

Oracle® Database

2 Day + Security Guide

12c Release 2 (12.2)

E49712-17

April 2017

Oracle Database 2 Day + Security Guide, 12c Release 2 (12.2)

E49712-17

Copyright © 2006, 2017, Oracle and/or its affiliates. All rights reserved.

Primary Author: Patricia Huey

Contributors: Todd Bottger, Peter Knaggs, Rahil Mir, Gopal Mulagund, Paul Needham, Deborah Owens, Dinesh Rajasekharan, Saikat Saha, Kamal Tbeileh, Peter Wahl, Alan Williams

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

| | |
|---|------|
| Preface | ix |
| Audience | ix |
| Documentation Accessibility | ix |
| Related Documents..... | x |
| Conventions..... | x |
| | |
| Changes in This Release for Oracle Database 2 Day + Security Guide | xiii |
| Changes in Oracle Database 12c Release 2 (12.2)..... | xiii |
| New Features | xiii |
| Desupported Features | xiv |
| | |
| 1 Introduction to Oracle Database Security | |
| 1.1 About This Guide | 1-1 |
| 1.1.1 Before Using This Guide | 1-1 |
| 1.1.2 What This Guide Is and Is Not..... | 1-2 |
| 1.2 Common Database Security Tasks..... | 1-2 |
| 1.3 Tools for Securing Your Database..... | 1-2 |
| 1.4 Securing Your Database: A Roadmap | 1-3 |
| | |
| 2 Securing the Database Installation and Configuration | |
| 2.1 About Securing the Database Installation and Configuration..... | 2-1 |
| 2.2 Securing Access to the Oracle Database Installation..... | 2-1 |
| 2.2.1 Default Security Settings | 2-2 |
| 2.2.2 Security for the Oracle Data Dictionary | 2-3 |
| 2.2.3 Initialization Parameters Used for Installation and Configuration Security..... | 2-5 |
| 2.2.4 Modifying the Value of an Initialization Parameter | 2-5 |
| 2.3 Security for the Network | 2-6 |
| 2.3.1 About Securing the Network..... | 2-6 |
| 2.3.2 Protecting Data on the Network by Using Network Encryption..... | 2-6 |
| 2.3.3 Initialization Parameters Used for Network Security..... | 2-10 |
| 2.4 Securing User Accounts..... | 2-10 |

| | | |
|----------|---|------|
| 2.4.1 | About Securing Oracle Database User Accounts | 2-11 |
| 2.4.2 | Predefined User Accounts Provided by Oracle Database | 2-11 |
| 2.4.3 | Expiring and Locking Database Accounts..... | 2-17 |
| 2.4.4 | Requirements for Creating Passwords..... | 2-17 |
| 2.4.5 | Finding and Changing Default Passwords..... | 2-18 |
| 2.4.6 | Parameters Used to Secure User Accounts..... | 2-20 |
| 3 | Managing User Privileges | |
| 3.1 | About Privilege Management..... | 3-1 |
| 3.2 | When to Grant Privileges to Users | 3-1 |
| 3.3 | When to Grant Roles to Users..... | 3-2 |
| 3.4 | Controlling Access to Applications with Secure Application Roles | 3-2 |
| 3.4.1 | About Secure Application Roles | 3-3 |
| 3.4.2 | Tutorial: Creating a Secure Application Role..... | 3-4 |
| 3.5 | Initialization Parameters Used for Privilege Security | 3-13 |
| 4 | Encrypting Data with Oracle Transparent Data Encryption | |
| 4.1 | About Encrypting Sensitive Data | 4-1 |
| 4.2 | When Should You Encrypt Data?..... | 4-2 |
| 4.3 | How Transparent Data Encryption Works | 4-2 |
| 4.4 | Configuring Data to Use Transparent Data Encryption | 4-4 |
| 4.4.1 | Step 1: Configure the Keystore Location..... | 4-4 |
| 4.4.2 | Step 2: Check the COMPATIBLE Initialization Parameter Setting..... | 4-5 |
| 4.4.3 | Step 3: Create the Software Password-Based Keystore | 4-6 |
| 4.4.4 | Step 4: Open (or Close) the Keystore..... | 4-7 |
| 4.4.5 | Step 5: Create the Master Encryption Key | 4-8 |
| 4.4.6 | Step 6: Encrypt Data..... | 4-8 |
| 4.5 | Checking Existing Encrypted Data | 4-12 |
| 4.5.1 | Finding the Type of Keystore That Was Created | 4-13 |
| 4.5.2 | Finding the Keystore Location | 4-13 |
| 4.5.3 | Checking Whether a Keystore Is Open or Closed | 4-13 |
| 4.5.4 | Checking Encrypted Columns of an Individual Table | 4-13 |
| 4.5.5 | Checking All Encrypted Table Columns in the Current Database Instance | 4-14 |
| 4.5.6 | Data Dictionary Views for Checking Encrypted Tablespace | 4-14 |
| 5 | Controlling Access with Oracle Database Vault | |
| 5.1 | About Oracle Database Vault..... | 5-1 |
| 5.2 | Tutorial: Controlling Administrator Access to a User Schema..... | 5-2 |
| 5.2.1 | Step 1: Enable Oracle Database Vault..... | 5-3 |
| 5.2.2 | Step 2: Grant SELECT on the OE.CUSTOMERS Table to User SCOTT | 5-3 |
| 5.2.3 | Step 3: Select from the OE.CUSTOMERS Table as Users SYS and SCOTT..... | 5-5 |
| 5.2.4 | Step 4: Create a Realm to Protect the OE.CUSTOMERS Table..... | 5-5 |
| 5.2.5 | Step 5: Test the OE Protections Realm..... | 5-7 |

| | | |
|----------|---|------|
| 5.2.6 | Step 6: Optionally, Remove the Components for This Tutorial | 5-8 |
| 6 | Restricting Access with Oracle Virtual Private Database | |
| 6.1 | About Oracle Virtual Private Database | 6-1 |
| 6.2 | Tutorial: Limiting Access to Data Based on the Querying User | 6-3 |
| 6.2.1 | About Limiting Access to Data Based on the Querying User..... | 6-3 |
| 6.2.2 | Step 1: Create User Accounts for This Tutorial..... | 6-4 |
| 6.2.3 | Step 2: If Necessary, Create the Security Administrator Account..... | 6-5 |
| 6.2.4 | Step 3: Update the Security Administrator Account..... | 6-5 |
| 6.2.5 | Step 4: Create the F_POLICY_ORDERS Policy Function | 6-6 |
| 6.2.6 | Step 5: Create the ACCESSCONTROL_ORDERS Virtual Private Database Policy | 6-8 |
| 6.2.7 | Step 6: Test the ACCESSCONTROL_ORDERS Virtual Private Database Policy | 6-9 |
| 6.2.8 | Step 7: Optionally, Remove the Components for This Tutorial | 6-10 |
| 7 | Limiting Access to Sensitive Data Using Oracle Data Redaction | |
| 7.1 | About Oracle Data Redaction | 7-1 |
| 7.2 | Tutorial: Redacting Data for a Select Group of Users | 7-2 |
| 7.2.1 | About Redacting Data for a Select Group of Users..... | 7-2 |
| 7.2.2 | Step 1: Create User Accounts and Grant Them the Necessary Privileges | 7-3 |
| 7.2.3 | Step 2: Create and Populate the SALES_OPPTS Sales Opportunities Table..... | 7-5 |
| 7.2.4 | Step 3: Create the SALES_OPPTS_POL Oracle Data Redaction Policy | 7-6 |
| 7.2.5 | Step 5: Test the SALES_OPPTS_POL Oracle Data Redaction Policy | 7-7 |
| 7.2.6 | Step 6: Optionally, Remove the Components for This Tutorial | 7-9 |
| 8 | Enforcing Row-Level Security with Oracle Label Security | |
| 8.1 | About Oracle Label Security | 8-1 |
| 8.2 | Virtual Private Database, Oracle Label Security, and Data Redaction Differences | 8-2 |
| 8.3 | Guidelines for Planning an Oracle Label Security Policy | 8-4 |
| 8.4 | Tutorial: Creating Levels of Access to Table Data Based on the User | 8-5 |
| 8.4.1 | About Creating Levels of Access to Table Data Based on the User..... | 8-6 |
| 8.4.2 | Step 1: Enable Oracle Label Security | 8-6 |
| 8.4.3 | Step 2: Enable the LBACSYS Account..... | 8-7 |
| 8.4.4 | Step 3: Create a Role and Three Users for the Oracle Label Security Tutorial..... | 8-8 |
| 8.4.5 | Step 4: Create the ACCESS_LOCATIONS Oracle Label Security Policy | 8-9 |
| 8.4.6 | Step 5: Define the ACCESS_LOCATIONS Policy-Level Components..... | 8-11 |
| 8.4.7 | Step 6: Create the ACCESS_LOCATIONS Policy Data Labels..... | 8-12 |
| 8.4.8 | Step 7: Create the ACCESS_LOCATIONS Policy User Authorizations..... | 8-13 |
| 8.4.9 | Step 8: Apply the ACCESS_LOCATIONS Policy to the HR.LOCATIONS Table | 8-16 |
| 8.4.10 | Step 9: Add the ACCESS_LOCATIONS Labels to the HR.LOCATIONS Data..... | 8-16 |
| 8.4.11 | Step 10: Test the ACCESS_LOCATIONS Policy | 8-19 |
| 8.4.12 | Step 11: Optionally, Remove the Components for This Tutorial | 8-21 |

9 Auditing Database Activity

| | |
|--|-----|
| 9.1 About Auditing..... | 9-1 |
| 9.2 Why Is Auditing Used?..... | 9-2 |
| 9.3 Tutorial: Creating a Unified Audit Policy..... | 9-3 |
| 9.3.1 Step 1: If Necessary, Enable Unified Auditing..... | 9-4 |
| 9.3.2 Step 2: Grant the SEC_ADMIN User the AUDIT_ADMIN Role..... | 9-5 |
| 9.3.3 Step 3: Create and Enable a Unified Audit Policy..... | 9-6 |
| 9.3.4 Step 4: Test the Unified Audit Policy..... | 9-7 |
| 9.3.5 Step 5: Optionally, Remove the Components for This Tutorial..... | 9-9 |
| 9.3.6 Step 6: Optionally, Remove the SEC_ADMIN Security Administrator Account..... | 9-9 |

Index

List of Tables

| | | |
|-----|---|------|
| 2-1 | Default Security Settings for Initialization and Profile Parameters..... | 2-2 |
| 2-2 | Initialization Parameters Used for Installation and Configuration Security..... | 2-5 |
| 2-3 | Initialization Parameters Used for Network Security..... | 2-10 |
| 2-4 | Predefined Oracle Database Administrative User Accounts..... | 2-12 |
| 2-5 | Predefined Oracle Database Non-Administrative User Accounts..... | 2-15 |
| 2-6 | Default Sample Schema User Accounts..... | 2-16 |
| 2-7 | Initialization and Profile Parameters Used for User Account Security..... | 2-20 |
| 3-1 | Initialization Parameters Used for Privilege Security..... | 3-14 |
| 4-1 | Data Dictionary Views for Encrypted Tablespaces..... | 4-15 |
| 8-1 | Comparing Virtual Private Database, Label Security, and Data Redaction..... | 8-3 |
| 8-2 | Values for Oracle Label Security Levels..... | 8-12 |

Preface

Welcome to *Oracle Database 2 Day + Security Guide*. This guide is for anyone who wants to perform common day-to-day security tasks with Oracle Database.

[Audience](#) (page ix)

[Documentation Accessibility](#) (page ix)

[Related Documents](#) (page x)

[Conventions](#) (page x)

Audience

Oracle Database 2 Day + Security Guide provides an introduction to configuring security in a default database. This guide expands on the security knowledge that you learned in *Oracle Database 2 Day DBA* to manage security in Oracle Database. The information in this guide applies to all platforms. For platform-specific information, see the installation guide, configuration guide, and platform guide for your platform.

This guide is intended for the following users:

- Oracle database administrators who want to acquire database security administrative skills
- Database administrators who have some security administrative knowledge but are new to Oracle Database

This guide is not an exhaustive discussion about security. For detailed information about security, see the Oracle Database Security documentation set. This guide does not provide information about security for Oracle E-Business Suite applications. For information about security in the Oracle E-Business Suite applications, see the documentation for those products.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc>.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/pls/>

[topic/lookup?ctx=acc&id=info](#) or visit <http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs> if you are hearing impaired.

Related Documents

For more information, you can refer to other security-related documentation in this library, Oracle Technology Network (OTN), and My Oracle Support.

Oracle Database Documentation

For more security-related information, see the following documents in the Oracle Database documentation set:

- *Oracle Database 2 Day DBA*
- *Oracle Database Administrator's Guide*
- *Oracle Database Security Guide*
- *Oracle Database Concepts*
- *Oracle Database Reference*
- *Oracle Database Vault Administrator's Guide*

Many of the examples in this guide use the sample schemas of the seed database, which is installed by default when you install Oracle. See *Oracle Database Sample Schemas* for information about how these schemas were created and how you can use them.

Oracle Technology Network (OTN)

You can download free release notes, installation documentation, updated versions of this guide, white papers, or other collateral from the Oracle Technology Network (OTN).

Visit:

<http://www.oracle.com/technetwork/index.html>

For security-specific information on OTN, visit

<http://www.oracle.com/technetwork/topics/security/whatsnew/index.html>

For the latest version of the Oracle documentation, including this guide, visit

<http://www.oracle.com/technetwork/documentation/index.html>

My Oracle Support (formerly OracleMetaLink)

You can find information about security patches, certifications, and the support knowledge base by visiting My Oracle Support at

<https://support.oracle.com>

Conventions

This document uses the standard Oracle Database documentation style conventions.

| Convention | Meaning |
|-------------------|--|
| boldface | Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary. |
| <i>italic</i> | Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values. |
| monospace | Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter. |

Changes in This Release for Oracle Database 2 Day + Security Guide

Oracle Database 2 Day + Security Guide has updates that affect Transparent Data Encryption, Oracle Database Vault, Oracle Data Redaction, Oracle Label Security, and auditing.

[Changes in Oracle Database 12c Release 2 \(12.2\)](#) (page xiii)

Changes in Oracle Database 12c Release 2 (12.2)

The following are changes in *Oracle Database 2 Day + Security Guide* for Oracle Database 12c release 2 (12.2):

[New Features](#) (page xiii)

[Desupported Features](#) (page xiv)

New Features

The following new features affect *Oracle Database 2 Day + Security Guide*:

- New initialization parameter to secure user accounts

Starting with this release, you can use the `INACTIVE_ACCOUNT_TIME` parameter to automatically lock the account of a database user who has not logged in to the database instance in a specified number of days.

See [Parameters Used to Secure User Accounts](#) (page 2-20) for more information.

- The following parameters have been changed to accommodate the Security Technical Implementation Guide (STIG) requirements, which standardize protocols that are used to enforce security:

- The default for `SEC_PROTOCOL_ERROR_FURTHER_ACTION` is now `(DROP , 3)`.

- The default for `SEC_MAX_FAILED_LOGIN_ATTEMPTS` is now 3.

- The default for `SQL92_SECURITY_PARAMETER` is now `TRUE`.

See *Oracle Database Reference* for more information about initialization parameters.

- Oracle Data Redaction enhancements

Oracle Data Redaction provides several new features for this release, including the ability to redact column data by replacing it with null values, and the ability to create a central library of regular expressions used in Data Redaction policies.

See [About Oracle Data Redaction](#) (page 7-1) for more information.

Desupported Features

The following feature has been desupported for this release:

- `MAX_ENABLED_ROLES` initialization parameter, because it has not been used since Oracle Database Release 10g

Introduction to Oracle Database Security

In a default Oracle Database installation, you can manage security using tools such as authentication, encryption, Oracle Data Redaction, and Oracle Database Vault.

[About This Guide](#) (page 1-1)

Oracle Database 2 Day + Security Guide teaches you how to perform day-to-day database security tasks.

[Common Database Security Tasks](#) (page 1-2)

Database administrators for Oracle Database are responsible for common security-related tasks.

[Tools for Securing Your Database](#) (page 1-2)

To achieve the goals of securing your Oracle database, you must use a specific set of products, tools, and utilities.

[Securing Your Database: A Roadmap](#) (page 1-3)

To learn the fundamentals of securing an Oracle database, you should follow a roadmap of specific tasks.

1.1 About This Guide

Oracle Database 2 Day + Security Guide teaches you how to perform day-to-day database security tasks.

[Before Using This Guide](#) (page 1-1)

Before using this guide, you should understand the basics of administering an Oracle database.

[What This Guide Is and Is Not](#) (page 1-2)

The objective of *Oracle Database 2 Day + Security Guide* is to use a task-oriented approach to describe how to perform default security tasks.

1.1.1 Before Using This Guide

Before using this guide, you should understand the basics of administering an Oracle database.

The goal of this guide is to help you understand the concepts behind Oracle Database security. You will learn how to perform common security tasks needed to secure your database. The knowledge you gain from completing the tasks in *Oracle Database 2 Day + Security Guide* helps you to better secure your data and to meet common regulatory compliance requirements, such as the Sarbanes-Oxley Act.

The primary administrative interface used in this guide is Oracle Enterprise Manager, featuring all the self-management capabilities introduced in Oracle Database.

Complete *Oracle Database 2 Day DBA*, which provides a good foundation for Oracle Database administration.

You should also obtain the necessary products and tools described in [Tools for Securing Your Database](#) (page 1-2).

1.1.2 What This Guide Is and Is Not

The objective of *Oracle Database 2 Day + Security Guide* is to use a task-oriented approach to describe how to perform default security tasks.

Where appropriate, this guide describes the concepts and steps necessary to understand and complete a task. This guide is not an exhaustive discussion of all Oracle Database concepts. For this type of information, see *Oracle Database Concepts*.

Where appropriate, this guide describes the necessary Oracle Database administrative steps to complete security tasks. This guide does not describe basic Oracle Database administrative tasks. For this type of information, see *Oracle Database 2 Day DBA*. Additionally, for a complete discussion of administrative tasks, see *Oracle Database Administrator's Guide*.

In addition, this guide is not an exhaustive discussion of all Oracle Database security features and does not describe available APIs that provide equivalent command line functionality to the tools used in this guide. For this type of information, see *Oracle Database Security Guide*.

1.2 Common Database Security Tasks

Database administrators for Oracle Database are responsible for common security-related tasks.

These tasks are as follows:

- Ensuring that the database installation and configuration is secure
- Managing the security aspects of user accounts: developing secure password policies, creating and assigning roles, restricting data access to only the appropriate users, and so on
- Ensuring that network connections are secure
- Encrypting sensitive data
- Ensuring the database has no security vulnerabilities and is protected against intruders
- Deciding what database components to audit and how granular you want this auditing to be
- Downloading and installing security patches

In a small to midsize database environment, you might perform these tasks as well and all database administrator-related tasks, such as installing Oracle software, creating databases, monitoring performance, and so on. In large, enterprise environments, the job is often divided among several database administrators—each with their own specialty—such as database security or database tuning.

1.3 Tools for Securing Your Database

To achieve the goals of securing your Oracle database, you must use a specific set of products, tools, and utilities.

These tools are as follows:

- **Oracle Database 12c Enterprise Edition**

Oracle Database 12c Enterprise Edition provides enterprise-class performance, scalability, and reliability on clustered and single-server configurations. It includes many security features that are used in this guide.
- **Oracle Enterprise Manager**

Oracle Enterprise Manager is a Web application that you can use to perform database administrative tasks for a single database instance or a clustered database. It enables you to manage multiple Oracle databases from one location. This guide explains how to use Enterprise Manager to perform database administrative tasks.
- **SQL*Plus**

SQL*Plus is a development environment that you can use to create and run SQL and PL/SQL code. It is part of the Oracle Database 12c release 1 (12.1) installation.
- **Database Configuration Assistant (DBCA)**

Database Configuration Assistant enables you to perform general database tasks, such as creating, configuring, or deleting databases. In this guide, you use DBCA to enable default auditing.
- **Oracle Net Manager**

Oracle Net Manager enables you to perform network-related tasks for Oracle Database. In this guide, you use Oracle Net Manager to configure network encryption.

1.4 Securing Your Database: A Roadmap

To learn the fundamentals of securing an Oracle database, you should follow a roadmap of specific tasks.

To use this guide:

1. Secure your Oracle Database installation and configuration.

Complete the tasks in [Securing the Database Installation and Configuration](#) (page 2-1) to secure access to an Oracle Database installation.
2. Understand how privileges work.

Complete the tasks in [Managing User Privileges](#) (page 3-1). You learn about the following:

 - How privileges work
 - Why you must be careful about granting privileges
 - How database roles work
 - How to create secure application roles
3. Encrypt data as it travels across the network.

Complete the tasks in [Encrypting Data with Oracle Transparent Data Encryption](#) (page 4-1) to learn how to secure client connections and to configure network encryption.

- 4.** Control system administrative access to sensitive data with Oracle Database Vault.
Complete the tasks in [Controlling Access with Oracle Database Vault](#) (page 5-1).
- 5.** Restrict the display of data with Oracle Virtual Private Database.
Complete the tasks in [Restricting Access with Oracle Virtual Private Database](#) (page 6-1).
- 6.** Control the display of data in real time by using data redaction.
Complete the tasks in [Limiting Access to Sensitive Data Using Oracle Data Redaction](#) (page 7-1).
- 7.** Enforce row-level security with Oracle Label Security.
[Enforcing Row-Level Security with Oracle Label Security](#) (page 8-1).
- 8.** Configure auditing so that you can monitor the database activities.
Complete the tasks in [Auditing Database Activity](#) (page 9-1) to learn about standard auditing.

Securing the Database Installation and Configuration

You should secure the Oracle Database installation, the network it users, and database user accounts.

[About Securing the Database Installation and Configuration](#) (page 2-1)

After you install Oracle Database, you should secure the database installation and configuration.

[Securing Access to the Oracle Database Installation](#) (page 2-1)

Oracle Database provides default security settings and initialization parameters to secure your installation.

[Security for the Network](#) (page 2-6)

Oracle Database provides ways that to protect client connections and encrypt data that travels through the network between the client and the server.

[Securing User Accounts](#) (page 2-10)

You can secure user accounts by creating secure passwords, changing default passwords, and using special parameters to further secure user accounts.

2.1 About Securing the Database Installation and Configuration

After you install Oracle Database, you should secure the database installation and configuration.

Oracle provides commonly used ways to do secure the database installation and configuration, all of which involve restricting permissions to specific areas of the database files.

Oracle Database is available on several operating systems. Consult the following guides for detailed platform-specific information about Oracle Database:

- *Oracle Database Platform Guide for Microsoft Windows*
- *Oracle Database Administrator's Reference for Linux and UNIX-Based Operating Systems*
- *Oracle Database Installation Guide for your platform*

2.2 Securing Access to the Oracle Database Installation

Oracle Database provides default security settings and initialization parameters to secure your installation.

[Default Security Settings](#) (page 2-2)

When you create a new database, Oracle Database provides a set of default security settings.

[Security for the Oracle Data Dictionary](#) (page 2-3)

The data dictionary is a set of database tables that provide information about the database, such as schema definitions or default values.

[Initialization Parameters Used for Installation and Configuration Security](#) (page 2-5)

Oracle Database provides initialization parameters to control installation and configuration security.

[Modifying the Value of an Initialization Parameter](#) (page 2-5)

You can use Enterprise Manager to modify the value of an initialization parameter.

2.2.1 Default Security Settings

When you create a new database, Oracle Database provides a set of default security settings.

These default security settings are as follows:

- **Enables default auditing settings.** See *Oracle Database Security Guide* for detailed information.
- **Creates stronger enforcements for new or changed passwords.** [Requirements for Creating Passwords](#) (page 2-17) describes the new password requirements.
- **Removes the CREATE EXTERNAL JOB privilege from the PUBLIC role.** For greater security, grant the CREATE EXTERNAL JOB privilege only to SYS, database administrators, and those trusted users who need it.
- **Sets security-related initialization and profile parameter settings.** [Table 2-1](#) (page 2-2) lists the default parameter settings.

Table 2-1 Default Security Settings for Initialization and Profile Parameters

| Setting | Default |
|-----------------------------|-----------|
| O7_DICTIONARY_ACCESSIBILITY | FALSE |
| PASSWORD_GRACE_TIME | 7 |
| PASSWORD_LOCK_TIME | 1 |
| FAILED_LOGIN_ATTEMPTS | 10 |
| PASSWORD_LIFE_TIME | 180 |
| PASSWORD_REUSE_MAX | UNLIMITED |
| PASSWORD_REUSE_TIME | UNLIMITED |
| REMOTE_OS_ROLES | FALSE |

2.2.2 Security for the Oracle Data Dictionary

The data dictionary is a set of database tables that provide information about the database, such as schema definitions or default values.

[About the Oracle Data Dictionary](#) (page 2-3)

The Oracle data dictionary contains information such as the names and privileges of Oracle Database users.

[Enabling Data Dictionary Protection](#) (page 2-4)

Setting the `O7_DICTIONARY_ACCESSIBILITY` initialization parameter to `FALSE` protects the data dictionary.

2.2.2.1 About the Oracle Data Dictionary

The Oracle data dictionary contains information such as the names and privileges of Oracle Database users.

The data dictionary has the following contents:

- The names of Oracle Database users
- Privileges and roles granted to each user
- The definitions of all schema objects in the database (tables, views, indexes, clusters, synonyms, sequences, procedures, functions, packages, triggers, and so on)
- The amount of space allocated for, and is currently used by, the schema objects
- Default values for columns
- Integrity constraint information
- Auditing information, such as who has accessed or updated various schema objects
- Other general database information

The data dictionary tables and views for a given database are stored in the `SYSTEM` tablespace for that database. All the data dictionary tables and views for a given database are owned by the user `SYS`. Connecting to the database with the `SYSDBA` administrative privilege gives full access to the data dictionary. Oracle strongly recommends limiting access to the `SYSDBA` administrative privilege to only those operations necessary such as patching and other administrative operations. The data dictionary is central to every Oracle database.

You can view the contents of the data dictionary by querying data dictionary views, which are described in *Oracle Database Reference*. Be aware that not all objects in the data dictionary are exposed to users. A subset of data dictionary objects, such as those beginning with `USER_` are exposed as read only to all database users.

[Example 2-1](#) (page 2-3) shows how you can find a list of database views specific to the data dictionary by querying the `DICTIONARY` view.

Example 2-1 Finding Views That Pertain to the Data Dictionary

```
sqlplus system
Enter password: password
```

```
SQL> SELECT TABLE_NAME FROM DICTIONARY;
```

2.2.2.2 Enabling Data Dictionary Protection

Setting the `O7_DICTIONARY_ACCESSIBILITY` initialization parameter to `FALSE` protects the data dictionary.

The `O7_DICTIONARY_ACCESSIBILITY` parameter prevents users who have the `ANY` system privilege from using those privileges on the data dictionary, that is, on objects in the `SYS` schema.

Oracle Database provides highly granular privileges. One such privilege, commonly referred to as the `ANY` privilege, should typically be granted to only application owners and individual database administrators. For example, you could grant the `DROP ANY TABLE` privilege to an application owner. You can protect the Oracle data dictionary from accidental or malicious use of the `ANY` privilege by turning on or off the `O7_DICTIONARY_ACCESSIBILITY` initialization parameter.

To enable data dictionary protection:

1. Access the Database home page.
See *Oracle Database 2 Day DBA* for more information.
2. From the **Administration** menu, select **Initialization Parameters**.
If the Database Login page appears, then log in as `SYS` with the **SYSDBA** role selected.
3. In the Initialization Parameters page, from the list, search for `O7_DICTIONARY_ACCESSIBILITY`.
In the **Name** field, enter `O7_` (the letter `O`), and then click **Go**. You can enter the first few characters of a parameter name. In this case, `O7_` displays the `O7_DICTIONARY_ACCESSIBILITY` parameter.
4. Set the value for `O7_DICTIONARY_ACCESSIBILITY` to `FALSE`.
5. Click **Apply**.
6. Restart the Oracle Database instance.

```
sqlplus sys as sysdba  
Enter password: password
```

```
SQL> SHUTDOWN IMMEDIATE  
SQL> STARTUP
```

Note:

- In a default installation, the `O7_DICTIONARY_ACCESSIBILITY` parameter is set to `FALSE`.
 - The `SELECT ANY DICTIONARY` privilege is not included in the `GRANT ALL PRIVILEGES` statement, but you can grant it through a role. Roles are described in [When to Grant Roles to Users](#) (page 3-2) and *Oracle Database Security Guide*.
-
-

2.2.3 Initialization Parameters Used for Installation and Configuration Security

Oracle Database provides initialization parameters to control installation and configuration security.

[Table 2-2](#) (page 2-5) lists these initialization parameters.

Table 2-2 Initialization Parameters Used for Installation and Configuration Security

| Initialization Parameter | Default | Description |
|----------------------------------|---------|---|
| SEC_RETURN_SERVER_RELEASE_BANNER | FALSE | Controls the display of the product version information, such as the release number, in a client connection. An intruder could use the database release number to find information about security vulnerabilities that may be present in the database software. You can enable or disable the detailed product version display by setting this parameter. See <i>Oracle Database Security Guide</i> for more information about this and similar parameters. <i>Oracle Database Reference</i> describes this parameter in detail. |
| O7_DICTIONARY_ACCESSIBILITY | FALSE | Controls restrictions on SYSTEM privileges. See Enabling Data Dictionary Protection (page 2-4) for more information about this parameter. <i>Oracle Database Reference</i> describes this parameter in detail. |

2.2.4 Modifying the Value of an Initialization Parameter

You can use Enterprise Manager to modify the value of an initialization parameter.

To modify the value of an initialization parameter:

1. Access the Database home page.
See *Oracle Database 2 Day DBA* for more information.
2. From the **Administration** menu, select **Initialization Parameters**.
If the Database Login page appears, then log in as SYS with the **SYSDBA** role selected.
3. In the Initialization Parameters page, in the **Name** field, enter the name of the parameter to change, and then click **Go**.

You can enter the first few letters of the parameter, for example, SEC_RETURN if you are searching for the SEC_RETURN_SERVER_RELEASE_NUMBER parameter. Alternatively, you can scroll down the list of parameters to find the parameter you want to change. The text is not case sensitive.
4. In the **Value** field, either enter the new value or if a list is presented, select from the list.
5. Click **Apply**.
6. If the parameter is static, then restart the Oracle Database instance.

```
sqlplus sys as sysdba
Enter password: password
```

```
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
```

To find out if an initialization parameter is static, check its description in *Oracle Database Reference*. If the Modifiable setting in its summary table shows No, then you must restart the database instance.

2.3 Security for the Network

Oracle Database provides ways that to protect client connections and encrypt data that travels through the network between the client and the server.

[About Securing the Network](#) (page 2-6)

When you encrypt data as it travels through the network, you should follow guidelines to secure the network connections for Oracle Database.

