

ORACLE 10G DATABASE ADMINISTRATION

Student Workbook

ORACLE 10G DATABASE ADMINISTRATION

Brian Peasland

Published by ITCourseware, LLC., 7245 South Havana Street, Suite 100, Centennial, CO 80112

Editor: Jan Waleri

Special thanks to: The many instructors, including Rob Roselius and Jeremy Russell, whose ideas and careful review have contributed to the quality of this workbook.

Copyright © 2011 by ITCourseware, LLC. All rights reserved. No part of this book may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying, recording, or by an information storage retrieval system, without permission in writing from the publisher. Inquiries should be addressed to ITCourseware, LLC., 7245 South Havana Street, Suite 100, Centennial, Colorado, 80112. (303) 302-5280.

All brand names, product names, trademarks, and registered trademarks are the property of their respective owners.

CONTENTS

| | |
|---|----|
| Chapter 1 - Course Introduction | 13 |
| Course Objectives | 14 |
| Course Overview | 16 |
| Using the Workbook | 17 |
| Suggested References | 18 |
| Chapter 2 - Overview of Oracle Database | 21 |
| ORACLE_HOME and ORACLE_SID | 22 |
| The Oracle Database vs. The Oracle Instance | 24 |
| Instance Memory Structures | 26 |
| Background Processes | 28 |
| Server Processes | 30 |
| Datafiles | 32 |
| Blocks, Extents, and Segments | 34 |
| Control Files | 36 |
| Redo Logs | 38 |
| The Oracle Architecture | 40 |
| SYS and SYSTEM Users | 42 |
| Labs | 44 |
| Chapter 3 - Starting and Shutting Down an Oracle Database | 47 |
| SYSDBA and SYSOPER System Privileges | 48 |
| Oracle Database Startup | 50 |
| Oracle Database Shutdown | 52 |
| Altering Database Availability | 54 |
| Suspending and Resuming a Database | 56 |
| Quiescing a Database | 58 |
| Tracking Database Activity - The Alert Log | 60 |
| Data Dictionary Views for Startup and Shutdown | 62 |
| Labs | 64 |

| | |
|---|-----|
| Chapter 4 - Using The Oracle Data Dictionary | 67 |
| Introducing the Data Dictionary | 68 |
| DBA, ALL, and USER Data Dictionary Views | 70 |
| V\$ Dynamic Performance Views | 72 |
| Using Oracle Documentation to Locate Data Dictionary Views | 74 |
| Combining Data Dictionary Views | 76 |
| Labs | 78 |
| Chapter 5 - Oracle Database Users and Schemas | 81 |
| Users and Schemas | 82 |
| Oracle Default Database Users | 84 |
| Oracle Sample Schemas | 86 |
| Creating Users | 88 |
| Altering and Dropping Users | 90 |
| Data Dictionary Views For Users | 92 |
| Labs | 94 |
| Chapter 6 - Oracle System Privileges | 97 |
| Overview of System Privileges | 98 |
| Granting and Revoking System Privileges | 100 |
| Data Dictionary Views for System Privileges | 102 |
| Roles | 104 |
| Creating and Removing Roles | 106 |
| Data Dictionary Views for Roles | 108 |
| Pre-defined Roles | 110 |
| User Group PUBLIC | 112 |
| Labs | 114 |
| Chapter 7 - Parameter Files | 117 |
| Oracle Database Parameters | 118 |
| The Parameter File (PFILE) and the Server Parameter File (SPFILE) | 120 |
| Dynamic vs. Static Parameters | 122 |
| Determining Current Parameter Settings | 124 |
| Benefits of SPFILES over PFILES | 126 |
| Creating an SPFILE from a PFILE and Back Again | 128 |
| Understanding Scope | 130 |
| Session-Level Parameters | 132 |
| Data Dictionary Views for Parameters | 134 |
| Labs | 136 |

| | |
|--|-----|
| Chapter 8 - Datafiles and Tablespaces | 139 |
| Datafiles Overview | 140 |
| Tablespaces Overview | 142 |
| SYSTEM and SYSAUX Tablespaces | 144 |
| Creating Tablespaces | 146 |
| Dictionary- and Locally Managed Tablespaces | 148 |
| Locally Managed Tablespace Extent Allocation | 150 |
| Temporary Tablespaces | 152 |
| Temporary Tablespace Groups | 154 |
| Default Tablespaces | 156 |
| Tablespace Quotas | 158 |
| Dropping and Altering a Tablespace | 160 |
| Renaming Tablespaces | 162 |
| Renaming or Relocating Datafiles | 164 |
| Bigfile Tablespaces | 166 |
| Data Dictionary Views for Datafiles and Tablespaces | 168 |
| Labs | 170 |
| Chapter 9 - Control Files | 173 |
| Control Files Overview | 174 |
| Database Parameters for Control Files | 176 |
| Backing Up Control Files | 178 |
| Restoring Control Files from Multiplexed Copies | 180 |
| Restoring Control Files from Backups | 182 |
| Moving Control Files | 184 |
| Data Dictionary Views for Control Files | 186 |
| Labs | 188 |
| Chapter 10 - Oracle Physical Structures — Online Redo Logs | 191 |
| Redo | 192 |
| Redo Log Files | 194 |
| Database Parameters for Redo | 196 |
| Sizing the Redo Log Files | 198 |
| How Many Redo Log Groups? | 200 |
| Creating Redo Logs | 202 |
| Removing Redo Logs | 204 |
| Renaming Redo Log Files | 206 |
| Forcing Log Switches | 208 |
| Archiving Redo Logs | 210 |
| Configuring Archive Log Mode | 212 |

| | |
|---|---------|
| Database Parameters for Archiving | 214 |
| Data Dictionary Views for Redo Logs | 216 |
| Labs | 218 |
| Chapter 11 - Oracle Physical Structures — Undo Segments | 221 |
| Undo Overview | 222 |
| Database Parameters for Undo | 224 |
| Creating Undo Tablespaces | 226 |
| Altering and Dropping Undo Tablespaces | 228 |
| Switching Undo Tablespaces | 230 |
| Undo Advisor | 232 |
| Data Dictionary Views for Undo | 234 |
| Labs | 236 |
| Chapter 12 - Segment Space Management | 239 |
| Blocks, Extents and Segments | 240 |
| Segment Space Management | 242 |
| Fragmentation | 244 |
| Coalescing Fragmented Space | 246 |
| Row Migration and Chaining | 248 |
| Manual Segment Space Management | 250 |
| Automatic Segment Space Management | 252 |
| Data Dictionary Views for Physical Objects | 254 |
| Labs | 256 |
| Chapter 13 - Tables | 259 |
| Tables Overview | 260 |
| Physical Properties of Heap Tables | 262 |
| Creating Tables | 264 |
| Table Storage and Logging | 266 |
| Altering Tables | 268 |
| Dropping Tables | 270 |
| The Recycle Bin | 272 |
| The TRUNCATE COMMAND | 274 |
| Temporary Tables | 276 |
| Clustered Tables | 278 |
| Index-Organized Tables | 280 |
| Creating an IOT | 282 |
| Data Dictionary Views for Tables | 284 |
| Labs | 286 |

| | |
|---|---------|
| Chapter 14 - Indexes | 289 |
| Indexes | 290 |
| B-tree Indexes | 292 |
| NULL Values and Indexes | 294 |
| NULL Values and Unique Indexes | 296 |
| Creating Indexes | 298 |
| Monitoring Index Usage | 300 |
| Dropping Indexes | 302 |
| Rebuilding and Moving Indexes | 304 |
| Coalescing Indexes | 306 |
| Bitmap Indexes | 308 |
| Special Index Types | 310 |
| Index Key Compression | 312 |
| Data Dictionary Views for Indexes | 314 |
| Labs | 316 |
| Chapter 15 - Constraints | 319 |
| Integrity Constraints | 320 |
| Constraint Names and Syntax | 322 |
| Constraint Checking | 324 |
| Managing Primary Key Constraints | 326 |
| Managing NOT NULL Constraints | 328 |
| Managing Check Constraints | 330 |
| Managing Foreign Key Constraints | 332 |
| Data Dictionary Views for Constraints | 334 |
| Labs | 336 |
| Chapter 16 - Views | 339 |
| Views | 340 |
| Creating and Replacing Views | 342 |
| Data Dictionary Views for Views | 344 |
| Security Through Views | 346 |
| Altering and Dropping Views | 348 |
| Dependencies and Views | 350 |
| Updating Data Through Views | 352 |
| Labs | 354 |

| | |
|---|-----|
| Chapter 17 - Object Privileges | 357 |
| Object Privileges | 358 |
| Granting and Revoking Object Privileges | 360 |
| Using WITH GRANT OPTION | 362 |
| Revoking and the GRANT OPTION | 364 |
| Data Dictionary Views for Object Privileges | 366 |
| Labs | 368 |
| Chapter 18 - Synonyms | 371 |
| Synonyms | 372 |
| Private and Public Synonyms | 374 |
| Creating and Dropping Synonyms | 376 |
| Security and Synonyms | 378 |
| Object Precedence | 380 |
| Labs | 382 |
| Chapter 19 - The Optimizer and Statistics | 385 |
| Optimizer Overview | 386 |
| Optimizer Statistics | 388 |
| Data Dictionary Views and Statistics | 390 |
| Collecting Statistics | 392 |
| Removing Statistics | 394 |
| Automated Statistics Collection | 396 |
| Labs | 398 |
| Chapter 20 - Oracle Net Services | 401 |
| Oracle Networking Explained | 402 |
| The Net Configuration Assistant | 404 |
| Configuring the Listener — GUI | 406 |
| Configuring the Listener — Manually | 408 |
| Configuring the Client — GUI | 410 |
| Configuring the Client — Manually | 412 |
| Labs | 414 |

| | |
|--|---------|
| Chapter 21 - Data Pump | 417 |
| Oracle Data Pump Architecture | 418 |
| Configuring for Data Pump | 420 |
| Command-Line Utilities | 422 |
| Exporting With expdp | 424 |
| Importing With impdp | 426 |
| Interactive Mode | 428 |
| Attaching to a Running Job | 430 |
| Parallel Data Pump | 432 |
| Data Dictionary Views for Data Pump | 434 |
| Labs | 436 |
| Chapter 22 - SQL*Loader and External Tables | 439 |
| SQL*Loader | 440 |
| SQL*Loader Control File | 442 |
| Loading Data with SQL*Loader | 444 |
| Conventional vs. Direct Path Loads | 446 |
| External Tables Overview | 448 |
| Creating an External Table | 450 |
| Creating a Writable External Table | 452 |
| Labs | 454 |
| Chapter 23 - Back Up Your Database | 457 |
| Physical Backups vs. Logical Backups | 458 |
| Offline Backup Basics | 460 |
| Oracle Offline Backup Steps | 462 |
| Other Backup Considerations | 464 |
| Advanced Backup Concepts | 466 |
| RMAN | 468 |
| Labs | 470 |
| Chapter 24 - Recover Your Database | 473 |
| Offline Recovery Basics | 474 |
| Restoration Using Offline Backups for a NOARCHIVE Database | 476 |
| Restoring Other Files | 478 |
| Other Recovery Considerations | 480 |
| Labs | 482 |

| | |
|---|---------|
| Chapter 25 - Oracle Enterprise Manager | 485 |
| Introducing Oracle Enterprise Manager | 486 |
| OEM Architecture and Physical Structure | 488 |
| Installing OEM When Using DBCA | 490 |
| Using EMCA to Install OEM | 492 |
| Managing OEM | 494 |
| Removing OEM | 496 |
| Troubleshooting OEM | 498 |
| Securing OEM | 500 |
| Starting OEM | 502 |
| Introducing the OEM Home Pages | 504 |
| Using OEM — Managing the Job Scheduler | 506 |
| Using OEM — Using Metrics, Alerts, and Thresholds | 508 |
| Automatic Workload Repository | 510 |
| The Automatic Database Diagnostic Monitor — ADDM | 512 |
| OEM and Advisors | 514 |
| The SQL Access Advisor | 516 |
| Appendix A - Installing Oracle Software | 519 |
| Optimal Flexible Architecture | 520 |
| OFA Directory Layouts | 522 |
| The Oracle Universal Installer | 524 |
| OUI Installation Modes | 526 |
| Preparing To Install Software | 528 |
| Starter Database | 530 |
| Installing the Oracle Software | 532 |
| Deinstalling Oracle Software | 534 |
| Appendix B - Creating a Database Using the DBCA | 537 |
| The DBCA | 538 |
| Select a Database Template | 540 |
| Specify the Database Name | 542 |
| Set System Account Passwords | 544 |
| Specify File Locations | 546 |
| Install Sample Schemas | 548 |
| Specify Storage Parameters | 550 |
| Create Your Oracle Database | 552 |

| | |
|---|---------|
| Oracle Net Configuration Assistant | 554 |
| Configure the Listener | 556 |
| Set TCP Parameters | 558 |
| Removing a Database with the DBCA | 560 |
| Appendix C - Manual Database Creation | 563 |
| Manual Database Creation | 564 |
| Prepare the System | 566 |
| Create the Parameter File | 568 |
| The CREATE DATABASE Statement | 570 |
| Manually Create the Database | 572 |
| Finalizing the Creation | 574 |
| SQL.BSQ File | 576 |
| Creating the Service on Windows | 578 |
| Appendix D - Auditing the Database | 581 |
| Auditing Explained | 582 |
| Data Dictionary Views for Auditing | 584 |
| Audit Trail, OS, and DB | 586 |
| Statement Auditing | 588 |
| Privilege Auditing | 590 |
| Object Auditing | 592 |
| Solutions | 595 |
| Index | 645 |

CHAPTER 1 - COURSE INTRODUCTION

COURSE OBJECTIVES

- * Create an Oracle Database using the Oracle Database Creation Assistant.
- * Manage an Oracle Database.
- * Configure various database structures.
- * Set up database and user security.
- * Add and administer users.
- * Monitor and tune main server areas.

COURSE OVERVIEW

- ✧ **Audience:** Database administrators, application developers, and system administrators.
- ✧ **Prerequisites:** *Introduction to Oracle 10g* or at least six months working in an Oracle technical environment. An understanding of relational database concepts, SQL, and PL/SQL programming skills are required. A solid understanding of Oracle schema is recommended.
- ✧ **Classroom Environment:**
 - A workstation per student.

USING THE WORKBOOK

This workbook design is based on a page-pair, consisting of a Topic page and a Support page. When you lay the workbook open flat, the Topic page is on the left and the Support page is on the right. The Topic page contains the points to be discussed in class. The Support page has code examples, diagrams, screen shots and additional information. **Hands On** sections provide opportunities for practical application of key concepts. **Try It** and **Investigate** sections help direct individual discovery.

In addition, there is an index for quick look-up. Printed lab solutions are in the back of the book as well as on-line if you need a little help.

The Topic page provides the main topics for classroom discussion.

The Support page has additional information, examples and suggestions.

Topics are organized into first (*), second (➤) and third (▪) level points.

JAVA SERVLETS

THE SERVLET LIFE CYCLE

- * The servlet container controls the life cycle of the servlet.
 - When the first request is received, the container loads the servlet class
 - The container uses a separate thread to call
 - The container calls the destroy ()
- As with Java's finalize () method, don't count on this being called.
- * Override one of the init () methods for one-time initializations, instead of using a constructor.
 - The simplest form takes no parameters.


```
public void init () { ... }
```
 - If you need to know container-specific configuration information, use the other version.


```
public void init (ServletConfig config) { ... }
```
 - Whenever you use the ServletConfig approach, always call the superclass method, which performs additional initializations.


```
super.init (config);
```

Page 16

Rev 2.0.0

© 2002 ITCourseware, LLC

Pages are numbered sequentially throughout the book, making lookup easy.

CHAPTER 2

SERVLET BASICS

Hands On:

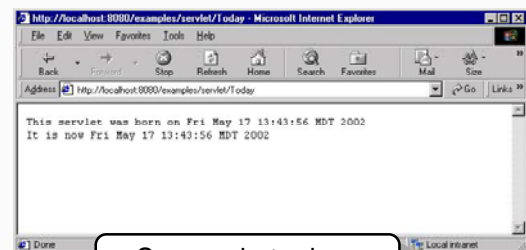
Add an init () method to your *Today* servlet that initializes along with the current date:

```
Today.java
...
public class Today extends GenericServlet {
    private Date bornOn;
    public void service(ServletRequest request,
        ServletResponse response) throws ServletException, IOException
    {
        ...
        Servlet was born on " + bornOn.toString();
        " + today.toString();
    }
}
```

Code examples are in a fixed font and shaded. The on-line file name is listed above the shaded area.

Callout boxes point out important parts of the example code.

