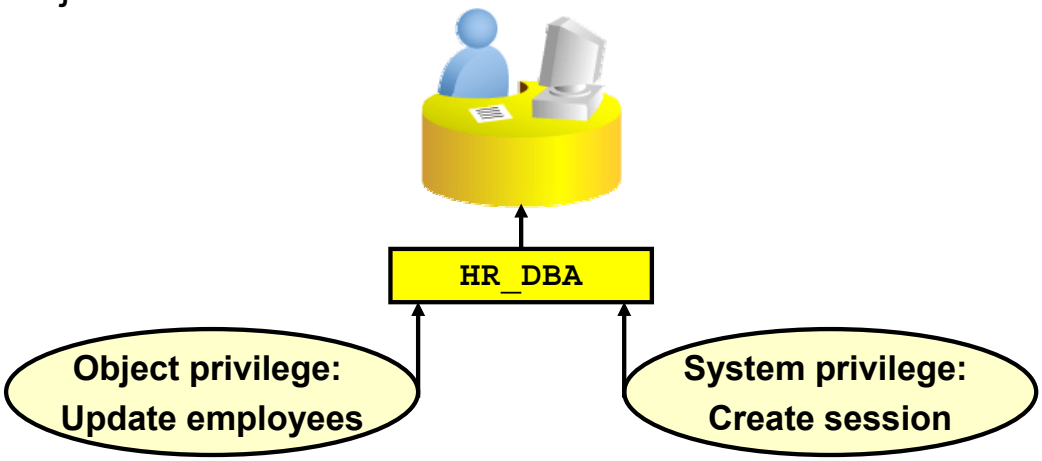
To unlock the user and reset the password, perform the following steps on the Edit Users page:

1. Enter the new password in the Enter Password and Confirm Password fields.
2. Select the Unlocked check box.
3. Click Apply to reset the password and unlock the user account.

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



ORACLE

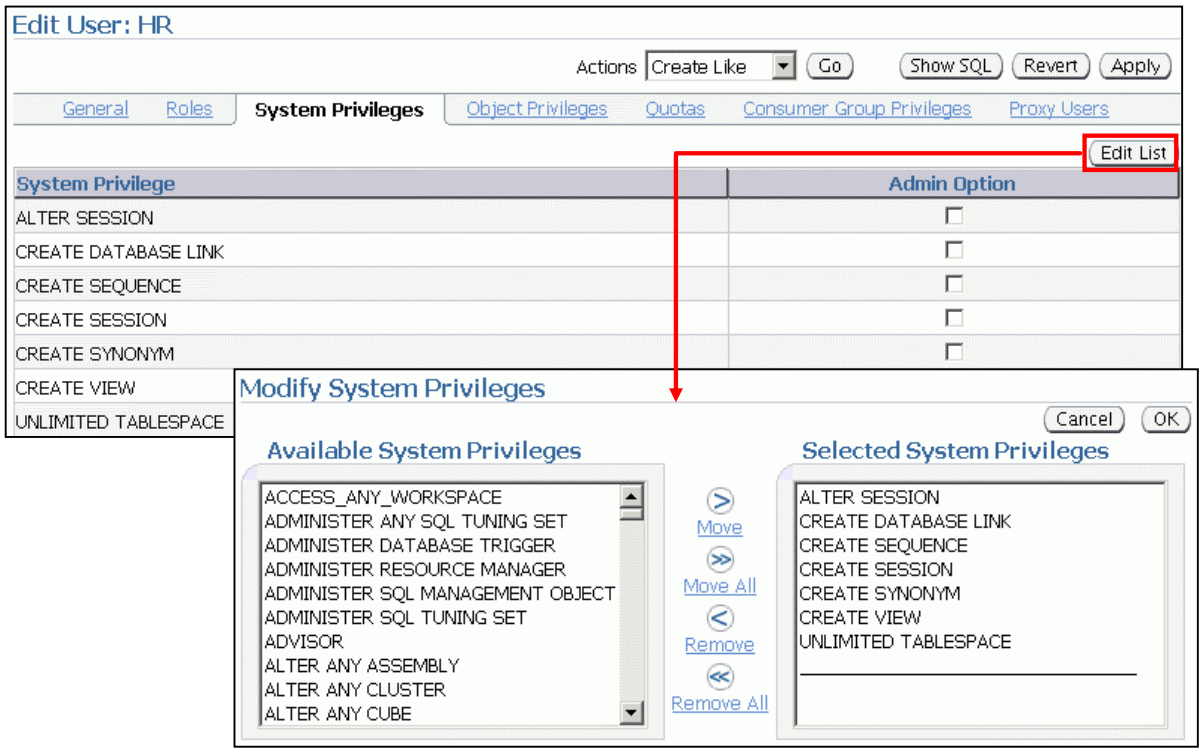
Privileges

A *privilege* is a right to execute a particular type of SQL statement or to access another user's object. The Oracle database enables you to control what the users can and cannot do in the database.

Privileges are divided into two categories:

- **System privileges:** Each system privilege allows a user to perform a particular database operation or class of database operations. For example, the privilege to create tablespaces is a system privilege. System privileges can be granted by the administrator or by someone who has been given explicit permission to administer the privilege. There are more than 170 distinct system privileges. Many system privileges contain the `ANY` clause.
- **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.

System Privileges



System Privileges

To grant system privileges, click the Systems Privileges tab on the Edit User page. Select the appropriate privileges from the list of available privileges, and move them to the Selected System Privileges list by clicking the Move arrow.

Granting a privilege with the `ANY` clause means that the privilege crosses schema lines. For example, if you have the `CREATE TABLE` privilege, you can create a table—but only in your own schema. The `SELECT ANY TABLE` privilege allows you to select from tables owned by other users. The `SYS` user and users with the `DBA` role are granted all of the `ANY` privileges; they can therefore do anything to any data object. The scope of the `ANY` system privileges can be controlled using the Oracle Database Vault Option.

Selecting the Admin Option check box enables the user to administer the privilege and grant the system privilege to other users.

The SQL syntax for granting system privileges is:

```
GRANT <system_privilege> TO <grantee clause> [WITH ADMIN
OPTION]
```

Carefully consider security requirements before granting system permissions. Some system privileges are usually granted only to administrators:

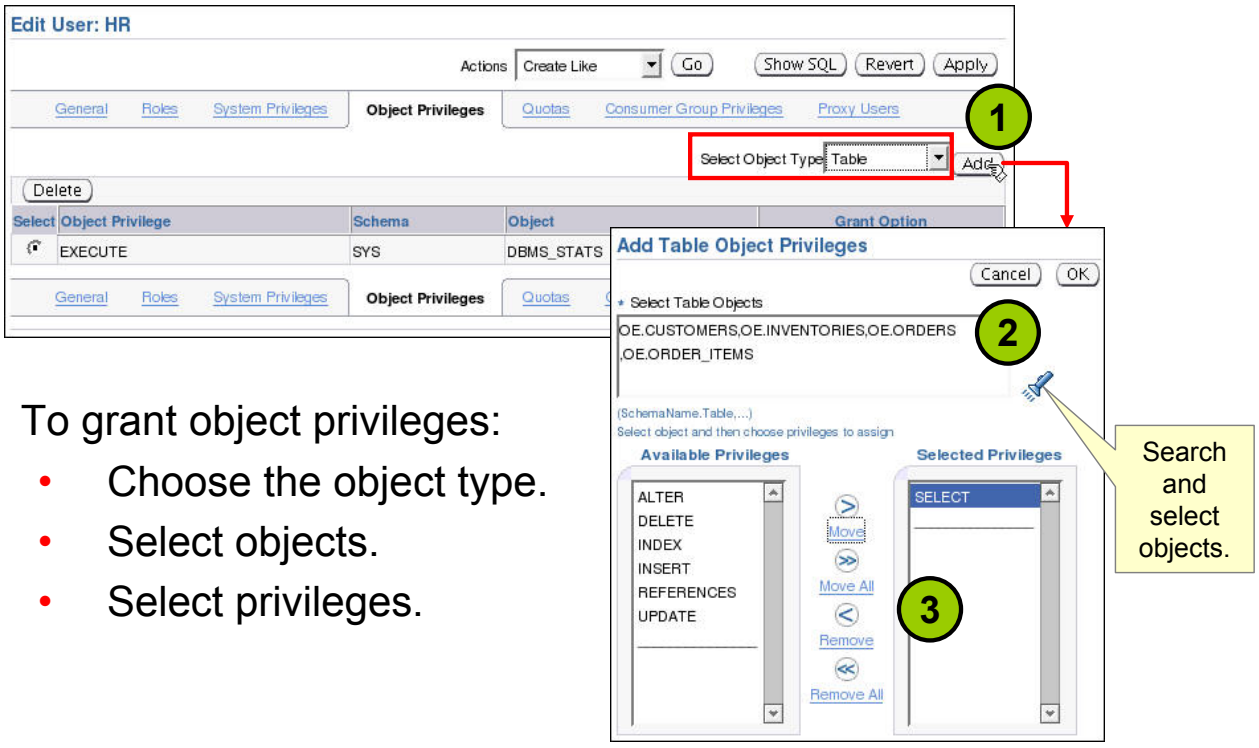
- **`RESTRICTED SESSION`:** This privilege allows you to log in even if the database has been opened in restricted mode.

System Privileges (continued)

- **SYSDBA and SYSOPER:** These privileges allow you to shut down, start up, and perform recovery and other administrative tasks in the database. SYSOPER allows a user to perform basic operational tasks, but without the ability to look at user data. It includes the following system privileges:
  - STARTUP and SHUTDOWN
  - CREATE SPFILE
  - ALTER DATABASE OPEN/MOUNT/BACKUP
  - ALTER DATABASE ARCHIVELOG
  - ALTER DATABASE RECOVER (Complete recovery only. Any form of incomplete recovery, such as UNTIL TIME | CHANGE | CANCEL | CONTROLFILE, requires connecting as SYSDBA.)
  - RESTRICTED SESSION

  The SYSDBA system privilege additionally authorizes incomplete recovery and the deletion of a database. Effectively, the SYSDBA system privilege allows a user to connect as the SYS user.
- **SYSASM:** This privilege allows you to start up, shut down and administer an ASM instance.
- **DROP ANY *object*:** The DROP ANY privilege allows you to delete objects that other schema users own.
- **CREATE, MANAGE, DROP, and ALTER TABLESPACE:** These privileges allow for tablespace administration, including creating, dropping, and changing tablespace attributes.
- **CREATE LIBRARY:** The Oracle database allows developers to create and call external code (for example, a C library) from PL/SQL. The library must be named by a LIBRARY object in the database. The CREATE LIBRARY privilege allows a user to create an arbitrary code library that is executable from PL/SQL.
- **CREATE ANY DIRECTORY:** As a security measure, the operating system directory where the code resides must be linked to a virtual Oracle directory object. With the CREATE ANY DIRECTORY privilege, you can potentially call insecure code objects.
  The CREATE ANY DIRECTORY privilege allows a user to create a directory object (with read and write access) to any directory that the Oracle software owner can access. This means that the user can access external procedures in those directories. The user can attempt to directly read and write any database file, such as data files, redo log, and audit logs. Ensure that your organization has a security strategy that prevents misuse of powerful privileges such as this one.
- **GRANT ANY OBJECT PRIVILEGE:** This privilege allows you to grant object permissions on objects that you do not own.
- **ALTER DATABASE and ALTER SYSTEM:** These very powerful privileges allow you to modify the database and the Oracle instance (for example, renaming a data file or flushing the buffer cache).

Object Privileges



To grant object privileges:

- Choose the object type.
- Select objects.
- Select privileges.

Object Privileges

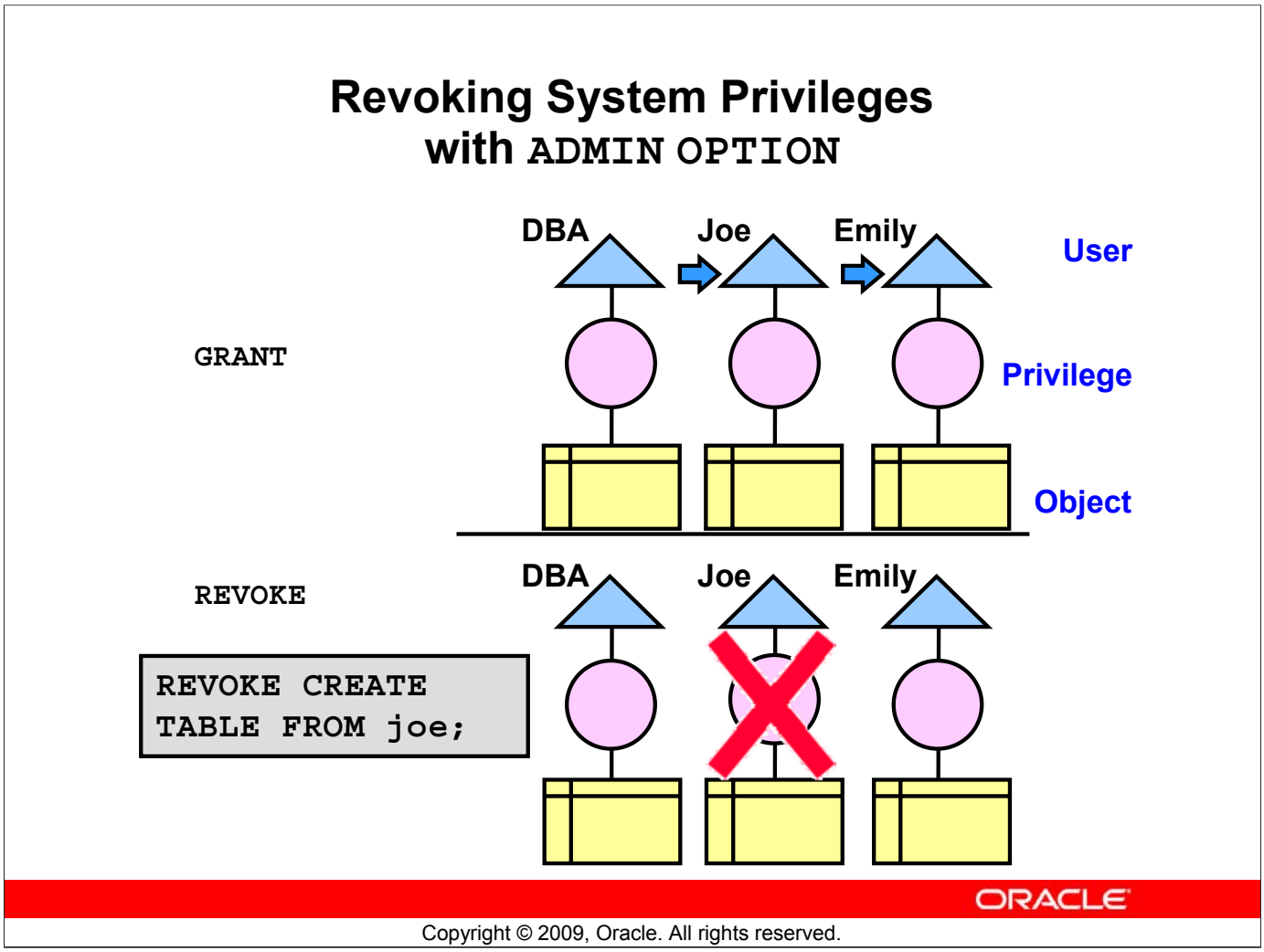
To grant object privileges, click the Object Privileges tab on the Edit User page. Select the type of object on which you want to grant privileges, and then click the Add button. Choose the objects by either entering *<username.object name>* or selecting them from the list.

Then select the appropriate privileges from the Available Privileges list and click the Move button. When you have finished selecting privileges, click OK.

On the Edit User page, select the Grant check box if this user is allowed to grant other users the same access.

The SQL syntax for granting object privileges is:

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



Revoking System Privileges with ADMIN OPTION

System privileges that have been granted directly with a GRANT command can be revoked by using the REVOKE SQL statement. Users with ADMIN OPTION for a system privilege can revoke the privilege from any other database user. The revoker does not have to be the same user who originally granted the privilege.

There are no cascading effects when a system privilege is revoked, regardless of whether it is given the ADMIN OPTION.

The SQL syntax for revoking system privileges is:

```
REVOKE <system_privilege> FROM <grantee clause>
```

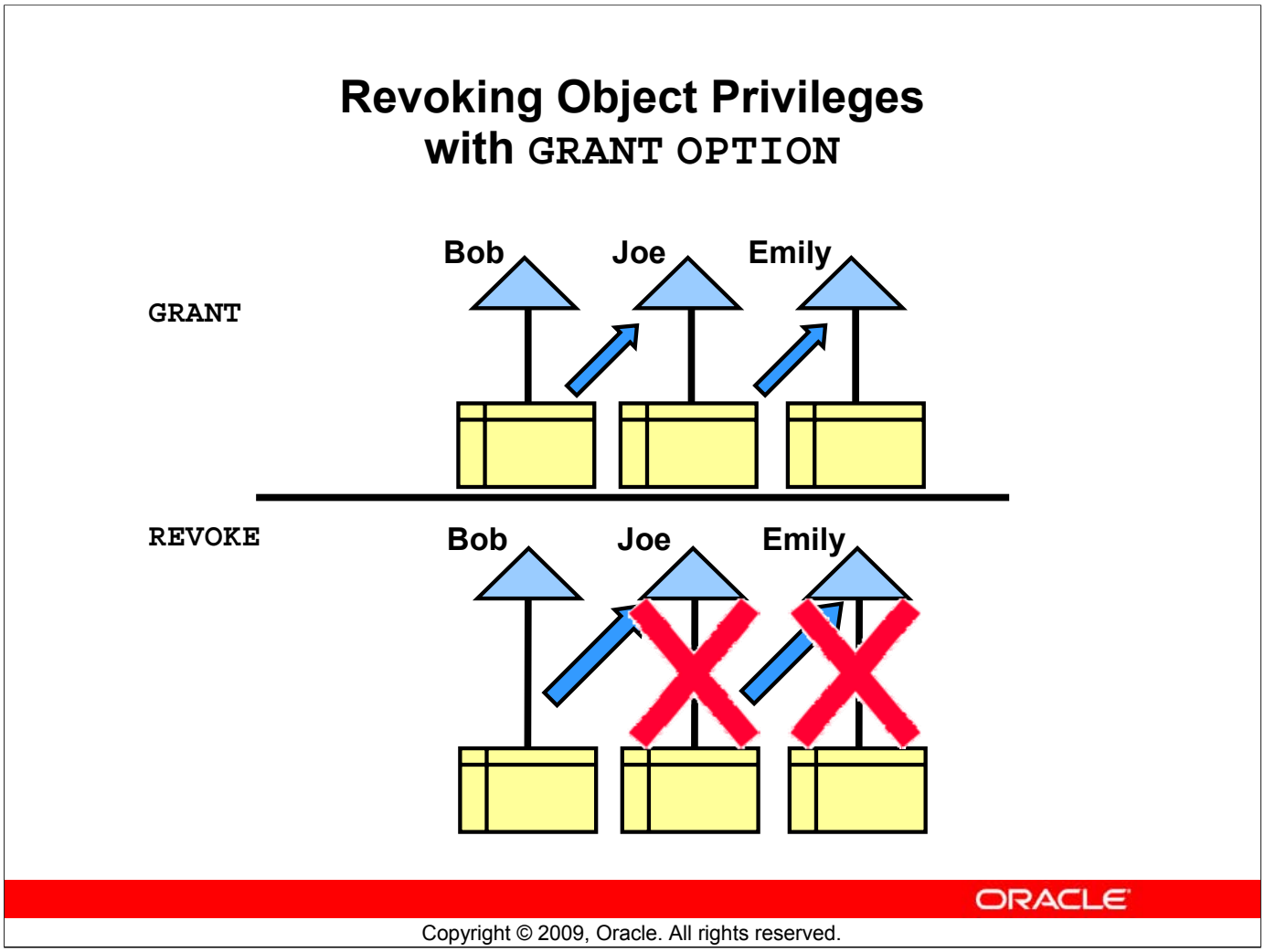
The slide illustrates the following situation.

Scenario

1. The DBA grants the CREATE TABLE system privilege to Joe with ADMIN OPTION.
2. Joe creates a table.
3. Joe grants the CREATE TABLE system privilege to Emily.
4. Emily creates a table.
5. The DBA revokes the CREATE TABLE system privilege from Joe.

Result

Joe's table still exists, but Joe cannot create new tables. Emily's table still exists, and she still has the CREATE TABLE system privilege.



Revoking Object Privileges with `GRANT OPTION`

Cascading effects can be observed when revoking a system privilege that is related to a data manipulation language (DML) operation. For example, if the `SELECT ANY TABLE` privilege is granted to a user, and if that user has created procedures that use the table, all procedures that are contained in the user's schema must be recompiled before they can be used again.

Revoking object privileges also cascades when given with `GRANT OPTION`. As a user, you can revoke only those privileges that you have granted. For example, Bob cannot revoke the object privilege that Joe granted to Emily. Only the grantee or a user with the privilege called `GRANT ANY OBJECT PRIVILEGE` can revoke object privileges.

Scenario

1. Joe is granted the `SELECT` object privilege on `EMPLOYEES` with `GRANT OPTION`.
2. Joe grants the `SELECT` privilege on `EMPLOYEES` to Emily.
3. The `SELECT` privilege is revoked from Joe. This revoke is cascaded to Emily as well.

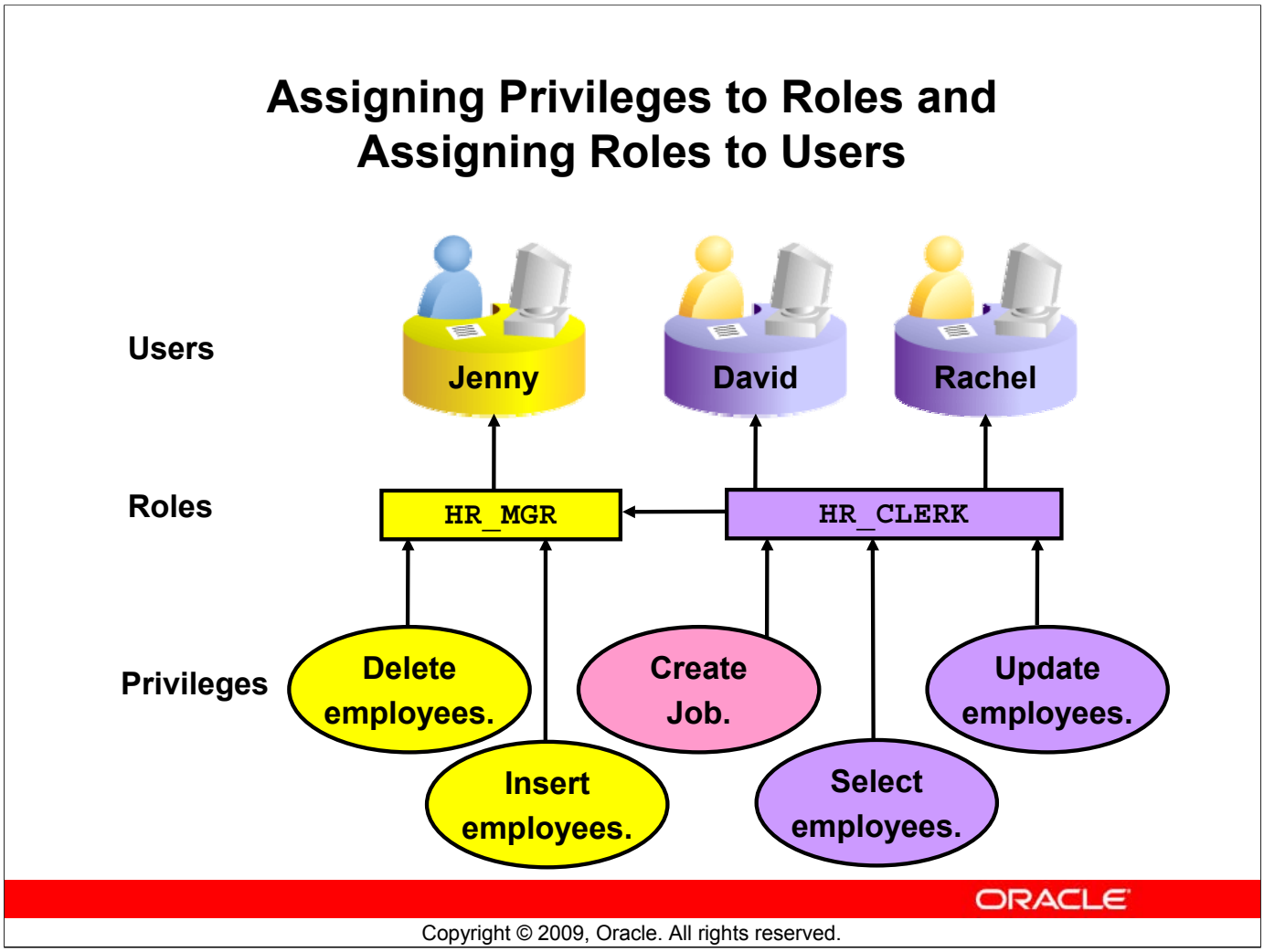
Benefits of Roles

- Easier privilege management
- Dynamic privilege management
- Selective availability of privileges

ORACLE

Benefits of Roles

- **Easier privilege management:** Use roles to simplify privilege management. Rather than granting the same set of privileges to several users, you can grant the privileges to a role and then grant that role to each user.
- **Dynamic privilege management:** If the privileges associated with a role are modified, all users who are granted the role acquire the modified privileges automatically and immediately.
- **Selective availability of privileges:** Roles can be enabled and disabled to turn privileges on and off temporarily. This allows the privileges of the user to be controlled in a given situation.



Assigning Privileges to Roles and Assigning Roles to Users

In most systems, it is time consuming and error prone to grant necessary privileges to each user individually. The Oracle software provides for easy and controlled privilege management through roles. Roles are named groups of related privileges that are granted to users or to other roles. Roles are designed to ease the administration of privileges in the database and, therefore, improve security.

Role characteristics

- Privileges are granted to and revoked from roles as though the role were a user.
- Roles are granted to and revoked from users or other roles as though they were system privileges.
- A role can consist of both system and object privileges.
- A role can be enabled or disabled for each user who is granted the role.
- A role can require a password to be enabled.
- Roles are not owned by anyone, and they are not in any schema.

In the slide example, the `SELECT` and `UPDATE` privileges on the `employees` table *and* the `CREATE JOB` system privilege are granted to the `HR_CLERK` role. `DELETE` and `INSERT` privileges on the `employees` table *and* the `HR_CLERK` role are granted to the `HR_MGR` role. The manager is granted the `HR_MGR` role and can now select, delete, insert, and update the `employees` table.

Predefined Roles

Role	Privileges Included
CONNECT	CREATE SESSION
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
DBA	Most system privileges; several other roles. Do not grant to nonadministrators.
SELECT_ CATALOG_ROLE	No system privileges; HS_ADMIN_ROLE and over 1,700 object privileges on the data dictionary

ORACLE

Predefined Roles

There are several roles that are defined automatically for Oracle databases when you run database creation scripts. CONNECT is granted automatically to any user that is created with Enterprise Manager. For security reasons, the CONNECT role has contained only the CREATE SESSION privilege since version 10.2.0 of the Oracle Database.

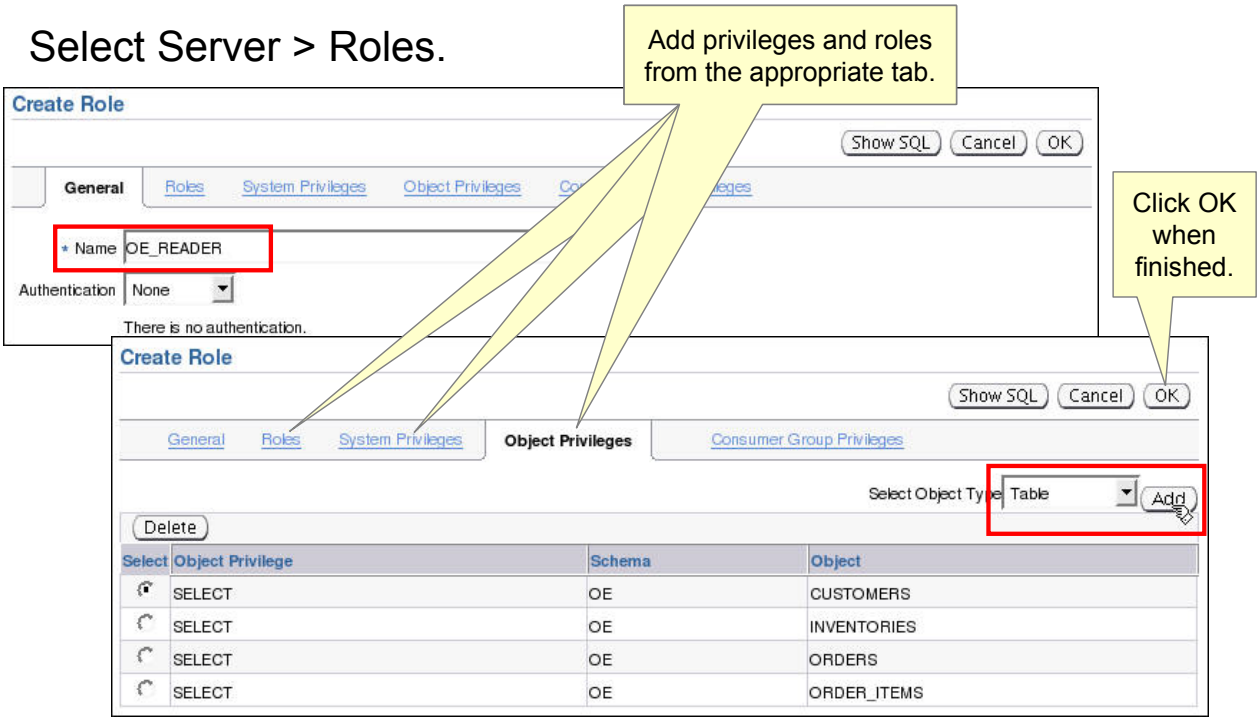
Note: Be aware that granting the RESOURCE role includes granting the UNLIMITED TABLESPACE privilege.

Functional Roles

Other roles that authorize you to administer special functions are created when that functionality is installed. For example, XDBADMIN contains the privileges required to administer the Extensible Markup Language (XML) database if that feature is installed. AQ_ADMINISTRATOR_ROLE provides privileges to administer advanced queuing. HS_ADMIN_ROLE includes the privileges needed to administer heterogeneous services.

You must not alter the privileges granted to these functional roles without the assistance of Oracle Support because you may inadvertently disable the needed functionality.

Creating a Role



ORACLE

Creating a Role

A *role* is a named group of related privileges that are granted to users or to other roles. A DBA manages privileges through roles.

To create a role, perform the following steps:

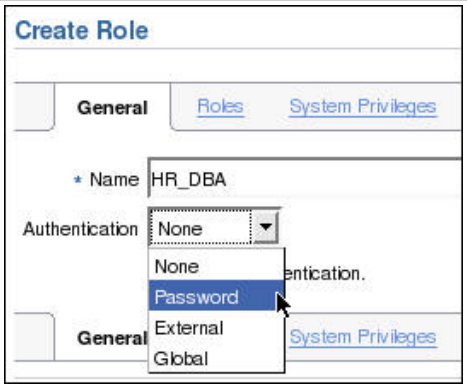
1. In Enterprise Manager Database Control, click the Server tab and then click Roles under the Security heading.
2. Click the Create button.
3. On the General tab, enter a name for the role.
4. Optionally, add the system privileges, object privileges, and other roles as required. The role can be edited at a later time to modify these settings if necessary.
5. Click OK when finished.

Secure Roles

- Roles can be nondefault and enabled when required.

```
SET ROLE vacationdba;
```

- Roles can be protected through authentication.



- Roles can also be secured programmatically.

```
CREATE ROLE secure_application_role
IDENTIFIED USING <security_procedure_name>;
```

ORACLE

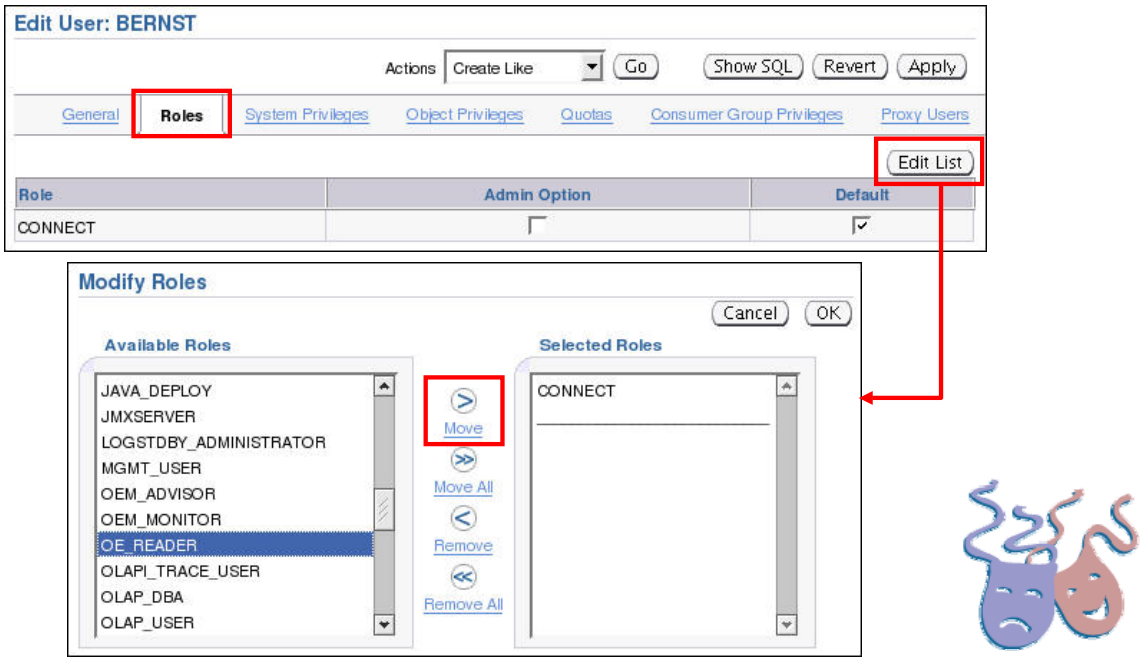
Secure Roles

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. Default roles are assigned to the user at connect time.

It is possible to:

- Make a role nondefault. When the role is granted to a user, deselect the DEFAULT check box. The user must now explicitly enable the role before the role's privileges can be exercised.
- Have a role require additional authentication. The default authentication for a role is None, but it is possible to have the role require additional authentication before it can be set.
- Create secure application roles that can be enabled only by executing a PL/SQL procedure successfully. The PL/SQL procedure can check things such as the user's network address, the program that the user is running, the time of day, and other elements needed to properly secure a group of permissions.
- Administer roles easily using the Oracle Database Vault option. Secure application roles are simplified, and traditional roles can be further restricted.

Assigning Roles to Users



Assigning Roles to Users

You can use roles to administer database privileges. You can add privileges to a role and grant the role to a user. The user can then enable the role and exercise the privileges granted by the role. A role contains all privileges that are granted to that role and all privileges of other roles that are granted to it.

By default, Enterprise Manager automatically grants the `CONNECT` role to new users. This allows users to connect to the database and create database objects in their own schemas.

To assign a role to a user:

1. In Enterprise Manager Database Control, click the Server tab and then click Users under the Security heading.
2. Select the user and click the Edit button.
3. Click the Roles tab, and then click the Edit List button.
4. Select the desired role under Available Roles and move it under Selected Roles.
5. When you have assigned all appropriate roles, click the OK button.

Quiz

All passwords created in Oracle Database 11*g* are not case-sensitive by default.

