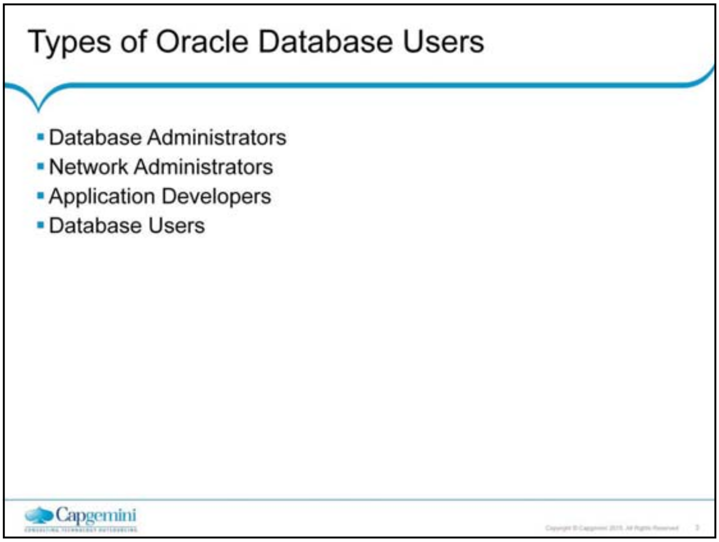
Oracle 11g DBA Fundamentals Overview

Lesson 01: Overview of Administering an Oracle Database

Lesson Objectives

- Types of Oracle Database Users
- Tasks of a Database Administrator
- DBA Security and Privileges
- Tools for Administering the Database
- Review the Oracle Database 11g architecture
- Managing Oracle Database Processes

Capgemini

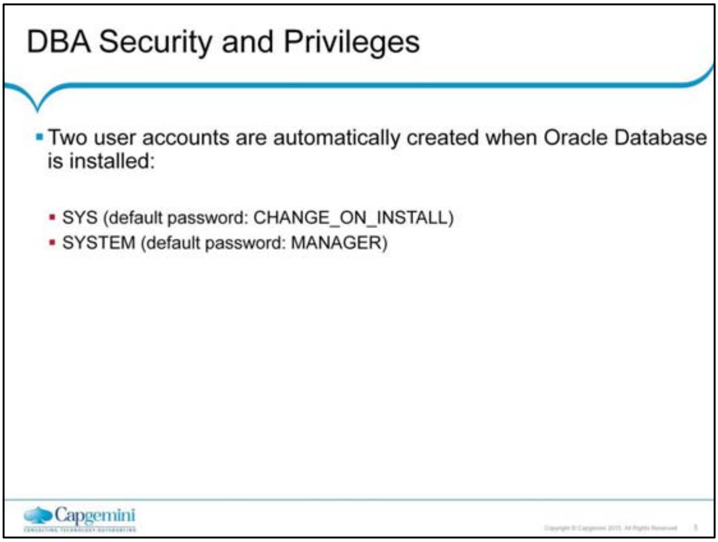

Types of Oracle Database Users
Database Administrators
Network Administrators
Application Developers
Database Users
Capgemini

Tasks of a Database Administrator

- Task 1: Evaluate the Database Server Hardware
- Task 2: Install the Oracle Database Software
- Task 3: Plan the Database
- Task 4: Create and Open the Database
- Task 5: Back Up the Database
- Task 6: Enroll System Users
- Task 7: Implement the Database Design
- Task 8: Back Up the Fully Functional Database
- Task 9: Tune Database Performance

Note:
- Keep titles concise – for example: Description, Features, Characteristics, Examples
- Ideas should be clean and simple
- No period at the end of bullet points

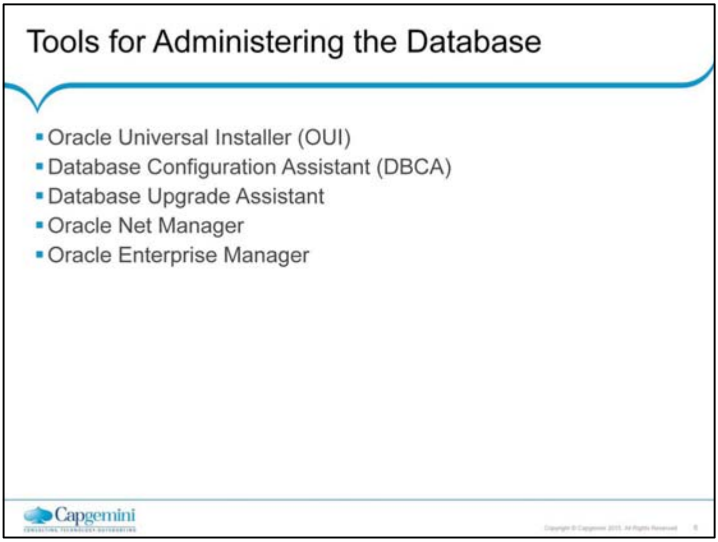
Add the notes here.



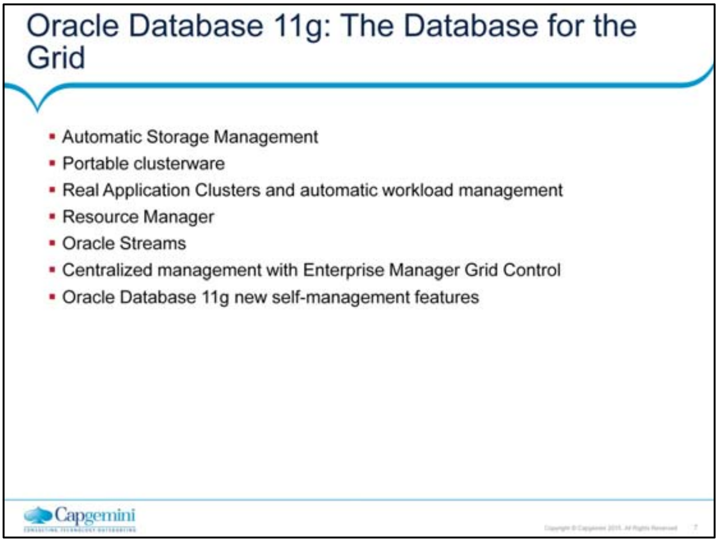
SYS
When you create an Oracle Database, the user SYS is automatically created and granted the DBA role. All of the base tables and views for the database data dictionary are stored in the schema SYS. These base tables and views are critical for the operation of Oracle Database. To maintain the integrity of the data dictionary, tables in the SYS schema are manipulated only by the database. They should never be modified by any user or database administrator, and no one should create any tables in the schema of user SYS.

SYSTEM
When you create an Oracle Database, the user SYSTEM is also automatically created and granted the DBA role. The SYSTEM username is used to create additional tables and views that display administrative information, and internal tables and views used by various Oracle Database options and tools.

The DBA Role
A predefined DBA role is automatically created with every Oracle Database installation. This role contains most database system privileges. Therefore, the DBA role should be granted only to actual database administrators.



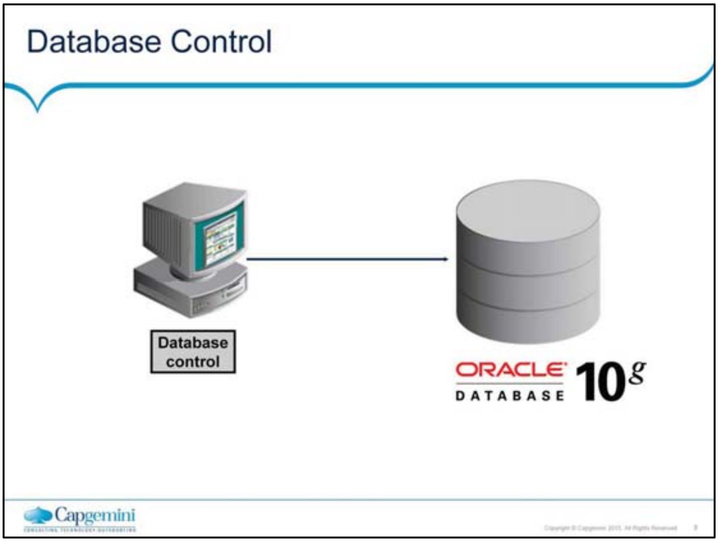
Add the notes here.



Oracle Database 11g: The Database for the Grid

Oracle Database 11g is the first database that is designed for grid computing. To summarize, some of the most important features are the following:

- Automatic Storage Management (ASM) virtualizes your storage and provides easy provisioning of your database storage.
- Oracle Database 11g offers portable clusterware that runs on all platforms.
- Oracle Database 11g offers automatic workload management for services within a RAC database.
- Oracle Database 11g provides additional mappings for consumer groups based on user host machine, application, OS username, or service.
- Oracle Streams can stream data between databases, nodes, or blade farms in a grid. It provides a unified framework for information sharing, combining message queuing, replication, events, and data warehouse loading into a single technology.
- Enterprise Manager Grid Control provides a single tool that can monitor and manage not only every Oracle software element (Oracle Application Server 11g and Oracle Database 11g) in your grid but also Web applications via Application Performance Management (APM), hosts, storage devices, and server load balancers.



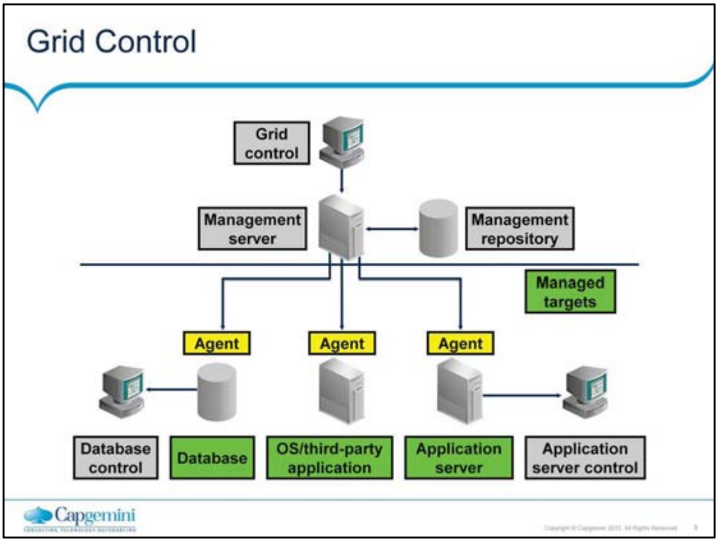
Database Control

Oracle Database 11g ships with Oracle Enterprise Manager's Database Control. Database Control is a web-enabled control console that the database administrator can use for:

- Performance monitoring
- Managing proactive alerts
- Controlling maintenance wizards and advisors
- User and database object administration
- Database backup and recovery
- Storage management

and much more.

Each Oracle Database 11g you create will have its own Database Control. You will be using Enterprise Manager Database Control in this course to manage the database on your classroom PC.

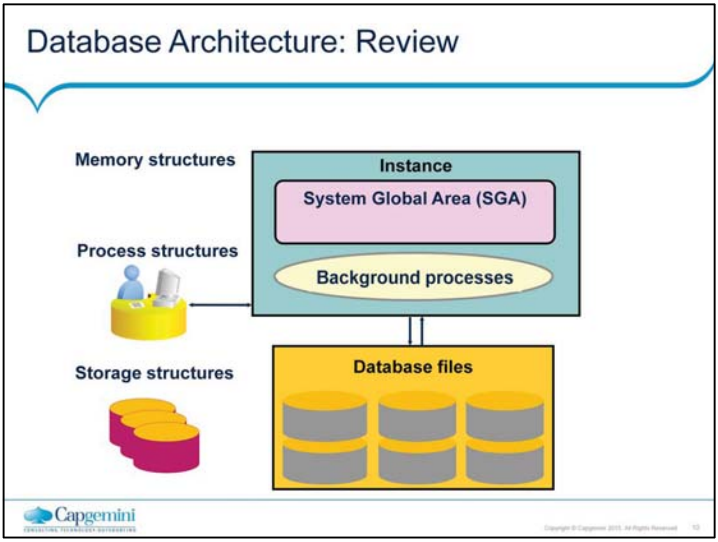


Grid Control

Database Control's capabilities can be extended and integrated with the rest of your systems using Oracle Enterprise Manager's Grid Control. The architecture of the Grid Control framework provides a high level of flexibility and functionality. You can easily customize Enterprise Manager to suit the monitoring and administrative needs of your environment.

The typical Enterprise Manager framework configuration consists of the following functional areas:

- Managed targets
- Management services
- Web-based grid control
- Database control
- Application server control



Database Architecture: Review

The following pages are a basic review of the Oracle database architecture. In this course, you enhance your knowledge of Oracle's database structures, processes, and utilities.

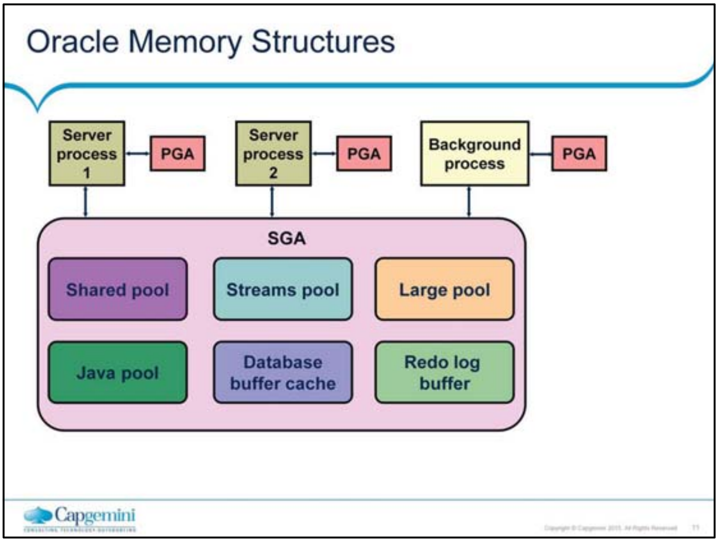
Each running Oracle database is associated with an Oracle instance. When a database is started on a database server, the Oracle software allocates a shared memory area called the System Global Area (SGA) and starts several Oracle background processes. This combination of the SGA and the Oracle processes is called an Oracle instance.

After starting an 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.

Multiple instances can execute concurrently on the same computer, each accessing its own physical database.

You can look at the Oracle database architecture as various interrelated structural components.

An Oracle database uses memory structures and processes to manage and access the database. All memory structures exist in the main memory of the computers that constitute the database server. Processes are jobs that work in the memory of these computers. A process is defined as a “thread of control” or a mechanism in an operating system that can run a series of steps.



Oracle Memory Structures

The basic memory structures associated with an Oracle instance include:

- System Global Area (SGA): Shared by all server and background processes
- Program Global Area (PGA): Private to each server and background process; there is one PGA for each process

The SGA is a memory area that contains data and control information for the instance.

The SGA includes the following data structures:

- Database buffer cache: Caches blocks of data retrieved from the database
- Redo log buffer: Caches redo information (used for instance recovery) until it can be written to the physical redo log files stored on the disk
- Shared pool: Caches various constructs that can be shared among users
- Large pool: Is an 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: Is used for all session-specific Java code and data within the Java Virtual Machine (JVM)
- Streams pool: Is used by Oracle Streams

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

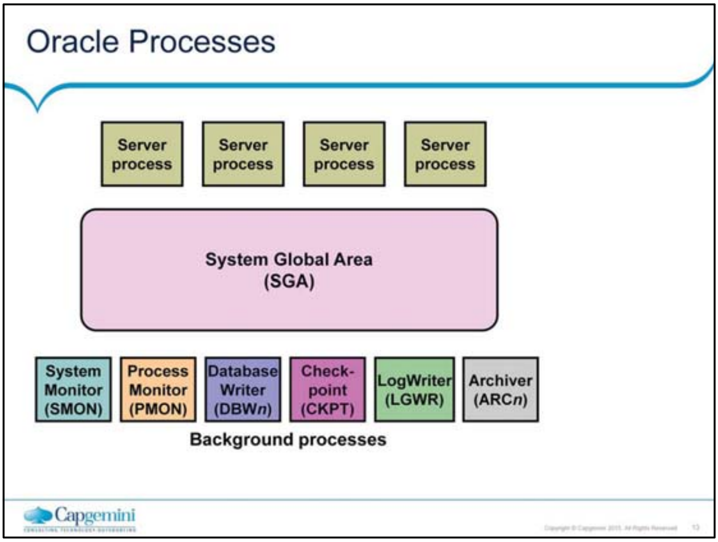
Oracle Memory Structures Full Notes Page

Oracle Memory Structures (continued)

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 created 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.

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

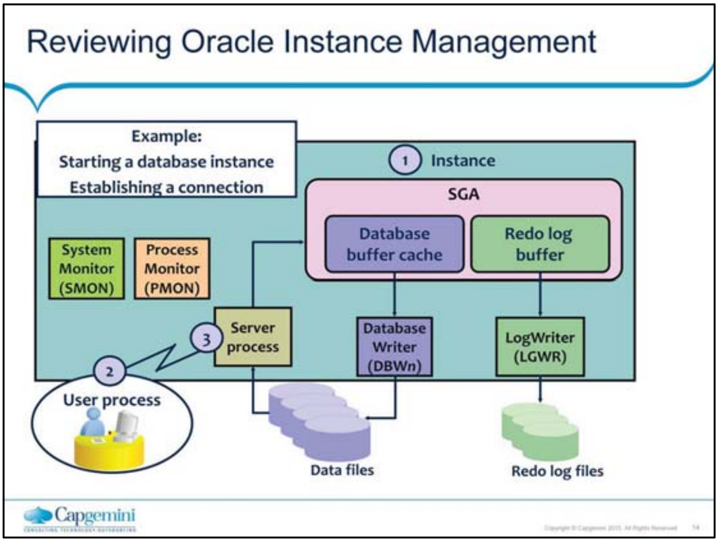
The Oracle database uses initialization parameters to create and configure memory structures. For example, the SGA_TARGET parameter specifies the total amount of space available to the SGA. If you set SGA_TARGET to 0, Automatic Shared Memory Management is disabled.



Oracle Processes

When you invoke an application program or an Oracle tool, such as Enterprise Manager, the Oracle server creates a server process to execute the commands issued by the application. The Oracle server also creates 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. Which background processes are present depends on the features that are being used in the database. The most common background processes are the following:

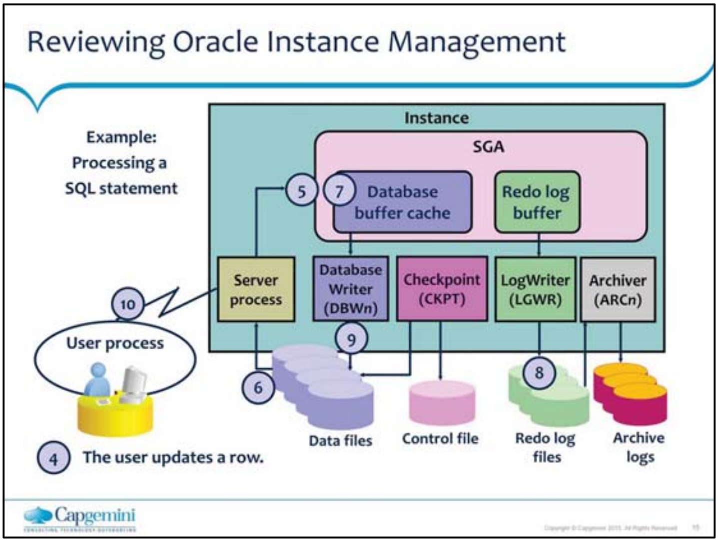
- System Monitor (SMON): Performs crash recovery when the instance is started following a failure
- Process Monitor (PMON): Performs process cleanup when a user process fails
- Database Writer (DBWn): Writes modified blocks from the database buffer cache to the data files on the disk
- Checkpoint (CKPT): Updates all the data files and control files of the database to indicate the most recent checkpoint
- LogWriter (LGWR): Writes redo log entries to the disk
- Archiver (ARCn): Copies redo log files to an archival storage when a log switch occurs



Reviewing Oracle Instance Management

The following example describes the most basic level of operations that the Oracle database performs. It illustrates an Oracle configuration where the user and associated server processes are on separate computers (connected through a network).

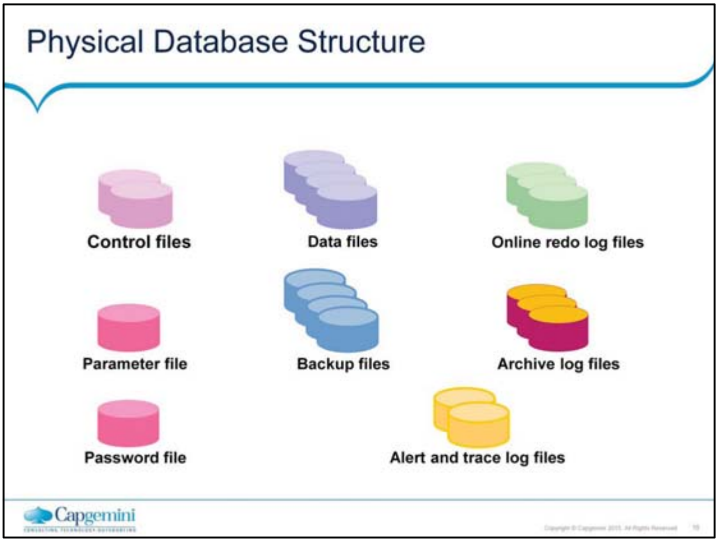
1. An instance has started on the computer running Oracle (often called the host or database server).
2. A computer running an application (a local computer or client workstation) runs the application in a user process. The client application attempts to establish a connection to the instance by using the Oracle Net Services driver.
3. The instance detects the connection request from the application and connects to a server process on behalf of the user process.



Reviewing Oracle Instance Management (continued)

4. The user updates a row.
5. The server process receives the statement and checks whether it is already in the shared pool of the SGA. If a shared SQL area is found, the server process checks the user's access privileges to the requested data, and the previously existing shared SQL area is used to process the statement. If the statement is not in the shared pool, then a new shared SQL area is allocated for the statement, so that it can be parsed and processed.
6. The server process retrieves any necessary data values from the actual data file (table) or from data blocks that are stored in the SGA.
7. The server process modifies the table data in the SGA.
8. When the transaction is committed, the LGWR process immediately records the transaction in the redo log file.
9. The DBWn process writes modified blocks to the disk when doing so is efficient.
10. The server process sends a success or error message across the network to the application.

Throughout this entire procedure, the other background processes run, watching for conditions that require intervention.



Physical Database Structure

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 within the database.
- Data files: Contain the user or application data of the database
- Online redo log files: Allow for instance recovery of the database. If the database crashes and does not lose any data files, then 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 to connect remotely to the database 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.
- Archive 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.

Physical Database Structure Full Notes Page

Physical Database Structure (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 files: Also known as alert logs, these are special trace files. The alert log of a database is a chronological log of messages and errors. Oracle recommends reviewing these files.

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	Defines the location for RMAN backups

Exam:

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

Capgemini

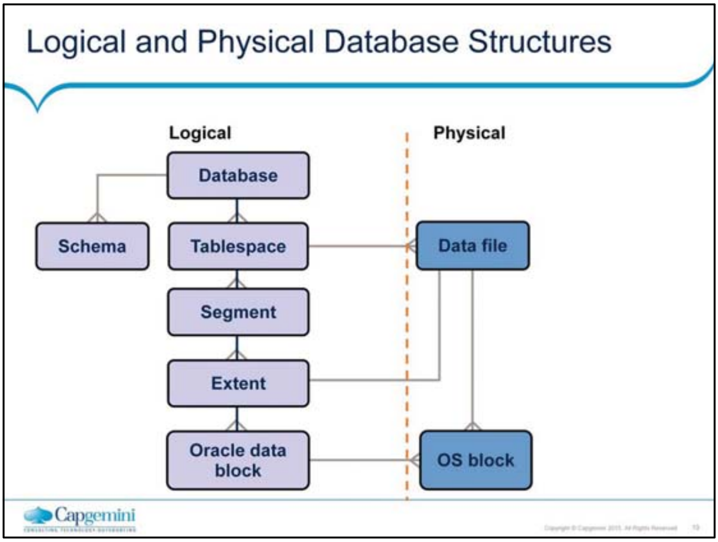
Oracle Managed Files (OMF)

Oracle Managed Files (OMF) eliminate the need for you to directly manage the operating system files that make up 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 to allow 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.



Logical and Physical Database Structures

An Oracle database is a collection of data that is treated as a unit. The general purpose of a database is to store and retrieve related information. The database has logical structures and physical structures.

Tablespaces

A database is divided into logical storage units called tablespaces, which group related logical structures together. For example, tablespaces commonly group all of an application's objects to simplify some administrative operations. You may have a tablespace for application data and an additional one for application indexes.

Databases, Tablespaces, and Data Files

The relationship among databases, tablespaces, and data files is illustrated in the slide. Each database is logically divided into one or more tablespaces. One or more data files are explicitly created for each tablespace to physically store the data of all logical structures in a tablespace. If it is a TEMPORARY tablespace, instead of a data file, the tablespace has a temporary file.

Logical and Physical Database Structures (continued)

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.

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 database space on the disk. A data block size is specified for each tablespace when it is created. A database uses and allocates free database space in Oracle data blocks.

Extents

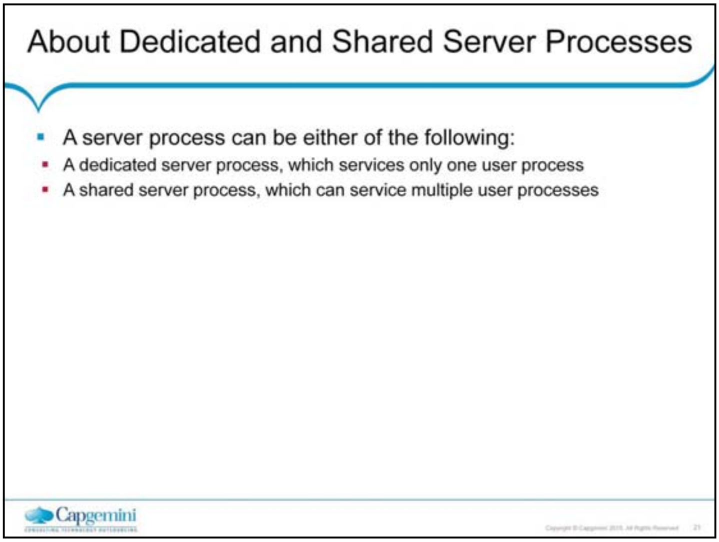
The next level of logical database space is called an extent. An extent is a specific number of contiguous data blocks (obtained in a single allocation) that are used to store a specific type of information.

Segments

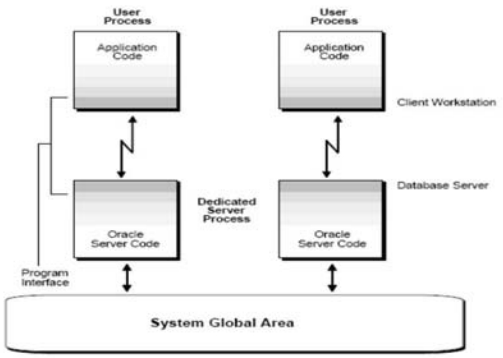
The level of logical database storage above an extent is called a segment. A segment is a set of extents allocated for a certain logical structure. For example, the different types of segments include:

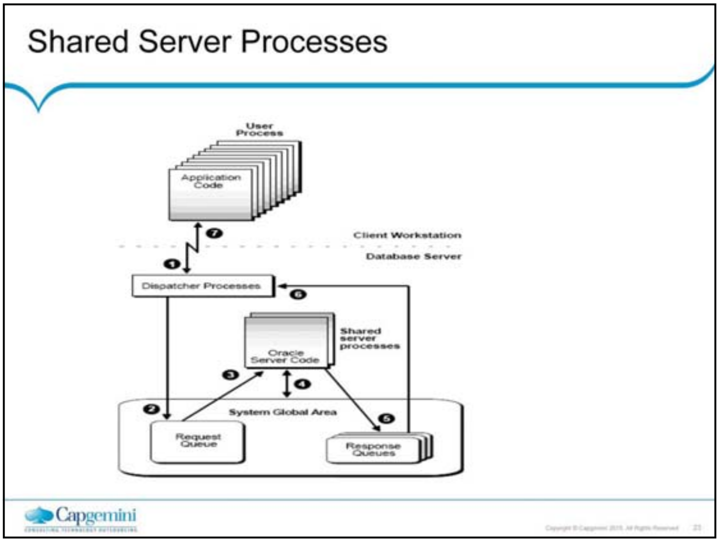
- ***Data segments*:** Each nonclustered, non-index-organized table has a data segment. All of the table's data is stored in the extents of its data segment. For a partitioned table, each partition has a data segment. Each cluster has a data segment. The data of every table in the cluster is stored in 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 by the database administrator 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 a default temporary tablespace for every user or a default temporary tablespace, which is used databasewide.

The Oracle database 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.

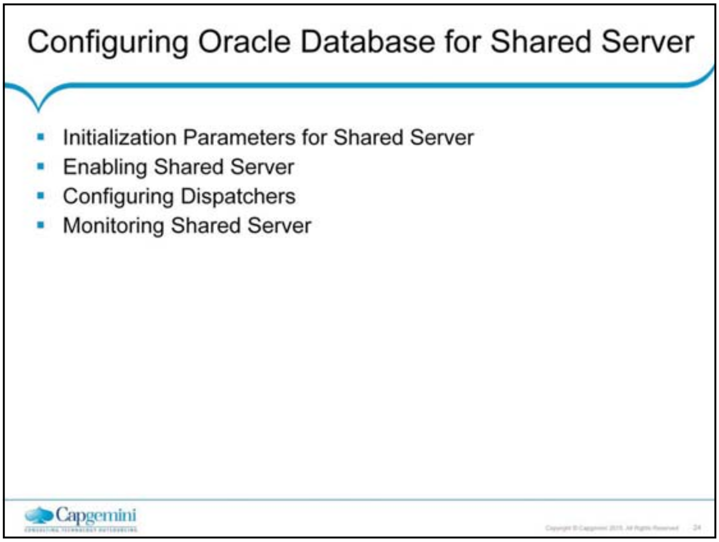

About Dedicated and Shared Server Processes
A server process can be either of the following:
A dedicated server process, which services only one user process
A shared server process, which can service multiple user processes
Capgemini

Dedicated Server Processes



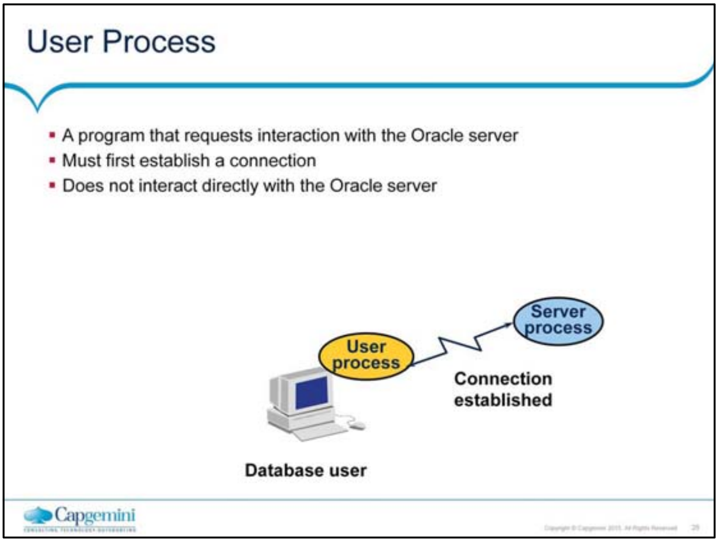


Add the notes here.


Configuring Oracle Database for Shared Server
Initialization Parameters for Shared Server
Enabling Shared Server
Configuring Dispatchers
Monitoring Shared Server
Capgemini

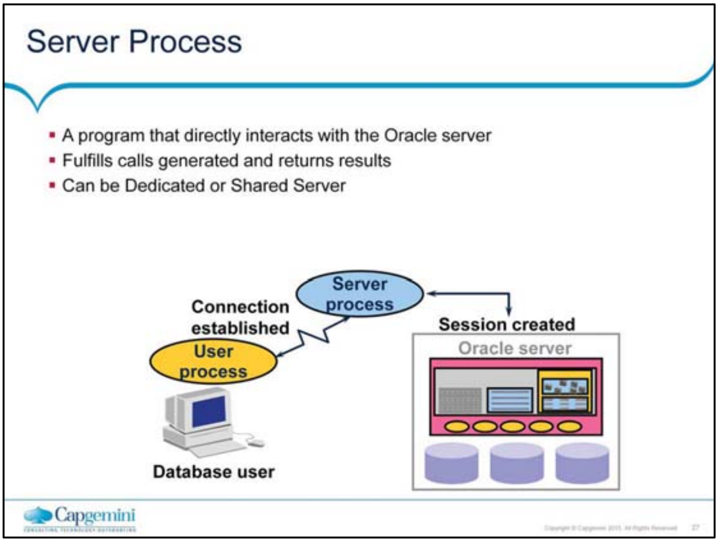
Process Structure

- Oracle takes advantage of various types of processes:
  - User process: Started at the time a database user requests connection to the Oracle server
  - Server process: Connects to the Oracle Instance and is started when a user establishes a session
  - Background processes: Started when an Oracle Instance is started



User Process

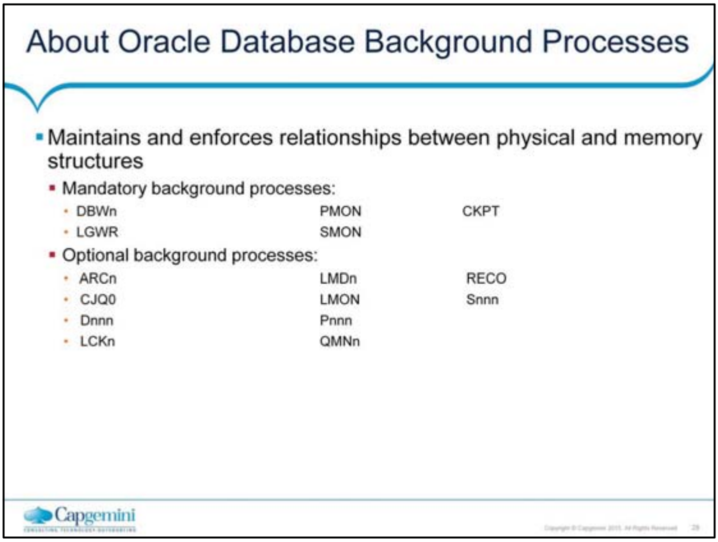
A database user who needs to request information from the database must first make a connection with the Oracle server. The connection is requested using a database interface tool, such as SQL*Plus, and beginning the user process. The user process does not interact directly with the Oracle server. Rather it generates calls through the user program interface (UPI), which creates a session and starts a server process.



Server Process

Once a user has established a connection, a server process is started to handle the user processes requests. A server process can be either a Dedicated Server process or a Shared Server process. In a Dedicated Server environment, the server process handles the request of a single user process. Once a user process disconnects, the server process is terminated. In a Shared Server environment, the server process handles the request of several user processes. The server process communicates with the Oracle server using the Oracle Program Interface (OPI).

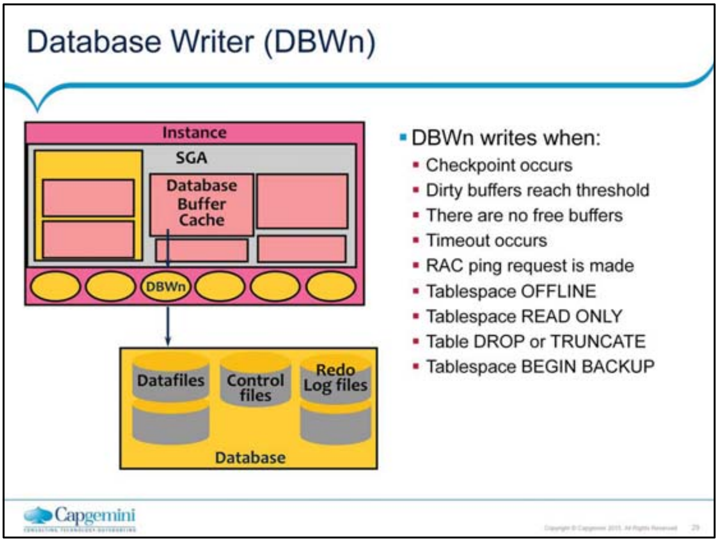
Note: Allocation of server process in a dedicated environment versus a shared environment is covered in further detail in the Oracle9i Database Performance Tuning course.



Background Processes

The Oracle architecture has five mandatory background processes that are discussed further in this lesson. In addition to the mandatory list, Oracle has many optional background process that are started when their option is being used. These optional processes are not within the scope of this course, with the exception of the background process, ARCn. Following is a list of some optional background processes:

RECO: Recoverer
QMNn: Advanced Queuing
ARCn: Archiver
LCKn: RAC Lock Manager–Instance Locks
LMON: RAC DLM Monitor–Global Locks
LMDn: RAC DLM Monitor–Remote Locks
CJQ0: Coordinator Job Queue background process
Dnnn: Dispatcher
Snnn: Shared Server
Pnnn: Parallel Query Slaves

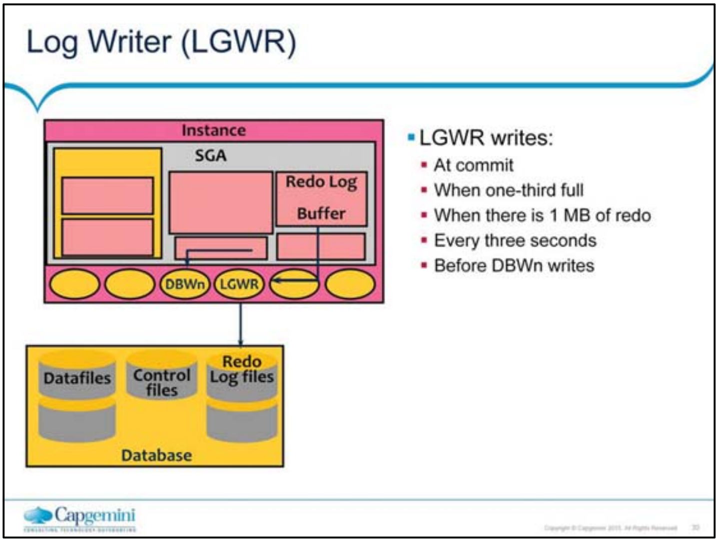


Database Writer (DBWn)

The server process records changes to undo and data blocks in the Database Buffer Cache. DBWn writes the dirty buffers from the Database Buffer Cache to the datafiles. It ensures that a sufficient number of free buffers (buffers that can be overwritten when server processes need to read in blocks from the datafiles) are available in the Database Buffer Cache. Database performance is improved because server processes make changes only in the Database Buffer Cache.

DBWn defers writing to the datafiles until one of the following events occurs:

- Incremental or normal checkpoint
- The number of dirty buffers reaches a threshold value
- A process scans a specified number of blocks when scanning for free buffers and cannot find any
- Timeout occurs
- A ping request in Real Application Clusters (RAC) environment
- Placing a normal or temporary tablespace offline
- Placing a tablespace in read-only mode
- Dropping or truncating a table
- ALTER TABLESPACE tablespace name BEGIN BACKUP



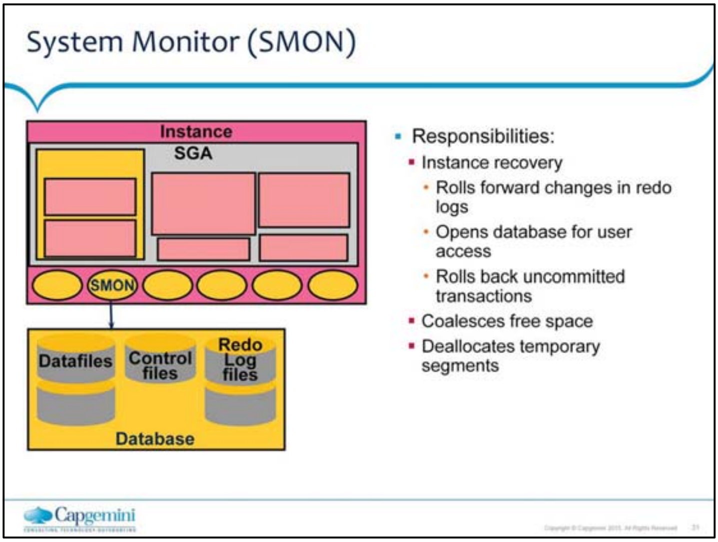
Log Writer (LGWR)

LGWR performs sequential writes from the Redo Log Buffer to the redo log file under the following situations:

- When a transaction commits
- When the Redo Log Buffer is one-third full
- When there is more than 1 MB of changes recorded in the Redo Log Buffer
- Before DBWn writes modified blocks in the Database Buffer Cache to the datafiles
- Every three seconds

Because the redo is needed for recovery, LGWR confirms the commit operation only after the redo is written to disk.

LGWR can also call on DBWn to write to the datafiles.



System Monitor (SMON)

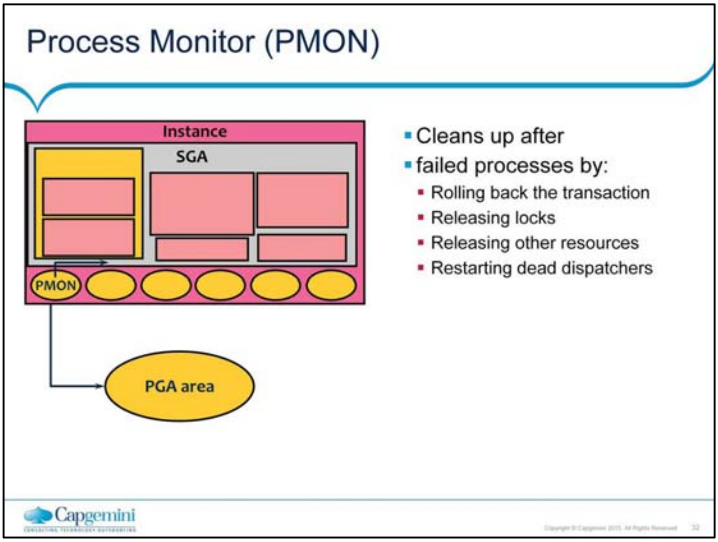
If the Oracle Instance fails, any information in the SGA that has not been written to disk is lost. For example, the failure of the operating system causes an instance failure. After the loss of the instance, the background process SMON automatically performs instance recovery when the database is reopened. Instance recovery consists of the following steps:

1. Rolling forward to recover data that has not been recorded in the datafiles but that has been recorded in the online redo log. This data has not been written to disk because of the loss of the SGA during instance failure. During this process, SMON reads the redo log files and applies the changes recorded in the redo log to the data blocks. Because all committed transactions have been written to the redo logs, this process completely recovers these transactions.
2. Opening the database so that users can log on. Any data that is not locked by unrecovered transactions is immediately available.
3. Rolling back uncommitted transactions. They are rolled back by SMON or by the individual server processes as they access locked data.

SMON also performs some space maintenance functions:

It combines, or coalesces, adjacent areas of free space in the datafiles.

It deallocates temporary segments to return them as free space in datafiles.

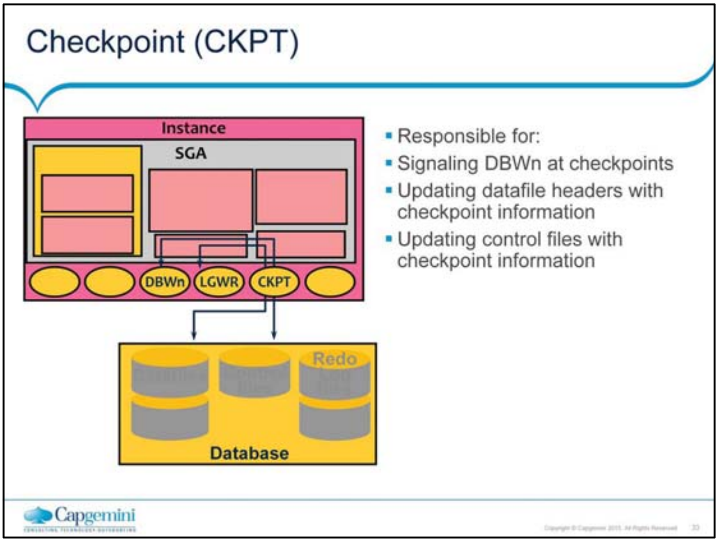


Process Monitor (PMON)

The background process PMON cleans up after failed processes by:

- Rolling back the user's current transaction
- Releasing all currently held table or row locks
- Freeing other resources currently reserved by the user
- Restarts dead dispatchers

Dispatchers are covered in further detail in the Oracle9i Database Administration Fundamentals II course.



Checkpoint (CKPT)

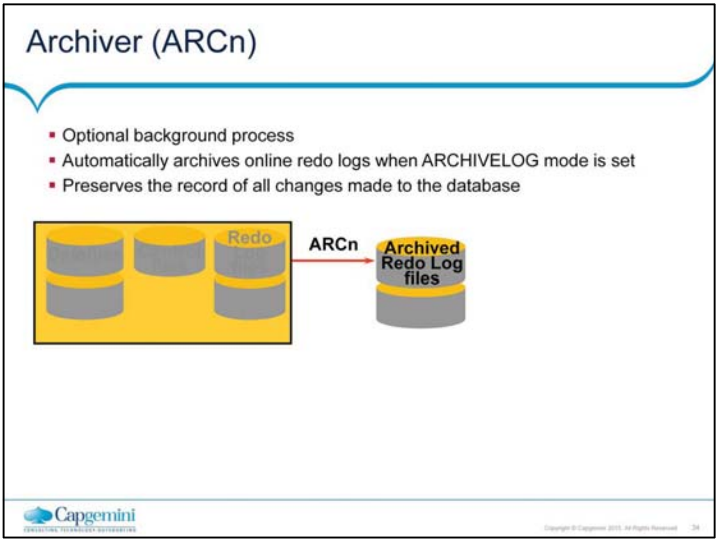
Every three seconds the CKPT process stores data in the control file to identify that place in the redo log file where recovery is to begin, this being called a checkpoint. The purpose of a checkpoint is to ensure that all of the buffers in the Database Buffer Cache that were modified prior to a point in time have been written to the datafiles. This point in time (called the checkpoint position) is where database recovery is to begin in the event of an instance failure. DBWn will already have written all of the buffers in the Database Buffer Cache that were modified prior to that point in time. Prior to Oracle9i, this was done at the end of the redo log. In the event of a log switch CKPT also writes this checkpoint information to the headers of the datafiles.

Checkpoints are initiated for the following reasons:

- To ensure that modified data blocks in memory are written to disk regularly so that data is not lost in case of a system or database failure
- To reduce the time required for instance recovery. Only the redo log entries following the last checkpoint need to be processed for recovery to occur
- To ensure that all committed data has been written to the datafiles during shutdown

Checkpoint information written by CKPT includes checkpoint position, system change number, location in the redo log to begin recovery, information about logs, and so on.

Note: CKPT does not write data blocks to disk or redo blocks to the online redo logs.



Archiver (ARCn)

All other background processes are optional, depending on the configuration of the database; however, one of them, ARCn, is crucial to recovering a database after the loss of a disk. As online redo log files get filled, the Oracle server begins writing to the next online redo log file. The process of switching from one redo log to another is called a log switch. The ARCn process initiates backing up, or archiving, of the filled log group at every log switch. It automatically archives the online redo log before the log can be reused, so that all of the changes made to the database are preserved. This enables the DBA to recover the database to the point of failure even if a disk drive is damaged.

Archiving redo log files:

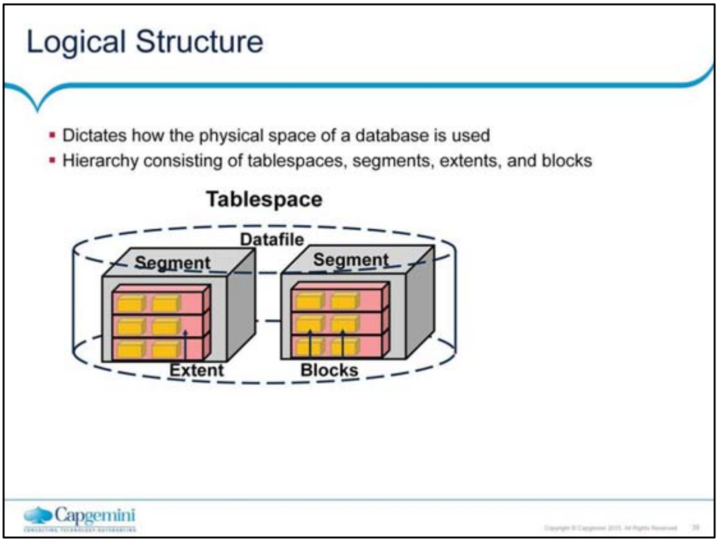
One of the important decisions that a DBA has to make is whether to configure the database to operate in ARCHIVELOG or in NOARCHIVELOG mode.

NOARCHIVELOG mode: In NOARCHIVELOG mode, the online redo log files are overwritten each time a log switch occurs. LGWR does not overwrite a redo log group until the checkpoint for that group is complete. This ensures that committed data can be recovered if there is an instance crash. During the instance crash, only the SGA is lost. There is no loss of disks, only memory. For example, an operating system crash causes an instance crash.

Archiving Redo Log Files (continued)

ARCHIVELOG mode: If the database is configured to run in ARCHIVELOG mode, inactive groups of filled online redo log files must be archived before they can be used again. Since changes made to the database are recorded in the online redo log files, the database administrator can use the physical backup of the datafiles and the archived online redo log files to recover the database without losing any committed data because of any single point of failure, including the loss of a disk. Usually, a production database is configured to run in ARCHIVELOG mode.

Archive log modes are covered in further detail in the Oracle9i Database Administration Fundamentals II course.



Logical Structure

A logical structure hierarchy exists as follows:

- An Oracle database contains at least one tablespace.
- A tablespace contains one or more segments.
- A segment is made up of extents.
- An extent is made up of logical blocks.
- A block is the smallest unit for read and write operations.

The Oracle database architecture includes logical and physical structures that make up the database.

- The physical structure includes the control files, online redo log files, and datafiles that make up the database.
- The logical structure includes tablespaces, segments, extents, and data blocks.

The Oracle server enables fine-grained control of disk space use through tablespace and logical storage structures, including segments, extents, and data blocks.

Logical Structure (continued)

Tablespaces:

The data in an Oracle database are stored in tablespaces.

- An Oracle database can be logically grouped into smaller logical areas of space known as tablespaces.
- A tablespace can belong to only one database at a time.
- Each tablespace consists of one or more operating system files, which are called datafiles.
- A tablespace may contains one or more segments.
- Tablespaces can be brought online while the database is running.
- Except for the SYSTEM tablespace or a tablespace with an active undo segment, tablespaces can be taken offline, leaving the database running.
- Tablespaces can be switched between read-write and read-only status.

Datafiles (Not a logical structure):

- Each tablespace in an Oracle database consists of one or more files called datafiles. These are physical structures that conform with the operating system on which the Oracle server is running.
- A data file can belong to only one tablespace.
- An Oracle server creates a data file for a tablespace by allocating the specified amount of disk space plus a small amount of overhead.
- The database administrator can change the size of a data file after its creation or can specify that a data file should dynamically grow as objects in the tablespace grow.

Segments:

- A segment is the space allocated for a specific logical storage structure within a tablespace.
- A tablespace may consist of one or more segments.
- A segment cannot span tablespaces; however, a segment can span multiple datafiles that belong to the same tablespace.
- Each segment is made up of one or more extents.

Extents:

Space is allocated to a segment by extents.

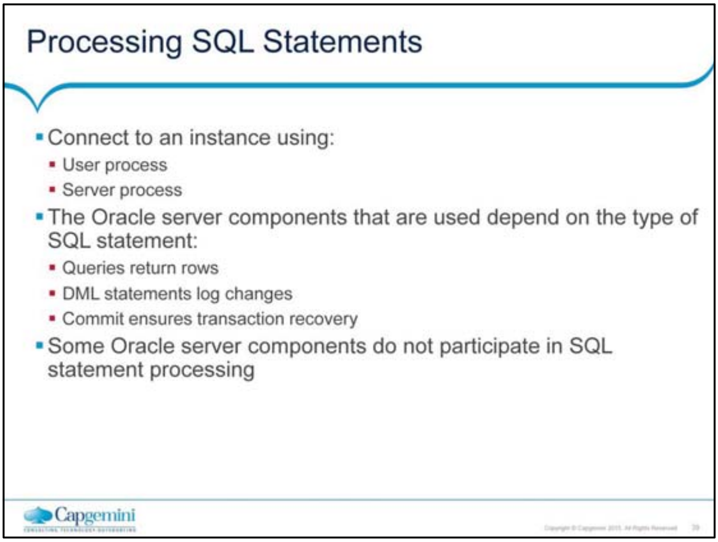
- One or more extents make up a segment.
  - When a segment is created, it consists of at least one extent.
  - As the segment grows, extents get added to the segment.
  - The DBA can manually add extents to a segment.
- An extent is a set of contiguous Oracle blocks.
- An extent cannot span datafiles, and therefore, it must exist in one datafile.

Logical Structure (continued)

Data blocks:

The Oracle server manages the storage space in the datafiles in units called Oracle blocks or data blocks.

- At the finest level of granularity, the data in an Oracle database is stored in data blocks.
- Oracle data blocks are the smallest units of storage that the Oracle server can allocate, read, or write.
- One data block corresponds to one or more operating system blocks allocated from an existing data file.
- The standard data block size for an Oracle database is specified by the DB_BLOCK_SIZE initialization parameter when the database is created.
- The data block size should be a multiple of the operating system block size to avoid unnecessary I/O.
- The maximum data block size is dependent on the operating system.



Processing SQL Statements

- Processing a query:
  - Parse:
    - Search for identical statement
    - Check syntax, object names, and privileges
    - Lock objects used during parse
    - Create and store execution plan
  - Bind: Obtains values for variables
  - Execute: Process statement
  - Fetch: Return rows to user process

Processing SQL Statements (continued)

Processing a DML statement:

Parse: Same as the parse phase used for processing a query.

Bind: Same as the bind phase used for processing a query.

Execute:

If the data and undo blocks are not already in the Database Buffer Cache, the server process reads them from the datafiles into the Database Buffer Cache.

The server process places locks on the rows that are to be modified. The undo block is used to store the before image of the data, so that the DML statements can be rolled back if necessary.

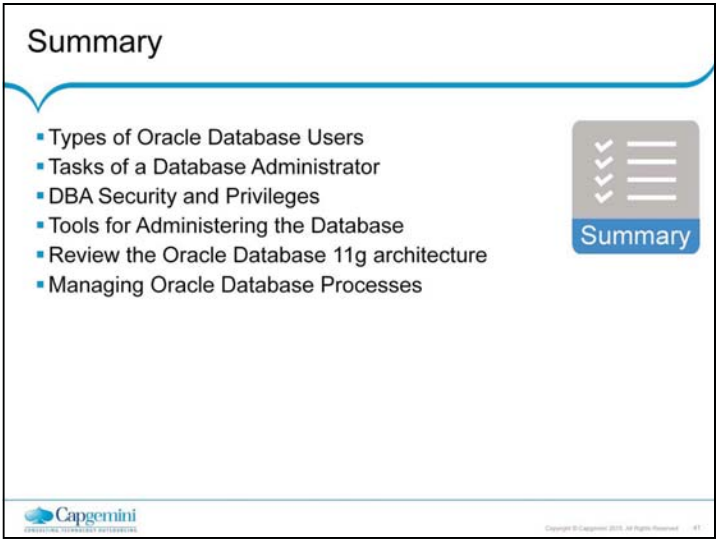
The data blocks record the new values of the data.

The server process records the before image to the undo block and updates the data block. Both of these changes are made in the Database Buffer Cache. Any changed blocks in the Database Buffer Cache are marked as dirty buffers. That is, buffers that are not the same as the corresponding blocks on the disk.

The processing of a DELETE or INSERT command uses similar steps. The before image for a DELETE contains the column values in the deleted row, and the before image of an INSERT contains the row location information.

Processing a DDL statement:

The execution of DDL (Data Definition Language) statements differs from the execution of DML (Data Manipulation Language) statements and queries, because the success of a DDL statement requires write access to the data dictionary. For these statements, parsing actually includes parsing, data dictionary lookup, and execution. Transaction management, session management, and system management SQL statements are processed using the parse and execute stages. To re-execute them, simply perform another execute.



Add the notes here.

Oracle 11g DBA Fundamentals Overview

Lesson 02: Creating an Oracle Database

Lesson Objectives

- Deciding How to Create an Oracle Database
- Manually Creating an Oracle Database
- Understanding the CREATE DATABASE Statement
- Initialization Parameters and Database Creation
- Dropping a Database
- Managing Initialization Parameters Using a Server Parameter File
- Viewing Information About the Database
  - Using Data Dictionaries
  - Using EM
- Using DBCA

Capgemini

Deciding How to Create an Oracle Database

- Use the Database Configuration Assistant (DBCA)
- Use the CREATE DATABASE statement
- Upgrade an existing database

Manually Creating an Oracle Database

- Step 1: Decide on Your Instance Identifier (SID)
- Step 2: Establish the Database Administrator Authentication Method
- Step 3: Create the Initialization Parameter File
- Step 4: Connect to the Instance
- Step 5: Create a Server Parameter File (Recommended)
- Step 6: Start the Instance
- Step 7: Issue the CREATE DATABASE Statement
- Step 8: Create Additional Tablespaces
- Step 9: Run Scripts to Build Data Dictionary Views
- Step 10: Run Scripts to Install Additional Options (Optional)
- Step 11: Back Up the Database.

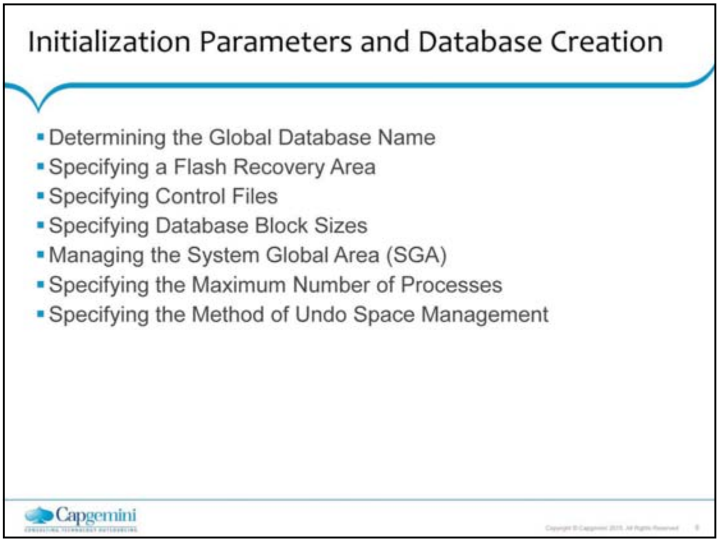
Add the notes here.

Understanding the CREATE DATABASE Statement

- Creates the datafiles for the database
- Creates the control files for the database
- Creates the redo log files for the database and establishes the ARCHIVELOG mode.
- Creates the SYSTEM tablespace and the SYSTEM rollback segment
- Creates the SYSAUX tablespace
- Creates the data dictionary
- Sets the character set that stores data in the database
- Sets the database time zone
- Mounts and opens the database for use

Capgemini

```
CREATE DATABASE rajita
MAXLOGFILES 5
   MAXLOGMEMBERS 5
   MAXLOGHISTORY 1
   MAXDATAFILES 100
   MAXINSTANCES 1
   CHARACTER SET US7ASCII
   NATIONAL CHARACTER SET AL16UTF16
   DATAFILE 'F:\rajita\data/system01.dbf' SIZE 325M REUSE
   EXTENT MANAGEMENT LOCAL
   SYSAUX DATAFILE 'F:\rajita\data\sysaux01.dbf' SIZE 325M REUSE
   DEFAULT TABLESPACE tbs_1
   DEFAULT TEMPORARY TABLESPACE tempts1
      TEMPFILE 'F:\rajita\data\temp01.dbf'
      SIZE 20M REUSE
   UNDO TABLESPACE ts_undo
      DATAFILE 'F:\rajita\data\undotbs01.dbf'
      SIZE 200M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED
   LOGFILE GROUP 1 ('F:\rajita\data\redo01.log') SIZE 100M,
         GROUP 2 ('F:\rajita\data\redo02.log') SIZE 100M,
         GROUP 3 ('F:\rajita\data\redo03.log') SIZE 100M
```



Add the notes here.

Dropping a Database

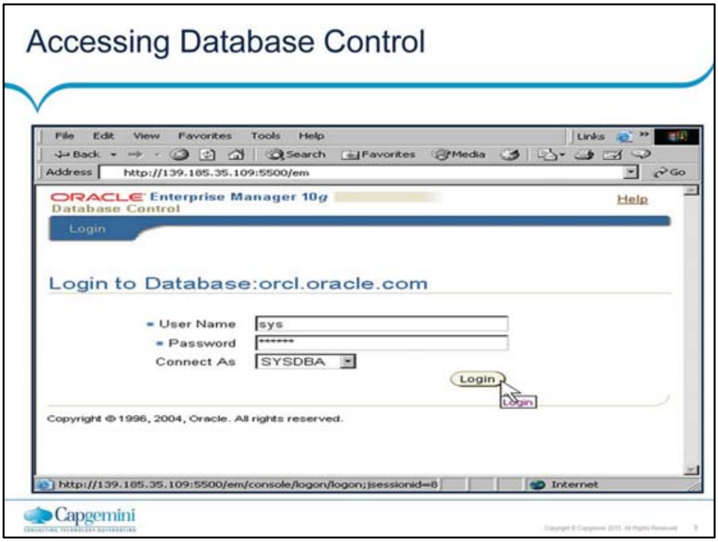
- To use the DROP DATABASE statement successfully, all of the following conditions must apply:
  - **The database must be mounted and closed**
  - **The database must be mounted exclusively--not in shared mode**
  - **The database must be mounted as RESTRICTED**
- An example of this statement is:
  - **DROP DATABASE;**

Capgemini

Managing Initialization Parameters Using a Server Parameter File

- What Is a Server Parameter File?
- Migrating to a Server Parameter File
- Creating a Server Parameter File
- The SPFILE Initialization Parameter
- Managing Initialization Parameters Using a Server Parameter File
- Using ALTER SYSTEM to Change Initialization Parameter Values
- Exporting the Server Parameter File
- Backing Up the Server Parameter File
- Errors and Recovery for the Server Parameter File
- Viewing Parameter Settings

Capgemini



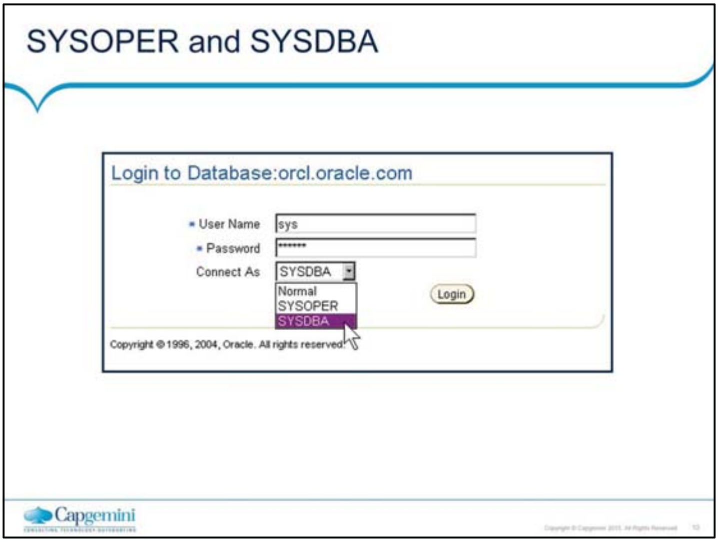
Accessing Database Control

Open your Web browser and enter the following URL (the default port is 5500):

http://hostname:portnumber/em

If the database is up, Enterprise Manager displays the Database Control Login page. Log in to the database using a username that is authorized to access Database Control. This initially will be SYS, SYSMAN or SYSTEM. Use the password you specified for the account during the database installation.

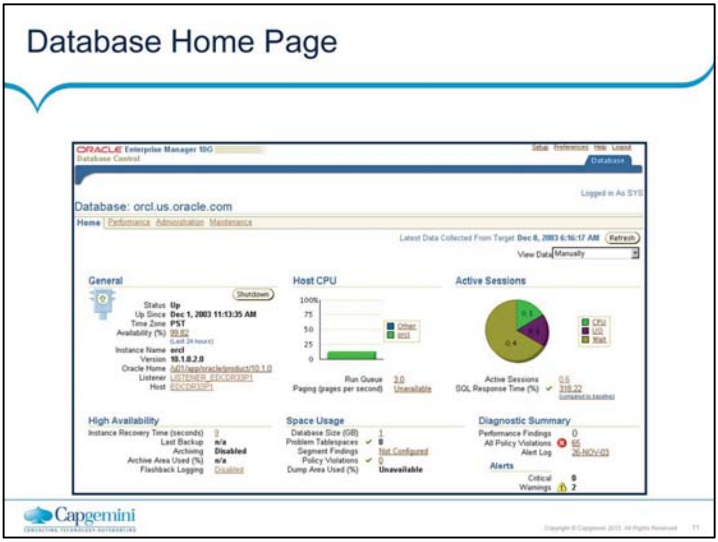
If the database is down and needs to be started, Enterprise Manager displays the Startup/Shutdown and Perform Recovery page. If this is the case, click the Startup/Shutdown button. You are then prompted for the host and target database login usernames and passwords, which you must enter. For the database user and password, use SYS and the password you specified during installation. Click OK to start the database. In the Confirmation page, click YES to start the database in open mode.



SYSOPER and SYSDBA

SYSOPER: Is a special database administration role that permits a database administrator to perform STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG, and RECOVER, and includes RESTRICTED SESSION privileges. When you connect with SYSDBA privileges, you are in the schema owned by SYS.

SYSDBA: Is a special database administration role that contains every system privilege with ADMIN OPTION and SYSOPER system privileges. SYSDBA also permits CREATE DATABASE actions and incomplete recovery. When you connect as SYSOPER, you are in the public schema. More details on user management will be covered in the lesson titled “Administering Users.”

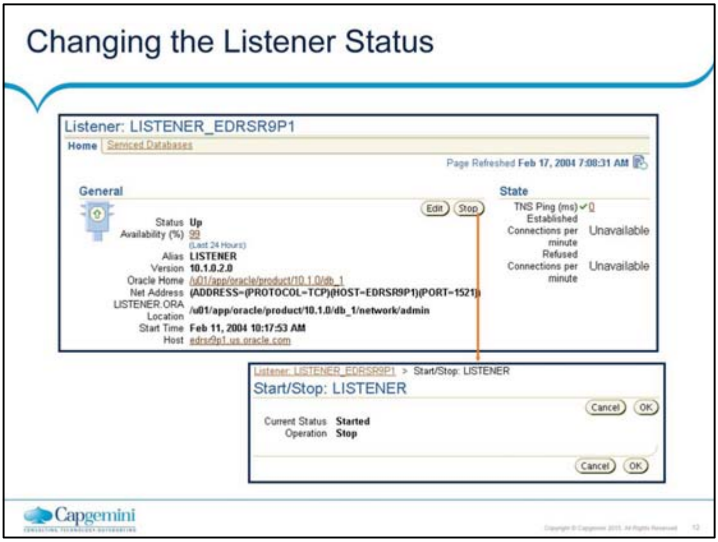


Database Home Page

The property pages across the top of the Database home page enable you to access performance, administration, and maintenance pages for managing your database. The various sections of the Database home page, and related links, provide a wealth of information about the database's environment and health.

To grant management access to other database users, use the following procedure:

1. Start your Web browser and log in to Database Control as the SYS or SYSTEM database user.
2. Click Setup at the top of the Database home page.
3. Click Administrators in the left navigation bar.
4. Click Create to create a new Enterprise Manager user by assigning the management privileges to an existing database user.
5. Click the flashlight icon next to the Name field and select an existing database user from the pop-up window.
6. Enter the password for the selected user and click Finish.



Changing the Listener Status

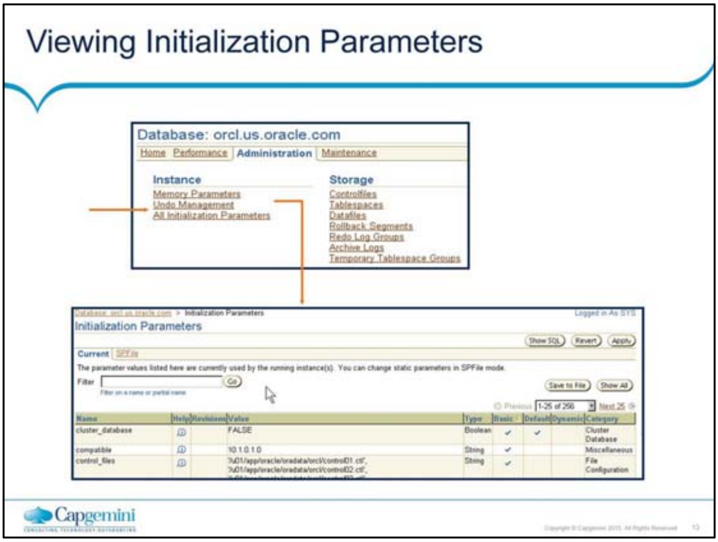
From the Database home page click the listener name to open the Listener home page. Click Stop to stop the listener if it is running or Start to start the listener if it is not running. You must log on to the host as an OS user if you have not done so already. This will be the OS user that starts or stops the listener.

The command line method for starting, stopping, and seeing the status of the listener is:

```
lsnrctl START [listener_name]
lsnrctl STOP [listener_name]
lsnrctl STATUS [listener_name]
```

Where listener_name is the name of the listener defined in the listener.ora file. It is not necessary to identify the listener if you are using the default listener, named LISTENER.

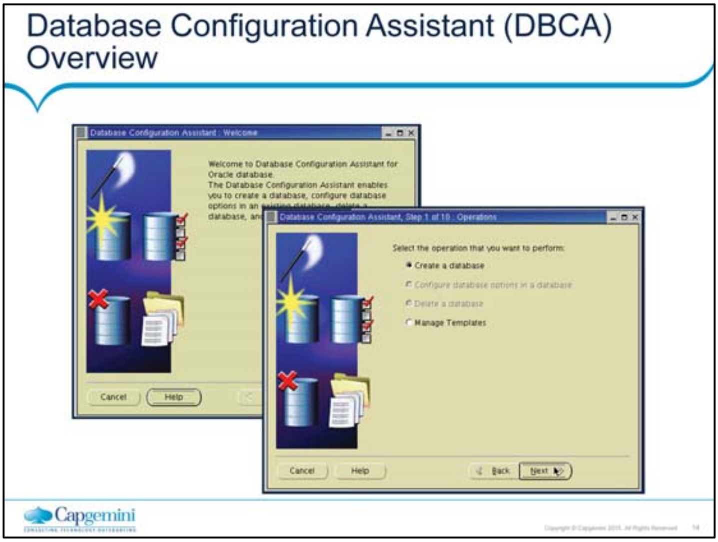
The STATUS command provides basic status information about a listener, including a summary of listener configuration settings, the listening protocol addresses, and a summary of services registered with the listener.



Viewing Initialization Parameters

The Oracle Database provides a number of initialization parameters to optimize its operation in diverse environments. Only a few of these parameters must be explicitly set because the default values are adequate in the majority of cases. There are 28 basic parameters.

The advanced parameters are preserved to allow expert DBAs to adapt the behavior of the Oracle Database to meet unique requirements without overwhelming those who have no such requirements.



Database Configuration Assistant Overview (DBCA)

Database Configuration Assistant (DBCA) enables you 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. A template is a predefined database that you use as a starting point for a new database.

Create a database: If you select this option, you can create a new database or template.

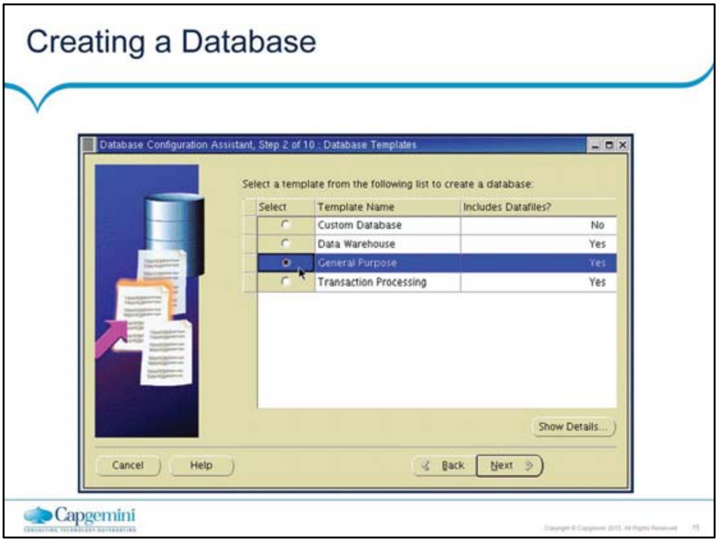
Configure options in a database: If you select the “Change database configuration” option, you can configure installed options that have not previously been configured for use with your database. You can also enable or disable shared server support.

Note: The “Change database configuration” option is not available for Oracle Real Application Clusters.

Delete a database: If you select this option, you can delete all the database files.

Manage Templates: If you select this option, you have three ways to create a template:

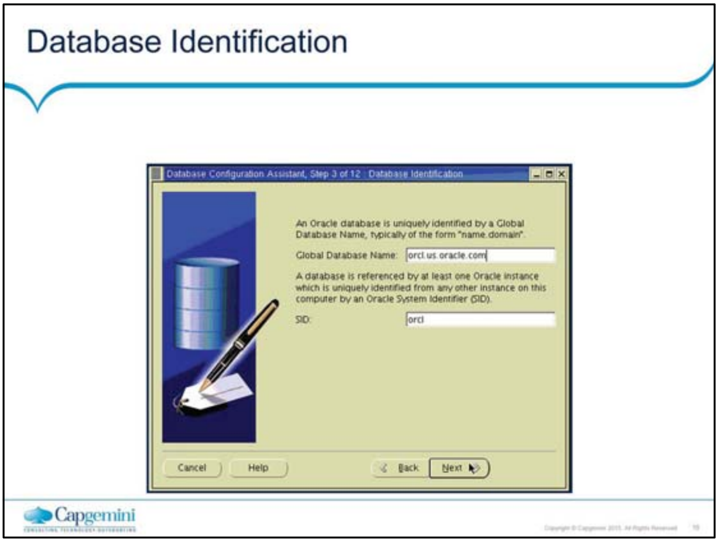
- From an existing template
- From an existing database (structure only)
- From an existing database (structure as well as data)



Creating a Database

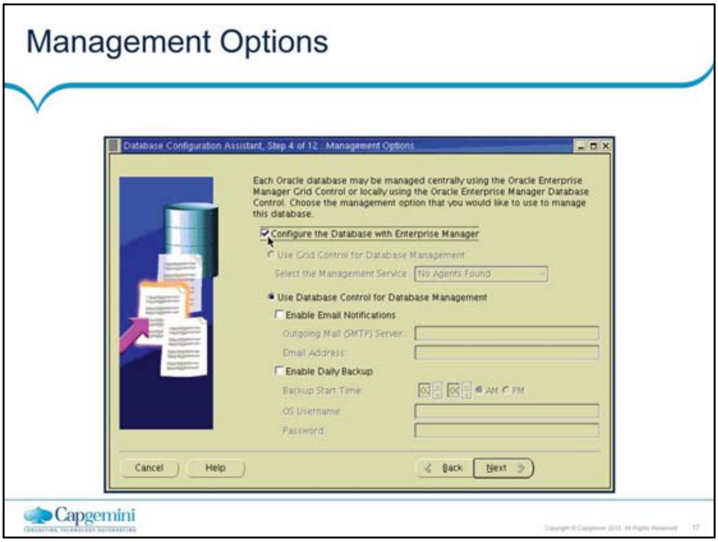
When creating a database with DBCA, you can select one of three predefined databases, or create a custom database. Oracle Corporation ships predefined templates. There are templates for data warehouse, general purpose, and transaction processing databases. The templates contain settings optimized for workload. Click Show Details to see the configuration for each type of database. Choose the template suited to the type of workload your database will support. If you are not sure, select the default General Purpose template.

For more complex environments, you may want to select the Custom Database option. This results in a more extensive interview and takes longer to create your database, because a database creation script must be run.



Database Identification

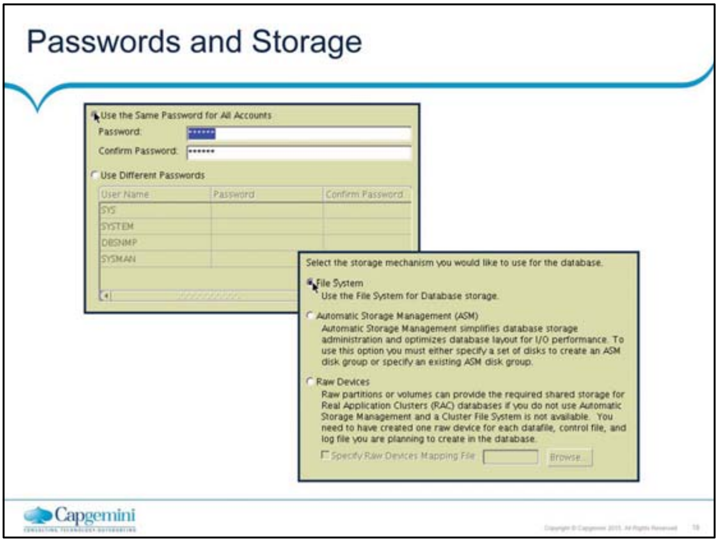
Enter the Global Database Name, in the form database_name.domain_name and SID (Oracle system identifier). The SID defaults to the database name and uniquely identifies the instance that runs the database. It is important to understand that the SID is the name of the instance that will connect to a database and not necessarily the name of the database. An instance and the database the instance connects to need not have the same name, although it is convenient. With Real Application Clusters, multiple instances open the same database and the SIDs for each instance will be different.