[Protecting Data on the Network by Using Network Encryption](#) (page 2-6)

In addition to protecting information by encrypting it at the database level, you must protect it as it travels across the network.

[Initialization Parameters Used for Network Security](#) (page 2-10)

Oracle Database provides initialization parameters to configure network security.

2.3.1 About Securing the Network

When you encrypt data as it travels through the network, you should follow guidelines to secure the network connections for Oracle Database.

You can configure the client connection to your Oracle Database installation by following the procedures in "Configuring the Network Environment" in *Oracle Database 2 Day DBA* and the *Oracle Database Installation Guide* for your platform.

2.3.2 Protecting Data on the Network by Using Network Encryption

In addition to protecting information by encrypting it at the database level, you must protect it as it travels across the network.

[About Network Encryption](#) (page 2-6)

Network encryption refers to encrypting data as it travels across the network between the client and server.

[Configuring Network Encryption](#) (page 2-7)

You can configure network encryption by using either Oracle Net Manager or by editing the `sqlnet.ora` file.

2.3.2.1 About Network Encryption

Network encryption refers to encrypting data as it travels across the network between the client and server.

The reason that you should encrypt data at the network level, and not just the database level, is because data can be exposed on the network level. For example, an intruder can use a network packet sniffer to capture information as it travels on the

network, and then spool it to a file for malicious use. Encrypting data on the network prevents this sort of activity.

To encrypt data on the network, you must have the following components:

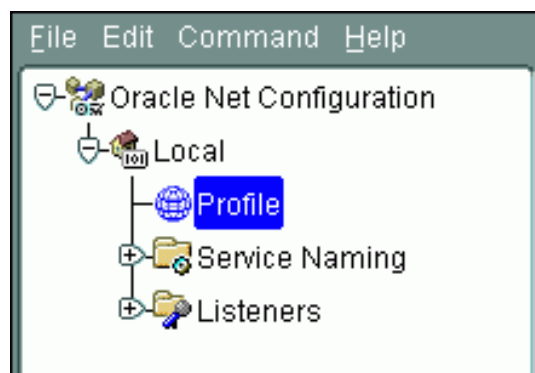
- **An encryption seed.** The encryption seed is a random string of up to 256 characters. It generates the cryptographic keys that encrypts data as it travels across the network.
- **An encryption algorithm.** You can specify any of the supported algorithm types: AES, RC4, DES, or 3DES.
- **Whether the settings apply to a client or server.** You must configure the server and each client to which it connects.
- **How the client or server should process the encrypted data.** The settings you select (you have four options) must complement both server and client.
- **A mechanism for configuring the encryption.** You can use Oracle Net Manager to configure the encryption. Alternatively, you can edit the `sqlnet.ora` configuration file. Both Oracle Net Manager and the `sqlnet.ora` file are available in a default Oracle Database installation.

2.3.2.2 Configuring Network Encryption

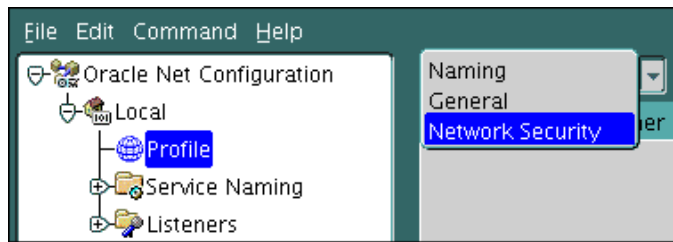
You can configure network encryption by using either Oracle Net Manager or by editing the `sqlnet.ora` file.

To configure network encryption:

1. On the server computer, start Oracle Net Manager.
 - **UNIX:** From `$ORACLE_HOME/bin`, enter the following at the command line:
`netmgr`
 - **Windows:** From the **Start** menu, click **All Programs**. Then, click **Oracle - HOME_NAME, Configuration and Migration Tools**, and then **Net Manager**
2. From the Oracle Net Configuration navigation tree, expand **Local**, and then select **Profile**.

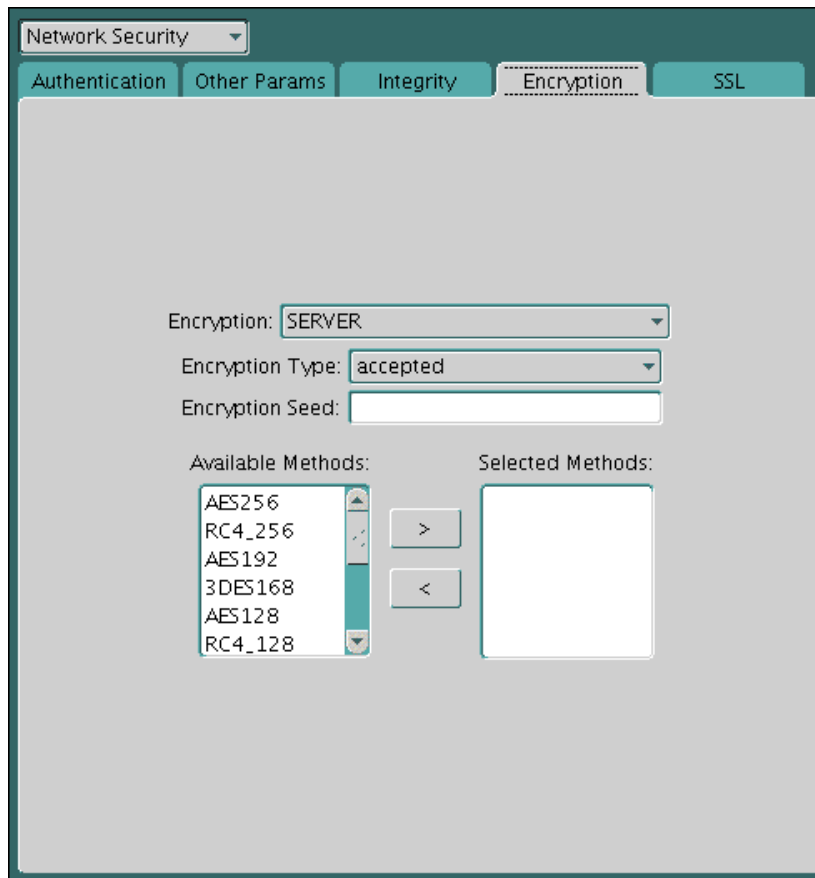


3. From the list, select **Network Security**.



4. Under Network Security, select the **Encryption** tab.

The Encryption settings pane appears.



5. Enter the following settings:
 - **Encryption:** From the list, select **SERVER** to configure the network encryption for the server. (For the client computer, you select **CLIENT**.)
 - **Encryption Type:** Select from the following values to specify the actions of the server (or client) when negotiating encryption and integrity:
 - **accepted:** Service will be active if the other side of the connection specifies either required or requested, and there is a compatible algorithm available on the receiving database; it will otherwise be inactive.
 - **rejected:** Service must not be active, and the connection will fail if the other side requires any of the methods in this list.

- **requested:** Service will be active if the other side of the connection specifies either accepted, required, or requested, and there is a compatible algorithm available on the other side. Otherwise, the service is inactive.
 - **required:** Service must be active, and the connection will fail if the other side specifies rejected, or if there is no compatible algorithm on the other side.
 - **Encryption Seed:** Enter a random string of up to 256 characters. Oracle Database uses the encryption seed to generate cryptographic keys. This is required when either encryption or integrity is enabled.

If you choose to use special characters such as a comma [,] or a right parenthesis [)] as a part of the **Encryption Seed** parameter, enclose the value within single quotation marks.
 - **Available Methods:** Select one or more of the following algorithms, and use the move button (>) to move them to the Selected Methods list. The order in which they appear in the Selected Methods list determines the preferred order for negotiation. That is, the first algorithm listed is selected first, and so on.
 - **AES256:** Advanced Encryption Standard (AES). AES was approved by the National Institute of Standards and Technology (NIST) to replace Data Encryption Standard (DES). AES256 enables you to encrypt a block size of 256 bits.
 - **RC4_256:** Rivest Cipher 4 (RC4), which is the most commonly used stream cipher that protects protocols such as Secure Sockets Layer (SSL). RC4_256 enables you to encrypt up to 256 bits of data.
 - **AES192:** Enables you to use AES to encrypt a block size of 192 bits.
 - **3DES168:** Triple Data Encryption Standard (TDES) with a three-key option. 3DES168 enables you to encrypt up to 168 bits of data.
 - **AES128:** Enables you to use AES to encrypt a block size of 128 bits.
 - **RC4_128:** Enables you to use RC4 to encrypt up to 128 bits of data.
 - **3DES112:** Enables you to use Triple DES with a two-key (112 bit) option.
 - **DES:** Data Encryption Standard (DES) 56-bit key. Note that National Institute of Standards and Technology (NIST) no longer recommends DES.
 - **RC4_40:** Enables you to use RC4 to encrypt up to 40 bits of data. (Not recommended.)
 - **DES40:** Enables you to use DES to encrypt up to 40 bits of data. (Not recommended.)
6. From the **File** menu, select **Save Network Configuration**, and then select **Exit** to exit Oracle Net Manager.
 7. Repeat these steps for each client computer that connects to the server.

See Also:

- *Oracle Database Net Services Reference* for information about editing the `sqlnet.ora` file parameters to configure network encryption

2.3.3 Initialization Parameters Used for Network Security

Oracle Database provides initialization parameters to configure network security.

[Table 2-3](#) (page 2-10) lists initialization parameters that you can set to better secure user accounts.

Table 2-3 Initialization Parameters Used for Network Security

| Initialization Parameter | Default | Description |
|-----------------------------------|--------------------|--|
| OS_AUTHENT_PREFIX | OP\$ | Specifies a prefix that Oracle Database uses to identify users attempting to connect to the database. Oracle Database concatenates the value of this parameter to the beginning of the user operating system account name and password. When a user attempts a connection request, Oracle Database compares the prefixed username with user names in the database. |
| REMOTE_LISTENER | No default setting | Specifies a network name that resolves to an address or address list of Oracle Net remote listeners (that is, listeners that are not running on the same computer as this instance). The address or address list is specified in the <code>tnsnames.ora</code> file or other address repository as configured for your system. |
| REMOTE_OS_AUTHENT | FALSE | Specifies whether remote clients will be authenticated with the value of the <code>OS_AUTHENT_PREFIX</code> parameter. |
| REMOTE_OS_ROLES | FALSE | Specifies whether operating system roles are allowed for remote clients. The default value, <code>FALSE</code> , causes Oracle Database to identify and manage roles for remote clients. |
| SEC_PROTOCOL_ERROR_TRACE_ACTION | TRACE | Specifies the action that the database should take when bad packets are received from a possibly malicious client. |
| SEC_PROTOCOL_ERROR_FURTHER_ACTION | DROP, 3 | Specifies the action that the database should take when bad packets are received from a possibly malicious client. |

To modify an initialization parameter, see [Modifying the Value of an Initialization Parameter](#) (page 2-5). For detailed information about initialization parameters, see *Oracle Database Reference*.

2.4 Securing User Accounts

You can secure user accounts by creating secure passwords, changing default passwords, and using special parameters to further secure user accounts.

[About Securing Oracle Database User Accounts](#) (page 2-11)

You can use many methods to secure both common and local database user accounts.

[Predefined User Accounts Provided by Oracle Database](#) (page 2-11)

The Oracle Database installation process creates predefined administrative, non-administrative, and sample schema user accounts in the database.

[Expiring and Locking Database Accounts](#) (page 2-17)

When you expire the password of a user, that password no longer exists.

[Requirements for Creating Passwords](#) (page 2-17)

Oracle provides password-creation requirements that help you create more secure passwords.

[Finding and Changing Default Passwords](#) (page 2-18)

You can find and change default passwords that may have come from earlier releases of Oracle Database.

[Parameters Used to Secure User Accounts](#) (page 2-20)

Oracle Database provides parameters to secure user accounts, such as setting the maximum failed login attempts.

2.4.1 About Securing Oracle Database User Accounts

You can use many methods to secure both common and local database user accounts.

For example, Oracle Database has a set of built-in protections for passwords. You can safeguard default database accounts and passwords, and use various ways to manage database accounts.

Oracle Database 2 Day DBA describes the fundamentals of creating and administering user accounts, including how to manage user roles, what the administrative accounts are, and how to use profiles to establish a password policy.

After you create user accounts, you can use the procedures in this section to further secure these accounts by following these methods:

- **Safeguarding predefined database accounts.** When you install Oracle Database, it creates a set of predefined accounts. You should secure these accounts as soon as possible by changing their passwords. You can use the same method to change all passwords, whether they are with regular user accounts, administrative accounts, or predefined accounts. This guide also provides guidelines on how to create the most secure passwords.
- **Managing database accounts.** You can expire and lock database accounts.
- **Managing passwords.** You can manage and protect passwords by setting initialization parameters. *Oracle Database Reference* describes the initialization parameters in detail.

2.4.2 Predefined User Accounts Provided by Oracle Database

The Oracle Database installation process creates predefined administrative, non-administrative, and sample schema user accounts in the database.

[Predefined Administrative Accounts](#) (page 2-12)

A default Oracle Database installation provides predefined administrative accounts to manage commonly used features, such as auditing.

[Predefined Non-Administrative User Accounts](#) (page 2-15)

A default Oracle Database installation provides non-administrative user accounts to manage features such as Oracle Spatial.

[Predefined Sample Schema User Accounts](#) (page 2-16)

Oracle Database creates a set of sample user accounts if you install the sample schemas.

2.4.2.1 Predefined Administrative Accounts

A default Oracle Database installation provides predefined administrative accounts to manage commonly used features, such as auditing.

These are accounts that have special privileges required to administer areas of the database, such as the `CREATE ANY TABLE` or `ALTER SESSION` privilege, or `EXECUTE` privileges on packages owned by the `SYS` schema. The default tablespace for administrative accounts is either `SYSTEM` or `SYSAUX`. In a multitenant environment, the predefined administrative accounts reside in the root database.

To protect these accounts from unauthorized access, the installation process expires and locks most of these accounts, except where noted in [Table 2-4](#) (page 2-12). As the database administrator, you are responsible for unlocking and resetting these accounts, as described in [Expiring and Locking Database Accounts](#) (page 2-17).

[Table 2-4](#) (page 2-12) lists the predefined administrative user accounts, which Oracle Database automatically creates when you run standard scripts (such as the various `cat*.sql` scripts). You can find user accounts that are created and maintained by Oracle by querying the `USERNAME` and `ORACLE_MAINTAINED` columns of the `ALL_USERS` data dictionary view. If the output for `ORACLE_MAINTAINED` is `Y`, then you must not modify the user account except by running the script that was used to create it.

Table 2-4 Predefined Oracle Database Administrative User Accounts

| User Account | Description | Status After Installation |
|--------------|---|---------------------------|
| ANONYMOUS | An account that allows HTTP access to Oracle XML DB. It is used in place of the <code>APEX_PUBLIC_USER</code> account when the Embedded PL/SQL Gateway (EPG) is installed in the database. EPG is a Web server that can be used with Oracle Database. It provides the necessary infrastructure to create dynamic applications. | Expired and locked |
| AUDSYS | The internal account used by the unified audit feature to store unified audit trail records. See <i>Oracle Database Security Guide</i> . | Expired and locked |

Table 2-4 (Cont.) Predefined Oracle Database Administrative User Accounts

| User Account | Description | Status After Installation |
|--------------------|---|--|
| CTXSYS | The account used to administer Oracle Text. Oracle Text enables you to build text query applications and document classification applications. It provides indexing, word and theme searching, and viewing capabilities for text. <i>See Oracle Text Application Developer's Guide.</i> | Expired and locked |
| DBSNMP | The account used by the Management Agent component of Oracle Enterprise Manager to monitor and manage the database. <i>See Enterprise Manager Cloud Control Administrator's Guide.</i> | Open Password is created at installation or database creation time. |
| LBACSYS | The account used to administer Oracle Label Security (OLS). It is created only when you install the Label Security custom option. <i>See Enforcing Row-Level Security with Oracle Label Security (page 8-1), and Oracle Label Security Administrator's Guide.</i> | Expired and locked |
| MDSYS | The Oracle Spatial and Oracle Multimedia Locator administrator account. <i>See Oracle Spatial and Graph Developer's Guide.</i> | Expired and locked |
| OLAPSYS | The account that owns the OLAP Catalog (CWM Lite). This account has been deprecated, but is retained for backward compatibility. | Expired and locked |
| ORDDATA | This account contains the Oracle Multimedia DICOM data model. <i>See Oracle Multimedia DICOM Developer's Guide</i> for more information. | Expired and locked |
| ORDPLUGINS | The Oracle Multimedia user. Plug-ins supplied by Oracle and third-party, format plug-ins are installed in this schema. Oracle Multimedia enables Oracle Database to store, manage, and retrieve images, audio, video, DICOM format medical images and other objects, or other heterogeneous media data integrated with other enterprise information. <i>See Oracle Multimedia User's Guide.</i> | Expired and locked |
| ORDSYS | The Oracle Multimedia administrator account. <i>See Oracle Multimedia User's Guide.</i> | Expired and locked |
| SI_INFORMTN_SCHEMA | The account that stores the information views for the SQL/MM Still Image Standard. <i>See Oracle Multimedia User's Guide.</i> Note: The SI_INFORMTN_SCHEMA account is deprecated in Oracle Database 12c release 2 (12.2). | Expired and locked |

Table 2-4 (Cont.) Predefined Oracle Database Administrative User Accounts

| User Account | Description | Status After Installation |
|--------------|--|--|
| SYS | An account used to perform database administration tasks. See <i>Oracle Database 2 Day DBA</i> . | Open Password is created at installation or database creation time. |
| SYSBACKUP | The account used to perform Oracle Recovery Manager recovery and backup operations. See <i>Oracle Database Backup and Recovery User's Guide</i> . | Expired and locked |
| SYSDG | The account used to perform Oracle Data Guard operations. See <i>Oracle Data Guard Concepts and Administration</i> . | Expired and locked |
| SYSKM | The account used to manage Transparent Data Encryption. See <i>Oracle Database Advanced Security Guide</i> . | Expired and locked |
| SYSRAC | The account used to manage Oracle Real Application Clusters. See <i>Oracle Real Application Clusters Administration and Deployment Guide</i> . | Expired and locked |
| SYSTEM | A default generic database administrator account for Oracle databases. For production systems, Oracle recommends creating individual database administrator accounts and not using the generic SYSTEM account for database administration operations. See <i>Oracle Database 2 Day DBA</i> . | Open Password is created at installation or database creation time. |
| WMSYS | The account used to store the metadata information for Oracle Workspace Manager. See <i>Oracle Database Workspace Manager Developer's Guide</i> . | Expired and locked |
| XDB | The account used for storing Oracle XML DB data and metadata. For better security, never unlock the XDB user account. Oracle XML DB provides high-performance XML storage and retrieval for Oracle Database data. See <i>Oracle XML DB Developer's Guide</i> . | Expired and locked |

Note:

If you create an Oracle Automatic Storage Management (Oracle ASM) instance, then the ASMSNMP account is created. Oracle Enterprise Manager uses this account to monitor ASM instances to retrieve data from ASM-related data dictionary views. The ASMSNMP account status is set to OPEN upon creation, and it is granted the SYSDBA administrative privilege. For more information, see *Oracle Automatic Storage Management Administrator's Guide*.

2.4.2.2 Predefined Non-Administrative User Accounts

A default Oracle Database installation provides non-administrative user accounts to manage features such as Oracle Spatial.

[Table 2-5](#) (page 2-15) lists the predefined non-administrative user accounts that Oracle Database automatically creates when you run standard scripts (such as the various `cat*.sql` scripts). You can find user accounts that are created and maintained by Oracle by querying the `USERNAME` and `ORACLE_MAINTAINED` columns of the `ALL_USERS` data dictionary view. If the output for `ORACLE_MAINTAINED` is `Y`, then you must not modify the user account except by running the script that was used to create it.

Non-administrative user accounts only have the minimum privileges needed to perform their jobs. Their default tablespace is `USERS`. In a multitenant environment, the predefined non-administrative accounts reside in the root database.

To protect these accounts from unauthorized access, the installation process locks and expires these accounts immediately after installation, except where noted in [Table 2-5](#) (page 2-15). As the database administrator, you are responsible for unlocking and resetting these accounts, as described in [Expiring and Locking Database Accounts](#) (page 2-17).

Table 2-5 Predefined Oracle Database Non-Administrative User Accounts

| User Account | Description | Status After Installation |
|-----------------------|--|---------------------------|
| DIP | The Oracle Directory Integration and Provisioning (DIP) account that is installed with Oracle Label Security. This profile is created automatically as part of the installation process for Oracle Internet Directory-enabled Oracle Label Security. <i>See Oracle Label Security Administrator's Guide.</i> | Expired and locked |
| MDDATA | The schema used by Oracle Spatial for storing Geocoder and router data. Oracle Spatial provides a SQL schema and functions that enable you to store, retrieve, update, and query collections of spatial features in an Oracle database. <i>See Oracle Spatial and Graph Developer's Guide.</i> | Expired and locked |
| ORACLE_OCM | The account used with Oracle Configuration Manager. This feature enables you to associate the configuration information for the current Oracle Database instance with My Oracle Support. Then when you log a service request, it is associated with the database instance configuration information. <i>See Oracle Database Installation Guide for your platform.</i> | Expired and locked |
| SPATIAL_CSW_ADMIN_USR | The Catalog Services for the Web (CSW) account. It is used by Oracle Spatial CSW Cache Manager to load all record-type metadata and record instances from the database into the main memory for the record types that are cached. <i>See Oracle Spatial and Graph Developer's Guide.</i> | Expired and locked |

Table 2-5 (Cont.) Predefined Oracle Database Non-Administrative User Accounts

| User Account | Description | Status After Installation |
|-----------------------|--|---------------------------|
| SPATIAL_WFS_ADMIN_USR | The Web Feature Service (WFS) account. It is used by Oracle Spatial WFS Cache Manager to load all feature type metadata and feature instances from the database into main memory for the feature types that are cached. See <i>Oracle Spatial and Graph Developer's Guide</i> . | Expired and locked |
| XS\$NULL | An internal account that represents the absence of database user in a session and the actual session user is an application user supported by Oracle Real Application Security. XS\$NULL has no privileges and does not own any database object. No one can authenticate as XS\$NULL, nor can authentication credentials ever be assigned to XS\$NULL. | Expired and locked |

2.4.2.3 Predefined Sample Schema User Accounts

Oracle Database creates a set of sample user accounts if you install the sample schemas.

The sample schema user accounts are all non-administrative accounts, and their tablespace is USERS.

To protect these accounts from unauthorized access, the installation process locks and expires these accounts immediately after installation. As the database administrator, you are responsible for unlocking and resetting these accounts, as described in [Expiring and Locking Database Accounts](#) (page 2-17). For more information about the sample schema accounts, see *Oracle Database Sample Schemas*.

[Table 2-6](#) (page 2-16) lists the sample schema user accounts, which represent different divisions of a fictional company that manufactures various products. You can find the status of these accounts by querying the DBA_USERS data dictionary view. Because the ORACLE_MAINTAINED column output for these accounts is N, you can modify these accounts without re-running the scripts that were used to create them.

Table 2-6 Default Sample Schema User Accounts

| User Account | Description | Status After Installation |
|--------------|--|---------------------------|
| HR | The account used to manage the HR (Human Resources) schema. This schema stores information about the employees and the facilities of the company. | Expired and locked |
| OE | The account used to manage the OE (Order Entry) schema. This schema stores product inventories and sales of the company's products through various channels. | Expired and locked |
| PM | The account used to manage the PM (Product Media) schema. This schema contains descriptions and detailed information about each product sold by the company. | Expired and locked |
| IX | The account used to manage the IX (Information Exchange) schema. This schema manages shipping through business-to-business (B2B) applications. | Expired and locked |

Table 2-6 (Cont.) Default Sample Schema User Accounts

| User Account | Description | Status After Installation |
|--------------|--|---------------------------|
| SH | The account used to manage the SH (Sales) schema. This schema stores business statistics to facilitate business decisions. | Expired and locked |

In addition to the sample schema accounts, Oracle Database provides another sample schema account, SCOTT. The SCOTT schema contains the tables EMP, DEPT, SALGRADE, and BONUS. The SCOTT account is used in examples throughout the Oracle Database documentation set. When you install Oracle Database, the SCOTT account is locked and expired.

2.4.3 Expiring and Locking Database Accounts

When you expire the password of a user, that password no longer exists.

Locking an account preserves the user password and other account information, but makes the account unavailable to anyone who tries to log in to the database using that account. Unlocking it makes the account available again.

Oracle Database 2 Day DBA explains how you can use Enterprise Manager to unlock database accounts. You also can use Enterprise Manager to expire or lock database accounts.

To expire and lock a database account:

1. Access the Database home page.
See *Oracle Database 2 Day DBA* for more information.
2. From the **Administration** menu, select **Security**, then **Users**.
If the Database Login page appears, then log in as an administrative user, such as SYSTEM.
The Users page lists the user accounts created for the current database instance. The Account Status column indicates whether an account is expired, locked, or open.
3. In the Select column, select the account you want to expire, and then click **Edit**.
4. In the Edit User page, do one of the following:
 - To expire a password, click **Expire Password now**.
To unexpire the password, enter a new password in the **Enter Password** and **Confirm Password** fields. See [Requirements for Creating Passwords](#) (page 2-17) for password requirements.
 - To lock the account, select **Locked**.
5. Click **Apply**.

2.4.4 Requirements for Creating Passwords

Oracle provides password-creation requirements that help you create more secure passwords.

When you create a user account, Oracle Database assigns a default password policy for that user. The password policy defines rules for how the password should be created, such as a minimum number of characters, when it expires, and so on. You can strengthen passwords by using password policies.

For greater security, follow these guidelines when you create passwords:

- Make the password between 12 and 30 characters and numbers.
- Use mixed case letters and special characters in the password. (See *Oracle Database Security Guide* for more information.)
- Use the database character set for the password characters, which can include the underscore (_), dollar (\$), and number sign (#) characters.
- Do not use an actual word for the entire password.

Oracle Database Security Guide describes more ways that you can further secure passwords.

2.4.5 Finding and Changing Default Passwords

You can find and change default passwords that may have come from earlier releases of Oracle Database.

[About Finding and Changing Default Passwords](#) (page 2-18)

After installation, the default database user accounts, including administrative accounts, are created without default passwords.

[Finding and Changing Default Passwords from SQL*Plus](#) (page 2-19)

You can use SQL*Plus to find and change default passwords.

[Finding and Changing Default Passwords from Enterprise Manager](#) (page 2-19)

You can use Enterprise Manager to change a user account passwords if you have administrative privileges.

2.4.5.1 About Finding and Changing Default Passwords

After installation, the default database user accounts, including administrative accounts, are created without default passwords.

Except for the administrative accounts whose passwords you create during installation (such as user `SYS`), the default user accounts arrive locked with their passwords expired. If you have upgraded from a previous release of Oracle Database, you may have database accounts that still have default passwords. These are default accounts that are created when you create a database, such as the `HR`, `OE`, and `SCOTT` accounts.

Security is most easily compromised when a default database user account still has a default password *after installation*. This is particularly true for the user account `SCOTT`, which is a well known account that may be vulnerable to intruders. Find accounts that use default passwords and then change their passwords.

See Also:

- *Oracle Database Security Guide* for additional methods of configuring password protection
- [Predefined User Accounts Provided by Oracle Database](#) (page 2-11)

2.4.5.2 Finding and Changing Default Passwords from SQL*Plus

You can use SQL*Plus to find and change default passwords.

To find and change default passwords:

1. Log into the database instance with administrative privileges.

```
sqlplus system
Enter password: password
```

2. Select from the DBA_USERS_WITH_DEFPWD data dictionary view.

```
SELECT * FROM DBA_USERS_WITH_DEFPWD;
```

The DBA_USERS_WITH_DEFPWD lists the accounts that still have user default passwords. For example:

```
USERNAME
-----
SCOTT
```

3. Change the password for the accounts the DBA_USERS_WITH_DEFPWD data dictionary view lists.

For example, to change the password for user SCOTT, enter the following:

```
PASSWORD SCOTT
Changing password for SCOTT
New password: password
Retype new password: password
Password changed
```

Replace *password* with a password that is secure, according to the guidelines listed in [Requirements for Creating Passwords](#) (page 2-17). For greater security, do not reuse the same password that was used in previous releases of Oracle Database.

Alternatively, you can use the ALTER USER SQL statement to change the password:

```
ALTER USER SCOTT IDENTIFIED BY password;
```

2.4.5.3 Finding and Changing Default Passwords from Enterprise Manager

You can use Enterprise Manager to change a user account passwords if you have administrative privileges.

Individual users can also use Enterprise Manager to change their own passwords.

To use Enterprise Manager to change the password of a database account:

1. Access the Database home page.

See *Oracle Database 2 Day DBA* for more information.

2. From the **Administration** menu, select **Security**, then **Users**.

If the Database Login page appears, then log in as an administrative user, such as SYS. User SYS must log in with the **SYSDBA** role selected.

The Users page lists the user accounts created for the current database instance. The Account Status column indicates whether an account is expired, locked, or open.

3. In the Select column, select the account you want to change, and then click **Edit**.
4. In the Edit User page, enter a new password in the **Enter Password** and **Confirm Password** fields.
5. Click **Apply**.

2.4.6 Parameters Used to Secure User Accounts

Oracle Database provides parameters to secure user accounts, such as setting the maximum failed login attempts.

[Table 2-7](#) (page 2-20) lists initialization and profile parameters that you can set to better secure user accounts.

Table 2-7 Initialization and Profile Parameters Used for User Account Security

| Parameter | Default | Description |
|-------------------------------|--------------------|---|
| SEC_CASE_SENSITIVE_LOGON | TRUE | Controls case sensitivity in passwords. TRUE enables case sensitivity; FALSE disables it. |
| SEC_MAX_FAILED_LOGIN_ATTEMPTS | 3 | Sets the maximum number of times user authentication is allowed before an Oracle Call Interface (OCI) connection is closed. |
| FAILED_LOGIN_ATTEMPTS | 10 | Sets the maximum times a user login is allowed to fail before locking the account. Note: You also can set limits on the number of times an unauthorized user (possibly an intruder) attempts to log in to Oracle Call Interface applications by using the SEC_MAX_FAILED_LOGIN_ATTEMPTS initialization parameter. |
| INACTIVE_ACCOUNT_TIME | 35 | Locks the account of a database user who has not logged in to the database instance in a specified number of days. |
| PASSWORD_GRACE_TIME | No default setting | Sets the number of days that a user has to change his or her password before it expires. |
| PASSWORD_LIFE_TIME | No default setting | Sets the number of days the user can use his or her current password. |
| PASSWORD_LOCK_TIME | No default setting | Sets the number of days an account will be locked after the specified number of consecutive failed login attempts. |

Table 2-7 (Cont.) Initialization and Profile Parameters Used for User Account Security

| Parameter | Default | Description |
|---------------------|--------------------|--|
| PASSWORD_REUSE_MAX | No default setting | Specifies the number of password changes required before the current password can be reused. |
| PASSWORD_REUSE_TIME | No default setting | Specifies the number of days before which a password cannot be reused. |

Note:

You can use most of these parameters to create a user profile. See *Oracle Database Security Guide* for more information about user profile settings.

