Oracle® Utilities Data Model

Implementation and Operations Guide

Release 12.2
E81921-01
September 2017

ORACLE

Oracle Utilities Data Model Implementation and Operations Guide, Release 12.2

E81921-01

Primary Author: Thomas Van Raalte

Contents

3 Access Layer Customization

4 ETL Implementation and Customization

5 Report and Query Customization

6 Metadata Collection and Reports

7 Working with User Roles and Privileges in Oracle Utilities Data Model

List of Tables

Preface

The *Oracle Utilities Data Model Implementation and Operations Guide* describes best practices for implementing a data warehouse based on the Oracle Utilities Data Model.

This preface contains the following topics:

Audience

This document is intended for business analysts, data modelers, data warehouse administrators, IT staff, and ETL developers who implement an Oracle Utilities Data Model warehouse.

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 Oracle Resources

Oracle provides many resources for you when implementing the Oracle Utilities Data Model.

Oracle Utilities Data Model Documentation Set

For more information on Oracle Utilities Data Model, see the following documents in the Oracle Utilities Data Model documentation set:

- *Oracle Utilities Data Model Installation Guide*
- *Oracle Utilities Data Model Reference*
- *Oracle Utilities Data Model Release Notes*

Oracle Technology Network

Visit the Oracle Technology Network (OTN to access to demos, whitepapers, Oracle By Example (OBE) tutorials, updated Oracle documentation, and other collateral.

Registering on OTN

You must register online before using OTN, Registration is free and can be done at

`www.oracle.com/technetwork/index.html`

Oracle Documentation on OTN

The Oracle Documentation site on OTN provides access to Oracle documentation. After you have a user name and password for OTN, you can go directly to the documentation section of the OTN Web site at

`docs.oracle.com`

Oracle Learning Library on OTN

The Oracle Learning Library provides free online training content (OBEs, Demos and Tutorials). After you have a user name and password for OTN, you can go directly to the Oracle Learning Library Web site at

`www.oracle.com/technetwork/tutorials/index.html`

Then you can search for the tutorial or demo (within "All") by name.

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.

1

Introduction to Oracle Utilities Data Model Customization

This chapter provides an introduction to customizing Oracle Utilities Data Model. It contains the following topics:

- What is the Oracle Utilities Data Model? (page 1-1)
  Oracle Utilities Data Model leverages several Oracle Database data warehouse and Business Intelligence concepts that need to be clarified to understand the structure and use of Oracle Utilities Data Model.
- Implementing an Oracle Utilities Data Model Warehouse (page 1-4)
  Although Oracle Utilities Data Model was designed following best practices, usually the model requires some customization to meet your business needs. The reasons that you might customize Oracle Utilities Data Model include: your business does not have a business area that is in the logical model of Oracle Utilities Data Model, or you must apply a new or different business rule.
- Before You Begin Customizing Oracle Utilities Data Model (page 1-5)
  Before you begin customizing Oracle Utilities Data Model, ensure that certain teams and committees exist.
- Performing Fit-Gap Analysis for Oracle Utilities Data Model (page 1-7)
  Fit-gap analysis is where you compare your information needs and business requirements with the structure that is available with Oracle Utilities Data Model. You identify any required functionality that is not included in the logical model and the default schema, as well as other modifications that are necessary to meet your requirements. The result of your fit-gap analysis is a customization report which is a brief explanation of the adaptations and adjustments required to customize Oracle Utilities Data Model to fit your environment.
- Data Encryption and Security for Oracle Utilities Data Model (page 1-8)
  To comply with privacy and data protection requirements, Oracle Utilities Data Model is certified with Transparent Data Encryption and Oracle Database Vault.