The init () method is called when the servlet is loaded into the container.



Screen shots show examples of what you should see in class.

© 2002 ITCourseware, LLC

Page 17

SUGGESTED REFERENCES

Dawes, Chip, Bob Bryla, Joseph C. Johnson, and Matthew Weishan. 2004. *OCA: Oracle 10g Administration I Study Guide (1Z0-042)*. Sybex, Inc., Alameda, CA. ISBN 0782143679.

Freeman, Robert G. 2004. *Oracle Database 10g New Features*. Oracle Press, Emeryville, CA. ISBN 0072229470

Greenwald, Rick, Robert Stackowiak, and Jonathan Stern. 2004. *Oracle Essentials*. O'Reilly & Associates, Sebastopol, CA. ISBN 0596005857

Freeman, Robert. 2004. *Portable DBA: Oracle*. McGraw-Hill Osborne Media. ISBN 0072229802

metalink.oracle.com — Requires Oracle product support contract

otn.oracle.com — Oracle Technology Network

tahiti.oracle.com — Oracle Database Documentation

www.quest-pipelines.com/pipelines/dba — Online discussions and resources

www.dbasupport.com

www.hot-oracle.com

news:comp.databases.oracle

news:comp.databases.oracle.server

news:comp.databases.oracle.misc

If you haven't done so before, now is the time to immerse yourself in the Oracle Database Online Documentation. You may have received this on CD-ROM with your Oracle distribution. If not, you can access it online at Oracle's web site. You can also download the complete set and install it locally on your system for quick access.

An easy way to find it is to go to:

<http://tahiti.oracle.com/>

Find the documentation for your version of Oracle.

If you have web access in the classroom, open a browser now and find the Oracle Database Online Documentation. Set a browser bookmark, and have it at hand throughout this class.

CHAPTER 2 - OVERVIEW OF ORACLE DATABASE

OBJECTIVES

- * Specify the location of your Oracle database software installation.
- * Identify the processes that make up the Oracle instance.
- * Identify the files that comprise your Oracle database.
- * Explain the basic storage allocation scheme used in Oracle datafiles.
- * Explain the architecture and basic operation of the Oracle database system.
- * Appropriately use the **SYS** and **SYSTEM** accounts.

ORACLE_HOME AND ORACLE_SID

- ✴ When you install the Oracle Database software, the software is installed in one directory, the *Oracle Home* directory, with many subdirectories.
 - Just about every Oracle Database program you run needs to know the location of this directory, under which are configuration files, error-message files, program libraries, etc.
 - The *Oracle Home* directory is typically located under a parent directory called the *Oracle Base* directory, under which other versions of Oracle Database, or Oracle non-database products may also be installed.
 - Each version of Oracle Database software you install on a host has its own *Oracle Home*.
 - On UNIX/Linux, set the *Oracle Home* location in your **ORACLE_HOME** environment variable, where programs you run will reference it.
- ✴ Each database on a host must be given a unique name or system identifier.
 - The system identifier is called the *SID* or Oracle *SID*.
 - On UNIX/Linux, set the *SID* in the **ORACLE_SID** environment variable.
- ✴ On Windows, Oracle applications find the **ORACLE_HOME** and **ORACLE_SID** in Windows Registry values created when you install Oracle (for **ORACLE_HOME**) or create a database (for **ORACLE_SID**).
- ✴ Both Windows and UNIX use a **PATH** environment variable to locate programs you run.
 - The *bin/* directory under **ORACLE_HOME** should be in your **PATH**.

The Oracle Home directory has many important subdirectories, including:

| Subdirectory | Contents |
|-----------------|--|
| bin | The executable programs |
| demo | Scripts to set up demo schema objects (the <i>demo</i> directory is only loaded if you specify it during installation, so this may or may not actually have files in it) |
| lib | The shared library files other programs use |
| db | Parameter and password files (UNIX/Linux) |
| database | Parameter and password files (Windows) |
| network | The network component and configuration files |
| rdbms | Database setup scripts |
| sqlplus | SQL*Plus component and configuration files |

When you install Oracle on Windows, the **%ORACLE_HOME%** environment variable is automatically set up for you. In UNIX and Linux environments, the **\$ORACLE_HOME** environment variable will need to be configured manually for your session. You can optionally set this parameter in your *.cshrc*, *.login*, or *.profile* files, depending on your UNIX/Linux shell.

```
ORACLE_HOME=/u01/app/oracle/product/10.1.0/db_1
export ORACLE_HOME
```

The **PATH** environment variable lets your command line sessions find executable programs without requiring you to type the directory path. If you install Oracle on Windows, the **PATH** environment variable has the **%ORACLE_HOME%\bin** directory automatically added to it. In UNIX and Linux environments, you will have to make this modification yourself:

```
PATH=$PATH:$ORACLE_HOME/bin
```

The Oracle Base directory is part of the Optimal Flexible Architecture (OFA) system. OFA is a methodology designed for maintaining separation between multiple Oracle versions and products. On UNIX/Linux, while not required, the location of the Oracle Base directory is usually set in the environment variable **ORACLE_BASE**.

For some UNIX/Linux systems, the **LD_LIBRARY_PATH** environment variable should include the *lib/* subdirectory under **ORACLE_HOME**.

THE ORACLE DATABASE VS. THE ORACLE INSTANCE

- ✳ A working Oracle system consists of two major components: a database and an instance.
- ✳ A collection of physical files makes up the Oracle database.
 - These files include:
 - Datafiles
 - Redo log files
 - Control file
 - Additional files, which are critical to running your Oracle server include:
 - Parameter file
 - Password file
 - Network configuration files
- ✳ The collection of running background processes and allocated memory makes up an Oracle instance.
 - An instance must be started to permit access to the database files.
 - An instance manages one and only one database.
 - It is possible for more than one instance to access the same database in a clustered environment, running Oracle's Real Application Clusters (RAC) software.

When you start an instance, Oracle will allocate memory to be shared by all Oracle processes and users. Oracle will also start various processes that manage and control the database. When you shut down the instance, the allocated memory and processes will cease to exist on the server, but the database files will still remain.

Hands On:

At this point, if you do not already have a database on your classroom system, your instructor might want to walk you through creating one using the DBCA right now. For the remainder of the course, you will use the database you create here.

INSTANCE MEMORY STRUCTURES

- * The *System Global Area* (SGA) is an area of shared memory accessed by all database users.
 - Each instance has its own SGA.
 - The SGA is allocated at instance startup and terminated at instance shutdown.
 - The SGA is broken down into different memory areas, including:
 - The Buffer Cache
 - The Redo Log Buffer
 - The Shared Pool
 - Additional optional areas (e.g. Large Pool, Java Pool)

Allocating too much memory for the SGA can cause poor performance if it forces the server host to perform memory page swapping. Allocating too little memory for the SGA can cause poor instance performance as well. It is important to find a good balance for your SGA allocation. Oracle's default recommendation is to assign less than 40% of system RAM.

There are multiple areas in the SGA. Each component of the SGA performs a specific function. The components of the SGA include:

| SGA Component | Function |
|-----------------|--|
| Buffer Cache | Memory access is quicker than access from disk. If data has been accessed multiple times, then it is beneficial to place that data in a cache for quicker future access. |
| Redo Log Buffer | As transactions make data modifications, a record of those changes are written to this buffer for quick transaction logging. |
| Shared Pool | The Shared Pool contains subcomponents to help manage SQL statements and a cache for the Data Dictionary objects. |
| Java Pool | Manages Java stored procedures. |
| Large Pool | Used for moving large blocks of data around the system. |

BACKGROUND PROCESSES

- ✴ When the instance starts, a number of background processes, each having a specific task, are also started.
 - On Windows, these run as threads within a single process.
- ✴ Some background processes are required, and the instance will terminate if they are not running.
 - Required background processes include:
 - Database Writer (DBW n)
 - System Monitor (SMON)
 - Process Monitor (PMON)
 - Log Writer (LGWR)
 - Checkpoint Process (CKPT)
- ✴ Some background processes are optional, such as:
 - Manageability Monitor (MMON)
 - Memory Manager (MMAN)
 - Archiver Processes (ARC n)
 - Queue Monitor Processes (QMN n)
 - Recoverer (RECO)
 - If an optional process is not running, the instance can remain up, but functionality may be lost.

The Oracle database has a lot of activity going on behind the scenes. Most database users are not even aware that these processes exist. The required background processes are:

| Process | Function |
|---------------------------|---|
| System Monitor (SMON) | Monitors the instance. Performs instance recovery on startup if needed. Cleans up temporary segments no longer needed, and coalesces free space. |
| Process Monitor (PMON) | Monitors users' sessions. Cleans up after a session if it abnormally terminates. |
| Log Writer (LGWR) | Writes redo log entries from the Log Buffer cache to the online redo log files. |
| Database Writer (DBWn) | Writes dirty buffers from the Buffer Cache to the datafiles. CKPT can prompt DBWn to begin writing. There can be multiple processes to handle a high I/O load. |
| Checkpoint Process (CKPT) | Performs a database checkpoint. On a checkpoint, all dirty buffers are written to disk. The datafiles are modified to include the time of the last checkpoint in the datafile header. |
| Recoverer Process (RECO) | Recovers from failed transactions involving more than one instance, in a distributed environment. |

Other background processes are optional. You must configure the database specifically to start the optional background processes:

| Process | Function |
|--------------------------------|---|
| Job Queue Processes (Jnnn) | Manages scheduled jobs within the database. |
| Coordinator Job Queue (CJQn) | The coordinator, or boss, of the Jnnn processes. |
| Archiver Processes (ARCn) | Responsible for archiving the contents of an online redo log. |
| Queue Monitor Processes (QMNn) | Used to manage Oracle's Advanced Queueing |
| Memory Manager (MMAN) | Manages the automatic sizing of SGA components. |
| Recovery Writer (RVWR) | Responsible for writing flashback logs. |
| Manageability Monitor (MMON) | Writes statistics required by the Automatic Workload Repository (AWR) on a regular basis. |
| Change Tracking Writer (CTWR) | Writes block-level change tracking information used for Recovery Manager's incremental backups. |

SERVER PROCESSES

- * When a user application connects to Oracle, a process must run on the server to perform the database work for that application.
- * The *server process* carries out two basic functions for the user that is connected to the database:
 - Parse and execute a user's SQL statements.
 - Fetch data from the datafiles if the data is not found in the Buffer Cache.
- * The server process also allocates a portion of memory called the *Program Global Area (PGA)*, containing such elements as:
 - SQL work areas
 - Session memory
 - Cursors
- * Each server process has its own PGA inaccessible by other server processes.
 - The PGA is typically allocated at session startup and terminated when the session ends; PGA size will change automatically as the requirements of the server process change.
- * Oracle supports two types of server process: Dedicated Server processes and Shared Server processes.
 - With *Dedicated Server* processes (the default configuration), each connected client has its own server process.
 - In a *Shared Server* process, users share a pool of processes.
 - The shared server process configuration is used when you need to support a high number of concurrent users.
 - The information in the PGA is not shared between database users, and will be cached in the Large Pool.

Out of the box, the Oracle instance is configured to run with dedicated server processes. In dedicated server mode, there is a one-to-one correspondence between each user's connection and a server process. The server process is started by Oracle in response to a user process connection. Each user's server process will have its own PGA.

If you have a very large number of concurrent users, then the allocation of each PGA can lead to a very large amount of memory allocated for those user's sessions. Such a large allocation of total PGA can quickly exceed total physical memory (RAM) in the database server. To alleviate the problem, you can configure the Oracle database to run with shared server processes. With a shared server configuration, a pool of server processes is created. Several sessions will share the same server process. They will share the same PGA. But each session will only be able to see its own information in that PGA.

Parsing is the procedure performed by Oracle to analyze a SQL statement and determine an optimum execution plan.

Background processes also have their own PGA memory regions.

DATAFILES

- ✴ Datafiles hold the permanent contents of the Oracle database.
- ✴ The Oracle datafile is like other physical files on your database server.
 - Oracle datafiles are binary files.
 - Other software should not modify the contents of the Oracle datafiles.
- ✴ When the instance is stopped normally, the datafiles are the persistent mechanism that stores the current state of your data.
- ✴ A session's server process is responsible for reading data blocks from the datafiles into the buffer cache.
- ✴ The *Database Writer* process (DBWn) is responsible for writing changed data from the SGA Buffer Cache to the datafiles.
 - The DBWn operates in a "lazy" style, writing only modified blocks that haven't been recently used.
 - The DBWn writes changed data to the datafiles in batches, not immediately after each has been modified.

Remember that the database is the collection of datafiles, control files, and online redo log files.

When the instance is running, data lives in memory, in the cache structures. Users access the data in the cache, not directly from the datafiles.

When a user requests data, and that data is not in any of the buffer caches, the user's server process reads the data from the datafile into the buffer cache. At this point, the user can access the data. The user never directly accesses the datafile, but the user's server process does read the data from the datafile.

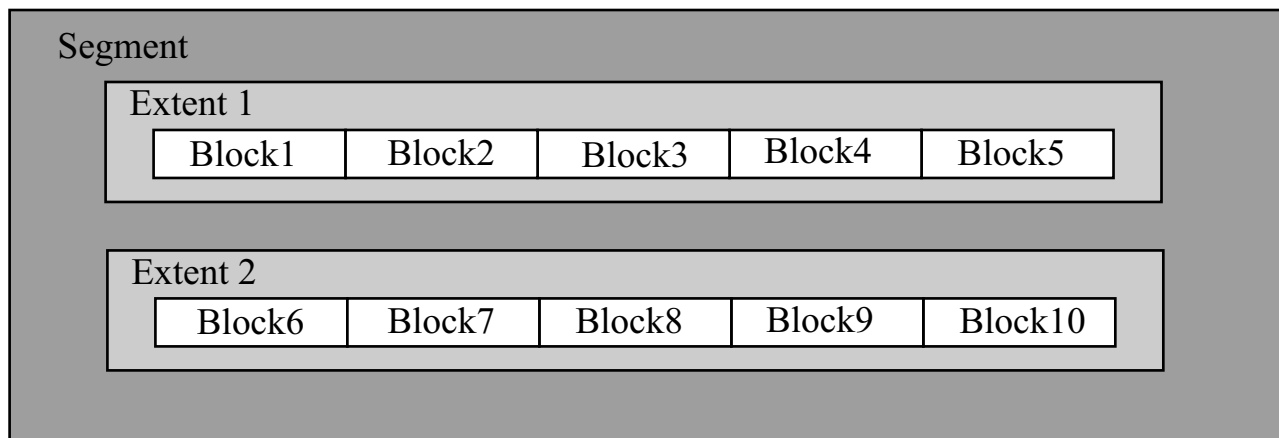
The Database Writer does as little work as possible, while remaining constant to its purpose of ensuring that changes to data blocks are permanently saved.

BLOCKS, EXTENTS, AND SEGMENTS

- * An Oracle *block* is the smallest unit of physical storage.
 - Each block is a physically contiguous range of disk sectors.
- * An *extent* is a contiguous allocation of data blocks.
- * Any database object that requires physical storage is called a *segment*.
 - Database objects that do not require physical storage are simply called *objects*.
- * A segment is made of at least one extent, which in turn, is made of at least one block.
- * A block can belong to one and only one extent.
- * An extent can belong to one and only one segment.
- * When a segment is created, at least one extent is allocated.
 - This first extent is called the *initial extent*.
 - Subsequent extents are called *next extents*.

The database block is either 2K, 4K, 8K, 16K, or 32K in size. The default size for 10g is 8K. The space for the block must be physically contiguous, or adjoining, on the disk unit. You cannot have a block that is split into more than one piece. Since the block is the smallest unit of storage, it is also the smallest unit of I/O the database performs. For instance, if the block is 4K in size, then one read or write operation that the database performs will read or write in 4K chunks.

An extent is a contiguous allocation of blocks for an object in the database. If you request a 64K extent and the block size is 4K, then the database must find 16 blocks, all physically adjacent to each other. Just like a block cannot be split into more than one piece, the extent cannot be split into more than one piece.



A segment is any database object that requires physical storage:

- Tables
- Partitions of a Table
- Indexes
- Partitions of an Index
- Clusters
- Materialized Views
- Materialized View Logs

Internal storage also utilizes the segment concept, including:

- Undo Segments
- Rollback Segments

CONTROL FILES

- ✴ The *control file* is a small binary file containing information about the database:
 - The database name
 - The names and locations of all datafiles and online redo logs
 - Database state information
- ✴ The control file is the master file of the database.
 - Without the control file, you will not be able to start the instance.
 - If the control file is lost, an open instance will crash.
- ✴ Normally, we have Oracle *multiplex* the control file: maintain multiple, identical copies of the same file.
 - The multiplexed copies should reside on different disk volumes for safety and security.
 - If some failure occurs on one disk unit which compromises one control file, one of the multiplexed copies can be used to restore the bad control file.