Management Options

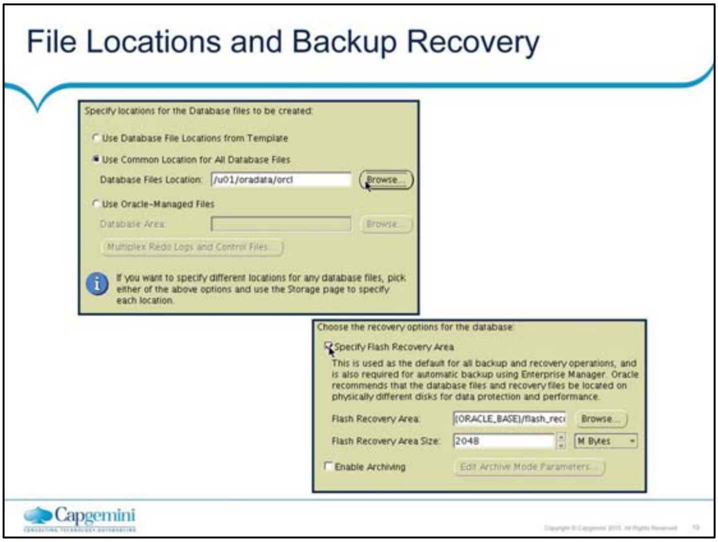
Use this page to set up your database so it can be managed with Oracle Enterprise Manager, which provides web-based management tools for individual databases, as well as central management tools for managing your entire Oracle environment. To use Enterprise Manager, select Configure the Database with Enterprise Manager.

If the Oracle Management Agent has been installed on your host computer, then you are given the option of selecting central management by selecting Use Grid Control for Database Management. If you select this type of management, you must also indicate which management service to use in the drop-down menu. Otherwise, select Use Database Control for Database Management to manage your database individually. If you choose this option, you can additionally enable Email Notifications and Enable Daily Backup. Click Help for more information about these options.



Passwords and Storage

- Database schema passwords: Provide passwords for the administrative users, SYS, SYSTEM, SYSMAN and DBSNMP. You can provide a password for each one separately or provide one password for all.
- File Storage Options
  - File System: This stores files in your OS-configured file system.
  - Automatic Storage Management (ASM): Automatic Storage Management files are created and managed automatically, and you get the additional benefits of features such as mirroring and striping. For details on how to set up ASM, see the Oracle Database Administrator's Guide.
  - Raw Devices (partitions): These are disk partitions without a file system on them. Generally you should use these only if you are very familiar with the use of raw partitions already. Check your OS documentation for details on setting up and maintaining raw partitions.



File Locations and Backup Recovery

File Locations

Use Database File Locations from Template: Selecting this option instructs the DBCA to use the directory information as specified in the template. You will have an opportunity later to make modifications to database file names and locations.

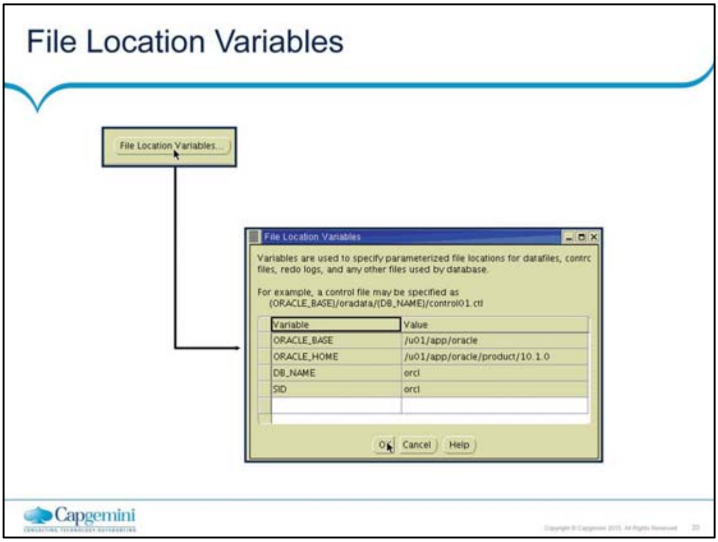
Use Common Location for All Database Files: This option requires you to specify a new common area for all your database files. You will have an opportunity later to make modifications to database file names and locations on the Storage page.

Use Oracle-Managed Files: Select this option to eliminate the need for you, the DBA, to directly manage operating system files that an Oracle database comprises. You must provide the path to the database area. For more details on Oracle Managed Files see the Database Administrator’s Guide.

Backup and Recovery Options

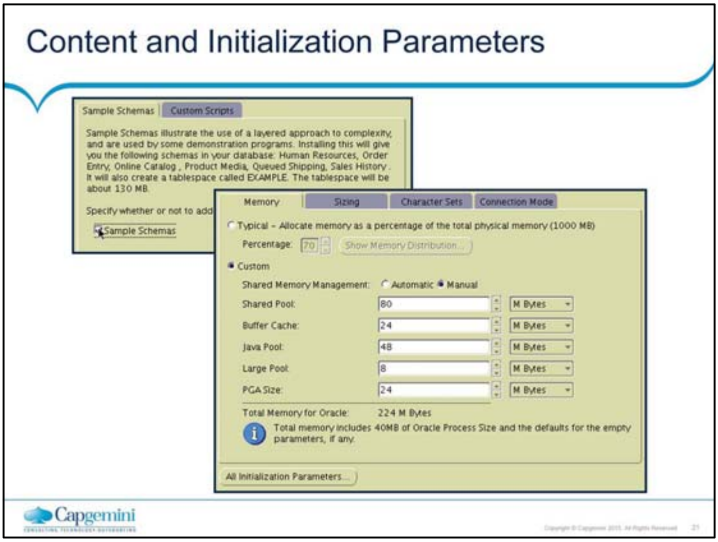
A flash recovery area is a location in which Oracle can store and manage files related to backup and recovery. For details on setting up and sizing the flash recovery area, see the Oracle Database Backup and Recovery Basics guide.

Enabling archiving puts the database in archive log mode at creation time. Archiving will be covered more detail in the lesson titled “Backup and Recovery Concepts.”



File Location Variables

On several pages you can click File Location Variables to open a page that shows you the definition of defined variables. These variables are used in the path definition for files of the database. You cannot change the values while in DBCA. If you need these values modified, you must exit DBCA, change them in the OS environment, and then restart DBCA.



Content and Initialization Parameters

Sample schemas: The sample schemas are a set of schemas used for demonstrations and training.

Custom scripts: Here you can specify any scripts you want run at creation time.

Initialization parameters: The four tabs can be used to set the most common parameters, and by clicking All Initialization Parameters you can view and set all the parameters.

- Memory: This allocates the memory used by the SGA and each PGA of the user processes.
- Sizing: Here you can set the block size, but if using a template the block size cannot be changed. You can also set the maximum number of OS processes that are allowed to connect to the instance.
- Character sets: Here you set the default character set for the database and the national character set. The default character set is used for most data types in the database. The NCHAR, NVARCHAR2, and NCLOB data types support Unicode data only, which is the national character set option. You can use either the AL32UTF8 or the AL16UTF16 character set. For more information on choosing a character set refer to Globalization Support Guide.

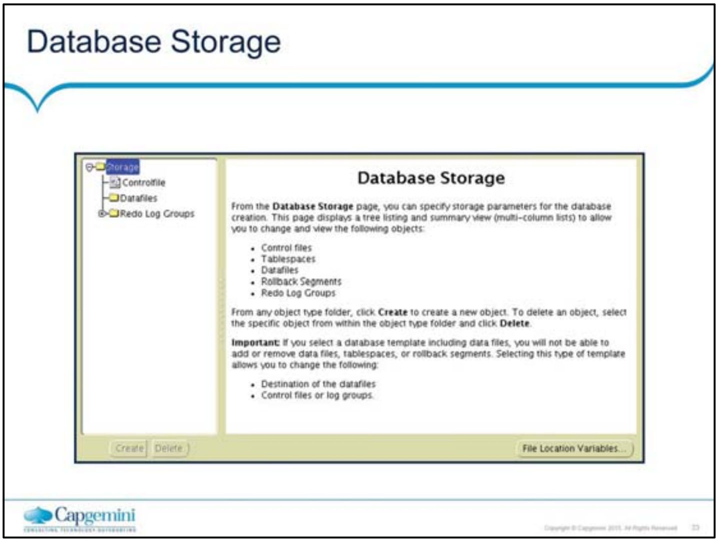
Content and Initialization Parameters Full Notes Page

Content and Initialization Parameters (continued)

Connection mode: Oracle Database creates server processes to handle the requests of user processes connected to an instance. A server process can be either of the following:

- A dedicated server process, where one server process services only one user process
- A shared server process, where a server process can service multiple user processes

Your database is always enabled to allow dedicated server processes, but you must specifically configure and enable shared server by setting one or more initialization parameters. Using Oracle Shared Servers will be discussed in a later lesson. You can also refer to the Database Administrator’s Guide.

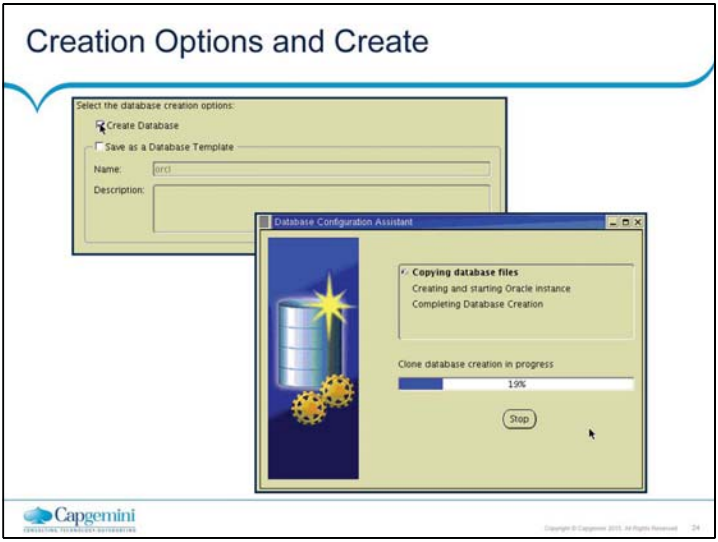


Database Storage

On this page you can see the storage settings for the control files, data files, and online redo log files. When using a template you cannot add any data files to the database, but you can add more control files and online redo log files.

Multiple control files are all maintained such that each is an exact copy of the others. DBCA automatically multiplexes the control file across three files. You can add more if you want.

Redo logs work in groups and should also be multiplexed. DBCA doesn’t automatically multiplex the online redo log files. Each file in a log group is an exact copy of the other members in the group. You can add in more members per group now or add them at anytime after creation of the database.

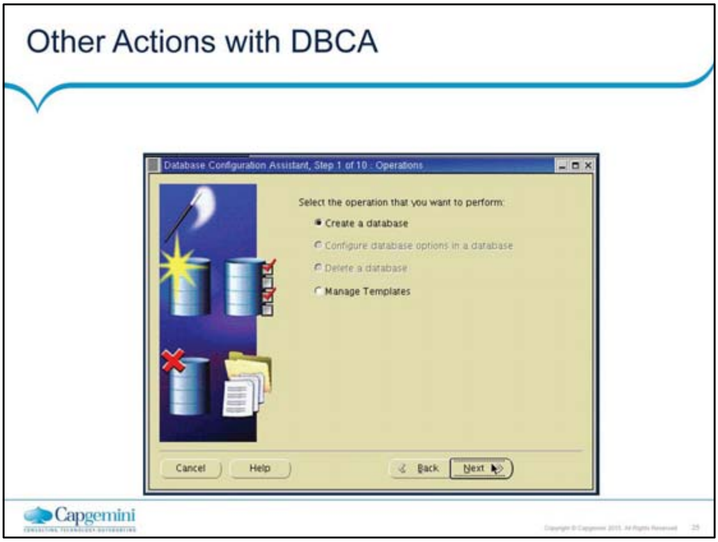


Creation Options and Create

You have the option of saving the database that you have defined as a template. This template can be used later to create databases with all the options you have defined.

After you click Next on the Creation Options page, a Confirmation page appears where you can review all the options taken. This is the last chance to make a change before the creation process starts. You can also save the Confirmation page as an HTML file to review later. After you click OK on the Confirmation page, the database creation starts.

At the end of the installation you will see a page presenting you the opportunity to unlock accounts created and change passwords if you desire. Click Password Management to unlock accounts and change password. Click Ok when don’t managing the accounts, then click Exit to DBCA.



Other Actions with DBCA

- Configure database options in a database: This allows you to add in options to an existing database.
- Delete a database: This permanently removes a database from your system.
- Manage Templates: This allows you to perform the following with templates.
  - Create a Template:
    - From an existing Template: Uses and existing template as a starting point to build a new template. Many of the pages are the same as those in the create database process.
    - From an existing database (structure only): This requires logging in to an existing database and uses its structure as a starting point.
    - From an existing database (structure as well as data):This requires logging into an existing database and uses it's structure as a base line; however it captures the data files as well.
  - Delete a database template: This permanently removes a database template from your system

Practice 3: Creating an Oracle Database

- This practice covers creating an Oracle database by using DBCA.

Practice 3 - Creating an Oracle Database Full Notes Page

Practice 3: Create an Oracle Database

Your IT manager returns from a meeting with a few of the users that will be using the new system you are going to support. They want a second database for storage of historical data.

Using DBCA you will create a database using the General Purpose template with the following information:

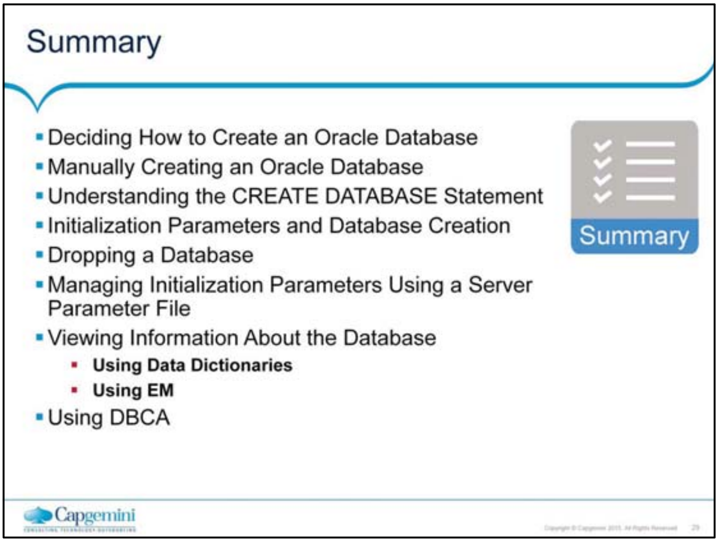
- Set the global database name hist.oracle.com and the SID to hist.
- Set the passwords to oracle.
- For the storage options use File System.
- Use Flash Recovery area, accept the default size and location, disable the backups.

After you create the database the users decided that they don't need to track the historical data. Drop the hist database

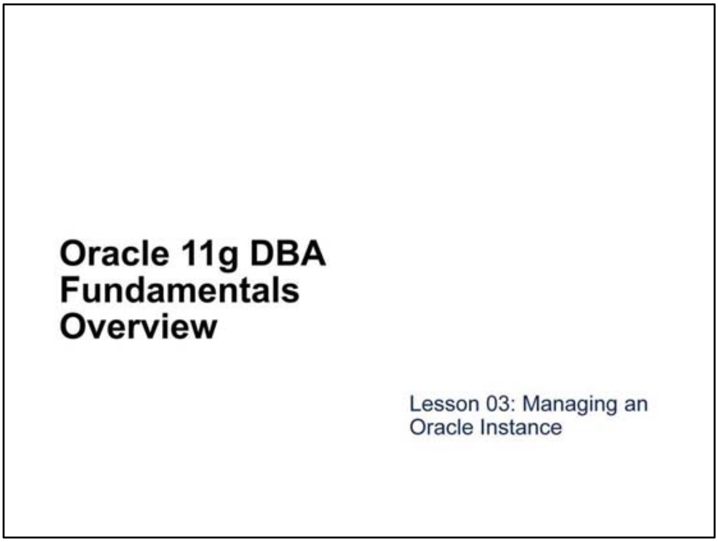
Lab: Creating an Oracle Database

- This practice covers creating an Oracle database manually.

Capgemini



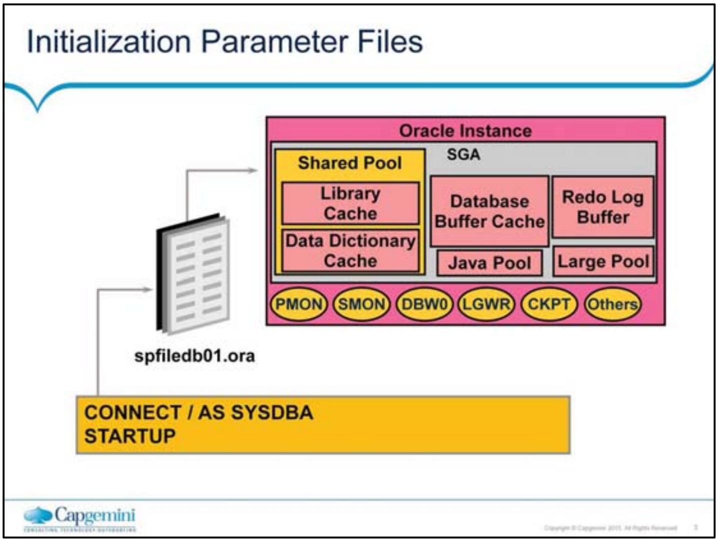
Add the notes here.


Oracle 11g DBA
Fundamentals
Overview
Lesson 03: Managing an
Oracle Instance

Objectives

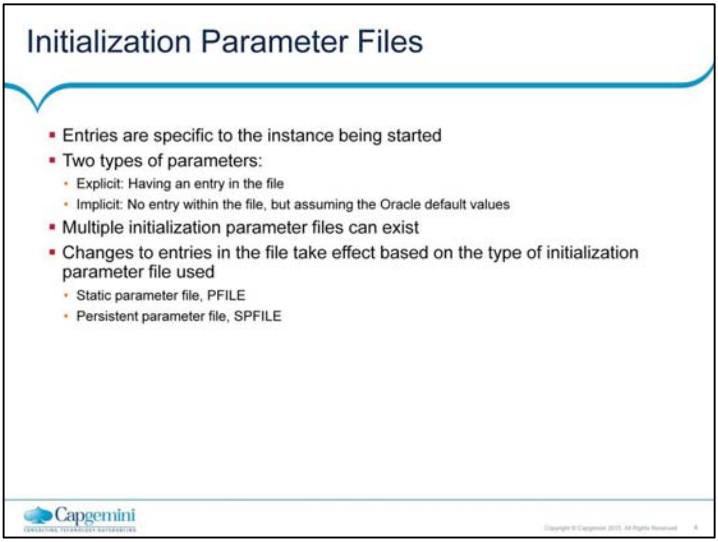
- After completing this lesson, you should be able to do the following:
  - Create and manage initialization parameter files
  - Start up and shut down an instance
  - Monitor and use diagnostic files

Capgemini



Initialization Parameter Files

To start an instance, the Oracle server must read the initialization parameter file.



Initialization Parameter Files

- To start an instance, the Oracle server reads the initialization parameter file.
- Two types of initialization parameter files exist:
  - Static parameter file, PFILE, commonly referred to as initSID.ora.
  - Persistent parameter file, SPFILE, commonly referred to as spfileSID.ora.
- Initialization parameter file contents:
  - A list of instance parameters
  - The name of the database the instance is associated with
  - Allocations for memory structures of the System Global Area (SGA)
  - What to do with filled online redo log files
  - The names and locations of control files
  - Information about undo segments
- Multiple initialization parameter files can exist for an instance in order to optimize performance in different situations.

Initialization Parameter Files

Using Oracle Enterprise Manager to View Initialization Parameters

From the OEM Console:

1. Navigate to Databases > Instance > Configuration.

Select All Initialization Parameters from the General page.

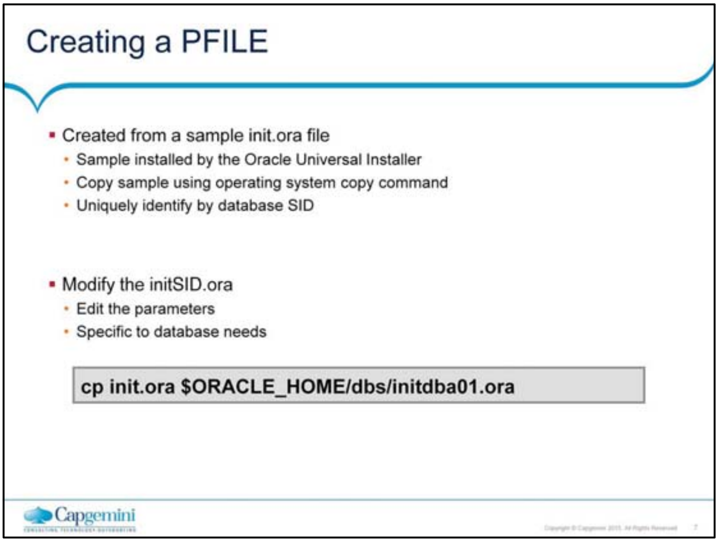
PFILE initSID.ora

- Text file
- Modified with an operating system editor
- Modifications made manually
- Changes take effect on the next startup
- Only opened during instance startup
- Default location is $ORACLE_HOME/dbs

PFILE

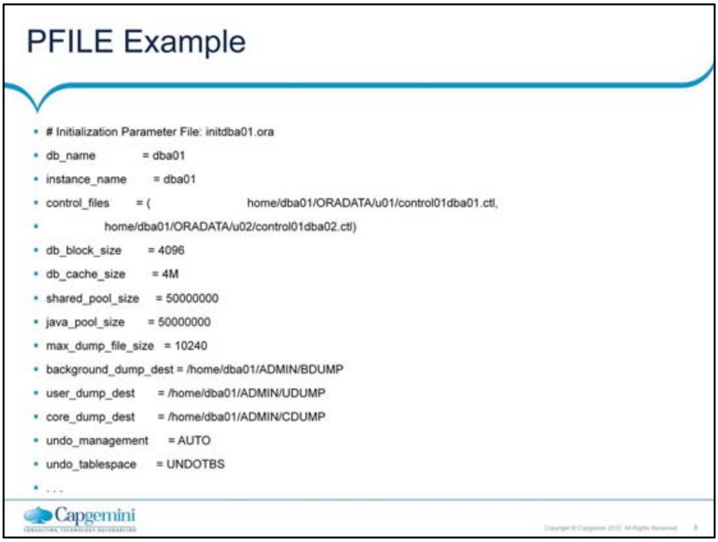
The PFILE is a text file that can be maintained using a standard operating system editor. The PFILE is read only during instance startup. If the file is modified, the instance must be shut down and restarted in order to make the new parameter values effective.

By default, located in the $ORACLE_HOME/dbs directory and named initSID.ora.



Creating a PFILE

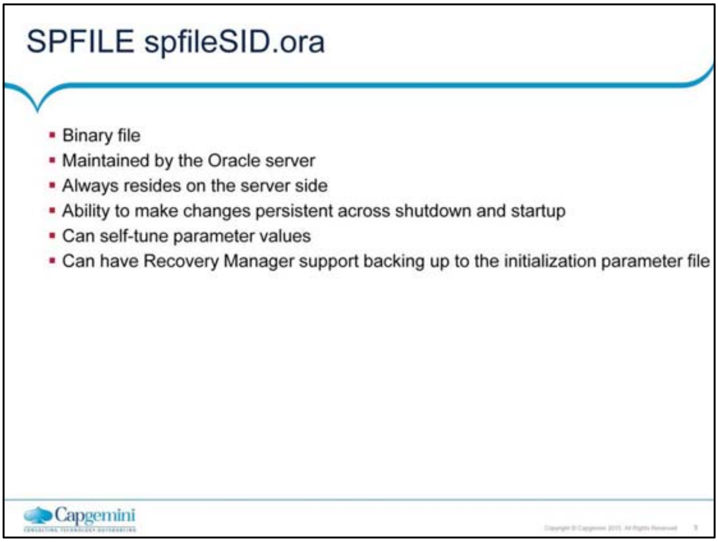
A sample init.ora file is created by the Universal Installer during installation. This sample init.ora file can be used to create an instance-specific initSID.ora. A text editor can be used to modify the parameters within the initSID.ora file.



PFILE Example

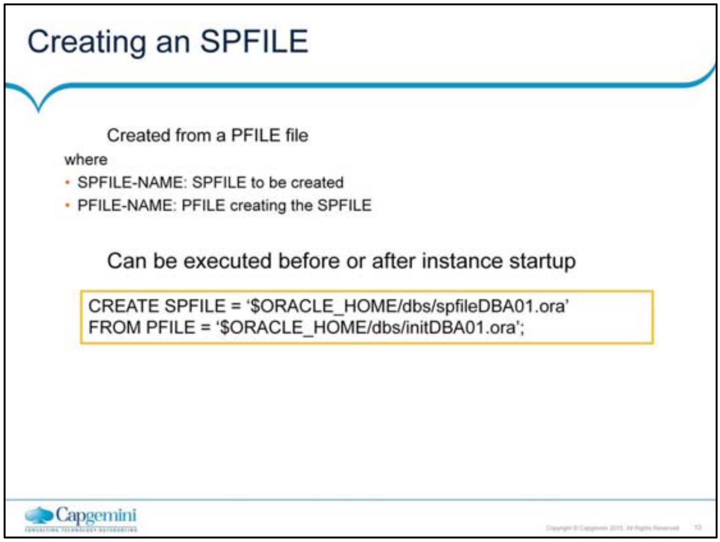
Specify the values in the following format: keyword=value.
The server has a default value for each parameter. This value may be operating system dependent, depending on the parameter.
Parameters can be specified in any order, although there are some exceptions.
Comment lines begin with the # symbol.
Enclose parameters in double quotation marks to include character literals.
Additional files can be included with the keyword IFILE.
If case is significant for the operating system, then it is also significant in filenames.
Multiple values are enclosed in parentheses and separated by commas.

Note: Develop a standard for listing parameters; either list them alphabetically or group them by functionality. The PFILE varies from instance to instance and does not necessarily look like the preceding example.



SPFILE

An SPFILE, new to Oracle9i, is a binary file. The file is not meant to be modified manually and must always reside on the server side. Once the file is created it is maintained by the Oracle server. If modified manually, the SPFILE is rendered useless. The SPFILE provides the ability to make changes to the database persistent across shutdown and startup. It also provides the ability to self-tune parameter values, which are recorded in the file. RMAN support for backing up the initialization parameter file is possible because the SPFILE resides on the server side. By default, the file is located in $ORACLE_HOME/dbs and has a default name in the format of spfileSID.ora.



Creating an SPFILE

An SPFILE is created from a PFILE file using the CREATE SPFILE command. This command requires the SYSDBA privilege to execute. This command can be executed before or after instance startup.

```
SQL> CREATE SPFILE [='SPFILE-NAME']
  2 FROM PFILE[='PFILE-NAME']
```

where:

SPFILE-NAME: Name of the SPFILE to be created

PFILE-NAME: Name of the PFILE being used to create the SPFILE. The PFILE must be available on the server side

If the SPFILE-NAME and PFILE-NAME are not included in the syntax, Oracle will use the default PFILE to generate an SPFILE with a system generated name.

```
SQL> CREATE SPFILE FROM PFILE;
```

Creating an SPFILE (continued)

Exporting an SPFILE:

The contents of an SPFILE can be exported into a PFILE.

```
SQL> CREATE PFILE FROM SPFILE;
```

The PFILE is created as a text file on the server side. This command can be executed either before or after instance startup. This provides an easy way to view the SPFILE and make modifications by:

- Exporting the SPFILE to a PFILE
- Editing the PFILE
- Recreating the SPFILE from the edited PFILE

Exporting an SPFILE to a PFILE can also serve as another alternative to creating a backup of the persistent parameter file.

V$SPPARAMETER

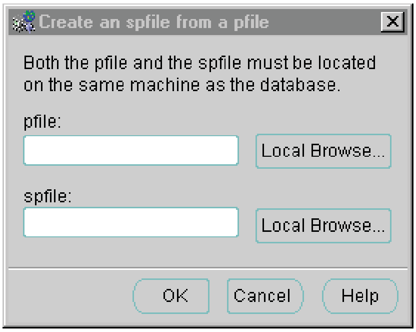
As shown above, there are several options for viewing the parameter settings within the SPFILE. V$SPPARAMETER is another source for presenting and viewing contents of the SPFILE.

Creating an SPFILE

Using Oracle Enterprise Manager to Create an SPFILE

From the OEM Console:

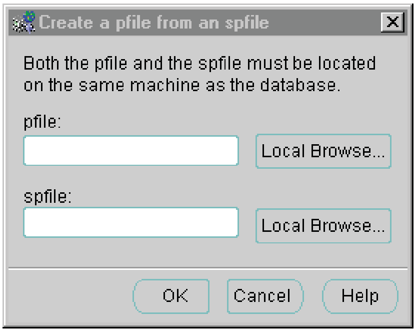
1. Select Object > Create spfile from the main menu.



Using Oracle Enterprise Manager to Export an SPFILE

From the OEM Console:

Select Object > Create pfile from the main menu.



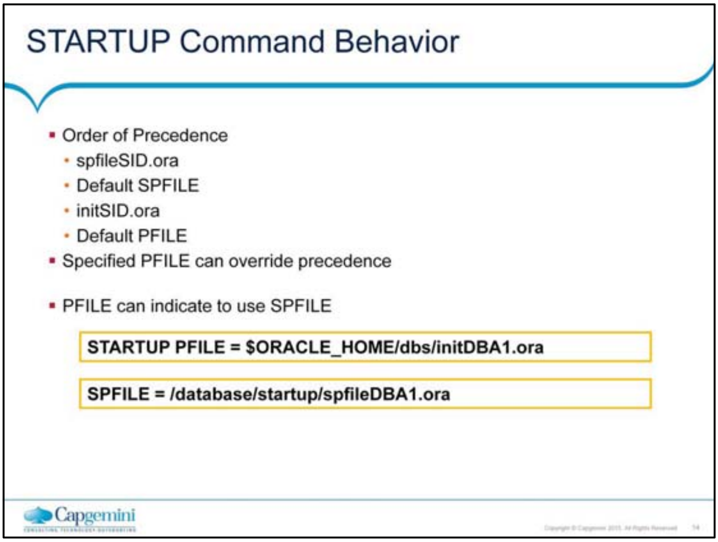
SPFILE Example

- *.background_dump_dest='/home/dba01/ADMIN/BDUMP'
- *.compatible='9.0.0'
- *.control_files='/home/dba01/ORADATA/u01/ctrl01.ctl' *.core_dump_dest='/home/dba01/ADMIN/CDUMP'
- *.db_block_size=4096
- *.db_name='dba01'
- *.db_domain='world'
- *.global_names=TRUE
- *.instance_name='dba01'
- *.remote_login_passwordfile='exclusive'
- *.java_pool_size=50000000'
- *.shared_pool_size=50000000
- *.undo_management='AUTO'
- *.undo_tablespace='UNDOTBS'
- . . .

Capgemini

SPFILE Example

The comments specified on the same lines as a parameter setting in the PFILE are maintained in the SPFILE. All other comments are ignored. Although the text of an SPFILE is easily viewed in UNIX, the SPFILE is binary, and manual modification of the SPFILE will render it unusable. If you need to view the specific contents of an SPFILE or make some modification, export the SPFILE to a PFILE.



STARTUP Command Behavior

Order of precedence:

When the command STARTUP is used, the spfileSID.ora on the server side is used to start up the instance.

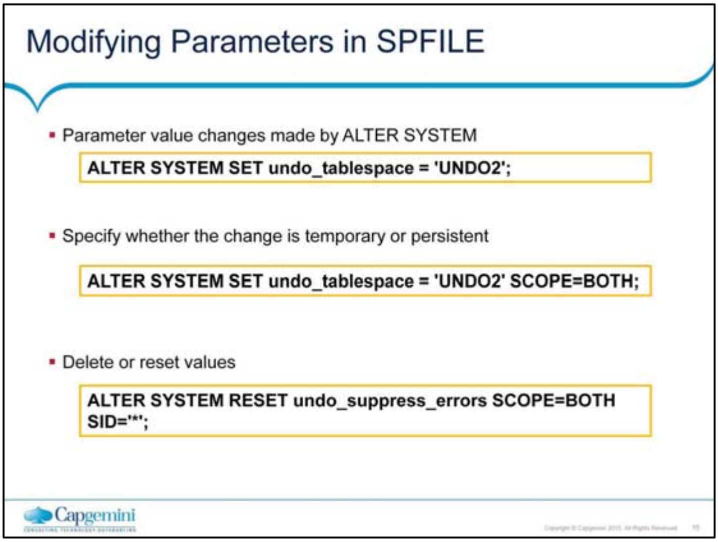
If the spfileSID.ora is not found, the default SPFILE on the server side is used to start the instance.

If the default SPFILE is not found, the initSID.ora on the server side will be used to start the instance.

A specified PFILE can override the use of the default SPFILE to start the instance.

A PFILE can optionally contain a definition to indicate use of an SPFILE. This is the only way to start the instance with an SPFILE in a non-default location. To start the database with an SPFILE not in the default location, SPFILE=<full path and filename> must be placed in the PFILE.

Example: SPFILE=$HOME/ADMIN/PFILE/$ORACLE_SID.ora.



Modifying Parameters in SPFILE

The ALTER SYSTEM SET command is used to change the value of instance parameters.

```
ALTER SYSTEM SET parameter_name = parameter_value
[COMMENT 'text'] [SCOPE = MEMORY|SPFILE|BOTH]
[SID= 'sid'|'*']
```

where

parameter_name: Name of the parameter to be changed
parameter_value: Value the parameter is being changed to
COMMENT: A comment to be added into the SPFILE next to the parameter being altered
SCOPE: Determines if change should be made in memory, SPFILE, or in both areas
MEMORY: Changes the parameter value only in the currently running instance
SPFILE: Changes the parameter value in the SPFILE only
BOTH: Changes the parameter value in the currently running instance and the SPFILE
SID: Identifies the ORACLE_SID for the SPFILE being used
'sid': Specific SID to be used in altering the SPFILE
'*': Uses the default SPFILE

Modifying Parameters in SPFILE (continued)

Example:

```
SQL>  SHOW PARAMETERS undo_suppress_errors
  NAME                  TYPE        VALUE
  --------------------- ----------- -------
  undo_suppress_errors  boolean     FALSE

SQL> ALTER SYSTEM SET undo_suppress_errors = TRUE
  2  COMMENT = ‘temporary testing’ SCOPE=BOTH
  3  SID=‘DBA01’;
SQL> SHOW PARAMETERS undo_suppress_errors
  NAME                  TYPE        VALUE
  --------------------- ----------- -------
  undo_suppress_errors  boolean     TRUE
```

The ALTER SYSTEM RESET command is used to delete or revert to the default value.

SQL> ALTER SYSTEM RESET parameter_name [SCOPE = MEMORY|SPFILE|<u>BOTH</u>] [SID= ‘sid’|’*’]

Example:

```
SQL> ALTER SYSTEM RESET undo_suppress_errors
  2  SCOPE=BOTH SID=‘dba01’;
```

Several ways exist to remove a parameter from the SPFILE:

- Set the parameter back to its default value to simulate deleting using ALTER SYSTEM SET.
- Recreate the SPFILE using CREATE SPFILE FROM PFILE.
- Use ALTER SYSTEM RESET to delete the parameter from the SPFILE.

Modifying Parameters in SPFILE (continued)

Using Oracle Enterprise Manager to Modify the SPFILE Configuration

From the OEM Console:

Navigate to Databases > Instance

Click Configuration.

3. In the General page, click All Initialization Parameters.

4. Modify a parameter in the value column.

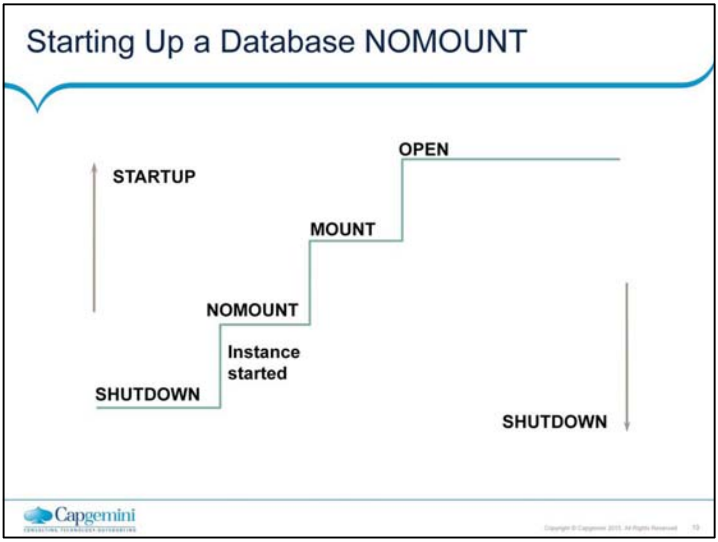
5. Click OK.

Parameters That Should Be Specified in the Initialization Parameter File

Parameter	**Description**
BACKGROUND_DUMP_DEST	Location where background process trace files are written (LGWR, DBWn, and so on). Also the location for the alert log file
COMPATIBLE	Version of the server with which this instance should be compatible
CONTROL_FILES	Names of the control files
DB_CACHE_SIZE	Specifies the size of the cache for standard block size buffers
DB_NAME	Database identifier of eight characters or fewer. This is the only parameter that is required when a new database is created
SHARED_POOL_SIZE	Size in bytes of the shared pool
USER_DUMP_DEST	Location where user debugging trace files are created on behalf of a user process

Note: The default values depend on the version of the Oracle server.
Commonly Modified Parameters

Parameter	**Description**
IFILE	Name of another parameter file to be embedded within the current parameter file. Up to three levels of nesting is possible
LOG_BUFFER	Number of bytes allocated to the redo log buffer in the SGA
MAX_DUMP_FILE_SIZE	Maximum size of the trace files, specified as number of operating system blocks
PROCESSES	Maximum number of operating system processes that can connect simultaneously to this instance
SQL_TRACE	Enables or disables the SQL trace facility for every user session
TIMED_STATISTICS	Enables or disables timing in trace files and in monitor screens



Starting Up a Database

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

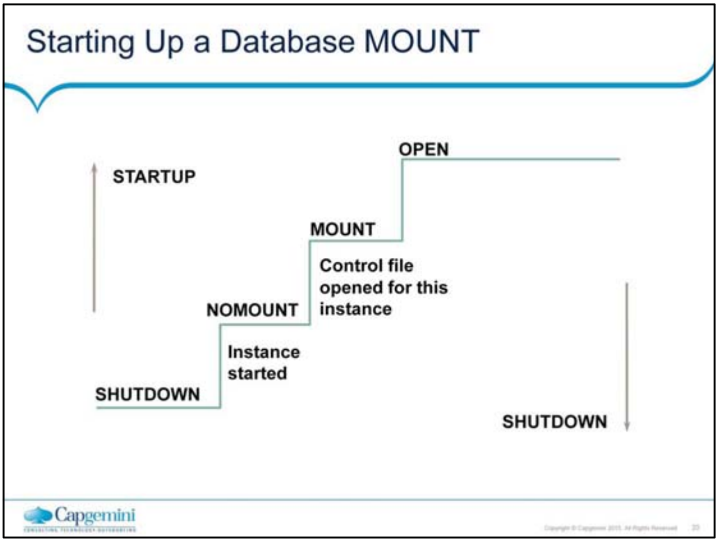
Starting the instance (NOMOUNT):

An instance would be started in the NOMOUNT stage only during database creation or the re-creation of control files.

Starting an instance includes the following tasks:

- Reading the initialization file from $ORACLE_HOME/dbs in the following order:
  - First spfileSID.ora
  - If not found then, spfile.ora
  - If not found then, initSID.ora
  - Specifying the PFILE parameter with STARTUP overrides the default behavior.
- Allocating the SGA
- Starting the background processes
- Opening the alertSID.log file and the trace files

The database must be named with the DB_NAME parameter either in the initialization parameter file or in the STARTUP command.



Starting Up a Database

Mounting the database (MOUNT):

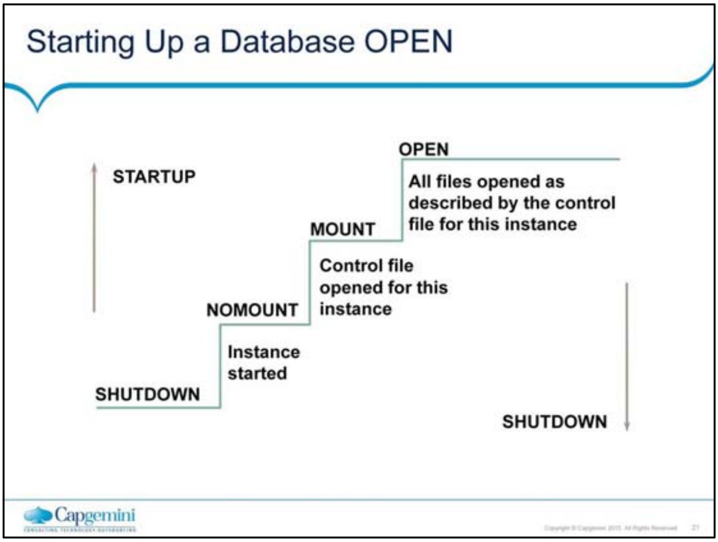
To perform specific maintenance operations, you start an instance and mount a database but do not open the database.

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

- Renaming datafiles
- Enabling and disabling redo log archiving options
- Performing full database recovery

Mounting a database includes the following tasks:

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



Starting Up a Database

Opening the database (OPEN):

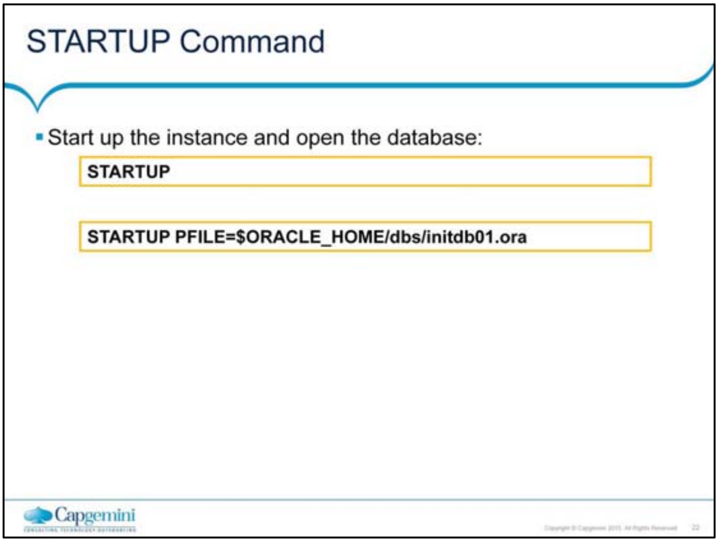
Normal database operation means that an instance is started and the database is mounted and open. With normal database operation, any valid user can connect to the database and perform typical data access operations.

Opening the database includes the following tasks:

- Opening the online datafiles
- Opening the online redo log files

If any of the datafiles 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 the datafiles 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.



STARTUP Command

To start up an instance, use the following command:

STARTUP [FORCE] [RESTRICT] [PFILE=filename]
[OPEN [RECOVER][database]
|MOUNT
|NOMOUNT]

(Note: This is not the complete syntax.)

where:

OPEN: Enables users to access the database
MOUNT: Mounts the database for certain DBA activities but does not provide user access to the database
NOMOUNT: Creates the SGA and starts up the background processes but does not provide access to the database
PFILE=parfile: Enables a nondefault parameter file to be used to configure the instance

Starting Up (continued)

FORCE: Aborts the running instance before performing a normal startup.

RESTRICT: Enables only users with RESTRICTED SESSION privilege to access the database.

RECOVER: Begins media recovery when the database starts.

Automating database startup:

On UNIX:

Automating database startup and shutdown can be controlled by the entries in a special operating system file; for example, oratab in the /var/opt/oracle directory.

Note: Refer to the installation guide of your operating system for more information.

Troubleshooting:

If any errors are encountered while issuing the STARTUP command the SHUTDOWN command must be issued before another STARTUP.

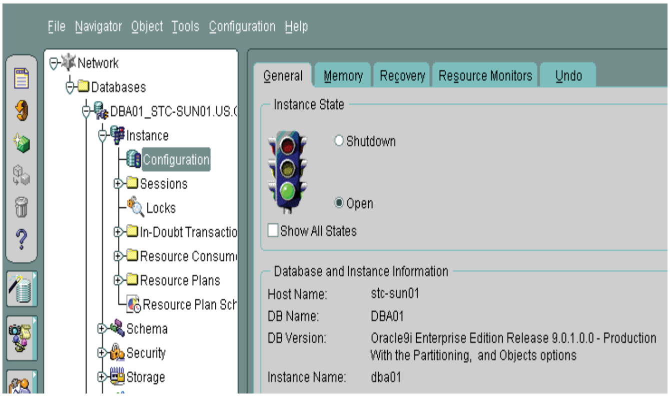
Note: STARTUP and SHUTDOWN commands are SQL*Plus commands, not SQL commands.

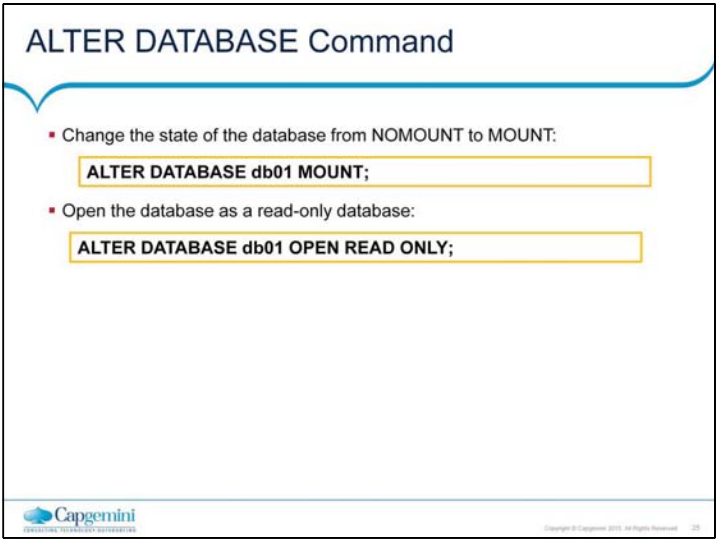
Shutdown Options (continued)

Using Oracle Enterprise Manager to Start Up a Database

From the OEM Console:
Navigate to Databases > Instance
Click Configuration
From the General tab, select the Open option.
Click Apply.

Note: You must be connected to the database with SYSDBA privileges to perform startup.





ALTER DATABASE Command

To move the database from the NOMOUNT to a MOUNT stage or from the MOUNT to an OPEN stage, use the ALTER DATABASE command:

ALTER DATABASE { MOUNT | OPEN }

To prevent data from being modified by user transactions, the database can be opened in read-only mode.

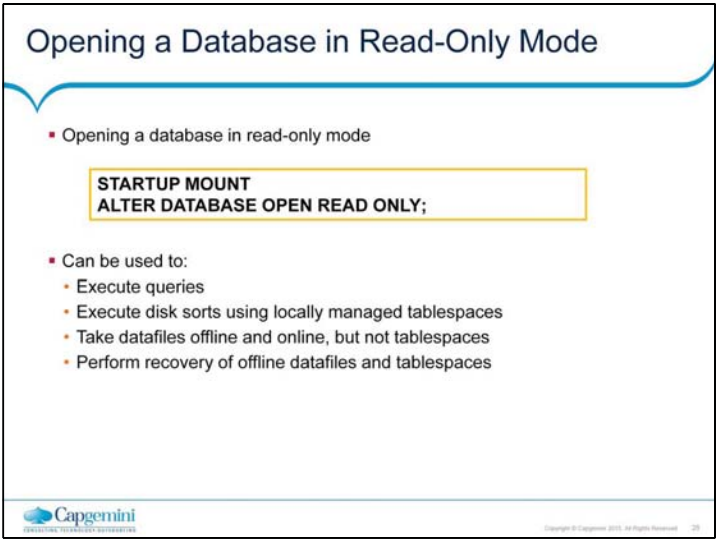
To start up an instance, use the following command:

ALTER DATABASE OPEN [READ WRITE| READ ONLY]

where:

READ WRITE: Opens the database in read-write mode, so that users can generate redo logs.

READ ONLY: Restricts users to read-only transactions, preventing them from generating redo log information.



Opening a Database in Read-Only Mode

A database can be opened as read-only, as long as it is not already open in read-write mode. The feature is especially useful for a standby database to offload query processing from the production database.

If a query needs to use a temporary tablespace, for example, to do disk sorts, the current user must have a locally managed tablespace assigned as the default temporary tablespace; otherwise, the query fails. For user SYS, a locally managed tablespace is required.

Note: Locally managed tablespaces are discussed in a later lesson.

Read-only mode does not restrict database recovery or operations that change the database state without generating redo data. For example, in read-only mode:

- Datafiles can be taken offline and online.
- Recovery of offline datafiles and tablespaces can be performed.

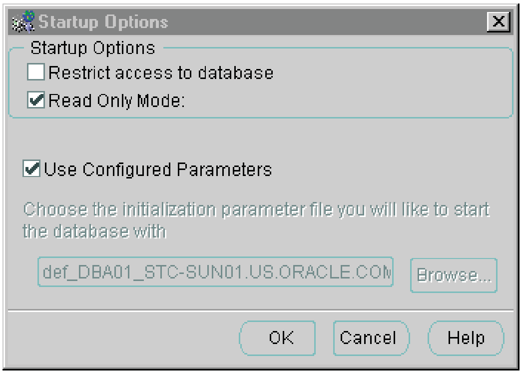
Disk writes to other files, such as control files, operating system audit trails, trace files, and alert log files, can continue in read-only mode.

Shutdown Options (continued)

Using Oracle Enterprise Manager to Start a Database in Read Only Mode

From the OEM Console:
Navigate to Instance > Configuration.
Select the General page.
Under Instance State, select the Shutdown option.
Select Apply.
The Shutdown Options dialog box will appear. Select the Immediate option.
Select OK.
Select Close when processing is complete.
Under Instance State, select the Open option.
Select OK.
The Startup Options dialog box will appear. Select Read Only Mode option.
Select OK.
Click Close when processing complete.

Note: You must be connected to the database with SYSDBA privileges.



Shutting Down the Database

- Shutdown mode:
  - A = ABORT
  - I = IMMEDIATE
  - T = TRANSACTIONAL
  - N = NORMAL

Shutdown Mode	A	I	T	N
Allow new connections	No	No	No	No
Wait until current sessions end	No	No	No	Yes
Wait until current transactions end	No	No	Yes	Yes
Force a checkpoint and close files	No	Yes	Yes	Yes

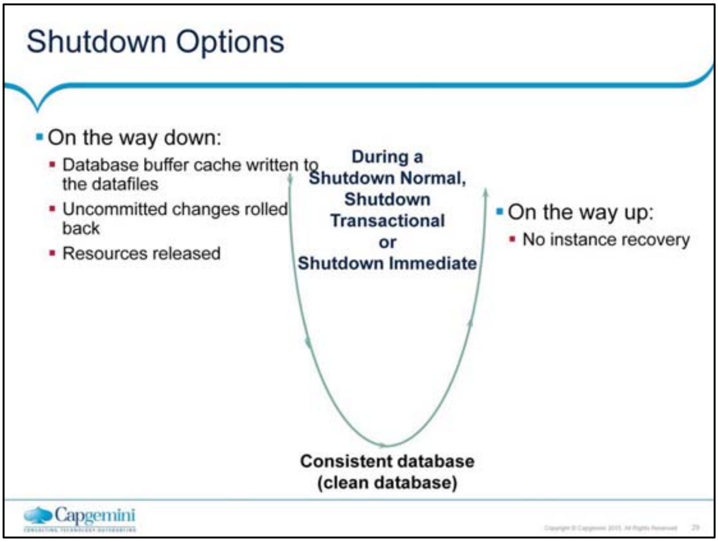
Capgemini

Shutting Down the Database

Shut down the database to make operating system offline backups of all physical structures and to have modified static initialization parameters take effect when restarted.

To shut down an instance you must connect as SYSOPER or SYSDBA and use the following command:

SHUTDOWN [NORMAL | TRANSACTIONAL | IMMEDIATE | ABORT]



Shutdown Options

Shutdown normal:

Normal is the default shutdown mode. 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.
- Oracle closes and dismounts the database before shutting down the instance.
- The next startup does not require an instance recovery.

Shutdown transactional:

A transactional shutdown prevents clients from losing work. A transactional database shutdown proceeds with the following conditions:

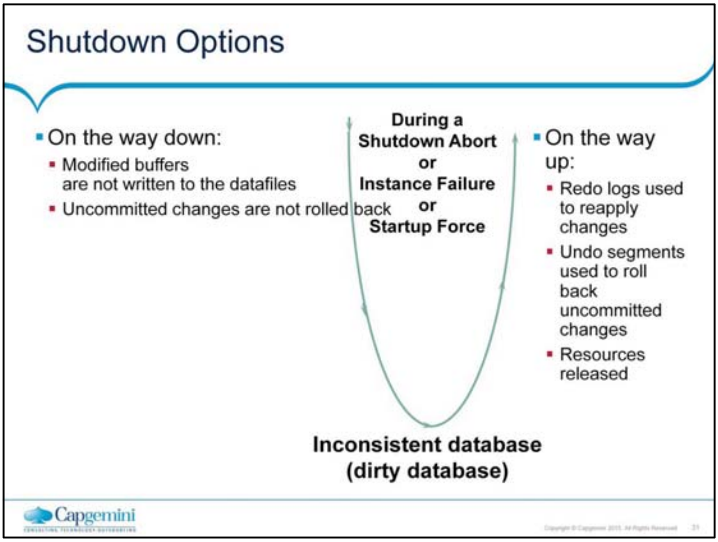
- No client can start a new transaction on this particular instance.
- A client is disconnected when the client ends the transaction that is in progress.
- When all transactions have finished, a shutdown immediately occurs.
- The next startup does not require an instance recovery.

Shutdown Options (continued)

Shutdown immediate:

Immediate database shutdown proceeds with the following conditions:

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



Shutdown Options

Shutdown abort:

If the normal and immediate shutdown options do 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.
- Oracle does not wait for users 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.

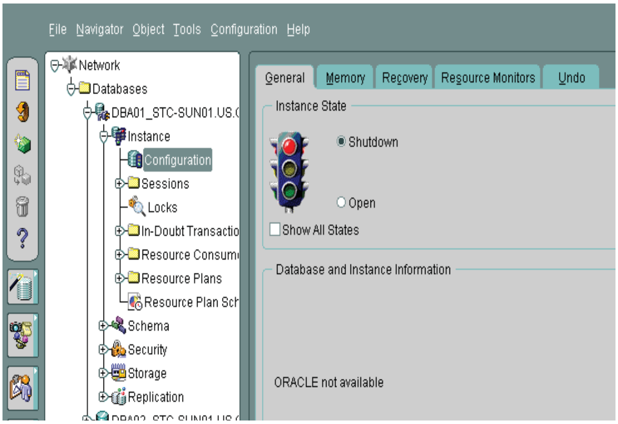
Note: It is not advisable to backup a database that is in an inconsistent state.

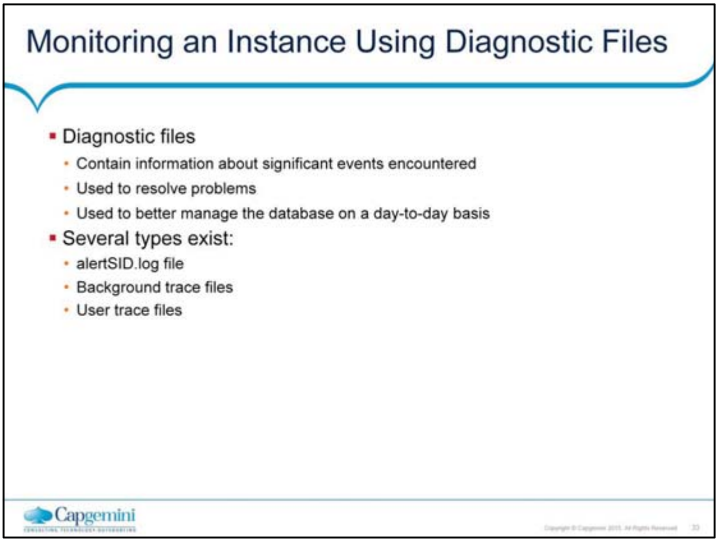
Shutdown Options (continued)

Using Oracle Enterprise Manager to Shut Down a Database

From the OEM Console:
Navigate to Databases > Instance
Click Configuration
From the General tab, select the Open option.
Click Apply.

Note: You must be connected to the database with SYSDBA privileges to perform shutdown.





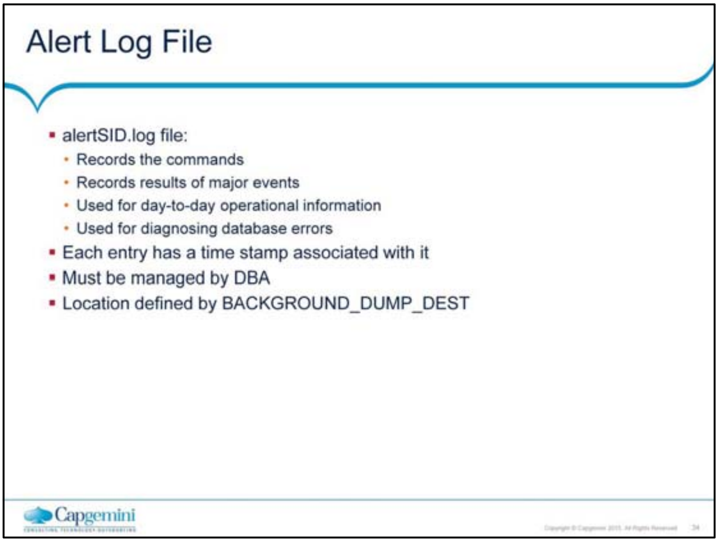
Monitoring an Instance Using Diagnostic Files

Diagnostic files are a means to capture information about the database's activities. They are also useful tools for you when you are managing an instance. Several types exist. The type of diagnostic file created depends on the problem that occurred or the information that is needed to be disseminated.

alertSID.log file: Information for day-to-day operation of the database

Background trace files: Vital information when background processes, such as SMON, PMON, DBWn, and others fail

User trace files: Vital information for fatal user errors or user forced traced files



Alert Log File

Each Oracle Instance has an alert log file. If not already created, it is created during instance startup. The alert log file is managed by you, as it continues to grow while the database continues to work. The alert log file should be the first place you look when diagnosing day-to-day operations or errors. The alert log file also contains pointers to trace files for more detailed information.

The alert log file keeps a record of the following information:

- When the database was started or shut down
- A list of all non-default initialization parameters
- The startup of background processes
- The thread being used by the instance
- The log sequence number LGWR is writing to
- Information regarding a log switch
- Creation of tablespaces and undo segments
- Alter statements that have been issued
- Information regarding error messages such as ORA-600 and extent errors

Alert Log File

The Alert log file consists of a chronological log of messages and errors.

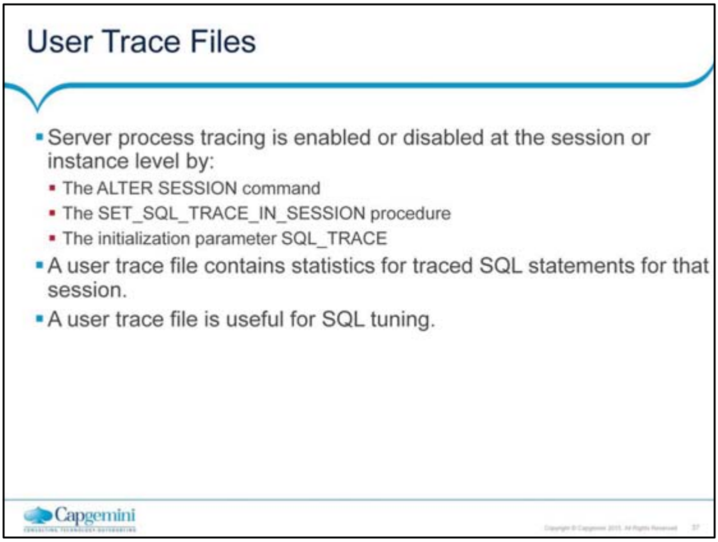
- Check the Alert log file regularly to:
- Detect internal errors (ORA-600) and block corruption errors.
- Monitor database operations.
- View the non default initialization parameters.
- Remove or trim it regularly after checking.

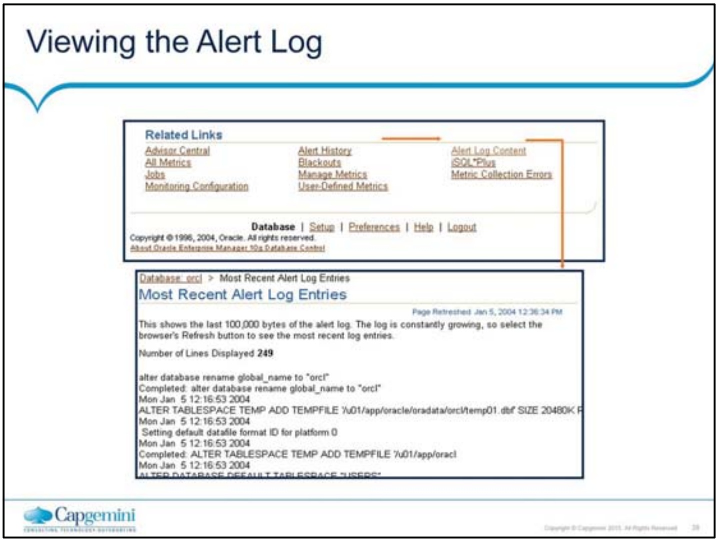
Capgemini

Background Processes Trace Files

- Oracle server dumps information about errors detected by any background process in trace files.
- Oracle support uses these trace files to diagnose and troubleshoot.

Capgemini


User Trace Files
Server process tracing is enabled or disabled at the session or instance level by:
The ALTER SESSION command
The SET_SQL_TRACE_IN_SESSION procedure
The initialization parameter SQL_TRACE
A user trace file contains statistics for traced SQL statements for that session.
A user trace file is useful for SQL tuning.
Capgemini

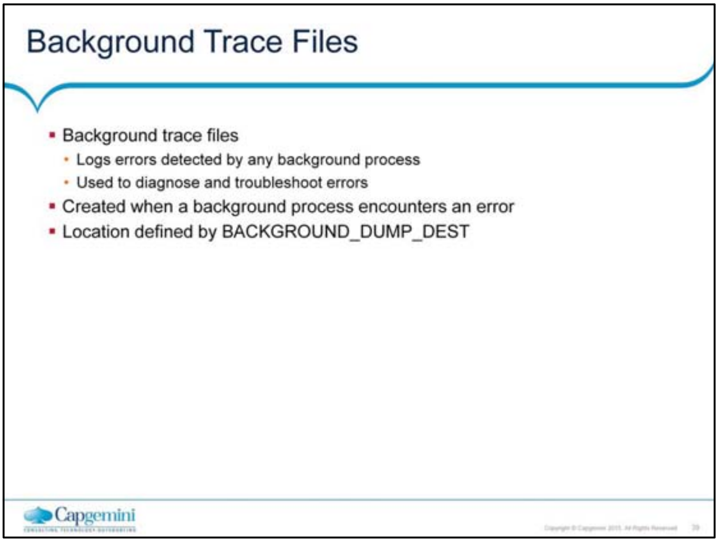


Viewing the Alert Log

Each database also has an alert_sid.log. The file is on the server with the database and is stored in the directory specified with the initialization parameter background_dump_dest. The alert file of a database is a chronological log of messages and errors, including the following:

- All internal errors (ORA-600), block corruption errors (ORA-1578), and deadlock errors (ORA-60) that occur
- Administrative operations, such as the SQL statements CREATE, ALTER, DROP DATABASE, TABLESPACE, ROLLBACK SEGMENT 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

EM monitors the alert log file and notifies you of critical errors. You can also view the log to see noncritical error and informative messages. Also the file can grow to an unmanageable size if left alone. You should make a back up of the alert file occasionally and delete the current alert file. When the database attempts to write to the alert file again, the database will then re-create a new alert file.



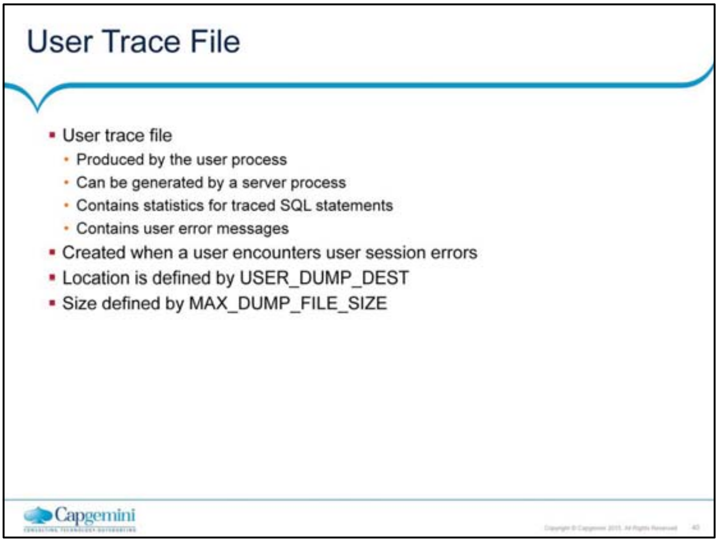
Background Trace Files

Background trace files are used to log errors that have been encountered by a background process, such are SMON, PMON, DBWn, and other background processes. These files exist only when an error requires writing to the trace files.You use these files to diagnose and troubleshoot problems.

Initially when a background trace file is created it contains header information indicating the version number of the data server and the operating system.

Naming convention for user trace file: sid_processname_PID.trc (db01_lgwr_23845.trc).

Its location is defined by the BACKGROUND_DUMP_DEST initialization parameter.



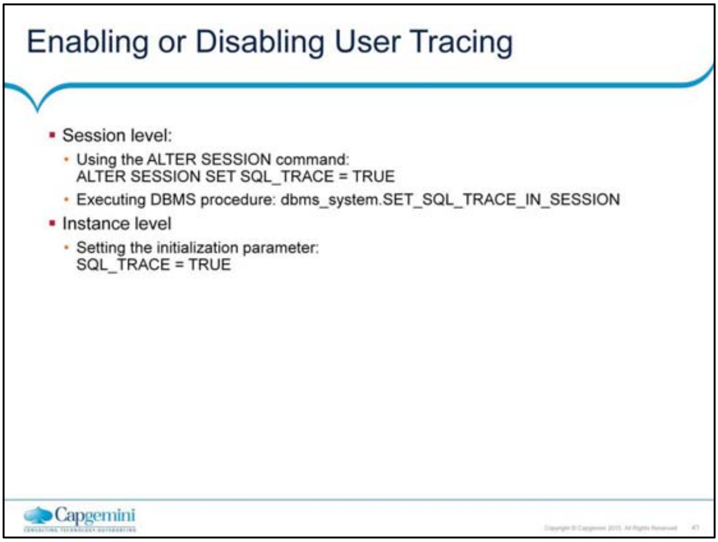
User Trace Files

User trace files contain statistics for traced SQL statements, which are useful for SQL tuning. In addition, user trace files contain user error messages.

Naming convention for user trace file:

sid_ora_PID.trc(db01_ora_23845.trc).

Its location is defined by the USER_DUMP_DEST initialization parameter.



Enabling or Disabling User Tracing

Note: Setting SQL_TRACE=TRUE at the instance level will produce a significant volume of trace data. This option should be used with care.

Enabling or Disabling User Tracing

Using Oracle Enterprise Manager to Enable or Disable User Tracing

From the OEM Console:

1. Navigate to Databases > Instance > Configuration.

Select All Initialization Parameters from the General page.

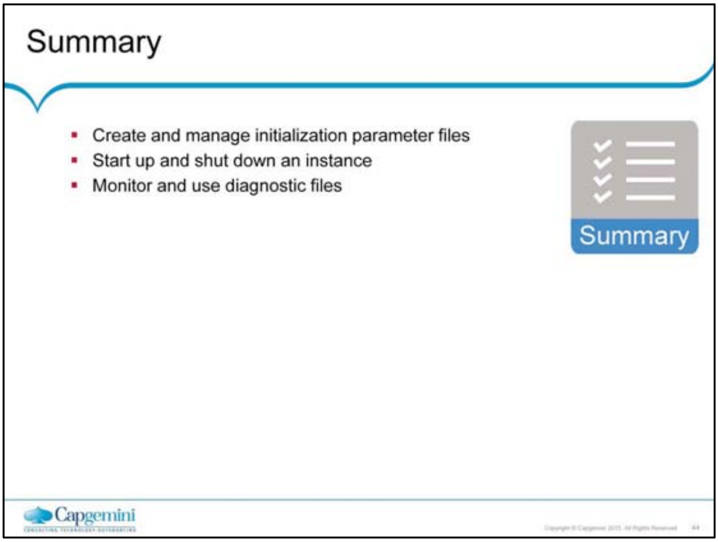
Set the parameter SQL_TRACE = TRUE.

Select OK.

Lab

- This practice covers the following topics:
  - Creating an SPFILE
  - Starting up and shutting down the database in different modes

Capgemini



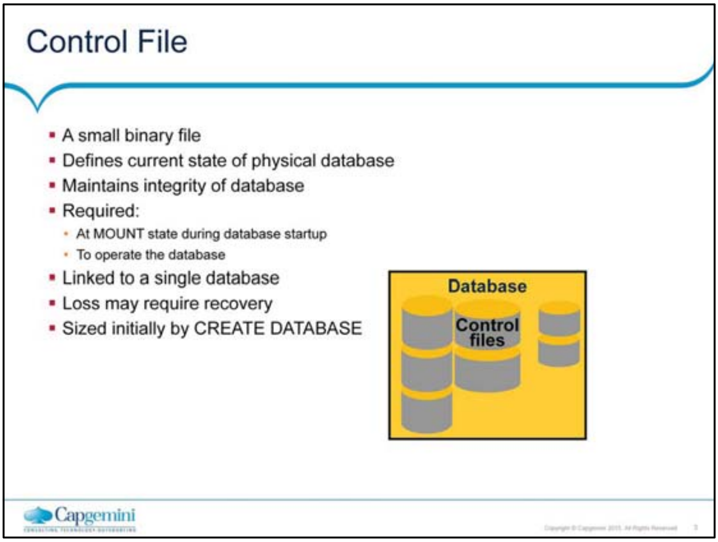
Add the notes here.

Oracle 11g DBA Fundamentals Overview

Lesson 04: Maintaining the Control File

Objectives

- After completing this lesson, you should be able to do the following:
  - Explain the uses of the control file
  - List the contents of the control file
  - Multiplex and manage the control file
  - Obtain control file information



Control File

The control file is a small binary file necessary for the database to start and operate successfully. Each control file is associated with only one Oracle database. Before a database is opened, the control file is read to determine if the database is in a valid state to use.

A control file is updated continuously by the Oracle server during database use, so it must be available for writing whenever the database is open. The information in the control file can be modified only by the Oracle server; no database administrator or end user can edit the control file.

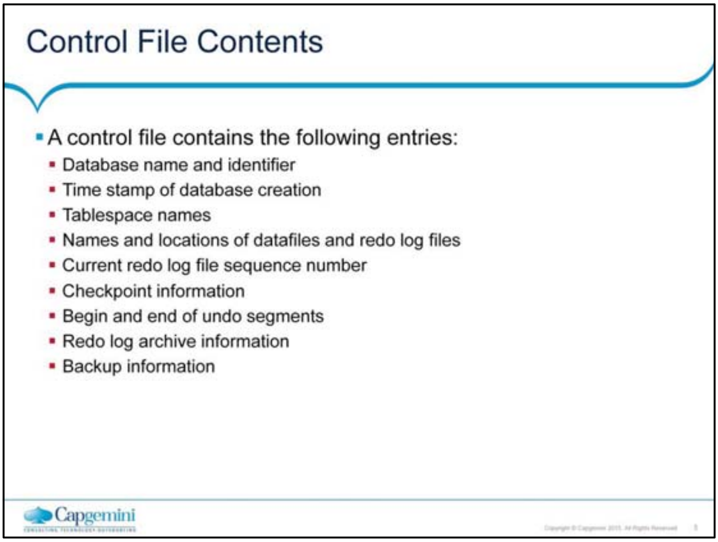
If for some reason the control file is not accessible, the database does not function properly. If all copies of a database's control files are lost, the database must be recovered before it can be opened.

Control File (continued)

Sizing the control file:

Keywords specified during the creation of the database affect the size of the control file. This is particularly significant when the parameters have large values. The size of the control file is influenced by the following keywords in the CREATE DATABASE or CREATE CONTROLFILE commands:

- MAXLOGFILES
- MAXLOGMEMBERS
- MAXLOGHISTORY
- MAXDATAFILES
- MAXINSTANCES



Control File Contents

The information in the control file includes the following:

Database name is taken from either the name specified by the initialization parameter DB_NAME or the name used in the CREATE DATABASE statement.

Database identifier is recorded when the database is created.

Time stamp of database creation is also recorded at database creation.

Names and locations of associated datafiles and online redo log files are updated when a datafile or redo log is added to, renamed in, or dropped from the database.

Tablespace information is updated as tablespaces are added or dropped.

Redo log history is recorded during log switches.

Location and status of archived logs are recorded when archiving occurs.

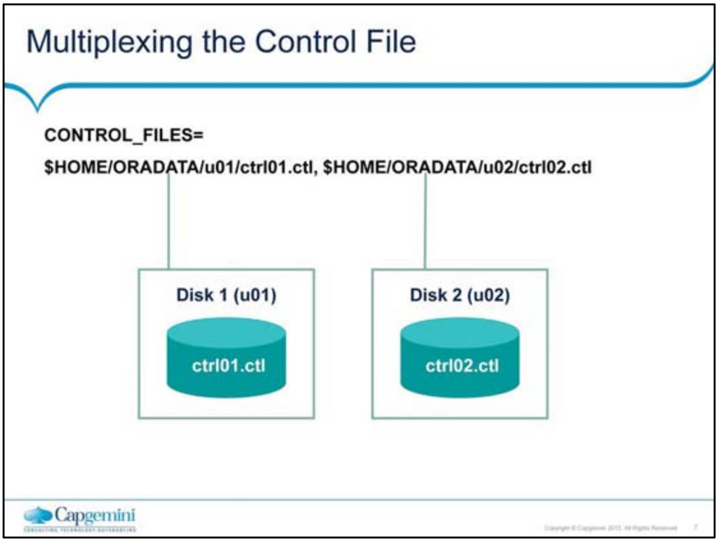
Location and status of backups are recorded by the Recovery Manager utility.

Current log sequence number is recorded when log switches occur.

Checkpoint information is recorded as checkpoints are made.

Contents of the Control File (continued)

- The control file consists of two types of sections:
  - Reusable
  - Not reusable
- Reusable sections store Recovery Manager information, such as backup datafile names and backup redo log file names. They are used in a circular manner and can be reused only by Recovery Manager.
- Note: Recovery Manager is covered in more detail in the course Oracle9i Database Administration Fundamentals II.



Multiplexing the Control File

To safeguard against a single point of failure of the control file, it is strongly recommended that the control file be multiplexed, storing each copy on a different physical disk. If a control file is lost, a multiplexed copy of the control file can be used to restart the instance without database recovery.

Control files can be multiplexed up to eight times by:

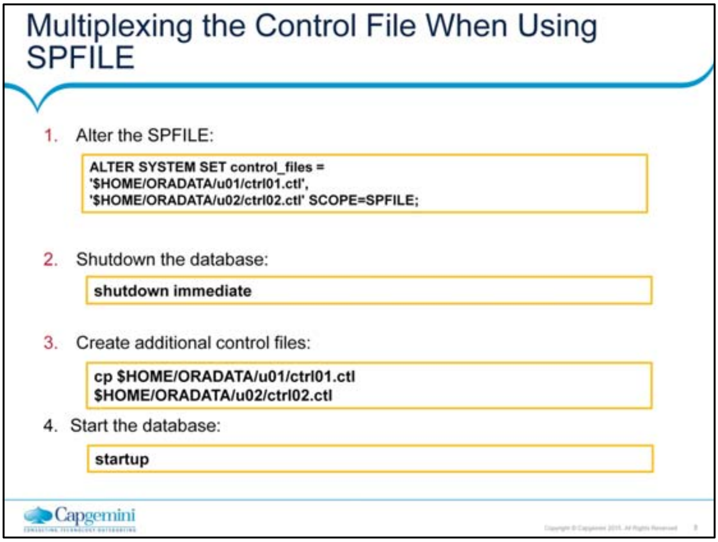
- Creating multiple control files when the database is created by including the control file names and full path in the initialization parameter file:

  CONTROL_FILES=$HOME/ORADATA/u01/ctrl01.ctl,
  $HOME/ORADATA/u02/ctrl02.ctl

- Adding a control file after the database is created

Backing up the control files:

Because the control file records the physical structure of the database, you should immediately make a backup of your control file after making changes to the physical structure of the database. Backup and recovery of the control file is covered in the course Oracle9i Database Administration Fundamentals II.