1.1 What is the Oracle Utilities Data Model?

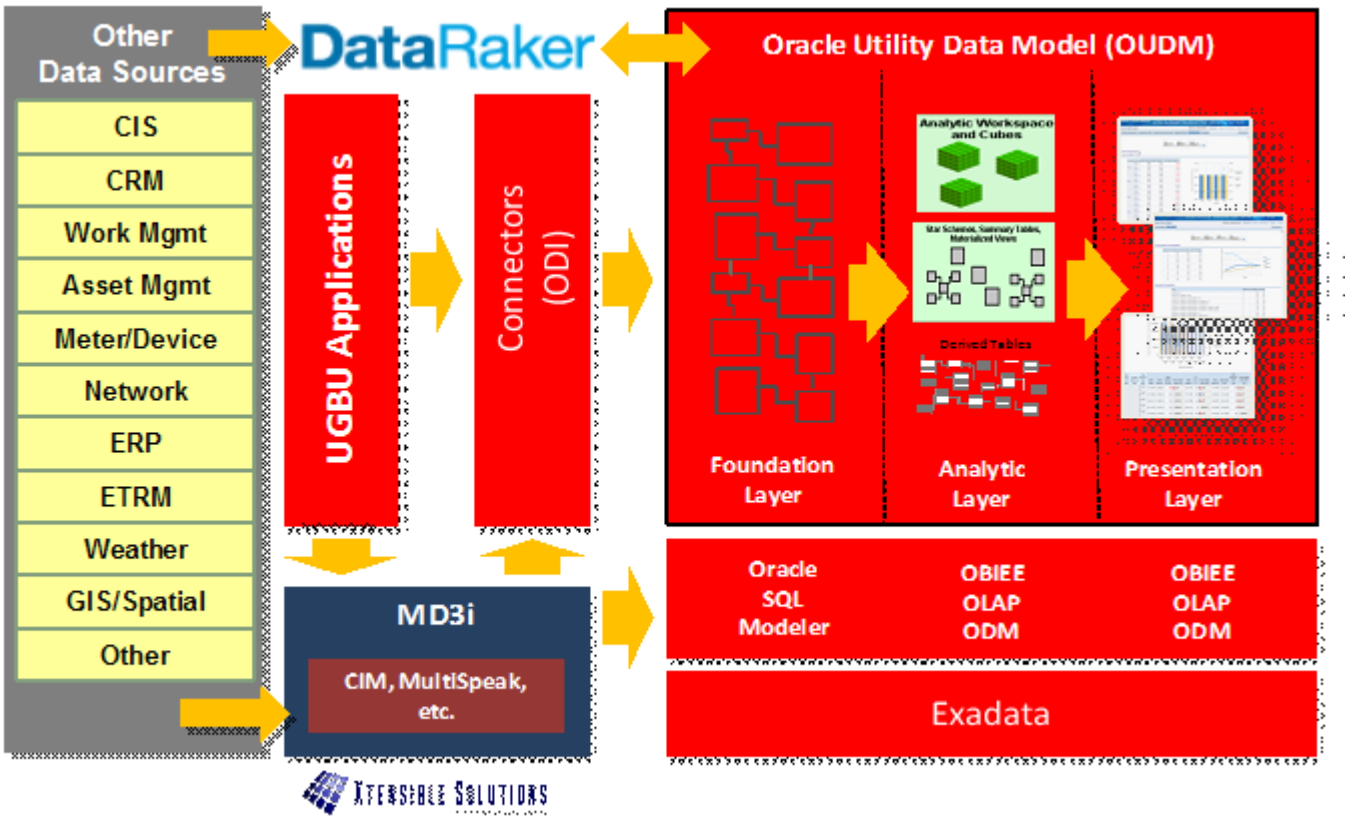
Oracle Utilities Data Model leverages several Oracle Database data warehouse and Business Intelligence concepts that need to be clarified to understand the structure and use of Oracle Utilities Data Model.

Oracle Utilities Data Model provides "One Single True Vision of the Business". This unique architecture provides the utilities Service Provider (CSP) Flexibility, Agility, Scalability and Accuracy to obtain a real competitive advantage.

A typical enterprise data warehouse architecture, as shown in Figure 1-1 (page 1-2), is composed of several layers ordered by the growing actionable value of the information in the warehouse:

- The Data Source layer (operational systems, Commercial-Off-The shelf solution, unstructured and syndicated data, with possibly a Master Data Management system).
- The Staging layer: Typically used for transformation and data cleansing. It is also sometimes used as Operational Data Store, in particular for real-time operational reporting.
- The Foundation layer: It is typically used to store all transactions and reference data at the most atomic level. Best practices require that this level is 3rd normal form, to avoid data redundancy.
- The Access and Performance or Analytical layer: this is the layer optimized for the business end-users. It usually contains the star schema to answer business questions, as well as OLAP tools and mining models.
- The Information (or Information Access) layer: This is the metadata layer and above, accessed by end-users through their Business Intelligence and/or reporting tools, or even external analytical tools (other OLAP or Mining tools). This layer is usually changeable by normal end-users (within their roles and responsibility). This is where the performance management applications provide their reports, where user roles, alerts, guided analytics, dashboards and reports are defined (usually by a specific BI administrator).
- The data movement from one layer to the other is run through ETL / ELT tools. One distinguishes the standard ETL/ELT (from data sources to foundation layer) from the intra-ETLs (from foundation layer up to the reporting).