Your database control file contains:

- The database name
- Names of all tablespaces and their datafiles
- Log history
- Database creation date
- Names and locations of all datafiles and online redo logs
- Log archive history
- Checkpoint information
- Current log sequence number
- RMAN backup sets and pieces

The control file should be multiplexed in all of your databases. The multiplexed copies of the control file should reside on separate disk volumes. If you lose a disk volume, you do not lose your complete control file information because you have another copy elsewhere.

Disk mirroring is not a substitute for multiplexing. With *disk mirroring*, the OS maintains duplicate copies, but all actions on one copy are performed on the mirror copy, as well. For instance, if a System Administrator accidentally deletes a control file, the mirrored copy is deleted. Oracle maintains identical copies of a multiplexed file, but, to the OS, the files are two distinctly different files.

Multiplexed control files should be used even when the primary storage uses Oracle's "software RAID" architecture called Automatic Storage Management (ASM).

REDO LOGS

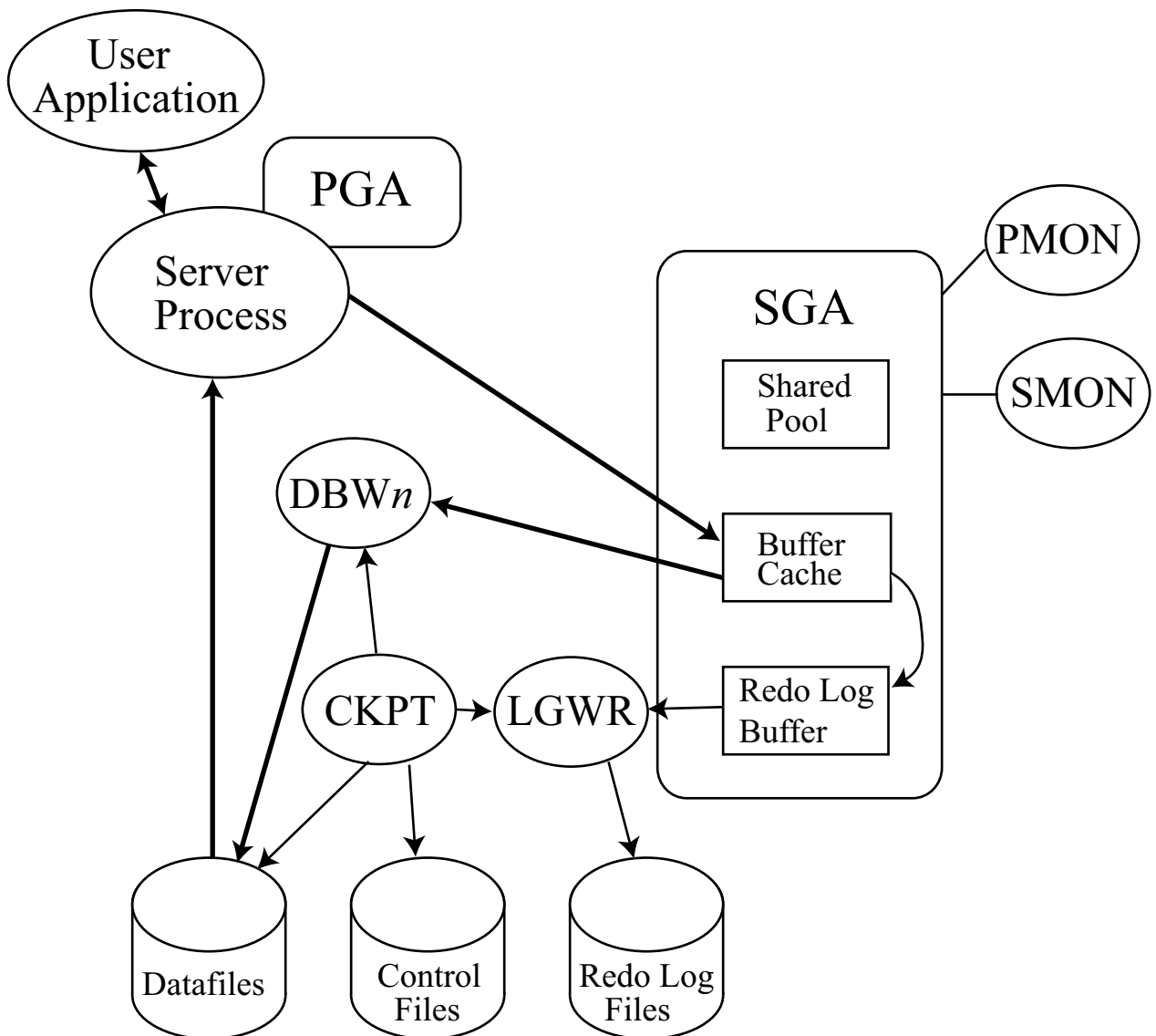
- ✴ The Oracle database maintains a record of all transactions called the *redo log*.
- ✴ Redo is the main recovery mechanism.
 - If the instance crashes before modified data blocks have been written back to their datafiles, Oracle automatically recovers the lost changes by reading the redo log.
 - If a failure occurs to a datafile, you can restore the datafile from a backup and use the redo information to reapply, or redo, the transactions that have occurred since the backup was taken.
- ✴ Redo records are initially written to a memory structure called the *redo buffer*.
 - The *Log Writer* (LGWR) process is responsible for writing the contents of the redo buffer to the online redo log files.

Whenever a transaction commits, the redo records must be written to the online redo log files. A commit is not complete until the redo information has been written to the online redo log files. Notice that a commit does not have to write to the database datafiles to complete.

Just as with control files, you should multiplex your online redo log files. To multiplex a redo log file, you create a group with multiple members. Multiplexing will safeguard you in case you lose one member of the group: you still have the redo records in another member of the group.

THE ORACLE ARCHITECTURE

- ✴ Now that we have seen the memory components, files, and processes, let's put it all together to see how it works.
 - A user connects to the instance; a server process starts and allocates its PGA.
 - The user issues a query; the server process checks the Shared Pool to see if the query has previously been issued.
 - If not, it parses, compiles, and caches the query in the Shared Pool.
 - The server process executes the query, which requests data not in the Buffer Cache.
 - The server process reads the blocks containing the data from the datafiles into the Buffer Cache.
- ✴ The user issues a DML statement; the server process copies the blocks containing the data to the buffer cache (if they aren't already there), as well as blocks from an undo segment.
 - The current state of the records is first written to the undo blocks.
 - Redo records of the changes and of the undo information are written to the redo buffer.
 - The changes are applied to the block in the buffer cache.
 - Frequently, the redo log writer process (LGWR) writes the redo records to the online redo log files.
 - Separately, the checkpoint process (CKPT) performs a checkpoint.
 - This prompts the database writer process (DBWn) to write the changed data blocks in the buffer cache back to the datafiles.
 - The checkpoint process modifies the control files to record that a checkpoint has occurred, and updates the datafile headers.



SYS AND SYSTEM USERS

- ✳ Two privileged users are created when the database is created: **SYS** and **SYSTEM**.
- ✳ The **SYS** user owns the Data Dictionary where the structural control information in the database is stored.
 - Only the Database Administrator should have the password for the **SYS** user.
 - While the **SYS** user can perform all database administration functions, the DBA should use a different account for administrative tasks.
 - In order to acquire **SYS** privileges, the keywords **AS SYSDBA** must be appended to the login connection string.
- ✳ Use the **SYSTEM** account for most administration tasks.
 - This does not preclude you from creating other users for administration tasks, including privileged operations usually requiring **SYS** privilege.

Oracle Command Tools for DBAs

Oracle 10g is installed with SQL*Plus, an interactive SQL entry program. To connect to a database, provide an argument as:

```
user/password@SID
```

On Windows systems, SQL*Plus has both character and graphical versions.

All database administration commands can be executed by issuing the appropriate SQL commands through SQL*Plus.

Oracle 10g is also installed with an optional browser-based management console, Oracle Enterprise Manager (OEM).

LABS

- ❶ If you don't already have your own database for use in this class, use the DBCA to create a new database. You will use this database for the remainder of the class. Please refer to Appendix C and the online documentation.
- ❷ When you create your database, specify a password for the **SYS** and **SYSTEM** users in the DBCA wizard. Using that password, connect to your database for the first time as **SYS** and as **SYSTEM**.

```
sqlplus /nolog  
connect sys/password as sysdba  
connect system/password
```

Note:

When you connect as **SYS**, you are required to connect with the **SYSDBA** or **SYSQUERY** privilege.

CHAPTER 8 - DATAFILES AND TABLESPACES

OBJECTIVES

- ✱ Describe the purpose and use of a tablespace and its datafiles.
- ✱ Describe the different types of tablespaces and discuss when they will be used.
- ✱ Create sample datafiles and tablespaces.
- ✱ Effectively administer tablespaces including dropping, renaming, and changing their availability.

DATAFILES OVERVIEW

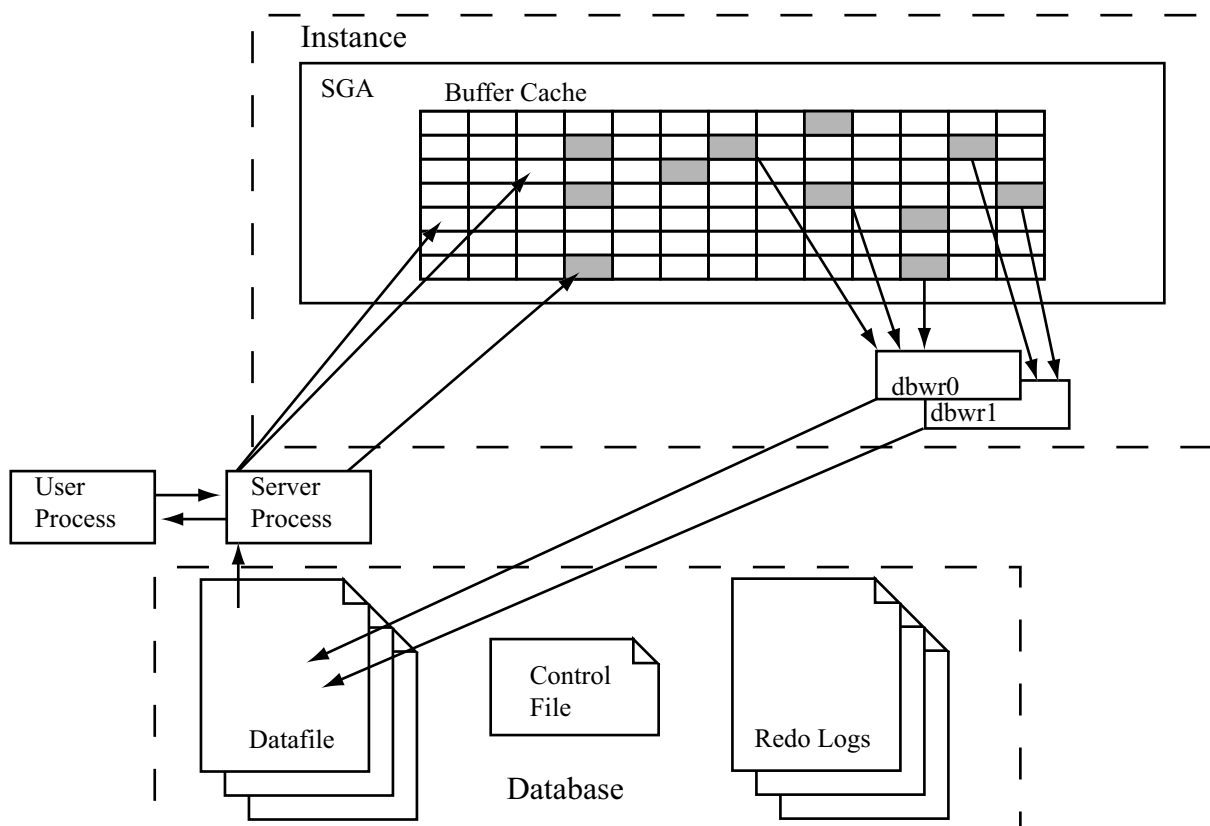
- * Datafiles are the physical structures that hold the contents of the Oracle database.
 - When the instance is stopped normally, the datafiles are the persistent mechanism that stores your data.
- * The datafiles for an Oracle database have the following properties:
 - Every Oracle 10g database has at least two datafiles.
 - Every pre-10g database has at least one datafile.
 - A typical database has many datafiles.
 - A datafile can be associated with only one Oracle database.
- * One or more Database Writer processes (DBWn) are responsible for writing changed data to the datafiles.
 - For efficient database operations, changed data is not written to the datafiles immediately.
 - Changed data is written to the datafiles when a checkpoint occurs or when the database is stopped normally.
 - In the event of an instance failure, the information in the online redo logs is used to reconstruct committed transactions whose changes were not written to the datafiles by the DBWn processes.
- * A session's server process is responsible for reading data from the datafiles.
 - Data that is read is stored in the instance's Buffer Cache.

The Oracle datafiles are binary files that comprise most of the disk space requirements of your Oracle database. Other software should not modify the contents of the Oracle datafiles.

Though it is not normally possible for two databases to share the same datafile, it is possible for multiple instances to access the same database. Remember that the database is the collection of datafiles, control files, and online redo log files. The instance is the collection of memory structures and processes. In order for multiple instances to use the same database, you must be running Oracle's Real Application Clusters (RAC). RAC is used for high availability and high performance.

The user process never directly accesses the datafile. When a user requests data, and that data is not in any of the buffer caches, the user's server process reads the data from the datafile into the Buffer Cache. At this point, the user can access the data.

While DBW_n writes changed data to the datafiles, if the data was not changed then there would be no need to modify that data in the datafile. The DBW_n writes changed data to the datafiles in batches, rather than immediately after a change is made. Whenever the database performs a checkpoint, DBW_n writes all changed data to the datafiles. Whenever you perform a **SHUTDOWN NORMAL**, **SHUTDOWN IMMEDIATE**, or **SHUTDOWN TRANSACTIONAL**, the DBW_n process writes changed data to the datafiles. Multiple DBW_n processes can be configured to handle a high-volume I/O load.



TABLESPACES OVERVIEW

- * *Tablespaces* are logical storage units that are used to group related objects together.
- * A database is logically divided into one or more tablespaces.
 - Oracle 10g databases must have, at a minimum, the **SYSTEM** and **SYSAUX** tablespaces.
 - Pre-Oracle 10g databases only require the **SYSTEM** tablespace.
- * The tablespaces for an Oracle database have the following properties:
 - A tablespace can belong to only one database.
 - A tablespace is made of one or more datafiles.
 - A datafile can belong to only one tablespace.
 - The total space allocated for a tablespace is the sum of the sizes of all of the tablespace's datafiles or tempfiles.
 - A tablespace is the unit of backup, so tables that should be associated for this purpose should share a tablespace.
- * A tablespace has one of the following types:
 - Permanent — Persistent object segments
 - Temporary — Transient session data
 - Undo — Previous versions of data for read consistency or recovery

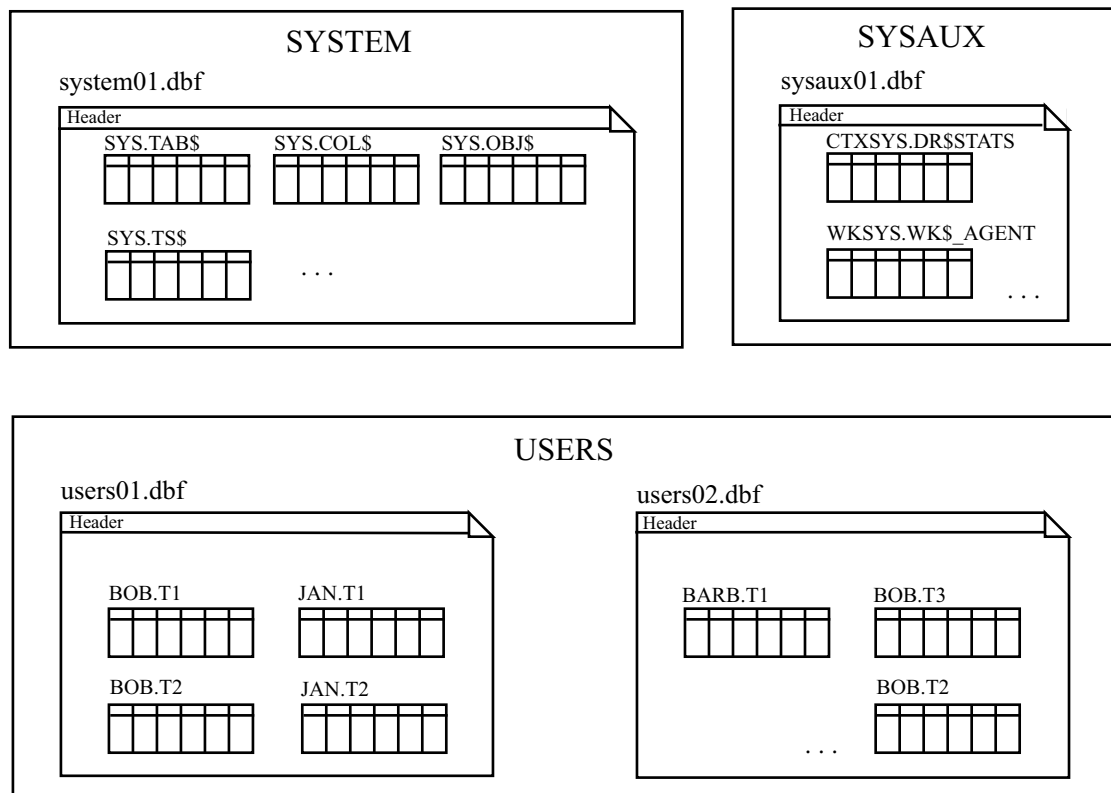
A tablespace is a logical container for any database object that requires a segment.

