Oracle®

Data Masking and Subsetting Guide

Enterprise Manager 13*c*

E85965-01

May 2017

Oracle Data Masking and Subsetting Guide, Enterprise Manager 13*c*

E85965-01

Primary Authors: Preethy P G, Bert Rich

Contributing Authors: Jim Garrison, Tulika Das, Dinesh Rajasekharan

Contributors: Eric Paapanen, Vipin Samar, Gopal Mulagund, John Kati, Amoghavarsha Ramappa

Contents

Preface

Changes in This Release for Oracle Data Masking and Subsetting User’s Guide

1 Introduction to Oracle Data Masking and Subsetting

2 Before You Begin

List of Figures

List of Tables

Preface

This preface contains the following topics:

Audience

This document provides information about how to mask and subset data for non-production usage such as development and testing using Oracle Data Masking and Subsetting Pack (DMS) of Oracle Enterprise Manager (EM) Database Plug-in. This document is intended for database administrators (DBAs), database designers, application administrators, application owners, business owners, testers, and developers.

Documentation Accessibility

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

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Related Documents

For more information about some of the topics discussed in this document, see the following documents in the Oracle documentation set:

- *Oracle Database 2 Day DBA*
- *Oracle Database 2 Day + Performance Tuning Guide*
- *Oracle Database Administrator's Guide*
- *Oracle Database Concepts*
- *Oracle Database Performance Tuning Guide*
- *Oracle Database SQL Tuning Guide*

- *Oracle Database Installation Guide for Linux*
- *Oracle Enterprise Manager Command Line Interface*
- *Oracle Enterprise Manager Installation Guide*
- *Oracle Enterprise Manager Advanced Installation Guide*
- *Oracle Enterprise Manager Administration Guide*
- *Enterprise Manager Licensing Guide*

Conventions

The following text conventions are used in this document:

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

Changes in This Release for Oracle Data Masking and Subsetting User's Guide

This preface contains:

- Versioning of Oracle Data Masking and Subsetting
- Changes in Oracle Data Masking and Subsetting Release 13.2
- Changes in Oracle Data Masking and Subsetting Release 13.1

Versioning of Oracle Data Masking and Subsetting

The Oracle Data Masking and Subsetting version is based on the version of the Oracle Enterprise Manager Database Plug-in.

Oracle Data Masking and Subsetting is a part of the Oracle Enterprise Manager Database Plug-in. In general, multiple Oracle Enterprise Manager Database Plug-in versions are released during the life cycle of the Oracle Enterprise Manager version. For example, Oracle Enterprise Manager Database Plug-in versions 12.1.0.6 and 12.1.0.7 were released during the life cycle of the Oracle Enterprise Manager version 12.1.0.4 and Oracle Enterprise Manager Database Plug-in version 13.1.0.0 was released during the life cycle of the Oracle Enterprise Manager version 13.1.

Changes in Oracle Data Masking and Subsetting Release 13.2

This section describes the features introduced in Oracle Data Masking and Subsetting Release 13.2.

- Enhanced Support for Discovering Non-Dictionary Based Referential Relationships
- Support for Handling Object Names With Size up to 128 Bytes
- Support for Importing and Exporting the Complete Database
- Support for Retaining the Existing Schema While Importing the Subset Dump

Enhanced Support for Discovering Non-Dictionary Based Referential Relationships

With Oracle Data Masking and Subsetting release 13.2, the ability to discover non-dictionary based referential relationships has been enhanced. Users can now discover potential non-dictionary based referential relationships by matching column values, in addition to column names. For more information on using this feature, see the Creating an ADM section**.**

Support for Handling Object Names With Size up to 128 Bytes

The repository tables in Oracle Data Masking and Subsetting are now compatible to store the target database object names with size up to 128 bytes. Prior releases of Oracle Data Masking and Subsetting could handle target database object names up to 30 bytes only.

Support for Importing and Exporting the Complete Database

Oracle Data Masking and Subsetting provides the flexibility to import and export the complete database while simultaneously masking or subsetting some schemas in the database. When a user chooses a Full database In-Export data masking option, the tables in the masking definition are exported as masked, and the remaining tables are exported as they are. Similarly, when a user chooses a Full database In-Export data subsetting option, the tables in the subset definition are subsetted based on the object rules defined in the subset definition, and the remaining tables are exported without applying subsetting.

Related Topics:

- Data Masking
- Data Subsetting

Support for Retaining the Existing Schema While Importing the Subset Dump

Oracle Data Masking and Subsetting allows users to retain the original tables of the schema by selecting the schema remap option while importing a subset dump. In prior releases of Oracle Data Masking and Subsetting, the existing schema was being dropped when users selected the schema remap option. For more information, see the Importing a Subset Dump section.

Changes in Oracle Data Masking and Subsetting Release 13.1

This section describes the features introduced in Oracle Data Masking and Subsetting Release 13.1.

- Support for Non-Dictionary Based Referential Relationships (Application Defined Relationships)
- Subset Data Based on Table Partitions
- Support for Application Data Modeling and Data Masking of Edition View Objects

Support for Non-Dictionary Based Referential Relationships (Application Defined Relationships)

Oracle Data Masking and Subsetting supports non-dictionary based referential relationships. The non-dictionary based referential relationships are application-specific referential relationships that are not defined in the Oracle data dictionary. For more information on using this feature, see the Application Data Modeling chapter.

Subset Data Based on Table Partitions

Oracle Data Masking and Subsetting provides the ability to subset data based on partitioning. Subsetting based on partition and sub-partition enables you to include only those rows in the subset that belong to a particular partition or sub-partition of a table. For more information on using this feature, see the Data Subsetting chapter.

Support for Application Data Modeling and Data Masking of Edition View Objects

Application Data Modeling and Data Masking components have been enhanced to support Edition Views. Support for Edition Views is essential to discover and mask applications such as Oracle E-Business Suite 12.2 or higher, which use edition-based redefinition for patching applications or upgrading databases, with minimal downtime.

You can create an application data model with edition views, and mark the edition view columns as sensitive. Data Masking will mask the base column that the edition view column is referring to.

For more information on Edition Views, see the Oracle Database Advanced Application Developer's Guide.

1

Introduction to Oracle Data Masking and Subsetting

This chapter explains the basics of Oracle Data Masking and Subsetting pack by providing an overview on:

- The Need to Mask and Subset Data
- Challenges Organization Face for Masking and Subsetting Data
- How Oracle Data Masking and Subsetting Addresses Masking/Subsetting Challenge
- Major Components of Oracle Data Masking and Subsetting
- Architecture
- Deployment Options
- Methodology

It is recommended that you understand the above concepts of Oracle Data Masking and Subsetting prior to the implementation. If you are already aware of these concepts or want to start masking and subsetting data, please refer to the outline of steps to mask and subset data mentioned below in this chapter.

> **Note:**
>
> For Oracle Data Masking and Subsetting licensing information, please refer to Oracle Database Licensing Guide.

1.1 The Need to Mask and Subset Data

The reasons to mask and subset data include the following:

- **Limit sensitive data proliferation:** The growing security threats have increased the need to limit exposure of sensitive information. At the same time, copying production data for non-production purposes such as test and development is proliferating sensitive data, expanding the security and compliance boundary, and increasing the likelihood of data breaches.
- **Share what is necessary:** Often, companies have to share a production data set with internal and external parties for various reasons. For example, a Cloud application provider may have to extract and share information specific to their individual subscribers on demand. Another example is a company serving a court order must extract and share a subset of production data with the court. In several cases, it is efficient to extract and share a portion or subset of information instead of sharing the entire production dataset.

- **Comply with data privacy laws and standards:** Data privacy standards such as PCI-DSS and European Union (EU) General Data Protection Regulation (GDPR) emphasize on protecting sensitive information in non-production environments because these environments are typically not as protected or monitored as production systems. EU GDPR also mandates an individual's right to be forgotten, erasure, portability or rectification which requires identifying and processing a subset of information.
- **Minimize storage costs:** Using the entire production data for test, development, and QA purposes will incur additional storage costs and prolong the test and development cycles, increasing the overall storage and operational cost.

Masking and subsetting data addresses the above use cases. Data Masking is the process of replacing sensitive data with fictitious yet realistic looking data. Data Subsetting is the process of downsizing either by discarding or extracting data. Masking limits sensitive data proliferation by anonymizing sensitive production data. Subsetting helps to minimize storage costs by deleting data or extracting a subset of data for sharing or archival. Data Masking is also known as Static Data Masking, and Data Subsetting is also known as Test Data Management.

1.2 Challenges Organization Face for Masking and Subsetting Data

Organizations typically mask and subset data using custom scripts or solutions. While these in-house solutions might work for few columns, they may not cater to large applications with distributed databases and thousands of columns, and result in challenges such as:

- How to locate sensitive data in the midst of numerous applications, databases, and environments?
- How to accurately protect sensitive data as data has different shapes and forms such as VISA, AMEX, Discoverer, Master, Social Security Numbers, and more?
- Is the protected data usable to developers, testers, and applications?
- Will the applications continue to work after masking and subsetting is done?

In addition to the above challenges, organizations may not have the resources to develop and maintain such a solution in this ever-changing IT landscape.

1.3 How Oracle Data Masking and Subsetting Addresses Masking/Subsetting Challenge

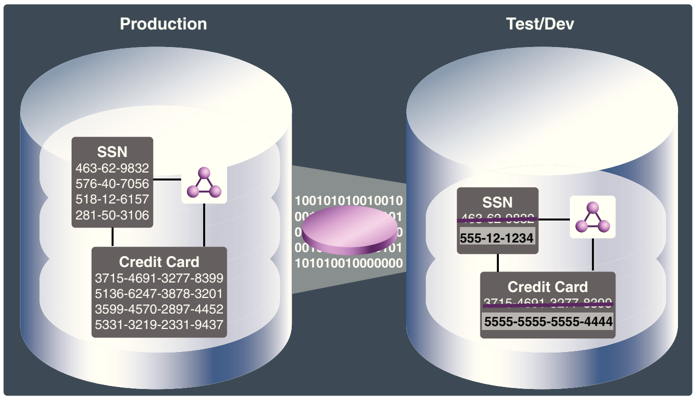
Oracle Data Masking and Subsetting addresses the above challenges by providing an automated, flexible, and easy-to-use solution that masks and subsets sensitive production data, thereby allowing data to be shared safely across non-production environments.

The Oracle Data Masking and Subsetting (DMS) pack of the Oracle Enterprise Manager (EM) helps:

- maximize the business value of data by masking sensitive information
- minimize the compliance boundary by not proliferating the sensitive production information

- lower the storage costs on test and development environments by subsetting data
- automate the discovery of sensitive data and parent-child relationships
- provide a comprehensive library of masking formats, masking transformations, subsetting techniques, and select application templates
- mask and subset data in-Database or on-the-file by extracting the data from a source database
- mask and subset both Oracle and non-Oracle databases
- mask and subset Oracle Databases hosted on the Oracle cloud
- preserve data integrity during masking and subsetting and offers many more unique features
- integrate with select Oracle testing, security, and integration products.

Figure 1-1 Oracle Data Masking and Subsetting Used in a Production and Test Database Setup



- Discover Sensitive Data
- Modeling Application Data
- Mask Data using Format Library
- Subset Based on Goal/Condition
- Mask/Subset in Export or on Staging
- Mask in Workload Captures and Clones
- Pre-installed in Enterprise Manager

1.4 Major Components of Oracle Data Masking and Subsetting

Oracle Data Masking and Subsetting consists of the following major components.

- Application Data Modeling
- Data Masking Format Library
- Data Masking Transformations
- Data Subsetting
- Integrated Masking and Subsetting

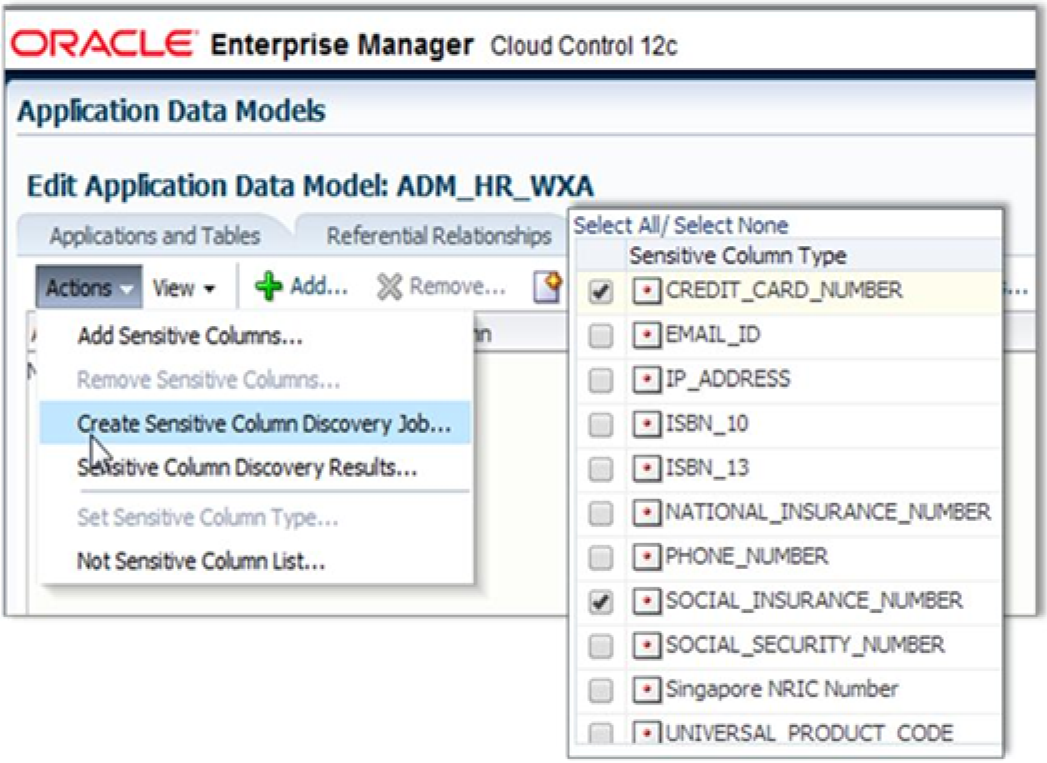
1.4.1 Application Data Modeling

The Application Data Modeling module of the Oracle Data Masking and Subsetting pack simplifies the effort of sensitive data discovery through automated discovery procedures and sensitive column types. These procedures not only discover columns holding sensitive information, but also discover the parent-child relationships between these columns that are defined in the database.

A sensitive column type creates forensics of sensitive data elements, such as national insurance numbers, using a combination of data patterns in column names, column data, and columns comments. Automated discovery procedures leverage sensitive column types to sample data in the database table columns while scanning the database for sensitive information.

Application Data Modeling provides several out-of-the-box sensitive columns types such as credit card numbers, social security numbers, phone numbers. Customer-sensitive column types can be easily created using regular expressions.

Figure 1-2   Editing an Application Data Model


ORACLE Enterprise Manager Cloud Control 12c
Application Data Models
Edit Application Data Model: ADM_HR_WXA
Applications and Tables
Referential Relationships
Actions
View
Add...
Remove...
Add Sensitive Columns...
Remove Sensitive Columns...
Create Sensitive Column Discovery Job...
Sensitive Column Discovery Results...
Set Sensitive Column Type...
Not Sensitive Column List...
Select All/ Select None
Sensitive Column Type
☑ CREDIT_CARD_NUMBER
□ EMAIL_ID
□ IP_ADDRESS
□ ISBN_10
□ ISBN_13
□ NATIONAL_INSURANCE_NUMBER
□ PHONE_NUMBER
☑ SOCIAL_INSURANCE_NUMBER
□ SOCIAL_SECURITY_NUMBER
□ Singapore NRIC Number
□ UNIVERSAL_PRODUCT_CODE

Related Topics:

- Application Data Modeling

1.4.2 Data Masking Format Library

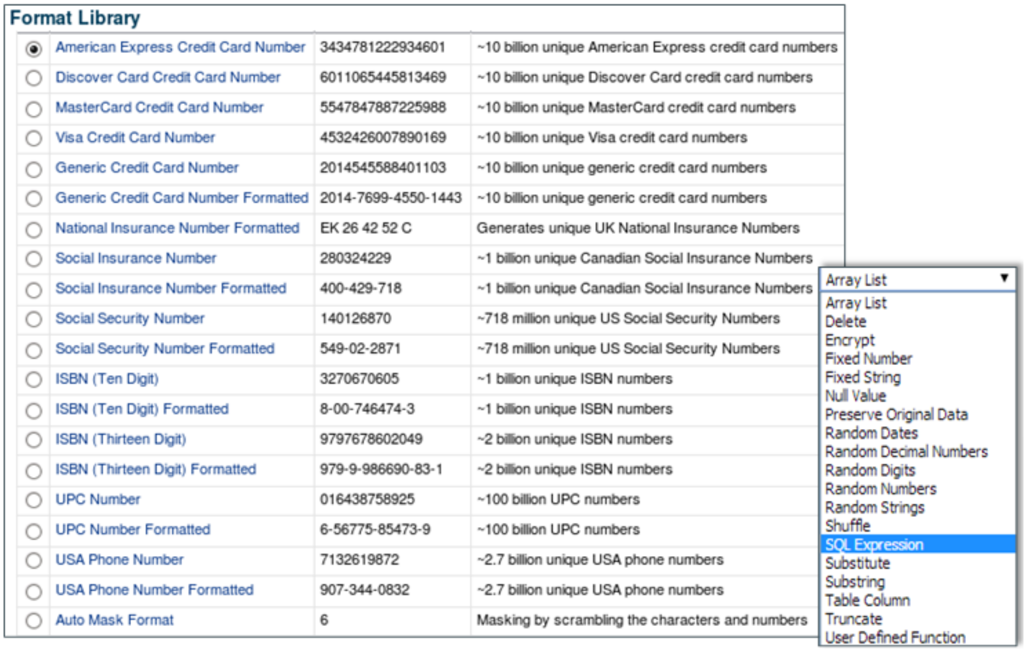
While Application Data Modeling automates the task of sensitive data discovery and modeling, a comprehensive library of masking formats and transformations simplify the effort of defining a masking criteria on the sensitive columns that are discovered.

One of the key aspects of the data masking exercise is to replace the sensitive information with fake data, without breaking the semantics and structure of the data element. The masked data must be realistic and pass format-specific checks such as Luhn check. For example, a masked credit card number must not only be a valid credit card number, but also a valid VISA, Master, American Express or Discover card number. Failing to maintain this data integrity will affect the development or test processes and may break the corresponding application.

Oracle Data Masking and Subsetting is packed with a comprehensive library of masking formats that covers most of the Personally Identifiable Information (PII) and Payment Card Information (PCI). Different types of credit card numbers or national identifiers of different countries or bank account numbers, the masking format library will meet the enterprise needs.

In addition to the several out-of-the-box masking formats, the product provides various built-in tools to easily create custom-masking formats. There are simple tools that generate fixed or random numbers, strings, and dates. There are tools that facilitate substitution from lookup tables. There are tools such as SQL Expression and User Defined Function to accommodate complex user-defined masking logic.

Figure 1-3   Data Masking Format Library



Format Library

●	American Express Credit Card Number	3434781222934601	~10 billion unique American Express credit card numbers
○	Discover Card Credit Card Number	6011065445813469	~10 billion unique Discover Card credit card numbers
○	MasterCard Credit Card Number	5547847887225988	~10 billion unique MasterCard credit card numbers
○	Visa Credit Card Number	4532426007890169	~10 billion unique Visa credit card numbers
○	Generic Credit Card Number	2014545588401103	~10 billion unique generic credit card numbers
○	Generic Credit Card Number Formatted	2014-7699-4550-1443	~10 billion unique generic credit card numbers
○	National Insurance Number Formatted	EK 26 42 52 C	Generates unique UK National Insurance Numbers
○	Social Insurance Number	280324229	~1 billion unique Canadian Social Insurance Numbers
○	Social Insurance Number Formatted	400-429-718	~1 billion unique Canadian Social Insurance Numbers
○	Social Security Number	140126870	~718 million unique US Social Security Numbers
○	Social Security Number Formatted	549-02-2871	~718 million unique US Social Security Numbers
○	ISBN (Ten Digit)	3270670605	~1 billion unique ISBN numbers
○	ISBN (Ten Digit) Formatted	8-00-746474-3	~1 billion unique ISBN numbers
○	ISBN (Thirteen Digit)	9797678602049	~2 billion unique ISBN numbers
○	ISBN (Thirteen Digit) Formatted	979-9-986690-83-1	~2 billion unique ISBN numbers
○	UPC Number	016438758925	~100 billion UPC numbers
○	UPC Number Formatted	6-56775-85473-9	~100 billion UPC numbers
○	USA Phone Number	7132619872	~2.7 billion unique USA phone numbers
○	USA Phone Number Formatted	907-344-0832	~2.7 billion unique USA phone numbers
○	Auto Mask Format	6	Masking by scrambling the characters and numbers

1.4.3 Data Masking Transformations

Data Masking format library and application templates accelerate the task of defining masking rules and preserving the integrity and structure of data elements.

Depending on the business use cases, organizations may have different requirements while mapping masking formats to sensitive columns. For example, one of the requirements in a large distributed database environment is to generate consistent masked outputs for the given input across multiple databases. Oracle Data Masking and Subsetting provides sophisticated masking transformations that fit into broader business context. If masking formats are considered as building blocks of a data masking definition, then masking transformations align these masking formats according to the different business requirements.