1. True
2. False

Answer: 2

Quiz

A database role:

1. Can be enabled or disabled
2. Can consist of system and object privileges
3. Is owned by its creator
4. Cannot be protected by a password

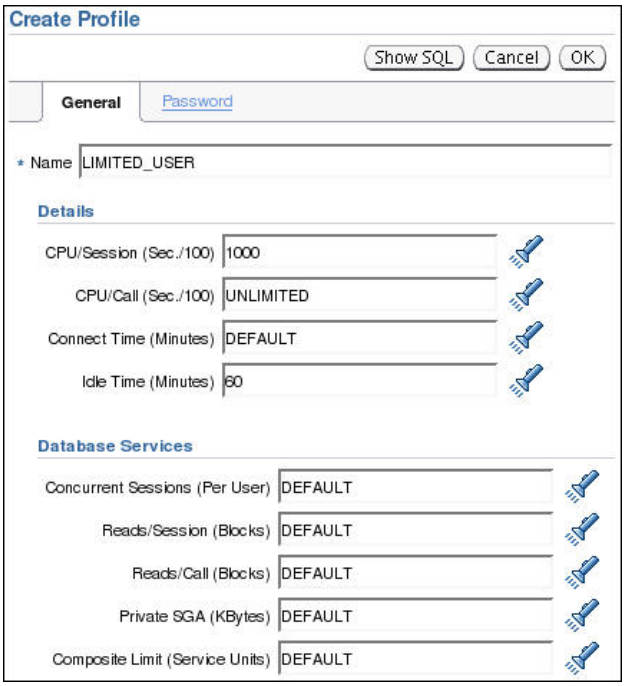
Answers: 1, 2

Profiles and Users

Users are assigned only one profile at a time.

Profiles:

- Control resource consumption
- Manage account status and password expiration



Note: `RESOURCE_LIMIT` must be set to `TRUE` before profiles can impose resource limitations.

ORACLE

Profiles and Users

Profiles impose a named set of resource limits on database usage and instance resources. Profiles also manage the account status and place limitations on users' passwords (length, expiration time, and so on). Every user is assigned a profile and may belong to only one profile at any given time. If users have already logged in when you change their profile, the change does not take effect until their next login.

The `DEFAULT` profile serves as the basis for all other profiles. As illustrated in the slide, limitations for a profile can be implicitly specified (as in CPU/Session), can be unlimited (as in CPU/Call), or can reference whatever setting is in the `DEFAULT` profile (as in Connect Time).

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. Profile password settings are always enforced

Profiles enable the administrator to control the following system resources:

- **CPU:** CPU resources may be limited on a per-session or per-call basis. A CPU/Session limitation of 1,000 means that if any individual session that uses this profile consumes more than 10 seconds of CPU time (CPU time limitations are in hundredths of a second), that session receives an error and is logged off:

```
ORA-02392: exceeded session limit on CPU usage, you are being
logged off
```

Profiles and Users (continued)

A per-call limitation does the same thing, but instead of limiting the user's overall session, it prevents any single command from consuming too much CPU. If CPU/Call is limited and the user exceeds the limitation, the command aborts. The user receives an error message such as the following:

```
ORA-02393: exceeded call limit on CPU usage
```

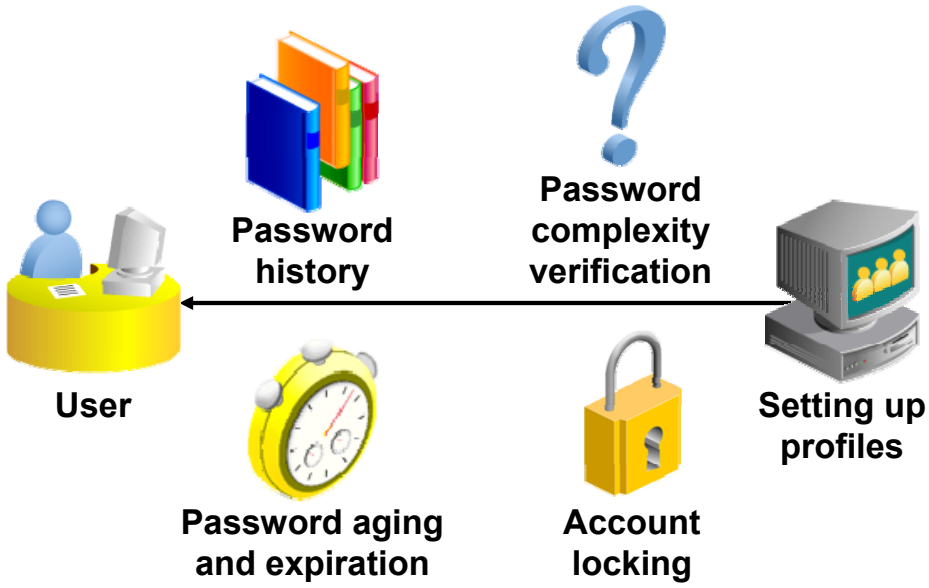
- **Network/Memory:** Each database session consumes system memory resources and (if the session is from a user who is not local to the server) network resources. You can specify the following:
  - **Connect Time:** Indicates for how many minutes a user can be connected before being automatically logged off
  - **Idle Time:** Indicates for how many minutes a user's session can remain idle before being automatically logged off. Idle time is calculated for the server process only. It does not take into account application activity. The `IDLE_TIME` limit is not affected by long-running queries and other operations.
  - **Concurrent Sessions:** Indicates how many concurrent sessions can be created by using a database user account
  - **Private SGA:** Limits the amount of space consumed in the System Global Area (SGA) for sorting, merging bitmaps, and so on. This restriction takes effect only if the session uses a shared server. (Shared servers are covered in the lesson titled "Configuring the Oracle Network Environment.")
- **Disk I/O:** This limits the amount of data a user can read at the per-session level or per-call level. Reads/Session and Reads/Call place a limitation on the total number of reads from both memory and the disk. This can be done to ensure that no I/O-intensive statements overuse memory and disks.

Profiles also allow a composite limit. Composite limits are based on a weighted combination of CPU/Session, Reads/Session, Connect Time, and Private SGA. Composite limits are discussed in more detail in the *Oracle Database Security Guide*.

To create a profile, click the Server tab and then click Profiles under the Security heading. On the Profiles page, click the Create button.

Note: Resource Manager is an alternative to many of the profile settings. For more details about Resource Manager, see the *Oracle Database Administrator's Guide*.

Implementing Password Security Features



Note: Do not use profiles that cause the `SYS`, `SYSMAN`, and `DBSNMP` passwords to expire and the accounts to be locked.

ORACLE

Implementing Password Security Features

Oracle password management is implemented with user profiles. Profiles can provide many standard security features.

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

- **`FAILED_LOGIN_ATTEMPTS`:** Specifies the number of failed login attempts before the lockout of the account
- **`PASSWORD_LOCK_TIME`:** Specifies the number of days for which the account is locked after the specified number of failed login attempts

Password aging and expiration: Enables user passwords to have a lifetime, after which the passwords expire and must be changed

- **`PASSWORD_LIFE_TIME`:** Determines the lifetime of the password in days, after which the password expires
- **`PASSWORD_GRACE_TIME`:** Specifies a grace period in days for changing the password after the first successful login after the password has expired

Note: Expiring passwords and locking the `SYS`, `SYSMAN`, and `DBSNMP` accounts prevent Enterprise Manager from functioning properly. The applications must catch the "password expired" warning message and handle the password change; otherwise, the grace period expires and the user is locked out without knowing the reason.

Implementing Password Security Features (continued)

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. These checks can be implemented by using one of the following:

- **PASSWORD_REUSE_TIME:** Specifies that a user cannot reuse a password for a given number of days
- **PASSWORD_REUSE_MAX:** Specifies the number of password changes that are required before the current password can be reused

Recall that the values of the profile parameters are either set or inherited from the DEFAULT profile.

If both password history parameters have a value of UNLIMITED, Oracle Database ignores both. The user can reuse any password at any time, which is not a good security practice.

If both parameters are set, password reuse is allowed—but only after meeting both conditions. The user must have changed the password the specified number of times, and the specified number of days must have passed since the old password was last used.

For example, the profile of user ALFRED has PASSWORD_REUSE_MAX set to 10 and PASSWORD_REUSE_TIME set to 30. User ALFRED cannot reuse a password until he has reset the password 10 times and until 30 days have passed since the password was last used.

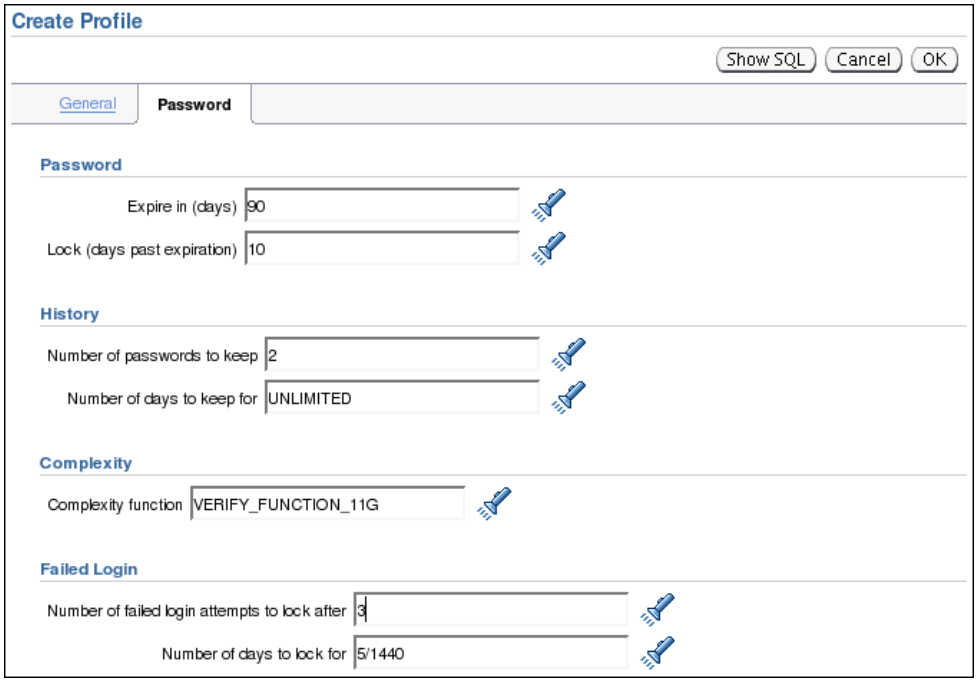
If one parameter is set to a number and the other parameter is specified as UNLIMITED, then the user can never reuse a password.

Password complexity verification: Makes a complexity check on the password to verify that it meets certain rules. The check must ensure that the password is complex enough to provide protection against intruders who may try to break into the system by guessing the password.

The PASSWORD_VERIFY_FUNCTION parameter names a PL/SQL function that performs a password complexity check before a password is assigned. Password verification functions must be owned by the SYS user and must return a Boolean value (TRUE or FALSE). A model password verification function is provided in the utlpwdmg.sql script found in the following directories:

- Unix and Linux platforms: $ORACLE_HOME/rdbms/admin
- Windows platforms: %ORACLE_HOME%\rdbms\admin

Creating a Password Profile



ORACLE

Creating a Password Profile

To create a password profile, click the Server tab and then click Profiles under the Security heading. On the Profiles page, click the Create button. Click the Password tab to set the password limits.

You can choose common values for each of the settings from a list of values (click the flashlight icon to browse), or you can enter a custom value.

All time periods are expressed in days but can also be expressed as fractions. There are 1,440 minutes in a day; 5/1,440 is therefore five minutes.

Enterprise Manager can also be used to edit existing password profiles.

If the `utlpwdmg.sql` script has been run, the `VERIFY_FUNCTION` and `VERIFY_FUNCTION_11G` functions are available. If you have created your own complexity function, the name of that function may be entered. The function name does not appear in the Select list. If the function produces run-time errors, the user is unable to change the password.

Dropping a Password Profile

In Enterprise Manager, you cannot drop a profile that is used by users. However, if you drop a profile with the `CASCADE` option (for example, in SQL*Plus), all users who have that profile are automatically assigned the `DEFAULT` profile.

Supplied Password Verification Function: VERIFY_FUNCTION_11G

The VERIFY_FUNCTION_11G function insures that the password is:

- At least eight characters
- Different from the username, username with a number, or username reversed
- Different from the database name or the database name with a number
- A string with at least one alphabetic and one numeric character
- Different from the previous password by at least three letters

Tip: Use this function as a template to create your own customized password verification.

ORACLE

Supplied Password Verification Function: VERIFY_FUNCTION_11G

The Oracle server provides two password complexity verification functions named VERIFY_FUNCTION and VERIFY_FUNCTION_11g. These functions are created with the <oracle_home>/rdbms/admin/utlpwdmg.sql script. The VERIFY_FUNCTION is provided for those who prefer the password function provided with previous versions. The password complexity verification function must be created in the SYS schema. It can be used as a template for your customized password verification.

In addition to creating VERIFY_FUNCTION, the utlpwdmg script also changes the DEFAULT profile with the following ALTER PROFILE command:

```
ALTER PROFILE default LIMIT
PASSWORD_LIFE_TIME 180
PASSWORD_GRACE_TIME 7
PASSWORD_REUSE_TIME UNLIMITED
PASSWORD_REUSE_MAX UNLIMITED
FAILED_LOGIN_ATTEMPTS 10
PASSWORD_LOCK_TIME 1
PASSWORD_VERIFY_FUNCTION verify_function_11g;
```

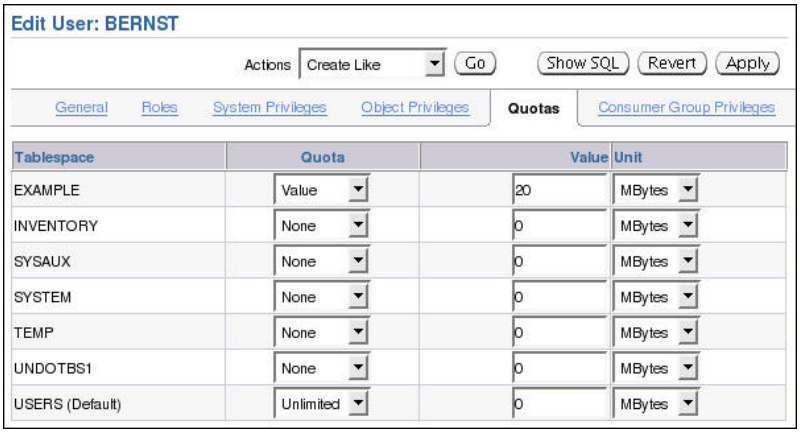
Remember that when users are created, they are assigned the DEFAULT profile unless another profile is specified.

Assigning Quotas to Users

Users who do not have the `UNLIMITED TABLESPACE` system privilege must be given a quota before they can create objects in a tablespace.

Quotas can be:

- A specific value in megabytes or kilobytes
- Unlimited



ORACLE

Assigning Quotas to Users

A *quota* is a space allowance in a given tablespace. By default, a user has no quota on any of the tablespaces. You have three options for providing a quota for a user on a tablespace.

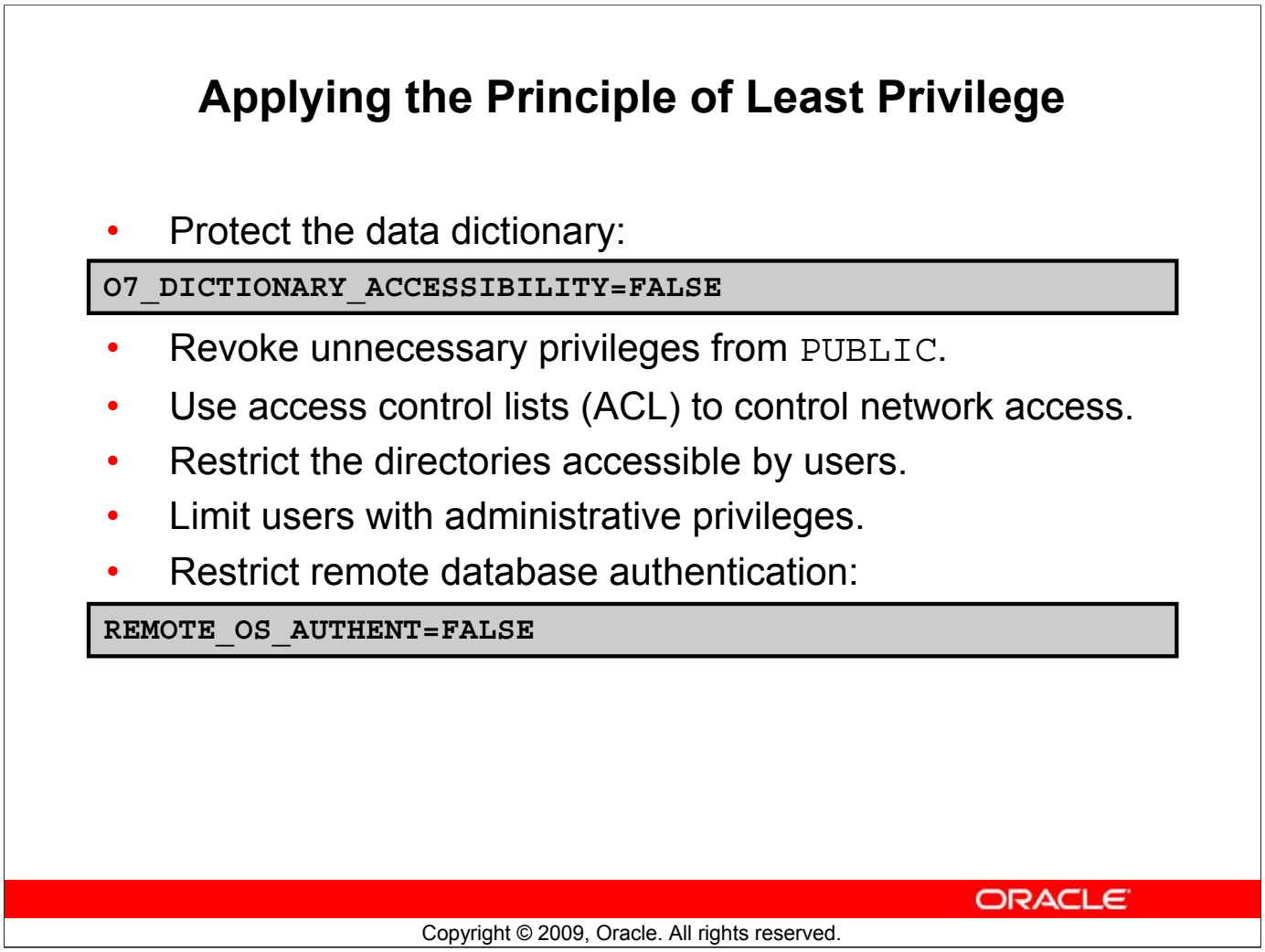
- **Unlimited:** Allows the user to use as much space as is available in the tablespace
- **Value:** Number of kilobytes or megabytes that the user can use. This does not guarantee that the space is set aside for the user. This value can be larger or smaller than the current space that is available in the tablespace.
- **`UNLIMITED TABLESPACE` system privilege:** Overrides all individual tablespace quotas and gives the user unlimited quota on all tablespaces, including `SYSTEM` and `SYSAUX`. This privilege must be granted with caution.
  Note: Be aware that granting the `RESOURCE` role includes granting this privilege.

You must not provide a quota to users on the `SYSTEM` or `SYSAUX` tablespaces. Typically, only the `SYS` and `SYSTEM` users are able to create objects in the `SYSTEM` or `SYSAUX` tablespaces.

You do not need a quota on an assigned temporary tablespace or any undo tablespaces. You do not need to have quota to be able to insert, update, and delete data in an Oracle database. The only users that need quota, are the accounts that own the database objects. It is typical when installing application code, that the installer creates database accounts to own the objects. Only these accounts need quota. Other database users can be granted permission to use these objects without quota.

Assigning Quotas to Users (continued)

- The Oracle instance checks the quota when a user creates or extends a segment.
- For activities that are assigned to a user schema, only those activities that use space in a tablespace count against the quota. Activities that do not use space in the assigned tablespace do not affect the quota (such as creating views or using temporary tablespaces).
- The quota is replenished when objects owned by the user are dropped with the `PURGE` clause or when the objects owned by the user in the recycle bin are purged.



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.

Protect the data dictionary: The `O7_DICTIONARY_ACCESSIBILITY` parameter is set by default to `FALSE`. You must not allow this to be changed without a very good reason because it prevents users with the `ANY TABLE` system privileges from accessing the data dictionary base tables. It also ensures that the `SYS` user can log in only as `SYSDBA`.

Revoke unnecessary privileges from `PUBLIC`: Several packages are extremely useful to applications that need them, but require proper configuration to be used securely. `PUBLIC` is granted execute privilege on the following packages: `UTL_SMTP`, `UTL_TCP`, `UTL_HTTP`, and `UTL_FILE`. In Oracle Database 11*g*, network access is controlled by an access control list (ACL) that may be configured to allow certain users access to specific network services. Network access is denied by default. An ACL must be created to allow network access. File through `UTL_FILE` access is controlled at two levels: at the OS level with permissions on files and directories, and in the database by `DIRECTORY` objects that allow access to specific file system directories. The `DIRECTORY` object may be granted to a user for read or for read and write. Execute privileges on other PL/SQL packages should be carefully controlled.

Applying the Principle of Least Privilege (continued)

The more powerful packages that may potentially be misused include:

- **UTL_SMTP:** Permits arbitrary email messages to be sent by using the database as a Simple Mail Transfer Protocol (SMTP) mail server. Use the ACL to control which machines may be accessed by which users.
- **UTL_TCP:** Permits outgoing network connections to be established by the database server to any receiving or waiting network service. Thus, arbitrary data can be sent between the database server and any waiting network service. Use the ACL to control access.
- **UTL_HTTP:** Allows the database server to request and retrieve data via HTTP. Granting this package to a user may permit data to be sent via HTML forms to a malicious Web site. Limit access by using the ACL.
- **UTL_FILE:** If configured improperly, allows text-level access to any file on the host operating system. When properly configured, this package limits user access to specific directory locations.

Restrict access to OS directories: The `DIRECTORY` object inside the database enables DBAs to map directories to OS paths and to grant privileges on those directories to individual users.

Limit users with administrative privileges: Do not provide database users more privileges than necessary. Nonadministrators must not be granted the DBA role. To implement least privilege, restrict the following types of privileges:

- Grants of system and object privileges
- `SYS`-privileged connections to the database, such as `SYSDBA` and `SYSOPER`
- Other DBA-type privileges, such as `DROP ANY TABLE`

Restrict remote database authentication: The `REMOTE_OS_AUTHENT` parameter is set to `FALSE` by default. It must not be changed unless all clients can be trusted to authenticate users appropriately. With the advent of Secure External Password Store (available in Oracle Database 10*g* Release 2), there are few compelling reasons ever to allow remote OS authentication.

In the remote authentication process:

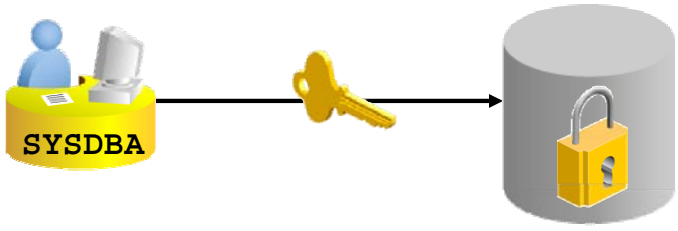
- The database user is authenticated externally
- The remote system authenticates the user
- The user logs in to the database without further authentication

Note: Always test your applications thoroughly if you have revoked privileges.

Protect Privileged Accounts

Privileged accounts can be protected by:

- Using password file with case-sensitive passwords
- Enabling strong authentication for administrator roles



ORACLE

Setting Database Administrator Authentication

Users with `SYSDBA`, `SYSOPER`, or `SYSASM` privileges must always be authenticated. When connecting locally, the user is authenticated by the local OS by being a member of a privileged OS group. If connecting remotely, a password file is used to authenticate privileged users. If the password file is configured, it will be checked first. In Oracle Database 11*g*, these passwords are case-sensitive. Oracle Database 11*g* provides other methods that make remote administrator authentication more secure and centralize the administration of these privileged users.

When a database is created using the Database Configuration Assistant, the password file is case-sensitive. If you upgrade from earlier database versions, be sure to make the password file case-sensitive for remote connections:

```
orapwd file=orapworcl entries=5 ignorecase=N
```

If your concern is that the password file might be vulnerable or that the maintenance of many password files is a burden, strong authentication can be implemented. The Advanced Security option is required if you want to use strong authentication methods. For more information about strong authentication, see the *Oracle Database Advanced Security Administrator's Guide*.

Quiz

Applying the principle of least privilege is not enough to harden the Oracle database.

1. True
2. False

Answer: 1

Quiz

With RESOURCE_LIMIT set at its default value of FALSE, profile password limitations are ignored.

1. True
2. False

Answer: 2

Summary

In this lesson, you should have learned how to:

- Create and manage database user accounts:
  - Authenticate users
  - Assign default storage areas (tablespaces)
- Grant and revoke privileges
- Create and manage roles
- Create and manage profiles:
  - Implement standard password security features
  - Control resource usage by users

ORACLE

Practice 8 Overview: Administering Users

This practice covers the following topics:

- Creating a profile to limit resource consumption
- Creating two roles:
  - `HRCLERK`
  - `HRMANAGER`
- Creating four new users:
  - One manager and two clerks
  - One schema user for the next practice session

ORACLE

9

Managing Data Concurrency

Objectives

After completing this lesson, you should be able to:

- Describe the locking mechanism and how Oracle manages data concurrency
- Monitor and resolve locking conflicts

ORACLE

Locks

- Prevent multiple sessions from changing the same data at the same time
- Are automatically obtained at the lowest possible level for a given statement
- Do not escalate

Transaction 1

```
SQL> UPDATE employees
  2  SET salary=salary+100
  3  WHERE employee_id=100;
```

Transaction 2

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3 WHERE employee_id=100;
```

ORACLE

Locks

Before the database allows a session to modify data, the session must first lock the data that is being modified. A lock gives the session exclusive control over the data so that no other transaction can modify the locked data until the lock is released.

Transactions can lock individual rows of data, multiple rows, or even entire tables. Oracle Database supports both manual and automatic locking. Automatically acquired locks always choose the lowest possible level of locking to minimize potential conflicts with other transactions.

Note: There are many types of locks used by the Oracle instance to maintain internal consistency. In this course, we will be concentrating only locking used to protect rows and tables.

Locking Mechanism

- High level of data concurrency:
  - Row-level locks for inserts, updates, and deletes
  - No locks required for queries
- Automatic queue management
- Locks held until the transaction ends (with the `COMMIT` or `ROLLBACK` operation)

Example

Assume that the rows for employee_id 100 and 101 reside in the same block:

Transaction 1

```
SQL> UPDATE employees
  2  SET salary=salary+100
  3  WHERE employee_id=100;
```

Transaction 2

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id=101;
```

ORACLE

Locking Mechanism

The locking mechanism is designed to provide the maximum possible degree of data concurrency within the database. Transactions that modify data acquire row-level locks rather than block-level or table-level locks. Modifications to objects (such as table moves) obtain object-level locks rather than whole database or schema locks.

Data queries do not require a lock, and a query succeeds even if someone has locked the data (always showing the original, prelock value reconstructed from undo information).

When multiple transactions need to lock the same resource, the first transaction to request the lock obtains it. Other transactions wait until the first transaction completes. The queue mechanism is automatic and requires no administrator interaction.

All locks are released as transactions are completed (that is, when a `COMMIT` or `ROLLBACK` is issued). In the case of a failed transaction, the same background process that automatically rolls back any changes from the failed transaction releases all locks held by that transaction.

Data Concurrency

Time:	Transaction 1	UPDATE hr.employees SET salary=salary+100 WHERE employee_id=100;
	Transaction 2	UPDATE hr.employees SET salary=salary+100 WHERE employee_id=101;
09:00:00	Transaction 3	UPDATE hr.employees SET salary=salary+100 WHERE employee_id=102;
	...	...
	Transaction *x*	UPDATE hr.employees SET salary=salary+100 WHERE employee_id=*xxx*;

ORACLE

Data Concurrency

The lock mechanism defaults to a fine-grained, row-level locking mode. Different transactions can be updating different rows within the same table without interfering with one another.

Although the default model is to lock at the row level, Oracle Database supports manual locking at higher levels if needed:

```
SQL> LOCK TABLE employees IN EXCLUSIVE MODE;
Table(s) Locked.
```

With the preceding statement, any other transaction that tries to update a row in the locked table must wait until the transaction that issued the lock request completes. `EXCLUSIVE` is the strictest lock mode. The following are the other lock modes:

- **`ROW SHARE`:** Permits concurrent access to the locked table but prohibits sessions from locking the entire table for exclusive access
- **`ROW EXCLUSIVE`:** Is the same as `ROW SHARE`, but also prohibits locking in `SHARE` mode. The `ROW EXCLUSIVE` locks are automatically obtained when updating, inserting, or deleting data. `ROW EXCLUSIVE` locks allow multiple readers and one writer.
- **`SHARE`:** Permits concurrent queries but prohibits updates to the locked table. A `SHARE` lock is required (and automatically requested) to create an index on a table. However, online index creation requires a `ROW SHARE` lock that is used when building the index.

Data Concurrency (continued)

Share locks allow multiple readers and no writers. Share locks are also used transparently when deleting or updating rows in a parent table that has a child table with foreign key constraints on the parent.

- **SHARE ROW EXCLUSIVE:** Is used to query a whole table and to allow others to query rows in the table, but prohibits others from locking the table in SHARE mode or updating rows
- **EXCLUSIVE:** Permits queries on the locked table but prohibits any other activity on it. An EXCLUSIVE lock is required to drop a table.

Like any request for a lock, manual lock statements wait until all sessions that either already have locks or have previously requested locks release their locks. The LOCK command accepts a special argument that controls the waiting behavior NOWAIT.

NOWAIT returns control to you immediately if the specified table is already locked by another session:

```
SQL> LOCK TABLE hr.employees IN SHARE MODE NOWAIT;
LOCK TABLE hr.employees IN SHARE MODE NOWAIT
               *
ERROR at line 1:
ORA-00054: resource busy and acquire with NOWAIT specified
```

It is usually not necessary to manually lock objects. The automatic locking mechanism provides the data concurrency needed for most applications. Oracle recommends that you avoid using manual locks, especially when developing applications. Severe performance issues often occur from unnecessarily high locking levels.

DML Locks

Transaction 1

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id= 107;
1 row updated.
```

Transaction 2

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id= 106;
1 row updated.
```

Each DML transaction must acquire *two* locks:

- `EXCLUSIVE` row lock on the row or rows being updated
- Table lock `(TM)` in `ROW EXCLUSIVE (RX)` mode on the table containing the rows

ORACLE

DML Locks

Each DML transaction obtains two locks:

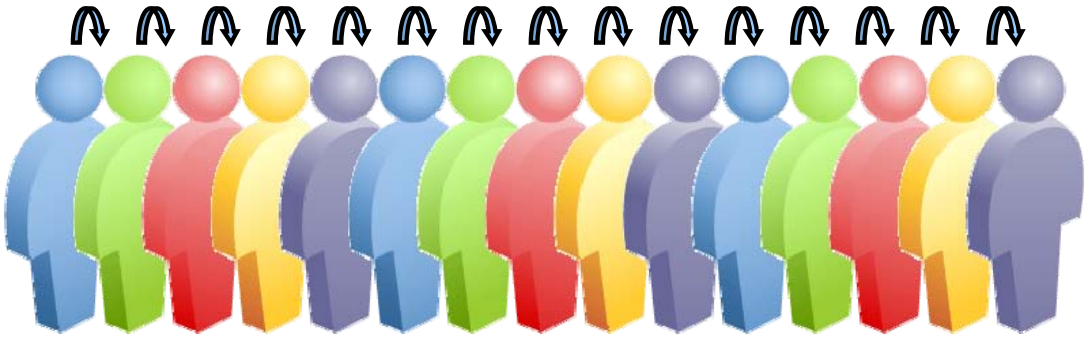
- An `EXCLUSIVE` row lock on the row or rows being updated
- A table lock `(TM)` in `ROW EXCLUSIVE (RX)` mode on the table being updated. This prevents another session from locking the whole table (possibly to drop or truncate it) while the change is being made. This mode is also called a subexclusive table lock `(SX)`.

A `ROW EXCLUSIVE` lock on the table prevents DDL commands from changing the dictionary metadata in the middle of an uncommited transaction. This preserves dictionary integrity and read consistency across the life of a transaction.

Enqueue Mechanism

The enqueue mechanism keeps track of:

- Sessions waiting for locks
- Requested lock mode
- Order in which sessions requested the lock



ORACLE

Enqueue Mechanism

Requests for locks are automatically queued. As soon as the transaction holding a lock is completed, the next session in line receives the lock.

The enqueue mechanism tracks the order in which locks are requested and the requested lock mode.

Sessions that already hold a lock can request that the lock be *converted* without having to go to the end of the queue. For example, suppose a session holds a `SHARE` lock on a table. The session can request that the `SHARE` lock be converted to an `EXCLUSIVE` lock. If no other transaction already has an `EXCLUSIVE` or `SHARE` lock on the table, the session holding the `SHARE` lock is granted an `EXCLUSIVE` lock without having to wait in the queue again.

Note: There are two categories of waiters for enqueues: those waiting without shared ownership and those with shared ownership that do not choose to escalate the lock level. Waiters in the second category are known as *converters*, which are always given priority over normal waiters even if they have been waiting less time.

Lock Conflicts

Transaction 1	Time	Transaction 2
`UPDATE employees SET salary=salary+100 WHERE employee_id=100;` `1 row updated.`	9:00:00	`UPDATE employees SET salary=salary+100 WHERE employee_id=101;` `1 row updated.`

`UPDATE employees SET COMMISION_PCT=2 WHERE employee_id=101;`
Session waits enqueued due to lock conflict.
	9:00:05 	`SELECT sum(salary) FROM employees;`
		`SUM(SALARY)`
		`-----------`
		`692634`

Session still waiting!	16:30:00	Many selects, inserts, updates, and deletes during the last 7.5 hours, but no commits or rollbacks!
`1 row updated.` Session continues.	16:30:01	`commit;`

ORACLE

Lock Conflicts

Lock conflicts occur often but are usually resolved through time and the enqueue mechanism. In certain rare cases, a lock conflict may require administrator intervention. In the case in the slide, transaction 2 obtains a lock on a single row at 9:00:00 and neglects to commit, leaving the lock in place. Transaction 1 attempts to update the entire table, requiring a lock on all rows, at 9:00:05. Transaction 1 is blocked by transaction 2 until transaction 2 commits at 16:30:01.

A user attempting to perform transaction 1 would almost certainly contact the administrator for help in this case, and the DBA would have to detect and resolve the conflict.

Possible Causes of Lock Conflicts

- Uncommitted changes
- Long-running transactions
- Unnecessarily high locking levels

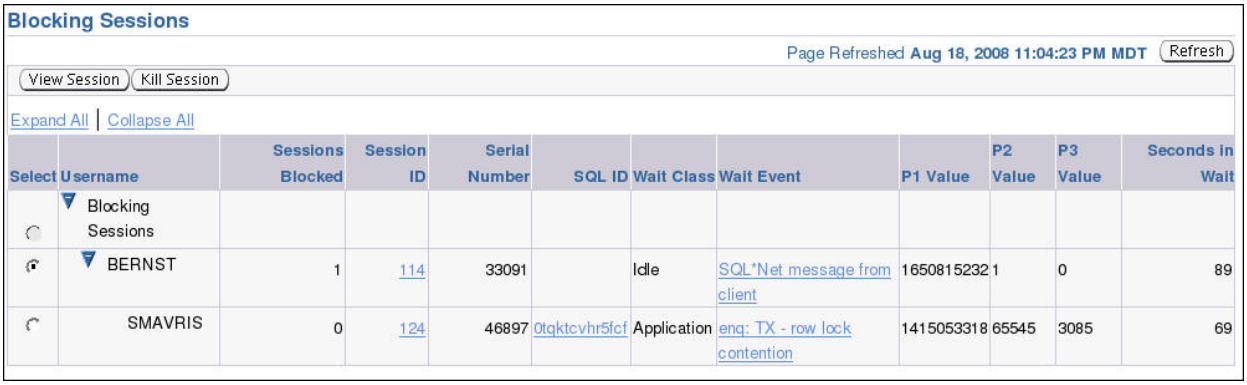
Possible Causes of Lock Conflicts

The most common cause of lock conflicts is an uncommitted change, but there are a few other possible causes:

- **Long-running transactions:** Many applications use batch processing to perform bulk updates. These batch jobs are usually scheduled for times of low or no user activity, but in some cases, they may not have finished or may take too long to run during the low activity period. Lock conflicts are common when transaction and batch processing are being performed simultaneously.
- **Unnecessarily high locking levels:** Not all databases support row-level locking (Oracle added support for row-level locks in 1988 with release 6). Some databases still lock at the page or table level. Developers writing applications that are intended to run on many different databases often write their applications with artificially high locking levels so that Oracle Database behaves similarly to these less capable database systems. Developers who are new to Oracle also sometimes unnecessarily code in higher locking levels than are required by Oracle Database.