Since this is a logical grouping of objects, the way you set up your tablespaces should make sense to your database and your data. For instance, you might put one user's data in one tablespace and another user's data in another tablespace. You could put small tables in one tablespace and large tables in another tablespace. You could put one application's data in one tablespace and another application's data in another tablespace.

The use of logical tablespaces as separate from physical datafiles delegates the management of disk space to the software, rather than developers. Using this model, applications can easily be moved to different operating environments, or the physical storage layout can be reorganized, without any need to alter software.

control.ctl - Control File

```
...
Tablespace: SYSTEM
Datafile: /u01/app/oracle/oradata/inst1/system01.dbf
Tablespace: SYSAUX
Datafile: /u01/app/oracle/oradata/inst1/sysaux02.dbf
Tablespace: USERS
Datafile: /u01/app/oracle/oradata/inst1/users01.dbf
Datafile: /u01/app/oracle/oradata/inst1/users02.dbf
...
```



SYSTEM AND SYSAUX TABLESPACES

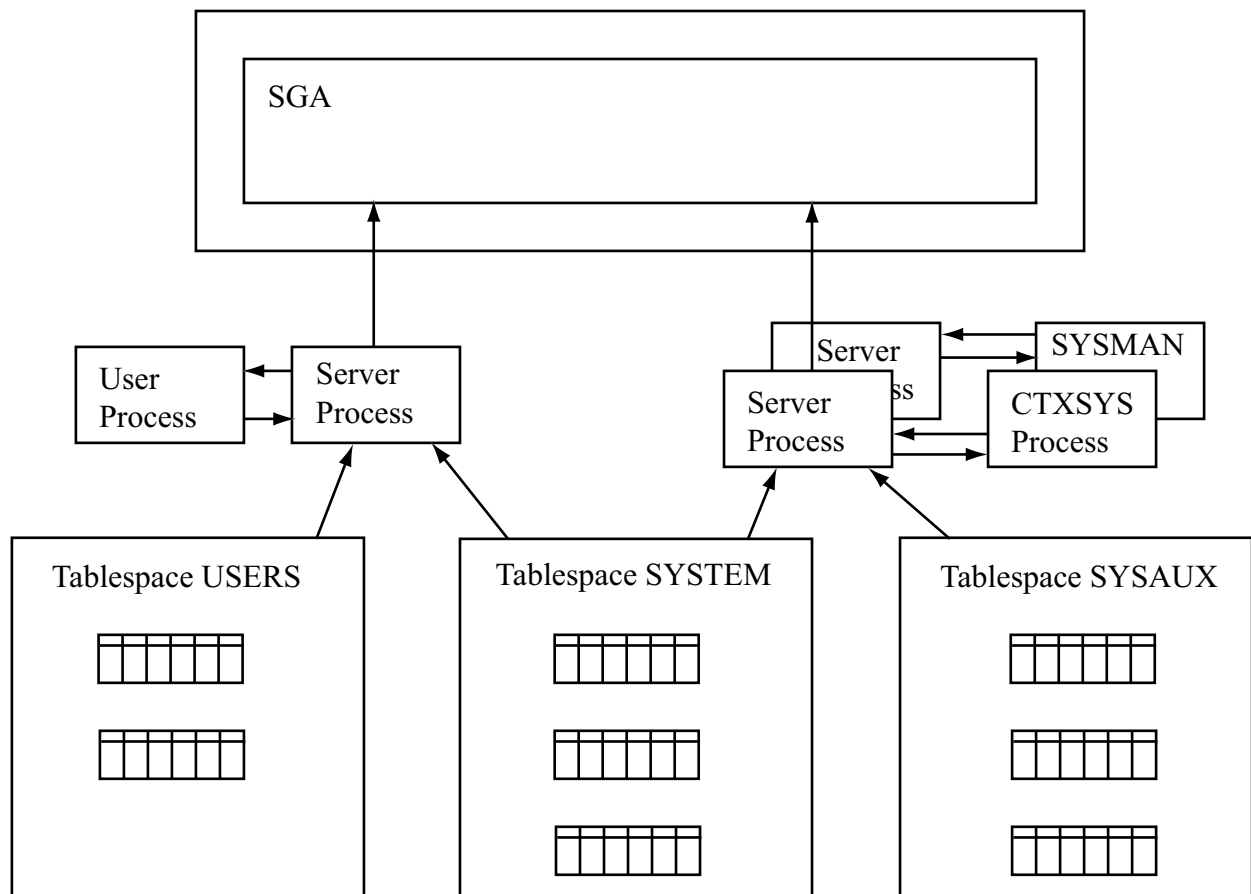
- ✳ All Data Dictionary tables are stored in the **SYSTEM** tablespace.
 - The **SYSTEM** tablespace is created when the database is created.
 - The **SYSTEM** tablespace must be available or the instance will not start.
 - A running instance will crash if this tablespace becomes unavailable.
- ✳ Oracle 10g databases also require a **SYSAUX** tablespace.
 - Many database components, such as the Oracle Scheduler, the Oracle Enterprise Manager (OEM), and the Recovery Manager (RMAN), use **SYSAUX** as the default storage location of their repository tables.
 - This additional tablespace keeps the Data Dictionary separate from these other database objects.
 - The **SYSAUX** tablespace must be available to start the instance, but the instance will not crash if this tablespace becomes unavailable.
 - However, you will lose the functionality of the components that store objects in this tablespace.

The **SYSTEM** tablespace has been around since the earliest days of Oracle. The only objects that should be stored in the **SYSTEM** tablespace are Data Dictionary tables, i.e. those tables owned by the **SYS** user. It is a misconception that the **SYSTEM** user is related to the **SYSTEM** tablespace.

A **SYSAUX** tablespace is created when an Oracle 10g database is created. If you are upgrading from a pre-10g version of Oracle, you must create the **SYSAUX** tablespace before your upgrade can proceed. If you perform your database upgrade with the DBUA, the **SYSAUX** tablespace will be created for you.

Many Oracle components, like the Oracle Scheduler, the Oracle Enterprise Manager (OEM), and the Recovery Manager (RMAN), require repository tables to store information. Prior to Oracle 10g, it had become common practice to store these non-Data Dictionary objects that supported database operations in the **SYSTEM** tablespace. Oracle 10g places only Data Dictionary tables in the **SYSTEM** tablespace and the non-Data Dictionary tables in the **SYSAUX** tablespace.

You cannot drop or rename the **SYSTEM** or **SYSAUX** tablespaces.



CREATING TABLESPACES

- * Create a permanent, undo, or temporary tablespace with the **CREATE TABLESPACE** command.

```
CREATE [UNDO|TEMPORARY] TABLESPACE ts_name  
[DATAFILE|TEMPFILE] file_specification;
```

- You must have the **CREATE TABLESPACE** system privilege to create a tablespace.
- You must have the **SYSDBA** privilege to be able to create the **SYSAUX** tablespace.
- The server account that runs the Oracle instance must have OS privileges to create files in the directory you specify.

- * The file specification has the following form:

```
'/directory/filename' SIZE xx[K|M|G|T]  
[AUTOEXTEND [ON|OFF] [NEXT xx[K|M|G|T]  
[MAXSIZE xx|UNLIMITED] ]
```

- The filename, including the directory path, is in single quotes.
 - For datafiles, we usually use a *.dbf* filename extension.
- The size in Kilobytes, Megabytes, Gigabytes, or Terabytes is specified.
- If you wish to let your datafile allocate more space when it becomes full, you can turn **AUTOEXTEND ON** and specify the **NEXT** size to be allocated.
- To restrict the growth of the file, which is recommended, specify a **MAXSIZE**; otherwise, you can let the file have **UNLIMITED** growth.

Here we cover the major components of the **CREATE TABLESPACE** command. There are many advanced options which can be found in the Oracle 10g SQL Reference Guide.

Let's look at an example of creating a permanent tablespace. Our tablespace will initially be 100MB in size and be allowed to grow to 500MB.

```
CREATE TABLESPACE course_data  
DATAFILE '/oradata1/orcl/course_data01.dbf' SIZE 100M  
AUTOEXTEND ON NEXT 100M MAXSIZE 500M;
```

If a datafile specification refers to an existing physical file, the **REUSE** keyword instructs Oracle to overwrite the contents of the existing file. If **REUSE** is not specified and a file is present, the **CREATE TABLESPACE** will fail.

DICTIONARY- AND LOCALLY MANAGED TABLESPACES

- * When you create an object in a tablespace, that object must allocate at least one extent.
- * A Dictionary-Managed Tablespace (DMT) records extent allocation and deallocation in Data Dictionary tables.
 - The **SYS.UE\$** table tracks all used extents in the database.
 - The **SYS.FE\$** table tracks all free extents in the database.
- * A Locally Managed Tablespace (LMT) records extent allocation and deallocation in bitmaps in the datafile headers.
- * LMTs have the following advantages over DMTs:
 - Managing space in bitmaps is more efficient.
 - Coalescing free extents is unnecessary.
 - Free space will not become fragmented.
 - Concurrent space allocation and deallocation operations can potentially cause a bottleneck on the Data Dictionary tables.
 - Managing extent sizes is removed from the DBA's list of duties.
- * When you create an Oracle 10g database, the **SYSTEM** tablespace is Locally Managed by default.
 - If the **SYSTEM** tablespace is Locally Managed, then all other tablespaces in the database must be Locally Managed or read-only.
 - If your **SYSTEM** tablespace is Dictionary-Managed, then you can have LMTs and DMTs in your database.
- * The **DBMS_SPACE_ADMIN**-supplied package can be used to convert tablespaces to or from local management.

How LMTs and DMTs Manage Extents

A Dictionary-Managed Tablespace (DMT) uses two Data Dictionary tables: **SYS.UE\$** and **SYS.FE\$**. When an extent is allocated, the system looks in **SYS.FE\$** for a free extent large enough. The table of free extents is updated to indicate that the space is no longer free. Similarly, the **SYS.UE\$** table is updated to indicate that the space is now being used by an object.

A Locally Managed Tablespace (LMT) manages extents locally in the tablespace using bitmaps in the datafile headers. The bits in the bitmap are turned on if the extent is being used and turned off if the extent is free.

LMTs are more efficient than DMTs, especially when allocating and deallocating space. Instead of searching for a free extent in a Data Dictionary table, a quick scan of a bitmap is performed. To allocate the space, a simple bitwise-**OR** operation on the bitmap is performed to indicate that the extent is now being used. When deallocating space, a simple bitwise-**AND** operation is performed on the bitmap. Furthermore, concurrent allocation and deallocation requests do not create a bottleneck on the two tables in the Data Dictionary.

In a DMT, it is possible for two or more physically adjacent extents to become free. Since they are adjacent, the free space can be coalesced, or combined, into one larger chunk of free space. The **SMON** process is responsible for coalescing free space. An LMT does not have to coalesce free space, since it makes better use of extent management. Relieving **SMON** from coalescing free space can save system resources. Furthermore, extents in LMTs are a predetermined size. If you are using DMTs, the extents could be virtually any multiple of the tablespace's block size. It is possible in a DMT to have free space become so fragmented as to make the space unusable for future allocation attempts. Free space in LMTs will not become fragmented in such a way that the free space cannot be reused, relieving the system from the need to periodically reorganize tablespace contents.

Before the advent of LMTs, the DBA would spend a large amount of time working with complex spreadsheets to determine an optimal extent size for database segments. Now that LMTs manage extents for us, the DBAs time is freed up to spend on other tasks. If at all possible, you should use LMTs for all of your tablespaces.

The **DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_TO_LOCAL** procedure will convert a DMT to an LMT. The **DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_FROM_LOCAL** procedure will convert an LMT to a DMT.

LOCALLY MANAGED TABLESPACE EXTENT ALLOCATION

- ✴ When you create a Locally Managed Tablespace, you define its extent allocation strategy.

```
CREATE TABLESPACE ts_name file_specification  
EXTENT MANAGEMENT LOCAL  
[AUTOALLOCATE|UNIFORM SIZE xx]
```

- The extent allocation strategy cannot be changed once the tablespace has been created.
- ✴ With the **UNIFORM** extent allocation strategy, all extents in the tablespace will be the same exact size.
 - You define the extent size when the tablespace is created.
 - Uniform extent sizes are good for tablespaces that contain similarly sized objects, such as small tables.
 - Undo tablespaces are good candidates for uniform extent sizes.
 - ✴ The second extent allocation strategy is to let Oracle **AUTOALLOCATE** the extents.
 - The initial extent size is 64k.
 - With **AUTOALLOCATE**, Oracle will determine the best size for your extents.
 - As the segment grows, the extent sizes grow to 1M or 8M in size.
 - Letting Oracle automatically determine the extent size is good for virtually all situations.
 - **AUTOALLOCATE** is not available for temporary tablespaces.

The **extent_management_clause** has the following form:

```
EXTENT MANAGEMENT [DICTIONARY|LOCAL  
[UNIFORM SIZE xx|AUTOALLOCATE]];
```

You can specify **EXTENT MANAGEMENT DICTIONARY** to create a DMT.

The **EXTENT MANAGEMENT LOCAL** clause creates an LMT. If creating an LMT, then you will also want to specify **UNIFORM SIZE xx** or **AUTOALLOCATE** for uniform extent sizes or the **AUTOALLOCATE** method, respectively.

```
CREATE TABLESPACE course_data  
DATAFILE '/oradata1/orcl/course_data01.dbf' SIZE 100M  
AUTOEXTEND ON NEXT 100M MAXSIZE 500M  
EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

The **UNIFORM** extent allocation will ensure that each and every extent in the tablespace is the same size. You determine the size of the uniform extents. If you specify an initial extent that differs from the **UNIFORM** size, the system will allocate enough extents to be at least the size you specified as the initial extent. For instance, if you set **UNIFORM** extent sizes to be 1M in size, and you request an initial extent of 2M, then the system will allocate two 1M extents for you. If you request an initial extent of 1500K, then the system will allocate two 1M extents for you. If you request an initial extent of 50K, then the system will allocate one 1M extent for you.

The **AUTOALLOCATE** method lets Oracle handle the extent sizes for you. Typically, the extents are 64K, 1M, or 8M in size. When an object is created, its initial extent is 64K. After 64 of these extents have been allocated, Oracle will start to allocate 1M extents. After 64 1M extents, Oracle will start to allocate 8M extents. As the object gets larger, the extents get larger.

When LMTs and **AUTOALLOCATE** were initially introduced, many Oracle DBAs were leery of letting Oracle determine the extent size. Now that LMTs have been around for a while, Oracle DBAs are finding that the **AUTOALLOCATE** method does a very good job of determining your object's extent sizes. The **AUTOALLOCATE** method is well-suited for virtually any situation. Temporary tablespaces that are Locally Managed cannot use the **AUTOALLOCATE** method.

TEMPORARY TABLESPACES

- ✴ If an operation cannot be performed in memory, the system breaks the operation down into manageable pieces, storing them in temporary segments.
 - Temporary tablespaces store this information only for the duration of a session.
 - Segments in temporary tablespaces are often used for sort operations that cannot be completed in memory, so they are commonly called *sort segments*.
 - You can create multiple temporary tablespaces in a database, but there is typically only one.
- ✴ When you create a temporary tablespace, you have the option of using standard **DATAFILE**s or using **TEMPFILE**s.
 - **TEMPFILE**s are similar to **DATAFILE**s, except that changes to their contents do not generate redo.
 - If you create your temporary tablespace with **TEMPFILE**s (which is highly recommended) then your tablespace must be Locally Managed with **UNIFORM** extent allocation.
- ✴ You can assign a user a temporary tablespace with the **CREATE USER** or **ALTER USER** statements.