To modify an initialization parameter, see [Modifying the Value of an Initialization Parameter](#) (page 2-5). For detailed information about initialization parameters, see *Oracle Database Reference*.

Managing User Privileges

A user privilege enables users to perform certain actions, such as creating other user accounts or modifying database tables.

[About Privilege Management](#) (page 3-1)

You can control user privileges in a variety of ways, such as granting and revoking privileges or creating roles.

[When to Grant Privileges to Users](#) (page 3-1)

You should only grant users the minimum privileges necessary to perform their jobs.

[When to Grant Roles to Users](#) (page 3-2)

A role is a named group of related privileges that you grant, as a group, to users or other roles.

[Controlling Access to Applications with Secure Application Roles](#) (page 3-2)

A secure application role is a role that can be enabled only by an authorized PL/SQL package.

[Initialization Parameters Used for Privilege Security](#) (page 3-13)

Oracle Database provides initialization parameters to configure privilege security, such as the restriction of SYSTEM privileges.

3.1 About Privilege Management

You can control user privileges in a variety of ways, such as granting and revoking privileges or creating roles.

- **Granting and revoking individual privileges.** You can grant individual privileges, for example, the privilege to perform the UPDATE SQL statement, to individual users or to groups of users.
- **Creating a role and assigning privileges to it.** A role is a named group of related privileges that you grant, as a group, to users or other roles.
- **Creating a secure application role.** A secure application role enables you to define conditions that control when a database role can be enabled. For example, a secure application role can check the IP address associated with a user session before allowing the session to enable a database role.

See Also: *Oracle Database Security Guide* for detailed information about privilege management

3.2 When to Grant Privileges to Users

You should only grant users the minimum privileges necessary to perform their jobs.

For an introduction to managing user privileges and roles, see *Oracle Database 2 Day DBA*. *Oracle Database 2 Day DBA* also provides an example of how to grant a privilege.

In other words, the *principle of least privilege* is that users be given only those privileges that are actually required to efficiently perform their jobs. To implement this principle, restrict the following as much as possible:

- The number of system and object privileges granted to database users
- The number of people who are allowed to make SYS-privileged connections to the database

For example, generally the `CREATE ANY TABLE` privilege is not granted to a user who does not have database administrator privileges.

You can find excessive system and object privilege grants, even with large numbers of user accounts in complex Oracle Database installations, by creating a privilege analysis policy. A privilege analysis policy finds privilege usage according to a specified condition and then stores the results in data dictionary views. *Oracle Database Vault Administrator's Guide* describes how to create a privilege analysis policy.

3.3 When to Grant Roles to Users

A role is a named group of related privileges that you grant, as a group, to users or other roles.

To learn the fundamentals of managing roles, see *Oracle Database 2 Day DBA*. In addition, see *Oracle Database 2 Day DBA* for an example of creating a role.

Roles are useful for quickly and easily granting permissions to users. Although you can use Oracle Database-defined roles, you have more control and continuity if you create your own roles that contain only the privileges pertaining to your requirements. Oracle may change or remove the privileges in an Oracle Database-defined role, as it has with the `CONNECT` role, which now has only the `CREATE SESSION` privilege. Formerly, this role had eight other privileges.

Ensure that the roles you define contain only the privileges required for the responsibility of a particular job. If your application users do not need all the privileges encompassed by an existing role, then apply a different set of roles that supply just the correct privileges. Alternatively, create and assign a more restrictive role.

Do not grant powerful privileges, such as the `CREATE DATABASE LINK` privilege, to regular users such as user `SCOTT`. (Particularly do not grant *any* powerful privileges to `SCOTT`, because this is a well known default user account that may be vulnerable to intruders.) Instead, grant the privilege to a database role, and then grant this role to the users who must use the privilege. And remember to only grant the minimum privileges the user needs.

3.4 Controlling Access to Applications with Secure Application Roles

A secure application role is a role that can be enabled only by an authorized PL/SQL package.

[About Secure Application Roles](#) (page 3-3)

A secure application role requires a PL/SQL package and a way to execute this package when the user logs in.

[Tutorial: Creating a Secure Application Role](#) (page 3-4)

This tutorial shows how two employees, Matthew Weiss and Winston Taylor, try to gain information from the OE.ORDERS table.

3.4.1 About Secure Application Roles

A secure application role requires a PL/SQL package and a way to execute this package when the user logs in.

This package defines one or more security policies that control access to the application. Both the role and the package are typically created in the schema of the person who creates them, which is typically a security administrator. A security administrator is a database administrator who is responsible for maintaining the security of the database.

The advantage of using a secure application role is you can create additional layers of security for application access, in addition to the privileges that were granted to the role itself. Secure application roles strengthen security because passwords are not embedded in application source code or stored in a table. This way, the decisions the database makes are based on the implementation of your security policies. Because these definitions are stored in one place, the database, rather than in your applications, you modify this policy once instead of modifying the policy in each application. No matter how many users connect to the database, the result is always the same, because the policy is bound to the role.

A secure application role has the following components:

- **The secure application role itself.** You create the role using the `CREATE ROLE` statement with the `IDENTIFIED USING` clause to associate it with the PL/SQL package. Then, you grant the role the privileges you typically grant a role.
- **A PL/SQL package, procedure, or function associated with the secure application role.** The PL/SQL package sets a condition that either grants the role or denies the role to the person trying to log in to the database. You must create the PL/SQL package, procedure, or function using invoker's rights, not definer's rights. An invoker's right procedure executes with the privileges of the current user, that is, the user who invokes the procedure. This user must be granted the `EXECUTE` privilege for the underlying objects that the PL/SQL package accesses. Invoker's rights procedures are not bound to a particular schema. They can be run by a variety of users and enable multiple users to manage their own data by using centralized application logic. To create the invoker's rights package, use the `AUTHID CURRENT_USER` clause in the declaration section of the procedure code.

The PL/SQL package also must contain a `SET ROLE` statement or `DBMS_SESSION.SET_ROLE` call to enable (or disable) the role for the user.

After you create the PL/SQL package, you must grant the appropriate users the `EXECUTE` privilege on the package.

- **A way to execute the PL/SQL package when the user logs on.** To execute the PL/SQL package, you must call it directly from the application before the user tries to use the privileges the role grants. You cannot use a logon trigger to execute the PL/SQL package automatically when the user logs on.

When a user logs in to the application, the policies in the package perform the checks as needed. If the user passes the checks, then the role is granted, which enables access to the application. If the user fails the checks, then the user is prevented from accessing the application.

3.4.2 Tutorial: Creating a Secure Application Role

This tutorial shows how two employees, Matthew Weiss and Winston Taylor, try to gain information from the `OE.ORDERS` table.

Access rights to the `OE.ORDERS` table are defined in the `emp_role` secure application role. Matthew is Winston's manager, so Matthew, as opposed to Winston, will be able to access the information in `OE.ORDERS`.

Step 1: Create User Accounts for This Tutorial (page 3-4)

Your first step is to create user accounts for Matthew and Winston.

Step 2: Create a Security Administrator Account (page 3-6)

For greater security, you should apply separation of duty concepts when you assign responsibilities to the system administrators on your staff.

Step 3: Create a Lookup View (page 3-7)

A lookup view contains a view of information from a larger table. The lookup view only contains the information that you need.

Step 4: Create the PL/SQL Procedure to Set the Secure Application Role (page 3-7)

After you create the administrative account and the lookup view, you can create the secure application role procedure.

Step 5: Create the Secure Application Role (page 3-10)

After you create the PL/SQL procedure to set the secure application role, you can create the `emp_role` secure application role.

Step 6: Grant SELECT for the EMP_ROLE Role to the OE.ORDERS Table (page 3-10)

User OE, who owns the `OE.ORDERS` table, must grant the `SELECT` privilege for the `ORDERS` table to the `emp_role` role.

Step 7: Grant the EXECUTE Privilege for the Procedure to Matthew and Winston (page 3-11)

At this stage, Matthew and Winston can try to access the `OE.ORDERS` table, but they are denied access.

Step 8: Test the EMP_ROLE Secure Application Role (page 3-12)

To test the `emp_role` secure application role, you must log on as Matthew and Winston and trying to access the `OE.ORDERS` table.

Step 9: Optionally, Remove the Components for This Tutorial (page 3-13)

You can remove the components that you created for this tutorial if you no longer need them.

3.4.2.1 Step 1: Create User Accounts for This Tutorial

Your first step is to create user accounts for Matthew and Winston.

Matthew and Winston both are sample employees in the `HR.EMPLOYEES` table. This table provides columns for the manager ID and email address of the employees, among other information. You must create user accounts for these two employees so that they can later test the secure application role.

To create the user accounts:

1. In Enterprise Manager, access the Database home page.

See *Oracle Database 2 Day DBA* for more information.

2. Access your target database and then log in as user `SYSTEM`.
3. From the **Schema** menu, select **Users**.
4. In the Users page, click **Create**.
5. In the Create User page, enter the following information:
 - **Name:** `mweiss` (to create the user account for Matthew Weiss)
 - **Profile:** `DEFAULT`
 - **Authentication:** `Password`
 - **Enter Password and Confirm Password:** Enter a password that meets the requirements in [Requirements for Creating Passwords](#) (page 2-17).
 - **Default Tablespace:** `USERS`
 - **Temporary Tablespace:** `TEMP`
 - **Status:** `Unlocked`
6. Click **System Privileges**.
7. Click **Edit List**.
8. In the Modify System Privileges, from the Available System Privileges lists, select the `CREATE SESSION` privilege, and then click **Move** to move it to the Selected System Privileges list.
9. Click **OK**.

The Create User page appears, with `CREATE SESSION` listed as the system privilege for user `mweiss`.
10. Ensure that the Admin Option for `CREATE SESSION` is not selected, and then click **OK**.
11. In the Users page, select the selection button for user **MWEISS** from the list of users, and then from the **Actions** list, select **Create Like**. Then, click **Go**.
12. In the Create User page, enter the following information to create the user account for Winston, which will be almost identical to the user account for Matthew:
 - **Name:** `wtaylor`
 - **Enter Password and Confirm Password:** Enter a password that meets the requirements in [Requirements for Creating Passwords](#) (page 2-17).

You do not need to specify the default and temporary tablespaces, or the `CREATE SESSION` system privilege, for user `wtaylor` because they are already specified.
13. Click **OK**.

Now both Matthew Weiss and Winston Taylor have user accounts that have identical privileges.

3.4.2.2 Step 2: Create a Security Administrator Account

For greater security, you should apply separation of duty concepts when you assign responsibilities to the system administrators on your staff.

For the tutorials used in this guide, you will create and use a security administrator account called `sec_admin`.

To create the `sec_admin` security administrator account:

1. From the **Schema** menu, select **Users**.
If the Database Login page appears, then log in as an administrative user, such as `SYS`. User `SYS` must log in with the `SYSDBA` role selected.
2. In the Users page, click **Create**.
3. In the Create User page, enter the following information:
 - **Name:** `sec_admin`
 - **Profile:** `Default`
 - **Authentication:** `Password`
 - **Enter Password and Confirm Password:** Enter a password that meets the requirements in [Requirements for Creating Passwords](#) (page 2-17).
 - **Default Tablespace:** `USERS`
 - **Temporary Tablespace:** `TEMP`
 - **Status:** `UNLOCKED`
4. Click **System Privileges**.
5. Click **Edit List**.
The Modify System Privileges page appears.
6. In the Available System Privileges list, select the following privileges and then click **Move** to move each one to the Selected System Privileges list. (Hold down the Control key to select multiple privileges.)
 - `CREATE PROCEDURE`
 - `CREATE ROLE`
 - `CREATE SESSION`
 - `INHERIT ANY PRIVILEGES`
 - `SELECT ANY DICTIONARY`
7. Click **OK**.
The Create User page appears. Under Admin Option, do not select the boxes.
8. Click **OK**.
The Users page appears. User `sec_admin` is listed in the User Name list.

3.4.2.3 Step 3: Create a Lookup View

A lookup view contains a view of information from a larger table. The lookup view only contains the information that you need.

The `grant_emp_role` procedure, which you will create later in this tutorial, grants the `emp_role` only to managers who report to Steven King, whose employee ID is 100. This information is located in the `HR.EMPLOYEES` table. However, you should not use that table in this procedure, because it contains sensitive data such as salary information, and for it to be used, everyone will need access to it. In most real world cases, you create a lookup view that contains only the information that you need. (You could create a lookup table, but a view will reflect the most recent data.) For this tutorial, you create your own lookup view that only contains the employee names, employee IDs, and their manager IDs.

To create the `HR.HR_VERIFY` lookup view:

1. In SQL*Plus, connect as user HR.

```
CONNECT HR
Enter password: password
```

If you receive an error message saying that HR is locked, then you can unlock the account and reset its password by entering the following statements. For greater security, do not reuse the same password that was used in previous releases of Oracle Database. Enter any password that is secure, according to the password guidelines described in [Requirements for Creating Passwords](#) (page 2-17).

```
CONNECT SYSTEM
Enter password: password

PASSWORD HR
Changing password for HR
New password: password
Retype new password: password
Password changed.

ALTER USER HR ACCOUNT UNLOCK;

CONNECT HR
Enter password: password
```

2. Enter the following `CREATE VIEW` SQL statement to create the lookup view:

```
CREATE VIEW hr_verify AS
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, MANAGER_ID
FROM EMPLOYEES;
```

3. Grant the `EXECUTE` privilege for this view to `mweiss`, `wtaylor`, and `sec_admin` by entering the following SQL statements:

```
GRANT SELECT ON hr_verify TO mweiss;
GRANT SELECT ON hr_verify TO wtaylor;
GRANT SELECT ON hr_verify TO sec_admin;
```

3.4.2.4 Step 4: Create the PL/SQL Procedure to Set the Secure Application Role

After you create the administrative account and the lookup view, you can create the secure application role procedure.

In most cases, you create a package to hold the procedure, but because this is a simple tutorial that requires only one secure application role test (as defined in the procedure), you will create a procedure by itself. If you want to have a series of procedures to test for the role, then you would create them in a package. At this stage, the role does not exist, but it does not need to. You will create it in the next step.

A PL/SQL package defines a simple, clear interface to a set of related procedures and types that can be accessed by SQL statements. Packages also make code more reusable and easier to maintain. The advantage here for secure application roles is that you can create a group of security policies that used together present a solid security strategy designed to protect your applications. For users (or potential intruders) who fail the security policies, you can add auditing checks to the package to record the failure.

To create the secure application role procedure:

1. In Enterprise Manager, click **Log Out** to log out of the database.
2. In the Confirmation dialog box, select **Logout of (Database Instance)** and then select the **Display login page after logout** check box. Then click **Logout**.
3. Log in as user `sec_admin` using the `NORMAL` role.
4. From the **Schema** menu, select **Programs**, then **Procedures**.
5. In the Procedures page, click **Create**.
6. In the Create Procedure page, enter the following information:
 - **Name:** `GRANT_EMP_ROLE`
 - **Schema:** `SEC_ADMIN`
 - **Source:** Delete the empty procedure code that has been provided and then enter following text to create the secure application role procedure.

```
AUTHID CURRENT_USER
AS
v_user varchar2(50);
v_manager_id number :=1;
BEGIN
  v_user := lower((sys_context ('userenv','session_user')));
  SELECT manager_id
  INTO v_manager_id FROM hr.hr_verify WHERE lower(email)=v_user;
  IF v_manager_id = 100
  THEN
    EXECUTE IMMEDIATE 'SET ROLE emp_role';
  ELSE NULL;
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN v_manager_id:=0;
  DBMS_OUTPUT.PUT_LINE(v_manager_id);
END;
```

In this specification:

- `AUTHID CURRENT USER`: Appends the `AUTHID CURRENT_USER` clause to the `CREATE PROCEDURE` statement. The `AUTHID CURRENT_USER` clause enables the invoking user to run the procedure using his or her privileges.

You *must* create the procedure to use invoker's rights for the procedure to work. Invoker's rights allow the user to have EXECUTE privileges on all objects that the procedure accesses.

Roles that are enabled inside an invoker's right procedure remain in effect even after the procedure exits, but after the user exits the session, he or she no longer has the privileges associated with the secure application role. In this case, you can have a dedicated procedure that enables the role for the rest of the session.

Because users cannot change the security domain inside definer's rights procedures, secure application roles can only be enabled inside invoker's rights procedures.

See [About Secure Application Roles](#) (page 3-3) for information about the importance of creating the procedure using invoker's rights.

- `v_user varchar2(50)`: Declares the `v_user` variable, which will store the user session information.
- `v_manager_id number :=1`: Declares the `v_manager_id` variable, which will store the manager's ID of the `v_user` user.
- `v_user := lower...`: Retrieves the user session information for the user logging on, in this case, Matthew or Winston. To retrieve user session information, use the `SYS_CONTEXT` SQL function with the `USERENV` namespace attributes (`'userenv', session_attribute`), and writes this information to the `v_user` variable.

The information returned by this function indicates the way in which the user was authenticated, the IP address of the client, and whether the user connected through a proxy. See *Oracle Database SQL Language Reference* for more information about `SYS_CONTEXT`.

- `SELECT manager_id ... INTO...`: Get the manager's ID of the current user. The `SELECT` statement copies the manager ID into the `v_manager_id` variable, and then checking the `HR.HR_VERIFY` view for the manager ID of the current user. This example uses the employees' email addresses because they are the same as their user names.
- `IF ... THEN ... END IF`: Use an `IF` condition to test whether the user should be granted the `grant_emp_role` role. In this case, the test condition is whether the user reports to Matthew's manager, Steven King, whose employee number is 100. If the user reports to King, as Matthew does, then the secure application role is granted to the user. Otherwise, the role is not granted.

The result is that the secure application role will grant Matthew Weiss the role because he is a direct report of Steven King, but will deny the role to Winston, because he is not a direct report of Steven King.

- `THEN ... ELSE NULL`: Within the `IF` condition, the `THEN` condition grants the role by executing immediately the `SET ROLE` statement. Otherwise, the `ELSE` condition denies the grant.
- `EXCEPTION`: Use an `EXCEPTION` statement to set `v_manager_id` to 0 if no data is found.
- `DBMS_OUTPUT.PUT_LINE`: Copies the manager's ID, which is now 0, into a buffer so that it is readily available.

- Click **OK**.

Tip:

If you have problems creating or running PL/SQL code, check the Oracle Database trace files. The `USER_DUMP_DEST` initialization parameter specifies the current location of the trace files. You can find the value of this parameter by issuing `SHOW PARAMETER USER_DUMP_DEST` in SQL*Plus. See *Oracle Database Administrator's Guide* for more information about trace files.

3.4.2.5 Step 5: Create the Secure Application Role

After you create the PL/SQL procedure to set the secure application role, you can create the `emp_role` secure application role.

To create the secure application role:

- Ensure that you are still logged in as user `sec_admin`.
- From the **Administration** menu, select **Security**, then **Roles**.
- In the Roles page, click **Create**.
- In the Create Role page, enter the following information:

- Name:** `emp_role`
- Authentication:** Select **Application**.

The page expands to show the additional prompts necessary to create the secure application role.

The screenshot shows the 'Create Role' dialog box in Oracle Enterprise Manager. The 'General' tab is selected. The 'Name' field is filled with 'EMP_ROLE'. Under 'Authentication', the 'Application' option is selected. Under 'Package', the 'Package' radio button is selected. The 'Package Name' field is empty. At the bottom of the dialog, there are four buttons: 'Execute On Multiple Databases', 'Show SQL', 'Cancel', and 'OK'.

- Select the **Procedure** option.
- In the **Procedure Name** field, enter `SEC_ADMIN.GRANT_EMP_ROLE`.
- Click **OK**.

3.4.2.6 Step 6: Grant SELECT for the EMP_ROLE Role to the OE.ORDERS Table

User `OE`, who owns the `OE.ORDERS` table, must grant the `SELECT` privilege for the `ORDERS` table to the `emp_role` role.

This gives anyone who has been authorized to use the `emp_role` role the ability to select from the `OE.ORDERS` table.

To grant the SELECT privilege for the EMP_ROLE role to the OE.ORDERS table:

- Connect as user `OE`.

```
CONNECT OE
Enter password: password
```

If you receive an error message saying that OE is locked, then you can unlock the OE account and reset its password by entering the following statements. For greater security, do not reuse the same password that was used in previous releases of Oracle Database. Enter any password that is secure, according to the password guidelines described in [Requirements for Creating Passwords](#) (page 2-17).

```
CONNECT SYSTEM
Enter password: sys_password
```

```
PASSWORD OE -- First, change the OE account password.
Changing password for OE
New password: password
Retype new password: password
Password changed.
```

```
ALTER USER OE ACCOUNT UNLOCK; -- Next, unlock the OE account.
```

Another way to unlock a user account and create a new password is to use the following syntax:

```
ALTER USER account_name ACCOUNT UNLOCK IDENTIFIED BY new_password:
```

Now you can connect as user OE.

```
CONNECT OE
Enter password: password
```

2. Enter the following statement to grant the `emp_role` role SELECT privileges on the `OE.ORDERS` table.

```
GRANT SELECT ON ORDERS TO emp_role;
```

Do not grant the role directly to the users. The PL/SQL package will do that for you, assuming the users pass its security policies.

3.4.2.7 Step 7: Grant the EXECUTE Privilege for the Procedure to Matthew and Winston

At this stage, Matthew and Winston can try to access the `OE.ORDERS` table, but they are denied access.

The next step is to grant them the EXECUTE privilege on the `grant_emp_role` procedure, so that the `grant_emp_role` procedure can execute, and then grant or deny access, when they try to select from the `OE.ORDERS` table.

To grant EXECUTE privileges for the `grant_emp_role` procedure:

1. In SQL*Plus, log in as user `sec_admin`.

```
connect sec_admin
Enter password: password
```

2. Run the following GRANT SQL statements for users `mweiss` and `wtaylor`:

```
GRANT EXECUTE ON grant_emp_role TO mweiss;
GRANT EXECUTE ON grant_emp_role TO wtaylor;
```

3.4.2.8 Step 8: Test the EMP_ROLE Secure Application Role

To test the `emp_role` secure application role, you must log on as Matthew and Winston and trying to access the `OE.ORDERS` table.

When Matthew and Winston log on, and before they issue a `SELECT` statement on the `OE.ORDERS` table, the `grant_emp_role` procedure must be executed for the role verification to take place.

[Testing the emp_role Secure Application Role as User MWEISS](#) (page 3-12)
You can connect to the database as user `mweiss` using SQL*Plus.

[Testing the emp_role Secure Application Role as User WTAYLOR](#) (page 3-12)
Next, Winston tries to access the secure application.

3.4.2.8.1 Testing the emp_role Secure Application Role as User MWEISS

You can connect to the database as user `mweiss` using SQL*Plus.

To test the emp_role secure application role as user MWEISS:

1. In SQL*Plus, connect as user `mweiss`.

```
CONNECT mweiss
Enter password: password
```

2. Enter the following SQL statement to run the `grant_emp_role` procedure:

```
EXEC sec_admin.grant_emp_role;
```

This statement executes the `grant_emp_role` procedure for the current session. (In a real world scenario, this statement would be automatically run when the user logs in to the application.)

3. Perform the following `SELECT` statement on the `OE.ORDERS` table:

```
SELECT COUNT(*) FROM OE.ORDERS;
```

Matthew has access to the `OE.ORDERS` table:

```
COUNT(*)
-----
        105
```

3.4.2.8.2 Testing the emp_role Secure Application Role as User WTAYLOR

Next, Winston tries to access the secure application.

To test the emp_role secure application role as user WTAYLOR:

1. In SQL*Plus, connect as user `wtaylor`.

```
CONNECT wtaylor
Enter password: password
```

2. Enter the following SQL statement to run the `grant_emp_role` procedure:

```
EXEC sec_admin.grant_emp_role;
```

This statement executes the `grant_emp_role` procedure for the current session.

3. Perform the following `SELECT` statement on the `OE.ORDERS` table:

```
SELECT COUNT(*) FROM OE.ORDERS;
```

```
ERROR at line 1:
ORA-00942: table or view does not exist
```

Because Winston does not report directly to Steven King, he does not have access to the OE.ORDERS table. He will never learn the true number of orders in the ORDERS table, at least not by performing a SELECT statement on it.

3.4.2.9 Step 9: Optionally, Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

To remove the components:

1. In SQL*Plus, connect as user SYSTEM.

```
CONNECT SYSTEM
Enter password: password
```

2. Enter the following DROP statements:

```
DROP USER mweiss;
DROP USER wtaylor;
```

Do not drop user sec_admin. You will need this user account for other tutorials in this guide.

3. In SQL*Plus, connect as user sec_admin.

```
CONNECT sec_admin
Enter password: password
```

4. Enter the following DROP SQL statements:

```
DROP ROLE emp_role;
DROP PROCEDURE grant_emp_role;
```

5. Connect as user HR, and then drop the HR_VERIFY view.

```
CONNECT HR
Enter password: password
DROP VIEW hr_verify;
```

6. Exit SQL*Plus.

```
EXIT
```

3.5 Initialization Parameters Used for Privilege Security

Oracle Database provides initialization parameters to configure privilege security, such as the restriction of SYSTEM privileges.

[Table 3-1](#) (page 3-14) lists initialization parameters that you can use to secure user privileges.

Table 3-1 Initialization Parameters Used for Privilege Security

| Initialization Parameter | Default | Description |
|-----------------------------|---------|---|
| O7_DICTIONARY_ACCESSIBILITY | FALSE | Controls restrictions on SYSTEM privileges. See Enabling Data Dictionary Protection (page 2-4) for more information about this parameter. |
| OS_ROLES | FALSE | Determines whether the operating system identifies and manages the roles of each user. |
| REMOTE_OS_ROLES | FALSE | Specifies whether operating system roles are allowed for remote clients. The default value, FALSE, causes Oracle to identify and manage roles for remote clients. |
| SQL92_SECURITY | TRUE | Specifies whether users must be granted the SELECT object privilege to execute UPDATE or DELETE statements. |

To modify an initialization parameter, see [Modifying the Value of an Initialization Parameter](#) (page 2-5). For detailed information about initialization parameters, see *Oracle Database Reference*.

Encrypting Data with Oracle Transparent Data Encryption

Transparent Data Encryption enables you to disguise data in table columns and in an entire tablespace.

[About Encrypting Sensitive Data](#) (page 4-1)

Encrypted data is data that has been disguised so that only an authorized recipient can read it.

[When Should You Encrypt Data?](#) (page 4-2)

In most cases, you must encrypt sensitive data on your site to meet a regulatory compliance.

[How Transparent Data Encryption Works](#) (page 4-2)

Transparent Data Encryption enables you to encrypt individual table columns or an entire tablespace.

[Configuring Data to Use Transparent Data Encryption](#) (page 4-4)

To start using Transparent Data Encryption, you must create a keystore and set a master key.

[Checking Existing Encrypted Data](#) (page 4-12)

You can query the database for the data that you have encrypted.

4.1 About Encrypting Sensitive Data

Encrypted data is data that has been disguised so that only an authorized recipient can read it.

You use encryption (Transparent Data Encryption, or TDE) to protect data in a potentially unprotected environment, such as data you have placed on backup media that is sent to an offsite storage location.

Encrypting data includes the following components:

- **An algorithm to encrypt the data.** Oracle Databases use the encryption algorithm to encrypt and decrypt data. Oracle Database supports several industry-standard encryption and hashing algorithms, including the Advanced Encryption Standard (AES) encryption algorithm, which has been approved by the National Institute of Standards and Technology (NIST).
- **A key to encrypt and decrypt data.** When you encrypt data, Oracle Database uses the key and plain text data as input into the encryption algorithm. Conversely, when you decrypt data, the key is used as input into the algorithm to reverse the process and retrieve the clear text data. Oracle Database uses a symmetric encryption key to perform this task, in which the same key is used to both encrypt and decrypt the data. The encryption key is stored in the data dictionary, but encrypted with another master key.

You can encrypt individual table columns or an entire tablespace. Be careful that you do not mix the two. For example, suppose you encrypt a table column and then encrypt its surrounding tablespace. This double encryption can cause performance problems. In addition, column encryption has limitations in data type support, and only supports B-tree indexes for equality searches. To check the current encrypted settings, you can query the `V$ENCRYPTED_TABLESPACES` data dictionary view for tablespaces and the `DBA_ENCRYPTED_COLUMNS` view for encrypted columns.

See Also: *Oracle Database Advanced Security Guide* for detailed information about TDE

4.2 When Should You Encrypt Data?

In most cases, you must encrypt sensitive data on your site to meet a regulatory compliance.

For example, sensitive data such as credit card numbers, Social Security numbers, or patient health information must be encrypted.

Historically, users have wanted to encrypt data to restrict data access from their database administrators. However, this problem is more of an access control problem, not an encryption problem. You can address this problem by using Oracle Database Vault to control the access to your application data from database administrators.

In most cases, you encrypt sensitive data, such as credit cards and Social Security numbers, to prevent access when backup tapes or disk drives are lost or stolen. In recent years, industry regulations such as the Payment Card Industry (PCI) Data Security Standard and the Healthcare Insurance Portability and Accountability Act (HIPAA) have become a driving factor behind increased usage of encryption for protecting credit card and health care information, respectively.

4.3 How Transparent Data Encryption Works

Transparent Data Encryption enables you to encrypt individual table columns or an entire tablespace.

When a user inserts data into an encrypted column, Transparent Data Encryption automatically encrypts the data. When authorized users select the column, then the data is automatically decrypted.

To encrypt data by using Transparent Data Encryption, you create the following components:

- **A keystore to store the master encryption key.** The keystore is an operating system file that is located outside the database. The database uses the keystore to store the master encryption key. To create the keystore, you can use the `ADMINISTER KEY MANAGEMENT` SQL statement. The keystore is encrypted using a password as the encryption key. You create the password when you create the keystore. Access to the contents (or master key) of the keystore is then restricted to only those who know the password. After the keystore is created, you must open the keystore using the password so that the database can access the master encryption key.

You can use either software keystores or hardware keystores. A software keystore is defined in a file that you create in a directory location. The software keystore can be one of the following types:

- **Password-based keystores:** Password-based keystores are protected by using a password that you create. You must open the keystore before the keys can be retrieved or used.
- **Auto-login keystores:** Auto-login keystores are protected by a system-generated password, and do not need to be explicitly opened by a security administrator. Auto-login keystores are automatically opened when accessed. Auto-login keystores can be used across different systems. If your environment does not require the extra security provided by a keystore that must be explicitly opened for use, then you can use an auto-login keystore.
- **Auto-login local keystores:** Auto-login local keystores are auto-login keystores that are local to the system on which they are created. Auto-login local keystores cannot be opened on any computer other than the one on which they are created.