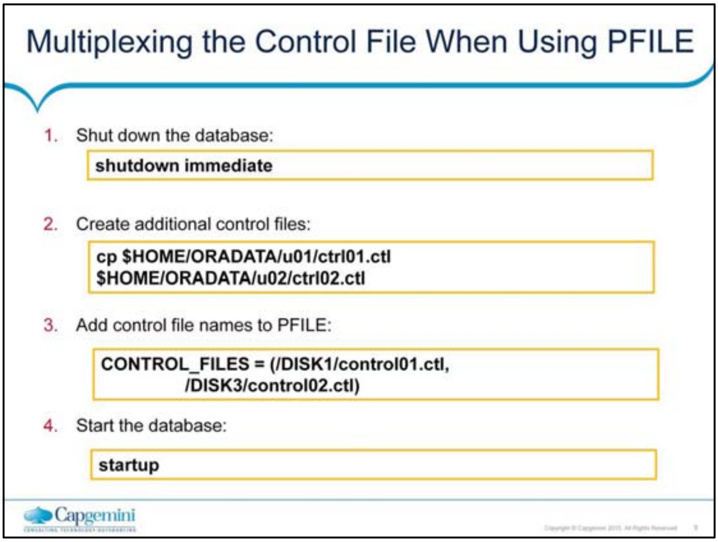
Multiplexing the Control File When Using SPFILE

Alter the SPFILE: Using the ALTER SYSTEM SET command alter the SPFILE to include a list of all control files to be used: main control file and multiplexed copies.

Shutdown the database: Shutdown the database in order to create the additional control files on the operating system.

Create additional control files: Using the operating system copy command, create the additional control files as required and verify that the files have been created in the appropriate directories.

Start the database: When the database is started the SPFILE will be read and the Oracle server will maintain all the control files listed in the CONTROL_FILES parameter.



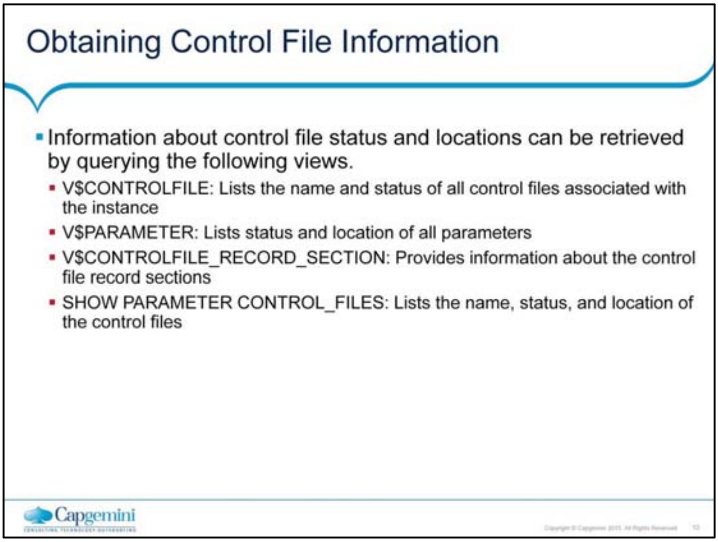
Multiplexing the Control File When Using PFILE

Shut down the database: Shutdown the database in order to create the additional control files on the operating system.

Create additional control files: Using the operating system copy command, create the additional control files as required and verify that the files have been created in the appropriate directories.

Add control file names to PFILE: Alter the PFILE to include a listing of all of the control files.

Start the database: When the database is started the PFILE will be read and the Oracle server will maintain all the control files listed in the CONTROL_FILES parameter.



Obtaining Control File Information

To obtain the location and names of the control files, query the V$CONTROLFILE view.

```
SELECT name FROM V$CONTROLFILE;
NAME
------------------------------------
/u01/home/db03/ORADATA/u01/ctrl01.ctl
/u01/home/db03/ORADATA/u01/ctrl01.ctl
2 rows selected.
```

The V$PARAMETER view can also be used.

```
SELECT name, value from V$PARAMETER
WHERE name = 'control_files';
NAME                Value
-------------   ----------------------------
control_files   /u01/home/db03/ORADATA/u01/ctrl01.ctl
```

Obtaining Control File Information (continued)

To obtain information about the different sections of the control files, query the V$CONTROLFILE_RECORD_SECTION view.

```
SQL> SELECT type, record_size, records_total, records_used
2  FROM v$controlfile_record_section
3  WHERE TYPE='DATAFILE';

TYPE              RECORD_SIZE RECORDS_TOTAL RECORDS_USED
------------------ ----------- ------------- ------------
DATAFILE                  180            40           10
1 row selected.
```

The column RECORDS_TOTAL specifies the number of records allocated for a special section. For example, you can view the maximum number of datafiles in our example 30, which is determined by the MAXDATAFILES parameter in the CREATE DATABASE command.

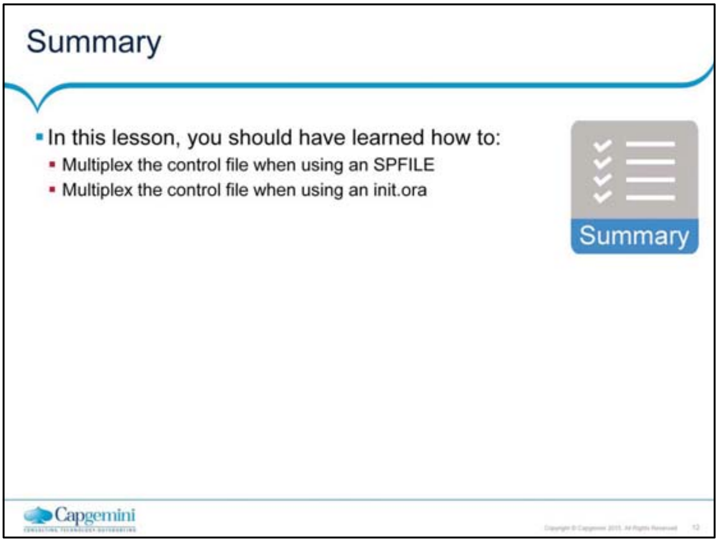
The SHOW PARAMETER command can also be used to find the location of the control files.

```
SQL>  SHOW PARAMETER control_files;
NAME          TYPE        VALUE
------------- ----------- ------------------------------
control_files  string     $HOME/ORADATA/u01/ctrl01.ctl,

$HOME/ORADATA/u02/ctrl02.ctl
```

Information in several dynamic performance views is obtained from the control file. Below is a list of examples:

V$BACKUP
V$DATAFILE
V$TEMPFILE
V$TABLESPACE
V$ARCHIVE
V$LOG
V$LOGFILE
V$LOGHIST
V$ARCHIVED_LOG
V$DATABASE


Summary
In this lesson, you should have learned how to:
Multiplex the control file when using an SPFILE
Multiplex the control file when using an init.ora
Summary
Capgemini

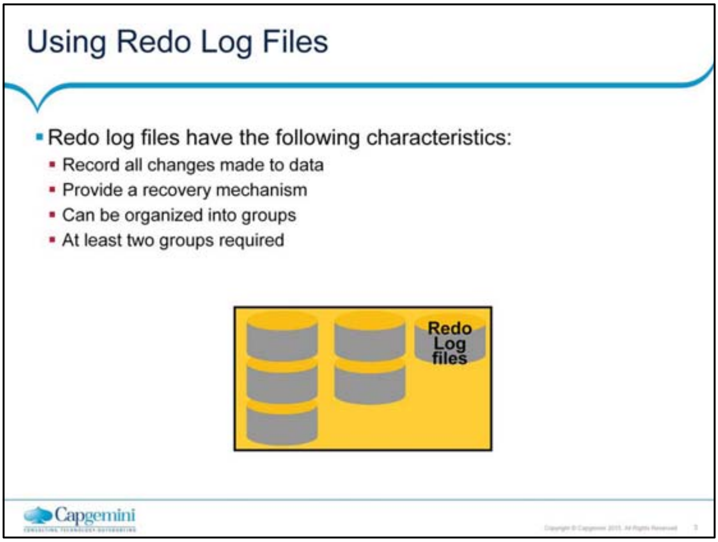
Oracle 11g DBA Fundamentals Overview

Lesson 05: Managing the Redo Log

Objectives

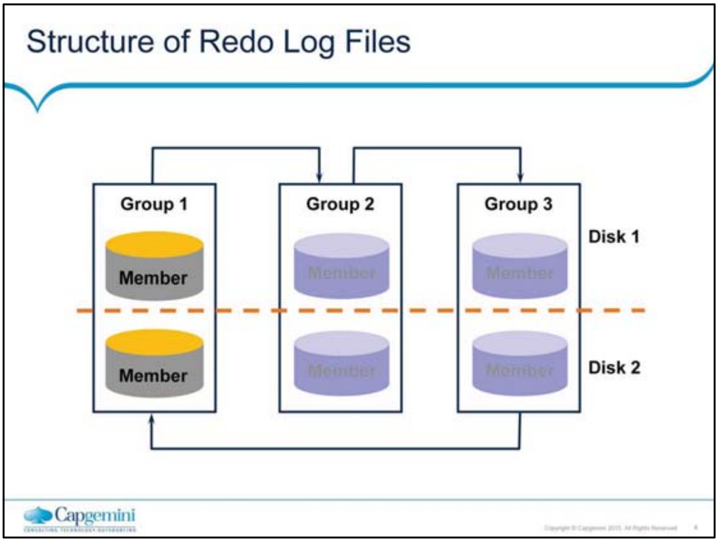
- After completing this lesson, you should be able to do the following:
  - Explain the purpose of online redo log files
  - Outline the structure of online redo log files
  - Control log switches and checkpoints
  - Multiplex and maintain online redo log files
  - Manage online redo logs files with OMF
  - What Is the Archived Redo Log?
  - Choosing Between NOARCHIVELOG and ARCHIVELOG Mode

Capgemini



Using Redo Log Files

Redo log files provide the means to redo transactions in the event of a database failure. Every transaction is written synchronously to the Redo Log Buffer, then gets flushed to the redo log files in order to provide a recovery mechanism in case of media failure. (With exceptions such as direct load inserts in objects with the NOLOGGING clause enabled.) This includes transactions that have not yet been committed, undo segment information, and schema and object management statements. Redo log files are used in a situation such as an instance failure to recover committed data that has not been written to the datafiles. The redo log files are used only for recovery.



Structure of the Redo Log Files

The database administrator can set up the Oracle database to maintain copies of online redo log files to avoid losing database information due to a single point of failure.

Online redo log file groups:

A set of identical copies of online redo log files is called an online redo log file group.

The LGWR background process concurrently writes the same information to all online redo log files in a group.

The Oracle server needs a minimum of two online redo log file groups for the normal operation of a database.

Online redo log file members:

Each online redo log file in a group is called a member.

Each member in a group has identical log sequence numbers and are of the same size. The log sequence number is assigned each time that the Oracle server writes to a log group to uniquely identify each redo log file. The current log sequence number is stored in the control file and in the header of all datafiles.

Structure of the Redo Log Files (continued)

- Creating initial redo log files:
- The initial set of online redo log file groups and members are created during the database creation.
- The following parameters limit the number of online redo log files:
  - The MAXLOGFILES parameter in the CREATE DATABASE command specifies the absolute maximum of online redo log file groups.
  - The maximum and default value for MAXLOGFILES is dependent on your operating system.
  - The MAXLOGMEMBERS parameter used in the CREATE DATABASE command determines the maximum number of members per group. The maximum and default value for MAXLOGMEMBERS is dependent on your operating system.

How Redo Log Files Work

- Redo log files are used in a cyclic fashion.
- When a redo log file is full, LGWR will move to the next log group.
  - Called a log switch
  - Checkpoint operation also occurs
  - Information written to the control file

Capgemini

How Redo Log Files Work

The Oracle server sequentially records all changes made to the database in the Redo Log Buffer. The redo entries are written from the Redo Log Buffer to one of the online redo log file groups called the current online redo log file group by the LGWR process. LGWR writes under the following situations:

- When a transaction commits
- When the Redo Log Buffer becomes one-third full
- When there is more than a megabyte of changed records in the Redo Log Buffer
- Before the DBWn writes modified blocks in the Database Buffer Cache to the datafiles

Redo log files are used in a cyclic fashion. Each redo log file group is identified by a log sequence number that is overwritten each time the log is reused.

Log switches:

LGWR writes to the online redo log files sequentially. When the current online redo log file group is filled, LGWR begins writing to the next group. This is called a log switch.

When the last available online redo log file is filled, LGWR returns to the first online redo log file group and starts writing again.

How Redo Log Files Work (continued)

Checkpoints:

During a checkpoint:

- DBWn writes a number of dirty database buffers, that are covered by the log that is being checkpointed, to the datafiles. The number of buffers that DBWn writes is determined by the FAST_START_MTTR_TARGET parameter, if specified. The default is zero.

Note: The FAST_START_MTTR_TARGET parameter is covered in detail in the *Oracle9i Database Administration Fundamentals II* course.

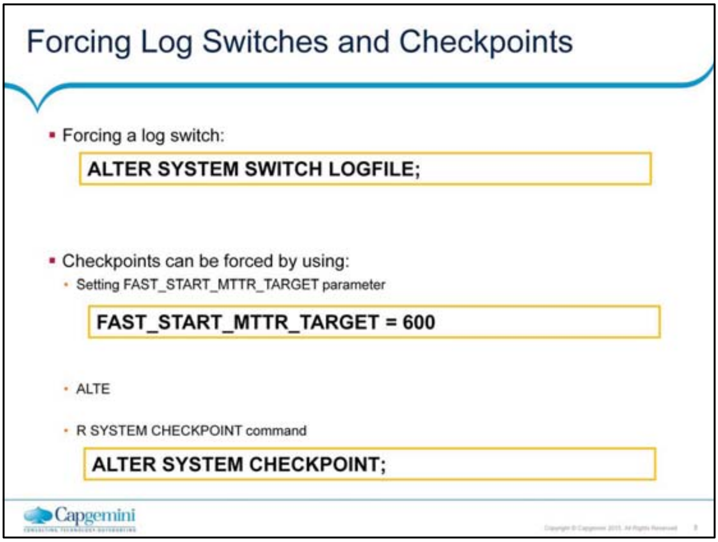
- The checkpoint background process CKPT updates the control file to reflect that it has completed a checkpoint successfully. If the checkpoint is caused by a log switch, CKPT also updates the headers of the datafiles.

Checkpoints can occur for all datafiles in the database or only for specific datafiles.

A checkpoint occurs, for example, in the following situations:

- At every log switch
- When an instance has been shut down with the normal, transactional, or immediate option
- When forced by setting the initialization parameter FAST_START_MTTR_TARGET
- When manually requested by the database administrator
- When the ALTER TABLESPACE [OFFLINE NORMAL|READ ONLY|BEGIN BACKUP]command causes checkpointing on specific datafiles

Information about each checkpoint is recorded in the alert_SID.log file if the LOG_CHECKPOINTS_TO_ALERT initialization parameter is set to TRUE. The default value of FALSE for this parameter does not log checkpoints.



Forcing Log Switches and Checkpoints

Log switches and checkpoints are automatically performed at certain points in the operation of the database, as identified previously. However, a DBA can force a log switch or a checkpoint to occur.

Forcing checkpoints:

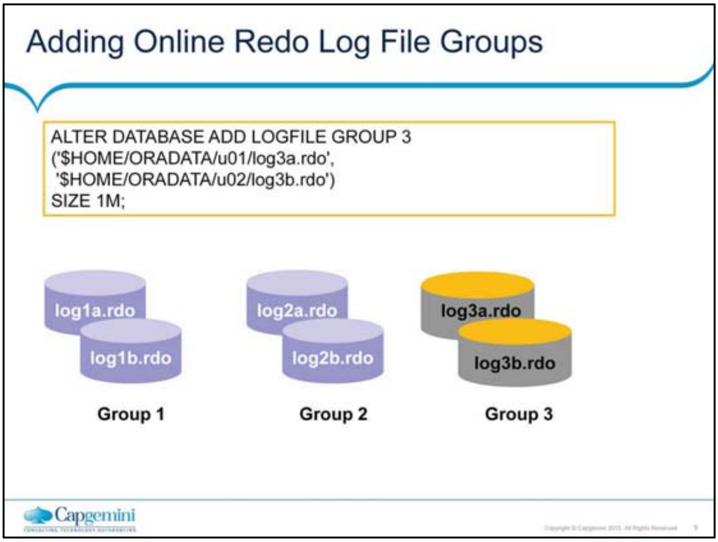
FAST_START_MTTR_TARGET parameter replaces the deprecated parameters:

FAST_START_IO_TARGET

LOG_CHECKPOINT_TIMEOUT

These deprecated parameters must not be used if the parameter FAST_START_MTTR_TARGET is used.

In the example above, the FAST_START_MTTR_TARGET parameter has been set so that instance recovery should not take more than 600 seconds. The database will adjust the other parameters to this goal.

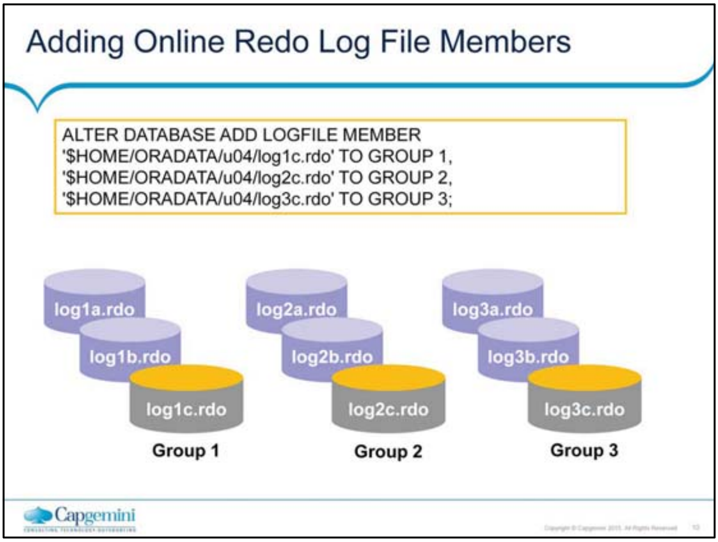


Adding Online Redo Log File Groups

In some cases you might need to create additional log file groups. For example, adding groups can solve availability problems. To create a new group of online redo log files, use the following SQL command:

```
ALTER DATABASE [database]
        ADD LOGFILE [GROUP integer] filespec
        [, [GROUP integer] filespec]...]
```

You specify the name and location of the members with the file specification. The value of the GROUP parameter can be selected for each redo log file group. If you omit this parameter, the Oracle server generates its value automatically.



Adding Online Redo Log File Members

You can add new members to existing redo log file groups using the following ALTER DATABASE ADD LOGFILE MEMBER command:

```
ALTER DATABASE [database]
            ADD LOGFILE MEMBER
            [ 'filename' [REUSE]
             [, 'filename' [REUSE]]...
            TO {GROUP integer
             |('filename'[, 'filename']...)
             }
            ]...
```

Use the fully specified name of the log file members; otherwise the files are created in a default directory of the database server.

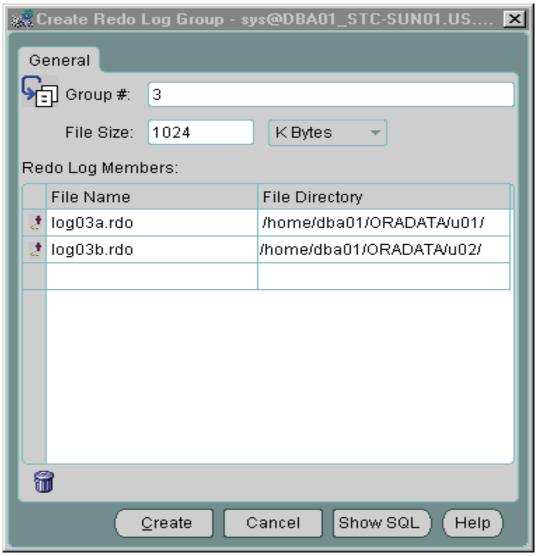
If the file already exists, it must have the same size, and you must specify the REUSE option. You can identify the target group either by specifying one or more members of the group or by specifying the group number.

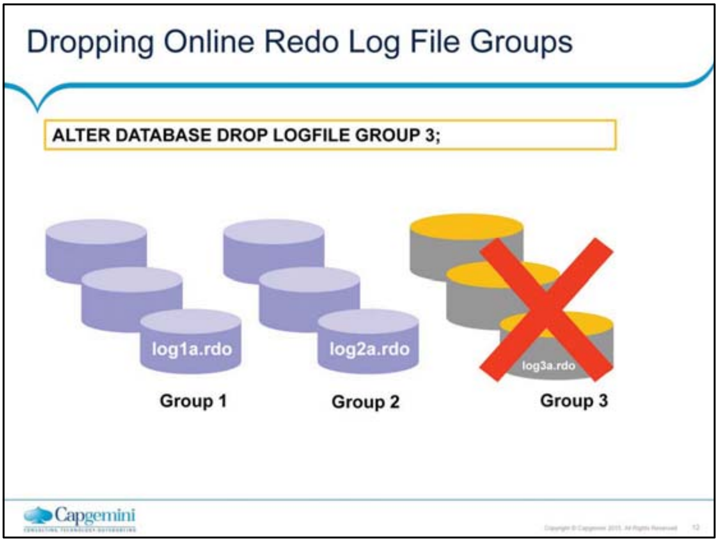
Adding Online Redo Log File Members (continued)

Using Oracle Enterprise Manager to Add Redo Log File Groups and Members

From the OEM Console:

- Navigate to Databases > Storage.
- Click the Redo Log Groups folder.
- Select Create from the right-mouse menu.
- In the General tab, complete the information to create the redo log file group and members.
- Click Create.





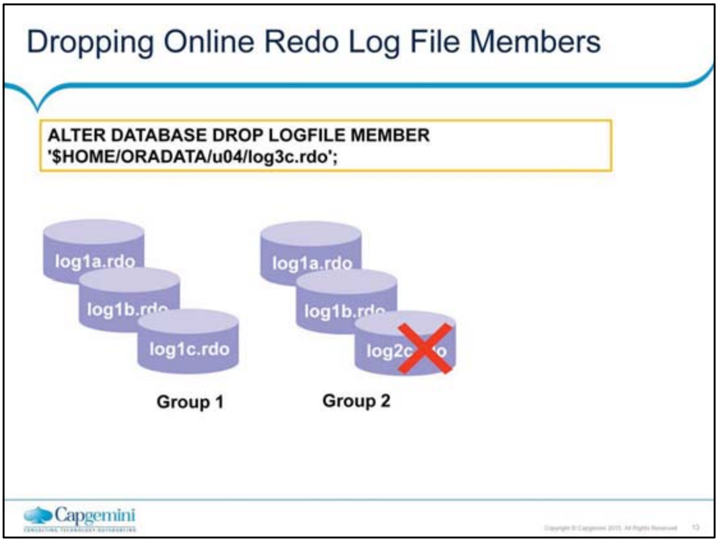
Dropping Online Redo Log File Groups

To increase or decrease the size of online redo log file groups, add new online redo log file groups (with the new size) and then drop the old ones. An entire online redo log file group can be dropped with the following ALTER DATABASE DROP LOGFILE command:

ALTER DATABASE [database]
DROP LOGFILE {GROUP integer|('filename'[, 'filename']...)}
[,{GROUP integer|('filename'[, 'filename']...)}]...

Restrictions:

An instance requires at least two groups of online redo log files.
An active or current group cannot be dropped.
When an online redo log file group is dropped, the operating system files are not deleted.



Dropping a Redo Log File Members

You may want to drop an online redo log file member because it is invalid. Use the following ALTER DATABASE DROP LOGFILE MEMBER command if you want to drop one or more specific online redo log file members:

ALTER DATABASE [database]
DROP LOGFILE MEMBER 'filename'[, 'filename']...

Restrictions:

If the member you want to drop is the last valid member of the group, you cannot drop that member.

If the group is current, you must force a log file switch before you can drop the member.

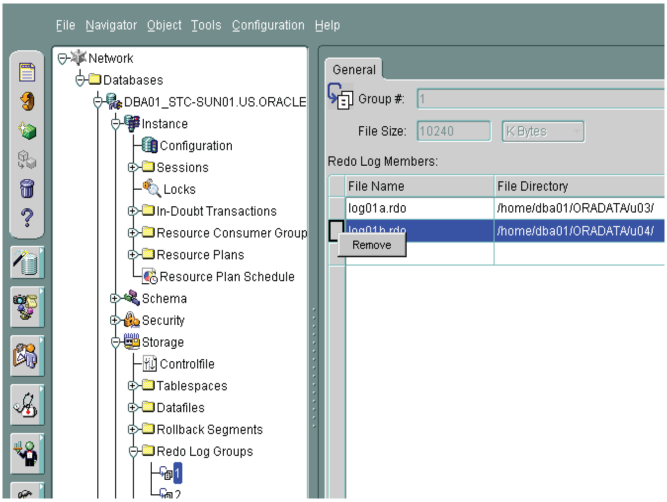
If the database is running in ARCHIVELOG mode and the log file group to which the member belongs is not archived, then the member cannot be dropped.

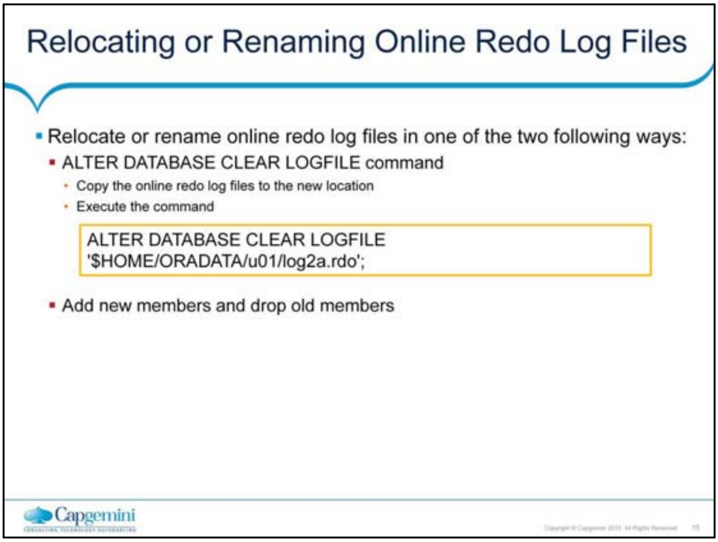
When an online redo log file member is dropped, the operating system file is not deleted if you are not using OMF feature.

Using Storage Manager to Drop Redo Log File Groups and Members

Using Oracle Enterprise Manager to Drop Redo Log File Groups and Members:

From the OEM Console:
Navigate to Databases > Storage.
To remove a group:
Expand the Redo Log Groups folder and select a redo log file group you want to remove.
Select Remove from right mouse menu.
Confirm remove.
To remove a member:
Expand the Redo Log Groups folder and navigate to the group containing the member you want to drop.
In the General page, highlight the member, and select Remove from the right mouse menu.
Confirm remove.





Relocating or Renaming Online Redo Log Files

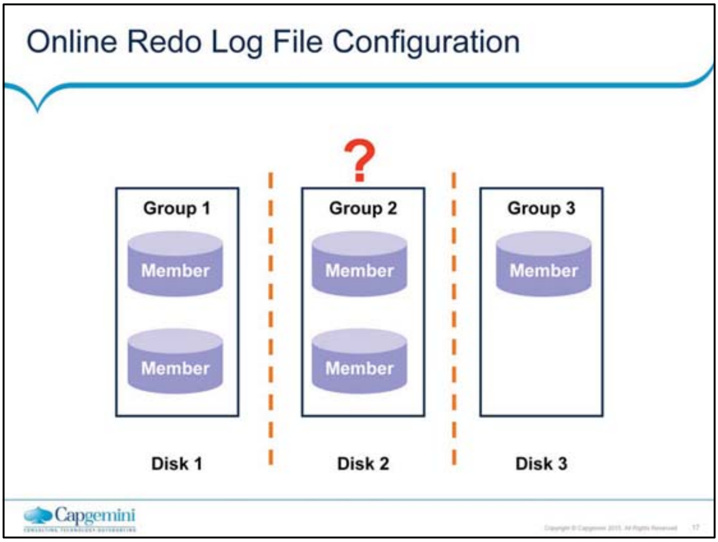
The locations of online redo log files can be changed by renaming the online redo log files. Before renaming the online redo log files, ensure that the new online redo log file exists. The Oracle server changes only the pointers in the control files, but does not physically rename or create any operating system files.

The following ALTER DATABASE RENAME FILE command changes the name of the online redo log file:

```
SQL> ALTER DATBASE [database}
 2        RENAME FILE 'filename' [,'filename']…
 3        TO 'filename']…
```

Using Storage Manager to Relocate or Rename Redo Log File Members

- Using Oracle Enterprise to Relocate or Rename Redo Log File Groups and Members
- From the OEM Console:
  - Navigate to Databases > Storage > Redo Log Groups.
  - Select a redo log file group.
  - Modify the redo log file member’s File Name or File Directory to rename or relocate members.
  - Click Apply.



Online Redo Log File Configuration

To determine the appropriate number of online redo log files for a database instance, you have to test different configurations.

In some cases, a database instance may require only two groups. In other situations, a database instance may require additional groups to guarantee that the groups are always available to LGWR. For example, if messages in the LGWR trace file or in the alert file indicate that LGWR frequently has to wait for a group because a checkpoint has not completed or a group has not been archived, you need to add groups.

Although with the Oracle server multiplexed groups can contain different numbers of members, try to build up a symmetric configuration. An asymmetric configuration should only be the temporary result of an unusual situation such as a disk failure.

Location of online redo log files:

When you multiplex the online redo log files, place members of a group on different disks. By doing this, even if one member is not available but other members are available, the instance does not shut down.

Separate archive log files and online redo log files on different disks to reduce contention between the ARCn and LGWR background processes.

Online Redo Log File Configuration (continued)

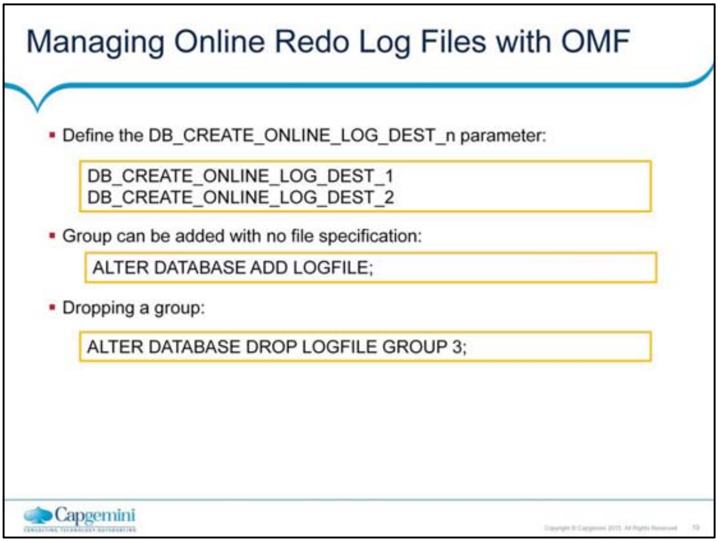
Datafiles and online redo log files should be placed on different disks to reduce LGWR and DBWn contention and reduce the risk of losing both datafiles and online redo log files in the event of media failure.

Sizing online redo log files:

The minimum size of an online redo log file is 50 KB, and the maximum size is specific to the operating system. Members of different groups can have different sizes; however, there is no benefit in having different-sized groups. Different-sized groups should be required as a temporary result only if you want to change the size of the members of the online redo log file groups. In this case, you have to create new online redo log file groups with different sizes, and then remove the old groups.

The following situations might influence the configuration of the online redo log files:

- Number of log switches and checkpoints
- Number and amount of redo entries
- Amount of space on the storage medium; for example, on a tape if archiving is enabled



Managing Online Redo Log Files with OMF

Define the DB_CREATE_ONLINE_LOG_DEST_n parameter: To create online redo log files to be managed by OMF, the DB_CREATE_ONLINE_LOG_DEST_n parameter must be defined. The parameter must be set for each multiplexed copy identified by the n value. In the example above, two groups have been created with two members each. The names will automatically be generated (such as ora_1_wo94n2xi.log) and displayed in the alertSID.log. The default size is 100 MB.

To create a new group of online redo log files, the DBA uses the ALTER DATABASE ADD LOGFILE command. The command has been modified so that the file specification is not necessary.

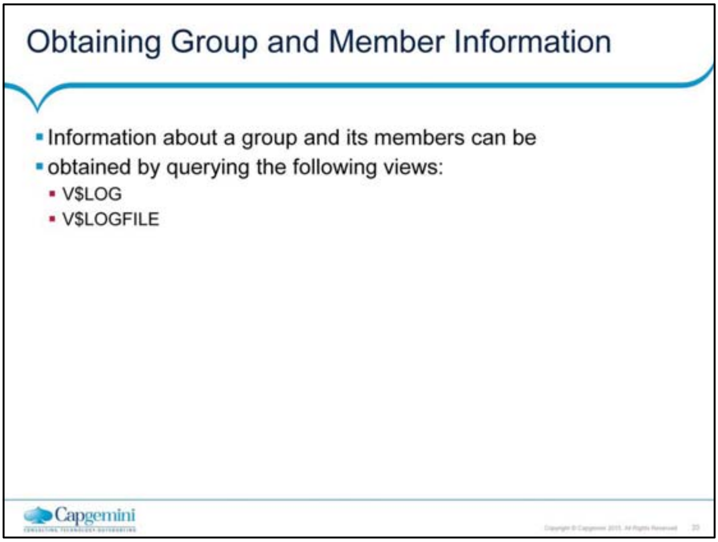
The example in the slide adds a log file with two members: one in the location defined by DB_CREATE_ONLINE_LOG_DEST_1 and one in DB_CREATE_ONLINE_LOG_DEST_2. Unique filenames for the log file members are generated automatically and displayed in the alertSID.log. The default size is 100 MB.

Dropping a group:

The above example drops the log file Group 3, and its operating system files associated with each OMF log file member in Group 3.

Archived Redo Log Files and OMF:

Archived redo log files cannot be OMF.



Obtaining Group and Member Information

V$LOG view:

The following query returns information about the online redo log file from the control file:

```
SQL>    SELECT group#, sequence#, bytes, members,
status
 2      FROM v$log;
GROUP#  SEQUENCE#  BYTES     MEMBERS
STATUS
------  ---------- --------  ---------  --
-------
        1          688       1048576    1
CURRENT
        2          689       1048576    1
INACTIVE
2 rows selected.
```

The following items are the most common values for the STATUS column:

UNUSED: Indicates that the online redo log file group has never been written to. This is the state of an online redo log file that was just added.

CURRENT: Indicates the current online redo log file group. This implies that the online redo log file group is active.

ACTIVE: Indicates that the online redo log file group is active but is not the current online redo log file group. It is needed for crash recovery. It may be in use for block recovery. It may or may not be archived.

Obtaining Group and Member Information (continued)

- CLEARING: Indicates that the log is being re-created as an empty log after an ALTER DATABASE CLEAR LOGFILE command. After the log is cleared, the status changes to UNUSED.
- CLEARING_CURRENT: Indicates that the current log file is being cleared of a closed thread. The log can stay in this status if there is some failure in the switch, such as an input/output (I/O) error writing the new log header.
- INACTIVE: Indicates that the online redo log file group is no longer needed for instance recovery. It may or may not be archived.

V$LOGFILE view:
To obtain the names of all the members of a group, query the V$LOGFILE view.

```
SQL> SELECT member FROM V$LOGFILE;
MEMBER
------------------------------------
/u01/home/db03/ORADATA/u03/log02a.rdo
/u01/home/db03/ORADATA/u03/log01a.rdo
```

The value of the STATUS column could be one of the following:

- INVALID: Indicates that the file is inaccessible
- STALE: Indicates that contents of the file are incomplete
- DELETED: Indicates that the file is no longer used
- Blank indicates that the file is in use

What Is the Archived Redo Log?

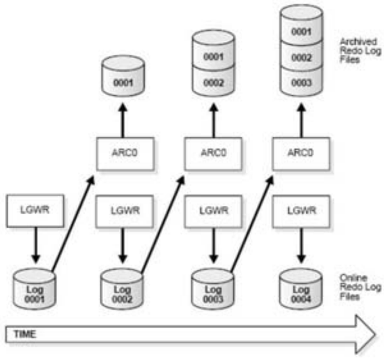
- Oracle Database lets you save filled groups of redo log files to one or more offline destinations, known collectively as the archived redo log, or more simply the archive log.
- The process of turning redo log files into archived redo log files is called archiving.
- This process is only possible if the database is running in ARCHIVELOG mode.
- You can choose automatic or manual archiving.

Choosing Between NOARCHIVELOG and ARCHIVELOG Mode

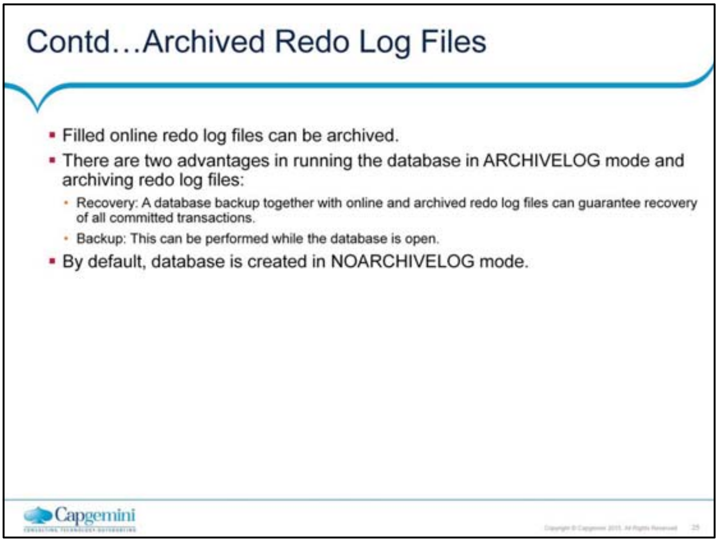
- Running a Database in NOARCHIVELOG Mode
  - When you run your database in NOARCHIVELOG mode, you disable the archiving of the redo log.
  - The database control file indicates that filled groups are not required to be archived.
  - Therefore, when a filled group becomes inactive after a log switch, the group is available for reuse by LGWR.
  - When you run a database in ARCHIVELOG mode, you enable the archiving of the redo log.
  - The database control file indicates that a group of filled redo log files cannot be reused by LGWR until the group is archived.
  - A filled group becomes available for archiving immediately after a redo log switch occurs.

Capgemini

Redo Log File Use in ARCHIVELOG Mode



Capgemini



Archived Redo Log Files

One of the important decisions that a database administrator (DBA) has to make is whether the database is configured to operate in ARCHIVELOG mode or in NOARCHIVELOG mode.

NOARCHIVELOG mode:

In NOARCHIVELOG mode, the online redo log files are overwritten each time an online redo log file is filled, and log switches occur. LGWR does not overwrite a redo log file group until the checkpoint for that group is completed.

ARCHIVELOG mode:

If the database is configured to run in ARCHIVELOG mode, inactive groups of filled online redo log files must be archived. Because all changes made to the database are recorded in the online redo log files, the database administrator can use the physical backup and the archived online redo log files to recover the database without losing any committed data.

There are two ways in which online redo log files can be archived:

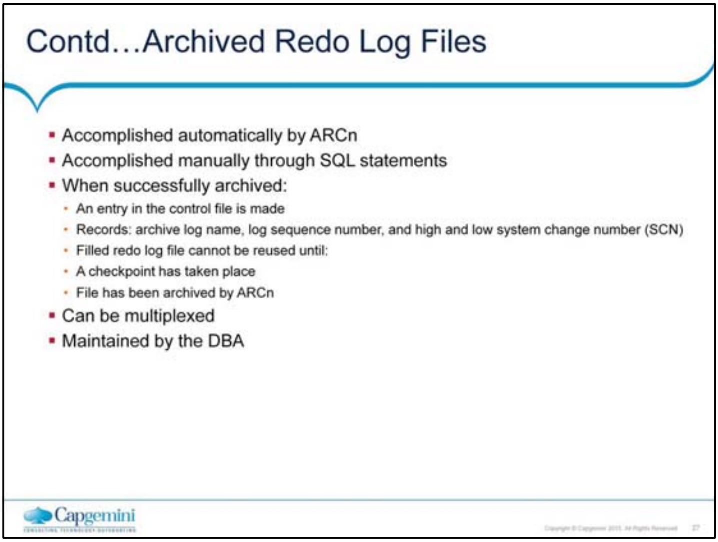
- Manually
- Automatically (recommended method)

Archived Redo Log Files (continued)

ARCHIVELOG mode (continued):

The LOG_ARCHIVE_START initialization parameter indicates whether archiving should be automatic or manual when the instance starts up.

- TRUE: TRUE indicates that archiving is automatic. ARCn initiates archiving of the filled log group at every log switch.
- FALSE: The default value, FALSE indicates that the DBA archives filled redo log files manually. The DBA must manually execute a command each time you want to archive an online redo log file. All or specific online redo log files can be archived manually.



Archived Redo Log Files

Information about archived logs can be obtained from V$INSTANCE.

```
SQL> SELECT archiver
  2 FROM v$instance;
ARCHIVE
---------
STOPPED
1 row selected.
```

Note: Archiving is covered in detail in the Oracle9i Database Administration Fundamentals II course.

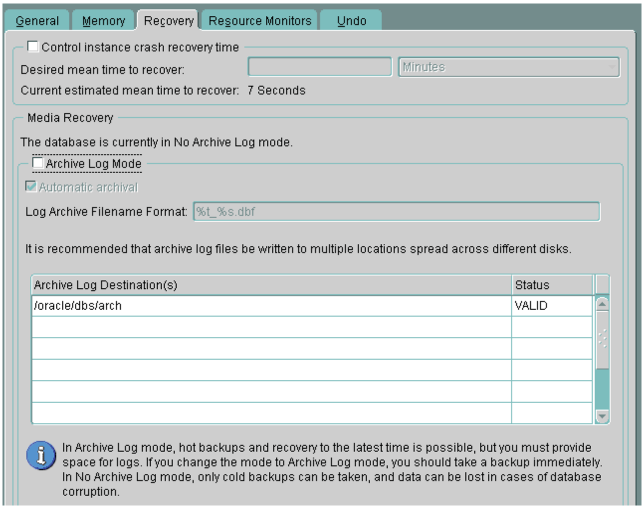
Archived Redo Log Files (continued)

Using Oracle Enterprise Manager to Obtain Archive Information

From OEM Console:

Navigate to Databases > Instance.

2. Click Configuration.
3. The General page identifies:
   - Database and Instance Information–Archive Log Mode: Identifies the mode the database is running in
   - All Initialization Parameters: Identifies any parameters set for archiving
4. The Recovery page allows you to set and identify the specifics of archiving such as: mode, filename format, and log destinations.



Summary

- In this lesson, you should have learned how to:
  - Explain the use of online redo log files
  - Obtain redo log file information
  - Control log switches and checkpoints
  - Multiplex and maintain online redo log files
  - Manage online redo log files with OMF
  - What Is the Archived Redo Log?
  - Choosing Between NOARCHIVELOG and ARCHIVELOG Mode

Summary

Capgemini

Oracle 11g DBA Fundamentals Overview

Lesson 06: Managing Tablespaces

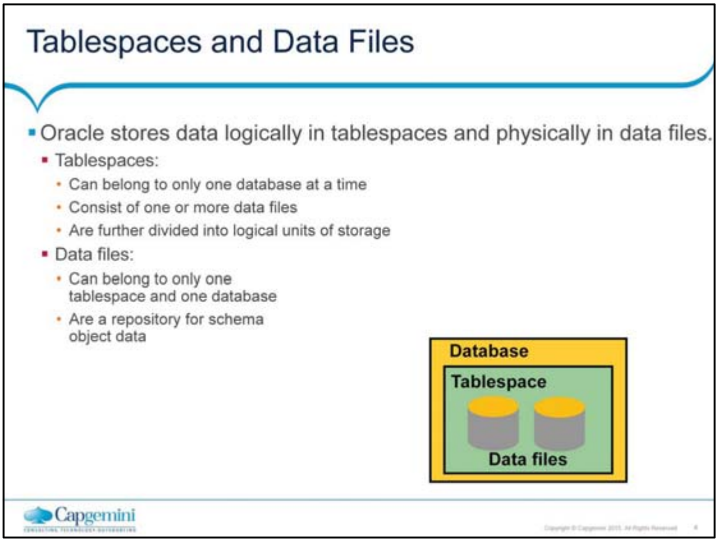
Lesson Objectives

- Creating Tablespaces
- Altering Tablespace Availability
- Using Read-Only Tablespaces
- Renaming Tablespaces
- Managing the SYSAUX Tablespace

Table spaces

- A tablespace is a logical storage unit within an Oracle database.
- It is logical because a table space is not visible in the file system of the machine on which the database resides.
- A table space, in turn, consists of at least one data file which, in turn, are physically located in the file system of the server.
- A datafile belongs to exactly one tablespace.
- Each table, index and so on that is stored in an Oracle database belongs to a table space.
- The table space builds the bridge between the Oracle database and the file system in which the table's or index' data is stored.
- There are three types of table spaces in Oracle:
  - Permanent table spaces
  - Undo table spaces
  - temporary table spaces

Capgemini



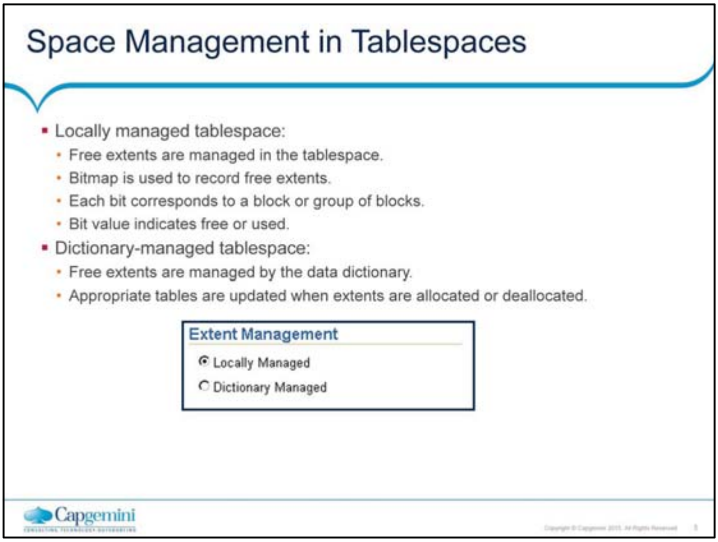
Tablespaces and Data Files

Databases, tablespaces, and data files are closely related, but they have important differences:

An Oracle database consists of one or more logical storage units called tablespaces, which collectively store all of the database's data.

Each tablespace in an Oracle database consists of one or more files called data files, which are physical structures that conform with the operating system on which Oracle is running.

A database's data is collectively stored in the data files that constitute each tablespace of the database. For example, the simplest Oracle database would have one tablespace and one data file. Another database can have three tablespaces, each consisting of two data files (for a total of six data files). A single database could potentially have as many as 65,535 data files.

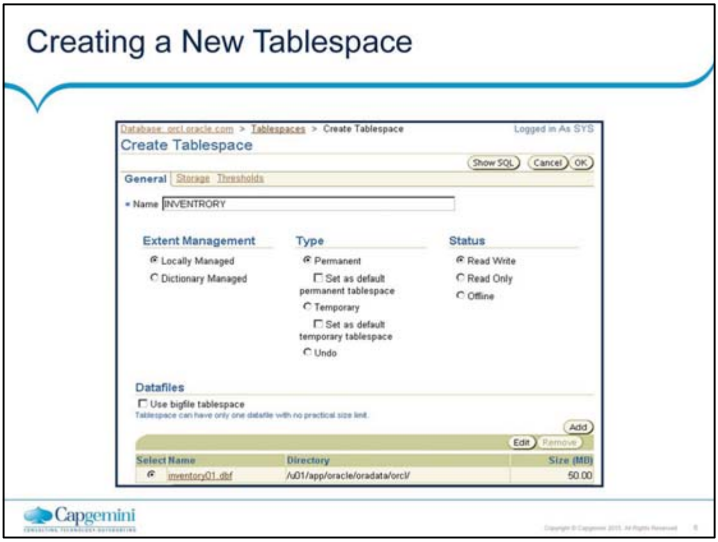


Space Management in Tablespaces

Tablespaces allocate space in extents. Tablespaces can be created to use one of the following two different methods of keeping track of free and used space:

Locally managed tablespaces: The extents are managed within the tablespace via bitmaps. Each bit in the bitmap corresponds to a block or a group of blocks. When an extent is allocated or freed for reuse, the Oracle server changes the bitmap values to show the new status of the blocks.

Dictionary-managed tablespaces: The extents are managed by the data dictionary. The Oracle server updates the appropriate tables in the data dictionary whenever an extent is allocated or deallocated. This is for backward compatibly; you should use locally managed for all tablespaces.



Creating a New Tablespace

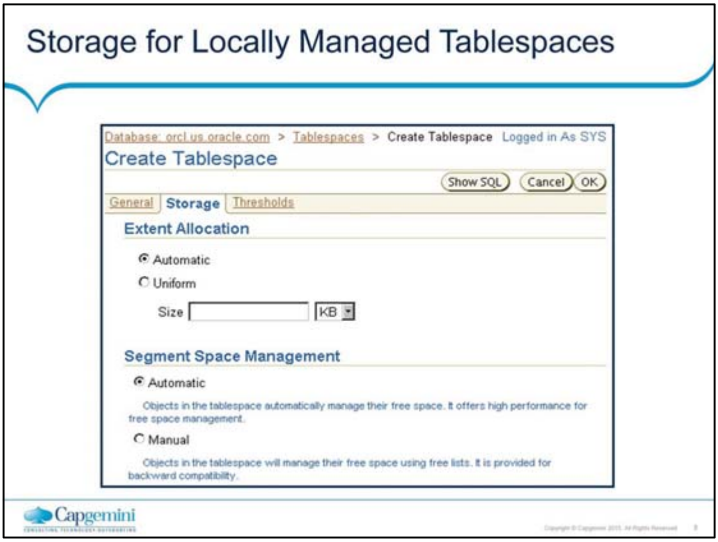
To create a tablespace, perform the following steps:

1. Navigate to the Tablespaces page. Go to the Administration tab, then click Tablespaces under the Storage heading.
2. Click the Create button.
   Note: If you want to create a tablespace that is like an existing tablespace, select an existing tablespace and select Create Like from the Actions menu. Click Go.
   The Create Tablespace General 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 within the tablespace by the Oracle database server. For a dictionary managed tablespace you must more actively manage extents and data dictionary access is required for tracking them. Dictionary managed tablespaces are being deprecated. Oracle does not suggest their use.
5. Under the Type heading, select Permanent. Permanent tablespaces store permanent database objects created by the system or users.
6. Under the Status heading, select Read Write. Read/write status means users can read and write to the tablespace after it is created. This is the default.

Creating a New Tablespace (continued)

7. In the Datafiles region of the page click Add to add datafiles to the tablespace, a tablespace must have at least one file. Bigfile tablespaces are used with ultra large databases where Oracle's Automatic Storage Management or other logical volume managers support striping or RAID, and dynamically extensible logical volumes.
8. In the Add Datafiles page, enter a file name. Accept the defaults for the File Directory and File Size.
9. Under the Storage region, select "Automatically extend datafile when full (AUTOEXTEND)" and specify an amount in the Increment field by which you want to extend the data file each time it fills. Leave the Maximum File Size set to Unlimited. Click OK. You are returned to the Create Tablespace General page.
10. Click the Storage tab. The Create Tablespace Storage page appears.
11. Accept all of the defaults on the Storage page.
12. Click the Thresholds tab to open the Thresholds page. This page enables you to set monitored thresholds for space usage. You receive advice and an option for action when the threshold is reached.
13. After specifying thresholds click OK to add the tablespace. You are returned to the Tablespaces page where you receive a confirmation of the creation of the tablespace. You can view your new tablespace in the Results section.

Note: These steps are intended to show you how to quickly create a tablespace for most situations. You may need to change some options depend on your particular system requirements.



Storage for Locally Managed Tablespaces

- Extents within a locally managed tablespace can be allocated in one of two ways:
  - Automatic: Also called autoallocate, specifies that the size of the extents within the tablespace are system managed. You cannot specify an extent size. You cannot specify automatic for a temporary tablespace.
  - Uniform: Specifies that the tablespace is managed with uniform extents of a size you specify. The default size is 1 megabyte. All extents of temporary tablespaces are of uniform size, so this is optional for a temporary tablespace. You cannot specify uniform for an undo tablespace.
- Segment space management within a locally managed tablespace:
  - Automatic: Oracle uses bitmaps to manage the free space within segments. A bitmap, in this case, is a map that describes the status of each data block within a segment with respect to the amount of space in the block available for inserting rows. As more or less space becomes available in a data block, its new state is reflected in the bitmap. Bitmaps allow Oracle to manage free space more automatically, and thus, this form of space management is called automatic segment-space management.

Storage for Locally Managed Tablespaces (continued)

Manual: This tells Oracle that you want to use free lists for managing free space within segments. Free lists are lists of data blocks that have space available for inserting rows. This form of managing space within 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.

Advantages of Locally Managed Tablespaces

Locally managed tablespaces have the following advantages over dictionary-managed tablespaces:

Local management avoids recursive space management operations. This can occur in dictionary-managed tablespaces if consuming or releasing space in an extent results in another operation that consumes or releases space in an undo segment or data dictionary table.

Because locally managed tablespaces do not record free space in data dictionary tables, they reduce contention on these tables.

Local management of extents automatically tracks adjacent free space, eliminating the need to coalesce free extents.

The sizes of extents that are managed locally can be determined automatically by the system.

Changes to the extent bitmaps do not generate undo information because they do not update tables in the data dictionary (except for special cases such as tablespace quota information).

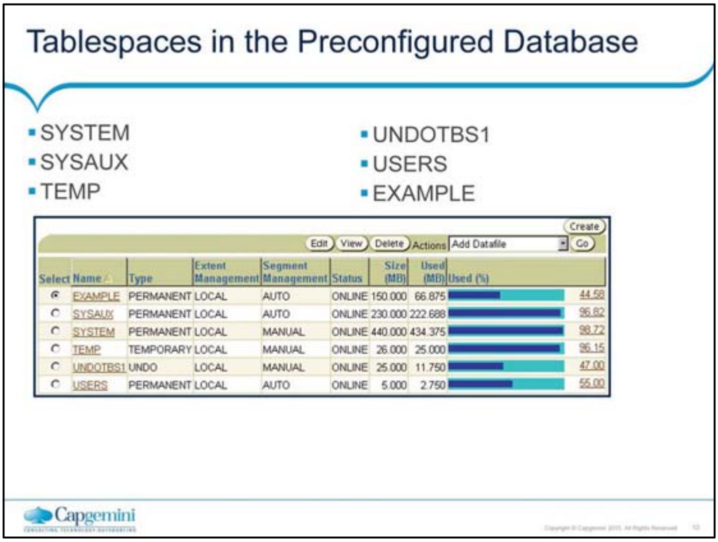
Note: If you are managing a database that has dictionary managed tablespaces and you want to convert them to locally managed, use the DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL procedure to do this. For details on the use of this procedure see the *PL/SQL Packages and Types Reference* and the *Database Administrator's Guide*.

Logging

When changes are made to objects in the tablespace, the change is logged in to what is called the redo stream. This redo stream starts in memory, the is written to the online redo log files and may be written to archive log files. You can turn this processes off. If you do turn off logging, the objects in this tablespace will be unrecoverable in the event of any kind of failure.

Thresholds

Use the Tablespace Thresholds tab to set the space used thresholds for the current database tablespace. You can choose to use the default usage thresholds for tablespaces for the database or you can specify the threshold for the percentage of space used for the current tablespace by entering the values for the Warning and Critical fields. You can also disable space used thresholds entirely for this tablespace. Thresholds are covered in more detail in lesson 15.



Select	Name	Type	Extent Management	Segment Management	Status	Size (MB)	Used (MB)	Used (%)
◉	EXAMPLE	PERMANENT	LOCAL	AUTO	ONLINE	150.000	66.875	44.58
○	SYSAUX	PERMANENT	LOCAL	AUTO	ONLINE	230.000	222.688	96.82
○	SYSTEM	PERMANENT	LOCAL	MANUAL	ONLINE	440.000	434.375	98.72
○	TEMP	TEMPORARY	LOCAL	MANUAL	ONLINE	26.000	25.000	96.15
○	UNDOTBS1	UNDO	LOCAL	MANUAL	ONLINE	25.000	11.750	47.00
○	USERS	PERMANENT	LOCAL	AUTO	ONLINE	5.000	2.750	55.00

Tablespaces in the Preconfigured Database

The following tablespaces are created for you in the preconfigured database:

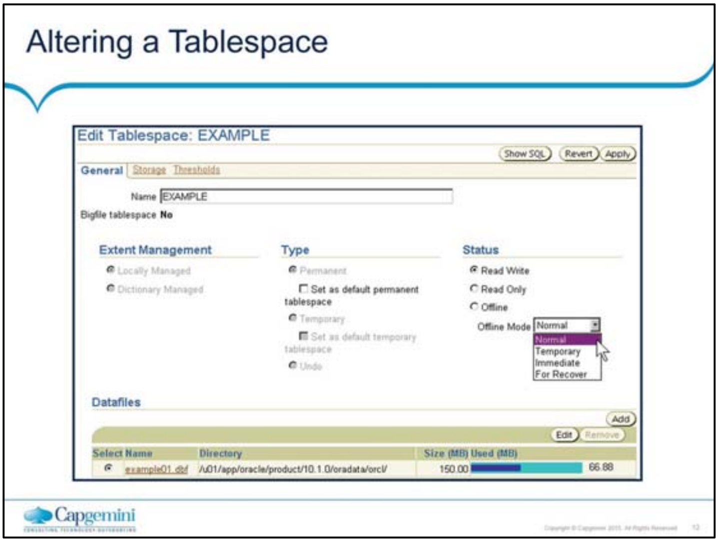
SYSTEM: The SYSTEM tablespace is used by the Oracle database server to manage the database. It contains the data dictionary and tables that contain administrative information about the database. These are all contained in the SYS schema, and can be accessed only by the user SYS, or other administrative 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 prior releases of Oracle, now use the SYSAUX tablespace. Every Oracle Database11g or higher-level database must have a SYSAUX tablespace.

TEMP: This tablespace is used to store temporary tables and indexes when processing SQL statements. It would, for example, be used for sort work space. Every database should have a temporary tablespace that is assigned to users as their temporary tablespace. 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, then Oracle assigns this tablespace to the user.

Tablespaces in the Preconfigured Database (continued)

- UNDOTBS1: This is the undo tablespace used by the database server to store undo information. Every database must have an undo tablespace that is created during database creation.
- USERS: This tablespace is used to store permanent user objects and data. In the preconfigured database, the USERS tablespace is the default tablespace for all objects created by nonsystem users. For the SYS and SYSTEM users (the system users), the default permanent tablespace remains 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 upon the sample schemas.



Altering a Tablespace

After you create a tablespace, you can later alter it in several ways as the needs of your system change.

Renaming: Simply 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. Depending on the type of tablespace, not all states may be available:

- 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 complete (commit or roll back), but no further DML operations are allowed to the tablespace except for rollback of existing transactions that previously modified blocks in the tablespace. The tablespace is online while in the read-only state.You cannot make the SYSTEM or SYSAUX tablespace 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 have choices of how to do this:Normal: A tablespace can be taken offline normally if no error conditions exist for any of the data files of the tablespace. Oracle takes 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 takes offline the data files that are not already offline, checkpointing 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 is 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 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.

Change the Size: You can add space to an existing tablespace by adding data files to the tablespace or you can change the size of an existing data file.

To add a new data file to the tablespace click Add, and fill in the information about the data file on the Add Data File page. Note that the tablespace name is fixed.

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 you get the following error:

ORA-03297: file contains used data beyond requested RESIZE value

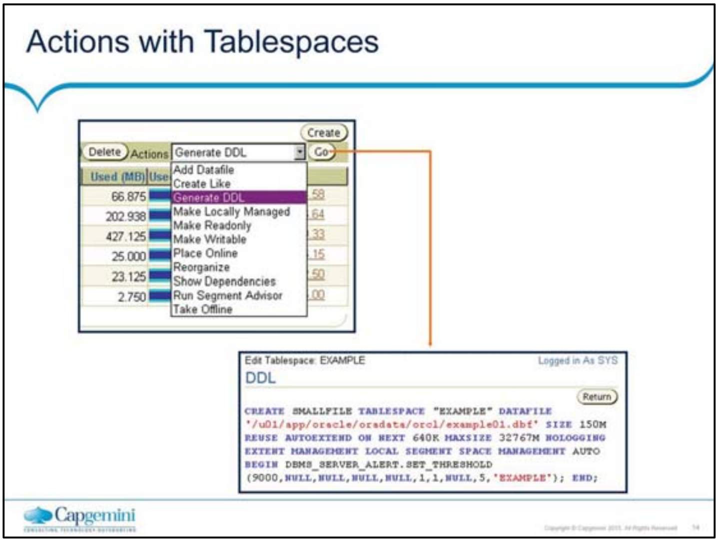
Storage Options: Click Storage to change change the logging behavior of the tablespace.

Thresholds: Click Thresholds to can change the warning and critical used space alters for the tablespace. You have three options:

Use Default Thresholds: This uses preset defaults, and you have the option of setting these defaults.

Specify Thresholds: This allows you to set thresholds for this particular tablespace.

Disable Thresholds: This turns off space usage alerts for this tablespace.



Actions with Tablespaces

With the Actions Menu you can perform a verity of tasks with your tablespaces. Select a tablespace and then the action you want to perform:

- Add Datafile: Adds a data file to the tablespace, which makes the tablespace larger.
- Create Like: Create another tablespace using the tablespace as a template.
- Generate DDL: Generate the 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: If the tablespace is currently dictionary managed, this will convert the tablespace to locally managed.
- Make Readonly: Stops all writes to the tablespace. Current transactions are allowed to compete but no new DML or other write activities are allowed to start on the tablespace.
- Make Writable: Allows DML and other write activities to be initiated on objects in the tablespace.

Actions with Tablespaces Full Notes Page

Actions with Tablespaces (continued)

Place Online: If the tablespace is currently offline, this brings it back online.

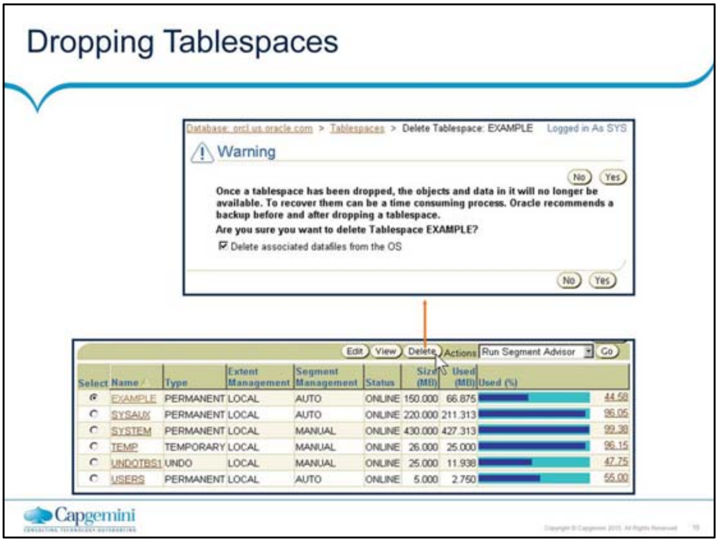
Reorganize: This starts the Reorganization Wizard, which you can use to move objects around within the tablespace to reclaim space that otherwise might not be used. This is a task that should be done during nonpeak usage of the objects in the tablespace.

Show Dependencies: This shows objects that this tablespace depends on or objects that depend on this tablespace.

Run Segment Advisor: The Segment Advisor helps you determine whether an object has space available for reclamation based on the level of space fragmentation within the object. At the tablespace level, advice is generated for every segment in the tablespace.

Take Offline: If the tablespace is currently online this will make the tablespace unavailable. The tablespace is not deleted or drop, just unavailable.

Note: Not all actions are available for each tablespace. Depending on the type of tablespace selected, some actions cannot be performed. For example, you cannot take the SYSTEM tablespace offline, nor can you make an undo tablespace read-only.

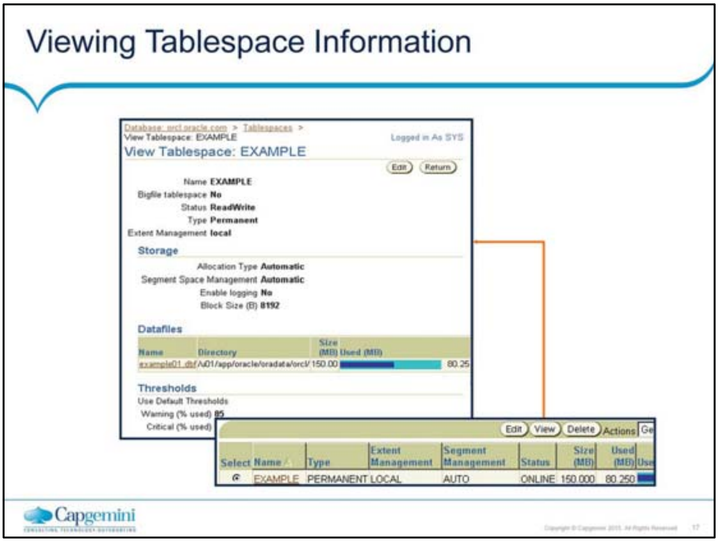


Dropping Tablespaces

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. You can optionally direct Oracle to delete the operating system files (data files) that constituted the dropped tablespace. If you do not direct Oracle 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 to delete them.

You cannot drop a tablespace that contains any active segments. For example, if a table in the tablespace is currently being used or 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

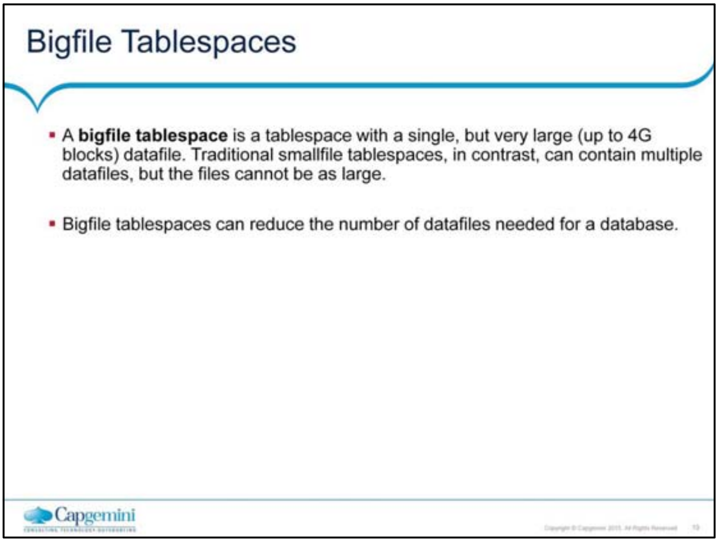
Click View to see information about the selected tablespace. In the View Tablespace page, you can also click edit to alter the tablespace.

Obtaining 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
- Temp file information:
  - DBA_TEMP_FILES
  - V$TEMPFILE

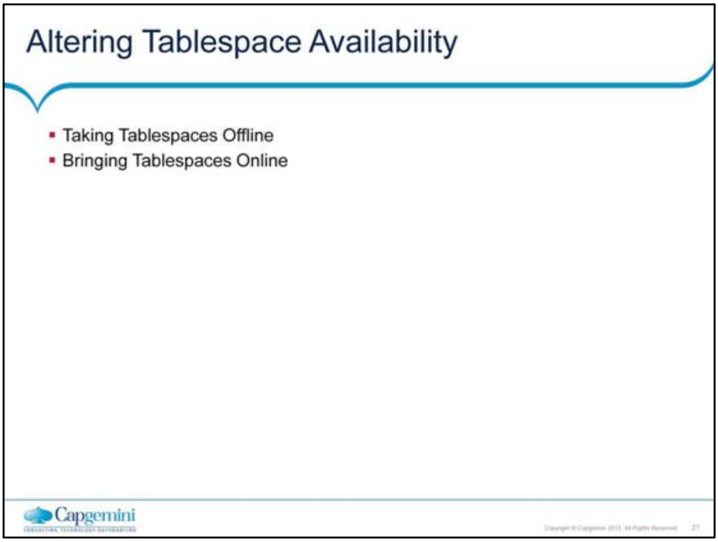
Locally Managed Tablespaces

- Fast, concurrent space operations. Space allocations and deallocations modify locally managed resources (bitmaps stored in header files).
- Enhanced performance
- Readable standby databases are allowed, because locally managed temporary tablespaces do not generate any undo or redo.
- Space allocation is simplified, because when the AUTOALLOCATE clause is specified, the database automatically selects the appropriate extent size.
- User reliance on the data dictionary is reduced, because the necessary information is stored in file headers and bitmap blocks.
- Coalescing free extents is unnecessary for locally managed tablespaces.


Bigfile Tablespaces
A **bigfile tablespace** is a tablespace with a single, but very large (up to 4G blocks) datafile. Traditional smallfile tablespaces, in contrast, can contain multiple datafiles, but the files cannot be as large.
Bigfile tablespaces can reduce the number of datafiles needed for a database.
Capgemini

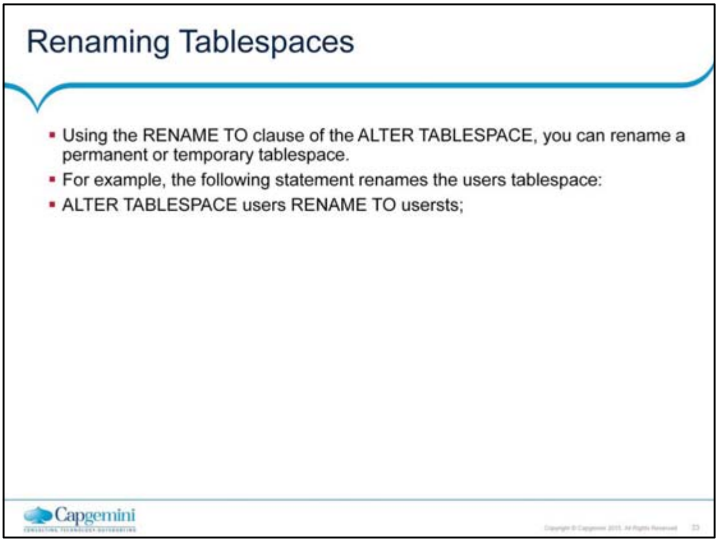
Temporary Tablespaces

- A temporary tablespace contains transient data that persists only for the duration of the session.
- It improve the concurrency of multiple sort operations, reduce their overhead, and avoid Oracle Database space management operations


Altering Tablespace Availability
Taking Tablespaces Offline
Bringing Tablespaces Online
Capgemini

Using Read-Only Tablespaces

- Making a tablespace read-only prevents write operations on the datafiles in the tablespace.
- The primary purpose of read-only tablespaces is to eliminate the need to perform backup and recovery of large, static portions of a database.
- Read-only tablespaces also provide a way to protecting historical data so that users cannot modify it.
- Making a tablespace read-only prevents updates on all tables in the tablespace, regardless of a user's update privilege level.


Renaming Tablespaces
Using the RENAME TO clause of the ALTER TABLESPACE, you can rename a permanent or temporary tablespace.
For example, the following statement renames the users tablespace:
ALTER TABLESPACE users RENAME TO usersts;
Capgemini

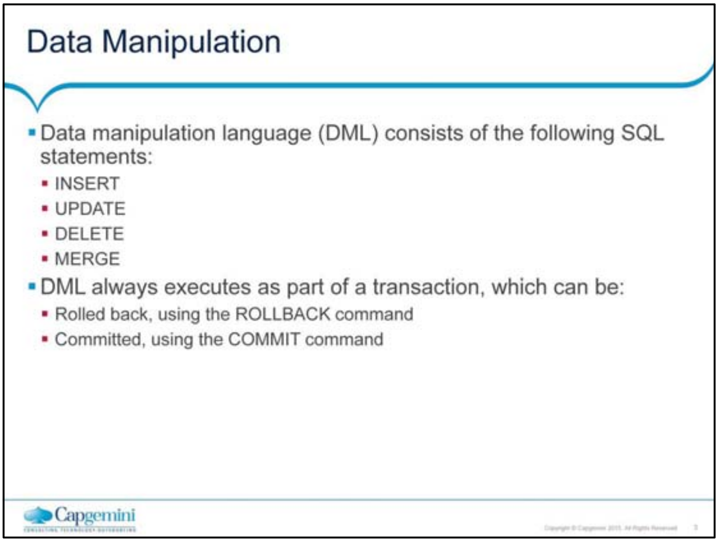
Oracle 11g DBA Fundamentals Overview

Lesson 07: Managing the Undo Tablespace

Objectives

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

Capgemini

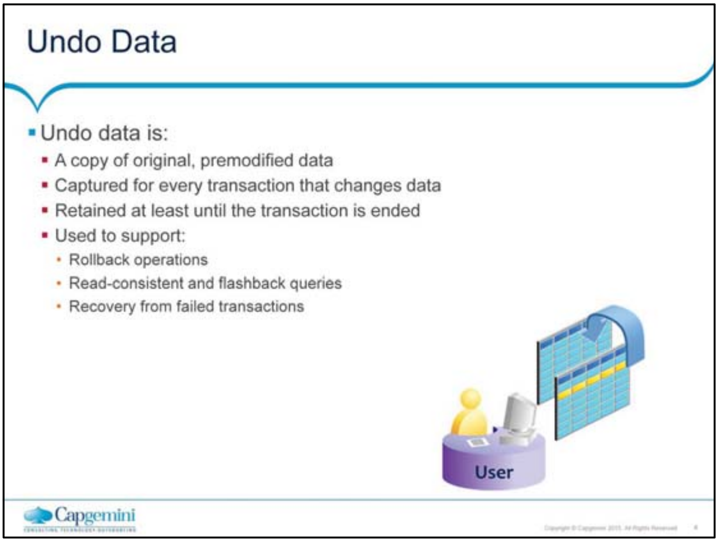


Data Manipulation

Data is manipulated, or modified, by the DML class of SQL statements: INSERT, UPDATE, DELETE, and MERGE. These statements execute as part of a transaction, which starts with the first successful DML statement and ends with either a COMMIT or ROLLBACK command. A transaction is either entirely committed or entirely rolled back.

Rollback may also occur if there is a process or system failure.

Note: The MERGE command performs a combination of inserts and updates to merge data from one table into another. It is covered in the lesson titled “Managing Data and Concurrency.”



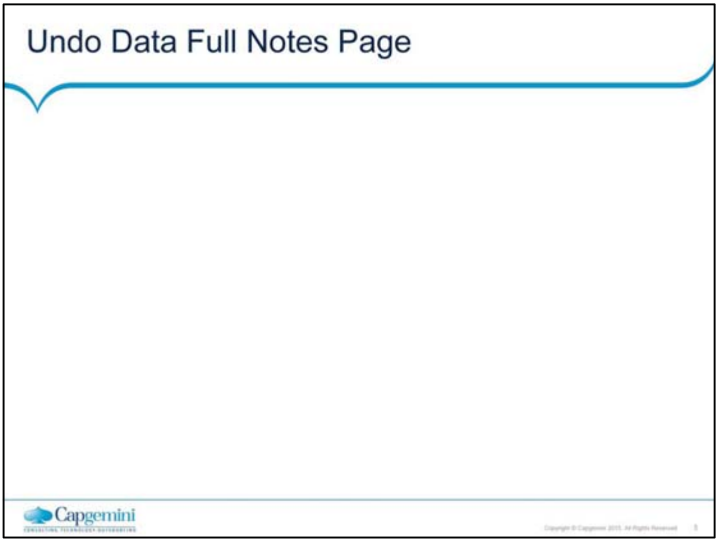
Undo Data

The Oracle database saves the old value (undo data) when a process changes data in a database. It stores the data as it existed before being modified. Capturing undo data enables you to roll back your uncommitted data. Undo also supports read-consistent and flashback queries.

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. As long as the undo information is retained, the Oracle database can reconstruct data to satisfy read-consistent queries.

Flashback queries are queries that 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.

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.