Detecting Lock Conflicts

Select Blocking Sessions on the Performance page.



Click the Session ID link to view information about the locking session, including the actual SQL statement.

ORACLE

Detecting Lock Conflicts

Use the Blocking Sessions page in Enterprise Manager to locate lock conflicts. Conflicting lock requests are shown in hierarchical layout, showing the session holding the lock at the top and, below that, any sessions that are enqueued for the lock.

For each session involved in the conflict, you are given the username, session ID, and number of seconds for which the session has been waiting. Drill down to the session ID to see actual SQL statements that are currently being executed or requested by the session.

The Automatic Database Diagnostic Monitor (ADDM) also automatically detects lock conflicts and can advise you on inefficient locking trends.

Resolving Lock Conflicts

To resolve a lock conflict:

- Have the session holding the lock commit or roll back
- Terminate the session holding the lock (in an emergency)

ORACLE

Resolving Lock Conflicts

To resolve a lock conflict, the session holding the lock must release it. The best way to have the session release the lock is to contact the user and ask that the transaction be completed.

In an emergency, it is possible for the administrator to terminate the session holding the lock by clicking the Kill Session button. Remember that when a session is killed, all work within the current transaction is lost (rolled back). A user whose session is killed must log in again and redo all work since the killed session's last commit.

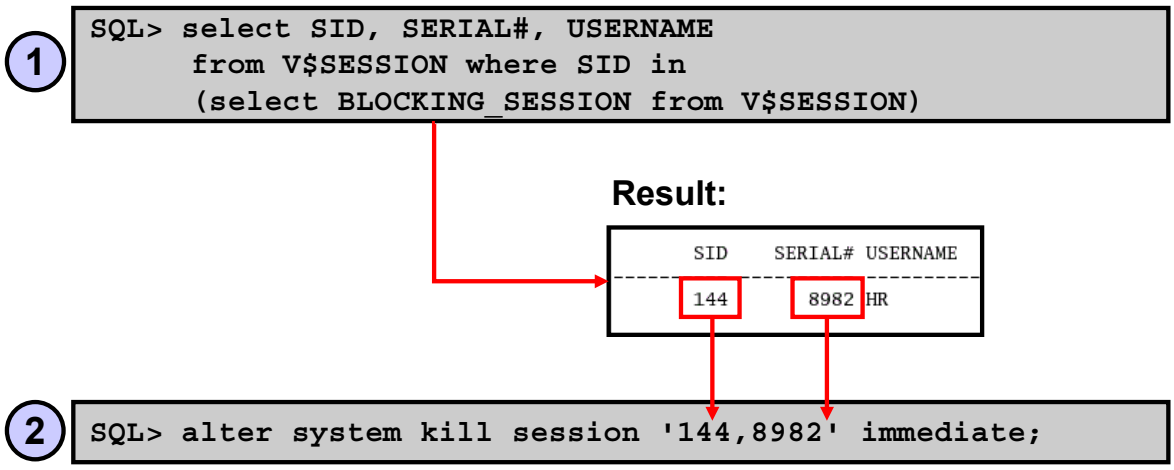
Users whose sessions have been killed receive the following error the next time they try to issue a SQL statement:

```
ORA-03135: connection lost contact
```

Note: The PMON session sniper can automatically kill sessions due to idle timeout, and this can be done by using profiles or Resource Manager.

Resolving Lock Conflicts with SQL

SQL statements can be used to determine the blocking session and kill it.



ORACLE

Resolving Lock Conflicts Using SQL

Session manipulation, like most other tasks in Enterprise Manager, can also be done by issuing SQL statements. The `V$SESSION` table contains details of all connected sessions. The value in `BLOCKING_SESSION` is the session ID of the session that is blocking. If you query for `SID` and `SERIAL#` (where `SID` matches a blocking session ID), you have the information needed to perform the `kill session` operation.

Note: Database Resource Manager can be used to automatically log out sessions that block others and are idle.

Deadlocks

Transaction 1		Transaction 2
`UPDATE employees SET salary = salary x 1.1 WHERE employee_id = 1000;`	9:00	`UPDATE employees SET manager = 1342 WHERE employee_id = 2000;`
`UPDATE employees SET salary = salary x 1.1 WHERE employee_id = 2000;`	9:15	`UPDATE employees SET manager = 1342 WHERE employee_id = 1000;`
`ORA-00060: Deadlock detected while waiting for resource`	9:16	

ORACLE

Deadlocks

A deadlock is a special example of a lock conflict. Deadlocks arise when two or more sessions wait for data that has been locked by the other. Because each is waiting for the other, neither can complete their transaction to resolve the conflict.

Oracle Database automatically detects deadlocks and terminates the statement with an error. The proper response to that error is either commit or rollback, which releases any other locks in that session so that the other session can continue its transaction.

In the example in the slide, transaction 1 must either commit or roll back in response to the deadlock detected error. If it commits, it must resubmit the second update to complete its transaction. If it performs a rollback, it must resubmit both statements to complete its transaction.

Quiz

The lock mechanism defaults to a fine-grained, row-level locking mode.

1. True
2. False

Answer: 1

Quiz

When the deadlock occurs, Oracle database automatically:

1. Waits 300 seconds before terminating both sessions
2. Terminates one statement with an error in one session
3. Terminates the statements with an error in both sessions
4. Takes no action by default and leaves it to DBA

Answer: 2

Summary

In this lesson, you should have learned how to:

- Describe the locking mechanism and how Oracle manages data concurrency
- Monitor and resolve locking conflicts

Practice 9 Overview: Managing Data and Concurrency

This practice covers the following topics:

- Identifying locking conflicts
- Resolving locking conflicts

10

Managing Undo Data

ORACLE

Objectives

After completing this lesson, you should be able to:

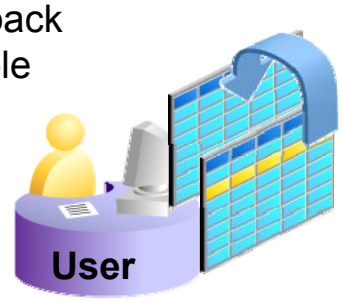
- Explain DML and undo data generation
- Monitor and administer undo data
- Describe the difference between undo data and redo data
- Configure undo retention
- Guarantee undo retention
- Use the Undo Advisor

ORACLE

Undo Data

Undo data is:

- A copy of original, premodified data
- 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



ORACLE

Undo Data

The Oracle database saves the old value (undo data) when a process changes data in a database. It stores the data as it exists before modifications. Capturing undo data enables you to roll back your uncommitted data. Undo supports read-consistent and flashback queries. Undo can also be used to "rewind" (flash back) transactions and tables.

Read-consistent queries provide results that are consistent with the data as of the time a query started. For a read-consistent query to succeed, the original information must still exist as undo information. If the original data is no longer available, you receive a "Snapshot too old" error (`ORA-01555`). As long as the undo information is retained, the Oracle database can reconstruct data to satisfy read-consistent queries.

Flashback queries purposely ask for a version of the data as it existed at some time in the past. As long as undo information for that past time still exists, flashback queries can complete successfully. Oracle Flashback Transaction uses undo to create compensating transactions, to back out a transaction and its dependent transactions. With Oracle Flashback Table, you can recover a table to a specific point in time.

Undo data is also used to recover from failed transactions. A failed transaction occurs when a user session ends abnormally (possibly because of network errors or a failure on the client computer) before the user decides to commit or roll back the transaction. Failed transactions may also occur when the instance crashes or you issue the `SHUTDOWN ABORT` command.

Undo Data (continued)

In case of a failed transaction, the safest behavior is chosen, and the Oracle database reverses all changes made by a user, thereby restoring the original data.

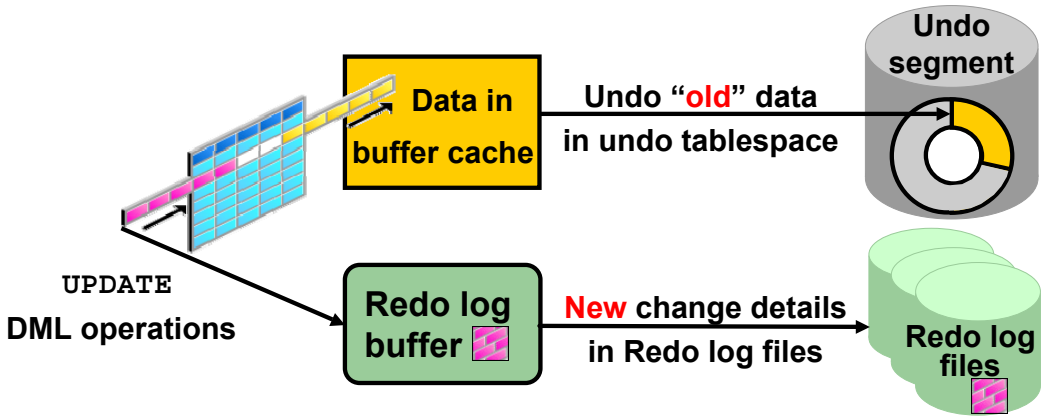
Undo information is retained for all transactions, at least until the transaction is ended by one of the following:

- User undoes a transaction (transaction rolls back).
- User ends a transaction (transaction commits).
- User executes a DDL statement, such as a `CREATE`, `DROP`, `RENAME` or `ALTER` statement. If the current transaction contains any DML statements, the database first commits the transaction and then executes and commits the DDL as a new transaction.
- User session terminates abnormally (transaction rolls back).
- User session terminates normally with an exit (transaction commits).

The amount of undo data that is retained and the time for which it is retained depend on the amount of database activity and the database configuration.

Note: Oracle Flashback Transaction leverages the online redo logs to mine the appropriate undo SQL for execution. It only uses Undo as an artificial time boundary, to determine a redo mining start time for the target transaction, if a transaction start time is not supplied in the flashback transaction invocation.

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.

ORACLE

Transactions and Undo Data

When a transaction starts, it is assigned to an undo segment. Throughout the life of the transaction, when data is changed, the original (before the change) values are copied into the undo segment. You can see which transactions are assigned to which undo segments by checking the `V$TRANSACTION` dynamic performance view.

Undo segments are specialized segments that are automatically created by the instance as needed to support transactions. Like all segments, undo segments are made up of extents, which, in turn, consist of data blocks. Undo segments automatically grow and shrink as needed, acting as a circular storage buffer for their assigned transactions.

Transactions fill extents in their undo segments until a transaction is completed or all space is consumed. If an extent fills up and more space is needed, the transaction acquires that space from the next extent in the segment. After all extents have been consumed, the transaction either wraps around back into the first extent or requests a new extent to be allocated to the undo segment.

Note: Parallel DML and DDL operations can actually cause a transaction to use more than one undo segment. To learn more about parallel DML execution, see the *Oracle Database Administrator's Guide*.

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

ORACLE

Storing Undo Information

Undo segments can exist only in a specialized form of tablespace called an *undo tablespace*. (You cannot create other segment types, such as tables, in the undo tablespace.)

The DBCA automatically creates a smallfile undo tablespace. You can also create a bigfile undo tablespace. However, in a high-volume online transaction processing (OLTP) environment with many short concurrent transactions, contention could occur on the file header. An undo tablespace, stored in multiple data files, can resolve this potential issue.

Although a database may have many undo tablespaces, only one of them at a time can be designated as the current undo tablespace for any instance in the database.

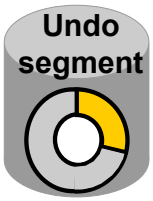
Undo segments are automatically created and always owned by `SYS`. Because the undo segments act as a circular buffer, each segment has a minimum of two extents. The default maximum number of extents depends on the database block size but is very high (32,765 for an 8 KB block size).

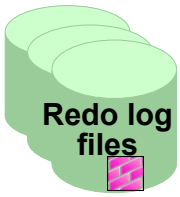
Undo tablespaces are permanent, locally managed tablespaces with automatic extent allocation. They are automatically managed by the database.

Because undo data is required to recover from failed transactions (such as those that may occur when an instance crashes), undo tablespaces can be recovered only while the instance is in the `MOUNT` state. Recovery considerations for undo tablespaces are covered in the lesson titled "Performing Database Recovery."

Undo Data Versus Redo Data

	Undo	Redo
Record of	How to undo a change	How to reproduce a change
Used for	Rollback, read consistency, flashback	Rolling forward database changes
Stored in	Undo segments	Redo log files
Protects against	Inconsistent reads in multiuser systems	Data loss





ORACLE

Undo Data Versus Redo Data

Undo data and redo data seem similar at first, but they serve different purposes. Undo data is needed if there is the need to undo a change, and this occurs for read consistency and rollback. Redo data is needed if there is the need to perform the changes again, in cases where they are lost for some reason. Undo block changes are also written to the redo log.

The process of committing entails a verification that the changes in the transaction have been written to the redo log file, which is persistent storage on the disk, as opposed to memory. In addition, the redo log file is typically multiplexed. As a result, there are multiple copies of the redo data on the disk. Although the changes may not yet have been written to the data files where the table's blocks are actually stored, writing to the persistent redo log is enough to guarantee consistency of the database.

Suppose that a power outage occurs just before committed changes have been reflected in the data files. This situation does not cause a problem because the transaction has been committed. When the system starts up again, it is thus able to roll forward any redo records that are not yet reflected in data files at the time of the outage.

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 undo tablespace to a fixed size
- Avoiding space and “snapshot too old” errors

ORACLE

Managing Undo

The Oracle database provides *automatic undo management*, which is a fully automated mechanism for managing undo information and space in a dedicated undo tablespace for all sessions. The system automatically tunes itself to provide the best possible retention of undo information . More precisely, the undo retention period for autoextending tablespaces is tuned to be slightly longer than the longest-running active query. For fixed-size undo tablespaces, the database dynamically tunes for best possible retention.

Automatic undo management is the default for Oracle Database 11*g* (and later releases). Manual undo management is supported for backward compatibility with Oracle8*i* and earlier releases but requires more DBA interaction. In manual undo management mode, undo space is managed through rollback segments (not through undo tablespace).

Note: Oracle strongly recommends that you use automatic undo management.

Although by default the Oracle database manages undo data and space automatically, you may need to perform some tasks if your database is using Flashback operations. The administration of undo should prevent space errors, the use of too much space, and “Snapshot too old” errors.

Configuring Undo Retention

UNDO_RETENTION specifies (in seconds) how long already committed undo information is to be retained. The only time you must set this parameter is when:

- The undo tablespace has the AUTOEXTEND option enabled
- You want to set undo retention for LOBs
- You want to guarantee retention

ORACLE

Configuring Undo Retention

The UNDO_RETENTION initialization parameter specifies (in seconds) the low threshold value of undo retention. Set the minimum undo retention period for the autoextending undo tablespace to be as long as the longest expected Flashback operation. For autoextending undo tablespaces, the system retains undo for at least the time specified in this parameter, and automatically tunes the undo retention period to meet the undo requirements of the queries. But this autotuned retention period may be insufficient for your Flashback operations.

For fixed-size undo tablespaces, the system automatically tunes for the best possible undo retention period on the basis of undo tablespace size and usage history; it ignores UNDO_RETENTION unless retention guarantee is enabled. So for automatic undo management, the UNDO_RETENTION setting is used for the three cases listed in the slide.
In cases other than these three, this parameter is ignored.

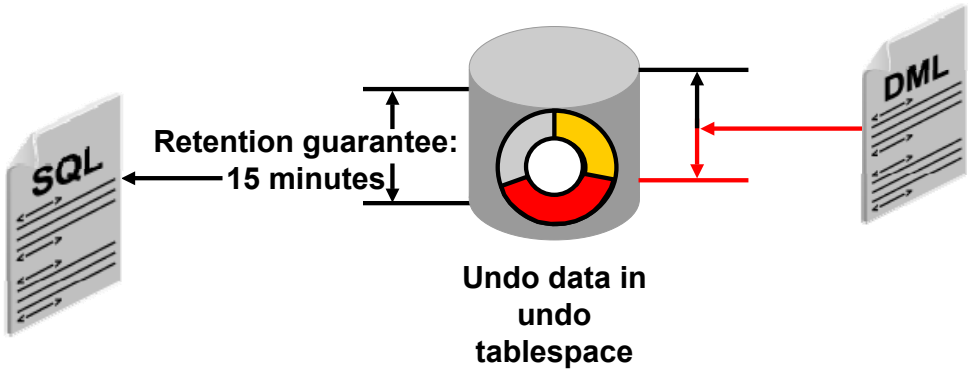
Configuring Undo Retention (continued)

Undo information is divided into three categories:

- **Uncommitted undo information (Active):** Supports a currently running transaction, and is required if a user wants to roll back or if the transaction has failed. Uncommitted undo information is never overwritten.
- **Committed undo information (Unexpired):** Is no longer needed to support a running transaction but is still needed to meet the undo retention interval. It is also known as "unexpired" undo information. Committed undo information is retained when possible without causing an active transaction to fail because of lack of space.
- **Expired undo information (Expired):** Is no longer needed to support a running transaction. Expired undo information is overwritten when space is required by an active transaction.

Guaranteeing Undo Retention

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



SELECT statements running 15 minutes or less are always satisfied.

A transaction will fail if it generates more undo than there is space.

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

ORACLE

Guaranteeing Undo Retention

The default undo behavior is to overwrite the undo information of committed transactions that has not yet expired rather than to allow an active transaction to fail because of lack of undo space.

This behavior can be changed by guaranteeing retention. With guaranteed retention, undo retention settings are enforced even if they cause transactions to fail.

`RETENTION GUARANTEE` is a tablespace attribute rather than an initialization parameter. This attribute can be changed only with SQL command-line statements. The syntax to change an undo tablespace to guarantee retention is:

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

To return a guaranteed undo tablespace to its normal setting, use the following command:

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

The retention guarantee applies only to undo tablespaces. Attempts to set it on a non-undo tablespace result in the following error:

```
SQL> ALTER TABLESPACE example RETENTION GUARANTEE;
ERROR at line 1:
ORA-30044: 'Retention' can only specified for undo tablespace
```

Changing an Undo Tablespace to a Fixed Size

Reasons:

- Supporting Flashback operations
- Limiting tablespace growth

Workflow:

1. Run regular workload.
2. Self-tuning mechanism establishes minimum required size.
3. (Optional) Use Undo Advisor, which calculates required size for future growth.
4. (Optional) Change undo tablespace to a fixed size.

ORACLE

Changing an Undo Tablespace to a Fixed Size

You might have two reasons for changing the undo tablespace to a fixed size: to support Flashback operations (where you expect future use of the undo) or to prevent the tablespace from growing too large.

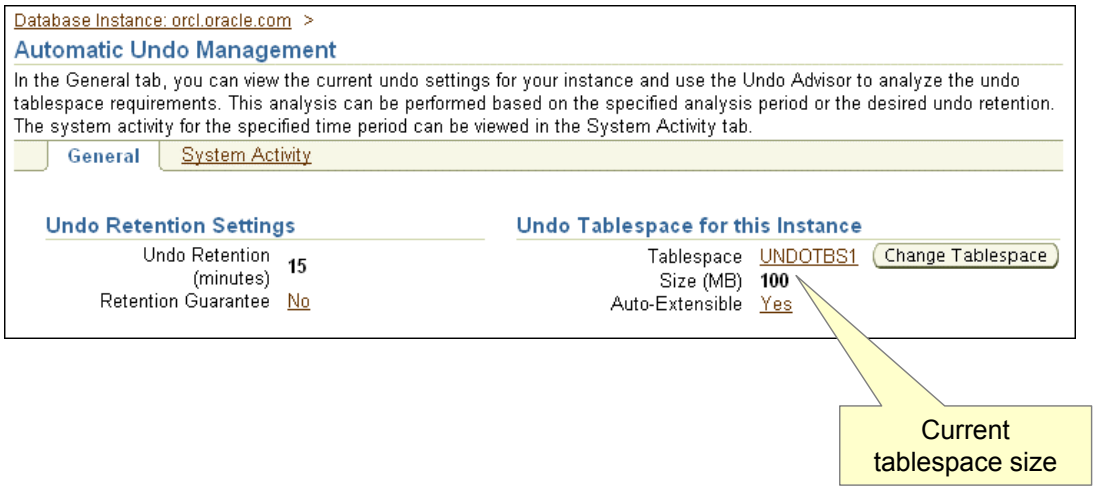
If you decide to change the undo tablespace to a fixed size, you must choose a large enough size to avoid the following two errors:

- DML failures (because there is not enough space to the undo for new transactions)
- "Snapshot too old" errors (because there was insufficient undo data for read consistency)

Oracle recommends that you run a regular, full workload, allowing the undo tablespace to grow to its minimum required size. The automatically gathered statistics include the duration of the longest-running query and the undo generation rate. Computing the minimum undo tablespace size based on these statistics is advisable for a system without Flashback operations, and for a system for which you do not expect longer-running queries in the future.

You can use the Undo Advisor to enter your desired duration for the undo period for longer-running queries and flashback.

General Undo Information

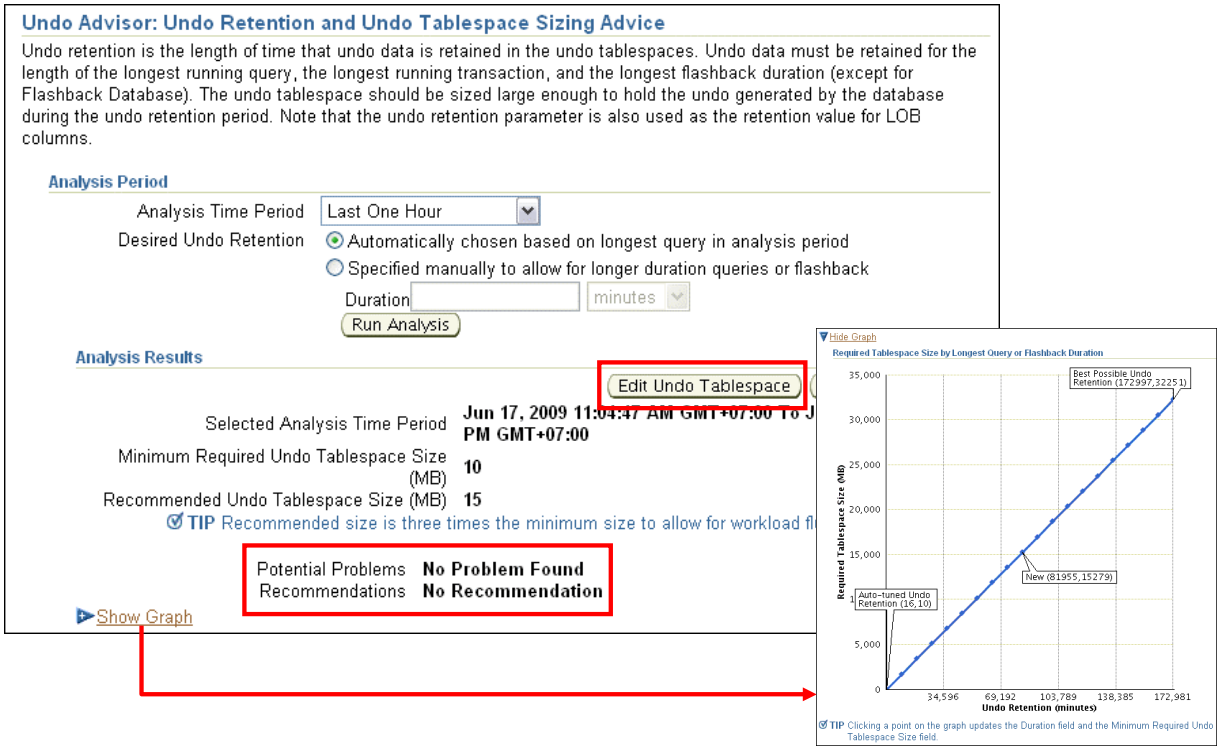


General Undo Information

In Enterprise Manager, select Server > Automatic Undo Management.

There are two pages: General and System Activity. In the top part of the General page, you see the Undo Retention Settings and information about the undo tablespace for this instance.

Using the Undo Advisor



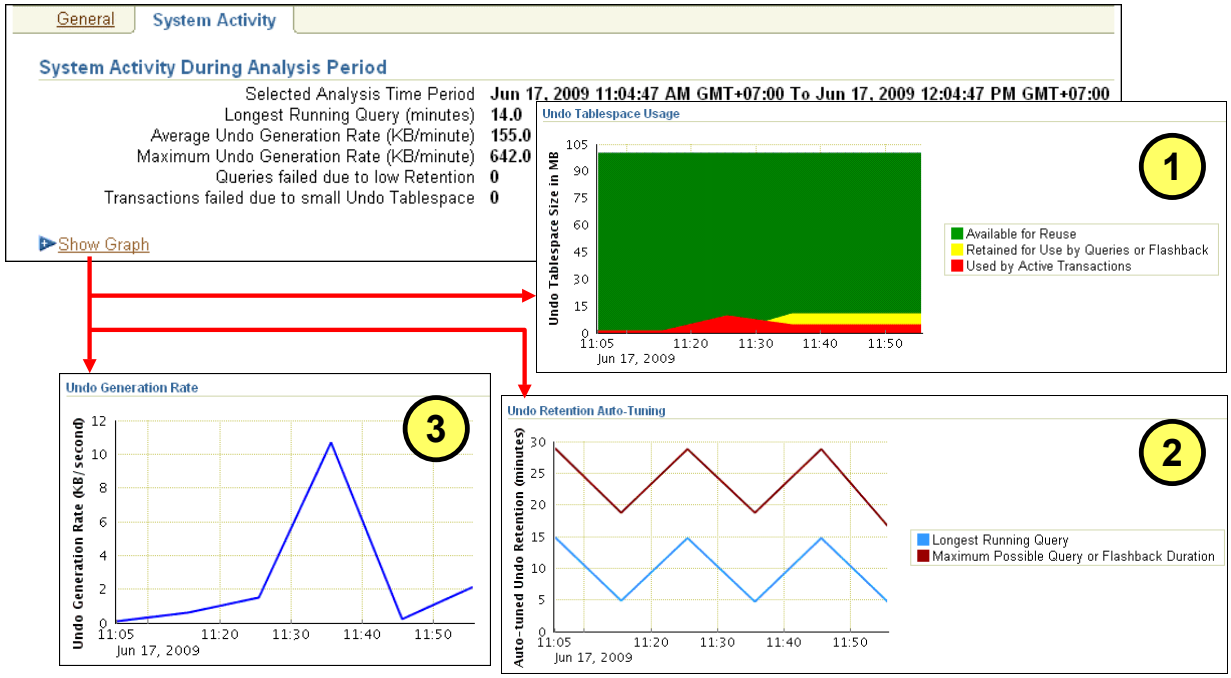
Using the Undo Advisor

The middle part of the General Undo page is your access to the Undo Advisor. It provides an estimate of the undo tablespace size required to satisfy a given undo retention.

The analysis region of the advisor displays the tablespace size required to support the retention period. You can click a point on the graph to see the tablespace size required to support the selected period.

Click the Edit Undo Tablespace button, and then click Edit in the Datafile section to change the undo tablespace to a fixed size.

Viewing System Activity



Viewing System Activity

The top part of the page displays system activity during the selected period.

Beneath this, there are three graphs:

1. **Undo Tablespace Usage:** Shows the tablespace size (in MB) by days of the month
2. **Undo Retention Auto-Tuning:** Visualizes the tuned undo retention (in minutes) by days of the month
3. **Undo Generation Rate:** Displays the undo generation (in KB per seconds) by days of the month

Quiz

All you need to do to guarantee that all queries under 15 minutes will find the undo data needed for read consistency, is set the `UNDO_RETENTION` parameter to 15 minutes.

1. True
2. False

Answer: 2

Quiz

Which statement does not relate to undo data?

1. Provides a record of how to undo a change
2. Is used for rollback, read consistency, and flashback
3. Is stored in memory only, not written to disk
4. Protects against inconsistent reads in a multiuser system

Answer: 3

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
- Use the Undo Advisor

ORACLE

Practice 10 Overview: Managing Undo Segments

This practice covers the following topics:

- Viewing system activity
- Calculating undo tablespace sizing to support a 48-hour retention interval
- Modifying an undo tablespace to support a 48-hour retention interval

ORACLE

11

Implementing Oracle Database Auditing

Objectives

After completing this lesson, you should be able to:

- Describe DBA responsibilities for security and auditing
- Enable standard database auditing
- Specify audit options
- Review audit information
- Maintain the audit trail

ORACLE

Objectives

This lesson is a starting point for learning about Oracle Security. Additional information is provided in the following documentation:

- *Oracle Database Concepts*
- *Oracle Database Administrator's Guide*
- *Oracle Database Security Guide*

Additional training is provided in the following courses:

- *Oracle Database 11g: Administration Workshop II*
- *Oracle Database 11g: Security*

Separation of Responsibilities

- Users with DBA privileges must be trusted.
  - Abuse of trust
  - Audit trails protecting the trusted position
- DBA responsibilities must be shared.
- Accounts must never be shared.
- The DBA and the system administrator must be different people.
- Separate operator and DBA responsibilities.

ORACLE

Separation of Responsibilities

These are the main requirements to satisfy the separation of duties.

DBAs must be trusted: It is difficult to restrict a DBA. To do his or her job, the DBA requires high-level privileges. A DBA has a position of trust and must be thoroughly vetted. Even a trusted DBA must have accountability. Consider the following:

- **Abuse of trust:** A DBA can potentially misuse the encrypted passwords from the `DBA_USERS` view.
- **Audit trails protecting the trusted position:** When auditing is carefully implemented and guidelines have been followed, the audit trail can show that a particular person has not violated procedures or committed a damaging act. If a malicious user tries to cast suspicion on a trusted user, well-designed audit trails catch the act.

Oracle Database Vault: The Oracle Database Vault option can be used for situations in which the separation of duties must be enforced by the database, or for situations in which the DBA is not allowed to view data in some or all database schemas.

Database Security

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

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

ORACLE

Database Security

Oracle Database 11*g* provides the industry's best framework for a secure system. But for that framework to be effective, the database administrator must follow best practices and continually monitor database activity.

Restricting Access to Data and Services

All users must not have access to all data. Depending on what is stored in your database, restricted access can be mandated by business requirements, by customer expectations, and (increasingly) by legal restrictions. Credit card information, health-care data, identity information, and so on must be protected from unauthorized access. The Oracle database provides extremely fine-grained authorization controls to limit database access. Restricting access must include applying the principle of least privilege.

Database Security (continued)

Authenticating Users

To enforce access controls on sensitive data, the system must first know who is trying to access the data. Compromised authentication can render all other security precautions useless. The most basic form of user authentication is challenging users to provide something that they know, such as a password. Ensuring that passwords follow simple rules can greatly increase the security of your system. Stronger authentication methods include requiring users to provide something that they have, such as a token or public key infrastructure (PKI) certificate. An even stronger form of authentication is to identify users through a unique biometric characteristic such as a fingerprint, an iris scan, bone structure patterns, and so on. The Oracle database supports advanced authentication techniques (such as token-, biometric-, and certificate-based identification) through the Advanced Security option. User accounts that are not in use must be locked to prevent attempts to compromise authentication.

Monitoring for Suspicious Activity

Even authorized and authenticated users can sometimes compromise your system. Identifying unusual database activity (such as an employee who suddenly begins querying large amounts of credit card information, research results, or other sensitive information) can be the first step to detecting information theft. The Oracle database provides a rich set of auditing tools to track user activity and identify suspicious trends.

Monitoring for Compliance

Monitoring or auditing must be an integral part of your security procedures.

Review the following:

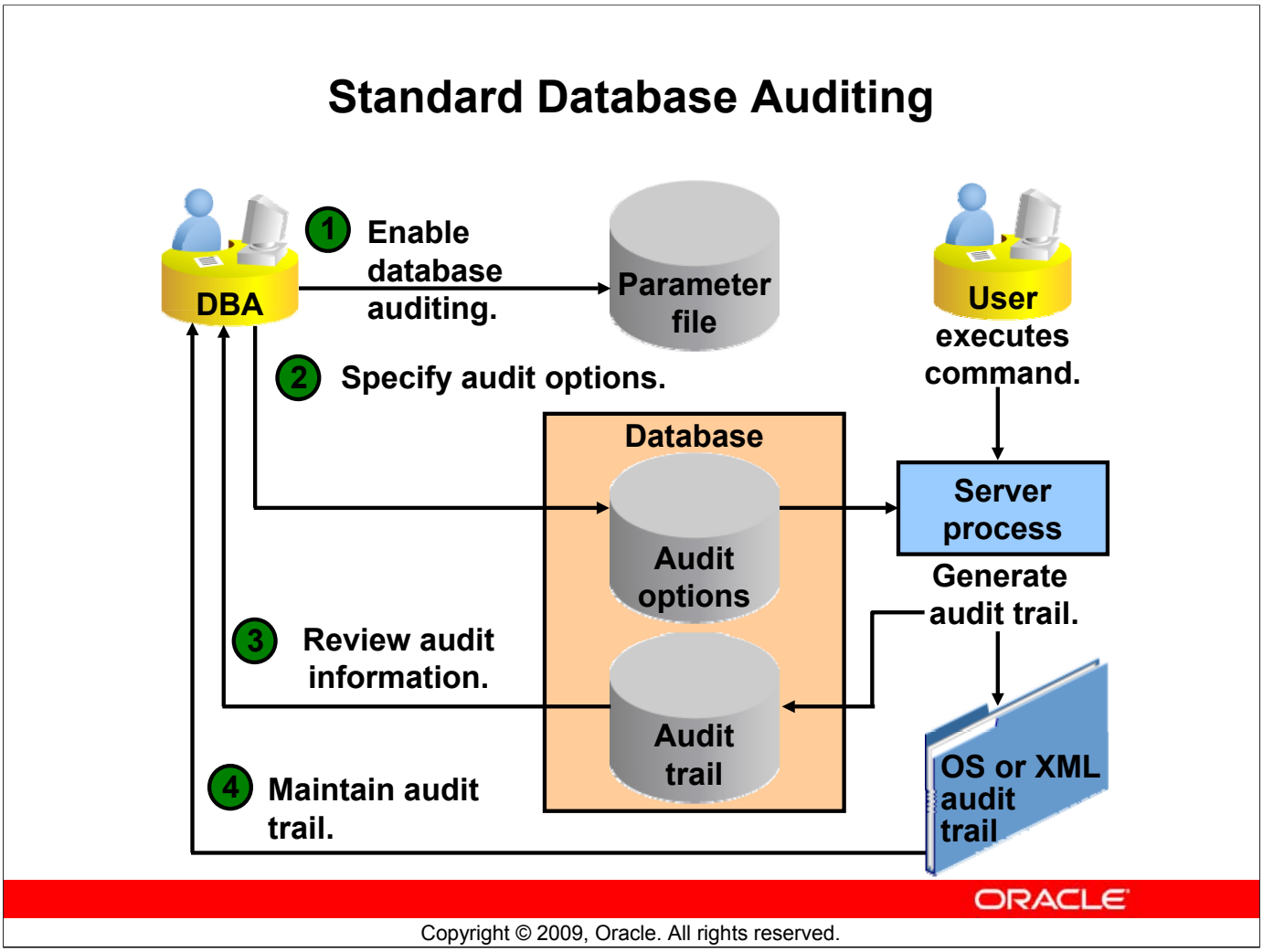
- Mandatory auditing
- Standard database auditing
- Value-based auditing
- Fine-grained auditing (FGA)
- SYSDBA (and SYSOPER) auditing

ORACLE

Monitoring for Compliance

Auditing, which means capturing and storing information about what is happening in the system, increases the amount of work the system must do. Auditing must be focused so that only events that are of interest are captured. Properly focused auditing has minimal impact on system performance. Improperly focused auditing can significantly affect performance.

- **Mandatory auditing:** All Oracle databases audit certain actions regardless of other audit options or parameters. The reason for mandatory audit logs is that the database needs to record some database activities, such as connections by privileged users.
- **Standard database auditing:** Enabled at the system level by using the `AUDIT_TRAIL` initialization parameter. After you enable auditing, select the objects and privileges that you want to audit and set the auditing properties with the `AUDIT` command.
- **Value-based auditing:** Extends standard database auditing, capturing not only the audited event that occurred but also the actual values that were inserted, updated, or deleted. Value-based auditing is implemented through database triggers.
- **Fine-grained auditing (FGA):** Extends standard database auditing, capturing the actual SQL statement that was issued rather than only the fact that the event occurred
- **SYSDBA (and SYSOPER) auditing:** Separates the auditing duties between the DBA and an auditor or security administrator who monitors the DBA activities in an operating system audit trail



Standard Database Auditing

To use database auditing, you must first set the static `AUDIT_TRAIL` parameter to point to a storage location for audit records. This enables database auditing. After you enable database auditing and specify the auditing options (login events, exercise of system and object privileges, or the use of SQL statements), the database begins collecting audit information.