A hardware keystore is used with a hardware security module, which is a physical device that is designed to provide secure storage for encryption keys. This guide explains how to configure software keystores only. For detailed information about hardware keystores, see *Oracle Database Advanced Security Guide*.

- **A location for the keystore.** You must specify the keystore location in the `sqlnet.ora` file.

Afterward, when a user enters data, Oracle Database performs the following steps:

1. Retrieves the master key from the keystore.
2. Decrypts the encryption key using the master key.
3. Uses the encryption key to encrypt the data the user entered.
4. Stores the data in encrypted format in the database.

If the user is selecting data, the process is similar: Oracle Database decrypts the data and then displays it in plain text format.

Transparent Data Encryption has the following benefits:

- As a security administrator, you can be sure that sensitive data is safe if the storage media or data file is stolen or lost.
- Implementing Transparent Data Encryption helps you address security-related regulatory compliance issues.
- Data from tables is transparently decrypted for the database user. You do not need to create triggers or views to decrypt data.
- Database users do not need to be aware that the data they are accessing is stored in encrypted form. Data is transparently decrypted for the database users and does not require any action on their part.
- Applications need not be modified to handle encrypted data. Data encryption and decryption is managed by the database.

Transparent Data Encryption has a minimal impact on performance. Transparent Data Encryption column encryption affects performance only when data is retrieved from or inserted into an encrypted column. There is no impact on performance for operations involving unencrypted columns, even if these columns are in a table

containing encrypted columns. However, be aware that encrypted data must have more storage space than plain text data. On average, encrypting a single column requires between 32 and 48 bytes of additional storage for each row. Transparent tablespace encryption provides even better performance because Oracle Database performs the encryption and decryption at the I/O block layer. Once blocks are decrypted, they are cached in Oracle Database memory for optimal performance.

See Also:

Oracle Database Advanced Security Guide for detailed information about using Transparent Data Encryption

4.4 Configuring Data to Use Transparent Data Encryption

To start using Transparent Data Encryption, you must create a keystore and set a master key.

The keystore should be a separate keystore specifically used by Transparent Data Encryption. This keystore will be used for all data that is being encrypted through Transparent Data Encryption.

Step 1: Configure the Keystore Location (page 4-4)

When you create a software password-based keystore, you must designate the directory location for the keystore in the `sqlnet.ora` file.

Step 2: Check the COMPATIBLE Initialization Parameter Setting (page 4-5)

To configure the full set of tablespace encryption features, you must set the `COMPATIBLE` initialization parameter for the database to `11.2.0.0` or later.

Step 3: Create the Software Password-Based Keystore (page 4-6)

To create the keystore, use the `ADMINISTER KEY MANAGEMENT SQL` statement.

Step 4: Open (or Close) the Keystore (page 4-7)

You can manually open and close keystores. Auto-login keystores open automatically when they are accessed.

Step 5: Create the Master Encryption Key (page 4-8)

The master encryption key, which stored in a keystore, protects the table keys and tablespace encryption keys.

Step 6: Encrypt Data (page 4-8)

Next, you are ready to encrypt either individual table columns or an entire tablespace.

4.4.1 Step 1: Configure the Keystore Location

When you create a software password-based keystore, you must designate the directory location for the keystore in the `sqlnet.ora` file.

You perform this step only once.

To configure the keystore location:

1. Create a directory in the `$ORACLE_HOME` directory to store the keystore.

For example, on Microsoft Windows, you could create a directory called `ORA_KEYSTORES` in the `C:\oracle\product\12.2.0\db_1` directory.

2. Create a backup copy of the `sqlnet.ora` file, which by default is located in the `$ORACLE_HOME/network/admin` directory.
3. At the end of the `sqlnet.ora` file, add code similar to the following, where `ORA_KEYSTORES` is the name of the directory where you plan to store the keystore:

```
ENCRYPTION_WALLET_LOCATION=
(SOURCE=
(METHOD=file)
(METHOD_DATA=
(DIRECTORY=C:\oracle\product\12.2.0\db_1\ORA_KEYSTORES)))
```

4. Save and close the `sqlnet.ora` file.

4.4.2 Step 2: Check the COMPATIBLE Initialization Parameter Setting

To configure the full set of tablespace encryption features, you must set the `COMPATIBLE` initialization parameter for the database to `11.2.0.0` or later.

Otherwise, ensure that it is at least `11.0.0.0`. Be aware that once you set this parameter, you cannot change it. Ideally, you should set `COMPATIBLE` to accommodate the most current release of Oracle Database.

To set the COMPATIBLE initialization parameter:

1. Log into the database instance.

For example:

```
sqlplus sec_admin
Enter password: password
Connected.
```

2. Check the current setting of the `COMPATIBLE` parameter.

For example:

```
SHOW PARAMETER COMPATIBLE
```

| NAME | TYPE | VALUE |
|------------|--------|----------|
| compatible | string | 11.0.0.0 |

3. If you must change the `COMPATIBLE` parameter, then complete the remaining steps in this procedure.

The value should be `11.2.0.0` or higher.

4. Locate the initialization parameter file for the database instance.
 - **UNIX systems:** This file is in the `ORACLE_HOME/dbs` directory and is named `initORACLE_SID.ora` (for example, `initmydb.ora`).
 - **Windows systems:** This file is in the `ORACLE_HOME\database` directory and is named `initORACLE_SID.ora` (for example, `initmydb.ora`).
5. Edit the initialization parameter file to use the correct `COMPATIBLE` setting.

For example:

```
COMPATIBLE = 12.2.0.0
```

6. In SQL*Plus, log in as a user who has the SYSDBA administrative privilege.

```
sqlplus sys as sysdba  
Enter password: password
```

7. Restart the Oracle Database instance.

For example:

```
SHUTDOWN IMMEDIATE  
STARTUP
```

8. Do not log out of SQL*Plus.

4.4.3 Step 3: Create the Software Password-Based Keystore

To create the keystore, use the `ADMINISTER KEY MANAGEMENT` SQL statement.

By default, the Oracle keystore stores a history of retired master keys, which enables you to change them and still be able to decrypt data that was encrypted under an old master key. A case-sensitive keystore password unknown to the database administrator provides separation of duty: a database administrator can restart the database, but the keystore is closed and must be manually opened by a security administrator before the database can encrypt or decrypt the data.

To create the keystore:

1. In SQL*Plus, connect as a user who has been granted the SYSKM administrative privilege.

For example:

```
CONNECT psmith / AS SYSKM  
Enter password: password
```

2. Run the following `ADMINISTER KEY MANAGEMENT` SQL statement:

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE 'keystore_location' IDENTIFIED BY  
software_keystore_password;
```

In this specification:

- *keystore_location* is the path to the keystore location that you defined in the `sqlnet.ora` file (for example, `oracle\product\12.2.0\db_1\ORA_KEYSTORES`). Enclose the *keystore_location* setting in single quotation marks. To find this location, query the `WRL_PARAMETER` column of the `V$ENCRYPTION_WALLET` view.
- *software_keystore_password* is a new password that you, the security administrator, creates.

For example, to create the keystore in the `c:\oracle\product\12.2.0\db_1\ORA_KEYSTORES` directory:

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE 'c:\oracle\product  
\12.2.0\db_1\ORA_KEYSTORES' IDENTIFIED BY password;
```

```
keystore altered.
```

After you run this statement, the `ewallet.p12` file, which contains the keystore, appears in the keystore location.

4.4.4 Step 4: Open (or Close) the Keystore

You can manually open and close keystores. Auto-login keystores open automatically when they are accessed.

[Opening a Keystore](#) (page 4-7)

After you create a software password-based keystore, you must manually open it before you can use Transparent Data Encryption.

[Closing a Keystore](#) (page 4-7)

You can close a keystore to disable access to the master key and prevent access to the encrypted columns.

4.4.4.1 Opening a Keystore

After you create a software password-based keystore, you must manually open it before you can use Transparent Data Encryption.

You do not need to open the auto-login or hardware keystores because they open automatically. You can check the status of whether a keystore is open or closed by querying the `STATUS` column of the `V$ENCRYPTION_WALLET` view.

To open a keystore:

1. Ensure that you are logged into SQL*Plus as a user who has been granted the `SYSKM` system privilege.
2. Enter the following `ADMINISTER KEY MANAGEMENT` SQL statement:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
software_keystore_password;
```

keystore altered.

Replace `software_keystore_password` with the password that you created in [Step 3: Create the Software Password-Based Keystore](#) (page 4-6).

4.4.4.2 Closing a Keystore

You can close a keystore to disable access to the master key and prevent access to the encrypted columns.

In most cases, leave the keystore open unless you have a reason for closing it. The keystore must be open for Transparent Data Encryption to work. To reopen the keystore, use the `ADMINISTER KEY MANAGEMENT` statement.

To close a keystore:

1. Ensure that you are logged into SQL*Plus as a user who has been granted the `SYSKM` system privilege.
2. Enter the following SQL statement:

```
ADMINISTER KEY MANAGEMENT SET KEYSTORE CLOSE IDENTIFIED BY
software_keystore_password;
```

4.4.5 Step 5: Create the Master Encryption Key

The master encryption key, which is stored in a keystore, protects the table keys and tablespace encryption keys.

By default, the master encryption key is a random key generated by Transparent Data Encryption (TDE).

To create the master encryption key:

1. Ensure that you are logged into SQL*Plus as a user who has been granted the SYSKM system privilege.
2. Run the following ADMINISTER KEY MANAGEMENT SQL statement:

```
ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY software_keystore_password [WITH  
BACKUP [USING 'backup_identifier']];
```

keystore altered.

In this specification:

- *software_keystore_password* is the password that you created in [Step 3: Create the Software Password-Based Keystore](#) (page 4-6).
- WITH BACKUP creates a backup of the keystore. You must use this option for password-based keystores. You do not need to use it for auto-login or auto-login local keystores. Optionally, you can use the USING clause to add a brief description of the backup. Enclose this description in double quotation marks. This identifier is appended to the named keystore file (for example, *ewallet_timestamp_emp_key_backup.p12*).

For example:

```
ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY  
software_keystore_password WITH BACKUP USING 'password key backup';
```

4.4.6 Step 6: Encrypt Data

Next, you are ready to encrypt either individual table columns or an entire tablespace.

[Encrypting Individual Table Columns](#) (page 4-8)

Oracle provides guidelines that you should follow before you encrypt columns, such as checking the column data types.

[Encrypting a Tablespace](#) (page 4-11)

You can encrypt a new tablespace while you are creating it, but you cannot encrypt an existing tablespace.

4.4.6.1 Encrypting Individual Table Columns

Oracle provides guidelines that you should follow before you encrypt columns, such as checking the column data types.

The decisions that you make when you identify columns to encrypt are determined by governmental security regulations, such as California Senate Bill 1386, or by industry standards such as the Payment Card Industry (PCI) Data Security Standard. Credit card numbers, Social Security numbers, and other personally identifiable information (PII) fall under this category. Your own internal security policies — trade secrets,

research results, or employee salaries and bonuses — determine your needs for encryption. See [When Should You Encrypt Data?](#) (page 4-2) for guidelines about when and when not to encrypt data.

Follow these guidelines when you select columns to encrypt:

- **Check the data types of the columns you plan to encrypt.** Transparent Data Encryption supports the following data types:

| Data Types B-L | Data Types N-V |
|--|-----------------------|
| BINARY_FLOAT | NUMBER |
| BINARY_DOUBLE | NVARCHAR2 |
| CHAR | RAW |
| DATE | TIMESTAMP |
| NCHAR | VARCHAR2 |
| Large object types (LOBs) such as BLOB and CLOB ¹ | - |

¹ You cannot encrypt external LOBs (BFILE).

- **Ensure that the columns you select are not part of a foreign key.** With Transparent Data Encryption, each table has its own encryption key, which is stored in the database data dictionary and encrypted with the external master key. Encrypted columns cannot be used as foreign keys.

To encrypt a column in a table:

1. Ensure that you have created and opened the keystore and created a master encryption key.

See the following sections, if necessary:

- [Step 3: Create the Software Password-Based Keystore](#) (page 4-6) to learn how to create a keystore key
 - [Step 4: Open \(or Close\) the Keystore](#) (page 4-7) to learn how to open or a keystore
 - [Step 5: Create the Master Encryption Key](#) (page 4-8) to create the master encryption key
2. In Enterprise Manager, access the Database home page.
See *Oracle Database 2 Day DBA* for more information.
 3. From the **Schema** menu, select **Database Objects**, then **Tables**.
If the Database Login page appears, then log in as SYS with the SYSDBA administrative privilege.
 4. In the Tables page, do one of the following:
 - To create a new table, click **Create**, and then answer the questions in the subsequent page to start creating the table.

- To modify an existing table, search for the table name by entering its schema name into the **Schema** field and the table name in the **Object Name** field. (You can use the percent sign (%) wildcard character to search for a group of tables, for example O% to find all tables beginning with the letter O.) When the table is listed in the Tables page, select the table, and then click **Edit**.

In the Create Table or Edit Table page, you can set the encryption options.

For example, to encrypt columns in the OE.ORDERS table, the Edit Table page appears as follows:

| Select | Name | Data Type | Size | Scale | Not NULL | Default Value | Encrypted |
|----------------------------------|--------------|-----------|------|-------|-------------------------------------|---------------|--------------------------|
| <input checked="" type="radio"/> | ORDER_ID | NUMBER | 12 | | <input checked="" type="checkbox"/> | | <input type="checkbox"/> |
| <input type="radio"/> | ORDER_DATE | TIMESTAMP | 6 | | <input checked="" type="checkbox"/> | | <input type="checkbox"/> |
| <input type="radio"/> | ORDER_MODE | VARCHAR2 | 8 | | <input type="checkbox"/> | | <input type="checkbox"/> |
| <input type="radio"/> | CUSTOMER_ID | NUMBER | 6 | | <input checked="" type="checkbox"/> | | <input type="checkbox"/> |
| <input type="radio"/> | ORDER_STATUS | NUMBER | 2 | | <input type="checkbox"/> | | <input type="checkbox"/> |
| <input type="radio"/> | ORDER_TOTAL | NUMBER | 8 | 2 | <input type="checkbox"/> | | <input type="checkbox"/> |
| <input type="radio"/> | SALES_REP_ID | NUMBER | 6 | | <input type="checkbox"/> | | <input type="checkbox"/> |
| <input type="radio"/> | PROMOTION_ID | NUMBER | 6 | | <input type="checkbox"/> | | <input type="checkbox"/> |
| <input type="radio"/> | | VARCHAR2 | | | <input type="checkbox"/> | | <input type="checkbox"/> |
| <input type="radio"/> | | VARCHAR2 | | | <input type="checkbox"/> | | <input type="checkbox"/> |

5. In the Create Table (or Edit Table) page, do the following:
 - a. Select the column that you want to encrypt.

Do not select columns that are part of a foreign key constraint (primary or unique key columns). You cannot encrypt these columns. These columns are indicated with a key or check mark icon to the left of their names.
 - b. Click **Encryption Options** to display the Encryption Options for the Table page.
 - c. From the Encryption Algorithm list, select from the following options:
 - **AES192**: Sets the key length to 192 bits. AES is the abbreviation for Advanced Encryption Standard.
 - **3DES168**: Sets the key length to 168 bits. 3DES is the abbreviation for Triple Data Encryption Standard.
 - **AES128**: Sets the key length to 128 bits. This option is the default.
 - **AES256**: Sets the key length to 256 bits.
 - d. Under Key Generation, select either **Generate Key Randomly** or **Specify Key**. If you select **Specify Key**, enter characters for the seed values in the **Enter Key** and **Confirm Key** fields.

The **Generate Key Randomly** setting enables salt. **Salt** is a way to strengthen the security of encrypted data. It is a random string added to the data before it is encrypted, causing the same text to appear different when encrypted. Salt

removes one method attackers use to steal data, namely, matching patterns of encrypted text.

- e. Click **Continue** to return to the Create Table (or Edit Table) page.
 - f. Enable encryption for the column by selecting its box under **Encrypted**.
6. Click **Apply**, and then click **Return**.

The Tables page appears.

While a table is being updated, read access is still possible. Afterward, existing and future data in the column is encrypted when it is written to the database file, and it is decrypted when an authorized user selects it. If data manipulation language (DML) statements are needed, you can use online redefinition statements.

4.4.6.2 Encrypting a Tablespace

You can encrypt a new tablespace while you are creating it, but you cannot encrypt an existing tablespace.

As a workaround, you can use the `CREATE TABLE AS SELECT`, `ALTER TABLE MOVE`, or use Oracle Data Pump import to get data from an existing tablespace into an encrypted tablespace. For details about creating a tablespace, see *Oracle Database 2 Day DBA*.

To encrypt a tablespace:

1. Ensure that you have created and opened the keystore, as described in the preceding steps of this section.
2. In Enterprise Manager, access the Database home page.
See *Oracle Database 2 Day DBA* for more information.
3. From the **Administration** menu, select **Storage**, then **Tablespaces**.
If the Database Login page appears, then log in as an administrative user, such as SYS. User SYS must log in with the **SYSDBA** role selected.
The Tablespaces page appears.
4. Click **Create**, and then answer the questions in the subsequent page to start creating the tablespace and its required data file.
5. In the Create Tablespace page, do the following:
 - a. Under Type, in the Permanent list, select the **Encryption** box.
 - b. Under Datafiles, select **Add** to add a data file. (Linux and Windows systems only)
 - c. Select **Encryption** options to display the Encryption Options page.
 - d. From the Encryption Algorithm list, select from the following options:
 - **AES192**: Sets the key length to 192 bits. AES is the abbreviation for Advanced Encryption Standard.
 - **3DES168**: Sets the key length to 168 bits. 3DES is the abbreviation for Triple Data Encryption Standard.

- **AES128:** Sets the key length to 128 bits. This option is the default.
- **AES256:** Sets the key length to 256 bits.

See "Available Methods" under Step 5 (page 2-8) in [Configuring Network Encryption](#) (page 2-7) for more information about these encryption algorithms.

e. Click **Continue**.

6. In the Create Tablespace page, click **OK**.

The new tablespace appears in the list of existing tablespaces. Remember that you cannot encrypt an existing tablespace.

See Also:

- [Data Dictionary Views for Checking Encrypted Tablespaces](#) (page 4-14) to query the database for existing encrypted tablespaces
 - *Oracle Database Advanced Security Guide* for detailed information about tablespace encryption
 - *Oracle Database Reference* for more information about the CREATE TABLESPACE statement
-
-

4.5 Checking Existing Encrypted Data

You can query the database for the data that you have encrypted.

You can check for individually encrypted columns, all tables in the current database instance that have encrypted columns, or all tablespaces that are encrypted.

[Finding the Type of Keystore That Was Created](#) (page 4-13)

The V\$ENCRYPTION_KEYS dynamic view lists the type of keystore that was created.

[Finding the Keystore Location](#) (page 4-13)

The V\$ENCRYPTION_WALLET dynamic view lists the location of a keystore.

[Checking Whether a Keystore Is Open or Closed](#) (page 4-13)

The V\$ENCRYPTION_WALLET dynamic view indicates if a keystore is open or closed.

[Checking Encrypted Columns of an Individual Table](#) (page 4-13)

The DESC (for DESCRIBE) statement in SQL*Plus checks the encrypted columns in a database table.

[Checking All Encrypted Table Columns in the Current Database Instance](#) (page 4-14)

The DBA_ENCRYPTED_COLUMNS data dictionary view lists all encrypted table columns in the current instance.

[Data Dictionary Views for Checking Encrypted Tablespaces](#) (page 4-14)

Oracle Database provides data dictionary views that describe encrypted tablespaces.

4.5.1 Finding the Type of Keystore That Was Created

The V\$ENCRYPTION_KEYS dynamic view lists the type of keystore that was created.

To find the type of keystore that was created:

- In SQL*Plus, query the V\$ENCRYPTION_KEYS view as follows:

```
SELECT KEYSTORE_TYPE FROM V$ENCRYPTION_KEYS;
```

The keystore location appears, similar to the following:

```
KEYSTORE_TYPE
-----
SOFTWARE KEYSTORE
```

4.5.2 Finding the Keystore Location

The V\$ENCRYPTION_WALLET dynamic view lists the location of a keystore.

To find the keystore location:

- In SQL*Plus, query the V\$ENCRYPTION_WALLET view as follows:

```
SELECT WRL_PARAMETER FROM V$ENCRYPTION_WALLET;
```

The keystore location appears, similar to the following:

```
WRL_PARAMETER
-----
C:\oracle\product\12.2.0\db_1
```

4.5.3 Checking Whether a Keystore Is Open or Closed

The V\$ENCRYPTION_WALLET dynamic view indicates if a keystore is open or closed.

To check whether a keystore is open or closed:

- In SQL*Plus, query the V\$ENCRYPTION_WALLET view as follows:

```
SELECT STATUS FROM V$ENCRYPTION_WALLET;
```

The keystore status appears, similar to the following:

```
STATUS
-----
OPEN
```

4.5.4 Checking Encrypted Columns of an Individual Table

The DESC (for DESCRIBE) statement in SQL*Plus checks the encrypted columns in a database table.

To check the encrypted columns of an individual table:

- In SQL*Plus, run the DESC statement using the following syntax.

```
DESC tablename;
```

For example:

```
DESC OE.ORDER_ITEMS;
```

A description of the table schema appears. The following output shows that the QUANTITY column is encrypted:

| Name | Null? | Type |
|--------------|----------|--------------------------|
| ORDER_ID | NOT NULL | NUMBER(12) |
| LINE_ITEM_ID | NOT NULL | NUMBER(3) |
| PRODUCT_ID | NOT NULL | NUMBER(6) |
| UNIT_PRICE | | NUMBER(8,2) |
| QUANTITY | | NUMBER(8) ENCRYPT |

4.5.5 Checking All Encrypted Table Columns in the Current Database Instance

The DBA_ENCRYPTED_COLUMNS data dictionary view lists all encrypted table columns in the current instance.

To check all encrypted table columns in the current database instance:

- In SQL*Plus, select from the DBA_ENCRYPTED_COLUMNS view:

For example:

```
SELECT * FROM DBA_ENCRYPTED_COLUMNS;
```

This SELECT statement lists all tables and column in the database that contain columns encrypted using Oracle Transparent Data Encryption. For example:

| OWNER | TABLE_NAME | COLUMN_NAME | ENCRYPTION_ALG | SALT | INTEGRITY_ALG |
|-------|------------|--------------|------------------|------|---------------|
| OE | CUSTOMERS | INCOME_LEVEL | AES 128 bits key | YES | SHA-1 |
| OE | UNIT_PRICE | ORADER_ITEMS | AES 128 bits key | YES | SHA-1 |
| HR | EMPLOYEES | SALARY | AES 192 bits key | YES | SHA-1 |

See Also:

Oracle Database Reference for more information about the DBA_ENCRYPTED_COLUMNS view

4.5.6 Data Dictionary Views for Checking Encrypted Tablespaces

Oracle Database provides data dictionary views that describe encrypted tablespaces.

[Table 4-1](#) (page 4-15) lists data dictionary views that you can use to check encrypted tablespaces.

Table 4-1 Data Dictionary Views for Encrypted Tablespaces

| Data Dictionary View | Description | | | | | | | | | | | | | | | | |
|--------------------------|---|-----------------|---------------|-------------|----|--------|-----|----------|----|------|----|-------|----|---------|----|-------------|-----|
| DBA_TABLESPACES | <p>Describes all tablespaces in the database. For example, to determine if the tablespace has been encrypted, enter the following:</p> <pre>SELECT TABLESPACE_NAME, ENCRYPTED FROM DBA_TABLESPACES;</pre> <table border="1"> <thead> <tr> <th>TABLESPACE_NAME</th> <th>ENC</th> </tr> </thead> <tbody> <tr><td>SYSTEM</td><td>NO</td></tr> <tr><td>SYSAUX</td><td>NO</td></tr> <tr><td>UNCOTBS1</td><td>NO</td></tr> <tr><td>TEMP</td><td>NO</td></tr> <tr><td>USERS</td><td>NO</td></tr> <tr><td>EXAMPLE</td><td>NO</td></tr> <tr><td>SECURESPACE</td><td>YES</td></tr> </tbody> </table> | TABLESPACE_NAME | ENC | SYSTEM | NO | SYSAUX | NO | UNCOTBS1 | NO | TEMP | NO | USERS | NO | EXAMPLE | NO | SECURESPACE | YES |
| TABLESPACE_NAME | ENC | | | | | | | | | | | | | | | | |
| SYSTEM | NO | | | | | | | | | | | | | | | | |
| SYSAUX | NO | | | | | | | | | | | | | | | | |
| UNCOTBS1 | NO | | | | | | | | | | | | | | | | |
| TEMP | NO | | | | | | | | | | | | | | | | |
| USERS | NO | | | | | | | | | | | | | | | | |
| EXAMPLE | NO | | | | | | | | | | | | | | | | |
| SECURESPACE | YES | | | | | | | | | | | | | | | | |
| USER_TABLESPACES | <p>Describes the tablespaces accessible to the current user. It has the same columns as DBA_TABLESPACES, except for the PLUGGED_IN column.</p> | | | | | | | | | | | | | | | | |
| V\$ENCRYPTED_TABLESPACES | <p>Displays information about the tablespaces that are encrypted. For example:</p> <pre>SELECT * FROM V\$ENCRYPTED_TABLESPACES;</pre> <table border="1"> <thead> <tr> <th>TS#</th> <th>ENCRYPTIONALG</th> <th>ENCRYPTEDTS</th> </tr> </thead> <tbody> <tr> <td>6</td> <td>AES128</td> <td>YES</td> </tr> </tbody> </table> <p>The list includes the tablespace number, its encryption algorithm, and whether its encryption is enabled or disabled.</p> <p>If you want to find the name of the tablespace, use the following join operation:</p> <pre>SELECT NAME, ENCRYPTIONALG ENCRYPTEDTS FROM V\$ENCRYPTED_TABLESPACES, V\$TABLESPACE WHERE V\$ENCRYPTED_TABLESPACES.TS# = V\$TABLESPACE.TS#;</pre> | TS# | ENCRYPTIONALG | ENCRYPTEDTS | 6 | AES128 | YES | | | | | | | | | | |
| TS# | ENCRYPTIONALG | ENCRYPTEDTS | | | | | | | | | | | | | | | |
| 6 | AES128 | YES | | | | | | | | | | | | | | | |

See Also:

Oracle Database Reference for more information about data dictionary views

Controlling Access with Oracle Database Vault

Oracle Database Vault enables you to restrict administrative access to an Oracle database.

[About Oracle Database Vault](#) (page 5-1)

You can use Oracle Database Vault to restrict administrative access to an Oracle database using a fine-grained approach.

[Tutorial: Controlling Administrator Access to a User Schema](#) (page 5-2)

In this tutorial, you create a realm around the OE schema to protect it from administrator access but allow SCOTT to access OE . CUSTOMERS.

5.1 About Oracle Database Vault

You can use Oracle Database Vault to restrict administrative access to an Oracle database using a fine-grained approach.

This helps you address the most difficult security problems remaining today: protecting against insider threats, meeting regulatory compliance requirements, and enforcing separation of duty. In addition to restricting administrator access to your databases, Database Vault enables you to enforce separation of duty, and control who, when, where and how applications, databases, and data are accessed.

Typically, the main job of an Oracle database administrator is to perform tasks such as database tuning, installing upgrades, monitoring the state of the database, and then remedying any problems that he or she finds. In a default Oracle Database installation, database administrators also have the ability to create users and access user data. For greater security, you should restrict these activities only to those users who must perform them. This is called **separation of duty**, and it frees the database administrator to focus on tasks ideally suited to his or her expertise, such as performance tuning.

By restricting administrator access to your Oracle databases, Oracle Database Vault helps you to follow common regulatory compliance requirements, such as the Payment Card Industry (PCI) Data Security Standard (DSS) requirements, Sarbanes-Oxley (SOX) Act, European Union (EU) Privacy Directive, and Healthcare Insurance Portability and Accountability Act (HIPAA). These regulations require strong internal controls on access, disclosure or modification of sensitive information that could lead to fraud, identity theft, financial irregularities and financial penalties.

Oracle Database Vault provides the following ways for you to restrict administrator access to an Oracle database:

- **Group database schemas, objects, and roles that you want to secure.** This grouping is called a **realm**, and all the components of the realm are protected. After you, the Database Vault administrator, create a realm, you designate a user

to manage access to the realm. For example, you can create a realm around one table within a schema, or around the entire schema itself.

- **Create PL/SQL expressions to customize your database restrictions.** You create an expression in a **rule**, and for multiple rules within one category, you can group the rules into a **rule set**. To enforce the rules within the rule set, you then associate the rule set with a realm or command rule. For example, if you wanted to prevent access to a database during a maintenance period (for example, from 10 to 12 p.m.), you can create a rule to restrict access only during those hours.
- **Designate specific PL/SQL statements that are accessible or not accessible to users.** These are called **command rules**. A command rule contains a command to be protected and a rule set that determines whether the execution of the command is permitted. You can create a command rule to protect `SELECT`, `ALTER SYSTEM`, database definition language (DDL), and data manipulation language (DML) statements that affect one or more database objects. You can associate a rule set to further customize the command rule.
- **Define attributes to record data such as session users or IP addresses that Oracle Database Vault can recognize and secure.** These attributes are called **factors**. You can use factors for activities such as authorizing database accounts to connect to the database or creating filtering logic to restrict the visibility and manageability of data. To further customize the factor, you can associate a rule set with it.
- **Design secure application roles that are enabled only by Oracle Database Vault rules.** After you create the secure application role in Oracle Database Vault, you associate a rule set with it. The rule set defines when and how the secure application role is enabled or disabled.

You can create policies using these components by using either Oracle Database Vault Administrator, or by using its PL/SQL packages. In a multitenant environment, each policy applies only to the current pluggable database (PDB). [Step 1: Enable Oracle Database Vault](#) (page 5-3)

See Also: *Oracle Database Vault Administrator's Guide* for detailed information about Oracle Database Vault

5.2 Tutorial: Controlling Administrator Access to a User Schema

In this tutorial, you create a realm around the OE schema to protect it from administrator access but allow SCOTT to access OE.CUSTOMERS.

The OE schema has several tables that contain confidential data, such as the credit limits allowed for customers and other information. Order Entry tables typically contain sensitive information, such as credit card or Social Security numbers. This type of information must be restricted only to individuals whose job requires access to this information, according to Payment Card Industry (PCI) Data Security Standards (DSS).

[Step 1: Enable Oracle Database Vault](#) (page 5-3)

After you install Oracle Database, you must register Oracle Database Vault and enable the Oracle Database Vault Account Manager user account.

[Step 2: Grant SELECT on the OE.CUSTOMERS Table to User SCOTT](#)
(page 5-3)

To test the tutorial later on, user SCOTT must select from the OE . CUSTOMERS table.

[Step 3: Select from the OE.CUSTOMERS Table as Users SYS and SCOTT](#)
(page 5-5)

At this stage, both users SYS and SCOTT can select from the OE . CUSTOMERS table.

[Step 4: Create a Realm to Protect the OE.CUSTOMERS Table](#) (page 5-5)

To restrict the OE . CUSTOMER table from administrative access, you must create a realm around the OE schema.

[Step 5: Test the OE Protections Realm](#) (page 5-7)

Now that you have created a realm to protect the OE schema, you are ready to test it.

[Step 6: Optionally, Remove the Components for This Tutorial](#) (page 5-8)

You can remove the components that you created for this tutorial if you no longer need them.

5.2.1 Step 1: Enable Oracle Database Vault

After you install Oracle Database, you must register Oracle Database Vault and enable the Oracle Database Vault Account Manager user account.

Oracle Database Vault is installed when you perform a default installation of Oracle Database.

If Oracle Label Security is not enabled, then the registration process enables it as well as Database Vault.

To register Oracle Database Vault:

1. Log into the database instance as user SYS with the SYSDBA administrative privilege.

For example:

```
sqlplus sys as sysdba
Enter password: password
```

2. Check if Oracle Database Vault has already been enabled. The PARAMETER column is case sensitive, so use the case shown here.