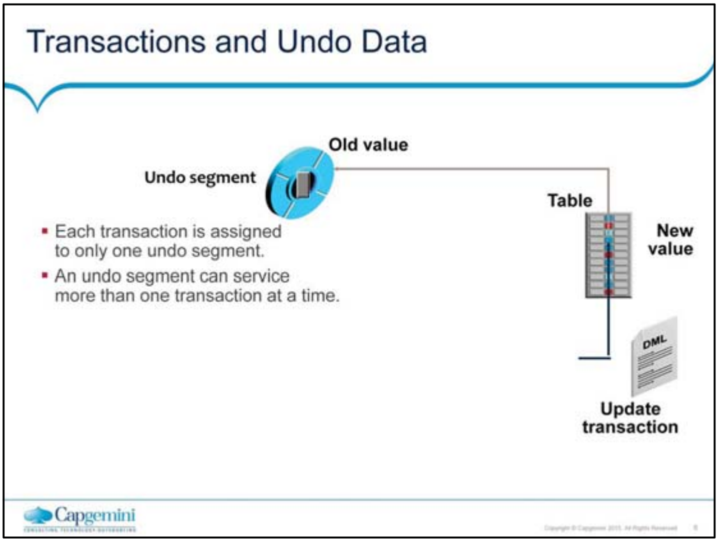
Undo Data (continued)

In case of a failed transaction, the safest behavior is chosen, and the Oracle database reverses all changes made by a user, restoring the original data.

Undo information is retained for all transactions, at least until the transaction is ended by:

- Users undoing the transaction (rolls back)
- Users ending a transaction (commits)
- User session abnormally terminating (rolls back)
- User session normally terminating with an exit (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.



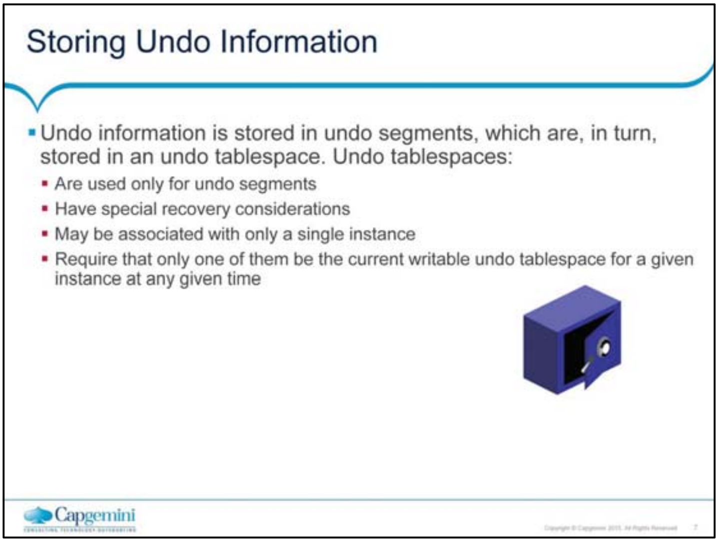
Transactions and Undo Data

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 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 11g.



Storing Undo Information

Undo segments can exist only in a specialized form of tablespace called an undo tablespace. Although a database may have many undo tablespaces, only one of them at a time can be designated as the current one to which undo data is written.

Undo segments are always owned by SYS. Because the 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 managed like any other tablespace with the exception of recovery. 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	Rolling forward database changes
Stored in	Undo segments	Redo log files
Protects against	Inconsistent reads in multiuser systems	Data loss

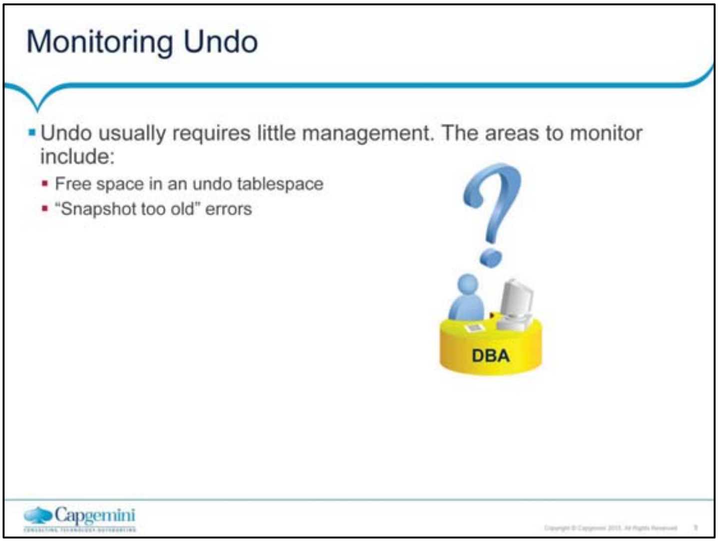
Capgemini

Undo Data Versus Redo Data

Undo data and redo data seem similar at first, but they serve different purposes. Undo data is needed in case there is the need to undo a change, and this occurs for read-consistency and rollback. Redo data is needed in case there is the need to perform the changes again, in case they are lost for some reason.

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, it is typically multiplexed. So, there are multiple copies of the redo data on the disk. Even though the changes may not have yet been written to the data files where the table’s blocks are actually stored, guaranteeing that the changes have been written to the redo log file is enough.

A power outage that occurs just before committed changes have been reflected into the redo files does not cause a problem because the transaction has been committed. So, when the system starts up again, it is able to roll forward any redo records that are not yet reflected in data files at the time of the outage.



Monitoring Undo

Most of the time, undo is managed automatically by the instance with little need for database administrator (DBA) intervention. A few things that may require administrator involvement include:

- Insufficient space for undo
- Users receiving the ORA-01555 snapshot too old error messages

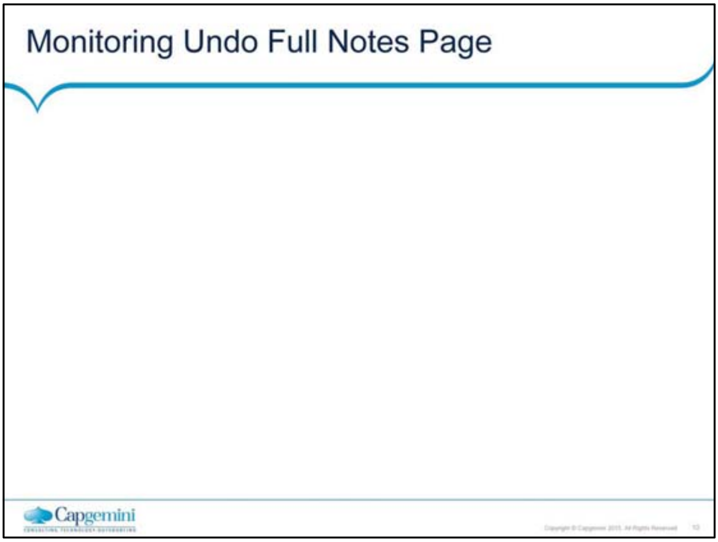
Undo information is always retained until a transaction ends. This means that if extremely large amounts of data are deleted or updated (Insert operations consume very little undo space because the original image of inserted data is a null value.) without being committed, the undo tablespace must be equally large to contain the original data. Imagine a case where a 50-GB table had all rows deleted with the following command:

```
SQL> DELETE FROM reallybigtable;
```

The undo tablespace would be required to make room for 50 GB of original information just in case the user who issued this statement changed his or her mind and wanted to roll back the change. When the undo tablespace runs out of room for undo data, users receive an error message such as the following:

```
ORA-01650: unable to extend rollback segment
```

Proactive monitoring detects space problems in an undo tablespace before they affect users.

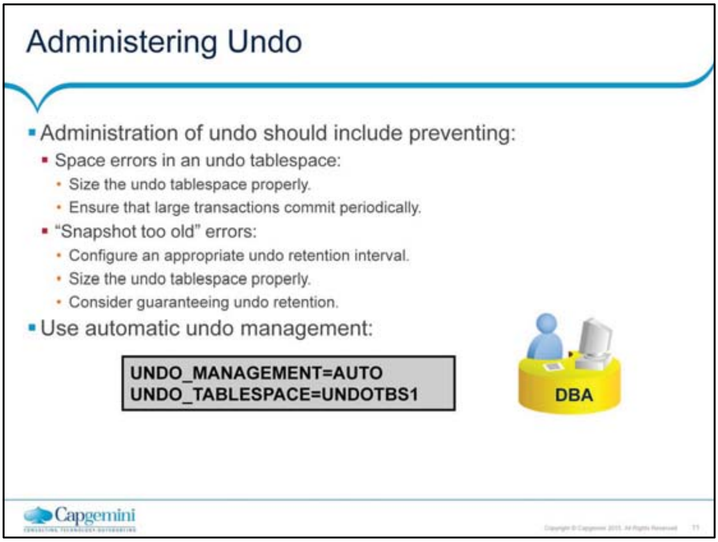


Monitoring Undo (continued)

Another problem that the administrator may encounter with undo information is when a query needs to access undo information that has already been overwritten. This may happen in a long-running or flashback query. When a query needs a “snapshot” of data as of some time in the past, and reconstructing that snapshot requires undo data that no longer exists, the query returns the following error:

ORA-01555: snapshot too old

This can happen because the Oracle database presents the user with a consistent view of the data as it exists at the time the query starts running. If there are uncommitted changes to the table being queried, the Oracle database reads the undo data to get the committed version of data. This is read consistency. If the query runs so long that in the meantime those modifications are indeed committed, and subsequently their undo data is released and overwritten, then the long-running query no longer can see a consistent view of the data as of when it first began to run. For this reason, undo retention should be configured to accommodate the longest-running query.



Administering Undo

It is recommended that you use automatic undo management, configured by setting the UNDO_MANAGEMENT initialization parameter to AUTO. Manual undo management is supported for backward compatibility with Oracle8i and earlier versions, but requires more DBA interaction.

With automatic undo management, the DBA manages undo at the tablespace level, controlling, with the UNDO_TABLESPACE initialization parameter, which undo tablespace an instance uses. After selecting the undo tablespace, the administrator need worry only about providing sufficient space and configuring an undo retention interval.

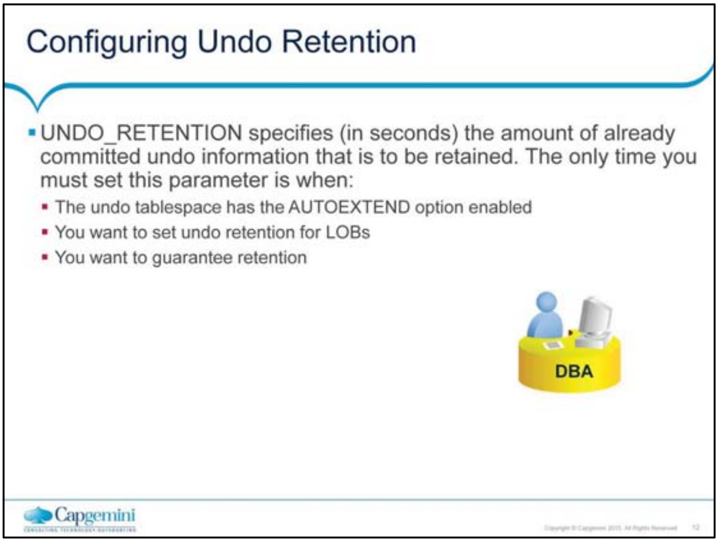
With manual management, the DBA must also consider:

Segment sizing, including maximum extents and extent sizing

Identifying and eliminating blocking transactions

Creating enough rollback segments to handle transactions (In manual mode, undo segments are known as rollback segments.)

Choosing a tablespace to contain the rollback segments (Undo tablespaces are used only with automatic undo management.)



Configuring Undo Retention

UNDO_RETENTION specifies (in seconds) the low threshold value of undo retention. For the AUTOEXTEND 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. For fixed-size undo tablespaces, the system automatically tunes for the maximum 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, for the three cases listed, the UNDO_RETENTION setting is used. In cases other than these three, this parameter is ignored.

Configuring Undo Retention Full Notes Page

Capgemini

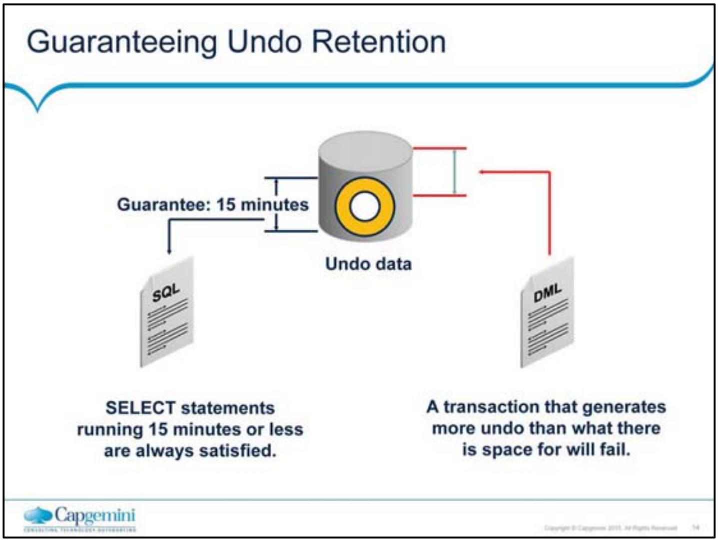
Configuring Undo Retention (continued)

Undo information is divided into three categories:

Uncommitted undo information: Supports a currently running transaction, and it is required if a user wants to roll back or if the transaction has failed. Uncommitted undo information is never overwritten.

Committed undo information: Is no longer needed to support a running transaction, but it 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: 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

The default undo behavior is to overwrite committed transactions that have 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:

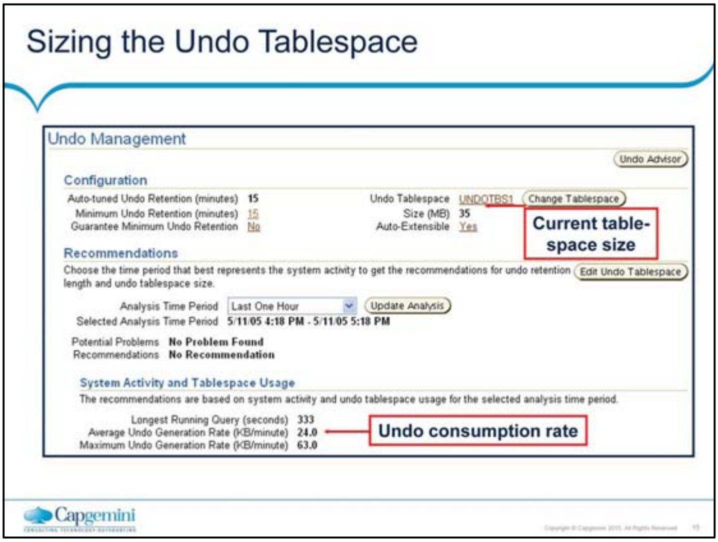
```
SQL> ALTER TABLESPACE undotbs1
RETENTION GUARANTEE;
```

To return a guaranteed undo tablespace to its normal setting, use the following command:

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

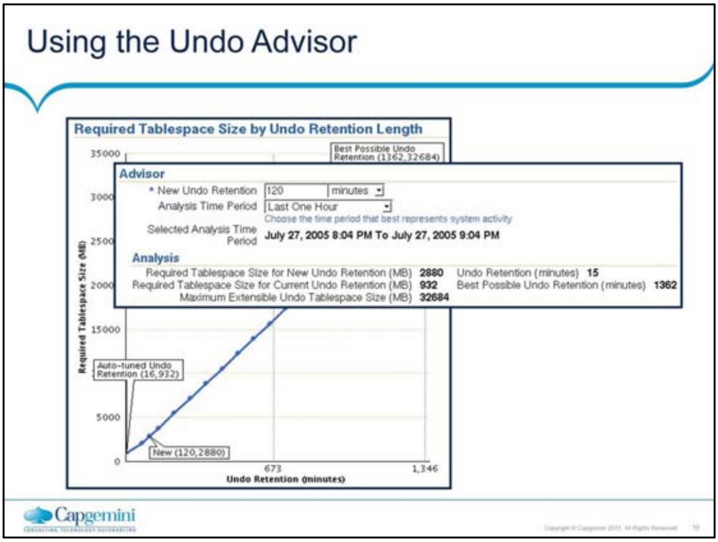
The retention guarantee applies only to undo tablespaces. Attempts to set it on a non-undo tablespace result in the following error:

```
SQL> ALTER TABLESPACE example
RETENTION GUARANTEE;
ERROR at line 1:
ORA-30044: 'Retention' can only specified for
undo tablespace
```



Sizing the Undo Tablespace

Undo tablespaces must be sized so that they can contain the original information for all transactions. Clicking the Undo Management link on the Enterprise Manager's Administration page reveals an overview of system undo, including current settings, undo consumption per minute, and the length of the longest-running query observed during a given time period. Data files belonging to an undo tablespace can automatically extend when they run out of free space. Oracle Corporation recommends that data files that are associated with undo tablespaces, unlike other tablespaces, should not have automatic extension enabled. When first determining undo space requirements, you may want to enable automatic extension of the data files, but after you have properly sized the tablespace, you must disable it. Disabling automatic extension in an undo tablespace's data files prevents a single user from inadvertently consuming large amounts of disk space by neglecting to commit transactions.

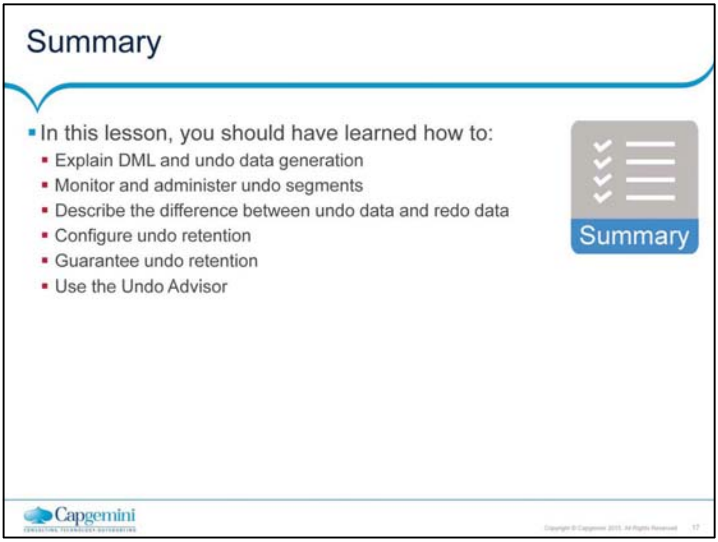


Using the Undo Advisor

The Undo Advisor is accessed through the Undo Management properties page. It provides an estimate of the undo tablespace size required to satisfy a given undo retention.

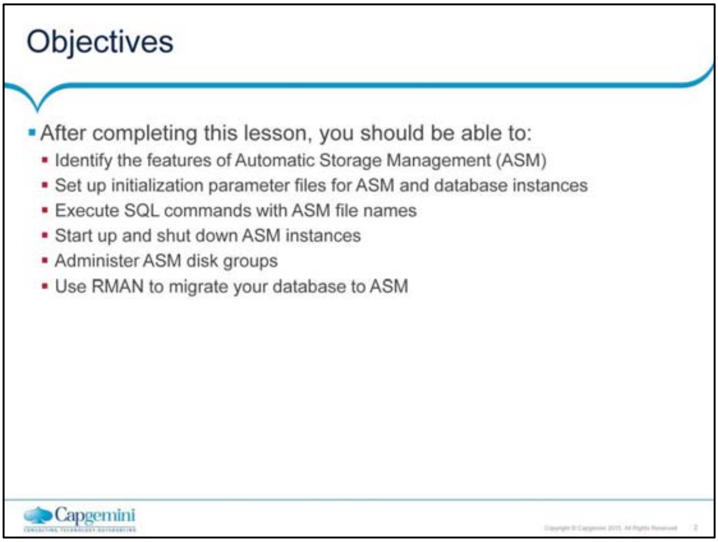
Enter the desired retention period, and the analysis region of the advisor displays the tablespace size required to support the retention period. You can also click a point on the graph to see the tablespace size required to support the selected period.

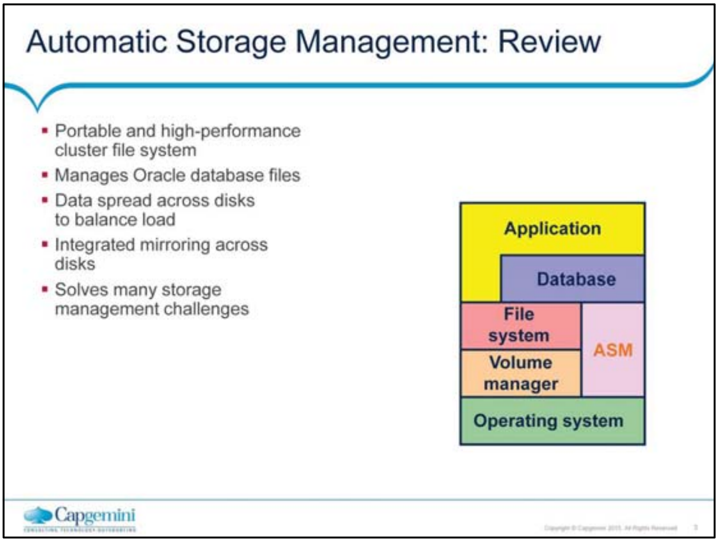
After you have selected an undo retention period, click OK to implement the new retention period.


Summary
In this lesson, you should have learned how to:
Explain DML and undo data generation
Monitor and administer undo segments
Describe the difference between undo data and redo data
Configure undo retention
Guarantee undo retention
Use the Undo Advisor
Summary
Capgemini

Oracle 11g DBA Fundamentals Overview

Lesson 08: Using Automatic Storage Management


Objectives
After completing this lesson, you should be able to:
Identify the features of Automatic Storage Management (ASM)
Set up initialization parameter files for ASM and database instances
Execute SQL commands with ASM file names
Start up and shut down ASM instances
Administer ASM disk groups
Use RMAN to migrate your database to ASM
Capgemini



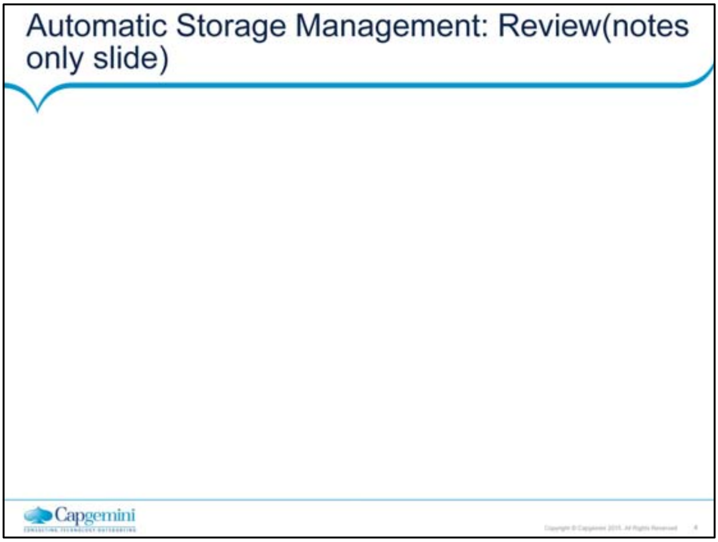
Automatic Storage Management: Review

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

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

ASM can maintain redundant copies of data to provide fault tolerance, or it can be built on top of vendor-supplied reliable 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 DBAs' time by automating manual storage and thereby increasing their ability to manage larger databases and more of them with increased efficiency.

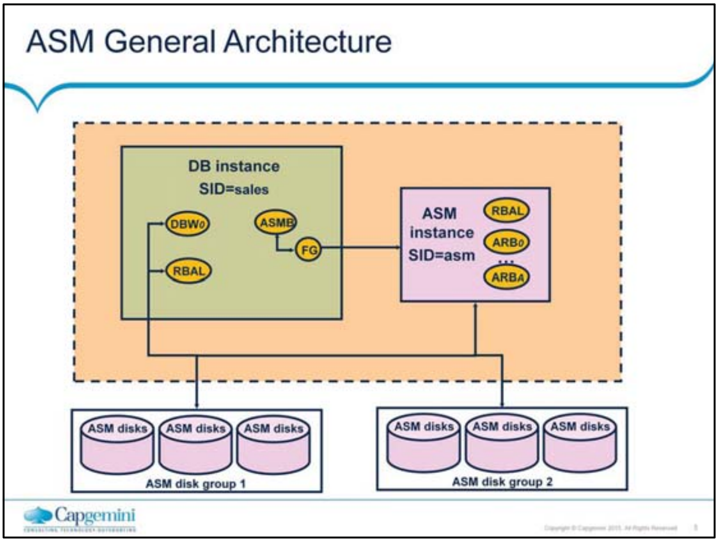


Automatic Storage Management: Review (continued)

ASM divides files into allocation units (AUs) and spreads the AUs for each file evenly across all the disks. ASM uses an index technique to track the placement of each AU. When your storage capacity changes, ASM does not restripe all of the data, but moves an amount of data proportional to the amount of storage added or removed to evenly redistribute the files and maintain a balanced load across the disks. This is done while the database is active.

You can increase the speed of a rebalance operation, or lower it to reduce the impact on the I/O subsystem. ASM provides mirroring protection without the need to purchase a third-party Logical Volume Manager. One unique advantage of ASM is that the mirroring is applied on a file basis, rather than on a volume basis. Therefore, the same disk group can contain a combination of files protected by mirroring, along with those that are not protected at all.

ASM supports data files, log files, control files, archive logs, RMAN backup sets, and other Oracle database file types. ASM supports Real Application Clusters and eliminates the need for a Cluster Logical Volume Manager or a Cluster File System.



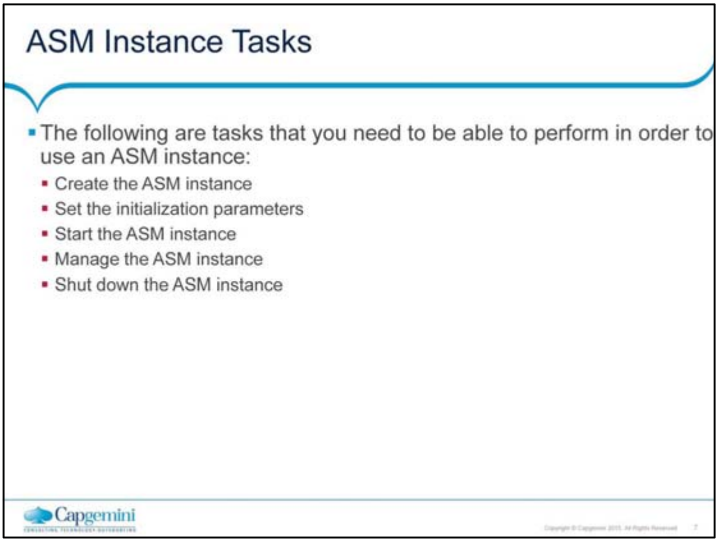
ASM General Architecture

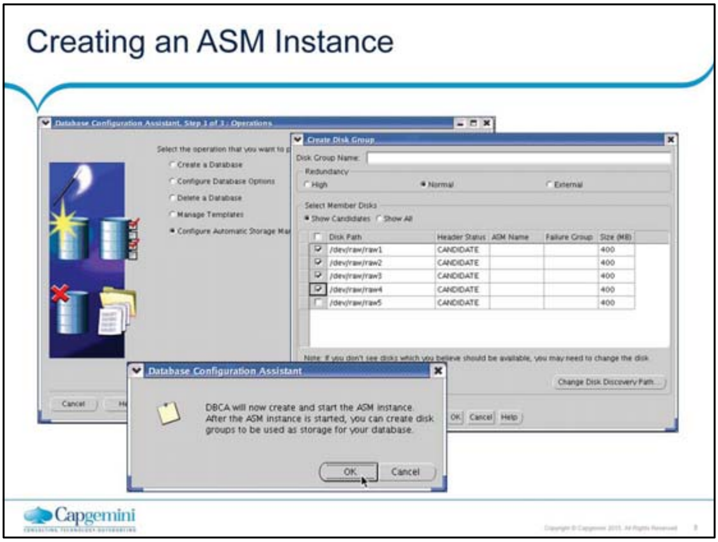
To use ASM, you must start a special instance, called an ASM instance, before you start your database instance. ASM instances do not mount databases, instead they manage the metadata needed to make ASM files available to ordinary database instances. Both ASM instances and database instances have access to some common set of disks called disk groups. Database instances access the contents of ASM files directly, communicating with an ASM instance only to get information about the layout of these files.

An ASM instance contains two new background processes. One coordinates rebalance activity for disk groups. It is called RBAL. The second one performs the actual rebalance AU movements. There can be many of these at a time, and they are called ARB0, ARB1, and so forth. An ASM instance also has some of the same background processes as a database instance, including SMON, PMON, LGWR, DBWR, and CKPT.

ASM General Architecture (continued)

Each database instance using ASM has two new background processes called ASMB and RBAL. RBAL performs global opens of the disks in the disk groups. At database instance startup, ASMB connects as a foreground process into the ASM instance. Communication between the database and the ASM instance is performed via this bridge. This includes physical file changes such as data file creation and deletion. Over this connection, periodic messages are exchanged to update statistics and to verify that both instances are healthy.


ASM Instance Tasks
The following are tasks that you need to be able to perform in order to use an ASM instance:
Create the ASM instance
Set the initialization parameters
Start the ASM instance
Manage the ASM instance
Shut down the ASM instance
Capgemini



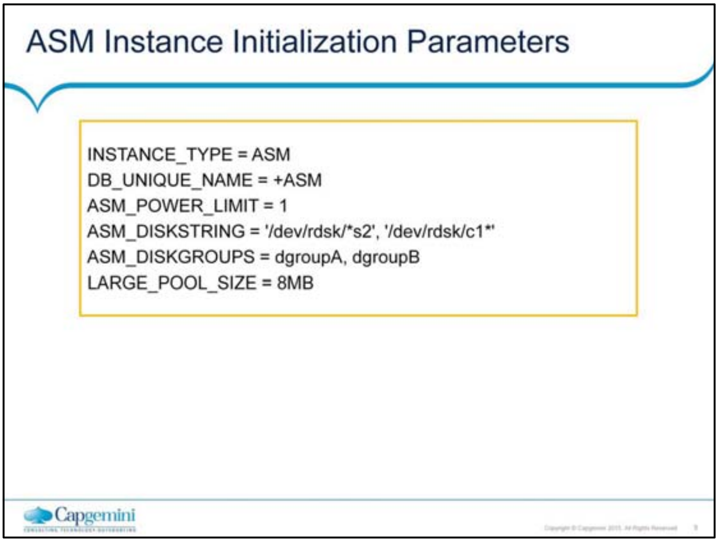
Creating an ASM Instance

You create an ASM instance by running the Database Configuration Assistant (DBCA). On the first screen, choose the option to Configure Automatic Storage Management, and follow the steps. The ASM instance is created and started for you. Then you are guided through the process of defining disk groups for the instance.

As part of the ASM instance creation process, the DBCA automatically creates an entry into the oratab file. This entry is used for discovery purposes. On the Windows platform where a services mechanism is used, the DBCA automatically creates an Oracle Service and the appropriate registry entry to facilitate the discovery of ASM instances.

When an ASM instance is configured, the DBCA creates an ASM instance parameter file and an ASM instance password file.

If you were to first create an ASM-enabled database, the DBCA determines whether an ASM instance already exists on your host. If ASM instance discovery returns an empty list, the DBCA creates a new ASM instance.



ASM Instance Initialization Parameters

INSTANCE_TYPE should be set to ASM for ASM instances.

DB_UNIQUE_NAME specifies the service provider name for which this ASM instance manages disk groups. The default value of +ASM must be modified only if you run multiple ASM instances on the same node.

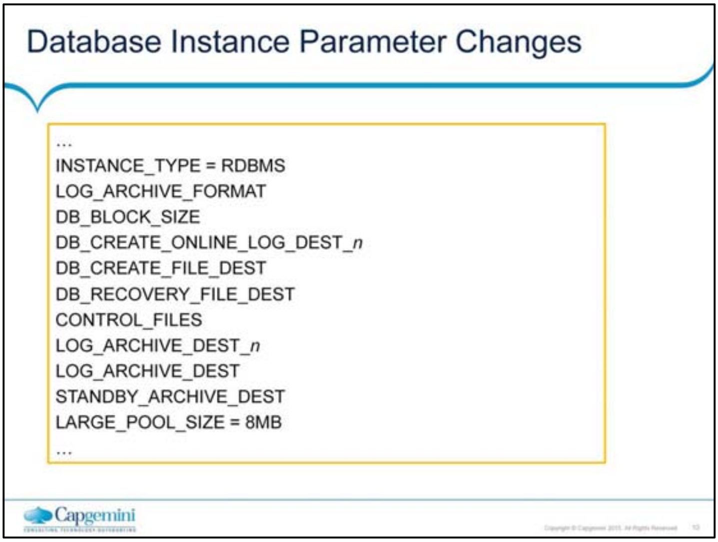
ASM_POWER_LIMIT controls the speed for a rebalance operation. Values range from 1 through 11, with 11 being the fastest. If omitted, this value defaults to 1. The number of slaves is derived from the parallelization level specified in a manual rebalance command (POWER), or by the ASM_POWER_LIMIT parameter.

ASM_DISKSTRING is an operating system–dependent value used by ASM to limit the set of disks considered for discovery.

ASM_DISK_GROUPS 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.

The INSTANCE_TYPE parameter is the only parameter that you must define. All other ASM parameters have default values that are suitable for most environments.

Note: If the ASM environment has been created using the command line instead of EM, then the disk groups must be created before they can be mounted.



Database Instance Parameter Changes

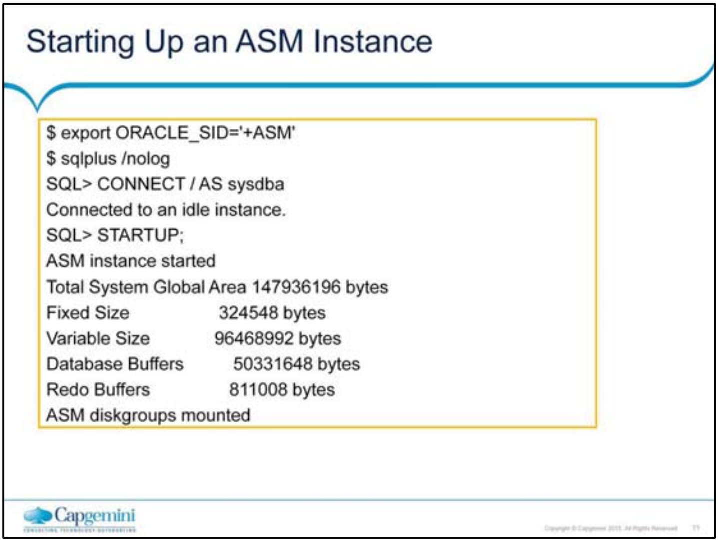
INSTANCE_TYPE defaults to RDBMS and specifies that this instance is an RDBMS instance.

LOG_ARCHIVE_FORMAT is ignored if LOG_ARCHIVE_DEST is set to an incomplete ASM file name, such as +dGroupA. If LOG_ARCHIVE_DEST is set to an ASM directory (for example, +dGroupA/myarchlogdir/), then LOG_ARCHIVE_FORMAT is used and the files are non-OMF. Unique file names for archived logs are automatically created by the Oracle database.

The following parameters accept the multifile creation context form of ASM file names as a destination:

- DB_CREATE_ONLINE_LOG_DEST_n
- DB_CREATE_FILE_DEST
- DB_RECOVERY_FILE_DEST
- CONTROL_FILES
- LOG_ARCHIVE_DEST_n
- LOG_ARCHIVE_DEST
- STANDBY_ARCHIVE_DEST

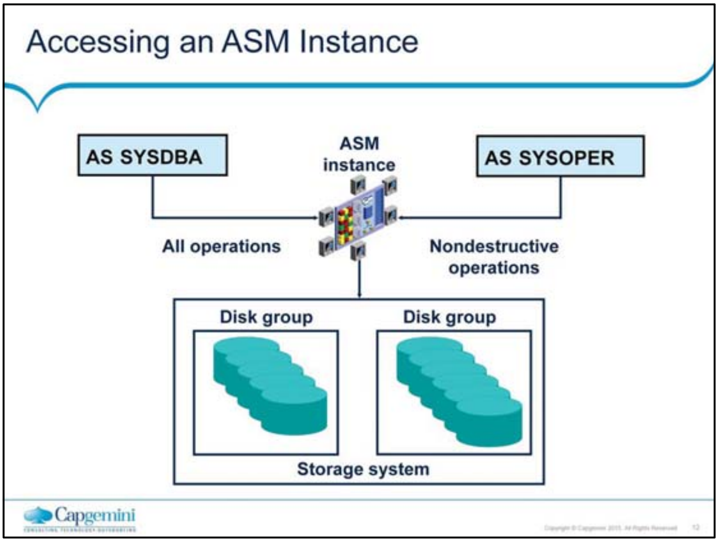
Note: Because allocation unit maps for ASM files are allocated from the LARGE_POOL, you must set the LARGE_POOL_SIZE initialization parameter to at least 8 MB, preferably higher.



Starting Up an ASM Instance

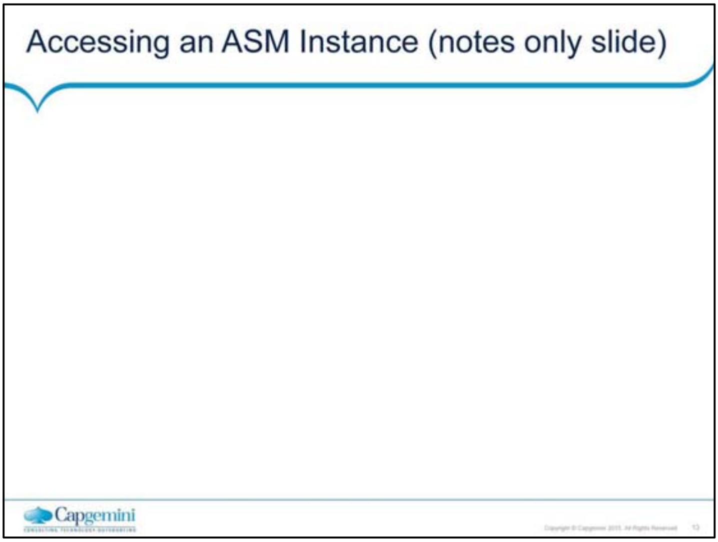
ASM instances are started similarly to database instances except that the initialization parameter file contains an entry like INSTANCE_TYPE=ASM. When this parameter is set to the value ASM, it informs the Oracle executable that an ASM instance is starting, not a database instance. Also, the ORACLE_SID variable must be set to the ASM instance name. When the ASM instance starts up, the mount stage attempts to mount the disk groups specified by the ASM_DISKGROUPS initialization parameter rather than mounting a database, as is done with non-ASM instances.

Other STARTUP clauses have comparable interpretation for ASM instances as they do for database instances. OPEN is invalid for an ASM instance. NOMOUNT starts up the ASM instance without mounting any disk group.



Accessing an ASM Instance

ASM instances do not have a data dictionary, so the only way to connect to one is by using OS authentication, that is, SYSDBA or SYSOPER. To connect remotely, a password file must be used. Normally, the SYSDBA privilege is granted through the use of an operating system group. On UNIX, this is typically the dba group. By default, members of the dba group have SYSDBA privilege on all instances on the node, including the ASM instance. Users who connect to the ASM instance with the SYSDBA privilege have administrative access to all disk groups in the system. The SYSOPER privilege is supported in ASM instances and limits the set of allowable SQL commands to the minimum required for basic operation of an already configured system.

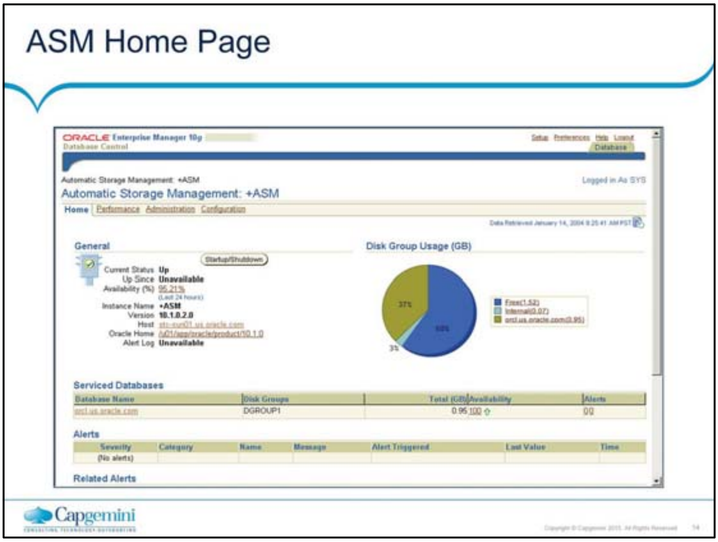


Accessing an ASM Instance (continued)

The following commands are available to SYSOPER users:

- STARTUP/SHUTDOWN
- ALTER DISKGROUP MOUNT/DISMOUNT
- ALTER DISKGROUP ONLINE/OFFLINE DISK
- ALTER DISKGROUP REBALANCE
- ALTER DISKGROUP CHECK
- SELECT all V$ASM_* views

All other commands, such as CREATE DISKGROUP, ADD/DROP/RESIZE DISK, and so on, require the SYSDBA privilege and are not allowed with the SYSOPER privilege.

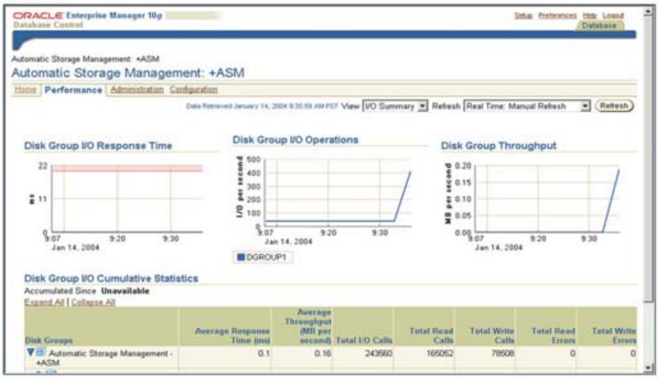


ASM Home Page

Enterprise Manager provides a user-friendly graphical interface to the Oracle database management, administration, and monitoring tasks. Oracle Database 11g extends the existing functionality to transparently support the management, administration, and monitoring of Oracle databases that use ASM storage. It also adds support for the new management tasks required for administration of ASM instance and ASM disk groups.

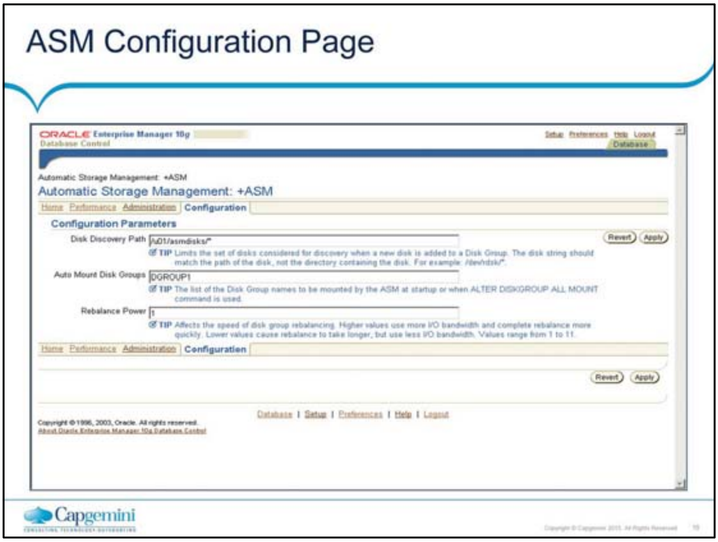
The ASM home page shows the status of the ASM instance along with the metrics and alerts generated by the collection mechanisms. It also provides the startup and shutdown functionality. Clicking the Alert link takes the user to an alert details page. The DiskGroup Usage chart shows space used by each client database along with free space.

ASM Performance Page



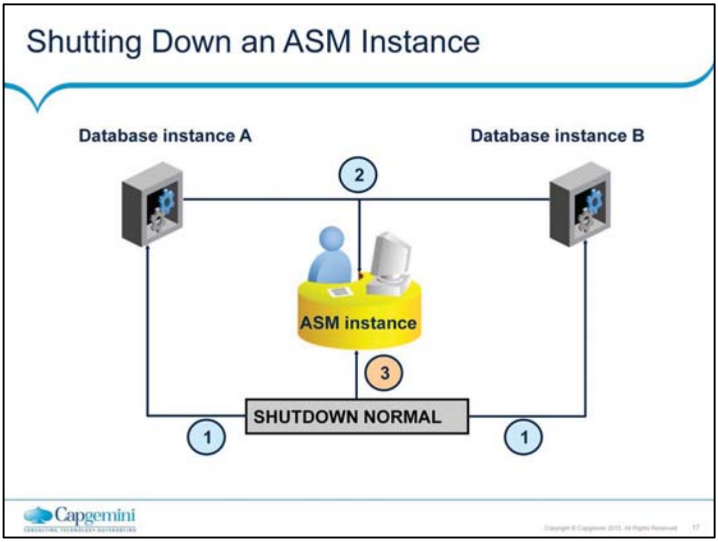
ASM Performance Page

The Performance tab of the Automatic Storage Management page shows the I/O response time and throughput for each disk group. You can further drill down to view disk-level performance metrics.



ASM Configuration Page

The Configuration tab of the Automatic Storage Management page enables you to view or modify the initialization parameters of the ASM instance.



Shutting Down an ASM Instance

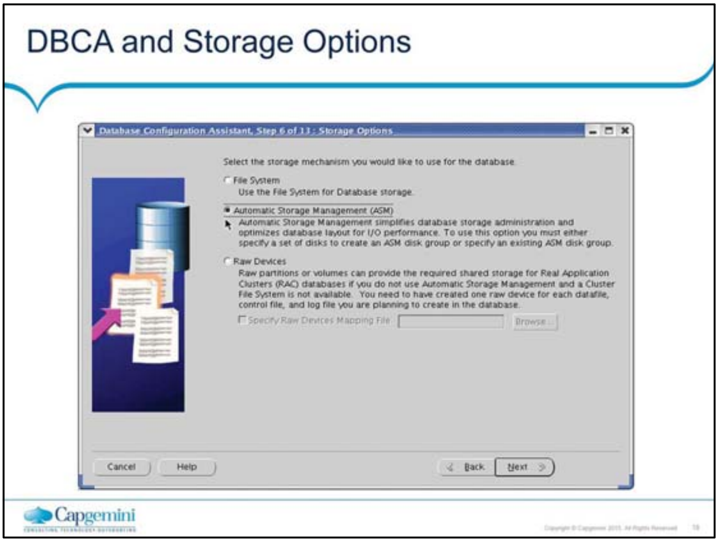
When you attempt to shutdown an ASM instance in the NORMAL, IMMEDIATE, or TRANSACTIONAL modes, it will only succeed if there are no database instances connected to the ASM instance. If there is at least one connected instance, you will receive the following error:

ORA-15097: cannot SHUTDOWN ASM instance with connected RDBMS instance

If you perform a SHUTDOWN ABORT on the ASM instance, it will shutdown, and it will require recovery at the time of the next startup. Any connected database instances will also eventually shutdown, reporting the following error:

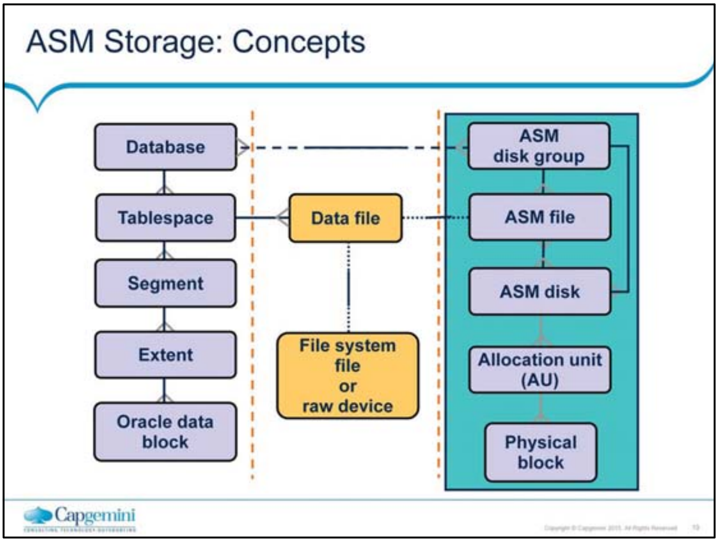
ORA-15064: communication failure with ASM instance

In a single ASM instance configuration, if the ASM instance fails while disk groups are open for update, then after the ASM instance reinitializes, it reads the disk group's log and recovers all transient changes. With multiple ASM instances sharing disk groups, if one ASM instance fails, another ASM instance automatically recovers transient ASM metadata changes caused by the failed instance. The failure of a database instance does not affect ASM instances. The ASM instance should be started automatically whenever the host is rebooted. ASM instance is expected to use the automatic startup mechanism supported by the underlying operating system. Note that file system failure usually crashes a node.



DBCA and Storage Options

In order to support ASM as a storage option, this screen appears in the Database Configuration Assistant (DBCA) when creating a database. This allows you to choose the storage options: file system, ASM, or raw devices.

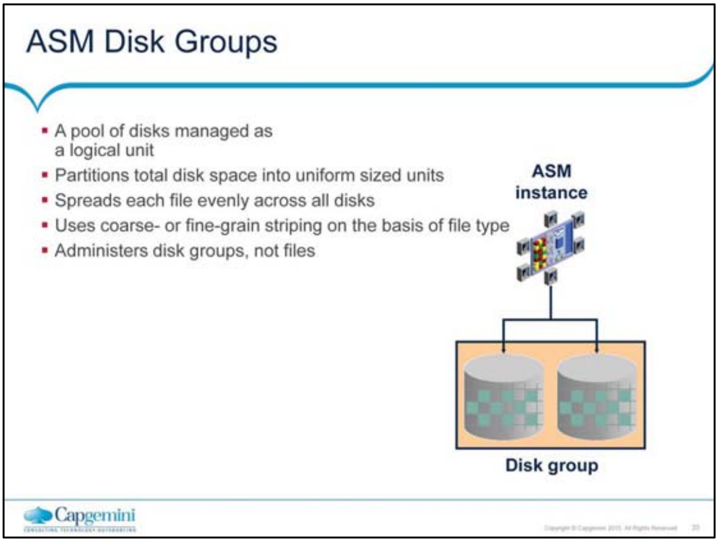


ASM Storage: Concepts

ASM does not eliminate any preexisting database functionality. Existing databases are able to operate as they always have. You can create new files as ASM files and leave existing files to be administered in the old way, or you can eventually migrate them to ASM.

The diagram depicts the relationships that exist between the various storage components inside an Oracle database with ASM available. The left and middle parts of the diagram show the relationships that exist in previous releases. On the right are the new concepts introduced by ASM. Database files can be stored as ASM files. At the top of the new hierarchy are ASM disk groups. Any single ASM file is contained in only one disk group. However, a disk group may contain files belonging to several databases, and a single database may use storage from multiple disk groups. As you can see, one disk group is made up of multiple ASM disks, and each ASM disk belongs to only one disk group. ASM files are always spread across all the ASM disks in the disk group. ASM disks are partitioned in allocation units (AU) of one megabyte each. An allocation unit is the smallest contiguous disk space that ASM allocates. ASM does not allow an Oracle block to be split across allocation units.

Note: This graphic deals with only one type of ASM file: data file. However, ASM can be used to store other database file types.

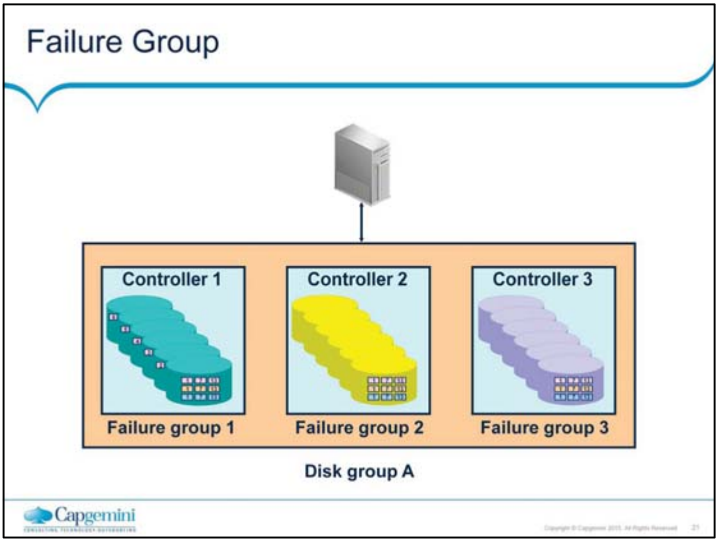


ASM Disk Groups

A disk group is a collection of disks managed as a logical unit. Storage is added and removed from disk groups in units of ASM disks. Every ASM disk has an ASM disk name, which is a name common to all nodes in a cluster. The ASM disk name abstraction is required because different hosts can use different names to refer to the same disk.

ASM always spreads files evenly in 1 MB allocation unit (AU) chunks across all the disks in a disk group. This is called coarse striping. That way, ASM eliminates the need for manual disk tuning. However, disks in a disk group should have similar size and performance characteristics to obtain optimal I/O. For most installations there is only a small number of disk groups. For instance, one disk group for a work area, and one for a recovery area. For files, such as log files, that require low latency, ASM provides fine-grained (128 KB) striping. Fine striping stripes each AU. Fine striping breaks up medium-sized I/O operations into multiple smaller I/O operations that execute in parallel. While the number of files and disks increase, you only have to manage a constant number of disk groups. From a database perspective, disk groups can be specified as the default location for files created in the database.

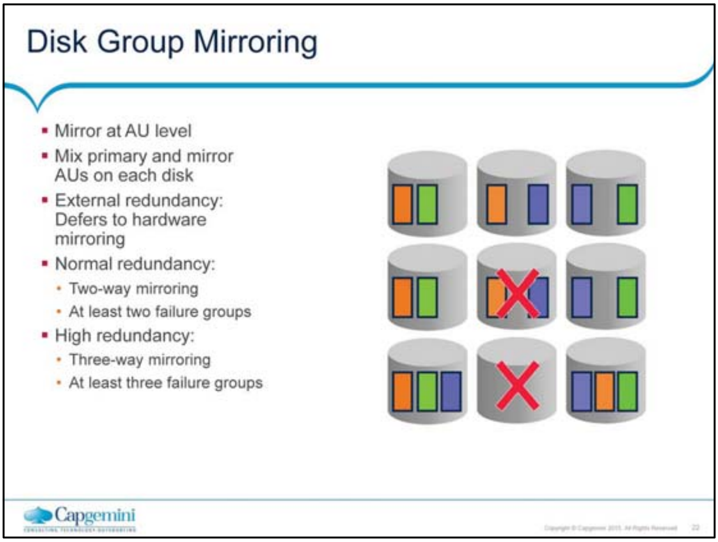
Note: Each disk group is self-describing, containing its own file directory and disk directory.



Failure Group

A failure group is a set of disks, inside one particular disk group, sharing a common resource whose failure needs to be tolerated. An example of a failure group is a string of SCSI disks connected to a common SCSI controller. A failure of the controller leads to all the disks on its SCSI bus becoming unavailable, although each of the individual disks is still functional.

What constitutes a failure group is site-specific. It is largely based upon failure modes that a site is willing to tolerate. By default, ASM assigns each disk to its own failure group. When creating a disk group or adding a disk to a disk group, administrators may specify their own grouping of disks into failure groups. After failure groups are identified, ASM can optimize file layout to reduce the unavailability of data due to the failure of a shared resource.



Disk Group Mirroring

ASM has three disk group types that support different types of mirroring:

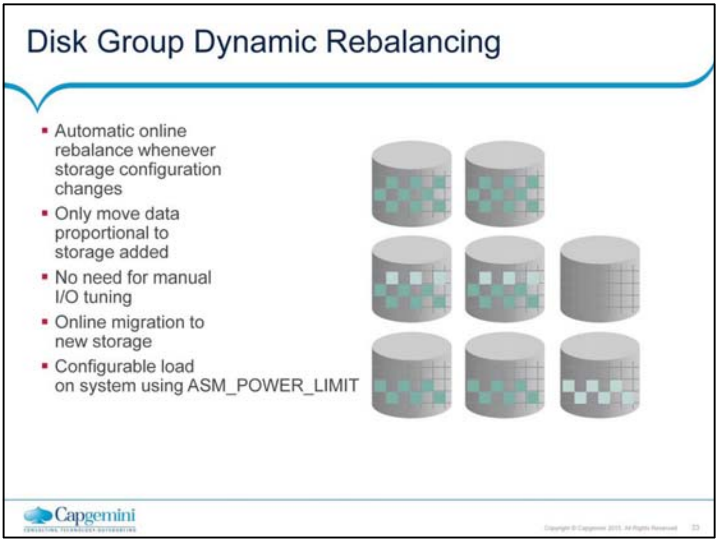
External redundancy: Do not provide mirroring. Use an external-redundancy disk group if you use hardware mirroring or if you can tolerate data loss as the result of a disk failure. Failure groups are not used with these types of disk groups.

Normal-redundancy: Support two-way mirroring

High-redundancy: Provide triple mirroring

ASM does not mirror disks; rather, it mirrors allocation units. As a result, you need only spare capacity in your disk group. When a disk fails, ASM automatically reconstructs the contents of the failed disk on the surviving disks in the disk group by reading the mirrored contents from the surviving disks. This spreads the I/O hit from a disk failure across several disks.

When ASM allocates a primary AU of a file to one disk in a disk group, it allocates a mirror copy of that AU to another disk in the disk group. Primary AUs on a given disk can have their mirror copies on one of several partner disks in the disk group. ASM ensures that a primary AU and its mirror copy never reside in the same failure group. If you define failure groups for your disk group, ASM can tolerate the simultaneous failure of multiple disks in a single failure group.



Disk Group Dynamic Rebalancing

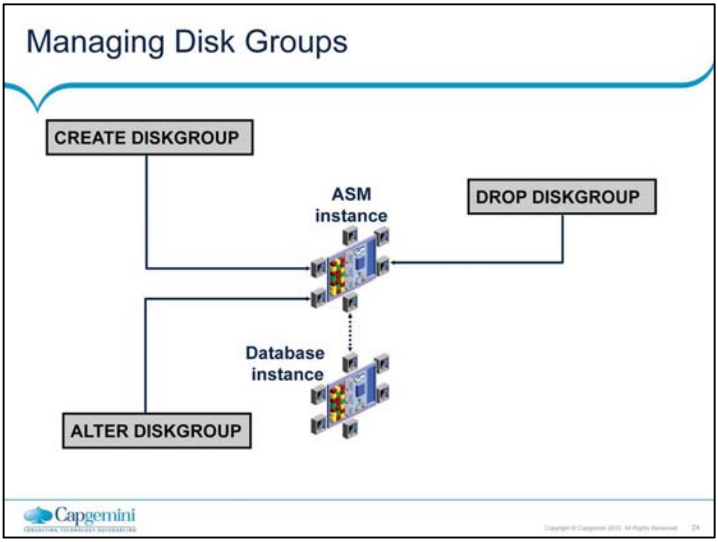
With ASM, the rebalance process is very easy and happens without any intervention from the DBA or system administrator. ASM automatically rebalances a disk group whenever disks are added or dropped.

By using index techniques to spread AUs on the available disks, ASM does not need to restripe all of the data, but instead needs to only move an amount of data proportional to the amount of storage added or removed to evenly redistribute the files and maintain a balanced I/O load across the disks in a disk group.

With the I/O balanced whenever files are allocated and whenever the storage configuration changes, the DBA never needs to search for hot spots in a disk group and manually move data to restore a balanced I/O load.

It is more efficient to add or drop multiple disks at the same time so that they are rebalanced as a single operation. This avoids unnecessary movement of data. With this technique, it is easy to achieve online migration of your data. All you need to do is add the new disks in one operation and drop the old ones in one operation.

You can control how much of a load the rebalance operation has on the system by setting the ASM_POWER_LIMIT initialization variable. Its range of values is 0 through 11. The lower the number, the lighter the load, whereas a higher setting has more of a load, and finishes sooner. A setting of 0 places rebalance operations on hold. The default value is 1.



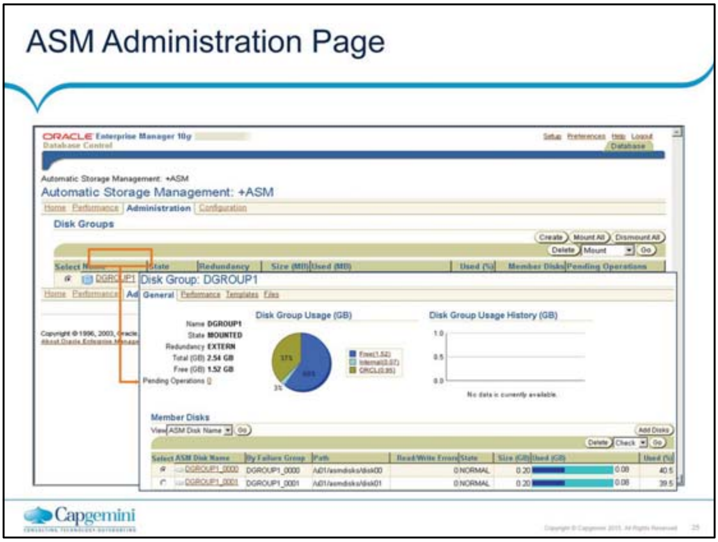
Managing Disk Groups

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 new disk group administrative commands. They all require the SYSDBA privilege and must be issued from an ASM instance.

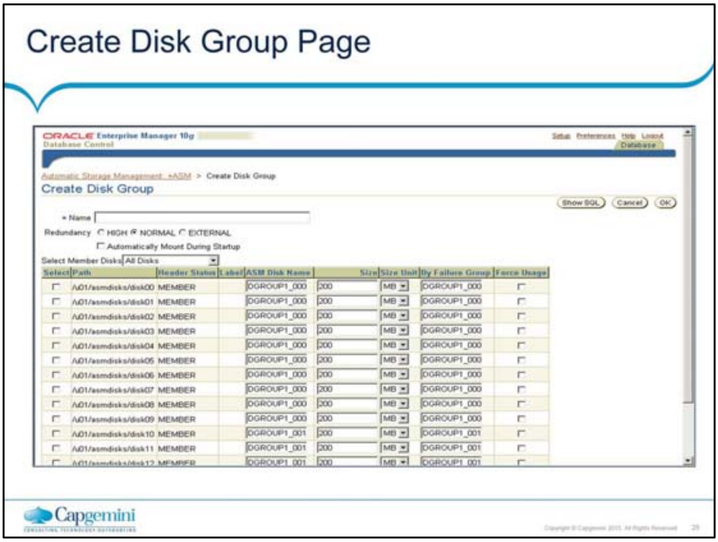
You can add new disk groups. You can also modify existing disk groups to add new disks, remove existing ones, and perform many other operations.

You can remove existing disk groups.



ASM Administration Page

When you click the Administration tabbed page of the Automatic Storage Management page, you can see the disk groups listed in the V$ASM_DISKGROUP view. From here, you can create, edit, or drop a disk group. You can also perform disk group operations such as mount, dismount, rebalance, check, and repair on a selected disk group.

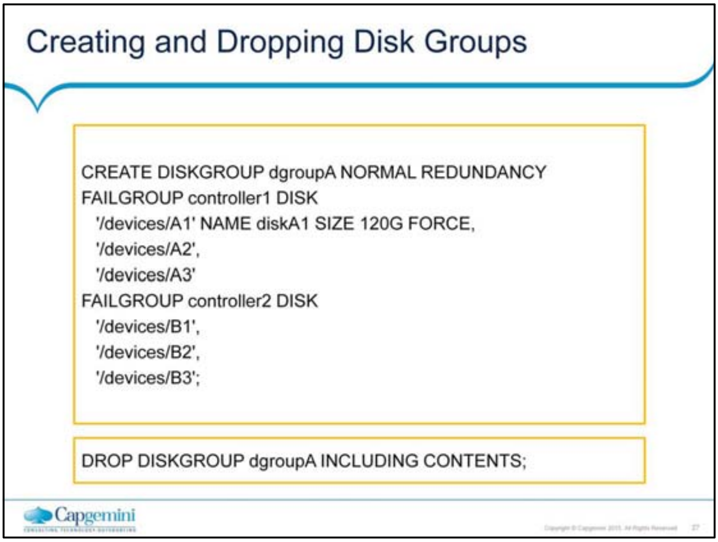


Create Disk Group Page

Clicking Create on the Disk Group Overview page displays the Create Disk Group page. You can enter the disk group name, redundancy mechanism, and the list of disks that you would like to include in the new disk group. The list of disks is obtained from the V$ASM_DISK fixed view. By default, only the disks that can be assigned to a disk group show up. Those are the ones with a status of one of the following:

- CANDIDATE: The disk has never been assigned to an ASM disk group.
- FORMER: The disk was once assigned to an ASM disk group, but is not now.
- PROVISIONED: ASMLib is being used, and this disk is not yet assigned to a disk group.

Note: ASMLib is an API that interfaces with other vendors' storage arrays. See the Database Administrator's Guide documentation for more information about ASMLib.

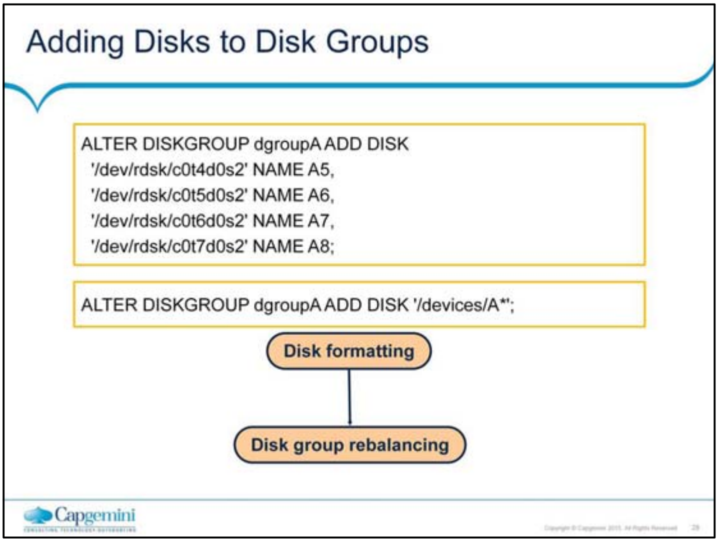


Creating and Dropping Disk Groups

Assume that ASM disk discovery identified the following disks in the /devices directory: A1, A2, A3, B1, B2, and B3. Also, assume that disks A1, A2, and A3 are on a separate SCSI controller from disks B1, B2, and B3. 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 characteristic, 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

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.

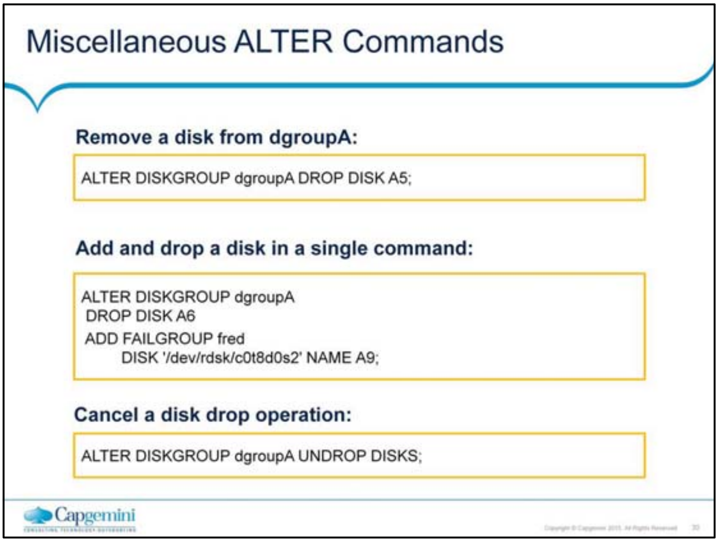
The second command adds A4 to the DGROUPA disk group. It ignores the other disks, even though they match the discovery string, because they are already part of the DGROUPA disk group. As shown by the diagram, when you add a disk to a disk group, the ASM instance ensures that the disk is addressable and usable. The disk is then formatted and rebalanced. The rebalance process is time consuming because it moves AUs from every file onto the new disk.

Adding Disks to Disk Groups (notes only slide)

Capgemini

Adding Disks to Disk Groups (continued)

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

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:

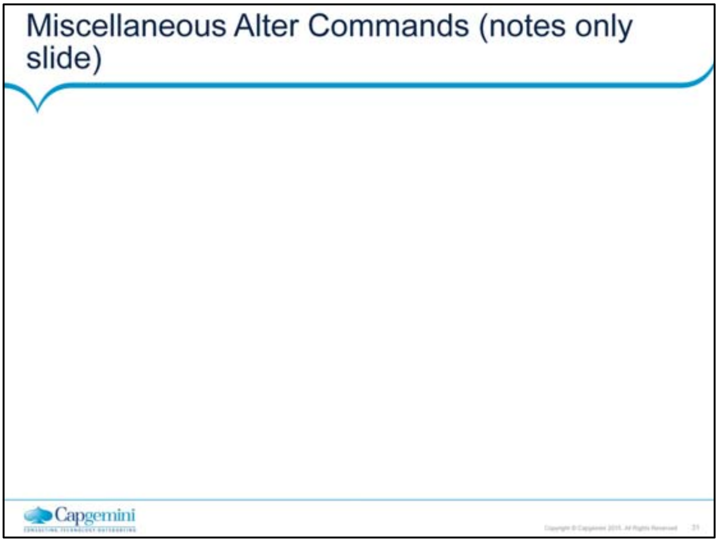
```
ALTER DISKGROUP dgroupB REBALANCE
POWER 5;
```

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.



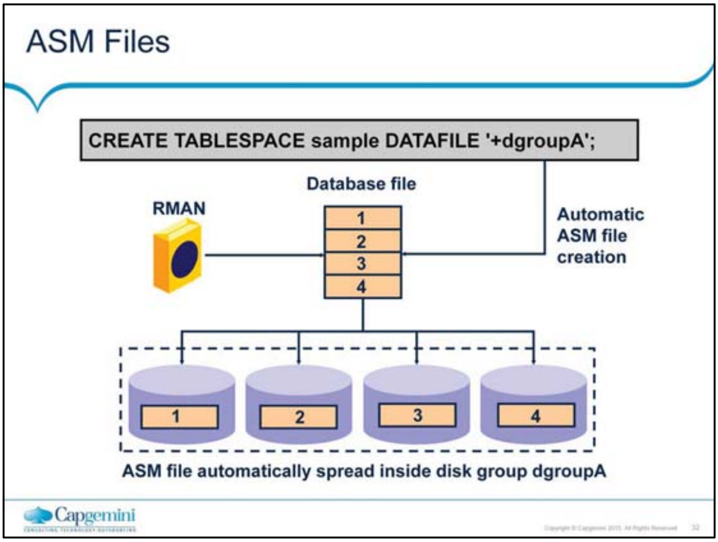
Miscellaneous ALTER Commands (continued)

Use the following statement to verify the internal consistency of disk group metadata and to repair any error found:

ALTER DISKGROUP dgroupA CHECK ALL;

It is also possible to use the NOREPAIR clause if you just want to be alerted about errors. While the example requests a check across all disks in the disk group, checking can be specified on a file or an individual disk. This command requires that the disk group be mounted. If any error is found, a summary error message is displayed and the details of the detected error are reported in the alert log.

Note: Of these six examples, the first four trigger a disk group rebalancing, and the last two do not.



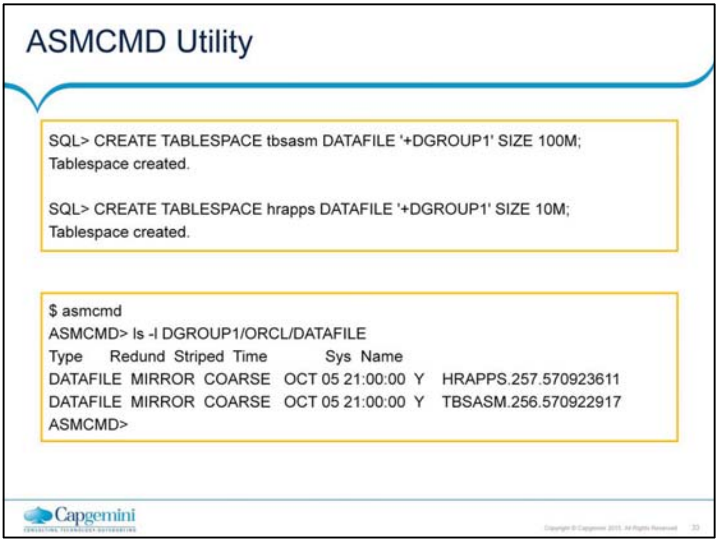
ASM Files

When you specify an ASM disk group as the data file name for a tablespace, ASM files are created in the disk group to provide storage for the tablespace.

When an ASM file is created, certain file attributes are permanently set. Among these are its protection policy, and its striping policy. ASM files are Oracle Managed Files. Any file that is created by ASM is automatically deleted when it is no longer needed.

With ASM, file operations are specified in terms of database objects. Administration of databases never requires knowing the name of a file, though the name of the file is exposed through some data dictionary views, or the ALTER DATABASE BACKUP CONTROLFILE TO TRACE command. Because each file in a disk group is physically spread across all disks in the disk group, a backup of a single disk is not useful. Database backups of ASM files must be made with RMAN.

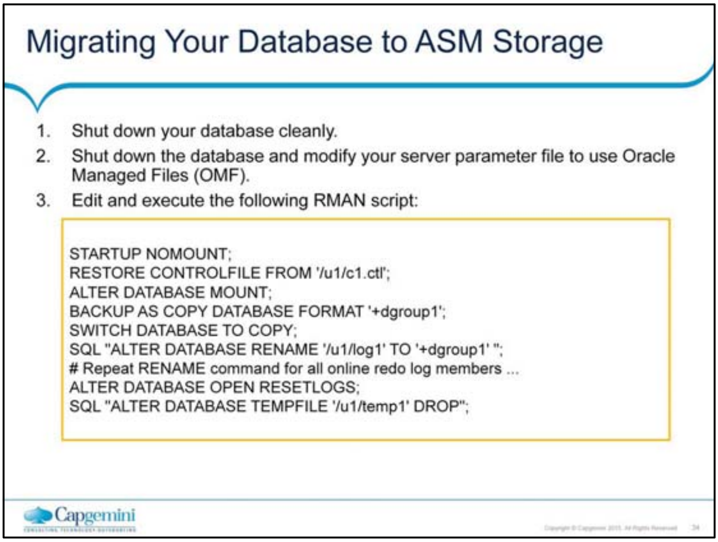
Note: ASM does not manage binaries, alert logs, trace files, or password files.



ASMCMD Utility

ASMCMD is a command-line utility that you can use to easily view and manipulate files and directories within ASM disk groups. It can list the contents of disk groups, perform searches, create and remove directories, and display space utilization, among other things.

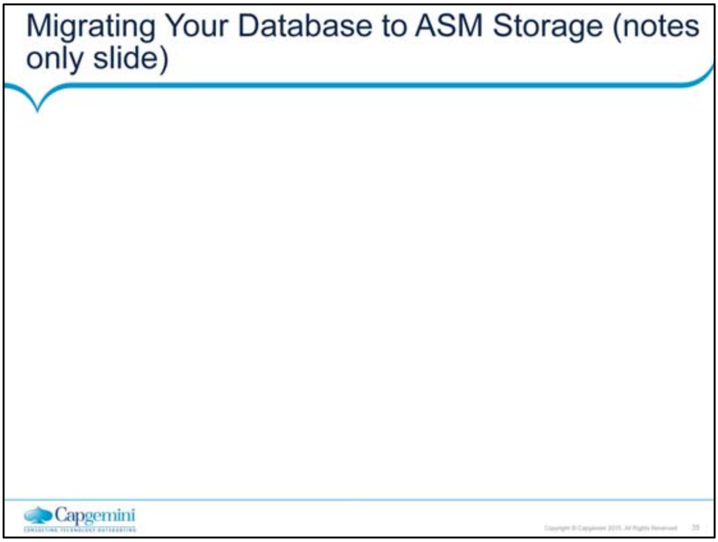
Note: For more information about ASMCMD, see the Oracle Database Utilities documentation.



Migrating Your Database to ASM Storage

Because ASM files cannot be accessed through normal operating system interfaces, RMAN is the only means for copying ASM files. Although files in a tablespace may be both ASM files and non-ASM files as a result of the tablespace history, RMAN commands enable non-ASM files to be relocated to an ASM disk group. You can use the following procedure to relocate your entire database to an ASM disk group: (It is assumed that you are using a server parameter file.)

1. Obtain the file names of the current control files and online redo logs by using V$CONTROLFILE and V$LOGFILE.
2. Shut down the database consistently. Modify the server parameter file of your database as follows:
   - Set the necessary OMF destination parameters to the desired ASM disk group.
   - Remove the CONTROL_FILES parameter.