If `AUDIT_TRAIL` is set to `OS`, the audit records are stored in the operating system's audit system. In a Windows environment, this is the event log. In a UNIX or Linux environment, audit records are stored in a file that is specified with the `AUDIT_FILE_DEST` parameter.

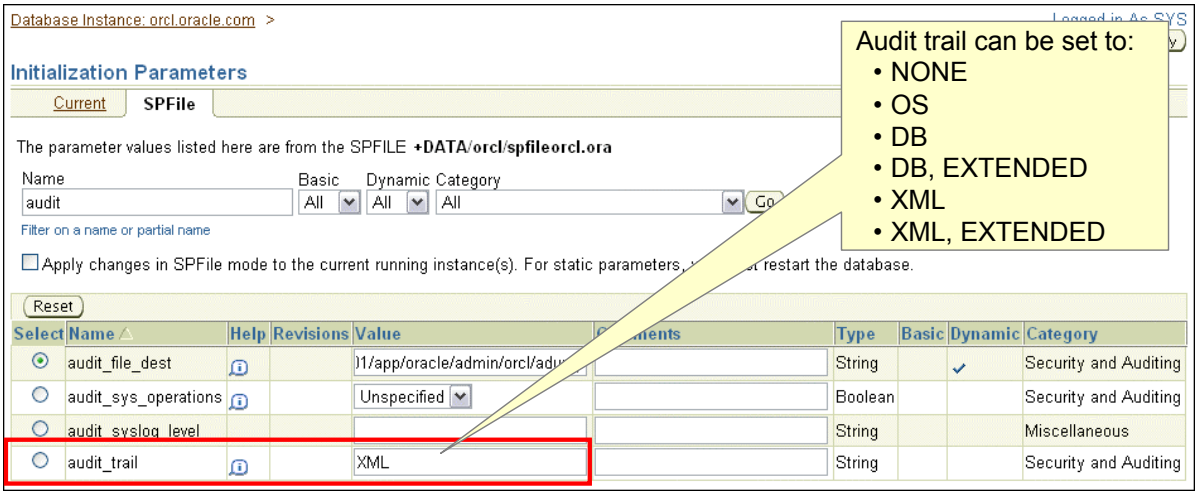
If the `AUDIT_TRAIL` parameter is set to `DB` or `DB, EXTENDED`, you can review audit records in the `DBA_AUDIT_TRAIL` view, which is part of the `SYS` schema.

If `AUDIT_TRAIL` is set to `XML` or to `XML,EXTENDED`, the audit records are written to XML files in the directory to which the `AUDIT_FILE_DEST` parameter points. The `V$XML_AUDIT_TRAIL` view allows you to view all the XML files in this directory.

Maintaining the audit trail is an important administrative task. Depending on the focus of the audit options, the audit trail can grow very large very quickly. If not properly maintained, the audit trail can create so many records that it affects the performance of the system. Audit overhead is directly related to the number of records that are produced.

Configuring the Audit Trail

Use AUDIT_TRAIL to enable database auditing.




```
ALTER SYSTEM SET AUDIT_TRAIL='XML' SCOPE=SPFILE;
```

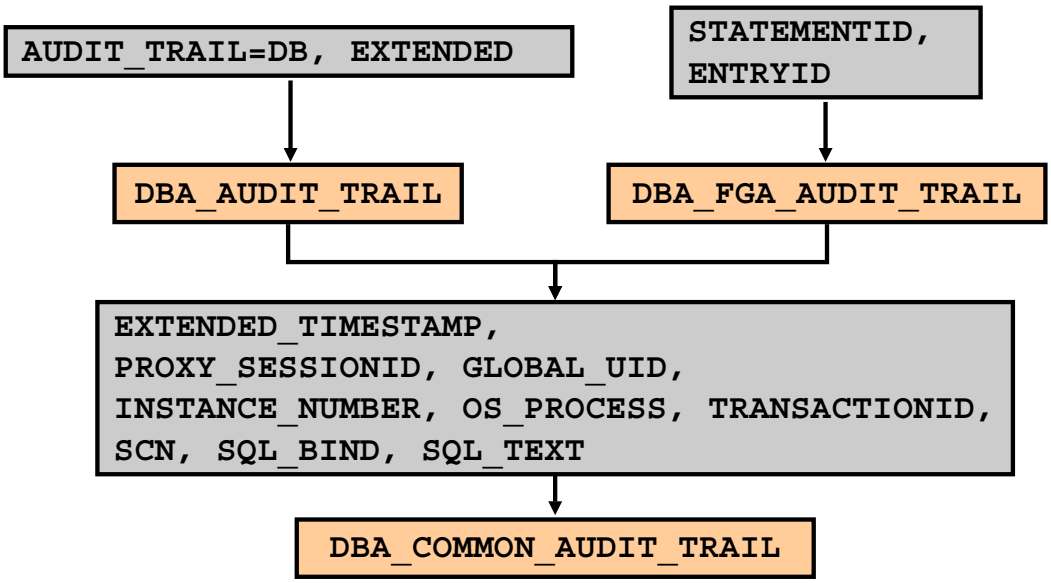
Restart database after modifying this static initialization parameter.

ORACLE

Configuring the Audit Trail

You can set the AUDIT_TRAIL parameter using either Enterprise Manager (from the Initialization Parameters page) or using the ALTER SYSTEM SET command from SQL*Plus. Because this is a static parameter, you will have to restart the database before the change will take effect. If you created the database using the Database Configuration Assistant (DBCA) then by default the audit_trail parameter is set to DB. When AUDIT_TRAIL is set to DB, the default behavior is to record the audit trail in the database AUD$ table. These audits should not have a large impact on database performance, for most sites. Oracle recommends the use of OS audit trail files. If you created the database manually (with the CREATE DATABASE command), AUDIT_TRAIL is set to NONE by default.

Uniform Audit Trails



ORACLE

Uniform Audit Trails

The Oracle database tracks the same fields for standard and fine-grained auditing, enabling you to easily analyze database activities. To accomplish this, both the standard audit trail and the fine-grained audit trail have attributes that complement each other.

The extra information that is collected by standard auditing includes:

- The system change number (SCN), which records every change to the system
- The exact SQL text executed by the user and the bind variables used with the SQL text. These columns appear only if you have specified `AUDIT_TRAIL=DB, EXTENDED`.

The extra information that is collected by fine-grained auditing includes:

- A serial number for each audit record
- A statement number that links multiple audit entries that originate from a single statement

Common attributes include:

- A global time stamp in Universal Time Coordinates (UTC). This field is useful for monitoring across servers in separate geographic locations and time zones.
- An instance number that is unique for each Real Application Clusters (RAC) instance
- A transaction identifier that helps you group audit records of a single transaction

The `DBA_COMMON_AUDIT_TRAIL` view combines standard and fine-grained audit log records.

Specifying Audit Options

- SQL statement auditing:

```
AUDIT table;
```

- System-privilege auditing (nonfocused and focused):

```
AUDIT select any table, create any trigger;
AUDIT select any table BY hr BY SESSION;
```

- Object-privilege auditing (nonfocused and focused):

```
AUDIT ALL on hr.employees;
AUDIT UPDATE,DELETE on hr.employees BY ACCESS;
```

ORACLE

Specifying Audited Options

SQL statement auditing: The statement shown in the slide can audit any data definition language (DDL) statement that affects a table, including `CREATE TABLE`, `DROP TABLE`, `TRUNCATE TABLE`, and so on. SQL statement auditing can be focused by username or by success or failure:

```
SQL> AUDIT TABLE BY hr WHENEVER NOT SUCCESSFUL;
```

System-privilege auditing: Can be used to audit the exercise of any system privilege (such as `DROP ANY TABLE`). It can be focused by username or by success or failure. By default, auditing is `BY ACCESS`. Each time an audited system privilege is exercised, an audit record is generated. You can choose to group those records with the `BY SESSION` clause so that only one record is generated per session. (In this way, if a user issues multiple update statements against a table belonging to another user, you gather only one audit record.) Consider using the `BY SESSION` clause to limit the performance and storage impact of system-privilege auditing.

Object-privilege auditing: Can be used to audit actions on tables, views, procedures, sequences, directories, and user-defined data types. This type of auditing can be focused by success or failure and grouped by session or access. Unlike system-privilege auditing, the default grouping is by session. You must explicitly specify `BY ACCESS` if you want a separate audit trail record to be generated for each action.

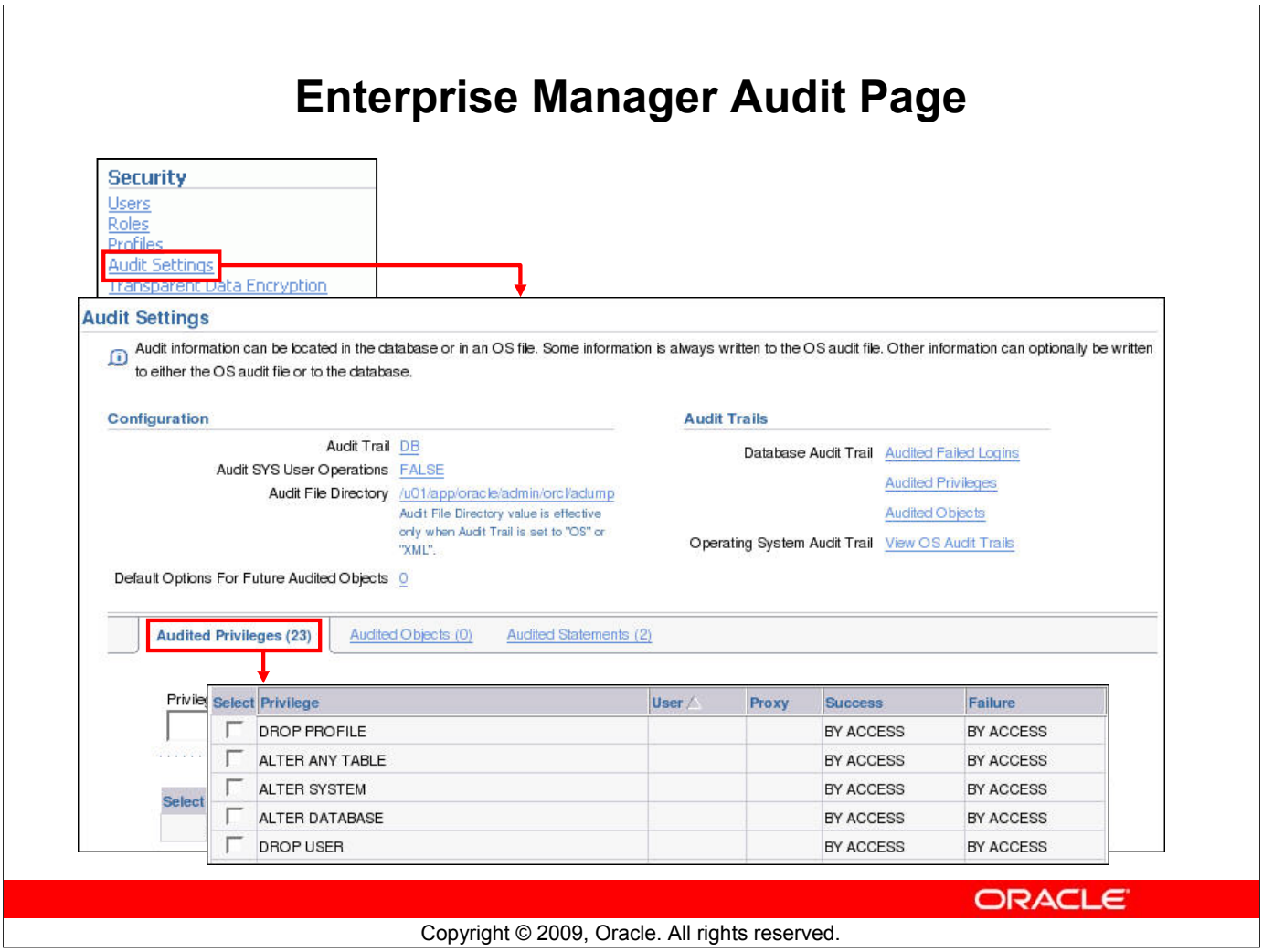
Default Auditing

Privileges Audited by Default		
ALTER ANY PROCEDURE	CREATE ANY LIBRARY	GRANT ANY PRIVILEGE
ALTER ANY TABLE	CREATE ANY PROCEDURE	GRANT ANY ROLE
ALTER DATABASE	CREATE ANY TABLE	DROP ANY PROCEDURE
ALTER PROFILE	CREATE EXTERNAL JOB	DROP ANY TABLE
ALTER SYSTEM	CREATE PUBLIC DATABASE LINK	DROP PROFILE
ALTER USER	CREATE SESSION	DROP USER
AUDIT SYSTEM	CREATE USER	EXEMPT ACCESS POLICY
CREATE ANY JOB	GRANT ANY OBJECT PRIVILEGE	

Statements Audited by Default
SYSTEM AUDIT BY ACCESS
ROLE BY ACCESS

Default Auditing

When auditing is enabled in Oracle Database 11g certain privileges and statements that are very important to security are audited by default. These privileges and statements are listed in the slide and are audited for all users on success and failure, and by access.



Enterprise Manager Audit Page

You can reach the Audit page from the Database Control Home page by clicking the Server tab and then clicking the Audit Settings link in the Security region.

The Audit page contains the following regions:

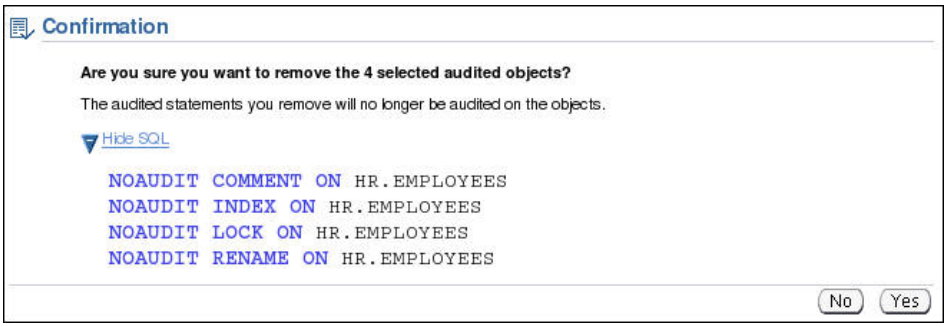
- **Configuration:** Shows the current configuration parameter values and contains links to edit the parameter values
- **Audit Trails:** Provides easy-to-use access to the audit information that has been collected

Use these tabbed pages to set and unset audit options:

- **Audited Privileges:** Shows privileges that are audited
- **Audited Objects:** Shows objects that are audited
- **Audited Statements:** Shows statements that are audited

Using and Maintaining Audit Information

Disable audit options if you are not using them.

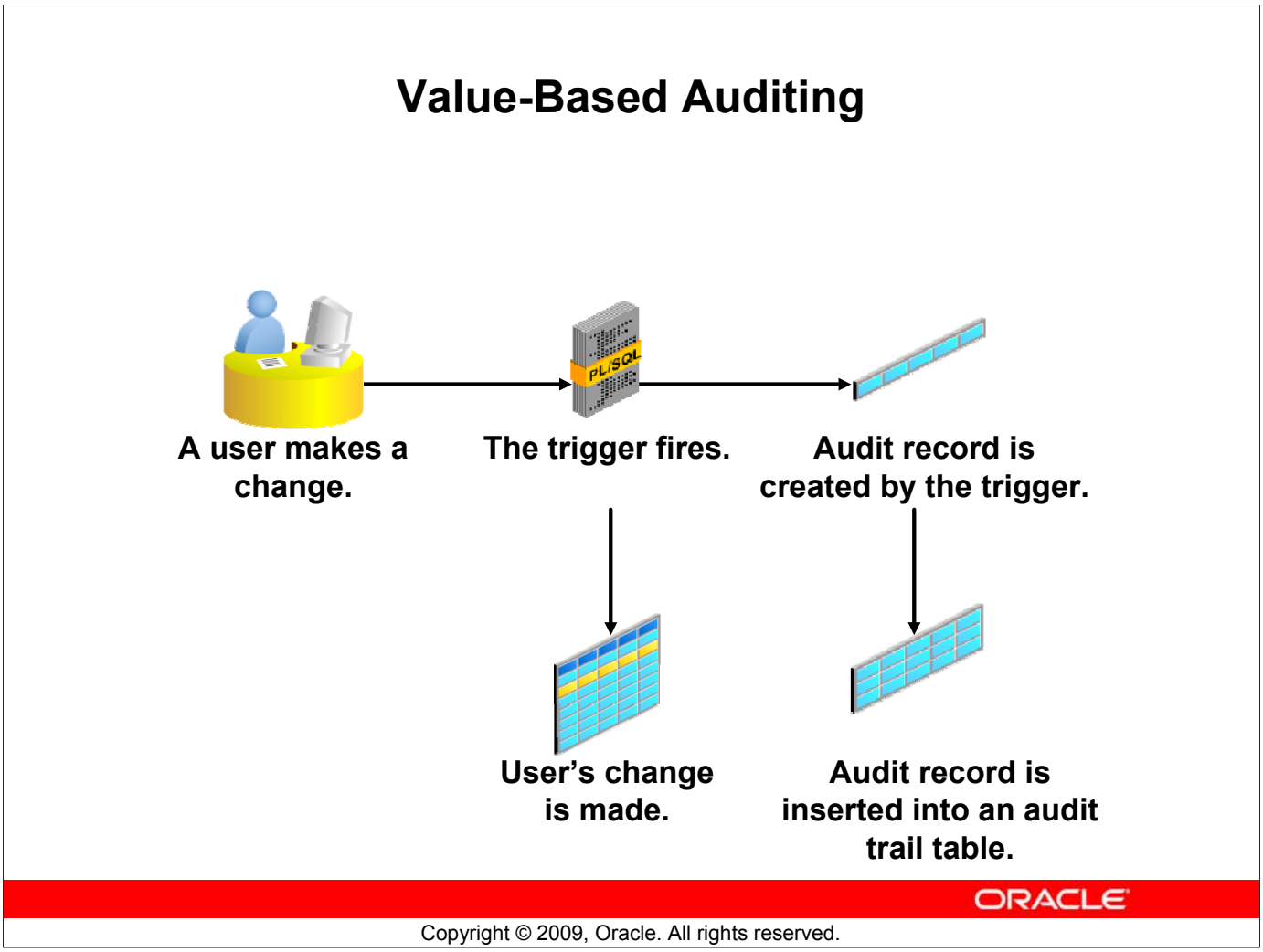


ORACLE

Using and Maintaining Audit Information

Best Practice Tip

Auditing incurs a performance penalty proportional to the number of writes to the audit trail. To tailor the audit options to the needs of your site, enable only those options that are necessary to meet the security policy. Focus the auditing to reduce the number of audit trail entries.



Value-Based Auditing

Database auditing records the inserts, updates, and deletes that have occurred in audited objects but does not capture the actual values that are changed. To extend database auditing, value-based auditing leverages database triggers (event-driven PL/SQL constructs) to capture the changed values.

When a user inserts, updates, or deletes data from a table with the appropriate trigger attached, the trigger works in the background to copy audit information to a table that is designed to contain the audit information. Value-based auditing tends to degrade performance more than standard database auditing because the audit trigger code must be executed each time the insert, update, or delete operation occurs. The degree of degradation depends on the efficiency of the trigger code. Value-based auditing must be used only in situations in which the information captured by standard database auditing is insufficient.

Value-based auditing is implemented by user or third-party code. The Oracle database provides the PL/SQL constructs to allow value-based audit systems to be built.

Value-Based Auditing (continued)

The key to value-based auditing is the audit trigger, which is simply a PL/SQL trigger that is constructed to capture audit information.

Example of a typical audit trigger:

```
CREATE OR REPLACE TRIGGER system.hrsalary_audit
       AFTER UPDATE OF salary
       ON hr.employees
       REFERENCING NEW AS NEW OLD AS OLD
       FOR EACH ROW
   BEGIN
    IF :old.salary != :new.salary THEN
        INSERT INTO system.audit_employees
        VALUES (sys_context('userenv','os_user'), sysdate,
        sys_context('userenv','ip_address'),
        :new.employee_id ||
        ' salary changed from '||:old.salary||
        ' to '||:new.salary);
  END IF;
   END;
/
```

This trigger focuses auditing to capture changes to the salary column of the `hr.employees` table. When a row is updated, the trigger checks the salary column. If the old salary is not equal to the new salary, the trigger inserts an audit record into the `audit_employees` table (created via a separate operation in the `SYSTEM` schema). The audit record includes the username, the IP address from which the change is made, the primary key identifying which record is changed, and the actual salary values that are changed.

Database triggers can also be used to capture information about user connections in cases where standard database auditing does not gather sufficient data. With login triggers, the administrator can capture data that identifies the user who is connecting to the database. Examples include the following:

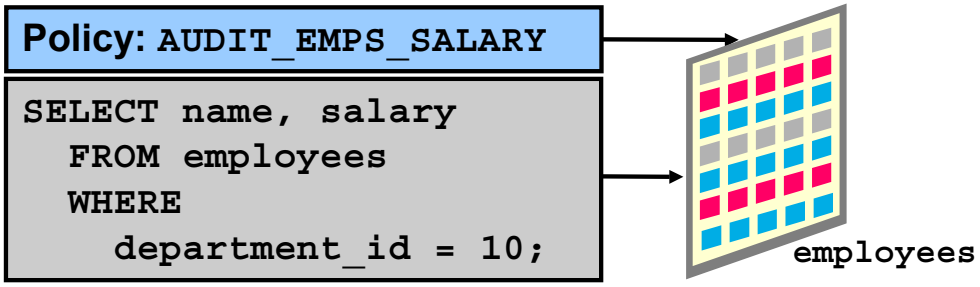
- IP address of the person logging in
- First 48 characters of the program name that is used to connect to the instance
- Terminal name that is used to connect to the instance

For a complete list of user parameters, see the section titled "`SYS_CONTEXT`" in the *Oracle Database SQL Reference*.

Value-based triggers have been superceded in many cases by the fine-grained auditing (FGA) feature.

Fine-Grained Auditing

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



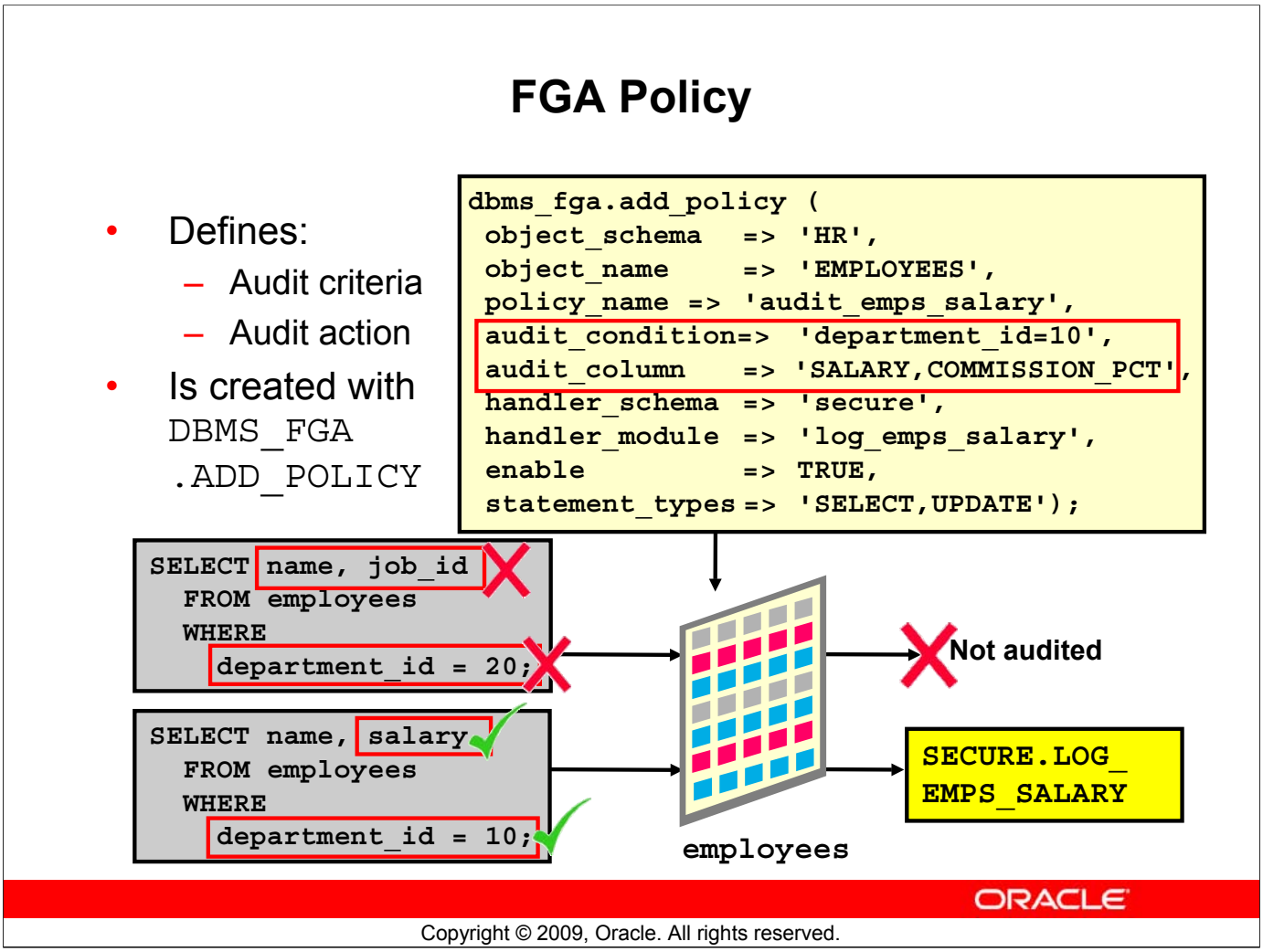
Fine-Grained Auditing

Database auditing records the fact that an operation has occurred but does not capture information about the statement that caused the operation. Fine-grained auditing (FGA) extends that capability to enable the capture of actual SQL statements that query or manipulate data.

FGA also allows auditing to be more narrowly focused than standard or value-based database auditing.

FGA options can be focused by individual columns in a table or view, and can even be conditional so that audits are captured only if certain administrator-defined specifications are met. More than one relevant column is supported for an FGA policy. By default, if any one of these columns is present in the SQL statement, it is audited. DBMS_FGA.ALL_COLUMNS and DBMS_FGA.ANY_COLUMNS are provided to audit on the basis of whether any or all of the relevant columns are used in the statement.

Use the DBMS_FGA PL/SQL package to create an audit policy on the target table or view. If any of the rows returned from a query block match the audited column and the specified audit condition, an audit event causes an audit record to be created and stored in the audit trail. As an option, the audit event can also execute a procedure. FGA automatically focuses auditing at the statement level. A SELECT statement that returns thousands of rows thus generates only one audit record.



FGA Policy

The example in the slide shows the creation of a fine-grained auditing policy with the `DBMS_FGA.ADD_POLICY` procedure, which accepts the following arguments.

Policy Name

You assign each FGA policy a name when you create it. The example in the slide names the policy `AUDIT_EMPS_SALARY` by using the following argument:

```
policy_name => 'audit_emps_salary'
```

Audit Condition

The audit condition is a SQL predicate that defines when the audit event must fire. In the slide example, all rows in department 10 are audited by using the following condition argument:

```
audit_condition => 'department_id = 10'
```

Note: Fine-grained auditing looks at the result set of the query, so with the FGA policy shown on the slide, queries that returns rows matching the policy specifications will cause an audit record to be created. For example, in the query "`select * from employees`", all rows including those having "10" in department_id may be returned, so an audit row is created.

FGA Policy (continued)

Audit Column

The audit column defines the data that is being audited. An audit event occurs if this column is included in the SELECT statement or if the audit condition allows the selection. The example in the slide audits two columns by using the following argument:

```
audit_column => 'SALARY,COMMISION_PCT'
```

This argument is optional. If it is not specified, only the AUDIT_CONDITION argument determines whether an audit event must occur.

Object

The object is the table or view that is being audited. It is passed as two arguments:

- The schema that contains the object
- The name of the object

The example in the slide audits the hr.employees table by using the following arguments:

```
object_schema => 'hr'
object_name => 'employees'
```

Handler

An optional event handler is a PL/SQL procedure that defines additional actions that must be taken during auditing. For example, the event handler can send an alert page to the administrator. If it is not defined, an audit event entry is inserted into the audit trail. If an audit event handler is defined, the audit entry is inserted into the audit trail and the audit event handler is executed.

The audit event entry includes the FGA policy that caused the event, the user executing the SQL statement, and the SQL statement and its bind variables.

The event handler is passed as two arguments:

- The schema that contains the PL/SQL program unit
- The name of the PL/SQL program unit

The example in the slide executes the SECURE.LOG_EMPS_SALARY procedure by using the following arguments:

```
handler_schema => 'secure'
handler_module => 'log_emps_salary'
```

By default, audit trail always writes the SQL text and SQL bind information to LOBs. The default can be changed (for example, if the system would suffer performance degradation).

Status

The status indicates whether the FGA policy is enabled. In the slide example, the following argument enables the policy:

```
enable => TRUE
```

Audited DML Statement: Considerations

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

Not audited because none of the records involved are for department 10.

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

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

ORACLE

Audited DML Statement: Considerations

With an FGA policy defined for DML statements, a DML statement is audited if the data rows (both new and old) that are being manipulated meet the policy predicate criteria.

However, if relevant columns are also specified in the policy definition, the statement is audited when the data meets the FGA policy predicate and the statement references the relevant columns defined.

For `DELETE` statements, specifying relevant columns during policy definition is not useful because all columns in a table are touched by a `DELETE` statement. Therefore, a `DELETE` statement is always audited regardless of the relevant columns.

`MERGE` statements are supported by FGA. The underlying `INSERT`, `UPDATE`, and `DELETE` statements are audited if they meet the defined `INSERT`, `UPDATE`, or `DELETE` FGA policies.

Using the previously defined FGA policy, the first statement is not audited whereas the second one is. None of the employees in department 10 receive a commission, but `employee_id=200` specifies an employee in department 10.

FGA Guidelines

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

FGA Guidelines

For the `SELECT` statements, FGA captures the statement itself and not the actual rows. However, when FGA is combined with Flashback Query, the rows can be reconstructed as they existed at that point in time.

For more details about Flashback Query, see the lesson titled "Performing Flashback."

For more details about the `DBMS_FGA` package, see the *Oracle Database PL/SQL Packages and Types Reference*.

SYSDBA Auditing

Users with SYSDBA or SYSOPER privileges can connect when the database is closed.

- Audit trail must be stored outside the database.
- Connections as SYSDBA or SYSOPER are always audited.
- You can enable additional auditing of SYSDBA or SYSOPER actions with AUDIT_SYS_OPERATIONS.
- You can control the audit trail with AUDIT_FILE_DEST.

ORACLE

SYSDBA Auditing

The SYSDBA and SYSOPER users have privileges to start up and shut down the database. Because they may make changes while the database is closed, the audit trail for these privileges must be stored outside the database. The Oracle database automatically captures login events by the SYSDBA and SYSOPER users. This provides a valuable way to track authorized or unauthorized SYSDBA and SYSOPER actions, but it is useful only if the OS audit trail is reviewed.

The Oracle database always captures the login events of privileged users. Other actions are captured if DBA auditing is specifically enabled. Enable auditing of the SYSDBA and SYSOPER users by setting the initialization parameter:

AUDIT_SYS_OPERATIONS=TRUE (The default is FALSE.)

If the SYS operations are audited, the AUDIT_FILE_DEST initialization parameter controls the storage location of the audit records. On a Windows platform, the audit trail defaults to the Windows event log. On UNIX and Linux platforms, audit records are stored in the AUDIT_FILE_DEST location.

Maintaining the Audit Trail

The audit trail should be maintained with the following best-practice guidelines:

- Review and store old records.
- Prevent storage problems.
- Avoid loss of records.

Maintaining the Audit Trail

Each type of audit trail must be maintained. Basic maintenance must include reviewing the audit records and removing older records from the database or operating system. Audit trails can grow to fill the available storage. If the file system is full, the system may crash or simply cause performance problems. If the database audit trail fills the tablespace, audited actions do not complete. If the audit trail fills the system tablespace, the performance of other operations is affected before audit operations halt.

The audit trail for standard auditing is stored in the `AUD$` table. The audit trail for FGA is the `FGA_LOG$` table. Both these tables are created in the `SYSTEM` tablespace by default. You can move these tables to another tablespace by using the Data Pump export and import utilities.

Note: Moving the audit tables out of the `SYSTEM` tablespace is not supported.

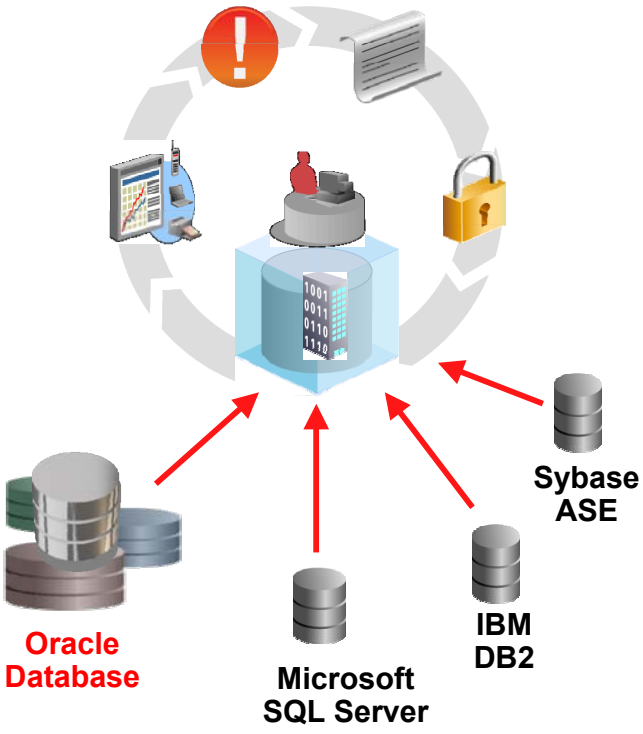
Audit records can be lost during the process of removing records from the audit tables.

Best Practice Tip

Use an export based on a time stamp, and then delete rows from the audit trail based on the same time stamp.

Oracle Audit Vault

- Consolidate and secure audit data
  - Oracle 9*i* Release 2 and higher
  - SQL Server 2000, 2005
  - IBM DB2 UDB 8.5 & 9.2
  - Sybase ASE 12.5 - 15.0
  - Secure and scalable
  - Cleanup of source Oracle audit data
- Centralized reporting
  - Updated reports interface using widely popular Oracle Application Express
  - Standard reports for compliance
  - New custom reports
- Alert on security threats
  - Detect and alert on security relevant events



Oracle Audit Vault

Key benefits of Oracle Audit Vault include the following:

- Oracle Audit Vault transparently collects and consolidates audit data from Oracle databases, beginning with Oracle9i Database Release 2, Microsoft SQL Server 2000 & 2005, IBM DB2 Unix, Linux, Windows 8.2 & 9.5 and Sybase ASE 12.5 – 15.0.
- Oracle Audit Vault helps organizations simplify compliance reporting with built-in reports and custom reports. In addition, Oracle Audit Vault provides an open audit warehouse schema that can be accessed from Oracle BI Publisher, Oracle Application Express, or any third-party reporting tools.
- Oracle Audit Vault helps detect and prevent insider threats by alerting you to suspicious activity.
- Central to Oracle Audit Vault is a secure and scalable audit warehouse built on Oracle's data warehousing technology and secured with Oracle's database security products, including Oracle Database Vault and Oracle Advanced Security. Oracle Audit Vault includes Oracle Partitioning to improve manageability and performance.
- Oracle Audit Vault helps organizations lower IT costs with centralized management of database audit settings (policies), making it easier for IT security officers and internal auditors to do their jobs.

Refer to the Audit Vault documentation for additional information.

Quiz

Standard database auditing captures the before and after changes of a DML transaction.

1. True
2. False

Answer: 2

Quiz

Auditing of SYSDBA and SYSOPER actions is enabled by default.