```
SELECT * FROM DBA_DV_STATUS;
```

If it returns TRUE for both the DV_CONFIGURE_STATUS and the DV_ENABLE_STATUS, then Oracle Database Vault is registered. Go to [Step 2: Grant SELECT on the OE.CUSTOMERS Table to User SCOTT](#) (page 5-3). If it returns FALSE, then register Database Vault with your database, as described in *Oracle Database Vault Administrator's Guide*.

5.2.2 Step 2: Grant SELECT on the OE.CUSTOMERS Table to User SCOTT

To test the tutorial later on, user SCOTT must select from the OE . CUSTOMERS table.

[Enabling User SCOTT for Oracle Database Vault](#) (page 5-4)

You can use Enterprise Manager to enable user SCOTT.

[Granting User SCOTT the SELECT Privilege on the OE.CUSTOMERS Table](#) (page 5-4)

After you enable user SCOTT, you can grant him the appropriate privileges.

5.2.2.1 Enabling User SCOTT for Oracle Database Vault

You can use Enterprise Manager to enable user SCOTT.

To enable user SCOTT:

1. In Enterprise Manager, ensure that you are logged in as the Database Vault Account Manager (a user who has been granted the role DV_ACCTMGR) with the **NORMAL** role selected.

After you enable Oracle Database Vault, you no longer can use the administrative accounts (such as SYS and SYSTEM) to create or enable user accounts. This is because right out of the box, Oracle Database Vault provides separation-of-duty principles to administrative accounts. From now on, to manage user accounts, you must use the Oracle Database Vault Account Manager account.

Administrative users still have the privileges they do need. For example, user SYS, who owns system privileges and many PL/SQL packages, can still grant privileges on these to other users. However, user SYS can no longer create, modify, or drop user accounts. Instead, you must log in as the Database Vault Account Manager.

2. From the **Administration** menu, select **Security**, then **Users**.
3. In the Users page, select the user **SCOTT**, and in the View User page, click **Edit**.
The Edit User page appears.
4. Enter the following settings:
 - **Enter Password** and **Confirm Password**: If the SCOTT account password status is expired, then enter a new password. Enter any password that is secure, according to the password guidelines described in [Requirements for Creating Passwords](#) (page 2-17).
 - **Status**: Click **Unlocked**.
5. Click **Apply**.
6. Do not exit Enterprise Manager.

5.2.2.2 Granting User SCOTT the SELECT Privilege on the OE.CUSTOMERS Table

After you enable user SCOTT, you can grant him the appropriate privileges.

To grant user SCOTT the SELECT privilege on the OE.CUSTOMERS table:

1. Log in to SQL*Plus as user OE.

```
sqlplus oe
Enter password: password
Connected.
```

2. Grant user SCOTT the SELECT privilege on the OE.CUSTOMERS table.

```
GRANT SELECT ON CUSTOMERS TO SCOTT;
```

5.2.3 Step 3: Select from the OE.CUSTOMERS Table as Users SYS and SCOTT

At this stage, both users `SYS` and `SCOTT` can select from the `OE.CUSTOMERS` table.

This is because `SYS` has administrative privileges and `SCOTT` has an explicit `SELECT` privilege granted by user `OE`.

To select from `OE.CUSTOMERS` as users `SYS` and `SCOTT`:

1. In SQL*Plus, connect as user `SYS` using the `SYSDBA` administrative privilege

```
sqlplus sys as sysdba
Enter password: password
```

2. Select from the `OE.CUSTOMERS` table as follows:

```
SELECT COUNT(*) FROM OE.CUSTOMERS;
```

The following output should appear

```
COUNT(*)
-----
      319
```

3. Connect as user `SCOTT`, and then perform the same `SELECT` statement.

```
CONNECT SCOTT
Enter password: password
Connected.
```

```
SELECT COUNT(*) FROM OE.CUSTOMERS;
```

The following output should appear:

```
COUNT(*)
-----
      319
```

5.2.4 Step 4: Create a Realm to Protect the OE.CUSTOMERS Table

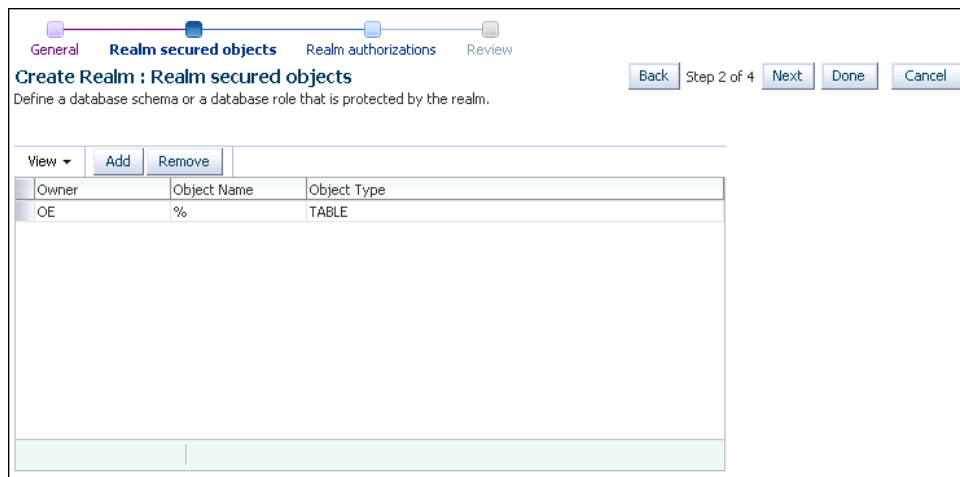
To restrict the `OE.CUSTOMER` table from administrative access, you must create a realm around the `OE` schema.

To create a realm around the `OE` schema:

1. In Enterprise Manager, click **Log Out** to log out of the database.
2. In the Confirmation dialog box, select **Logout of (Database Instance)** and then select the **Display login page after logout** check box. Then click **Logout**.
3. Log in as a user who has been granted the `DV_OWNER` or `DV_ADMIN` account (for example, `dbv_owner`). Connect using the **Normal Role** role.
4. From the **Security** menu, select **Database Vault**.
5. In the Database Vault page, select the **Administration** tab.
6. Under Database Vault Components, select **Realms**.

The Realms page appears.

7. Click **Create**.
8. In the Create Realm page, enter the following information:
 - **Name:** OE Protections
 - **Description:** Realm to protect the OE schema
 - **Status:** Click **Enabled**.
 - **Audit Options:** Select **Audit on Failure**.
9. Click **Next**.
The Realm Secured Objects page appears.
10. Click **Add**.
11. In the Add Secured Objects window, add the following information:
 - **Owner:** OE
 - **Object Type:** TABLE
 - **Object Name:** %
12. Click **OK**.
The OE table is now listed as a realm-secured object.

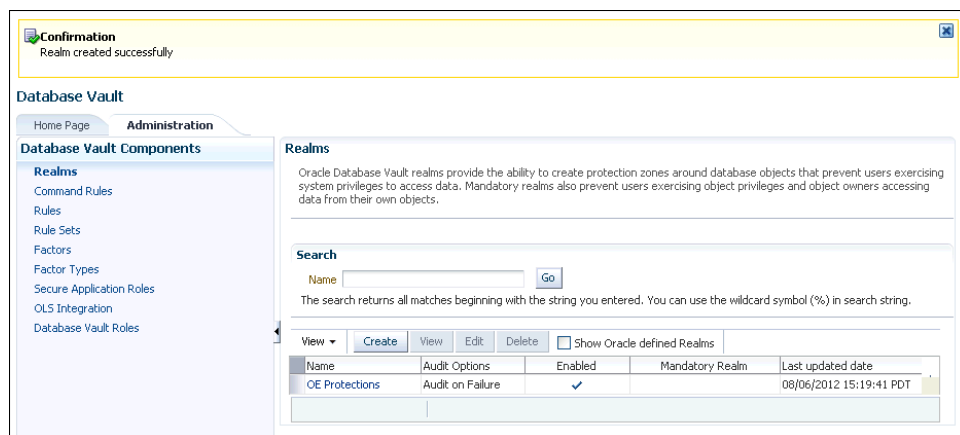


13. Click **Next**.
14. In the Realm Authorizations page, click **Add**.
The Add Authorizations window appears.
15. Enter the following information:
 - **Realm Authorization Grantee:** Select **OE**.
 - **Realm Authorization Type:** Select **Owner**.
 - **Realm Authorization Rule Set:** Select **Disabled**.
16. Click **OK**, and then click **Next**.

The Review page appears, so that you can check your settings.

17. Click Finish.

The Realms page now shows the OE Protections realm.



18. Do not exit Enterprise Manager.

5.2.5 Step 5: Test the OE Protections Realm

Now that you have created a realm to protect the OE schema, you are ready to test it.

You do not need to restart the database session, because any protections you define in Oracle Database Vault take effect right away.

To test the OE Protections realm:

1. Connect to SQL*Plus as user SYS using the SYSDBA administrative privilege.

```
CONNECT SYS AS SYSDBA
Enter password: password
Connected.
```

If you were connected as SYS before, then you do not need to reconnect. The changes that you just made take effect immediately.

2. Try selecting from the OE.CUSTOMERS table.

```
SELECT COUNT(*) FROM OE.CUSTOMERS;
```

The following output should appear:

```
ERROR at line 1:
ORA-01031: insufficient privileges
```

The OE Protections realm prevents the administrative user from accessing the OE.CUSTOMERS table. Because you defined the OE Protections realm to protect the entire schema, the administrative user does not have access to any of the other tables in OE, either.

3. Connect as user SCOTT.

```
CONNECT SCOTT
Enter password: password
Connected.
```

4. Try selecting from the OE . CUSTOMERS table.

```
SELECT COUNT(*) FROM OE.CUSTOMERS;
```

The following output should appear:

```
COUNT(*)
-----
      319
```

The OE Protections realm does not apply to user SCOTT because user OE has explicitly granted this user the SELECT privilege on the OE . CUSTOMERS table. Oracle Database Vault sets up the protections that you need, but does not override the explicit privileges you have defined. SCOTT still can query this table.

5.2.6 Step 6: Optionally, Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

[Dropping the OE Protections Realm](#) (page 5-8)

You can use Enterprise Manager to drop the OE protections realm.

[Revoking the SELECT Privilege on OE.CUSTOMERS from User SCOTT](#) (page 5-8)

You can use SQL*Plus to revoke the SELECT privilege on OE . CUSTOMERS from user SCOTT.

[Disabling Oracle Database Vault and Oracle Label Security](#) (page 5-9)

You can use SQL*Plus to disable Oracle Database Vault and Oracle Label Security.

5.2.6.1 Dropping the OE Protections Realm

You can use Enterprise Manager to drop the OE protections realm.

To drop the OE Protections realm:

1. In Enterprise Manager, if you have logged out of the Database Vault Administrator pages, then log back in as the Database Vault Owner account that you created when you installed Oracle Database Vault (for example, dbv_owner).
2. From the **Security** menu, select **Database Vault**.
3. In the Oracle Database Vault page, select the **Administration** tab.
4. Under Database Vault Feature Administration, click **Realms**.
The Realms page appears.
5. Select **OE Protections** from the list of realms, and then click **Delete**. Then click **Yes** in the Confirmation page.
6. Log out of Oracle Database Vault Administrator.

5.2.6.2 Revoking the SELECT Privilege on OE.CUSTOMERS from User SCOTT

You can use SQL*Plus to revoke the SELECT privilege on OE . CUSTOMERS from user SCOTT.

To revoke the SELECT privilege on OE.CUSTOMERS from user SCOTT:

1. In SQL*Plus, connect as user OE.

```
CONNECT OE
Enter password: password
Connected.
```

2. Revoke the SELECT privilege from user SCOTT.

```
REVOKE SELECT ON CUSTOMERS FROM SCOTT;
```

5.2.6.3 Disabling Oracle Database Vault and Oracle Label Security

You can use SQL*Plus to disable Oracle Database Vault and Oracle Label Security.

To disable Oracle Database Vault and if necessary, Oracle Label Security:

1. Connect as a user who has been granted the DV_OWNER role.

For example:

```
CONNECT dbv_owner
Enter password: password
```

2. Run the following procedure to disable Oracle Database Vault:

```
EXEC DVSYS.DBMS_MACADM.DISABLE_DV;
```

3. Connect as user SYS with the SYSDBA administrative privilege

```
CONNECT SYS AS SYSDBA
Enter password: password
```

4. Run the following procedure to disable Oracle Label Security:

```
EXEC LBACSYS.OLS_ENFORCEMENT.DISABLE_OLS;
```

When you register and enable Oracle Database Vault, Oracle Label Security is also enabled. If you choose to not disable Oracle Database Vault, then do not disable Oracle Label Security, because Database Vault uses Oracle Label Security. (This guide assumes that you are disabling Database Vault.) However, you can have Oracle Label Security enabled and Database Vault disabled.

5. Restart the database.

```
SHUTDOWN IMMEDIATE
STARTUP
```

Restricting Access with Oracle Virtual Private Database

Oracle Virtual Private Database restricts access to data based on a dynamic `WHERE` clause that is added to the SQL statements that users enter.

[About Oracle Virtual Private Database](#) (page 6-1)

Oracle Virtual Private Database (VPD) enables you to dynamically add a `WHERE` clause in any SQL statement that a user executes.

[Tutorial: Limiting Access to Data Based on the Querying User](#) (page 6-3)

In this tutorial, you create two users whose individual data access will be based on their roles.

6.1 About Oracle Virtual Private Database

Oracle Virtual Private Database (VPD) enables you to dynamically add a `WHERE` clause in any SQL statement that a user executes.

The `WHERE` clause filters the data the user is allowed to access, based on the identity of a user.

This feature restricts row and column level data access by creating a policy that enforces a `WHERE` clause for all SQL statements that query the database. The `WHERE` clause allows only users whose identity passes the security policy, and hence, have access to the data that you want to protect. You create and manage the VPD policy at the database table or view level, which means that you do not modify the applications that access the database.

In a multitenant environment, each Virtual Private Database policy applies only to the current pluggable database (PDB).

An Oracle Virtual Private Database policy has the following components, which are typically created in the schema of the security administrator:

- **A PL/SQL function to append the dynamic `WHERE` clause to SQL statements that affect the Virtual Private Database tables.** For example, a PL/SQL function translates the following `SELECT` statement:

```
SELECT * FROM ORDERS;
```

to the following:

```
SELECT * FROM ORDERS
WHERE SALES_REP_ID = 159;
```

In this example, the user can only view orders by Sales Representative 159. The PL/SQL function used to generate this `WHERE` clause is as follows:

```
CREATE OR REPLACE FUNCTION auth_orders(
  schema_var IN VARCHAR2,
```

```
table_var IN VARCHAR2
)
RETURN VARCHAR2
IS
return_val VARCHAR2 (400);
BEGIN
return_val := 'SALES_REP_ID = 159';
RETURN return_val;
END auth_orders;
/
```

In this example:

- `schema_var` and `table_var`: Create parameters to store the schema name, OE, and table name, ORDERS. (The second parameter, `table_var`, for the table, can also be used for views and synonyms.) Always create these two parameters in this order: create the parameter for the schema first, followed by the parameter for the table, view, or synonym object. Note that the function itself does not specify the OE schema or its ORDERS table. The Virtual Private Database policy you create uses these parameters to specify the OE.ORDERS table.
- `RETURN VARCHAR2`: Returns the string that will be used for the WHERE predicate clause.
- `IS ... RETURN return_val`: Encompasses the creation of the WHERE `SALES_REP_ID = 159` predicate.

You can design the WHERE clause to filter the user information based on the session information of that user, such as the user ID. To do so, you create an application context. Application contexts can be used to authenticate both database and nondatabase users. An application context is a name-value pair. For example:

```
SELECT * FROM oe.orders
WHERE sales_rep_id = SYS_CONTEXT('userenv','session_user');
```

In this example, the WHERE clause uses the SYS_CONTEXT PL/SQL function to retrieve the user session ID (`session_user`) designated by the `userenv` context. See *Oracle Database Security Guide* for detailed information about application contexts.

- **A way to attach the policy the package.** Use the `DBMS_RLS.ADD_POLICY` function to attach the policy to the package. Before you can use the `DBMS_RLS` PL/SQL package, you must be granted EXECUTE privileges on it. User SYS owns the `DBMS_RLS` package.

The advantages of enforcing row-level security at the database level rather than at the application program level are enormous. Because the security policy is implemented in the database itself, where the data to be protected is, this data is less likely to be vulnerable to attacks by different data access methods. This layer of security is present and enforced no matter how users (or intruders) try to access the data it protects. The maintenance overhead is low because you maintain the policy in one place, the database, rather than having to maintain it in the applications that connect to this database. The policies that you create provide a great deal of flexibility because you can write them for specific DML operations.

See Also:

- *Oracle Database Security Guide* for detailed information about Oracle Virtual Private Database

6.2 Tutorial: Limiting Access to Data Based on the Querying User

In this tutorial, you create two users whose individual data access will be based on their roles.

[About Limiting Access to Data Based on the Querying User](#) (page 6-3)

To limit a user's data access, you must create an Oracle Virtual Private Database (VPD) policy to define the necessary restrictions.

[Step 1: Create User Accounts for This Tutorial](#) (page 6-4)

The first step is to create accounts for the employees who must access the OE.ORDERS table.

[Step 2: If Necessary, Create the Security Administrator Account](#) (page 6-5)

The sec_admin security administrator account enables you to perform the tasks a security administrator can perform.

[Step 3: Update the Security Administrator Account](#) (page 6-5)

The sec_admin account user must have privileges to use the DBMS_RLS packages.

[Step 4: Create the F_POLICY_ORDERS Policy Function](#) (page 6-6)

The f_policy_orders policy is a PL/SQL function that defines the policy used to filter users who query the ORDERS table.

[Step 5: Create the ACCESSCONTROL_ORDERS Virtual Private Database Policy](#) (page 6-8)

Next, you can create the Virtual Private Database policy, accesscontrol_orders, and then attach it to the ORDERS table.

[Step 6: Test the ACCESSCONTROL_ORDERS Virtual Private Database Policy](#) (page 6-9)

At this stage, you can test the policy by logging on as each user and attempting to select data from the ORDERS table.

[Step 7: Optionally, Remove the Components for This Tutorial](#) (page 6-10)

You can remove the components that you created for this tutorial if you no longer need them.

6.2.1 About Limiting Access to Data Based on the Querying User

To limit a user's data access, you must create an Oracle Virtual Private Database (VPD) policy to define the necessary restrictions.

In this tutorial, you will use the ORDERS table in the Order Entry database, OE.

This table contains the following information:

| Name | Null? | Type |
|-------------|---------|-----------------------------------|
| ORDER_ID | NOTNULL | NUMBER(12) |
| ORDER_DATE | NOTNULL | TIMESTAMP(6) WITH LOCAL TIME ZONE |
| ORDER_MODE | | VARCHAR2(8) |
| CUSTOMER_ID | NOTNULL | NUMBER(6) |

| | |
|--------------|-------------|
| ORDER_STATUS | NUMBER(2) |
| ORDER_TOTAL | NUMBER(8,2) |
| SALES_REP_ID | NUMBER(6) |
| PROMOTION_ID | NUMBER(6) |

The Virtual Private Database policy that you will create is associated with a PL/SQL function. Because VPD policies are controlled by PL/SQL functions or procedures, you can design the policy to restrict access in many different ways. For this tutorial, the function you create will restrict access by the employees based on to whom they report. The function will restrict the customer access based on the customer's ID.

You may want to store VPD policies in a database account separate from the database administrator and from application accounts. In this tutorial, you will use the `sec_admin` account, which was created in [Tutorial: Creating a Secure Application Role](#) (page 3-4), to create the VPD policy. This provides better security by separating the VPD policy from the applications tables.

To restrict access based on the sensitivity of row data, you can use Oracle Label Security (OLS). OLS lets you categorize data into different levels of security, with each level determining who can access the data in that row. This way, the data access restriction is focused on the data itself, rather than on user privileges. See [Enforcing Row-Level Security with Oracle Label Security](#) (page 8-1) for more information.

6.2.2 Step 1: Create User Accounts for This Tutorial

The first step is to create accounts for the employees who must access the `OE.ORDERS` table.

To create the employee user accounts:

1. In Enterprise Manager, access the Database home page for your target database as user `SYS` with the `SYSDBA` administrative privilege.

See *Oracle Database 2 Day DBA* for more information.

2. From the **Administration** menu, select **Security**, then **Users**.
3. In the Users Page, click **Create**.
4. In the Create User page, enter the following information:
 - **Name:** LDORAN (to create the user account Louise Doran)
 - **Profile:** DEFAULT
 - **Authentication:** Password
 - **Enter Password and Confirm Password:** Enter a password that meets the requirements in [Requirements for Creating Passwords](#) (page 2-17).
 - **Default Tablespace:** USERS
 - **Temporary Tablespace:** TEMP
 - **Status:** Unlocked
5. Select the **Object Privileges** tab.
6. From the **Select Object Type** list, select **Table**, and then click **Add**.

- In the Add Table Object Privileges page, in the **Select Table Objects** field, enter the following text:

```
OE.ORDERS
```

Do not include spaces in this text.

- In the Available Privileges list, select **SELECT**, and then click **Move** to move it to the Selected Privileges list. Click **OK**.

The Create User page appears, with **SELECT** privileges for **OE.ORDERS** listed.

- Click **OK**.

The Users page appears, with user **ldoran** is listed in the User Name column.

- Select the selection button for user **LDORAN**, and from the **Actions** list, select **Create Like**. Then, click **Go**.

- In the Create User page, enter the following information:

- Name:** **LPOPP** (to create the user account for Finance Manager Luis Popp.)
- Enter Password** and **Confirm Password:** Enter a password that meets the requirements in [Requirements for Creating Passwords](#) (page 2-17).

- Click **OK**.

Both employee accounts have been created, and they have identical privileges. If you check the privileges for user **LPOPP**, you will see that they are identical to those of user **LDORAN**'s. At this stage, if either of these users performs a **SELECT** statement on the **OE.ORDERS** table, he or she will be able to see all of its data.

6.2.3 Step 2: If Necessary, Create the Security Administrator Account

The **sec_admin** security administrator account enables you to perform the tasks a security administrator can perform.

In [Tutorial: Creating a Secure Application Role](#) (page 3-4), you created the **sec_admin** for that tutorial. You can use that account for this tutorial.

If you have not yet created this account, then follow the steps in [Step 2: Create a Security Administrator Account](#) (page 3-6) to create **sec_admin**.

6.2.4 Step 3: Update the Security Administrator Account

The **sec_admin** account user must have privileges to use the **DBMS_RLS** packages.

User **SYS** owns this package, so you must log on as **SYS** to grant these package privileges to **sec_admin**. The user **sec_admin** also must have **SELECT** privileges on the **CUSTOMERS** table in the **OE** schema and the **EMPLOYEES** table in the **HR** schema.

To grant **sec_admin** privileges to use the **DBMS_RLS** package:

- In Enterprise Manager, access the Database home page and ensure that you are logged in as user **SYS** with the **SYSDBA** role selected.

See *Oracle Database 2 Day DBA* for more information.

- From the **Schema** menu, then **Users**.

3. In the Users Page, select the **SEC_ADMIN** user, and in the View User page, click **Edit**.
4. In the Edit User page, click **Object Privileges**.
5. From the **Select Object Type** list, select **Package**, and then click **Add**.
6. In the Add Package Object Privileges page, under Select Package Objects, enter `SYS.DBMS_RLS` so that `sec_admin` will have access to the `DBMS_RLS` package.
7. Under Available Privileges, select **EXECUTE**, and then click **Move** to move it to the Selected Privileges list.
8. Click **OK**.
9. In the Edit User page, from the **Select Object Type** list, select **Table**, and then click **Add**.
10. In the Add Table Object Privileges page, in the **Select Table Objects** field, enter `HR.EMPLOYEES` so that `sec_admin` will have access to the `HR.EMPLOYEES` table.
11. Under Available Privileges, select **SELECT**, and then click **Move** to move it to the Selected Privileges list.
12. Click **OK**.

The Edit User page appears. It shows that user `sec_admin` has object privileges for the `HR.EMPLOYEES` table and `DBMS_RLS PL/SQL` package. Ensure that you do not select the grant option for either of these objects.

13. Click **Apply**.

All the changes you have made, in this case, the addition of the two object privileges, are applied to the `sec_admin` user account.

6.2.5 Step 4: Create the **F_POLICY_ORDERS** Policy Function

The `f_policy_orders` policy is a PL/SQL function that defines the policy used to filter users who query the `ORDERS` table.

To filter the users, the policy function uses the `SYS_CONTEXT` PL/SQL function to retrieve session information about users who are logging in to the database.

To create the application context and its package:

1. Select **Logout** to log out of the database instance.
2. In the Confirmation dialog box, select **Logout of (Database Instance)** and then select the **Display login page after logout** check box. Then click **Logout**.
3. Log in as user `sec_admin` using the `NORMAL` role.
4. From the **Schema** menu, select **Programs**, then **Functions**.
5. In the Database Login page, log in as user `sec_admin` with the `NORMAL` role selected.
6. From the **Schema** menu, select **Programs**, and then **Functions**.

7. In the Functions page, ensure that the **Object Type** menu is set to **Function**, and then click **Create**.
8. In the Create Function page, enter the following information:
 - **Name:** F_POLICY_ORDERS
 - **Schema:** SEC_ADMIN
 - **Source:** Delete the empty function code that has been provided, and then enter the following code (but not the line numbers on the left side of the code) to create a function that checks whether the user who has logged on is a sales representative.

The `f_policy_orders` function uses the `SYS_CONTEXT` PL/SQL function to get the session information of the user. It then compares this information with the job ID of that user in the `HR.EMPLOYEES` table, for which `sec_admin` has `SELECT` privileges.

```
(schema in varchar2,
tab in varchar2)
return varchar2
as
v_job_id  varchar2(20);
v_user    varchar2(100);
predicate varchar2(400);

begin
  v_job_id := null;
  v_user   := null;
  predicate := '1=2';

v_user := lower(sys_context('userenv','session_user'));

  select lower(job_id) into v_job_id from hr.employees
     where lower(email) = v_user;

  if v_job_id='sa_rep' then
    predicate := '1=1';
  else
    null;
  end if;

  return predicate;

exception
  when no_data_found then
    null;
end;
```

In this specification:

- `(schema in varchar2, tab in varchar2)`: Defines parameters for the schema (`schema`) and table (`tab`) that must be protected. Notice that the function does not mention the `OE.ORDERS` table. The `ACCESSCONTROL_ORDERS` policy that you create in [Step 5: Create the ACCESSCONTROL_ORDERS Virtual Private Database Policy](#) (page 6-8) uses these parameters to specify the `OE` schema and `ORDERS` table. Ensure that you create the `schema` parameter first, followed by the `tab` parameter.

- `return varchar2`: Returns the string that will be used for the `WHERE` predicate clause. Always use `VARCHAR2` as the data type for this return value.
- `as ... predicate`: Defines variables to store the job ID, user name of the user who has logged on, and predicate values.
- `begin ... return predicate`: Encompasses the creation of the `WHERE` predicate, starting the with the `BEGIN` clause for the `v_job_id` and `v_user` settings.
- `v_job_id varchar2(20) and v_user varchar2(100)`: Sets the `v_job_id` and `v_user` variables to null, and the `predicate` variable to `1=2`, that is, to a false value. At this stage, no `WHERE` predicate can be generated until these variables pass the tests starting with `select lower(job_id) into v_job_id`.
- `v_user := lower(sys_context ...)`: Uses the `SYS_CONTEXT` function to retrieve the session information of the user and write it to the `v_user` variable.
- `select lower(job_id) into v_job_id ... end if`: Checks if the user is a sales representative by comparing the job ID with the user who has logged on. If the job ID of the user who has logged on is `sa_rep` (sales representative), then the `predicate` variable is set to `1=1`. In other words, the user, by being a sales representative, has passed the test.
- `return predicate`: Returns the `WHERE` predicate, which translates to `WHERE role_of_user_logging_on IS "sa_rep"`. Oracle Database appends this `WHERE` predicate onto any `SELECT` statement that users `LDORAN` and `LPOPP` issue on the `OE.ORDERS` table.
- `exception ... null`: Provide an `EXCEPTION` clause for cases where a user without the correct privileges has logged on.

9. Click **OK**.

6.2.6 Step 5: Create the `ACCESSCONTROL_ORDERS` Virtual Private Database Policy

Next, you can create the Virtual Private Database policy, `accesscontrol_orders`, and then attach it to the `ORDERS` table.

To increase performance, add the `CONTEXT_SENSITIVE` parameter to the policy, so that Oracle Database only executes the `f_policy_orders` function when the content of the application context changes, in this case, when a new user logs on. Oracle Database only activates the policy when a user performs a SQL `SELECT` statement on the `ORDERS` table. Hence, the user cannot run the `INSERT`, `UPDATE`, and `DELETE` statements, because the policy does not allow him or her to do so.

To create the `ACCESSCONTROL_ORDERS` Virtual Private Database policy:

1. From the **Security** menu, select **Virtual Private Database Policies**.
2. In the Virtual Private Database Policies page, click **Create**.
3. In the Create Policy page, under **General**, enter the following:
 - **Policy Name:** `ACCESSCONTROL_ORDERS`

- **Object Name:** OE . ORDERS
- **Policy Type:** Select **CONTEXT_SENSITIVE**.
This type reevaluates the policy function at statement run-time if it detects context changes since the last use of the cursor. For session pooling, where multiple clients share a database session, the middle tier must reset the context during client switches. Note that Oracle Database does not cache the value that the function returns for this policy type; it always runs the policy function during statement parsing. The **CONTEXT_SENSITIVE** policy type applies to only one object.
To enable the Policy Type, select the **Enabled** box.
- 4. Under Policy Function, enter the following:
 - **Policy Function:** Enter the name of the function that generates a predicate for the policy, in this case, SEC_ADMIN . F_POLICY_ORDERS.
 - **Long Predicate:** Do not select this box.
Typically, you select this box to return a predicate with a length of up to 32K bytes. By not selecting this box, Oracle Database limits the predicate to 4000 bytes.
- 5. Under Enforcement, select the **SELECT** option and deselect the remaining options that already may be selected.
- 6. Do not select any options under Security Relevant Columns.
- 7. Click **OK**.
The Virtual Private Database Policies page appears, with the ACCESSCONTROL_ORDERS policy listed in the list of policies.
- 8. Do not log out of Enterprise Manager.

6.2.7 Step 6: Test the ACCESSCONTROL_ORDERS Virtual Private Database Policy

At this stage, you can test the policy by logging on as each user and attempting to select data from the ORDERS table.

To test the ACCESSCONTROL_ORDERS policy:

1. Start SQL*Plus.

From a command prompt, enter the following command to start SQL*Plus, and log in as Sales Representative Louise Doran, whose user name is ldoran:

```
sqlplus ldoran
Enter password: password
```

SQL*Plus starts, connects to the default database, and then displays a prompt.

For detailed information about starting SQL*Plus, see *Oracle Database 2 Day DBA*.

2. Enter the following **SELECT** statement:

```
SELECT COUNT(*) FROM OE.ORDERS;
```

The following results should appear for Louise. As you can see, Louise is able to access all the orders in the OE . ORDERS table.