Conditional Masking

Conditional transformation provides an ability to arrange masking formats according to different conditions. For example, consider masking a column containing unique person identifiers. Identifiers that belong to country USA can be masked using Social Security Number format and that belong to country UK can be masked using National Insurance Number format.

Compound Masking

Compound transformation (also known as grouping option), masks related columns as a group, ensuring the masked data across the related columns retain the same relationship. For example, consider masking address fields such as city, state, and postal codes. These values must be consistent after masking.

Deterministic/Consistent Masking

Deterministic transformation generates consistent outputs for a given input across databases. This transformation will be helpful to maintain data integrity across multiples applications and preserve system integrity in a single sign-on environment. For example, consider three applications: a human capital management application, a customer relationship management application, and a sales data warehouse. These three applications may have key common fields such as EMPLOYEE ID that must be masked consistently across these applications. Substitute and Encrypt masking formats provide deterministic masking transformation.

Shuffle

Shuffle transformation shuffles fields within a column in a random fashion. This transformation is helpful in breaking one-to-one mapping between sensitive data elements. For example, columns containing personal health records can be shuffled while masking health care information.

Key Based Reversible Masking (Encrypt Format)

This transformation encrypts and decrypts the original data using a secure key string. The input data format is preserved during encryption and decryption. This transformation uses powerful industry-standard 3DES algorithm. This transformation is helpful when businesses need to mask and send their data to a third-party for analysis, reporting, or any other business processing purpose. After the processed data is received from the third-party, the original data can be recovered using the same key string that was used to encrypt the data.

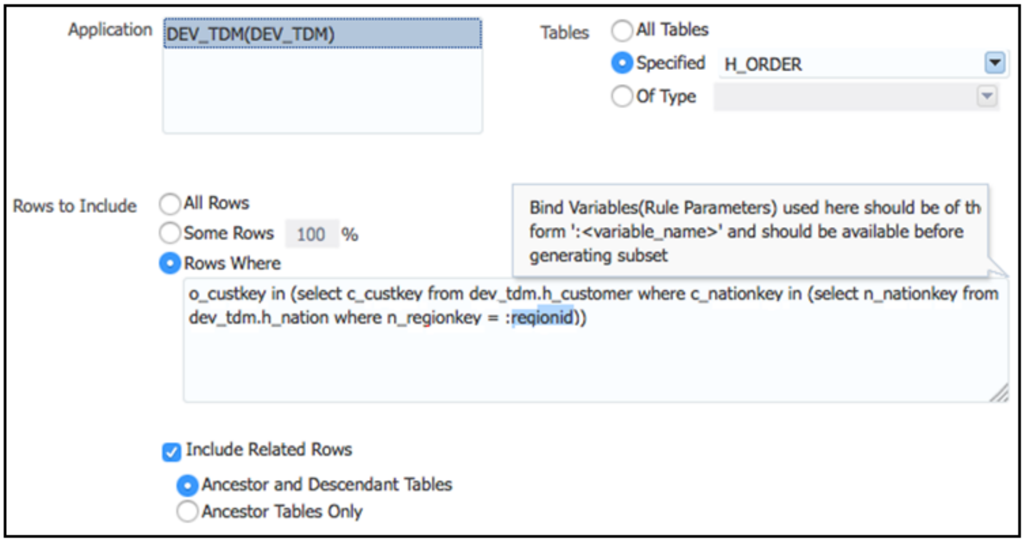
Format Preserving Randomization (Auto Mask Format)

This transformation randomizes the data, preserving the input length, position of the characters and numbers, case of the character (upper or lower), and special characters in the input.

1.4.4 Data Subsetting

Subsetting modern enterprise class applications is a challenging task. Oracle Data Masking and Subsetting simplifies this effort through its easy-to-define goal and condition-based subsetting techniques. Data can be subsetted based on different goals. A goal can be relative table size, for example, extracting 1% subset of a table containing 1 million rows. Data can also be subsetted based on different conditions. A condition can be based on time, for example, discarding all user records created prior to a particular year. A condition can be based on region, for example, extracting Asia Pacific information for a new application development. The conditions are specified using “SQL where clause”. The “SQL where clause” also supports bind variables.

Figure 1-4 Subsetting based on a condition



Application: DEV_TDM(DEV_TDM)

Tables: ○ All Tables ● Specified H_ORDER ○ Of Type

Rows to Include: ○ All Rows ○ Some Rows 100 % ● Rows Where

Bind Variables(Rule Parameters) used here should be of th form ':<variable_name>' and should be available before generating subset

```
o_custkey in (select c_custkey from dev_tdm.h_customer where c_nationkey in (select n_nationkey from
dev_tdm.h_nation where n_regionkey = :regionid))
```

☑ Include Related Rows

● Ancestor and Descendant Tables
○ Ancestor Tables Only

Data Subsetting generates a real-time dynamic view of the application schema with before and after storage size and the percentage of data within the tables being subsetted, including the dependent tables. Administrators can use this view to validate the subsetting criteria even before subsetting the data.

1.4.5 Integrated Masking and Subsetting

Oracle Data Masking and Subsetting provides quick masking and detailed masking options in-line to subsetting data. Quick masking can be used in cases where the primary use case is data subsetting with a requirement to mask few sensitive columns. Quick masking provides minimal masking formats such as fixed number or fixed string to replace all the sensitive fields to a common masked value. If the requirement is to mask several sensitive columns with different formats and preserving the integrity of the masked data, use the data masking component to create detailed masking definitions. These pre-created detailed masking definitions can be called in-line with the data subsetting definitions. During execution, data is subsetted followed by data being masked.

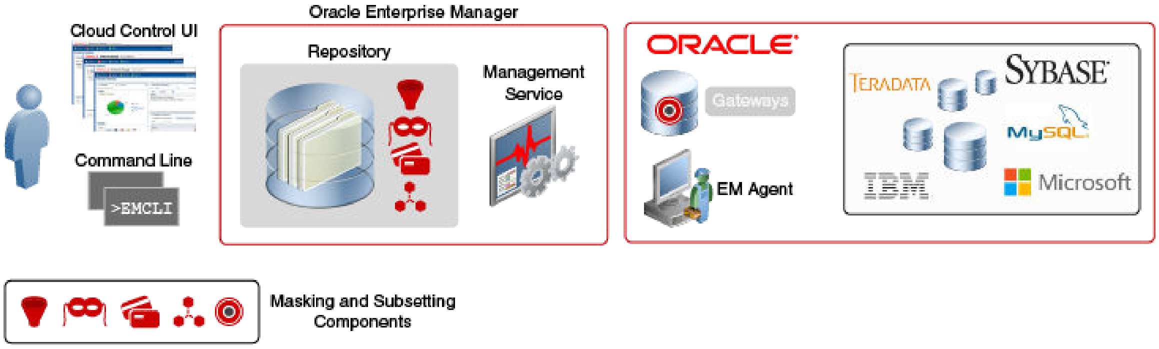
Data Subsetting integrated with Data Masking maximizes the business value of the production data without compromising sensitive information.

1.5 Architecture

Oracle Data Masking and Subsetting is part of the Oracle Enterprise Manager infrastructure. Organizations using Oracle Enterprise Manager do not need to download and install the Oracle Data Masking and Subsetting Pack separately. Oracle Enterprise Manager provides unified browser-based user interface for administration. All Data Masking and Subsetting objects are centrally located in the Oracle Enterprise

Manager repository, which facilitates centralized creation and administration of Application Data Models, Data Masking and Subsetting rules or definitions. In addition to its intuitive cloud control Graphical User Interface (GUI), Oracle Enterprise Manager also provides Command Line Interface (EMCLI) to automate select Data Masking and Subsetting tasks.

Figure 1-5 Architecture of the Oracle Data Masking and Subsetting



For more details on Oracle Enterprise Manager Architecture, please refer to the *Oracle Enterprise Manager Introduction Guide*.

1.6 Deployment Options

Oracle Data Masking and Subsetting provides the following modes for masking and subsetting data:

- **In-Database mode** directly masks and subsets data within a non-production database with minimal or zero impact on production environments. As In-Database mode permanently changes the data in a database, it is recommended for non-production environments such as staging, test or development databases instead of production databases.
- **In-Export mode** masks and subsets the data in near real-time while extracting the data from a database. The masked and subsetted data that is extracted is written to data pump export files, which can be further imported into test, development or QA databases. In general, In-Export mode is used for production databases. In-Export method of masking and subsetting is a unique offering from Oracle that sanitizes sensitive information within the product perimeter.
- **Heterogeneous mode** Oracle Data Masking and Subsetting can mask and subset data in non-Oracle databases. Target production data is first copied from the non-Oracle environment into Oracle Database using an Oracle Database Gateway, and is then masked and subsetted within the Oracle Database, and is finally copied back to the non-Oracle environment. This approach is very similar to the steps used in various ETL (Extract, Transform, and Load) tools, except that the Oracle Database is the intermediary that transforms the data. Oracle Database Gateways enable Oracle Data Masking and Subsetting to operate on data from Oracle MySQL, Microsoft SQLServer, Sybase SQLServer, IBM DB2 (UDB, 400, z/OS), IBM Informix, and Teradata.

Note:

In order to mask and subset non-Oracle databases, you must have a license for Oracle Data Masking and Subsetting for non-Oracle Databases. For more information on licensing, please refer to *Oracle Database Licensing Guide* .

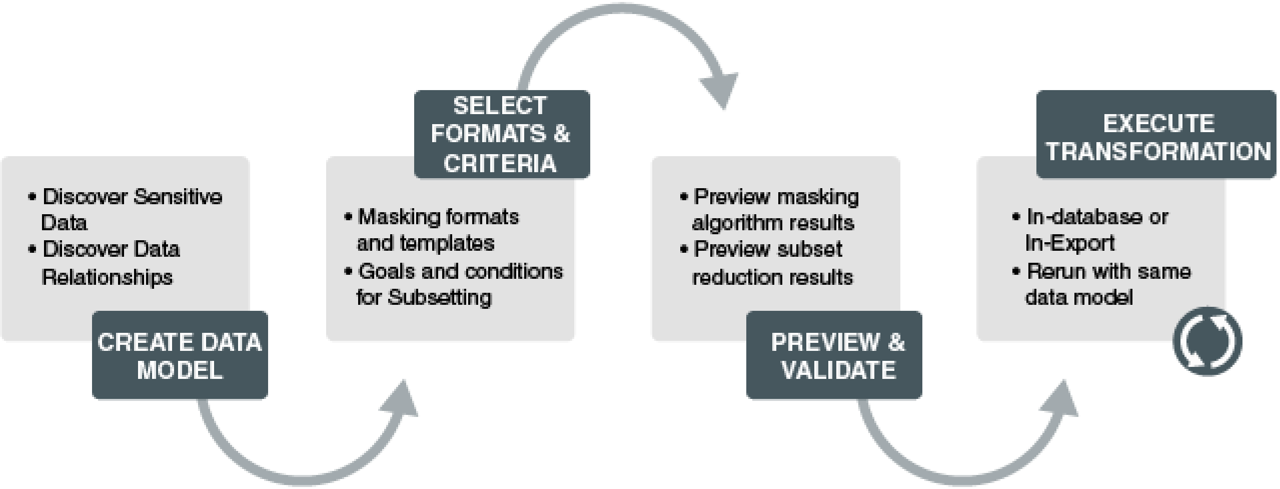
1.7 Methodology

Oracle Data Masking and Subsetting uses the following methodology to secure non-production database and replace sensitive data with fictitious, but relevant data that meets compliance requirements.

- **Creating an Application Data Model**— Discover sensitive data and data relationships, and then create or assign an Application Data Model
- **Selecting Masking Formats and Criteria**— Create data masking definition and masking format types and templates based on the sensitive data that is discovered
- **Previewing and Validating** — Secure sensitive data by previewing the masking algorithm results and the subset reduction results
- **Executing Masking Transformations**— Execute In-Database or In-Export masking and subsetting transformations and validate the data that is masked

The following figure describes the methodology used in Oracle Data Masking and Subsetting.

Figure 1-6 Methodology


• Discover Sensitive Data
• Discover Data Relationships
CREATE DATA MODEL
SELECT FORMATS & CRITERIA
• Masking formats and templates
• Goals and conditions for Subsetting
• Preview masking algorithm results
• Preview subset reduction results
PREVIEW & VALIDATE
EXECUTE TRANSFORMATION
• In-database or In-Export
• Rerun with same data model

1.8 Outline of Steps to Mask and Subset Data

This section outlines the steps to mask and subset data in the context of the rest of this document.

1. Start with the Before You Begin chapter to understand the privileges, roles, and storage requirements.

2. Refer to the Application Data Modeling chapter to discover and model sensitive columns.
3. Refer to the Data Masking chapter to define and execute masking criteria .
4. Refer to the Data Subsetting chapter to define and execute subsetting criteria.

Related Topics:

- Before You Begin
- Application Data Modeling
- Data Masking
- Data Subsetting

2

Before You Begin

This chapter helps you prepare with the prerequisites and other important things that you must consider before installing and using Oracle Data Masking and Subsetting.

2.1 Privileges and Roles

This section explains the privileges and roles required for using Oracle Data Masking and Subsetting.

- Privileges Required for Creating an Application Data Model
- Privileges Required for Creating a Masking Definition
- Privileges Required for Creating a Subsetting Definition

Privileges Required for Creating an Application Data Model

The following basic privileges (without *Oracle Database Vault*) are required for creating an application data model:

- `CREATE ANY PROCEDURE`
- `SELECT ANY DICTIONARY`
- `EXECUTE ANY PROCEDURE`
- `CREATE SESSION`

The following additional privileges/authorizations are required on *Oracle Database Vault* environment:

- `Participant` authorization on the Oracle Enterprise Manager realm
- Execute `dbms_macadm.authorize_ddl ('<username>', 'dbsnmp')`

> **Note:**
>
> This is required for *Oracle Enterprise Manager* versions 12.1.0.2 and greater only.

- `Owner` authorization on the Oracle Enterprise Manager realm for the user that creates the application data model.

Privileges Required for Creating a Masking Definition

The following basic privileges (without *Oracle Database Vault*) are required for creating a masking definition:

- `SELECT ANY DICTIONARY`
- `CREATE ANY DICTIONARY`
- `ALTER ANY INDEX`

- `ALTER ANY TABLE`
- `ALTER SYSTEM`
- `ANALYZE ANY`
- `CREATE ANY INDEX`
- `CREATE ANY TABLE`
- `CREATE PROCEDURE`
- `CREATE SEQUENCE`
- `CREATE SESSION`
- `CREATE TABLE`
- `CREATE TABLESPACE`
- `CREATE TYPE`
- `DROP ANY INDEX`
- `DROP ANY TABLE`
- `DROP TABLESPACE`
- `INSERT ANY TABLE`
- `LOCK ANY TABLE`
- `RESUMABLE`
- `SELECT ANY TABLE`
- `UNLIMITED TABLESPACE`

The following extra privileges are required:

- `EXECUTE ON DBMS_CRYPTO`
- `EXECUTE ON DBMS_AQADM`

The following additional privileges/authorizations are required on *Oracle Database Vault* environment:

`Execute on utl_file`

The following are the additional privileges are required for group shuffle:

- `Owner` authorization on “Oracle Enterprise Manager” realm
- Execute `dbms_macadm.authorize_ddl ('<username>', 'dbsnmp')`

> **Note:**
>
> This is required for *Oracle Enterprise Manager* versions 12.1.0.2 and greater only.

- `Participant` authorization on realm for masking table
- `Owner` authorization on realm for masking tables

Privileges Required for Creating a Subsetting Definition

The following basic privileges (without *Oracle Database Vault*) are required for creating a subsetting definition:

- `CREATE TABLE`
- `RESUMABLE`
- `ALTER ANY TABLE`
- `UNLIMITED TABLESPACE`
- `ANALYZE ANY`
- `CREATE TYPE`
- `DROP ANY INDEX`
- `ALTER ANY INDEX`
- `INSERT ANY TABLE`
- `DROP ANY TABLE`
- `CREATE ANY DIRECTORY`
- `CREATE ANY TABLE`
- `ALTER SYSTEM`
- `LOCK ANY TABLE`
- `CREATE SESSION`
- `SELECT ANY TABLE`
- `CREATE ANY INDEX`
- `SELECT ANY DICTIONARY`
- `CREATE SEQUENCE`
- `CREATE PROCEDURE`
- `EXECUTE ON DBMS_AQADM`

2.2 Oracle Data Masking and Subsetting Access Rights

Prerequisites

The following privileges must be assigned to the users on Oracle Enterprise Manager repositories to administer and view the Oracle Data Masking and Subsetting user interface pages.

- **DB_MASK_ADMIN**: to manage and use data masking feature in Oracle Enterprise Manager.
- **DB_ADM_ADMIN**: to manage and use the application data model feature in Oracle Enterprise Manager.

  DB_SUBSET_ADMIN: to manage and use the data subsetting feature in Oracle Enterprise Manager

By default, Enterprise Manager administrators can access the following primary Oracle Data Management and Subsetting pages:

- Application Data Models
- Data Subset Definitions
- Data Masking Definitions
- Data Masking Formats

This is by virtue of having the TDM_ACCESS privilege, which is included in the PUBLIC role. The Super Administrator can revoke this privilege for designated administrators, thereby restricting access to the TDM pages. Without the privilege, the respective menu items do not appear in the Cloud Control console.

Additionally, Enterprise Manager provides a privilege access model that enables Super Administrators and administrators to limit access to TDM objects to authorized users only. The model involves the ability to grant Operator or Designer privileges to selected users.

Operator Privileges

Those granted Operator privileges can perform data masking and subsetting operations. Privileges can be granted on TDM objects; that is, on Application Data Models (ADM), data subsetting definitions, and data masking definitions. Operator privileges do not include the ability to edit and delete these objects.

- ADM–a user (other than Super Administrator) with ADM Operator privileges can view an ADM, but cannot edit and delete it, nor view its properties. To enforce this, the Edit and Delete icons, and the Properties menu are disabled. Additionally, the Sync option on the Create Verification Job page is disabled.
- Data subset definition–a user (other than Super DSD Administrator) with Operator privileges can view but not edit and delete a subset definition. To enforce this, the Edit and Delete icons are disabled.

  A user with Data Subset Definition Operator privileges can do any other operation except for editing and deleting the data subset definition, and has the following rights:

  - Viewing the data subset definition.
  - Creating a data subset to export files.
  - Creating a data subset on a database.
  - Saving the subset script.
- Data masking definition–a user with Data Masking Definition Operator privileges can do any other operation except for editing and deleting the data masking definition, and has the following rights:
  - Viewing the data masking definition.
  - Generating a data masking script.
  - Scheduling a data masking job.
  - Exporting a data masking definition.

Designer Privileges

Those granted Designer privileges can enhance, modify, and manage TDM objects. These users can also grant and revoke Operator and Designer privileges to others. Designer privileges imply the corresponding Operator privileges on a TDM object.

- ADM–a user with Designer privileges can perform all operations on an ADM including delete.
- Data subset definition–a user with Designer privileges can perform all operations on a subset definition including delete.
- Data masking definition–a user with Designer privileges can perform all operations on a masking definition including delete.

2.3 Access Control For Oracle Data Masking and Subsetting Objects

This section describes the procedure to grant privileges on Application Data Models, Data Masking definitions, and Data Subsetting definitions.

- Assigning Privileges to an Existing ADM
- Granting Privileges on a Subset Definition

2.4 Storage Requirements

Although Oracle Data Masking and Subsetting objects such as data models, masking and subsetting definitions consume a negligible amount of storage space, depending on the amount of data being stored over a period of time, you may need to allocate additional storage space to Oracle Enterprise Manager's repository database.

This section details the storage recommendations for masking and subsetting.

- **In-Database Masking:** 3X of additional space in the user tablespace (X being the largest table in size) 2X of additional space in temporary tablespace
- **In-Export Masking:** 2X additional space in the user tablespace (X being the largest table in size) 2X of additional space in temporary tablespace Sufficient disk space to store the generated export dump file
- **In-Database Subsetting:** 2X additional space in the user tablespace (X being the largest table in size) 2X additional space in temporary tablespace
- **In-Export Subsetting:** X additional space in the user tablespace (X being the largest table in size) Sufficient space to store the generated dump files

> **Note:**
>
> The recommended storage requirement for integrated masking and subsetting is the sum total of the storage requirement for masking and subsetting as mentioned above.

2.5 Supported Data Types

The list of supported data types varies by release.

