## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>ix</td>
</tr>
<tr>
<td>Audience</td>
<td>ix</td>
</tr>
<tr>
<td>Documentation Accessibility</td>
<td>ix</td>
</tr>
<tr>
<td>Related Documents</td>
<td>x</td>
</tr>
<tr>
<td>Conventions</td>
<td>xi</td>
</tr>
<tr>
<td>1 Introduction to Oracle Database Security</td>
<td>1-1</td>
</tr>
<tr>
<td>About This Guide</td>
<td>1-1</td>
</tr>
<tr>
<td>Before Using This Guide</td>
<td>1-1</td>
</tr>
<tr>
<td>What This Guide Is and Is Not</td>
<td>1-1</td>
</tr>
<tr>
<td>Common Database Security Tasks</td>
<td>1-2</td>
</tr>
<tr>
<td>Tools for Securing Your Database</td>
<td>1-2</td>
</tr>
<tr>
<td>Securing Your Database: A Roadmap</td>
<td>1-3</td>
</tr>
<tr>
<td>2 Securing the Database Installation and Configuration</td>
<td>2-1</td>
</tr>
<tr>
<td>About Securing the Database Installation and Configuration</td>
<td>2-1</td>
</tr>
<tr>
<td>Enabling the Default Security Settings</td>
<td>2-1</td>
</tr>
<tr>
<td>Securing the Oracle Data Dictionary</td>
<td>2-3</td>
</tr>
<tr>
<td>About the Oracle Data Dictionary</td>
<td>2-3</td>
</tr>
<tr>
<td>Enabling Data Dictionary Protection</td>
<td>2-4</td>
</tr>
<tr>
<td>Guidelines for Securing Operating System Access to Oracle Database</td>
<td>2-5</td>
</tr>
<tr>
<td>Guideline for Granting Permissions to Run-Time Facilities</td>
<td>2-5</td>
</tr>
<tr>
<td>Initialization Parameters Used for Installation and Configuration Security</td>
<td>2-6</td>
</tr>
<tr>
<td>Modifying the Value of an Initialization Parameter</td>
<td>2-6</td>
</tr>
<tr>
<td>3 Securing Oracle Database User Accounts</td>
<td>3-1</td>
</tr>
<tr>
<td>About Securing Oracle Database User Accounts</td>
<td>3-1</td>
</tr>
<tr>
<td>Predefined User Accounts Provided by Oracle Database</td>
<td>3-2</td>
</tr>
<tr>
<td>Predefined Administrative Accounts</td>
<td>3-3</td>
</tr>
<tr>
<td>Predefined Non-Administrative User Accounts</td>
<td>3-5</td>
</tr>
<tr>
<td>Predefined Sample Schema User Accounts</td>
<td>3-6</td>
</tr>
<tr>
<td>Expiring and Locking Database Accounts</td>
<td>3-7</td>
</tr>
<tr>
<td>Requirements for Creating Passwords</td>
<td>3-8</td>
</tr>
<tr>
<td>Finding and Changing Default Passwords</td>
<td>3-9</td>
</tr>
<tr>
<td>Guideline for Handling the Default Administrative User Passwords</td>
<td>3-10</td>
</tr>
</tbody>
</table>
Using Proxies to Audit SQL Statements and Privileges in a Multitier Environment 7-8
Individually Auditing Schema Objects ................................................................. 7-8
Auditing Network Activity ..................................................................................... 7-8

Tutorial: Creating a Standard Audit Trail ............................................................. 7-9
Step 1: Log In and Enable Standard Auditing ....................................................... 7-9
Step 2: Enable Auditing for SELECT Statements on the OE.CUSTOMERS Table .... 7-10
Step 3: Test the Audit Settings ............................................................................ 7-11
Step 4: Optionally, Remove the Components for This Tutorial ......................... 7-11
Step 5: Remove the SEC_ADMIN Security Administrator Account ..................... 7-12

Guidelines for Auditing ....................................................................................... 7-12
Guideline for Using Default Auditing of SQL Statements and Privileges ............. 7-12
Guidelines for Managing Audited Information .................................................. 7-13
Guidelines for Auditing Typical Database Activity .......................................... 7-13
Guidelines for Auditing Suspicious Database Activity ....................................... 7-14

Initialization Parameters Used for Auditing ....................................................... 7-15

Index
List of Tables

2–1 Default Security Settings for Initialization Parameters................................. 2-2
2–2 Initialization Parameters Used for Installation and Configuration Security .... 2-6
3–1 Predefined Oracle Database Administrative User Accounts ......................... 3-2
3–2 Predefined Oracle Database Non-Administrative User Accounts .................... 3-5
3–3 Default Sample Schema User Accounts............................................................ 3-7
3–4 Initialization and Profile Parameters Used for User Account Security ............ 3-12
4–1 Initialization Parameters Used for Privilege Security ........................................ 4-12
5–1 Initialization Parameters Used for Network Security ........................................ 5-8
6–1 Data Dictionary Views for Encrypted Tablespaces ........................................... 6-11
7–1 Initialization Parameters Used for Auditing ..................................................... 7-15
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.

The contents of this preface are as follows:
- Audience
- Documentation Accessibility
- Related Documents
- Conventions

**Audience**

Oracle Database 2 Day + Security 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**

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at
Accessibility of Code Examples in Documentation
Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation
This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services
Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, 7 days a week. For TTY support, call 800.446.2398. Outside the United States, call +1.407.458.2479.

Related Documents
For more information, use the following resources:

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/technology/index.html

If you are not already a member, you can register for free at:
http://www.oracle.com/technology/membership/

For security-specific information about OTN, visit:

For the latest version of the Oracle documentation, including this guide, visit:
Oracle Documentation Search Engine
To access the database documentation search engine directly, visit:
http://tahiti.oracle.com/

Oracle Store
Printed documentation is available for sale in the Oracle Store at:
https://oraclestore.oracle.com/OA_HTML/ibeCZzpHome.jsp

OracleMetaLink
You can find information about security patches, certifications, and the support knowledge base by visiting OracleMetaLink at:
https://metalink.oracle.com/

Conventions
The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>boldface</td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td>italic</td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
Introduction to Oracle Database Security

This chapter contains:
- About This Guide
- Common Database Security Tasks
- Tools for Securing Your Database
- Securing Your Database: A Roadmap

About This Guide

*Oracle Database 2 Day + Security Guide* teaches you how to perform day-to-day database security tasks. Its goal 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 in Database Console mode, featuring all the self-management capabilities introduced in Oracle Database.

This section contains the following topics:
- Before Using This Guide
- What This Guide Is and Is Not

Before Using This Guide

Before using this guide:
- Complete *Oracle Database 2 Day DBA*
- Obtain the necessary products and tools described in “Tools for Securing Your Database” on page 1-2

What This Guide Is and Is Not

*Oracle Database 2 Day + Security Guide* is task oriented. The objective of this guide is to describe why and when you need to perform 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*. 
Common Database Security Tasks

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.

Common Database Security Tasks

As a database administrator for Oracle Database, you should be involved in the following security-related tasks:

- 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.

Tools for Securing Your Database

To achieve the goals of securing your database, you need the following products, tools, and utilities:

- Oracle Database 11g Release 1 (11.1) Enterprise Edition
  Oracle Database 11g Release 1 (11.1) 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 Database Control
  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.

- 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 11g Release 1 (11.1) installation.
Securing Your Database: A Roadmap

To learn how to secure your database, you follow these general steps:

1. **Secure your Oracle Database installation and configuration.**
   Complete the tasks in Chapter 2, “Securing the Database Installation and Configuration” to secure access to an Oracle Database installation.

2. **Secure user accounts for your site.**
   Complete the tasks in Chapter 3, “Securing Oracle Database User Accounts”, which builds on *Oracle Database 2 Day DBA*, where you learned how to create user accounts. You learn the following:
   - How to expire, lock, and unlock user accounts
   - Guidelines to choose secure passwords
   - How to change a password
   - How to enforce password management
   - Why you need to encrypt passwords in Oracle Database tables

3. **Understand how privileges work.**
   Complete the tasks in Chapter 4, “Managing User Privileges”. 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

4. **Secure data as it travels across the network.**
   Complete the tasks in Chapter 5, “Securing the Network” to learn how to secure client connections and to configure network encryption.

5. **Encrypt sensitive data.**
   Complete the tasks in Chapter 6, “Securing Data”, in which you learn about the following:
   - How to use transparent data encryption to automatically encrypt database table columns and tablespaces
   - How to control data access with Oracle Virtual Private Database
   - How to enforce row-level security with Oracle Label Security
How to control system administrative access to sensitive data with Oracle Database Vault.

6. Configure auditing so that you can monitor the database activities.
Complete the tasks in Chapter 7, “Auditing Database Activity” to learn about standard auditing.
Securing the Database Installation and Configuration

This chapter contains:

- About Securing the Database Installation and Configuration
- Enabling the Default Security Settings
- Securing the Oracle Data Dictionary
- Guidelines for Securing Operating System Access to Oracle Database
- Guideline for Granting Permissions to Run-Time Facilities
- Initialization Parameters Used for Installation and Configuration Security

About Securing the Database Installation and Configuration

After you install Oracle Database, you should secure the database installation and configuration. The methods in this chapter describe commonly used ways to do this, 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
- Oracle Database Installation Guide for your platform

Enabling the Default Security Settings

When you create a new database or modify an existing database, you can use the Security Settings window in Database Configuration Assistant (DBCA) to enable or disable the default security settings. Oracle recommends that you enable these settings. These settings enable the following default security settings:

- Enables default auditing settings. See "Using Default Auditing for Security-Relevant SQL Statements and Privileges" on page 7-5 for detailed information.
- Creates stronger enforcements for new or changed passwords. "Requirements for Creating Passwords" on page 3-8 describes the new password requirements.
- Removes the CREATE EXTERNAL JOB privilege from PUBLIC. For greater security, grant the CREATE EXTERNAL JOB privilege only to SYS, database administrators, and those users who need it.
Enabling the Default Security Settings

Modifies initialization parameter settings. Table 2–1 lists the modified initialization parameter settings.

### Table 2–1 Default Security Settings for Initialization Parameters

<table>
<thead>
<tr>
<th>Setting</th>
<th>Previous Setting</th>
<th>New Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIT_TRAIL</td>
<td>NONE</td>
<td>DB</td>
</tr>
<tr>
<td>O7_DICTIONARY_ACCESSIBILITY</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>PASSWORD_GRACE_TIME</td>
<td>UNLIMITED</td>
<td>7</td>
</tr>
<tr>
<td>PASSWORD_LOCK_TIME</td>
<td>UNLIMITED</td>
<td>1</td>
</tr>
<tr>
<td>PASSWORD_LOGIN_FAILURES</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>PASSWORD_LIFE_TIME</td>
<td>UNLIMITED</td>
<td>180</td>
</tr>
<tr>
<td>PASSWORD_REUSE_MAX</td>
<td>UNLIMITED</td>
<td>UNLIMITED</td>
</tr>
<tr>
<td>PASSWORD_REUSE_TIME</td>
<td>UNLIMITED</td>
<td>UNLIMITED</td>
</tr>
<tr>
<td>REMOTE_OS_ROLES</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
</tbody>
</table>

To enable the default profile security settings using Database Configuration Assistant:

1. Start Database Configuration Assistant:
   - UNIX: Enter the following command at a terminal window:
     ```
     dbca
     ```
     Typically, dbca is in the `$ORACLE_HOME/bin` directory.
   - Windows: From the Start menu, click All Programs. Then click Oracle - ORACLE_HOME, then Configuration and Migration Tools, and then Database Configuration Assistant.
     Alternatively, you can start Database Configuration assistant at a command prompt:
     ```
     dbca
     ```
     As with UNIX, typically, dbca is in the `ORACLE_BASE\ORACLE_HOME\bin` directory.

2. In the Welcome window, click Next.
   The Operations window appears.

3. Select Configure Database Options, and then click Next.
   The Database window appears.

4. Select the database that you want to configure, and then click Next.
   The Security Settings window appears.

5. Select the Keep the enhanced 11g default security settings (recommended). These settings include enabling auditing and a new default password profile option.

6. Click Next.
   The Database Components window appears.
Securing the Oracle Data Dictionary

This section describes how you can secure the data dictionary. The data dictionary is a set of database tables that provide information about the database, such as schema definitions or default values.

This section contains:
- About the Oracle Data Dictionary
- Enabling Data Dictionary Protection

About the Oracle Data Dictionary

The Oracle data dictionary is a set of database tables that provides information about the database. A data dictionary has the following contents:

- 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
- The names of Oracle Database users
- Privileges and roles granted to each user
- 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. The data dictionary is structured in tables and views, just like other database data. All the data dictionary tables and views for a given database are owned by the user SYS. Connecting to the database with the SYSDBA privilege gives full access to the data dictionary. Oracle strongly recommends limiting access to the SYSDBA privilege to only those operations necessary such as patching and other administrative operations. The data dictionary central to every Oracle database.

You can use SQL statements to access the data dictionary. Because the data dictionary is read only if you do not connect with the SYSDBA privilege, you can issue only queries (SELECT statements) against its tables and views. 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. Oracle Database Reference provides a list of database views that you can query to find information about the data dictionary.

Example 2–1 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

7. Select any additional options, and then click Next. Answer the remaining questions as necessary.
8. Click Finish.
Securing the Oracle Data Dictionary

Enter password: password
Connected.

SQL> SELECT TABLE_NAME FROM DICTIONARY;

Enabling Data Dictionary Protection

You can protect the data dictionary by enabling the `O7_DICTIONARY_ACCESSIBILITY` initialization parameter. This 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, is typically 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 the `O7_DICTIONARY_ACCESSIBILITY` initialization parameter.

To enable data dictionary protection:

1. Start Oracle Enterprise Manager Database Control (Database Control).
   See Oracle Database 2 Day DBA for instructions about how to start Database Control.

2. Log in as `SYS` and connect with the `SYSDBA` privilege.
   - **User Name**: Enter the name of a user has administrative privileges. In this case, you enter SYS.
   - **Password**: Enter the user’s password.
   - **Connect As**: From the list, select either `SYSDBA`, `SYSOPER`, or `Normal`. In this case, you select `SYSDBA`.

   The Oracle Enterprise Manager Database Home page (Database Home page) appears.

3. Click **Server** to display the Server subpage.

4. In the Database Configuration section, click **Initialization Parameters**.
   The Initialization Parameters page appears.

5. In 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.

   Depending on the parameter, you may have to modify the value from the SPFile subpage. Click the **SFFile** tab to display the SPFile subpage.

6. Set the value for `O7_DICTIONARY_ACCESSIBILITY` to `FALSE`.

7. Click **Apply**.

8. Restart the Oracle Database instance.
   - Click the **Database Instance** link.
   - Click **Home** to display the Database Control home page.
   - Under General, click **Shutdown**.
d. In the Startup/Shutdown Credentials page, enter your credentials. See Oracle Database 2 Day DBA for more information.

e. After the shutdown completes, click Startup.

After you set the O7_DICTIONARY_ACCESSIBILITY parameter to FALSE, only users who have the SELECT ANY DICTIONARY privilege and those users authorized to make DBA-privileged (for example CONNECT / AS SYSDBA) connections can use the ANY system privilege on the data dictionary. If the O7_DICTIONARY_ACCESSIBILITY parameter is not set to FALSE, then any user with a DROP ANY TABLE (for example) system privilege can drop parts of the data dictionary.

---

**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 “Guideline for Granting Roles to Users” on page 4-2 and Oracle Database 2 Day DBA.

---

**Guidelines for Securing Operating System Access to Oracle Database**

You can secure access to Oracle Database on the operating system level by following these guidelines:

- Limit the number of operating system users.
- Limit the privileges of the operating system accounts (administrative, root-privileged, or DBA) on the Oracle Database host (physical computer). Only grant the user the least number of privileges needed to perform his or her tasks.
- Restrict the ability to modify the default file and directory permissions for the Oracle Database home (installation) directory or its contents. Even privileged operating system users and the Oracle owner should not modify these permissions, unless instructed otherwise by Oracle.
- Restrict symbolic links. Ensure that when you provide a path or file to the database, neither the file nor any part of the path is modifiable by an untrusted user. The file and all components of the path should be owned by the database administrator or some trusted account, such as root.

This recommendation applies to all types of files: data files, log files, trace files, external tables, BFILES, and so on.

**Guideline for Granting Permissions to Run-Time Facilities**

Many Oracle Database products use run-time facilities such as Oracle Java Virtual Machine (OJVM). Do not assign all permissions to a database run-time facility. Instead, grant specific permissions to the explicit document root file paths for facilities that might run files and packages outside the database.

Here is an example of a vulnerable run-time call, in which individual files are specified:

```
call dba_java.grant_permission('wsmith',
    'SYS:java.io.FilePermission','filename','read');
```
The following example is a better (more secure) run-time call, which specifies a directory path (in bold typeface) instead.

call dbms_java.grant_permission('wsmith', 'SYS:java.io.FilePermission', 'directory_path', 'read');

Initialization Parameters Used for Installation and Configuration Security

Table 2–2 lists initialization parameters that you can set to better secure your Oracle Database installation and configuration.

<table>
<thead>
<tr>
<th>Initialization Parameter</th>
<th>Default Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEC_RETURN_SERVER_RELEASE_BANNER</td>
<td>FALSE</td>
<td>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 Oracle Database Security Guide for more information about this and similar parameters. Oracle Database Reference describes this parameter in detail.</td>
</tr>
<tr>
<td>O7_DICTIONARY_ACCESSIBILITY</td>
<td>FALSE</td>
<td>Controls restrictions on SYSTEM privileges. See &quot;Enabling Data Dictionary Protection&quot; on page 2-4 for more information about this parameter. Oracle Database Reference describes this parameter in detail.</td>
</tr>
</tbody>
</table>

See Also: Oracle Database Reference for more information about initialization parameters

Modifying the Value of an Initialization Parameter

This section explains how to use Database Control to modify the value of an initialization parameter. To find detailed information about the initialization parameters available, see Oracle Database Reference.

To modify the value of an initialization parameter:
1. Start Database Control.
2. Log in as user SYS with the SYSDBA privilege.
   - **User Name:** SYS
   - **Password:** Enter your password.
   - **Connect As:** SYSDBA
3. Click Server to display the Server subpage.
4. In the Database Configuration section, click Initialization Parameters.
   The Initialization Parameters page appears.
5. 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.

Depending on the parameter, you might have to modify the value from the SPFile subpage. Click the SPFile tab to display the SPFile subpage.

6. In the Value field, either enter the new value or if a list is presented, select from the list.

7. Click Apply.

8. If the parameter is static, restart the Oracle Database instance.

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.

a. Click the Database Instance link.

b. Click Home to display the Database Control home page.

c. Under General, click Shutdown.

d. In the Startup/Shutdown Credentials page, enter your credentials.

See Oracle Database 2 Day DBA for more information.

e. After the shutdown completes, click Startup.
Securing Oracle Database User Accounts

This chapter contains:
- About Securing Oracle Database User Accounts
- Predefined User Accounts Provided by Oracle Database
- Expiring and Locking Database Accounts
- Requirements for Creating Passwords
- Finding and Changing Default Passwords
- Guideline for Handling the Default Administrative User Passwords
- Guideline for Enforcing Password Management
- Parameters Used to Secure User Accounts

See Also: Oracle Database Security Guide for detailed information about securing user accounts

About Securing Oracle Database User Accounts

You can use many methods to secure database user accounts. For example, Oracle Database has a set of built-in protections for passwords. This chapter explains how you can safeguard default database accounts and passwords, and describes 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 for your site, 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, lock, and unlock database accounts.

- **Managing passwords.** You can manage and protect passwords by using the tools provided with Oracle Database, such as initialization parameters.
Predefined User Accounts Provided by Oracle Database

When you install Oracle Database, the installation process creates a set of predefined accounts. These accounts are in the following categories:

- Predefined Administrative Accounts
- Predefined Non-Administrative User Accounts
- Predefined Sample Schema User Accounts

Predefined Administrative Accounts

A default Oracle Database installation provides a set of predefined administrative accounts. 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`.

To protect these accounts from unauthorized access, the installation process expires and locks most of these accounts, except where noted in Table 3–1. As the database administrator, you are responsible for unlocking and resetting these accounts, as described in "Expiring and Locking Database Accounts" on page 3-7.

Table 3–1 lists the administrative user accounts provided by Oracle Database.