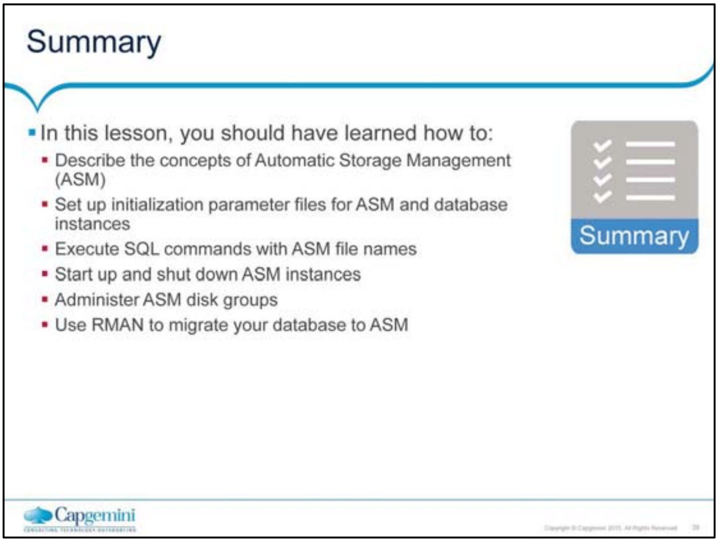


Migrating Your Database to ASM Storage (continued)

3. Edit and run the RMAN command file, which backs up the database, switches the current data files to the backups, and renames the online redo logs. You can move only tablespaces or data files by using the BACKUP AS COPY command.
4. Delete the old database files.

Note: If you create an OMF control file, and if there is a server parameter file, then a CONTROL_FILES initialization parameter entry is created in the server parameter file.

See the Oracle Database Backup and Recovery Advanced User’s Guide for details about how to migrate a database to ASM.


Summary
In this lesson, you should have learned how to:
Describe the concepts of Automatic Storage Management (ASM)
Set up initialization parameter files for ASM and database instances
Execute SQL commands with ASM file names
Start up and shut down ASM instances
Administer ASM disk groups
Use RMAN to migrate your database to ASM
Summary
Capgemini

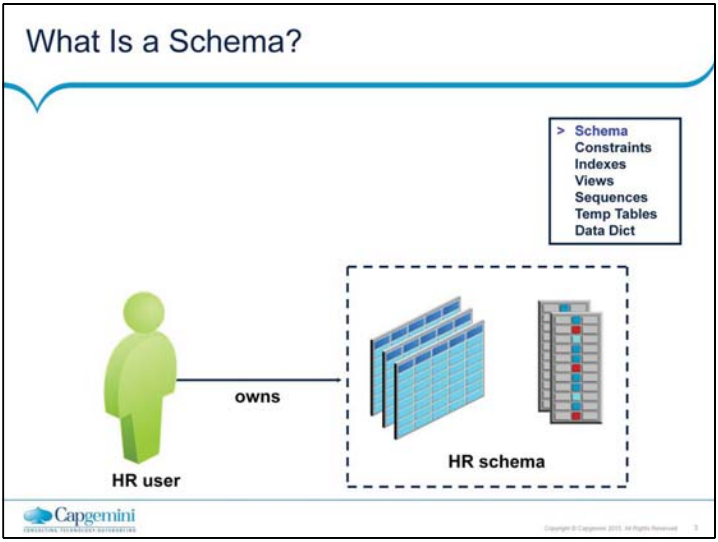
Oracle 11g DBA Fundamentals Overview

Lesson 09: Managing Schema Objects

Objectives

- After completing this lesson, you should be able to do the following:
  - Define schema objects and data types
  - Create and modify tables
  - Define constraints
  - View the columns and contents of a table
  - Create indexes
  - Create views
  - Create sequences
  - Explain the use of temporary tables
  - Use the data dictionary

Capgemini



What Is a Schema?

A schema is a collection of database objects that are owned by a particular user. Typically, for a production database, this user does not represent a person, but an application. A schema has the same name as the user that owns the schema. Schema objects are the logical structures that directly refer to database's data. Schema objects include structures such as tables, views, and indexes.

You can create and manipulate schema objects by using SQL or Enterprise Manager. When you use Enterprise Manager, the underlying SQL is generated for you.

Note: A schema does not necessarily have to be directly related to a single tablespace. You can define configurations such that objects in a single schema can be in different tablespaces, and a tablespace can hold objects from different schemas.

What Is a Schema? (continued)

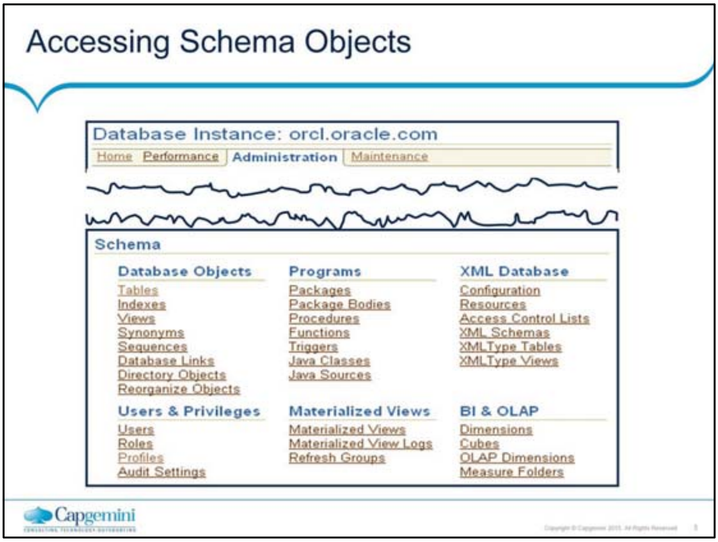
When you create the database, several schemas are created for you. Two of particular importance are the following:

- **SYS schema:** This contains the data dictionary, as described in the lesson titled "Administering User Security."

- **SYSTEM schema:** It contains additional tables and views that store administrative information. This is described in the lesson titled "Administering User Security."

During a complete installation of an Oracle database, sample schemas are installed automatically. Sample schemas serve the purpose of providing a common platform for examples in Oracle documentation and curricula. They are a set of interlinked schemas aimed at providing examples of different levels of complexity and include the following:

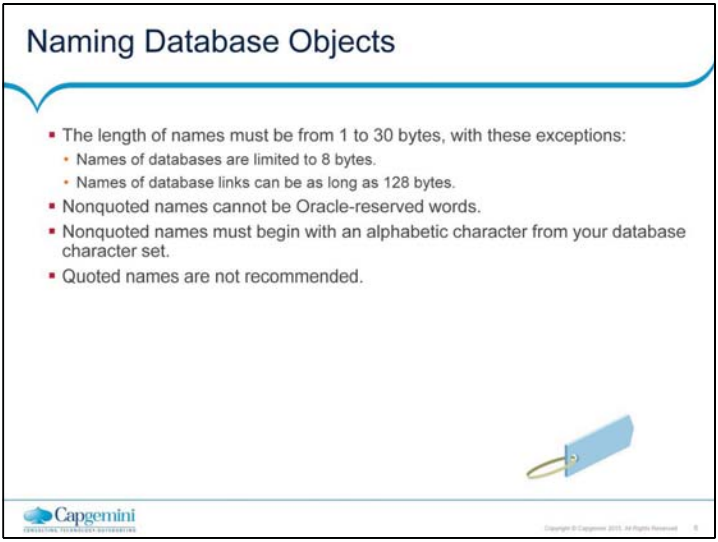
- **HR:** The Human Resources schema is a simple schema for introducing basic topics. An extension to this schema supports Oracle Internet Directory demonstrations.
- **OE:** The Order Entry schema deals with matters of intermediate complexity. A multitude of data types are available in the OE schema. The OC (Online Catalog) subschema is a collection of object-relational database objects built inside the OE schema.
- **PM:** The Product Media schema is dedicated to multimedia data types.
- **QS:** The Queued Shipping schema contains a set of schemas that are used to demonstrate Oracle Advanced Queuing capabilities.
- **SH:** The Sales History schema allows demonstrations with larger amounts of data. An extension to this schema provides support for advanced analytic processing.



Accessing Schema Objects

You can quickly access many types of schema objects from the Schema region of the Database Administration page.

After clicking one of the links, the Results page is displayed. In the Search region of the page, you can enter a schema name and object name to search for a specific object. In addition, you can search for other types of objects from the Search region by selecting the object type from the drop-down list. The drop-down list includes additional object types that are not shown as links on the Database Administration page.



Naming Database Objects

When you name an object in the database, you can enclose the name in double quotation marks (“”). If you do this, you can break several of the naming rules mentioned in the slide. However, this is not recommended because if you name an object this way, you must always refer to it with the quotation marks around the name. For example, if you name a table “Local Temp,” you must do the following:

```
SQL>  select * from "Local Temp";
TEMP_DATE    LO_TEMP    HI_TEMP
--------- ---------- ----------
01-DEC-03        30         41
```

If you enter the name in the wrong case, then you get an error:

```
SQL> select * from "local temp";
select * from "local temp"
              *
ERROR at line 1:
ORA-00942: table or view does not exist
```

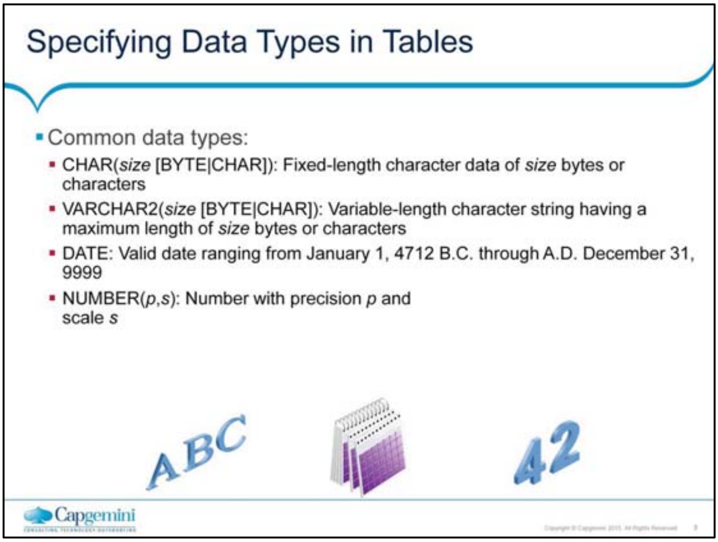
Nonquoted names are stored in uppercase and are not case sensitive. When a SQL statement is processed, nonquoted names are converted to all uppercase.

Naming Database Objects Full Notes Page

Naming Database Objects (continued)

Nonquoted identifiers can contain only alphanumeric characters from your database character set and the underscore (_), the dollar sign ($), and the pound sign (#). Database links can also contain periods (.) and the "at" sign (@). You are strongly discouraged from using $ and # in nonquoted identifiers.

Quoted identifiers can contain any characters and punctuation marks as well as spaces. However, neither quoted nor nonquoted identifiers can contain double quotation marks.



Specifying Data Types in Tables

When you create a table, you must specify a data type for each of its columns. When you create a procedure or function, you must specify a data type for each of its arguments. These data types define the domain of values that each column can contain or each argument can have.

Built-in data types in the Oracle database include the following:

- CHAR: Fixed-length character data of size bytes or characters. The maximum size is 2,000 bytes or characters; the default and minimum size is 1 byte.
  - BYTE indicates that the column has byte length semantics.
  - CHAR indicates that the column has character semantics.
- VARCHAR2: Variable-length character string having maximum length size bytes or characters. The maximum size is 4,000 bytes. You must specify the size for VARCHAR2.
- DATE: Valid date ranging from January 1, 4712 B.C. through A.D. December 31, 9999. It also stores the time: hours, minutes, and seconds.
- NUMBER: Number with precision p and scale s. The precision can range from 1 through 38. The scale can range from –84 through 127.

Specifying Data Types in Tables (continued)

BINARY_FLOAT: This is a 32-bit floating-point number. This data type requires 5 bytes, including the length byte.
BINARY_DOUBLE: This is a 64-bit floating-point number. This data type requires 9 bytes.
FLOAT(*p*): This is an American National Standards Institute (ANSI) data type. The FLOAT data type is a floating-point number with a binary precision *p*. The default precision for this data type is 126 binary or 38 decimal.
INTEGER: This is equivalent to NUMBER(p,0).
NCHAR(*length*): The NCHAR data type is a Unicode-only data type. When you create a table with an NCHAR column, you define the column length in characters. You define the national character set when you create your database. The maximum length of a column is determined by the national character set definition. The width specifications of the NCHAR data type refer to the number of characters. The maximum column size allowed is 2,000 bytes. If you insert a value that is less than the column length, the Oracle database pads the value with blanks for full column length. You cannot insert a CHAR value into an NCHAR column, nor can you insert an NCHAR value into a CHAR column.

NVARCHAR2(size [BYTE|CHAR]): The NVARCHAR2 data type is a Unicode-only data type. It is like NCHAR except that its maximum length is 4,000 bytes and it is not blank-padded.
LONG: This is a character data of variable length of up to 2 gigabytes or 2^{31} –1 bytes. The LONG data type is deprecated; use the large object (LOB) data type instead.
LONG RAW: This is raw binary data of variable length of up to 2 gigabytes.
RAW(*size*): This is raw binary data of length *size* bytes. The maximum size is 2,000 bytes. You must specify the size for a RAW value.
ROWID: This is a base 64 string representing the unique address of a row in its table. This data type is primarily for values returned by the ROWID pseudocolumn.
UROWID: This is a base 64 string representing the logical address of a row of an index-organized table. The optional size is the size of a column of the UROWID type. The maximum size and default is 4,000 bytes.
BLOB: This is a binary large object.
CLOB: This is a character large object containing single-byte or multibyte characters. Both fixed-width and variable-width character sets are supported, and both use the CHAR database character set.

Specifying Data Types in Tables (continued)

NCLOB: This is a character large object containing Unicode characters. Both fixed-width and variable-width character sets are supported, and both use the NCHAR database character set. It stores national character set data.

Note: The maximum size for all LOB data types (BLOB, CLOB, and NCLOB) is:

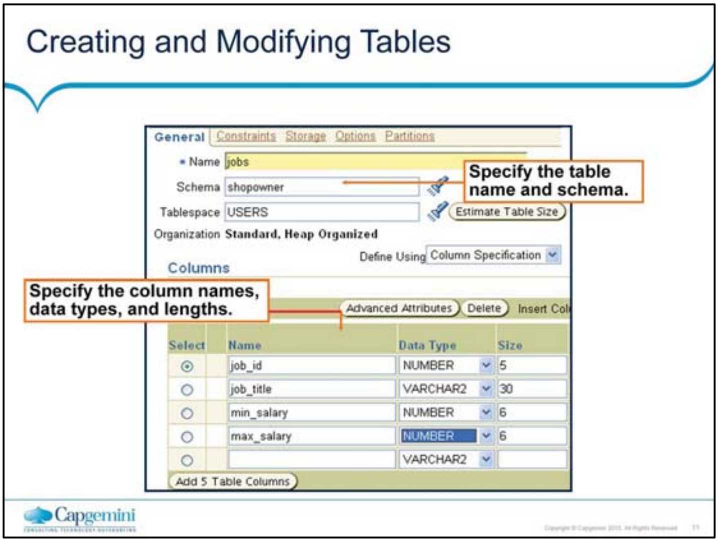
(4 gigabytes – 1) * (the value of CHUNK).

CHUNK is an optional attribute that you can set when defining a LOB. CHUNK specifies the number of bytes to be allocated for LOB manipulation. If the size is not a multiple of the database block size, then the database rounds up the size in bytes to the next multiple. For example, if the database block size is 2,048 and the CHUNK size is 2,050, then the database allocates 4,096 bytes (2 blocks). The maximum value is 32,768 (32 KB), which is the largest Oracle database block size allowed. The default CHUNK size is one Oracle database block.

BFILE: The BFILE data type contains a locator to a large binary file stored outside the database. It enables byte stream I/O access to external LOBs residing on the database server. The maximum size is 4 gigabytes.

TIMESTAMP(*fractional_seconds_precision*): With this data type, you can specify the year, month, and day values of date, as well as hour, minute, and second values of time, where *fractional_seconds_precision* is the number of digits in the fractional part of a second. The accepted values are 0 to 9. The default is 6.

For a complete list of built-in data types and user-defined types, refer to *Oracle Database SQL Reference*.



Creating and Modifying Tables

Tables are the basic units of data storage in an Oracle database. They hold all user-accessible data. Each table has columns and rows.

Creating a Table

To create a table by using Enterprise Manager, perform the following steps:

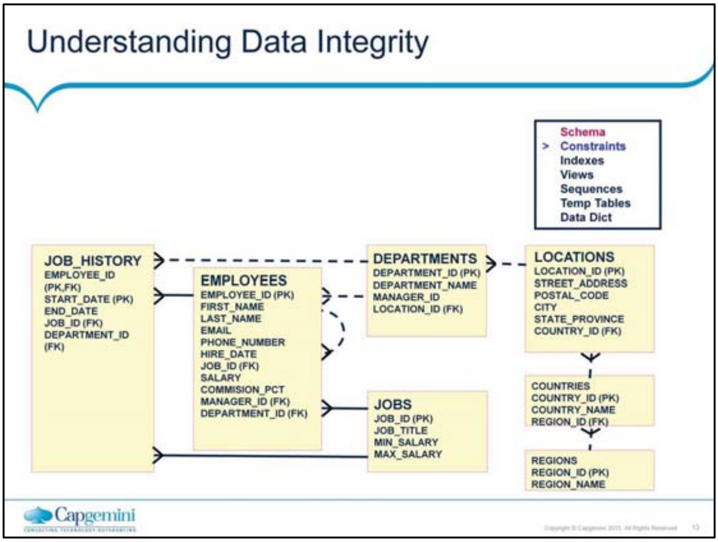
1. Click Tables in the Schema region of the Administration page. The Tables page appears.
2. If you know the schema name, enter all or part of it in the Schema field in the Search region. If you do not know the schema name, click the flashlight icon next to the Schema field. The Search and Select: Schema window appears. You can browse through the schema names and select the one that you are looking for.
3. Click Create. The Create Table: Table Organization page appears.
4. Accept the default of Standard, Heap Organized by clicking Continue. The Create Table page appears.
5. Enter the table name in the Name field.
6. Enter the schema name in the Schema field, or click the flashlight icon to invoke the search function.

Creating and Modifying Tables Full Notes Page

Creating and Modifying Tables (continued)

7. Enter the tablespace name in the Tablespace field, or click the flashlight icon to invoke the search function.
8. In the Columns region, enter the column name and data types.
9. Click OK. An update message appears indicating that the table has been successfully created.

Modifying a Table
You can modify a table by using Enterprise Manager as described in the following steps. In this example, a column is added to a table.
On the Tables page, select the table in the results list and click Edit.
On the Edit Table page, click the Add 5 Table Columns button. An editable columns list appears.
Enter the new column name, data type, and size.
Click Apply. An update message appears indicating that the table has been modified successfully.



Understanding Data Integrity

You can use the following integrity constraints to impose restrictions on the input of column values:

NOT NULL: By default, all columns in a table allow null values. Null means the absence of a value. A NOT NULL constraint requires that a column of a table must contain no null values. For example, you can define a NOT NULL constraint to require that a value be input in the LAST_NAME column for every row of the EMPLOYEES table.

UNIQUE Key: A UNIQUE key integrity constraint requires that every value in a column or set of columns (key) be unique—that is, no two rows of a table have duplicate values in a specified column or set of columns. For example, a UNIQUE key constraint is defined on the DEPARTMENT_NAME column of the DEPARTMENTS table to disallow rows with duplicate department names. Except for special circumstances, this is enforced with a unique index.

PRIMARY KEY: Each table in the database can have at most one PRIMARY KEY constraint. The values in the group of one or more columns subject to this constraint constitute the unique identifier of the row. In effect, each row is named by its primary key values.

Understanding Data Integrity (continued)

The Oracle server's implementation of the PRIMARY KEY integrity constraint guarantees that both the following are true:
No two rows of a table have duplicate values in the specified column or set of columns.
The primary key columns do not allow nulls. That is, a value must exist for the primary key columns in each row.

Under normal circumstances, the database enforces the PRIMARY KEY constraints by using indexes. The primary key constraint created for the DEPARTMENT_ID column in the DEPARTMENTS table is enforced by the implicit creation of:

A unique index on that column
A NOT NULL constraint for that column

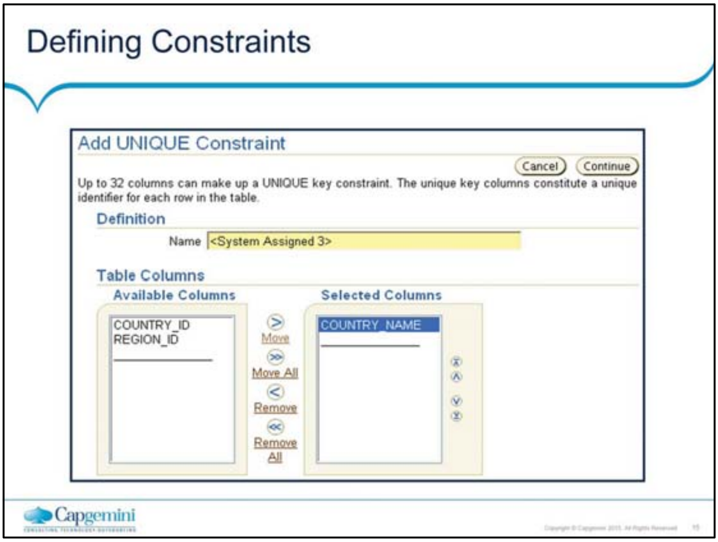
Referential integrity constraints: Different tables in a relational database can be related by common columns, and the rules that govern the relationship of the columns must be maintained. Referential integrity rules guarantee that these relationships are preserved.

A referential integrity constraint requires that for each row of a table, the value in the foreign key must match a value in a parent key.

As an example, a foreign key is defined on the DEPARTMENT_ID column of the EMPLOYEES table. It guarantees that every value in this column must match a value in the primary key of the DEPARTMENTS table. Therefore, no erroneous department numbers can exist in the DEPARTMENT_ID column of the DEPARTMENTS table.

Another type of referential integrity constraint is called a self-referential integrity constraint. This type of foreign key references a parent key in the same table

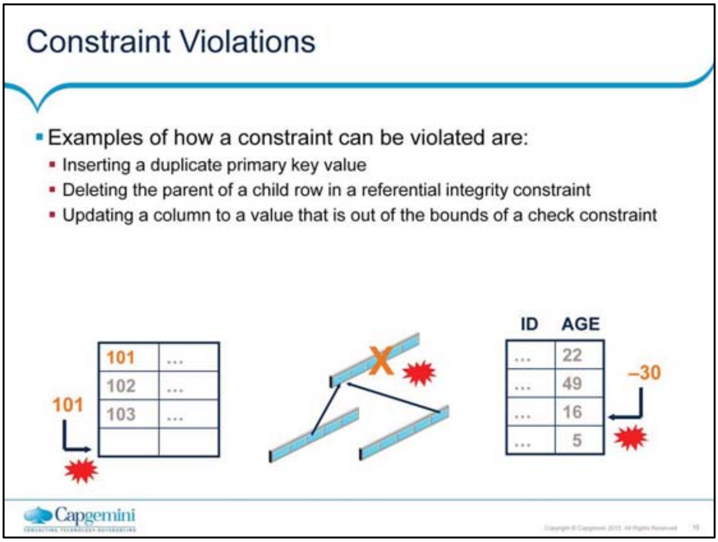
Check constraints: A CHECK integrity constraint on a column or set of columns requires that a specified condition be true or unknown for every row of the table. If a data manipulation language (DML) statement results in the condition of the CHECK constraint evaluating to false, then the statement is rolled back.



Defining Constraints

To add a constraint to a table by using Enterprise Manager, perform the following steps:

1. Select the table on the Tables page, and click Edit.
2. Click Constraints. The Constraints page is displayed showing all constraints that have been defined on the table.
3. Select the type of constraint that you want to add from the drop-down list, and click Add.
4. Enter the appropriate information for the type of constraint that you are defining. Click OK.



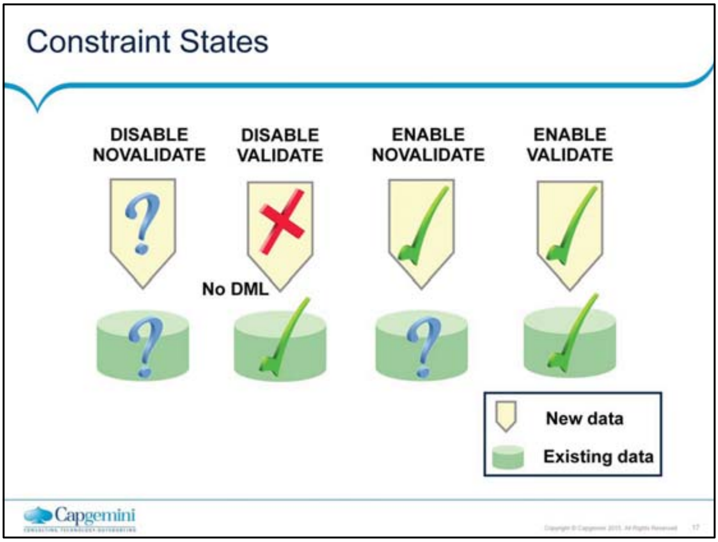
Constraint Violations

A constraint violation occurs when DML is submitted, which does not comply with the constraint. Constraint violations can come in many forms. Among these are the following:

Uniqueness: An attempt is made to have duplicate values in a column that has a unique constraint, such as in the case where a column is the primary key, or it is uniquely indexed.

Referential integrity: The rule of every child row having a parent row is violated.

Check: An attempt is made to store a value in a column that does not follow the rules defined for that column. For example, an AGE column could have a check constraint on it enforcing it to be a positive number.



Constraint States

To better deal with situations where data must be temporarily in violation of a constraint, you can designate a constraint to be in various states. An integrity constraint can be enabled (ENABLE) or disabled (DISABLE). If a constraint is enabled, the data is checked as it is entered or updated in the database. Data that does not conform to the constraint’s rule is prevented from being entered. If a constraint is disabled, then the nonconforming data can be entered into the database. An integrity constraint can be in one of the following states:

- DISABLE NOVALIDATE
- DISABLE VALIDATE
- ENABLE NOVALIDATE
- ENABLE VALIDATE

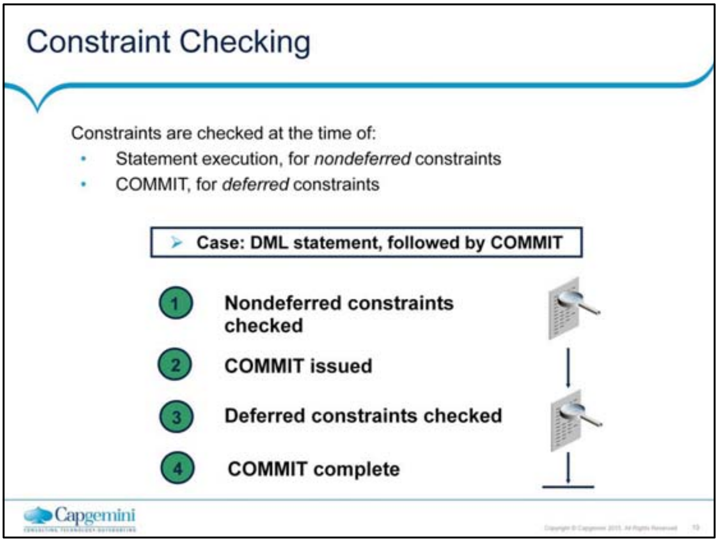
Constraint States (continued)

DISABLE NOVALIDATE: New as well as existing data may not conform to the constraint because it is not checked. This is often used when the data is from an already validated source and the table is read-only, so no new data is being entered into the table.

DISABLE VALIDATE: If a constraint is in this state, then any modification of the constrained columns is not allowed because it would be inconsistent to have validated the existing data and then allow unchecked data to enter the table. This is often used when the existing data must be validated but the data is not going to be modified and the index is not otherwise needed for performance.

ENABLE NOVALIDATE: New data conforms to the constraint but existing data is in an unknown state. This is frequently used so that existing constraint violations can be corrected, and at the same time, new violations are not allowed to enter the system.

ENABLE VALIDATE: Both new and existing data conform to the constraint. This is the typical and default state of a constraint.



Constraint Checking

You can defer checking constraints for validity until the end of the transaction.

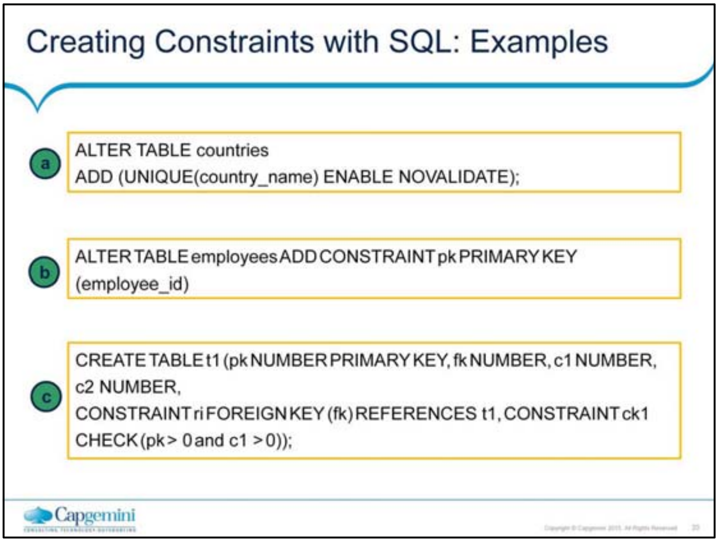
Nondeferred constraints, also known as immediate constraints, are enforced at the end of every DML statement. A constraint violation causes the statement to be rolled back. If a constraint causes an action such as delete cascade, the action is taken as part of the statement that caused it. A constraint that is defined as nondeferrable cannot be changed to a deferrable constraint.

Deferred constraints are constraints that are checked only when a transaction is committed. If any constraint violations are detected at commit time, then the entire transaction is rolled back. These constraints are most useful when both the parent and child rows in a foreign key relationship are entered at the same time, as in the case of an order entry system, where the order and the items in the order are entered at the same time.

A constraint that is defined as deferrable can be specified as one of the following:

Initially immediate specifies that by default it must function as an immediate constraint unless explicitly set otherwise.

Initially deferred specifies that by default the constraint must be enforced only at the end of the transaction.



Creating Constraints with SQL: Examples

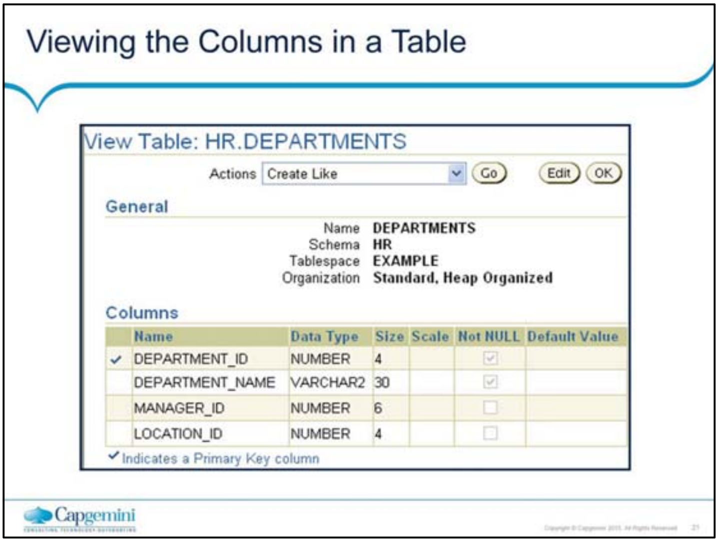
Three examples of constraint creation are shown in the slide:

a. After this statement executes, any inserts or updates done on the COUNTRIES table are required to have a COUNTRY_NAME value that is unique. But it is possible that when this statement is issued, there already exist COUNTRY_NAME values in the table that are not unique. The NOVALIDATE keyword indicates that they should be ignored. Only new rows are constrained.

b. This statement adds a primary key to the employee table. The constraint name is PK and the primary key is the EMPLOYEE_ID column.

c. This statement defines constraints at the time the table is created, rather than using an ALTER TABLE statement later. The RI constraint enforces that the values in the FK column must be present in the primary key column of the T1 table. The CK1 constraint enforces that the PK and C1 columns are greater than zero.

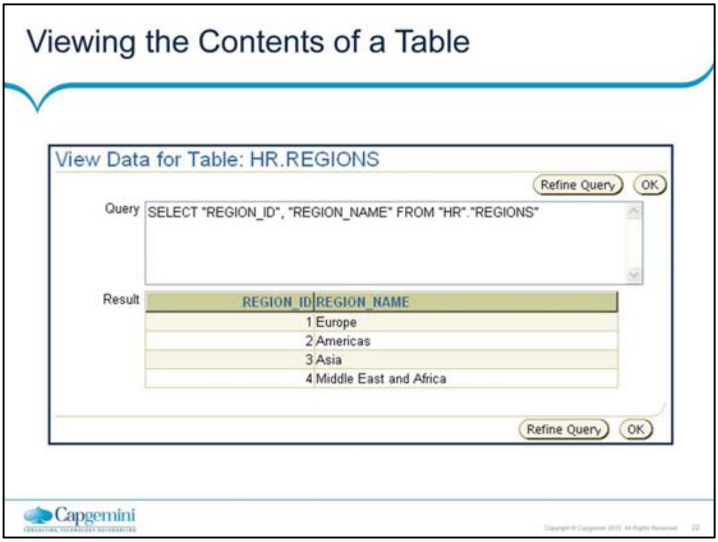
Note: Each constraint has a name. If one is not supplied in the DDL statement, then a system-supplied name is assigned, which starts with SYS_.



Viewing the Columns in a Table

To view the attributes of a table by using Enterprise Manager, perform the following steps:

1. Click the Tables link in the Schema region of the Database Administration page.
2. Select a table from the Results list and click the View button to see the attributes of the table.



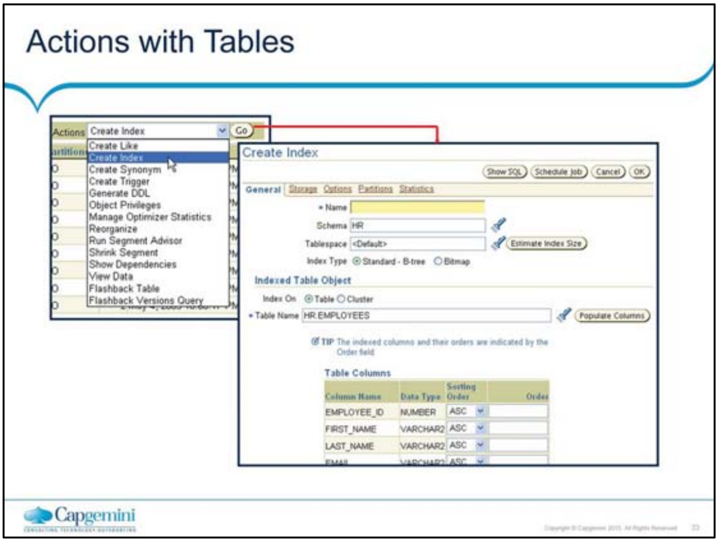
Viewing the Contents of a Table

To view the rows in a table by using Enterprise Manager, perform the following steps:

1. Select the table on the Tables page.
2. Select View Data from the Actions menu and click Go.

The View Data for Table page appears. The row data for the table is shown in the Result region. The Query box displays the SQL query that is executed to produce the results. On this page, you can click any column name and sort the data in the column in either ascending or descending order. If you want to change the query, click the Refine Query button. On the Refine Query for Table page, you can select the columns that you want to display and specify a WHERE clause for the SQL statement to limit the results.

For more information about the WHERE clauses in SQL statements, refer to Oracle Database SQL Reference.



Actions with Tables

You can select a table and then perform actions on that table. Here are some of those actions:

Create Like: With this action, you can create a table that has the same structure as the selected table. You must change the constraint names. You can add or delete columns and make other changes to the table structure before it is created.

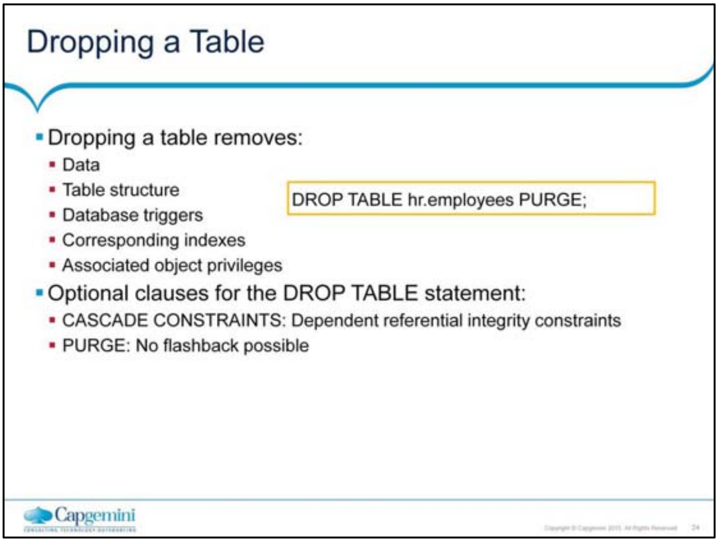
Create Index: Use this option to create indexes on a table.

Generate DDL: This generates the DDL that represents the table as it already exists. This can then be copied to a text file for use as a script or for documentation purposes.

Grant Privileges: By default, when a table is created, only the owner can do anything with it. The owner must grant privileges to other users in order for them to perform DML or possibly DDL on the table.

Show Dependencies: This shows objects that this table depends on or objects that depend on this table.

View Data: This selects and displays data from the table in a read-only manner.



Dropping a Table

Syntax:

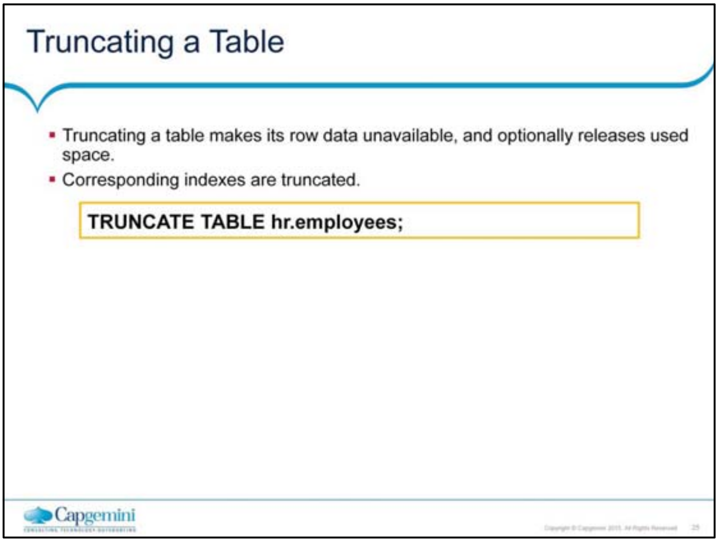
DROP TABLE [schema.] table [CASCADE CONSTRAINTS] [PURGE]

The DROP TABLE command removes data, the table structure, and associated object privileges. Some DROP TABLE considerations are as follows:

Without the PURGE clause, the table definition, associated indexes, and triggers are placed in a recycle bin. The table data still exists, but is inaccessible without the table definition. If you drop a table through Enterprise Manager, the PURGE clause is not used. Use the FLASHBACK TABLE command to recover schema objects from the recycle bin. The PURGE RECYCLEBIN command empties the recycle bin.

The CASCADE CONSTRAINTS option is required to remove all dependent referential integrity constraints.

Note: If you do not use the PURGE option, the space taken up by the table and its indexes still counts against the user’s allowed quota for the tablespaces involved. That is, the space is still considered as being used.



Truncating a Table

Syntax:

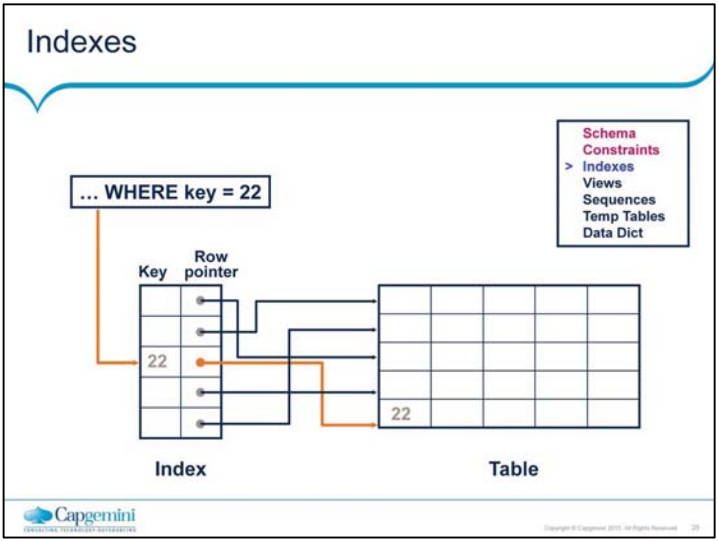
TRUNCATE TABLE [schema.] table [{DROP | REUSE} STORAGE]

The effects of using this command are as follows:

- The table is marked as empty by setting the high-water mark (HWM) to the beginning of the table, making its rows unavailable.
- No undo data is generated and the command commits implicitly because TRUNCATE TABLE is a DDL command.
- Corresponding indexes are also truncated.
- A table that is being referenced by a foreign key cannot be truncated.
- The delete triggers do not fire when this command is used.

This is usually many times faster than issuing a DELETE statement to delete all the rows of the table due to the following reasons:

- The Oracle database resets the table's HWM instead of processing each row as a DELETE operation.
- No undo data is generated.



Indexes

Indexes are optional structures associated with tables. They can be created to improve the performance of data update and retrieval. An Oracle index provides a direct access path to a row of data.

Indexes can be created on one or more columns of a table. After an index is created, it is automatically maintained and used by the Oracle server. Updates to a table's data, such as adding new rows, updating rows, or deleting rows, are automatically propagated to all relevant indexes with complete transparency to users.

Types of Indexes

- These are several types of index structures available to you, depending on the need:
  - A B-tree index is in the form of a binary tree and is the default index type.
  - A bitmap index has a bitmap for each distinct value indexed, and each bit position represents a row that may or may not contain the indexed value. This is best for low-cardinality columns.

Capgemini

Types of Indexes

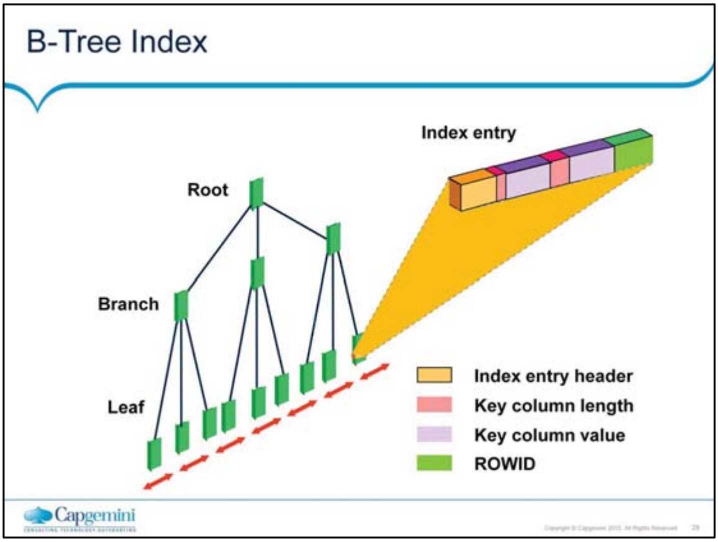
The following are the most common forms of indexes:

B-tree

Bitmap

A B-tree index has its key values stored in a balanced tree (B-tree), allowing for fast binary searches.

A bitmap index has a bitmap for each distinct key value being indexed. Within each bitmap, there is a bit set aside for each row in the table being indexed. This allows for fast lookups when there are few distinct values; that is, the indexed column has low cardinality. An example of this is a gender indicator. It can have values of “M” and “F” only. So, there are only two bitmaps to search. For example, if a bitmap index were used for a phone_number column, there would be so many bitmaps to manage and search that it would be very inefficient. Use bitmap indexes for low-cardinality columns.



B-Tree Index

Structure of a B-tree index

At the top of the index is the root, which contains entries that point to the next level in the index. At the next level are branch blocks, which in turn point to blocks at the next level in the index. At the lowest level are the leaf nodes, which contain the index entries that point to rows in the table. The leaf blocks are doubly linked to facilitate the scanning of the index in an ascending as well as descending order of key values.

Format of index leaf entries

An index entry is made up of the following components:

- An entry header, which stores the number of columns and locking information
- Key column length-value pairs, which define the size of a column in the key followed by the value for the column (The number of such pairs is a maximum of the number of columns in the index.)
- ROWID of a row that contains the key values

B-Tree Index (continued)

Index leaf entry characteristics

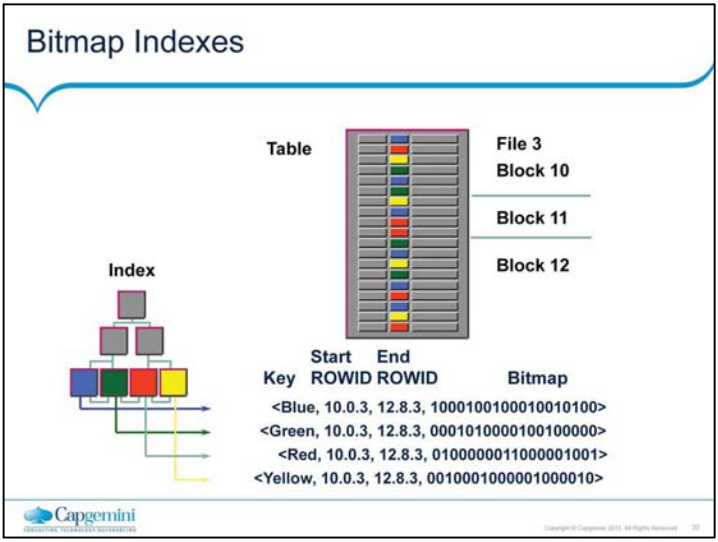
In a B-tree index on a nonpartitioned table:

- Key values are repeated if there are multiple rows that have the same key value unless the index is compressed
- There is no index entry corresponding to a row that has all key columns that are NULL. Therefore, a WHERE clause specifying NULL will always result in a full table scan.
- Restricted ROWID is used to point to the rows of the table because all rows belong to the same segment

Effect of DML operations on an index

The Oracle server maintains all the indexes when DML operations are carried out on the table. Here is an explanation of the effect of a DML command on an index:

- Insert operations result in the insertion of an index entry in the appropriate block.
- Deleting a row results only in a logical deletion of the index entry. The space used by the deleted row is not available for new entries until all the entries in the block are deleted.
- Updates to the key columns result in a logical delete and an insert to the index. The PCTFREE setting has no effect on the index except at the time of creation. A new entry may be added to an index block even if it has less space than that specified by PCTFREE.



Bitmap Indexes

Bitmap indexes are more advantageous than B-tree indexes in certain situations:

When a table has millions of rows and the key columns have low cardinality—that is, there are very few distinct values for the column. For example, bitmap indexes may be preferable to B-tree indexes for the gender and marital status columns of a table containing passport records.

When queries often use a combination of multiple WHERE conditions involving the OR operator

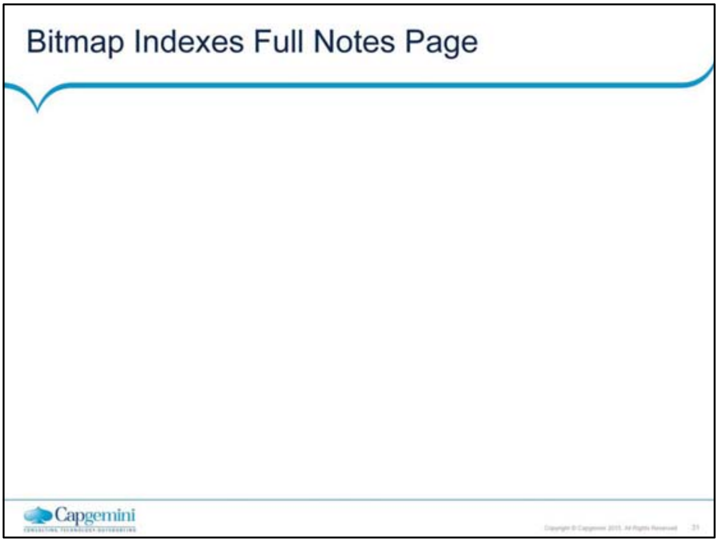
When there is read-only or low update activity on the key columns

Structure of a bitmap index

A bitmap index is also organized as a B-tree, but the leaf node stores a bitmap for each key value instead of a list of ROWIDs. Each bit in the bitmap corresponds to a possible ROWID, and if the bit is set, it means that the row with the corresponding ROWID contains the key value.

As shown in the diagram, the leaf node of a bitmap index contains the following:

An entry header that contains the number of columns and lock information



Bitmap Indexes (continued)

Structure of a bitmap index (continued)

Key values consisting of length and value pairs for each key column. In the example, the key consists of only one column, and the first entry has a key value of Blue.

Start ROWID, which in the example specifies block number ten, row number zero, and file number three

End ROWID, which in the example specifies block number twelve, row

number eight, and file number three

A bitmap segment consisting of a string of bits. (The bit is set when the corresponding row contains the key value and is unset when the row does not contain the key value. The Oracle server uses a patented compression technique to store bitmap segments.)

The start ROWID is the ROWID of the first row pointed to by the bitmap segment of the bitmap—that is, the first bit of the bitmap corresponds to that ROWID, the second bit of the bitmap corresponds to the next row in the block, and the end ROWID is a pointer to the last row in the table covered by the bitmap segment. Bitmap indexes use restricted ROWIDs.

Using a bitmap index

The B-tree is used to locate the leaf nodes that contain bitmap segments for a given value of the key. Start ROWID and the bitmap segments are used to locate the rows that contain the key value.

When changes are made to the key column in the table, bitmaps must be modified. This results in the locking of the relevant bitmap segments. Because locks are acquired on the whole bitmap segment, a row that is covered by the bitmap cannot be updated by other transactions until the first transaction ends.

Index Options

- A unique index ensures that every indexed value is unique.
- An index can have its key values stored in ascending or descending order.
- A reverse key index has its key value bytes stored in reverse order.
- A composite index is one that is based on more than one column.
- A function-based index is an index based on a function's return value.
- A compressed index has repeated key values removed.

Capgemini

Index Options

For efficiency of retrieval, it may be advantageous to have an index store the keys in descending order. This decision is made on the basis of how the data is accessed most frequently.

A reverse key index has the bytes of the indexed value stored in reverse order. This can reduce activity in a particular hot spot in the index. If many users are processing data in the same order, then the prefix portions of the key values (that are currently being processed) are close in value at any given instant. Consequently, there is a lot of activity in that area of the index structure. A reverse key index spreads that activity out across the index structure by indexing a reversed-byte version of the key values.

An index created by the combination of more than one column is called a composite index. For example, you can create an index based on a person's last name and first name:

```
CREATE INDEX name_ix ON employees
(last_name, first_name);
```

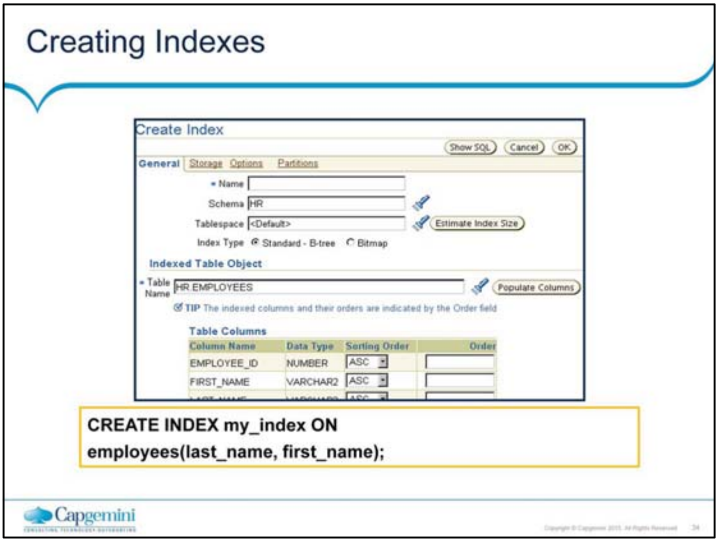
Index Options (continued)

A function-based index indexes a function's return value. This function can be a built-in SQL function, a supplied PL/SQL function, or a user-written function. This relieves the server from having to invoke the function for every key value as it performs a search on the indexed expression. The following example indexes the returned tree volume that is computed by the function, based on each tree's species, height, and volume (which are columns in the TREES table):

```
CREATE INDEX tree_vol_ix ON
TREES(volume(species,height,circumferenc
e));
```

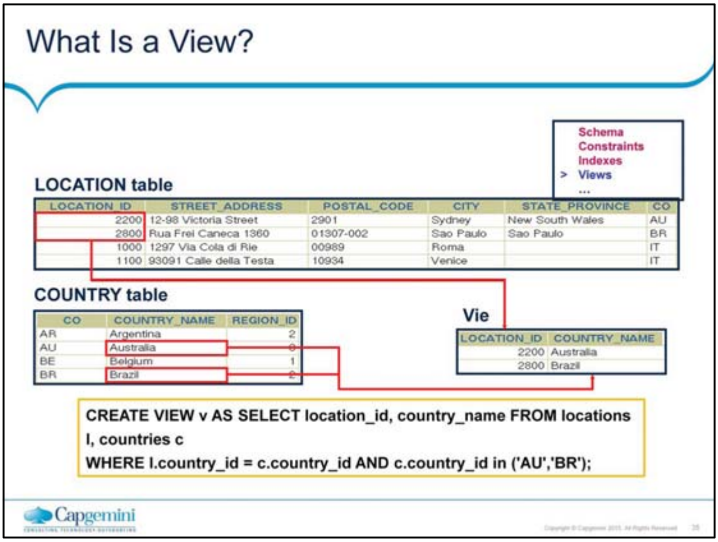
Then, any query that contains the expression volume(species,height,circumference) in the WHERE clause may be able to take advantage of this index, and execute much more quickly because the volume computation is already done for each tree. Function-based indexes are maintained automatically, as are normal indexes.

You can use a compressed index to reduce disk consumption at execution time. Because repeated key values are removed, more index entries can fit in a given amount of disk space, resulting in the ability to read more entries from the disk in the same amount of time. Compression and decompression must be performed for the writing and reading of the index, respectively.



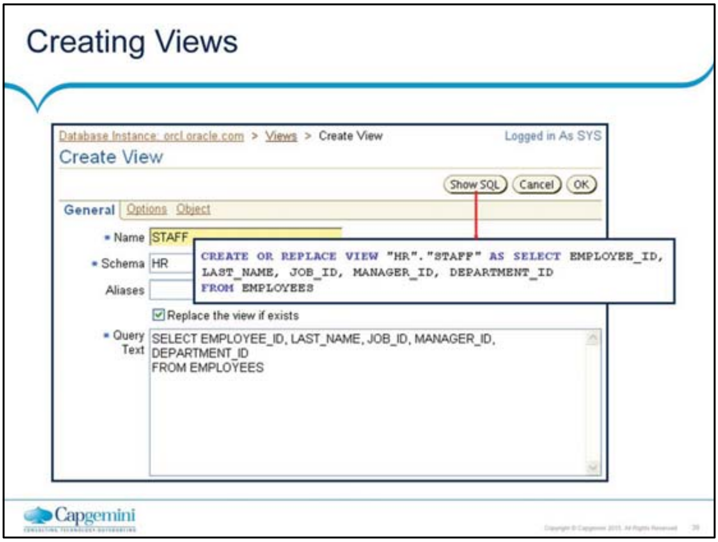
Creating Indexes

You can click the Indexes link under the Schema heading of the Administration page to view the Indexes page. You can view index attributes or use the Actions menu to view dependencies for an index. Indexes can be created explicitly, or implicitly through constraints that are placed on a table. An example of an implicitly created index is the definition of a primary key, in which case a unique index would be automatically created to enforce uniqueness on the column.



What Is a View?

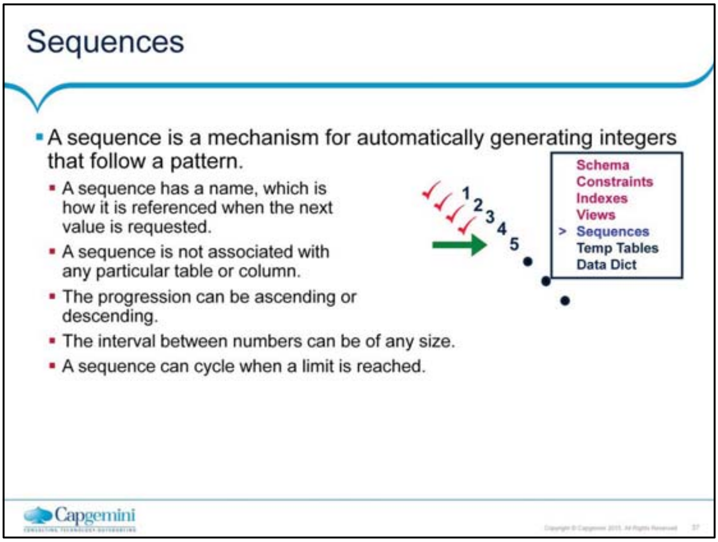
Views are customized representations of data in one or more tables or other views. They can be thought of as stored queries because they can hide very complex conditions, joins, and other complex expressions and SQL constructs. Views do not actually contain data; instead, they derive their data from the tables on which they are based. These tables are referred to as the base tables of the view.



Creating Views

Like tables, views can be queried, updated, inserted into, and deleted from, with some restrictions. All operations performed on a view actually affect the base tables of the view. Views provide an additional level of security by restricting access to a predetermined set of rows and columns of a table. They also hide data complexity and store complex queries.

To see the views defined in the database, click the Views link under the Schema heading on the Administration page.



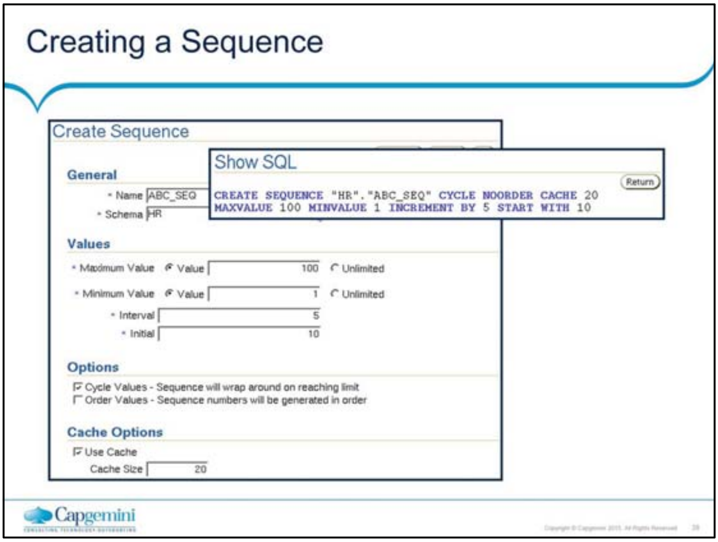
Sequences

To retrieve the next value from a sequence, you reference it by its name; there is no association of a sequence to a table or a column.

After a given number is issued, it will not be issued again, unless the sequence is defined as cyclical. Sometimes an application requests a value it never ends up using or storing in the database. This may result in gaps in the numbers that reside in the table that they are being stored into.

Caching of sequence numbers improves performance because a set of numbers is preallocated in memory for faster access. If there is an instance failure, any cached sequence numbers are not used, which results in gaps.

Note: If an application requires that there be no gaps, then the application should implement a custom number generator. However, this method can result in very poor performance. If you use a table to store a value, and increment that value and update the table for each request, that process would be a systemwide bottleneck. This is because every session would have to wait for that mechanism, which, to guarantee no duplicates or gaps, can handle only a single request at a time.



Creating a Sequence

You can view and create sequences with Enterprise Manager by clicking the Sequences link under the Schema heading of the Administration page. Here is a summary of the sequence creation options:

Name: This is the name of the sequence, which is how it is referenced.

Schema: This is the owner of the sequence.

Maximum Value: Specify the maximum value that the sequence can generate. This integer value can have 28 or fewer digits. It must be greater than Minimum Value and Initial. Using Unlimited indicates the maximum value of 1027 for an ascending sequence or –1 for a descending sequence. The default is Unlimited.

Minimum Value: Specify the minimum value of the sequence. This integer value can have 28 or fewer digits. It must be less than or equal to Initial and less than Maximum Value. Using Unlimited indicates the minimum value of 1 for an ascending sequence or –1026 for a descending sequence. The default is Unlimited.

Creating a Sequence (continued)

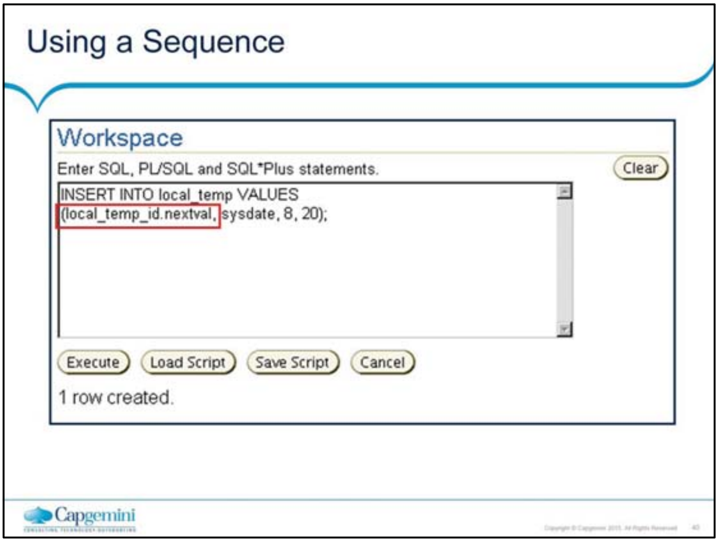
Interval: Specify the interval between sequence numbers. This integer value can be any positive or negative integer, but it cannot be zero. It can have 28 or fewer digits. The default value is one.

Initial: Specify the first sequence number to be generated. Use this clause to start an ascending sequence at a value greater than its minimum or to start a descending sequence at a value less than its maximum.

Cycle Values: After an ascending sequence reaches its maximum value, it generates its minimum value. After a descending sequence reaches its minimum, it generates its maximum value. If you do not choose this option, an error is returned when you attempt to retrieve a value after the sequence has been exhausted.

Order Values: This guarantees that sequence numbers are generated in the order of request. This clause is useful if you are using sequence numbers as timestamps. Guaranteeing order is usually not important for sequences that are used to generate primary keys. This option is necessary only to guarantee ordered generation if you are using the Oracle database with Real Application Clusters.

Cache Options: Specify how many values of the sequence the Oracle database preallocates and keeps in memory for faster access. This integer value can have 28 or fewer digits. The minimum value for this parameter is 2. For sequences that cycle, this value must be less than the number of values in the cycle. You cannot cache more values than what would fit in a given cycle of sequence numbers.



Using a Sequence

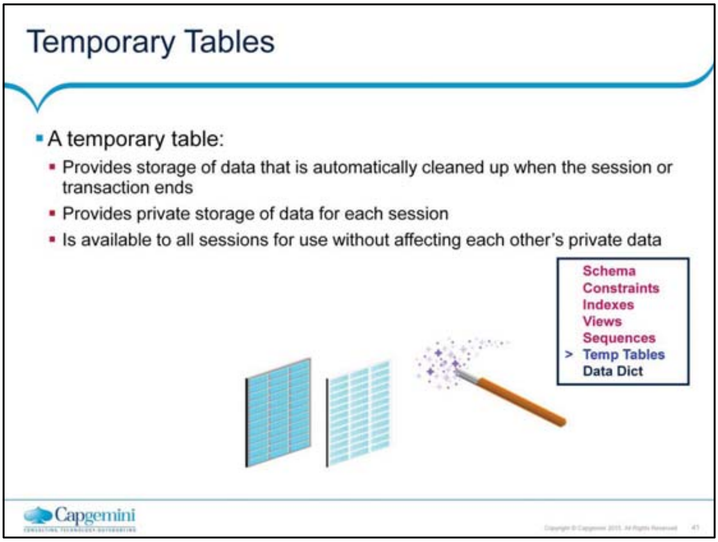
Refer to sequence values in SQL statements with the following pseudocolumns:

- CURRVAL: Returns the current value of a sequence
- NEXTVAL: Increments the sequence and returns the next value

You must qualify CURRVAL and NEXTVAL with the name of the sequence:

sequence.CURRVAL
sequence.NEXTVAL

The first reference to NEXTVAL returns the initial value of the sequence. Subsequent references to NEXTVAL increment the sequence value by the defined increment and return the new value. Any reference to CURRVAL always returns the current value of the sequence, which is the value returned by the last reference to NEXTVAL.



Temporary Tables

You can take advantage of temporary tables when you need to privately store data for the purpose of performing a task, and you want the data to be cleaned up when that task is performed, at the end of either a transaction or a session. Temporary tables provide this functionality while relieving you of the responsibilities of hiding your data from other sessions, and removing the generated data when you have finished. The only temporary table data visible to a session is the data that the session has inserted.

A temporary table can be transaction specific or session specific. For transaction-specific temporary tables, data exists for the duration of the transaction whereas for session-specific temporary tables, data exists for the duration of the session. In both cases, the data inserted by a session is private to the session. Each session can view and modify only its own data. As a result, DML locks are never acquired on the data of temporary tables.

The following clauses control the lifetime of the rows:

- ON COMMIT DELETE ROWS: To specify that the lifetime of the inserted rows is for the duration of the transaction only
- ON COMMIT PRESERVE ROWS: To specify that the lifetime of the inserted rows is for the duration of the session

Temporary Tables Full Notes Page

Temporary Tables (continued)

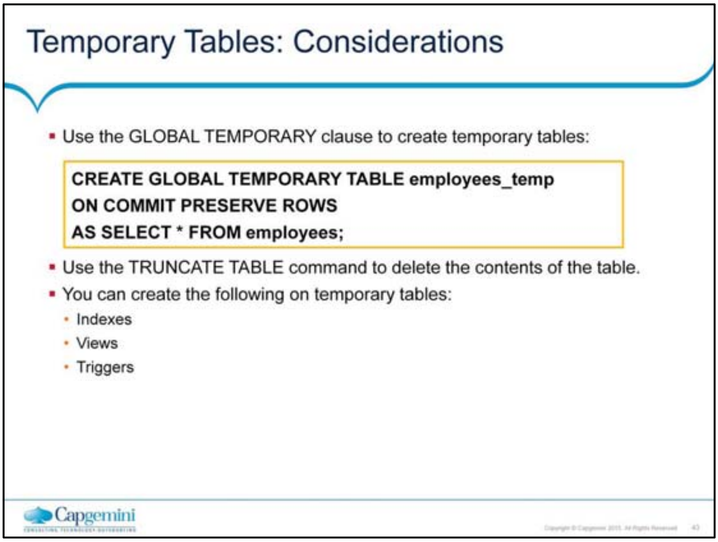
The CREATE GLOBAL TEMPORARY TABLE statement creates a temporary table. You can create indexes, views, and triggers on temporary tables, and you can also use Export and Import or Data Pump to export and import the definition of a temporary table. However, no data is exported, even if you use the ROWS option.

In addition to the already mentioned events that cause the data to be deleted, you can force the data to be removed efficiently with the TRUNCATE TABLE command. This removes all the data that you have inserted. It is more efficient than using the DELETE command.

You can create indexes, views, and triggers on temporary tables.

Temporary tables can be created using Enterprise Manager by clicking the Temporary option on the Create Table: Table Organization page. Click Continue, and the next page enables you to specify whether the temporary table is session specific or transaction specific. The Tablespace field is disabled because a temporary table is always created in the user's temporary tablespace; no other tablespace can be specified.

Note: The GLOBAL keyword is based on the terminology specified in the International Organization for Standardization (ISO) standard for SQL.

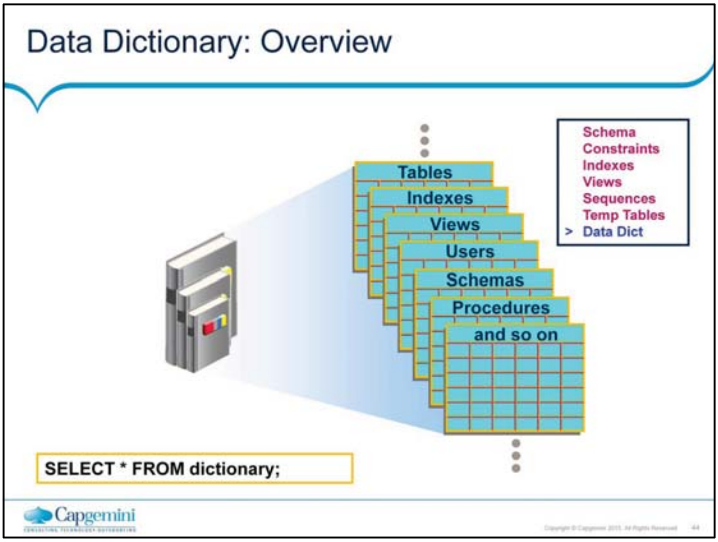


Temporary Tables: Considerations

The CREATE GLOBAL TEMPORARY TABLE statement creates a temporary table. You can create indexes, views, and triggers on temporary tables, and you can also use Export and Import or Data Pump to export and import the definition of a temporary table. However, no data is exported even if you use the ROWS option.

In addition to the already mentioned events that cause the data to be deleted, you can force the data to be removed efficiently with the TRUNCATE TABLE command. This removes all the data that you have inserted. It is more efficient than using the DELETE command.

Note: The GLOBAL keyword is based on the terminology specified in the International Organization for Standardization (ISO) standard for SQL.



Data Dictionary: Overview

Oracle’s data dictionary is the description of a database. It 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

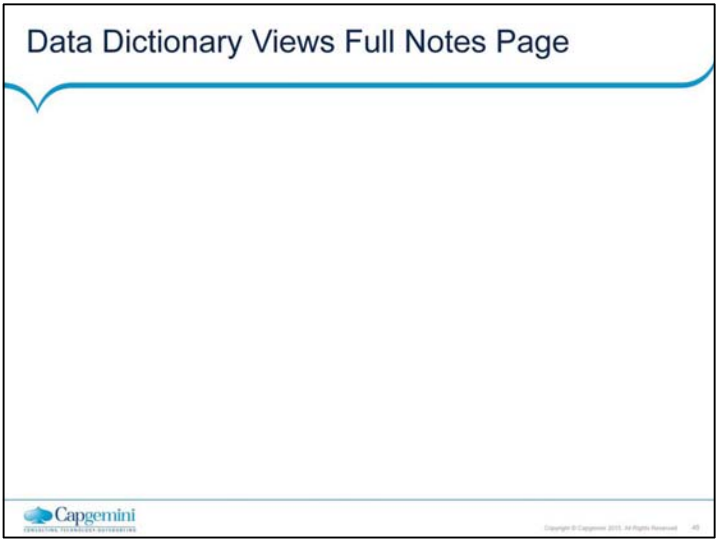
Note: The DICTIONARY data dictionary view, or the DICT synonym for this, contains the names and descriptions of everything in the data dictionary. Use the DICT_COLUMNS view to see the view columns and their definitions. For complete definitions of each view, see the Oracle Database Reference documentation.

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
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.

Data Dictionary Views

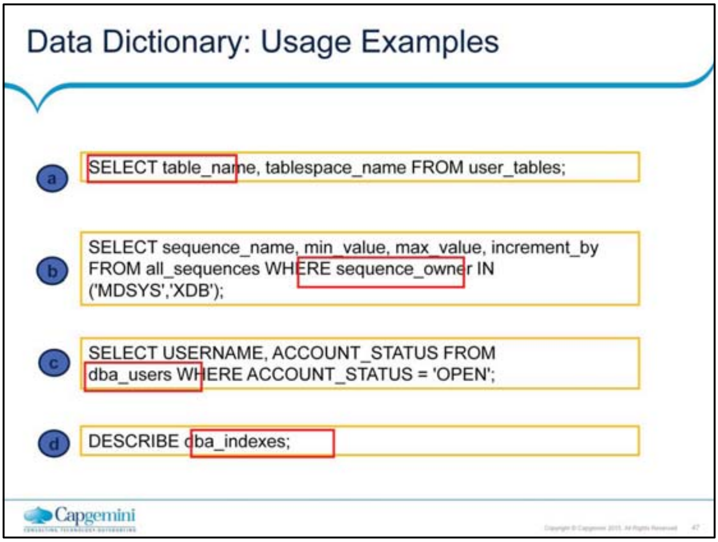
The view prefixes indicate what or how much data a given user can see. The global view of everything is accessed only by users with DBA privileges, using the DBA_ prefix. The next level of privilege is at the ALL_ prefix level, which represents all objects that the querying user is privileged to see, whether he or she 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 shows only those objects that the querying user owns; that is, those that are present in his or her own schema.



Data Dictionary Views (continued)

Generally, each view set is a subset of the higher privileged view set, row-wise and columnwise. 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. You should be certain 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 you know that it is something that you own and you do not want other objects to be added to your result set.