- Grid Control, Database, and Cloud Control
  - Numeric Types

    The following Numeric Types can use Array List, Delete, Fixed Number, Null Value, Post Processing Function, Preserve Original Data, Random Decimal Numbers, Random Numbers, Shuffle, SQL Expression, Substitute, Table Column, Truncate, Encrypt, and User Defined Function masking formats:

    - `NUMBER`
    - `FLOAT`
    - `RAW`
    - `BINARY_FLOAT`
    - `BINARY_DOUBLE`

  - String Types

    The following String Types can use Array List, Delete, Fixed Number, Fixed String, Null Value, Post Processing Function, Preserve Original Data, Random Decimal Numbers, Random Digits, Random Numbers, Random Strings, Shuffle, SQL Expression, Substitute, Substring, Table Column, Truncate, Encrypt, and User Defined Function masking formats:

    - `CHAR`
    - `NCHAR`
    - `VARCHAR2`
    - `NVARCHAR2`

  - Date Types

    The following Date Types can use Array List, Delete, Null Value, Post Processing Function, Preserve Original Data, Random Dates, Shuffle, SQL Expression, Substitute, Table Column, Truncate, Encrypt, and User Defined Function masking formats:

    - `DATE`
    - `TIMESTAMP`

- Grid Control and Cloud Control
  - Large Object (LOB) Data Types

    The following Data Types can use Fixed Number, Fixed String, Null Value, Regular Expression, and SQL Expression masking formats:

    - BLOB
    - CLOB
    - NCLOB

2.6 Unsupported Objects

Oracle Data Masking and Subsetting does not support:

- external tables
- clustered tables
- long columns
- column of type “XML”; XML-type columns

- virtual columns

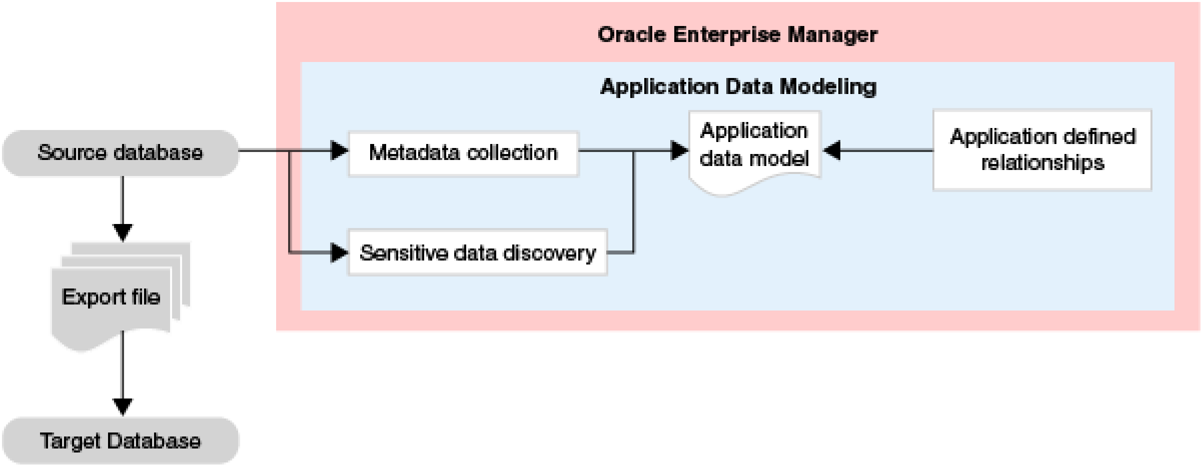
> **Note:**
>
> Masking is supported for relational tables and tables containing long columns.

3 Application Data Modeling

Secure Test Data Management provides Enterprise Manager the capability to enable operations such as sensitive data discovery, data subsetting, and data masking. These capabilities enable scanning and tagging of sensitive data and modeling of data relationships incorporated within an Application Data Model (ADM). You must have the Oracle Data Masking and Subsetting Pack license to use test data management features.

The following figure shows the workflow associated with an Application Data Model.

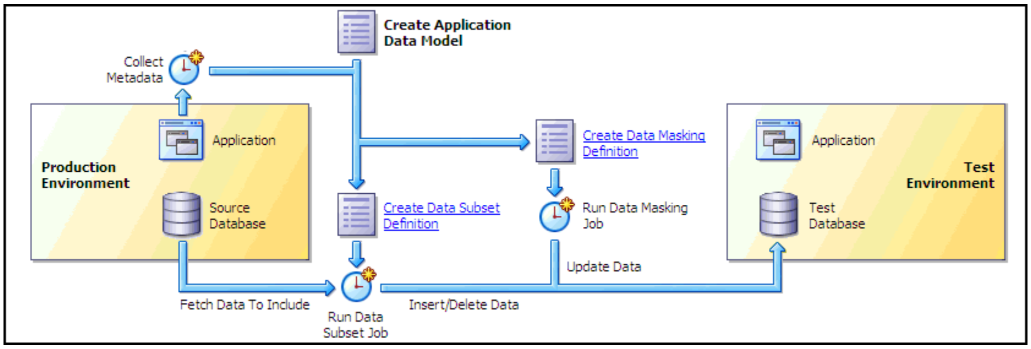
Figure 3-1 Workflow of an Application Data Model



The ADM stores the list of applications, tables, and relationships between table columns that are either declared in the data dictionary, imported from application metadata, or user-specified. The ADM maintains sensitive data types and their associated columns, and is used by test data operations, such as data subsetting and data masking, to securely produce test data. Creating an ADM is a prerequisite for data subsetting and data masking operations.

The following figure shows the Application Data Model's relationship to other test data management components as well as the production and test environments.

Figure 3-2 Test Data Management Architecture



You can perform several tasks related to Application Data Modeling, including the following tasks discussed in this chapter:

- Creating an Application Data Model
- Creating and Managing Custom Sensitive Column Types
- Associating a Database to an Existing ADM
- Verifying or Synchronizing an ADM
- Importing and Exporting an ADM
- Assigning Privileges to an Existing ADM

> **Note:**
>
> The procedures described in this chapter are applicable to Oracle Enterprise Manager Cloud Control 12.1 and higher only.

> **See Also:**
>
> - Data Subsetting, for information about data subsetting
> - Masking Sensitive Data, for information about data masking

3.1 Creating an Application Data Model

Before proceeding, ensure that you have the following privileges:

- **Target Privileges (applicable to all targets):**
  - Connect to any viewable target
  - Execute Command Anywhere
  - View Any Target
- **Resource Privileges:**

  - Job System
  - Named Credential
  - Oracle Data Masking and Subsetting resource privilege

> **Note:**
>
> The EM_ALL_OPERATOR privilege for Enterprise Manager Cloud Control users includes all of the above privileges.

- SELECT_CATALOG_ROLE for database users
- SELECT ANY DICTIONARY privilege for database users
- EXECUTE privileges for the DBMS_CRYPTO package

> **Note:**
>
> When you create an Application Data Model, the PL/SQL metadata collection packages are automatically deployed on the target database. The Database user must have DBA privileges to auto-deploy the packages.
>
> We recommend that you create an Application Data Model for the first time with a highly privileged user so that all the necessary packages are deployed. Subsequently, all other Application Data Models can be created with a less privileged user.

3.1.1 Creating an ADM

To create an Application Data Model:

1. From the Application Data Modeling page, view the diagram that explains how you can create a database for a test environment.
2. Click **Create**.

   A pop-up window requesting general properties information appears.
3. Specify a name for the ADM to be created.
4. Select the Source Database by clicking the Select Database Target icon.

   The Source Database is the source from which the metadata is extracted.
5. Select an Application Suite:
   - If you select **Custom Application Suite**:
     - By default, metadata collection is enabled for creating the ADM.
     - If "Create One Application For Each Schema" is not selected, a shell ADM is created. You must later edit the ADM to add applications and tables to it. Also, please note that metadata collection job is not submitted, like how it is done for the default choice.
   - If you select **Oracle Application Suite**:

  - **Oracle E-Business Suite**– Provide the database credentials for APPS user (or equivalent) , and click **Submit** to create the ADM.
  - **Oracle Fusion Applications**– Provide database credentials for FUSION user (or equivalent), and click **Submit** to create the ADM.

Note the following points about metadata collections:

- The metadata collection for the selected application suite populates the ADM with the applications and tables in the suite.
- The ADM can collect metadata for one or more schemas. An ADM application typically represents a schema. Each schema you select becomes an ADM application, and the ADM becomes populated with the tables in the schema, particularly in the case of custom applications. However, please note that multiple applications can also map to a single schema, as in the case of Fusion Applications. The actual mapping depends on the application metadata discovered by the metadata collection job.

6. Select the following:
   - **Discover Dictionary Based Relationships**— identifies dictionary based referential relationships. These are referential relationships that are defined in the Oracle data dictionary.
   - **Discover Non-Dictionary Based Relationships**
     - **By Sampling Column Names** — identifies the potential parent-child relationships that are not defined as referential integrity constraints (primary key and foreign key) in the data dictionary by matching the column name. Column name patterns can be specified using a regular expression.
     - **By Sampling Column Values** — identifies the potential parent-child relationships that are not defined as referential integrity constraints (primary key and foreign key) in the data dictionary by matching the values or value patterns of a column.

       Users can choose from the available value patterns or specify a custom data pattern that uses a regular expression. Oracle Data Masking and Subsetting will match the values in the column to identify the potential primary key and foreign key using the data pattern specified by the user, and return the results.

       The potential foreign keys are verified for containment within the potential primary key column, that is, the values of potential foreign key must already be present in the potential primary key. A containment test is done to meet 90% accuracy, that is, if the foreign key column is 90% contained in the primary key, a match is flagged. This test is done to include any orphan rows that might be present in applications such as Oracle's E-Business Suite and Oracle Fusion Applications.

> **Note:**
>
> For accurate results, ensure that you use `dbms_stats.gather_table_stats` to gather the stats of all the tables.

7. Click **Continue**.

If you selected Custom Application Suite in Step 5, a Schemas pop-up is displayed. From the Available list, select the schemas you want to include as applications in the ADM being created.

8. Click **Continue**.
9. If you selected **Discover Non-Dictionary Based Relationships** in Step 6, click the **+** icon, and specify the primary key and foreign key columns that must be matched.
10. Click **Continue**.
11. Specify the parameters for scheduling the metadata collection job.

    You can chose to either run the metadata collection job immediately or schedule it to start at a later time.
12. Click **Submit** to submit the metadata collection job.

    The ADM you created appears in the Application Data Modeling page.
13. Click **View Job Details** to view the status and details of the metadata collection process.
14. Review the *Most Recent Job Status* table column to monitor the status of the metadata collection job.

    The application data model is locked and cannot be edited when the metadata is being collected. You must wait until the status indicates that the job is complete.

3.1.2 Editing an ADM to View the Application Schemas and Tables

To view and edit application tables:

1. From the Application Data Modeling page, view the diagram that explains how you can create a database for a test environment.
2. Select the Application Data Model you previously created, and then click **Edit**.

    The Applications and Objects subpage appears, displaying the applications discovered during the metadata collection process.
    To view the tables associated with an application, click the Expand **(>)** icon.
3. To edit an application, select the application, open the **Actions** menu, then select **Add Application**.

    The Add Application pop-up window appears.
4. Specify a name for the application, a nick name/short name, description for the application, and click the search icon to search a schema.
5. Select a schema from the list, and click **OK**.
6. Click **OK** to add the application table to the data model.

    The table now appears in the Applications and Tables view.
7. To discover non-dictionary based referential relationships, click the **Referential Relationships** tab, and then click **Discover Non-Dictionary Based Relationships**.

3.1.3 Adding and Removing Tables From the Application Schema

To add or remove tables from the application schema:

1. From the Application Data Modeling page, select the Application Data Model you previously created, and then click **Edit**.

   The Applications and Objects subpage appears, displaying the applications and objects found during metadata collection.
   To see the tables for an application, click the expand **(>)** icon.

2. To add a table, click **Add Application**.

   The Add Application pop-up window appears.

3. Specify a name for the application, a nick name/short name, description for the application, and click the search icon to search a schema.

   The Search and Select pop-up appears, displaying all of the tables from the selected schema that are not assigned to an application.

4. Select an unassigned table, then click **OK**.

   The table name now appears in the Add Application pop-up.

5. After selecting a Table Type, click **OK**.

   The table now appears in the Applications and Objects view.

6. To remove a table, select the table from the Application and Objects view, and click **Remove**.

3.1.4 Viewing the Referential Relationships

To view referential relationships:

1. From the Application Data Modeling page, select the model you created, then click **Edit**.

   The Applications and Objects subpage appears, displaying the applications found during metadata collection.
   To view the tables for an application, click the expand **(>)** icon.

2. Click the **Referential Relationships** tab.

   The following types of referential relationships are supported:

   - **Dictionary-defined**

     This is the referential relationship that the metadata collection extracted, resulting from primary key and foreign key relationship. You can remove relationship from the ADM if desired.

   - **Non-Dictionary Based**

     This is the referential relationship that is not defined in the Oracle data dictionary, and is achieved by matching the column names and column values of potential foreign keys with column names and column values of potential primary keys along with their data types. You must evaluate the potential non-dictionary based referential relationships listed here, and if you consider these relationships valid, select the relationship, and click **Add to ADM** to add it to the Application Data Model.

   - **Imported from template**

     If there are application templates available from the vendor of the enterprise application, for example, Oracle Fusion Applications or Oracle E-Business

Suite, then the ADM can be created from the application vendor-supplied template by using the **Import** action on the ADM home page.

- **User-defined**

3. Open an application view by selecting it, then use the chevron icon (>) to view the parent and dependent key relationships, or select **Expand All** from the **View** menu to view all relationships.

3.1.5 Adding and Removing Referential Relationships

To manually add a referential relationship:

1. From the Application Data Modeling page, select the model you created, then click **Edit**.

   The Applications and Objects subpage appears, displaying the applications and objects found during metadata collection.
   To view the tables and objects for an application, click the expand **(>)** icon.

2. From the Referential Relationships tab, open the **Actions** menu, then select **Add Referential Relationship**.

   The Add Referential Relationship pop-up window appears.

3. Select the requisite Parent Key and Dependent Key information.

4. In the Columns Name list, select a dependent key column to associate with a parent key column.

5. Click **OK** to add the referential relationship to the ADM.

   The new dependent column now appears in the referential relationships list.

3.1.6 Performing Sensitive Data Discovery

To discover sensitive columns:

1. From the Application Data Modeling page, select the model you created, then click the **Edit** .

   The Applications and Objects subpage appears, displaying the applications and objects found during metadata collection. To view the tables for an application, click the expand **(>)** icon.

2. From the Sensitive Columns tab, open the **Actions** menu, then select **Create Sensitive Column Discovery Job**.

   The Parameters pop-up appears.

3. Select the applications and sensitive column types.

   The sensitive column types you select is processed for each application to search for columns that match the type.

4. Click **Continue**.

   The schedule pop-up window appears.

5. Specify the required information, schedule the job, then click **Submit** when you have finished.

   The Sensitive Columns subpage reappears.

6. Click **Save and Return** to return to the Application Data Modeling home page.

Modifying the Sensitive Column Type

To modify the sensitive column type:

1. From the Application Data Modeling page, select the model you created, then click the **Edit**.

   The Applications and Objects subpage appears, displaying the applications and objects found during metadata collection. To view the tables for an application, click the expand **(>)** icon.

2. Click the **Sensitive Columns** tab.

   This view shows the sensitive columns that have already been identified.

3. Select the sensitive column for which you want to change the type.
4. Open the **Actions** menu, then select **Set Sensitive Column Type**.

   The Set Sensitive Column Type pop-up window appears.

5. Select the new type and click **OK**.

3.1.6.1 Viewing the Discovery Results

To view the sensitive column discovery results:

1. From the Application Data Modeling page, select the model you created, then click **Edit**.

   The Applications and Objects subpage appears, displaying the applications found during metadata collection.
   To view the tables for an application, click the expand **(>)** icon.

2. Select the **Sensitive Columns** tab, then click **Discovery Results** to view the discovery results.

3.1.6.2 Setting Sensitive Status on the Discovery Results

To set sensitive status on the discovery results:

1. When the Most Recent Job Status column indicates that the job is Successful, select the ADM, then click **Edit**.
2. Select the **Sensitive Columns** tab, then click **Discovery Results** to view the job results.
3. To set the sensitive status of any column, select the row for the column you want to define, open the **Set Status** menu, then select either **Sensitive** or **Not Sensitive**.
4. Click **OK** to save and return to the Sensitive Columns tab.

   The sensitive columns you defined in the previous step now appear in the list.

5. Click **Save and Return** to return to the Application Data Modeling page.

3.1.6.3 Adding and Removing Sensitive Columns

To add/remove sensitive columns:

1. From the Application Data Models page, select an ADM, then click **Edit**.
2. Select the **Sensitive Columns** tab, then click **Add**.

   The Add Sensitive Column pop-up appears.
3. Provide the required information and an optional Sensitive Column Type, then click **OK**.

   The sensitive column now appears in the table for the Sensitive Columns tab.

3.2 Creating and Managing Custom Sensitive Column Types

After you have successfully created an ADM, the next task is to create either a new sensitive column type or one based on an existing type.

To create a sensitive column type:

1. From the **Actions** menu of the Application Data Models page, select **Sensitive Column Types**.

   The Sensitive Column Types page appears.
2. Click **Create**.

   The Create Sensitive Column Type pop-up appears.
3. Specify a required name and regular expressions for the Column Name, Column Comment, and Column Data search patterns.

   - The Or Search Type means that any of the patterns can match for a candidate sensitive column.
   - The And Search Type means that all of the patterns must match for a candidate sensitive column.

   If you do not provide expressions for any of these parameters, the system does not search for the entity.
4. Click **OK**.

   The sensitive column appears in the table in the Sensitive Column Types page.

To create a sensitive column type based on an existing type:

1. From the **Actions** menu of the Application Data Models page, select **Sensitive Column Types**.

   The Sensitive Column Types page appears.
2. Select either a sensitive column type you have already defined, or select one from the out-of-box types that the product provides.
3. Click **Create Like**.

   The Create Sensitive Column Type pop-up appears.
4. Specify a required name and alter the existing expressions for the Column Name, Column Comment, and Column Data search patterns to suit your needs.
5. Click **OK**.

   The sensitive column appears in the table in the Sensitive Column Types page.

3.3 Associating a Database to an Existing ADM

After you have created an Application Data Model (ADM), you can select additional databases to be associated databases of an ADM, as explained in the following procedure.

To associate a database to an ADM:

1. From the Application Data Models page, select an ADM, select **Actions**, then **Associated Databases**.

   This dialog lists all of the databases associated with this ADM and the schemas assigned to each application per database. You can add more databases that give you a choice of data sources when subsetting and databases to mask during masking.
2. Click **Add**, then select a database from the pop-up.

   The selected database now appears in the Database section of the Associated Databases dialog.
3. To change a schema, select the associated database on the left, select the application on the right for which the schema is to be changed, then click **Select Schema**.
4. Select the missing schema from the list in the pop-up, then click **Select**.

Related Topics:

- Creating an ADM

3.4 Verifying or Synchronizing an ADM

After you have created an Application Data Model (ADM), the Source Database Status column can indicate Valid, Invalid, Needs Verification, or Needs Upgrade.

- **Invalid status**–Verify the source database to update the referential relationships in the application data model with those found in the data dictionary, and to also determine if each item in the application data model has a corresponding object in the database.
- **Needs Verification status**–You have imported an Oracle supplied template and you must verify the ADM before you can use it. This is to ensure that necessary referential relationships from data dictionary are pulled into the ADM.
- **Needs Upgrade status**–You have imported a pre-12*c* masking definition, so you now need to upgrade the ADM.

To verify a source database:

1. Select the ADM to be verified, indicated with an Invalid status.
2. From the **Actions** menu, select **Verify**.
3. Select the source database with the Invalid status, then click **Create Verification Job**.
4. Specify job parameters in the Create Verification Job pop-up, then click **Submit**.

5. After the job completes successfully, click the source database and note the object problems listed.
6. Fix the object problems, rerun the Verification Job, then check that the Source Database Status is now Valid.

3.5 Importing and Exporting an ADM

You can share Application Data Models (ADM) with other Enterprise Manager environments that use a different repository by exporting an ADM, which can subsequently be imported into the new repository.

An exported ADM is by definition in the XML file format required for import. You can edit an exported ADM XML file prior to import. When exporting an ADM for subsequent import, it is best to have one that uses most or all of the features—applications, tables, table types, referential relationships, sensitive columns. This way, if you are going to edit the exported file prior to import, it is clear which XML tags are required and where they belong in the file.

- Importing an ADM
- Exporting an ADM

> **Note:**
>
> There are EMCLI verbs to export and import an ADM if you want to perform these operations remotely or script them.

3.5.1 Importing an ADM

There are two methods of import:

- Importing an ADM XML File from your Desktop
- Importing an ADM XML file from the Software Library

3.5.1.1 Importing an ADM XML File from your Desktop

1. From the **Application Data Models** page, select the ADM you want to import.
2. From the **Actions** menu, select **Import**, then select **File from Desktop**.
3. In the pop-up that appears, specify a name for the ADM, the source database you want to assign to the ADM, and location on your desktop from which you want to import the ADM.
4. Click **OK**.

   The ADM now appears on the Application Data Models page.

3.5.1.2 Importing an ADM XML file from the Software Library

1. From the **Application Data Models** page, select the ADM you want to import.
2. From the **Actions** menu, select **Import**, then select **File from Software Library**.

3. In the Export File from Software Library pop-up that appears, select the desired ADM XML file on the left, then specify a name and the source database you want to assign to the ADM on the right.
4. Click **Import**.

   The ADM now appears on the Application Data Models page.

After importing an ADM, you may want to discover sensitive columns or run a verification job. In the process of performing these tasks, the PL/SQL metadata collection packages are automatically deployed on the target database. The Database user must have DBA privileges to auto-deploy the packages.