<table>
<thead>
<tr>
<th>User Account</th>
<th>Description</th>
<th>Status After Installation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANONYMOUS</td>
<td>Account that allows HTTP access to Oracle XML DB. It is 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.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>CTXSYS</td>
<td>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.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>DBSNMP</td>
<td>The account used by the Management Agent component of Oracle Enterprise Manager to monitor and manage the database.</td>
<td>Open</td>
</tr>
</tbody>
</table>

See Also:

- Oracle Database Security Guide for detailed information about managing user accounts and authentication
- "Predefined User Accounts Provided by Oracle Database" on page 3-2 for a description of the predefined user accounts that are created when you install Oracle Database
### Predefined User Accounts Provided by Oracle Database

#### Securing Oracle Database User Accounts

<table>
<thead>
<tr>
<th>User Account</th>
<th>Description</th>
<th>Status After Installation</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXFSYS</td>
<td>The account used internally to access the EXFSYS schema, which is associated with the Rules Manager and Expression Filter feature. This feature enables you to build complex PL/SQL rules and expressions. The EXFSYS schema contains the Rules Manager and Expression Filter DDL, DML, and associated metadata. See Oracle Database Rules Manager and Expression Filter Developer’s Guide.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>LBACSYS</td>
<td>The account used to administer Oracle Label Security (OLS). It is created only when you install the Label Security custom option. See &quot;Enforcing Row-Level Security with Oracle Label Security&quot; on page 6-20 and Oracle Label Security Administrator’s Guide.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>MDSYS</td>
<td>The Oracle Spatial and Oracle Multimedia Locator administrator account. See Oracle Spatial Developer’s Guide.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>MGMT_VIEW</td>
<td>An account used by Oracle Enterprise Manager Database Control. Open Password is randomly generated at installation or database creation time. Users do not need to know this password.</td>
<td>Open</td>
</tr>
<tr>
<td>OLAPSYS</td>
<td>The account that owns the OLAP Catalog (CWMLite). This account has been deprecated, but is retained for backward compatibility.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>ORDSYS</td>
<td>The account for administrating the Oracle Warehouse Builder repository. Access this account during the installation process to define the base language of the repository and to define Warehouse Builder workspaces and users. A data warehouse is a relational or multidimensional database that is designed for query and analysis. See Oracle Warehouse Builder Installation and Administration Guide.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>ORDPLUGINS</td>
<td>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. See Oracle Multimedia User’s Guide and Oracle Multimedia Reference.</td>
<td>Expired and locked</td>
</tr>
</tbody>
</table>
Predefined User Accounts Provided by Oracle Database

<table>
<thead>
<tr>
<th>User Account</th>
<th>Description</th>
<th>Status After Installation</th>
</tr>
</thead>
<tbody>
<tr>
<td>OUTLN</td>
<td>The account that supports plan stability. Plan stability prevents certain database environment changes from affecting the performance characteristics of applications by preserving execution plans in stored outlines. OUTLN acts as a role to centrally manage metadata associated with stored outlines. See Oracle Database Performance Tuning Guide.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>SYS</td>
<td>An account used to perform database administration tasks. See Oracle Database 2 Day DBA.</td>
<td>Open</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>An account used to perform Oracle Enterprise Manager database administration tasks. The SYS and SYSTEM accounts can also perform these tasks. See Oracle Enterprise Manager Grid Control Installation and Basic Configuration.</td>
<td>Open</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>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 Oracle Database 2 Day DBA.</td>
<td>Open</td>
</tr>
<tr>
<td>TSMSYS</td>
<td>An account used for transparent session migration (TSM).</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>WK_TEST</td>
<td>The instance administrator for the default instance, WK_INST. After you unlock this account and assign this user a password, then you must also update the cached schema password using the administration tool Edit Instance Page. Ultra Search provides uniform search-and-location capabilities over multiple repositories, such as Oracle databases, other ODBC compliant databases, IMAP mail servers, HTML documents managed by a Web server, files on disk, and more. See Oracle Ultra Search Administrator’s Guide.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>WKSYS</td>
<td>An Ultra Search database super-user. WKSYS can grant super-user privileges to other users, such as WK_TEST. All Oracle Ultra Search database objects are installed in the WKSYS schema. See Oracle Ultra Search Administrator’s Guide.</td>
<td>Expired and locked</td>
</tr>
</tbody>
</table>
Predefined Non-Administrative User Accounts

Table 3–2 lists default non-administrative user accounts that are created when you install Oracle Database. Non-administrative user accounts only have the minimum privileges needed to perform their jobs. Their default tablespace is USERS.

To protect these accounts from unauthorized access, the installation process locks and expires these accounts immediately after installation, except where noted in Table 3–2.

As the database administrator, you are responsible for unlocking and resetting these accounts, as described in "Expiring and Locking Database Accounts" on page 3-7.

<table>
<thead>
<tr>
<th>User Account</th>
<th>Description</th>
<th>Status After Installation</th>
</tr>
</thead>
<tbody>
<tr>
<td>APEX_PUBLIC_USER</td>
<td>The Oracle Database Application Express account. Use this account to specify the Oracle schema used to connect to the database through the database access descriptor (DAD). Oracle Application Express is a rapid, Web application development tool for Oracle Database. See Oracle Database Application Express User’s Guide.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>DIP</td>
<td>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. See Oracle Label Security Administrator’s Guide.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>FLONG_30000</td>
<td>The account that owns most of the database objects created during the installation of Oracle Database Application Express. These objects include tables, views, triggers, indexes, packages, and so on. See Oracle Database Application Express User’s Guide.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>FLONG_FILES</td>
<td>The account that owns the database objects created during the installation of Oracle Database Application Express related to modplsql document conveyance, for example, file uploads and downloads. These objects include tables, views, triggers, indexes, packages, and so on. See Oracle Database Application Express User’s Guide.</td>
<td>Expired and locked</td>
</tr>
</tbody>
</table>
Predefined User Accounts Provided by Oracle Database

<table>
<thead>
<tr>
<th>User Account</th>
<th>Description</th>
<th>Status After Installation</th>
</tr>
</thead>
<tbody>
<tr>
<td>MDDATA</td>
<td>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.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>ORACLE_OCM</td>
<td>The account used with Oracle Configuration Manager. This feature enables you to associate the configuration information for the current Oracle Database instance with Oracle MetaLink. When you log a service request, it is associated with the database instance configuration information.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>Account used for the PUBLIC user group.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>SPATIAL_CSW_ADMIN_USR</td>
<td>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 main memory for the record types that are cached.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>SPATIAL_WFS_ADMIN_USR</td>
<td>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.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>XS$NULL</td>
<td>An internal account that represents the absence of a user in a session. Because XS$NULL is not a user, this account can only be accessed by the Oracle Database instance. XS$NULL has no privileges and no one can authenticate as XS$NULL, nor can authentication credentials ever be assigned to XS$NULL.</td>
<td>Expired and locked</td>
</tr>
</tbody>
</table>
In addition to the sample schema accounts, Oracle Database provides another sample schema account, SCOTT. The SCOTT schema contains the tables EMP, DEPT, SALGRAD, 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.

### Table 3-3 Default Sample Schema User Accounts

<table>
<thead>
<tr>
<th>User Account</th>
<th>Description</th>
<th>Status After Installation</th>
</tr>
</thead>
<tbody>
<tr>
<td>BI</td>
<td>The account that owns the BI (Business Intelligence) schema included in the Oracle Sample Schemas. See also Oracle Warehouse Builder User’s Guide.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>HR</td>
<td>The account used to manage the HR (Human Resources) schema. This schema stores information about the employees and the facilities of the company.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>OE</td>
<td>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.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>PM</td>
<td>The account used to manage the PM (Product Media) schema. This schema contains descriptions and detailed information about each product sold by the company.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>IX</td>
<td>The account used to manage the IX (Information Exchange) schema. This schema manages shipping through business-to-business (B2B) applications.</td>
<td>Expired and locked</td>
</tr>
<tr>
<td>SH</td>
<td>The account used to manage the SH (Sales) schema. This schema stores business statistics to facilitate business decisions.</td>
<td>Expired and locked</td>
</tr>
</tbody>
</table>

In addition to the sample schema accounts, Oracle Database provides another sample schema account, SCOTT. The SCOTT schema contains the tables EMP, DEPT, SALGRAD, 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.

### Expiring and Locking Database Accounts

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

When you expire the password of a user, that password no longer exists. If you want to unexpire the password, you change the password of that account. Locking an account preserves the user password, as well as 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.

**To expire and lock a database account:**

1. Start Database Control.
   
   See *Oracle Database 2 Day DBA* for instructions about how to start Database Control.

2. Log in with administrative privileges.
   
   For example:

   ![Login to Database screen](image_url)

   - **User Name:** system
   - **Password:** ***
   - **Connect As:** [Your username]
   - **Submit** button

---

Securing Oracle Database User Accounts 3-7
The Database Home page appears.

3. Click Server to display the Server subpage.

4. In the Security section, click Users.

   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.

5. In the Select column, select the account you want to expire, and then click Edit.

   The Edit User page appears.

6. 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” on page 3-8 for password requirements.
   - To lock the account, select Locked.

7. Click Apply.

Requirements for Creating 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.

At a minimum, passwords must be no longer than 30 characters. However, for greater security, follow these additional guidelines:

- Make the password between 10 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’s 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.
Finding and Changing Default Passwords

In Oracle Database, database user accounts, including administrative accounts, are installed without default passwords. During installation, you either create a password for the account (always an administrative account), or Oracle Database installs the default accounts, such as those in the sample schemas, locked with their passwords expired. If you have upgraded from a previous release of Oracle Database, you may have database accounts that 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.

To find and change default passwords:

1. Log into SQL*Plus 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

See Also:

- "Finding and Changing Default Passwords" on page 3-9 for information about changing user passwords
- "Expanding and Locking Database Accounts" on page 3-7 for information about_locking accounts and expiring passwords
- "Predefined User Accounts Provided by Oracle Database" on page 3-2 for a description of the predefined user accounts that are created when you install Oracle Database
- Oracle Database 2 Day DBA for an introduction to password policies
- Oracle Database Security Guide for detailed information about managing passwords

Finding and Changing Default Passwords

In Oracle Database, database user accounts, including administrative accounts, are installed without default passwords. During installation, you either create a password for the account (always an administrative account), or Oracle Database installs the default accounts, such as those in the sample schemas, locked with their passwords expired. If you have upgraded from a previous release of Oracle Database, you may have database accounts that 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.

To find and change default passwords:

1. Log into SQL*Plus 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

Securing Oracle Database User Accounts 3-9
Guideline for Handling the Default Administrative User Passwords

Replace *password* with a password that is secure, according to the guidelines listed in "Requirements for Creating Passwords" on page 3-8. 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;
```

You can use Database Control to change a user account passwords (not just the default user account passwords) if you have administrative privileges. Individual users can also use Database Control to change their own passwords.

**To use Database Control to change the password of a database account:**

1. Start Database Control.
   
   See *Oracle Database 2 Day DBA* for instructions about how to start Database Control.

2. Enter an administrator user name and password (for example, `SYSTEM`), and then click **Login**.

3. Click **Server** to display the Server subpage.

4. In the Security section, click **Users**.
   
   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.

5. In the Select column, select the account you want to change, and then click **Edit**.
   
   The Edit User page appears.

6. Enter a new password in the **Enter Password** and **Confirm Password** fields.

7. Click **Apply**.

**See Also:**

- Oracle Database Security Guide for additional methods of configuring password protection
- "Predefined User Accounts Provided by Oracle Database" on page 3-2

Guideline for Handling the Default Administrative User Passwords

You can use the same or different passwords for the `SYS`, `SYSTEM`, `SYSMAN`, and `DBSNMP` administrative accounts. Oracle recommends that you use different passwords for each. In any Oracle Database environment (production or test), assign strong, secure, and distinct passwords to these administrative accounts. If you use Database Configuration Assistant to create a new database, then it requires you to create passwords for the `SYS` and `SYSTEM` accounts.

Similarly, for production environments, do not use default passwords for any administrative accounts, including `SYSMAN` and `DBSNMP`. Oracle Database 11g Release 1 (11.1) and later does not install these accounts with default passwords, but if you have upgraded from an earlier release of Oracle Database, you may still have accounts that use default passwords. You should find and change these accounts by using the procedures in "Finding and Changing Default Passwords" on page 3-9.
At the end of database creation, Database Configuration Assistant displays a page that requires you to enter and confirm new passwords for the SYS and SYSTEM user accounts.

After installation, you can use Database Control to change the administrative user passwords. See “Finding and Changing Default Passwords” on page 3-9 for more information on changing a password.

**Guideline for Enforcing Password Management**

Apply basic password management rules (such as password length, history, complexity, and so forth) to all user passwords. Oracle Database has password policies enabled for the default profile. ‘Requirements for Creating Passwords’ on page 3-8 provides guidelines for creating password policies. Table 3-4 on page 3-12 lists initialization parameters that you can set to enforce password management.

You can find information about user accounts by querying the DBA_USERS view. This view contains a column for passwords, but for stronger security, Oracle Database encrypts the data in this column. The DBA_USERS view provides useful information such as the user account status, whether or not the account is locked, and password versions. You can query DBA_USERS as follows:

```sql
SQLPLUS SYSTEM
Enter password: password
Connected.
SQL> SELECT * FROM DBA_USERS;
```

Oracle also recommends, if possible, using Oracle Advanced Security (an option to Oracle Database Enterprise Edition) with network authentication services (such as Kerberos), token cards, smart cards, or X.509 certificates. These services provide strong authentication of users, and provide better protection against unauthorized access to Oracle Database.

**See Also:**
- Oracle Database Security Guide for more information about password management
- Oracle Database Security Guide for additional views you can query to find information about users and profiles
- Oracle Database Advanced Security Administrator’s Guide for more information about Oracle Database Advanced Security

**Parameters Used to Secure User Accounts**

Table 3–4 lists initialization and profile parameters that you can set to better secure user accounts.
### Parameters Used to Secure User Accounts

#### Table 3–4 Initialization and Profile Parameters Used for User Account Security

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEC_CASE_SENSITIVE_LOGON</td>
<td>TRUE</td>
<td>Controls case sensitivity in passwords. TRUE enables case sensitivity; FALSE disables it.</td>
</tr>
<tr>
<td>SEC_MAX_FAILED_LOGIN_ATTEMPTS</td>
<td>No default setting</td>
<td>Sets the maximum number of times a user is allowed to fail when connecting to an application.</td>
</tr>
<tr>
<td>FAILED_LOGIN_ATTEMPTS</td>
<td>10</td>
<td>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.</td>
</tr>
<tr>
<td>PASSWORD_GRACE_TIME</td>
<td>7</td>
<td>Sets the number of days that a user has to change his or her password before it expires.</td>
</tr>
<tr>
<td>PASSWORD_LIFE_TIME</td>
<td>180</td>
<td>Sets the number of days the user can use his or her current password.</td>
</tr>
<tr>
<td>PASSWORD_LOCK_TIME</td>
<td>1</td>
<td>Sets the number of days an account will be locked after the specified number of consecutive failed login attempts.</td>
</tr>
<tr>
<td>PASSWORD_REUSE_MAX</td>
<td>UNLIMITED</td>
<td>Sets the number of days before which a password cannot be reused.</td>
</tr>
<tr>
<td>PASSWORD_REUSE_TIME</td>
<td>UNLIMITED</td>
<td>Sets the number of password changes required before the current password can be reused.</td>
</tr>
</tbody>
</table>

**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” on page 2-6. For detailed information about initialization parameters, see Oracle Database Reference and Oracle Database Administrator’s Guide.
Managing User Privileges

This chapter contains:

- About Privilege Management
- Guideline for Granting Privileges
- Guideline for Handling Privileges for the PUBLIC User Group
- Guideline for Granting Roles to Users
- Controlling Access to Applications with Secure Application Roles
- Initialization Parameters Used for Privilege Security

See Also:
- Oracle Database Security Guide
- Oracle Label Security Administrator’s Guide

About Privilege Management

You can control user privileges in the following ways:

- **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.

Guideline for Granting Privileges

Because privileges are the rights to perform a specific action, such as updating or deleting a table, do not provide database users more privileges than are necessary. For an introduction to managing privileges, see “About User Privileges and Roles” in 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:

Managing User Privileges 4-1
Guideline for Handling Privileges for the PUBLIC User Group

- 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.

Guideline for Handling Privileges for the PUBLIC User Group

You should revoke unnecessary privileges and roles from the database server user group PUBLIC. PUBLIC acts as a default role granted to every user in an Oracle database. Any database user can exercise privileges that are granted to PUBLIC. These privileges include EXECUTE on various PL/SQL packages, potentially enabling someone with minimal privileges to access and execute functions that this user would not otherwise be permitted to access directly.

Guideline for Granting 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 "Administering Roles" in Oracle Database 2 Day DBA. In addition, see "Example: Creating a Role" in Oracle Database 2 Day DBA.

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.

For example, it is imperative to strictly limit the privileges of user SCOTT, because this is a well known default user account that may be vulnerable to intruders. Because the CREATE DBLINK privilege allows access from one database to another, drop its privilege for SCOTT. Then, drop the entire role for the user, because privileges acquired through a role cannot be dropped individually. Recreate your own role with only the privileges needed, and grant that new role to that user. Similarly, for even better security, drop the CREATE DBLINK privilege from all users who do not require it.

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. The PL/SQL package itself reflects the security policies necessary to control access to the application.

This section contains:
- About Secure Application Roles
- Tutorial: Creating a Secure Application Role
About Secure Application Roles

A secure application role is a role that can be enabled only by an authorized PL/SQL package. 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 that 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.
  
  Do not grant the role directly to the user; the PL/SQL package will do that for you. However, if the policy for your site is to grant roles to users, you can grant the secure application role to the user if you alter the user account to not have any default roles. For example:
  ```sql
  ALTER USER psmith DEFAULT ROLE NONE;
  ```

- **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. Invoker’s rights enable the user to have `EXECUTE` privileges on all objects that the package accesses. An invoker’s right procedure executes with the privileges of the current user, that is, the user who invokes the procedure. These 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 `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 `EXECUTE` privileges 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.
Controlling Access to Applications with Secure Application Roles

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 this table are defined in the EMPLOYEE_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.

In this tutorial:
- Step 1: Create a Security Administrator Account
- Step 2: Create User Accounts for This Tutorial
- Step 3: Create the Secure Application Role
- Step 4: Create a Lookup Table
- Step 5: Create the PL/SQL Procedure to Set the Secure Application Role
- Step 6: Grant EXECUTE Privileges for the Procedure to Matthew and Winston
- Step 7: Test the EMPLOYEE_ROLE Secure Application Role
- Step 8: Optionally, Remove the Components for This Tutorial

Step 1: 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. Start Database Control.
   - See Oracle Database 2 Day DBA for instructions about how to start Database Control.
2. Enter an administrator user name (for example, SYSTEM) and password, and then click Login.
   - The Database Home page appears.
3. Click Server to display the Server subpage.
   - The Users page appears.
5. Click Create.
   - The Create User page appears.
6. 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" on page 3-8.
   - Default Tablespace: SYSTEM
   - Temporary Tablespace: TEMP
   - Status: UNLOCKED
7. Click **System Privileges** to display the System Privileges subpage.
8. Click **Edit List**.
   The Modify System Privileges page appears.
9. 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
   - SELECT ANY DICTIONARY
10. Click **OK**.
11. Under Admin Option, do not select the boxes.
12. Click **OK**.

**Step 2: Create User Accounts for This Tutorial**
Matthew and Winston both are sample employees in the HR. EMPLOYEES schema, which provides columns for the manager ID and e-mail 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 Database Control, select the **Database Instance** link to display the Database Home page.
   If you are not logged in to Database Control, see Oracle Database 2 Day DBA for instructions about how to start Database Control. In the Login page, enter an administrator user name (for example, **SYSTEM**) and password, and then click **Login**.
2. Click **Server** to display the Server subpage.
3. Under Security, select **Users**.
   The Users page appears.
4. Click **Create**.
   The Create User page appears.
5. 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’ on page 3-8.
   - **Default Tablespace**: USERS
   - **Temporary Tablespace**: TEMP
   - **Status**: Unlocked
6. Click **System Privileges** to display the System Privileges subpage.
7. Click Edit List.  
   The Modify System Privileges page appears.
8. In 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.  
    The Users page appears.
11. Select 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" on page 3-8.
13. Click OK.  
    You do not need to grant wtaylor the CREATE SESSION privilege, because the Create Like action has done this for you.
14. Exit Database Control.

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

Step 3: Create the Secure Application Role

Now, you are ready to create the employee_role secure application role. To do so, you must log on as the security administrator sec_admin. "Step 1: Create a Security Administrator Account" on page 4-4 explains how to create the sec_admin account.

To create the secure application role:
1. Start SQL*Plus and log on as the security administrator sec_admin.  
   SQLPLUS sec_admin  
   Enter password: password  
   SQL*Plus starts, connects to the default database, and then displays a prompt.
   SQL>
   For detailed information about starting SQL*Plus, see Oracle Database 2 Day DBA.

2. Create the following secure application role:
   CREATE ROLE employee_role IDENTIFIED USING sec_roles;
   The IDENTIFIED USING clause sets the role to be enabled (or disabled) only within the associated PL/SQL package, in this case, sec_roles. At this stage, the sec_roles PL/SQL package does not need to exist.

3. Connect as user OE.
Controlling Access to Applications with Secure Application Roles

4. Enter the following statement to grant the EMPLOYEE_ROLE role SELECT privileges on the OE.ORDERS table:

   GRANT SELECT ON OE.ORDERS TO employee_role;

   Do not grant the role directly to the user. The PL/SQL package will do that for you, assuming the user passes its security policies. If your site requires that you directly grant users the role, then you must disable the role for that user. This is because the role must be initially disabled before the security policies in the package can begin performing their checks. For example, to disable the role for user wsmith (assuming wsmith was granted the role in the first place), enter the following statement:

   ALTER USER wsmith DEFAULT ROLE NONE;

Step 4: Create a Lookup Table

You are almost ready to create the procedure that determines who is granted the employee_role role. The procedure will grant the employee_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 use an existing application table as the lookup table. For this tutorial, you create your own lookup table that only contains the employee names, employee IDs, and their manager IDs.

To create the HR.VERIFY lookup table:

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” on page 3-8.

   CONNECT sys/as sysdba
   Enter password: sys_password
   PASSWORD HR
   Changing password for HR
   New password: password
   Retype new password: password
   Password changed.

   CONNECT hr
   Enter password: password
2. Enter the following CREATE TABLE SQL statement to create the lookup table:

```
CREATE table hr_verify AS
SELECT employee_id, first_name, last_name, email, manager_id
FROM employees;
```

3. Grant EXECUTE privileges for this table to mweiss and wtaylor by entering the following SQL statements:

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

Step 5: Create the PL/SQL Procedure to Set the Secure Application Role

Now, you are ready to 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, create them in a package.

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 SQL*Plus, connect as user sec_admin.

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

2. Enter the following CREATE PROCEDURE statement to create the secure application role procedure:
In this example:

- Line 1: Appends the `AUTHID CURRENT_USER` clause to the `CREATE PROCEDURE` statement, which creates the procedure using invoker’s rights. The `AUTHID CURRENT_USER` clause creates the package using invoker’s rights, using the privileges of the current user.

  You must create the package using invoker’s rights for the package to work. Invoker’s rights allow the user to have `EXECUTE` privileges on all objects that the package 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" on page 4-3 for information about the importance of creating the procedure using invoker’s rights.

- Line 3: Declares the `v_user` variable, which will store the user session information.

- Line 4: Declares the `v_manager_id` variable, which will store the manager’s ID of the `v_user` user.

- Line 6: 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_user`), and write 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`. 