```
ALTER USER username TEMPORARY TABLESPACE temp_ts;
```

Since these segments do not need to be permanent, if the system or session crashes in the middle of the operation, the temporary segment is simply dropped.

There are many operations that may require a temporary segment:

| CREATE INDEX | SELECT ... ORDER BY | SELECT DISTINCT ... | SELECT ... GROUP BY |
|------------------|---|--------------------------------------|---------------------|
| SELECT ... UNION | SELECT ... INTERSECT | SELECT ... MINUS | CREATE BITMAP INDEX |
| Merging bitmaps | Any select statement that performs a has join | Some joins and correlated subqueries | |

To create a temporary tablespace with tempfiles, use the **CREATE TEMPORARY TABLESPACE** command as follows:

```
CREATE TEMPORARY TABLESPACE tablespace_name
TEMPFILE '/directory/filename' SIZE xxM
EXTENT MANAGEMENT LOCAL UNIFORM SIZE yyK;
```

To create a temporary tablespace with datafiles, use the **TEMPORARY** clause at the end of the **CREATE TABLESPACE** command similar to the following:

```
CREATE TABLESPACE bri_temp
DATAFILE '/oradata1/frccd/bri_temp.dbf' SIZE 12M TEMPORARY;
```

It should be noted that the **TEMPORARY** clause is still available in Oracle 10g, but it has been deprecated. It will be removed in a future version.

TEMPORARY TABLESPACE GROUPS

- ✴ A temporary tablespace group lets parallel operations use multiple temporary tablespaces to assist with sort processing.
- ✴ A temporary tablespace group must contain at least one tablespace, but it can contain any number of tablespaces.
- ✴ A temporary tablespace group is implicitly created when you create the first tablespace in the group.

```
CREATE TEMPORARY TABLESPACE ts_name
TEMPFILE '/directory/filename' SIZE xx [K|M|G|T]
TABLESPACE GROUP groupname;
```

- ✴ Use **ALTER TABLESPACE** to add or remove a temporary tablespace to or from an existing group.

```
ALTER TABLESPACE ts_name TABLESPACE GROUP groupname;
```

```
ALTER TABLESPACE ts_name TABLESPACE GROUP ' ';
```

- If the result of the **ALTER TABLESPACE** operation leaves a group with no members, the group is automatically removed from the database.
- ✴ Use **CREATE USER** or **ALTER USER** to assign a user to a temporary tablespace group.

```
ALTER USER username TEMPORARY TABLESPACE groupname;
```

- A user is then free to utilize space in any tablespace in the group.
- ✴ Use **ALTER DATABASE** to make the temporary tablespace group the default for the database.

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE groupname;
```


DEFAULT TABLESPACES

- ✴ When a user creates an object that requires a segment, like a table or an index, the user has the option of specifying a tablespace for that object.

```
CREATE TABLE car (  
    vin CHAR(17),  
    make VARCHAR(25),  
    model VARCHAR2(25)  
) TABLESPACE user_space;
```

- If a user does not specify a tablespace for the segment, the segment will be created in the user's default tablespace.

- ✴ You can change the default tablespace for a user with the **ALTER USER** command.

```
ALTER USER bob DEFAULT TABLESPACE user_space;
```

- ✴ Out of the box, the default tablespace for all users is the **SYSTEM** tablespace.

- Only **SYS** should create segments in the **SYSTEM** tablespace.
- It is a very good idea to define a non-**SYSTEM** default tablespace for all users.

- ✴ Use the **ALTER DATABASE** command to define a system-wide default tablespace for all users who do not have a specifically assigned default tablespace.

```
ALTER DATABASE DEFAULT TABLESPACE user_space;
```

- ✴ You can also designate a default temporary tablespace for all users.

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE temp_space;
```

- If a user is not given a new default temporary tablespace, they will create all temporary segments in the **SYSTEM** tablespace. (This is not a recommended solution.)

TABLESPACE QUOTAS

- * A user's *quota* on a tablespace defines the limit on how much total space, for all schema objects owned by the user, they can allocate in a tablespace.

- The quota can be specified in bytes, kilobytes, megabytes, or defined as **UNLIMITED**, when creating or altering a user.

```
CREATE USER username IDENTIFIED BY password  
QUOTA UNLIMITED ON ts_name;
```

```
ALTER USER username QUOTA xx [K|M] ON ts_name;
```

- * A quota of zero assigned to a user for a tablespace means that the user cannot create any objects in that tablespace.

- By default, users have a quota of zero on all tablespaces until you change their quota.

- * The **UNLIMITED TABLESPACE** system privileges lets a user consume an unlimited amount of space in all tablespaces.

- Users should not be able to create objects in the **SYSTEM** tablespace, so the **UNLIMITED TABLESPACE** system privilege should rarely be granted.
- Granting the deprecated **RESOURCE** role implicitly grants the **UNLIMITED TABLESPACE** system privilege.

The only user who should be able to allocate space in the **SYSTEM** tablespace is **SYS**. All other users should have a zero quota on the **SYSTEM** tablespace. Application users should not have any quota in the **SYSAUX** tablespace, either. It is good practice to grant quotas only on those tablespaces a user needs to allocate space.

If a user has a quota on a tablespace and you no longer wish the user to allocate further space in the tablespace, then set their quota to zero for that tablespace. Any objects in the tablespace will remain, but they will not be allowed to grow and the user will not be able to create new objects in the tablespace. Space assigned to existing objects will be retained and may be reused if required.

If a user has quotas on various tablespaces and the user is granted the **UNLIMITED TABLESPACE** system privilege, this privilege overrides all quota settings previously defined for the user. If you subsequently revoke this privilege from the user, the original quota settings are in effect.

Granting the **RESOURCE** supplied role has the side effect of granting the **UNLIMITED TABLESPACE** system privilege to the user. This privilege is not part of the **RESOURCE** role. If you revoke the **RESOURCE** role, the **UNLIMITED TABLESPACE** system privilege is still granted to the user. The **RESOURCE** and **DBA** roles are deprecated and may not be supported in future releases of Oracle. It is highly recommended that you create your own role, instead of using the **RESOURCE** role and its implied privileges.

DROPPING AND ALTERING A TABLESPACE

- ✴ You must have the **DROP TABLESPACE** system privilege in order to be able to remove a tablespace from the database.

```
DROP TABLESPACE ts_name [INCLUDING CONTENTS [AND  
DATAFILES] [CASCADE CONSTRAINTS]] ;
```

- You cannot drop the **SYSTEM** tablespace.
- You can only drop the **SYSAUX** tablespace if you are connected as **SYSDBA** and you have started the database in **MIGRATE** mode.
- You cannot drop an active undo tablespace or a database's default tablespace.
 - If the tablespace has been designated as the database's default tablespace, you must designate a new tablespace as the database default before you can drop the tablespace.
- ✴ The **MANAGE TABLESPACE** system privilege lets you change the tablespace's availability, make it read-only or read/write, or start and stop a backup on the tablespace.
- ✴ The **ALTER TABLESPACE** system privilege will additionally allow the user to perform coalescing, resizing, and other operations on a tablespace.

```
ALTER TABLESPACE ts_name [BEGIN|END] BACKUP ;
```

```
ALTER TABLESPACE ts_name [READ ONLY|READ WRITE] ;
```

```
ALTER TABLESPACE ts_name [ONLINE|OFFLINE] ;
```

If you attempt to drop a tablespace that contains objects, you will receive an error, unless you specify the optional **INCLUDING CONTENTS** clause. When you drop a tablespace, the datafiles will remain on the server to be removed with OS commands. If you want to drop the tablespace and remove the datafiles in the same operation, use the **INCLUDING CONTENTS AND DATAFILES** clause. If any tables to be dropped have foreign key constraints to tables outside the tablespace, you will receive an error, unless you include the **CASCADE CONSTRAINTS** clause.

RENAMING TABLESPACES

- ✴ Starting in Oracle 10g, you can rename tablespaces.

```
ALTER TABLESPACE ts_name RENAME new_ts_name;
```

- ✴ The tablespace and all of its datafiles must be online before you can rename the tablespace.
- ✴ The **COMPATIBLE** parameter must be at least 10.0.0.
- ✴ You cannot rename the **SYSTEM** and **SYSAUX** tablespaces.
- ✴ You must have the **ALTER TABLESPACE** system privilege.
- ✴ If the tablespace is **READ ONLY**, a rename operation will complete, but the datafile headers will not be updated.
 - The Alert Log will show an error until you put the tablespace in **READ WRITE** mode and the datafile headers are updated.
- ✴ The ability to rename a tablespace was added to support Transportable Tablespaces.

Transportable Tablespaces is a feature whereby you can copy a tablespace to another database and plug it in, making the data available in another database. Transportable Tablespaces are a quick and easy way to move large amounts of data. If the tablespace name is already being used in the destination tablespace, then the operation to plug in the tablespace will fail. Starting in Oracle 10g, you can rename the tablespace in the destination database and transport the new tablespace without a problem.

If the tablespace being renamed is an undo tablespace, then the **UNDO_TABLESPACE** initialization parameter will be changed in the **SPFILE** when you rename the tablespace. If you started the database with a **PFILE**, then you need to manually modify this parameter. A message will be placed in the Alert Log reminding you to do this.

RENAMING OR RELOCATING DATAFILES

- ✴ To the Oracle database, there is no difference between renaming and relocating a datafile.
 - A datafile's name contains both the directory path and the file name.
- ✴ You must have the **ALTER DATABASE** system privilege to be able to rename or relocate a datafile.
- ✴ To rename or relocate a datafile with the instance stopped, use the following steps:
 1. Bring down the instance.
 2. Rename and/or relocate the file with OS commands.
 3. Start the instance in **MOUNT** mode. In **MOUNT** mode, the datafiles are not accessed, therefore the instance will not generate an error that it cannot find the datafile.
 4. Issue the **ALTER DATABASE RENAME FILE** command, which updates the control file record of the file's location.
 5. Open the database.
- ✴ If you need to rename or relocate the **SYSTEM** tablespace's datafiles, then you must use the method where the instance is not running.
 - If you rename or relocate a datafile with the system up and running, your users may experience application errors until the operation is complete.
 - Additionally, to rename or relocate a datafile while the instance is running, you must be able to take the datafile offline, which has some restrictions, e.g. active **UNDO** spaces.
 - Active transactions may cause the offline operation to take a very long time to complete, since the operation will not complete until the activity terminates.

To rename or relocate a datafile with the instance stopped:

1. Bring down the instance.

```
SHUTDOWN IMMEDIATE
```

2. Rename and/or relocate the file with OS commands.

In UNIX:

```
mv /mydirectory/datafile /mynewdirectory/mynewdatafile
```

In Windows:

```
rename filename new_file OR move c:\directory\filename c:\new_dir\
```

3. Start the instance in **MOUNT** mode. In **MOUNT** mode, the datafiles are not accessed, therefore the instance will not generate an error that it cannot find the datafile.

```
STARTUP MOUNT
```

4. Issue the **ALTER DATABASE RENAME** command.

```
ALTER DATABASE RENAME FILE  
'/directory/filename' TO '/new_dir/new_file';
```

5. Open the database for business.

```
ALTER DATABASE OPEN;
```

To rename or relocate a datafile with the instance running:

1. Take the datafile offline. This operation can take a long time to complete. You will not be able to take the **SYSTEM** tablespace's datafiles and active undo tablespace's datafiles offline.

```
ALTER DATABASE DATAFILE '/directory/filename' OFFLINE;
```

2. Rename and/or relocate the file with OS commands.

In UNIX:

```
mv /mydirectory/datafile /mynewdirectory/mynewdatafile
```

In Windows:

```
rename filename new_file OR move c:\directory\filename c:\new_dir\
```

3. Issue the **ALTER DATABASE RENAME** command.

```
ALTER DATABASE RENAME FILE  
'/directory/filename' TO '/new_dir/new_file';
```

4. Bring the datafile online.

```
ALTER DATABASE DATAFILE '/directory/filename' ONLINE;
```

BIGFILE TABLESPACES

- * Bigfile tablespaces were introduced in Oracle 10g to support Very Large Databases (VLDB).
 - Using bigfile tablespaces, the database can be as large as 8 Exabytes (8 million Terabytes).

- * To create a bigfile tablespaces, use the **CREATE BIGFILE TABLESPACE** command.

```
CREATE BIGFILE TABLESPACE ts_name  
DATAFILE '/directory/filename' SIZE xx[K|M|G|T];
```

- * Bigfile tablespaces simplify management of the database by reducing the number of files that make up the tablespace.
- * A bigfile tablespace can only consist of a single bigfile.
- * Bigfile tablespaces also simplify management by making the datafile transparent in the tablespace for SQL commands.
 - Use the **ALTER TABLESPACE** command to resize the bigfile.

```
ALTER TABLESPACE ts_name RESIZE xx[K|M|G|T];
```

- Use the **ALTER TABLESPACE** command to make the bigfile auto-extensible.

```
ALTER TABLESPACE ts_name AUTOEXTEND ON  
NEXT xx[K|M|G|T];
```

Prior to 10g, Oracle tablespaces required multiple datafiles in order to grow beyond a relatively low size limit. These smallfile tablespaces (still the default type) can contain 1022 files, with 4M blocks in each file. But, the entire database has a limit of 64K files. With bigfile tablespaces, it is much more difficult to hit the 64K datafile limit.

Bigfile tablespaces were introduced in Oracle 10g to support Very Large Databases (VLDB). Without bigfile tablespaces, the Oracle database can only be a maximum of 512 Petabytes (approximately 525,000 Terabytes) in size. With bigfile tablespaces, the Oracle database can be as large as 8 Exabytes (approximately 8 million Terabytes).

Bigfile tablespaces simplify management of large databases. By using bigfiles instead of regular datafiles (also called *smallfiles*), you can significantly reduce the total number of datafiles in your database. To the DBA, there is no difference between the bigfile and the tablespace. This means that you can modify a bigfile simply by using the **ALTER TABLESPACE** command. Instead of using an **ALTER DATABASE** command to resize a datafile, you can use the **ALTER TABLESPACE** command:

```
ALTER TABLESPACE ts_name RESIZE xx[K|M|G|T] ;
```

Instead of using the **ALTER DATABASE** command to modify the autoextensibility of a bigfile, you can use the **ALTER TABLESPACE** command:

```
ALTER TABLESPACE ts_name AUTOEXTEND ON NEXT xx[K|M|G|T] ;
```

To create a bigfile tablespace, use the **CREATE BIGFILE TABLESPACE** command:

```
CREATE BIGFILE TABLESPACE ts_name  
DATAFILE 'directory/filename' SIZE xx[K|M|G|T] ;
```

Since a bigfile tablespace can only have a single bigfile, the **ALTER ADD DATAFILE** command cannot be used on such a tablespace.

DATA DICTIONARY VIEWS FOR DATAFILES AND TABLESPACES

- * The **DBA_TABLESPACES** view obtains its information from the Data Dictionary and requires the database to be open.
 - **DBA_TABLESPACES** contains a **BIGFILE** column to indicate if the tablespace is a bigfile tablespace.
 - The **CONTENTS** column indicates if it is **PERMANENT** or **TEMPORARY**.
 - The **V\$TABLESPACE** view can only be queried with the database mounted and gets information from the control files.
- * The **DBA_DATA_FILES** and **DBA_TEMP_FILES** views give you details on datafiles and tempfiles.
 - The corresponding **V\$DATAFILE** and **V\$TEMPFILE** views obtain their information from the control files, the information being available after the database has been mounted.
 - These views require the database to be open.
- * The **DBA_SEGMENTS** view gives details on every segment in the database.
 - This view is accompanied by the **ALL_SEGMENTS** and **USER_SEGMENTS** views.
- * Each segment is comprised of multiple extents.
 - To obtain details on extents, you can query the **DBA_EXTENTS**, **USER_EXTENTS**, or **ALL_EXTENTS** views.

| View | Contents |
|------------------------------|--|
| V\$DATAFILE | All datafiles in the database. |
| V\$TEMPFILE | All tempfiles in the database. |
| DBA_DATA_FILES | All datafiles in the database. |
| DBA_TEMP_FILES | All tempfiles in the database. |
| V\$TABLESPACE | All tablespaces in the database. |
| DBA_TABLESPACES | All tablespaces in the database. |
| DBA_FREE_SPACE | All free extents in all tablespaces. |
| DBA_USERS | Default and temporary tablespaces for all users. |
| DBA_TS_QUOTAS | User's quotas on all tablespaces. |
| DBA_SEGMENTS | All segments in all tablespaces. |
| DBA_EXTENTS | All extents in all tablespaces. |
| DBA_TABLESPACE_GROUPS | Temporary tablespace groups in the database. |

The **DBA_FREE_SPACE** view is often queried by the DBA to see how much free space is available in each tablespace. Many applications have data requirements that grow over time. If there is not enough free space in the tablespace, then the application may run out of room and experience errors.

The **DBA_USERS** view contains the default tablespace and temporary tablespace for every user in the database. If you want to know how much quota a user has on a specific tablespace, query the **DBA_TS_QUOTAS** view.

Display all files for all tablespaces, including the file sizes:

file.sql