3.5.2 Exporting an ADM

There are three methods of export:

- Exporting an ADM as an XML File to Your Desktop
- Exporting an ADM
- Exporting an ADM to a Transparent Sensitive Data Protection Catalog

3.5.2.1 Exporting an ADM as an XML File to Your Desktop

1. From the Application Data Models page, select the ADM you want to export.
2. From the **Actions** menu, select **Export**, then select **Selected Application Data Model**.
3. In the File Download pop-up that appears, click **Save**.
4. In the Save As pop-up that appears, navigate to a file location and click **Save**.

   The system converts the ADM into an XML file that now appears at the specified location on your desktop.

3.5.2.2 Exporting an ADM

1. From the **Actions** menu, select **Export**, then select **File from Software Library**.
2. From the **Actions** menu, select **Export**, then select **File from Software Library**.
3. In the Export File from Software Library pop-up that appears, select the desired ADM and click **Export**.
4. In the File Download pop-up that appears, click **Save**.
5. In the Save As pop-up that appears, navigate to a file location and click **Save**.

   The system converts the ADM into an XML file that now appears at the specified location on your desktop.

3.5.2.3 Exporting an ADM to a Transparent Sensitive Data Protection Catalog

1. From the Application Data Models page, select the ADM you want to export.
2. From the **Actions** menu, select **Export**, then select **Export to TSDP Catalog.**
3. The Application Data Models page displays a table of associated databases. Select a database and click the **Export Sensitive Data** button.

4. In the Export Sensitive Data pop-up that appears, provide credentials for the selected database and click **OK**.

   A message appears on the Application Data Models page confirming that the sensitive data was copied to the database.

For detailed information on TSDP, see *Oracle Database Security Guide*.

3.6 Assigning Privileges to an Existing ADM

You can grant privileges on an Application Data Model that you create so that others can have access. To do so, you must be an Enterprise Manager Administrator with at least Designer privileges on the ADM.

To assign privileges to an existing ADM:

1. From the **Enterprise** menu, select **Quality Management**, then select **Application Data Models**.
2. Select the ADM to which you want to grant privileges.
3. From the **Actions** menu, select **Grant**, then select one of the following:
   - **Operator**–to grant Operator privileges on the ADM to selected roles or administrators, which means the grantees can view and copy but not edit and delete the definition.
   - **Designer**–to grant Designer privileges on the ADM to selected roles or administrators, which means the grantees can view, edit, and delete the definition.
4. In the dialog that opens, select the type (administrator or role, or both). Search by name, if desired. Make your selections and click **Select**.

   The selected names now have privileges on the ADM.
5. Use the **Revoke** action if you want to deny any privileges that were previously granted.

4

Data Masking

This chapter provides conceptual information about the components that comprise Oracle Data Masking, and procedural information about performing the task sequence, such as creating masking formats and masking definitions. Data masking presupposes that you have created an Application Data Model (ADM) with defined sensitive columns.

The procedures in this chapter are applicable to Oracle Enterprise Manager Cloud Control 12.1 and higher only. You must have the Oracle Data Masking and Subsetting Pack license to use data masking features.

> **Note:**
>
> Performing masking on an 11.2.0.3 database that uses Database Plug-in 12.1.0.3 and higher requires that the database patch #16922826 is applied for masking to run successfully. The Mask-In Export feature (also known as At Source masking) works with Oracle Database 11.1 and higher.

4.1 Overview of Oracle Data Masking

Enterprises run the risk of breaching sensitive information when copying production data into non-production environments for the purposes of application development, testing, or data analysis. Oracle Data Masking helps reduce this risk by irreversibly replacing the original sensitive data with fictitious data so that production data can be shared safely with non-production users. Accessible through Oracle Enterprise Manager, Oracle Data Masking provides end-to-end secure automation for provisioning test databases from production in compliance with regulations.

4.1.1 Data Masking Concepts

Data masking (also known as data scrambling and data anonymization) is the process of replacing sensitive information copied from production databases to test non-production databases with realistic, but scrubbed, data based on masking rules. Data masking is ideal for virtually any situation when confidential or regulated data needs to be shared with non-production users. These users may include internal users such as application developers, or external business partners such as offshore testing companies, suppliers and customers. These non-production users need to access some of the original data, but do not need to see every column of every table, especially when the information is protected by government regulations.

Data masking enables organizations to generate realistic and fully functional data with similar characteristics as the original data to replace sensitive or confidential information. This contrasts with encryption or Virtual Private Database, which simply hides data, and the original data can be retrieved with the appropriate access or key. With data masking, the original sensitive data cannot be retrieved or accessed.

Names, addresses, phone numbers, and credit card details are examples of data that require protection of the information content from inappropriate visibility. Live production database environments contain valuable and confidential data—access to this information is tightly controlled. However, each production system usually has replicated development copies, and the controls on such test environments are less stringent. This greatly increases the risks that the data might be used inappropriately. Data masking can modify sensitive database records so that they remain usable, but do not contain confidential or personally identifiable information. Yet, the masked test data resembles the original in appearance to ensure the integrity of the application.

4.1.2 Roles of Data Masking Users

The following types of users participate in the data masking process for a typical enterprise:

- Application database administrator or application developer

  This user is knowledgeable about the application and database objects. This user may add additional custom database objects or extensions to packaged applications, such as the Oracle E-Business suite.

- Information security administrator

  This user defines information security policies, enforces security best practices, and also recommends the data to be hidden and protected.

4.1.3 Related Oracle Security Offerings

Besides data masking, Oracle offers the following security products:

- Virtual Private Database or Oracle Label Security — Hides rows and data depending on user access grants.
- Transparent Data Encryption — Hides information stored on disk using encryption. Clients see unencrypted information.
- `DBMS_CRYPTO` — Provides server packages that enable you to encrypt user data.
- Database Vault — Provides greater access controls on data.

4.1.4 Agent Compatibility for Data Masking

Data masking supports Oracle Database 9*i* and newer releases. If you have a version prior to 11.1, you can use it by implementing the following workaround.

Replace the following file...

```
AGENT_HOME/sysman/admin/scripts/db/reorg/reorganize.pl
```

... with this file:

```
OMS_HOME/sysman/admin/scripts/db/reorg/reorganize.pl
```

4.2 Format Libraries and Masking Definitions

To mask data, the Data Masking Pack provides two main features:

- Masking format library

The format library contains a collection of ready-to-use masking formats. The library consists of format routines that you can use for masking. A masking format can either be one that you create, or one from the list of Oracle-supplied default masking formats.

As a matter of best practice, organizations should create masking formats for all commonly regulated information so that the formats can be applied to the sensitive data regardless of which database the sensitive data resides in. This ensures that all sensitive data is consistently masked across the entire organization.

- Masking definitions

  A masking definition defines a data masking operation to be implemented on one or more tables in a database. Masking definitions associate table columns with formats to use for masking the data. They also maintain the relationship between columns that are not formally declared in the database using related columns.

  You can create a new masking definition or use an existing definition for a masking operation. To create a masking definition, you specify the column of the table for which the data should be masked and the format of masked data. If the columns being masked are involved in unique, primary key, or foreign key constraints, data masking generates the values so that the constraints are not violated. Masking ensures uniqueness per character using decimal arithmetic. For example, a 5-character string generates a maximum of only 99999 unique values. Similarly, a 1-character string generates a maximum of only 9 unique values.

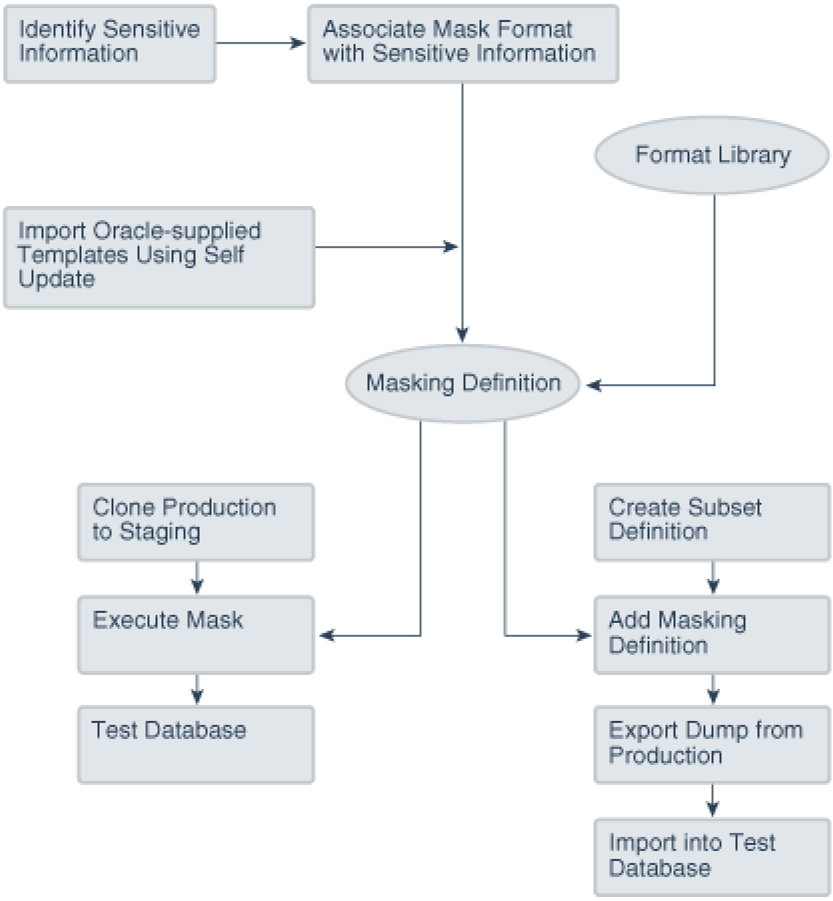
  You would typically export masking definitions to files and import them on other systems. This is important when the test and production sites reside on different Oracle Management Systems or on entirely different sites.

> **See Also:**
>
> The online help topic "Creating a Data Masking Definition" as well as the help for each Data Masking page

4.3 Recommended Data Masking Workflow

The following figure shows that the production database is cloned to a staging region and then masked there. During the masking process, the staging and test areas are tightly controlled like a production site.

Figure 4-1   Data Masking Workflow


Identify Sensitive Information
Associate Mask Format with Sensitive Information
Format Library
Import Oracle-supplied Templates Using Self Update
Masking Definition
Clone Production to Staging
Execute Mask
Test Database
Create Subset Definition
Add Masking Definition
Export Dump from Production
Import into Test Database

Data masking is an iterative and evolving process handled by the security administrator and implemented by the database administrator. When you first configure data masking, try out the masking definition on a test system, then add a greater number of columns to the masking definition and test it to make sure it functions correctly and does not break any application constraints. During this process, you should exercise care when removing all imbedded references to the real data while maintaining referential integrity.

After data masking is configured to your satisfaction, you can use the existing definition to repeatedly mask after cloning. The masking definition, however, would need to evolve as new schema changes require new data and columns to be masked.

After the masking process is complete, you can distribute the database for wide availability. If you need to ship the database to another third-party site, you are required to use the Data Pump Export utility, and then ship the dump file to the remote site. However, if you are retaining the masked data in-house, see "Data Masking Task Sequence".

You can also perform inline, or at the source, data masking while creating a subset definition.

Related Topics:

- Creating a Data Subset Definition

4.4 Data Masking Task Sequence

The task sequence in this section demonstrates the data masking workflow and refers you to additional information about some of the tasks in the sequence. Before reviewing this sequence, note that there are two options for completing this process:

- Exporting/importing to another database

  You can clone the production database to a staging area, mask it, then export/ import it to another database before delivering it to in-house testers or external customers. This is the most secure approach.

- Making the staging area the new test region

  You can clone the production database to a mask staging area, then make the staging area the new test region. In this case, you should not grant testers `SYSDBA` access or access to the database files. Doing so would compromise security. The masked database contains the original data in unused blocks and in the free list. You can only purge this information by exporting/importing the data to another database.

The following basic steps guide you through the data masking process, with references to other sections for supporting information.

1. Review the application database and identify the sources of sensitive information.
2. Define mask formats for the sensitive data. The mask formats may be simple or complex depending on the information security needs of the organization.
3. Create a masking definition to associate table columns and edition view objects to these mask formats. Data masking determines the database foreign key relationships and adds foreign key columns to the mask.
4. Save the masking definition and generate the masking script.
5. Verify if the masked data meets the information security requirements. Otherwise, refine the masking definition, restore the altered tables, and reapply the masking definition until the optimal set of masking definitions has been identified.
6. Clone the production database to a staging area, selecting the masking definition to be used after cloning. Note that you can clone using Oracle Enterprise Manager, which enables you to add masking to the Oracle Enterprise Manager clone workflow. However, if you clone outside of Oracle Enterprise Manager, you must initiate masking from Oracle Enterprise Manager after cloning is complete. The cloned database should be controlled with the same privileges as the production system, because it still contains sensitive production data.

   After cloning, make sure you change the passwords as well as update or disable any database links, streams, or references to external data sources. Back up the cloned database, or minimally the tables that contain masked data. This can help you restore the original data if the masking definition needs to be refined further.
7. After masking, test all of your applications, reports, and business processes to ensure they are functional. If everything is working, you can export the masking definition to keep it as a back-up.
8. After masking the staging site, make sure to drop any tables named `MGMT_DM_TT` before cloning to a test region. These temporary tables contain a mapping between the original sensitive column value and the mask values, and are therefore sensitive in nature.

During masking, Oracle Enterprise Manager automatically drops these temporary tables for you with the default "Drop temporary tables created during masking" option. However, you can preserve these temporary tables by deselecting this option. In this case, you are responsible for deleting the temporary tables before cloning to the test region.

9. After masking is complete, ensure that all tables loaded for use by the substitute column format or table column format are going to be dropped. These tables contain the mask values that table column or substitute formats will use. It is recommended that you purge this information for security reasons.
10. Clone the database to a test region, or use it as the new test region. When cloning the database to an external or unsecured site, you should use Export or Import. Only supply the data in the database, rather than the database files themselves.
11. As part of cloning production for testing, provide the masking definition to the application database administrator to use in masking the database.

Related Topics:

- Creating New Masking Formats
- Using Oracle-supplied Predefined Masking Formats
- Masking with an Application Data Model and Workloads
- Cloning the Production Database
- Deterministic Masking Using the Substitute Format

4.5 Defining Masking Formats

A masking definition requires one or more masking formats for any columns included in the masking definition. When adding columns to a masking definition, you can either create masking formats manually or import them from the format library. It is often more efficient to work with masking formats from the format library.

4.5.1 Creating New Masking Formats

This section describes how to create new masking formats using Enterprise Manager.

To create a masking format in the format library:

1. From the Enterprise menu, select **Quality Management**, then **Data Masking Formats**. Alternatively, if you are in the Database home page, select **Data Masking Format Library** from the Schema menu.

   The Format Library appears with predefined formats that Oracle Enterprise Manager provides.
2. Click **Create**.

   The Create Format page appears, where you can define a masking format.

> **See Also:**
>
> The online help for information on page user controls

3. Provide a required name for the new format, select a format entry type from the **Add** list, then click **Go**.

   A page appears that enables you to provide input for the format entry you have selected. For instance, if you select Array List, the subsequent page enables you to enter a list of values, such as New York, New Jersey, and New Hampshire.

4. Continue adding additional format entries as needed.
5. When done, provide an optional user-defined or post-processing function, then click **OK** to save the masking format.

   The Format Library page reappears with your newly created format displayed in the Format Library table. You can use this format later to mask a column of the same sensitive type.

> **See Also:**
>
> The online help for information on the Format Library and Create Format pages

Related Topics:

- Providing User-Defined and Post-Processing Functions

4.5.1.1 Providing User-Defined and Post-Processing Functions

If desired, you can provide user-defined and post-processing functions on the Create Format page. A user-defined choice is available in the Add list, and a post-processing function field is available at the bottom of the page.

- User-defined functions

  To provide a user-defined function, select **User Defined Function** from the Add list, then click **Go** to access the input fields.

  A user-defined function passes in the original value as input, and returns a mask value. The data type and uniqueness of the output values must be compatible with the original output values. Otherwise, a failure occurs when the job runs. Combinable, a user-defined function is a PL/SQL function that can be invoked in a SELECT statement. Its signature is returned as:

  ```
  Function udf_func (rowid varchar2, column_name varchar2, original_value
  varchar2) return varchar2;
  ```

  – rowid is the min (rowid) of the rows that contain the value original_value 3rd argument.

  – column_name is the name of the column being masked.

  – original_value is the value being masked.

  That is, it accepts the original value as an input string, and returns the mask value.

  Both the input and output values are varchar2. For instance, a user-defined function to mask a number could receive 100 as input, the string representation of the number 100, and return 99, the string representation of the number 99. Values are cast appropriately when inserting to the table. If the value is not castable, masking fails.

- Post-processing functions

  To provide a post-processing function, enter it in the **Post Processing Function** field.

  A post-processing function has the same signature as a user-defined function, but passes in the mask value the masking engine generates, and returns the mask value that should be used for masking, as shown in the following example:

  ```
  Function post_proc_udf_func (rowid varchar2, column_name varchar2, mask_value
  varchar2) return varchar2;
  ```

  - `rowid` is the min (rowid) of the rows that contain the value `mask_value`.
  - `column_name` is the name of the column being masked.
  - `mask_value` is the value being masked.

4.5.1.2 Using Masking Format Templates

After you have created at least one format, you can use the format definition as a template in the Create Format page, where you can implement most of the format using a different name and changing the entries as needed, rather than needing to create a new format from scratch.

To create a new format similar to an existing format, select a format on the Format Library page and click **Create Like**. The masking format you select can either be one you have previously defined yourself, or one from the list of out-of-box masking formats. You can use these generic masking format definitions for different applications.

For instructional details about the various Oracle-supplied predefined masking format definitions and how to modify them to suit your needs, see "Using Oracle-supplied Predefined Masking Formats".

4.5.2 Using Oracle-supplied Predefined Masking Formats

Enterprise Manager provides several out-of-box predefined formats. All predefined formats and built-in formats are random. The following sections discuss the various Oracle-supplied format definitions and how to modify them to suit your needs:

- Patterns of Format Definitions
- Category Definitions

> **See Also:**
>
> "Installing the DM_FMTLIB Package" for information on installing the DM_FMTLIB package so that you can use the predefined masking formats

4.5.2.1 Patterns of Format Definitions

All of the format definitions adhere to these typical patterns:

- Generate a random number or random digits.

- Perform post-processing on the above-generated value to ensure that the final result is a valid, realistic value.

For example, a valid credit card number must pass Luhn's check. That is, the last digit of any credit card number is a checksum digit, which is always computed. Also, the first few digits indicate the card type (MasterCard, Amex, Visa, and so forth). Consequently, the format definition of a credit card would be as follows:

- Generate random and unique 10-digit numbers.
- Using a post-processing function, transform the values above to a proper credit card number by adding well known card type prefixes and computing the last digit.

This format is capable of generating 10 billion unique credit card numbers.

4.5.2.2 Category Definitions

The following sections discuss different categories of these definitions:

- Credit Card Numbers
- United States Social Security Numbers
- ISBN Numbers
- UPC Numbers
- Canadian Social Insurance Numbers
- North American Phone Numbers
- UK National Insurance Numbers
- Auto Mask

By default, these mask formats are also available in different format styles, such as a hyphen (-) format. If needed, you can modify the format style.

4.5.2.2.1 Credit Card Numbers

Out of the box, the format library provides many different formats for credit cards. The credit card numbers generated by these formats pass the standard credit card validation tests by the applications, thereby making them appear like valid credit card numbers.

Some of the credit card formats you can use include:

- MasterCard numbers
- Visa card numbers
- American Express card numbers
- Discover Card numbers
- Any credit card number (credit card numbers belong to all types of cards)

You may want to use different styles for storing credit card numbers, such as:

- Pure numbers
- 'Space' for every four digits
- 'Hyphen' (-) for every four digits, and so forth

To implement the masked values in a certain format style, you can set the DM_CC_FORMAT variable of the DM_FMTLIB package. To install the package, see "Installing the DM_FMTLIB Package".

4.5.2.2.2 United States Social Security Numbers

Out of the box, you can generate valid U.S. Social Security (SSN) numbers. These SSNs pass the normal application tests of a valid SSN.

You can affect the format style by setting the DM_SSN_FORMAT variable of the DM_FMTLIB package. For example, if you set this variable to '-', the typical social security number would appear as '123-45-6789'.

4.5.2.2.3 ISBN Numbers

Using the format library, you can generate either 10-digit or 13-digit ISBN numbers. These numbers adhere to standard ISBN number validation tests. All of these ISBN numbers are random in nature. Similar to other format definitions, you can affect the "style" of the ISBN format by setting values to DM_ISBN_FORMAT.

4.5.2.2.4 UPC Numbers

Using the format library, you can generate valid UPC numbers. They adhere to standard tests for valid UPC numbers. You can affect the formatting style by setting the DM_UPC_FORMAT value of the DM_FMTLIB package.

4.5.2.2.5 Canadian Social Insurance Numbers

Using the format library, you can generate valid Canadian Social Insurance Numbers (SINs). These numbers adhere to standard tests of Canadian SINs. You can affect the formatting style by setting the DM_CN_SIN_FORMAT value of the DM_FMTLIB package.