```sql
CREATE OR REPLACE procedure sec_roles 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 employee_role';
  ELSE NULL;
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN v_manager_id:=0;
  DBMS_OUTPUT.PUT_LINE(v_manager_id);
END;
/
```
Controlling Access to Applications with Secure Application Roles

- Lines 7–8: 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` table for the manager ID of the current user.

- Lines 9–13: Use an IF condition to test whether or not the user should be granted the `sec_roles` 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.

- Lines 10–12: Within the IF condition, the THEN condition grants the role by executing immediately the `SET ROLE` statement. Otherwise, its ELSE condition denies the grant.

- Lines 14–15: Use an EXCEPTION statement to set `v_manager_id` to 0 if no data is found.

- Line 16: Copies the manager’s ID into a buffer so that it is readily available.

Step 6: Grant EXECUTE Privileges for the Procedure to Matthew and Winston

At this stage, Matthew and Winston can try to access the `OE.ORDERS` table, but they are not able to, even if they should. The next step is to grant them EXECUTE privileges on the `sec_roles` procedure, so that the `sec_roles` 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 `sec_roles` procedure:

- In SQL*Plus, as user `sec_admin`, enter the following GRANT SQL statements:
  
  ```sql
  GRANT EXECUTE ON sec_admin.sec_roles TO mweiss;
  GRANT EXECUTE ON sec_admin.sec_roles TO wtaylor;
  ```

Step 7: Test the EMPLOYEE_ROLE Secure Application Role

You are ready to test the `employee_role` secure application role by logging 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 `sec_roles` procedure must be executed for the role verification to take place.

To test the `employee_role` secure application role, as user `MWEISS`:

1. Connect as user `mweiss`.
   
   ```sql
   CONNECT mweiss
   Enter password: password
   ```

2. Enter the following SQL statement to run the `sec_roles` procedure:
   
   ```sql
   EXEC sec_admin.sec_roles;
   ```

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

3. Perform the following `SELECT` statement on the `OE.ORDERS` table:
   
   ```sql
   SELECT count(*) FROM oe.orders;
   ```

   Matthew has access to the `OE.ORDERS` table:
   
   ```sql
   COUNT(*)
   ```
Now, Winston will try to access the secure application.

**To test the employee_role secure application role as user WTAYLOR:**

1. In SQL*Plus, connect as user wtaylor.
   
   ```sql
   CONNECT wtaylor
   Enter password: password
   ```

2. Enter the following SQL statement to run the sec_roles procedure:
   
   ```sql
   EXEC sec_admin.sec_roles;
   ```
   
   This statement executes the sec_roles procedure for the current session.

3. Perform the following `SELECT` statement on the OE.ORDERS table:
   
   ```sql
   SELECT count(*) FROM oe.orders;
   ```
   
   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.

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

**Step 8: Optionally, Remove the Components for This Tutorial**

Remove the components that you created for this tutorial.

**To remove the components:**

1. In SQL*Plus, connect as SYS with the SYSDBA privilege.
   
   ```sql
   CONNECT SYS/AS SYSDBA
   Enter password: password
   ```

2. Enter the following `DROP` statements:
   
   ```sql
   DROP USER mweiss;
   DROP USER wtaylor;
   ```
   
   Do not drop user sec_admin. You will need this user for other tutorials in this guide.

3. In SQL*Plus, connect as user sec_admin.
   
   ```sql
   CONNECT sec_admin
   Enter password: password
   ```

4. Enter the following `DROP` SQL statements:
   
   ```sql
   DROP ROLE employee_role;
   DROP PROCEDURE sec_roles;
   ```

5. Connect as user HR, and then drop the HR_VERIFY table.
   
   ```sql
   CONNECT HR
   Enter password: password
   DROP TABLE hr_verify;
   ```

### Initialization Parameters Used for Privilege Security

Table 4–1 lists initialization parameters that you can use to secure user privileges.

<table>
<thead>
<tr>
<th>Initialization Parameter</th>
<th>Default Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>O7_DICTIONARY_ACCESSIBILITY</td>
<td>FALSE</td>
<td>Controls restrictions on SYSTEM privileges. See &quot;Enabling Data Dictionary Protection&quot; on page 2-4 for more information about this parameter.</td>
</tr>
<tr>
<td>OS_ROLES</td>
<td>FALSE</td>
<td>Determines whether Oracle or the operating system identifies and manages the roles of each user name.</td>
</tr>
<tr>
<td>MAX_ENABLED_ROLES</td>
<td>30</td>
<td>Specifies the maximum number of database roles that users can enable, including roles contained within other roles.</td>
</tr>
<tr>
<td>REMOTE_OS_ROLES</td>
<td>FALSE</td>
<td>Specifies whether or not operating system roles are allowed for remote clients. The default value, FALSE, causes Oracle to identify and manage roles for remote clients.</td>
</tr>
<tr>
<td>SQL92_SECURITY</td>
<td>FALSE</td>
<td>Specifies whether or not users must be granted the SELECT object privilege to execute UPDATE or DELETE statements.</td>
</tr>
</tbody>
</table>

To modify an initialization parameter, see "Modifying the Value of an Initialization Parameter" on page 2-6. For detailed information about initialization parameters, see Oracle Database Reference and Oracle Database Administrator’s Guide.
This chapter contains:

■ About Securing the Network
■ Securing the Client Connection on the Network
■ Protecting Data on the Network by Using Network Encryption
■ Initialization Parameters Used for Network Security

About Securing the Network
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. This chapter explains how you can encrypt data as it travels through the network, and also provides guidelines that you can follow to secure the network connections for Oracle Database.

Securing the Client Connection on the Network
This section describes how you can improve security for the client connection to ensure thorough protection. Using SSL is an essential element in these lists, enabling strict security for authentication and communications.

These guidelines are as follows:

■ Guidelines for Securing Client Connections
■ Guidelines for Securing the Network Connection

Guidelines for Securing Client Connections
Because authenticating client computers is problematic over the Internet, typically, user authentication is performed instead. This approach avoids client system issues that include falsified IP addresses, compromised operating systems or applications, and falsified or stolen client system identities. Nevertheless, the following guidelines improve the security of client connections:

1. **Enforce access controls effectively and authenticate clients stringently.**

   By default, Oracle allows operating system-authenticated logins only over secure connections, which precludes using Oracle Net and a shared server configuration. This default restriction prevents a remote user from impersonating another operating system user over a network connection.
Setting the initialization parameter `REMOTE_OS_AUTHENT` to `TRUE` forces the database to accept the client, operating-system user name received over a nonsecure connection and use it for account access. (To modify an initialization parameter, see “Modifying the Value of an Initialization Parameter” on page 2-6.) Because clients, such as PCs, are not trusted to perform operating-system authentication properly, it is poor security practice to use this feature.

The default setting, `REMOTE_OS_AUTHENT = FALSE`, creates a more secure configuration that enforces proper, server-based authentication of clients connecting to an Oracle database.

Do not alter the default setting of the `REMOTE_OS_AUTHENT` initialization parameter, which is `FALSE`.

Setting this parameter to `FALSE` does not mean that users cannot connect remotely. It means that the database will not trust that the client has already authenticated, and will apply its standard authentication processes.

2. **Configure the connection to use Secure Sockets Layer (SSL).**

   Using SSL communication makes eavesdropping difficult and enables the use of certificates for user and server authentication. To learn how to configure SSL, see Oracle Database Advanced Security Administrator’s Guide.

3. **Set up certificate authentication for clients and servers.**

   See Oracle Database Advanced Security Administrator’s Guide for more information about ways to manage certificates.

4. **Monitor the users who access your systems.**

   Authenticating client computers over the Internet is problematic. Perform user authentication instead, which avoids client-system issues that include falsified IP addresses, hacked operating systems or applications, and falsified or stolen client-system identities. The following steps improve client computer security:

   a. Configure the connection to use Secure Sockets Layer (SSL). Using SSL communication makes eavesdropping unfruitful, and enables the use of certificates for user and server authentication. To learn how to configure SSL, see Oracle Database Advanced Security Administrator’s Guide.

   b. Set up certificate authentication for clients and servers so that:

      - The organization is identified by unit and certificate issuer, and the user is identified by distinguished name and certificate issuer.
      - Applications test for expired certificates.
      - Certificate revocation lists are audited.

   See Oracle Database Advanced Security Administrator’s Guide for more information about ways to manage certificates.

**Guidelines for Securing the Network Connection**

Protecting the network and its traffic from inappropriate access or modification is the essence of network security. You should consider all paths the data travels, and assess the threats that impinge on each path and node. Then, take steps to lessen or eliminate those threats and the consequences of a breach of security. In addition, monitor and audit to detect either increased threat levels or successful penetration.
To manage network connections, you can use Oracle Net Manager. For an introduction to using Oracle Net Manager, see Oracle Database 2 Day DBA. See also Oracle Database Net Services Administrator’s Guide.

The following practices improve network security:

1. **Monitor listener activity.**

   You can monitor listener activity by using Oracle Enterprise Manager Database Control. In the Database Control home page, under General, click the link for your listener. The Listener page appears. This page provides detailed information, such as the category of alert generated, alert messages, when the alert was triggered, and so on. This page provides other information, such as performance statistics for the listener.

2. **Prevent online administration by requiring the administrator to have write privileges on the listener.ora file and the listener password:**

   a. Add or modify this line in the `listener.ora` file:
      ```ora
      ADMINISTRITIONS_LISTENER=ON
      ```
   b. Use `RELOAD` to reload the configuration.
   c. Use SSL when administering the listener, by making the TCPS protocol the first entry in the address list as follows:
      ```ora
      LISTENER=
      (DESCRIPTION=
       (ADDRESS_LIST=
        (ADDRESS=
         (PROTOCOL=tcps)
         (HOST = shobeen.us.example.com)
         (PORT = 8281))
       )
      )
      ```

      To administer the listener remotely, define the listener in the `listener.ora` file on the client computer. For example, to access listener USER281 remotely, use the following configuration:
      ```ora
      user281 =
      (DESCRIPTION =
       (ADDRESS =
        (PROTOCOL = tcp)
        (HOST = shobeen.us.example.com)
        (PORT = 8281))
       )
      )
      ```

      For more information about the parameters in `listener.ora`, see Oracle Database Net Services Reference.

3. **Do not set the listener password.**

   Ensure that the password has not been set in the `listener.ora` file. The local operating system authentication secures the listener administration. The remote listener administration is disabled when the password has not been set.

4. **When a host has multiple IP addresses associated with multiple NIC cards, configure the listener to the specific IP address.**

   This enables the listener to monitor all the IP addresses. You can restrict the listener to monitor a specific IP address. Oracle recommends that you specify the specific IP addresses on these types of computers, rather than enabling the listener...
to monitor all IP addresses. Restricting the listener to specific IP addresses helps to prevent an intruder from stealing a TCP end point from the listener process.

5. **Restrict the privileges of the listener, so that it cannot read or write files in the database or the Oracle server address space.**

   This restriction prevents external procedure agents spawned by the listener (or procedures executed by an agent) from inheriting the ability to perform read or write operations. The owner of this separate listener process should not be the owner that installed Oracle Database or executes the Oracle Database instance (such as ORACLE, the default owner).

   For more information about configuring external procedures in the listener, see Oracle Database Net Services Administrator’s Guide.

6. **Because you cannot protect physical addresses when transferring data over the Internet, use encryption when this data must be secure.**

   See “Protecting Data on the Network by Using Network Encryption” on page 5-5 to learn about how to protect Oracle data over the network. Oracle Database Advanced Security Administrator’s Guide describes network encryption in detail.

7. **Use a firewall.**

   Appropriately placed and configured firewalls can prevent outside access to your intranet when you allow internal users to have Internet access.

   ■ Keep the database server behind a firewall. Oracle Database network infrastructure, Oracle Net (formerly known as Net8 and SQL*Net), provides support for a variety of firewalls from various vendors. Supported proxy-enabled firewalls include Gauntlet from Network Associates and Raptor from Asent. Supported packet-filtering firewalls include PIX Firewall from Cisco, and supported stateful inspection firewalls (more sophisticated packet-filtered firewalls) include Firewall-1 from CheckPoint.

   ■ Ensure that the firewall is placed outside the network to be protected.

   ■ Configure the firewall to accept only those protocols, applications, or client/server sources that you know are safe.

   ■ Use a product such as Oracle Connection Manager to multiplex multiple-client, network sessions through a single network connection to the database. It can filter using the source, destination, and host name. This product enables you to ensure that connections are accepted only from physically secure terminals or from application Web servers with known IP addresses. (Filtering using the IP address alone is not enough for authentication, because it can be falsified.)

8. **Prevent unauthorized administration of the Oracle listener.**

   For more information about the listener, see Oracle Database Net Services Administrator’s Guide.

9. **Check network IP addresses.**

   Use the Oracle Net valid node checking security feature to allow or deny access to Oracle server processes from network clients with specified IP addresses. To use this feature, set the following sqlnet.ora configuration file parameters:

   ```
   tcp.validnode_checking = YES
   tcp.excluded_nodes = {list of IP addresses}
   tcp.invited_nodes = {list of IP addresses}
   ```
The `tcp.validnode_checking` parameter enables the feature. The `tcp.excluded_nodes` and `tcp.invited_nodes` parameters deny and enable specific client IP addresses from making connections to the Oracle listener. This helps to prevent potential Denial of Service attacks.

You can use Oracle Net Manager to configure these parameters. See Oracle Database Net Services Administrator’s Guide for more information.

10. Encrypt network traffic.

If possible, use Oracle Advanced Security to encrypt network traffic among clients, databases, and application servers. For an introduction to Oracle network encryption, see ‘Protecting Data on the Network by Using Network Encryption’ on page 5-5. For detailed information about network encryption, see Oracle Database Advanced Security Administrator’s Guide.

11. Secure the host operating system (the system on which Oracle Database resides).

Secure the host operating system by disabling all unnecessary operating system services. Both UNIX and Windows platforms provide a variety of operating system services, most of which are not necessary for typical deployments. These services include FTP, TFTP, TELNET, and so forth. Be sure to close both the UDP and TCP ports for each service that is being disabled. Disabling one type of port and not the other does not make the operating system more secure.

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.

This section contains:

- About Network Encryption
- Configuring Network Encryption

See Also: Oracle Database Advanced Security Administrator’s Guide for detailed information about network encryption

About Network Encryption

Network encryption refers to encrypting data as it travels across the network between the client and server. The reason you should encrypt data at the network level, and not just the database level, is because data can be exposed on the network level even though you have carefully encrypted it in the database. 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 need 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.
Protecting Data on the Network by Using Network Encryption

- 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.

Configuring Network Encryption

You can configure network encryption by using either Oracle Net Manager or by editing the sqlnet.ora file. This guide explains how to use Oracle Net Manager to configure network encryption.

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 Oracle Advanced Security.
   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 other side; it will otherwise be inactive.
  
  - **rejected**: Service must not be active, and the connection will fail if the other side requires.
  
  - **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; it will otherwise be 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.
Initialization Parameters Used for Network Security

- **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.
- **DES40**: Enables you to use DES to encrypt up to 40 bits of data.

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
- Oracle Database Advanced Security Administrator’s Guide for more information about network data encryption

### Initialization Parameters Used for Network Security

Table 5–1 lists initialization parameters that you can set to better secure user accounts.

<table>
<thead>
<tr>
<th>Initialization Parameter</th>
<th>Default Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OS_AUTHENT_PREFIX</td>
<td>OPS$</td>
<td>Specifies a prefix that Oracle Database uses to authenticate users attempting to connect to the database. Oracle Database concatenates the value of this parameter to the beginning of the user operating system account name and password. When a user attempts a connection request, Oracle Database compares the prefixed username with user names in the database. The default value of this parameter is OPS$ for backward compatibility with previous versions. However, you can set the prefix value to “” (a null string), thereby eliminating the addition of any prefix to operating system account names.</td>
</tr>
<tr>
<td>REMOTE_LISTENER</td>
<td>No default setting</td>
<td>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 tnsnames.ora file or other address repository as configured for your system.</td>
</tr>
<tr>
<td>REMOTE_OS_AUTHENT</td>
<td>FALSE</td>
<td>Specifies whether remote clients will be authenticated with the value of the OS_AUTHENT_PREFIX parameter.</td>
</tr>
<tr>
<td>REMOTE_OS_ROLES</td>
<td>FALSE</td>
<td>Specifies whether operating system roles are allowed for remote clients. The default value, FALSE, causes Oracle Database to identify and manage roles for remote clients.</td>
</tr>
</tbody>
</table>
To modify an initialization parameter, see "Modifying the Value of an Initialization Parameter" on page 2-6. For detailed information about initialization parameters, see Oracle Database Reference and Oracle Database Administrator’s Guide.
 Initialization Parameters Used for Network Security
Securing Data

This chapter contains:

- About Securing Data
- Encrypting Data Transparently with Transparent Data Encryption
- Controlling Data Access with Oracle Virtual Private Database
- Enforcing Row-Level Security with Oracle Label Security
- Controlling Administrator Access with Oracle Database Vault

About Securing Data

Oracle Database provides many ways to secure data. This chapter describes the following methods that you can use to secure data on your site:

- **Transparent data encryption.** Transparent data encryption encrypts data in one or more database table columns, or it can encrypt an entire tablespace. Transparent data encryption is the quickest and easiest way to encrypt data. Transparent data encryption supports the Advanced Encryption Standard (AES) and Triple Data Encryption Standard (3DES) algorithms.
  
  You can also encrypt data on the network. "Protecting Data on the Network by Using Network Encryption" on page 5-5 explains how.

- **Oracle Virtual Private Database (VPD).** This feature restricts data access by creating a policy that enforces a WHERE clause for all SQL statements that query the database. 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.

- **Oracle Label Security (OLS).** This feature secures your database tables at the row level, and assigns these rows different levels of security based on the needs of your site. You then create a security authorization for users based on the OLS labels.

- **Oracle Database Vault.** This feature enables you to restrict administrator access to your databases, enforce separation of duty, and control who, when, where and how applications, databases, and data are accessed.
Encrypting Data Transparently with Transparent Data Encryption

Transparent data encryption enables you to quickly encrypt one or more table columns or a tablespace. It is easy to implement and has many advantages over other types of database encryption.

This section contains:
■ About Encrypting Sensitive Data
■ When Should You Encrypt Data?
■ How Transparent Data Encryption Works
■ Configuring Data to Use Transparent Data Encryption
■ Checking Existing Encrypted Data

About Encrypting Sensitive Data

Encrypted data can only be read by its recipient. You use encryption to protect data in a potentially unprotected environment, such as on backup media sent to offsite storage.

Encrypted data has the following components:
■ An algorithm to encrypt the data. The encryption algorithm is a formula that Oracle Database uses to encrypt data. It translates the clear text (that is, human-readable) version of the data into a format that only can be unencrypted by another algorithm to decrypt the data. Oracle Database supports several industry-standard encryption and hashing algorithms, including the Advanced Encryption Standard (AES) encryption algorithm. AES has been approved by the National Institute of Standards and Technology (NIST) to replace the Data Encryption Standard (DES).
■ An algorithm to decrypt the data. The decryption algorithm performs the task of the encryption algorithm in reverse: it takes the data and translates it back into clear text.
■ A key to encrypt the data for the sender and to decrypt the data for the receiver. When you encrypt data, Oracle Database uses the key and clear 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.

When Should You Encrypt Data?

In most cases, you 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 because they want 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 Standards require encryption of credit card data at rest.
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.

**See Also:** Oracle Database Security Guide for common misconceptions about encrypting stored data

## 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 users select the column, the data is automatically decrypted.

To encrypt data by using transparent data encryption, you create the following components:

- **A wallet to store the master encryption key.** The wallet is a storage space in the form of a binary file located outside the database. The database uses the wallet to store the master encryption key. To create the wallet, you can use Enterprise Manager or the `ALTER SYSTEM` command. The wallet is encrypted using a password as the encryption key. You create the password when you create the wallet. Access to the contents (or master key) of the wallet is thus restricted to only those who know the password. After the wallet is created, you must open the wallet using the password so that the database can access the master encryption key.

- **A location for the wallet.** You can specify the wallet location by modifying the `sqlnet.ora` file.

Afterward, when a user enters data into an encrypted column, Oracle Database performs the following steps:

1. Retrieves the master key from the wallet.
2. Decrypts the encryption key of the table from the data dictionary.
3. Uses the encryption key to encrypt the data the user entered into the encrypted column.
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 clear text format.

Transparent data encryption has the following advantages:

- As a security administrator, you can be sure that sensitive data is safe if the storage media or data file gets stolen.
- Implementing transparent data encryption helps you address security-related regulatory compliance issues.
- You do not need to create triggers or views to decrypt data. Data from tables is transparently decrypted for the database user.
- Database users need not be aware of the fact 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.
Encrypting Data Transparently with Transparent Data Encryption

Transparent data encryption affects performance only when data is retrieved from or inserted into an encrypted column. No reduction in performance occurs for operations involving unencrypted columns, even if these columns are in a table containing encrypted columns. However, be aware that encrypted data needs more storage space than clear text data. On average, encrypting a single column requires between 32 and 48 bytes of additional storage for each row.

See Also: Oracle Database Advanced Security Administrator’s Guide for detailed information about using transparent data encryption

Configuring Data to Use Transparent Data Encryption

To start using transparent data encryption, you must create a wallet and set a master key. The wallet can be the default database wallet shared with other Oracle Database components, or a separate wallet specifically used by transparent data encryption. Oracle recommends that you use a separate wallet to store the master encryption key. This wallet will be used for all data that is being encrypted through transparent data encryption.

You follow these steps to configure table columns to use transparent data encryption:

- **Step 1: Configure the Wallet Location**
- **Step 2: Create the Wallet**
- **Step 3: Open (or Close) the Wallet**
- **Step 4: Encrypt (or Decrypt) Data**

**See Also:** Oracle Database Advanced Security Administrator’s Guide for detailed information about using tablespace encryption

**Step 1: Configure the Wallet Location**

You designate the directory location for the wallet in the sqlnet.ora file. You perform this step once.

To configure the wallet location:

1. Create a backup copy of the sqlnet.ora file, which by default is located in the $ORACLE_HOME/network/admin directory.
2. Create a directory in the $ORACLE_HOME directory in which to store the wallet.
   
   For example, create a directory called ORA_WALLETS in the C:\oracle\product\11.1.0\db_1 directory.
3. At the end of the sqlnet.ora file, add code similar to the following, where ORA_WALLETS is the name of the directory where you plan to store the wallet:
   
   ```
   ENCRYPTION_WALLET_LOCATION=(SOURCE=(METHOD=file)(DIRECTORY=C:\oracle\product\11.1.0\db_1\ORA_WALLETS))
   ```
4. Save and close the sqlnet.ora file.
5. Start SQL*Plus and then log on as SYS, connecting as SYSOPER.