1. True
2. False

Answer: 2

Summary

In this lesson, you should have learned how to:

- Describe DBA responsibilities for security and auditing
- Enable standard database auditing
- Specify audit options
- Review audit information
- Maintain the audit trail

Practice 11 Overview: Implementing Oracle Database Security

This practice covers the following topics:

- Enabling standard database auditing
- Specifying audit options for the HR.JOBS table
- Updating the table
- Reviewing audit information
- Maintaining the audit trail

ORACLE

12

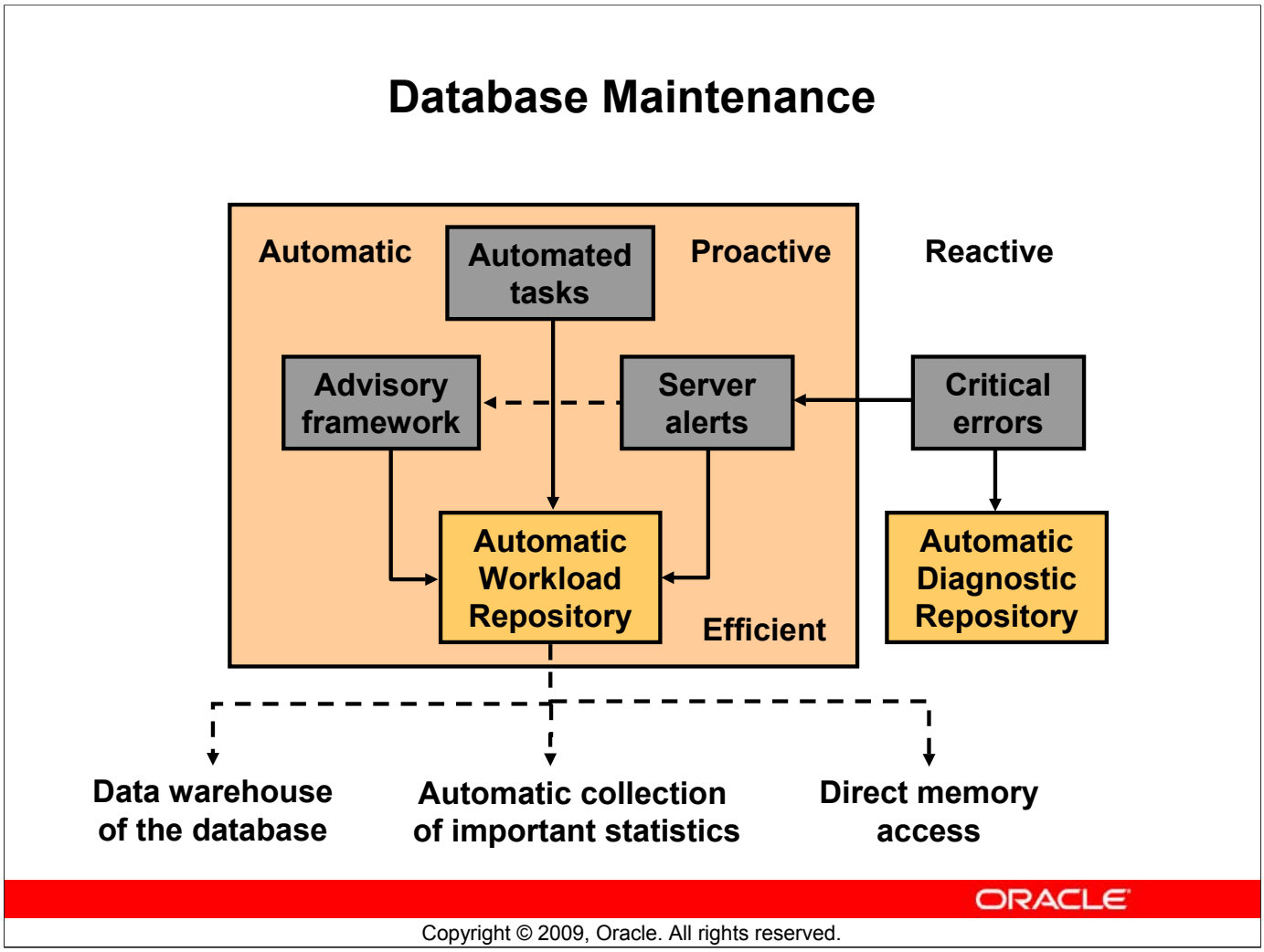
Database Maintenance

Objectives

After completing this lesson, you should be able to:

- Manage optimizer statistics
- Manage the Automatic Workload Repository (AWR)
- Use the Automatic Database Diagnostic Monitor (ADDM)
- Describe and use the advisory framework
- Set alert thresholds
- Use server-generated alerts
- Use automated tasks

ORACLE



Database Maintenance

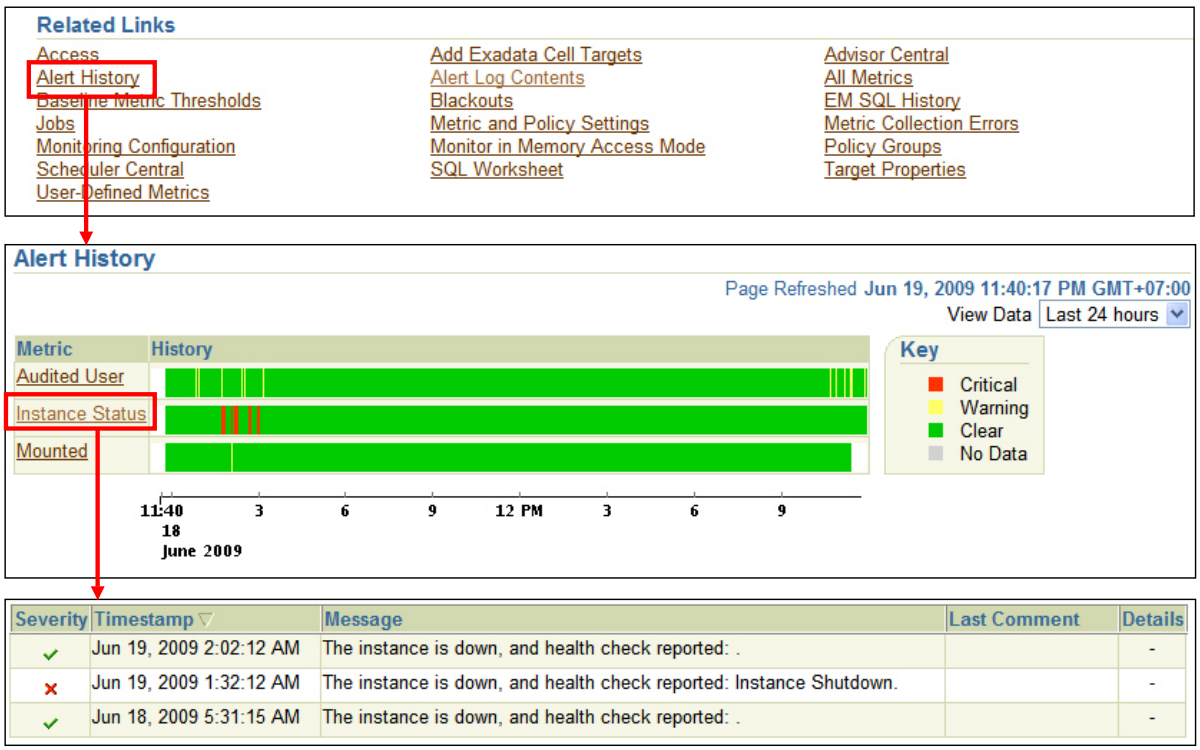
Proactive database maintenance is made easy by the sophisticated infrastructure of the Oracle database, including the following main elements:

- The Automatic Workload Repository (AWR) is a built-in repository in each Oracle database.
  At regular intervals, the Oracle database server makes a snapshot of all its vital statistics and workload information and stores this data in the AWR. The captured data can be analyzed by you, by the database itself, or by both.
- Using automated tasks, the database performs routine maintenance operations such as regular backups, refreshing optimizer statistics, and database health checks.

Reactive database maintenance includes critical errors and conditions discovered by database health checkers:

- For problems that cannot be resolved automatically and require administrators to be notified (such as running out of space), the Oracle database server provides server-generated alerts. The Oracle database server by default monitors itself and sends out alerts to notify you of problems. The alerts notify you and often also provide recommendations on how to resolve reported problem.
- Recommendations are generated from a number of advisors, each of which is responsible for a subsystem. For example, there are memory, segment, and SQL advisors.

Viewing the Alert History



Viewing the Alert History

The Alert History page displays a chart that shows the alert history of the current database in segments of time that you designate. An alert indicates a potential problem: either a warning or critical threshold for a monitored metric, or an indication that a target is no longer available. Click the metric name listed on the Alert History page to get detailed statistics, graphs, and actual timestamps for each alert. There is also a location to enter comments related to the alert such as resolution information.

Terminology

- Automatic Workload Repository (AWR): Infrastructure for data gathering, analysis, and solutions recommendations
- AWR Baseline: A set of AWR snapshots for performance comparison
- Metric: Rate of change in a cumulative statistic
- Statistics: Data collections providing database and object detail
  - Optimizer statistics: Used by query optimizer
  - Database statistics: Used for performance
- Threshold: A boundary value against which metric values are compared

ORACLE

Terminology

The *Automatic Workload Repository* (AWR) provides services to internal Oracle server components to collect, process, maintain, and use performance statistics for problem detection and self-tuning purposes. *Active Session History* (ASH) is the history of recent session activity stored in the AWR.

Statistics are collections of data that provide more details about the database and the objects in it. Optimizer statistics are used by the query optimizer to choose the best execution plan for each SQL statement. Database statistics provide information for performance monitoring.

AWR snapshots include database statistics and metrics, application statistics (transaction volumes, response time), operating system statistics, and other measures. An *AWR baseline* is a set of AWR snapshots collected over a period of time. The baseline is used for performance comparison, either current performance versus the baseline or one baseline compared to another.

The *System Moving Window* baseline is collected by default in Oracle Database 11*g*. The System Moving Window baseline is a changing set of snapshots that include the last eight days of snapshots by default. This baseline becomes valid after sufficient data has been collected and the statistics calculation occurs. The statistics calculation is scheduled for every Saturday at midnight by default.

Oracle Optimizer: Overview

The Oracle optimizer determines the most efficient execution plan and is the most important step in the processing of any SQL statement.

The optimizer:

- Evaluates expressions and conditions
- Uses object and system statistics
- Decides how to access the data
- Decides how to join tables
- Determines the most efficient path

ORACLE

Oracle Optimizer: Overview

The optimizer is the part of the Oracle database that creates the execution plan for a SQL statement. The determination of the execution plan is an important step in the processing of any SQL statement and can greatly affect execution time.

The execution plan is a series of operations that are performed in sequence to execute the statement. The optimizer considers many factors related to the referenced objects and the conditions specified in the query. The information necessary to the optimizer includes:

- Statistics gathered for the system (I/O, CPU, and so on) as well as schema objects (number of rows, index, and so on)
- Information in the dictionary
- `WHERE` clause qualifiers
- Hints supplied by the developer

When you use diagnostic tools such as Enterprise Manager, `EXPLAIN PLAN`, and SQL*Plus `AUTOTRACE`, you can see the execution plan that the optimizer chooses.

Note: The Oracle optimizer has two names based on its functionality: the *query optimizer* and the *Automatic Tuning Optimizer*.

Optimizer Statistics

Optimizer statistics are:

- A snapshot at a point in time
- Persistent across instance restarts
- Collected automatically

```
SQL> SELECT COUNT(*) FROM hr.employees;
  COUNT(*)
----------
       214
SQL> SELECT num_rows FROM dba_tables
  2 WHERE owner='HR' AND table_name = 'EMPLOYEES';
  NUM_ROWS
----------
       107
```

ORACLE

Optimizer Statistics

Optimizer statistics include table, column, index, and system statistics. Statistics for tables and indexes are stored in the data dictionary. These statistics are not intended to provide real-time data. They provide the optimizer a *statistically* correct snapshot of data storage and distribution, which the optimizer uses to make decisions on how to access data.

The statistics that are collected include:

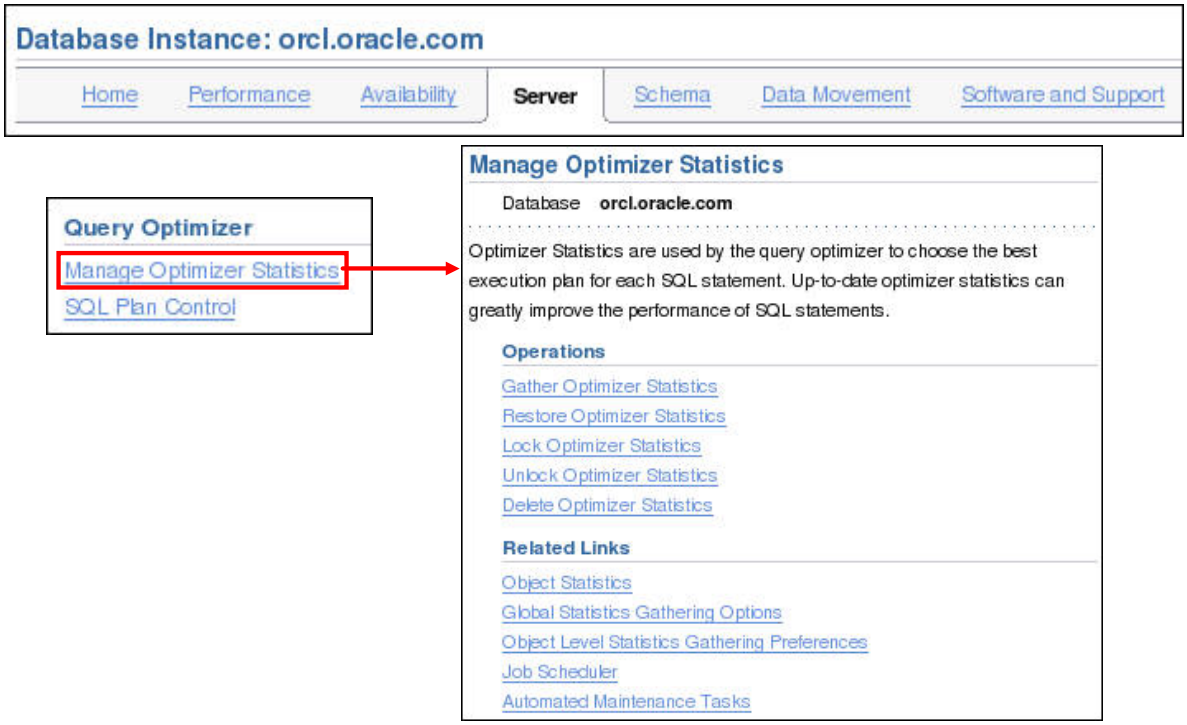
- Size of the table or index in database blocks
- Number of rows
- Average row size and chain count (tables only)
- Height and number of deleted leaf rows (indexes only)

As data is inserted, deleted, and modified, these facts change. Because the performance impact of maintaining real-time data distribution statistics is prohibitive, these statistics are updated by periodically gathering statistics on tables and indexes.

Optimizer statistics are collected automatically by an automatic maintenance job that runs during predefined maintenance windows once daily by default. System statistics are operating system characteristics that are used by the optimizer. These statistics are not collected automatically. For details about collecting system statistics, see the *Oracle Database Performance Tuning Guide*.

Optimizer statistics are not the same as the database performance statistics that are gathered in the AWR snapshot.

Using the Manage Optimizer Statistics Page



Using the Manage Optimizer Statistics Page

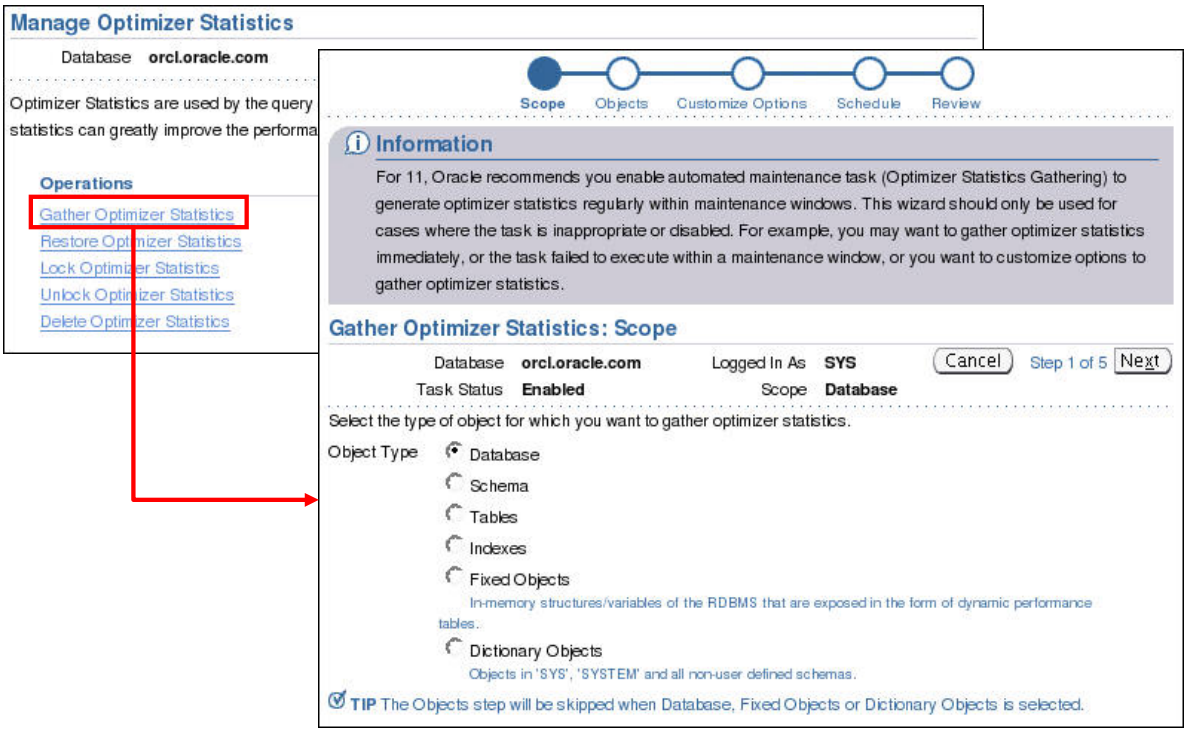
To manage optimizer statistics in Enterprise Manager, click the Server tab and then click Manage Optimizer Statistics under the Query Optimizer section. From this page, you can perform the following tasks on statistics:

- Gather optimizer statistics manually.
- Restore optimizer statistics to a point in the past. The chosen point in time must be within the optimizer statistics retention period, which defaults to 30 days.
- Lock optimizer statistics to guarantee that the statistics for certain objects are never overwritten. This is useful if statistics have been calculated for a certain table at a time when well-representative data is present, and you want to always have those statistics. No fluctuations in the table affect the statistics if they are locked.
- Unlock optimizer statistics to undo the previously done lock.
- Delete optimizer statistics to delete statistics.

Best Practice Tip

Use the automatic maintenance tasks to gather optimizer statistics. To enable the task for gathering optimizer statistics gathering, you must ensure that the `STATISTICS_LEVEL` initialization parameter is set to `TYPICAL` or `ALL`.

Gathering Optimizer Statistics Manually



Gathering Optimizer Statistics Manually

You may need to gather statistics manually at certain times, such as when the contents of a table have changed so much between automatic gathering jobs that the statistics no longer represent the table accurately. This is common for large tables that experience more than a 10 percent change in size in a 24-hour period.

Best practice tip: Collect statistics often enough that the table never changes by more than about 10 percent between collection periods. This may require manual statistics collection or additional maintenance windows.

Statistics can be manually collected by using either Enterprise Manager or the `DBMS_STATS` package. System statistics can be gathered only by using the `DBMS_STATS` package. System statistics describe the system's hardware characteristics, such as I/O and CPU performance and utilization, to the query optimizer.

The Gather Optimizer Statistics menu selection starts a wizard that allows you to select the scope, objects, options, and schedule for a job that will gather optimizer statistics. The wizard submits a `DBMS_STATS.GATHER_*_STATS` job at the scope you specify: table, schema, or database. In this wizard, you set the preferences for the default values used by the `DBMS_STATS` package and you schedule this job to run at a time that you determine.

Gathering Optimizer Statistics Manually (continued)

Gathering statistics manually for routine statistics collection is not recommended because the statistics are gathered more efficiently and with less impact on users during the maintenance windows. A manual job can also be submitted if the automatic job has failed or been disabled.

You can also gather optimizer statistics with the DBMS_STATS package directly:

```
SQL> EXEC dbms_stats.gather_table_stats('HR','EMPLOYEES');
SQL> SELECT num_rows FROM dba_tables
  2  WHERE owner='HR' AND table_name = 'EMPLOYEES';
  NUM_ROWS
----------
       214
```

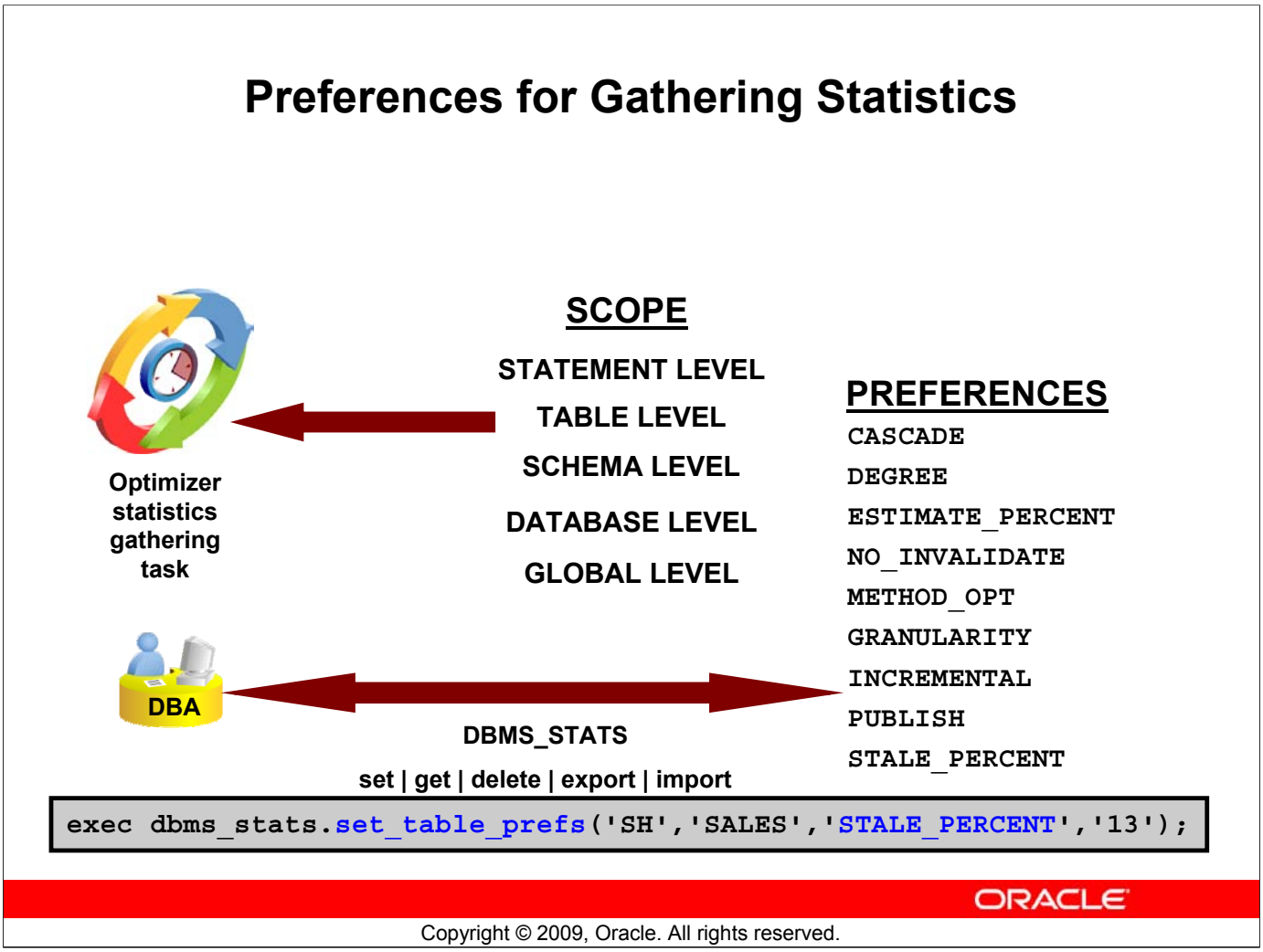
Notice that the number of rows now correctly reflects what was in the table at the time that the statistics were gathered. DBMS_STATS also enables manual collection of statistics for an entire schema or even for the whole database.

System statistics do not change unless the workload significantly changes. As a result, system statistics do not need frequent adjustment. The DBMS_STATS.GATHER_SYSTEM_STATS procedure will collect system statistics over a specified period, or you can start the gathering of system statistics and make another call to stop gathering.

Best practice tip: Use the following command when you create a database:

```
SQL> EXEC dbms_stats.gather_system_stats('NOWORKLOAD');
```

The NOWORKLOAD option takes a few minutes (depending on the size of the database) and captures estimates of I/O characteristics such as average read seek time and I/O transfer rate.



Preferences for Gathering Statistics

The `DBMS_STATS.GATHER_*_STATS` procedures can be called at various levels to gather statistics for an entire database or for individual objects such as tables. When the `GATHER_*_STATS` procedures are called, several of the parameters are often allowed to default. The supplied defaults work well for most of the objects in the database, but for some objects or schemas the defaults need to be changed. Instead of running manual jobs for each of these objects, Oracle Database 11*g* allows you to set values (called *preferences*) for individual objects, schemas, or databases, or to change the default values with the global-level command.

The preferences specify the parameters that are given to the gather procedures. The `SET_*_PREFS` procedures create preference values for any object that is not owned by `SYS` or `SYSTEM`. The expected use is that the DBA will set the global preferences for any parameters that should be database-wide. These will be applied for any parameter that is allowed to default.

The `SET_DATATBASE_PREFS` procedure iterates over all the tables and schemas in the database setting the specified preference. `SET_SCHEMA_PREFS` iterates over the tables in the specified schema. `SET_TABLE_PREFS` sets the preference value for a single table.

All object preferences—whether set at the database, schema, or table level—are held in a single table. Changing the preferences at the schema level overwrites the preferences that were previously set at the table level.

Preferences for Gathering Statistics (continued)

When the various gather procedures execute, they retrieve the object-level preferences that were set for each object. You can view the object-level preferences in the DBA_TAB_STAT_PREFS view. Any preferences that are not set at the object level will be set to the global-level preferences. You can see the global preferences by calling the DBMS_STATS.GET_PREFS procedure for each preference.

You can set, get, delete, export, and import those preferences at the table, schema, database, and global levels. The preference values are expected to be set from global to table levels, applying the preferences to the smallest group last.

Preferences in Oracle Database 11*g*:

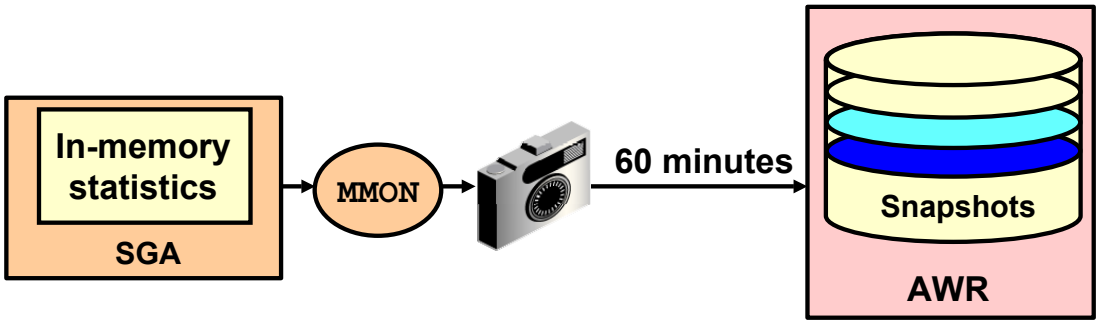
- CASCADE determines whether index statistics are collected as part of gathering table statistics.
- DEGREE sets the degree of parallelism that is used for gathering statistics.
- PUBLISH is used to decide whether to publish the statistics to the dictionary or store them in a private area. This enables the DBA to validate the statistics before publishing them to the data dictionary with the PUBLISH_PENDING_STATS procedure.
- STALE_PERCENT is used to determine the threshold level at which an object is considered to have stale statistics. The value is a percentage of the rows modified since the last statistics gathering. The example changes the 10 percent default to 13 percent for SH.SALES only.
- INCREMENTAL is used to gather global statistics on partitioned tables in an incremental way.
- METHOD_OPT determines the columns and histogram parameters that are used to gather column statistics.
- GRANULARITY determines the granularity of statistics to collect (which is pertinent only if the table is partitioned).
- NO_INVALIDATE is used to determine whether to invalidate cursors.
- ESTIMATE_PERCENT is used to determine the number of rows to sample to obtain good statistics. It is a percentage of the number of rows in the table.

Note: For details about these preferences, see the DBMS_STATS documentation in the *Oracle Database PL/SQL Packages and Types Reference*.

Preferences may be deleted with the DBMS_STATS.DELETE_*_PREFS procedures at the table, schema, and database levels. You can reset the global preferences to the recommended values with the DBMS_STATS.RESET_PARAM_DEFAULTS procedure.

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



ORACLE

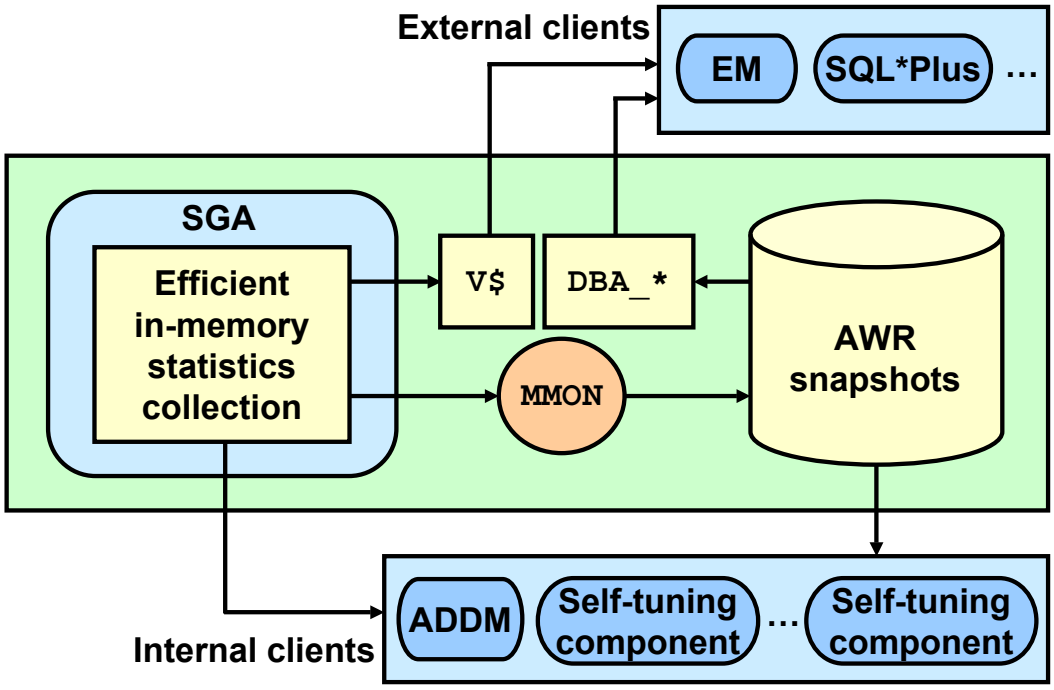
Automatic Workload Repository (AWR)

The AWR is the infrastructure that provides services to Oracle Database 11*g* components to collect, maintain, and utilize statistics for problem detection and self-tuning purposes. You can view it as a data warehouse for database statistics, metrics, and so on.

Every 60 minutes (by default) the database automatically captures statistical information from the SGA and stores it in the AWR in the form of snapshots. These snapshots are stored on the disk by a background process called Manageability Monitor (`MMON`). By default, snapshots are retained for eight days. You can modify both the snapshot interval and the retention intervals.

The AWR contains hundreds of tables, all belonging to the `SYSMAN` schema and stored in the `SYSAUX` tablespace. Oracle recommends that the repository be accessed only though Enterprise Manager or the `DBMS_WORKLOAD_REPOSITORY` package to work with the AWR. Direct DML against the repository tables is not supported.

AWR Infrastructure



AWR Infrastructure

The AWR infrastructure has two major parts:

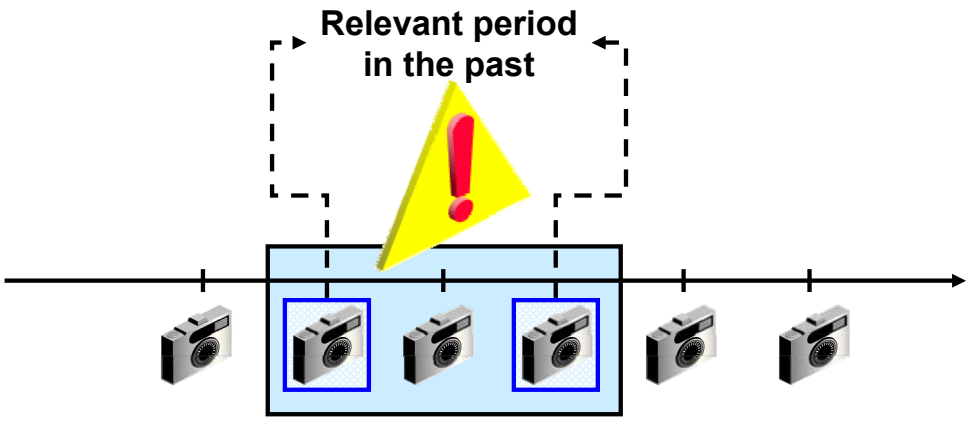
- An in-memory statistics collection facility that is used by Oracle Database 11*g* components to collect statistics. These statistics are stored in memory for performance reasons. Statistics stored in memory are accessible through dynamic performance (`V$`) views.
- The AWR snapshots that represent the persistent portion of the facility. AWR snapshots are accessible through data dictionary views and Enterprise Manager Database Control.

Statistics are stored in persistent storage for several reasons:

- The statistics need to survive instance crashes.
- Some analyses need historical data for baseline comparisons.
- A memory overflow can occur. When old statistics are replaced by new ones because of memory shortage, the replaced data can be stored for later use.

The memory version of the statistics is transferred to disk on a regular basis by the `MMON` background process. With the AWR, the Oracle database provides a way to capture historical statistics data automatically without DBA intervention.

AWR Baselines




```
DBMS_WORKLOAD_REPOSITORY.CREATE_BASELINE ( -
        start_snap_id IN NUMBER,
        end_snap_id   IN NUMBER,
        baseline_name IN VARCHAR2);
```


AWR Baselines

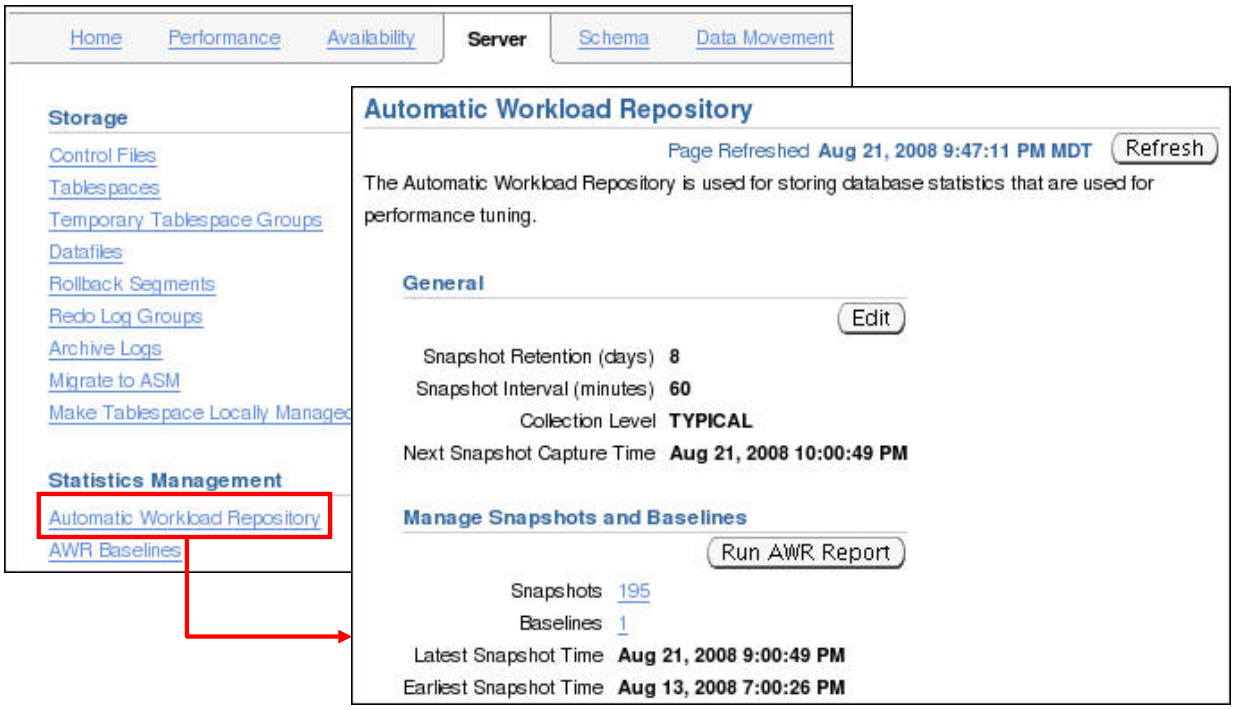
An AWR baseline is a set of AWR snapshots. This is usually a set of snapshot data for an important period that you tag and retain in the AWR. A baseline is defined on a pair of snapshots; the snapshots are identified by their snapshot sequence numbers (`snap_ids`) or a start and end time. Each snapshot set has starting and ending snapshots and includes all the snapshots in between. Snapshot sets are used to retain snapshot data. Therefore, by default, snapshots belonging to snapshot sets are retained until the snapshot sets are dropped. An expiration value can be set to a number of days that the snapshot will be retained.