4.5.2.2.6 North American Phone Numbers

Out of the box, the format library provides various possible U.S. and Canadian phone numbers. These are valid, realistic looking numbers that can pass standard phone number validation tests employed by applications. You can generate the following types of numbers:

- Any North American phone numbers
- Any Canadian phone number
- Any U.S.A. phone number

4.5.2.2.7 UK National Insurance Numbers

Using the format library, you can generate valid unique random UK National Insurance Numbers (NINs). These numbers adhere to standard tests of UK NINs. A typical national insurance number would appear as 'GR 12 56 34 RS'.

4.5.2.2.8 Auto Mask

This format scrambles characters and numbers into masked characters and numbers and while retaining the format and length of the data, including special characters; for example, 'ABCD_343-ddg' masked as 'FHDT_657-tte'.

4.5.2.3 Installing the DM_FMTLIB Package

The predefined masking formats use functions defined in the `DM_FMTLIB` package. This package is automatically installed in the `DBSNMP` schema of your Enterprise Manager repository database. To use the predefined masking formats on a target database (other than the repository database), you must manually install the `DM_FMTLIB` package on that database.

To install the DM_FMTLIB package:

1. Locate the following scripts in your Enterprise Manager installation:

   ```
   $PLUGIN_HOME/sql/db/latest/masking/dm_fmtlib_pkgdef.sql
   $PLUGIN_HOME/sql/db/latest/masking/dm_fmtlib_pkgbody.sql
   ```

   Where PLUGIN_HOME can be any of the locations returned by the following SQL SELECT statement, executed as SYSMAN:

   ```
   select PLUGIN_HOME from gc_current_deployed_plugin where
   plugin_id='oracle.sysman.db' and destination_type='OMS';
   ```

2. Copy these scripts to a directory in your target database installation and execute them using SQL*Plus, connected as a user that can create packages in the DBSNMP schema.

   You can now use the predefined masking formats in your masking definitions.

3. Select and import any predefined masking format into a masking definition by clicking the **Import Format** button on the Define Column Mask page.

4.5.3 Providing a Masking Format to Define a Column

When you create a masking definition ("Masking with an Application Data Model and Workloads"), you will be either importing a format or selecting one from the available types in the Define Column Mask page. Format entry options are as follows:

- **Array List**

  Accepts a list of values as input and maps each value in the list to a value in the input column. The number of values in the list should be greater than or equal to the number of distinct values in the masked column. The values in the user-provided list are ordered randomly before mapping them to the original column values. For example, if the original column contains values [10,20,30,40,50] and the Array List specified by the user is [99,100,101,102,103], the first masking run could produce the mapping [10,101], [20,103], [30,100], [40,99], [50,102] and a different masking run can produce [10,100], [20,99], [30,101], [40,102], [50,103].

  A mapping table is created. The CTAS that creates the mapping table queries from:

  1. The original table to fetch the column values being masked and a row number for each column value. The row number is derived from the Oracle-supplied ROW_NUMBER function.
  2. The user-passed list of values — values in the user-passed array list are converted into a table-like record set using the SQL TABLE function. A row number is also retrieved corresponding to each value in the record set. The row number is derived from the ROWNUM pseudo column. The values in the record set are randomly ordered using `DBMS_RANDOM.VALUE` function.

3. The mapping table CTAS then joins the row numbers in both sub-queries to map the original value (from sub-query in step 1 above) and a value from the user-list (sub-query in step 2 above). Multiple executions of the CTAS will create a mapping table with different original-masked value mappings because of the random ordering of the user list in step 2.

- **Delete**

  Deletes a row based on a condition. If the condition matches, then the row is deleted on the target. A mapping table is created. The “DELETE_VAL” column in the mapping table is set to 1 for rows that are candidates to be deleted. For example, we are masking the SALARY column and the masking definition has conditions on the EMPID column and formats defined as:

  ```
  EMPID < 100
        DELETE
  EMPID < 200
          RANDOM NUMBER [Start Value:1 End Value:100]
  DEFAULT
          PRESERVE
  ```

  The mapping table will have the DELETE_VAL column set to 1 for SALARY rows with EMPID < 100. DELETE_VAL for all other rows is set to 0. The final masking CTAS SQL which joins the original table and the mapping table to create the masked table filters out rows with DELETE_VAL set to 1. Therefore, the rows in the original table that match the join condition are effectively “deleted”.

- **Encrypt**

  The data in the table is fixed and does not change. Encrypts a column based on a regular expression. The column values in all the rows must match the regular expression. For a given value, the format always generates the same masked value. This format implements Format Preserving Encryption (FPE) by invoking dbms_crypto using 3DES with multiple cycles. The format is not trivial and involves significant processing. The function used is:

  ```
  function encrypt( orig_value varchar2, fmt mgmt_dm_formatmap,
  max_value number,inpsd number) return varchar2; max_value
  number,inpsd number) return varchar2;
  ```

  `orig_value` is the original value of the column before it was masked. `fmt` and `max_value` are specific to the FPE and are known at the time of generating the script. `inpsd` is the input seed the user specifies when executing the generated script. This value is not known during script generation time. When the SQL invoking this function is executed during incremental masking, the seed value is required. One option is to persist it in a table on the target when the user executes the masking script. The PL/SQL code executed during replication will look like this:

  1. Fetch the seed from the table.
  2. Invoke the SQL query that has a call to encrypt (and possibly other formats). Pass the replicated column value and the seed obtained from step 1 to the procedure. The query will return the masked value.

  Encrypt creates a mapping table when there is a combination of formats under conditions. No mapping table is created when Encrypt is the only format available for the column, since the PL/SQL code implementing encrypt is invoked inline from the final masking SQL.

- **Fixed Number**

This format does not use a lookup or a mapping table. It assigns a fixed number value to a string/number column.

The type of column applicable to this entry is a `NUMBER` column or a `STRING` column. For example, if you mask a column that has a social security number, one of the entries can be Fixed Number 900. This format is combinable.

- **Fixed String**

  This format does not use a lookup or a mapping table. It assigns a fixed string value to a string column.

  The type of column applicable to this entry is a `STRING` column. For example, if you mask a column that has a License Plate Number, one of the entries can be Fixed String CA. This format is combinable.

- **Null Value**

  Masks the column with a value of NULL. It does not use a lookup or a mapping table.

> **Note:**
>
> Fixed number/string and null value formats are implemented through mapping tables when a column is masked through multiple formats. For example, mapping table is used when SALARY is masked with fixed number for EMPID<100 and with random numbers for EMPID>100. Similar to the encrypt format, no mapping table is created when the column does not have a combination of formats. In the example, no mapping table is created when only fixed number is used to mask SALARY for all EMPID values. In this case, the formats are implemented inline in the final masking SQL.

- **Post-Processing Function**

  The format allows users to use a custom function to process column values after they are masked using standard data masking formats. For example, the SALARY column can be masked with a SQL expression first, and a post processing function can be applied on the masked values to add a currency symbol, like ‘$’. The function has a fixed signature:

  ```
  function post_proc_func(rowid varchar2, column_name varchar2,
  mask_value varchar2) returns varchar2;
  ```

  The **ROWID** input allows a user to fetch column values from the masked table. The function could these values to mask the input column value, basically to transform the column further after a standard format is applied on the column. This format creates a mapping table. The post processing function gets invoked as part of the mapping table CTAS SQL. The input to the **mask_value** argument of the function is the masked value of the original column. For example, say we are masking the **SALARY** column and the mask definition has conditions on the EMPID column and formats are defined this way:

  ```
  EMPID < 100
        RANDOM NUMBERS [START:100000 END: 10000000]
        POST PROCESSING FUNCTION ppf
  EMPID < 200
          FIXED NUMBER 100000
  DEFAULT
          PRESERVE
  ```

- **Preserve Original Data**

  Preserves the original column value. Used in conditional masking with a combination of other formats where only a subset of values needs to be masked based on a condition.

- **Random Dates**

  The format creates a mapping table. The mapping table CTAS contains code to generate random dates within a user specified date range. A random date is generated using the following logic:

  ```
  TO_DATE(start_date','YYYY-DD-MM HH24:MI:SS') +
  mask_util.genrnd(0, <#of days between the specified date range>)
  ```

- **Random Decimal Numbers**

  If used as part of a mixed random string, these have limited usage for generating unique values. This masking format generates unique values within the specified range. For example, a starting value of 5.5 and ending value of 9.99 generates a decimal number ranging from 5.5 to 9.99, both inclusive. This masking format is combinable.

- **Random Digits**

  This format generates unique values within the specified range. For example, for a random digit with a length of [5,5], an integer between [0, 99999] is randomly generated, left padded with '0's to satisfy the length and uniqueness requirement. This is a complementary type of random number, which will not be padded. When using random digits, the random digit pads to the appropriate length in a string. It does not pad when used for a number column. This format is combinable.

  Data masking ensures that the generated values are unique, but if you do not specify enough digits, you could run out of unique values in that range.

- **Random Numbers**

  If used as part of a mixed random string, these have limited usage for generating unique values. This format generates unique values within the specified range. For example, a starting value of 100 and ending value of 200 generates an integer number ranging from 100 to 200, both inclusive. Note that Oracle Enterprise Manager release 10.2.0.4.0 does not support float numbers. This format is combinable.

- **Random Strings**

  This format generates unique values within the specified range. For example, a starting length of 2 and ending length of 6 generates a random string of 2 - 6 characters in length. This format is combinable.

- **Regular Expression**

  The format uses a lookup table. No mapping table is created. The PL/SQL function that implements the format is invoked directly from the final CTAS which creates the masked table. The lookup table has two columns to store the regular expression and the replacement value specified by the user. The `SQL REGEXP_REPLACE` function is used to implement this format.

  The function has the signature:

  ```
  regexp_replace(column_value, regex, replacement_val);
  ```

  For example, phone numbers in the format `nnn.nnn.nnnn` can be masked using a regex `[1-9]{3}[.][0-9]{3}[.][0-9]{4}` with a replacement value `***.***.****`. The format invokes the `regexp_replace` for each regexp-replacement value pair. If the phone number column was masked using regular expression:

```
EMPID < 100
  Regular Expression      Regex: [1-9]{3}[.][0-9]{3}[.][0-9]{4}  Replacement 
Value: 999.444.555
  Regular Expression      Regex: [9]{3}[.][4]{3}[.][5]{4}       Replacement 
Value: ***.***.***
```

Each column value matching the first regular expression is first replaced with `999.444.555`, this value then matches the second regular expression and is replaced with `***.***.***`. The example is not a real world use case. The behavior probably is a side effect of how the format is implemented, the real use case of specifying multiple regular expression formats to mask a column is to handle cases when the data in the column could match multiple regular expressions. For example:

```
EMPID < 100
  Regular Expression      Regex: [1-9]{3}[.][0-9]{3}[.][0-9]{4}  Replacement 
Value: ***.***.****
  Regular Expression      Regex: [1-9]{3}[.][0-9]{3}[.][0-9]{3}  Replacement 
Value: ***.***.***
```

can be used to mask column values that store 10 digit - nnn.nnn.nnnn - or 9 digit - nnn.nnn.nnn - phone numbers.

- **Shuffle**

  This format does not use a lookup or a mapping table. The final CTAS which creates the mapping table includes a sub query to order the column contents randomly using the `DBMS_RANDOM.VALUE` function. If shuffle is used with a grouping column, the `PARTITION` clause is used to partition by the grouping column and the column is ordered randomly within each partition. The implementation is similar to that of Array List and Table Column. The random ordering for the shuffle format occurs on the column being “shuffle masked”, whereas in Array List, it is on the user-passed list, and in Table Column, the ordering is on the user-specified column.

- **Substitute**

  The format creates a mapping table. It uses a user specified “substitution” table as a source for masked values. The format uses the Oracle supplied hash based partitioning function `ORA_HASH` to map a column value to its mask value in a lookup (substitution) table. Processing involves querying the substitution table to get a count of distinct values in the mask column. This count – say `n` - is then used as the `max_bucket` parameter of `ORA_HASH` to hash the original column values into n buckets. For example, if we are masking `EMPLOYEE.SALARY` and using `SUBST.SUB_COL` column as the substitution column, we first get the count of distinct values in `SUB_COL`. The mapping table CTAS then queries:

  1. The original column, `EMPLOYEE.SALARY`
  2. The user provided substitution table to fetch all the distinct values in `SUBST.SUB_COL` and also fetches the `ROWNUM` associated with each row

  The CTAS SQL then joins 1 and 2 using `ORA_HASH` and equating its output to the `ROWNUM` from step 2. The `SELECT` part of the CTAS SQL is listed below. `max_bckt` is the count of distinct values in the substitution column `SUBST.SUB_COL`:

```
select s.orig_val,
       a0.new_val
```

```
from ( select orig_val
       from (select "SALARY" orig_val
             from "TESTU"."EMPLOYEE")
       group by orig_val) s,
     (select rownum rn,
             SUB_COL new_val
      from (select distinct SUB_COL
            from TESTU.SUBST
            order by SUB_COL)) a0
 where ora_hash(s.orig_val, max_bckt, seed)+1 = a0.rn
```

- **SQL Expression**

  The format does not create a mapping table. It allows a user to use a SQL Expression for masking a column. Data masking uses this expression to generate masked values to replace the original values. The expression is invoked directly from the masking CTAS SQL. The SQL Expression can consist of one or more values, operators, and SQL functions that evaluates to a value. It can also contain substitution columns (columns from the same table as the masked column). Some examples of valid expressions:

  ```
  1.    dbms_random.string('u', 8) || '@company.com'
  2.    %first_name% || '.' || %last_name% || '@company.com'
  3.    dbms_lob.empty_clob()
  4.    custom_mask_clob(%CLOB_COL%)
  5.    (case when %PARTY_TYPE%='PERSON' then %PERSON_FIRST_NAME%|| ' ' ||
  %PERSON_LAST_NAME% else (select dbms_random.string('U', 10) from dual) end)
  6.    select MASK_ZIPCODE from data_mask.DATA_MASK_ADDR where ADDR_SEQ =
  ora_hash( %ZIPCODE% , 1000, 1234)
  ```

- **Substring**

  The format creates a mapping table. The mapping table CTAS invokes the Oracle SUBSTR function on the input column. The format accepts a start position and length as input, extracts that data from the input column using `SUBSTR`, and uses that as a mask value.

- **Table Column**

  The format creates a mapping table. The format maps original column values to column values in a user specified table. The processing is similar to the array list format. The values in the user specified table are randomly ordered using `DBMS_RANDOM.VALUE` before mapping each value to the original column. Unlike the Substitute format, the format is not deterministic since the substitution column is randomly ordered.

- **Truncate**

  The format truncates all rows in a table. It does not create a mapping table. If one of the columns in a table is masked using this format, so no other mask formats can be specified for any of the other columns.

- **User Defined Function**

  The format creates a mapping table. The return value of the user defined function is used to mask the column. The function is invoked as part of the mapping table CTAS. The function has a fixed signature:

  ```
  function userdef_func(rowid varchar2, col_name varchar2, orig_val varchar2)
  returns varchar2;Function udf_func (rowid varchar2, column_name varchar2,
  original_value varchar2) return varchar2;
  ```

4.5.4 Deterministic Masking Using the Substitute Format

You may occasionally need to consistently mask multiple, distinct databases. For instance, if you run HR, payroll, and benefits that have an employee ID concept on three separate databases, the concept may be consistent for all of these databases, in that an employee's ID can be selected to retrieve the employee's HR, payroll, or benefits information. Based on this premise, if you were to mask the employee's ID because it actually contains his/her social security number, you would have to mask this consistently across all three databases.

Deterministic masking provides a solution for this problem. You can use the Substitute format to mask employee ID column(s) in all three databases. The Substitute format uses a table of values from which to substitute the original value with a mask value. As long as this table of values does not change, the mask is deterministic or consistent across the three databases.

> **See Also:**
>
> The online help for Define Column Mask page for more information on the Substitute format

4.6 Masking with an Application Data Model and Workloads

Before creating a masking definition, note the following prerequisites and advisory information:

- Ensure that you have the following minimum privileges for data masking:
  - **Target Privileges (applicable to all targets):**
    - Connect to any viewable target
    - Execute Command Anywhere
    - View Any Target
  - **Resource Privileges:**
    - Job System
    - Named Credential
    - Oracle Data Masking and Subsetting resource privilege

> **Note:**
>
> The `EM_ALL_OPERATOR` privilege for Enterprise Manager Cloud Control users includes all of the above privileges.

  - `SELECT_CATALOG_ROLE` for database users
  - `SELECT ANY DICTIONARY` privilege for database users
  - `EXECUTE` privileges for the DBMS_CRYPTO package

- Ensure the format you select does not violate check constraints and does not break any applications that use the data.
- For triggers and PL/SQL packages, data masking recompiles the object.
- Exercise caution when masking partitioned tables, especially if you are masking the partition key. In this circumstance, the row may move to another partition.
- Data Masking does not support clustered tables, masking information in object tables, XML tables, and virtual columns. Relational tables are supported for masking.
- If objects are layered on top of a table such as views, materialized views, and PL/SQL packages, they are recompiled to be valid.

If you plan to mask a test system intended for evaluating performance, the following practices are recommended:

- Try to preserve the production statistics and SQL profiles after masking by adding a pre-masking script to export the SQL profiles and statistics to a temporary table, then restoring after masking completes.
- Run a SQL Performance Analyzer evaluation to understand the masking impact on performance. Any performance changes other than what appears in the evaluation report are usually related to application-specific changes on the masked database.

To create a masking definition:

1. From the **Enterprise** menu, select **Quality Management**, then **Data Masking Definitions**.

   The Data Masking Definitions page appears, where you can create and schedule new masking definitions and manage existing masking definitions.

2. Click **Create** to go to the Create Masking Definition page.

   A masking definition includes information regarding table columns and the format for each column. You can choose which columns to mask, leaving the remaining columns intact.

3. Provide a required **Name**, **Application Data Model**, and **Reference Database**.

   When you click the search icon and select an Application Data Model (ADM) name from the list, the system automatically populates the Reference Database field.

   - Optional: Check Ensure Workload Masking Compatibility if you want to mask Capture files and SQL Tuning Sets.

     When you enable this check box, the masking definition is evaluated to determine if the SQL Expression format or conditional masking is being used. If either is in use when you click OK, the option becomes unchecked and an error message appears asking you to remove these items before selecting this option.

> **Note:**
>
> Before proceeding to the next step, one or more sensitive columns must already be defined in the Application Data Model. See "Creating and Managing Custom Sensitive Column Types" for more information.

4. Click **Add** to go to the Add Columns page, where you can choose which sensitive columns in the ADM you want to mask.

   The results appear in the Columns table. Primary key and foreign key columns appear below the sensitive columns.

5. Use filtering criteria to refine sensitive column results. For example, perhaps you want to isolate all columns that have order in the name (column name=order%). You first may have to expose the filter section (**Show Filters**).

6. Use the disable feature to exclude certain columns from masking consideration. All columns are enabled by default. You can disable selected or all columns. You can also search for a subset of columns (column name=order%) to disable. The Status column on the right changes to reflect a column's disabled state. Note that a column's disabled state persists on export of a data masking definition.

7. Optional. Click the edit icon in the Format column to review and edit the masking format.

8. Expand **Show Advanced Options** and decide whether the selected default data masking options are satisfactory.

9. Click **OK** to save your definition and return to the Data Masking Definitions page.

   At this point, super administrators can see each other's masking definitions.

10. Select the definition and click **Generate Script**. The schedule job dialog opens. You may have to log in to the database first.

    Complete the schedule job dialog by providing the required information, then click **Submit**.

11. A message appears denoting that the job was submitted successfully and you return to the Data Masking Definitions page, where the status is "Generating Script." Click **View Job Details** to open the job summary page.

    When the job completes, click **Log Report** to check whether sufficient disk space is available for the operation, and to determine the impact on other destination objects, such as users, after masking. If any tables included in the masking definition have columns of data type `LONG`, a warning message may appear.

12. When the status on the Data Masking Definitions page is "Script Generated," select the script and choose from the following actions:

    - **Clone Database**–to clone and mask the database using the Clone Database wizard (this requires a Database Lifecycle Management Pack license).
    - **Save Script**–to save the entire PL/SQL script to your desktop.
    - **Save Mask Bundle**–to download a zip file containing the SQL files generated as part of the At source masking script generation option. You can then extract and execute the script to create a masked dump of the database.
    - **View Script**–to view the PL/SQL script, which you can edit and save. You can also view errors and warnings, if any, in the impact report.
    - **Storage Requirement Report**–to view the space required for intermittent objects during masking.

> **Note:**
>
> For accurate storage estimates, ensure that you use `dbms_stats.gather_table_stats` to gather the stats of all the tables participating in the masking process.

Click **Go** to execute the selected action.

13. If you are already working with a test database and want to directly mask the data in this database, click **Schedule Job**.

    - Provide the requisite information and desired options. You can specify the database at execution time to any database. The system assumes that the database you select is a clone of the source database. By default, the source database from the ADM is selected.
    - Click **Submit**.

      The Data Masking Definitions page appears. The job has been submitted to Enterprise Manager and the masking process appears. The Status column on this page indicates the current stage of the process.

Note that you can also perform data masking at the source as part of a data subsetting definition. See "Creating a Data Subset Definition" for more information.