```
SQLPLUS "SYS/AS SYSOPER"
Enter password: password
```
SQL*Plus starts, connects to the default database, and then displays a SQL> prompt.

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

6. Enter the following SQL statements to shut down and then restart the database:

```
SHUTDOWN IMMEDIATE
STARTUP
```

Step 2: Create the Wallet

To create the wallet, use the `ALTER SYSTEM` SQL statement. By default, the Oracle wallet 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 wallet password that might be unknown to the database administrator provides separation of duty: The database administrator might be able to restart the database, but the wallet is closed and must be manually opened by a security administrator who knows the wallet password.

To create the wallet:

1. In SQL*Plus, connect as a user with administrative privileges, such as `SYSTEM`, or as a security administrator.

   For example:
   ```sql
   CONNECT SYSTEM
   Enter password: password
   ```

2. Enter the following `ALTER SYSTEM` statement, where `password` is the password you want to assign to the encryption key:

   ```sql
   ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "password";
   ```

   Enclose the password in double quotation marks. As with other passwords that you create in Oracle Database, the password will not appear in clear text or in any dynamic views or logs.

   This statement generates the wallet with a new encryption key and sets it as the current transparent data encryption master key. If you plan to use public key infrastructure (PKI) to configure the master encryption key, then specify a certificate ID, which is an optional string that contains the unique identifier of a certificate stored in the Oracle wallet. Use the following syntax:

   ```sql
   ALTER SYSTEM SET ENCRYPTION KEY certificate_ID IDENTIFIED BY "password";
   ```

Step 3: Open (or Close) the Wallet

Immediately after you create the wallet key, the wallet is open, and you are ready to start encrypting data. However, if you have restarted the database after you created the wallet, you must manually open the wallet before you can use transparent data encryption.

To open the wallet:

- In SQL*Plus, enter the following `ALTER SYSTEM` statement, where `password` is the password you assigned to the encryption key:

  ```sql
  ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY "password";
  ```
In most cases, leave the wallet open unless you have a reason for closing it. You can close the wallet to disable access to the master key and prevent access to the encrypted columns. However, the unencrypted data is still available. The wallet must be open for transparent data encryption to work. To reopen the wallet, use the `ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY password` statement.

To close the wallet:
- In SQL*Plus, enter the following statement:
  ```sql
  ALTER SYSTEM SET ENCRYPTION WALLET CLOSE;
  ```

**Step 4: Encrypt (or Decrypt) Data**

After you have created a directory location for the wallet in the `sqlnet.ora` file and created the wallet itself, you are ready to encrypt either individual table columns or an entire tablespace.

This section contains the following topics:
- Encrypting Individual Table Columns
- Encrypting a Tablespace

**Encrypting Individual Table Columns**

The decisions that you make when you identify columns to be encrypted are determined by governmental security regulations, such as California Senate Bill 1386, or by private standards used by companies such as MasterCard or VISA. Credit card numbers, Social Security numbers, and other personally identifiable information (PII) fall under this category. Another need for encryption is defined by your own internal security policies — trade secrets, research results, or employee salaries and bonuses. See "When Should You Encrypt Data?" on page 6-2 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:
  ```
  BINARY_FLOAT
  BINARY_DOUBLE
  CHAR
  DATE
  NCHAR
  NUMBER
  RAW
  TIMESTAMP
  VARCHAR2
  ```

- 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 a wallet key.
   - ‘Step 2: Create the Wallet’ on page 6-5 explains how to create a wallet key. To open an existing wallet key, see ‘Step 3: Open (or Close) the Wallet’ on page 6-5.

2. Start Database Control.
Encrypting Data Transparently with Transparent Data Encryption

See Oracle Database 2 Day DBA for instructions about how to start Database Control.

3. Enter an administrator user name (for example, SYSTEM, or the name of a security administrator) and password, and then click Login.

The Database Home page appears.

4. Click Schema to display the Schema subpage.

5. Under Database Objects, select Tables.

The Tables page appears.

6. 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 its encryption options.

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

7. In the Create Table (or Edit Table) page, do the following:
   - Select the column that you want to encrypt.
     Do not select any indexed columns or columns that use 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.
   - Click Encryption Options to display the Encryption Options for the Table page.
   - 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.
Encrypting Data Transparently with Transparent Data Encryption

- **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 repetition of text in the clear 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.

t. Enable encryption for the column by selecting its box under Encrypted.

8. Click Continue.

The Create Table (or Edit Table) page appears. 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. When a table is updated, read access is still possible. If data manipulation language (DML) statements are needed, you can use online redefinition statements.

**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 a wallet key.
   "Step 2: Create the Wallet" on page 6-5 explains how to create a wallet key. To open an existing wallet key, see "Step 3: Open (or Close) the Wallet" on page 6-5.

2. Start Database Control.
   See Oracle Database 2 Day DBA for instructions about how to start Database Control.

3. Enter an administrator user name (for example, SYSTEM, or the name of a security administrator) and password, and then click Login.
   The Database Home page appears.

4. Click Server to display the Server subpage.

5. Under Storage, click Tablespaces.
   The Tablespaces page appears.

6. Click Create, and then answer the questions in the subsequent page to start creating the tablespace and its required data file.

7. In the Create Tablespace page, do the following:
   a. Under Type, select the Encryption box, under Permanent.
b. Select Encryption options to display the Encryption Options 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.
   See “Available Methods” under Step 5 in “Configuring Network Encryption” on page 5-6 for more information about these encryption algorithms.

d. Click Continue.
   The Create Tablespace page appears.

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

See Also:
   - "Checking Encrypted Tablespaces in the Current Database Instance" on page 6-10 to query the database for existing encrypted tablespaces
   - Oracle Database Advanced Security Administrator’s Guide for detailed information about tablespace encryption
   - Oracle Database SQL Language Reference for more information about the CREATE TABLESPACE statement

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.

This section contains:
   - Checking Whether a Wallet Is Open or Closed
   - Checking Encrypted Columns of an Individual Table
   - Checking All Encrypted Table Columns in the Current Database Instance
   - Checking Encrypted Tablespaces in the Current Database Instance

Checking Whether a Wallet Is Open or Closed

You can find out if a wallet is open or closed by running the $V$ENCRYPTION_WALLET view.

To check whether a wallet is open or closed:
   - In SQL*Plus, run the $V$ENCRYPTION_VIEW view as follows:

```
SELECT * FROM $V$ENCRYPTION_WALLET;
```

The wallet status appears, similar to the following:
Checking Encrypted Columns of an Individual Table

You use the DESCRIBE (for DESCRIBE) statement in SQL*Plus to check the encrypted columns in a database table.

To check the encrypted columns of an individual table:

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

  ```sql
  DESCR tablename;
  ```

  For example:

  ```sql
  DESCR OE.ORDER_ITEMS;
  ```

  A description of the table schema appears. For example:

  ```
  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) ENCRYPT
  QUANTITY
  ```

Checking All Encrypted Table Columns in the Current Database Instance

To check all encrypted table columns, you use the DBA_ENCRYPTED_COLUMNS view.

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

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

  ```sql
  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
  -----------  ----------    -----------    ----------------   ----
  OE           CUSTOMERS     INCOME_LEVEL   AES 128 bits key   YES
  OE           UNIT_PRICE    ORDER_ITEMS    AES 128 bits key   YES
  HR           EMPLOYEES     SALARY         AES 128 bits key   YES
  ```

  See Also: Oracle Database Reference for more information about the DBA_ENCRYPTED_COLUMNS view

Checking Encrypted Tablespaces in the Current Database Instance

Table 6–1 lists data dictionary views that you can use to check encrypted tablespaces.
Controlling Data Access with 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 credentials of a user.

This section contains:

- **About Oracle Virtual Private Database**
- **Tutorial: Creating an Oracle Virtual Private Database Policy**

### About Oracle Virtual Private Database

Oracle Virtual Private Database (VPD) provides row-level security at the database table or view level. You can extend it to provide column-level security as well. Essentially, Virtual Private Database inserts an additional `WHERE` clause to any SQL statement that is used on any table or view to which a Virtual Private Database security policy has been applied. (A security policy is a function that allows or prevents access to data.) The `WHERE` clause allows only users whose credentials pass the security policy, and hence, have access to the data that you want to protect.

### Table 6–1   Data Dictionary Views for Encrypted Tablespaces

<table>
<thead>
<tr>
<th>Data Dictionary View</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DBA_TABLESPACES</strong></td>
<td>Describes all tablespaces in the database. For example, find out if the tablespace has been encrypted, enter the following:</td>
</tr>
<tr>
<td></td>
<td><code>SELECT TABLESPACE_NAME, ENCRYPTED FROM DBA_TABLESPACES</code></td>
</tr>
<tr>
<td></td>
<td><code>TABLESPACE_NAME            ENC</code></td>
</tr>
<tr>
<td></td>
<td><code>-----------------------   -------</code></td>
</tr>
<tr>
<td></td>
<td><code>SYSTEM                      NO</code></td>
</tr>
<tr>
<td></td>
<td><code>SYSAUX                     NO</code></td>
</tr>
<tr>
<td></td>
<td><code>UNDOTBS1                   NO</code></td>
</tr>
<tr>
<td></td>
<td><code>TEMP                        NO</code></td>
</tr>
<tr>
<td></td>
<td><code>USERS                       NO</code></td>
</tr>
<tr>
<td></td>
<td><code>EXAMPLE                    NO</code></td>
</tr>
<tr>
<td></td>
<td><code>SECURESPACE                 YES</code></td>
</tr>
<tr>
<td><strong>USER_TABLESPACES</strong></td>
<td>Describes the tablespaces accessible to the current user. It has the same columns as <code>DBA_TABLESPACES</code>, except for the <code>PLUGGED_IN</code> column.</td>
</tr>
<tr>
<td><strong>V$ENCRYPTED_TABLESPACE</strong></td>
<td>Displays information about the tablespaces that are encrypted. For example:</td>
</tr>
<tr>
<td></td>
<td><code>SELECT * FROM V$ENCRYPTED_TABLESPACE</code></td>
</tr>
<tr>
<td></td>
<td><code>TS#  ENCRYPTIONALG  ENCRYPTEDTS</code></td>
</tr>
<tr>
<td></td>
<td><code>-----------  --------------  ----</code></td>
</tr>
<tr>
<td></td>
<td><code>-------  -------</code></td>
</tr>
<tr>
<td></td>
<td><code>6   AES128      YES</code></td>
</tr>
</tbody>
</table>

See Also: [Oracle Database Reference](#) for more information about data dictionary views

See Also: [Oracle Database Security Guide](#) for detailed information about how Oracle Virtual Private Database works
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:

- Lines 2-3: 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.

- Line 5: Returns the string that will be used for the WHERE predicate clause.

- Lines 6-10: Encompass 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. An application context is a name-value pair. For example:

```
SELECT * FROM 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 Database Control or 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.

Tutorial: Creating an Oracle Virtual Private Database Policy

The ORDERS table in the Order Entry database, OE, contains the following information:

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORDER_ID</td>
<td>NOTNULL</td>
<td>NUMBER(12)</td>
</tr>
<tr>
<td>ORDER_DATE</td>
<td>NOTNULL</td>
<td>TIMESTAMP(6) WITH LOCAL TIME ZONE</td>
</tr>
<tr>
<td>ORDER_MODE</td>
<td></td>
<td>VARCHAR2(8)</td>
</tr>
<tr>
<td>CUSTOMER_ID</td>
<td>NOTNULL</td>
<td>NUMBER(6)</td>
</tr>
<tr>
<td>ORDER_STATUS</td>
<td></td>
<td>NUMBER(2)</td>
</tr>
<tr>
<td>ORDER_TOTAL</td>
<td></td>
<td>NUMBER(8,2)</td>
</tr>
<tr>
<td>SALES_REP_ID</td>
<td></td>
<td>NUMBER(6)</td>
</tr>
<tr>
<td>PROMOTION_ID</td>
<td></td>
<td>NUMBER(6)</td>
</tr>
</tbody>
</table>

Suppose you want to limit access to this table based on the person who is querying the table. For example, a sales representative should only see the orders that have been created, but other employees should not. In this tutorial, you create a sales representative user account and an account for a finance manager. Then, you create an Oracle Virtual Private Database policy that will limit the data access to these users based on their roles.

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 ID of the customer.

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" on page 4-4, 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” on page 6-20 for more information.

In this tutorial:

- Step 1: If Necessary, Create the Security Administrator Account
- Step 2: Update the Security Administrator Account
- Step 3: Create User Accounts for This Tutorial
- Step 4: Create the F_POLICY_ORDERS Policy Function
- Step 5: Create the ACCESSCONTROL_ORDERS Virtual Private Database Policy
Step 1: If Necessary, Create the Security Administrator Account

In Tutorial: Creating a Secure Application Role on page 4-4, you created a security administrator account called sec_admin for that tutorial. You can use that account for this tutorial. If you have not yet created this account, follow the steps in Step 1: Create a Security Administrator Account on page 4-4 to create sec_admin.

Step 2: 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:

1. Start Database Control.

   See Oracle Database 2 Day DBA for instructions about how to start Database Control.

2. Log in as user SYS and connect with the SYSDBA privilege:

   - User Name: SYS
   - Password: Enter the password for SYS
   - Connect As: SYSDBA

3. Click Server to display the Server subpage.


   The Users Page appears.

5. Select SEC_ADMIN and then click Edit.

   The Edit User page appears.

6. Click Object Privileges to display the Object Privileges page.

7. From the Select Object Type list, select Package, and then click Add.

   The Add Package Object Privileges page appears.

8. Under Select Package Objects, enter SYS.DBMS_RLS so that sec_admin will have access to the DBMS_RLS package.

9. Under Available Privileges, select EXECUTE, and then click Move to move it to the Selected Privileges list.

10. Click OK.

    The Edit User page appears.

11. From the Select Object Type list, select Table, and then click Add.

    The Add Table Object Privileges page appears.

12. Select Table Objects, and then enter HR.EMPLOYEES so that sec_admin will have access to the HR.EMPLOYEES table.

13. Under Available Privileges, select SELECT, and then click Move to move it to the Selected Privileges list.
14. Click OK.
   The Edit User page appears.
15. Click Apply.

**Step 3: Create User Accounts for This Tutorial**

You are ready to create accounts for the employees who need to access the `OE.ORDERS` table.

**To create the employee user accounts:**

1. In Database Control, click Users in the Database Instance link to return to the Users page.
   The Users page appears.
2. Click Create.
   The Create User page appears.
3. 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" on page 3-8.
   - **Default Tablespace**: USERS
   - **Temporary Tablespace**: TEMP
   - **Status**: Unlocked
4. Click OK.
   The Users page appears, with LDORAN listed as a new user.
5. Select LDORAN from the Users page.
   The Edit User page appears.
6. Select **Object Privileges** to display the Object Privileges subpage.
7. From the Select Object Type list, select **Table**, and then click **Add**.
   The Add Table Object Privileges page appears.
8. In the **Select Table Objects** field, enter the following text:
   `OE.ORDERS`
   Do not include spaces in this text.
9. 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.
10. Click **Apply**.
11. Select LDORAN, and from the **Actions** list, select **Create Like**. Then, click **Go**.
    The Create User page appears.
12. Enter the following information:
   ■ Name: LPOP (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” on page 3-8.

13. Click OK.

Both employee accounts have been created, and they have identical privileges. If either performs a SELECT statement on the OE.ORDERS table, he or she will be able to see all of its data.

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. In Database Control, click Logout and then Login.
2. Log in as user sec_admin.
3. Click Schema to display the Schema subpage.
4. Under Programs, select Functions.
   The Functions page appears.
5. Click Create.
   The Create Function page appears.
6. Enter the following information:
   ■ Name: F_POLICY_ORDERS
   ■ Schema: SEC_ADMIN
   ■ Source: 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 accomplishes this by using the SYS_CONTEXT PL/SQL function to get the session information of the user, and then it compares this information with the job ID of that user in the HR.EMPLOYEES table, for which sec_admin has SELECT privileges.
In this example:

- Lines 1-2: Define 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 uses these parameters to specify the OE schema and ORDERS table. Ensure that you create the schema parameter first, followed by the tab parameter.

- Line 3: Returns the string that will be used for the WHERE predicate clause. Always use VARCHAR2 as the data type for this return value.

- Lines 4-7: Define variables to store the job ID, user name of the user who has logged on, and predicate values.

- Lines 9-25: Encompass the creation of the WHERE predicate, starting the with the BEGIN clause at Line 9.

- Lines 10-12: 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 Line 16.

- Line 14: Uses the SYS_CONTEXT function to retrieve the session information of the user and write it to the v_user variable.

- Lines 16-23: 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.

- Line 25: 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 LPORP issue on the OE.ORDERS table.

- Lines 27–29: Provide an EXCEPTION clause for cases where a user without the correct privileges has logged on.

7. Click OK.

Step 5: Create the ACCESSCONTROL_ORDERS Virtual Private Database Policy

Now that you have created the Virtual Private Database policy function, 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. The SELECT, UPDATE, and DELETE statements are impossible to use, because the user was not granted permissions.

To create the ACCESSCONTROL_ORDERS Virtual Private Database policy:

1. In Database Control, click the Database Instance link to display the Database Home page.
2. Click Server to display the Server subpage.
   The Virtual Private Database Policies page appears.
4. Click Create.
   The Create Policy page appears.
5. 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 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.
6. 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.