A baseline is identified by a user-supplied name. Execute the `CREATE_BASELINE` procedure to create a baseline from a set of snapshots, and specify a name and a pair of snapshot identifiers. A baseline identifier that is unique for the life of a database is assigned to the newly created baseline. Usually you set up baselines from representative periods in the past, to be used for comparisons with current system behavior. You can also set up threshold-based alerts by using baselines from Database Control. You can set the expiration time in a number of days with the expiration parameter of this procedure. The default is `NULL`, meaning "never expire."

You can get the `snap_ids` directly from `DBA_HIST_SNAPSHOT`, or from Database Control.

Note: For more information about the `DBMS_WORKLOAD_REPOSITORY` package, see the *Oracle Database PL/SQL Packages and Types Reference* guide.

Enterprise Manager and the AWR



Enterprise Manager and the AWR

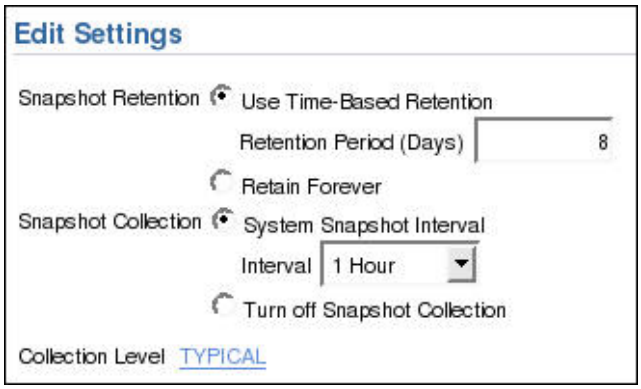
Click the Server tab, and then click Automatic Workload Repository in the Statistics Management section. On the Automatic Workload Repository page, click Edit to change the settings.

From the Automatic Workload Repository page, you can:

- Edit the workload repository settings
- Look at the detailed information about created snapshots and manually create new ones
- Create AWR baselines.
- Generate an AWR report

Managing the AWR

- Retention period
  - Default: Eight days
  - Consider storage needs
- Collection interval
  - Default: 60 minutes
  - Consider storage needs and performance impact
- Collection level
  - Basic (disables most ADDM functionality)
  - Typical (recommended)
  - All (adds additional SQL tuning information to snapshots)



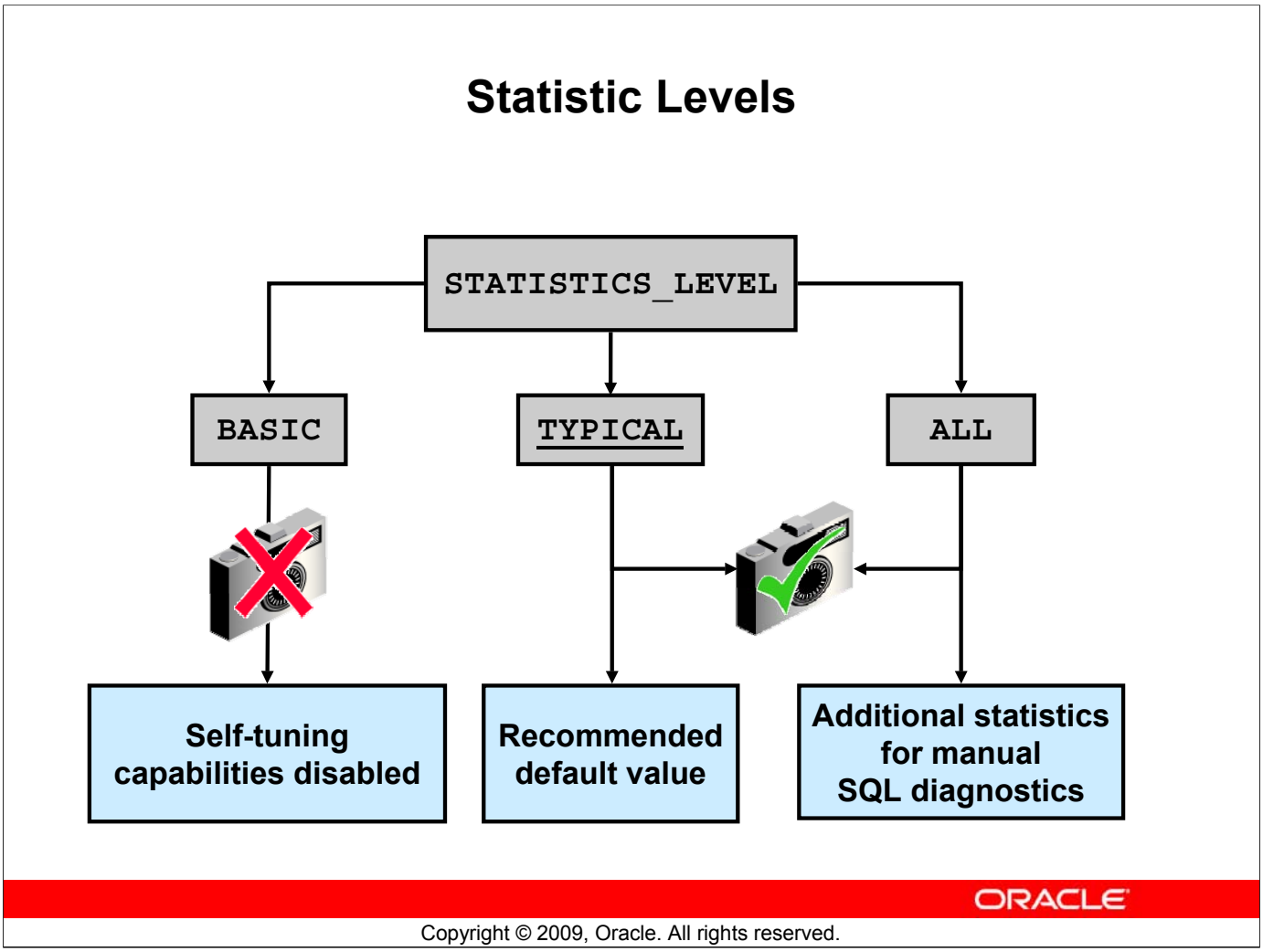
ORACLE

Managing the AWR

AWR settings include retention period, collection interval, and collection level. Remember that decreasing any of these settings affects the functionality of components that depend on the AWR, including the advisors.

Increasing the settings can provide improved advisor recommendations—but at the cost of the space that is required to store the snapshots and the performance expended in collecting the snapshot information.

Consider setting collection level to `ALL` when tuning a new application. The `ALL` setting collects SQL execution plans and timing statistics that enhance the recommendations of the SQL advisors. When tuning is complete, this setting should be returned to the `TYPICAL` setting.



Statistic Levels

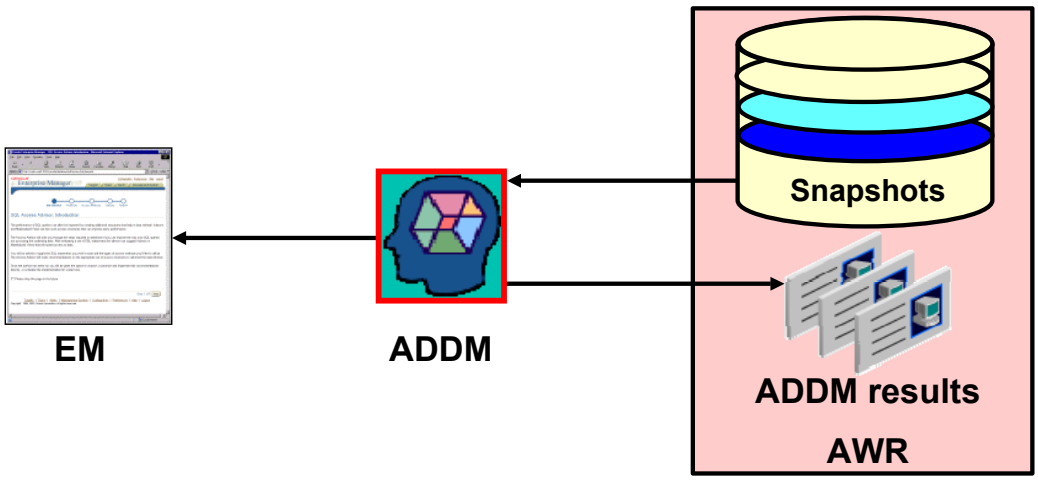
The `STATISTICS_LEVEL` initialization parameter controls the capture of a variety of statistics and various advisors, including the automatic maintenance tasks. The automatic maintenance tasks include gathering optimizer statistics. The `STATISTICS_LEVEL` parameter can be set to the following levels:

- **`BASIC`:** The computation of AWR statistics and metrics is turned off. The automatic optimizer statistics task is disabled, as are all advisors and server-generated alerts.
- **`TYPICAL`:** Major statistics are collected that are required for database self-management. They represent what is typically needed to monitor Oracle database behavior. This includes automatic gathering of statistics to reduce the likelihood of poorly performing SQL statements due to stale or invalid statistics.
- **`ALL`:** All possible statistics are captured. This level of capture adds timed OS statistics and plan execution statistics. These statistics are not needed in most cases and should not be enabled for best performance; they are sometimes needed for specific diagnostics tests.

Oracle recommends that the default value of `TYPICAL` be set for the `STATISTICS_LEVEL` initialization parameter. Setting the value to `BASIC` disables the automatic gathering of optimizer statistics.

Automatic Database Diagnostic Monitor (ADDM)

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



ORACLE

Automatic Database Diagnostic Monitor (ADDM)

Unlike the other advisors, the ADDM runs automatically after each AWR snapshot. Each time a snapshot is taken, the ADDM performs an analysis of the period corresponding to the last two snapshots. The ADDM proactively monitors the instance and detects most bottlenecks before they become a significant problem.

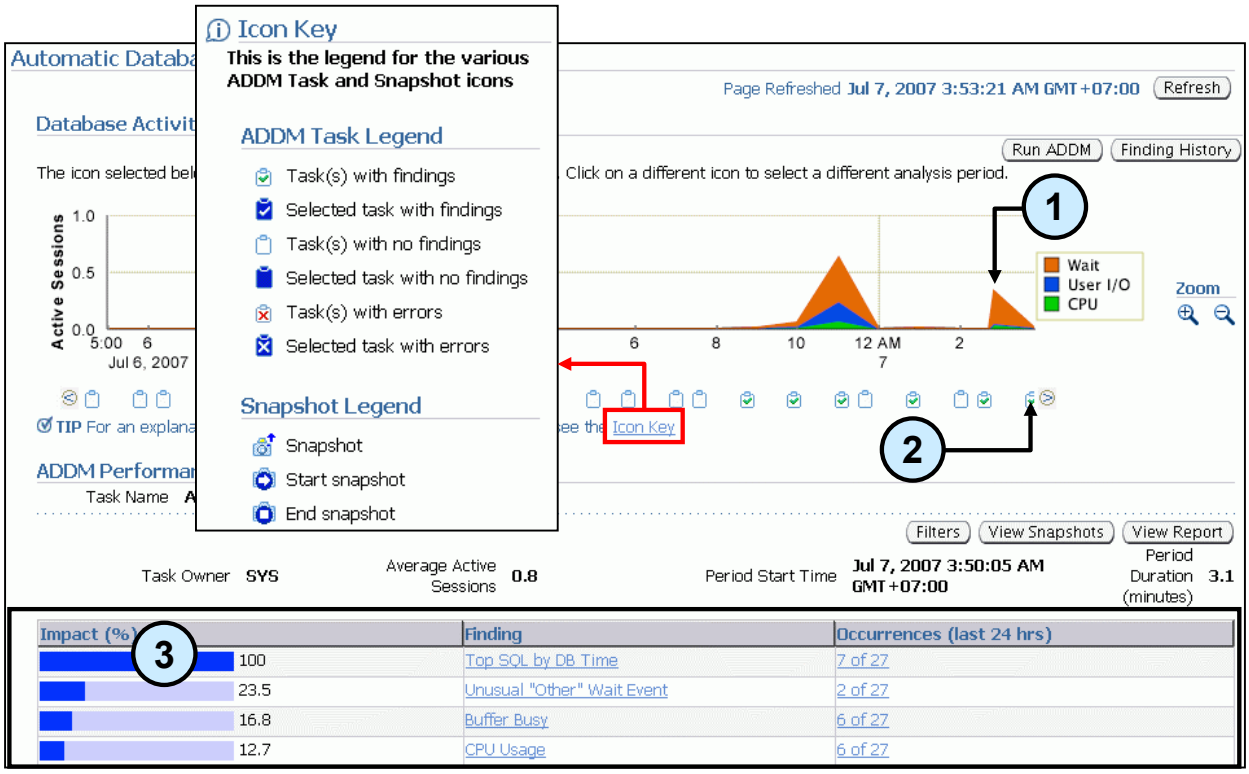
In many cases, the ADDM recommends solutions for detected problems and even quantifies the benefits for the recommendations.

Some common problems that are detected by the ADDM:

- CPU bottlenecks
- Poor Oracle Net connection management
- Lock contention
- Input/output (I/O) capacity
- Undersizing of database instance memory structures
- High-load SQL statements
- High PL/SQL and Java time
- High checkpoint load and cause (for example, small log files)

The results of each ADDM analysis are stored in the AWR and are also accessible through Enterprise Manager.

ADDM Findings



ADDM Findings

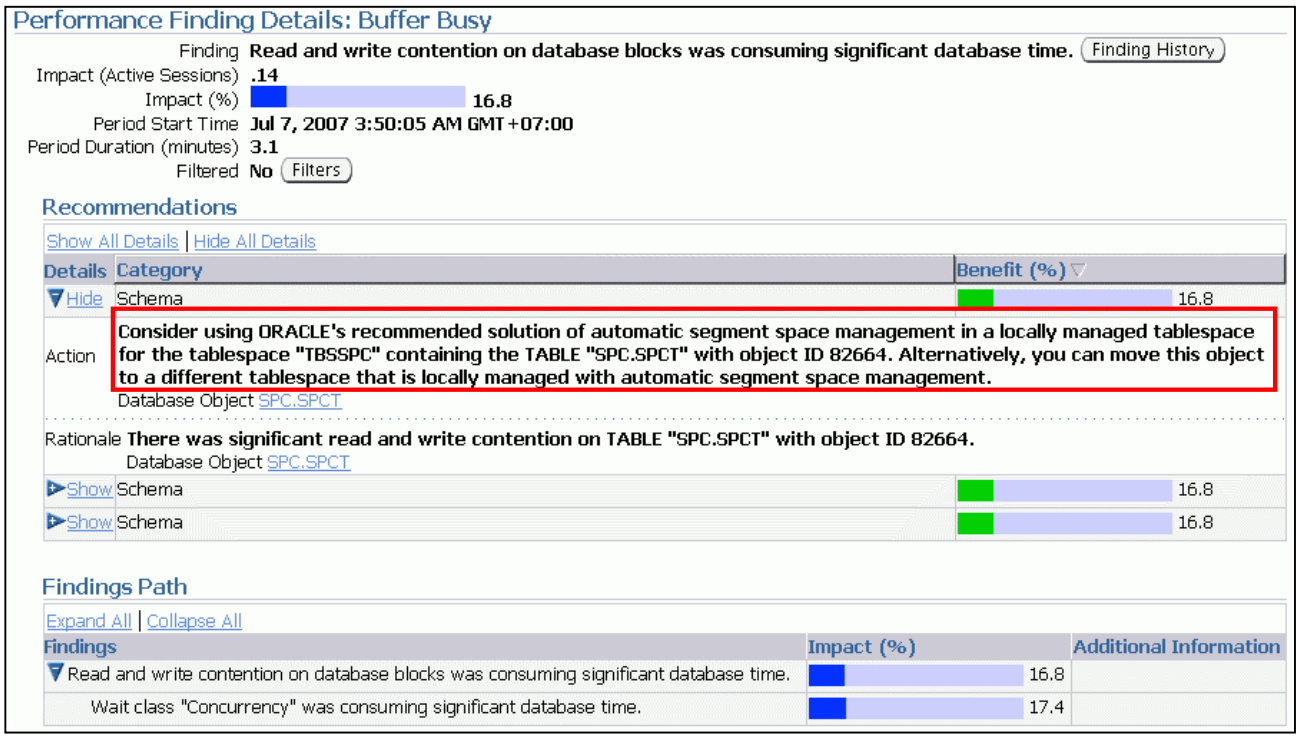
On the Automatic Database Diagnostic Monitor (ADDM) page, you see the detailed findings for the latest ADDM run. Database Time represents the sum of the nonidle time spent by sessions in the database for the analysis period. A specific impact percentage is given for each finding. The impact represents the time consumed by the corresponding issue compared with the database time for the analysis period.

In the slide, note the following:

1. The graphic shows that the number of average active users increased dramatically at this point. In addition, the major problem was a `Wait` problem.
2. The icon shows that the ADDM output displayed at the bottom of the page corresponds to this point in time. You can go into the past (to view previous analyses) by clicking the other icons.
3. The findings give you a short summary of what the ADDM found as tunable areas. By clicking a particular issue, you are directed to the Performance Finding Details page.

Click the View Report button to get details about the performance analysis in the form of a text report.

ADDM Recommendations

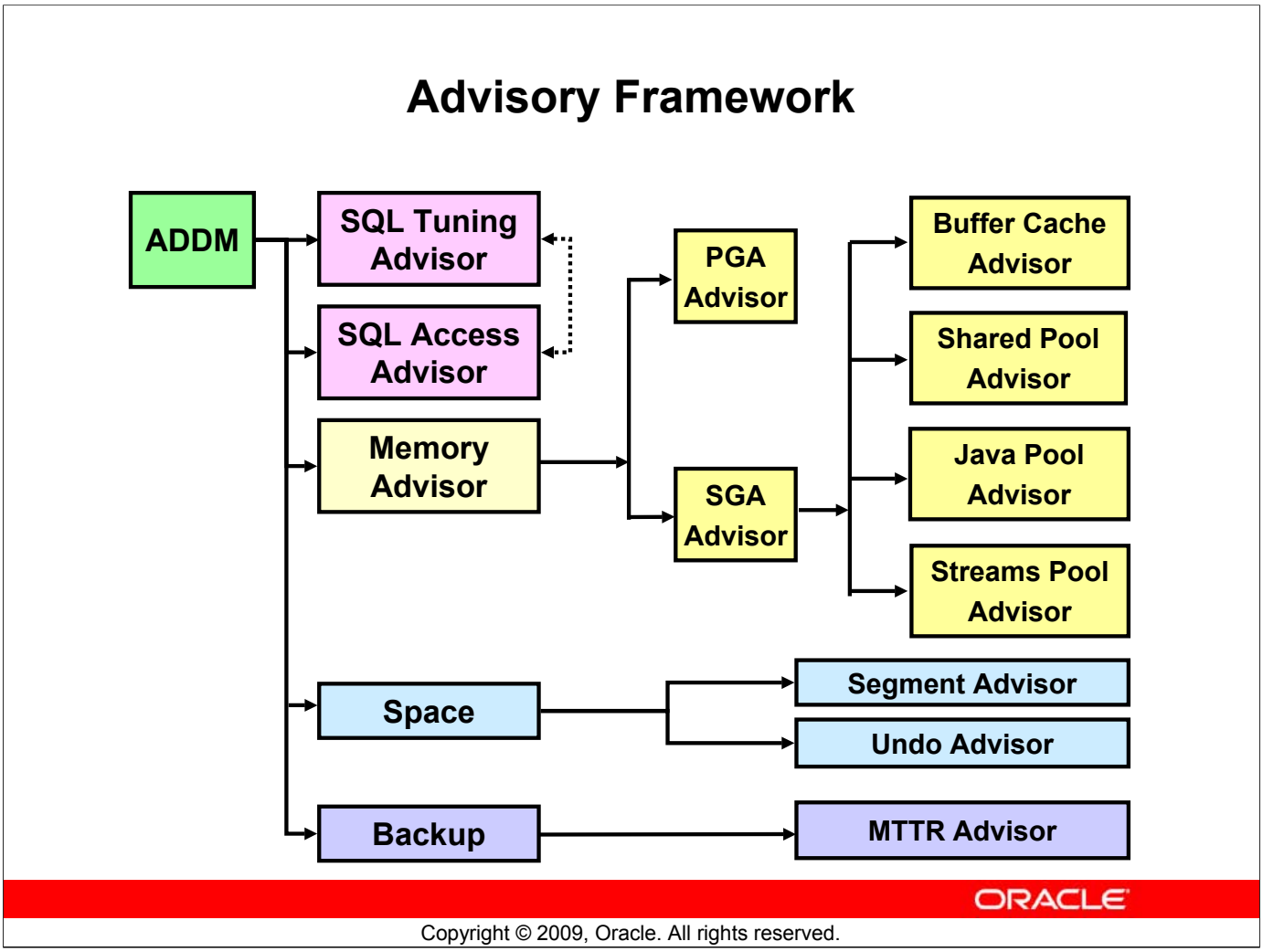


ADDM Recommendations

On the Performance Finding Details page, you are given recommendations for solving the corresponding issue. Recommendations are grouped into Schema, SQL Tuning, Database Configuration, and many other categories. The `Benefit (%)` column gives you the maximum reduction in database elapsed time if the recommendation is implemented.

The ADDM considers a variety of changes to a system. Its recommendations can include:

- **Hardware changes:** Adding CPUs or changing the I/O subsystem configuration
- **Database configuration:** Changing initialization parameter settings
- **Schema changes:** Hash-partitioning a table or index, or using Automatic Segment Space Management (ASSM)
- **Application changes:** Using the cache option for sequences, or using bind variables
- **Using other advisors:** Running the SQL Tuning Advisor on high-load SQL, or running the Segment Advisor on hot objects



Advisory Framework

Advisors provide you with useful feedback about resource utilization and performance for their respective server components. For example, the Memory Advisor provides a recommended value for the `MEMORY_TARGET` initialization parameter, which controls the total amount of memory used by the Oracle database instance.

By building on the data captured in the AWR, the ADDM enables the Oracle database to diagnose its own performance and determine how identified problems can be resolved. ADDM runs automatically after each AWR statistics capture. It can potentially call other advisors.

Here are the major benefits that are provided by the advisor infrastructure:

- All advisors use a uniform interface.
- All advisors have a common data source and results storage by using the workload repository.

Not all advisors are shown in the slide (for example, the Data Recovery Advisor and the SQL Repair Advisor are not listed).

Automatic Database Diagnostic Monitor (ADDM)

The ADDM is a server-based expert that reviews database performance every 60 minutes. Its goal is to detect possible system bottlenecks early and recommend fixes before system performance degrades noticeably.

Advisory Framework (continued)

Memory Advisors

The Memory Advisor is actually a collection of several advisory functions that help determine the best settings for the total memory used by the database instance. The System Global Area (SGA) has a set of advisors for the shared pool, database buffer cache, Java pool, and streams pool. The Java pool and streams pool advisors are not exposed on the EM Memory Advisor page. There is an advisor for the Program Global Area (PGA). In addition to the advisory functions, this advisor provides a central point of control for the large pool and the Java pool.

Mean-Time-To-Recover (MTTR) Advisor

Using the MTTR Advisor, you set the length of time required for the database to recover after an instance crash.

Segment Advisor

This advisor looks for tables and indexes that consume more space than they require. The advisor checks for inefficient space consumption at the tablespace or schema level and produces scripts to reduce space consumption where possible.

SQL Access Advisor

This advisor analyzes all SQL statements that are issued in a given period and suggests the creation of additional indexes or materialized views that will improve performance.

SQL Tuning Advisor

This advisor analyzes an individual SQL statement and makes recommendations for improving its performance. Recommendations may include actions such as rewriting the statement, changing the instance configuration, or adding indexes. The SQL Tuning Advisor is not invoked directly. Instead, it is called from within other tools (such as Top SQL or Top Sessions) to help optimize high-impact SQL statements.

Undo Management Advisor

With the Undo Management Advisor, you can determine the undo tablespace size that is required to support a given retention period. Undo management and the use of the advisor are covered in the lesson titled “Managing Undo Data.”

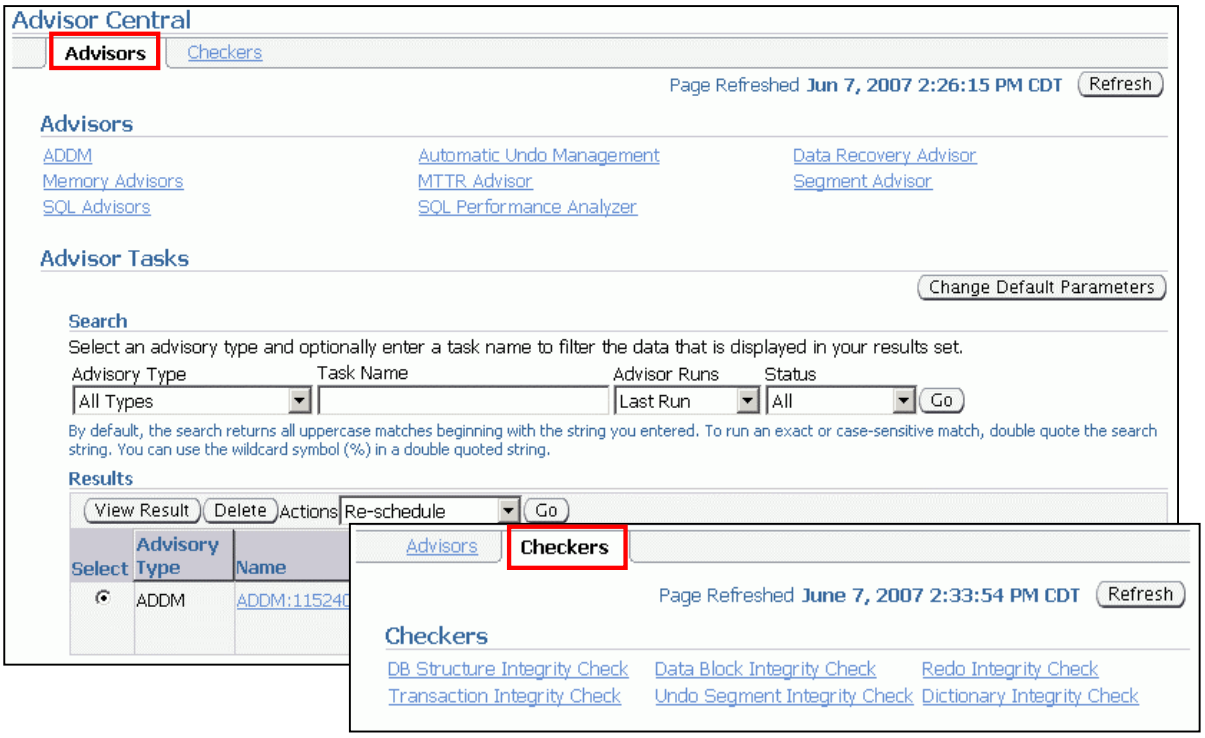
Data Recovery Advisor

This advisor automatically diagnoses persistent data failures, presents repair options to the user, and executes repairs at the user’s request. The purpose of the Data Recovery Advisor is to reduce the mean time to recover (MTTR) and provide a centralized tool for automated data repair.

SQL Repair Advisor

You run the SQL Repair Advisor after a SQL statement fails with a critical error that generates a problem in the Automatic Diagnostic Repository. The advisor analyzes the statement and, in many cases, recommends a patch to repair the statement. If you implement the recommendation, the applied SQL patch circumvents the failure by causing the query optimizer to choose an alternative execution plan for future executions. This is done without changing the SQL statement itself.

Enterprise Manager and Advisors



Enterprise Manager and Advisors

The Advisor Central page is the main page of all advisors. You can reach this page by clicking the Advisor Central link in the list of Related Links on the Database Control Home page. However, this is not the only place in Database Control where advisors can be invoked. It is also possible to have access to advisors in certain contexts.

On the Advisors tab of the Advisor Central page, you can list all the advisor tasks that are registered in the workload repository. You can also filter this list by advisor type and for predefined time periods.

The Checkers tab of the Advisor Central page enables you to schedule various database integrity checkers. You can list all the checker runs by name, type, or time period.

Some advisors are described in greater detail in the lessons titled “Managing Undo Data,” “Performance Management,” and “Backup and Recovery Concepts.”

Note: Use the Change Default Parameters page to change the default expiration (in days) for all future tasks. You can also use this page to change the parameters of some important advisors.

DBMS_ADVISOR Package

Procedure	Description
CREATE_TASK	Creates a new task in the repository
DELETE_TASK	Deletes a task from the repository
EXECUTE_TASK	Initiates execution of the task
INTERRUPT_TASK	Suspends a task that is currently executing
GET_TASK_REPORT	Creates and returns a text report for the specified task
RESUME_TASK	Causes a suspended task to resume
UPDATE_TASK_ATTRIBUTES	Updates task attributes
SET_TASK_PARAMETER	Modifies a task parameter
MARK_RECOMMENDATION	Marks one or more recommendations as accepted, rejected, or ignored
GET_TASK_SCRIPT	Creates a script of all the recommendations that are accepted

ORACLE

DBMS_ADVISOR Package

The DBMS_ADVISOR package contains all constants and procedure declarations for all advisor modules. You can use this package to execute tasks from the command line.

To execute advisor procedures, you must be granted the ADVISOR privilege. The ADVISOR privilege permits full access to the advisor procedures and views.

Note: For more information about all the procedures found in the DBMS_ADVISOR package, see the *Oracle Database PL/SQL Packages and Types Reference*.

Quiz

The optimizer statistic `num_rows` will always reflect the true row count for a table.

1. True
2. False

Answer: 2

Automated Maintenance Tasks

Autotask maintenance process:

1. Maintenance Window opens.
2. Autotask background process schedules jobs.
3. Scheduler initiates jobs.
4. Resource Manager limits impact of Autotask jobs.

Default Autotask maintenance jobs:

- Gathering optimizer statistics
- Automatic Segment Advisor
- Automatic SQL Advisor

ORACLE

Automated Maintenance Tasks

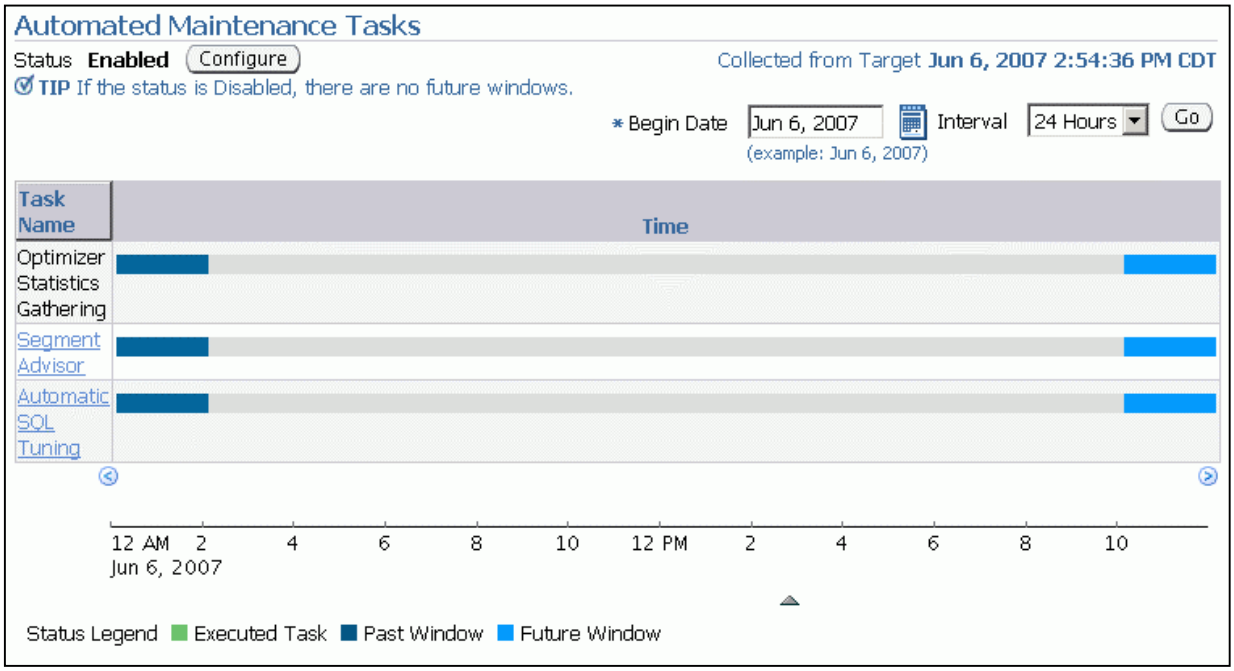
By analyzing the information stored in the AWR, the database can identify the need to perform routine maintenance tasks, such as optimizer statistics refresh. The automated maintenance tasks infrastructure enables the Oracle database to automatically perform such operations. It uses the Scheduler to run such tasks in predefined maintenance windows.

By default, the weekday maintenance windows start at 10:00 PM and lasts 4 hours. On Saturday and Sunday, the maintenance window starts at 6:00 AM and lasts for 20 hours. All attributes of the maintenance windows are customizable, including the start and end time, frequency, days of the week, and so on. In addition, the impact of automated maintenance tasks on normal database operations can be limited by associating a Database Resource Manager resource plan to the maintenance window.

Examples of maintenance:

- Optimizer statistics are automatically refreshed by using the automatic maintenance task infrastructure.
- The Automatic Segment Advisor has default jobs, which run in the maintenance window.
- When creating a database with the DBCA, you can initiate regular database backups.