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



Static Data Dictionary: Usage Examples

The examples in the slide show queries that answer these questions:

a. What are the names of the tables (along with the name of the tablespace where they reside) that have been created in your schema?

b. What is the significant information about any sequences in the database that you have access to?

c. What users in this database are currently able to log in?

d. 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)
```

Summary

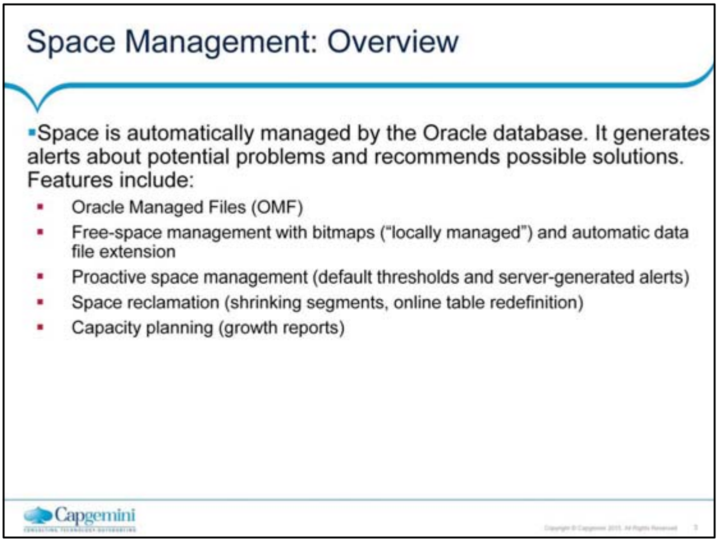
- In this lesson, you should have learned how to:
  - Define schema objects and data types
  - Create and modify tables
  - Define constraints
  - View the columns and contents of a table
  - Create indexes
  - Create views
  - Create sequences
  - Explain the use of temporary tables
  - Use the data dictionary

Oracle 11g DBA Fundamentals Overview

Lesson 10: Managing Storage

Objectives

- After completing this lesson, you should be able to:
  - Describe how the Oracle database automatically manages space
  - Proactively monitor and manage tablespace space usage
  - Use the Segment Advisor
  - Reclaim wasted space from tables and indexes by using the segment shrink functionality
  - Manage resumable space allocation
  - Describe the concepts of transportable tablespaces and databases



Space Management: Overview

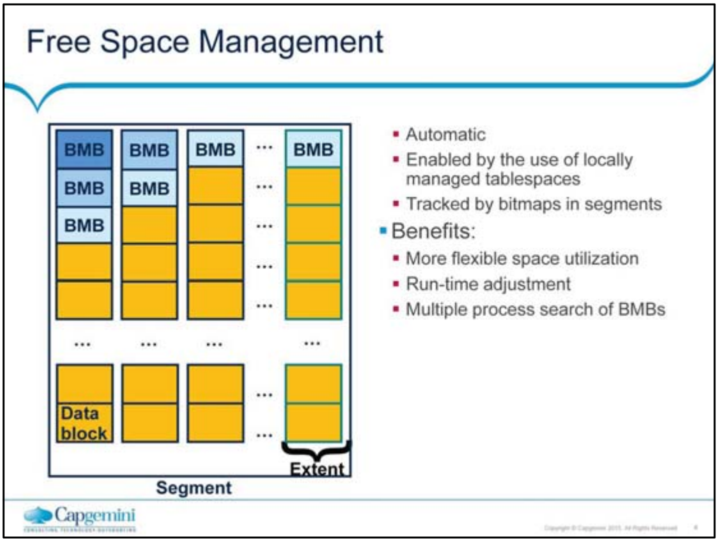
With Oracle Managed Files (OMF), you can specify operations in terms of database objects rather than file names. For more details, see the lesson titled "Introduction."

The Oracle database manages free space within a table with bitmaps. This is known as a "locally managed" tablespace. (Dictionary-managed tablespaces are supported only for backward compatibility.) The bitmapped implementation eliminates much space-related tuning of tables, while providing improved performance during peak loads. Additionally, the Oracle database provides automatic extension of data files, so the files can grow automatically based on the amount of data in the files.

When you create a database, proactive space monitoring is enabled by default. (This causes no performance impact.) The Oracle database monitors space utilization during normal space allocation and deallocation operations and alerts you if the free space availability falls below the predefined thresholds (which you can override). Advisors and wizards assist you with space reclamation.

For capacity planning, the Oracle database provides space estimates based on table structure and number of rows and a growth trend report based on historical space utilization stored in the Automatic Workload Repository (AWR).

The Oracle Database 11g: Administration Workshop I course provides an introduction to space and storage concepts, related utilities, and DBA tasks. Through this or other means, you should be familiar with the basic concepts and storage features.



Free Space Management

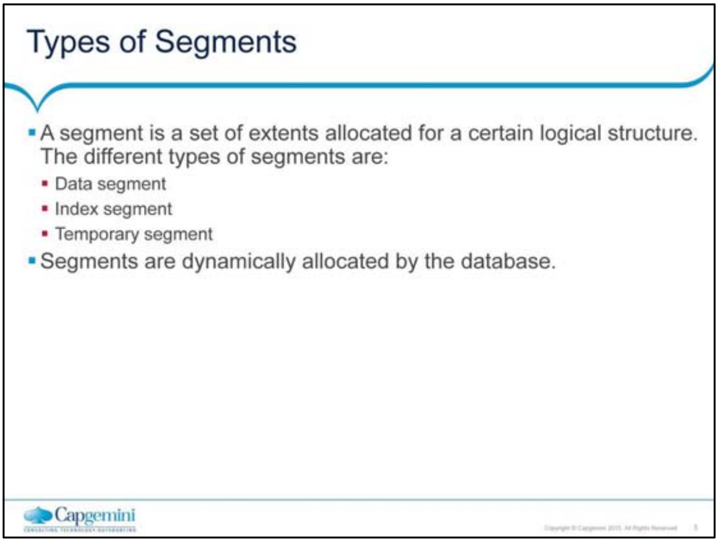
Free space can be managed automatically inside database segments. The in-segment free or used space is tracked with bitmaps. You specify Automatic Segment Space Management, when you create a locally managed tablespace. Your specification then applies to all segments subsequently created in this tablespace.

Automatic space management segments have a set of bitmap blocks (BMBs) describing the space utilization of the data blocks in that segment. BMBs are organized in a tree hierarchy. The root level of the hierarchy, which contains the references to all intermediate BMBs, is stored in the segment header. The leaves of this hierarchy represent the space information for a set of contiguous data blocks that belong to the segment. The maximum number of levels inside this hierarchy is three.

Benefits of using automatic space management (compared to manual space management, which uses "freelist" data structures and is synonymous with "dictionary-managed" tablespaces):

- Better space utilization, especially for the objects with highly varying row sizes
- Better run-time adjustment to variations in concurrent access
- Better multi-instance behavior in terms of performance or space utilization

Therefore, less work for you, the DBA.



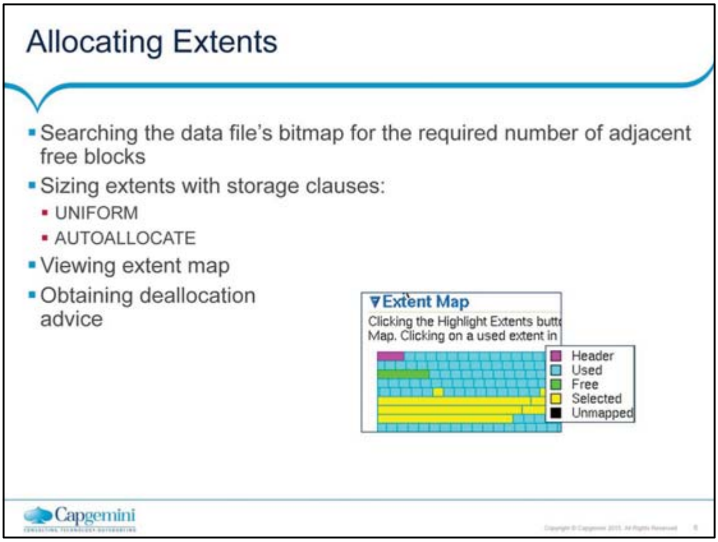
Types of Segments

Data segment: Each nonclustered table has a data segment. All table data is stored in the extents of the 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 segment: Each index has an index segment that stores all of its data. For a partitioned index, each partition has an index segment.

Temporary segment: A temporary segment is created by the Oracle database when a SQL statement needs a temporary database area to complete execution. When the statement finishes execution, the extents in the temporary segment are returned to the system for future use.

The Oracle database dynamically allocates space when the existing extents of a segment become full. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.



Allocating Extents

With locally managed tablespaces, the Oracle database looks for free space to allocate to a new extent by first determining a candidate data file in the tablespace and then searching the data file's bitmap for the required number of adjacent free blocks. If that data file does not have enough adjacent free space, then the Oracle database looks in another data file.

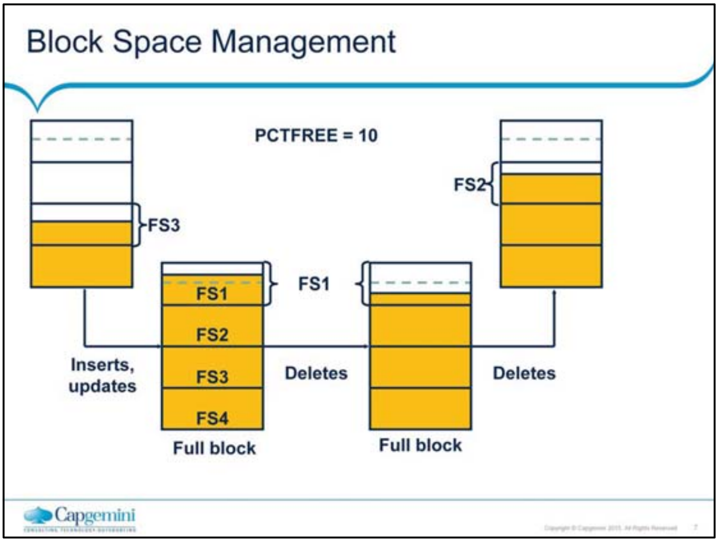
Two clauses affect the sizing of extents:

With the UNIFORM clause, the database creates all extents of a uniform size that you specified (or a default size) for any objects created in the tablespace.

With the AUTOALLOCATE clause, the database determines the extent-sizing policy for the tablespace.

To view the extent map in Enterprise Manager, choose Administration > Tablespaces > View Tablespace > Show Tablespace Contents.

The Oracle database provides a Segment Advisor that helps you determine whether an object has space available for reclamation on the basis of the level of space fragmentation within the object.

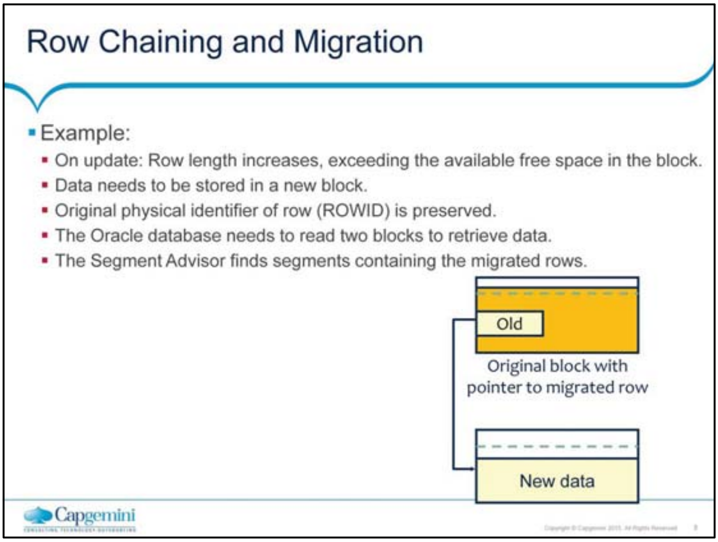


Block Space Management

Space management involves the management of free space at the block level. With Automatic Segment Space Management, each block is divided into four sections, named FS1 (between 0 and 25% of free space), FS2 (25% to50% free), FS3 (50% to 75% free), and FS4 (75% to 100% free). Depending on the level of free space in the block, its status is automatically updated. That way, depending on the length of an inserted row, you can tell whether a particular block can be used to satisfy an insert operation. Note that a “full” status means that a block is no longer available for inserts.

In the slide example, the block on the left is an FS3 block because it has between 50% and 75% free space. After some insert and update statements, PCTFREE is reached (the dashed line) and it is no longer possible to insert new rows in that block. The block is now considered as a “full” or FS1 block. The block is considered for insertion again, as soon as its free space level drops below the next section. In the above case, it gets status FS2 as soon as the free space is more than 25%.

Note: Large object (LOB) data types (BLOB, CLOB, NCLOB, and BFILE) do not use the PCTFREE storage parameter. For more information, see the Oracle Database Application Developer’s Guide - Large Objects 11g Release 2 (10.2).



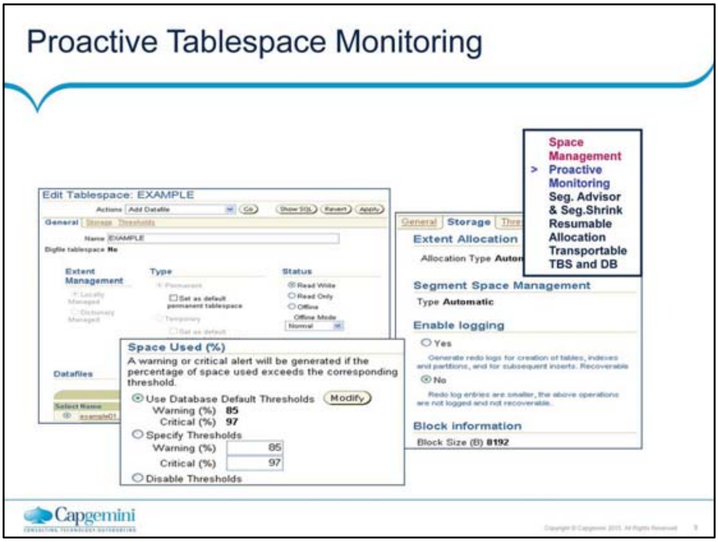
Row Chaining and Migrating

In two circumstances, the data for a row in a table may be too large to fit into a single data block. In the first case, the row is too large to fit into one data block when it is first inserted. In this case, the Oracle database stores the data for the row in a chain of data blocks (one or more) reserved for that segment. Row chaining most often occurs with large rows, such as rows that contain a column of data type LONG or LONG RAW. Row chaining in these cases is unavoidable.

However, in the second case, a row that originally fit into one data block is updated, so that the overall row length increases, and the block's free space is already completely filled. In this case, the Oracle database migrates the data for the entire row to a new data block, assuming that the entire row can fit in a new block. The database preserves the original row piece of a migrated row to point to the new block containing the migrated row. The ROWID of a migrated row does not change.

When a row is chained or migrated, I/O performance associated with this row decreases because the Oracle database must scan more than one data block to retrieve the information for the row.

The Segment Advisor finds the segments containing migrated rows that result from an UPDATE.

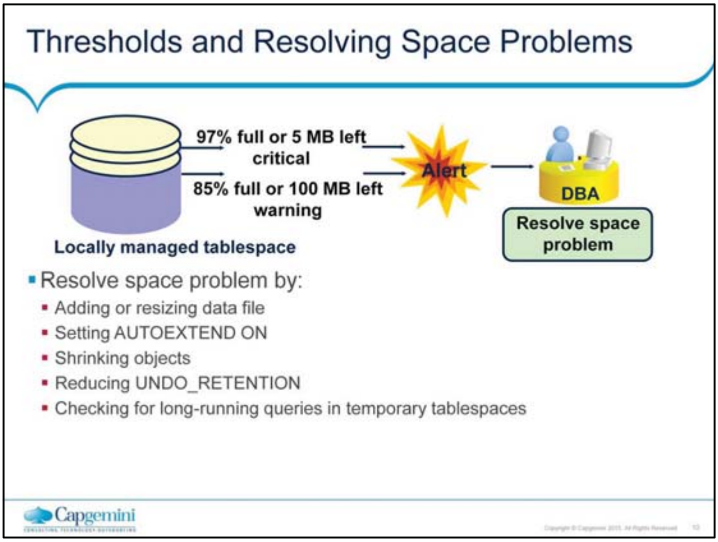


Proactive Tablespace Monitoring

Tablespace disk space usage is proactively managed by the database in the following ways:

- Through the use of database alerts, you are informed when a tablespace runs low on available disk space as well as when particular segments are running out of space. You can then provide the tablespace with more disk space, thus avoiding out-of-space conditions.
- Information gathered is stored in the Automatic Workload Repository (AWR) and is used to perform growth trend analysis and capacity planning of the database.

To view and modify tablespace information in Enterprise Manager, select Administration > Tablespaces. Select the tablespace of your choice and click the Edit button.



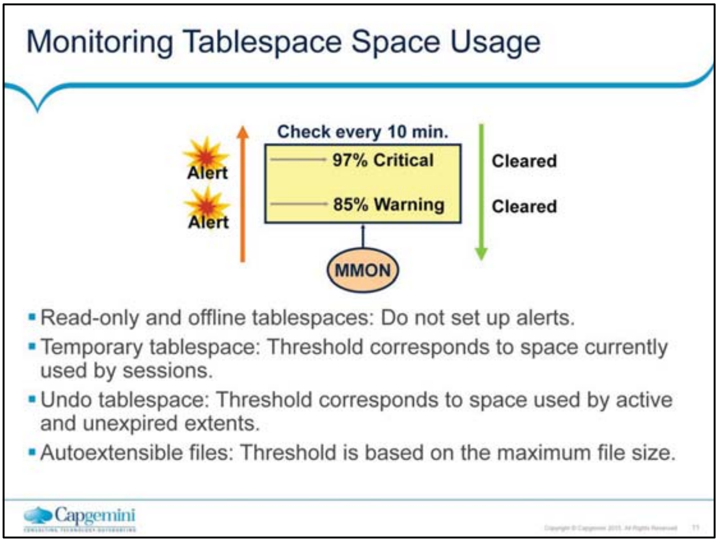
Thresholds and Resolving Space Problems

Tablespace thresholds are defined either as full or as available space in the tablespace. Critical and warning thresholds are the two thresholds that apply to a tablespace. The DBMS_SERVER_ALERT package contains procedures to set and get the threshold values. When the tablespace limits are reached, an appropriate alert is raised. The threshold is expressed in terms of a percentage of the tablespace size or in remaining bytes free. It is calculated in memory. You can have both a percentage and a byte-based threshold defined for a tablespace. Either or both of them may generate an alert.

The ideal setting for the warning threshold trigger value results in an alert that is early enough to ensure that there is enough time to resolve the problem before it becomes critical, but late enough so that you are not bothered when space is not a problem.

The alert indicates that the problem can be resolved by doing one or more of the following:

- Adding more space to the tablespace by adding a file or resizing existing files, or making an existing file autoextendable
- Freeing up space on disks that contain any autoextendable files
- Shrinking sparse objects in the tablespace



Monitoring Tablespace Space Usage

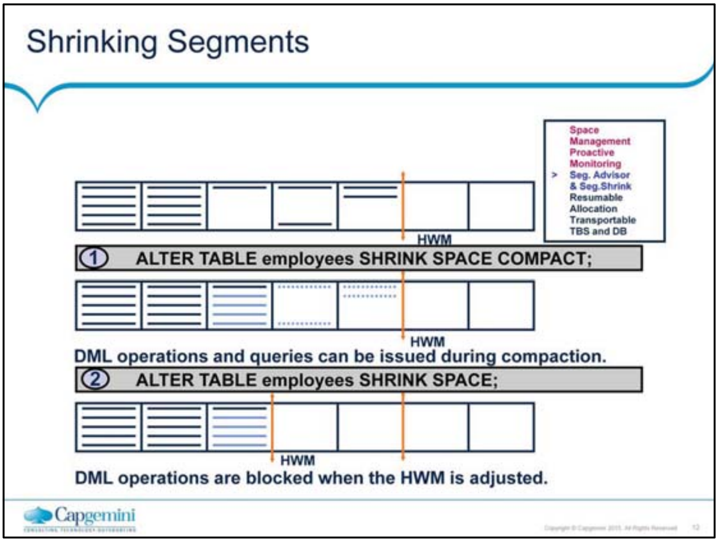
The database tracks space utilization while performing regular space management activities. This information is aggregated every 10 minutes by the MMON process. An alert is triggered when the threshold for a tablespace has been reached or cleared.

Alerts should not be flagged on tablespaces that are in read-only mode, or tablespaces that were taken offline, because there is not much to do for them.

In temporary tablespaces, the threshold value has to be defined as a limit on the used space in the tablespace.

For undo tablespaces, an extent is reusable if it does not contain active or unexpired undo. For the computation of threshold violation, the sum of active and unexpired extents is considered as used space.

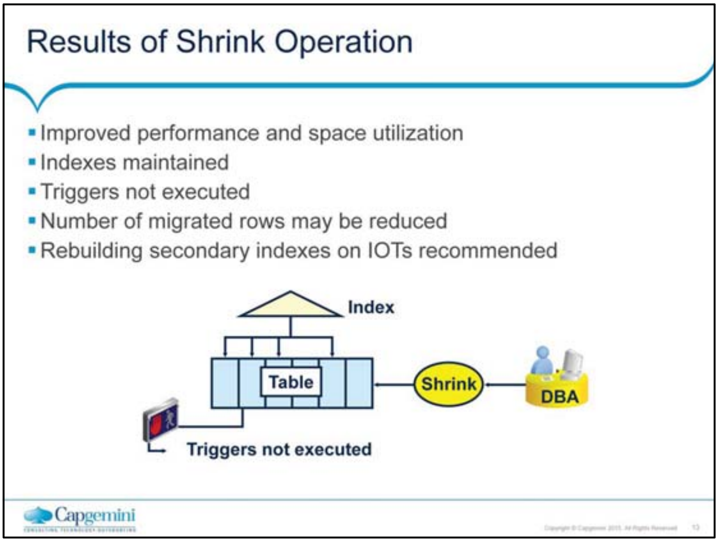
For tablespaces with autoextensible files, the thresholds are computed according to the maximum file size you specified, or the maximum OS file size.



Shrinking Segments

The diagram in the slide describes the two phases of a table shrink operation. The first phase does the compaction. During this phase, rows are moved to the left part of the segment as much as possible. Internally, rows are moved by packets to avoid locking issues. After the rows have been moved, the second phase of the shrink operation is started. During this phase, the high-water mark (HWM) is adjusted and the unused space is released.

The COMPACT clause is useful if you have long-running queries that might span the shrink operation and attempt to read from blocks that have been reclaimed. When you specify the SHRINK SPACE COMPACT clause, the progress of the shrink operation is saved in the bitmap blocks of the corresponding segment. This means that the next time a shrink operation is executed on the same segment, the Oracle database remembers what has been done already. You can then reissue the SHRINK SPACE clause without the COMPACT clause during off-peak hours to complete the second phase.



Results of Shrink Operation

Shrinking a sparsely populated segment improves the performance of scan and DML operations on that segment. This is because there are fewer blocks to look at after the segment has been shrunk. This is especially true for:

Full table scans (fewer and denser blocks)

Better index access (fewer I/Os on range ROWID scans due to a more compact tree)

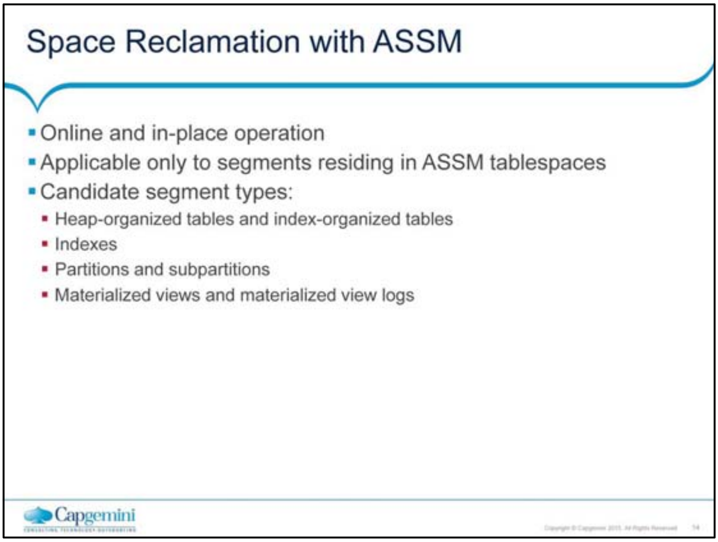
Also, by shrinking sparsely populated segments, you enhance the efficiency of space utilization inside your database because more free space is made available for objects in need.

Index dependency is taken care of during the segment shrink operation. The indexes are in a usable state after shrinking the corresponding table. Therefore, no further maintenance is needed.

The actual shrink operation is handled internally as an INSERT/DELETE operation. However, any DML triggers are not executed because the data itself is not changed.

As a result of a segment shrink operation, it is possible that the number of migrated rows is reduced. However, you should not always depend on reducing the number of migrated rows after a segment has been shrunk. This is because a segment shrink operation may not touch all the blocks in the segment. Therefore, it is not guaranteed that all the migrated rows are handled.

Note: It is recommended to rebuild secondary indexes on an index-organized table (IOT) after a shrink operation.



Space Reclamation with ASSM

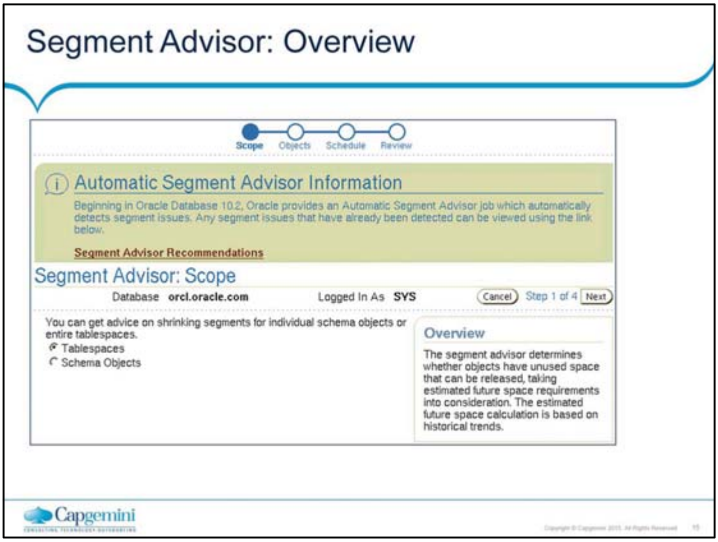
A shrink operation is an online and in-place operation because it does not need extra database space to be executed.

You cannot execute a shrink operation on segments managed by free lists. Segments in automatic segment-space managed tablespaces can be shrunk. However, the following objects stored in ASSM tablespaces cannot be shrunk:

- Tables in clusters
- Tables with LONG columns
- Tables with on-commit materialized views
- Tables with ROWID-based materialized views
- IOT mapping tables
- Tables with function-based indexes

ROW MOVEMENT must be enabled for heap-organized segments.

Note: Automatic Segment Space Management (ASSM) is the default type of segment space management for all new, permanent, locally managed tablespaces in Oracle Database 11g Release 2.



Segment Advisor: Overview

The Segment Advisor determines whether an object is a good candidate for a shrink operation. The advisor also finds the segments containing migrated rows that result from an UPDATE. (Beginning with Oracle Database 10.2, the Segment Advisor jobs are automatically run for you.) The advisor makes recommendations based on the amount of unused space that can be released, and takes into consideration estimated future space requirements by using criteria from the gathered information about segment growth trends.

After the recommendations are made, you can choose to implement the recommendations. The shrink advisor can be invoked at the segment or tablespace level.

The EM Database Control Console is the interface to the Segment Advisor. You can access the Segment Advisor from several places within EM:

- Advisor Central page
- Tablespaces page
- Schema object pages

The Database Control Console provides the option to select various inputs and schedule a job that calls the Segment Advisor to get shrink advice. The Segment Advisor wizard can be invoked with no context, in the context of a tablespace, or in the context of a schema object.

The Segment Advisor makes recommendation on the basis of sampled analysis, historical information, and future growth trends.

Segment Advisor

From the Administration page, select Tablespaces in the Storage section. On the Tablespaces page, select the tablespace on which you want to perform the shrink analysis, and then select Run Segment Advisor in the Actions drop-down list. Click Go to open the Segment Advisor initial page. You must choose “comprehensive” or “limited’ analysis mode. In comprehensive mode, the analysis is longer because the advisor is sampling the segments to identify the right targets.

Keep clicking Continue to answer the various questions of the advisor. You end up on the Segment Advisor: Review page, where you can review the details of your analysis. The Segment Advisor analysis is run as a scheduled job, so you can review the scheduled task from the Advisor Central page. When completed, you can review the advisor’s recommendations.

Note: In the Segment Advisor, you can specify the duration of the analysis. This enables you to limit the time the advisor takes to produce recommendations. Generally speaking, a longer analysis period produces more comprehensive results. The results are stored in the AWR and can be viewed later. Use the “Number of days to retain” option to instruct the Oracle database how long these results should be preserved before being purged from the AWR.

Segment Advisor Full Notes Page

Segment Advisor (continued)

Scope Objects Schedule **Review**

Execute task script

Review: Show SQL

Return

Create task and objects script

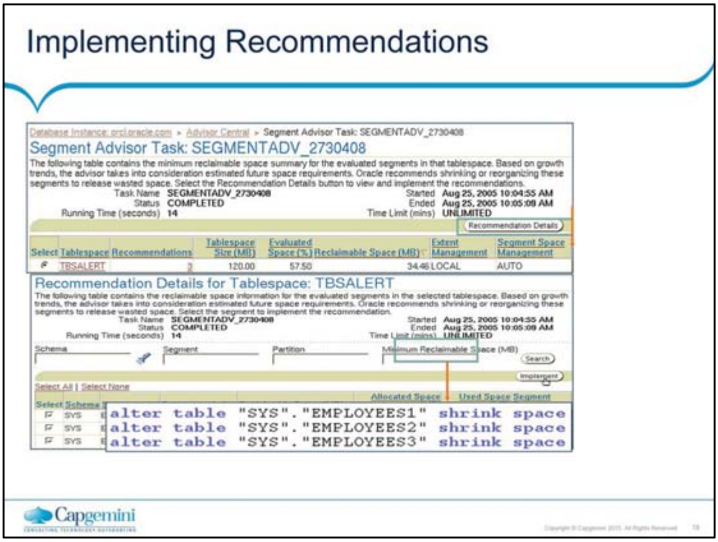
```
DECLARE

taskname varchar2(100);
taskdesc varchar2(128);
task_id number;
object_id number;
timeLimit varchar2(25);
numDaysToRetain varchar2(25);
objectName varchar2(100);
objectType varchar2(100);

BEGIN
taskname := 'SEGMENTADV_3712320';
taskdesc :='Get shrink advice based on object growth trend';
numDaysToRetain :='30';
dbms_advisor.create_task('Segment Advisor',?,taskname,taskdesc,NULL);
dbms_advisor.create_object(taskname, 'TABLESPACE', 'EXAMPLE', ' ', ' ',
NULL, object_id);
dbms_advisor.set_task_parameter(taskname, 'RECOMMEND_ALL', 'TRUE');
dbms_advisor.set_task_parameter(taskname, 'DAYS_TO_EXPIRE',
numDaysToRetain);
END;
```

Execute task script Return to Top

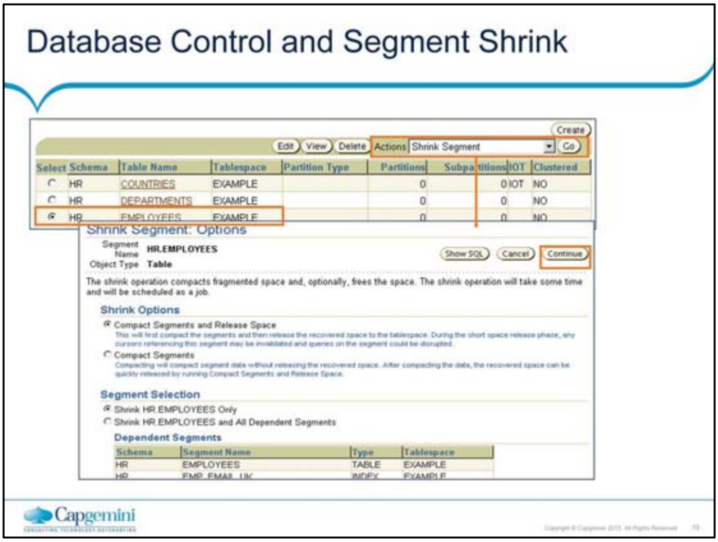
```
DECLARE
taskname varchar2(100);
BEGIN
taskname := 'SEGMENTADV_3712320';
dbms_advisor.reset_task(taskname);
dbms_advisor.execute_task(taskname);
END;
```



Implementing Recommendations

After the Segment Advisor completes its job, you can view the recommendation details and implement them directly.

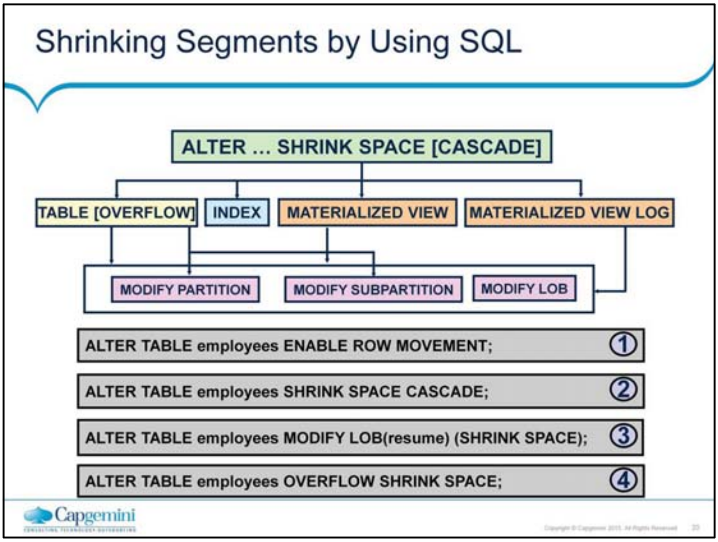
Note: Before shrinking a heap-organized table, you must enable row movement on that table. You can do this with Database Control from the Options tab on the Edit Table page.



Database Control and Segment Shrink

Alternatively (to implementing the Segment Advisor recommendations), you can shrink individual segments. For example, from the Database Control home page, click the Tables link in the Storage section. On the Tables page, select your table, and then select Shrink Segment in the Actions drop-down list. Then click the Go button. This brings you to the Shrink Segment page, where you can choose the dependent segments to shrink. You have the opportunity to compact only or to compact and release the space. You can also choose the CASCADE option.

When done, click the Continue link. This submits the shrink statements as a scheduled job.



Shrinking Segments by Using SQL

Because a shrink operation may cause ROWIDs to change in heap-organized segments, you must enable row movement on the corresponding segment before executing a shrink operation on that segment. Row movement by default is disabled at segment level. To enable row movement, the ENABLE ROW MOVEMENT clause of the CREATE TABLE or ALTER TABLE command is used. This is illustrated in the first example in the slide.

Use the ALTER command to invoke segment shrink on an object. The object’s type can be one of the following: table (heap- or index-organized), partition, subpartition, LOB (data and index segment), index, materialized view, or materialized view log.

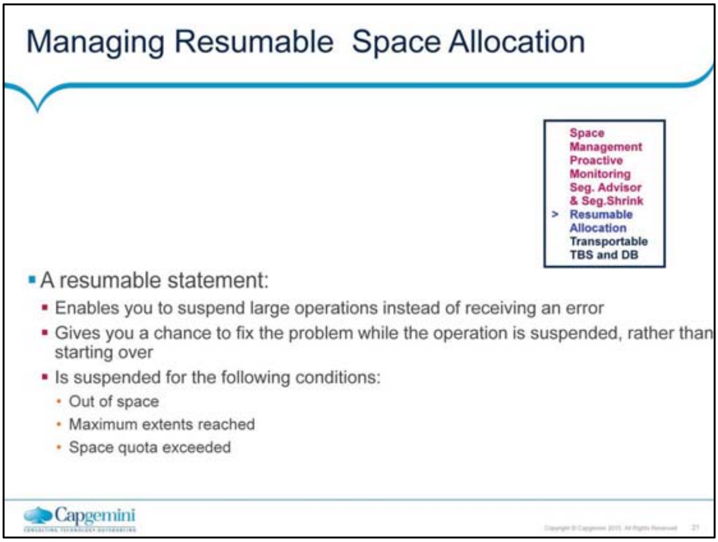
Use the SHRINK SPACE clause to shrink space in a segment. If CASCADE is specified, the shrink behavior is cascaded to all the dependent segments that support a shrink operation, except materialized views, LOB indexes, and IOT (index-organized tables) mapping tables. The SHRINK SPACE clause is illustrated in the second example.

In an index segment, the shrink operation coalesces the index before compacting the data.

Example 3 shows a command that shrinks a LOB segment, given that the RESUME column is a CLOB.

Example 4 shows a command that shrinks an IOT overflow segment belonging to the EMPLOYEES table.

Note: For more information, refer to the Oracle Database SQL Reference guide.



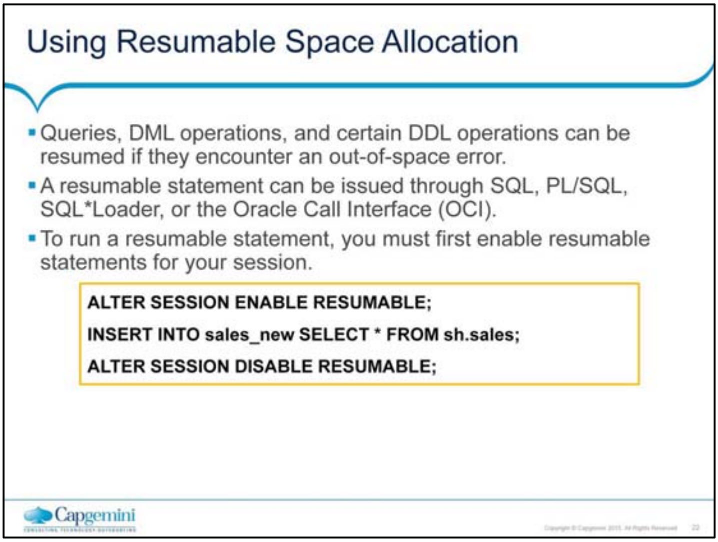
Managing Resumable Space Allocation

The Oracle database provides a means for suspending, and later resuming, the execution of large database operations in the event of space allocation failures. This enables you to take corrective action instead of the Oracle database server returning an error to the user. After the error condition is corrected, the suspended operation automatically resumes. This feature is called “resumable space allocation.” The statements that are affected are called “resumable statements.”

A statement executes in resumable mode only when the resumable statement feature has been enabled for the system or session.

Suspending a statement automatically results in suspending the transaction. Thus all transactional resources are held through the suspension and resuming of a SQL statement. When the error condition disappears (for example, as a result of user intervention or perhaps sort space released by other queries), the suspended statement automatically resumes execution.

A suspension time-out interval is associated with resumable statements. A resumable statement that is suspended for the time-out interval (the default is 7,200 seconds (2 hours)) reactivates itself and returns the exception to the user. A resumable statement can be suspended and resumed multiple times during execution.



Using Resumable Space Allocation

Resumable space allocation is possible only when statements are executed within a session that has resumable mode enabled. There are two means of enabling and disabling resumable space allocation:

- Issue the ALTER SESSION ENABLE RESUMABLE command.
- Set the RESUMABLE_TIMEOUT initialization parameter to a nonzero value with an ALTER SESSION or ALTER SYSTEM statement.

When enabling resumable mode for a session or the database, you can specify a time-out period, after which a suspended statement errors out if no intervention has taken place. The RESUMABLE_TIMEOUT initialization parameter indicates the number of seconds before a time-out occurs. You can also specify the time-out period with the following command:

```
ALTER SESSION ENABLE RESUMABLE
TIMEOUT 3600;
```

The value of TIMEOUT remains in effect until it is changed by another ALTER SESSION ENABLE RESUMABLE statement, it is changed by another means, or the session ends. The default time-out interval when using the ENABLE RESUMABLE TIMEOUT clause to enable resumable mode is 7,200 seconds, or 2 hours.

Using Resumable Space AllocationFull Notes Page

Using Resumable Space Allocation (continued)

You can also give a name to resumable statements. For example:

```
ALTER SESSION ENABLE RESUMABLE TIMEOUT 3600
NAME 'multitab insert';
```

The name of the statement is used to identify the resumable statement in the DBA_RESUMABLE and USER_RESUMABLE views.

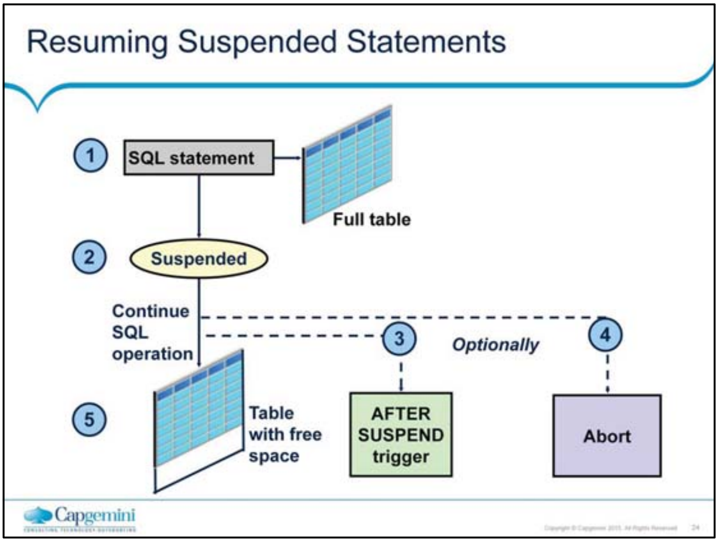
For example:

```
SELECT name, sql_text FROM user_resumable;

NAME            SQL_TEXT
--------------- ---------------------------------------------
multitab insert INSERT INTO oldsales SELECT * FROM
sh.sales;
```

To automatically configure resumable statement settings for individual sessions, you can create and register a database-level LOGON trigger that alters a user’s session. The trigger issues commands to enable resumable statements for the session, specifies a time-out period, and associates a name with the resumable statements issued by the session.

Because suspended statements can hold up some system resources, users must be granted the RESUMABLE system privilege before they are allowed to enable resumable space allocation and execute resumable statements.



Resuming Suspending Statements

Example:

1. An INSERT statement encounters an error saying the table is full.
2. The INSERT statement is suspended, and no error is passed to client.
3. Optionally, an AFTER SUSPEND trigger is executed.
4. Optionally, the SQLERRROR exception is activated to abort the statement.
5. If the statement is not aborted and free space is successfully added to the table, the INSERT statement resumes execution.

Detecting a Suspended Statement

When a resumable statement is suspended, the error is not raised to the client. In order for corrective action to be taken, the Oracle database provides alternative methods for notifying users of the error and for providing information about the circumstances.

Resuming Suspended Statements Full Notes Page

Resuming Suspended Statements (continued)

Possible Actions During Suspension

When a resumable statement encounters a correctable error, the system internally generates the AFTER SUSPEND system event. Users can register triggers for this event at both the database and schema level. If a user registers a trigger to handle this system event, the trigger is executed after a SQL statement has been suspended. SQL statements executed within an AFTER SUSPEND trigger are always nonresumable and are always autonomous. Transactions started within the trigger use the SYSTEM rollback segment. These conditions are imposed to overcome deadlocks and reduce the chance of the trigger experiencing the same error condition as the statement.

Within the trigger code, you can use the USER_RESUMABLE or DBA_RESUMABLE views, or the DBMS_RESUMABLE.SPACE_ERROR_INFO function to get information about the resumable statements.

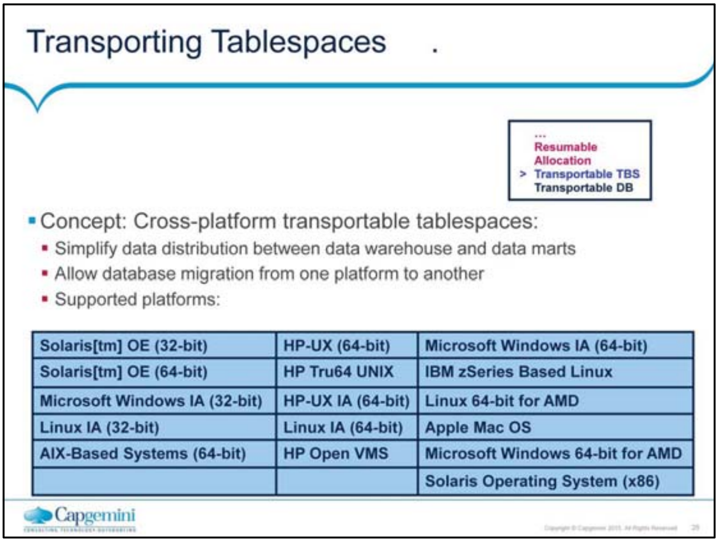
When a resumable statement is suspended:

- The session invoking the statement is put into a wait state. A row is inserted into V$SESSION_WAIT for the session with the EVENT column containing “statement suspended, wait error to be cleared”.
- An operation-suspended alert is issued on the object that needs additional resources for the suspended statement to complete.

Ending a Suspended Statement

When the error condition is resolved (for example, as a result of DBA intervention or perhaps sort space released by other queries), the suspended statement automatically resumes execution and the “resumable session suspended” alert is cleared.

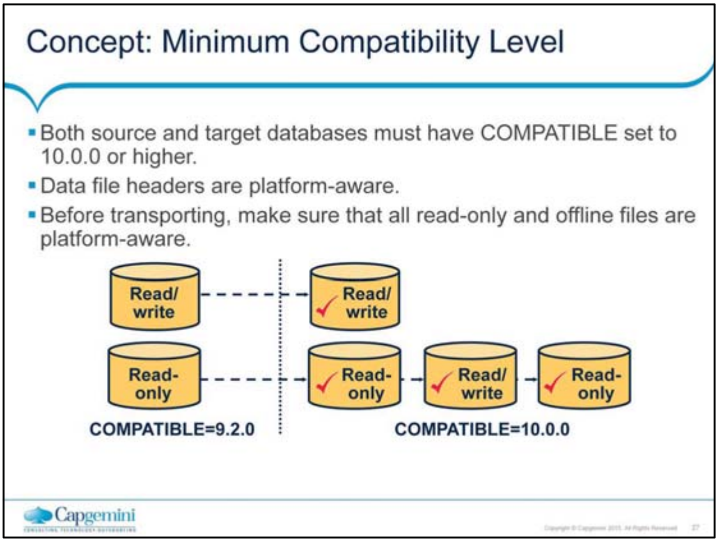
A suspended statement can be forced to activate the SERVERERROR exception by using the DBMS_RESUMABLE.ABORT() procedure. This procedure can be called by a DBA, or by the user who issued the statement. If the suspension time-out interval associated with the resumable statement is reached, the statement aborts automatically and an error is returned to the user.



Transporting Tablespaces

You can use the transportable tablespace feature to move data across platform boundaries. This simplifies the distribution of data from a data warehouse environment to data marts, which often run on smaller platforms. It also allows a database to be migrated from one platform to another by rebuilding the dictionary and transporting the user tablespaces. To be able to transport data files from one platform to another, you must ensure that both the source system and the target system are running on one of the supported platforms (see slide).

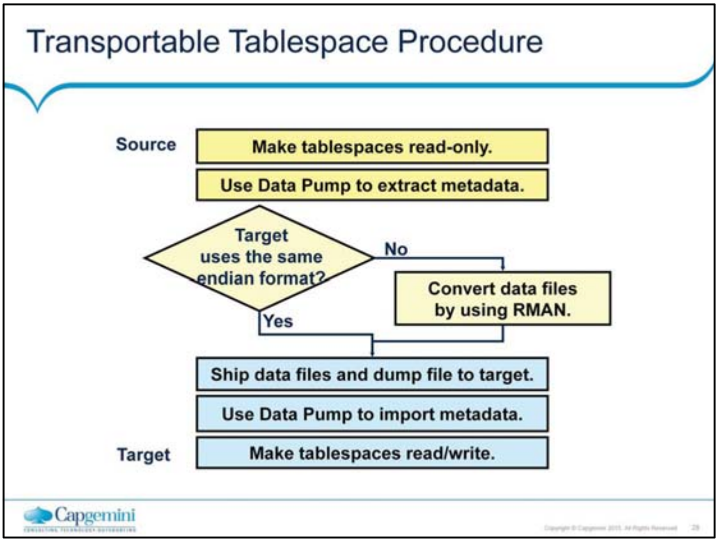
Note: The cross-platform transportable tablespace feature requires both platforms to be using the same character sets.



Concept: Minimum Compatibility Level

Both source and target databases need to advance their database COMPATIBLE initialization parameter to 10.0.0 or greater before they can use the cross-platform transportable tablespace feature.

When data files are first opened in Oracle Database 11g with COMPATIBLE set to 10.0.0 (or greater), the files are made platform-aware. This is represented by the check marks in the diagram. Each file identifies the platform that it belongs to. These files have identical on-disk formats for file header blocks that are used for file identification and verification. Read-only and offline files get the compatibility advanced only after they are made read/write or are brought online. This implies that tablespaces that are read-only in databases before Oracle Database 11g must be made read/write at least once before they can use the cross-platform transportable feature.



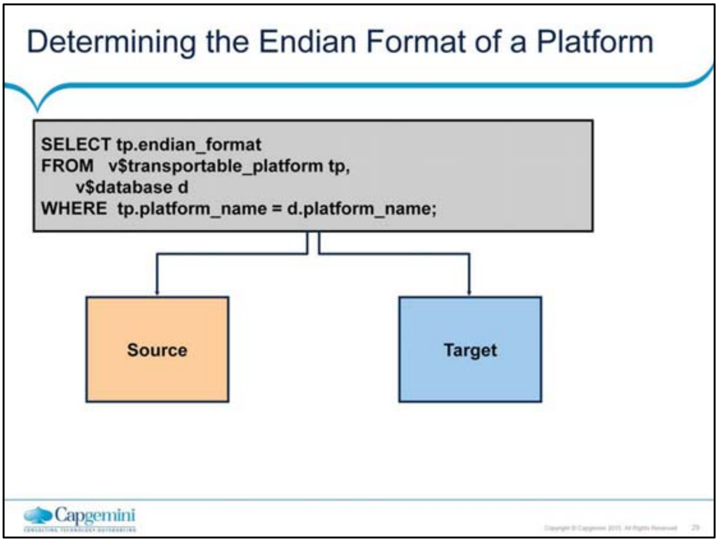
Transportable Tablespace Procedure

To transport a tablespace from one platform to another (source to target), data files belonging to the tablespace set must be converted to a format that can be understood by the target or destination database. Although with Oracle Database 11g, disk structures conform to a common format, it is possible for the source and target platforms to use different endian formats (byte ordering). When going to a different endian platform, you must use the CONVERT command of the RMAN utility to convert the byte ordering. This operation can be performed on either the source or the target platforms. For platforms that have the same endian format, no conversion is needed.

The slide graphic depicts the possible steps to transport tablespaces from a source platform to a target platform. However, it is possible to perform the conversion after shipping the files to the target platform. The last two steps must be executed on the target platform.

Basically, the procedure is the same as when using previous releases of the Oracle database except when both platforms use different endian formats. It is assumed that both platforms are cross-transportable compliant.

Note: Byte ordering can affect the results when data is written and read. For example, the 2-byte integer value 1 is written as 0x0001 on a big-endian system (such as Sun SPARC Solaris) and as 0x0100 on a little-endian system (such as an Intel-compatible PC).



Determining the Endian Format of a Platform

You can query V$TRANSPORTABLE_PLATFORM to determine whether the endian ordering is the same on both platforms. V$DATABASE has two columns that can be used to determine your own platform name and platform identifier.

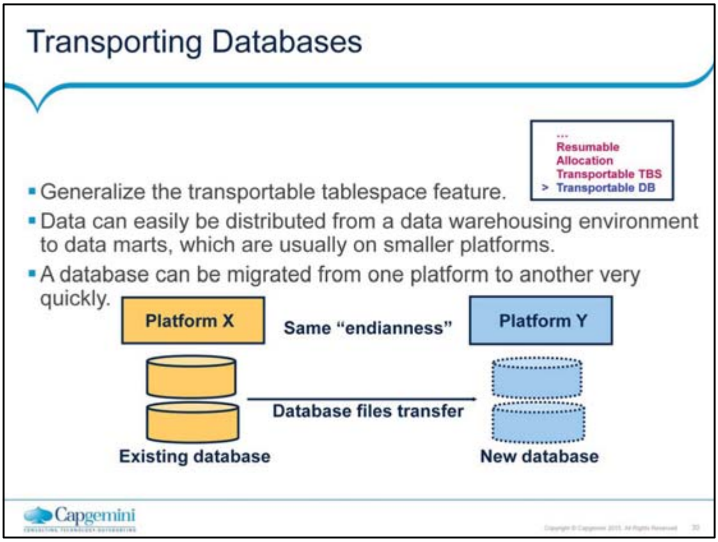
Use the query in the slide on both platforms, and then compare the results.

On a Sun SPARC Solaris system, the SELECT statement produces the following output:

```
ENDIAN_FORMAT
-------------
Big
```

On a Microsoft Windows Intel-based platform, the SELECT statement produces the following output:

```
ENDIAN_FORMAT
-------------
Little
```



Transporting Databases

With the transportable tablespace feature, moving data across different platforms becomes much faster. However, metadata still needs to be unloaded, because the system tablespace cannot be transported.

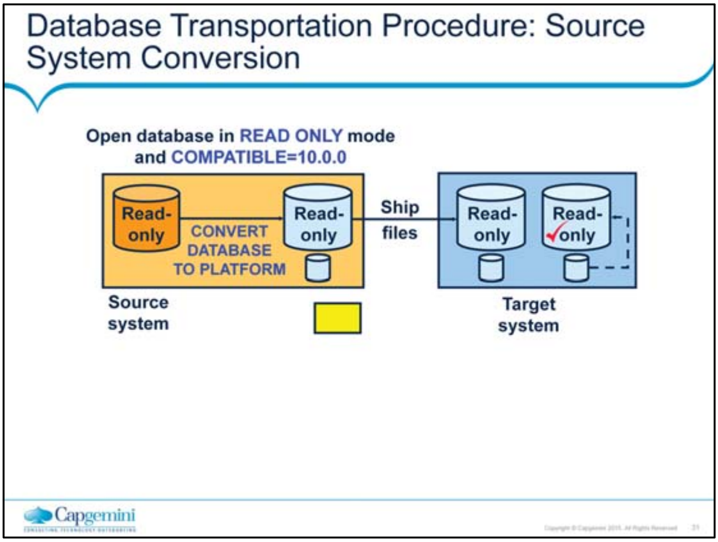
The purpose of the database transport feature is to provide a fast and easy way to transport a database across different platforms with the same endian format. However, the source platform and the target platform can have different disk alignments. For example, HP-UX and Solaris both have big endian, but the disk alignment is eight on HP-UX and four on Solaris.

To transport databases from one platform to another, you must ensure that both the source system and the target system are running on one of the platforms that are listed in V$TRANSPORTABLE_PLATFORM and that both have the same endian format. For example, you can transport a database running on Linux IA (32-bit) to one of the Windows platforms.

If one or both of the databases uses Automatic Storage Management (ASM), you may need to use the DBMS_FILE_TRANSFER package to ftp the files.

Unlike transportable tablespace, where there is a target database to plug data into, this feature creates a new database on the target platform. The newly created database contains the same data as the source database. Except for things such as database name, instance name, and location of files, the new database also has the same settings as the source database.

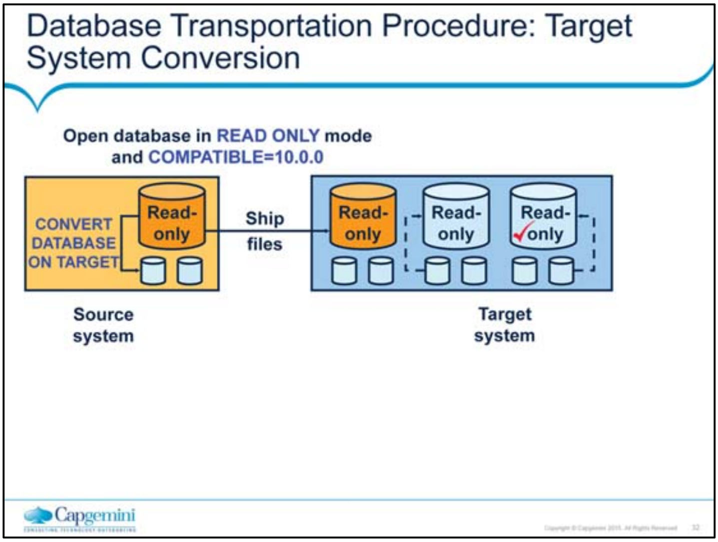
Note: Transporting database is faster than using Data Pump to move data.



Database Transportation Procedure: Source System Conversion

Before you can transport your database, you must open it in READ ONLY mode. Then use RMAN to convert the necessary data files of the database. When you do the conversion on the source platform, the new RMAN command CONVERT DATABASE generates a script containing the correct CREATE CONTROLFILE RESETLOGS command that is used on the target system to create the new database. The CONVERT DATABASE command then converts all identified data files so that they can be used on the target system. You then ship the converted data files and the generated script to the target platform. By executing the generated script on the target platform, you create a new copy of your database.

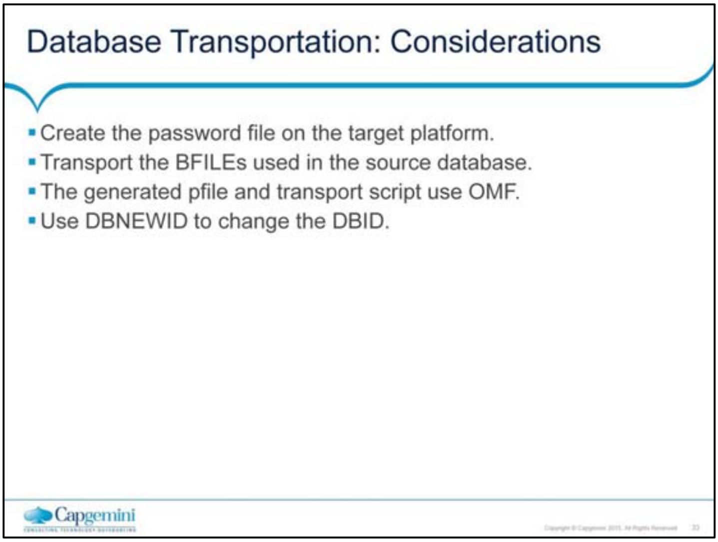
Note: The source database must be running with the COMPATIBLE initialization parameter set to 10.0.0 or higher. All identified tablespaces must have been READ WRITE at least once since COMPATIBLE was set to 10.0.0 or higher.



Database Transportation Procedure: Target System Conversion

Before you can transport your database, you must open it in READ ONLY mode. Then use RMAN to convert the necessary data files of the database. When you do the conversion on the target platform, the CONVERT DATABASE command (which is executed on the source system) generates only two scripts used on the target system to convert the data files, and to re-create the control files for the new database. Then, you ship the identified data files and both scripts to the target platform. After this is done, execute both scripts in the right order. The first one uses the existing CONVERT DATAFILE RMAN command to do the conversion, and the second issues the CREATE CONTROLFILE RESETLOGS SQL command with the converted data files to create the new database.

Note: The source database must be running with the COMPATIBLE initialization parameter set to 10.0.0 or higher. All identified tablespaces must have been READ WRITE at least once since COMPATIBLE was set to 10.0.0 or higher.



Database Transportation: Considerations

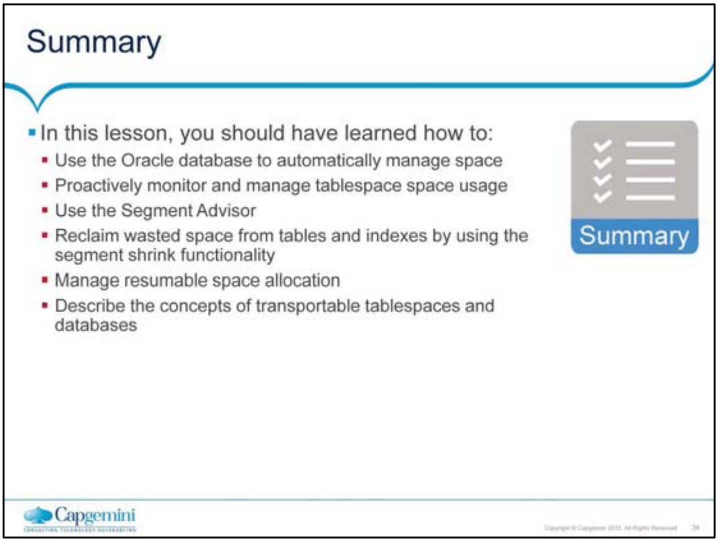
Redo logs, control files, and tempfiles are not transported. They are re-created for the new database on the target platform. As a result, the new database on the target platform must be opened with the RESETLOGS option.

If a password file is used, it is not transported and you need to create it on the target platform. This is because the types of file names allowed for the password file are OS specific. However, the output of the CONVERT DATABASE command lists all the usernames and their system privileges, and advises to re-create the password file and add entries for these users on the target platform.

The CONVERT DATABASE command lists all the directory objects and objects that use BFILE data types or external tables in the source database. You may need to update these objects with new directory and file names. If BFILEs are used in the database, you have to transport the BFILEs.

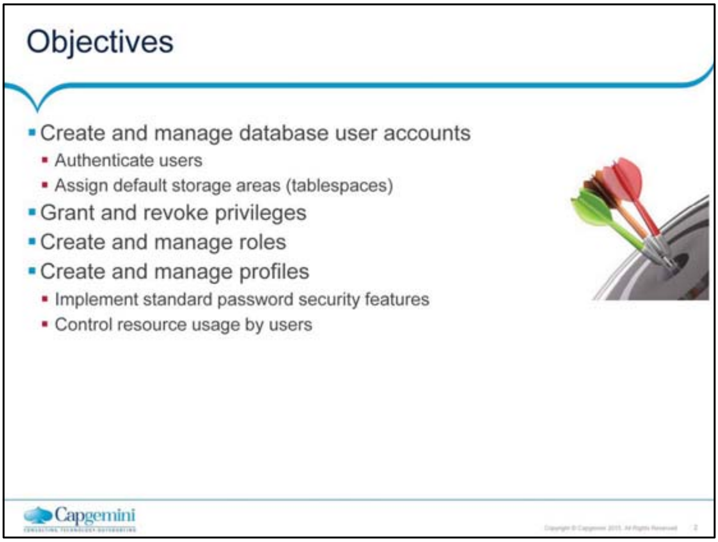
The generated pfile and transport script use Oracle Managed Files (OMF) for database files. If you do not want to use OMF, you must modify the pfile and transport script.

The transported database has the same DBID as the source database. You can use the DBNEWID utility to change the DBID. In the transport script as well as the output of the CONVERT DATABASE command, you are prompted to use the DBNEWID utility to change the database ID.


Summary
In this lesson, you should have learned how to:
Use the Oracle database to automatically manage space
Proactively monitor and manage tablespace space usage
Use the Segment Advisor
Reclaim wasted space from tables and indexes by using the segment shrink functionality
Manage resumable space allocation
Describe the concepts of transportable tablespaces and databases
Summary
Capgemini

Oracle 11g DBA Fundamentals Overview

Lesson 11: Managing Users and Securing the Database



Objectives

The following terms relate to administering database users and assist you in understanding the objectives:

A database user account is a means to organize the ownership of and access to database objects.

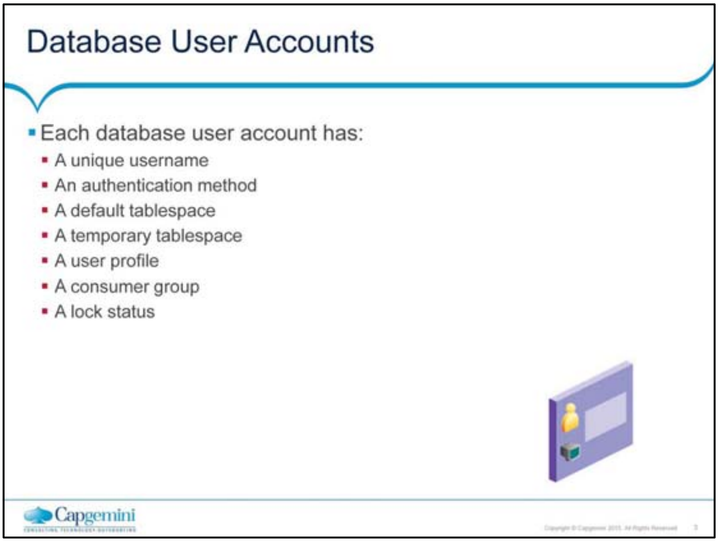
A password is an authentication by the Oracle database.

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.

Quota is a space allowance in a given tablespace. This is one of the ways by which you can control resource usage by users.



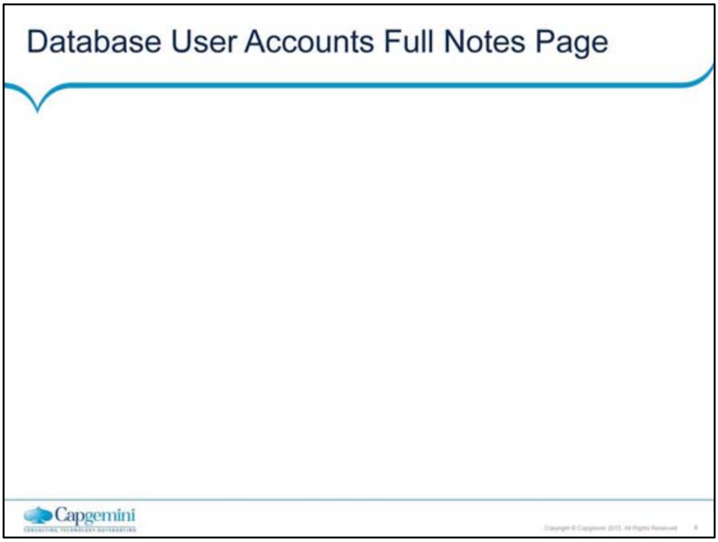
Database User Accounts

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 his or her own database account. This is Oracle's best practice recommendation to avoid potential security holes and provide meaningful data for certain audit activities. However, in rare cases, users share a common database account. In this case, 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.

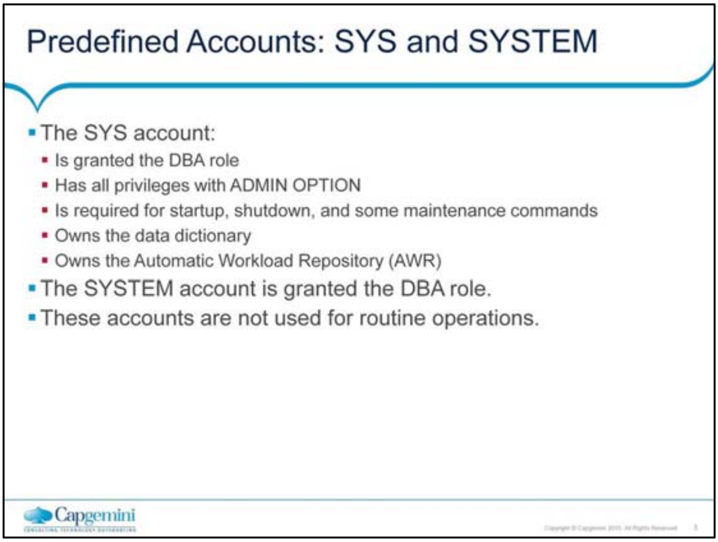
An authentication method: The most common authentication method is a password, but Oracle Database 11g supports several other authentication methods, including biometric, certificate, and token authentication.

A default tablespace: This is a place where the user creates objects if he or she 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 within that tablespace in which to create objects. Both these are granted separately.



Database User Accounts (continued)

A temporary tablespace: This is a place where the user can create temporary objects, such as sorts and temporary tables.
A user profile: This is a set of resource and password restrictions assigned to the user.
A consumer group: This is used by the resource manager.
A lock status: Users can access only “unlocked” accounts.



Predefined Accounts: SYS and SYSTEM

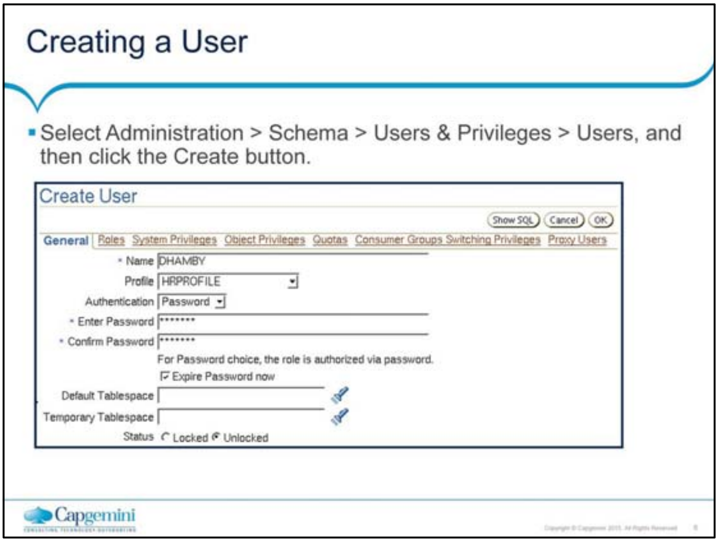
The SYS and SYSTEM accounts have the database administrator (DBA) role granted to them by default.

The SYS account in addition has all privileges with ADMIN OPTION and owns the data dictionary. To connect to the SYS account, you must use the AS SYSDBA clause. 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 or SYSOPER privilege, are allowed to start up and shut down the database instance.

The SYSTEM account is granted the DBA role by default, but not 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.

The SYS and SYSTEM accounts are required accounts in the database. They cannot be dropped.



Creating a User

In Enterprise Manager, you can manage the list of database users, who are allowed to access the current database, by using the Users page. You can use this page to create, delete, and modify the settings of a user.

To create a database user, perform the following steps:

1. In Enterprise Manager Database Control, select Administration > Schema > Users & Privileges > Users.
2. Click the Create button.

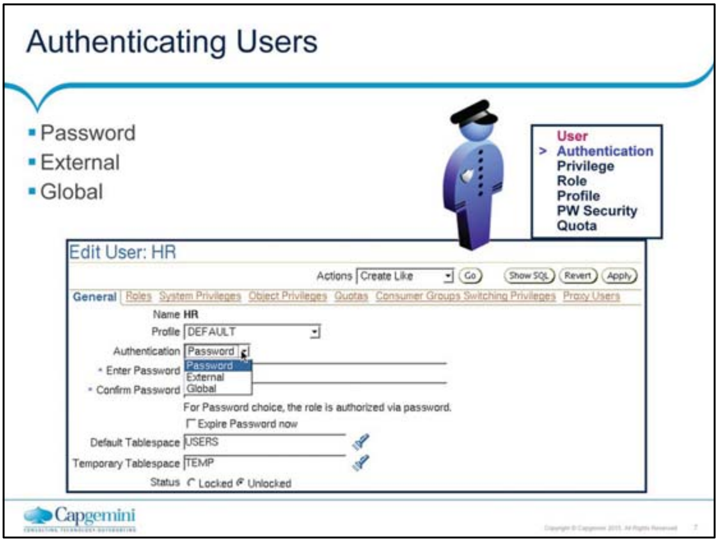
Provide the required information. Mandatory items, such as Name, are marked with a star.

The following pages give you more information about authentication.

Profiles are covered later in this lesson.

Assign a default tablespace and a temporary tablespace to each user. This allows you to control where their objects are created, if users do not specify a tablespace in the creation of an object.

If you do not choose a default tablespace, then the system-defined default permanent tablespace is used. Similarly for the temporary tablespace: if you do not specify one, then the system-defined temporary tablespace is used.



Authenticating Users

Authentication means verifying the identity of someone (a user, device, or other entity) who 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 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.

Passwords are always automatically and transparently encrypted during network (client/server and server/server) connections, by using a modified Data Encryption Standard (DES) algorithm, before sending them across the network.

Authenticating Users (continued)

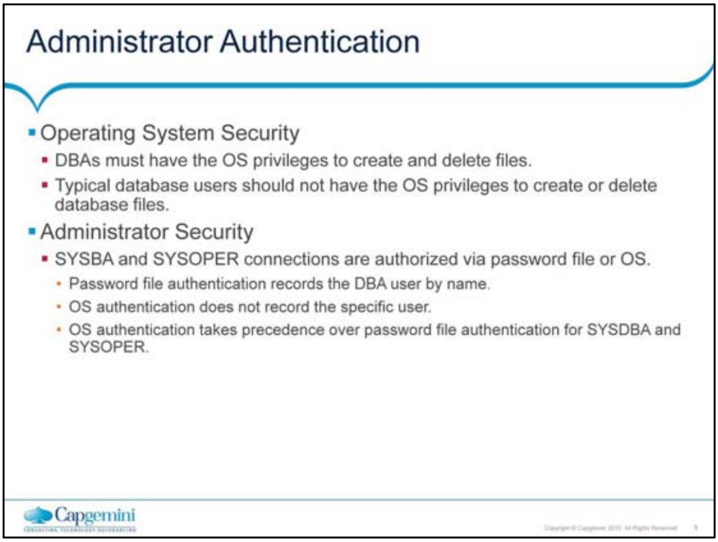
External: This is also referred to as authentication by the operating system. Users can connect to the Oracle database without specifying a username or password. With external authentication, your database relies on the underlying operating system or network authentication service to restrict access 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 do so, 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, assume 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, then 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 your operating system–specific Oracle documentation for more information about this initialization parameter.

Global: Using the Oracle Advanced Security option, global authentication (which is a strong authentication) allows users to be identified through the use of biometrics, x509 certificates, token devices, and Oracle Internet Directory. For more information about advanced authentication methods, refer to the *Oracle Enterprise Identity Management* course.



Administrator Authentication

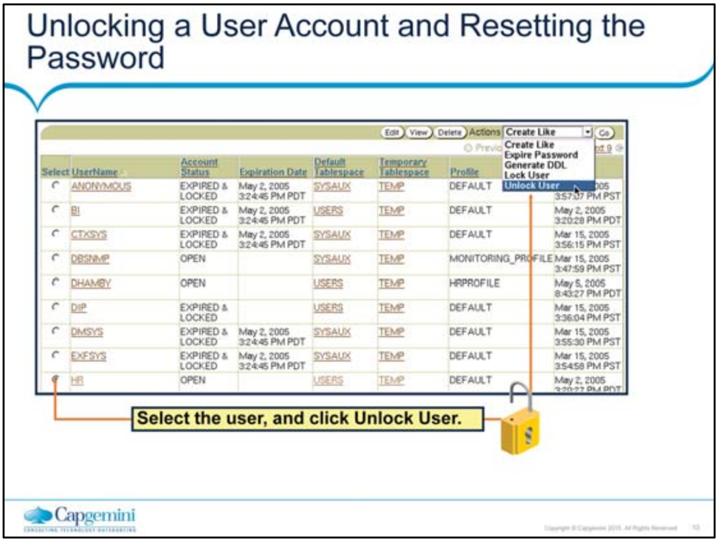
Operating System Security: In UNIX and Linux, by default, DBAs belong to the install OS group, which has the required privileges to create and delete database files.

Administrator Security: SYSBA and SYSOPER connections are authorized only after verification with the password file or with the operating system privileges and permissions. If operating system authentication is used, then the database does not use the supplied username and password.

Operating system 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 is supplied.

However, if authentication succeeds by means of the password file, then the connection is logged with the username. If authentication succeeds by means of the operating system, then it is a CONNECT / connection that does not record the specific user.

Note: OS authentication takes precedence over password file authentication. Specifically, 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.

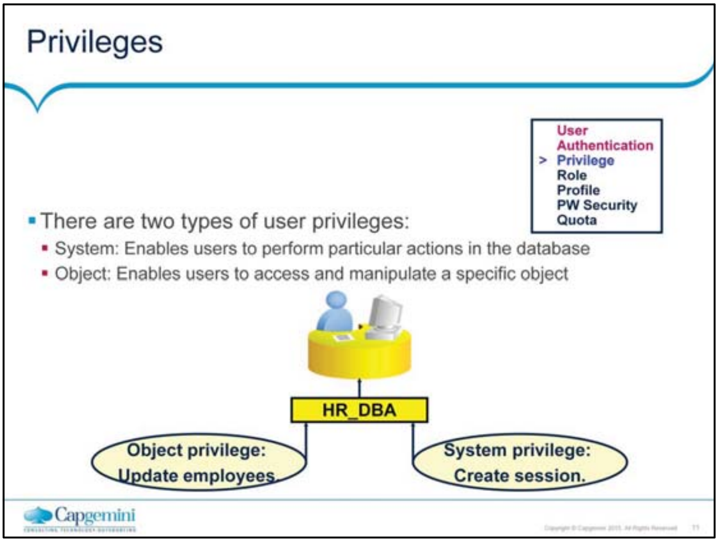


Unlocking a User Account and Resetting the Password

During installation and database creation, you can unlock and reset many of the Oracle-supplied database user accounts. If you have not chosen to unlock the user accounts at that time, you can unlock the users and reset the passwords by selecting the user on the Users page and clicking Unlock User.

Alternatively, if you are on the Edit Users page, perform the following steps:

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.

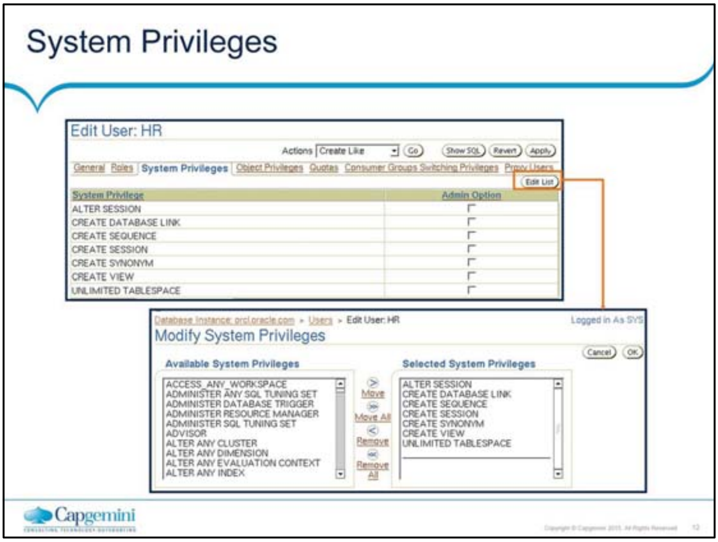


Privileges

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 users can or cannot do within 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 explicitly gives permission to administer the privilege. There are more than a hundred 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

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, the CREATE TABLE privilege allows you to create a table but only within your own schema. The SELECT ANY TABLE privilege allows you to select from tables owned by other users.

Selecting the Admin Option check box enables you to administer the privilege and grant the system privilege to other users.

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.

DROP ANY *object*: The DROP ANY privilege allows you to delete objects that other schema users own.

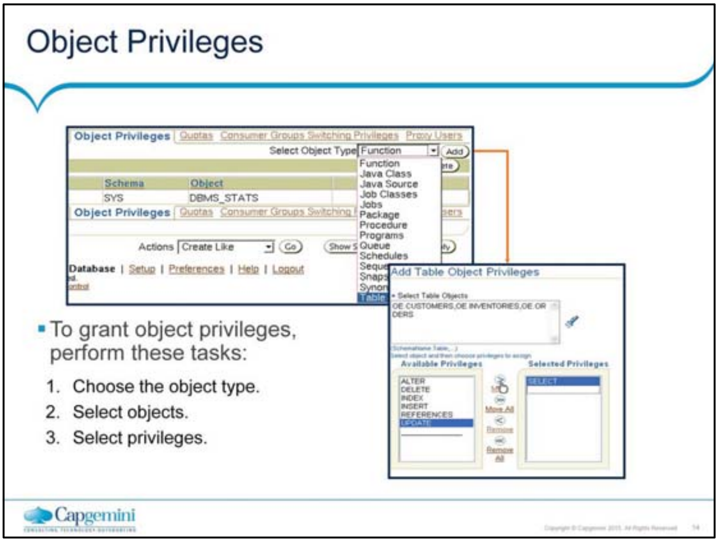
CREATE, MANAGE, DROP, and ALTER TABLESPACE: These privileges allow for tablespace administration including creating, dropping, and changing their attributes.

CREATE ANY DIRECTORY: The Oracle database allows developers to call external code (for example, a C library) from within PL/SQL. 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, such as renaming a data file or flushing the buffer cache.

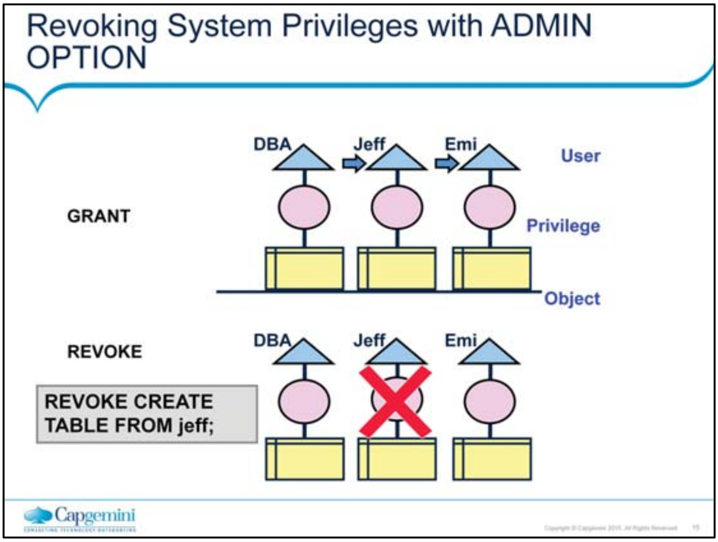


Object Privileges

To grant object privileges, click the Object Privileges tab on the Edit User page. Select the type of object you want to grant privileges on, and click the Add button. Choose the objects you want to grant privileges on by either entering <username.object name> or selecting them from the list.

Next, select the appropriate privileges from the Available Privileges list, and click the Move button. When you have finished selecting privileges, click OK.

Back on the Edit User page, select the Grant check box if this user is allowed to grant other users the same access.



Revoking System Privileges

System privileges, which 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.

Read through the following steps that illustrate this:

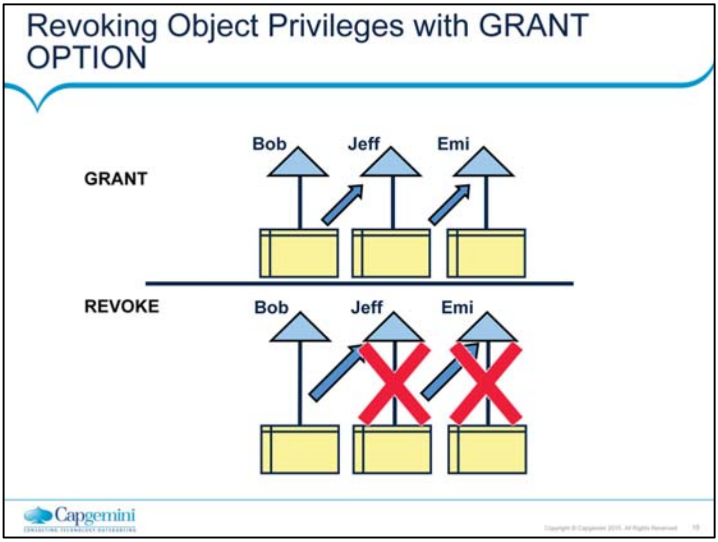
Scenario

1. The DBA grants the CREATE TABLE system privilege to Jeff with ADMIN OPTION.
2. Jeff creates a table.
3. Jeff grants the CREATE TABLE system privilege to Emi.
4. Emi creates a table.
5. The DBA revokes the CREATE TABLE system privilege from Jeff.

The result

Jeff's table still exists, but no new tables can be created.

Emi's table still exists, and she still has the CREATE TABLE system privilege.



Revoking Object Privileges

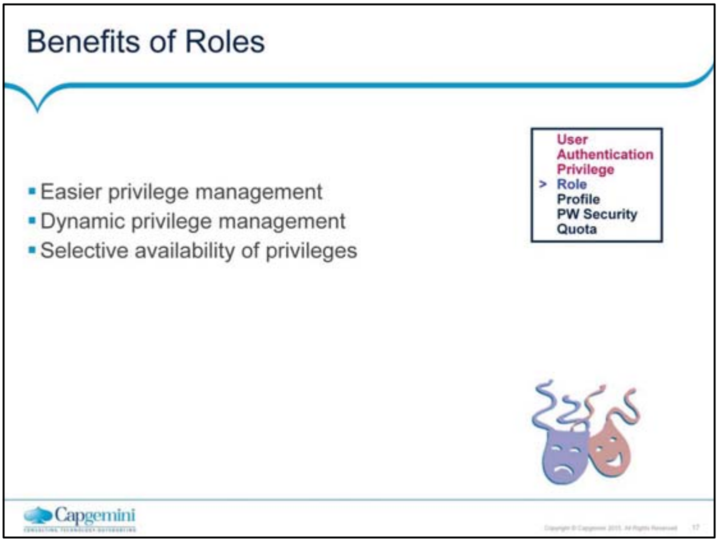
Cascading effects can be observed when revoking a system privilege that is related to a data manipulation language (DML) operation. For example, if the SELECT ANY TABLE privilege is granted to a user, and that user has created procedures that use the table, all procedures that are contained in the user's schema must be recompiled before they can be used again.

Revoking object privileges also cascades when given WITH GRANT OPTION.

Read through the following steps that illustrate this:

Scenario

1. Jeff is granted the SELECT object privilege on EMPLOYEES with GRANT OPTION.
2. Jeff grants the SELECT privilege on EMPLOYEES to Emi.
3. Later, the SELECT privilege is revoked from Jeff. This revoke is cascaded to Emi as well.

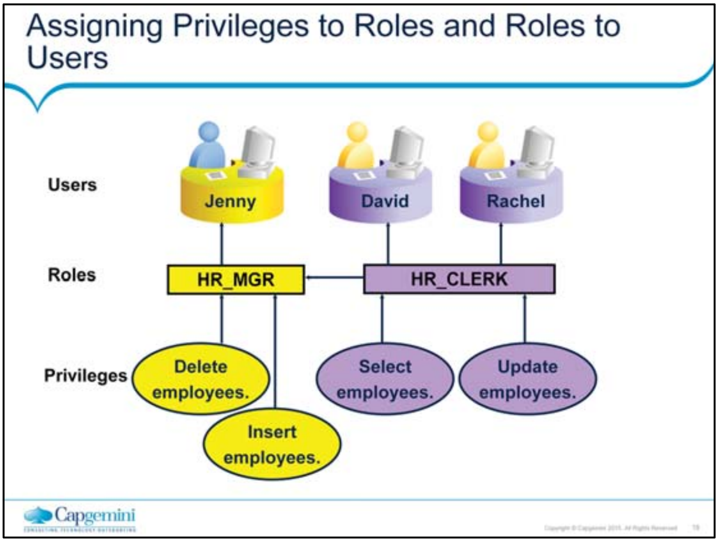


Benefits of Roles

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 the 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. Enabling a role can also be used to verify that a user has been granted that role.



Assigning Privileges to Roles and Roles to Users

In most systems, it is too time-consuming to grant necessary privileges to each user individually, and there is too great a chance of error. 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 can be 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 HR_CLERK role is granted the SELECT and UPDATE privileges on the employees table. The HR_MGR role is granted the DELETE and INSERT privileges on the employees table and the HR_CLERK role. The manager is granted the HR_MGR role and can now select, delete, insert, and update the employees table.

Predefined Roles

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, but HS_ADMIN_ROLE and over 1,700 object privileges on the data dictionary

Capgemini

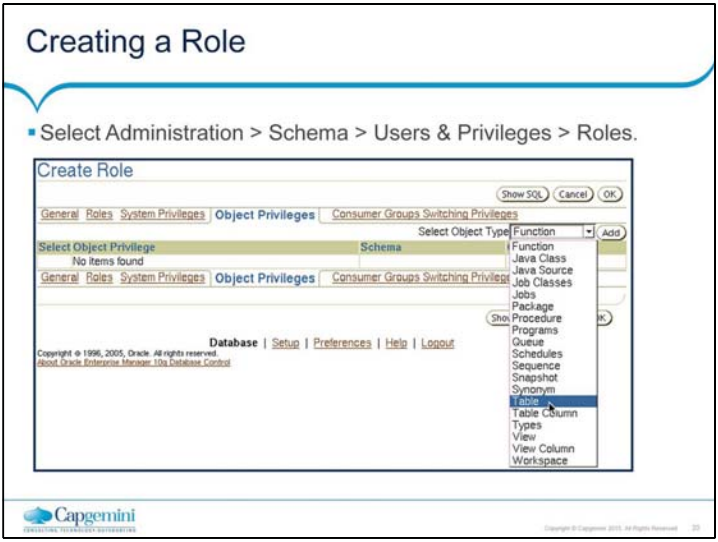
Predefined Roles

There are several roles that are defined automatically for Oracle databases when you run database creation scripts. CONNECT is granted automatically to any user created with Enterprise Manager. In earlier versions of the database (before Oracle Database 11g Release 2), the CONNECT role included more privileges, such as CREATE TABLE and CREATE DATABASE LINK, which have been removed for security reasons.

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.

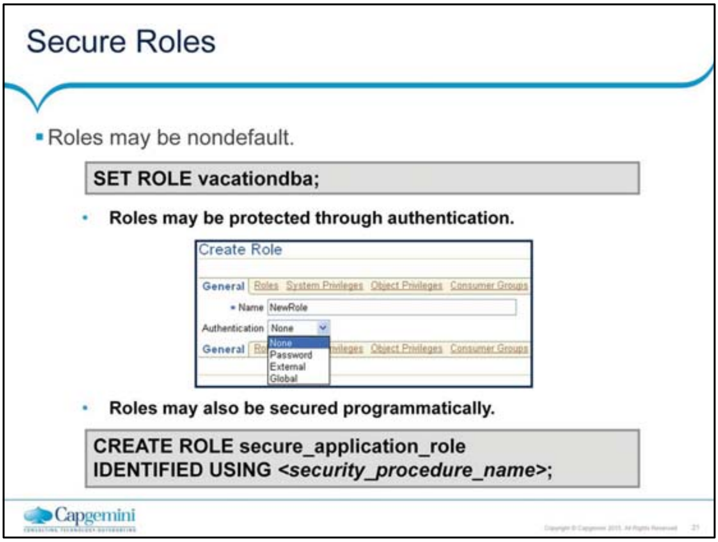


Creating a Role

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, select Administration > Schema > Users & Privileges > Roles.
2. Click the Create button.



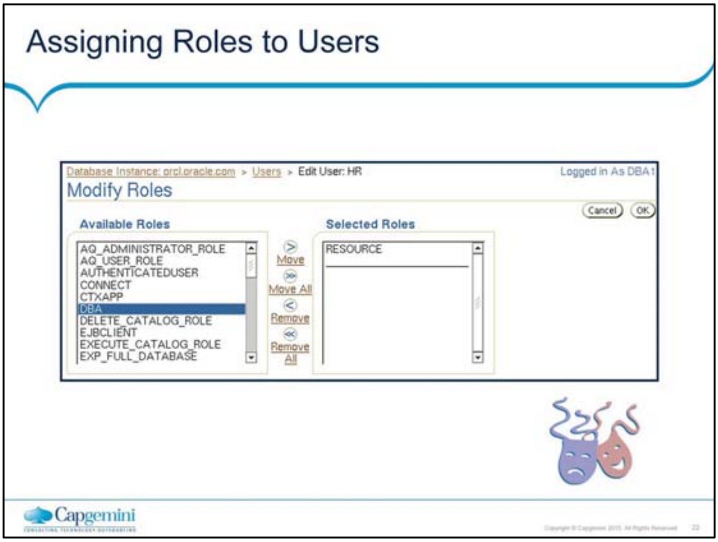
Secure Roles

Roles are usually enabled by default, which means that if a role is granted to a user, that user can exercise the privileges given to that role. 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, which program the user is running, time of day, or other elements needed to properly secure a group of permissions.



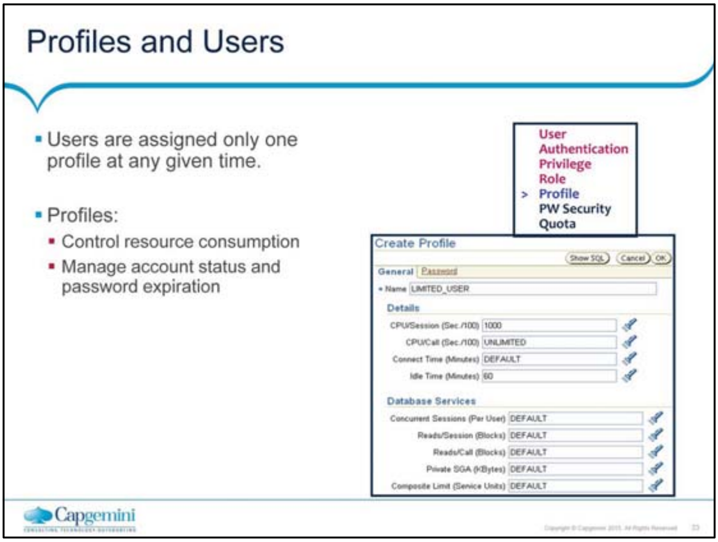
Assigning Roles to Users

A role is a set of privileges that can be granted to users or to other roles. You can use roles to administer database privileges. You can add privileges to a role and then 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 granted to that role and all privileges of other roles 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, perform the following steps:

1. In Enterprise Manager Database Control, choose Administration > Schema > Users & Privileges > Users.
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.



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), be unlimited (as in CPU/Call), or 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 limitations are ignored.

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.), then 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, and the user gets 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:

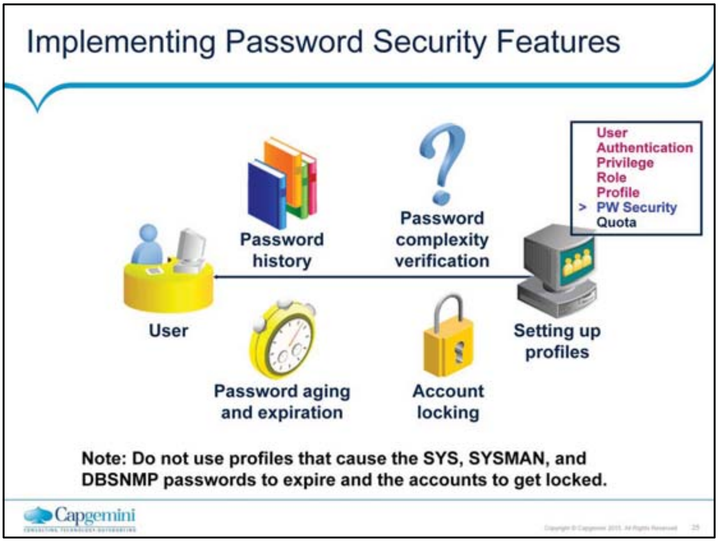
- Connect Time: Indicates for how many minutes a user can be connected before being automatically logged off
- Idle Time: Indicates for how many minutes a user's session can remain idle before being automatically logged off. Idle time is calculated for the server process only. It does not take into account 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 within 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 discussed in the lesson titled "Configuring the Oracle Network Environment").

Disk I/O: This limits the amount of data a user can read either at the per-session 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 input/output (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, select Administration > Schema > Users & Privileges > Profiles, and 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

Oracle password management is implemented with user profiles. Profiles can provide many standard security features including the following:

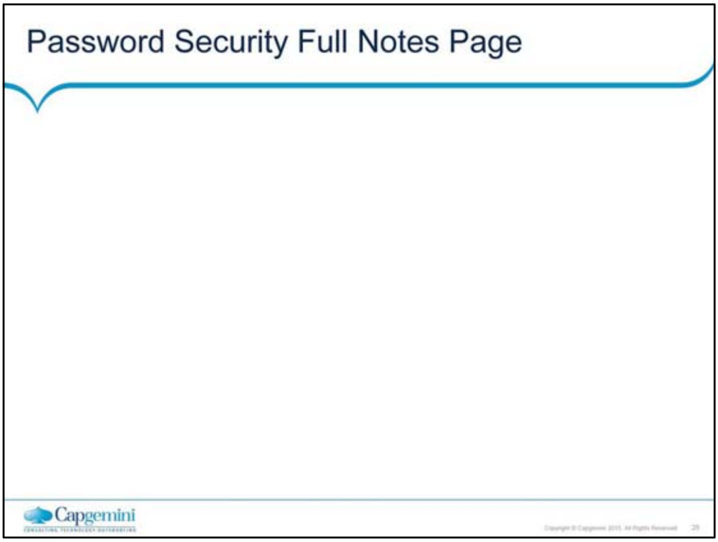
Account locking: Enables automatic locking of accounts for a set duration when users fail to log in to the system in the specified number of attempts.

- The FAILED_LOGIN_ATTEMPTS parameter specifies the number of failed login attempts before the lockout of the account.
- The PASSWORD_LOCK_TIME parameter 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

- The PASSWORD_LIFE_TIME parameter determines the lifetime of the password in days, after which the password expires.
- The PASSWORD_GRACE_TIME parameter 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)

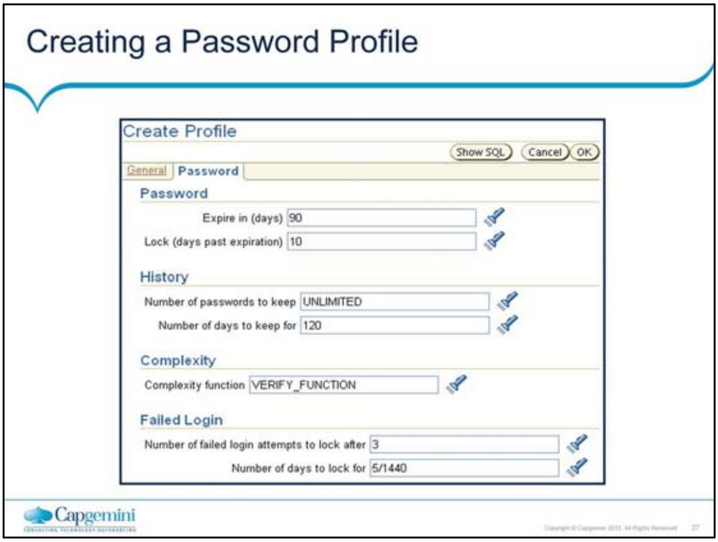
Password history: Checks the new password to ensure that the password is not reused for a specified amount of time or a specified number of password changes. These checks can be implemented by using one of the following:

- 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

These two parameters are mutually exclusive, and so when one parameter is set to a value other than UNLIMITED (or DEFAULT, if the DEFAULT profile has the value set to UNLIMITED), the other parameter must be set to UNLIMITED.

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).



Creating a Password Profile

To create a password profile, select Administration > Schema > Users & Privileges > Profiles, and click the Create button.

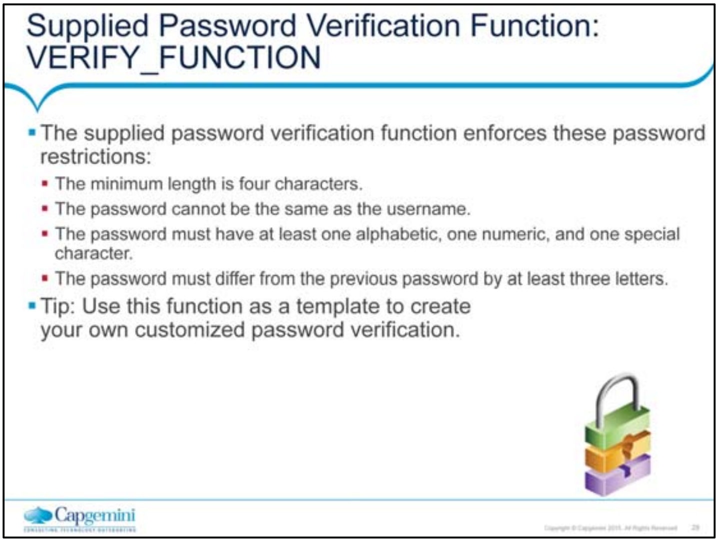
Common values for each of the settings can be chosen from a list of values (Click the flashlight icon to browse.), or you can enter a custom value.

All time periods are expressed in days, but can be expressed as fractions also. There are 1,440 minutes in a day, and so 5/1440 is five minutes.

Enterprise Manager can also be used to edit existing password profiles.

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), then all users who have that profile are automatically assigned the DEFAULT profile.



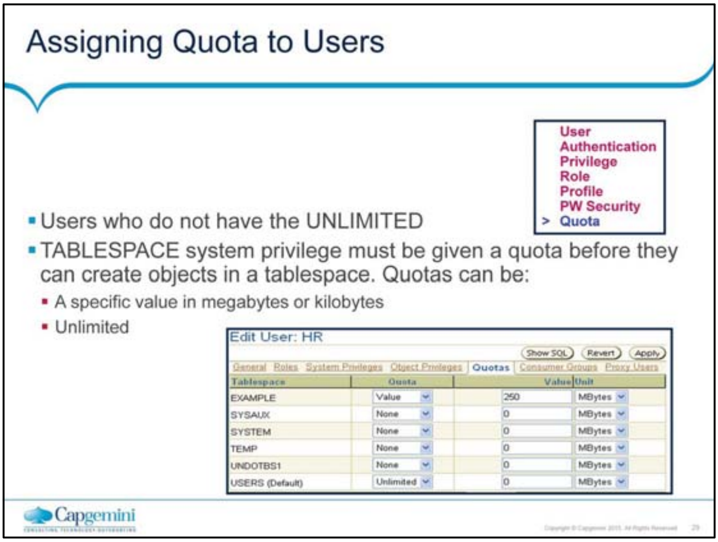
Supplied Password Verification Function: VERIFY_FUNCTION

The Oracle server provides a password complexity verification function named VERIFY_FUNCTION. This function is created with the <oracle_home>/rdbms/admin/utlpwdmg.sql script. 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 60
PASSWORD_GRACE_TIME 10
PASSWORD_REUSE_TIME 1800
PASSWORD_REUSE_MAX UNLIMITED
FAILED_LOGIN_ATTEMPTS 3
PASSWORD_LOCK_TIME 1/1440
PASSWORD_VERIFY_FUNCTION
verify_function;
```