7. Under Enforcement, select SELECT.
8. Click OK.

**Step 6: Test the ACCESSCONTROL_ORDERS Virtual Private Database Policy**

At this stage, you are ready to test the accesscontrol_orders 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:
   
   ```sql
   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:
   
   ```sql
   SELECT COUNT(*) FROM OE.ORDERS;
   ```
   
   The following results should appear, because Mr. Popp, who is not a sales representative, does not have access to the data in the OE.ORDERS table.
   
   ```
   COUNT(*)
   -------
   0
   ```
   
5. Exit SQL*Plus:
   
   ```
   EXIT
   ```

**Step 7: Optionally, Remove the Components for This Tutorial**

After completing this tutorial, you can remove the data structures that you used if you no longer need them.
To remove the data structures created by sec_admin:
1. In Database Control, log in as user sec_admin.
2. Click Server to display the Server subpage.
   The Virtual Private Database Policies page appears.
4. 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.
5. Select ACCESSCONTROL_ORDERS, and then click Delete.
6. In the Confirmation page, click Yes.

To remove the user accounts and roles:
1. In Database Control, click Logout, and then Login.
2. Log in as the administrative user who created the user accounts and roles used in this tutorial.
3. Click Server to display the Server subpage.
   The Users page appears.
5. 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. Exit Database Control.

Enforcing Row-Level Security with 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 section contains:
■ About Oracle Label Security
■ Guidelines for Planning an Oracle Label Security Policy
■ Tutorial: Applying Security Labels to the HR.LOCATIONS Table

About Oracle Label Security
You use Oracle Label Security to secure your database tables at the row level, and assign these rows different levels of security based on the needs of your site. 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
Enforcing Row-Level Security with Oracle Label Security

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.

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 or not 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, or 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 to 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 Database Control, or you can create them using the SA_SYSDBA, SA_COMPONENTS, and SA_LABEL_ADMIN PL/SQL packages. For information about using the PL/SQL packages, see Oracle Label Security Administrator’s Guide. This guide explains how to create Oracle Label Security labels and policies by using Database Control.

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 or not 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.

### Guidelines for Planning an Oracle Label Security Policy

Before you create an Oracle Label Security policy, you must determine where and 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 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. They must correspond correctly for each table identified during the schema analysis you performed in Step 1. Then, compare the organizational structure of the user population with the data groups that you identified in Step 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.
Tutorial: Applying Security Labels to the HR.LOCATIONS Table

This tutorial demonstrates the general concepts of using Oracle Label Security. In it, you will apply security labels to the HR.LOCATIONS table. Three users, sking, kpartner, and idoran 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 the user has in your organization, you can create a PL/SQL function or procedure to use with a Virtual Private Database policy. See “Controlling Data Access with Oracle Virtual Private Database” on page 6-11 for more information.

The schema for HR.LOCATIONS is as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCATION_ID</td>
<td>NOT NULL</td>
<td>NUMBER (4)</td>
</tr>
<tr>
<td>STREET_ADDRESS</td>
<td></td>
<td>VARCHAR2 (40)</td>
</tr>
<tr>
<td>POSTAL_CODE</td>
<td></td>
<td>VARCHAR2 (12)</td>
</tr>
<tr>
<td>CITY</td>
<td>NOT NULL</td>
<td>VARCHAR2 (30)</td>
</tr>
<tr>
<td>STATE_PROVINCE</td>
<td></td>
<td>VARCHAR2 (25)</td>
</tr>
<tr>
<td>COUNTRY_ID</td>
<td></td>
<td>CHAR (2)</td>
</tr>
</tbody>
</table>

You will apply the following labels:

<table>
<thead>
<tr>
<th>Label</th>
<th>Privileges</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIDENTIAL</td>
<td>Read access to the cities Munich, Oxford, and Roma</td>
</tr>
<tr>
<td>SENSITIVE</td>
<td>Read access to the cities Beijing, Tokyo, and Singapore</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>Read access to all other cities listed in HR.LOCATIONS</td>
</tr>
</tbody>
</table>

In this tutorial:
- Step 1: Install Oracle Label Security and Enable User LBACSYS
- Step 2: Create a Role and Three Users for the Oracle Label Security Tutorial
- Step 3: Create the ACCESS_LOCATIONS Oracle Label Security Policy
- Step 4: Define the ACCESS_LOCATIONS Policy-Level Components
- Step 5: Create the ACCESS_LOCATIONS Policy Data Labels
- Step 6: Create the ACCESS_LOCATIONS Policy User Authorizations
- Step 7: Apply the ACCESS_LOCATIONS Policy to the HR.LOCATIONS Table
- Step 8: Add the ACCESS_LOCATIONS Labels to the HR.LOCATIONS Data
- Step 9: Test the ACCESS_LOCATIONS Policy
- Step 10: Optionally, Remove the Components for This Tutorial

Step 1: Install Oracle Label Security and Enable User LBACSYS

In a default Oracle Database installation, Oracle Label Security is not installed, but it is part of the products available in Oracle Database. You can install it in an existing database by using Oracle Universal Installer, and then Database Configuration Assistant (DBCA) to register it. Oracle Label Security provides its own user account, LBACSYS, which you will need to enable after the installation.
Installing Oracle Label Security

This procedure explains how to install Oracle Label Security in an existing database.

To install Oracle Label Security:
1. Shut down the database instance in which you plan to install Oracle Label Security.
   Log in to SQL*Plus as SYS, connecting with the SYSOPER privilege. At the SQL prompt, enter the following command:
   `SHUTDOWN IMMEDIATE`
2. Exit SQL*Plus.
   `EXIT`
3. Stop the Oracle Database processes.
   - UNIX: Go to the `$ORACLE_HOME/bin` directory and run the following commands to stop the Database Console and the listener:
     ```
     ./emctl stop dbconsole
     ./lsnrctl stop
     ```
   - Windows: In the Windows Services tool, right-click the Oracle listener, console, and database service services, and then from the menu, select Stop. The names of these services begin with Oracle and include the name of the database instance. For example, assuming the database instance is `orcl`, the names would be similar to the following:
     - `OracleDBConsoleorcl`
     - `OracleJobSchedulerORCL`
     - `OracleOraDB1g-home1TNSListener`
     - `OracleServiceORCL`
4. Run Oracle Universal Installer from the installation media.
   - UNIX: Use the following command:
     ```
     /mnt/cdrom/runInstaller
     ```
   - Windows: Double-click the file, `setup.exe`, on the installation media.
5. In the Select a Product to Install window, select Oracle Database 11g, and then click Next.
6. Select Advanced Installation, and then click Next.
   The Select Installation Type window appears.
7. Select Custom, and then click Next.
   The Specify Home Details screen appears.
8. Select the Oracle base directory and the Oracle home directory in which you want to install Oracle Label Security. Click Next.
By default, Oracle Universal Installer offers to create a new Oracle home for you, so ensure that you select the correct existing Oracle home. Oracle Universal Installer then verifies that your system meets the minimum requirements. Next, the Available Product Components window is displayed.

9. Select the box corresponding to Oracle Label Security.
   You can find this option under Enterprise Edition Options. Oracle Universal Installer also selects Oracle Services For Microsoft Transaction Server, but if you do not need this product, you can deselect it. Then click Next.
   The Summary window is displayed.
10. Review your choices and then click Install.
    The progress window is displayed. When the installation completes, Oracle Universal Installer displays the End of Installation window.
11. Click Exit, and then click Yes to confirm the exit.
12. Restart the services and the database instance in which you installed Oracle Label Security.

   ■ UNIX: Go to the $ORACLE_HOME/bin directory and run the following commands to start the Database Console and the listener:
     ```
     ./emctl start dbconsole
     ./lsnrctl start
     ```
   Start SQL*Plus and then restart the database instance:
   ```
   SQLPLUS "SYS/AS SYSOPER"
   Enter password: password
   Connected to an idle instance
   SQL> STARTUP
   ```

   ■ Windows: In the Windows Services tool, right-click the Oracle listener, console, and database service services, and then from the menu, select Start. The names of these services begin with Oracle and include the name of the database instance. For example, assuming the database instance is orcl, the names would be similar to the following:
   - OracleDBConsoleorcl
   - OracleJobSchedulerORCL (Optional; you do not need to start it for the tutorials in this guide.)
   - OracleOraDB1g-home1TNSListener
   - OracleServiceORCL (This service starts when you start OracleDBConsole.)

Registering Oracle Label Security with Oracle Database
After you complete the installation, you must register Oracle Label Security with Oracle Database.

To register Oracle Label Security with Oracle Database:
1. Start Database Configuration Assistant.

   ■ UNIX: Enter the following command at a terminal window:
   ```
   dbca
   ```
   Typically, dbca is in the $ORACLE_HOME/bin directory.
Enforcing Row-Level Security with Oracle Label Security

- **Windows:** From the Start menu, click All Programs. Then, click Oracle - ORACLE_HOME, Configuration and Migration Tools, and then Database Configuration Assistant.
  Alternatively, you can start Database Configuration Assistant at a command prompt:
  
  `dbca`

  As with UNIX, typically, `dbca` is in the `ORACLE_BASE\ORACLE_HOME\bin` directory.

2. In the Welcome page, click Next.
   The Operations page appears.
3. Select Configure Database Options, and then click Next.
   The Database page appears.
4. From the list, select the database where you installed Oracle Label Security and then click Next.
   The Management Options page appears.
5. Select Keep the database configured with Database Control.
   The Security Settings page appears.
6. Select the security option you prefer, and then click Next.
   Oracle recommends that you take advantage of the enhanced security settings for this release.
   The Database Components page appears.
7. Select Oracle Label Security, and then click Next.
   The Connection Mode page appears.
8. Select either Dedicated Server Mode or Shared Server Mode (depending on the selection you made when you created this database), click Finish, and then click OK in the confirmation prompts.
   Database Configuration Assistant registers Oracle Label Security, and then restarts the database instance.
9. Exit Database Configuration Assistant.

**Enabling the Default Oracle Label Security User Account LBACSYS**

The Oracle Label Security installation process creates a default user account, LBACSYS, who manages the Oracle Label Security features. An administrator can create a user who has the same privileges as this user, that is, EXECUTE privileges on the SA_SYSDBA, SA_COMPONENTS, and SA_LABEL_ADMIN PL/SQL packages. By default, LBACSYS is created as a locked account with its password expired. Your next step is to unlock LBACSYS and create a new password. Because user LBACSYS is using Database Control to create the Oracle Label Security policy, you must grant the SELECT ANY DICTIONARY privilege to LBACSYS.

**To unlock LBACSYS, create a new password, and grant it SELECT ANY DICTIONARY privileges:**

1. Log in to Database Control as the user SYSTEM.
In the Login page, enter \texttt{SYSTEM} and the password assigned to \texttt{SYSTEM}. Set \texttt{Connect As} to \texttt{Normal}. Select \texttt{Login} to log in.

2. Click \texttt{Server} to display the Server subpage.

3. Under Security, select \texttt{Users}.

   The Users page appears.

4. Select user \texttt{LBACSYS}.

   To quickly find \texttt{LBACSYS}, enter \texttt{lba} in the Object Name field, and then click \texttt{Go}.

5. With \texttt{LBACSYS} selected, click \texttt{Edit}.

   The Edit User page appears.

6. Next to Status, select \texttt{Unlocked}.

7. In the Enter Password and Confirm Password fields, enter a secure password, according to the guidelines in “Requirements for Creating Passwords” on page 3-8.

   For greater security, do not reuse the same password that was used in previous releases of Oracle Database.

8. Click \texttt{System Privileges} to display the Edit User: \texttt{LBACSYS} page.

9. Click \texttt{Edit List}.

   The Modify System Privileges page appears.

10. In the Available System Privileges list, select \texttt{SELECT ANY DICTIONARY}, and then click \texttt{Move} to move it to the Selected System Privileges list. Then click \texttt{OK}.

11. Click \texttt{Apply}.

Step 2: 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
- Creating the Users

Creating a Role

The \texttt{emp_role} role provides the necessary privileges for the three users you will create.

To create the role \texttt{emp_role}:

1. Ensure that you are logged in to Database Control as \texttt{SYSTEM}.

   If you are not already logged in as \texttt{SYSTEM}, then select \texttt{Logout}, and then select \texttt{Login}. In the Login page, enter \texttt{SYSTEM} and the password assigned to that account. Set \texttt{Connect As} to \texttt{Normal}. Select \texttt{Login} to log in.

   If you are logged in as \texttt{SYSTEM}, click the Database Instance link to display the home page.

2. Click \texttt{Schema} to display the Schema subpage.

3. In the Users & Privileges section, click \texttt{Roles}.

   The Roles page appears.

4. Click \texttt{Create}.

   The Create Role page appears.
5. In the Name field, enter EMP_ROLE and leave Authentication set to None.
6. Select the Object Privileges subpage.
7. From the Select Object Type list, select Table, and then click Add.
   The Add Table Object Privileges page appears.
8. 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.
9. Click OK to return to the Create Role page, and then click OK to return to the Roles page.

Creating the Users
The three users you 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. Ensure that you are logged in to Database Control as SYSTEM.
   If you are not already logged in as SYSTEM, then select Logout, and then select Login. In the Login page, enter SYSTEM and the password assigned to that account. Set Connect As to Normal. Select Login to log in.
   If you are logged in as SYSTEM, click the Database Instance link to display the home page.
2. Click Server to display the Server subpage.
3. In the Security section, click Users.
   The Users page appears.
4. Click Create.
   The Create User page appears.
5. 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' on page 3-8.
   - Default Tablespace: USERS
   - Temporary Tablespace: TEMP
   - 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.
Enforcing Row-Level Security with Oracle Label Security

6. **System Privileges**: Select the System Privileges subpage and then click Edit List to grant the CREATE SESSION privileges. Do not grant the ADMIN OPTION option.

7. Click OK.

8. In the Users page, select SKING, set Actions to Create Like, and then click Go.

The Create User page appears.

8. Create accounts for kpartner and l doran.

Create their names and passwords. (See “Requirements for Creating Passwords” on page 3-8.) 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 SELECT privileges on the HR.LOCATIONS table.

**Step 3: Create the ACCESS_LOCATIONS Oracle Label Security Policy**

Next, you are ready to create the ACCESS_LOCATIONS policy.

To create the ACCESS_LOCATIONS policy:

1. Log in to Database Control as user LBACSYS.

Select Logout, and then select Login. In the Login page, log in as user LBACSYS. Set Connect As to Normal. Select Login to log in.

2. Click Server to display the Server subpage.


The Label Security Policies page appears.

4. Click Create.

5. 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.
   - **Enabled**: Select this box to enable the policy. (It should be enabled by default.)
   - **Enforcement Options**: Select Apply Policy Enforcements, and then select the following options:
     
     For all queries (READ_CONTROL)
     
     To use session’s default label for label column update (LABEL_DEFAULT)

6. Click OK.

The ACCESS_LOCATIONS policy appears in the Label Security Policies page.
Enforcing Row-Level Security with Oracle Label Security

Step 4: Define the ACCESS_LOCATIONS Policy-Level Components

At this stage, you have the policy and have set enforcement options for it. 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. The Edit Label Security Policy page appears.

2. Select the Label Components subpage.

3. 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>
<thead>
<tr>
<th>Long Name</th>
<th>Short Name</th>
<th>Numeric Tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENSITIVE</td>
<td>SENS</td>
<td>3000</td>
</tr>
<tr>
<td>CONFIDENTIAL</td>
<td>CONF</td>
<td>2000</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>PUB</td>
<td>1000</td>
</tr>
</tbody>
</table>
4. Click Apply.

**Step 5: Create the ACCESS_LOCATIONS Policy Data Labels**

In this step, you create data labels for the policy you created in Step 4: Define the ACCESS_LOCATIONS Policy-Level Components. 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:**
1. Return to the Label Security policies page by selecting the Label Security Policies link.
2. Select the ACCESS_LOCATIONS policy.
3. In the Actions list, select Data Labels, and then click Go.
   The Data Labels page appears.
4. Click Add.
   The Create Data Label page appears.
5. Enter the following information:
   - **Numeric Tag:** Enter 1000.
   - **Level:** From the list, select PUB. (To use the keyboard to select an item, enter the first letter of its name. For example, enter P to select PUB.)
6. Click OK.
   The data label appears in the Data Labels page.
7. Click Add again, and then create a data label for the CONF level. For the numeric tag, enter 2000.
8. Click OK.
9. Click Add again, and then create a data label for the SENS level. For the numeric tag, enter 3000.
10. Click OK.
    At this stage, the CONF, PUB, and SENS labels appear in the Data Labels page.

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.
Step 6: 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 ACCESS_LOCATIONS policy.
3. In the Actions list, select Authorization, and then click Go.
   The Authorization page appears.
4. Click Add Users.
   The Add User: Users page appears.
5. 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 box for user SKING, and then click Select.
   The Create User page lists user SKING.
7. Click Next.
8. In the Privileges page, select Next.
   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, use the flashlight icon to select data to enter for the following fields, so that user SKING will be able to read sensitive and confidential data in HR_LOCATIONS:
   - Maximum Level: SENS (for SENSITIVE)
   - Minimum Level: CONF (for CONFIDENTIAL)
   - Default Level: SENS
Enforcing Row-Level Security with Oracle Label Security

**Row Level**: SENS

10. Click Next.

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.

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 through Step 12 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 all 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.
Step 7: 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 ACCESS_LOCATIONS policy.
3. In the Actions list, select Apply, and then click Go.
   - The Apply page appears.
4. 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.
   - The default policy enforcement options for ACCESS_LOCATIONS are:
     - 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.

Step 8: Add the ACCESS_LOCATIONS Labels to the HR.LOCATIONS Data

After you have applied the ACCESS_LOCATIONS policy to the HR.LOCATIONS table, you apply the labels of the policy to the OLS_COLUMN in LOCATIONS. For the user HR (the owner of that table) to accomplish this, the user must have FULL access to locations before being able to add the data labels to the hidden OLS_COLUMN column in LOCATIONS.

- Granting HR FULL Policy Privilege for the HR.LOCATIONS Table
- Updating the OLS_COLUMN Table in HR.LOCATIONS

Granting HR FULL Policy Privilege for the HR.LOCATIONS Table

The label security administrative user, LBACSYS, can grant 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 ACCESS_LOCATIONS policy.
3. Select Authorization from the Actions list, and then click Go.
   - The Authorization page appears.
4. Click Add Users.
   The Add User page appears.
5. Under Database Users, click Add.
   The Search and Select window appears.
6. Select the box for user HR, and then click Select.
   The Create User page lists user HR.
7. Click Next.
   The Privileges step appears.
8. Select the Bypass all Label Security checks (FULL) privilege, and then click Next.
   The Labels, Compartments, and Groups page appears.
9. Click Next.
   The Audit step appears.
10. Click Next.
    The Review step appears.
11. Click Finish.

At this stage, HR is listed in the Authorization page with the other users.

12. Exit Database Control.

**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.
   
   ```sql
   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" on page 3-8.
   
   ```sql
   ALTER USER HR ACCOUNT UNLOCK IDENTIFIED BY password
   ```

2. Enter the following UPDATE statement to apply the SEIS label to the cities Beijing, Tokyo, and Singapore:
   
   ```sql
   UPDATE LOCATIONS
   ```
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;
```

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.

6. Revoke the FULL access from user HR.

Refer to the steps in "Granting HR FULL Policy Privilege for the HR.LOCATIONS Table" on page 6-34 to revoke FULL access from user HR.

Step 9: Test the ACCESS_LOCATIONS Policy

The ACCESS_LOCATIONS policy is complete and ready to be tested. You can test it by logging in to SQL*Plus as each of the three users and performing 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:

COL city HEADING City FORMAT a25
COL country_id HEADING Country FORMAT a11
COL label format a10
```
Now enter the `SELECT` statement as follows:

```sql
SELECT city, country_id, LABEL_TO_CHAR (OLS_COLUMN) AS Label FROM hr.locations ORDER BY ols_column;
```

User skiing 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`.

<table>
<thead>
<tr>
<th>City</th>
<th>Country</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Venice</td>
<td>IT</td>
<td>PUB</td>
</tr>
<tr>
<td>Utrecht</td>
<td>NL</td>
<td>PUB</td>
</tr>
<tr>
<td>Bern</td>
<td>CH</td>
<td>PUB</td>
</tr>
<tr>
<td>Geneva</td>
<td>CH</td>
<td>PUB</td>
</tr>
<tr>
<td>San Paulo</td>
<td>BR</td>
<td>PUB</td>
</tr>
<tr>
<td>Stratford</td>
<td>UK</td>
<td>PUB</td>
</tr>
<tr>
<td>Mexico City</td>
<td>MX</td>
<td>PUB</td>
</tr>
<tr>
<td>Hiroshima</td>
<td>JP</td>
<td>PUB</td>
</tr>
<tr>
<td>Southlake</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>South San Francisco</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>South Brunswick</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>Seattle</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>Toronto</td>
<td>CA</td>
<td>PUB</td>
</tr>
<tr>
<td>Whitehorse</td>
<td>CA</td>
<td>PUB</td>
</tr>
<tr>
<td>Bombay</td>
<td>IN</td>
<td>PUB</td>
</tr>
<tr>
<td>Sydney</td>
<td>AU</td>
<td>PUB</td>
</tr>
<tr>
<td>London</td>
<td>UK</td>
<td>PUB</td>
</tr>
<tr>
<td>Oxford</td>
<td>UK</td>
<td>CONF</td>
</tr>
<tr>
<td>Munich</td>
<td>DE</td>
<td>CONF</td>
</tr>
<tr>
<td>Roma</td>
<td>IT</td>
<td>CONF</td>
</tr>
<tr>
<td>Singapore</td>
<td>SG</td>
<td>SENS</td>
</tr>
<tr>
<td>Tokyo</td>
<td>JP</td>
<td>SENS</td>
</tr>
<tr>
<td>Beijing</td>
<td>CH</td>
<td>SENS</td>
</tr>
</tbody>
</table>

23 rows selected.