Automated Maintenance Tasks



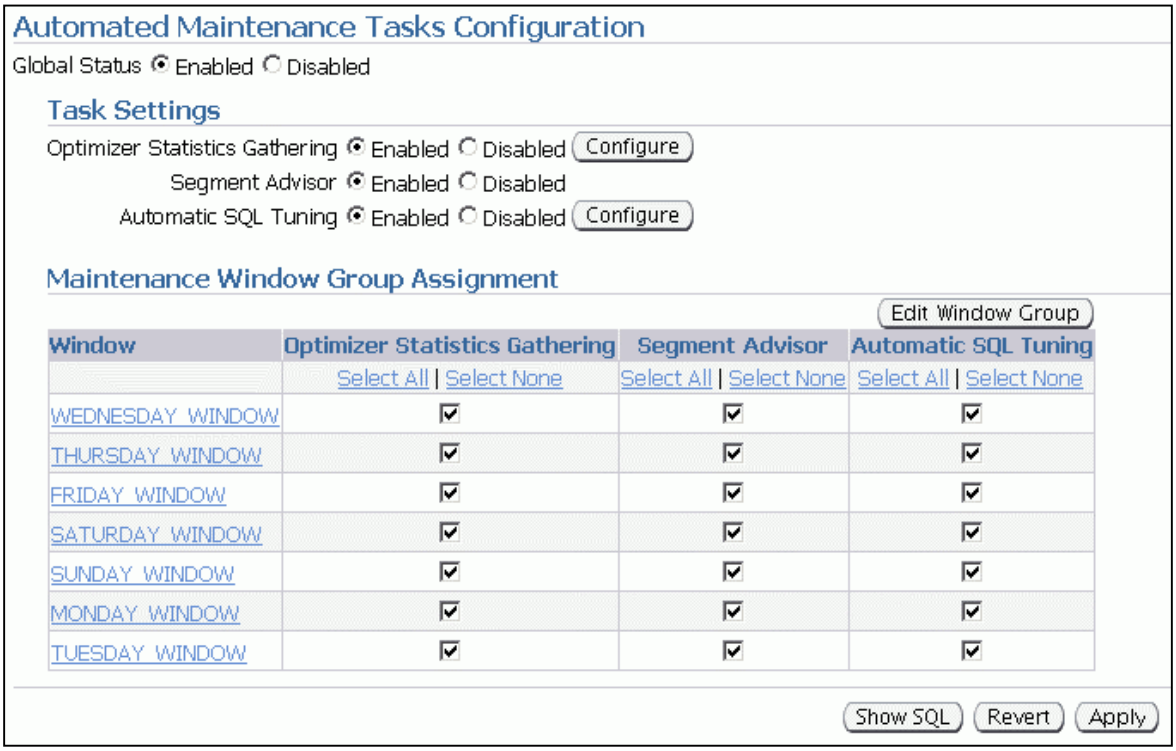
ORACLE

Automated Maintenance Tasks (continued)

Click Automated Maintenance Tasks under the Scheduler heading on the Server page to access the Automated Maintenance Task page, where you can view the automated maintenance task schedule and recent history. From here you can drill down to details on some tasks. Click Configure to go to the Automated Maintenance Tasks Configuration page. A task executes in a window. The graph shows the last window in which a task was executed and the next window in which the task is scheduled to be executed.

Note: The default windows for tasks are shown in the example. When the maintenance window closes, the Scheduler terminates the optimizer statistics–gathering job by default. The remaining objects are then processed in the next maintenance window.

Automated Maintenance Tasks Configuration



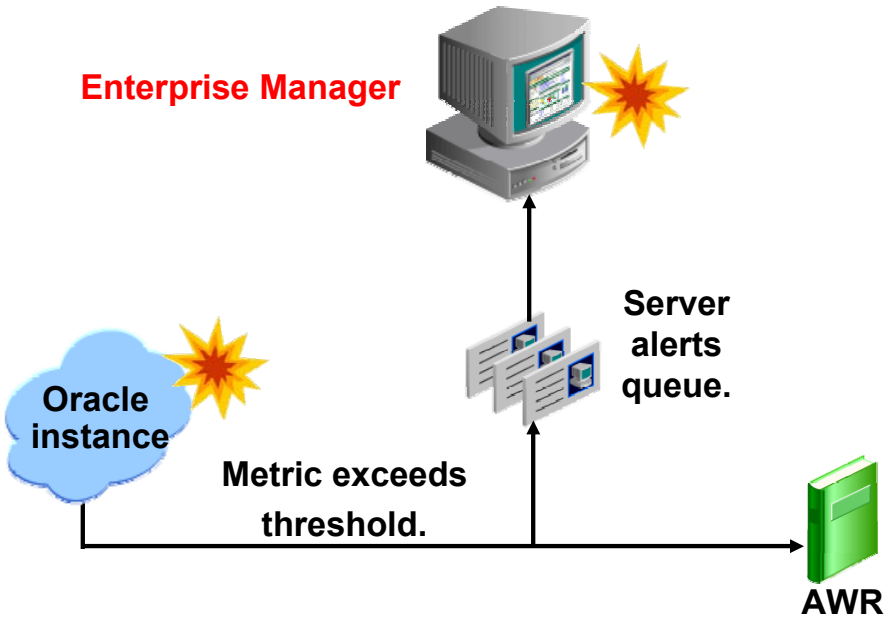
Automated Maintenance Tasks Configuration

On the Automated Maintenance Tasks Configuration page, you can enable and disable automatic maintenance tasks—all at once, by individual tasks, or by particular windows. You can also configure the settings that are used for optimizer statistics gathering and the job control parameters for the automatic SQL Tuning Advisor.

Select the window name to view or edit the window schedule.

Click Edit Window Group to add and remove windows in the window group.

Server-Generated Alerts



Server-Generated Alerts

Alerts are notifications of when a database is in an undesirable state and needs your attention. By default, the Oracle database provides alerts via Enterprise Manager Database Control. Optionally, Enterprise Manager can be configured to send an email message to the administrator about problem conditions as well as display alert information on the console.

You can also set thresholds on many of the pertinent metrics for your system. Oracle Database 11*g* proactively notifies you if the database deviates sufficiently from normal readings to reach those thresholds. An early notification of potential problems enables you to respond quickly and, in many cases, resolve issues before users even notice them.

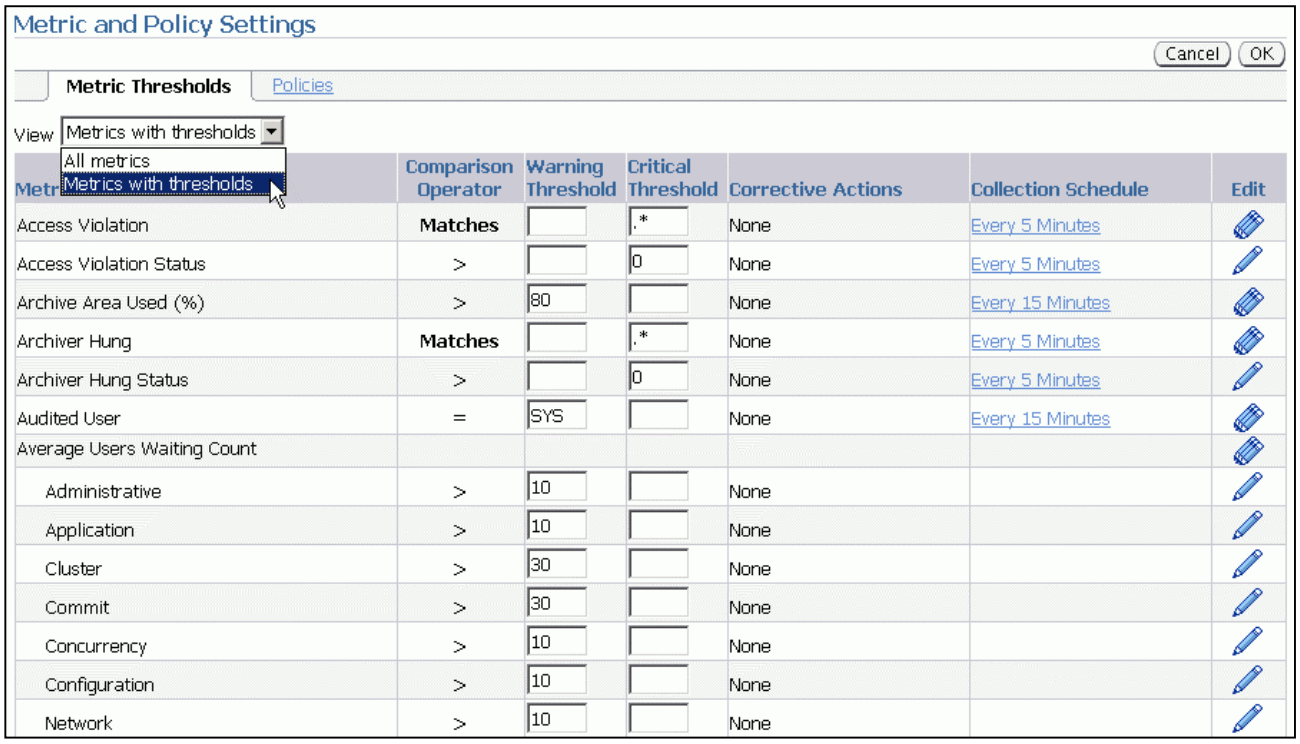
Approximately 60 metrics are monitored by default, among which are:

- Broken Job Count
- Database Time Spent Waiting (%)
- Dump Area Used (%)
- SQL Response Time (%) compared to baseline
- Tablespace Used (%)
- Generic Incident

A few additional key metrics can provide early problem notification:

- Average File Read Time (centiseconds)
- Response Time (per transaction)
- Wait Time (%)

Setting Thresholds



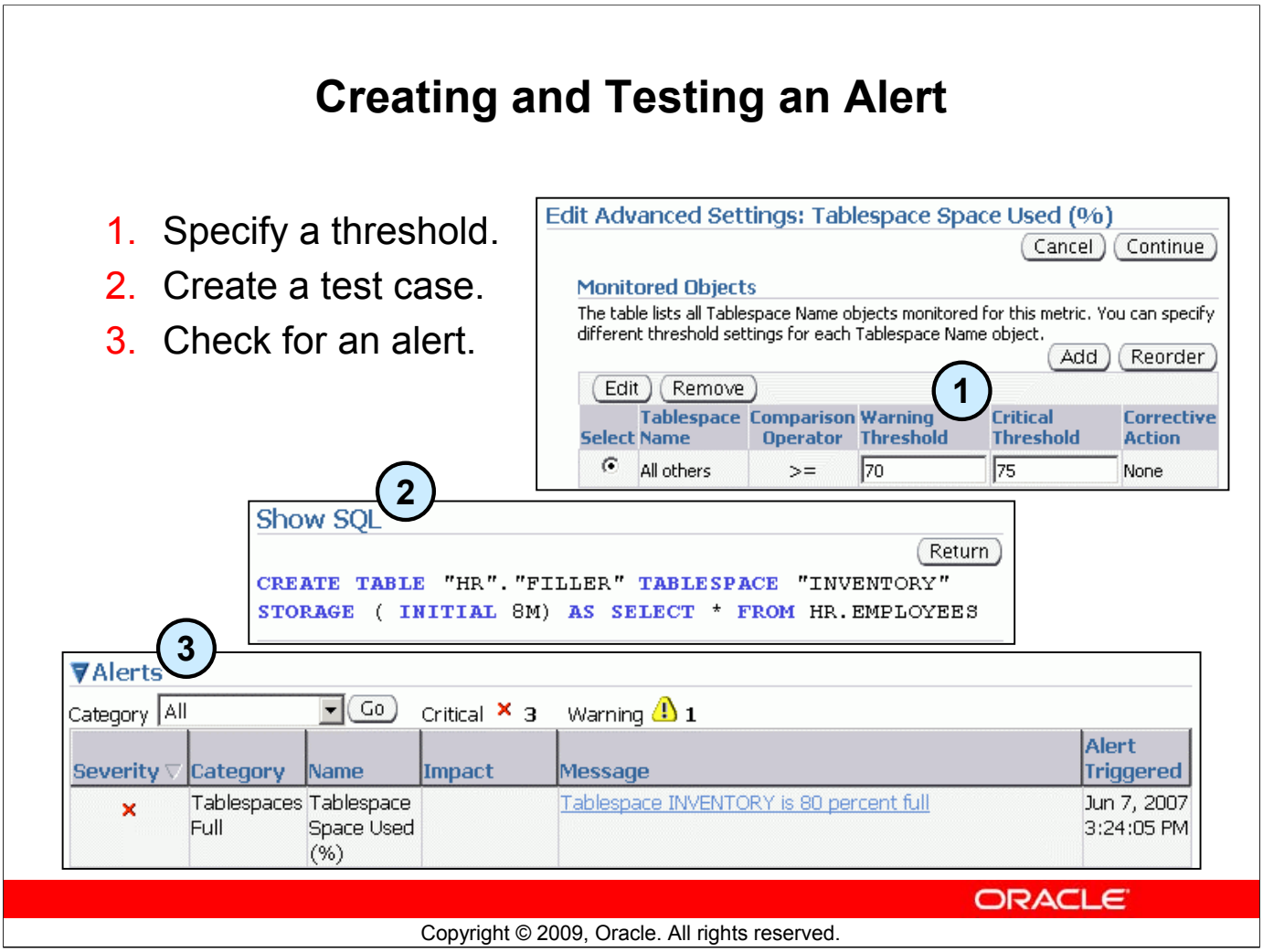
Setting Thresholds

To set or edit a threshold for your whole database, click “Metric and Policy Settings” in the Related Links region of the database home page. Enter your desired warning and critical threshold values. The appropriate alerts appear when the database reaches your specified values.

The thresholds that are already set appear in the “Metrics with thresholds” list. By default, approximately 60 metrics have preset thresholds; you may change these as needed. The “All metrics” list shows the metrics that do not have thresholds set.

Click one of the Edit icons to access a page where you can specify additional corrective actions for either warning or critical thresholds.

Click a Collection Schedule link and change the scheduled collection interval. Be aware that each schedule affects a group of metrics.



Creating and Testing an Alert

You can also set thresholds for a specific object.

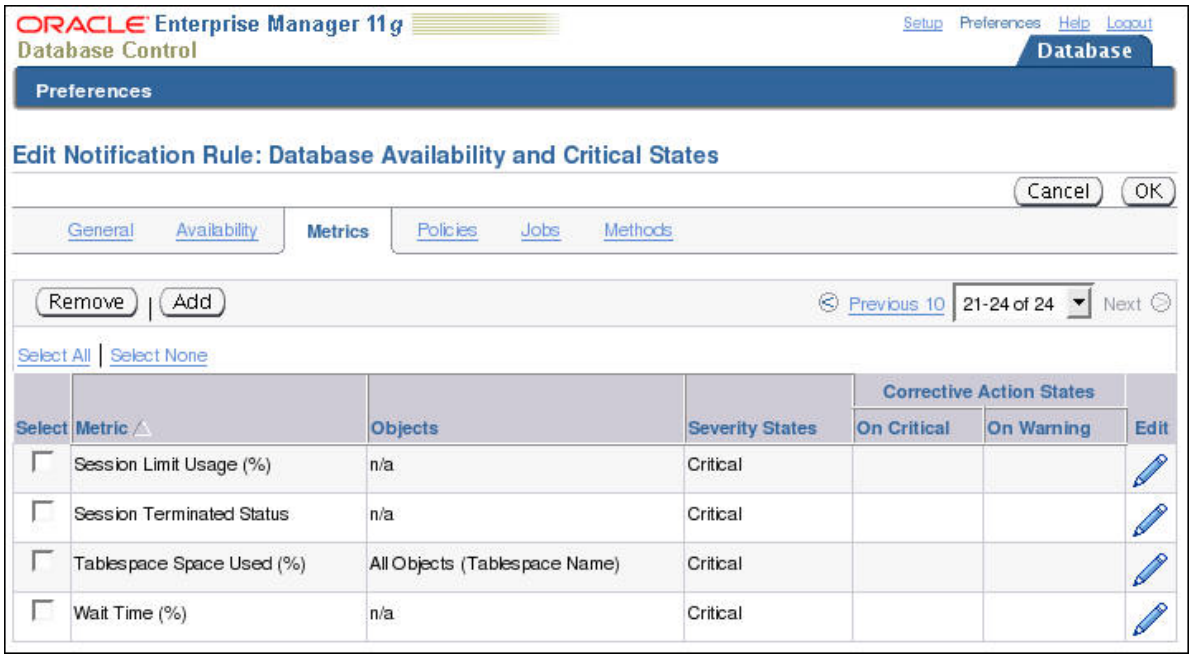
Example

You decide that you need to receive a critical alert if the space used in the `INVENTORY` tablespace exceeds 75%. (This tablespace does not allow its data files to automatically extend.) To create and test the alert, perform the following steps:

1. In Enterprise Manager, navigate to the "Metrics and Policy Settings" page, and then click the Edit icon for the Tablespace Used (%) threshold. Set your desired threshold for the tablespace.
2. Under the Schema tab on the Tables page, create a table to test the alert. Use the "Define using SQL" action to duplicate an existing table. The initial setting of 8 MB in the `STORAGE` clause causes the table to allocate 80% of the 10 MB `INVENTORY` tablespace immediately.
3. After you receive an error that this table is unable to extend, check the Database Home page for the associated alert. Tablespace Space Used (%) is collected every 10 minutes by default.

Most alerts contain the name of an associated advisor that can be invoked to give you more detailed advice. For each corresponding alert message, Database Control provides a link to invoke the corresponding advisor.

Alerts Notification



ORACLE

Alerts Notification

The notification mechanism uses the Enterprise Manager user interface. It is based on the concept of a notification rule that establishes the appropriate notification mechanism for a set of upcoming alerts.

Using Database Control, you edit the notification rules. On the home page, click the Preferences link to display the General page, where you specify the email address at which you want to receive notifications.

On the General page, click the Rules link in the Notification region. Select the “Database Availability and Critical States” rule, and then click the Edit button. This takes you to the “Edit Notification Rule Database Availability and Critical States” page, where you click the Metrics tab and edit the metrics for which you want to receive notifications.

Alerts Notification (continued)

As an option, you can specify that Enterprise Manager provide you with direct notification when specific alerts arise. For example, if you specify that you want email notification for critical alerts, and you have a critical threshold set for the system response time for each call metric, you can send an email message containing a message similar to the following:

```
Host Name=mydb.us.mycompany.com
Metric=Response Time per Call
Timestamp=08-NOV-2005 10:10:01 (GMT -7:00)
Severity=Critical
Message=Response time per call has exceeded the threshold.
   See the latest ADDM analysis.
Rule Name= Rule
Owner=SYSMAN
```

The email contains a link to the host name and the latest ADDM analysis.

By default, alerts in critical state (such as DB Down, Generic Alert Log Error Status, and Tablespace Used) are set up for notification. However, to receive these notifications, you must set up your email information by following these steps:

1. On any Database Control page, click the Setup link in the header and footer area.
2. On the Setup page, select Notification Methods.
3. Enter the required information in the Mail Server region of the Notifications Methods page.

There are other methods of notification, including scripts and Simplified Network Management Protocol (SNMP) traps. The latter can be used to communicate with third-party applications.

To receive notifications:

1. On any Database Control page, click the Preferences link in the header and footer area.
2. On the Preferences page, select General. Enter your email address in the E-mail Addresses region.
3. You can optionally edit notification rules (for example, to change the severity state for receiving notification). To do so, click Notification Rules. The Notification Rules page appears.
   Note: For more information about configuring notification rules, see the *Oracle Enterprise Manager Advanced Configuration* documentation.

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.

ORACLE

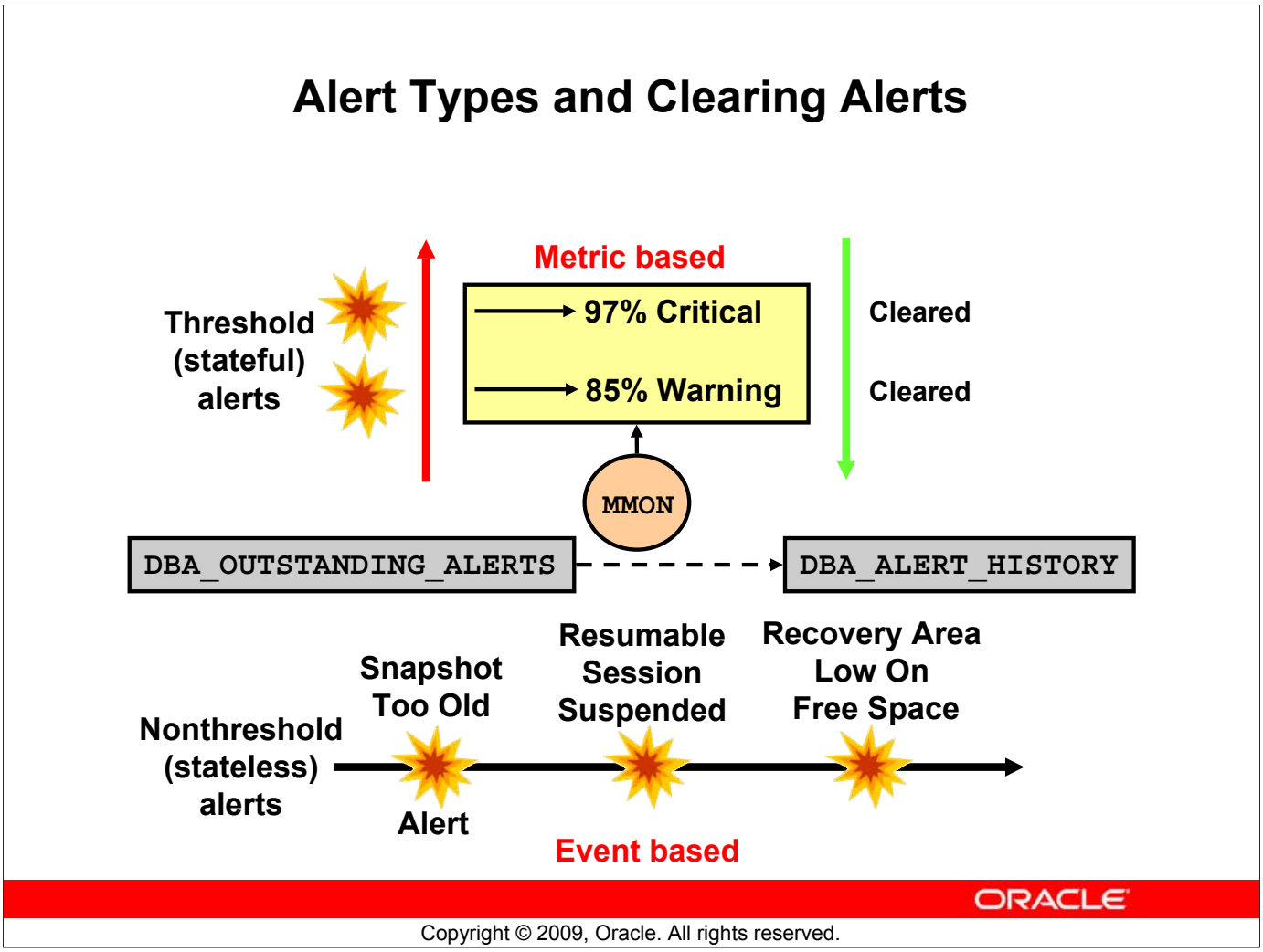
Reacting to Alerts

When you receive an alert, follow the recommendations that it provides. Or you can consider running the ADDM (or another advisor as appropriate) to obtain more detailed diagnostics of system or object behavior.

Alerts and incidents are generated for critical errors. Critical errors usually generate incidents that are collected into problems. You use the Support Workbench to investigate and possibly report the problem to Oracle Support.

Most alerts (such as “Out of Space”) are cleared automatically when the cause of the problem disappears. However, other alerts (such as Generic Alert Log Error) are sent to you for notification and must be acknowledged by you. After taking the necessary corrective measures, you acknowledge an alert by clearing or purging it. Clearing an alert sends the alert to the Alert History, which is viewable from the home page under Related Links. Purging an alert removes it from the Alert History.

To clear an alert such as Generic Alert Log Error, click the Alert Log link on the home page under Diagnostic Summary. The Alert Log Errors page appears. Select the alert to clear, and then click Clear. To purge an alert, select it and click Purge. You can also click the Clear Every Open Alert button or the Purge Every Alert button.



Alert Types and Clearing Alerts

There are two kinds of server-generated alerts: threshold and nonthreshold.

Most server-generated alerts are configured by setting a warning and critical threshold values on database metrics. You can define thresholds for more than 120 metrics, including the following:

- Physical Reads Per Sec
- User Commits Per Sec
- SQL Service Response Time

Except for the Tablespace Space Usage metric, which is database related, the other metrics are instance related. Threshold alerts are also referred to as *stateful alerts*, which are automatically cleared when an alert condition clears. Stateful alerts appear in `DBA_OUTSTANDING_ALERTS` and, when cleared, go to `DBA_ALERT_HISTORY`.

Other server-generated alerts correspond to specific database events such as `ORA-*` errors, "Snapshot too old" errors, Recovery Area Low On Free Space, and Resumable Session Suspended. These are non-threshold-based alerts, also referred to as *stateless alerts*. Stateless alerts go directly to the history table. Clearing a stateless alert makes sense only in the Database Control environment because Database Control stores stateless alerts in its own repository.

Quiz

Stateless alerts such as `SNAPSHOT TOO OLD` can be found in the dictionary view `DBA_OUTSTANDING_ALERTS`.

1. True
2. False

Answer: 2

Summary

In this lesson, you should have learned how to:

- Manage optimizer statistics
- Manage the Automatic Workload Repository (AWR)
- Use the Automatic Database Diagnostic Monitor (ADDM)
- Describe and use the advisory framework
- Set alert thresholds
- Use server-generated alerts
- Use automated tasks

ORACLE

Practice 12 Overview: Proactive Maintenance

This practice covers proactively managing your database with ADDM, including:

- Setting up an issue for analysis
- Reviewing your database performance
- Implementing a solution

ORACLE

13

Performance Management

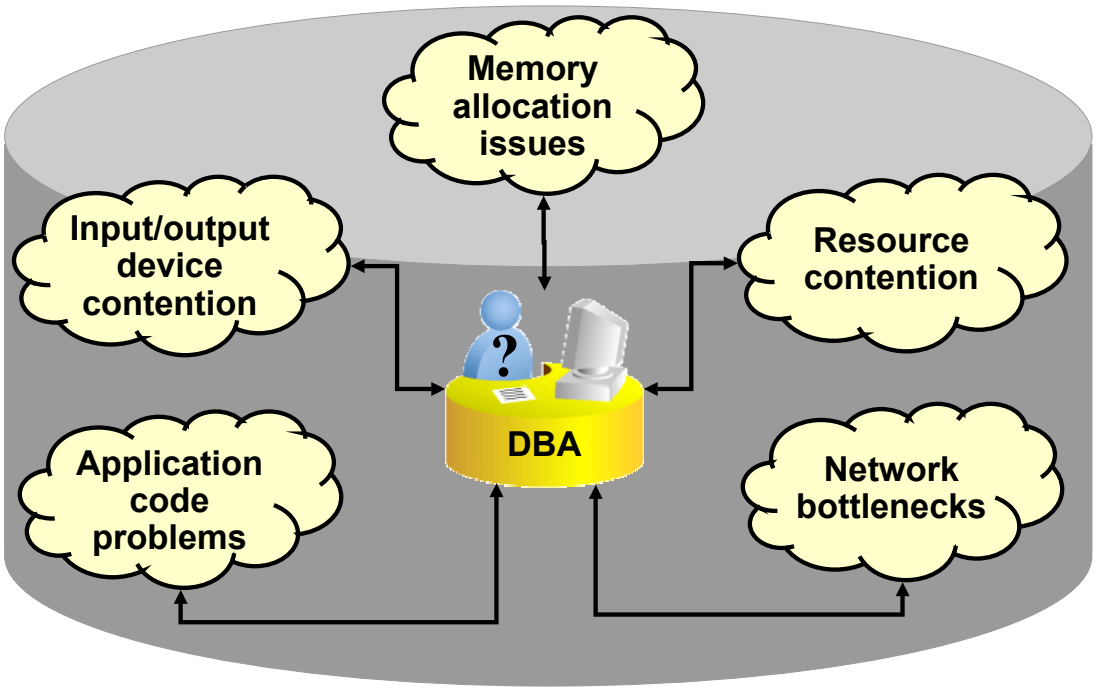
Objectives

After completing this lesson, you should be able to:

- Use Enterprise Manager to monitor performance
- Use Automatic Memory Management (AMM)
- Use the Memory Advisor to size memory buffers
- View performance-related dynamic views
- Troubleshoot invalid and unusable objects

ORACLE

Performance Monitoring



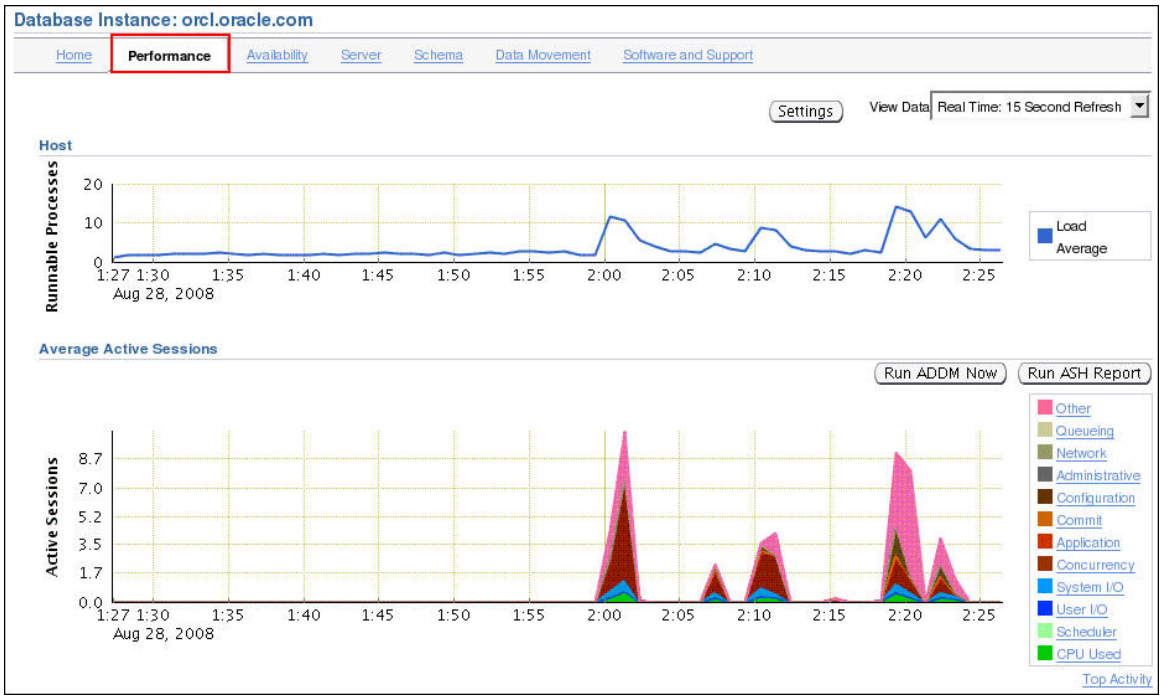
Performance Monitoring

To administer Oracle Database 11*g* and keep it running smoothly, the database administrator (DBA) must regularly monitor its performance to locate bottlenecks and correct problem areas.

A DBA can look at hundreds of performance measurements, covering everything from network performance and disk input/output (I/O) speed to the time spent working on individual application operations. These performance measurements are commonly referred to as *database metrics*.

Note: For more information about Oracle database performance, see the *Oracle Database 11g: Performance Tuning* course.

Enterprise Manager Performance Page

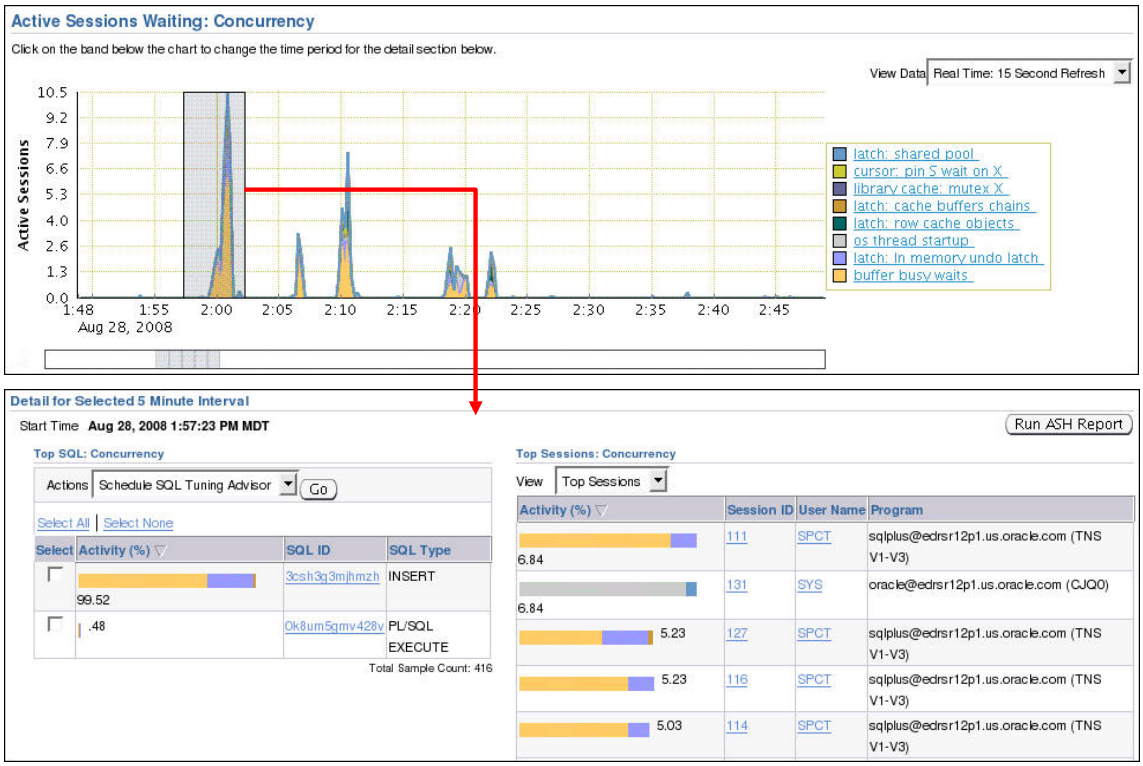


Enterprise Manager Performance Page

The Performance page in Enterprise Manager is the portal to a powerful set of performance monitoring and tuning tools. The first set of graphs on this page summarizes processes and active session activity. The Average Active Sessions graph shows the level of CPU usage and the resources that are causing the most wait events.

In the slide, you see that there was a recent increase in Concurrency and Other waits. During each of these spikes there was also a slight increase in System I/O and CPU usage. You can click these categories to see more details about the waits. The I/O data is divided into types of I/O (for example, log file read, control file write, and so on).

Drilling Down to a Particular Wait Category

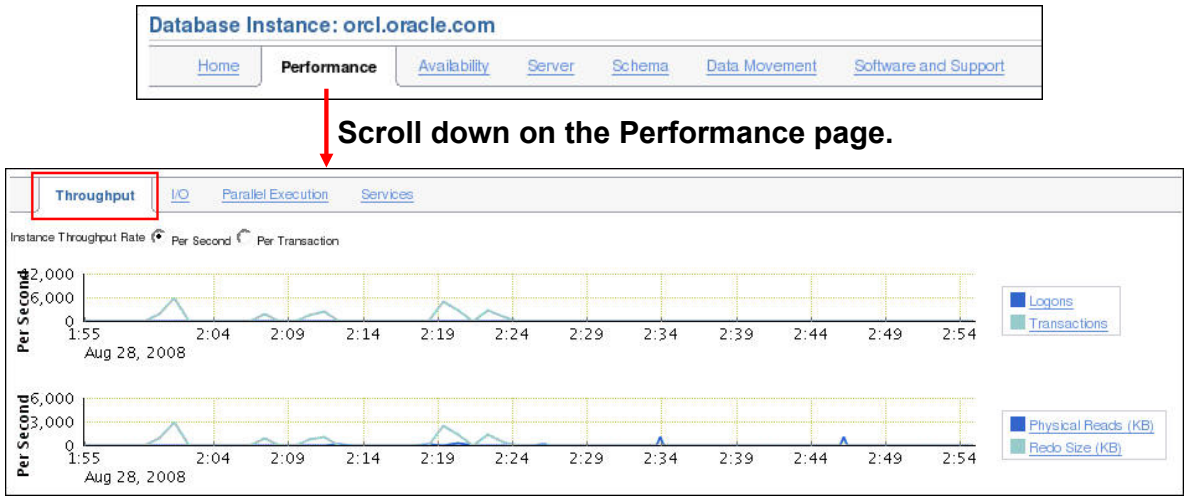


Drilling Down to a Particular Wait Category

When you drill down to a particular wait category, you can view details of specific five-minute intervals and also see the Top Working SQL and the Top Working Sessions associated with that particular wait event during that time. This enables you to perform after-the-fact analysis of system slowdowns and determine potential causes.

The example in the slide shows the results of drilling down on the Concurrency category from the Active Session graph in the previous slide.