Related Topics:

- Selecting Data Masking Advanced Options
- Scheduling a Script Generation Job
- Cloning the Production Database
- Scheduling a Data Masking Job

4.6.1 Editing a Data Masking Definition

Use the following procedure to edit or otherwise work with a masking definition:

1. From the Data Masking Definitions page, click the radio button next to the definition you want to edit, then click **Edit**.
2. Add columns or edit format entries as you would when creating a new definition, which is explained starting in Step 3 above.
3. Click **OK** on the Edit Masking Definitions page when you have finished editing.
4. Perform these other actions with an existing masking definition by choosing the respective menu item from the **Actions** menu:

    - **Create Like**–to display the masking definition in the Create Masking Definition page where you can customize the definition.
    - **Grant Designer**–to grant Designer privileges on the masking definition to selected roles or administrators, which means the grantees can view and edit the definition. Privileges granted can also be revoked.
    - **Grant Operator**–to grant Operator privileges on the masking definition to selected roles or administrators, which means the grantees can view and copy but not edit the definition. Privileges granted can also be revoked.
    - **Export**–to export the definition as an XML file for use in other repositories.

Click **Go** to execute the selected action.

4.6.2 Adding Columns for Masking

Use this page to add one or more columns for masking and automatically add foreign key columns. Select the database columns you want to mask from the corresponding schema. After you select the columns, you specify the format used to mask the data within.

> **Note:**
>
> You need to add at least one column in the masking definition. Otherwise, you cannot generate a script that creates an impact report that provides information about the objects and resources examined and lists details of any warnings or errors detected.

1. Enter search criteria, then click **Search**.

   The sensitive columns you defined in the ADM appear in the table below.

2. Either select one or more columns for later formatting on the Create Masking Definition page, or formatting now if the data types of the columns you have selected are identical.

3. *Optional*: if you want to mask selected columns as a group, enable Mask selected columns as a group. The columns that you want to mask as a group must all be from the same table.

   Enable this check box if you want to mask more than one column together, rather than separately. When you select two or more columns and then later define the format on the Define Group Mask page, the columns appear together, and any choices you make for format type or masking table apply collectively to all of the columns.

   After you define the group and return to this page, the Column Group column in the table shows an identical number for each entry row in the table for all members of the group. For example, if you have defined your first group containing four columns, each of the four entries in this page will show a number 1 in the Column Group column. If you define another group, the entries in the page will show the number 2, and so forth. This helps you to distinguish which columns belong to which column groups.

4. Either click **Add** to add the column to the masking definition, return to the Create Masking Definition page and define the format of the column later, or click **Define Format and Add** to define the format for the column now.

   The Define Format and Add feature can save you significant time. When you select multiple columns to add that have the same data type, you do not need to define the format for each column as you would when you click Add. For instance, if you search for Social Security numbers (SSN) and the search yields 100 SSN columns, you could select them all, then click **Define Format and Add** to import the SSN format for all of them.

5. Do one of the following:

   - If you clicked **Add** in the previous step:

You will eventually need to define the format of the column in the Create Masking Definition page before you can continue. When you are ready to do so, click the icon in the page Format column for the column you want to format. Depending on whether you decided to mask selected columns as a group on the Add Columns page, either the Define Column mask or Define Group mask appears. Read further in this step for instructions for both cases.

- If you clicked **Define Format and Add** in the previous step and did not check **Mask selected columns as a group**:

  The Define Column Mask page appears, where you can define the format for the column before adding the column to the Create Masking Definition page, as explained below:

  - Provide a format entry for the required Default condition by either selecting a format entry from the list and clicking **Add**, or clicking **Import Format**, selecting a predefined format on the Import Format page, then clicking **Import**.

    The Import Format page displays the formats that are marked with the same sensitive type as the masked column.

  - Add another condition by clicking **Add Condition** to add a new condition row, then provide one or more format entries as described in the previous step.
  - When you have finished formatting the column, click **OK** to return to the Create Masking Definition page.

- If you clicked **Define Format and Add** in the previous step and checked **Mask selected columns as a group**:

  The Define Group Mask page appears, where you can add format entries for group columns that appear in the Create Masking Definition page, as explained below:

  - Select one of the available format types. For complete information on the format types, see the online help for the Defining the Group Masking Format topic.
  - Optionally add a column to the group.
  - When you have finished formatting the group, click **OK** to return to the Create Masking Definition page.

    The results appear in the Columns table. The sensitive columns you selected earlier now appear on this page. Primary key and foreign key columns appear below the sensitive columns.

> **Note:**
>
> When you add a column for masking, and select the available format types, you can specify the value for the masking format, and click the **Show Sample** icon to view a sample data masking format. However, the **Show Sample** icon isn't displayed for the delete, truncate, and null value formats.

4.6.3 Selecting Data Masking Advanced Options

The following options on the Masking Definitions page are all checked by default, so you need to uncheck the options that you do not want to enable:

- Data Masking Options
- Random Number Generation
- Pre- and Post-mask Scripts

4.6.3.1 Data Masking Options

The data masking options include:

- Disable redo log generation during masking

  Masking disables redo logging and flashback logging to purge any original unmasked data from logs. However, in certain circumstances when you only want to test masking, roll back changes, and retry with more mask columns, it is easier to uncheck this box and use a flashback database to retrieve the old unmasked data after it has been masked. You can use Enterprise Manager to flashback a database.

  > **Note:**
  >
  > Disabling this option compromises security. You must ensure this option is enabled in the final mask performed on the copy of the production database.

- Refresh statistics after masking

  If you have already enabled statistics collection and would like to use special options when collecting statistics, such as histograms or different sampling percentages, it is beneficial to turn off this option to disable default statistics collection and run your own statistics collection jobs.

- Drop temporary tables created during masking

  Masking creates temporary tables that map the original sensitive data values to mask values. In some cases, you may want to preserve this information to track how masking changed your data. Note that doing so compromises security. These tables must be dropped before the database is available for unprivileged users.

- Decrypt encrypted columns

  This option decrypts columns that were previously masked using Encrypt format. To decrypt a previously encrypted column, the seed value must be the same as the value used to encrypt.

  Decrypt only recovers the original value if the original format used for the encryption matches the original value. If the originally encrypted value did not conform to the specified regular expression, when decrypted, the encrypted value cannot reproduce the original value.

- Use parallel execution when possible

  Oracle Database can make parallel various SQL operations that can significantly improve their performance. Data Masking uses this feature when you select this option. You can enable Oracle Database to automatically determine the degree of parallelism, or you can specify a value. For more information about using parallel execution and the degree of parallelism, see the *Oracle Database Data Warehousing Guide*.

- Recompile invalid dependent objects after masking

  The masking process re-creates the table to be masked and as a consequence, all existing dependent objects (packages, procedures, functions, MViews, Views, Triggers) become invalid. You can specify that the masking process recompile these invalid objects after creating the table, by selecting the check box. Otherwise, invalid objects are not recompiled using `utl_comp` procedures at the end of masking.

  If you choose this option, indicate whether to use serial or parallel execution. You can enable Oracle Database to automatically determine the degree, or you can specify a value. For more information about using parallel execution and the degree of parallelism, see the *Oracle Database Data Warehousing Guide*.

4.6.3.2 Random Number Generation

The random number generation options include:

- Favor Speed

  The `DBMS_RANDOM` package is used for random number generation.

- Favor Security

  The `DBMS_CRYPTO` package is used for random number generation. Additionally, if you use the Substitute format, a seed value is required when you schedule the masking job or database clone job.

4.6.3.3 Pre- and Post-mask Scripts

When masking a test system to evaluate performance, it is beneficial to preserve the object statistics after masking. You can accomplish this by adding a pre-masking script to export the statistics to a temporary table, then restoring them with a post-masking script after masking concludes.

Use the Pre Mask Script text box to specify any user-specified SQL script that must run before masking starts.

Use the Post Mask Script text box to specify any user-specified SQL script that must run after masking completes. Since masking modifies data, you can also perform tasks, such as rebalancing books or calling roll-up or aggregation modules, to ensure that related or aggregate information is consistent.

The following examples show pre- and post-masking scripts for preserving statistics.

This example shows a pre-masking script for preserving statistics.

```
variable sts_task  VARCHAR2(64);

/*Step :1 Create the staging table for statistics*/

exec dbms_stats.create_stat_table(ownname=>'SCOTT',stattab=>'STATS');

/* Step 2: Export the table statistics into the staging table. Cascade results
in all index and column statistics associated with the specified table being
```

```
exported as well. */

exec
dbms_stats.export_table_stats(ownname=>'SCOTT',tabname=>'EMP',
partname=>NULL,stattab=>'STATS',statid=>NULL,cascade=>TRUE,statown=>'SCOTT');
exec
dbms_stats.export_table_stats(ownname=>'SCOTT',tabname=>'DEPT',
partname=>NULL,stattab=>'STATS',statid=>NULL,cascade=>TRUE,statown=>'SCOTT');

/* Step 3: Create analysis task */
3. exec :sts_task := DBMS_SQLPA.create_analysis_task(sqlset_name=>
'scott_test_sts',task_name=>'SPA_TASK', sqlset_owner=>'SCOTT');

/*Step 4: Execute the analysis task before masking */
exec DBMS_SQLPA.execute_analysis_task(task_name => 'SPA_TASK',
execution_type=> 'explain plan', execution_name  => 'pre-mask_SPA_TASK');
```

This example shows a post-masking script for preserving statistics.

```
*Step 1: Import the statistics from the staging table to the dictionary tables*/

exec
dbms_stats.import_table_stats(ownname=>'SCOTT',tabname=>'EMP',
partname=>NULL,stattab=>'STATS',statid=>NULL,cascade=>TRUE,statown=>'SCOTT');
exec
dbms_stats.import_table_stats(ownname=>'SCOTT',tabname=>'DEPT',
partname=>NULL,stattab=>'STATS',statid=>NULL,cascade=>TRUE,statown=>'SCOTT');

/* Step 2: Drop the staging table */

exec dbms_stats.drop_stat_table(ownname=>'SCOTT',stattab=>'STATS');

/*Step 3: Execute the analysis task before masking */
exec DBMS_SQLPA.execute_analysis_task(task_name=>'SPA_TASK',
execution_type=>'explain plan', execution_name=>'post-mask_SPA_TASK');

/*Step 4: Execute the comparison task */
exec DBMS_SQLPA.execute_analysis_task(task_name =>'SPA_TASK',
execution_type=>'compare', execution_name=>'compare-mask_SPA_TASK');
```

> **See Also:**
>
> "Masking a Test System to Evaluate Performance" for a procedure that explains how to specify the location of these scripts when scheduling a data masking job

4.6.4 Scheduling a Script Generation Job

To schedule a script generation job:

1. Select the masking definition to generate a script for, then click **Generate Script**.
2. Change the default job name to something meaningful, if desired, and provide an optional job description.
3. Select a reference database from the drop-down list.
4. Select a script generation option:

- **Mask In-Database**–to replace sensitive data in-place with masked data on a specified database (usually copied from production). Use this option only in non-production environments. This differs from the **Actions** menu option **Clone Database**, which clones the database and then masks the data.
- **Mask In-Export**–to export masked data from the specified source database (usually production) using Oracle Data Pump. This option is safe to run in a production environment as it does not modify customer data. Note, however, that this option creates temporary tables that get dropped when the masking operation completes.

Note that you can choose both options; that is, a script to mask the database directly and a script to create a masked dump.

5. Specify credentials to log in to the reference database.
6. Specify to start the job immediately or at a later specified date and time, then click **Submit.**

   A message confirms that the job has been scheduled. Refresh the page to see the job results.

4.6.5 Scheduling a Data Masking Job

To set up the data masking job and schedule its execution:

1. Select the masking definition for which a script has been generated, then click **Schedule Job**.
2. Change the default job name if desired and enter an optional job description.
3. Select a database from the drop-down menu and indicate your preference:
   - **Mask In-Database**–to replace sensitive data in-place with masked data on a specified database (usually copied from production). Use this option only in non-production environments. This differs from the **Actions** menu option **Clone Database**, which clones the database and then masks the data.

     Note:

     You must enable the check box indicating that the selected target is not a production database in order to proceed.

   - **Mask In-Export**–to export masked data from the specified source database (usually production) using Oracle Data Pump. This option is safe to run in a production environment as it does not modify customer data. Note, however, that this option creates temporary tables that get dropped when the masking operation completes.

   Your selections affect the check box text that appears below the radio buttons as well as other regions on the page.
4. Proceed according to your selections in Step 3:
   - **Data Mask Options**–Provide the requisite information as follows:
     – After script generation completes, the data masking script is stored in the Enterprise Manager repository. By default, the data masking job retrieves the script from the repository and copies it to the `$ORACLE_HOME/dbs`

directory on the database host, using a generated file name. The Script File Location and Name fields enable you to override the default location and generated file name.

- Workloads–Select options to mask SQL tuning sets and capture files, as appropriate. Browse to the file location where you want to capture the files.
- Detect SQL Plan Changes Due to Masking–Run the SQL Performance Analyzer to assess the impact of masking. Provide a task name and browse to the corresponding SQL tuning set.

- **Data Export Options**–Provide the requisite information as follows:
  - Specify a directory where to save the mask dump. The drop-down list consists of directory objects that you can access. Alternatively, you can select a custom directory path. Click the check box if you want to speed the process by using an external directory. Recommended default: `DATA_FILE_DIR`.
  - Specify appropriate values if you want to override the defaults: enter a name for the export file; specify a maximum file size in megabytes; specify the maximum number of threads of active execution operating on behalf of the export job. This enables you to consider trade-offs between resource consumption and elapsed time.
  - Specify whether to export only the masked data or the entire database along with the masked data.
  - Select whether to enable dump file compression and encryption. Enter and confirm an encryption password, if appropriate. Log file generation is selected by default.

5. Specify credentials to log in to the database host.
6. Specify credentials to log in to the reference database.
7. Choose one of the following options to create temporary objects:
   - **Optimize storage using default settings** — creates all the mapping tables in the user’s default tablespace. While creating the mapping tables, basic compression is used to compress the mapping tables.

     Mapping tables are used to transform one value to another in an Oracle database.
   - **Optimize storage using recommended settings** — creates a compressed tablespace. All the intermittent objects (mapping tables as well as the copy of original tables being masked) will be created in this tablespace.

> **Note:**
>
> If the `DB_CREATE_FILE_DEST` is set, the datafile required for the new tablespace will be managed by Oracle. Otherwise, the datafile for the new tablespace will be created in the directory from where the masking script is being run.
>
> This tablespace will be dropped when the masking job is completed. However, the tablespace is not dropped if the user chooses not to drop the mapping tables while creating the script.

- **Optimize storage using custom settings**
  - **Create temporary objects in a custom tablespace** — creates temporary objects in the tablespace specified in step 8. Basic compression will be used to compress these temporary objects.
  - **Create temporary objects and copy of table being masked in a custom tablespace** — creates the temporary objects that are compressed, and creates a copy of the original table that is being masked while moving it to the tablespace specified at Step 8.

> **Note:**
>
> This option might increase the masking time considerably as it makes a copy of the original table and moves it to the custom tablespace.
>
> For accurate storage estimates, ensure that you use `dbms_stats.gather_table_stats` to gather the stats of all the tables participating in the masking process.

8. If you chose custom settings in Step 7, select a custom tablespace where the objects must be created.
9. Specify to start the job immediately or at a later specified date and time, then click **Submit.**

   A message confirms that the job has been scheduled. Refresh the page to see the job results.

4.6.6 Estimating Space Requirements for Masking Operations

Here are some guidelines for estimating space requirements for masking operations. These estimates are based on a projected largest table size of 500GB. In making masking space estimates, assume a "worst-case scenario."

- For in-place masking:
  - 2 * 500GB for the mapping table (the mapping table stores both the original and the masked columns. Worse case is every column is to be masked).
  - 2 * 500GB to accommodate both the original and the masked tables (both exist in the database at some point during the masking process).
  - 2 * 500GB for temporary tablespace (needed for hash joins, sorts, and so forth).

  Total space required for the worst case: 3TB.
- For at-source masking:
  - 2 * 500GB for the mapping table (as for in-place masking).
  - 2 * 500GB for temporary tablespace (as for in-place masking).
  - Sufficient file system space to accommodate the dump file.

  Total space required for the worst case: 2TB plus the necessary file system space.

In either case, Oracle recommends that you set the temp and undo tablespaces to auto extend.

You can specify a tablespace for mapping tables during script generation. If you do not specify a tablespace, the tables are created in the tablespace of the executing user. Note that space estimations are provided during script generation, with resource warnings as appropriate. There are some situations, for example when using the shuffle format, that do not require a mapping table. In these cases, updates to the original table happen in-line.

4.6.7 Adding Dependent Columns

Dependent columns are defined by adding them to the Application Data Model. The following prerequisites apply for the column to be defined as dependent:

- A valid dependent column should not already be included for masking.
- The column should not be a foreign key column or referenced by a foreign key column.
- The column data should conform to the data in the parent column.

If the column does not meet these criteria, an "Invalid Dependent Columns" message appears when you attempt to add the dependent column.

4.6.8 Masking Dependent Columns for Packaged Applications

The following procedure explains how to mask data across columns for packaged applications in which the relationships are not defined in the data dictionary.

To mask dependent columns for packaged applications:

1. Go to Data Discovery and Modeling and create a new Application Data Model (ADM) using metadata collection for your packaged application suite.

   When metadata collection is complete, edit the newly created ADM.

2. Manually add a referential relationship:

   a. From the Referential Relationships tab, open the **Actions** menu, then select **Add Referential Relationship**.

      The Add Referential Relationship pop-up window appears.

   b. Select the requisite Parent Key and Dependent Key information.

   c. In the Columns Name list, select a dependent key column to associate with a parent key column.

   d. Click **OK** to add the referential relationship to the ADM.

      The new dependent column now appears in the referential relationships list.

3. Perform sensitive column discovery.

   When sensitive column discovery is complete, review the columns found by the discovery job and mark them sensitive or not sensitive as needed.

   When marked as sensitive, any discovery sensitive column also marks its parent and the other child columns of the parent as sensitive. Consequently, it is advisable to first create the ADM with all relationships. ADM by default, or after running drivers, may not contain denormalized relationships. You need to manually add these.

For more information about sensitive column discovery, see Performing Sensitive Data Discovery.

4. Go to Data Masking and create a new masking definition.
5. Select the newly created ADM and click **Add**, then **Search** to view this ADM's sensitive columns.
6. Select columns based on your search results, then import formats for the selected columns.

   Enterprise Manager displays formats that conform to the privacy attributes.
7. Select the format and generate the script.
8. Execute the masking script.

   Enterprise Manager executes the generated script on the target database and masks all of your specified columns.

4.6.9 Importing a Data Masking Template

You can import and re-use a previously exported data masking definition saved as an XML file to the current Enterprise Manager repository. You also can import an Oracle-supplied data masking definition from the Software Library.

Importing a Previously Exported Masking Definition

Note the following advisory information:

- The XML file format must be compliant with the masking definition XML format.
- Verify that the name of the masking definition to be imported does not already exist in the repository, and the source database name identifies a valid Enterprise Manager target.
- Verify that the value in the XML file to be imported refers to a valid database target.

1. From the Data Masking Definitions page, click **Import**.

   The Import Masking Definition page appears.
2. Specify a name for the masking definition and select the ADM to associate with the template. The Reference Database is automatically provided.
3. Browse for the XML file, or specify the name of the XML file, then click **Continue**.

   The Data Masking Definitions Page reappears and displays the imported definition in the table list for subsequent viewing and masking.

Importing a Data Masking Template from the Software Library

The Self Update feature ensures that the latest Oracle-supplied data masking templates are available in the Software Library. You can also check for updates. Go to the Self Update page and check for Test Data Management updates. If present, download and apply them so that they are available in the Software Library.

1. On the Data Masking Definitions page, click **Import from Software Library**.

   The Import Masking Definition page appears.
2. Select a masking template in the Software Library list.

3. Specify a name for the masking definition and select the ADM to associate with the template. The Reference Database is automatically provided.
4. Click **Continue**.

   The Data Masking Definitions Page reappears and displays the imported definition in the table list for subsequent viewing and masking.

You can also export a data masking definition from the Software Library.

1. On the Data Masking Definitions page, click **Export from Software Library**.
2. Select a masking template in the Software Library list.
3. Click **Export**.

   Save the template file for import into a different repository.

4.6.10 Cloning the Production Database

When you clone and mask the database, a copy of the masking script is saved in the Enterprise Manager repository and then retrieved and executed after the clone process completes. Therefore, it is important to regenerate the script after any schema changes or modifications to the production database.

To clone and optionally mask the masking definition's target database:

1. From the Data Masking Definitions page, select the masking definition you want to clone, select **Clone Database** from the Actions list, then click **Go**.

   The Clone Database: Source Type page appears.

   The Clone Database wizard appears, where you can create a test system to run the mask.
2. Specify the type of source database backup to be used for the cloning operation, then click **Continue**.
3. Proceed through the wizard steps as you ordinarily would to clone a database. For assistance, refer to the online help for each step.
4. In the Database Configuration step of the wizard, add a masking definition, then select the Run SQL Performance Analyzer option as well as other options as desired or necessary.
5. Schedule and then run the clone job.

4.7 Masking a Test System to Evaluate Performance

After you have created a data masking definition, you may want to use it to analyze the performance impact from masking on a test system. The procedures in the following sections explain the process for this task for masking only, or cloning and masking.

4.7.1 Using Only Masking for Evaluation

To use only masking to evaluate performance:

1. From the Data Masking Definitions page, select the masking definition to be analyzed, then click **Schedule Job**.

The Schedule Data Masking Job page appears.