Remember that when users are created, they are assigned the DEFAULT profile, unless another profile is specified.



Assigning Quota to Users

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 user quota on a tablespace.

Unlimited: This allows the user to use as much space as is available in the tablespace.

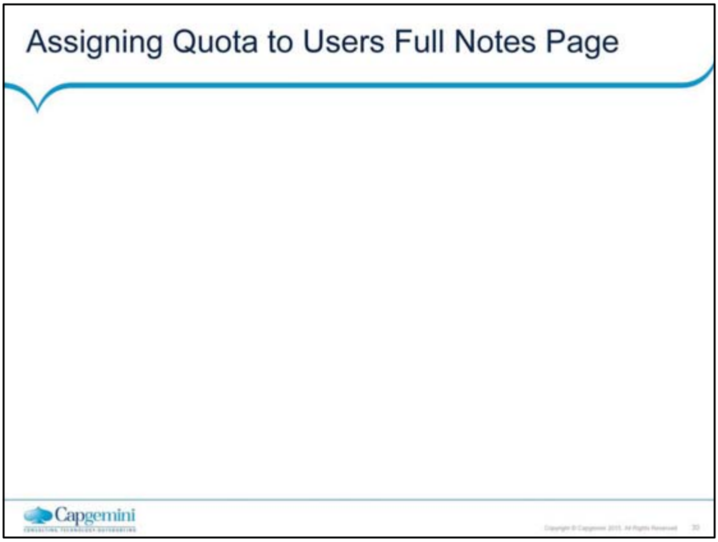
Value: This is a number of kilobytes or megabytes that the user can use. This does not guarantee that the space is set aside for the user. This value can be larger or smaller than the current space that is available in the tablespace.

UNLIMITED TABLESPACE system privilege: This 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 quota to users on the SYSTEM or SYSAUX tablespace. Typically, only the SYS and SYSTEM users must be able to create objects in the SYSTEM or SYSAUX tablespace.

You do not need quota on an assigned temporary tablespace or any undo tablespaces.



Assigning Quota to Users (continued)

When does the Oracle instance use quota?

Quotas are used when a user creates or extends a segment.

Which activities do not 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 tablespace.

When is the quota replenished?

The quota is replenished when objects owned by the user are dropped with the PURGE clause or the objects in the recycle bin are automatically purged.

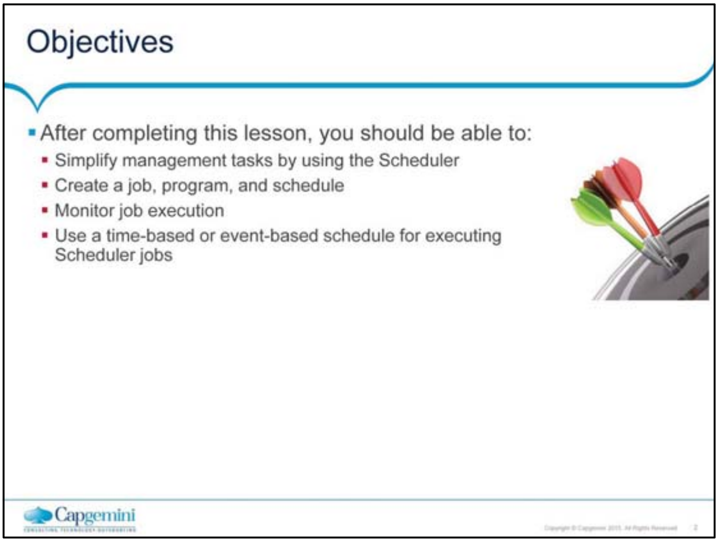
Summary

- 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

Summary

Oracle 11g DBA Fundamentals Overview

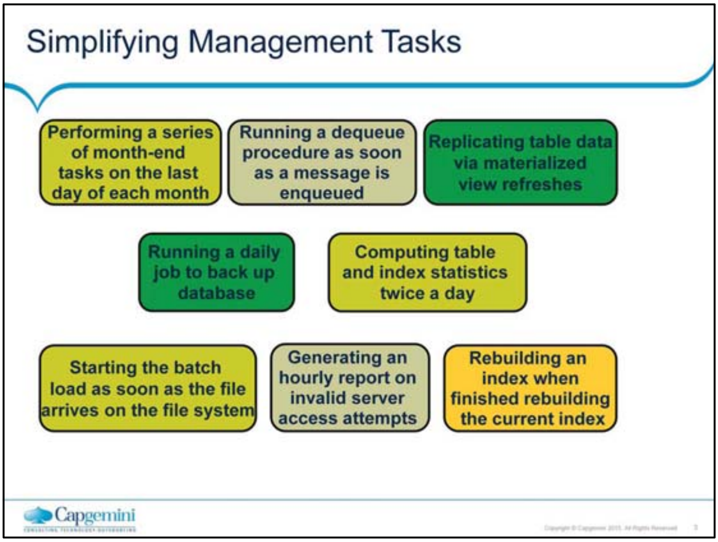
Lesson 12: Automating Tasks with the Scheduler



Objectives

For information about the various Scheduler components and their interaction, see the Oracle Database Administrator’s Guide.

For detailed information about the DBMS_SCHEDULER package, see the Oracle Database PL/SQL Packages and Types Reference.



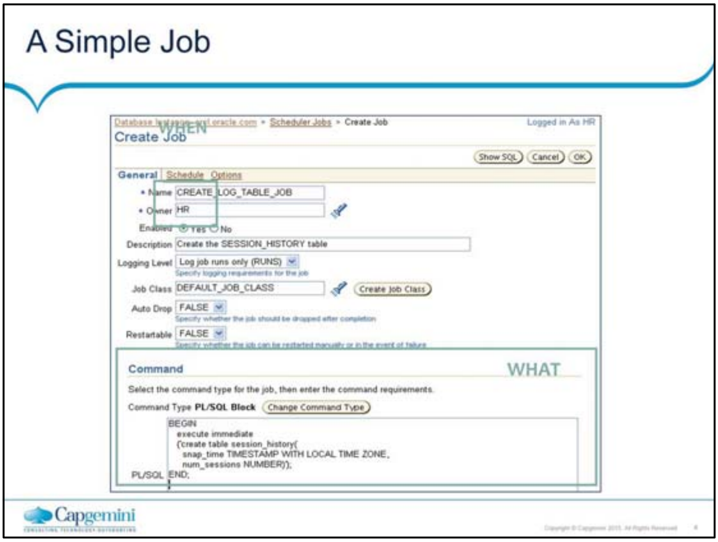
Simplifying Management Tasks

Many tasks in the Oracle environment need job-scheduling capabilities. Routine database maintenance and application logic require jobs to be scheduled and run periodically. Business-to-business (B2B) applications require scheduling for their business events. DBAs need to schedule regular maintenance jobs in specified time windows.

Oracle Database 11g provides advanced scheduling capabilities through the database Scheduler, which is a collection of functions and procedures in the DBMS_SCHEDULER package. The Scheduler can be invoked in any SQL environment, or through Enterprise Manager.

The Scheduler enables database administrators and application developers to control when and where various tasks take place in the database environment. These tasks can be time consuming and complicated; using the Scheduler, you can manage and plan these tasks.

Scheduler jobs can be started based on time or when a specified event occurs, and the Scheduler can raise events when a job's state changes (for example, from RUNNING to COMPLETE). You can also use a named series of programs that are linked together for a combined objective.

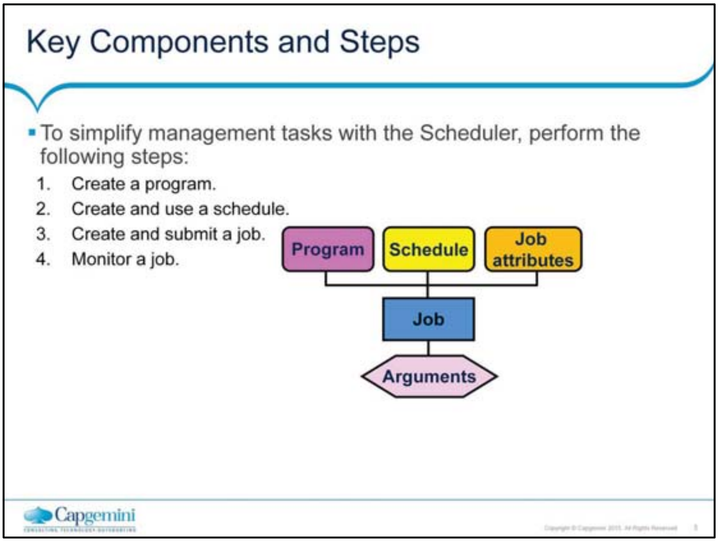


A Simple Job

A job has two key components: action, “what” needs to be done and schedule, “when” action occurs. The “what” is expressed in the Command region of the screenshot shown in the slide and the job_type and job_action parameters.

```
BEGIN
sys.dbms_scheduler.create_job(
job_name => '"HR"."CREATE_LOG_TABLE_JOB"',
job_type => 'PLSQL_BLOCK', job_action => 'begin
   execute immediate (''create table session_history(
   snap_time TIMESTAMP WITH LOCAL TIME ZONE,
   num_sessions NUMBER)'');  end;',
start_date => systimestamp at time zone 'America/New_York',
job_class => 'DEFAULT_JOB_CLASS',
comments => 'Create the SESSION_HISTORY table',
auto_drop => FALSE, enabled => TRUE);
END;
```

A job is also defined by “when” the desired action needs to take place. The “when” is expressed in a “schedule,” which can be based on time (see the start_date parameter) or events, or be dependent on the outcome of other jobs. These options are discussed in this lesson.



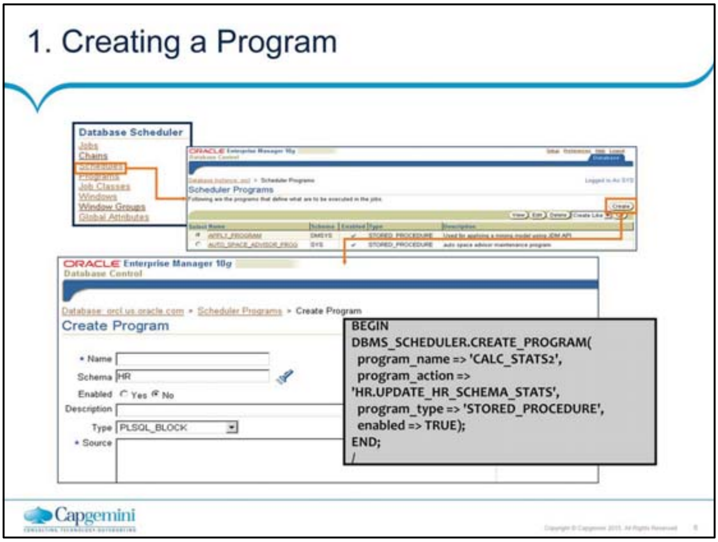
Key Components and Key Steps

The Scheduler offers a modularized approach for managing tasks within the Oracle database. By breaking down a task into its components, such as time, location, and database object, the Scheduler offers you an easier way to manage your database environment. The Scheduler uses three basic components:

A job specifies what needs to be executed and when. For example, the "what" could be a PL/SQL procedure, a native binary executable, a Java application, or a shell script. You can specify the program (what) and schedule (when) as part of the job definition, or you can use an existing program or schedule instead. You can use arguments for a job to customize its run-time behavior.

A schedule specifies when and how many times a job is executed. A schedule can be based on time or an event. You can define a schedule for a job by using a series of dates, an event, or a combination of the two, along with additional specifications to denote repeating intervals. You can store the schedule for a job separately and then use the same schedule for multiple jobs.

A program is a collection of metadata about a particular executable, script, or procedure. An automated job executes some task. Using a program enables you to modify the job task, or the "what," without modifying the job itself. You can define arguments for a program, enabling users to modify the run-time behavior of the task.



1. Creating a Program

Use the CREATE_PROGRAM procedure to create a program. Creating a program is an optional part of using the Scheduler. You can also encode the action to be performed within an anonymous PL/SQL block in the CREATE_JOB procedure. By creating the program separately, you can define the action once, and then reuse this action within multiple jobs. This enables you to change the schedule for a job without having to re-create the PL/SQL block. You can also customize the job by specifying argument values.

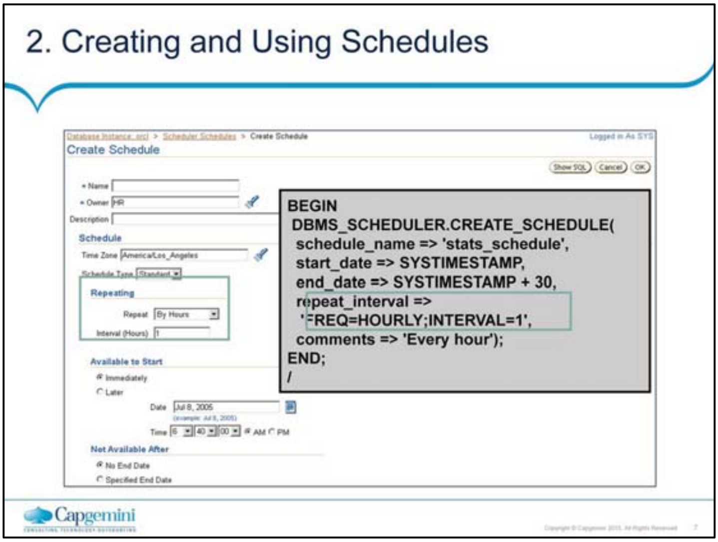
To create a program in your own schema, you need the CREATE JOB privilege. A user with the CREATE ANY JOB privilege can create a program in any schema.

A program is created in a disabled state by default (unless the enabled parameter is set to TRUE). A disabled program cannot be executed by a job until it is enabled. You can specify that a program should be created in the enabled state by specifying a value of TRUE for enabled.

The program action is a string specifying a procedure, executable name, or a PL/SQL anonymous block, depending on what program_type is set to.

If you have a procedure called UPDATE_HR_SCHEMA_STATS that collects the statistics for the hr schema, then you can create a program to call this procedure.

In Enterprise Manager, select Administration > Programs and click the Create button.



2. Creating and Using Schedules

By using a schedule (instead of specifying the execution times for a job within the job definition), you can manage the scheduled execution of multiple jobs without having to update multiple job definitions. If a schedule is modified, then each job that uses that schedule automatically uses the new schedule.

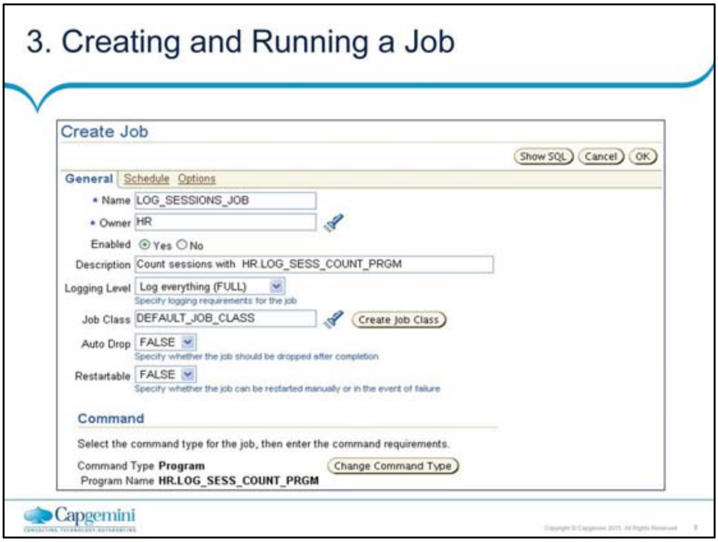
Use the CREATE_SCHEDULE procedure in the DBMS_SCHEDULER PL/SQL package to create a schedule.

The start_date represents the date on which the schedule becomes active. The schedule cannot refer to any dates before this date. The schedule is not valid after the end_date.

You can schedule repeated executions by supplying a calendaring expression for repeat_interval. This calendaring expression is used to generate the next date of the schedule. Dates falling after the end_date time are not included in the schedule.

In the example shown in the slide, a schedule named STATS_SCHEDULE is created, specifying a repeat interval of every four hours, starting now, and continuing for 30 days.

You can use Enterprise Manager to create schedules as shown in the slide.



3. Creating and Running a Job

A job is a combination of a schedule and a description of what to do, along with any additional arguments that are required by the job.

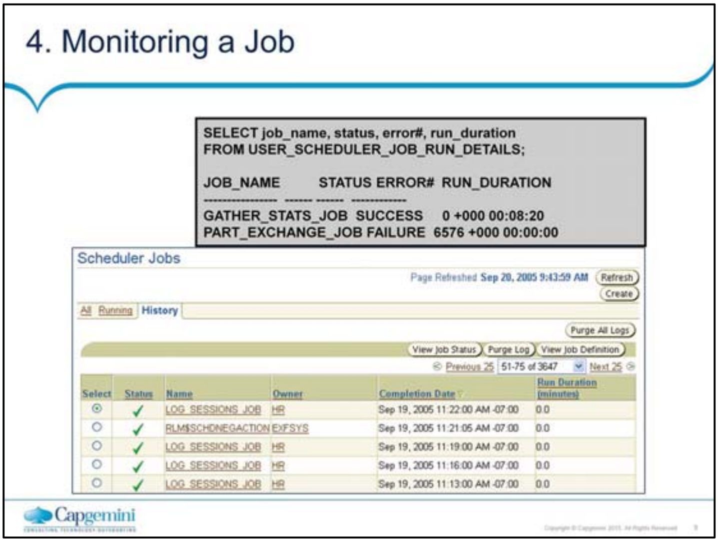
The program or “Command” can be a precreated PL/SQL or Java program, an anonymous PL/SQL block, or an executable that is run from the operating system’s command line.

The schedule for a job can be a predefined schedule (created with the DBMS_SCHDULER.CREATE_SCHEDULE procedure) or defined as part of the job creation. The schedule specifies attributes about when the job is run, such as:

- A start time, which defines when the job is picked for execution and an end time, which specifies the time after which the job is no longer valid and is not scheduled any more
- An expression specifying a repeating interval for the job
- A complex schedule created by combining existing schedules
- A condition or change in state, called an event, that must be met before the job is started

There are many attributes that you can set for a job. Attributes control how the job executes.

To run a job in Enterprise Manager, select Administration > Jobs.

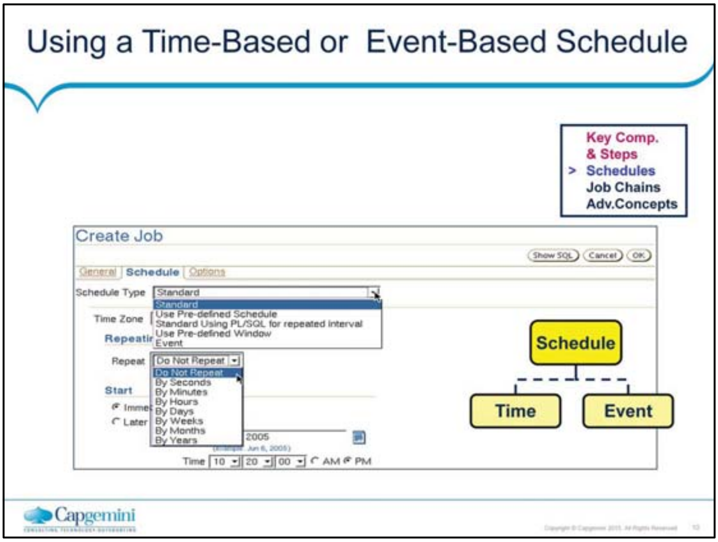


4. Monitoring a Job

The DBA_SCHEDULER_JOB_RUN_DETAILS view has a row for each job instance. Each row contains information about the job execution for that instance.

[DBA|ALL]_SCHEDULER_JOB_RUN_DETAILS views have the following columns:

LOG_ID: The unique identifier of the log entry
LOG_DATE: The time stamp of the log entry
OWNER: The job owner
JOB_NAME: The name of the job
STATUS: The status of the job execution
ERROR#: The number of the first error encountered
REQ_START_DATE: The time at which the job was scheduled to start
ACTUAL_START_DATE: The time at which the job was actually started
RUN_DURATION: The duration of execution of the job
INSTANCE_ID: The instance upon which the job ran
SESSION_ID: The session the job ran within
SLAVE_PID: The process ID of the slave process used to perform the job execution
CPU_USED: The amount of CPU used for the job run
ADDITIONAL_INFO: Additional information about the job run



Using a Time-Based or Event-Based Schedule

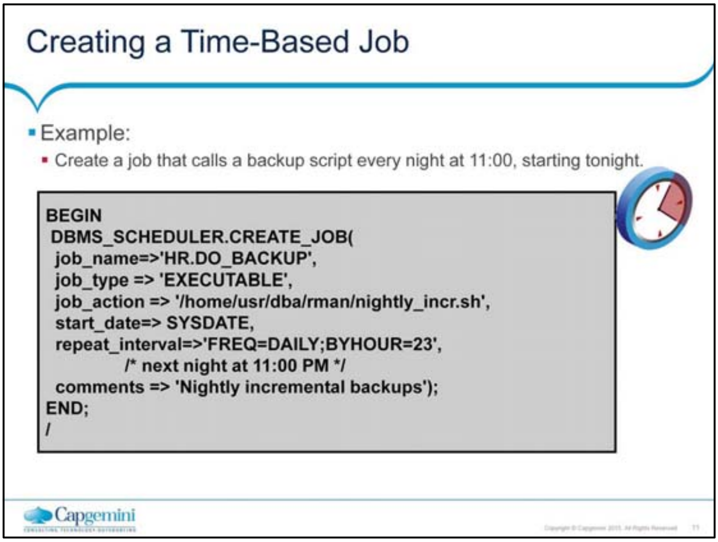
To specify a time-based schedule for a job, you can specify either a calendaring expression or a datetime expression

When using a calendaring expression, the next start time for a job is calculated using the repeat interval and the start date of the job. When using datetime expressions, the specified expression determines the next time that the job should run. If no repeat interval is specified, the job runs only once on the specified start date.

If a job uses an event-based schedule, the job runs when the event is raised. At a high level, an event can be viewed as a change in state. An event occurs when a Boolean condition changes its state from FALSE to TRUE, or TRUE to FALSE.

The Scheduler uses Oracle Streams Advanced Queuing (AQ) to raise and consume events.

Note: The Scheduler does not guarantee that a job executes on the exact time because the system may be overloaded and thus resources may be unavailable.



Creating a Time-Based Job

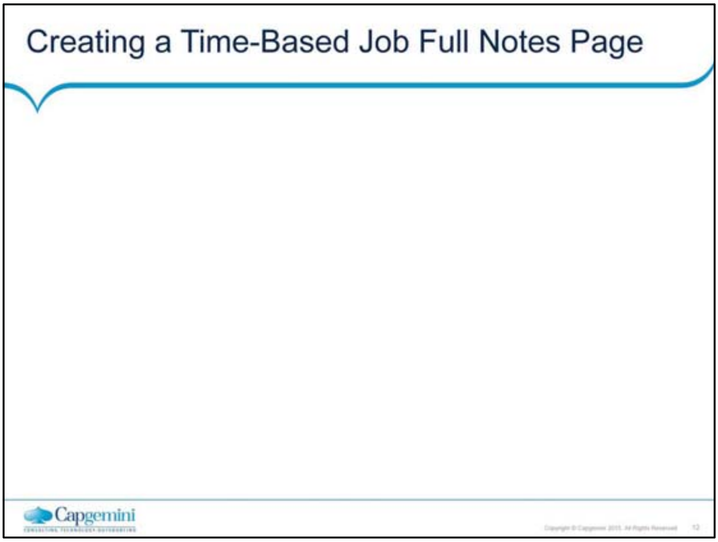
Use the CREATE_JOB procedure of the DBMS_SCHEDULER package to create a job. Jobs are created disabled by default and they become active and scheduled only when they are explicitly enabled. All job names are of the form: [schema.]name.

You should use SYSTIMESTAMP and specify a time zone so that when the time changes because of daylight saving time, your job adjusts its execution time automatically.

By default, a job is created in the current schema. You can create a job in another schema
by specifying the name of the schema, as shown in the example in the slide. The job owner is the user in whose schema the job is created, whereas the job creator is the user who created the job. Jobs are executed with the privileges of the job owner. The national language support (NLS) environment of the job when it runs is the same as that which was present at the time the job was created.

The job_type parameter indicates the type of task to be performed by the job. The possible values are:

- PLSQL_BLOCK: An anonymous PL/SQL block
- STORED_PROCEDURE: A named PL/SQL, Java, or external procedure
- EXECUTABLE: A command that can be executed from the operating system command line



Creating a Time-Based Job (continued)

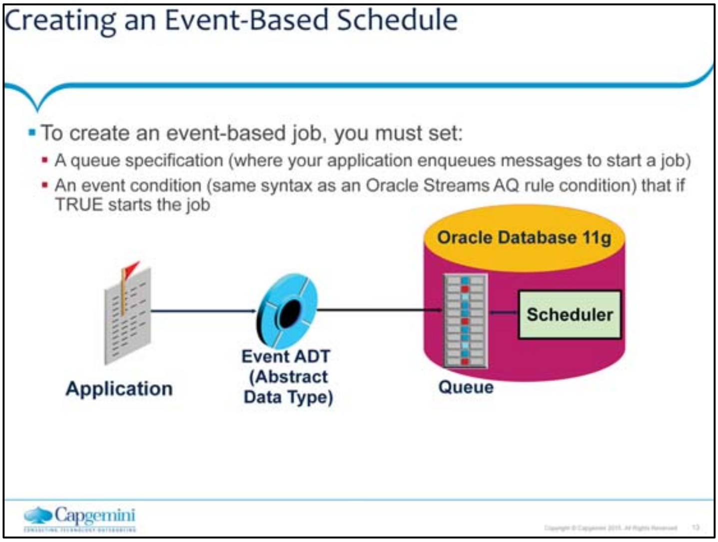
The job_action parameter can be the name of the procedure to run, the name of a script or operating system command, or an anonymous PL/SQL code block, depending on the value of the job_type parameter.

In the example in the slide, job_type is specified as EXECUTABLE and job_action is the full OS-dependent path of the desired external executable plus optionally any command-line arguments.

An external job is a job that runs outside the database. All external jobs run as a low-privileged guest user, as has been determined by the database administrator while configuring external job support. Because the executable is run as a low-privileged guest account, you should verify that it has access to necessary files and resources. Most, but not all, platforms support external jobs. For platforms that do not support external jobs, creating or setting the attribute of a job or a program to type EXECUTABLE returns an error.

Refer to your Oracle database platform-specific documentation for more information about configuring the environment to run external programs with the Scheduler. For example, you may need to reference one or more of the following books:

- Oracle Database Platform Guide 11g for Windows
- Oracle Database Installation Guide for UNIX Systems
- Oracle Database Release Notes 11g for AIX-Based Systems
- Oracle Database Release Notes 11g for hp HP-UX PA-RISC (64-bit



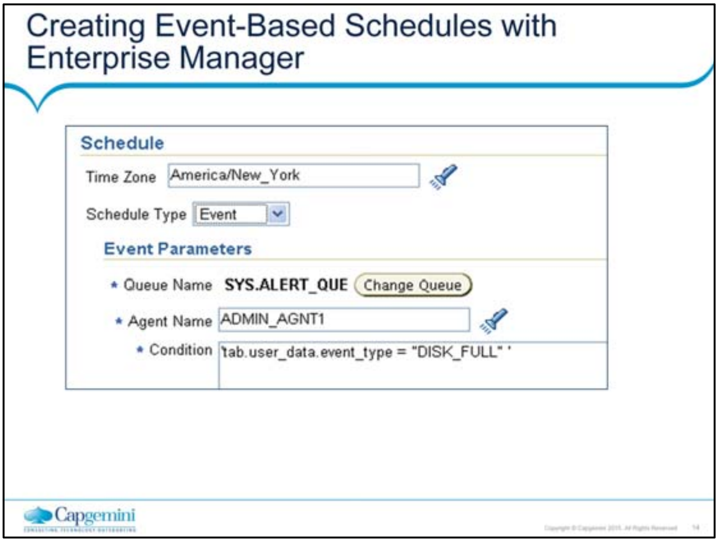
Creating an Event-Based Schedule

Jobs can be triggered based on events. An application can notify the Scheduler to start a job by enqueuing a message onto an Oracle Streams AQ queue. A job started in this way is referred to as an event-based job. To create an event-based job, you must set the following two additional attributes with the CREATE_JOB procedure:

queue_spec: A queue specification that includes the name of the queue where your application enqueues messages to raise job start events, or in the case of a secure queue, the <queue_name>, <agent_name> pair

event_condition: A conditional expression based on message properties that must evaluate to TRUE for the message to start the job. The expression must use the same syntax as an Oracle Streams AQ rule condition. You can include user data properties in the expression, provided that the message payload is a user-defined object type, and that you prefix object attributes in the expression with tab.user_data.

You can either specify queue_spec and event_condition as in-line job attributes, or create an event-based schedule with these attributes and then create a job that references this schedule.



Creating Event-Based Schedules with Enterprise Manager

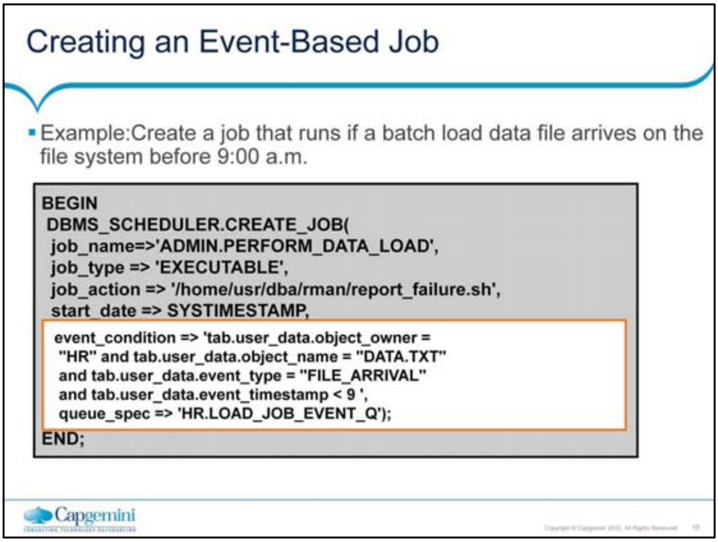
The Create Schedule page enables you to choose between a standard, time-based schedule and an event-based schedule. If you choose an event-based schedule, then the interface changes and you can specify the queue name, agent name, and event condition, in addition to the other schedule attributes.

Note

The Scheduler runs the event-based job for each occurrence of an event that matches event_condition. However, events that occur while the job is already running are ignored; the event gets consumed, but does not trigger another run of the job.

References:

- See the Oracle Streams Advanced Queuing User's Guide and Reference for information about how to create queues and enqueue messages.
- For more information about Oracle Streams AQ rules and event conditions, see the DBMS_AQADM.ADD_SUBSCRIBER procedure in the Oracle Database PL/SQL Packages and Types Reference 11g Release 2 manual.



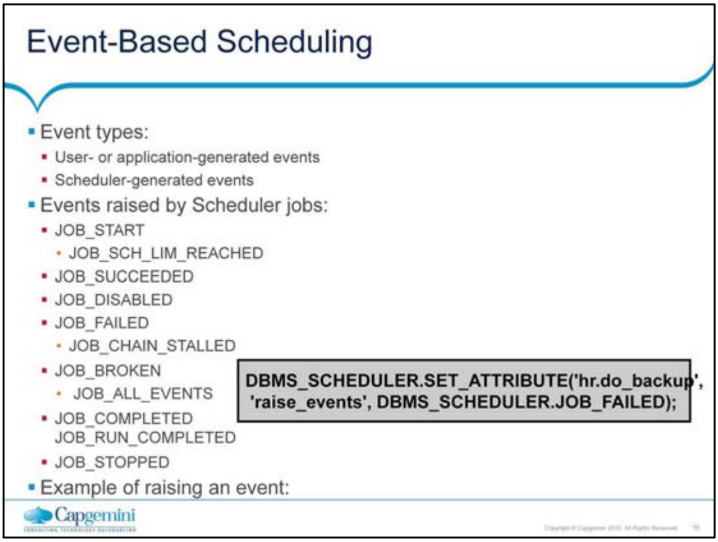
Creating an Event-Based Job

To specify event information as job attributes, you use an alternate syntax of CREATE_JOB that includes the queue_spec and event_condition attributes. The job can include event information in-line as job attributes or can specify event information by pointing to an event schedule. The example shown in the slide uses an in-line event-based schedule.

The example in the slide shows a job that is started when a file arrives on the operating system, as long as the file arrives before 9:00 a.m. Assume that the message payload is an object with four attributes named object_owner, object_name, event_type, and event_timestamp.

The example uses a user-defined event. Therefore, before this job can be started, when the file arrives on the file system, a program or procedure must enqueue the event object type with the proper information into the specified event queue. The HR.LOAD_JOB_EVENT_Q queue must be of the same type as the event object type used for notifying the Scheduler of an event occurrence. That is, the HR.LOAD_JOB_EVENT_Q queue must be a typed queue where the type has four attributes named object_owner, object_name, event_type, and event_timestamp.

For more information about how to create queues and enqueue messages, refer to the Oracle Streams Advanced Queuing User's Guide and Reference documentation.



Event-Based Scheduling

You can create a job that directly references an event as the means to start the job, instead of assigning a schedule to the job. There are two types of events:

User- or application-generated events: An application can raise an event to be consumed by the Scheduler. The Scheduler reacts to the event by starting a job. An example of such events: a running job completes; a file arrives on the file system; an account within the database is locked; and the inventory reaches a low threshold.

Scheduler-generated events: The Scheduler can raise an event to indicate state changes that occur within the Scheduler itself. For example, the Scheduler can raise an event when a job starts, when a job completes, when a job exceeds its allotted run time, and so on. The consumer of the event is an application that performs some action in response to the event.

You can configure a job so that the Scheduler raises an event when the job's state changes. You do this by setting the raise_events job attribute. By default, a job does not raise any state change events until you alter the raise_events attribute for a job. To alter this attribute, you must first create the job by using the CREATE_JOB procedure and then use the SET_ATTRIBUTE procedure to modify the attribute's default value. The example shows that the hr.do_backup job is altered, so that it raises an event if the job fails.

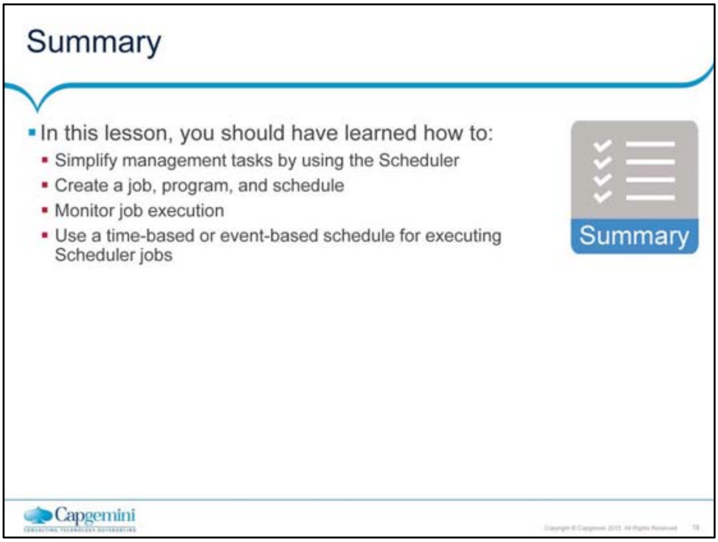
Event-Based Scheduling (continued)

After you enable job state change events for a job, the Scheduler raises these events by enqueuing messages onto the default event queue SYS.SCHEDULER$_EVENT_QUEUE.

The default Scheduler event queue is a secure queue. Depending on your application, you may have to configure the queue to enable certain users to perform operations on it. See the *Oracle Streams Concepts and Administration* documentation for information about secure queues.

The default Scheduler event queue is intended primarily for Scheduler-generated events. Oracle does not recommend the use of this queue for user applications, or user-defined events.

Event Type	Description
JOB_START	The job is started.
JOB_SUCCEEDED	The job is successfully completed.
JOB_FAILED	The job failed, either by raising an error or by abnormally terminating.
JOB_BROKEN	The job is disabled and changed to the BROKEN state, because it exceeded the number of failures defined by the MAX_FAILURES job attribute.
JOB_COMPLETED	The job is completed, because it reached the values set by the MAX_RUNS or END_DATE job attributes.
JOB_STOPPED	The job is stopped by a call to the STOP_JOB procedure.
JOB_SCH_LIM_REACHED	The job's schedule limit is reached. The job is not started, because the delay in starting the job exceeded the value of the SCHEDULE_LIMIT job attribute.
JOB_DISABLED	The job is disabled by the scheduler or by a call to the SET_ATTRIBUTE procedure.
JOB_CHAIN_STALLED	A job running a chain is put into the CHAIN_STALLED state. A running chain becomes stalled if there are no steps running or scheduled to run and the chain EVALUATION_INTERVAL is set to NULL. The chain waits for manual intervention.
JOB_ALL_EVENTS	JOB_ALL_EVENTS is not an event, but a constant, that provides an easy way for you to enable all events.
JOB_RUN_COMPLETED	A job run is completed. It either failed, succeeded, or is stopped.


Summary
In this lesson, you should have learned how to:
Simplify management tasks by using the Scheduler
Create a job, program, and schedule
Monitor job execution
Use a time-based or event-based schedule for executing Scheduler jobs
Summary
Capgemini

Oracle 11g DBA Fundamentals Overview

Lesson 13: Backup and Recovery Concepts

Objectives

- After completing this you should be able to do the following:
  - Describe the basics of database backup, restore and recovery.
  - List the types of failure that may occur in an Oracle Database.
  - Describe ways to tune instance recovery.
  - Identify the importance of checkpoints, redo log files, and archived log files.

Capgemini

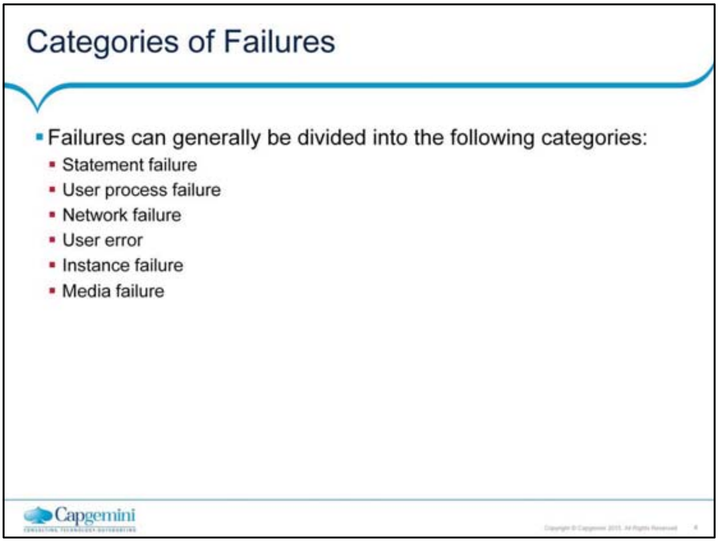
Backup and Recovery Issues

- The administrator's duty is to:
  - Protect the database from failure wherever possible.
  - Increase the Mean-Time-Between-Failures (MTBF).
  - Decrease the Mean-Time-To-Recover (MTTR).
  - Minimize the loss of data.

Backup and Recovery Issues

The DBA's goal is to ensure the database is open and available when users need it. In support of that goal, the DBA, usually working with the system administrator:

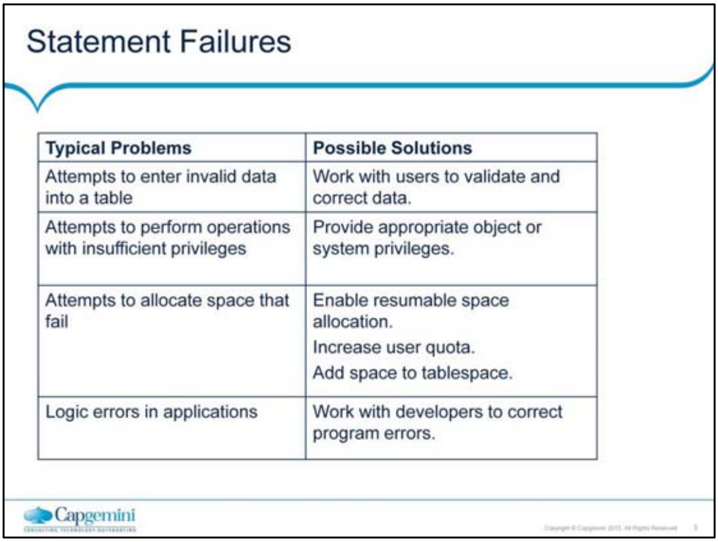
- Anticipates and works against common causes of failure
- Works to increase the mean-time-between-failure, ensuring hardware is as reliable as possible, that critical components are protected by redundancy, and that operating system maintenance is performed in a timely manner. Oracle provides advanced configuration options to increase MTBF including:
  - Real Application Clusters (discussed in another course)
  - Streams (discussed in another course)
- Decreases the mean-time-to-recover, practicing recovery procedures in advance and configuring backups so they are readily available when needed
- Minimizes the loss of data. DBAs who follow accepted best practices can configure their databases so that no committed transaction is ever lost. Tools to assist in guaranteeing this include:
  - Archived redo logs (discussed later in this lesson)
  - Standby databases and Oracle Data Guard (discussed in another course)



Categories of Failures

Failures can be divided into a few broad categories:

- Statement failure: A single database operation (select, insert, update, delete) fails.
- User process failure: A single database session fails.
- Network failure: Connectivity to the database is lost.
- User error: A user successfully completes an operation, but the operation was incorrect (dropping a table, entering incorrect data).
- Instance failure: The database instance shuts down unexpectedly.
- Media failure: One or more of the database files are lost (deleted, failed disk).

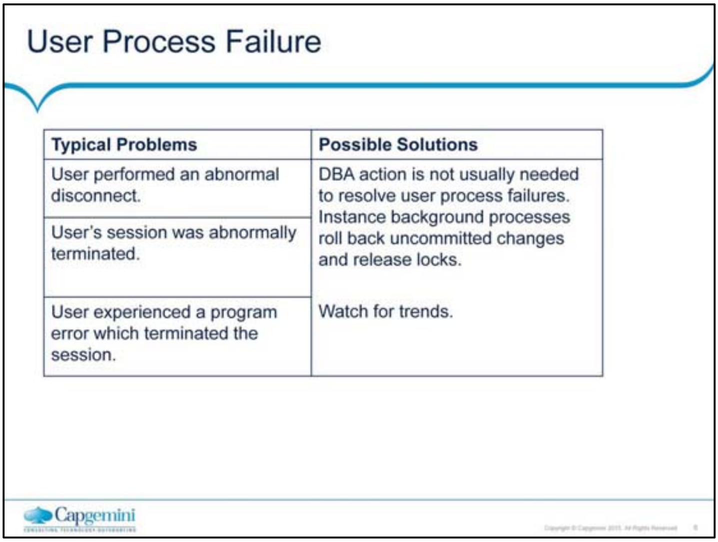


Statement Failures

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

Statement Failures

When a single database operation fails, DBA involvement may be needed to correct errors with user privileges or database space allocation.



User Process Failure

Users who are abnormally disconnected from the instance may have uncommitted work in progress that needs to be cleaned up. The PMON background process periodically polls server processes to ensure that their sessions are still connected. If PMON finds a server process whose user is no longer connected, PMON recovers from any ongoing transactions, including rolling back uncommitted changes and releasing any locks held by the failed session.

DBA intervention should not be required to recover from user process failure but the administrator should watch for trends. One or two users disconnecting abnormally is not a cause for concern. A small percentage of user process failures is normal. Consistent and systemic failures indicate other problems. A large percentage of abnormal disconnects may indicate a need for user training (teach them to log out rather than just terminating their programs). It may also be indicative of network or application problems.

Network Failure

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

Network Failure

The best solution to network failures is to provide redundant paths for network connections. Backup listeners, network connection, and network interface cards reduce the chance of network failures affecting system availability.

User Errors

Typical Causes	Possible Solutions
User inadvertently deletes or modifies data.	Roll back or use flashback query to recover.
User drops a table.	Recover table from recycle bin.

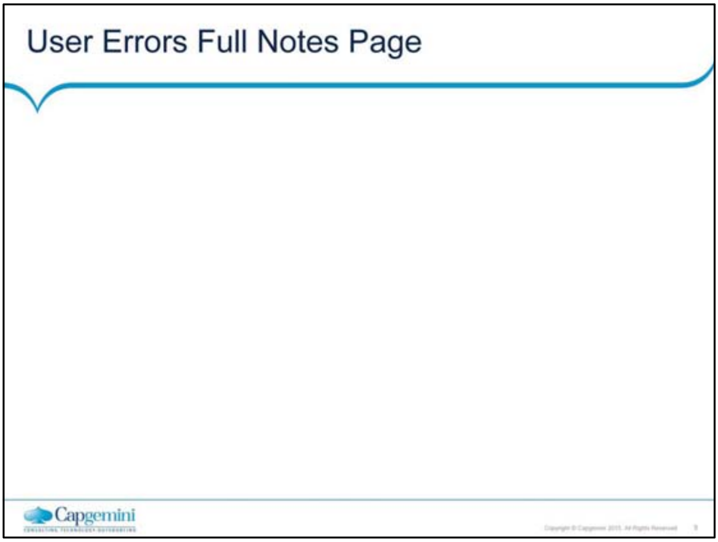
Capgemini

User Errors

Users may inadvertently delete or modify data. When that happens, the DBA may need to assist the user in recovering from the error. If the user has not yet committed or exited their program, they can simply roll back their operation. If the user has already committed the changes, flashback queries can be used to determine what the previous values were (and then the data can be updated to restore the original information.)

```
SQL> SELECT salary FROM employees WHERE
employee_id=100;
SALARY
------
25
SQL> SELECT salary FROM employees
2    AS OF TIMESTAMP(SYSTIMESTAMP-
INTERVAL'10' minute)
3    WHERE employee_id=100;
SALARY
------
24000
```

In cases where flashback queries are not possible because the undo retention period has been exceeded, the DBA may still be able to recover the original information through the use of Oracle LogMiner.



User Errors (continued)

Oracle LogMiner allows you to query your online redo logs and archived redo logs through an SQL interface. Transaction data may persist in online redo logs longer than it does in undo, and if you have configured archiving of redo information redo persists until you delete the archived files.

Oracle LogMiner is discussed in the Oracle Database 11g: Administration Workshop II course and in the Oracle Database: Utilities reference manual.

Users who drop tables can recover those tables from the recycle bin by flashing the table back to before the drop.

```
SQL> DROP TABLE hr.job_history;
Table dropped.

SQL> SELECT COUNT(*) FROM hr.job_history;
SELECT COUNT(*) FROM hr.job_history
              *
ERROR at line 1:
ORA-00942: table or view does not exist

SQL> FLASHBACK TABLE hr.job_history TO BEFORE
DROP;
Flashback complete.

SQL> SELECT COUNT(*) FROM hr.job_history;
  COUNT(*)
----------
        10
```

If the recycle bin has already been purged, or if the user dropped the table with the PURGE option, the dropped table can still be recovered by using point-in-time recovery (PITR) if the database has been properly configured.

PITR is discussed in the Oracle Database 11g: Administration Workshop II course and in the Oracle Database: Backup and Recovery Advanced User's Guide.

Instance Failure

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

Capgemini

Instance Failure

Instance failure occurs when the database instance is shut down before synchronizing all database files. An instance failure can occur due to hardware or software failure, or through the use of the emergency shutdown commands SHUTDOWN ABORT and STARTUP FORCE. Administrator involvement in recovering from instance failure is usually limited to restarting the instance and working to prevent future occurrences.

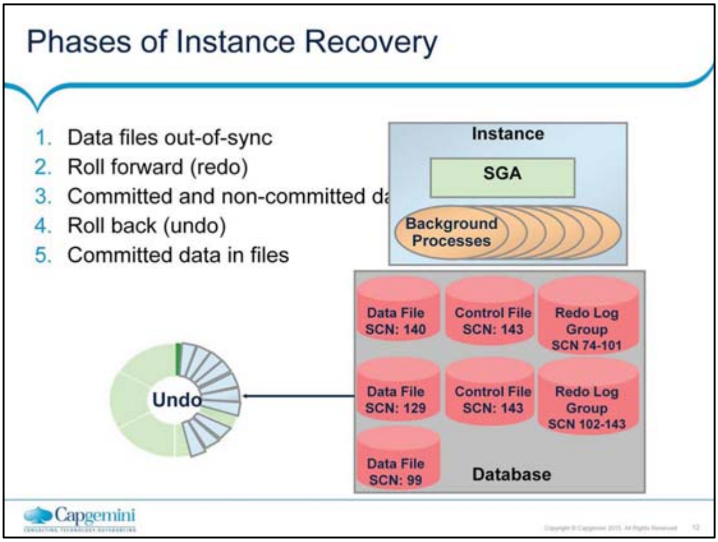
Instance Recovery

- Instance or crash recovery:
  - Is caused by attempts to open a database whose files were not synchronized on shutdown
  - Is automatic
  - Uses information stored in redo log groups to synchronize files
  - Involves two distinct operations
    - Rolling forward: Data files are restored to their state before the instance failed.
    - Rolling back: Changes made but not committed are returned to their original state.

Capgemini

Instance Recovery

The Oracle Database 11g automatically recovers from instance failure. All the DBA needs to do is start the instance normally. The instance will mount the control files and then attempt to open the data files. When it discovers the data files have not been synchronized during shutdown, the instance uses information contained in the redo log groups to roll the data files forward to the time of shutdown and then (because the undo tablespace was also rolled forward) roll back any uncommitted transactions.

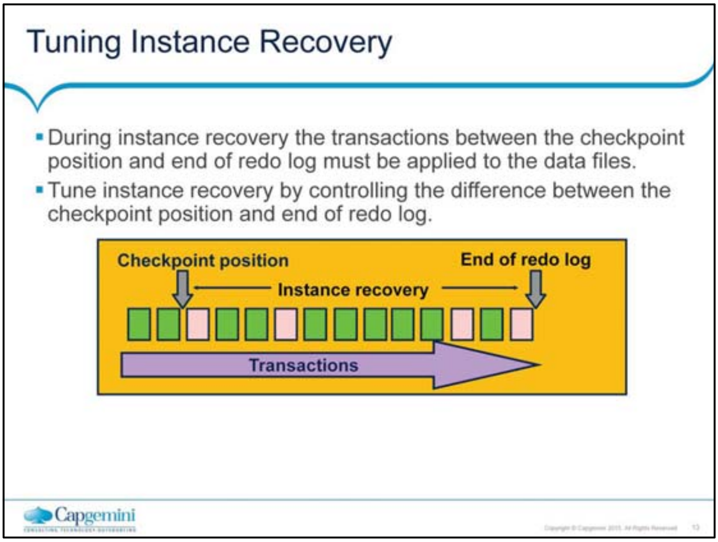


Phases of Instance Recovery

In order for an instance to open a data file, the system change number (SCN) contained within the data file's header must match the current SCN stored in the database's control files.

If the numbers do not match, the instance applies redo from the online redo logs, sequentially "redoing" transactions until the data files are up-to-date. After all data files have been synchronized with the control files, the database is opened and users may now log in.

When redo was applied, all transactions were applied to bring the database up to the state as of the time of failure. This usually includes transactions that were in progress but had not yet been committed. After the database has been opened, those uncommitted transactions are rolled back. At the end of the rollback phase of instance recovery, the data files will contain only committed data.



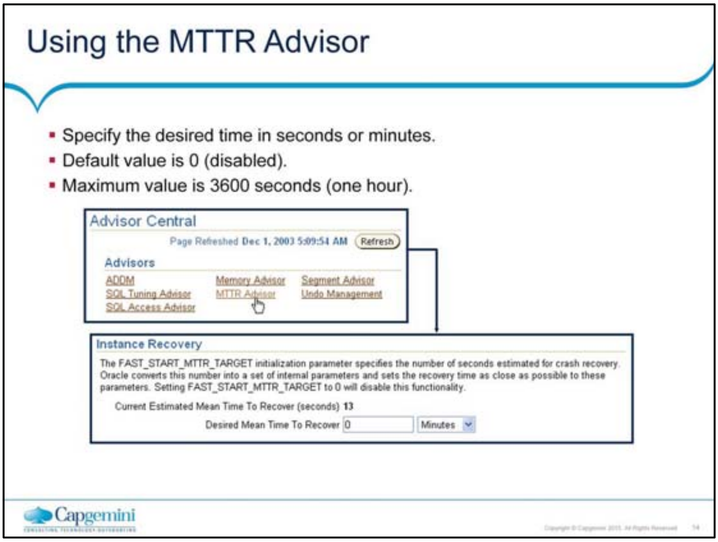
Tuning Instance Recovery

Transaction information is always recorded in the redo log groups before the instance returns commit complete for a transaction. The information in the redo log groups guarantees that the transaction can be recovered in case of a failure. That same transaction information also needs to be written to the data file. The data file write usually happens sometime after the information is recorded in the redo log groups because the data file write process is much slower than the redo writes (random writes for data files are slower than serial writes for redo log files).

To keep track of what has already been written to the data files, the database uses checkpoints. A checkpoint guarantees that as of the time the checkpoint occurs, all data up to a certain SCN is recorded in the data file. Transactions after the checkpoint position may or may not have yet been written to the appropriate data file. In the graphic above, the striped blocks have not yet been written to disk.

The time required for instance recovery is the time required to bring the data files from their last checkpoint to the latest SCN recorded in the control file. The administrator controls that time by setting a MTTR target (in seconds) and through the size of the redo log groups.

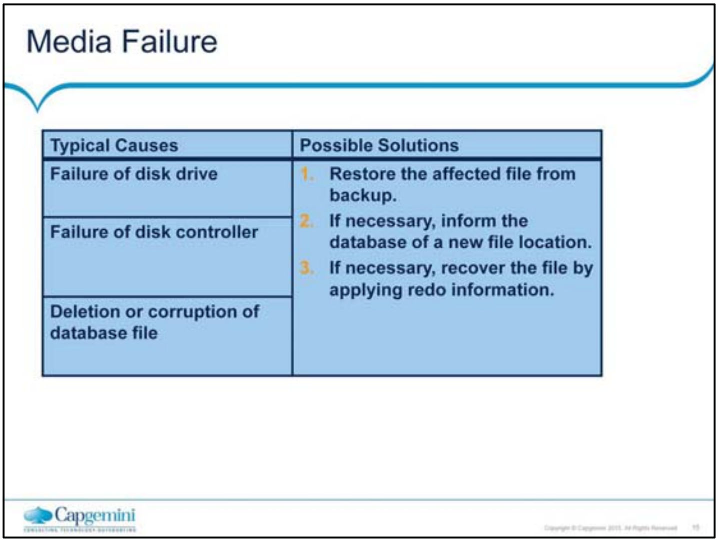
The distance between the checkpoint position and the end of the redo log group can never be more than 90% of the smallest redo log group.



MTTR Advisor

Click the MTTR Advisor from Enterprise Manager's Advisor Center for assistance in setting the MTTR target. The advisor allows you to specify a desired mean-time-to-recover and translates that into settings for the FAST_START_MTTR_TARGET initialization parameter.

The default setting of 0 disables the MTTR target, reducing the likelihood that writes to the log groups will wait on writes to the data files. This should be set to a value that supports the service level agreement for your system. Setting the MTTR target too small means that writes to the log groups wait for writes to the data files (impacting performance). Setting the MTTR target too large means that the instance takes longer to recover after a crash.