3. Repeat Steps 1 and 2 for users kpartner and ldoman.

User KPARTNER can access the rows labeled `CONF` and `PUB`:

<table>
<thead>
<tr>
<th>City</th>
<th>Country</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Venice</td>
<td>IT</td>
<td>PUB</td>
</tr>
<tr>
<td>Utrecht</td>
<td>NL</td>
<td>PUB</td>
</tr>
<tr>
<td>Bern</td>
<td>CH</td>
<td>PUB</td>
</tr>
<tr>
<td>Mexico City</td>
<td>MX</td>
<td>PUB</td>
</tr>
<tr>
<td>Hiroshima</td>
<td>JP</td>
<td>PUB</td>
</tr>
<tr>
<td>Southlake</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>South San Francisco</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>South Brunswick</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>Seattle</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>Toronto</td>
<td>CA</td>
<td>PUB</td>
</tr>
<tr>
<td>Whitehorse</td>
<td>CA</td>
<td>PUB</td>
</tr>
<tr>
<td>Bombay</td>
<td>IN</td>
<td>PUB</td>
</tr>
<tr>
<td>Sydney</td>
<td>AU</td>
<td>PUB</td>
</tr>
<tr>
<td>London</td>
<td>UK</td>
<td>PUB</td>
</tr>
<tr>
<td>Stratford</td>
<td>UK</td>
<td>PUB</td>
</tr>
<tr>
<td>Sao Paulo</td>
<td>BR</td>
<td>PUB</td>
</tr>
<tr>
<td>Geneva</td>
<td>CH</td>
<td>PUB</td>
</tr>
<tr>
<td>Oxford</td>
<td>UK</td>
<td>CONF</td>
</tr>
<tr>
<td>Munich</td>
<td>DE</td>
<td>CONF</td>
</tr>
</tbody>
</table>

23 rows selected.
User LDORAN can access the rows labeled PUB:

<table>
<thead>
<tr>
<th>City</th>
<th>Country</th>
<th>LABEL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Venice</td>
<td>IT</td>
<td>PUB</td>
</tr>
<tr>
<td>Hiroshima</td>
<td>JP</td>
<td>PUB</td>
</tr>
<tr>
<td>Southlake</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>South San Francisco</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>South Brunswick</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>Seattle</td>
<td>US</td>
<td>PUB</td>
</tr>
<tr>
<td>Toronto</td>
<td>CA</td>
<td>PUB</td>
</tr>
<tr>
<td>Whitehorse</td>
<td>CA</td>
<td>PUB</td>
</tr>
<tr>
<td>Bombay</td>
<td>IN</td>
<td>PUB</td>
</tr>
<tr>
<td>Sydney</td>
<td>AU</td>
<td>PUB</td>
</tr>
<tr>
<td>London</td>
<td>UK</td>
<td>PUB</td>
</tr>
<tr>
<td>Stratford</td>
<td>UK</td>
<td>PUB</td>
</tr>
<tr>
<td>Sao Paulo</td>
<td>BR</td>
<td>PUB</td>
</tr>
<tr>
<td>Geneva</td>
<td>CH</td>
<td>PUB</td>
</tr>
<tr>
<td>Bern</td>
<td>CH</td>
<td>PUB</td>
</tr>
<tr>
<td>Utrecht</td>
<td>NL</td>
<td>PUB</td>
</tr>
<tr>
<td>Mexico City</td>
<td>MX</td>
<td>PUB</td>
</tr>
<tr>
<td>Geneva</td>
<td>CH</td>
<td>PUB</td>
</tr>
</tbody>
</table>

Step 10: Optionally, Remove the Components for This Tutorial

Remove the components that you created for this tutorial.

To remove the components for this tutorial:

1. In Database Control, connect as user SYSTEM.
2. Click Server to display the Server subpage.
3. In the Security section, click Users.
4. Select user kpartner, and then click Delete.
5. In the Confirmation page, click Yes.
6. Repeat Step 4 and Step 5 for users ldoran and sking.
7. Click Server to display the Server subpage.
8. Click the Database Instance link to return to the Database Home page.
10. Select the role emp_role, and then click Delete.
11. In the Confirmation dialog box, click Yes.
12. Log out of Database Control, and then log back in as LABCSYS.
13. Click Server to display the Server subpage.
15. In the Label Security Policies page, in the Name field, enter ACCESS% and then click Go.
16. Ensure that ACCESS_LOCATIONS is selected, and then click Delete. In the Confirmation page, click Yes.

Deleting the ACCESS_LOCATIONS policy also drops the OLS_COLUMN column from the HR.LOCATIONS table.

Controlling Administrator Access with Oracle Database Vault

Oracle Database Vault enables you to restrict administrative access to an Oracle database. This helps you address the most difficult security problems remaining today: protecting against insider threats, meeting regulatory compliance requirements, and enforcing separation of duty.

- About Oracle Database Vault
- Tutorial: Controlling Administrator Access to the OE Schema

See Also: Oracle Database Vault Administrator’s Guide

About Oracle Database Vault

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 (HIPAA) Act. 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 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. You can associate a rule set with a realm, to further customize the type of protection you want for that realm. 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. 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 these components by using either Oracle Database Vault Administrator, or by using its PL/SQL packages.

**Tutorial: Controlling Administrator Access to the OE Schema**

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).

In this tutorial, you create a realm around the OE schema, which will protect it from administrator access. However, user SCOTT needs access to the OE.CUSTOMERS table, so you must ensure that he can continue to access this data.

In this tutorial:

- Step 1: Install and Register Oracle Database Vault, and Enable Its User Accounts
- Step 2: Grant the SELECT Privilege on the OE.CUSTOMERS Table to User SCOTT
- Step 3: Select from the OE.CUSTOMERS Table as Users SYS and SCOTT
- Step 4: Create a Realm to Protect the OE.CUSTOMERS Table
- Step 5: Test the OE Protections Realm
- Step 6: Optionally, Remove the Components for This Tutorial
Step 1: Install and Register Oracle Database Vault, and Enable Its User Accounts
This section contains:
■ Installing Oracle Database Vault
■ Registering Oracle Database Vault
■ Enabling Access to Database Control

Installing Oracle Database Vault
Oracle Database Vault is not installed in a default Oracle Database installation, but it is part of the products available in the Oracle Database installation media. You can install it into an existing database by using Oracle Universal Installer.

To install Oracle Database Vault:
1. Shut down the database instance in which you plan to install Oracle Database Vault.
   Log in to SQL*Plus as SYS, connecting with the SYSOPER privilege. At the SQL prompt, enter the following command:
   SHUTDOWN IMMEDIATE
2. Exit SQL*Plus.
   EXIT
3. Stop the Oracle Database processes.
   ■ UNIX: Go to the $ORACLE_HOME/bin directory and run the following commands to stop the Database Console and the listener:
     ./emctl stop dbconsole
     ./lsnrctl stop
   ■ Windows: In the Windows Services tool, right-click the Oracle listener, console, and database service services, and then from the menu, select Stop. The names of these services begin with Oracle and include the name of the database instance. For example, assuming the database instance is orcl, the names would be similar to the following:
     – OracleDBConsoleorcl
     – OracleJobSchedulerORCL
     – OracleOraDB1g-home1TNSListener
     – OracleServiceORCL
4. Run Oracle Universal Installer from the installation media.
   ■ UNIX: Use the following command:
     /mnt/cdrom/runInstaller
   ■ Windows: Double-click the file, setup.exe, on the installation media.
5. In the Select a Product to Install window, select Oracle Database 11g, and then click Next.
6. Select Advanced Installation, and then click Next.
   The Select Installation Type window appears.
7. Select **Custom**, and then click **Next**.
   The Specify Home Details screen appears.

8. Select the Oracle base directory and the Oracle home directory in which you want to install Oracle Database Vault. Click **Next**.
   By default, Oracle Universal Installer offers to create a new Oracle home for you, so ensure that you select the correct existing Oracle home. Oracle Universal Installer then verifies that your system meets the minimum requirements. Next, the Available Product Components window is displayed.

9. Select the box corresponding to **Oracle Database Vault option**.
   You can find this option under Enterprise Edition Options. You also must have Oracle Label Security installed, so Oracle Universal Installer selects it for you. Oracle Universal Installer also selects Oracle **Services For Microsoft Transaction Server**, but if you do not need this product, you can deselect it. Then click **Next**.
   The Summary window is displayed.

10. Review your choices and then click **Install**.
    The new products should include Oracle Database Vault J2EE Application, Oracle Database Vault option, and Oracle Label Security.
    After you click **Install**, the progress window is displayed. When the installation completes, Oracle Universal Installer displays the End of Installation window.

11. Click **Exit**, and then click **Yes** to confirm the exit.

12. Restart the services and the database instance in which you installed Oracle Database Vault.
   - **UNIX**: Go to the $ORACLE_HOME/bin directory and run the following commands to start the Database Console and the listener:
     ```
     ./emctl start dbconsole
     ./lsnrctl start
     ```
   - Start SQL"Plus and then restart the database instance:
     ```
     SQLPLUS "SYS/as SYSOPER"
     Enter password: password
     Connected to an idle instance
     SQL> STARTUP
     ```
   - **Windows**: In the Windows Services tool, right-click the Oracle listener, console, and database service services, and then from the menu, select **Start**.
     The names of these services begin with Oracle and include the name of the database instance. For example, assuming the database instance is **orcl**, the names would be similar to the following:
     - OracleDBConsoleorcl
     - OracleJobSchedulerORCL (Optional; you do not need to start it for the tutorials in this guide.)
     - OracleOraDB1g-home1TNSListener
     - OracleServiceORCL (This service starts when you start OracleDBConsole.)
Registering Oracle Database Vault

After you install Oracle Database Vault, you must register it with the database and then create its accounts.

To register Oracle Database Vault:

1. Start Database Configuration Assistant.

   - **UNIX**: Enter the following command at a terminal window:
     
     ```
     dbca
     ```

     Typically, `dbca` is in the `$ORACLE_HOME/bin` directory.

   - **Windows**: From the **Start** menu, click **All Programs**, then click **Oracle - ORACLE_HOME**, **Configuration and Migration Tools**, and then **Database Configuration Assistant**.

     Alternatively, you can start Database Configuration Assistant at a command prompt:

     ```
     dbca
     ```

     As with UNIX, typically, `dbca` is in the `ORACLE_BASE\ORACLE_HOME\bin` directory.

2. In the Welcome page, click **Next**.

   The Operations page appears.

3. Select **Configure Database Options**, and then click **Next**.

   The Database page appears.

4. From the list, select the database where you installed Oracle Database and then click **Next**.

   The Database Content page appears.

5. Select **Oracle Database Vault** (and **Oracle Label Security** if it is not already installed), and then click **Next**.

   If **Oracle Database Vault** is already checked and its name grayed out, then it has already been registered.

   After you select **Oracle Database Vault**, the Oracle Database Vault Credentials page appears.

6. Specify the name and password for the Database Vault Owner account (for example, `DBVOWNER`) and the Database Vault Account Manager (for example, `DBVACCTMGR`).

   Enter any password that is secure, according to the password guidelines described in "Requirements for Creating Passwords" on page 3-8. Oracle Database Vault has additional password requirements, which are displayed if you try to create an incorrect password.

7. Click **Next**.

   The Connection Mode page appears.

8. Select either **Dedicated Server Mode** or **Shared Server Mode** (depending on the selection you made when you created this database), click **Finish**, and then click **OK** in the confirmation prompts.
Database Configuration Assistant registers Oracle Database Vault, and then restarts the database instance.

9. Exit Database Configuration Assistant.

Enabling Access to Database Control

The Database Vault Account Manager and OE accounts must have the SELECT ANY DICTIONARY privilege to use Database Control.

To grant the SELECT ANY DICTIONARY privilege:

1. Log in to Database Control as the user SYS.
   - In the Login page, enter SYS and the password assigned to SYS. Set Connect As to SYSDBA. Select Login to log in. See Oracle Database 2 Day DBA for instructions about how to start Database Control.

2. Click Server to display the Server subpage.

   - The Users page appears.

4. Select the Database Vault Account Manager account, for example, DBVACCTMGR.
   - To quickly find DBVACCTMGR, enter DBV in the Object Name field, and then click Go.

5. With DBVACCTMGR selected, click Edit.
   - The Edit User page appears.

6. Click System Privileges to display the Edit User page.

7. Click Edit List.
   - The Modify System Privileges page appears.

8. In the Available System Privileges list, select SELECT ANY DICTIONARY, and then click Move to move it to the Selected System Privileges list. Then click OK.

9. Click Apply.

10. Repeat these steps to grant the SELECT ANY DICTIONARY privilege to user OE.

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

To test the tutorial later on, user SCOTT must select from the OE.CUSTOMERS table. First, you should ensure that he SCOTT account is active.

To enable user SCOTT:

1. Start Database Control.
   - See Oracle Database 2 Day DBA for instructions about how to start Database Control.

2. Connect as the Oracle Database Vault Account Manager account and connect as Normal.
   - After you install Oracle Database Vault, you no longer can use the administrative accounts 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.
However, administrative users still have the privileges they do need. For example, user SYS, who owns system privileges and quite a number of PL/SQL packages, can still grant privileges on these to other users.

3. Click Server to display the Server subpage.
   The Users page appears.
5. Select SCOTT from the list of users, and then click Edit.
   The Edit User page appears.
6. 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” on page 3-8.
   ■ Status: Click Unlocked.
7. Click Apply.
8. Click Logout.

To grant user SCOTT the SELECT privilege on the OE.CUSTOMERS table:
1. In the Login page of Database Control, log in as user OE.
   See Oracle Database 2 Day DBA for instructions about how to start Database Control.
2. Click Server to display the Server subpage.
   The Users page appears.
4. Select SCOTT and then click Edit.
   The Edit User page appears.
5. In the Edit User page, select the Object Privileges subpage.
   The Object Privileges subpage appears.
6. From the Select Object Type list, select Table, and then click Add.
   The Add Table Object Privileges page appears.
7. In the Select Table Objects field, enter OE.CUSTOMERS or use the flashlight icon to find this table.
8. Under Available Privileges, select SELECT and then click Move to move it to selected Privileges.
9. Click OK.
   The Edit User page appears.
10. Click Apply.

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, because SYS has administrative privileges and because SCOTT has an explicit SELECT privilege granted by user OE.
To select from OE.CUSTOMERS as users SYS and SCOTT:

1. Start SQL*Plus and connect as user SYS using the SYSDBA privilege

   SQLPLUS "SYS/AS SYSDBA"
   Enter password: password
   Connected.

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

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

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

1. Start Oracle Database Vault Administrator.

   In a browser, enter the following URL:

   https://host_name:port/dva

   Replace host_name with the name of the server on which you installed Oracle Database Vault, and port with the Oracle Enterprise Manager Console HTTPS port number. In most cases, the name of the server and port number are the same as those used by Database Control.

   If you cannot start Database Vault Administrator, you may need to manually deploy it. See Oracle Database Vault Administrator's Guide for more information.

2. In the Login to Database page, enter the following information:

   ■ User Name: Enter the name of the DV_OWNER account that you created when you installed Oracle Database Vault, for example, DEVOWNER.

   ■ Password: Enter the password of the user whose name you entered.

   ■ Host: Enter the host name or IP address of the computer where you installed Oracle Database Vault, for example, myserver.us.example.com.

   ■ Port: Enter the port number for the database, for example, 1521.

   ■ SID/Service: Enter either the SID (for example, orcl) of the database, or the service (for example, myserver.us.example.com).

   The Database Instance Administration page appears.
3. Under Database Vault Feature Administration, select Realms.
   The Realms page appears.
4. Click Create.
   The Create Realm page appears.
5. Enter the following information:
   ■ Name: OE Protections
   ■ Description: Realm to protect the OE schema
   ■ Status: Click Enabled.
   ■ Audit Options: Select Audit on Failure.
6. Click OK.
   The Realms page appears, with OE listed as a realm. However, it has no protected
   objects or authorized users yet.
7. Select the OE Protections realm and then click Edit.
   The Edit Realm page appears.
8. Under Realm Secured Objects, click Create.
   The Create Realm Secured Object page appears.
9. From the Object Owner list, select OE.
   OE is the account that owns the OE schema. Selecting the OE user ensures that this
   account can still maintain the OE schema tables.
10. From the Object Type list, select TABLE.
11. In the Object Name field, enter % to specify all tables within the OE schema, and
    then click OK.
    The Edit Realm page appears.
12. Under Realm Authorizations, click Create.
    The Create Realm Authorization page appears.
13. From the Grantee list, select OE, and then set the Authorization Type to Owner.
    Then set Authorization Rule Set to <Non Selected>.
    This authorizes the OE user to manage access to the objects within the OE schema.
    As an Owner, the OE user can grant or revoke realm-secured database roles, and
    access, manipulate, and create objects protected by the OE Protections realm.
    The Authorization Rule Set list enables you to select a rule that further controls
    access, such as the time the realm is in effect, and so on.
14. Click OK to return to the Edit Realm page, and then click OK again to return to
    the Realms page.
15. Click Logout to exit Oracle Database Vault Administrator.

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 privilege.
   ```sql
   CONNECT SYS/AS SYSDBA
   Enter password: password
   Connected.
   ```

2. Try selecting from the OE.CUSTOMERS table.
   ```sql
   SELECT COUNT(*) FROM OE.CUSTOMERS;
   ```
   The following output should appear:
   ```sql
   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.
   ```sql
   CONNECT SCOTT
   Enter password: password
   Connected.
   ```

4. Try selecting from the OE.CUSTOMERS table.
   ```sql
   SELECT COUNT(*) FROM OE.CUSTOMERS;
   ```
   The following output should appear:
   ```sql
   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 you need, but does not override the explicit privileges you have define. SCOTT still can query this table.

5. Exit SQL*Plus.
   ```sql
   EXIT
   ```

Step 6: Optionally, Remove the Components for This Tutorial
After completing this tutorial, you can remove the data structures that you used if you no longer need them.

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

1. Start Database Control.
   See Oracle Database 2 Day DBA for instructions about how to start Database Control.

2. Log in as the OE user.

3. In the Database Home page, click Server to display the Server subpage.

   The Users page appears.
5. Select SCOTT and then click Edit.
The Edit User page appears.
6. Click Object Privileges to display the Object Privileges subpage.
7. Select the SELECT object privilege for the OE.CUSTOMERS table, and then click Delete. Then click Apply.
8. Click Logout.

To revoke the SELECT ANY DICTIONARY privilege from user OE:
1. In Database Control, click Login.
The Login page appears.
2. Log in as user SYS and connect using the SYSDBA privilege.
The Database Control Home page appears.
3. Click Server, and then select Users from the Security list.
The Users page appears.
4. Select OE and then click Edit.
The Edit User page appears.
5. Click System Privileges, and then click Edit List.
The Modify System Privileges page appears.
6. From the Selected System Privileges list, select SELECT ANY DICTIONARY, and then click Remove. Then click OK, and then Apply.
7. Exit Database Control.

To drop the OE Protections realm:
1. Start Oracle Database Vault Administrator.
   See Step 1 in "Step 4: Create a Realm to Protect the OE.CUSTOMERS Table" on page 6-46 for how to start Database Vault Administrator.
2. Log in using the name of the DV_OWNER account that you created when you installed Oracle Database Vault, for example, DBVOWNER.
The Administration page appears.
3. Under Database Vault Feature Administration, click Realms.
The Realms page appears.
4. Select OE Protections from the list of realms, and then click Remove. Then click Yes in the Confirmation page.
5. Exit Oracle Database Vault Administrator.
About Auditing

Auditing is the monitoring and recording of selected user database actions. You can use standard auditing to audit SQL statements, privileges, schemas, objects, and network and multitier activity. In standard auditing, you use initialization parameters and the `AUDIT` and `NOAUDIT` SQL statements to audit SQL statements, privileges, and schema objects, as well as network and multitier activities.

There are also activities that Oracle Database always audits, regardless of whether or not auditing is enabled. These activities are administrative privilege connections, database startups, and database shutdowns. See Oracle Database Security Guide for more information.

Another type of auditing is fine-grained auditing. Fine-grained auditing enables you to audit at the most granular level, data access, and actions based on content, using Boolean measurement, such as `value > 1000`. You can use fine-grained auditing to audit activities based on access to or changes in a column. You can create security policies to trigger auditing when someone accesses or alters specified elements in an Oracle database, including the contents within a specified object. You can create policies that define specific conditions that must take place for the audit to occur. For example, you can audit a particular table column to find out when and who tried to access it during a specified period of time. Furthermore, you can create alerts that are

See Also:
- Oracle Database Security Guide for other ways that you can audit user and database activities
- Oracle Audit Vault Administrator’s Guide for information about Oracle Audit Vault, which provides advanced auditing features
Why Is Auditing Used?