2. At the top of the page, provide the requisite information.

   The script file location pertains to the masking script, which also contains the pre- and post-masking scripts you created in "Pre- and Post-mask Scripts".

3. In the Encryption Seed section, provide a text string that you want to use for encryption.

   This section only appears for masking definitions that use the Substitute or Encrypt formats. The seed is an encryption key used by the encryption/hash-based substitution APIs, and makes masking more deterministic instead of being random.

4. In the Workloads section:

   a. Select the **Mask SQL Tuning Sets** option, if desired.

      If you use a SQL Tuning Set that has sensitive data to evaluate performance, it is beneficial to mask it for security, consistency of data with the database, and to generate correct evaluation results.

   b. Select the **Capture Files** option, if desired, then select a capture directory.

      When you select this option, the contents of the directory is masked. The capture file masking is executed consistently with the database.

5. In the Detect SQL Plan Changes Due to Masking section, leave the Run SQL Performance Analyzer option unchecked.

   You do not need to enable this option because the pre- and post-masking scripts you created, referenced in step 2, already execute the analyzer.

6. Provide credentials and scheduling information, then click **Submit**.

   The Data Masking Definitions page reappears, and a message appears stating that the Data Masking job has been submitted successfully.

   During masking of any database, the AWR bind variable data is purged to protect sensitive bind variables from leaking to a test system.

7. When the job completes successfully, click the link in the SQL Performance Analyzer Task column to view the executed analysis tasks and Trial Comparison Report, which shows any changes in plans, timing, and so forth.

4.7.2 Using Cloning and Masking for Evaluation

Using both cloning and masking to evaluate performance is very similar to the procedure described in the previous section, except that you specify the options from the Clone Database wizard, rather than from the Schedule Data Masking Job page.

To use both cloning and masking to evaluate performance:

1. From the Data Masking Definitions page, select the masking definition you want to clone, select **Clone Database** from the Actions list, then click **Go**.

   The Clone Database: Source Type page appears.

   The Clone Database wizard appears, where you can create a test system to run the mask.

2. Specify the type of source database backup to be used for the cloning operation, then click **Continue**.

3. Proceed through the wizard steps as you ordinarily would to clone a database. For assistance, refer to the online help for each step.
4. In the Database Configuration step of the wizard, add a masking definition, then select the Run SQL Performance Analyzer option as well as other options as desired or necessary.

> **Note:**
>
> The format of the Database Configuration step appears different from the Schedule Data Masking Job page discussed in "Using Only Masking for Evaluation", but select options as you would for the Schedule Data Masking Job page.

5. Continue with the wizard steps to complete and submit the cloning and masking job.

4.8 Upgrade Considerations

Upgrading data masking definitions from 10 or 11 Grid Control to 12*c* Cloud Control assumes that you have completed the following tasks:

- Upgraded Enterprise Manager to 12*c*
- Downloaded the latest database plug-in using Self Update and deployed the plug-in to OMS and Management Agent

Completing these tasks automatically upgrades the masking definitions and creates for each a shell Application Data Model (ADM) that becomes populated with the sensitive columns and their dependent column information from the legacy mask definition. The ADM, and hence data masking, then remains in an unverified state, because it is missing the dictionary relationships.

Proceed as follows to complete the masking definition upgrade:

1. From the **Enterprise** menu, select **Quality Management**, then select **Data Discovery and Modeling**.
2. For each shell ADM (verification status is Needs Upgrade), do the following:
   a. Select the ADM in the table.
   b. From the Actions menu, select **Upgrade and Verify**.
   c. Schedule and submit the job.

      When the job completes, verification status should be Valid.
3. From the **Enterprise** menu, select **Quality Management**, then select **Data Masking Definitions**.
4. For each upgraded masking definition, do the following:
   a. Open the masking definition for editing.
   b. In **Advanced Options**, select the "Recompile invalid dependent objects after masking" option, with Parallel and Default settings.
   c. Click **OK** to save your changes.

5. Next, schedule a script generation job for each upgraded masking definition.

You can now resume masking with the upgraded data masking definitions.

> **See Also:**
>
> "Adding Dependent Columns" for information on dependent columns

Consider these other points regarding upgrades:

- You can combine multiple upgraded ADMs by exporting an ADM and performing an Import Content into another ADM.
- An upgraded ADM uses the same semantics as for upgrading a legacy mask definition (discussed above), in that you would need to perform a validation.
- An 11.1 Grid Control E-Business Suite (EBS) masking definition based on an EBS masking template shipped from Oracle is treated as a custom application after the upgrade. You can always use the approach discussed in the first bulleted item above to move into a newly created EBS ADM with all of the metadata in place. However, this is not required.

4.9 Using the Shuffle Format

A shuffle format is available that does not preserve data distribution when the column values are not unique and also when it is conditionally masked. For example, consider the Original Table (Table 4-1) that shows two columns: EmpName and Salary. The Salary column has three distinct values: 10, 90, and 20.

Table 4-1 Original Table (Non-preservation)

EmpName	Salary
A	10
B	90
C	10
D	10
E	90
F	20

If you mask the Salary column with this format, each of the original values is replaced with one of the values from this set. Assume that the shuffle format replaces 10 with 20, 90 with 10, and 20 with 90 (Table 4-2).

Table 4-2 Mapping Table (Non-preservation)

EmpName	Salary
10	20
90	10
20	90

The result is a shuffled Salary column as shown in the Masked Table (Table 4-3), but the data distribution is changed. While the value 10 occurs three times in the Salary column of the Original Table, it occurs only twice in the Masked Table.

Table 4-3 Masked Table (Non-preservation)

EmpName	Salary
A	20
B	10
C	20
D	20
E	10
F	90

If the salary values had been unique, the format would have maintained data distribution.

4.10 Using Group Shuffle

Group shuffle enables you to perform a shuffle within discrete units, or groups, where there is a relationship among the members of the group. Consider the case of shuffling the salaries of employees. Table 4-4 illustrates the group shuffle mechanism, where employees are categorized as managers (M) or workers (W), and salaries are shuffled within job category.

Table 4-4 Group Shuffle Using Job Category

Employee	Job Category	Salary	Shuffled Salary
Alice	M	90	88
Bill	M	88	90
Carol	W	72	70
Denise	W	57	45
Eddie	W	70	57
Frank	W	45	72

4.11 Using Conditional Masking

To demonstrate how conditional masking can handle duplicate values, add to Table 4-4 another job category, assistant (A), where the employee in this category, George, earns the same as Frank. Assume the following conditions:

- If job category is M, replace salary with a random number between 1 and 10.
- If job category is W, set salary to a fixed number (01).
- Default is to preserve the existing value.

Applying these conditions results in the masked values shown in Table 4-5:

Table 4-5   Using Job Category for Group Shuffle

Employee	Job Category	Salary	Conditional Result
Alice	M	90	5
Bill	M	88	7
Carol	W	72	01
Denise	W	57	01
Eddie	W	70	01
Frank	W	45	01
George	A	45	45

Conditional masking works when there are duplicate values provided there are no dependent columns or foreign keys. If either of these is present, a "bleeding condition" results in the first of two duplicate values becoming the value of the second. So, in the example, George's salary is not preserved, but becomes 01.

4.12 Using Data Masking with LONG Columns

When data masking script generation completes, an impact report appears. If the masking definition has tables with columns of data type LONG, the following warning message is displayed in the impact report:

```
The table <table_name> has a LONG column. Data Masking uses "in-place" UPDATE
to mask tables with LONG columns. This will generate undo information and the
original data will be available in the undo tablespaces during the undo
retention period. You should purge undo information after masking the data.
Any orphan rows in this table will not be masked.
```

4.13 Debugging a Data Masking Job

The `reorganize.pl` script is used internally while a data masking job is being executed. This script contains some useful debugging messages such as `ORACLE_HOME`, `Protocol`, `Host`, `Port`, `Sid`, and `Connection String` that the masking job uses in order to connect to the target database, and run the script.

To print the debugging message to `user_perl.trc` file in `$EMSTATE/sysman/log/` , enable agent debug tracing by doing the following:

```
$EMSTATE/bin/emctl setproperty agent -name
"EMAGENT_PERL_TRACE_LEVEL" -value "DEBUG"
```

```
$EMSTATE/bin/emctl setproperty agent -name "userPerlTraceDir" -
value "$EMSTATE/sysman/log"
```

5

Data Subsetting

This chapter covers the Integrated Subset and Mask capability, where you perform data masking and subsetting in a single taskflow and outlines a number of scenarios to demonstrate the process. You must have the Oracle Data Masking and Subsetting Pack license to use data subsetting features.

> **Note:**
>
> Data subsetting is supported only in Oracle Database versions 10.1 and higher. The procedures in this chapter are applicable only to Oracle Enterprise Manager Cloud Control 12.1 and higher.

5.1 Creating a Data Subset Definition

The procedure described in this section enables you to create a subset database, after which you can perform other tasks, such as editing the properties of the subset definition or exporting a subset definition.

The interface also allows you to perform inline, or at the source, masking while creating the subset definition.

Before proceeding, ensure that you have the following privileges:

- `EM_ALL_OPERATOR` for Enterprise Manager Cloud Control users

> **Note:**
>
> The EM_ALL_OPERATOR privilege is not required, if you have the following privileges:
>
> **Target Privileges (applicable to all targets):**
>
> – Connect to any viewable target
> – Execute Command Anywhere
> – View Any Target
>
> **Resource Privileges:**
>
> – Job System
> – Named Credential
> – Oracle Data Masking and Subsetting resource privilege

- `SELECT_ANY_DICTIONARY` privilege for database users

- To perform an in-place delete operation, the DBA user must be granted `EXECUTE_ANY_TYPE` privilege.
- To subset tables that have Virtual Private Database (VPD) policies, the user performing subsetting must be granted `Exempt Access Policy` privilege.

To create a data subset definition:

1. From the Enterprise menu, select **Quality Management**, then **Data Subsetting Definitions**.
2. Open the **Actions** menu in the Data Subsetting Definitions page, then select **Create**, or just click the **Create** icon.
3. Define the data subset definition properties:

    a. Provide the requisite information in the General pop-up that appears, then click **Continue**.

    You can select any source database associated with the Application Data Model.

    If you are performing masking within the subset definition, you must select the same ADM and target used in creating the masking definition.

    b. Provide a job name, credentials, and specify a schedule in the Schedule Application Detail Collection pop-up that appears, then click **Submit**.

    If you want to use new credentials, choose the New Credentials option. Otherwise, choose the Preferred Credentials or Named Credentials option.

    The space estimate collection job runs, and then displays the Data Subset Definitions page. Your definition appears in the table, and the Most Recent Job Status column should indicate Scheduled, Running, or Succeeded, depending on the schedule option selected and time required to complete the job.

4. Select the definition within the table, open the **Actions** menu, then select **Edit**.

    The Database Login page appears.

5. Select either Named Credentials or New Credentials if you have not already set preferred credentials, then click **Login**.
6. In the Applications subpage of the Edit page, move applications from the Available list to the Selected list as follows:

    - If you intend only to mask the data (no subsetting), select all applications.
    - If you intend only to subset the data (no masking), select specific applications as appropriate.
    - If you intend both to subset and mask the data, the applications selected must include those that the masking definitions require.

    The names of application suites, applications, or application modules are maintained in the Application Data Model.

7. Click the **Object Rules** tab.

> **Note:**
>
> If you are masking only, set the Default Object Rows option to include all rows and skip to Step 13. The **Column Mask Rules** tab, **Rule Parameters** tab, and additional features on the **Object Rules** tab pertain specifically to subsetting.

You can add rules here to define the data to include in the subset.

8. Select **Actions**, then **Create** to display the Object Rule pop-up, or just click the **Create** icon.

   a. Select the application for which you want to provide a rule.

      Associate the rule with all objects, a specific object, or a specific type of object.

   b. If you select **All Objects**, in the Rows to Include section, select all rows or some rows by specifying a percentage portion of the rows.

   c. If you select **Specified** as the object type, the tables from the selected Application appear in the drop-down list.

      In the Rows to Include section, select all rows, or some rows by specifying a percentage portion of the rows. For finer granularity, you could specify a Where clause, such as where region_id=6.

      If the selected table is partitioned, click **Add/Remove Partitions** to choose the partitions and sub-partitions from which the objects must be included in the subset.

   d. In the Include Related Rows section, do one of the following:

      - Select **Ancestor and Descendant Objects**

        This rule impacts the parent and child columns, and ensures that referential integrity is maintained, and that child columns are also selected as part of the subset.

      - Select **Ancestor Objects Only**

        This option will be disabled if the global rule is set to “Include All Rows”, and will be enabled if it is set to “Include No Rows”. This rule only impacts the parent columns, and ensures that referential integrity is maintained.

        Select **Descendant Objects Only**

        This option will be disabled if the global rule is set to “Include No Rows”, and will be enabled if it is set to “Include All Rows”.

      In Oracle Data Masking and Subsetting release 13.2, users can now set the rule scope to “Include Related Rows” with global rule scope set to “Include All Rows” and vice versa, and all unrelated tables are processed using the global rule “Include All Rows” .

      If you disable the Include Related Rows check box, referential integrity might not be maintained. However, you can define additional rules on the related tables to restore the referential integrity. You can disable this check box whether or not you specify a Where clause.

   e. If you want to specify a Where clause, go to the next step. Otherwise, skip to Step 9.

   f. Provide a rule parameter, if desired, for the clause.

For instance, if you specify a particular value for an employee ID as employee_id=:emp_id, you could enter query values for the default of 100:

- Select the Rows Where button and enter employee_id=:emp_id.
- Click **OK** to save the rule and return to the Object Rules tab.

  If this is a new rule, a warning appears stating that "Rule parameters corresponding to the bind variables 'emp_id' should be created before generating subset."
- Select the table rule, click the **Rule Parameters** tab, then click **Create**.

  The Rule Parameter Properties pop-up appears.
- Enter emp_id for the Name and 100 for the Value.

> **Note:**
>
> The colon (:) preceding emp_id is only present in the Where clause, and not required when creating a new rule parameter.

- Click **OK** to save the properties, which now appear in the **Rule Parameters** tab.
- Skip to Step 10.

9. Click **OK** to save the rule and return to the **Object Rules** tab.

   The new rule is displayed in the list. The related objects are displayed in the table below. Related rows from the objects are included in the subset to provide referential integrity in the subset database.

10. In the Related Objects section of the **Object Rules** tab, you can manage the size of the subset by controlling the levels of ancestors and descendants within the subset. Notice that each node in the table has a check box. By default, all nodes are included in the subset, as indicated by the check mark. Deselect the check box to exclude a node from the subset. The deselection option is disabled for parent rows (the join columns to the right identify parent and child rows). In addition, you can make these other refinements to subset content:

    - Click **Allow Excluding Parent Objects**. This enables the check marks that were grayed out. You can now selectively exclude parent rows from the subset by deselecting the check box.
    - Select a node within the table and click **Add Descendants** to include related rows. In the dialog that opens, make appropriate selections and click **OK**.

    As you make these refinements, columns on the right reflect the effect on space estimates for the subset. The Related Objects section also denotes the processing order of the ancestor and descendant tables, including the detailed impact of including each object. When you are done with the refinements, go to the **Space Estimates** tab to see a finer granularity of the impact on the overall size of the subset.

11. In the Default Object Rows section of the **Object Rules** tab, choose whether you want to include or exclude the objects not affected by the defined rules in the subset.

    When you select the Include All Rows option, all of the rows for the object are selected as part of the subset.

This is a global rule and applies to the entire subset. You can only select the Include All Rows option when all of the rules have a scope of None. A scope of None is established when you uncheck the Include Related Rows option in the Object Rule pop-up.

> **Note:**
>
> For a subset definition that has column rules (see Step 12), be sure to use object rules to include the corresponding objects. You can use the Default Object Rows option to include all objects not affected by object rules, if required.

12. *Optional*: Click the **Column Mask Rules** tab to perform inline masking as part of the subset definition.

    a. Click **Create** and enter search criteria to filter on columns within the schema. These would typically be vertical columns such as CLOB AND BLOB columns.

    > **Note:**
    >
    > If you are using column mask rules instead of masking definitions (see Step 13), you can select no more than 10 columns in a given table. This restriction applies to the export method but not to the in-place delete method.

    Click **OK**.

    b. Select a row or rows in the column search results and click **Manage Masking Formats**.

    c. In the pop-up dialog, select a masking format and value to apply to the columns. For multiselection, the same format must be appropriate for all columns. If you select multiple columns, ensure that the column rule format you choose is applicable to the selected columns. Use the columns (flags) **not null** and **unique** to enforce compliance.

    Click **OK** to apply the masking format to the columns.

13. *Optional*: Click the **Data Masking Definitions** tab to include masking definitions as part of the subsetting operation or to perform at the source data masking only.

    a. Click **Add**.

    b. In the pop-up dialog, enter search criteria to retrieve appropriate definitions. Be sure to select the desired radio button (All or Any). All formats except compound masking are supported for inline masking.

> **Note:**
>
> No single table within a masking definition can have more than 10 masked columns if you are using the export method. The restriction does not apply to the in-place delete method.

Click **OK**.

The search results appear in the data masking table.

14. Click the **Space Estimates** tab.

    - Note the value in the Estimated Subset Size MB column. The space estimates depend on optimizer statistics, and the actual distribution of data can only be calculated if histogram statistics are present.
    - Whenever you add new rules, recheck the space estimates for updated values.
    - Data in the Space Estimates subpage is sorted with the largest applications appearing at the top.

    > **Note:**
    >
    > Space estimates are accurate only if “dbms_stats.gather_table_stats” is used. Also, space estimates do not reflect the effect of data masking, if used.

    If you provide a Where clause and subsequent rule parameter properties, the Space Estimates subpage is updated with the value contained in the **Rule Parameters** tab.

15. *Optional*: click the **Pre/Post Subset Script** tab.

    - You can specify a pre-subset script to run on the subset database before you select subset data.
    - You can specify a post-subset script to run on the subset database after you assemble the subset data.
    - Either script type runs on the source database.

    > **Note:**
    >
    > Ensure that the PL/SQL block defined the pre-susbset or post-subset script includes a “/” at the end.

16. Click **Return**.

    The definition is complete and displayed in the Data Subsetting Definitions table.

You can now proceed with script generation. Alternatively, you may want to save the script for future use. In either case, you must decide whether to export data to a dump file or delete data from a target database.

> **Tip:**
>
> If you have a very large database of 4 terabytes, for instance, and you want to export a small percentage of the rows, such as 10%, it is more advantageous to use the export method. Using the in-place delete method would require 3.6 terabytes of data, which would not perform as quickly as the export method.
>
> The in-place delete method is recommended when the amount of data being deleted is a small percentage of the overall data size.
>
> There is an EMCLI verb if you want to perform an in-place delete remotely or script it.

5.1.1 Generating a Subset Script

To prepare and submit a job to generate a subset script:

1. Select the definition within the table, open the **Actions** menu, then select **Generate Subset**.

   The Generate Subset pop-up appears.

2. Select a target database that is either the same target database you used to create the subset model, or similar to this database regarding the table schema and objects.

3. Decide if you want to create a subset by writing subset data to export files, or by deleting data from a target database.

   > **Note:**
   >
   > Choosing to delete data creates an in-place subset by removing or deleting unwanted data from a cloned copy of the production database, rather than a production database. Only data satisfying the rules are retained. You should never use this option on a production database.

   Select either Named Credentials or New Credentials if you have not already set preferred credentials.

   If you have defined any parameters from the **Rule Parameters** tab, they appear in the table at the bottom. You can change a parameter value by clicking on the associated field in the Value column.

4. Click **Continue** to access the Parameters pop-up. The contents of the pop-up depend on whether you chose the export or delete option in the previous step.

   For **Writing Subset Data to Export Files**, provide the requisite information, then click **Continue** to schedule the job.

   - Specify a subset directory where to save the export dump. The drop-down list consists of directory objects for which you have access. Alternatively, you can select a custom directory path. Click the check box if you want to speed the process by using an external directory. Recommended default: `DATA_PUMP_DIR`.

- Specify appropriate values if you want to override the defaults: enter a name for the export file; specify a maximum file size in megabytes; specify the maximum number of threads of active execution operating on behalf of the export job. This enables you to consider trade-offs between resource consumption and elapsed time.
- Specify whether you want to export only the subsetted data or the entire database along with the subsetted data.
- Select whether to enable dump file compression and encryption. Enter and confirm an encryption password, if appropriate. Log file generation is selected by default.

For **Deleting Data From a Target Database**, provide the requisite information, then click **Continue** to schedule the job.

- Specify a subset directory where to save the subset scripts. The drop-down list consists of directory objects for which you have access. Alternatively, you can select a custom directory path. Recommended default: `DATA_FILE_DIR`.
- You must enable the check box indicating that the selected target is not a production database in order to proceed.

5. Click **Continue** to review the estimated storage requirement for generating the subset script.

> **Note:**
>
> If the required storage for generating the susbset script is much more than the available space, allocate the necessary space to ensure that you have enough space to proceed with generating the subset script.

6. Specify a name for the subset job, and provide a description. Schedule the job to run immediately or at a later point of time, then click **Submit**.

   The Data Subset Definitions page reappears, and the Most Recent Job Status column shows that the subset job is running, and subsequently that it has succeeded.

After performing this procedure, you can now create a subset database with the generated export files at any time.