```
COUNT(*)
-----
      105
```

3. Connect as Finance Manager Luis Popp.

```
CONNECT lpopp
Enter password: password
```

4. Enter the following SELECT statement:

```
SELECT COUNT(*) FROM OE.ORDERS;
```

The following result should appear, because Mr. Popp, who is not a sales representative, does not have access to the data in the OE.ORDERS table. Because Mr. Popp does not have access, Oracle Database only allows him access to 0 rows.

```
COUNT(*)
-----
      0
```

5. Exit SQL*Plus:

```
EXIT
```

6.2.8 Step 7: Optionally, Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

[Removing the Data Structures Created by sec_admin](#) (page 6-10)

You can use Enterprise Manager to remove the data structures that user `sec_admin` created.

[Removing the User Accounts](#) (page 6-11)

You can use Enterprise Manager to remove the user accounts.

[Revoking Privileges on DBMS_RLS from User sec_admin](#) (page 6-11)

You can use Enterprise Manager to revoke the EXECUTE privilege on the DBMS_RLS package from user `sec_admin`.

6.2.8.1 Removing the Data Structures Created by sec_admin

You can use Enterprise Manager to remove the data structures that user `sec_admin` created.

To remove the data structures created by sec_admin:

1. In Enterprise Manager, ensure that you are logged in as user `sec_admin`.
2. From the **Security** menu, select **Virtual Private Database Policies**.
3. In the Virtual Private Database Policies page, under Search, enter the following information, and then click **Go**:

- **Schema Name:** OE
- **Object Name:** ORDERS
- **Policy Name:** %

The policy you created, `ACCESSCONTROL_ORDERS`, is listed.

4. Select **ACCESSCONTROL_ORDERS**, and then click **Delete**.
5. In the Confirmation page, click **Yes**.
6. From the **Schema** menu, select **Programs**, then **Functions**.
7. If the **F_POLICY_ORDERS** function is not listed, then use the **Search** field to search for it.
8. Select the selection button for the **F_POLICY_ORDERS** function and then click **Delete**.
9. In the Confirmation window, click **OK**.

6.2.8.2 Removing the User Accounts

You can use Enterprise Manager to remove the user accounts.

To remove the user accounts:

1. From Enterprise Manager, select **Logout** to log out of the database instance.
2. Log in as user **SYSTEM** with the **NORMAL** role selected.
3. In the Database home page, from the **Schema** menu, select **Users**.
4. In the Users page, select each of the following users, and then click **Delete** to remove them:
 - **LDORAN**
 - **LPOPP**

Do not remove `sec_admin` because you will need this account for later tutorials in this guide.

6.2.8.3 Revoking Privileges on DBMS_RLS from User sec_admin

You can use Enterprise Manager to revoke the **EXECUTE** privilege on the **DBMS_RLS** package from user `sec_admin`.

To revoke the EXECUTE privilege on the DBMS_RLS package from user sec_admin:

1. From Enterprise Manager, select **Logout** to log out of the database instance.
2. Log in as the **SYS** administrative user with the **SYSDBA** role selected.
3. From the **Schema** menu, select **Users**.
4. In the Users page, select user **SEC_ADMIN** and then click **Edit**.
5. Select the **Object Privileges** tab.
6. From the list of object privileges, select the listing for the **SELECT** privilege for the `HR.EMPLOYEES` table.
7. Click **Delete**.
8. From the list of object privileges, select the listing for the **EXECUTE** privilege for the **DBMS_RLS** package.

9. Click **Delete**.
10. Click **Apply**.
11. Exit Enterprise Manager.

Limiting Access to Sensitive Data Using Oracle Data Redaction

Oracle Data Redaction limits access to sensitive data by redacting this data in real time.

[About Oracle Data Redaction](#) (page 7-1)

Oracle Data Redaction enables you to redact (mask) column data.

[Tutorial: Redacting Data for a Select Group of Users](#) (page 7-2)

In this tutorial, you create an Oracle Data Redaction policy that redacts data based on the user who has logged in.

7.1 About Oracle Data Redaction

Oracle Data Redaction enables you to redact (mask) column data.

You can redact data using one of the following methods:

- **Full redaction.** You redact all the contents of the column data. The redacted value returned to the querying user depends on the data type of the column. For example, columns of the NUMBER data type are redacted with a zero (0) and character data types are redacted with a blank space.
- **Partial redaction.** You redact a portion of the column data. For example, you can redact most of a credit card number with asterisks (*), except for the last four digits.
- **Regular expressions.** You can use regular expressions in both full and partial redaction. This enables you to redact data based on a search pattern for the data. For example, you can use regular expressions to redact specific phone numbers or email addresses in your data.
- **Redaction using NULL values.** This feature enables you to use the `DBMS_REDACT.NULLIFY` function to hide all of the sensitive data in a table or view column and replace it with null values. You can set this function by using the `function_type` parameter of the `DBMS_REDACT.ADD_POLICY` or `DBMS_REDACT.ALTER_POLICY` procedure.
- **Random redaction.** The redacted data presented to the querying user appears as randomly-generated values each time it is displayed.
- **No redaction.** This option enables you to test the internal operation of your redaction policies, with no effect on the results of queries against tables with policies defined on them. You can use this option to test the redaction policy definitions before applying them to a production environment.

- **Central management of named Data Redaction expressions.** This feature enables you to create a library of named policy expressions that can be used in the columns of multiple tables and views. By having named policy expressions, you can centrally manage all of the policy expressions within a database.

Data Redaction performs the redaction at run time, that is, the moment that the user tries to view the data. This functionality is ideally suited for dynamic production systems in which data constantly changes. While the data is being redacted, Oracle Database can process all data normally and preserve the back-end referential integrity constraints. Data redaction can help you to comply with industry regulations such as Payment Card Industry Data Security Standard (PCI DSS) and the Sarbanes-Oxley Act.

See Also: *Oracle Database Advanced Security Guide* for detailed information about Oracle Data Redaction

7.2 Tutorial: Redacting Data for a Select Group of Users

In this tutorial, you create an Oracle Data Redaction policy that redacts data based on the user who has logged in.

[About Redacting Data for a Select Group of Users](#) (page 7-2)

The scenario for this tutorial is a sales office in which a sales manager must see all data in a table.

[Step 1: Create User Accounts and Grant Them the Necessary Privileges](#) (page 7-3)

First, you must create and grant privileges to the necessary users and roles.

[Step 2: Create and Populate the SALES_OPPTS Sales Opportunities Table](#) (page 7-5)

The `sales_oppts` table contains information for small businesses that are sales opportunities.

[Step 3: Create the SALES_OPPTS_POL Oracle Data Redaction Policy](#) (page 7-6)

As user `sec_admin`, create the `sales_oppts_pol` Oracle Data Redaction policy.

[Step 5: Test the SALES_OPPTS_POL Oracle Data Redaction Policy](#) (page 7-7)

Next, you are ready to test the policy.

[Step 6: Optionally, Remove the Components for This Tutorial](#) (page 7-9)

You can remove the components that you created for this tutorial if you no longer need them.

7.2.1 About Redacting Data for a Select Group of Users

The scenario for this tutorial is a sales office in which a sales manager must see all data in a table.

The sales manager, user `ezlotkey`, must have access to `sales_oppts` table, which describes various sales opportunities. However, the sales representatives under this sales manager, users `ahutton` and `eabel`, must have limited access to the `sales_oppts` table columns that describe account names and closing dates.

To solve this problem, you will create an Oracle Data Redaction policy that accomplishes these needs by performing the following actions:

- The policy redacts data in two columns, each using a different redaction style.
- The effect of the policy is the display of the query result with either the actual data or the redacted data based on the enabled roles of the querying user.

For this tutorial, you will interact directly with the database by using database user accounts. This is for simplicity. The intended use scenarios for Oracle Data Redaction are twofold: redact database applications and redact direct database access. Oracle Data Redaction on its own is a good solution for redacting sensitive data from packaged and custom applications. After completing the tutorial, you will have the knowledge necessary to apply what you have learned (using the database user accounts) for scenarios that involve actual application users. When direct database access is the target scenario, you should couple Oracle Data Redaction with preventive and detective controls that provide security for privileged database users (for example, Oracle Database Vault, Oracle Label Security, Oracle Audit Vault, and Oracle Database Firewall).

7.2.2 Step 1: Create User Accounts and Grant Them the Necessary Privileges

First, you must create and grant privileges to the necessary users and roles.

The user `sec_admin` must have the `EXECUTE` privilege on the `DBMS_REDACT` PL/SQL package, which is required to create Oracle Data Redaction policies.

To create user accounts for this tutorial:

1. In Enterprise Manager, access the Database home page for your target database as user `SYS` with the `SYSDBA` administrative privilege.
See *Oracle Database 2 Day DBA* for more information.
2. From the **Security** menu, select **Roles**.
3. In the Roles page, select **Create**.
4. In the Create Role page Name field, enter `SUPERVISOR` and then click **OK**.
5. From the **Schema** menu, select **Users**.
6. In the Users Page, click **Create**.
7. In the Create User page, enter the following information:
 - **Name:** `EZLOTKEY` (to create the user account for Eleni Zlotkey)
 - **Profile:** `DEFAULT`
 - **Authentication:** `Password`
 - **Enter Password and Confirm Password:** Enter a password that meets the requirements in [Requirements for Creating Passwords](#) (page 2-17).
 - **Default Tablespace:** `USERS`
 - **Temporary Tablespace:** `TEMP`
 - **Status:** `Unlocked`
8. Select the **System Privileges** tab.

9. Select the **Edit List** button.
 10. In the **Modify System Privileges** list, select the following privileges and then move them to the **Selected System Privileges** list.
 - **CREATE SESSION**
 - **CREATE TABLE**
 - **UNLIMITED TABLESPACE**
 11. Click **OK**.
 12. In the Create User page, select the **Roles** tab, and then select the **Edit List** button.
 13. In the Modify Roles page, double-click the **SUPERVISOR** role in the **Available Roles** list to move it to the **Selected Roles** list.
 14. Click **OK**, and then click **OK** again to return to the Users page.
 15. Select the **Create** button.
 16. In the Create User page, enter the following information:
 - **Name:** EABEL (to create the user account for Ellen Abel)
 - **Profile:** DEFAULT
 - **Authentication:** Password
 - **Enter Password and Confirm Password:** Enter a password that meets the requirements in [Requirements for Creating Passwords](#) (page 2-17).
 - **Default Tablespace:** USERS
 - **Temporary Tablespace:** TEMP
 - **Status:** Unlocked
 17. Select the **System Privileges** tab.
 18. Select the **Edit List** button.
 19. In the **Modify System Privileges** list, double-click the **CREATE SESSION** system privilege to move it to Selected System Privileges list.
 20. Click **OK**.
 21. In the Users page, select the **EABEL** user.
 22. From the **Actions** list, select the **Create Like** button and then click **Go**.
 23. In the Create User page, enter the following information:
 - **Name:** Enter **AHUTTON** (for user Alyssa Hutton).
 - **Enter Password and Confirm Password:** Enter a password that meets the requirements in [Requirements for Creating Passwords](#) (page 2-17).
- Note that user `ezlotkey` has been granted a role, `supervisor`, but the users `eabel` and `ahutton` are not granted any roles.

24. Click **OK**.
25. In the Users page, select the **SEC_ADMIN** user and then click the **Edit** button.
If user **SEC_ADMIN** does not exist, then you can quickly create this user in SQL*Plus by entering the following statement:

```
GRANT CREATE PROCEDURE, CREATE ROLE, CREATE SESSION, INHERIT ANY PRIVILEGES,  
SELECT ANY DICTIONARY TO sec_admin IDENTIFIED BY password;
```


See also [Step 2: Create a Security Administrator Account](#) (page 3-6) to create this account.
26. Select the **Object Privileges** tab.
27. In the Object Privileges tab, from the **Select Object Type** list, select **Package**, and then click **Add**.
28. In the Select Package Objects field, enter `SYS.DBMS_REDACT`, and then in the Available Privileges list, move the **EXECUTE** privilege to the Selected Privileges list.
29. Click **OK**, and then click **Apply**.

7.2.3 Step 2: Create and Populate the SALES_OPPS Sales Opportunities Table

The `sales_opps` table contains information for small businesses that are sales opportunities.

This table contains two columns that you must later on and on which you must create an Oracle Data Redaction policy.

To create and populate the sales_opps sales opportunities table:

1. Log into the database instance as user `ezlotkey`.

```
sqlplus ezlotkey  
Enter password: password
```

2. Create the `sales_opps` table.

```
CREATE TABLE sales_opps (  
  account varchar2(30),  
  region varchar2(20),  
  product varchar2(20),  
  sales_rep varchar2(15),  
  close_date date,  
  price number,  
  quantity number);
```

3. Populate the `sales_opps` table with some data.

```
INSERT INTO sales_opps VALUES ('Rising Dough Bakery', 'north-east', 'AAL  
AccountPro', 'ahutton', '07-JUL-12', 400.00, 4);  
INSERT INTO sales_opps VALUES ('Shear Madness Hair Salon', 'south-west', 'AAL  
AccountPro', 'eabel', '20-APR-12', 400.00, 1);  
INSERT INTO sales_opps VALUES ('Doublecheck Accounting', 'north-east', 'AAL  
AccountPro', 'ahutton', '14-MAR-12', 400.00, 12);  
INSERT INTO sales_opps VALUES ('State of Art Framing', 'south-west', 'AAL  
TaxPro', 'eabel', '21-MAY-12', 300.00, 2);
```

```
INSERT INTO sales_opps VALUES ('Shady Trees Arborists', 'north-east', 'AA1
AccountPro', 'ahutton', '17-JUN-12', 400.00, 16);
```

4. Query the `account` and `close_date` columns of the `sales_opps` table, to see the data that must be redacted.

```
SELECT account, close_date, product, quantity FROM sales_opps;
```

| ACCOUNT | CLOSE_DAT | PRODUCT | QUANTITY |
|--------------------------|-----------|----------------|----------|
| Rising Dough Bakery | 07-JUL-12 | AA1 AccountPro | 4 |
| Shear Madness Hair Salon | 20-APR-12 | AA1 AccountPro | 1 |
| Doublecheck Accounting | 14-MAR-12 | AA1 AccountPro | 12 |
| State of Art Framing | 21-MAY-12 | AA1 TaxPro | 2 |
| Shady Trees Arborists | 17-JUN-12 | AA1 AccountPro | 16 |

7.2.4 Step 3: Create the SALES_OPPTS_POL Oracle Data Redaction Policy

As user `sec_admin`, create the `sales_opps_pol` Oracle Data Redaction policy.

To create the `sales_opps_pol` Data Redaction policy:

1. In Enterprise Manager, log out, and then log back in again as user `sec_admin`. See *Oracle Database 2 Day DBA* for more information.
2. From the **Security** menu, select **Oracle Data Redaction**.
3. In the Data Redaction page, select the **Policies** tab.
4. Select the **Create** button.
5. In the Create Data Redaction Policy page, enter the following information to design the basics of the policy:

- **Schema:** Enter `EZLOTKEY` (in capital letters).
- **Table/View:** Enter `SALES_OPPTS` (in capital letters).
- **Policy Name:** Enter `SALES_OPPTS_POL`.
- **Policy Expression:** Enter the following expression:

```
SYS_CONTEXT('SYS_SESSION_ROLES', 'SUPERVISOR') = 'FALSE'
```

The expression translates to "Redact the data in the `account` column for any user who does not have the `supervisor` role enabled." In other words, only the supervisor, `ezlotkey`, will be able to see the data in the `account` column.

6. Still in the Data Redaction page, apply the `sales_opps_policy` policy to a column, as follows:
 - a. Select the **Add** button.
 - b. In the Add window, from the **Column** list, select **ACCOUNT**.
The Column Datatype field displays the data type of `ACCOUNT`, which is `VARCHAR2`.

- c. From the **Redaction Template** list, select **Custom** (which should be the default selection.)
 - d. From the **Redaction Function** list, select **FULL**.
FULL means all the characters in the `account` column will be redacted. Because the data type of the `account` column is `VARCHAR2`, the data will appear as a blank space.
 - e. Click **OK**.
7. Apply the `sales_opps_pol` policy to a second column, as follows:
- a. Select the **Add** button.
 - b. In the Add window, from the **Column** list, select **CLOSE_DATE**.
The **Column Datatype** field determines that the data type for `CLOSE_DATE` is `DATE`.
 - c. From the **Redaction Function** list, select **PARTIAL**.
In a moment, the Add window expands to show the **Function Attributes** area.
 - d. In the **Date Redaction Format** field, enter the following attribute:

```
m06d7YHMS
```

This setting redacts the month and day of each date, setting them to appear as `07-JUNE-12`.
 - e. Click **OK**.
The Create Data Redaction Policy page should appear as follows:

Create Data Redaction Policy: SALES_OPPTS_POL

* Schema: EZLOTKEY

* Table/View: SALES_OPPTS

* Policy Name: SALES_OPPTS_POL

* Policy Expression: SYS_CONTEXT('SYS_SESSION_ROLES','SUPERVISOR') = 'FALSE'

Object Columns

| Column | Column Datatype | Redaction Function | Function Attributes |
|------------|-----------------|--------------------|---------------------|
| ACCOUNT | VARCHAR2 | FULL | |
| CLOSE_DATE | DATE | PARTIAL | m06d7YHMS |

Instructions

Policy expression can be based on SYS_CONTEXT values from a specific namespace such as USERENV and SYS_SESSION_ROLES. Also boolean expressions that always evaluates to TRUE/FALSE can be specified. Example:
- SYS_CONTEXT (USERENV, SESSION_USER) = 'APP_USER'
+ SYS_CONTEXT ('SYS_SESSION_ROLES','SUPERVISOR') = 'FALSE'
- '1=1' (Default Expression which evaluates to TRUE)

8. In the Create Data Redaction Policy page, click **OK**.
The policy appears in the Data Redaction Policies list.
Do not exit Enterprise Manager.

7.2.5 Step 5: Test the SALES_OPPTS_POL Oracle Data Redaction Policy

Next, you are ready to test the policy.

To test the `sales_opps_pol` Oracle Data Redaction policy, log in as the users you created earlier and query the redacted columns in the `sales_opps` table.

To test the `sales_opps_pol` policy:

1. Connect to SQL*Plus as user `ezlotkey`.

```
connect ezlotkey
Enter password: password
```

2. Grant the sales representatives the `SELECT` privilege for the `sales_opps` table.

```
GRANT SELECT ON sales_opps TO eabel;
GRANT SELECT ON sales_opps TO ahutton;
```

3. Connect as user `eabel`.

```
connect eabel
Enter password: password
```

4. Query the `sales_opps` tables as follows:

```
SELECT account, close_date, product, quantity FROM ezlotkey.sales_opps;
```

Output similar to the following should appear:

| ACCOUNT | CLOSE_DAT | PRODUCT | QUANTITY |
|---------|-----------|----------------|----------|
| | 07-JUN-12 | AA1 AccountPro | 4 |
| | 07-JUN-12 | AA1 AccountPro | 1 |
| | 07-JUN-12 | AA1 AccountPro | 12 |
| | 07-JUN-12 | AA1 TaxPro | 2 |
| | 07-JUN-12 | AA1 AccountPro | 16 |

No data appears in the `account` column because it has been redacted to display a blank space for each row. The `close_date` column shows dates, but they are all partial date values. The `product` and `quantity` columns show their data, as expected, because the Data Redaction policy does not apply to them.

5. Connect as user `ahutton` and perform the same query.

```
connect ahutton
Enter password: password
```

```
SELECT account, close_date, product, quantity FROM ezlotkey.sales_opps;
```

| ACCOUNT | CLOSE_DAT | PRODUCT | QUANTITY |
|---------|-----------|----------------|----------|
| | 07-JUN-12 | AA1 AccountPro | 4 |
| | 07-JUN-12 | AA1 AccountPro | 1 |
| | 07-JUN-12 | AA1 AccountPro | 12 |
| | 07-JUN-12 | AA1 TaxPro | 2 |
| | 07-JUN-12 | AA1 AccountPro | 16 |

The data is redacted for user `ahutton` as well.

6. Connect as user `ezlotkey` and perform the same query.

```
connect ezlotkey
Enter password: password
```

```
SELECT account, close_date, product, quantity FROM sales_opps;
```

| ACCOUNT | CLOSE_DAT | PRODUCT | QUANTITY |
|--------------------------|-----------|----------------|----------|
| Rising Dough Bakery | 07-JUL-12 | AA1 AccountPro | 4 |
| Shear Madness Hair Salon | 20-APR-12 | AA1 AccountPro | 1 |
| Doublecheck Accounting | 14-MAR-12 | AA1 AccountPro | 12 |
| State of Art Framing | 21-MAY-12 | AA1 TaxPro | 2 |
| Shady Trees Arborists | 17-JUN-12 | AA1 AccountPro | 16 |

The `sales_opps_pol` Data Redaction policy shows the actual data for user `ezlotkey` because she has the `supervisor` role enabled. However, if this role is disabled for `ezlotkey`, then when she queries this table, the `account` and `close_date` columns will be redacted, even though she created and owns the `sales_opps` table.

7. Log out of SQL*Plus.

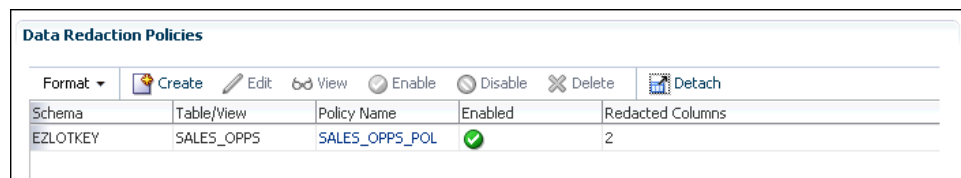
EXIT

7.2.6 Step 6: Optionally, Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

To remove the components for this tutorial:

1. In Enterprise Manager, ensure that you are logged in as user `sec_admin`.
2. In the Data Redaction Policies page, select the `SALES_OPPTS_POL` policy.



3. Click the **Delete** button.
4. In the Confirmation page, click **Yes**.
5. Log out of Enterprise Manager and then log back in again as user `SYS` with the `SYSDBA` administrative privilege
6. From the Database home page, select **Schema**, then **Users**.
7. Select user `AHUTTON`, click **Delete**, and then in the Confirmation window, select **Yes**.
8. Select user `EABEL`, click **Delete**, and then in the Confirmation window, select **Yes**.
9. Select user `EZLOTKEY`, click **Delete**, and then in the Confirmation window, select **Yes**.
10. From the **Security** menu, select **Roles**.
11. Select the role `SUPERVISOR`, click **Delete**, and in the Confirmation window, select **Yes**.
12. In the **Users** page, select the `SEC_ADMIN` user and then click **Edit**.

13. In the Edit User: SEC_ADMIN page, select the **Object Privileges** tab.
14. Select the **EXECUTE** privilege for the **DBMS_REDACT** package, and then click **Delete**.
15. Click the **Apply** button.
16. Exit Enterprise Manager by clicking the **Log Out** button.

Enforcing Row-Level Security with Oracle Label Security

Oracle Label Security enables you to enforce row-level security.

[About Oracle Label Security](#) (page 8-1)

Oracle Label Security (OLS) provides row-level security for your database tables.

[Virtual Private Database, Oracle Label Security, and Data Redaction Differences](#) (page 8-2)

Oracle Virtual Private Database, Oracle Label Security, and Oracle Data Redaction restrict the data that different users can see in database tables.

[Guidelines for Planning an Oracle Label Security Policy](#) (page 8-4)

Before you create an Oracle Label Security policy, determine how to apply the labels to the application schema.

[Tutorial: Creating Levels of Access to Table Data Based on the User](#) (page 8-5)

You can create different levels of Oracle Label Security access to table data based on who the user is.

8.1 About Oracle Label Security

Oracle Label Security (OLS) provides row-level security for your database tables.

You can accomplish this by assigning one or more security labels that define the level of security you want for the data rows of the table.

This feature secures your database tables at the row level, and assigns these rows different levels of security based on security labels. You then create a security authorization for users based on the OLS labels.

For example, rows that contain highly sensitive data can be assigned a label entitled `HIGHLY SENSITIVE`; rows that are less sensitive can be labeled as `SENSITIVE`, and so on. Rows that all users can have access to can be labeled `PUBLIC`. You can create as many labels as you need, to fit your site's security requirements. In a multitenant environment, the labels apply to the local pluggable database (PDB) and the session labels apply to local users.

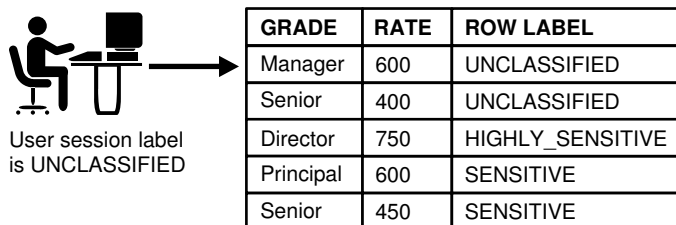
After you create and assign the labels, you can use Oracle Label Security to assign specific users authorization for specific rows, based on these labels. Afterward, Oracle Label Security automatically compares the label of the data row with the security clearance of the user to determine whether the user is allowed access to the data in the row.

An Oracle Label Security policy has the following components:

- **Labels.** Labels for data and users, along with authorizations for users and program units, govern access to specified protected objects. Labels are composed of the following:
 - **Levels.** Levels indicate the type of sensitivity that you want to assign to the row, for example, `SENSITIVE` or `HIGHLY SENSITIVE`.
 - **Compartments.** (Optional) Data can have the same level (Public, Confidential and Secret), but can belong to different projects inside a company, for example ACME Merger and IT Security. Compartments represent the projects in this example, that help define more precise access controls. They are most often used in government environments.
 - **Groups.** (Optional) Groups identify organizations owning or accessing the data, for example, UK, US, Asia, Europe. Groups are used both in commercial and government environments, and frequently used in place of compartments due to their flexibility.
- **Policy.** A policy is a name associated with these labels, rules, and authorizations.

You can create Oracle Label Security labels and policies in Enterprise Manager, or you can create them using the `SA_SYSDBA`, `SA_COMPONENTS`, and `SA_LABEL_ADMIN` PL/SQL packages. This guide explains how to create Oracle Label Security labels and policies by using Enterprise Manager.

For example, assume that a user has the `SELECT` privilege on an application table. As illustrated in the following figure, when the user runs a `SELECT` statement, Oracle Label Security evaluates each row selected to determine whether the user can access it. The decision is based on the privileges and access labels assigned to the user by the security administrator. You can also configure Oracle Label Security to perform security checks on `UPDATE`, `DELETE`, and `INSERT` statements.



See Also: *Oracle Label Security Administrator's Guide* for detailed information about Oracle Label Security

8.2 Virtual Private Database, Oracle Label Security, and Data Redaction Differences

Oracle Virtual Private Database, Oracle Label Security, and Oracle Data Redaction restrict the data that different users can see in database tables.

But which of the features should you use? Virtual Private Database is effective when there is existing data you can use to determine the access requirements. For example, you can configure a sales representative to see only the rows and columns in a customer order entry table for orders he or she handles. Oracle Label Security is useful if you have no natural data (such as user accounts or employee IDs) that can be used to indicate a table's access requirements. To determine this type of user access, you assign different levels of sensitivity to the table rows. Oracle Data Redaction enables

you to select from three differing (redaction) styles, and it applies the redaction when the user accesses the data, not directly in the database table.

In some cases, Oracle Virtual Private Database and Oracle Label Security can complement each other. The following Oracle Technology Network hands-on tutorial demonstrates how a Virtual Private Database policy can compare an Oracle Label Security user clearance with a minimum clearance. When the user clearance dominates the threshold, the salary column is not hidden.

<http://www.oracle.com/technetwork/database/security/ols-cs1-099558.html>

Table 8-1 (page 8-3) compares the features of Oracle Virtual Private Database, Oracle Label Security, and Oracle Data Redaction.

Table 8-1 Comparing Virtual Private Database, Label Security, and Data Redaction

| Feature | VPD | OLS | Data Redaction |
|---|-----------------|------------------|----------------|
| Provides full masking, partial masking, and random masking | No | No | Yes |
| Redacts data in real-time, as the user is accessing it | No | No | Yes |
| Provides row-level security | Yes | Yes | No |
| Provides column-level security (column masking) | Yes | No | Yes |
| Binds a user-defined PL/SQL package to a table, view, or synonym | Yes | No ¹ | No |
| Modifies SQL by dynamically adding a WHERE clause returned from the PL/SQL procedures | Yes | No | No |
| Restricts database operations by privileged users ² | No | No | No |
| Controls access to a set of rows based on the sensitivity label of the row and the security level of the user | No | Yes | No |
| Adds a column (optionally hidden) designed to store sensitivity labels for rows in the protected table ³ | No | Yes | No |
| Provides a user account to manage its administration | No ⁴ | Yes ⁵ | No |
| Provides pre-defined PL/SQL packages for row-level security | No | Yes | No |
| Is provided in the default installation of Oracle Database | Yes | Yes | Yes |
| Is provided as an additional option to Oracle Database and must be licensed | No | No | Yes |

¹ Oracle Label Security uses predefined PL/SQL packages, not user-created packages, to attach security policies to tables.

² If you want to restrict privileged user access, consider using Oracle Database Vault.

³ Usually, this column is hidden to achieve transparency and not break applications that are not designed to show an additional column.

⁴ Oracle Virtual Private Database does not provide a user account, but you can create a user account that is solely responsible for managing Virtual Private Database policies.

⁵ The LBACSYS account manages Oracle Label Security policies. This provides an additional layer of security in that one specific user account is responsible for these policies, which reduces the risk of another user tampering with the policies.

See Also:

- [Controlling Access with Oracle Database Vault](#) (page 5-1)
 - [Restricting Access with Oracle Virtual Private Database](#) (page 6-1)
 - [Limiting Access to Sensitive Data Using Oracle Data Redaction](#) (page 7-1)
-
-

8.3 Guidelines for Planning an Oracle Label Security Policy

Before you create an Oracle Label Security policy, determine how to apply the labels to the application schema.

To determine where and how to apply Oracle Label Security policies for application data, follow these guidelines:

1. **Analyze the application schema.** Identify the tables that require an Oracle Label Security policy. In most cases, only a small number of the application tables will require an Oracle Label Security policy. For example, tables that store lookup values or constants usually do not need to be protected with a security policy. However, tables that contain sensitive data, such as patient medical histories or employee salaries, do.
2. **Analyze the use of data levels.** After you identify the candidate tables, evaluate the data in the tables to determine the level of security for the table. Someone who has broad familiarity with business operations can provide valuable assistance with this stage of the analysis.

Data levels refer to the sensitivity of the data. `PUBLIC`, `SENSITIVE`, and `HIGHLY SENSITIVE` are examples of data levels. You should also consider future sensitivities. Doing so creates a robust set of label definitions.

Remember that if a data record is assigned a sensitivity label whose level component is lower than the clearance of the user, then a user attempting to read the record is granted access to that row.

3. **Analyze the use of data compartments.** Data compartments are used primarily in government environments. If your application is a commercial application, in most cases, you will not create data compartments.
4. **Analyze the data groups.** Data groups and data compartments are typically used to control access to data by organization, region, or data ownership. For example, if the application is a sales application, access to the sales data can be controlled by country or region.

When a data record is assigned a sensitivity label with compartments and groups, a user attempting to read the record must have a user clearance that contains a level that is equal to or greater than the level of the data label, all of its compartments, and at least one of the groups in the sensitivity label. Because groups are hierarchical, a user could have the parent of one of the groups in the sensitivity label assigned to the data label and still be able to access that record.

5. **Analyze the user population.** Separate the users into one or more designated user types. For example, a user might be designated as a typical user, privileged user, or administrative user. After you create these categories of users, compare the categories with the data levels you created in Step 2 (page 8-4). They must correspond correctly for each table identified during the schema analysis you

- performed in Step 1 (page 8-4). Then, compare the organizational structure of the user population with the data groups that you identified in Step 4 (page 8-4).
6. **Examine the highly privileged and administrative users to determine which Oracle Label Security authorizations should be assigned to the user.** Oracle Label Security has several special authorizations that can be assigned to users. In general, typical users do not require any special authorizations. See *Oracle Label Security Administrator's Guide* for a complete list of these authorizations.
 7. **Review and document the data you gathered.** This step is crucial for continuity across the enterprise, and the resulting document should become part of the enterprise security policy. For example, this document should contain a list of protected application tables and corresponding justifications.

8.4 Tutorial: Creating Levels of Access to Table Data Based on the User

You can create different levels of Oracle Label Security access to table data based on who the user is.

[About Creating Levels of Access to Table Data Based on the User](#) (page 8-6)

This tutorial demonstrates the general concepts of using Oracle Label Security.

[Step 1: Enable Oracle Label Security](#) (page 8-6)

In a default Oracle Database installation, Oracle Label Security is installed but you must manually enable it.

[Step 2: Enable the LBACSYS Account](#) (page 8-7)

After you have enabled Oracle Label Security, you must enable the default Oracle Label Security account, which is called LBACSYS.

[Step 3: Create a Role and Three Users for the Oracle Label Security Tutorial](#) (page 8-8)

You are ready to create a role and three users, and then grant these users the role.

[Step 4: Create the ACCESS_LOCATIONS Oracle Label Security Policy](#) (page 8-9)

After you create the user accounts, you are ready to create the ACCESS_LOCATIONS policy.

[Step 5: Define the ACCESS_LOCATIONS Policy-Level Components](#) (page 8-11)

Next, you are ready to create label components for the policy.

[Step 6: Create the ACCESS_LOCATIONS Policy Data Labels](#) (page 8-12)

In this step, you create data labels for the ACCESS_LOCATION policy.

[Step 7: Create the ACCESS_LOCATIONS Policy User Authorizations](#) (page 8-13)

Next, you are ready to create user authorizations for the policy.

[Step 8: Apply the ACCESS_LOCATIONS Policy to the HR.LOCATIONS Table](#) (page 8-16)

Next, you are ready to apply the policy to the HR.LOCATIONS table.

[Step 9: Add the ACCESS_LOCATIONS Labels to the HR.LOCATIONS Data](#) (page 8-16)

You must apply the labels of the policy to the OLS_COLUMN in LOCATIONS of the HR.LOCATIONS table.

Step 10: Test the ACCESS_LOCATIONS Policy (page 8-19)

You can test the ACCESS_LOCATIONS policy by having the three users and perform a SELECT on the HR.LOCATIONS table.

Step 11: Optionally, Remove the Components for This Tutorial (page 8-21)

You can remove the components that you created for this tutorial if you no longer need them.

8.4.1 About Creating Levels of Access to Table Data Based on the User

This tutorial demonstrates the general concepts of using Oracle Label Security.

In this tutorial, you will apply security labels to the HR.LOCATIONS table. Three users, sking, kpartner, and ldoran will have access to specific rows within this table, based on the cities listed in the LOCATIONS table.

With Oracle Label Security, you restrict user access to data by focusing on row data, and designing different levels of access based on the sensitivity of your data. If you must restrict user access by focusing on user privileges, or some other method such as the job title that the user in your organization has, you can create a PL/SQL function or procedure to use with a Virtual Private Database policy. See [Restricting Access with Oracle Virtual Private Database](#) (page 6-1), for more information.

The schema for HR.LOCATIONS is as follows:

| Name | Null? | Type |
|----------------|----------|--------------|
| LOCATION_ID | NOT NULL | NUMBER(4) |
| STREET_ADDRESS | | VARCHAR2(40) |
| POSTAL_CODE | | VARCHAR2(12) |
| CITY | NOT NULL | VARCHAR2(30) |
| STATE_PROVINCE | | VARCHAR2(25) |
| COUNTRY_ID | | CHAR(2) |

You will apply the following labels:

| Label | Privileges |
|--------------|---|
| CONFIDENTIAL | Read access to the cities Munich, Oxford, and Roma |
| SENSITIVE | Read access to the cities Beijing, Tokyo, and Singapore |
| PUBLIC | Read access to all other cities listed in HR.LOCATIONS |

8.4.2 Step 1: Enable Oracle Label Security

In a default Oracle Database installation, Oracle Label Security is installed but you must manually enable it.

To enable Oracle Label Security:

1. Log into the database instance as user SYS with the SYSDBA administrative privilege.

For example:

```
sqlplus sys as sysdba
Enter password: password
```

2. Check if Oracle Label Security has been registered with the database.

```
SELECT STATUS FROM DBA_OLS_STATUS WHERE NAME = 'OLS_CONFIGURE_STATUS';
```

If it returns TRUE, then Oracle Label Security has been registered with the database. If the output is FALSE, then run the following procedure:

```
EXEC LBACSYS.CONFIGURE_OLS;
```

3. Check if Oracle Label Security is enabled. The PARAMETER column is case sensitive, so use the case shown here.

```
SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Oracle Label Security';
```

If it returns TRUE, then Oracle Label Security is enabled. Go to [Step 2: Enable the LBACSYS Account](#) (page 8-7). If it returns FALSE, then run the following procedure to enable it:

```
EXEC LBACSYS.OLS_ENFORCEMENT.ENABLE_OLS;
```

4. If you needed to register and enable Oracle Label Security, then connect as user SYS with the SYSOPER privilege.

```
CONNECT sys as sysoper
Enter password: password
```

5. Restart the database.

```
SHUTDOWN IMMEDIATE
STARTUP
```

8.4.3 Step 2: Enable the LBACSYS Account

After you have enabled Oracle Label Security, you must enable the default Oracle Label Security account, which is called LBACSYS.

To enable the Oracle Label Security LBACSYS user account:

1. Access the Database home page for your target database as user SYS with the SYSDBA administrative privilege.

See *Oracle Database 2 Day DBA* for more information.

2. From the **Schema** menu, select **Users**.
3. Select the **LBACSYS** account and click **Edit**.

If the account is active (the status will say OPEN), then go to [Step 3: Create a Role and Three Users for the Oracle Label Security Tutorial](#) (page 8-8).

4. In the Edit User: LBACSYS page, enter the following settings:
 - **Enter Password** and **Confirm Password**: Enter a password that meets the requirements in [Requirements for Creating Passwords](#) (page 2-17).
 - **Status**: Set the status to **Unlocked**.
5. Click **Apply**.
6. Select the **System Privileges** tab.
7. In the System Privileges page, select the **Edit List** button.

8. In the Modify System Privileges page, select **SELECT ANY DICTIONARY** from the Available System Privileges list, and then move it to the Selected System Privileges list. Click **OK**.
9. In the Edit User page, click **Apply**.

8.4.4 Step 3: Create a Role and Three Users for the Oracle Label Security Tutorial

You are ready to create a role and three users, and then grant these users the role.

[Creating a Role](#) (page 8-8)

The `emp_role` role provides the necessary privileges for the three users you will create.

[Creating the Oracle Label Security Users](#) (page 8-8)

The three users that you must create will have different levels of access to the `HR.LOCATIONS` table, depending on their position.

8.4.4.1 Creating a Role

The `emp_role` role provides the necessary privileges for the three users you will create.

To create the role `emp_role`:

1. Access the Database home page.
See Oracle Database 2 Day DBA for more information. Log in as user SYSTEM with the NORMAL privilege.
2. From the **Security** menu, select **Roles**.
3. In the Roles page, click **Create**.
4. In the Create Role page, in the **Name** field, enter `EMP_ROLE` and leave Authentication set to **None**.
5. Select the **Object Privileges** subpage.
6. From the **Select Object Type** list, select **Table**, and then click **Add**.
The Add Table Object Privileges page appears.
7. Under Select Table Objects, enter `HR.LOCATIONS` to select the `LOCATIONS` table in the `HR` schema, and then under Available Privileges, move `SELECT` to the Selected Privileges list.
8. Click **OK** to return to the Create Role page, and then click **OK** to return to the Roles page.

8.4.4.2 Creating the Oracle Label Security Users

The three users that you must create will have different levels of access to the `HR.LOCATIONS` table, depending on their position.

Steven King (`sking`) is the advertising president, so he has full read access to the `HR.LOCATIONS` table. Karen Partners (`kpartner`) is a sales manager who has less access, and Louise Doran (`ldoran`) is a sales representative who has the least access.

To create the users:

1. From the **Schema** menu, select **Users**.
The Users page appears.
2. Click **Create**.
3. In the Create User page, enter the following information:
 - **Name:** SKING
 - **Profile:** DEFAULT
 - **Authentication:** Password
 - **Enter Password and Confirm Password:** Enter a password that meets the requirements in [Requirements for Creating Passwords](#) (page 2-17).
 - **Default Tablespace:** USERS
 - **Temporary Tablespace:** TEMP
 - **Status:** Set to **Unlocked**.
 - **Roles:** Select the **Roles** subpage, and then grant the emp_role role to sking by selecting **Edit List**. From the Available Roles list, select emp_role, and then click **Move** to move it to the Selected Roles list. Click **OK**. In the Create User page, ensure that the **Default** box is selected for both the CONNECT and emp_role roles.
 - **System Privileges:** Select the **System Privileges** subpage and then click **Edit List** to grant the CREATE SESSION privilege. Do not grant sking the ADMIN OPTION option.
4. Click **OK** to return to the Create User page, and then from there, click **OK** to return to the Users page.
5. In the Users page, select SKING, set **Actions** to **Create Like**, and then click **Go**.
6. In the Create User page, create accounts for kpartner and ldoran.
Create their names and passwords. (See [Requirements for Creating Passwords](#) (page 2-17).) You do not need to grant roles or system privileges to them. Their roles and system privileges, defined in the sking account, are automatically created.

At this stage, you have created three users who have identical privileges. All of these users have the SELECT privilege on the HR.LOCATIONS table, through the EMP_ROLE role.

8.4.5 Step 4: Create the ACCESS_LOCATIONS Oracle Label Security Policy

After you create the user accounts, you are ready to create the ACCESS_LOCATIONS policy.

To create the ACCESS_LOCATIONS policy:

1. Log in to the Enterprise Manager target database as user as user LBACSYS with the **NORMAL** role selected.

2. From the **Security** menu, select **Label Security**.
3. In the Label Security Policies page, click **Create**.
4. In the Create Label Security Policy page, enter the following information:
 - **Name:** ACCESS_LOCATIONS
 - **Label Column:** OLS_COLUMN
Later on, when you apply the policy to a table, the label column is added to that table. By default, the data type of the policy label column is NUMBER (10).
 - **Hide Label Column:** Deselect this box so that the label column will not be hidden. (It should be deselected by default.)
Usually, the label column is hidden, but during the development phase, you may want to have it visible so that you can check it. After the policy is created and working, hide this column so that it is transparent to applications. Many applications are designed not to show an another column, so hiding the column prevents the application from breaking.
 - **Enabled:** Select this box to enable the policy. (It should be enabled by default.)
 - **Inverse user's read and write groups (INVERSE_GROUP):** Do not select this option.
 - **Default Policy Enforcement Options:** Select **Apply Policy Enforcement**, and then select the following options:
 - For all queries (READ_CONTROL)**
 - To use session's default label for label column update (LABEL_DEFAULT)**
5. Click **OK**.
The ACCESS_LOCATIONS policy appears in the Label Security Policies page.

Label Security Policies

Update Message
Label Security Policy ACCESS_LOCATIONS has been created successfully

Search
Specify a policy to filter the data that is displayed in your results set

Policy Name

Selection Mode

Actions

| Select | Policy Name | Enabled | Label Column |
|--------------------------|------------------|-------------------------------------|--------------|
| <input type="checkbox"/> | ACCESS_LOCATIONS | <input checked="" type="checkbox"/> | OLS_COLUMN |

TIP Use caution while disabling a policy as anyone who connects to the database can access all the data normally protected by the policy

8.4.6 Step 5: Define the ACCESS_LOCATIONS Policy-Level Components

Next, you are ready to create label components for the policy.

At a minimum, you must create one or more levels, such as `PUBLIC` or `SENSITIVE`; and define a long name, a short name, and a number indicating the sensitivity level. Compartments and groups are optional.

The level numbers indicate the level of sensitivity needed for their corresponding labels. Select a numeric range that can be expanded later on, in case your security policy needs more levels. For example, to create the additional levels `LOW_SENSITIVITY` and `HIGH_SENSITIVITY`, you can assign them numbers 7300 (for `LOW_SENSITIVITY`) and 7600 (for `HIGH_SENSITIVITY`), so that they fit in the scale of security your policy creates. Generally, the higher the number, the more sensitive the data.

Compartments identify areas that describe the sensitivity of the labeled data, providing a finer level of granularity within a level. Compartments are optional.

Groups identify organizations owning or accessing the data. Groups are useful for the controlled dissemination of data and for timely reaction to organizational change. Groups are optional.

In this step, you define the level components, which reflect the names and relationships of the `SENSITIVE`, `CONFIDENTIAL`, and `PUBLIC` labels that you must create for the `ACCESS_LOCATIONS` policy.

To define the label components for the ACCESS_LOCATIONS policy:

1. In the Label Security policies page, select the `ACCESS_LOCATIONS` policy, and then select **Edit**.
2. In the Edit Label Security Policy page, select the **Label Components** subpage.

- Under Levels, click **Add 5 Rows**, and then enter a long name, short name, and a numeric tag as follows. (To move from one field to the next, press the **Tab** key.)

Table 8-2 Values for Oracle Label Security Levels

| Long Name | Short Name | Numeric Tag |
|--------------|------------|-------------|
| SENSITIVE | SENS | 3000 |
| CONFIDENTIAL | CONF | 2000 |
| PUBLIC | PUB | 1000 |

- Click **Apply**.

8.4.7 Step 6: Create the ACCESS_LOCATIONS Policy Data Labels

In this step, you create data labels for the `ACCESS_LOCATION` policy.


To create the data label, you must assign a numeric tag to each level. Later on, the tag number will be stored in the security column when you apply the policy to a table. It has nothing to do with the sensitivity of the label; it is only used to identify the labels for the policy.

To create the data labels:

- Return to the Label Security policies page by selecting the **Label Security Policies** link.
- Select the selection button for the `ACCESS_LOCATIONS` policy.
- In the Actions list, select **Data Labels**, and then click **Go**.
- In the Data Labels page, click **Add**.
- In the Create Data Label page, enter the following information:
 - Numeric Tag:** Enter 1000.
 - Level:** Enter PUB.

Create Data Label
 Every label should have a level field and optionally one or more compartments and/or groups. Also every label needs to have a numeric tag associated with it that uniquely identifies it across all policies in the database

* Numeric Tag
should not be more than 8 digits

* Level 

- Click **OK**.
 The data label appears in the Data Labels page.
- Click **Add** again, and then create a data label for the `CONF` label as follows:
 - Numeric Tag:** Enter 2000.
 - Level:** Select `CONF` from the list.
- Click **OK**.

9. Click **Add** again, and then create a data label for the `SENS` label as follows:
 - **Numeric Tag:** Enter 3000.
 - **Level:** Select `SENS` from the list.
10. Click **OK**.

The `CONF`, `PUB`, and `SENS` labels appear in the Data Labels page.

| <input type="button" value="Add"/> | | |
|--|-------------------|-------------|
| <input type="button" value="Edit"/> <input type="button" value="View"/> <input type="button" value="Create Like"/> <input type="button" value="Delete"/> | | |
| Select | Label | Numeric Tag |
| <input checked="" type="radio"/> | <code>CONF</code> | 2000 |
| <input type="radio"/> | <code>PUB</code> | 1000 |
| <input type="radio"/> | <code>SENS</code> | 3000 |

Later, the tag number will be stored in the security column when you apply the policy to the `HR.LOCATIONS` table. It has nothing to do with the sensitivity of the label; it is only used to identify the labels for the policy.

8.4.8 Step 7: Create the `ACCESS_LOCATIONS` Policy User Authorizations

Next, you are ready to create user authorizations for the policy.

To create user authorizations for the policy:

1. Return to the Label Security policies page by selecting the **Label Security Policies** link.
2. Select the selection button for the `ACCESS_LOCATIONS` policy.
3. In the **Actions** list, select **Authorization**, and then click **Go**.
4. In the Authorization page, click **Add Users**.
5. In the Add User: Users page, under Database Users, click **Add**.

The Search and Select: Userpage appears. Enter `SKING`, and then click **Go**.

Typically, a database user account already has been created in the database, for example, by using the `CREATE USER SQL` statement.

The other option is **Non Database Users**. Most application users are considered nondatabase users. A nondatabase user does not exist in the database. This can be any user name that meets the Oracle Label Security naming standards and can fit into the `VARCHAR2(30)` length field. However, be aware that Oracle Database does not automatically configure the associated security information for the nondatabase user when the application connects to the database. In this case, the application must call an Oracle Label Security function to assume the label authorizations of the specified user who is not a database user.

6. Select the check box for user `SKING`, and then click **Select**.

The Create User page lists user `SKING`.

| Select | Name |
|--------------------------|-------|
| <input type="checkbox"/> | SKING |

7. Select the check box for user **SKING** and then click **Next**.
(You may need to refresh the page to display user SKING's check box.)
8. In the Privileges page, select **Next** to move to the Audit page.
Oracle Label Security enforces the policy through the label authorizations. The Privileges page enables the user to override the policy label authorization, so do not select any of its options.
9. In the Labels, Compartments and Groups page, enter the following settings:
 - **Maximum Level:** SENS (for SENSITIVE)
 - **Minimum Level:** CONF (for CONFIDENTIAL)
 - **Default Level:** SENS
 - **Row Level:** SENS
10. Click **Next** to go to the Audit page.
11. In the Audit pane of the Add Users: Audit page, ensure that all of the audit operations are set to None, and then click **Next**.
The Review page appears.

Add Users: Review

Policy Name: ACCESS_LOCATIONS
Users: SKING
Privileges:

Levels

Maximum Level: SENS
Minimum Level: CONF
Default Level: SENS
Row Level: SENS

Compartments

| Short Name | Write | Default | Row |
|-----------------------|-------|---------|-----|
| No Compartments Found | | | |

Groups

| Short Name | Write | Default | Row |
|-----------------|-------|---------|-----|
| No Groups Found | | | |

Audit

| Operation | Audit On Success By | Audit On Failure By |
|--------------------------------|---------------------|---------------------|
| Policy Applied | None | None |
| Policy Removed | None | None |
| Labels And Privileges Set | None | None |
| All Policy Specific Privileges | None | None |

12. Ensure that the settings are correct, and then click **Finish**.

The Review page lists all the authorization settings you have selected.

13. Repeat Step 4 (page 8-13) through Step 12 (page 8-15) to create the following authorizations for user KPARTNER, so that she can read confidential and public data in HR.LOCATIONS.
- **Privileges:** Select no privileges.
 - **Labels, Compartments And Groups:** Set the four levels to the following:
 - **Maximum Level:** CONF (for CONFIDENTIAL)
 - **Minimum Level:** PUB (for PUBLIC)
 - **Default Level:** CONF
 - **Row Level:** CONF
 - **Audit:** Set all to None.
14. Create the following authorizations for user LDORAN, who is only allowed to read public data from HR.LOCATIONS:
- **Privileges:** Select no privileges.
 - **Labels, Compartments And Groups:** Set all four levels to PUB.
 - **Audit:** Set all to None.

8.4.9 Step 8: Apply the ACCESS_LOCATIONS Policy to the HR.LOCATIONS Table

Next, you are ready to apply the policy to the HR.LOCATIONS table.

To apply the ACCESS_LOCATIONS policy to the HR.LOCATIONS table:

1. Return to the Label Security policies page by selecting the **Label Security Policies** link.
2. Select the selection button for the ACCESS_LOCATIONS policy.
3. In the **Actions** list, select **Apply**, and then click **Go**.
4. In the Apply page, click **Create**.

The Add Table page appears.

5. In the **Table** field, enter HR.LOCATIONS.
6. Ensure that the **Hide Policy Column** box is not selected.
7. Ensure that the **Enabled** box is selected.
8. Under Policy Enforcement Options, select **Use Default Policy Enforcement**.

By choosing **Use Default Policy Enforcement**, you are automatically choosing these options:

- **For all queries (READ_CONTROL)**
- **Use session's default label for label column update (LABEL_DEFAULT)**

9. Click **OK**.

The ACCESS_LOCATIONS policy is applied to the HR.LOCATIONS table.

| Select | Table | Schema | Enforcement Options | Enabled |
|--------|-----------|--------|-----------------------------|-------------------------------------|
| | LOCATIONS | HR | Read Control, Label Default | <input checked="" type="checkbox"/> |

8.4.10 Step 9: Add the ACCESS_LOCATIONS Labels to the HR.LOCATIONS Data

You must apply the labels of the policy to the OLS_COLUMN in LOCATIONS of the HR.LOCATIONS table.

[Granting HR FULL Policy Privilege for the HR.LOCATIONS Table](#) (page 8-17)

The label security administrative user, LBACSYS, can grant HR the necessary privilege.

[Updating the OLS_COLUMN Table in HR.LOCATIONS](#) (page 8-17)

The user HR now can update the OLS_COLUMN column in the HR.LOCATIONS table to include data labels that will be assigned to specific rows in the table, based on the cities listed in the CITY column.

8.4.10.1 Granting HR FULL Policy Privilege for the HR.LOCATIONS Table

The label security administrative user, LBACSYS, can grant HR the necessary privilege.

To grant HR FULL access to the ACCESS_LOCATIONS policy:

1. Return to the Label Security policies page by selecting the **Label Security Policies** link.
2. Select the selection button for the **ACCESS_LOCATIONS** policy.
3. Select **Authorization** from the **Actions** list, and then click **Go**.
4. In the Authorization page, click **Add Users**.
5. In the Add Users page, under Database Users, click **Add**.
6. In the Search and Select window, select the box for user HR, and then click **Select**.
The Add User page lists user HR.
7. Click **Next** to display the Privileges page.
8. Select the **Bypass all Label Security checks (FULL)** privilege, and then click **Next**.
9. Click **Next** to display the Levels, Compartments and Groups page.
10. Click **Next**.
11. In the Audit page, click **Next** to display the Review page.
12. Click **Finish**.

At this stage, HR is listed in the Authorization page with the other users.

| Add Users | | | | |
|--|----------|--------------------|---------------------|------------|
| Edit View Create Like Delete | | | | |
| Select | User | Maximum Read Label | Maximum Write Label | Privileges |
| <input checked="" type="radio"/> | HR | | | Full |
| <input type="radio"/> | KPARTNER | CONF | CONF | |
| <input type="radio"/> | LDORAN | PUB | PUB | |
| <input type="radio"/> | SKING | SENS | SENS | |

13. Do not exit Enterprise Manager.

8.4.10.2 Updating the OLS_COLUMN Table in HR.LOCATIONS

The user HR now can update the OLS_COLUMN column in the HR.LOCATIONS table to include data labels that will be assigned to specific rows in the table, based on the cities listed in the CITY column.

To update the OLS_COLUMN table in HR.LOCATIONS:

1. In SQL*Plus, connect as user HR.

```
CONNECT HR
Enter password: password
```

If you cannot log in as HR because this account locked and expired, log in as SYSTEM and then enter the following statement. Replace *password* with an appropriate password for the HR account. For greater security, do not reuse the same password that was used in previous releases of Oracle Database. See [Requirements for Creating Passwords](#) (page 2-17).

```
ALTER USER HR ACCOUNT UNLOCK IDENTIFIED BY password
```

After you complete this ALTER USER statement, try logging in as user HR again.

2. Enter the following UPDATE statement to apply the SENS label to the cities Beijing, Tokyo, and Singapore:

```
UPDATE LOCATIONS
SET ols_column = CHAR_TO_LABEL('ACCESS_LOCATIONS','SENS')
WHERE UPPER(city) IN ('BEIJING', 'TOKYO', 'SINGAPORE');
```

3. Enter the following UPDATE statement to apply the CONF label to the cities Munich, Oxford, and Roma:

```
UPDATE LOCATIONS
SET ols_column = CHAR_TO_LABEL('ACCESS_LOCATIONS','CONF')
WHERE UPPER(city) IN ('MUNICH', 'OXFORD', 'ROMA');
```

4. Enter the following UPDATE statement to apply the PUB label to the remaining cities:

```
UPDATE LOCATIONS
SET ols_column = CHAR_TO_LABEL('ACCESS_LOCATIONS','PUB')
WHERE ols_column IS NULL;
```

5. To check that the columns were updated, enter the following statement:

```
SELECT LABEL_TO_CHAR (OLS_COLUMN) FROM LOCATIONS;
```

The following output should appear:

```
LABEL_TO_CHAR(OLS_COLUMN)
```

```
-----
CONF
PUB
SENS
PUB
PUB
PUB
PUB
PUB
PUB
PUB
PUB
PUB
SENS
PUB
PUB
SENS
PUB
CONF
PUB
CONF
PUB
PUB
PUB
PUB
PUB
```


23 rows selected.

Note:

Using the label column name (OLS_COLUMN) explicitly in the preceding query enables you to see the label column, even if it was hidden.

If the label column is hidden, and you do not specify the label column name explicitly, then the label column is not displayed in the query results. For example, using the `SELECT * FROM LOCATIONS` query does not show the label column if it is hidden. This feature enables the label column to remain transparent to applications. An application that was designed before the label column was added does not know about the label column and will never see it.

8.4.11 Step 10: Test the ACCESS_LOCATIONS Policy

You can test the ACCESS_LOCATIONS policy by having the three users and perform a SELECT on the HR.LOCATIONS table.

To test the ACCESS_LOCATIONS policy:

1. In SQL*Plus, connect as user sking.

```
CONNECT sking
Enter password: password
```

2. Enter the following:

The following commands format the width of the table columns so that you can read them easier. You only need to perform this step once for the entire session (including when kpartner and ldoran log in.)