```
SELECT tablespace_name, file_name, bytes
FROM dba_data_files ORDER BY 1,2;
```

Display information on all tablespaces in the database:

tbspace.sql

```
SELECT tablespace_name, status, extent_management, allocation_type
FROM dba_tablespaces;
```

Display allocated space usage summary by tablespace:

tbs_alloc.sql

```
SELECT f.tablespace_name, f.bytes AS bytes_free,
       d.bytes as bytes_alloc,
       ((d.bytes-f.bytes)/d.bytes)*100 as pct_used
FROM (SELECT tablespace_name,SUM(bytes) AS bytes
      FROM dba_free_space GROUP BY tablespace_name) f,
     (SELECT tablespace_name,SUM(bytes)
      FROM dba_data_files GROUP BY tablespace_name) d
WHERE f.tablespace_name = d.tablespace_name
ORDER BY 1;
```

LABS

- ❶ Write a query to determine the locations and sizes of all of your database's data files.
(Solution: *datafiles.sql*)
- ❷ Write a query that lists all tablespaces, their total sizes, and amount of free space.
(Solution: *dataspace.sql*)
- ❸ Create a 10MB tablespace named **UNIFORM_LMT** that is locally managed, with a uniform extent size of 128KB. Use the same directory location as your database's existing datafiles.
(Solution: *uniform_lmt.sql*)
- ❹ Create a tablespace named **STUDENT_DATA** that is locally managed with the **AUTOALLOCATE** method. Initially allocate 100MB for the datafile, but allow the datafile to grow to 500MB, 50MB at a time.
(Solution: *student_data.sql*)
- ❺ Query **DBA_TABLESPACES** and view all of the attributes of your two tablespaces.
(Solution: *tbs_review.sql*)
- ❻ Drop the **UNIFORM_LMT** tablespace. Confirm that it's no longer listed in the Data Dictionary. Look for the datafile. Is it still there?
(Solution: *druniform_lmt.sql*)

CHAPTER 25 - ORACLE ENTERPRISE MANAGER

OBJECTIVES

- * Describe the benefits of Oracle Enterprise Manager (OEM).
- * Describe the OEM architecture and directory configurations.
- * Install and configure OEM.
- * Manage OEM.
- * Connect to OEM.
- * Navigate OEM and learn about functionality present in OEM.

INTRODUCING ORACLE ENTERPRISE MANAGER

- * *Oracle Enterprise Manager (OEM)* is a tool used to manage your Oracle database.
- * With OEM you can perform typical DBA tasks:
 - Create, alter, and remove user accounts and database objects.
 - Schedule database jobs to run at specific times.
 - Back up or recover your database.
 - Keep your database updated with the most current Oracle patch sets.
- * OEM provides proactive database management features.
 - Proactively monitor database performance.
 - Email and paging facilities make for easy alerting.
- * OEM provides tuning facilities.
 - Locate bad SQL running on your database.
 - Use the Oracle database advisors to tune your database and your SQL.

One of the main goals of the Oracle Database 10g release was to ease the administration of valuable data. Oracle Enterprise Manager (OEM) is a key component in this strategy. OEM allows you to easily manage, monitor, and tune your Oracle database.

OEM allows you to manage many databases, and each database has its own home page which contains summary information. You can then drill down from that home page into more detailed information. The home page also contains tabbed navigation to allow you to access specific functions.

OEM is installed on your system when you install the Oracle database server. OEM comes in three different editions:

- Oracle Database Control — The standard edition of OEM.
- Oracle Grid Control — An edition of OEM specific to management of RAC clusters.
- Oracle App Server Control — Available from 10gR2, this edition allows you to manage the Oracle application server.

OEM comes with a standard set of functionality as well as a number of optionally licensed add on products such as the Tuning Pack for Database and the Diagnostics Pack for Database. When using OEM you should make sure you are licensed for the functionality you are using, because all of the optional packs will be installed when OEM is installed.

As with previous versions, OEM offers a great deal of functionality including the ability to manage users and database objects. OEM is easily accessed through your web browser. OEM provides an easy interface into RMAN, allowing the DBA to easily backup and recover the Oracle database.

OEM also offers proactive database monitoring. The various OEM processes and agents will continuously monitor the database for trouble. Should a problem be detected, OEM can be configured to send out alerts to DBAs should specific conditions occur. This improves the uptime of your database since you will be notified of conditions with outage potential before they happen, not after.

Oracle Database 10g introduced a number of self-diagnostic packages called advisors. These advisors can be used to help the DBA configure the Oracle database and can also be used to locate potential database problems and correct them. OEM is the principle interface into these advisors. We will discuss the various advisors more later in this lesson.

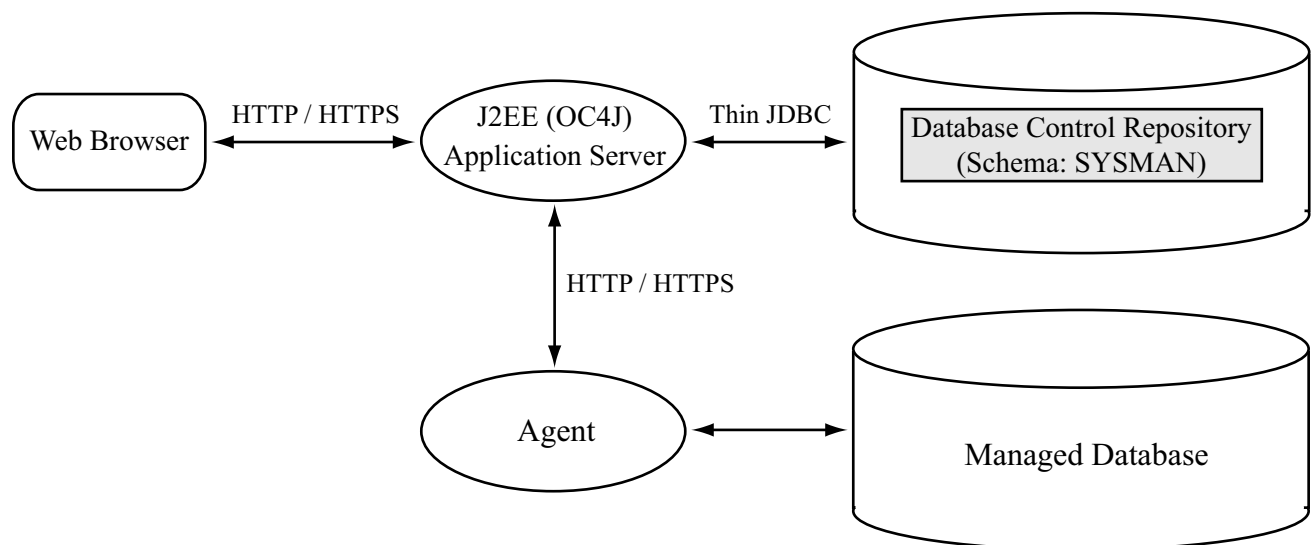
OEM has many features. Indeed an entire class could be built off of using OEM. In this chapter we will introduce you to OEM, show you how it works, and give you the tools you need to explore all the functionality available in this Oracle product.

OEM ARCHITECTURE AND PHYSICAL STRUCTURE

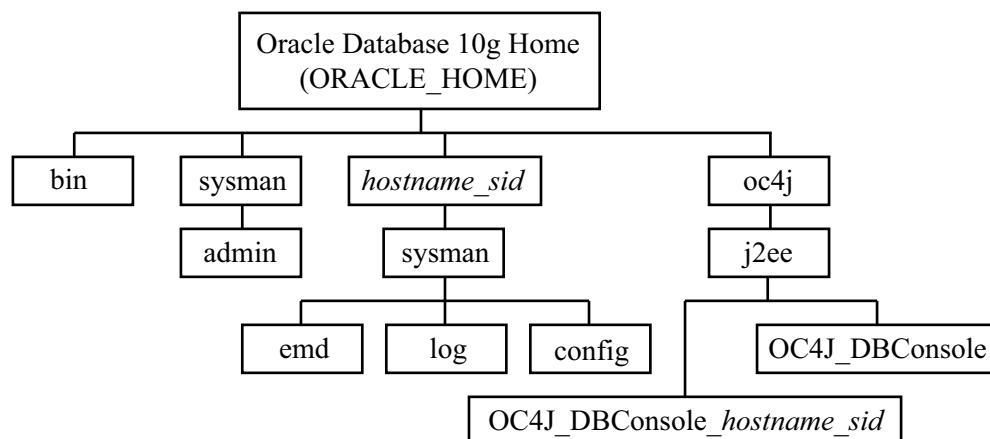
- * OEM is a multi-tiered architecture that includes:
 - The *Oracle Management Service* (OMS) — A J2EE (OC4J) application that provides the web services required by OEM.
 - The OMS uses an OEM *repository* stored in an Oracle database.
 - An Oracle Intelligent Agent, a process which monitors and interacts with the managed database.
 - The OMS can interact with many agents, on the local host or other hosts.
 - A Web browser.
- * OEM has its own directory structure under *ORACLE_HOME*:
 - *bin* — OEM Executables
 - *sysman* — Source files and scripts
 - *oc4j/j2ee/OC4J_DBConsole/<host>_<SID>* — OMS files
 - *<host>_<SID>/sysman* — Agent files
- * When you install OEM, an OEM repository is created.
 - This repository is stored in the **SYSMAN** schema (in the **SYSAUX** tablespace).
 - You should never need to manually deal with the repository.
 - OEM uses the **DBSNMP** account for SNMP access.
- * OEM can be installed in the default *ORACLE_HOME* directory, or in its own *ORACLE_HOME* directory.

OEM Architecture

More complex configurations of OEM allow for agent processes on multiple databases and a single OEM repository so a single OEM instance can manage many databases at one time. These types of complex configurations are beyond the scope of this training class, and can take some time to setup and configure correctly. Here is a diagram of the basic OEM architecture



OEM Directory Structure



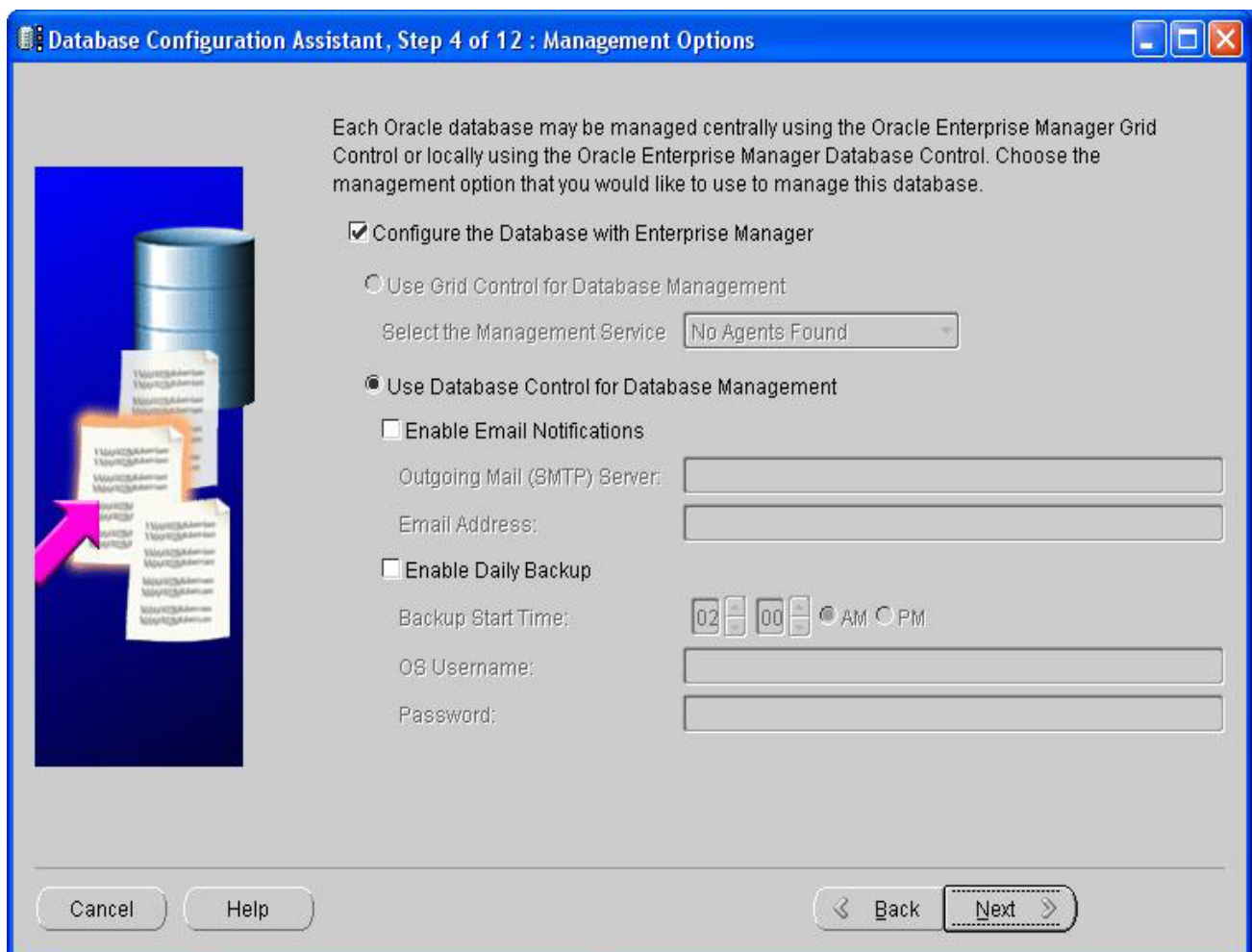
INSTALLING OEM WHEN USING DBCA

- ✱ OEM can be installed when you create a database with the *Oracle Database Configuration Assistant* (DBCA).
 - The DBCA will ask you if you want to install OEM.
 - If you install OEM you can also choose to setup scheduled backups and configure email notifications for alerts.
 - Using DBCA to install OEM makes for a very easy, straightforward OEM install.

If we do not want to install OEM, then we simply uncheck the **Configure the Database with Enterprise Manager** box. After selecting **Next**, the DBCA will continue to create the database and install OEM. Once the database is created, DBCA will start the web server, tell you that it has installed OEM, and give you the URL that you can use to access the OEM home page.

Note in the dialogue box below that you can choose to enable email notifications. If you choose to do this you need to make sure you know your SMTP server for outgoing email. You also need to know what email address you want to send notifications to.

Additionally you can enable daily backups when you choose to have OEM installed. You can define when the backup should begin, and you will need to define an OS username and password that the backup will be started from. This OS account should be an account that has access to the Oracle database.



Database Configuration Assistant, Step 4 of 12 : Management Options

Each Oracle database may be managed centrally using the Oracle Enterprise Manager Grid Control or locally using the Oracle Enterprise Manager Database Control. Choose the management option that you would like to use to manage this database.

☒ **Configure the Database with Enterprise Manager**

☐ Use Grid Control for Database Management

Select the Management Service: No Agents Found

☒ **Use Database Control for Database Management**

☐ **Enable Email Notifications**

Outgoing Mail (SMTP) Server:

Email Address:

☐ **Enable Daily Backup**

Backup Start Time: ☒ AM ☐ PM

OS Username:

Password:

Cancel Help Back Next

USING EMCA TO INSTALL OEM

- * You use the Enterprise Manager Configuration Assistant, **emca**, utility to install OEM in your database.
- * To use **emca** to create an OEM instance, you will need the following:
 - The listener port number (default is 1521).
 - The database SID and service name.
 - If you want to use email, the email server address and gateway.
 - Passwords for **db snmp**, **sysman**, and **sys**.
- * When using **emca** to create an OEM repository:
 - The database and database listener must be running.
 - Make sure that the machine host name resolves to the machine IP address.
- * OEM autostart:
 - After installing OEM on a UNIX system, you will need to configure the appropriate startup files so that OEM is started automatically.
 - These files differ across vendors so check your vendor documentation for more details.
 - In Windows, OEM installs as a windows service which will start automatically by default.
- * **emca** can also operate in silent mode.
 - This requires the creation of a text-based parameter file.
 - Use the **-respfile** parameter to indicate the name and location of the parameter file.

The Enterprise Manager Configuration Assistant, **emca**, utility is used to create, drop, and manage OEM. You use **emca** to install OEM in a database if it does not already have OEM installed. **emca** can also be used to remove OEM control structures from a database and change various OEM related parameters.

Something important to be aware of is that if Oracle cannot resolve the IP address to a host name, then it will use the IP address for all OEM configuration. This can cause serious problems if you are running DHCP. See Metalink note 339425.1 for more on this issue.

MANAGING OEM

- ✧ You use the Enterprise Manager Control Program, **emctl**, to check the status of, start, and stop OEM.
- ✧ To administer the OEM web server, you must first set the **ORACLE_SID** environment variable to identify the related database.

For Windows:

```
set ORACLE_SID=orcl
```

For Linux:

```
export ORACLE_SID=orcl
```

- ✧ Start the OEM Web Server with **emctl**.