5.1.2 Saving a Subset Script

To prepare and submit a job to save a subset script:

1. Select the definition within the table, open the **Actions** menu, then select **Save Subset Script**. The Subset Mode pop-up appears.
2. Select a target database that is either the same target database you used to create the subset model, or similar to this database regarding the table schema and objects.
3. Decide if you want to create a subset by writing subset data to export files, or by deleting data from a target database.

   Choosing to delete data creates an in-place subset by removing/deleting unwanted data from a cloned copy of the production database, rather than a production database. Only data satisfying the rules are retained. You should never use this option on a production database.

Select either Named Credentials or New Credentials if you have not already set preferred credentials.

If you have defined any parameters from the **Rule Parameters** tab, they appear in the table at the bottom. You can change a parameter value by clicking on the associated field in the Value column.

4. Click **Continue** to access the Parameters pop-up. The contents of the pop-up depend on whether you chose the export or delete option in the previous step.

   For **Writing Subset Data to Export Files**, provide the requisite information, then click **Continue** to schedule the job.

   - Specify a subset directory where to save the export dump. The drop-down list consists of directory objects for which you have access. Alternatively, you can select a custom directory path. Click the check box if you want to speed the process by using an external directory. Recommended default: `DATA_PUMP_DIR`.
   - Specify appropriate values if you want to override the defaults: enter a name for the export file; specify a maximum file size in megabytes; specify the maximum number of threads of active execution operating on behalf of the export job. This enables you to consider trade-offs between resource consumption and elapsed time.
   - Specify whether to export only the subsetted data or the entire database along with the subsetted data.
   - Select whether to enable dump file compression and encryption. Enter and confirm an encryption password, if appropriate. Log file generation is selected by default.

   For **Deleting Data From a Target Database**, provide the requisite information, then click **Continue** to schedule the job.

   - Specify a subset directory where to save the subset scripts. The drop-down list consists of directory objects for which you have access. Alternatively, you can select a custom directory path. Recommended default: `DATA_FILE_DIR`.
   - You must enable the check box indicating that the selected target is not a production database in order to proceed.

5. Click **Continue**. A progress indicator tracks script generation. When complete, the Files table lists the results of script generation.
6. Click **Download**. In the File Download pop-up that appears, click **Save**.
7. In the Save As pop-up that appears, navigate to a file location and click **Save**.

   The file containing the scripts (SubsetBundle.zip) now appears at the specified location on your desktop.

To run the saved script at a later time:

1. Port the ZIP file to the target database and extract it to a directory on which you have the requisite privileges.
2. Change directory to where you extracted the files.
3. Execute the following script from the SQL command line:

   ```
   subset_exec.sql
   ```

Note that if you want to change generated parameter settings, you can do so by editing the following file in a text editor prior to executing the script:

```
subset_exec_params.lst
```

5.2 Importing and Exporting Subset Templates and Dumps

A subset template is an XML file that contains the details of the subset, consisting of the application, subset rules, rule parameters, and pre-scripts or post-scripts. When you create a subset definition and specify that you want to write subset data to export files, the export files become a template that you can subsequently import for reuse. You would import the template to perform subset operations on a different database.

Typically, the workflow is that you would first import a previously exported ADM template, which is another XML file, while creating an ADM. You would then import the related subset template while creating a data subset definition. You could alternatively select an existing ADM (skipping the import ADM flow) while importing the subset template.

> **Note:**
>
> There are EMCLI verbs to export and import subset definitions and subset dumps if you want to perform these operations remotely or script them.

5.2.1 Importing a Subset Definition

There are three methods of import:

- Importing a Subset Definition XML File From Your Desktop
- Importing a Subset Dump
- Importing a Subset Definition XML File From the Software Library

5.2.1.1 Importing a Subset Definition XML File From Your Desktop

1. From the **Actions** menu, select **Import**, then select **File from Desktop**.
2. In the pop-up that appears:
   a. Specify a name for the subset definition.
   b. The ADM on which the subset is based.
   c. A source database.
   d. The location on your desktop from which you want to import the subset definition.
   e. Click **Continue**.
3. In the pop-up that appears:
   a. Enter a descriptive job name (or accept the default).
   b. Provide credentials.

c. Schedule the job.

d. Click **Submit**.

After the job runs successfully, the imported subset appears in the list of subsets in the table on the Data Subset Definitions page.

5.2.1.2 Importing a Subset Dump

1. From the **Actions** menu, select **Import**, then select **Subset Dump**.
2. In the pop-up that appears:
   a. Select a target database.

   b. Provide database and host credentials, then click **Login**.

   c. Specify the location of the dump file, which can be in a selected directory on the target database or at a custom path on the target. Note that if the original export action used an external location for the dump files, the location must be specified as well.

   d. Click **Continue**.
3. In the pop-up that appears:
   a. Select whether to import both metadata and data, or data only. If data only, indicate if you want to truncate, that is, overlay existing data or append to existing data.

   b. Enable OID transform during type or object creation to create a new OID.

   Creating a new OID will break the REF columns that point to the object.

   c. Specify whether to use the default tablespace on the target database or remap the tablespaces from the existing tablespace to another tablespace.

   d. Perform tablespace remapping as necessary.

   e. Specify whether you want to retain the existing tables in the destination schema or drop the existing tables in the destination schema.

   f. Perform schema remapping as necessary.

   g. Select log file options.

   h. Click **Continue**.
4. In the pop-up that appears:
   a. Enter a descriptive job name (or accept the default).

   b. Schedule the job.

   c. Click **Submit**.

The job reads the dump files and loads the data into the selected target database.

5.2.1.3 Importing a Subset Definition XML File From the Software Library

1. From the **Actions** menu, select **Import**, then select **File from Software Library**.
2. In the Import Data Subset Definition from Software Library pop-up that appears:
   a. Selected the desired subset definition XML file on the left.

   b. Provide the ADM properties on the right.
   c. Click **Continue**.
3. In the pop-up that appears:
   a. Enter a descriptive job name (or accept the default).
   b. Provide credentials.
   c. Schedule the job.
   d. Click **Submit**.

After the job runs successfully, the imported subset appears in the list of subsets in the table on the Data Subset Definitions page.

5.2.2 Exporting a Subset Definition

There are two methods of export:

- Exporting a Subset Definition as an XML File to Your Desktop
- Exporting a Subset Definition From the Software Library

5.2.2.1 Exporting a Subset Definition as an XML File to Your Desktop

1. From the Data Subset Definitions page, select the subset definition you want to export.
2. From the **Actions** menu, select **Export**, then select **Selected Subset Definition**.
3. In the File Download pop-up that appears, click **Save**.
4. In the Save As pop-up that appears, navigate to a file location and click **Save**.

   The system converts the subset definition into an XML file that now appears at the specified location on your desktop.

5.2.2.2 Exporting a Subset Definition From the Software Library

1. From the **Actions** menu, select **Export**, then select **File from Software Library**.
2. In the Export Subset Definition from Software Library pop-up that appears, select the desired subset definition and click **Export**.
3. In the File Download pop-up that appears, click **Save**.
4. In the Save As pop-up that appears, navigate to a file location and click **Save**.

   The system converts the subset definition into an XML file that now appears at the specified location on your desktop.

After the job runs successfully, the subset template appears in the list of subsets in the table on the Data Subset Definitions page.

5.3 Creating a Subset Version of a Target Database

After a subset is defined, analyzed, and validated, you can execute the subset operation to create a subset version of the source data.

The procedure assumes the following prerequisites:

- A subset definition already exists that contains the rules needed to subset the database.
- You have the requisite privileges to extract the data from the source and create the subset version in a target database. Depending on the subset technique, different levels of file or database privileges may be created. The required privileges include:
  - **Target Privileges (applicable to all targets):**
    - Connect to any viewable target
    - Execute Command Anywhere
    - View Any Target
  - **Resource Privileges:**
    - Job System
    - Named Credential
    - Oracle Data Masking and Subsetting resource privilege

> **Note:**
>
> The `EM_ALL_OPERATOR` privilege for Enterprise Manager Cloud Control users includes all of the above privileges.

  - `SELECT_CATALOG_ROLE` for database users
  - `SELECT ANY DICTIONARY` privilege for database users
  - DBA privileges on a database for target database users
- Additionally, to perform an in-place delete operation, the DBA user must be granted the `EXECUTE_ANY_TYPE` privilege

To create a subset version of a target database:

1. Create a subset operation by selecting a subset definition and associating it with a source database.

   Enterprise Manager validates the subset definition against the source database and flags schema differences. Note that this association may be different from the original association that an application developer may have created.

2. Edit the definition to remap the defined schema to a test schema.

   You are prompted to connect to a database, whereupon the database is associated with the subset definition. This also enables you to remap the vendor-provided schema names to actual schema names in the database.

3. Select one of the various subset creation techniques:
   - Data Pump dump file followed by a Data Pump import
   - In-place delete, in which rows in the specified database not matching the rule conditions are deleted
   - In-transit subset creation or refresh

   Enterprise Manager generates the appropriate response file (that is, SQL script, Data Pump script, or OS script), checks the target system for appropriate privileges to be able proceed with the operation, and estimates the size of the target.
4. After reviewing the analysis, submit the subset process.

   Enterprise Manager executes the subset process and summarizes the results of the execution.

5.4 Synchronizing a Subset Definition with an Application Data Model

Changes to an ADM, adding referential relationships or deleting tables, for example, can render a subset definition stale. The Subset Definitions page clearly indicates this condition with a lock icon next to the subset name and an invalid status. Also, most menu items on the **Actions** menu are disabled. To revert the status to valid and unlock the subset definition, you have to synchronize the definition with its associated ADM.

1. On the Subset Definitions page, select the locked subset definition.
2. From the **Actions** menu, select **Synchronize**.
3. Complete the job submission dialog, then click **Submit**.

   When the job completes, the subset definition is unlocked and available for use.

5.5 Granting Privileges on a Subset Definition

You can grant privileges on a subset definition that you create so that others can have access. To do so, you must be an Enterprise Manager Administrator with at least Designer privileges on the subset definition.

1. From the **Enterprise** menu, select **Quality Management**, then select **Data Subset Definitions**.
2. Select the subset definition to which you want to grant privileges.
3. From the **Actions** menu, select **Grant**, then select as follows:
   - **Operator**–to grant Operator privileges on the subset definition to selected roles or administrators, which means the grantees can view and copy but not edit the definition.
   - **Designer**–to grant Designer privileges on the subset definition to selected roles or administrators, which means the grantees can view and edit the definition.
4. In the dialog that opens, select the type (administrator or role, or both). Search by name, if desired. Make your selections and click **Select**.

Those selected now have privileges on the subset definition.

5. Use the **Revoke** action if you want to deny privileges previously granted.

See "Oracle Data Masking and Subsetting Access Rights" for more information on privileges within the test data management area.

5.6 About Inline Masking and Subsetting

You can reduce the size of the database simultaneous with masking sensitive data. This serves the dual purpose of obscuring exported production data while greatly reducing hardware costs related to storing large masked production databases for testing.

> **Note:**
>
> Inline masking is available only with Oracle Database 11*g* and higher releases.

The benefits of integrating data masking with subsetting include the following:

- Prepare the test system in a single flow
- Avoid the necessity of maintaining large-size masked databases for test purposes
- Exported data in the form of a dump file can be imported into multiple databases without exposing sensitive data
- Subsetting is enhanced by ability to discard columns containing chunks of large data

You can select one or more data masking definitions during subset creation. The masking definitions must be based on the same ADM as the current subset definition. At the same time, you can significantly reduce the subset size by defining column rules to set CLOB and BLOB columns to null (or another supported format such as Fixed String, Fixed Number).

You generate a subset in two ways:

- Export Dump–if masking definitions are part of the subset model, mapping tables are created during generation, and the resulting dump contains masked values
- In-Place Delete–subsetting is performed on a cloned copy of the production database; if data masking is part of the subset model, pregenerated masking scripts are executed on the target sequentially

Advantages of inline masking include the following:

- Sensitive data never leaves the production environment and thus is not exposed (Export Dump option).
- There is no need to temporarily store data in a staging area.
- Exported data can subsequently be imported into multiple environments.
- You can define table rules to export only a subset of data, and can further trim the volume by using column rules to eliminate large vertical columns.

- You can mask the same data in different ways and import into different test databases.
- You can use the provisioning framework to create multiple copies of trimmed down, referentially intact databases containing no sensitive data (in-place delete), or import a dump file into multiple databases (export dump).

The section "Creating a Data Subset Definition" includes instructions for combining data subsetting and data masking within the process of creating a subset definition. See Data Masking,for information on data masking and creating a data masking definition.

5.7 Inline Masking and Subsetting Scenarios

The scenarios described below assume that an Application Data Model (ADM) exists for a production (or test) database in which sensitive column details are captured. The steps outlined are at a high level. See "Masking with an Application Data Model and Workloads" for details on creating a masking definition; see "Creating a Data Subset Definition" for details on creating and editing a subset definition.

5.7.1 Masking and Exporting Production Data

As the Security Administrator, you want to create copies of the production database by exporting the data with masked values; that is, the export dump will have only masked values and no sensitive data.

1. Create a masking definition. Implies the following:
   a. Select an appropriate ADM.
   b. Search and select sensitive columns (includes dependent columns and recommended masking formats).
   c. Review suggested formats and edit as necessary.
   d. Save the results.
2. Create a subset definition. Implies the following:
   a. Select an appropriate ADM.
   b. Submit the create subset job.
3. Edit the subsetting definition.

   On the **Data Masking** tab, search for and select masking definitions. System validation checks for overlapping columns that use multiple masking definitions.
4. Generate the subset using the Export option.

Summarizing the outcome:

- Generates and executes a script to create a mapping table and a mapping function. Also creates a table to map the column(s) to the respective mapping function.
- Copies subsetting and masking scripts to the target database.
- Generates an export dump of production data, replacing sensitive data with masked values using the mapping function.

5.7.2 Masking and Deleting Operations on a Test Database

As the Security Administrator, you want to create a usable test database by masking sensitive information. The resulting database will have only masked values and no sensitive data.

1. Create a masking definition on a cloned database. Implies the following:
   1. Select an appropriate ADM.
   2. Search and select sensitive columns (includes dependent columns and recommended masking formats).
   3. Review suggested formats and edit as necessary.
   4. Save.
2. Create a subset definition. Implies the following:
   1. Select an appropriate ADM.
   2. Submit the create subset job.
3. Edit the subsetting definition.

   On the **Data Masking** tab, search and select masking definitions. System validation checks for overlapping columns that use multiple masking definitions.
4. Generate the subset using the In-Place Delete option.

Summarizing the outcome:

- Copies subsetting and masking scripts to the target database.
- Performs data subsetting based on subset rules, if specified.
- Sequentially executes the pregenerated data masking scripts on the target database.
- Creates a masked copy of the production database for use in testing.

5.7.3 Masking Sensitive Data and Exporting a Subset of a Production Database

As the Security Administrator, you want to create copies of the production database by exporting a subset of production data with masked values.

1. Create a masking definition. Implies the following:
   1. Select an appropriate ADM.
   2. Search and select sensitive columns (includes dependent columns and recommended masking formats).
   3. Review suggested formats and edit as necessary.
   4. Save.
2. Create a subset definition. Implies the following:
   1. Select an appropriate ADM.
   2. Submit the create subset job.

3. Edit the subset definition.
   a. Define table rules, resulting in space estimates.
   b. On the **Data Masking** tab, search and select masking definitions. System validation checks for overlapping columns that use multiple masking definitions.
4. Generate the subset using the Export option.

Summarizing the outcome:

- Generates and executes a script to create a mapping table and a mapping function. Also creates a table to map the column(s) to the respective mapping function.
- Copies subsetting and masking scripts to the target database.
- Generates an export dump of production data, replacing sensitive data with masked values using the mapping function.

5.7.4 Performing Subset, Mask, and Delete Operations on a Test Database

As the Security Administrator, you want to create a usable test database by masking sensitive information. On import, the database will have only masked values and no sensitive data.

1. Create a masking definition. Implies the following:
   a. Select an appropriate ADM.
   b. Search and select sensitive columns (includes dependent columns and recommended masking formats).
   c. Review suggested formats and edit as necessary.
   d. Save.
2. Create a subset definition. Implies the following:
   a. Select an appropriate ADM.
   b. Submit the create subset job.
3. Edit the subset definition.
   a. Define table rules, resulting in space estimates.
   b. On the **Data Masking** tab, search and select masking definitions. System validation checks for overlapping columns that use multiple masking definitions.
4. Generate the subset using the In-Place Delete option.

Summarizing the outcome:

- Copies subsetting and masking scripts to the target database.
- Performs data subsetting based on subset rules, if specified.
- Following subset completion, sequentially executes the pregenerated data masking scripts on the target database.

- Applies masking definitions and subsetting rules, resulting in a masked database of reduced size.

5.7.5 Applying Column Rules

As the Security Administrator, you want to create a targeted subset by selecting large-sized columns and setting them to null or a fixed value. Table rules can also be used to further reduce database size. Impact of size reduction is immediately visible and applied to the final subset.

1. Create a subset definition. Implies the following:
   a. Select an appropriate ADM.
   b. Submit the create subset job.
2. Edit the subset definition.
   a. Click the **Table Rules** tab and select from existing options, if desired.
   b. Click the **Column Rules** tab, then click **Create**.
   c. Specify filtering criteria to search for large-sized columns and select the desired columns in the results table.
   d. Click **Manage Masking Formats** and select a format from the drop-down list. Enter a value if appropriate to the selection.
   e. Click **OK** and review the updated space estimates.
3. Generate the subset, using either the Export or In-Place Delete option.

Summarizing the outcome:

- Generates an export dump/subset of production data.
- Column rules are applied on the target database.
- If table rules were also applied, the resulting subset reflects the combined effect of table and column rules.

5.7.6 Exporting a Subset Definition That Uses Inline Masking

As the Security Administrator, you want to export a subset definition for reuse.

1. Create a subset definition. Implies the following:
   a. Select an appropriate ADM.
   b. Submit the create subset job.
2. Edit the subset definition.
   a. Create rules to compute space estimates.
   b. On the **Data Masking** tab, search and select masking definitions. System validation checks for overlapping columns that use multiple masking definitions.
3. Select the subset definition on the Subset home page and export it.

The subset definition is saved on the client machine as an XML file that potentially contains the following:

- Information on selected applications

- Rules and rule parameters
- Selected masking definitions
- Columns to set to null
- Pre- and post-scripts

If column rules were used, they would replace the masking definitions in the list.

5.7.7 Importing a Subset Definition That Uses Inline Masking

As the Security Administrator, you want to import a subset definition XML file to create replicas of the subset definition previously exported.

1. Import a subset definition.
2. Select an exported XML template that contains exported masking definitions. System validation:
   - Checks for overlapping columns that use multiple masking definitions.
   - Ensures that the masking definition specified is part of the same ADM as the current subset model.
3. Submit the job to create the subset model.

Summarizing the outcome:

- Creates a subset definition model
- Applies specified rules and calculates space estimates
- Remembers masking definitions that were part of the XML

5.7.8 Importing a Subset Dump

As the Security Administrator, you want to import a subset dump, which might contain either or both of the following:

- A masked version of a production database
- A subset version of a production database

Note that this example assumes a previous export dump.

1. On the subset home page, select **Import Subset Dump** from the **Actions** menu.
2. Provide credentials, a dump name, and select the dump location.
3. Provide the import type, tablespace options, and log file location details.
4. Schedule the job and submit.

The job reads the dump files and loads the data into the selected target database.

5.7.9 Saving a Subset Script Bundle

As the Security Administrator, you want to save a subset script bundle so that it can be executed on a target database external to Enterprise Manager.

This example presupposes the existence of a subset model that has required table rules and masking definitions.

1. On the subset home page, from the **Actions** menu, select **Generate**, then select **Subset**.
2. Complete the mode page as follows:
   1. Indicate the method of subset creation.
   2. Specify which credentials to use.
   3. Provide rule parameters as appropriate.
   4. Click **Continue**.
3. Complete the parameters page as follows:
   1. Select the location where to save the subset export.
   2. If the subset is to be stored externally, click the check box and select the location.
   3. Specify an export file name. Note that you can use the % wildcard.
   4. Specify the maximum file size and number of threads.
   5. Indicate whether to generate a log file and specify a log file name.
   6. Click **Continue**.
4. Note the progress of the script file generation. When complete, click **Download**.
5. Specify where to save the `SubsetBundle.zip` file.

5.8 Lifecycle Management

This section discusses the lifecycle management of Application Data Models, Data Masking, and Data Subsetting definitions.

In the event of an Enterprise Manager user being dropped or modified, the user can reassign the Application Data Model, data masking and data subsetting definitions to another user in the system. However, if the user doesn't reassign the Application Data Model, data masking and data subsetting definitions to another user, these definitions are automatically reassigned to the SYSMAN user.

When you try to reassign the Application Data Model, data masking and data subsetting definitions to another user in the system, and if the reassigned user already has a definition with the same name, the original definitions are renamed.

For example: User A has an Application Data Model, ADM1, and user B also has an Application Data Model, named ADM1. If user B is being dropped, and you choose to assign its definitions to user A, the original definition ADM1 is renamed to ADM1_B, The original definitions with the same name are renamed by suffixing "_" and adding the user name that is being dropped. After the reassignment, user A will now have both definitions ADM1 and ADM1_B.

Index