Oracle Database records audit activities in audit records. Audit records provide information about the operation that was audited, the user performing the operation, and the date and time of the operation. Audit records can be stored in either a data dictionary table, called the database audit trail, or in operating system files, called an operating system audit trail. Oracle Database also provides a set of data dictionary views that you can use to track suspicious activities. See Oracle Database Security Guide for more information about these views.
When you use standard auditing, Oracle Database writes the audit records to either to
DBA_AUDIT_TRAIL (the sys.aud$ table), the operating system audit trail, or to the
DBA_COMMON_AUDIT_TRAIL view, which combines standard and fine-grained audit
log records.

In addition, the actions performed by administrators are recorded in the syslog audit
trail.

Auditing General Activities Using Standard Auditing

This section explains how to use standard auditing to audit activities performed on
SQL statements, privileges, schema objects, and network or multitier activities.

This section contains:
- About Standard Auditing
- Enabling or Disabling the Standard Audit Trail
- Using Default Auditing for Security-Relevant SQL Statements and Privileges
- Individually Auditing SQL Statements
- Individually Auditing Privileges
- Using Proxies to Audit SQL Statements and Privileges in a Multitier Environment
- Individually Auditing Schema Objects
- Auditing Network Activity
- Using Proxies to Audit SQL Statements and Privileges in a Multitier Environment
- Tutorial: Creating a Standard Audit Trail

See Also: Oracle Database Security Guide for detailed information about managing the standard audit trail

About Standard Auditing

In standard auditing, you enable auditing of SQL statements, privileges, schema
objects, and network or multitier activities. You can direct the audit for a specific
schema table if you want. To perform this type of audit, you use Database Control.

Standard audit records can be written either to DBA_AUDIT_TRAIL (the sys.aud$ table), the operating system audit trail, or to the DBA_COMMON_AUDIT_TRAIL view, which combines standard and fine-grained audit log records.

Enabling or Disabling the Standard Audit Trail

Before you perform the standard auditing procedures described in this section, you
must enable standard auditing. When you enable standard auditing, you can create
the audit trail in the database audit trail or write the audit activities to an operating
system file. If you write to an operating system file, you can create the audit record in
text or XML format.

To enable or disable the standard audit trail:
1. Start Database Control.
2. Log in as SYS and connect with the SYSDBA privilege.
   - User Name: SYS
3. Click Server to display the Server subpage.

4. In the Database Configuration section, click Initialization Parameters.
   The Initialization Parameters page appears.

5. Click SPFile to display the SPFile subpage.
   If the SPFile tab does not display in your installation, then you did not install Oracle Database using a server parameters file. Go to the next step.

6. In the Name field, enter audit_trail to find the AUDIT_TRAIL parameter, and then click Go.
   You can enter the first few characters of the parameter, for example, AUDIT_.
   Alternatively, you can scroll down the list of parameters to find the AUDIT_TRAIL parameter.

7. In the Value field, select one of the following values:
   - DB: Enables database auditing and directs all audit records to the database audit trail (SYS.AUD$), except for records that are always written to the operating system audit trail. (This value is the default if you created the database using Database Configuration Assistant. Otherwise, the default is NONE.)
   - OS: Enables database auditing and directs all audit records to an operating system file. If you are using a highly secure database configuration, Oracle recommends that you use this setting because it reduces the likelihood of a Denial of Service (DoS) attack. This setting also makes it easier to secure the audit trail. If the auditor is distinct from the database administrator, you must use the operating system setting. Any auditing information stored in the database is viewable and modifiable by the database administrator.
   - NONE: Disables standard auditing. (This value is the default if you created the database using a method other than Database Configuration Assistant. Otherwise, the default is DB.)
   - DB, EXTENDED: Performs all actions of the AUDIT_TRAIL=DB setting and also populates the SQL bind and SQL text CLOB-type columns of the SYS.AUDS table, when available. (These two columns are populated only when this parameter is specified.)
   - XML: Writes to the operating system audit record file in XML format. Prints all elements of the AuditRecord node except Sql_Text and Sql_Bind to the operating system XML audit file.
   - EXTENDED: Specifies XML, EXTENDED, which performs all actions of XML and also populates the SQL bind and SQL text CLOB-type columns of the SYS.AUDS table, whenever possible. (These columns are populated only when this parameter is specified.)

8. Click Apply.

9. Restart the Oracle Database instance:
   a. Click the Database Instance link.
b. Click **Home** to display the Database Control home page.

c. Under General, click **Shutdown**.

d. In the Startup/Shutdown Credentials page, enter your credentials.

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

e. After the shutdown completes, click **Startup**.

Note the following:

- You do not need to restart the database if you change the auditing of objects. You only need to restart the database if you made a universal change, such as turning on or off all auditing.

- You do not need to set **AUDIT_TRAIL** to enable either fine-grained auditing or **SYS** auditing. (SYS auditing enables you to monitor the activities of a system administrator. See *Oracle Database Security Guide* for more information.) For fine-grained auditing, you add and remove fine-grained auditing policies as necessary, applying them to the specific operations or objects you want to monitor. You can use the **AUDIT_SYS_OPERATIONS** parameter to enable and disable **SYS** auditing.

### Using Default Auditing for Security-Relevant SQL Statements and Privileges

This section explains how you can enable the Oracle-recommended audit parameters. It covers the following topics:

- About Default Auditing
- Enabling Default Auditing

**About Default Auditing**

When you create a new database or modify an existing database, you use the Security Settings window in Database Configuration Assistant (DBCA) to enable or disable its default security settings. This section explains how to start DBCA and enable the default security settings. Oracle recommends that you enable these settings. When these settings are enabled, Oracle Database audits some of the security-relevant SQL statements and privileges. It also sets the **AUDIT_TRAIL** initialization parameter to **DB**.

If you decide to use a different auditing option, for example, **OS** if you want to write the audit trail records to operating system files, you can do that: Oracle Database continues to audit the privileges that are audited by default. If you disable auditing by setting the **AUDIT_TRAIL** parameter to **NONE**, then no auditing takes place.

Oracle Database audits the **AUDIT ROLE** SQL statement by default. The privileges that are audited by default are as follows:

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Default Auditing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ALTER ANY PROCEDURE</strong></td>
<td>CREATE ANY LIBRARY</td>
</tr>
<tr>
<td><strong>ALTER ANY TABLE</strong></td>
<td>CREATE ANY PROCEDURE</td>
</tr>
<tr>
<td><strong>ALTER DATABASE</strong></td>
<td>CREATE ANY TABLE</td>
</tr>
<tr>
<td><strong>ALTER PROFILE</strong></td>
<td>CREATE ANY USER</td>
</tr>
<tr>
<td><strong>ALTER SYSTEM</strong></td>
<td>CREATE PUBLIC DB LINK</td>
</tr>
<tr>
<td><strong>ALTER USER</strong></td>
<td>CREATE SESSION</td>
</tr>
<tr>
<td><strong>AUDIT SYSTEM</strong></td>
<td>CREATE USER</td>
</tr>
<tr>
<td><strong>CREATE ANY JOB</strong></td>
<td>DROP ANY PROCEDURE</td>
</tr>
</tbody>
</table>

**AUDITING DATABASE ACTIVITY 7-6**
Oracle Database also audits all privileges and statements that have the BY ACCESS clause.

If you are concerned that auditing these statements and privileges will adversely affect your applications, you can disable auditing by using Database Configuration Assistant (DBCA). When you modify your applications to use auditing, you can reenable the default auditing of these statements and privileges.

Oracle strongly recommends that you enable auditing by default. Auditing is an effective method of enforcing strong internal controls so that your site can meet its regulatory compliance requirements, as defined in the Sarbanes-Oxley Act. This enables you to monitor business operations and catch any activities that may deviate from company policy. Doing so translates into tightly controlled access to your database and the application software, ensuring that patches are applied on schedule, and preventing ad hoc changes. By enabling auditing by default, you can generate an audit record for audit and compliance personnel. However, be aware that auditing may affect database performance.

See Also: Oracle Database SQL Language Reference for detailed information about the SQL statements described in this section and the AUDIT_TRAIL initialization parameter

Enabling Default Auditing
This section explains how to use Database Configuration Assistant to enable default auditing.

To enable the default profile security settings using Database Configuration Assistant:

1. Start Database Configuration Assistant:
   - UNIX: Enter the following command at a terminal window:
     
     dbca
     
     Typically, dbca is in the $ORACLE_HOME/bin directory
   - Windows: From the Start menu, click All Programs. Then click Oracle - ORACLE_HOME, Configuration and Migration Tools, and then Database Configuration Assistant.
     
     Alternatively, you can start Database Configuration assistant at a command prompt:
     
     dbca
     
     As with UNIX, typically, dbca is in the ORACLE_BASE\ORACLE_HOME\bin directory.

2. In the Welcome window, click Next.
   
The Operations window appears.

3. From the list, select Configure Database Options, and then click Next.
   
The Database window appears.

4. From the list, select the current database instance, and then click Next.
   
The Management Options window appears next.

5. Select Keep the database configured with Database Control.
The Security Settings page appears.

6. Select the security option you prefer, and then click Next.
   Oracle recommends that you take advantage of the enhanced security settings for this release.
   The Database Components page appears.

7. Click Next.
   The Connection Mode page appears.

8. Select either Dedicated Server Mode or Shared Server Mode (depending on the selection you made when you created this database), click Finish, and then click OK in the confirmation prompts.

Individually Auditing SQL Statements

The SQL statements that you can audit are in the following categories:

- **DDL statements.** For example, enabling the auditing of tables (AUDIT TABLE) audits all CREATE and DROP TABLE statements.
- **DML statements.** For example, enabling the auditing of SELECT TABLE audits all SELECT ... FROM TABLE/VIEW statements, regardless of the table or view.

Statement auditing can be broad or focused, for example, by auditing the activities of all database users or of only a select list of users.

**See Also:** Oracle Database Security Guide for detailed information about auditing SQL statements.

Individually Auditing Privileges

Privilege auditing is a way to audit statements that can use a system privilege, such as the SELECT ANY TABLE statement. You can audit the use of any system privilege. Similar to statement auditing, privilege auditing can audit the activities of all database users or of only a specified list. As with SQL statement auditing, you use the AUDIT and NOAUDIT statements to enable and disable privilege auditing. In addition, you must have the AUDIT SYSTEM system privilege before you can enable auditing.

Privilege audit options match the corresponding system privileges. For example, the option to audit use of the DELETE ANY TABLE privilege is DELETE ANY TABLE. For example:

```sql
AUDIT DELETE ANY TABLE BY ACCESS WHENEVER NOT SUCCESSFUL;
```

To audit all successful and unsuccessful uses of the DELETE ANY TABLE system privilege, enter the following statement:

```sql
AUDIT DELETE ANY TABLE;
```

To audit all unsuccessful SELECT, INSERT, and DELETE statements on all tables and unsuccessful uses of the EXECUTE PROCEDURE system privilege, by all database users, and by individual audited statement, issue the following statement:

```sql
AUDIT SELECT TABLE, INSERT TABLE, DELETE TABLE, EXECUTE PROCEDURE BY ACCESS
WHENEVER NOT SUCCESSFUL;
```

**See Also:** Oracle Database Security Guide for detailed information about auditing privileges.

Auditing Database Activity 7-7
Using Proxies to Audit SQL Statements and Privileges in a Multitier Environment

You can audit the activities of a client in a multitier environment by specifying a proxy in the Add Audited Statements or Add Audited Privileges page in Database Control. In a multitier environment, Oracle Database preserves the identity of the client through all tiers. Thus, you can audit actions performed on behalf of the client by a middle-tier application.

The middle tier can also set the user client identity in a database session, enabling the auditing of user actions through the middle-tier application. The user client identity then shows up in the audit trail.

You can use the SQL `AUDIT` statement to audit the activities of a client in a multitier environment. To do so, use the `BY PROXY` clause in the `AUDIT` statement.

For example, to audit `SELECT TABLE` statements issued on behalf of client `jackson` by the proxy application server `appserve`:

```
AUDIT SELECT TABLE BY jackson ON BEHALF OF appserve;
```

Afterward, user `jackson` can connect using the `appserve` proxy user as follows:

```
CONNECT appserve[jackson]
Enter password: password
```

See Also: Oracle Database Security Guide for detailed information about auditing in a multitier environment

Individually Auditing Schema Objects

Schema object auditing can audit all `SELECT` and `DML` statements permitted by schema object privileges, such as `SELECT` or `DELETE` statements on a particular table. The `GRANT` and `REVOKE` statements that control those privileges are also audited.

See Also: Oracle Database Security Guide for detailed information about auditing schema objects

Auditing Network Activity

You can use the `AUDIT` statement to audit unexpected errors in network protocol or internal errors in the network layer. The types of errors uncovered by network auditing are not connection failures, but can have several other possible causes. One possible cause is an internal event set by a database engineer for testing purposes. Other causes include conflicting configuration settings for encryption, such as the network not finding the information required to create or process expected encryption.

To enable network auditing:

1. Start SQL*Plus and log on with administrative privileges, such as `SYSTEM`, or as a security administrator. For example:

   ```
   SQLPLUS SYSTEM
   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 statement:

   ```
   AUDIT NETWORK;
   ```

   See Also: Oracle Database Security Guide for detailed information about auditing in a multitier environment
Tutorial: Creating a Standard Audit Trail

Suppose you wanted to audit SELECT statements on the OE.CUSTOMERS table. In this tutorial, you enable standard auditing, enable auditing for the SELECT SQL statement, run the SELECT SQL statement on the OE.CUSTOMERS table, and then check its audit file.

In this tutorial:
- Step 1: Log In and Enable Standard Auditing
- Step 2: Enable Auditing for SELECT Statements on the OE.CUSTOMERS Table
- Step 3: Test the Audit Settings
- Step 4: Optionally, Remove the Components for This Tutorial
- Step 5: Remove the SEC_ADMIN Security Administrator Account

Step 1: Log In and Enable Standard Auditing

First, log in, and, if necessary, enable standard auditing.

To enable standard auditing:
1. Start Database Control.
2. Log in as SYS and connect with the SYSDBA privilege.
   - User Name: SYS
   - Password: Enter your password.
   - Connect As: SYSDBA
3. Click Server to display the Server subpage.
4. In the Database Configuration section, click Initialization Parameters.
   The Initialization Parameters page appears.
5. Click SPFile to display the SPFile subpage.
   If the SPFile tab does not display in your installation, then you did not install Oracle Database using a server parameters file. Go to the next step.
6. In the Name field, enter AUDIT_TRAIL to find the AUDIT_TRAIL parameter, and then click Go.
   You can enter the first few characters of the parameter, for example, AUDIT. Alternatively, you can scroll down the list of parameters to find the AUDIT_TRAIL parameter.
7. In the Value field, select the DB (Database) option.
The DB option enables database auditing and directs all audit records to the database audit trail (SYS.AUD$), except for records that are always written to the operating system audit trail.

8. Click Apply.
9. Restart the Oracle Database instance.
   a. Click the Database Instance link.
   b. Click Home to display the Database Control home page.
   c. Under General, click Shutdown.
   d. In the Startup/Shutdown Credentials page, enter your credentials.
      See Oracle Database 2 Day DBA for more information.
   e. After the shutdown completes, click Startup.

Step 2: Enable Auditing for SELECT Statements on the OE.CUSTOMERS Table

Next, enable auditing for SELECT statements on the OE.CUSTOMERS table.

To enable auditing of SELECT statements for the OE.CUSTOMERS table:
1. Ensure that the sample user sec_admin exists.
   Log on as SYSTEM, and then from the Database Control home page, click Server to display the Server subpage. Select Users under Security, and check the list of accounts for sec_admin. "Step 1: Create a Security Administrator Account" on page 4-4 explains how to create the sec_admin security administrator account.
2. Grant sec_admin the SELECT privilege on the OE.CUSTOMERS table.
3. Log in to Database Control as user sec_admin.
4. Click Server to display the Server subpage.
5. In the Security section, click Audit Settings.
   The Audit Settings page appears.
6. Select the Audited Objects subpage.
7. Click Add.
   The Add Audited Object page appears.
8. Enter the following information:
   • Object Type: Select Table.
   • Table: Enter OE.CUSTOMERS.
   • Available Statements: Select SELECT, and then click Move to move it to the Selected Statements list.
9. Click OK.
10. Shut down the database instance and then restart it.
    a. In the upper, right corner of the Database Control page, select Logout.
    b. Click Login.
    c. In the Login page, enter the following login information:
       User Name: SYS
Password: The password of the system administrator

Connect As: SYSDBA

Use the SYSDBA system privilege to shut down and restart the database.

d. Under General, click Shutdown.

e. In the Startup/Shutdown Credentials page, enter your credentials.

See Oracle Database 2 Day DBA for more information.

f. After the shutdown completes, click Startup.

g. Exit Database Control.

Step 3: Test the Audit Settings

At this stage, auditing is enabled and any SELECT statements performed on the OE.CUSTOMERS table are written to the DBA_AUDIT_TRAIL view. Now, you are ready to test the audit settings.

To test the audit settings:

1. Start SQL*Plus, and connect as user sec_admin.

   SQLPLUS sec_admin
   Enter password: password

2. Enter the following SELECT statement to create an alert in the audit trail:

   SELECT COUNT(*) FROM oe.customers;

3. Enter the following statement to view the DBA_AUDIT_TRAIL view:

   SELECT USERNAME, TIMESTAMP FROM DBA_AUDIT_TRAIL;

   Output similar to the following should display:

   USERNAME          TIMESTAMP
   ---------------------------
   SEC_ADMIN         07-MAY-08

4. Exit SQL*Plus:

   EXIT

Step 4: Optionally, Remove the Components for This Tutorial

Optionally, remove the audit settings you created earlier.

To remove the audit settings in Database Control:

1. Log in to Database Control using administrative privileges.

2. Go to the Database Control home page.

3. Click Server to display the Server subpage.

4. In the Security section, click Audit Settings.

   The Audit Settings page appears.

5. Select the Audited Objects subpage.

6. Under Schema, enter OE.
Guidelines for Auditing

7. Under Object Name, enter CUSTOMERS.
8. Click Search.
9. Select the box next to the OE.CUSTOMERS audited schema, and then click Remove. A Confirmation dialog box appears.
10. Select Yes.
11. Exit Database Control.

To set AUDIT_TRAIL to its original value:
- Follow the procedure in "Step 1: Log In and Enable Standard Auditing" on page 7-9 to log in to SQL*Plus and set the AUDIT_TRAIL parameter back to its original value. Afterward, shut down and then restart the database.

Step 5: Remove the SEC_ADMIN Security Administrator Account

This is the last example in this guide. If you no longer need the sec_admin administrator account, you should remove it.

To remove the sec_admin security administrator account:
1. Log in to Database Control using administrative privileges.
2. Go to the Database Control home page.
3. Click Server to display the Server subpage.
4. In the Security section, click Users.
   The Users page appears.
5. In the Name field, enter sec_admin.
6. Click Search.
7. Select the box next to the sec_admin user account, and then click Remove. A Confirmation dialog box appears.
8. Select Yes.
9. Exit Database Control.

Guidelines for Auditing

This section contains the following topics:
- Guideline for Using Default Auditing of SQL Statements and Privileges
- Guidelines for Managing Audited Information
- Guidelines for Auditing Typical Database Activity
- Guidelines for Auditing Suspicious Database Activity

Guideline for Using Default Auditing of SQL Statements and Privileges

When you create a new database, you can enable the auditing of a select set of SQL statements and privileges. Oracle recommends that you enable default auditing. Auditing is an effective method of enforcing strong internal controls so that your site meets its regulatory compliance requirements, as defined in the Sarbanes-Oxley Act.
Guidelines for Auditing

See "Using Default Auditing for Security-Relevant SQL Statements and Privileges" on page 7-5 for more information about default auditing.

Guidelines for Managing Audited Information

Although auditing does not severely affect database performance, limit the number of audited events as much as possible. This minimizes the performance impact on the execution of audited statements and the size of the audit trail, making it easier to analyze and understand.

Follow these guidelines when devising an auditing strategy:

1. **Evaluate your reason for auditing.**
   
   After you understand the reasons for auditing, you can devise an appropriate auditing strategy and avoid unnecessary auditing.
   
   For example, suppose you are auditing to investigate suspicious database activity. This information by itself is not specific enough. What types of suspicious database activity do you suspect or have you noticed? A more focused auditing purpose might be to audit unauthorized deletions from arbitrary tables in the database. This purpose narrows the type of action being audited and the type of object being affected by the suspicious activity.

2. **Audit knowledgeably.**

   Audit the minimum number of statements, users, or objects required to get the targeted information. This prevents unnecessary audit information from cluttering the meaningful information and using valuable space in the SYSTEM tablespace. Balance your need to gather sufficient security information with your ability to store and process it.

   For example, if you are auditing to gather information about database activity, then determine exactly what types of activities you want to track, audit only the activities of interest, and audit only for the amount of time necessary to gather the information that you want. As another example, do not audit objects if you are only interested in logical I/O information for each session.

Guidelines for Auditing Typical Database Activity

When your purpose for auditing is to gather historical information about particular database activities, follow these guidelines:

1. **Audit only pertinent actions.**

   To avoid cluttering meaningful information with useless audit records and to reduce the amount of audit trail administration, audit only the targeted database activities. You can audit specific actions by using fine-grained auditing. *Oracle Database Security Guide* describes fine-grained auditing in detail.

2. **Archive audit records and purge the audit trail.**

   After you collect the required information, archive the audit records of interest, and purge the audit trail of this information.

   To archive audit records, you copy the relevant records to a database table, for example, using `INSERT INTO table SELECT ... FROM SYS.AUDS ...` for the standard audit trail. (Fine-grained audit records are in the `SYS.FGA_LOG$` table.) Alternatively, you can export the audit trail table to an operating system file. *Oracle Database Utilities* explains how to export tables by using Oracle Data Pump.
To purge audit records, you delete standard audit records from the `SYS.AUD$` table and fine-grained audit records from the `SYS.FGA_LOG$` table. For example, to delete all audit records from the standard audit trail, enter the following statement:

```
DELETE FROM SYS.AUD$;
```

Alternatively, to delete all audit records from the standard audit trail generated as a result of auditing the table `emp`, enter the following statement:

```
DELETE FROM SYS.AUD$
WHERE obj$name='EMP';
```

3. Remember the privacy considerations of your company.

Privacy regulations often lead to additional business privacy policies. Most privacy laws require businesses to monitor access to personally identifiable information (PII), and this type of monitoring is implemented by auditing. A business-level privacy policy should address all relevant aspects of data access and user accountability, including technical, legal, and company policy concerns.

### Guidelines for Auditing Suspicious Database Activity

When you audit to monitor suspicious database activity, follow these guidelines:

1. **Audit general information, and then audit specific information.**

   When you start to audit for suspicious database activity, often not much information is available to target specific users or schema objects. Therefore, set audit options more generally at first, that is, by using the standard audit options described in "Auditing General Activities Using Standard Auditing" on page 7-3.

   After you have recorded and analyzed the preliminary audit information, disable general auditing, and then audit specific actions. You can use fine-grained auditing, described in Oracle Database Security Guide, to audit specific actions. Continue this process until you gather enough evidence to draw conclusions about the origin of the suspicious database activity.

2. **Protect the audit trail.**

   When auditing for suspicious database activity, protect the audit trail so that audit information cannot be added, changed, or deleted without being audited. You audit the standard audit trail by using the `AUDIT SQL` statement. For example:

   ```sql
   SQLPLUS 'SYS/AS SYSDBA'
Enter password: password
   SQL> AUDIT SELECT ON SYS.AUD$ BY ACCESS;
   ```
Initialization Parameters Used for Auditing

Table 7-1 lists initialization parameters that you can use to secure auditing.

<table>
<thead>
<tr>
<th>Initialization Parameter</th>
<th>Default Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUDIT_TRAIL</td>
<td>DB</td>
<td>Enables or disables auditing. See &quot;Enabling or Disabling the Standard Audit Trail&quot; on page 7-3 for detailed information.</td>
</tr>
<tr>
<td>AUDIT_FILE_DEST</td>
<td>ORACLE_BASE/admin/ORACLE_SID/adump or ORACLE_HOME/rdbms/audit</td>
<td>Specifies the operating system directory into which the audit trail is written when the AUDIT_TRAIL initialization parameter is set to OS, XML, or XML, EXTENDED. Oracle Database writes the audit records in XML format if the AUDIT_TRAIL initialization parameter is set to XML. Oracle Database also writes mandatory auditing information to this location, and if the AUDIT_SYS_OPERATIONS initialization parameter, writes audit records for user SYS.</td>
</tr>
<tr>
<td>AUDIT_SYS_OPERATIONS</td>
<td>FALSE</td>
<td>Enables or disables the auditing of operations issued by user SYS, and users connecting with SYSDBA or SYSOPER privileges. Oracle Database writes the audit records to the audit trail of the operating system. Furthermore, it writes the audit records in XML format if the AUDIT_TRAIL initialization parameter is set to XML or XML, EXTENDED. On UNIX systems, if you have also set the AUDIT_SYSLOG_LEVEL parameter, then it overrides the AUDIT_TRAIL parameter, which writes the SYS audit records to the system audit log using the SYSLOG utility.</td>
</tr>
<tr>
<td>AUDIT_SYSLOG_LEVEL</td>
<td>No default setting</td>
<td>On UNIX systems, writes the SYS and standard OS audit records to the system audit log using the SYSLOG utility.</td>
</tr>
</tbody>
</table>

To modify an initialization parameter, see "Modifying the Value of an Initialization Parameter" on page 2-6. For detailed information about initialization parameters, see Oracle Database Reference and Oracle Database Administrator’s Guide.
Initialization Parameters Used for Auditing
Index

A
access control
  data encryption, 6-2
  enforcing, 5-1
Oracle Label Security, 6-21
administrative accounts
  about, 3-2
  access, 5-3
  passwords, 3-10
  predefined, listed, 3-2
administrators
  privileges for listener.ora file, 5-3
  restricting access of, 6-39
  separation of duty, 6-39
ANONYMOUS user, 3-2
ANY system privilege, protecting data dictionary, 2-4
APEX_PUBLIC_USER user, 3-5
application contexts
  Oracle Virtual Private Database, used with, 6-12
audit files
  archiving and purging, 7-13
  operating system file, writing to, 7-4
audit records
  types, 7-2
  viewing, 7-2
audit trail
  DB setting, 7-4
  XML file output, 7-4
auditing
  about, 7-1
  DDL statements, 7-7
  default security setting, modified by, 7-5
  DML statements, 7-7
  fine-grained auditing, 7-1
  guidelines, security, 7-12
  historical information, 7-13
  keeping information manageable, 7-13
  monitoring user actions, 7-1
  privilege audit options, 7-7
  reasons to audit, 7-2
Sarbanes-Oxley Act
  default auditing, 7-12
  requirements, 7-6
  suspicious activity, 7-14
  viewing audit records, 7-2
  where recorded, 7-2
authentication
  certificate, 5-2
  client, 5-1, 5-2
  remote, 5-1, 5-2
  strong, 3-11
  user, 5-2
AUTHID CURRENT USER invoker's rights clause, 4-9
B
BFILEs
  restricting access, 2-5
  BI user, 3-7
C
certificate authentication, 5-2
certificates for user and server authentication, 5-2
client connections
  stolen, 5-1
  client guidelines, 5-1
configuration files
  listener.ora
    administering listener remotely, 5-3
    sample, 5-3
  CONNECT role, privilege available to, 4-2
  CONNECT statement
    AS SYSDBA privilege, connecting with, 2-5
    connections
    AS SYSDBA privilege, 2-5
    securing, 5-1
  SYS user, 4-2
  CREATE ANY TABLE statement, 4-2
  CREATE DRLINK statement, 4-2
  CREATE EXTERNAL JOB privilege
    default security setting, modified by, 2-1
  CREATE SESSION statement, 4-2
  CREATE TABLE statement, auditing, 7-7
 CTXSYS user, 3-2
D
data definition language
auditing, 7-7
data dictionary
about, 2-3
securing, 2-4
data dictionary views
DBA_USERS, 3-11
DBA_USERS_WITH_DEFPWD, 3-9
data files
restricting access, 2-5
data manipulation language, auditing, 7-7
database
restarting, 7-11
shutting down, 7-11
starting, 7-11
database accounts
See user accounts
Database Configuration Assistant
auditing by default, 7-5
default passwords, changing, 3-10
Oracle Database Vault, installing, 6-41
Oracle Label Security, installing, 6-23
Database Control
See Oracle Enterprise Manager Database Control
DBA_USERS data dictionary view, 3-11
DBA_USERS_WITH_DEFPWD data dictionary view, 3-9
DBCA
See Database Configuration Assistant
DBSNMP user
about, 3-2
passwords, default, 3-10
default passwords
administrative accounts, using with, 3-10
default permissions, 2-5
default security settings
about, 2-1
enabling, 2-2
Denial of Service (DoS) attacks
audit trail, writing to operating system file, 7-4
networks, addressing, 5-5
See also security attacks
DIP user, 3-5
disabling unnecessary services, 5-5
DROP ANY TABLE statement, 2-5
DROP TABLE statement, auditing, 7-7
encryption
about, 6-2
algorithms, described, 5-7
components, 6-2
data transfer, 5-4
network, 5-5 to 5-8
network traffic, 5-5
reasons not to encrypt, 6-2
reasons to encrypt, 6-2
Enterprise Edition, 3-11
errors
WHEN NO_DATA_FOUND exception
example, 4-10
examples
user session information, retrieving with SYS_CONTEXT, 6-16
See also tutorials
exceptions
WHEN NO_DATA_FOUND example, 4-10
EXECUTE privilege, 4-2
EXFSYS user, 3-3
external tables, 2-5
F
files
audit archiving, 7-13
DOS attacks, recommendations, 7-4
configuration, 5-3
listener.ora, 5-3
restrict listener access, 5-4
restricting access, 2-5
symbolic links, restricting, 2-5
fine-grained auditing, 7-1
firewalls
Avant, 5-4
CheckPoint, 5-4
Cisco, 5-4
database server, keeping behind, 5-4
Firewall-1, 5-4
Gauntlet, 5-4
guidelines, 5-4
Network Associates, 5-4
PIX Firewall, 5-4
Raptor, 5-4
supported
packet-filtered, 5-4
proxy-enabled, 5-4
FLOWs_30000 user, 3-5
FLOWs_FILES user, 3-5
FTP service
disabling, 5-5
G
GRANT ALL PRIVILEGES privilege, 2-5
guidelines for security
auditing
audited information, managing, 7-13
database activity, typical, 7-13
default auditing, 7-12
database activity, suspicious, 7-14
database connections, 5-1
network connections, 5-2
operating access to database, 2-5
operating system accounts, limiting privileges, 2-5
operating system users, limiting number of, 2-5
Oracle home default permissions, disallowing modifying of, 2-5
Oracle Label Security policies, planning, 6-21
passwords
administrative, 3-10
creating, 3-8
management, enforcing, 3-11
privileges, granting, 4-1
PUBLIC user group, privileges, 4-2
roles, granting to users, 4-2
run-time facilities, granting permissions to, 2-5
symbolic links, restricting, 2-5

HR user, 3-7

identity theft
See security attacks
initialization parameters
AUDIT_FILE_DESTINATION, 7-15
AUDIT_SYS_OPERATIONS, 7-15
AUDIT_SYS_ADDR, 7-15
AU/DIT_TRAIL, 7-15
configuration related, 2-6
default security, modified by, 2-2
FAILED_LOGIN_ATTEMPTS, 3-12
installation related, 2-6
MAX_ENABLED_ROLES, 4-12
modifying, 2-6
O7_DICTIONARY_ACCESSIBILITY about, 2-6
data dictionary, protecting, 2-4
default setting, 2-5
setting in Database Control, 2-4
OS_AUTHTYPE, 5-8
OS_ROLES, 4-12
PASSWORD_USAGE_TIME, 3-12
PASSWORD_USAGE_TIME, 3-12
PASSWORD_REUSE_MAX, 3-12
PASSWORD_REUSE_TIME, 3-12
REMOTE_LISTENER, 5-8
REMOTE_OS_AUTHTYPE, 5-2, 5-8
REMOTE_OS_AUTHTYPE, 5-2, 5-8
SEC_CASE_SENSITIVE_LOGIN, 3-12
SEC_MAX_FAILED_LOGIN_ATTEMPTS, 3-12
SEC_RETURN_SERVER_RELEASE_BANNER, 2-6
SQLID_SECURITY, 4-12
invoker’s rights, 4-9
IP addresses
falsifying, 5-4
guidelines, 5-2
IX user, 3-7

Kerberos authentication
password management, 3-11

LBACSYS user, 3-3
least privilege principle, 4-1
listener
not an Oracle owner, 5-4
preventing online administration, 5-3
restrict privileges, 5-4
secure administration, 5-4
listener.ora file
administering remotely, 5-3
online administration, preventing, 5-3
log files
restricting access, 2-5

MDDATA user, 3-6
MDSYS user, 3-3
MGMT_VIEW user, 3-3
monitoring
See auditing
multiplex multiple-client network connections, 5-4
multithread environments, auditing, 7-8

Net8 network utility
See Oracle Net
network activity
auditing, 7-8
network authentication services, 3-11
smart cards, 3-11
token cards, 3-11
X.509 certificates, 3-11
network encryption
about, 5-5
components, 5-5
crypting, 5-6
network IP addresses, 5-4
network security
Denial of Service attacks, addressing, 5-5
guidelines for clients, 5-1

object privileges, 4-2
OE user, 3-7
OLAPSYS user, 3-3
operating system access, restricting, 2-5
operating system account privileges, limiting, 2-5
operating systems
limiting number of, 2-5
operating systems
compromised, 5-1
default permissions, 2-5
Oracle Advanced Security
authentication protection, 3-11
network traffic encryption, 5-5
Oracle Connection Manager
firewall all configuration, 5-4
Oracle Database Vault

Index-3
about, 6-39
components, 6-39
installing, 6-41
registering with database, 6-43
tutorial, 6-40 to 6-49
Oracle Enterprise Manager Database Control
about, 1-2
starting, 2-4
Oracle home
default permissions, disallowing modifying of
Oracle Java Virtual Machine (OJVM), 2-5
Oracle Label Security (OLS)
about, 6-20
components, 6-21
guidelines in planning, 6-21
how it works, 6-21
installing, 6-23
tutorial, 6-23 to 6-39
Oracle Net
encrypting network traffic, 5-6
firewall support, 5-4
Oracle Virtual Private Database (VPD)
about, 6-11
advantages, 6-13
application contexts, 6-12
components, 6-12
tutorial, 6-13 to 6-20
Oracle Wallet Manager
wallet, creating, 6-4
with transparent data encryption, 6-5
ORACLE_OCM user, 3-6
ORDPLUGINS user, 3-3
ORDSYS user, 3-3
OUTLN user, 3-4
OWBSYS user, 3-3
passwords
administrative, 3-10
administrative user, 3-10
changing, 3-10
complexity, 3-11
default security setting, modified by, 2-1
default user account, 3-9
history, 3-11
length, 3-11
management, 3-11
management rules, 3-11
profiles
administering default settings, 7-6
SYS user, 3-10
SYSTEM user, 3-10
for security
requirements, 3-8
permissions
default, 2-5
run-time facilities, 2-5
PM user, 3-7
privileges
about, 4-1
auditing, 7-7
CREATE DBLINK statement, 4-2
monitor, 2-4
DROP ANY TABLE, 2-5
SELECT ANY DICTIONARY, 2-5
SYSTEM and OBJECT, 4-2
using proxies to audit, 7-8
PUBLIC user, 3-6
PUBLIC user group, revoking unnecessary privileges and roles, 4-2
remote authentication, 3-1, 3-2
REMOTE_OAUTH parameter, 5-2
roles
CONNECT, 4-2
create your own, 4-2
job responsibility privileges only, 4-2
root file paths for files and packages outside the database, 2-5
run-time facilities, restricting permissions, 2-5
Sarbanes-Oxley Act
auditing requirements, 7-6
debugging, 7-12
schema objects, auditing, 7-8
SCOTT user
about, 3-7
restricting privileges of, 4-2
sec_admin example security administrator
creating, 4-4
removing, 7-12
secure application roles
about, 4-3
advantages, 4-3
components, 4-3
invoker’s rights, 4-9
tutorial, 4-4 to 4-12
user environment information from SYS_CONTEXT SQL function, 4-9
Secure Sockets Layer (SSL)
administering listener remotely, 5-3
certificates, enabling for user and server, 5-2
security administrator
example of creating, 4-4
removing sec_admin, 7-12
security attacks
applications, 5-1
client connections, 5-1
Denial of Service, 5-5
eavesdropping, 5-2
falsified IP addresses, 5-1
falsified or stolen client system identities, 5-1
network connections, 5-2
security tasks, common, 1-2
SELECT ANY DICTIONARY privilege
data dictionary, accessing, 2-5
GRANT ALL PRIVILEGES privilege, not included in, 2-5
sensitive data
Oracle Label Security, 6-20
secure application roles, 4-3
separation of duty concepts, 4-4
separation-of-duty principles about, 6-39
Oracle Database Vault, 6-44
session information, retrieving, 6-12
SH user, 3-7
SI_INFORMTN_SCHEMA user, 3-4
smart cards, 3-11
SPATIAL_CSW_ADMIN_USR user, 3-6
SPATIAL_WFS_ADMIN_USR user, 3-6
SQL statements
auditing, 7-7
using proxies to audit, 7-8
SQL*Net network utility, 5-4
standard auditing
about, 7-3
auditing by default, 7-5
enabling or disabling audit trail, 7-3
in multitenant environment, 7-8
network activity, 7-8
privileges, 7-7
proxies, 7-8
schema objects, 7-8
SQL statements, 7-7
tutorial, 7-9 to 7-12
strong authentication, 3-11
symbolic links, restricting, 2-5
SYS user
about, 3-4
password use, 3-10
SYS_CONTEXT SQL function
example, 6-16
validating users, 4-9
SYSAUD database audit trail table
about, 7-4
DB (database) option, 7-10
DB, EXTENDED option, 7-4
XML, EXTENDED option, 7-4
SYSDBA system privilege, 7-11
SYSMAN user
about, 3-4
password use, 3-10
passwords, default, 3-10
SYS-privileged connections, 4-2
system administrator
See administrative accounts, security administrator
system identities, stolen, 5-1
system privileges, 4-2

ANY, 2-4
DROP ANY TABLE statement, 2-5
SELECT ANY DICTIONARY, 2-5
SYSTEM user
about, 3-4
password use, 3-10
T
tablespace
encrypting, 6-8
TCP ports
closing for ALL disabled services, 5-5
TCPS protocol
Secure Sockets Layer, used with, 5-3
TDE
See transparent data encryption
TELNET service, disabling, 5-5
TFTP service
disabling, 5-5
token cards, 3-11
trace files restricting access, 2-5
transparent data encryption
about, 6-3
advantages, 6-3
components, 6-3
configuring, 6-4
how it works, 6-3
performance effects, 6-4
storage space, 6-4
table columns
checking in database instances, 6-10
checking individual tables, 6-10
encrypting, 6-6
tablespace
checking, 6-10
tablespace, encrypting, 6-8
wallets, 6-5
TSM_SYS user, 3-4
tutorials
Oracle Database Vault, 6-40 to 6-49
Oracle Label Security, 6-23 to 6-39
Oracle Virtual Private Database, 6-13 to 6-20
secure application roles, 4-4 to 4-12
standard auditing, 7-9 to 7-12
U
UDP ports
closing for ALL disabled services, 5-5
user accounts
about, 3-1
administrative user passwords, 3-10
default, changing password, 3-9
expiring, 3-7
finding information about, 3-11
locking, 3-7
password requirements, 3-8
predefined administrative, 3-2

Index-5
non-administrative, 3-5
sample schema, 3-6
securing, ?? to 3-12
unlocking, 3-7
user accounts, predefined
  ANONYMOUS, 3-2
  APEX_PUBLIC_USER, 3-5
  BI, 3-7
  CTXSYS, 3-2
  DBSNMP, 3-2
  DIP, 3-5
  EXPSYS, 3-3
  FLOWS_30000, 3-5
  FLOWS_FILES, 3-5
  HR, 3-7
  IX, 3-7
  LIBACSYS, 3-3
  MDDATA, 3-6
  MDSYS, 3-3
  MGMT_VIEW, 3-3
  OE, 3-7
  OLAPSYS, 3-3
  ORACLE_OCM, 3-6
  ORDPPLUGINS, 3-3
  ORDSYS, 3-3
  OUTLN, 3-4
  OWBSYS, 3-3
  PM, 3-7
  PUBLIC, 3-6
  SCOTT, 3-7, 4-2
  SH, 3-7
  SI_INFORMTN_SCHEMA, 3-4
  SPATIAL_CSW_ADMIN_USR, 3-6
  SPATIAL_WFS_ADMIN_USR, 3-6
  SYS, 3-4
  SYSDMB, 3-4
  SYSTEM, 3-4
  TSMSYS, 3-4
  WK_TEST, 3-4
  WKPROXY, 3-5
  WKSYS, 3-5
  WMSYS, 3-5
  XDB, 3-5
  XS$NULL, 3-6
user session information, retrieving, 6-12

V
valid node checking, 5-4
views
See data dictionary views
Virtual Private Database
See Oracle Virtual Private Database
VPD
See Oracle Virtual Private Database
vulnerable run-time call, 2-5
made more secure, 2-6

W
WK_TEST user, 3-4
WKPROXY user, 3-5
WKSYS user, 3-4
WMSYS user, 3-5

X
X.509 certificates, 3-11
XDB user, 3-5
XS$NULL user, 3-6