```
emctl start dbconsole
```

- ✧ Stop the OEM Web Server with **emctl**.

```
emctl stop dbconsole
```

- ✧ Check the status of OEM with **emctl**.

```
emctl status dbconsole
```

Start the OEM Web Server

In most cases, a properly configured OEM will automatically start when the system is started. As a result, there is nothing the DBA or system administrator needs do. If OEM does not start, you can manually start it. If you are running Windows, start OEM using the services window, or use **emctl**.

If you are running in UNIX you will use **emctl** to control OEM services. Also in UNIX you will want to start OEM in your startup scripts (for example, in the *rc3.d* directory). Start OEM after the Oracle database has been started. You will need to make sure that the listener is running before you start OEM.

To start the OEM service, simply use the **emctl start dbconsole** command as seen in this example:

```
C:\Oracle\product\10.2.0\rob10R2\admin\create>emctl start dbconsole
Oracle Enterprise Manager 10g Database Control Release 10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation. All rights reserved.
http://roberts-sony:1159/em/console/aboutApplication
Starting Oracle Enterprise Manager 10g Database Control ...
The OracleDBConsoleTestoem service is starting.....
The OracleDBConsoleTestoem service was started successfully.
```

Use **emctl** to stop the OEM service, with the **emctl stop dbconsole** command:

```
C:\Oracle\product\10.2.0\rob10R2\admin\create>emctl stop dbconsole
Oracle Enterprise Manager 10g Database Control Release 10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation. All rights reserved.
http://roberts-sony:1159/em/console/aboutApplication
The OracleDBConsoleTestoem service is stopping.....
The OracleDBConsoleTestoem service was stopped successfully.
```

emctl can be used to check the status of OEM, too:

```
C:\Oracle\product\10.2.0\rob10R2\192.168.2.7_rob10R2\sysman\log>emctl
status dbconsole
Oracle Enterprise Manager 10g Database Control Release 10.2.0.1.0
Copyright (c) 1996, 2005 Oracle Corporation. All rights reserved.
http://roberts-sony:1159/em/console/aboutApplication
Oracle Enterprise Manager 10g is running.
-----
Logs are generated in directory C:\oracle\product\10.2.0\rob10R2\
roberts-sony_testoem/sysman/log
```

REMOVING OEM

- ✧ Use the **emca** utility if you need to remove OEM in your database.

```
emca -deconfig dbcontrol db
```

- ✧ You may need to remove and reinstall OEM if your machine's IP address changes and the IP address was used during the install instead of the machine host name.
- ✧ After removing OEM, make sure you remove any startup files that you might have created when you installed OEM.

Hands On:

1. Let's remove, and then reinstall, OEM in our database. Before we can install OEM there are a few things we need to know:
 - a. What is the database SID and the listener port (typically 1521)?
 - b. What is your machine's IP address and host name?
 - c. What is the password to **SYS**, **SYSMAN**, and **DBSMNP**?

2. Check the status of OEM with **emctl**. It should be up and running.

```
prompt>emctl status dbconsole
```

3. Open your web browser, and start OEM to verify that it is running.
4. Open a command-line window and set the *ORACLE_SID* directory and then remove OEM from your database with **emca**:

```
export ORACLE_SID=testoem
emca -deconfig dbcontrol db
```

5. Open your web browser, and start OEM. OEM should not start since you have removed it.
6. Reinstall OEM with **emca**.

```
set ORACLE_SID=testoem
emca -config dbcontrol db
```

7. Open your web browser, and start OEM at the URL listed in the output from the **emca** install. OEM should now open for you.

TROUBLESHOOTING OEM

- ✴ Under *ORACLE_HOME/cfgtoollogs/cfgfw/*, you can find logs of your **emca** activities.
- ✴ Review logs in *ORACLE_HOME/<host>_<SID>/sysman/log/* for agent activity.
- ✴ *ORACLE_HOME/oc4j/j2ee/OC4J_DBConsole_<host>_<SID>/* has files, including logs, for the OMS.
- ✴ The most common reasons that OEM fails to load in the web browser:
 - The environment is not setup correctly.
 - Check parameters such as *ORACLE_HOME*.
 - The OEM service has not been started.
 - The listener is not up and running.
 - Your host name does not resolve properly to your system IP address.
 - On a Windows system, your windows account does not have the correct privileges to run unattended batch programs.
 - An administrator must resolve this configuration issue.

When using **emca** to create or drop an OEM instance, Oracle creates logs that you can review to attempt to troubleshoot OEM problems. These logs are stored in `$ORACLE_HOME/cfgtoollogs` under subdirectories for each tool. The log file name varies by OS, but typically starts with **emca**. Use the log files to attempt to determine the cause of **emca** failures. Also, you will find additional logs specific to each database in `$ORACLE_HOME/cfgtoollogs/<dir>` where `<dir>` is a directory specific to the individual database instance.

Logging for **emctl** can be found in the `emoms.log` log file which is in the directory `$ORACLE_HOME/<hostname>_<db>/sysman/log`.

To set correct privileges on a Windows system:

1. Click the **Start** button.
2. Select **Settings → Control Panel**.
3. Select **Administrative Tools**.
4. Select **Local Security Policy**.
5. Expand **Local Policies → User Rights Assignment**.
6. Open **Logon** as batch job.
7. Add user account name to list.
8. Click **OK** to save amended setting.

SECURING OEM

- * You need to change the passwords for various Oracle OEM related accounts before you install OEM.
 - Each of these accounts has very basic default passwords on database install or may be locked, depending on how your database was created.
- * **SYSDBA** privileges are required to use the full OEM functionality.
- * Developer (non-DBA) accounts can also use OEM.
 - They will need to have the **SELECT ANY DICTIONARY** privilege to use most of the OEM.
- * Oracle security add-on products offer network encryption that can further secure OEM.

Securing A Database When Using OEM

OEM uses four specific accounts: **SYS** (or another DBA privileged account with **SYSDBA** privileges), **SYSTEM**, **SYSMAN** and **DBSNMP**. Optionally you can use a non-**SYSDBA** account although you will have fewer privileges within OEM.

Each of these accounts must be properly secured because they each have very powerful privileges. Before you install OEM you should change the passwords for the **SYS**, **SYSMAN**, and **DBSNMP** accounts.

Creating a Non DBA OEM Account

Non-DBA users can also use OEM. Any account that is going to use OEM must have the **CREATE SESSION** privilege and the **SELECT ANY DICTIONARY** privilege in order to use OEM. With this configuration, the OEM user will not be able to perform DBA activities such as starting and stopping the database. They will have access to the various OEM summary reports, and to the different tuning tools that developers typically require.

Permitting developers access to OEM and to the information provided should not be considered a security risk. If the developer attempts any DBA operation they will be returned an Oracle error, which will indicate that they have insufficient privileges. OEM will help the developer write better code, and will help them in their tuning efforts. It might mean that you, the DBA, will have to field more developer questions, because the developer will have access to much more performance information. This is a short term problem however, and in the long run developer access to all that OEM offers will benefit you as the DBA.

OEM Network Security

You can further secure OEM by enabling **SQLNET** encryption available through Oracle security add-on products. Anyone who has access to OEM and the **SYS** account can run any OEM function on your database, so protect the **SYS** account password and change it often.

Since OEM is a web application, it is important to secure access to that application. Securing OEM with firewalls and other types of network security are important. OEM uses specific ports so you will need to provide access to that port to anyone outside the firewall (via some form of tunneling, for example).

Changing Passwords

If you need to change the password to the **SYSMAN** account, you will need to modify the OEM properties file. This can get a bit complex, and we suggest you pull up Metalink note 259379.1 for instructions on performing this operation.

STARTING OEM

- ✴ To start OEM, go to the URL provided when the DBCA created the database, or when you installed OEM with **emca**.
 - The typical URL is *http://machine_name:1158/em* as reported during database creation.
 - The port number will increase by 1 for each database on the system.
- ✴ When OEM first starts, you will need to log on to the database from the logon screen.
- ✴ When you use OEM to start up or shut down the database, you must provide both a database and an operating system logon.
- ✴ Oracle licenses different parts of OEM separately.
 - Basic OEM functionality comes with the database license.
 - Diagnostic and performance tuning packs are licensed separately from the base OEM product.
 - These packs are installed with OEM, even if you are not licensed for them.

Starting OEM

When we installed OEM, **emca** gave us the URL that we can use to connect to OEM as seen here:
The Database Control URL is *http://localhost:1158/em*.

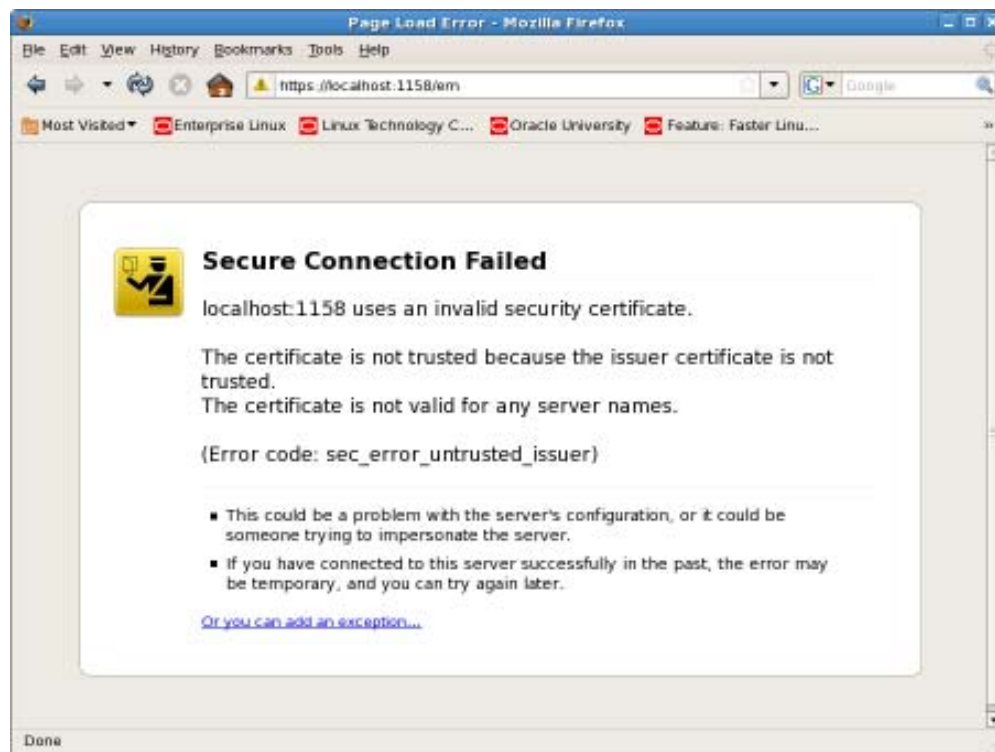
We are given a specific port, 1158. OEM uses port 1158 for the first OEM database install and then will increment by one port for each additional database installed. Once the install is complete, start OEM by typing the URL into your web browser. The first OEM screen you will see is the logon screen.

From the logon screen you can log into OEM using any Oracle account. Obviously, accounts with higher level of privileges (such as those with **SYSDBA** privileges) will be able to do much more with OEM than those accounts without such privileges.

OEM Licensing

The first time you log into the database with OEM, you will see the OEM Licensing Information screen. When you install OEM, you only are licensed for the base product functionality. Additional OEM packs (such as the database tuning pack), which are listed on this screen, require you pay additional licensing fees. Even though these additional license fees are required, Oracle still installs all of this functionality into each OEM install, so be careful about what functionality you use and make sure it is within the licenses you have purchased from Oracle.

When you log into OEM, you may find that the database is not started. OEM will give you the option to startup the database with a **startup** button. When you click the **startup** button, the startup/shutdown login page will appear. From this screen you provide both OS authentication privileges and the database logon privileges required to start the database being managed by OEM.



Hands On:

1. Start OEM in your web browser (the URL is something like *http://<hostname>:1158/em*).
2. Log on to the database as **SYS** using **SYSDBA** privileges.

INTRODUCING THE OEM HOME PAGES

- * Each OEM Grid Control "Target" (component) has a home page.
 - Database Home Page — This is the default home page.
 - Host Home Page — Contains host (OS) related information.
 - Listener Home Page — Contains information on the listener status.
- * The OEM Database Control Home page allows you to review the overall health of the database at a glance.
- * The OEM Database Control Home Page provides a summary peek into a number of database areas:
 - General database summary
 - System and database performance summaries
 - Space and availability summaries
 - Database alerts and diagnostic summary information
 - Job scheduler summary
- * In each home page tab, you can drill down for more detailed information.
- * Each home page also provides tabbed links to different OEM areas such as performance, availability, server and schema administration, maintenance, and more.
- * Each home page provides links to specific "high use" areas in the related links section.
- * Some home pages can be set to refresh automatically while some require manual refreshes.

Each of the home pages are similar in nature. After you log on to OEM you will find yourself at the OEM database home page. The OEM database home page provides a general overview of different types of database information. You can click on a given graphic to drill down for more information.

The other home pages (Host and Listener) can be navigated to from the database home page. Under the **General** section of the database home page you will find host and listener hot links to the host and listener home pages. Click on these links to get to those home pages.

On the database home page you will find a number of graphics that allow you to view and then drill down for more information on that area. For example, note the **HOST CPU** summary area on the OEM home page. This area contains a graphic that represents the current CPU usage on the system. Click on that graphic and OEM will take you to the host performance summary page which provides many more host performance related graphics and other related information such as CPU, memory and disk IO utilization. From these graphics you could further drill down to get more detailed system related information.

The OEM home page includes seven tabbed pages that provide access to the other OEM functionality. These links include:

- **Home** — Returns to the database home page.
- **Performance** — Displays the performance summary page.
- **Availability** — Backup and recovery settings and options.
- **Server** — Storage, configuration, scheduler, and other system wide configuration.
- **Schema** — Schema object management options.
- **Data Movement** — Data loading and export/import operations.
- **Software and Support** — Installation patching and Oracle support options.

At the bottom of each of the tabbed pages is an area called **related links**. These link to frequently used functions that you might want to utilize.

Hands On:

1. From the home page, click the **active sessions** graphic to display the **Performance** page.
2. Scroll to the **Additional Monitoring Links** section and click the **Top Activity** link.
3. Note the **Top SQL** and **Top Sessions** regions on the **Top Activity** page. These can be useful in determining which sessions are causing your database problems.
4. Return to the database home page by clicking on the **database** tab at the top right of the page.
5. Bring up the host home page by clicking on the **host** link in the general region.
6. Review the host home page format. Note that it is much like the database home page. Explore the different host tabbed pages (**performance**, **administration**, **targets**, **configuration**). Once you are done exploring, return to the database home page by clicking on the **database** tab at the top of the page.
7. Review the listener home page by clicking on the **listener** link that is in the general region.
8. Review the listener home page format. Note that it is much like the database home page. Explore the different host tabbed pages (**performance**, **serviced databases**). Once you are done exploring, return to the database home page by clicking on the **database** tab at the top of the page.

USING OEM – MANAGING THE JOB SCHEDULER

- ✴ OEM allows you to manage a variety of database functionality.
 - The "look and feel" of OEM is similar in each area.
 - By dealing with a few areas, you will become comfortable with the way OEM works.
- ✴ Oracle Database 10g has an integrated job scheduler.
 - The job scheduler can be complex.
 - OEM can manage the job scheduler for you.
- ✴ Access the job activity page from the **server** tab of OEM.
 - You will find a section titled Oracle Scheduler.
- ✴ The JOBS section is the principle page for the job scheduler.
 - The JOBS section allows you to add, delete, and modify jobs.
 - The JOBS links leads you to the scheduler jobs page.
 - The scheduler jobs page has links that allow you to monitor and manage:
 - Running jobs
 - Job History
 - Current jobs stored in the job scheduler

The job scheduler can get quite complex, so we will just be dealing with its base level of functionality in this topic to demonstrate how OEM helps make the management of jobs much easier. Let's create a job and see how the job scheduler interface of OEM works!

Hands On:

1. From the OEM database home page, click on the **server** tab. On the administration page, find the Oracle scheduler section.
2. Click on **Jobs** under the Oracle scheduler. You will see a number of jobs that are loaded in the job scheduler currently.
3. Click on the **running** tab at the top or bottom of the page. You will see the running jobs window. There are probably no jobs currently running.
4. Click on the **history** tab at the top or bottom of the page. You will see the history of the jobs that have already run.
5. Return to the **All** tab. Let's create a job. Click on the **create** button at the right of the page. This will bring up the create job page.
6. Let's create a PL/SQL job called **my_test_job**. Enter the following:
 - a. Name: **my_test_job**
 - b. Description: My first job scheduler job!
 - c. Leave the following defaults: Owner (**SYS**), Enabled (**YES**), Logging Level (Log job runs only (**RUNS**)), Job Class (**DEFAULT_JOB_CLASS**), Auto drop (**FALSE**), Restartable (**FALSE**).
 - d. Enter the following PL/SQL block:

```
declare
    v_sql          varchar2(2000);
begin
    v_sql:='create table my_tab (id number)';
    execute immediate v_sql;
end;
```

- e. Click on the **schedule** tab. The **Start Immediately** radio button should be selected.
 - f. Click **OK**. The job will be submitted. You are returned to the scheduler jobs page.
7. Click on the **history** tab and find the **my_test_job** job. Click on the job. This will bring up the job status page for the **my_test_job** job.
 8. Review the page and the contents of the page. Go to the **operation detail** region at the bottom. This area contains detailed logging information on the job runs. Your job run should have a **succeed** status. Click on the **log_id** and review the status screen.
 9. Log into the database as **SYS** and determine if the **MY_TAB** table was created by your job.
 10. Return to the scheduler jobs page and rerun the **my_test_job** job.
 - a. After the job runs, check the status and the job log. If it failed, why?
 11. Return to the scheduler jobs page and delete **my_test_job**.
 - a. Click on the radio button next to the **my_test_job**.
 - b. Click the **delete** button on the top of the region.
 - c. The job should be removed.

USING OEM – USING METRICS, ALERTS, AND THRESHOLDS

- * Oracle collects a number of database metrics.
 - Metrics are units of measurement used to track database performance.
- * Database metrics can have thresholds associated with them that indicate when they have exceeded specific allowable limits.
 - You can view and edit these thresholds from the **all metrics** link at the bottom of the database home page.
 - Some metrics have preset thresholds.
 - These default thresholds vary based on the version of Oracle you are running and the OS you are running on.
- * When a threshold is reached, an alert is triggered.
- * Alerts are detected by the OEM polling mechanisms, therefore alerting may not be instantaneous.
- * Alerts are reported in the **Alerts** and **Related Alerts** regions of the database home page.
 - Alerts pertain to database specific alerts
 - Related alerts pertain to associated components such as the OS or the listener.

Oracle does not currently support changing the polling frequency. Let's take a quick look at how you set and use thresholds in the Oracle database.

Hands On:

1. Open the OEM database home page.
2. Go to the **Alerts** region and review the alerts listed there. Database-specific alerts that are generated as a result of a threshold being violated will appear in this region.
3. Look at the **related alerts** region. This region will also provide alerts that might be generated as a result of a threshold being violated on related systems such as the listener, or the operating system (like running out of disk space).
4. First, let's force a threshold to alert. Let's check the current setting for blocked session reporting. From the **related links** region at the bottom of the database home page, click on the **metric and policy setting** link.
5. Under the metric find the metric titled **Database Time Spend Waiting (%)**. The metric should be set to a value of > 30 (seconds) by default.
6. Let's force a blocking session situation.
 - a. Open two SQL*Plus sessions and connect to the **SCOTT** schema.
 - b. From SQL*Plus session one, enter the following SQL statement:

```
update emp set ename='FREEMAN' where empno=7369;
```

- c. From SQL*Plus session two enter the following:

```
update emp set ename='SCOTT' where empno=7369;
```

- d. It may take up to a minute for the database to recognize the blocking situation and report it. Wait for 30 seconds and refresh the database home page, refresh it again in 30 seconds until the alert for blocking sessions appears. Eventually, you should see the **Database time spend waiting** alert appear in the **alerts** section.
 - e. Additionally, the blocking session will report in the **alerts** section as a blocking session (this takes longer to appear).
 - f. Commit the changes in the SQL*Plus sessions.
 - g. The alerts should clear from the database home page.
7. Let's change one of the threshold metrics. In this case, let's change the monitoring on the **tablespace space used (%)** threshold. The default values are a warning level at 85 and a critical level at 97%. Let's change these to 90 and 95%.
 - a. Open the metric and policy setting page.
 - b. Click on the **Edit Thresholds** button.
 - c. Find the **Tablespace Space Used (%)** metric.
 - d. Change the warning threshold to 50% (so we can simulate the alert), and the critical threshold to 95%.
 - e. Change the tablespace space used (%) metric for dictionary managed tablespaces to the same value.
 - f. Click **OK**.
 - g. Go to the database home page. An alert will appear if your **SYSTEM** tablespace usage is greater than 50%. This alert might take several minutes to register with OEM (in our case, it took over 15 minutes for the alert to actually show up!).

AUTOMATIC WORKLOAD REPOSITORY

- ✴ The Automatic Workload Repository collects performance statistics along with SQL text for all SQL statements executed in the database.
 - It is a historical performance data warehouse that stores SQL statement CPU, memory, and I/O resource consumption.
 - The AWR runs by default on an hourly schedule and does not add a noticeable level of overhead.
 - The information in this repository is used as input for diagnostic tools like Advisors and Automatic Workload Management, as well as for root cause analysis by ADDM.
- ✴ The new background process MMON takes snapshots at regular intervals; manual snapshots can also be performed.
 - This information is stored in the **SYSAUX** tablespace and named in the format **WRM\$_*** and **WRH\$_***.
 - Control information is stored in **DBA_HIST_WR_CONTROL**.
- ✴ The AWR has a report generation mechanism that produces a summary report on statistics, stored in the workload repository.
- ✴ The report-generation interface is also provided in the form of a SQL*Plus script (*awrrpt.sql*) that generates a general report with information on the overall system behavior over a selected time period.
 - The script generates a text file or an HTML report.
 - AWR reports can be generated and viewed using Enterprise Manager.
- ✴ AWR data is automatically purged data after eight days but this retention period can be altered if necessary.

When the Oracle Database 10g database is created, AWR is installed and scheduled jobs to collect AWR data are created. These jobs run in the job scheduler every hour by default and the data collected will be maintained for 7 days. You can modify the snapshot collection interval using the **dbms_workload_repository.modify_snapshot_settings** package.

Hands On:

Modify AWR so it will collect statistics every 15 minutes and retain the data for 2 weeks (20160 seconds).

1. Open a SQL*Plus session as **SYS**.
2. Execute the following SQL command to change the retention period to 14 days and the interval to 15 minutes:

```
EXEC dbms_workload_repository.modify_snapshot_settings
(retention=>20160, interval=> 15);
```

3. Check the **DBA_HIST_WR_CONTROL** view to make sure the change took effect.
4. Create a manual snapshot by executing the following SQL command:

```
EXEC dbms_workload_repository.create_snapshot;
```

5. Query the **DBA_HIST_SNAPSHOT** view to see the newly created snapshot:

```
SELECT snap_id, begin_interval_time, end_interval_time
FROM dba_hist_snapshot
ORDER BY 1;
```

6. Disable the automatic collection of snapshots:

```
EXEC dbms_scheduler.disable('GATHER_STATS_JOB');
```

7. Enable the automatic collection of snapshots:

```
EXEC dbms_scheduler.enable('GATHER_STATS_JOB');
```

8. From OEM go to the administration page. On the page find the link to the Automatic Workload Repository.
9. This brings up the AWR page. Change the snapshot retention to a period of 14 days. Change the snapshot collection interval to 30 minutes.
10. Click **OK**. This will return you to the AWR page. Note the changed snapshot information.

THE AUTOMATIC DATABASE DIAGNOSTIC MONITOR — ADDM

- * Once an AWR snapshot is taken, the ADDM analysis occurs automatically.
 - The **STATISTICS_LEVEL** parameter must be set to **TYPICAL** or **ALL**.
 - ADDM uses the AWR snapshots and other information to perform a database analysis.
 - The MMON process is responsible for ADDM analysis after each snapshot.
- * The results of an ADDM analysis run are stored in the AWR for later use.
 - Results are stored as findings.
 - There are three kinds of findings:
 - Problem — A problem indicates a root cause problem that is causing a database performance problem.
 - Symptom — A symptom indicates a performance issue that normally points to one or more specific problem findings.
 - Information — This is just basic database related information that is not related to a specific problem.
 - Findings are ranked by how much total DB Time they have consumed.
 - Each finding has one or more recommendations associated with them.
 - These recommendations can provide solutions to correcting the problem found in the finding.
 - Recommendations consist of two elements, the action and the justifying rationale.
 - Findings can be viewed in OEM or manually from the database.

Prior to Oracle Database 10g, the primary native Oracle approach to database analysis was a manual process involving the collection and analysis of statspack reports. The analysis of statspack reports was complex and often required an experienced DBA to deal with the harder-to-find problems. To try to rectify this problem, Oracle Database 10g introduced ADDM. ADDM provides an Oracle automated way to perform this analysis so that we can analyze many enterprise databases with results that were consistent. The purpose of ADDM is to help the beginning DBA determine what problems his database is experiencing and to provide solutions to correct those problems.

Hands On:

Let's look at using ADDM through OEM:

1. Go to the OEM database home page and connect as **SYSTEM**.
2. Click on Advisor Central from the related links region.
3. Down in the results section, you will find an ADDM advisory type. This is the last ADDM run. Click on the link under name (it should be named starting with ADDM).
4. Review the ADDM report, were there any impacts reported by ADDM?
5. The Database Activity region provides a graph that provides an image of the number of active sessions and the impacts of waits, CPU and IO on those sessions. The graph is related to the time of the associated AWR snapshot along the bottom. Along the bottom of this graph OEM provides images of when previous ADDM reports were executed automatically.
6. Click on the link that is marked icon key. This will provide you with some insight into the OEM icons on this page. Return to the previous ADDM page.
7. From the database activity region find a previously executed ADDM run with findings and click on it. Review the results of those ADDM runs.
8. Each finding has a link for that finding that will take you to a different OEM application page that will help you further drill down into that finding.
9. Click on the informational findings region and review the details on the ADDM findings.
10. Different links on the findings detail page will allow you to further drill down into the finding and determine a correct solution. Review the different finding detail pages for your ADDM runs.

OEM AND ADVISORS

- * OEM provides other advisors beyond ADDM, including:
 - The memory advisors — Provide help in managing instance memory.
 - The Materialized View Advisor — Provides help in the creation of materialized views and problem diagnosis in existing materialized views. (Not available in Advisor Central.)
 - The SQL Access Advisor — Provides information on potential additional structures (e.g. indexes and materialized views) that could improve SQL performance on objects in the system.
 - The SQL Tuning Advisor — Provides SQL tuning assistance for single statements.
 - The Segment Advisor — Provides diagnostic information on Oracle segments, such as when they might benefit from being rebuilt.
 - The UNDO Advisor — Provides administrative guidance on undo tablespace configuration. (Not available through Advisor Central.)
 - The MTTR Advisor — Provides administrative guidance on online redo log configuration.
- * The advisors provide the DBA with assistance in tuning and configuring the database.

Oracle started providing advisors in Oracle9i, and has been adding them ever since. In Oracle Database 10g, Oracle has provided a number of advisors to help the beginning or experienced DBA better manage the database. The advisors can be accessed from OEM or directly from the database. The advisors can help you size memory, the online redo logs, or tune SQL statements.

Hands On:

Let's look at one of the advisors right now.

1. Go to the OEM database home page and connect as **SYS**.
2. Click on **Advisor Central** from the related links region.
3. Click on the **memory advisors**.
4. From the memory advisors page, you can see how much memory has been allocated to the SGA, and you can see the setting of **SGA_MAX_SIZE**.
5. Click on the **PGA** tab. From here you can look at details related to PGA memory management. Click on **PGA Memory Usage Details**.
6. The PGA Memory Usage Details page will open in a new window. This page provides you with current PGA memory execution statistics and can also provide you with extrapolated memory usage estimates.
7. Return to the SGA page by closing the PGA Memory Usage Details window.
8. Click on the **advice** button. The Memory Size Advice page will open. This page makes recommendations as to how much memory should be allocated to the SGA. Note that the graph indicates that you should increase the amount of memory to the SGA to get better performance. We can't increase the amount of memory currently because **SGA_MAX_SIZE** is set too low. We need to increase **SGA_MAX_SIZE**, and then we can increase the amount of memory allocated to the SGA.
9. Return to the Memory Advisor home page.
10. Add 50 MB to the maximum memory size of your database. Click on the **show SQL** button to see the SQL that will be generated. Click **Return** to redisplay the previous screen.
11. Click the **apply** button. OEM will make the change in the database.
12. Click **yes** to restart the database.
13. Log in to shutdown the database. Click the checkbox to save preferred credentials. Wait for a few moments and then refresh the page. The Database home page will appear indicating the database instance is down.
14. Wait for the instance to restart, then log in as **SYS**.
15. Click on **advisor central**, and then on **memory advisors** again.
16. Click on the **advice** button. The SGA Size Advice page will open. Note that there is now a difference between the current SGA size and the maximum SGA size of the database. However, the memory advisor recommendations are not that good at this point since we restarted the database and therefore, have no relevant workload history.
17. Change the SGA size, by adding 10MB to it.
18. Apply the change.

THE SQL ACCESS ADVISOR

- * The *SQL Access Advisor* is available to help you find indexes, materialized views, or materialized view logs that can be used to improve access to Oracle database objects.
- * The SQL Access Advisor can be used to proactively tune your database based on single SQL statements or database workload.
- * The SQL Access Advisor helps the newer DBA who may not understand the database model to find SQL tuning opportunities.
- * The SQL Access Advisor can provide tuning recommendations based on:
 - Information in the AWR repository (Current and recent SQL activity).
 - A SQL Tuning set.
 - User-defined SQL statements.
 - A list of specific tables.
- * You can further filter the SQL worksets based on a number of criteria, such as:
 - Resource consumption.
 - Specific users.
 - Specific tables that are accessed by SQL statements.
 - Specific text in SQL statements.

It can be very difficult to make decisions about where to place indexes or where to create materialized views in your database. Often this requires that you be well acquainted with the data and how it is being used. This can be difficult for even the most experienced DBA who is new on a site, or for a Junior DBA who is just learning how Oracle works.

The SQL Access Advisor will analyze your database and make recommendations on where to add indexes and materialized views. You then review those recommendations and make those changes that make the most sense. The SQL Access Advisor is just an advisor, so you need to carefully review its results before you implement them. It can, however, be one tool that can lead to greater success in a database tuning effort.

Hands On:

1. From the database home page, connected as **SYSTEM**, select **advisor central** from the related links region.
2. From the advisors region of the advisor central page, select **SQL Advisors** and **SQL Access Advisor**.
3. From the SQL Access Advisor page click **continue**.
4. From the Workload Source page select **Current and Recent SQL Activity** and click **next**.
5. From the Recommendation Options select **Both Indexes and Materialized Views**. Also select **limited mode** for the Advisor Mode. Click **Next**.
6. From the schedule page, take the default values and click **Next**.
7. From the review page, click the **Submit** button to begin the SQL Access advisor.
8. The Advisor Central home page will appear. In the results region you should see your SQL Access Advisor session appear with a status of **CREATED** or **RUNNING**. Wait about 15 seconds and click **refresh**. Continue to refresh until the status reads **COMPLETED**.
9. Once the status is **COMPLETED**, click on the name of the advisor to see the results.
10. The result page for the advisor task will appear. Review the first page, which is a summary overview of the recommendations.
11. Click on the **Recommendations** tab. Each recommendation is listed on the bar graph in order of the total cost improvement that Oracle thinks that recommendation will result in.
12. If a recommendation appears in the **Select recommendations for implementation**, review those recommendations.
13. Click on the **SQL Statements** tab. This provides a list of SQL statements that will benefit from the recommendations made in the **Recommendations** page.
14. Click on the **details** tab. This provides a summary of the options that were used when the advisor task was executed.