```
COL city HEADING City FORMAT a25
COL country_id HEADING Country FORMAT a11
COL Label format a10
```

Now enter the SELECT statement as follows:

```
SELECT CITY, COUNTRY_ID, LABEL_TO_CHAR (OLS_COLUMN)
AS LABEL FROM HR.LOCATIONS ORDER BY OLS_COLUMN;
```

User sking is able to access all 23 rows of the HR.LOCATIONS table. Even though he is only authorized to access rows that are labeled CONF and SENS, he can still read (but not write to) rows labeled PUB.

| City | Country | LABEL |
|---------------------|---------|-------|
| ----- | | |
| Venice | IT | PUB |
| Utrecht | NL | PUB |
| Bern | CH | PUB |
| Geneva | CH | PUB |
| Sao Paulo | BR | PUB |
| Stretford | UK | PUB |
| Mexico City | MX | PUB |
| Hiroshima | JP | PUB |
| Southlake | US | PUB |
| South San Francisco | US | PUB |

| | | |
|-----------------|----|------|
| South Brunswick | US | PUB |
| Seattle | US | PUB |
| Toronto | CA | PUB |
| Whitehorse | CA | PUB |
| Bombay | IN | PUB |
| Sydney | AU | PUB |
| London | UK | PUB |
| Oxford | UK | CONF |
| Munich | DE | CONF |
| Roma | IT | CONF |
| Singapore | SG | SENS |
| Tokyo | JP | SENS |
| Beijing | CN | SENS |

23 rows selected.

- Repeat Steps 1 (page 8-19) and 2 (page 8-19) for users kpartner and ldoran.

User KPARTNER can access the rows labeled CONF and PUB:

| City | Country | LABEL |
|---------------------|---------|-------|
| ----- | ----- | ----- |
| Venice | IT | PUB |
| Utrecht | NL | PUB |
| Bern | CH | PUB |
| Mexico City | MX | PUB |
| Hiroshima | JP | PUB |
| Southlake | US | PUB |
| South San Francisco | US | PUB |
| South Brunswick | US | PUB |
| Seattle | US | PUB |
| Toronto | CA | PUB |
| Whitehorse | CA | PUB |
| Bombay | IN | PUB |
| Sydney | AU | PUB |
| London | UK | PUB |
| Stretford | UK | PUB |
| Sao Paulo | BR | PUB |
| Geneva | CH | PUB |
| Oxford | UK | CONF |
| Munich | DE | CONF |
| Roma | IT | CONF |

20 rows selected.

User LDORAN can access the rows labeled PUB:

| City | Country | LABEL |
|---------------------|---------|-------|
| ----- | ----- | ----- |
| Venice | IT | PUB |
| Hiroshima | JP | PUB |
| Southlake | US | PUB |
| South San Francisco | US | PUB |
| South Brunswick | US | PUB |
| Seattle | US | PUB |
| Toronto | CA | PUB |
| Whitehorse | CA | PUB |
| Bombay | IN | PUB |
| Sydney | AU | PUB |
| London | UK | PUB |
| Stretford | UK | PUB |
| Sao Paulo | BR | PUB |
| Geneva | CH | PUB |

| | | |
|-------------|----|-----|
| Bern | CH | PUB |
| Utrecht | NL | PUB |
| Mexico City | MX | PUB |

17 rows selected.

4. Exit SQL*Plus.

8.4.12 Step 11: Optionally, Remove the Components for This Tutorial

You can remove the components that you created for this tutorial if you no longer need them.

To remove the components for this tutorial:

1. Return to the Label Security policies page by selecting the **Label Security Policies** link.

2. Select the `ACCESS_LOCATIONS` policy and then click **Delete**. In the Confirmation page, select the **Drop column** check box and then click **Yes**.

Deleting the `ACCESS_LOCATIONS` policy also drops the `OLS_COLUMN` column from the `HR.LOCATIONS` table.

3. Log out of Enterprise Manager and then log back in as user `SYSTEM`.
4. From the **Schema** menu, select **Users**.
5. Select user `KPARTNER`, and then click **Delete**.
6. In the Confirmation page, click **Yes**.
7. Repeat Step 5 (page 8-21) and Step 6 (page 8-21) to remove users `ldoran` and `sking`.
8. Click the **Database Instance** link to return to the Database home page.
9. From the **Security** menu, select **Roles**.
10. Select the role `emp_role`, and then click **Delete**.
11. In the Confirmation dialog box, click **Yes**.
12. Log out of Enterprise Manager.
13. If necessary, disable Oracle Label Security.
 - a. Log into the database instance as either `SYS` with the `SYSDBA` administrative privilege, or as a user who has been granted the `LBAC_DBA` role.

For example:

```
sqlplus lbacsys
Enter password: password
```

- b. Run the following procedure:

```
EXEC LBACSYS.OLS_ENFORCEMENT.DISABLE_OLS
```

- c. Restart the database:

```
CONNECT SYS AS SYSOPER  
Enter password: password
```

```
SHUTDOWN IMMEDIATE  
STARTUP
```

Auditing Database Activity

Unified auditing enables you to create policies that audit user behavior in the database in a variety of ways.

[About Auditing](#) (page 9-1)

Auditing is the monitoring and recording of selected user database actions.

[Why Is Auditing Used?](#) (page 9-2)

Auditing enables activities such as enabling user accountability, deterring inappropriate user actions, and investigating suspicious activity.

[Tutorial: Creating a Unified Audit Policy](#) (page 9-3)

In this tutorial, you create unified audit policy that monitors SELECT statements on the OE.CUSTOMERS table.

9.1 About Auditing

Auditing is the monitoring and recording of selected user database actions.

To perform auditing, you must be granted the appropriate system privileges. To better facilitate separation of duty, the following two default roles are provided:

- `AUDIT_ADMIN` role, which enables you to configure auditing and administer both unified audit policies and fine-grained audit policies. It also enables you to view and analyze audit data. Typically, security administrators are granted this role.
- `AUDIT_VIEWER` role, which enables you to view and analyze audit data only. Typically, external auditors are granted this role.

This section provides an introduction to unified auditing, which captures audit records from the following locations:

- Audit records (including `SYS` audit records) from unified audit policies and `AUDIT` settings
- Fine-grained audit records from the `DBMS_FGA` PL/SQL package
- Oracle Real Application Security audit records
- Oracle Recovery Manager audit records
- Oracle Database Vault audit records
- Oracle Label Security audit records

Oracle Database consolidates these records in one location, in one format, viewable from the `UNIFIED_AUDIT_TRAIL` view for single database instances and `GV$UNIFIED_AUDIT_TRAIL` for Oracle Real Application Clusters environments.

When you upgrade your database to the current release, you must manually migrate to unified auditing if you want to use it. After you complete the migration, in an upgraded database, the audit records from the previous release are still available. You then can archive and purge these older audit trails. Afterwards, the new audit records will be written to the unified audit trail.

For a newly created database, Oracle Database provides mixed mode-enabled auditing, which enables both the old and new audit facilities to run simultaneously.

When you create and enable a unified audit policy, the policy begins to collect audit records right away. You do not need to set initialization parameters to enable overall auditing, as was necessary in previous releases. The policy can be as simple as auditing the activities of a single user or you can create complex audit policies that use conditions. You can have more than one audit policy in effect at a time in a database. An audit policy can contain both system-wide and object-specific audit options. Most of the auditing that you will do for general activities (including standard auditing) requires the use of audit policies

Another type of auditing is fine-grained auditing. Fine-grained auditing provides most of the auditing capabilities as unified auditing, plus the following functionality:

- **Auditing specific columns.** You can audit specific relevant columns that hold sensitive information, such as salaries or Social Security numbers.
- **Using event handlers.** For example, you can write a function that sends an email alert to a security administrator when an audited column that should not be changed at midnight is updated.

Oracle provides three predefined audit policies that cover commonly used security relevant audit settings. These policies are designed to provide an effective method of enforcing strong internal controls so that your site meets its regulatory compliance requirements.

See Also:

- *Oracle Database Upgrade Guide* for information about migrating to unified auditing and about mixed mode auditing
 - *Oracle Database Security Guide* for information about archiving and purging the audit trail
 - *Oracle Database Security Guide* for information about fine-grained auditing
 - *Oracle Database Security Guide* for more information about the predefined audit policies
-
-

9.2 Why Is Auditing Used?

Auditing enables activities such as enabling user accountability, deterring inappropriate user actions, and investigating suspicious activity.

Auditing is used for the following reasons:

- **Enable accountability for actions.** These include actions taken in a particular schema, table, or row, or affecting specific content.
- **Deter users from inappropriate actions based on that accountability.**

- **Investigate suspicious activity.** For example, if a user is deleting data from tables, then a security administrator might decide to audit all connections to the database and all successful and unsuccessful deletions of rows from all tables in the database.
- **Notify an auditor of actions by an unauthorized user.** For example, an unauthorized user could change or delete data, or a user has more privileges than expected, which can lead to reassessing user authorizations.
- **Detect problems with an authorization or access control implementation.** For example, you can create audit policies that you expect will never generate an audit record because the data is protected in other ways. However, if these policies do generate audit records, then you will know the other security controls are not properly implemented.
- **Address auditing requirements for compliance.** Regulations such as the following have common auditing-related requirements:
 - Sarbanes-Oxley Act
 - Health Insurance Portability and Accountability Act (HIPAA)
 - International Convergence of Capital Measurement and Capital Standards: a Revised Framework (Basel II)
 - Japan Privacy Law
 - European Union Directive on Privacy and Electronic Communications
- **Monitor and gather data about specific database activities.** For example, the database administrator can gather statistics about which tables are being updated, how many logical I/O operations are performed, or how many concurrent users connect at peak times.

9.3 Tutorial: Creating a Unified Audit Policy

In this tutorial, you create unified audit policy that monitors `SELECT` statements on the `OE.CUSTOMERS` table.

Step 1: If Necessary, Enable Unified Auditing (page 9-4)

In this procedure, you check if unified auditing has been enabled, and if it has not, you enable it.

Step 2: Grant the SEC_ADMIN User the AUDIT_ADMIN Role (page 9-5)

Next, you are ready to grant the `sec_admin` user the `AUDIT_ADMIN` role, which enables `SEC_ADMIN` to create audit policies.

Step 3: Create and Enable a Unified Audit Policy (page 9-6)

In Enterprise Manager as user `sec_admin`, you can create a unified auditing policy for `SELECT` statements on the `OE.CUSTOMERS` table.

Step 4: Test the Unified Audit Policy (page 9-7)

With auditing enabled, you are ready to test the audit settings.

Step 5: Optionally, Remove the Components for This Tutorial (page 9-9)

You can remove the components of this tutorial if you no longer need them.

[Step 6: Optionally, Remove the SEC_ADMIN Security Administrator Account](#)
(page 9-9)

If you no longer need the `sec_admin` administrator account, then you should remove it.

9.3.1 Step 1: If Necessary, Enable Unified Auditing

In this procedure, you check if unified auditing has been enabled, and if it has not, you enable it.

To enable unified auditing:

1. Log in to SQL*Plus as user `SYS` with the `SYSDBA` administrative privilege.

```
sqlplus sys as sysdba
Enter password: password
```

2. Run the following query to find out if your database has been migrated to use unified auditing. Enter `Unified Auditing` in the case shown here.

```
SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Unified Auditing';
```

If the output for the `VALUE` column is `FALSE`, then complete the remaining steps in this section to migrate to unified auditing. If the output is `TRUE`, then unified auditing is enabled and you can go to [Step 2: Grant the SEC_ADMIN User the AUDIT_ADMIN Role](#) (page 9-5).

3. Stop the database.

For single-instance installations, enter the following commands from SQL*Plus:

```
SHUTDOWN IMMEDIATE
EXIT
```

For Windows systems, stop the Oracle service:

```
net stop OracleService%ORACLE_SID%
```

For Oracle Real Application Clusters (Oracle RAC) installations, shut down each database instance as follows:

```
srvctl stop database -db db_name
```

4. Stop the listener. (Stopping the listener is not necessary for Oracle RAC and Grid Infrastructure listeners.)

```
lsnrctl stop listener_name
```

You can find the name of the listener by running the `lsnrctl status` command. The name is indicated by the `Alias` setting.

5. Go to the `$ORACLE_HOME/rdbms/lib` directory.
6. Enable the unified auditing executable.

- **UNIX:** Run the following command:

```
make -f ins_rdbms.mk uniaud_on ioracle ORACLE_HOME=$ORACLE_HOME
```

- **Windows:** Rename the `%ORACLE_HOME%/bin/orauniaud12.dll.option` file to `%ORACLE_HOME%/bin/orauniaud12.dll`.

- Restart the listener.

```
lsnrctl start listener_name
```

- Restart the database. Log in to SQL*Plus and then enter the `STARTUP` command as follows:

```
sqlplus sys as sysoper
Enter password: password
```

```
SQL> STARTUP
```

For Windows systems, start the Oracle service again.

```
net start OracleService%ORACLE_SID%
```

For Oracle RAC installations, from a command line, restart the database as follows:

```
srvctl setenv database -db orcl
```

9.3.2 Step 2: Grant the SEC_ADMIN User the AUDIT_ADMIN Role

Next, you are ready to grant the `sec_admin` user the `AUDIT_ADMIN` role, which enables `SEC_ADMIN` to create audit policies.

To grant the SEC_ADMIN user the AUDIT_ADMIN role:

- Access the Database home page for your target database as user `SYS` with the `SYSDBA` administrative privilege.

See *Oracle Database 2 Day DBA* for more information.

- From the **Schema** menu, select **Users**.
- Select the **SEC_ADMIN** account and click **Edit**.

If the `sec_admin` user account does not exist, see [Step 2: Create a Security Administrator Account](#) (page 3-6) for instructions on how to create the `sec_admin` security administrator account.

- In the Edit User: `SEC_ADMIN` page, select the **Roles** tab.
- Select the **Edit List** button.
- In the Modify Roles page, select **AUDIT_ADMIN** from the Available Roles list, and then move it to the Selected Roles list. Click **OK**.
- In the Edit User: `SEC_ADMIN` page, select the **Object Privileges** tab.
- In the Edit User page: `SEC_ADMIN` page, from the **Object Type** list, select **Package**, and then click **Add**.
- In the Add Package Object Privileges page, do the following:
 - In the **Select Package Objects** field, enter `SYS.DBMS_AUDIT_MGMT`.
 - Under Available Packages, select **EXECUTE** and then click **Move** to send it to the Selected Privileges list.
 - Click **OK**.

10. In the Edit User page: SEC_ADMIN page, click **Apply**.

9.3.3 Step 3: Create and Enable a Unified Audit Policy

In Enterprise Manager as user `sec_admin`, you can create a unified auditing policy for `SELECT` statements on the `OE.CUSTOMERS` table.

To create the unified audit policy:

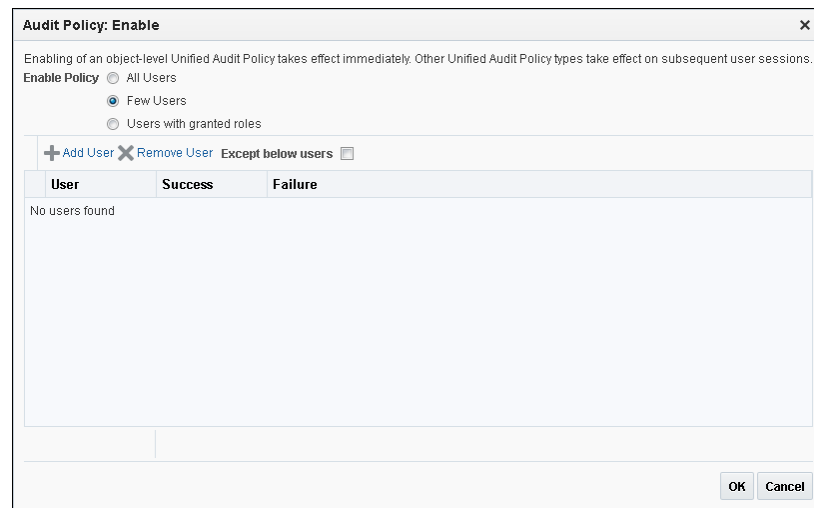
1. In Enterprise Manager, log out and then log back in again as user `sec_admin`.
2. From the **Security** menu, select **Audit Settings**.
3. In the Audit Settings page, click **Create**.
4. In the Create Audit Policy : Privileges and Roles page, enter the following settings:
 - **Name:** Enter `select_cust_pol` for the policy name.
 - **Comments:** Enter `Audit policy for SELECT statements on OE.CUSTOMERS`.
5. Click the **Next** button.
6. In the Create Audit Policy : Component Actions page, click **Next**.
7. In the Create Audit Policy : Object Actions page, click **Add Object**.
8. In the Add Object dialog box, enter the following settings:
 - **Schema:** `OE`
 - **Type:** `TABLE`
 - **Object:** `CUSTOMERS`
 - **Action:** `SELECT`
9. Click **OK**.

The Create Audit Policy : Object Actions page should appear as follows:

| Type | Action | Schema | Name |
|-------|--------|--------|-----------|
| TABLE | SELECT | OE | CUSTOMERS |

10. In the Create Audit Policy : Object Actions window click **Next**.
11. In the Create Audit Policy : Conditions window, click **Next**. In the Create Audit Policy : Review page, click **Submit**.
The Audit Settings page appears, with `SELECT_CUST_POL` in the list of policies.
12. Select `SELECT_CUST_POL` and then click the **Enable** button.
13. In the Enable Audit Policy dialog box, select the **Few Users** check box.

The Enable Audit Policy dialog expands so that you can add specific users.



14. Click the **Add User** button.
15. In the Add User dialog box, search for and enter OE (select the Show Oracle Supplied check box) and ensure that the **Audit on Success** and **Audit on Failure** check boxes are selected.
16. Click **OK**.
17. Repeat these steps to add user **HR** to the list.
18. Do not exit Enterprise Manager.

9.3.4 Step 4: Test the Unified Audit Policy

With auditing enabled, you are ready to test the audit settings.

Any `SELECT` statements that are performed on the `OE.CUSTOMERS` table will be written to the unified audit trail. To find the audit records, you must query the `UNIFIED_AUDIT_TRAIL` dynamic view.

A unified auditing policy takes effect in the next user session for the users who are being audited. So, before their audit records can be captured, the users must connect to the database *after* the policy has been created.

To test the audit settings:

1. In SQL*Plus, connect as user OE and query the `OE.CUSTOMERS` table so that you can generate a record for the audit trail.

```
connect OE
Enter password: password

SELECT COUNT(*) FROM CUSTOMERS;

COUNT(*)
-----
        319
```

2. Connect as user HR and try to run this query.

```
connect HR
Enter password: password

SELECT COUNT(*) FROM OE.CUSTOMERS;

ERROR at line 1:
ORA-00942: table or view does not exist
```

At this point, the audit trail contains two records, one for a successful query by user OE and another for a failed query attempt by user HR.

3. Connect as user SH and try to run the same query.

```
connect SH
Enter password: password

SELECT COUNT(*) FROM OE.CUSTOMERS;

ERROR at line 1:
ORA-00942: table or view does not exist
```

Because this user is not being audited, there should be no audit record for this action.

4. Connect as user sec_admin.

```
connect sec_admin
Enter password: password
```

5. Run then the following procedure.

```
EXEC SYS.DBMS_AUDIT_MGMT.FLUSH_UNIFIED_AUDIT_TRAIL;
```

If the audit trail mode is QUEUED, then audit records are not written to disk until the in-memory queues are full. This command explicitly flushes the queues to disk, so that you can see the audit trail records in the UNIFIED_AUDIT_TRAIL view.

6. Enter the following statement to query the UNIFIED_AUDIT_TRAIL view:

```
col dbusername format a12
col sql_text format a30
col event_timestamp format a38

SELECT DBUSERNAME, SQL_TEXT, EVENT_TIMESTAMP
FROM UNIFIED_AUDIT_TRAIL
WHERE SQL_TEXT LIKE 'SELECT %';
```

For this SELECT statement, enter the text for the SQL_TEXT column ('SELECT %') using the same case that you used when you entered the SELECT statement for users OE, HR, and SH. In other words, if you entered that SELECT statement in lowercase letters, then enter 'select %' when you query the DBA_AUDIT_TRAIL view, not 'SELECT %'.

Output similar to the following appears:

| DBUSERNAME | SQL_TEXT | EVENT_TIMESTAMP |
|------------|-----------------------------------|------------------------------|
| OE | SELECT COUNT(*) FROM CUSTOMERS | 04-JAN-13 03.39.02.468963 PM |
| HR | SELECT COUNT(*) FROM OE.CUSTOMERS | 04-JAN-13 03.38.05.974127 PM |

Because the audit policy only applied to users OE and HR, there is no record for user SH's actions.

9.3.5 Step 5: Optionally, Remove the Components for This Tutorial

You can remove the components of this tutorial if you no longer need them.

To remove the audit settings for this tutorial:

1. In Enterprise Manager, ensure that you are logged in as user `sec_admin`.
2. Select Administration, then Security, and then Audit Settings to display the Audit Settings page.
3. Select the `SELECT_CUST_POL` policy, click **Disable**, and then select **Yes** in the Confirmation dialog box.
4. Select the `SELECT_CUST_POL` policy again, click **Delete**, and then select **Yes** in the Confirmation dialog box. Then click **Yes** again.
5. Log out of Enterprise Manager and then log back in again as user `SYS` with the `SYSDBA` administrative privilege.
6. From the **Schema** menu, select **Users**.
7. Select the `SEC_ADMIN` user and then click **Edit**.
8. In the Edit User : `SEC_ADMIN` page, select **Roles**.
9. Select the **Edit List** button.
10. In the Modify Roles page, move the `AUDIT_ADMIN` role to the Available Roles list and then click **OK**.
11. In the Edit User : `SEC_ADMIN` page, select the **Object Privileges** tab.
12. Select the `EXECUTE` privilege for the `DBMS_AUDIT_MGMT` package, and then click **Delete**.
13. Click **Apply**.

9.3.6 Step 6: Optionally, Remove the SEC_ADMIN Security Administrator Account

If you no longer need the `sec_admin` administrator account, then you should remove it.

To remove the sec_admin security administrator account:

1. Log in to the database as user `SYSTEM`.
2. From the Database home page, select **Schema**, then **Users**.
3. Select the user `sec_admin`.
4. Click **Delete**.
5. In the Confirmation page, select **Yes**.
6. Exit Enterprise Manager.

A

- access control
 - data encryption, [4-2](#)
 - Oracle Label Security, [8-1](#)
- administrative accounts
 - about, [2-12](#)
 - predefined, listed, [2-12](#)
- administrators
 - restricting access of, [5-1](#)
 - separation of duty, [5-1](#)
- ANONYMOUS user account, [2-12](#)
- application contexts
 - Oracle Virtual Private Database, used with, [6-1](#)
- ASMSNMP user account, [2-12](#)
- auditing
 - about, [9-1](#)
 - fine-grained auditing, [9-1](#)
 - monitoring user actions, [9-1](#)
 - reasons to audit, [9-2](#)
- AUDSYS user account, [2-12](#)
- AUTHID CURRENT USER invoker's rights clause, [3-7](#)

C

- CONNECT role, privilege available to, [3-2](#)
- CREATE DATABASE LINK statement, [3-2](#)
- CREATE EXTERNAL JOB privilege
 - default security setting, modified by, [2-2](#)
- CREATE SESSION statement, [3-2](#)
- CTXSYS user account, [2-12](#)

D

- data dictionary
 - about, [2-3](#)
 - securing, [2-4](#)
- data dictionary views
 - DBA_USERS_WITH_DEFPWD, [2-19](#)
- database
 - checking compatibility, [4-5](#)
- database accounts
 - See* user accounts

- DBA_USERS_WITH_DEFPWD data dictionary view,
 - [2-19](#)
- DBSNMP user account
 - about, [2-12](#)
- default passwords
 - importance of changing, [2-18](#)
- default security settings
 - about, [2-2](#)
- DIP user account, [2-15](#)

E

- encryption
 - about, [4-1](#)
 - algorithms, described, [2-7](#)
 - components, [4-1](#)
 - network, [2-6](#)
 - reasons not to encrypt, [4-2](#)
 - reasons to encrypt, [4-2](#)
- errors
 - checking trace files, [3-7](#)
 - WHEN NO_DATA_FOUND exception example,
 - [3-7](#)
- examples, [3-4](#)
 - See also* tutorials
- exceptions
 - WHEN NO_DATA_FOUND example, [3-7](#)

F

- fine-grained auditing, [9-1](#)

G

- GRANT ALL PRIVILEGES privilege, [2-4](#)
- guidelines for security
 - auditing
 - predefined unified audit policies, [9-1](#)
 - Oracle Label Security policies, planning, [8-4](#)
 - passwords
 - creating, [2-17](#)
 - privileges, granting, [3-1](#)
 - roles, granting to users, [3-2](#)

H

HR user account, [2-16](#)

I

initialization parameters

- configuration related, [2-5](#)
- default security, modified by, [2-2](#)
- FAILED_LOGIN_ATTEMPTS, [2-20](#)
- INACTIVE_ACCOUNT_TIME, [2-20](#)
- installation related, [2-5](#)
- modifying, [2-5](#)
- O7_DICTIONARY_ACCESSIBILITY
 - about, [2-5](#)
 - data dictionary, protecting, [2-4](#)
 - default setting, [2-4](#)
- OS_AUTHENT_PREFIX, [2-10](#)
- OS_ROLES, [3-13](#)
- PASSWORD_GRACE_TIME, [2-20](#)
- PASSWORD_LIFE_TIME, [2-20](#)
- PASSWORD_LOCK_TIME, [2-20](#)
- PASSWORD_REUSE_MAX, [2-20](#)
- PASSWORD_REUSE_TIME, [2-20](#)
- REMOTE_LISTENER, [2-10](#)
- REMOTE_OS_AUTHENT, [2-10](#)
- REMOTE_OS_ROLES, [2-10](#), [3-13](#)
- SEC_CASE_SENSITIVE_LOGIN, [2-20](#)
- SEC_MAX_FAILED_LOGIN_ATTEMPTS, [2-20](#)
- SEC_RETURN_SERVER_RELEASE_BANNER,
[2-5](#)
- SQL92_SECURITY, [3-13](#)

invoker's rights, [3-7](#)

IX user account, [2-16](#)

K

keystores

- closing, [4-7](#)
- creating, [4-4](#)
- creating master encryption key, [4-8](#)
- creating software password keystore, [4-6](#)
- opening, [4-7](#)

L

LBACSYS user account, [2-12](#)

least privilege principle, [3-1](#)

M

master encryption key

- creating, [4-8](#)

MDDATA user account, [2-15](#)

MDSYS user account, [2-12](#)

monitoring

- See* auditing

multitenant container databases

See CDBs

My Oracle Support

about, [x](#)

user account for logging service requests, [2-15](#)

N

network encryption

- about, [2-6](#)
- components, [2-6](#)
- configuring, [2-7](#)

nondatabase users, [6-1](#)

O

object privileges, [3-1](#)

OE user account, [2-16](#)

OLAPSYS user account, [2-12](#)

Oracle Data Redaction

- about, [7-1](#)
- compared with VPD and OLS, [8-2](#)
- industry compliance, [7-1](#)
- redaction performed in real time, [7-1](#)
- tutorial, [7-2](#)
- types of redaction, [7-1](#)

Oracle Database Vault

- about, [5-1](#)
- components, [5-1](#)
- registering with database, [5-3](#)
- regulatory compliances, how meets, [5-1](#)
- tutorial, [5-2](#)

Oracle Enterprise Manage

about, [1-2](#)

Oracle Label Security (OLS)

- compared with VPD and Data Redaction, [8-2](#)
- components, [8-1](#)
- guidelines in planning, [8-4](#)
- how it works, [8-1](#)
- tutorial, [8-6](#)
- used with Oracle Virtual Private Database, [8-2](#)

Oracle MetaLink

See My Oracle Support

Oracle Net

encrypting network traffic, [2-7](#)

Oracle Virtual Private Database (VPD)

- about, [6-1](#)
- advantages, [6-1](#)
- application contexts, [6-1](#)
- compared with OLS and Data Redaction, [8-2](#)
- components, [6-1](#)
- tutorial, [6-3](#)
- used with Oracle Label Security, [8-2](#)

ORACLE_OCM user account, [2-15](#)

ORDDATA user account, [2-12](#)

ORDPLUGINS user account, [2-12](#)

ORDSYS user account, [2-12](#)

P

- passwords
 - changing, [2-19](#)
 - default security setting, modified by, [2-2](#)
 - default user account, [2-18](#)
- passwords for security requirements, [2-17](#)
- pluggable databases
 - See PDBs
- PM user account, [2-16](#)
- principle of least privilege, [3-1](#)
- privileges
 - about, [3-1](#)
 - CREATE DATABASE LINK statement, [3-2](#)
 - SYSTEM and OBJECT, [3-1](#)

R

- roles
 - CONNECT, [3-2](#)
 - create your own, [3-2](#)
 - job responsibility privileges only, [3-2](#)

S

- SCOTT user
 - about, [2-16](#)
 - restricting privileges of, [3-2](#)
- sec_admin example security administrator
 - removing, [9-9](#)
- secure application roles
 - about, [3-3](#)
 - advantages, [3-3](#)
 - components, [3-3](#)
 - invoker's rights, [3-7](#)
 - tutorial, [3-4](#)
- security administrator
 - removing sec_admin, [9-9](#)
- security tasks, common, [1-2](#)
- SELECT ANY DICTIONARY privilege
 - GRANT ALL PRIVILEGES privilege, not included in, [2-4](#)
- sensitive data
 - Oracle Virtual Private Database, [6-1](#)
 - secure application roles, [3-3](#)
- separation of duty concepts, [3-6](#)
- separation-of-duty principles
 - about, [5-1](#)
 - Oracle Database Vault, [5-4](#)
- session information, retrieving, [6-1](#)
- SH user account, [2-16](#)
- SI_INFORMTN_SCHEMA user account, [2-12](#)
- SPATIAL_CSW_ADMIN_USR user account, [2-15](#)
- SPATIAL_WFS_ADMIN_USR user account, [2-15](#)
- standard auditing
 - tutorial, [9-3](#)

- SYS user account
 - about, [2-12](#)
- SYSBACKUP user account
 - about, [2-12](#)
- SYSDG user account
 - about, [2-12](#)
- SYSKM user account
 - about, [2-12](#)
- system administrator
 - See administrative accounts, security administrator
- system privileges, [3-1](#)
- SYSTEM user account
 - about, [2-12](#)

T

- tablespaces
 - encrypting, [4-11](#)
- TDE
 - See Transparent Data Encryption (TDE)
- trace files
 - checking for errors, [3-7](#)
- Transparent Data Encryption (TDE)
 - about, [4-2](#)
 - advantages, [4-2](#)
 - closing keystore, [4-7](#)
 - components, [4-2](#)
 - configuring, [4-4](#)
 - creating master encryption key, [4-8](#)
 - creating software password keystores, [4-6](#)
 - finding if keystore is open or closed, [4-13](#)
 - finding keystore location, [4-13](#)
 - finding keystore type, [4-13](#)
 - how it works, [4-2](#)
 - opening keystore, [4-7](#)
 - performance effects, [4-2](#)
 - storage space, [4-2](#)
 - table columns
 - checking in database instances, [4-14](#)
 - encrypting, [4-8](#)
 - tablespaces
 - checking, [4-14](#)
 - tablespaces, encrypting, [4-11](#)
- troubleshooting
 - checking trace files, [3-7](#)
- tutorials
 - Oracle Database Vault, [5-2](#)
 - Oracle Label Security, [8-6](#)
 - Oracle Virtual Private Database, [6-3](#)
 - secure application roles, [3-4](#)
 - standard auditing, [9-3](#)

U

- user accounts
 - about, [2-11](#)
 - default, changing password, [2-18](#)

user accounts (*continued*)
 expiring, [2-17](#)
 locking, [2-17](#)
 password requirements, [2-17](#)
 predefined
 administrative, [2-12](#)
 non-administrative, [2-15](#)
 sample schema, [2-16](#)
 unlocking, [2-17](#)
user accounts, predefined
 ANONYMOUS, [2-12](#)
 ASMSNMP, [2-12](#)
 AUDSYS, [2-12](#)
 CTXSYS, [2-12](#)
 DBSNMP, [2-12](#)
 DIP, [2-15](#)
 HR, [2-16](#)
 IX, [2-16](#)
 LBACSYS, [2-12](#)
 MDDATA, [2-15](#)
 MDSYS, [2-12](#)
 OE, [2-16](#)
 OLAPSYS, [2-12](#)
 ORACLE_OCM, [2-15](#)
 ORDDATA, [2-12](#)
 ORDPLUGINS, [2-12](#)
 ORDSYS, [2-12](#)
 PM, [2-16](#)
 SCOTT, [2-16](#), [3-2](#)
 SH, [2-16](#)

user accounts, predefined (*continued*)
 SI_INFORMTN_SCHEMA, [2-12](#)
 SPATIAL_CSW_ADMIN_USR, [2-15](#)
 SPATIAL_WFS_ADMIN_USR, [2-15](#)
 SYS, [2-12](#)
 SYSBACKUP, [2-12](#)
 SYSDG, [2-12](#)
 SYSKM, [2-12](#)
 SYSTEM, [2-12](#)
 WMSYS, [2-12](#)
 XDB, [2-12](#)
 XS\$NULL, [2-15](#)
user session information, retrieving, [6-1](#)

V

views
 See data dictionary views
Virtual Private Database
 See Oracle Virtual Private Database
VPD
 See Oracle Virtual Private Database

W

WMSYS user account, [2-12](#)

X

XDB user account, [2-12](#)
XS\$NULL user account, [2-15](#)