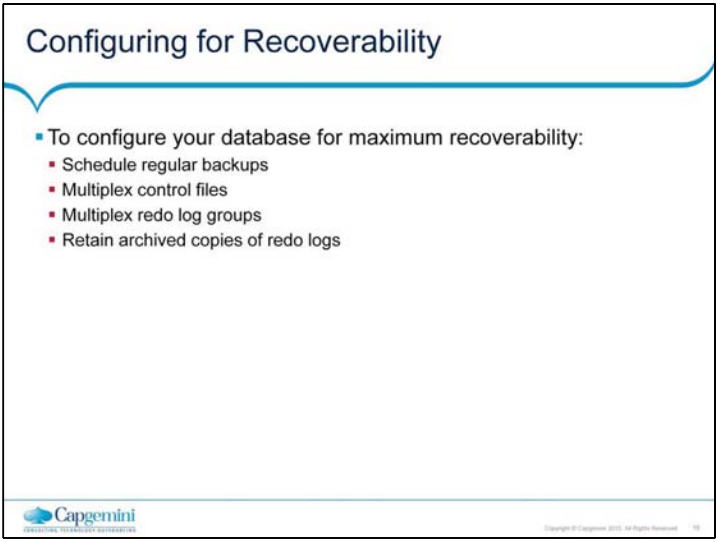


Media Failure

Typical Causes	Possible Solutions
Failure of disk drive	1. Restore the affected file from backup. 2. If necessary, inform the database of a new file location. 3. If necessary, recover the file by applying redo information.
Failure of disk controller	
Deletion or corruption of database file	

Media Failure

Oracle defines media failure as any failure which results in the loss or corruption of one or more database files (data, control, or redo log file). Recovering from media failure requires that you restore and recover the missing files. To ensure your database can be recovered from media failure, follow best practices as outlined in the next few pages. Recovery from media failure will be discussed in more detail in a future lesson.



Configuring for Recoverability

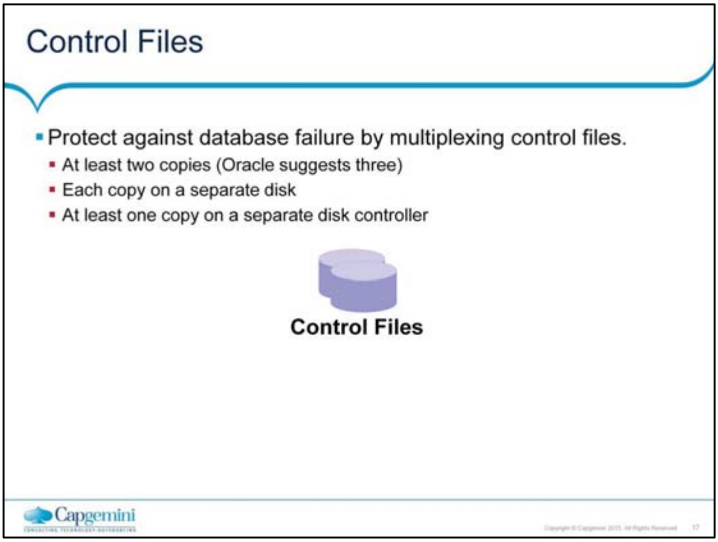
To provide the best protection for your data you should:

Schedule regular backups. Most media failures require that you restore the lost or damaged file from backup.

Multiplex control files. All control files associated with a database are identical. Recovering from the loss of single control file is not difficult. Recovering from the loss of all control files is much more challenging. Guard against losing all control files by having multiple copies (at least three).

Multiplex redo log groups. To recover from instance or media failure, redo log information is used to roll data files forward to the last committed transaction. If your redo log groups rely on a single redo log file, then the loss of that file means data is likely to be lost. Ensure there are at least two copies of each redo log group.

Retain archived copies of redo logs. If a file is lost and restored from backup, the instance must apply redo information to bring that file up to the latest SCN contained in the control file. The default setting is to overwrite redo information once it has been written to the data files. Your database can be configured to retain redo information in archived copies of the redo logs. This is known as placing the database in ARCHIVELOG mode.



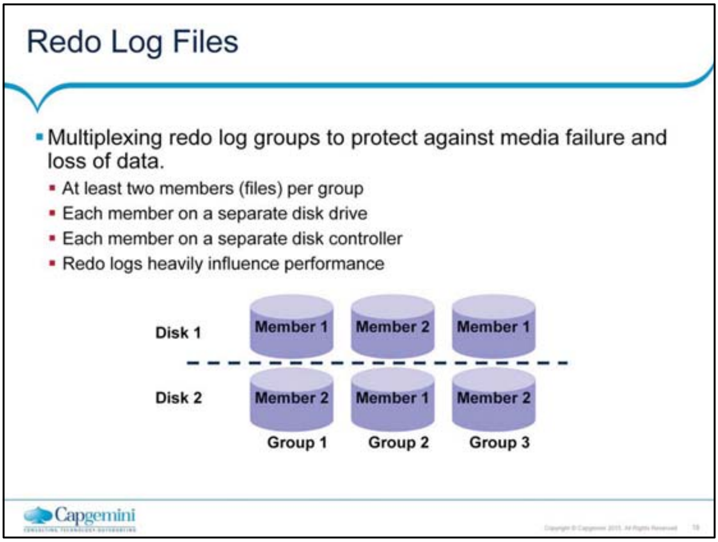
Control Files

The control file is a small binary file that describes the structure of the database. It must be available for writing by the Oracle server whenever the database is mounted or open. Without this file, the database cannot be mounted and recovery or re-creation of the control file will be required.

Your database should have a minimum of two control files (three is preferred) on different disks to minimize the impact of a loss of one control file.

If your database was created with the DBCA, you should already have three control files.

Loss of a single control file will cause the instance to fail because all control files must be available at all times, but recovery is a simple matter of copying one of the other control files. Loss of all control files is slightly more difficult to recover from, but not usually catastrophic. Recovery from loss of control files will be discussed in a later lesson.



Redo Log Files

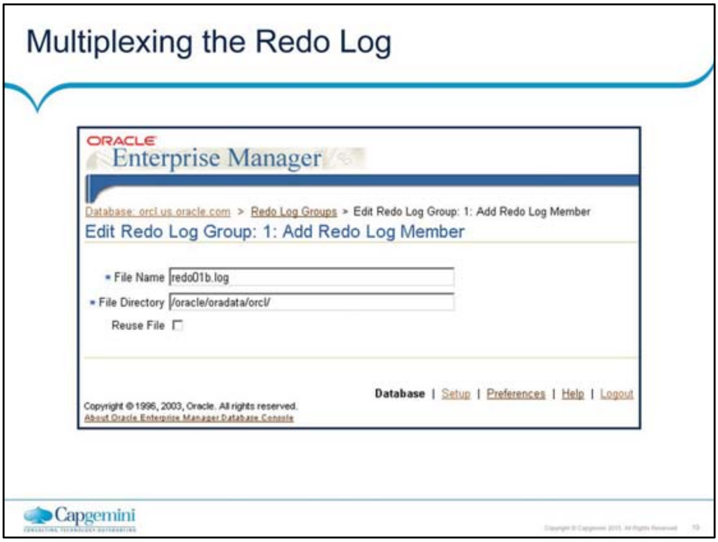
Redo log groups are made up one or more redo log files. Each log file within a group is a duplicate of the others. Oracle recommends that redo log groups have at least two files per group, with the files distributed on separate disks/controllers so that no single equipment failure will destroy an entire log group.

Loss of an entire log group is one of the most serious possible media failures because it can result in loss of data. Loss of a single member within a multiple-member log group is trivial, and will not affect database operation other than causing an alert to be published in the alert log.

Recovery from loss of a single log file will be discussed in a later lesson.

Recovery from loss of an entire log group requires advanced recovery techniques and is discussed in Oracle Database 11g: Workshop II.

Remember that redo logs heavily influence database performance because a commit cannot complete until the transaction information has been written to the logs. You should place your redo log files on your fastest disks served by your fastest controllers. If possible, do not place any other database files on the same disks as your redo log files. Because only one group is written to at a given time, there is no harm in having members from several groups on the same disk.



Multiplexing the Redo Log

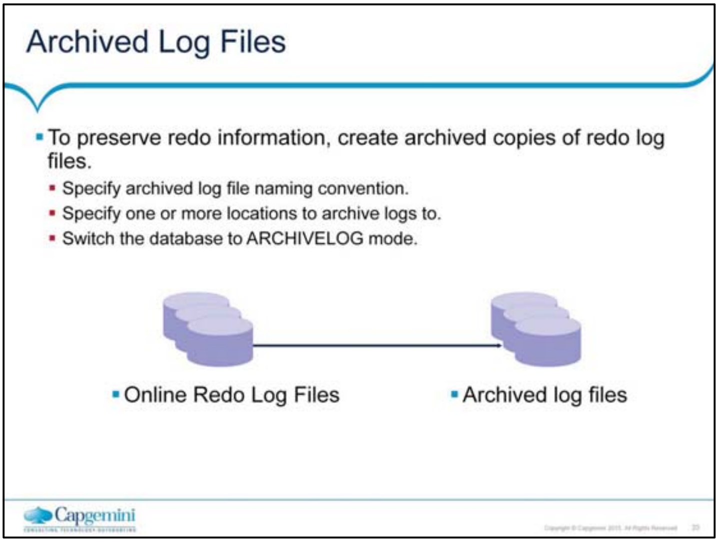
You can multiplex your redo log by adding a member to an existing log group. Perform the following steps to add a member to a redo log group, this can be done when the database is open with no impact on user performance:

1. Navigate to the Redo Log Groups page.
2. Select a group and click the Edit button, or click the group number link.The Edit Redo Log Group page appears.
3. In the Redo Log Members section, click Add. The Add Redo Log Member page appears.
4. Enter the file name and the file directory. Click OK.

   Note: It is recommended that you store members on separate drives to protect against total loss of the redo log entries in the event of a disk failure.

   Repeat these steps for every existing group.

When you add the redo log member to a group, the group's status is marked INVALID. This is the expected state because a member of the group has not yet been written to. When a log switch occurs and the invalid group becomes the current group, the status changes to CURRENT.



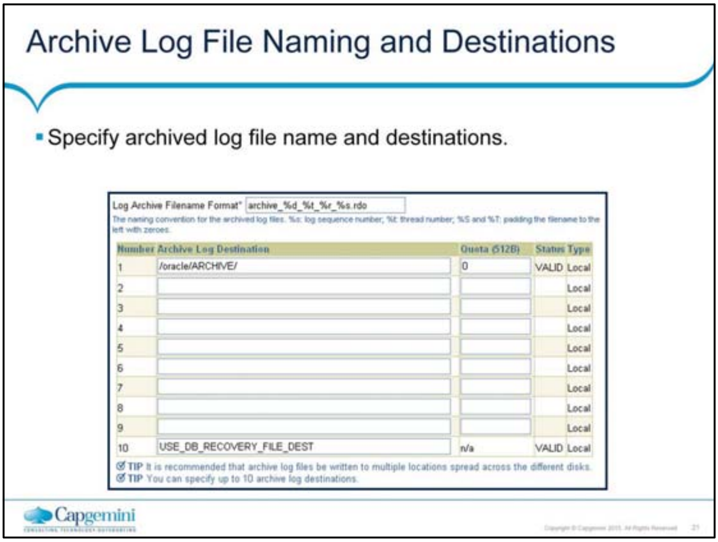
Archived Log Files

The instance treats the online redo log groups as a circular buffer in which to store transaction information, filling one group and then moving on to the next. After all groups have been written to, the instance begins overwriting information in the first log group.

To configure your database for maximum recoverability, you should instruct the database to make a copy of the online redo log group before allowing it to be overwritten. These copies are known as archived logs. To facilitate the creation of archive logs you should:

1. Specify a naming convention for your archived logs.
2. Specify a destination or destinations for storing your archived logs.
3. Place the database in ARCHIVELOG mode.

Note: The destination must exist prior to placing the database in ARCHIVELOG mode. When a directory is specified as a destination, there should be a trailing slash at the end of the directory name.



Archived Log File Naming and Destinations

Configure log file naming an destinations by clicking Configure Recovery Settings from the Maintenance page.

Each archive log file must have a unique name to avoid overwriting older log files. You specify the naming format as show above. To help create unique file names, Oracle Database 11g allows several wildcard characters in the name format:

- %s: Includes the log sequence number as part of the file name
- %t: Includes the thread number as part of the file name
- %r: Resetlogs ID. Ensures that the archive log file name remains unique even after certain advanced recovery techniques that reset log sequence numbers
- %d: Includes the database ID as part of the file name

The format must include %s, %t, and %r. The use of %d is optional but it should be included if multiple databases share the same archive log destination.

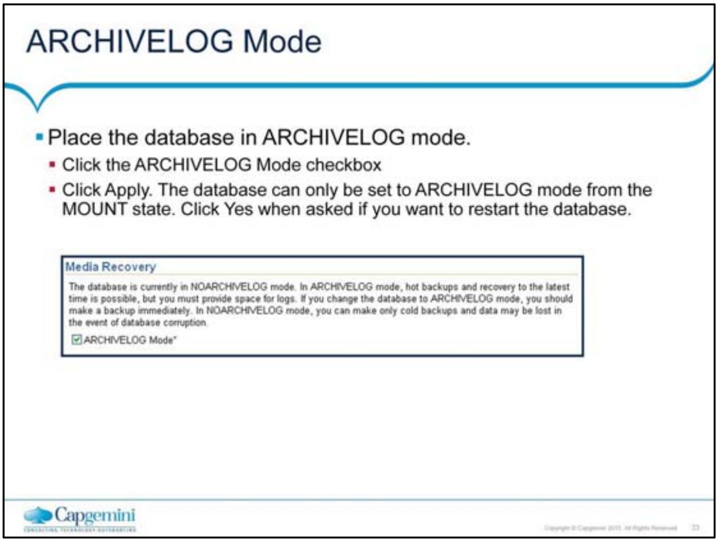
Archived log files can be written to as many as ten different destinations. Destinations may be local (a directory) or remote (an Oracle Net alias for a standby database). Local destinations should end in a slash (/), or a backslash (\) if using Windows.

Archive Log File Naming and Destinations Full Notes Page

Archive Log File Naming and Destinations (Continued)

The default destination (number 10) sends archived log files to a location determined by the DB_RECOVERY_FILE_DEST initialization parameter. DB_RECOVERY_FILE_DEST is also known as the flash recovery area. This destination is visible at the bottom of the Configure Recovery Settings properties page as the Flash Recovery Area Location. If you do not want archives sent to this location, simply delete USE_DB_RECOVERY_FILE_DEST.

In order to change recovery settings you must be connected as SYSDBA or SYSOPER.



ARCHIVELOG Mode

Placing the database in ARCHIVELOG mode prevents redo logs from being overwritten until they have been archived, and is the last step in configuring the database for archiving redo information.

The SQL command to place the database in ARCHIVELOG mode is:

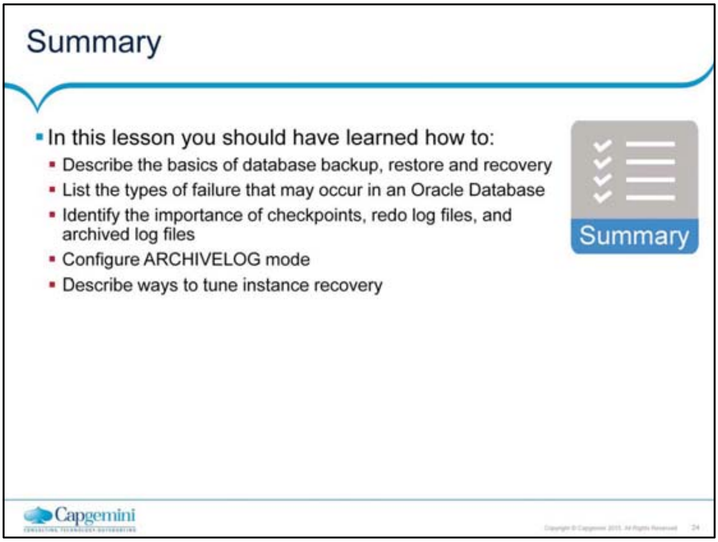
```
SQL> ALTER DATABASE ARCHIVELOG;
```

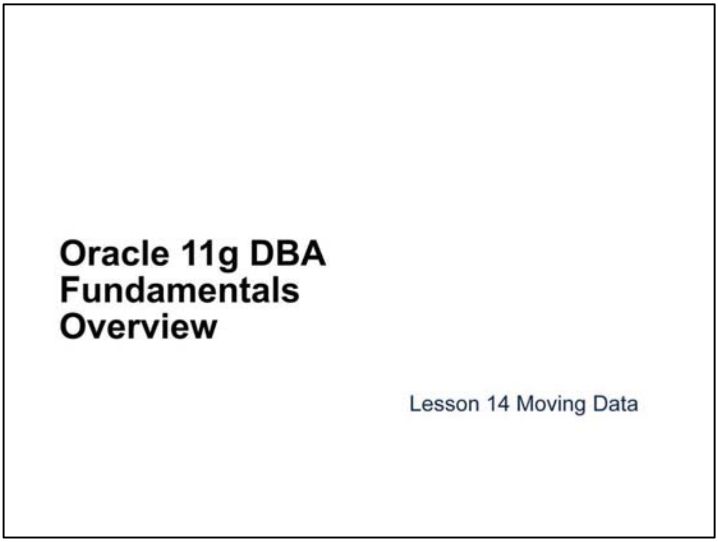
This command can be issued only while the database is in the MOUNT state, so the instance must be restarted to complete this last step. You will be asked for operating system and database credentials during the restart of the database. The database credentials must be for a user with SYSDBA privileges.

After the instance is restarted, the changes you made to the archive processes, log format, and log destinations will take effect.

With the database in NOARCHIVELOG mode (the default), recovery is possible only up until the time of the last backup. All transactions made after that backup are lost.

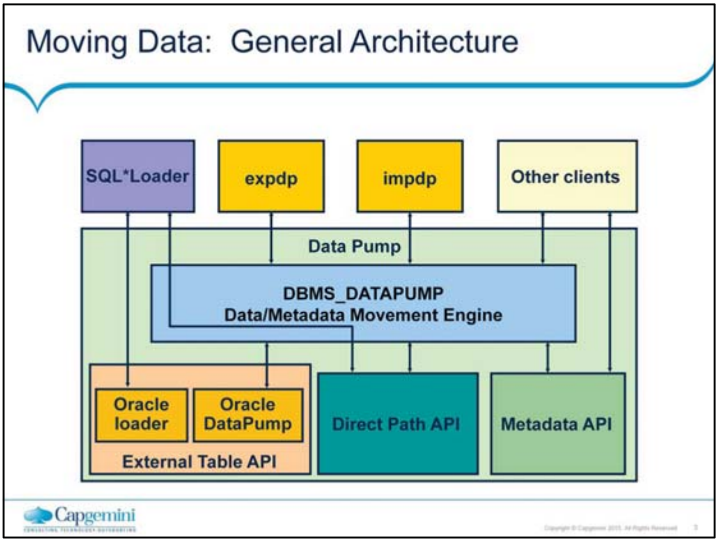
In ARCHIVELOG mode, recovery is possible up until the time of the last commit. Most production databases are run in ARCHIVELOG mode.


Summary
In this lesson you should have learned how to:
Describe the basics of database backup, restore and recovery
List the types of failure that may occur in an Oracle Database
Identify the importance of checkpoints, redo log files, and archived log files
Configure ARCHIVELOG mode
Describe ways to tune instance recovery
Summary
Capgemini


Oracle 11g DBA
Fundamentals
Overview
Lesson 14 Moving Data

Objectives

- After completing this lesson, you should be able to do the following:
  - Describe available ways for moving data
  - Create and use directory objects
  - Use SQL*Loader to load data from a non-Oracle database (or user files)
  - Explain the general architecture of Data Pump
  - Use Data Pump Export and Import to move data between Oracle databases
  - Use external tables to move data via platform-independent files



Moving Data: General Architecture

This is a block diagram of the major functional components:

DBMS_DATAPUMP: This package embodies the API for high-speed export and import utilities for bulk data and metadata movement.

Direct Path API (DPAPI): Oracle Database 11g supports a direct path API interface that minimizes data conversion and parsing at both unload and load time.

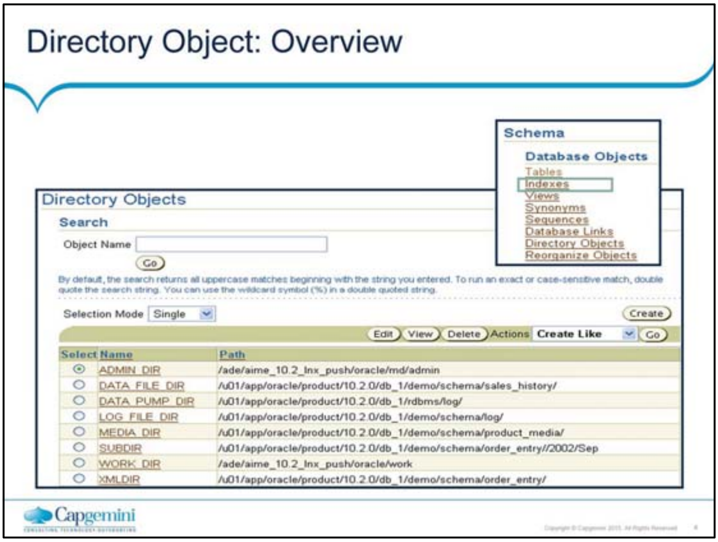
DBMS_METADATA: This package is used by worker processes for all metadata unloading and loading. Database object definitions are stored using XML rather than SQL.

External Table API: With the ORACLE_DATAPUMP and ORACLE_LOADER access drivers, you can store data in external tables (that is, in platform-independent files). The SELECT statement reads external tables as though they were stored in an Oracle database.

SQL*Loader: The SQL*Loader client has been integrated with external tables, thereby providing automatic migration of loader control files to external table access parameters.

expdp and impdp: The expdp and impdp clients are thin layers that make calls to the DBMS_DATAPUMP package to initiate and monitor Data Pump operations.

Other clients: They are applications, such as Database Control, replication, transportable tablespaces, and user applications, that benefit from this infrastructure. SQL*Plus may also be used as a client of DBMS_DATAPUMP for simple status queries against ongoing operations.



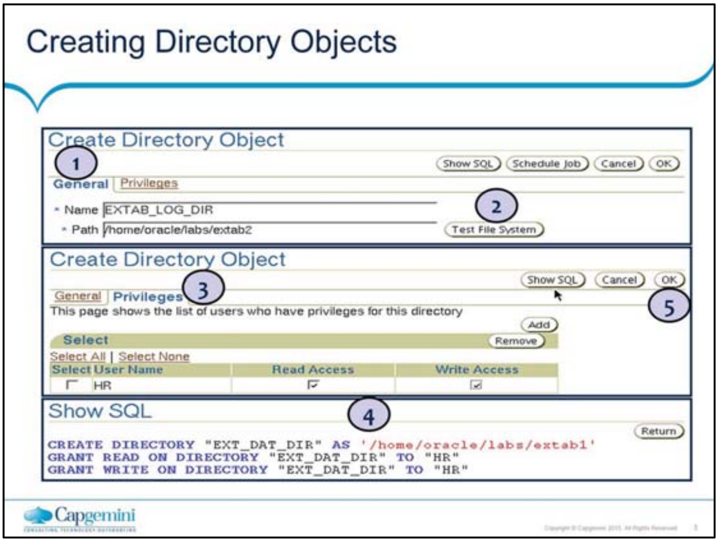
Directory Object: Overview

Directory objects are logical structures that represent a physical directory on the server's file system. They contain the location of a specific operating system directory. This directory object name can be used in Enterprise Manager, so you do not need to hard-code directory path specifications. Therefore, you get greater file management flexibility. Directory objects are owned by the SYS user. Directory names are unique across the database because all the directories are located in a single name space (that is, SYS).

Directory objects are required when you specify file locations for Data Pump because it accesses files on the server rather than on the client.

In Enterprise Manager, select Administration > Directory Objects.

To edit or delete a directory object, select the directory object and click the appropriate button.



Creating Directory Objects

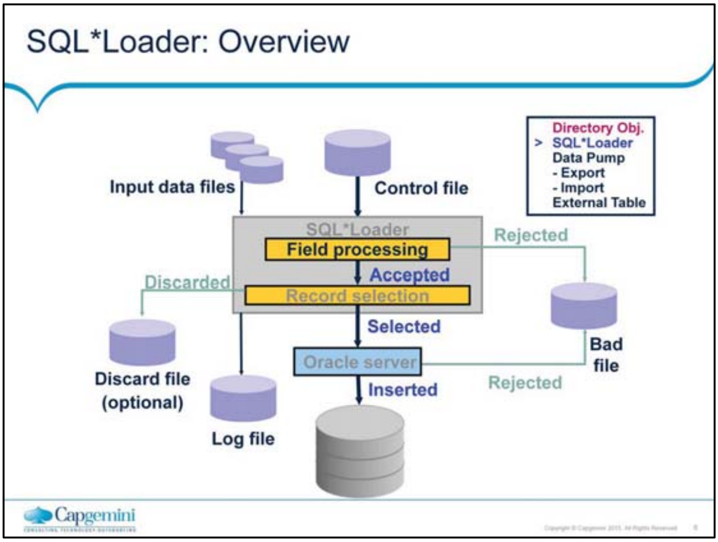
On the Directory Objects page, click the Create button.

2. Enter the name of the directory object and the OS path to which it maps. OS directories should be created before they are used. You can test this by clicking the “Test File System” button. For the test, provide the host login credentials (that is, the OS user who has privileges on this OS directory).

3. Permissions for the directory objects are not the same as the OS permissions on the physical directory on the server file system. You can manage user privileges on individual directory objects. This increases the level of security and gives you granular control over these objects. On the Privileges tabbed page, click Add to select the user to which you give read or write privileges or both.

4. Click Show SQL to view the underlying statements.

5. Click OK to create the object.



SQL*Loader: Overview

SQL*Loader loads data from external files into tables of an Oracle database. It has a powerful data parsing engine that puts little limitation on the format of the data in the data file. The files that are used by SQL*Loader are as follows:

Input data files: SQL*Loader reads data from one or more files (or operating system–equivalents of files) that are specified in the control file. From SQL*Loader's perspective, the data in the data file is organized as records. A particular data file can be in fixed record format, variable record format, or stream record format. The record format can be specified in the control file with the INFILE parameter. If no record format is specified, the default is stream record format.

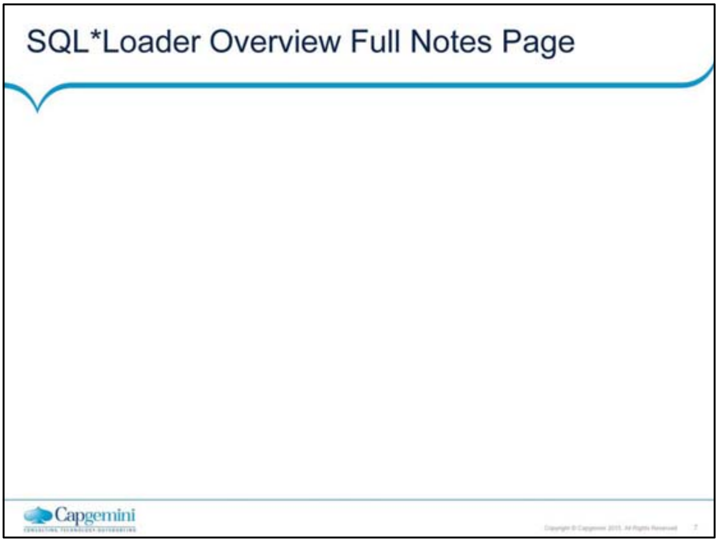
Control file: The control file is a text file that is written in a language that SQL*Loader understands. The control file indicates to SQL*Loader where to find the data, how to parse and interpret the data, where to insert the data, and so on. Although not precisely defined, a control file can be said to have three sections.

The first section contains sessionwide information, for example:

Global options, such as the input data file name, and records to be skipped.

INFILE clauses to specify where the input data is located

Data to be loaded



SQL*Loader: Overview (continued)

The second section consists of one or more INTO TABLE blocks. Each of these blocks contains information about the table (such as the table name and the columns of the table) into which the data is to be loaded.

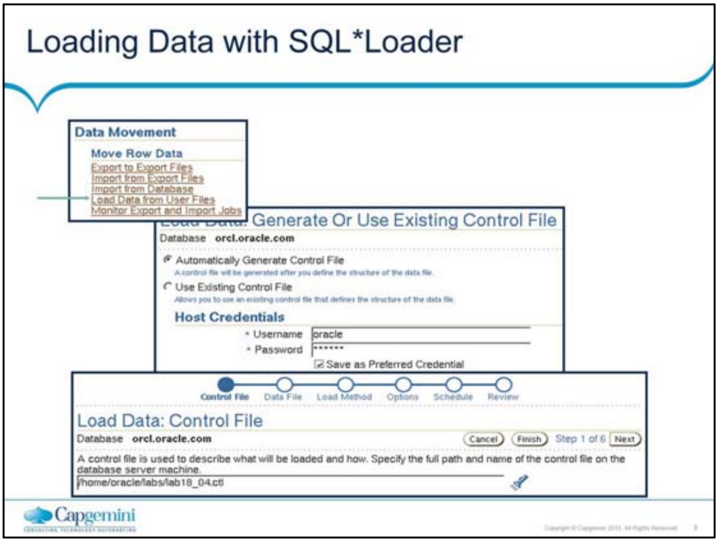
The third section is optional and, if present, contains input data.

Log file: When SQL*Loader begins execution, it creates a log file. If it cannot create a log file, execution terminates. The log file contains a detailed summary of the load, including a description of any errors that occurred during the load.

Bad file: The bad file contains records that are rejected, either by SQL*Loader or by the Oracle database. Data file records are rejected by SQL*Loader when the input format is invalid. After a data file record is accepted for processing by SQL*Loader, it is sent to the Oracle database for insertion into a table as a row. If the Oracle database determines that the row is valid, then the row is inserted into the table. If the row is determined to be invalid, then the record is rejected and SQL*Loader puts it in the bad file.

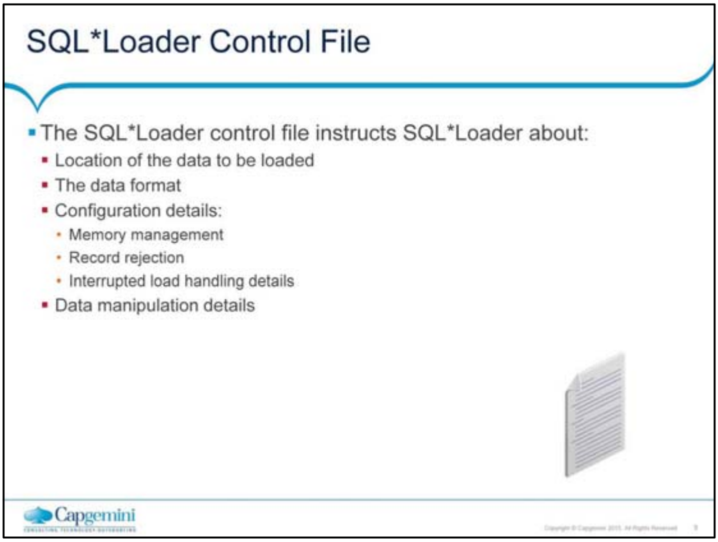
Discard file: This file is created only when it is needed, and only if you have specified that a discard file should be enabled. The discard file contains records that are filtered out of the load because they do not match any record-selection criteria specified in the control file.

For more information about SQL*Loader, refer to the Oracle Database Utilities documentation.



Loading Data with SQL*Loader

Use the Load Data from User Files Wizard to load data from a flat file into an Oracle database. To display the wizard, select Enterprise Manager Maintenance > Data Movement > Move Row Data > Load Data from User Files. The wizard guides you through the required steps.



SQL*Loader Control File

The SQL*Loader control file is a text file that contains data definition language (DDL) instructions. DDL is used to control the following aspects of a SQL*Loader session:

- Where SQL*Loader finds the data to load
- How SQL*Loader expects that data to be formatted
- How SQL*Loader is being configured (including memory management, selection and rejection criteria, interrupted load handling, and so on) as it loads the data
- How SQL*Loader manipulates the data being loaded

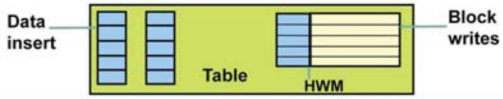
SQL*Loader Control File (continued)

```
1     -- This is a sample control file
2 LOAD DATA
3 INFILE 'SAMPLE.DAT'
4 BADFILE 'sample.bad'
5 DISCARDFILE 'sample.dsc'
6 APPEND
7 INTO TABLE emp
8 WHEN (57) = '.'
9 TRAILING NULLCOLS
10 (hiredate SYSDATE,
            deptno POSITION(1:2) INTEGER EXTERNAL(3)
      NULLIF deptno=BLANKS,
            job POSITION(7:14) CHAR TERMINATED BY WHITESPACE
            NULLIF job=BLANKS "UPPER(:job)",
            mgr POSITION(28:31) INTEGER EXTERNAL
            TERMINATED BY WHITESPACE, NULLIF mgr=BLANKS,
      ename POSITION(34:41) CHAR
            TERMINATED BY WHITESPACE "UPPER(:ename)",
            empno POSITION(45) INTEGER EXTERNAL
            TERMINATED BY WHITESPACE,
            sal POSITION(51) CHAR TERMINATED BY WHITESPACE
            "TO_NUMBER(:sal,'$99,999.99')",
            comm INTEGER EXTERNAL ENCLOSED BY '(' AND '%'
            ":comm * 100"
    )
```

The explanation of a sample control file by line numbers is as follows:

1. Comments can appear anywhere in the command section of the file, but they must not appear within the data. Precede any comment with two hyphens. All text to the right of the double hyphen is ignored, until the end of the line.
2. The LOAD DATA statement indicates to SQL*Loader that this is the beginning of a new data load. If you are continuing a load that has been interrupted in progress, use the CONTINUE LOAD DATA statement.
3. The INFILE keyword specifies the name of a data file containing data that you want to load.
4. The BADFILE keyword specifies the name of a file into which rejected records are placed.
5. The DISCARDFILE keyword specifies the name of a file into which discarded records are placed.
6. The APPEND keyword is one of the options that you can use when loading data into a table that is not empty. To load data into a table that is empty, use the INSERT keyword.
7. The INTO TABLE keyword enables you to identify tables, fields, and data types. It defines the relationship between records in the data file and tables in the database.
8. The WHEN clause specifies one or more field conditions that each record must match before SQL*Loader loads the data. In this example, SQL*Loader loads the record only if the 57th character is a decimal point. That decimal point delimits dollars and cents in the field and causes records to be rejected if SAL has no value.
9. The TRAILING NULLCOLS clause prompts SQL*Loader to treat any relatively positioned columns that are not present in the record as null columns.
10. The remainder of the control file contains the field list, which provides information about column formats in the table that is being loaded.

Loading Methods



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

Capgemini

Comparing Direct and Conventional Path Loads

Method of Saving Data

Conventional path loads use SQL processing and a database COMMIT operation for saving data. The insertion of an array of records is followed by a COMMIT operation. Each data load may involve several transactions.

Direct path loads use data saves to write blocks of data to Oracle data files. This is why the direct path loads are faster than the conventional ones. The following features differentiate a data save from COMMIT:

- During a data save, only full database blocks are written to the database.
- The blocks are written after the high-water mark (HWM) of the table.
- After a data save, the high-water mark (HWM) is moved.
- Internal resources are not released after a data save.
- A data save does not end the transaction.
- Indexes are not updated at each data save.

Note: Direct path and parallel direct path loads are so similar regarding DML activities that they are not separated in this comparison.

Comparing Direct and Conventional Path Loads (continued)

Logging Changes

Conventional path loading generates redo entries similar to any DML statement. When using a direct path load, redo entries are not generated if:

- The database is in NOARCHIVELOG mode
- The database is in ARCHIVELOG mode, but logging is disabled. Logging can be disabled by setting the NOLOGGING attribute for the table or by using the UNRECOVERABLE clause in the control file.

Enforcing Constraints

During a conventional path load, all enabled constraints are enforced in the same way that they are during any DML operation.

During direct path loads, the constraints are handled as follows:

- NOT NULL constraints are checked when arrays are built.
- FOREIGN KEY and CHECK constraints are disabled, and they can be enabled at the end of the load by using the appropriate commands in the control file. The FOREIGN KEY constraints are disabled because they reference other rows or tables, and the CHECK constraints are disabled because they may use SQL functions. If only a small number of rows are to be inserted into a large table, then use conventional loads.
- PRIMARY KEY and UNIQUE constraints are checked during and at the end of the load, and they may be disabled if they are violated.

Firing the INSERT Triggers

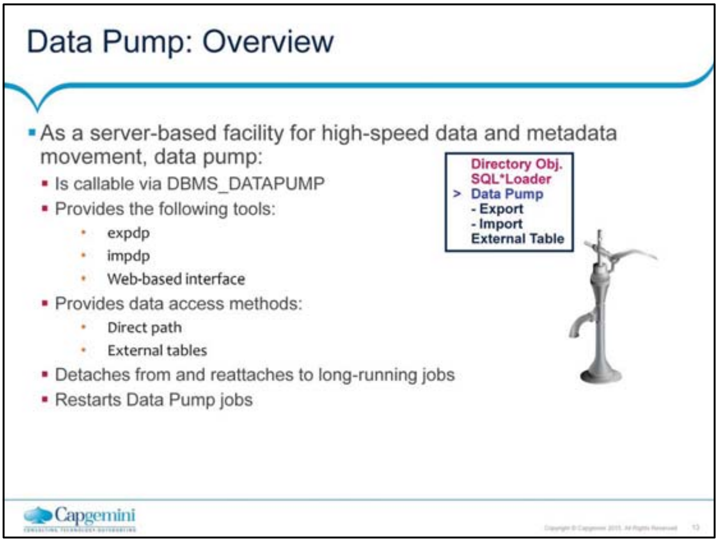
The WHILE INSERT triggers are fired during conventional loads; they are disabled before a direct path load and reenabled at the end of the load. They may remain disabled if a referenced object is not accessible at the end of the run. Consider using conventional path loads to load data into tables with the INSERT triggers.

Loading into Clustered Tables

Direct Loads cannot be used to load rows into clustered tables. Clustered tables can be loaded with conventional path loads only.

Locking

While a direct path load is in progress, other transactions cannot make changes to the tables that are being loaded. The only exception to this rule is when several parallel direct load sessions are used concurrently.



Data Pump: Overview

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

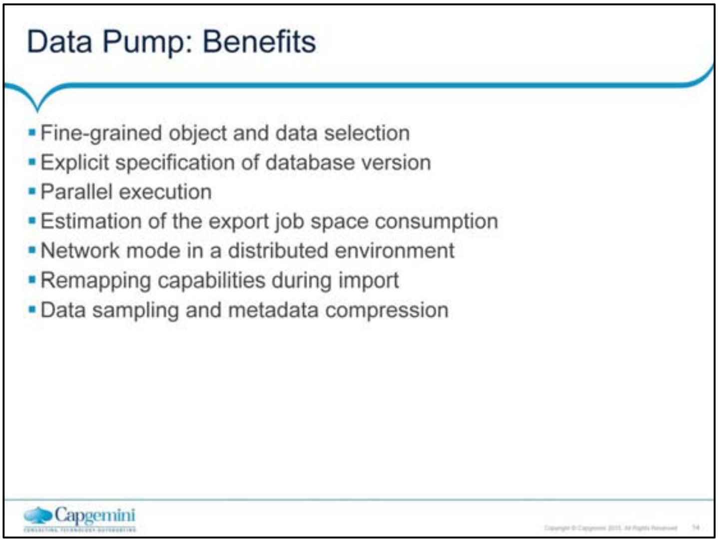
Oracle Database 11g provides the following tools:

- Command-line export and import clients called expdp and impdp respectively
- A Web-based export and import interface that is accessible from Database Control

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

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

Note: Data Pump is an integral feature of Oracle Database 11g and is, therefore, available in all configurations. However, parallelism is available in Enterprise Edition only.



Data Pump: Benefits

The EXCLUDE, INCLUDE, and CONTENT parameters are used for fine-grained object and data selection.

You can specify the database version for objects to be moved (using the VERSION parameter) to create a dump file set that is compatible with a previous release of the Oracle database that supports Data Pump.

You can use the PARALLEL parameter to specify the maximum number of threads of active execution servers operating on behalf of the export job.

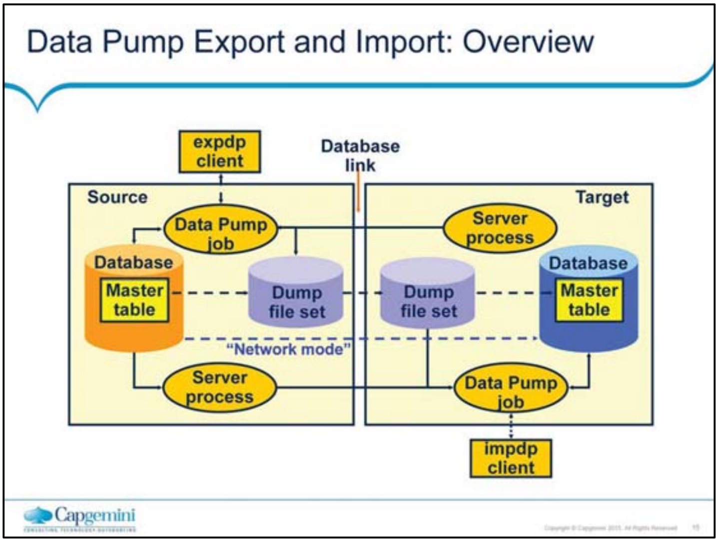
You can estimate how much space an export job would consume (without actually performing the export) by using the ESTIMATE_ONLY parameter.

Network mode enables you to export from a remote database directly to a dump file set. This can be done by using a database link to the source system.

During import, you can change the target data file names, schemas, and tablespaces.

In addition, Oracle Database 11g enables you to specify a percentage of data to be sampled and unloaded from the source database when performing a Data Pump export. This can be done by specifying the SAMPLE parameter.

You can use the COMPRESSION parameter to indicate whether the metadata should be compressed in the export dump file so that it consumes less disk space. If you compress the metadata, it is automatically uncompressed during import.



Data Pump Export and Import: Overview

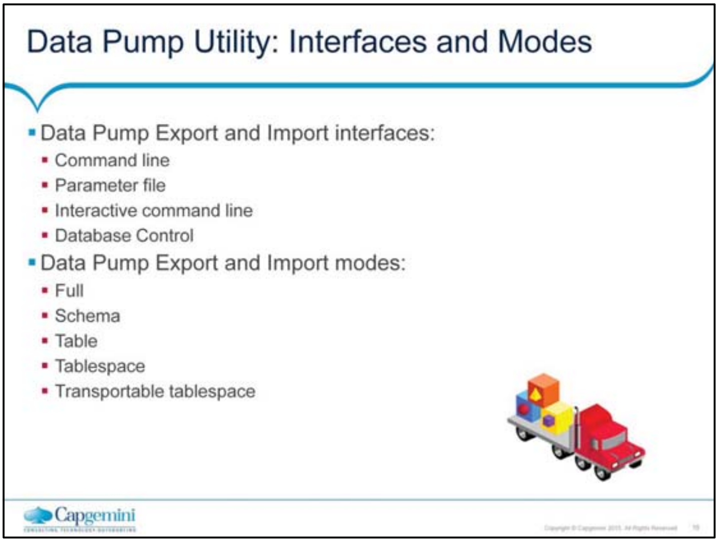
Data Pump Export is a utility for unloading data and metadata into a set of operating system files called dump file sets. Data Pump Import is used to load metadata and data stored in an export dump file set into a target system.

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

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

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

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



Data Pump Utility: Interfaces and Modes

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

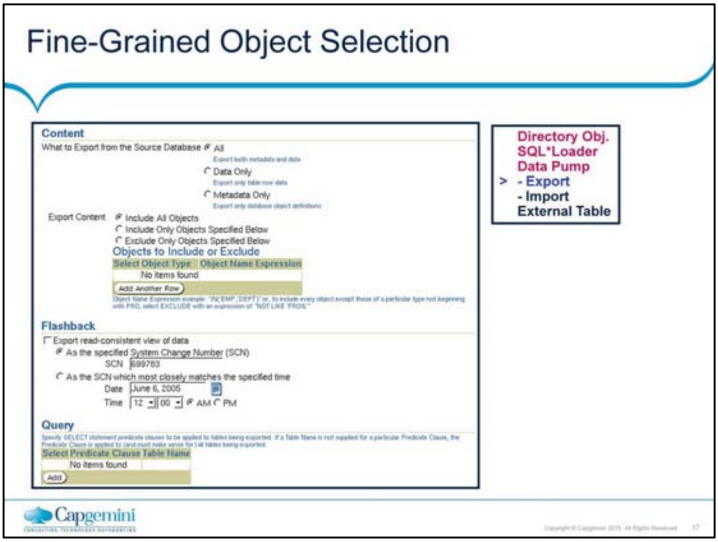
The command-line interface enables you to specify most of the export parameters directly on the command line.

The parameter file interface enables you to specify all command-line parameters in a parameter file. The only exception is the PARFILE parameter.

The interactive-command interface stops logging to the terminal and displays the export or import prompts, where you can enter various commands. This mode is enabled by pressing [Ctrl] + [C] during an export operation that is started with the command-line interface or the parameter file interface. Interactive-command mode is also enabled when you attach to an executing or stopped job.

You can also access the Web interface. On the Database Control home page, click the Maintenance tab, and then select one of the following links from the Utilities region: Export to Files, Import from Files, or Import from Database.

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



Fine-Grained Object Selection

The Data Pump job can include or exclude virtually any type of object. The EXCLUDE parameter enables any database object type to be excluded from an export or import operation. The optional name qualifier enables you to have finer selectivity within each object type that is specified. Examples:

EXCLUDE=VIEW
EXCLUDE=PACKAGE
EXCLUDE=INDEX:"LIKE 'EMP%'"

The INCLUDE parameter includes only the specified object types and objects in an operation. Syntax:

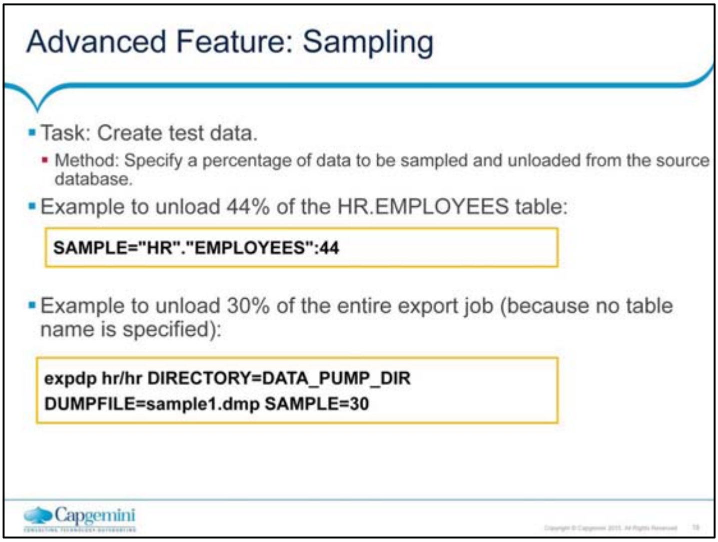
INCLUDE = object_type[:"name_expr"]

The CONTENT parameter enables you to request for the current operation, only the metadata, only the data, or both. Syntax:

CONTENT = ALL | METADATA_ONLY | DATA_ONLY

The QUERY parameter operates in a similar manner as the original export utility, with two significant enhancements: It can be qualified with a table name so that it applies to only that table, and it can be used during import as well. Example:

QUERY=hr.employees:"WHERE department_id in (10,20) and salary < 1600 ORDER BY department_id"



Advanced Feature: Sampling

With the SAMPLE parameter, you can specify a percentage of data to be sampled and unloaded from the source database when performing a Data Pump export.

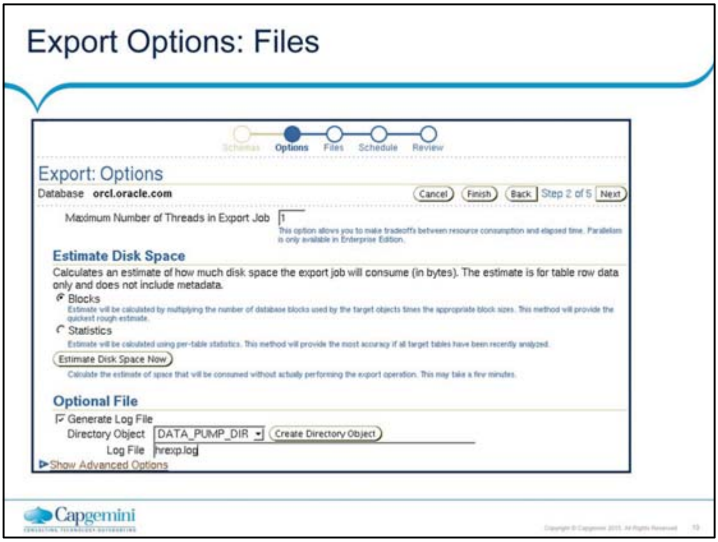
Syntax:

SAMPLE=[[schema_name.]table_name:]sample_percent

Range for sample_percent: .000001 to (but not including) 100

Sample percentage indicates the likelihood that a block of rows will be included.

Note: The SAMPLE parameter is not valid for network exports.

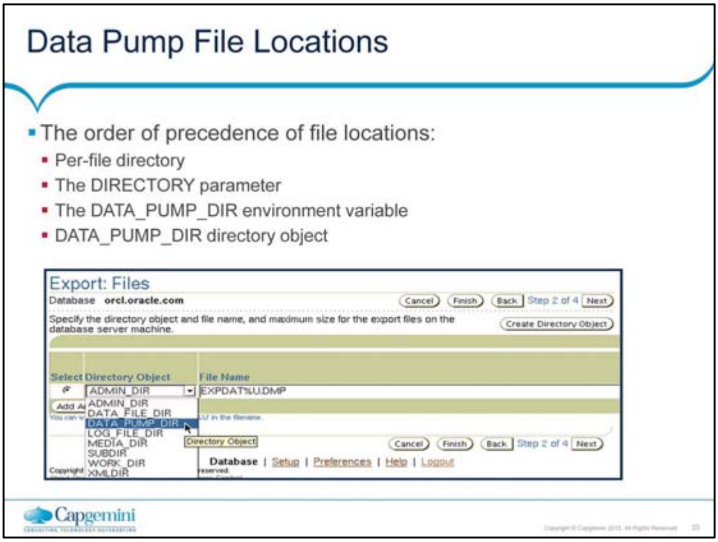


Export Options: Files

There are three types of files that are managed by Data Pump jobs:

- Dump files for data and metadata that is to be moved
- Log files for messages
- SQL files for the output of a SQLFILE operation

Because Data Pump is server based and not client based, Data Pump files are accessed relative to Oracle directory paths. Absolute paths are not supported for security reasons.



Data Pump File Locations

The slide shows you the order of precedence used by Data Pump clients to locate these files:

Per-file directory objects may be specified for each dump file, log file, and SQL file. If specified, they are separated from the file name by a colon (:).

The Data Pump Export and Import clients provide a DIRECTORY parameter, which specifies the name of a directory object. These directory objects describe the location in which the files are accessed.

You can alternatively define an environment variable, DATA_PUMP_DIR, to specify the directory object name rather than use the DIRECTORY parameter. The Data Pump clients look for this environment variable if no explicit directory object is specified.

There is a default directory object created for every database. This directory object is named DATA_PUMP_DIR. Access to the DATA_PUMP_DIR directory is granted automatically to the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles.

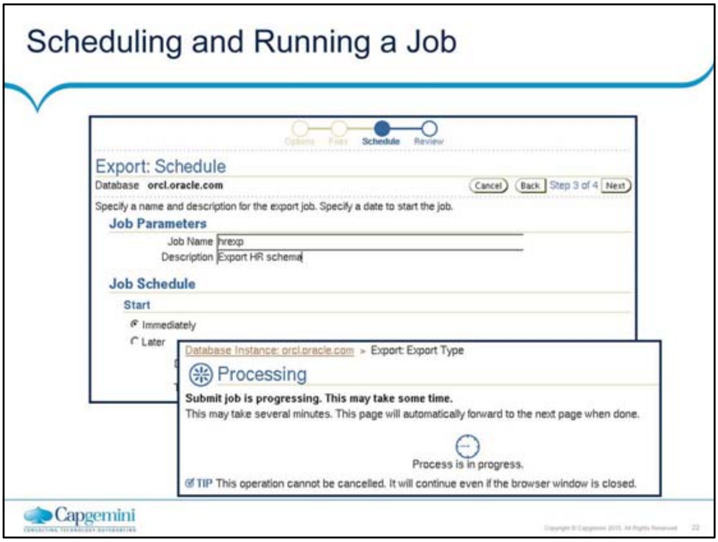
Data Pump File Locations (continued)

You do not need to create a directory object manually before using Data Pump Export. There is a default directory object created for every database, whether newly created or upgraded by a script on UNIX or Windows platforms. This directory object is named DATA_PUMP_DIR. Access to the DATA_PUMP_DIR directory is granted automatically to the EXP_FULL_DATABASE and IMP_FULL_DATABASE roles. The DATA_PUMP_DIR directory is created in one of the following locations:

<ORACLE_BASE>/admin/DB_UNIQUE_NAME
<ORACLE_HOME>/admin/DB_UNIQUE_NAME

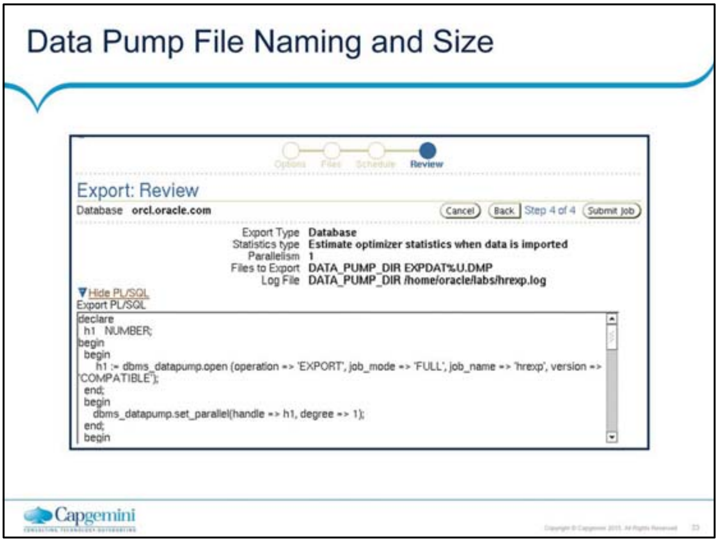
The exact directory path specification for DATA_PUMP_DIR varies, depending on the value of the ORACLE_BASE and ORACLE_HOME system environment variables and on the existence of the DATA_PUMP_DIR subdirectory. If ORACLE_BASE is defined on the target system, then that value is used. Otherwise, the value of ORACLE_HOME is used. If, for some reason, the DATA_PUMP_DIR subdirectory is not found, the following default path is used: ORACLE_HOME/rdbms/log.

Note: In all cases, you must have the appropriate access privileges to the directory object for the operation attempted. For export, you need write access for all files; for import, you need read access for dump files and write access for log files and SQL files.



Scheduling and Running a Job

Data Pump jobs (created through this wizard) are scheduled as repeatable jobs by Enterprise Manager Database Control.



Data Pump File Naming and Size

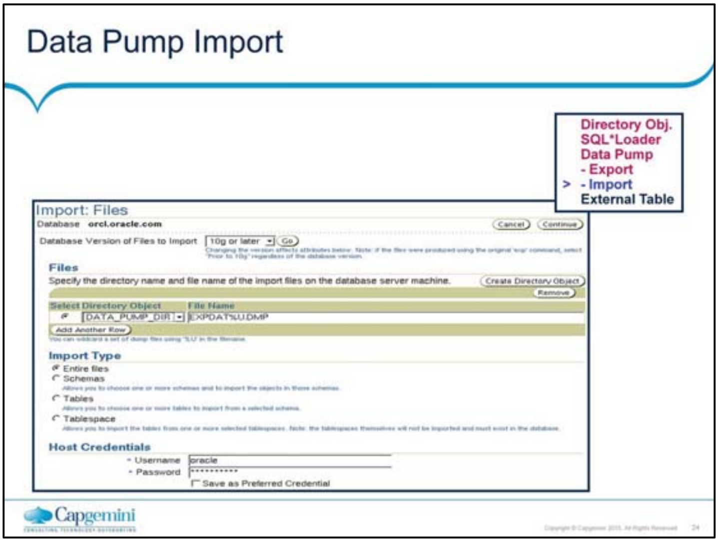
The DUMPFILE parameter specifies the names and (optionally) directories of disk-based dump files. Multiple file specifications may be provided as a comma-separated list or in separate DUMPFILE parameter specifications. File names may contain the substitution variable %U, which implies that multiple files may be generated. %U is expanded in the resulting file names into a two-character, fixed-width, monotonically increasing integer starting at 01. If no DUMPFILE is specified, expdat.dmp is used by default. By default, created dump files are autoextensible.

If FILESIZE is specified, each file is FILESIZE bytes in size and nonextensible. If more dump space is required and a template with %U has been supplied, then a new file is automatically created with FILESIZE bytes; otherwise, the client receives a message to add a new file.

If a template with %U is specified, the number of files initially created is equal to the PARALLEL parameter.

Preexisting files that match the resulting file names are not overwritten; they result in an error and cause the job to be aborted.

Note: If multiple dump file templates are provided, then they are used to generate dump files in a circular fashion.



Data Pump Import

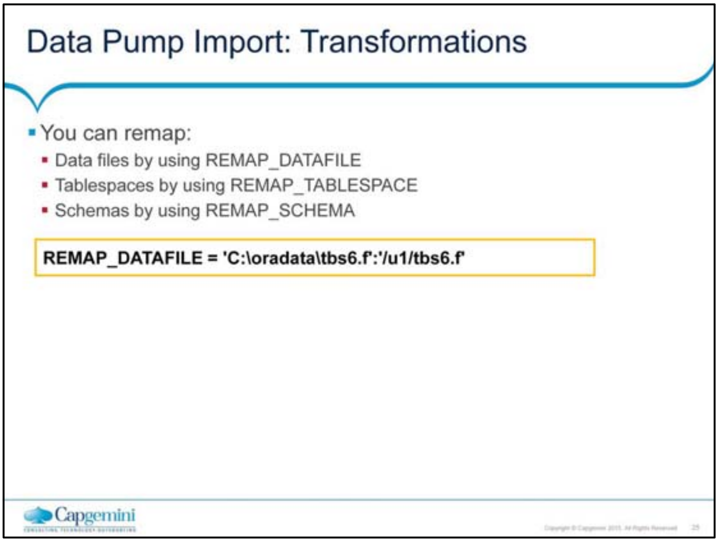
Data Pump Import is a utility for loading an export dump file set into a target system. The dump file set is made up of one or more disk files that contain table data, database object metadata, and control information. The files are written in a proprietary, binary format. During an import operation, the Data Pump Import utility uses these files to locate each database object in the dump file set.

You can interact with Data Pump Import by using a command line, a parameter file, or an interactive-command mode:

You can use the impdp command and specify parameters directly on the command line.

You can enter command-line parameters in a file (the PARFILE parameter is excluded because parameter files cannot be nested).

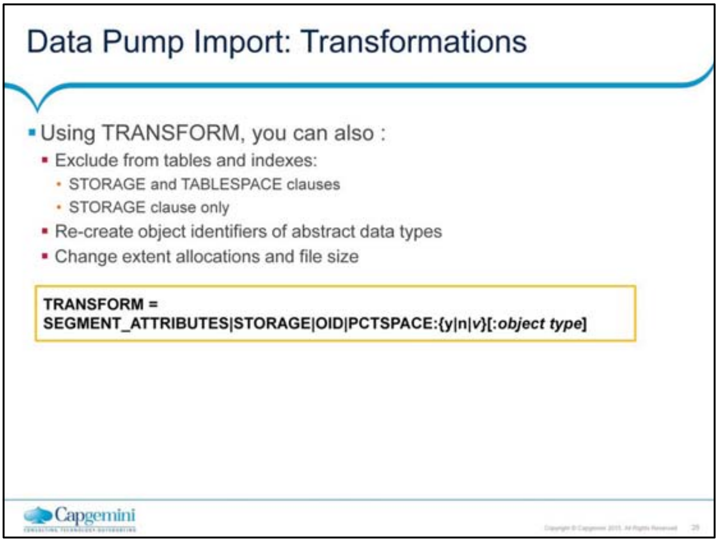
In interactive-command mode, the current job continues running, but logging to the terminal is stopped and the Import prompt is displayed. You can, for example, attach additional jobs to an executing or stopped job.



Data Pump Import: Transformations

Because object metadata is stored as XML in the dump file set, it is easy to apply transformations when DDL is being formed during import. Data Pump Import supports several transformations:

- REMAP_DATAFILE is useful when moving databases across platforms that have different file-system semantics.
- REMAP_TABLESPACE allows objects to be moved from one tablespace to another.
- REMAP_SCHEMA provides the old FROMUSER /TOUSER capability to change object ownership.



Data Pump Import: Transformations (continued)

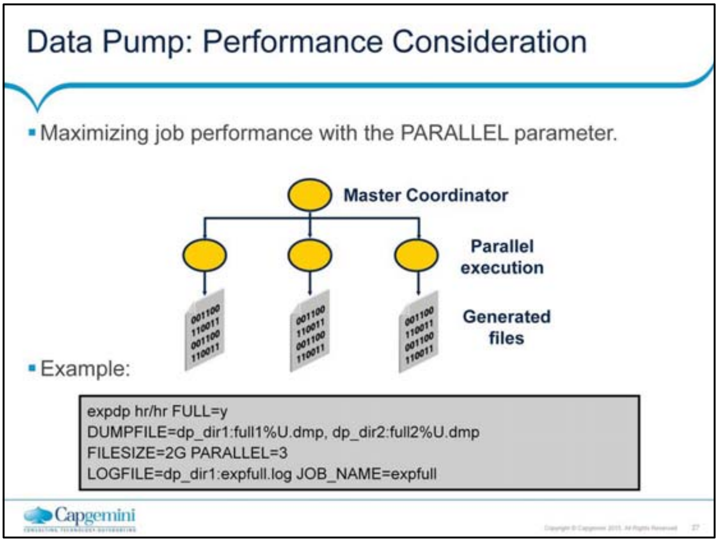
The TRANSFORM parameter enables you to alter the object creation DDL for specific objects or for all applicable objects being loaded. Specify the TRANSFORM parameter as shown in the slide. The following are possible options:

SEGMENT_ATTRIBUTES: If the value is specified as Y, segment attributes (physical attributes, storage attributes, tablespaces, and logging) are included.

STORAGE: If the value is specified as Y, the STORAGE clauses are included.

OID: You can use this parameter to determine whether the object ID (OID) of abstract data types is reused or created anew. If the value is specified as N, then the generation of the export OID clause for object types is suppressed. This is useful when you need to duplicate schemas across databases by using export and import, but you cannot guarantee that the object types will have identical OID values in those databases.

PCTSPACE: You can use the PCTSPACE parameter to reduce the amount of space required for tablespaces by performing a shrink operation on tablespace storage allocation. The value supplied for this transformation must be a number greater than zero. It represents the percentage multiplier used to alter extent allocations and the size of data files.



Data Pump: Performance Consideration

You can improve throughput of a job with the PARALLEL parameter. The parallelism setting is enforced by the master process, which allocates work to be executed to worker processes that perform the data and metadata processing within an operation. These worker processes operate in parallel. In general, the degree of parallelism should be set to more than twice the number of CPUs on an instance. To maximize parallelism, you must supply at least one file for each degree of parallelism. If there are not enough dump files, the performance will not be optimal because multiple threads of execution will be trying to access the same dump file. The degree of parallelism can be reset at any time during a job.

The example in the slide shows a full database export. All data and metadata in the database will be exported. Dump files (full101.dmp, full201.dmp, full102.dmp, and so on) will be created in a round-robin fashion in the directories pointed to by the dp_dir1 and dp_dir2 directory objects. For best performance, these should be on separate I/O channels. Each file will be up to 2 gigabytes in size, as necessary. Initially, up to three files will be created. More files will be created, if needed. The job and master table have the same name: expfull. The log file will be written to expfull.log in the dp_dir1 directory.

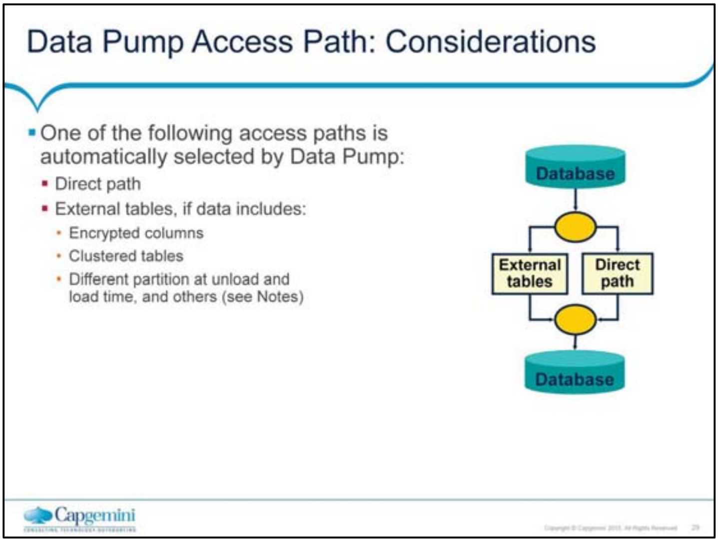
Performance Initialization Parameters

- Performance of Data Pump can be affected by:
  - DISK_ASYNCH_IO=TRUE
  - DB_BLOCK_CHECKING=FALSE
  - DB_BLOCK_CHECKSUM=FALSE
- The following should be set high to allow for maximum parallelism:
  - PROCESSES
  - SESSIONS
  - PARALLEL_MAX_SERVERS
- The following should be sized generously:
  - SHARED_POOL_SIZE
  - UNDO_TABLESPACE

Capgemini

Performance Initialization Parameters

You can try using the parameters (shown in the slide) to improve performance, although the effect may not be the same on all platforms. Additionally, the SHARED_POOL_SIZE and UNDO_TABLESPACE initialization parameters should be generously sized. The exact values will depend upon the size of your database.



Data Pump Direct Path: Considerations

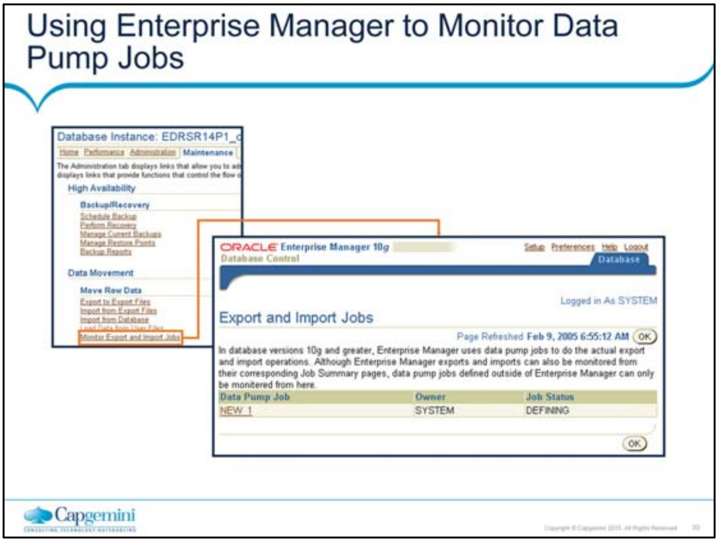
Data Pump automatically selects the most appropriate access method for each table.

Data Pump uses direct path load and unload when a table's structure allows it and when maximum single-stream performance is desired.

Data Pump uses external tables, if any of the conditions exist:

- Tables with fine-grained access control enabled in insert and select modes
- Domain index, which exists for a LOB column
- Tables with active triggers defined
- Global index on partitioned tables with a single-partition load
- BFILE or opaque type columns
- Referential integrity constraint
- VARRAY columns with an embedded opaque type

Note: Because both methods support the same external data representation, data that is unloaded with one method can be loaded using the other method.

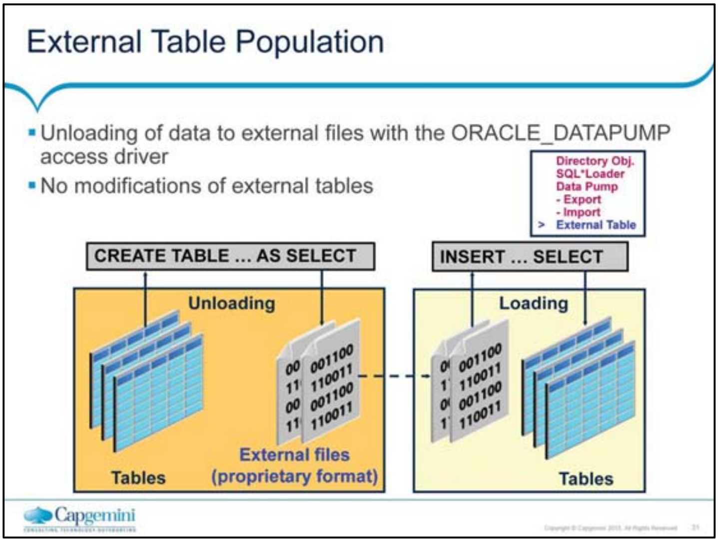


Using Enterprise Manager to Monitor Data Pump Jobs

You can use the Enterprise Manager graphical user interface (GUI) to monitor all Data Pump jobs, including those created by using the expdp or impdp command-line interfaces or by using the DBMS_DATAPUMP package.

You can view the current status of the job and also change the status to EXECUTE, STOP, or SUSPEND.

To access the Export and Import Jobs page, click the Monitor Export and Import Jobs link in the Move Row Data region of the Maintenance page.

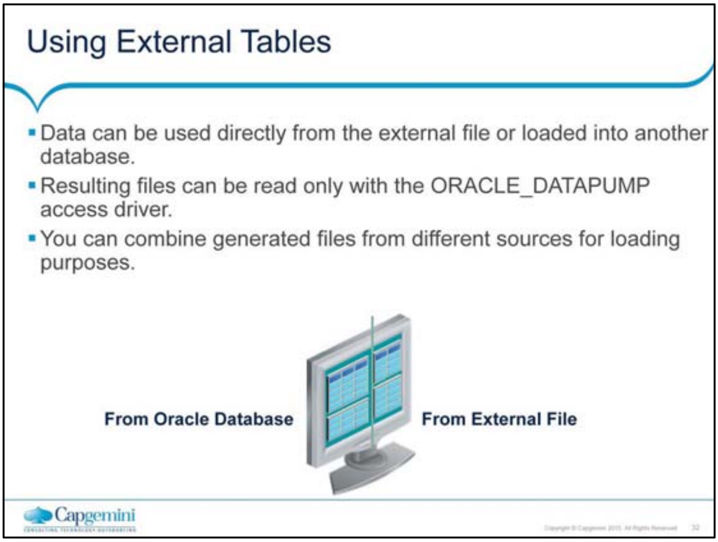


External Table Population

An “external table” is composed of proprietary format (that is, Direct Path API) flat files that are operating system independent. As data is extracted from the Oracle database and “unloaded” into files, it is transparently converted from its Oracle internal representation into an equivalent Oracle native external representation (that is, DPAPI).

You may use the CREATE TABLE AS SELECT command to populate an external table. After an external table has been created and populated, no rows may be added, updated, or deleted from the external table. Any attempt to modify the data in the external table fails. An external table may not have indexes.

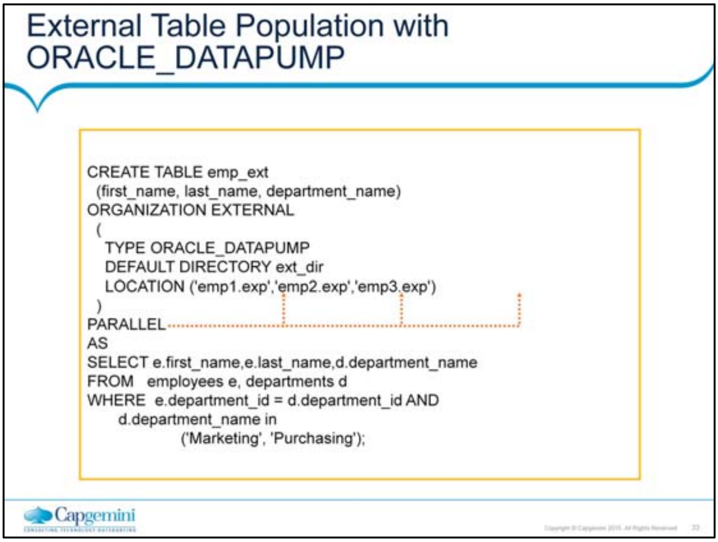
The Data Pump access driver enables the unloading and loading operations for external tables.



Using External Tables

The data files created for the external table can be moved and used as the data files for another external table in the same database or different database. They can be read only by the ORACLE_DATAPUMP access driver. You can choose whether your applications should directly access external tables with the SELECT command, or if the data should first be loaded into a target database.

Data files populated by different external tables can all be specified in the LOCATION clause of another external table. This provides an easy way of aggregating data from multiple sources. The only restriction is that the metadata for all the external tables must be exactly the same.



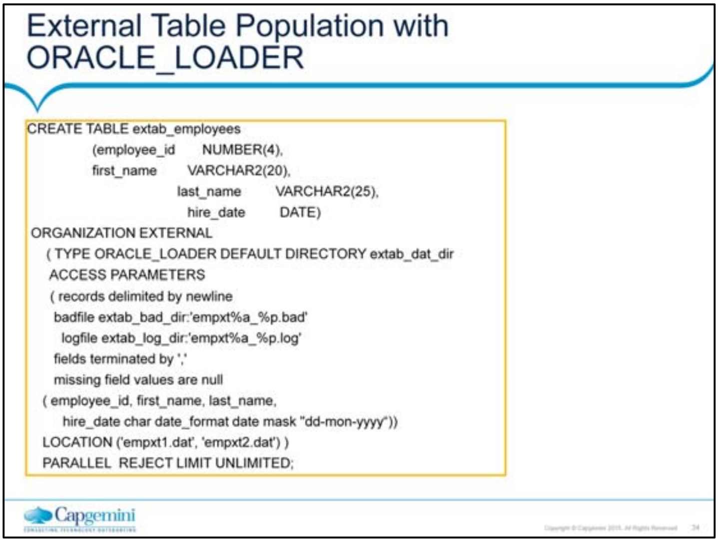
External Table Population with ORACLE_DATAPUMP

This example shows you how the new external table population operation can help to export a selective set of records resulting from the join of the EMPLOYEES and DEPARTMENTS tables.

Because the external table can be large, you can use a parallel populate operation to unload your data to an external table. As opposed to a parallel query from an external table, the degree of parallelism of a parallel populate operation is constrained by the number of concurrent files that can be written to by the access driver. There is never more than one parallel execution server writing into one file at a particular point in time.

The number of files in the LOCATION clause must match the specified degree of parallelism because each input/output (I/O) server process requires its own file. Any extra files that are specified are ignored. If there are not enough files for the specified degree of parallelism, the degree of parallelization is lowered to match the number of files in the LOCATION clause.

Note: For more information about the ORACLE_DATAPUMP access driver parameters, see the Oracle Database Utilities guide.



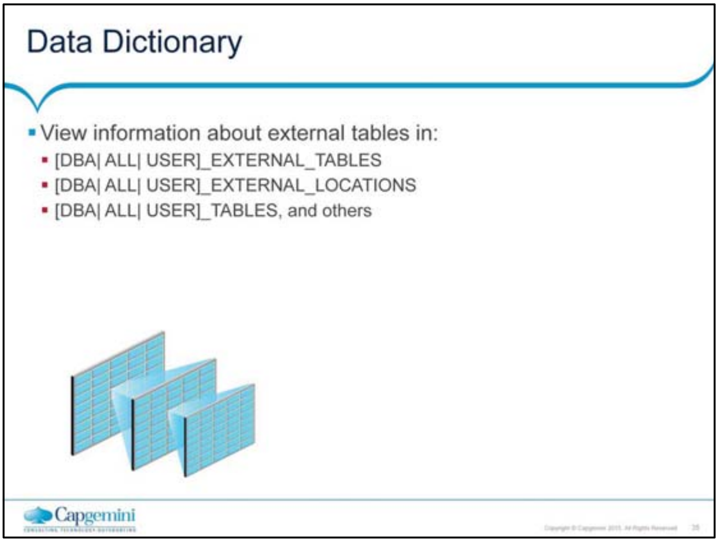
External Table Population with ORACLE_LOADER

The ORACLE_LOADER access driver uses the SQL*Loader syntax to create external tables.

The example shown in the slide assumes that three directory objects (extab_dat_dir, extab_bad_dir, and extab_log_dir) are created and mapped to existing OS directories, to which the user is granted access.

Tip: If you have a lot of data to load, enable PARALLEL for the load operation:

```
ALTER SESSION ENABLE PARALLEL DML;
```



Data Dictionary

[DBA| ALL| USER]_EXTERNAL_TABLES list the specific attributes of external tables in the database.
[DBA| ALL| USER]_EXTERNAL_LOCATIONS list the data sources for external tables.
[DBA| ALL| USER]_TABLES describe relational tables in the database.
[DBA| ALL| USER]_TAB_COLUMNS describe the columns of tables, views, and clusters in the database.

Summary

- In this lesson, you should have learned how to:
  - Describe available ways for moving data
  - Create and use directory objects
  - Use SQL*Loader to load data from a non-Oracle database (or user files)
  - Explain the general architecture of Data Pump
  - Use Data Pump Export and Import to move data between Oracle databases
  - Use external tables to move data via platform-independent files

Capgemini