Performance Page: Throughput

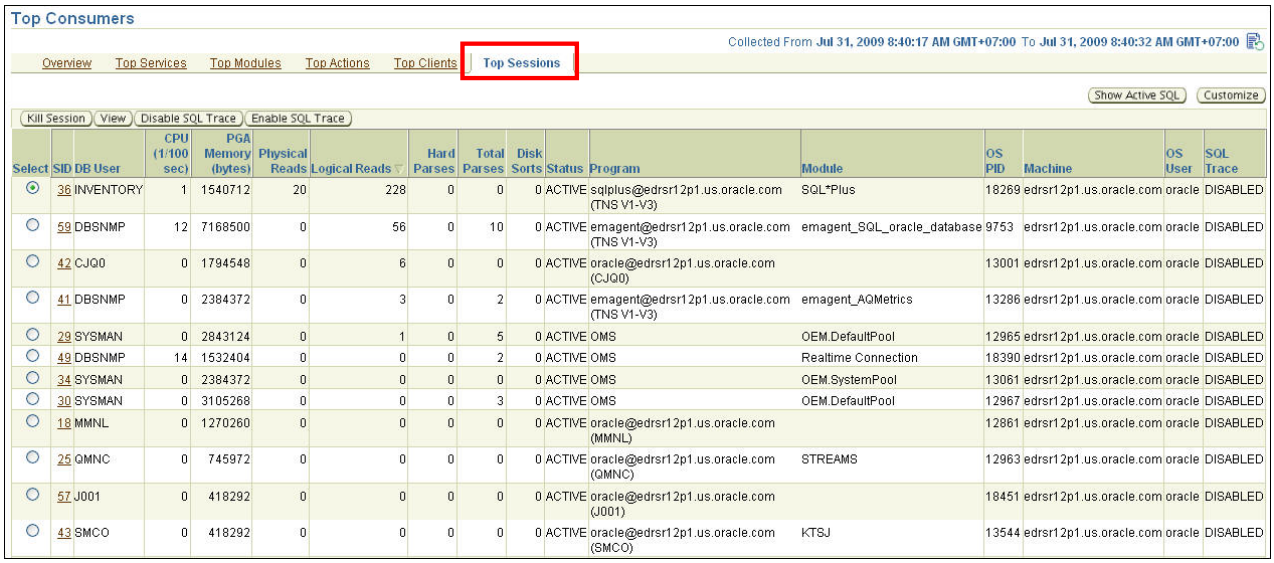


ORACLE

Performance Page: Throughput

You can see graphs of Instance throughput and Instance Disk I/O by clicking the Throughput and I/O tabs on the main Performance page. The Throughput tab has been selected in the slide.

Performance Monitoring: Top Sessions



Select	SID	DB User	CPU (1/100 sec)	PGA Memory (bytes)	Physical Reads	Logical Reads	Hard Parses	Total Parses	Disk Sorts	Status	Program	Module	OS PID	Machine	OS User	SQL Trace
◉	36	INVENTORY	1	1540712	20	228	0	0	0	ACTIVE	sqlplus@edrsr12p1.us.oracle.com (TNS V1-V3)	SQL*Plus	18269	edrsr12p1.us.oracle.com	oracle	DISABLED
○	59	DBSNMP	12	7168500	0	56	0	10	0	ACTIVE	emagent@edrsr12p1.us.oracle.com (TNS V1-V3)	emagent_SQL_oracle_database	9753	edrsr12p1.us.oracle.com	oracle	DISABLED
○	42	CJQ0	0	1794548	0	6	0	0	0	ACTIVE	oracle@edrsr12p1.us.oracle.com (CJQ0)		13001	edrsr12p1.us.oracle.com	oracle	DISABLED
○	41	DBSNMP	0	2384372	0	3	0	2	0	ACTIVE	emagent@edrsr12p1.us.oracle.com (TNS V1-V3)	emagent_AQMetrics	13286	edrsr12p1.us.oracle.com	oracle	DISABLED
○	29	SYSMAN	0	2843124	0	1	0	5	0	ACTIVE	OMS	OEM.DefaultPool	12965	edrsr12p1.us.oracle.com	oracle	DISABLED
○	49	DBSNMP	14	1532404	0	0	0	2	0	ACTIVE	OMS	Realtime Connection	18390	edrsr12p1.us.oracle.com	oracle	DISABLED
○	34	SYSMAN	0	2384372	0	0	0	0	0	ACTIVE	OMS	OEM.SystemPool	13061	edrsr12p1.us.oracle.com	oracle	DISABLED
○	30	SYSMAN	0	3105268	0	0	0	3	0	ACTIVE	OMS	OEM.DefaultPool	12967	edrsr12p1.us.oracle.com	oracle	DISABLED
○	18	MMNL	0	1270260	0	0	0	0	0	ACTIVE	oracle@edrsr12p1.us.oracle.com (MMNL)		12861	edrsr12p1.us.oracle.com	oracle	DISABLED
○	25	QMNC	0	745972	0	0	0	0	0	ACTIVE	oracle@edrsr12p1.us.oracle.com (QMNC)	STREAMS	12963	edrsr12p1.us.oracle.com	oracle	DISABLED
○	57	J001	0	418292	0	0	0	0	0	ACTIVE	oracle@edrsr12p1.us.oracle.com (J001)		18451	edrsr12p1.us.oracle.com	oracle	DISABLED
○	43	SMCO	0	418292	0	0	0	0	0	ACTIVE	oracle@edrsr12p1.us.oracle.com (SMCO)	KTSJ	13544	edrsr12p1.us.oracle.com	oracle	DISABLED

Performance Monitoring: Top Sessions

Click Top Consumers in the Additional Monitoring Links section will take you to the Top Consumers page.

The Top Consumers Overview page shows in graphical format:

- Top services
- Top modules (by Service)
- Top Actions (by Service and Module)
- Top Clients

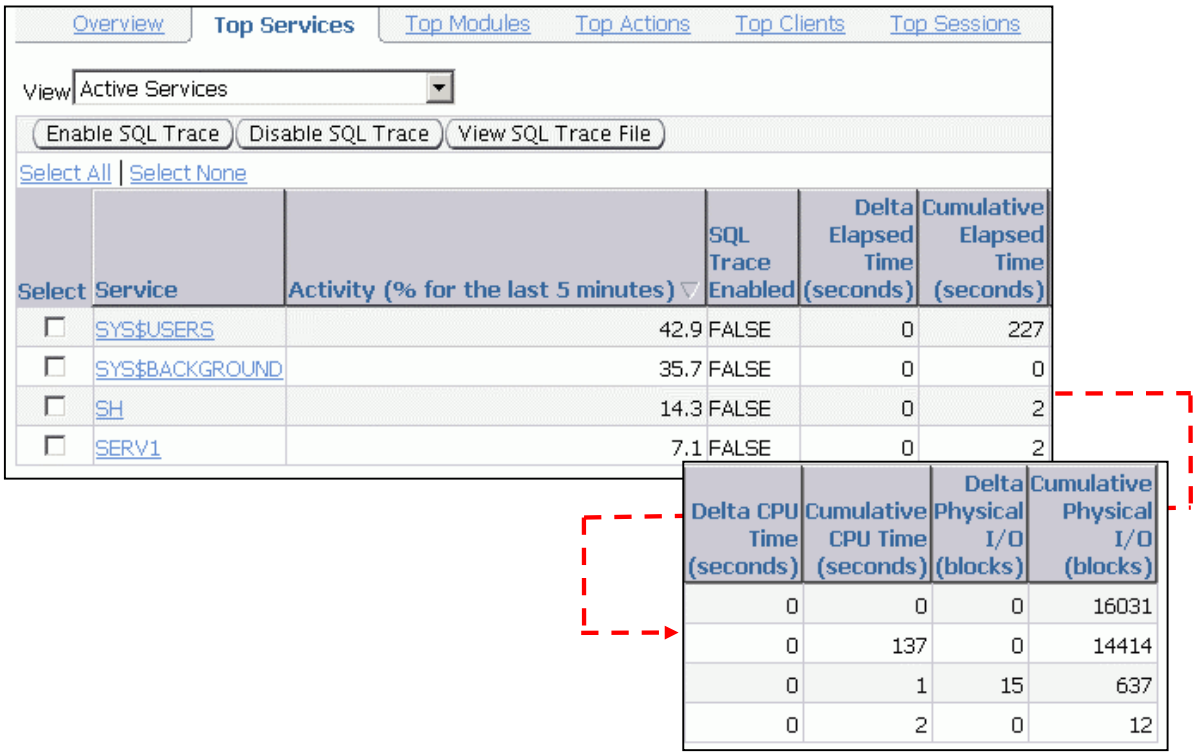
On the Top Consumers page, click the Top Sessions tab to see critical statistics of the sessions using the most resources

- CPU
- PGA Memory
- Logical Reads
- Physical Read
- Hard Parse count,
- Sort count.

Click a column name to have the results sorted by the value in that column.

The table on this page lists the sessions sorted by logical reads. This shows that the user `INVENTORY` in session 36 is producing the greatest number of logical reads at this particular time.

Performance Monitoring: Top Services



Performance Monitoring: Top Services

In multitier systems where there is an application server that is pooling database connections, viewing sessions may not provide the information you need to analyze performance. Grouping sessions into service names enables you to monitor performance more accurately.

In the example in the slide, there are four services: `SYS$USERS`, `SYS$BACKGROUND`, `SH`, and `SERV1`. Regardless of the session that was used for a particular request, if it connected via one of these services, the performance data of the session is captured under that service name. Of the application services shown (`SH` and `SERV1`), it is clear from this listing that the `SH` service was more active during this five-minute interval.

Managing Memory Components

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

ORACLE

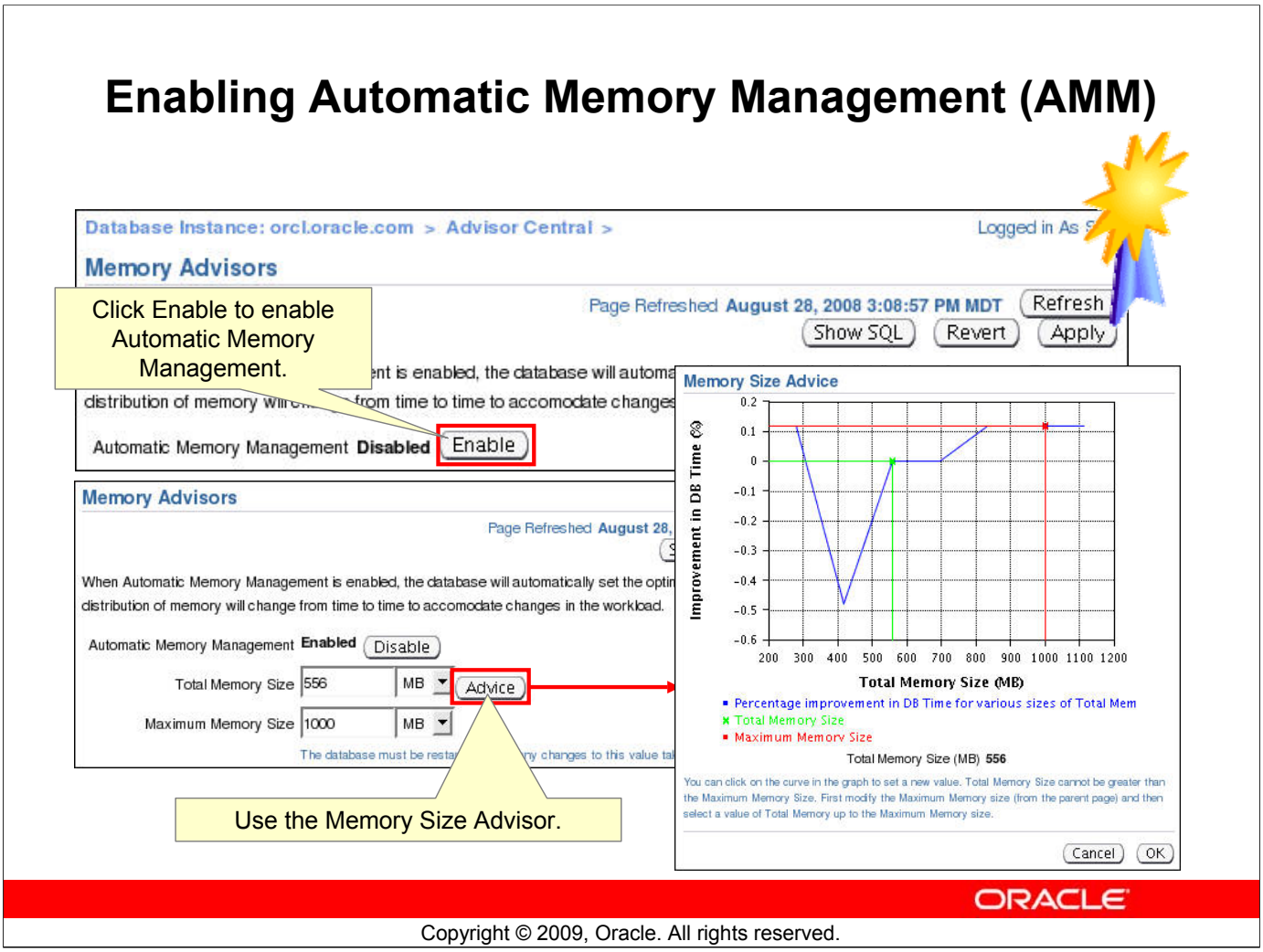
Managing Memory Components

Oracle Database 11*g* enables you to specify the total memory allocated to the instance. Memory will be dynamically reallocated between the System Global Area (SGA) and Program Global Area (PGA) as needed. This method is called Automatic Memory Management (AMM) and is available on only those platforms that support dynamic release of memory. This simplifies your memory management tasks.

Memory advisors are available to help you set the initialization parameters on various levels. Which advisor is available depends on the level on which you are specifying the memory parameters. If you enable AMM, only the Memory Size Advisor is available.

Automatic Shared Memory Management (ASMM) enables you to manage the SGA as a whole. The SGA comprises several components. The sizes of many of these components are dynamically adjusted for best performance within the limits of the initialization parameters. When the AMM is enabled, the ASMM is automatically enabled. If the ASMM is enabled but not the AMM, the SGA Size Advisor is available.

You can manage the size of individual components manually by setting the initialization parameter for each component. If the Oracle server notifies you of a performance problem that is related to the size of an SGA or PGA component, you can use the Memory Advisor for the component to determine appropriate new settings. The Memory Advisor can model the effect of parameter changes.



Enabling Automatic Memory Management (AMM)

If you did not enable Automatic Memory Management (AMM) when you configured your database, you can enable it by performing the following steps:

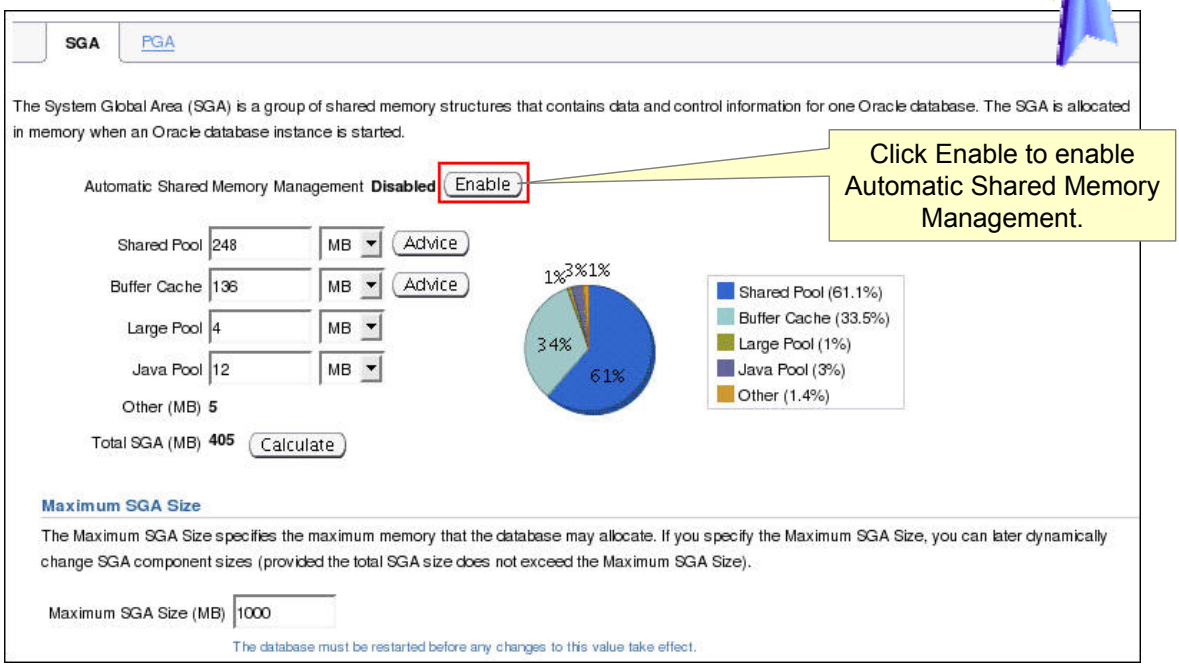
1. On the Database home page, click the Server tab.
2. Click Memory Advisors in the Database Configuration region.
   The Memory Advisors page appears.
3. Click Enable for Automatic Memory Management.
   The Enable Automatic Memory Management page appears.
4. Set the values for Total Memory Size and Maximum Memory Size for Automatic Memory Management.
   Note: If you change the Maximum Memory Size, the database instance must be restarted.
5. Click OK.

You can increase the size at a later time by increasing the value of the Total Memory Size field or the `MEMORY_TARGET` initialization parameter. However, you cannot set it higher than the value specified by the Maximum Memory Size field or the `MEMORY_MAX_TARGET` parameter. For more information, see the *Oracle Database Administrator's Guide*.

After AMM is enabled, the Memory Size Advisor is available to help you adjust the maximum and target memory sizes.

Note: Oracle recommends that you use Automatic Memory Management to simplify memory management tasks.

Enabling Automatic Shared Memory Management (ASMM)



ORACLE

Enabling Automatic Shared Memory Management (ASMM)

Automatic Shared Memory Management is automatically enabled if you have enabled AMM. If you have not enabled AMM or did not enable ASMM when you configured your database, you can enable Automatic Shared Memory Management by performing the following steps:

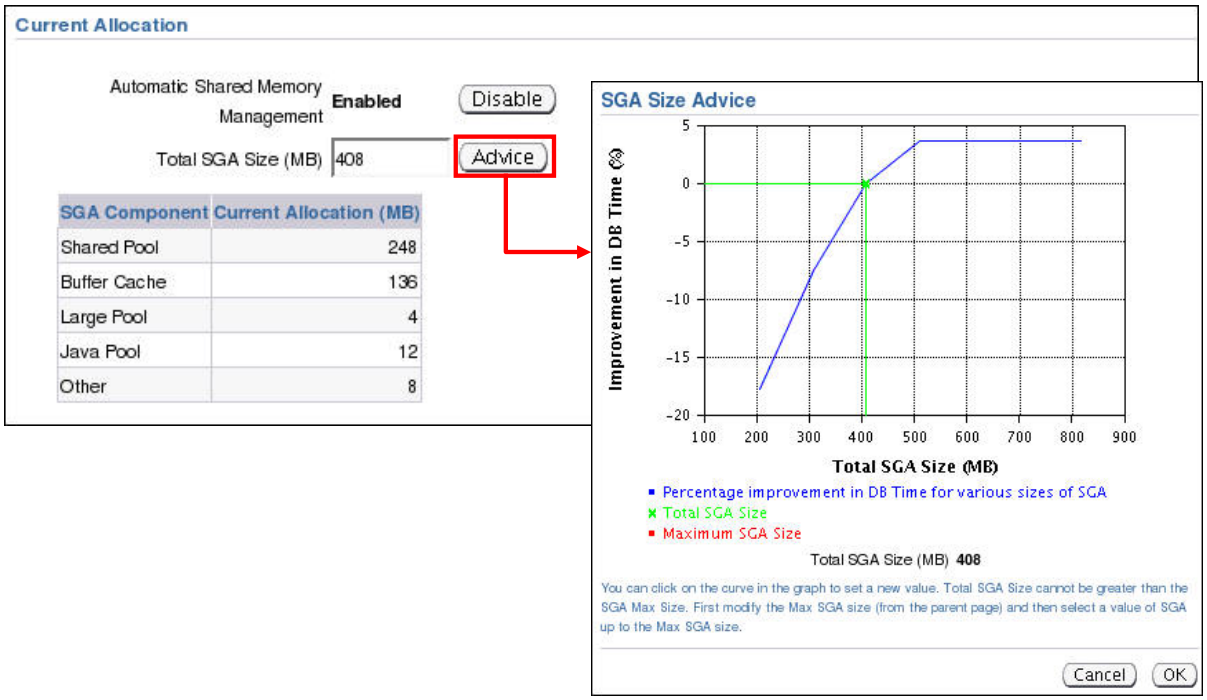
1. On the Database home page, click the Server tab.
2. Click Memory Advisors in the Database Configuration region.
   The Memory Advisors page appears.
3. Scroll down to the SGA section. Click Enable for Automatic Shared Memory Management.
   The Enable Automatic Shared Memory Management page appears.
4. Specify the total SGA size. Click OK.

You can increase the total SGA size at a later time by increasing the value of the Total SGA Size field or the `SGA_TARGET` initialization parameter. However, you cannot set it higher than the value specified by the Maximum SGA Size field or the `SGA_MAX_SIZE` parameter. For more information, see the *Oracle Database Administrator's Guide*.

When AMM is disabled, the PGA advisor is accessible. The PGA advisor is recommended for setting the PGA memory value. Click the PGA tab to access the PGA property page. Click Advice to invoke the PGA Advisor.

Note: Oracle recommends that you use Automatic Shared Memory Management to simplify your memory management tasks.

Automatic Shared Memory Advisor



Automatic Shared Memory Advisor

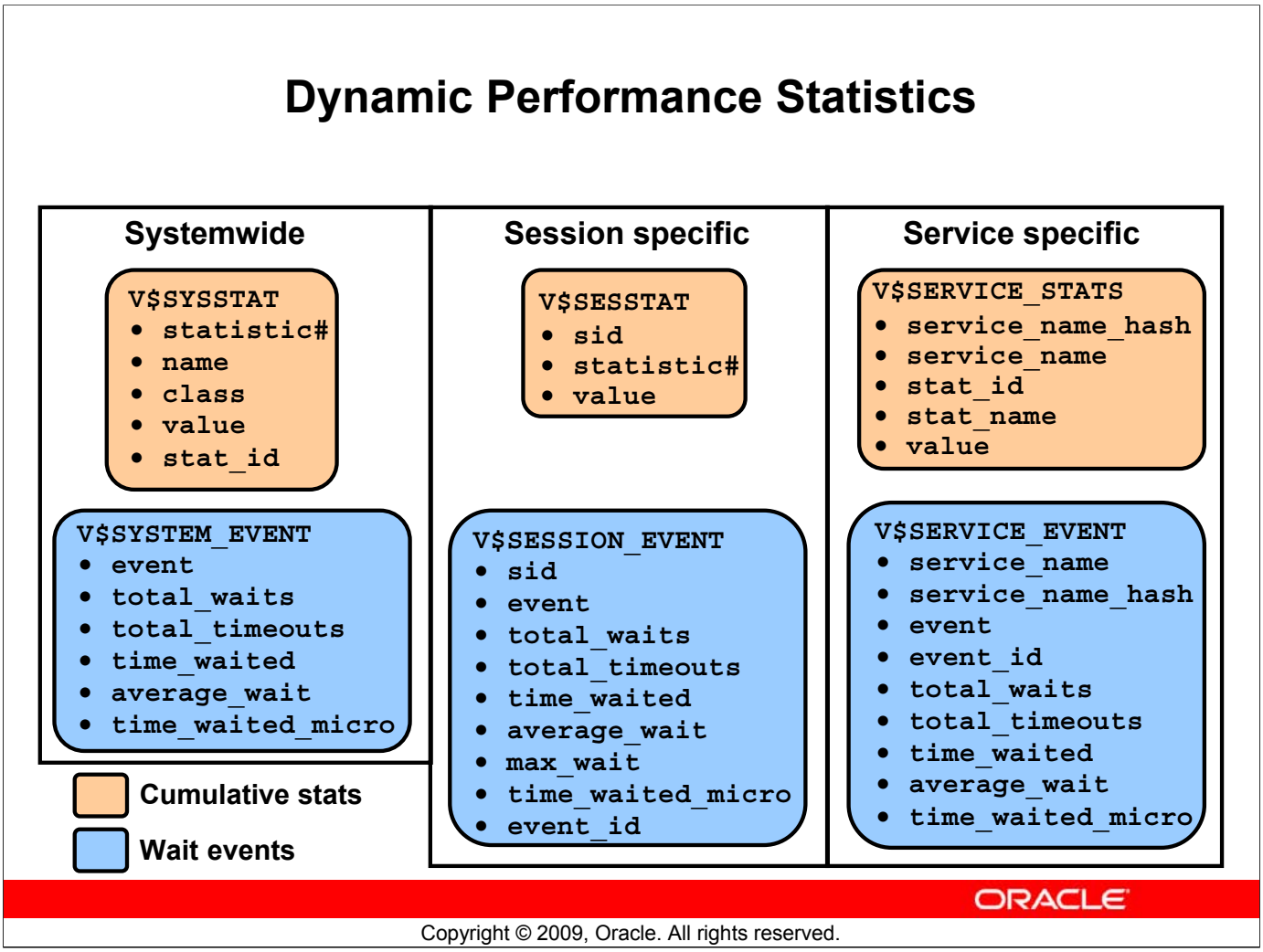
When ASMM is enabled, you should not set the initialization parameters for the specific components of shared memory that ASMM manages. After ASMM is enabled, the SGA Size Advisor is available to help you choose the best value for total SGA size.

Before enabling ASMM, you should remove the individual memory area parameters from the SPFILE because having them set can impose restrictions on ASMM. If, after seeing the effects of the ASMM allocations, you decide that you want to adjust certain component allocations, you can specify values for those components. If the values you are specifying are lower than the current values, then those values are treated as minimum memory sizes for their respective components. If the values you are specifying are larger than the current values, then the size of the memory components are resized upward to the values you provided as long as free memory is available. Doing this limits the amount of memory available for automatic adjustment, but the capability is available if your environment requires special sizing that is not accommodated by ASMM.

The initialization parameters of concern are the following:

- `SHARED_POOL_SIZE`
- `LARGE_POOL_SIZE`
- `JAVA_POOL_SIZE`
- `DB_CACHE_SIZE`
- `STREAMS_POOL_SIZE`

To adjust these parameters while ASMM is enabled, you must use the `ALTER SYSTEM` command.



Dynamic Performance Statistics

Statistics must be available for the effective diagnosis of performance problems. The Oracle server generates many types of statistics for different levels of granularity.

At the systemwide level, session level, and service level, both wait events and accumulated statistics are computed. In the slide, the top row of views shows the cumulative statistics. The bottom row shows the wait event views.

When analyzing a performance problem in any of these scopes, you typically look at the change in statistics (delta value) over the period of time you are interested in. All the possible wait events are cataloged in the `V$EVENT_NAME` view. All statistics are cataloged in the `V$STATNAME` view; approximately 480 statistics are available in Oracle Database.

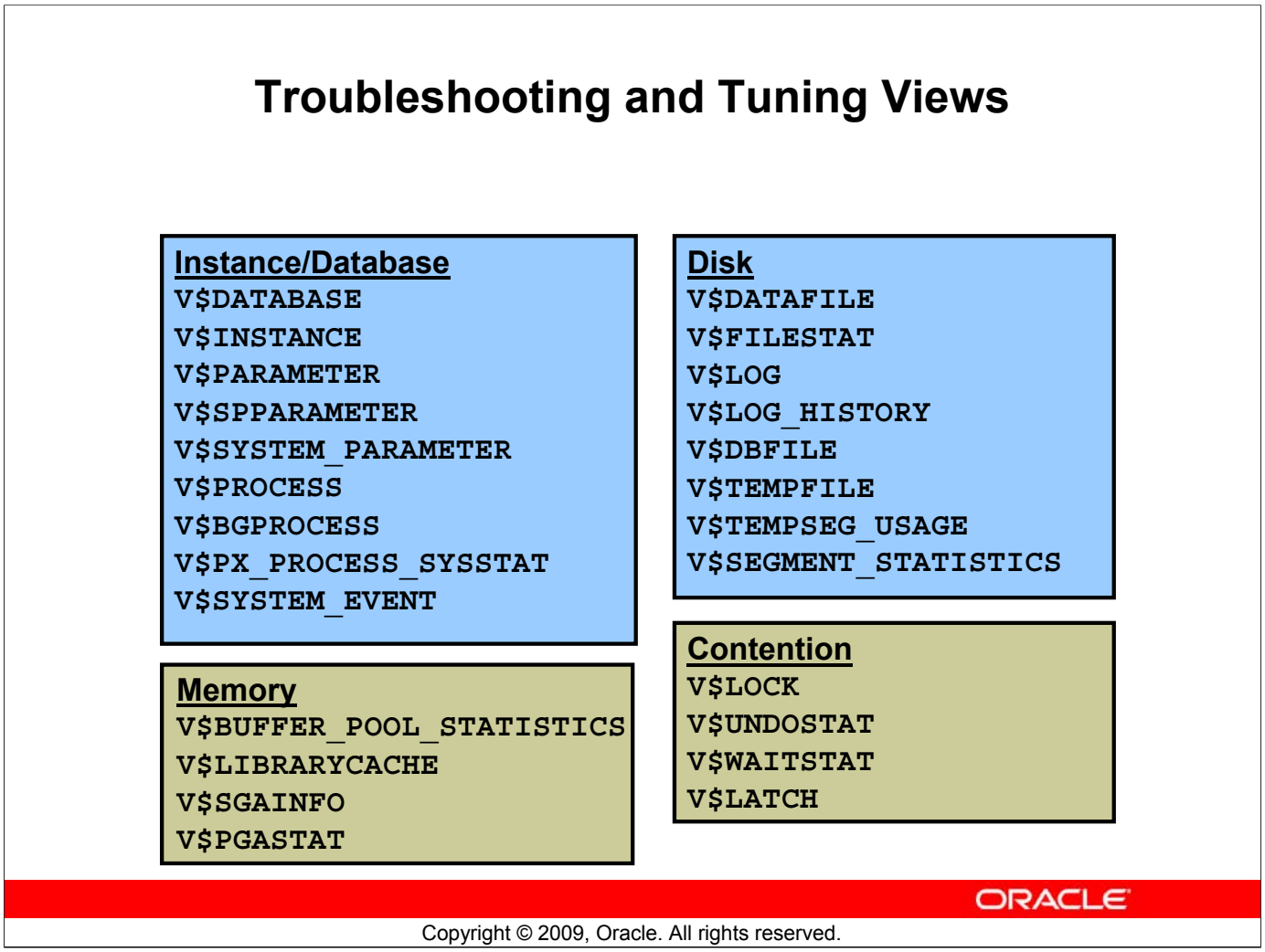
Dynamic Performance Statistics (continued)

Displaying Systemwide Statistics

Example:

```
SQL> SELECT name, class, value FROM v$sysstat;
NAME                                 CLASS      VALUE
-------------------------------- ------ ----------
...
table scans (short tables)              64     135116
table scans (long tables)               64        250
table scans (rowid ranges)              64          0
table scans (cache partitions)          64          3
table scans (direct read)               64          0
table scan rows gotten                  64   14789836
table scan blocks gotten                64     558542
...
```

Systemwide statistics are classified by the tuning topic and the debugging purpose. The classes include general instance activity, redo log buffer activity, locking, database buffer cache activity, and so on.



Troubleshooting and Tuning Views

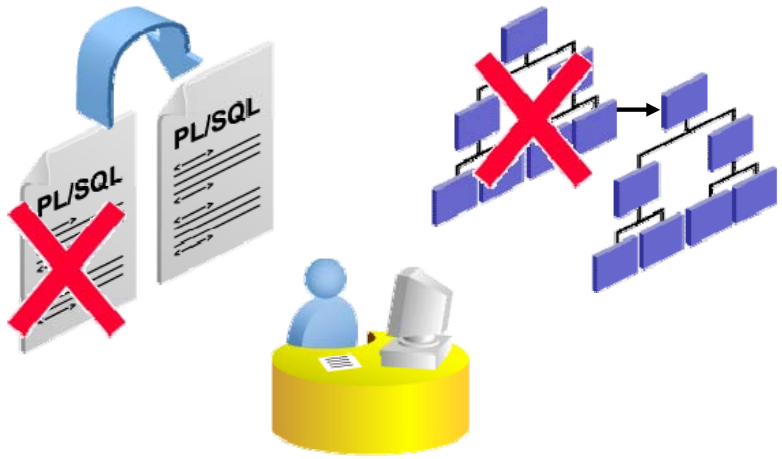
The slide lists some of the views that can help you determine the cause of performance problems or analyze the current status of your database.

For a complete description of these views, see the *Oracle Database Reference*.

Invalid and Unusable Objects

Effect on performance:

- PL/SQL code objects are recompiled.
- Indexes are rebuilt.



ORACLE

Invalid and Unusable Objects

Invalid PL/SQL objects and unusable indexes have an impact on performance. Invalid PL/SQL object must be recompiled before they can be used. This requires the compile time to be added to the first action that attempts to access the PL/SQL package, procedure, or function. If the PL/SQL does not recompile successfully, the operation fails with an error. Unusable indexes are ignored by the optimizer. If proper performance of a SQL statement depends on an index that has been marked unusable, the performance does not improve until the index is rebuilt.

Invalid PL/SQL objects: The current status of PL/SQL objects can be viewed by querying the data dictionary. You can find invalid PL/SQL objects with:

```
SELECT object_name, object_type FROM DBA_OBJECTS
WHERE status = 'INVALID';
```

By default, the Owner's Invalid Object Count metric is checked every 24 hours. If the number of objects for an individual owner exceeds two, an alert is issued.

If you find PL/SQL objects with a status of `INVALID`, the first question that you need to answer is "Has this object *ever* been `VALID`?" An application developer often neglects to clean up code that does not work. If the PL/SQL object is invalid because of a code error, there is little that can be done until that error is fixed. If the procedure was valid at some time in the past and has recently become invalid, you have two options for fixing the problem:

Invalid and Unusable Objects (continued)

- Do nothing. Most PL/SQL objects automatically recompile if needed when they are called. Users experience a slight delay while the objects recompile. (In most cases, this delay is not even noticeable.)
- Manually recompile the invalid object.

Invalid PL/SQL objects can be manually recompiled by using Enterprise Manager or through SQL commands:

```
ALTER PROCEDURE HR.add_job_history COMPILE;
```

Manually recompiling PL/SQL packages requires two steps:

```
ALTER PACKAGE HR.maintainemp COMPILE;
ALTER PACKAGE HR.maintainemp COMPILE BODY;
```

Unusable indexes: Invalid indexes can be found by querying the DBA_INDEXES data dictionary view:

```
SELECT index_name, table_name FROM DBA_INDEXES
WHERE status = 'UNUSABLE';
```

For partitioned indexes, the status is held in the DBA_IND_PARTITIONS view.

Unusable indexes are made valid by rebuilding them to recalculate the pointers. Rebuilding an unusable index re-creates the index in a new location and then drops the unusable index. This can be done either by using Enterprise Manager or through SQL commands:

```
ALTER INDEX HR.emp_empid_pk REBUILD;
ALTER INDEX HR.emp_empid_pk REBUILD ONLINE;
ALTER INDEX HR.email REBUILD TABLESPACE USERS;
```

If the TABLESPACE clause is left out, the index is rebuilt in the same tablespace where it already exists. The REBUILD ONLINE clause enables users to continue updating the index's table while the rebuild takes place. (Without the ONLINE keyword, users must wait for the rebuild to finish before performing DML on the affected table. If the index is unusable, it is not used during the rebuild even if the ONLINE keyword is used.)

Enterprise Manager uses the Reorganize action to repair an UNUSABLE index.

Note: Rebuilding an index requires that free space be available for the rebuild. Verify that there is sufficient space before attempting the rebuild. Enterprise Manager automatically checks space requirements.

Quiz

Automatic Memory Management allows the Oracle instance to reallocate memory from the _____ to the SGA .

1. Large Pool
2. Log Buffer
3. PGA
4. Streams Pool

Answer: 3

Quiz

SGA_TARGET may not be sized greater than _____ .

1. LOG_BUFFER
2. SGA_MAX_SIZE
3. STREAMS_POOL_SIZE
4. PGA_AGGREGATE_TARGET

Answer: 2

Summary

In this lesson, you should have learned how to:

- Use Enterprise Manager to monitor performance
- Use Automatic Memory Management (AMM)
- Use the Memory Advisor to size memory buffers
- View performance-related dynamic views
- Troubleshoot invalid and unusable objects

ORACLE

Practice 13 Overview: Monitoring and Improving Performance

This practice covers the following topics:

- Detecting and repairing unusable indexes
- Using the Performance page in Enterprise Manager