Figure 1-1 Data Warehouse Reference Architecture with Oracle Utilities Data Model



Within a standard enterprise data warehouse architecture, as shown in Figure 1-1 (page 1-2), if an adapter is used, for example DataRaker, the Staging area is also provided. Oracle Utilities Data Model covers Foundation Layer, plus the intra-ETL part, and includes parts of the reporting area if OBIEE is used (Oracle Utilities Data Model also includes the pre-built OBIEE repository).

- Components of Oracle Utilities Data Model (page 1-3)
  Oracle Utilities Data Model includes several components.
- Oracle Products That Make Up Oracle Utilities Data Model (page 1-3)
  Several Oracle technologies are involved in building the infrastructure for Oracle Utilities Data Model:

1.1.1 Components of Oracle Utilities Data Model

Oracle Utilities Data Model includes several components.

- Logical model which is a third normal form (3NF) entity-object model. The logical model follows industry standards.
- Physical model defined as an Oracle Database schema.
- Intra-ETL database packages and SQL scripts to extract, transform, and load (ETL) data from the Oracle Utilities Data Model 3NF physical tables to the derived and aggregate tables in Oracle Utilities Data Model.
- Sample reports and dashboards developed using Oracle Business Intelligence Suite Extended Edition.
- DDL and installation scripts

> **Note:**
>
> When you use the Oracle Installer to install Oracle Utilities Data Model, you have the choice of performing two different types of installations:
>
> - Installation of the Oracle Utilities Data Model component
> - Installation of sample reports (and schemas)

Related Topics:

- *Oracle Utilities Data Model Installation Guide*

1.1.2 Oracle Products That Make Up Oracle Utilities Data Model

Several Oracle technologies are involved in building the infrastructure for Oracle Utilities Data Model:

- Oracle Database with OLAP, Data Mining and Partitioning Option (page 1-4)
- Oracle Development Tools (page 1-4)
- Oracle Business Intelligence Suite Enterprise Edition Presentation Tools (page 1-4)

Oracle Database with OLAP, Data Mining and Partitioning Option

Oracle Utilities Data Model uses a complete Oracle technical stack. It leverages the following data warehousing features of the Oracle Database: SQL model, compression, partitioning, advanced statistical functions, materialized views, data mining, and online analytical processing (OLAP).

Oracle Development Tools

You can use the following Oracle tools to customize the predefined physical models provided with Oracle Utilities Data Model, or to populate the target relational tables and materialized cube views.

Table 1-1 Oracle Development Tools Used with Oracle Utilities Data Model

Name	Use
SQL Developer or SQL*Plus	To modify, customize, and extend database objects
Analytic Workspace Manager	To view, create, develop, and manage OLAP dimensional objects.

Oracle Business Intelligence Suite Enterprise Edition Presentation Tools

Oracle Business Intelligence Suite Enterprise Edition is a comprehensive suite of enterprise BI products that delivers a full range of analysis and reporting capabilities. You can use Oracle Business Intelligence Suite Enterprise Edition Answers and Dashboard presentation tools to customize the predefined sample dashboard reports that are provided with Oracle Utilities Data Model.

Related Topics:

- Reporting Approaches in Oracle Utilities Data Model (page 5-1)
  There are two main approaches to creating reports from data in an Oracle Utilities Data Model warehouse: Relational Reporting and OLAP Reporting.

1.2 Implementing an Oracle Utilities Data Model Warehouse

Although Oracle Utilities Data Model was designed following best practices, usually the model requires some customization to meet your business needs. The reasons that you might customize Oracle Utilities Data Model include: your business does not have a business area that is in the logical model of Oracle Utilities Data Model, or you must apply a new or different business rule.

Typical physical model modifications include: adding, deleting, modifying, or renaming tables and columns; or altering foreign keys, constraints, or indexes.

To implement an Oracle Utilities Data Model warehouse, perform the following steps:

1. Perform the required organizational tasks.
2. Create a fit-gap analysis report.
3. In a development environment, install a copy of the Oracle Utilities Data Model.
4. Customize Oracle Utilities Data Model by making the changes you documented in the fit-gap analysis report. Make the changes in the following order:

a. Foundation layer of the physical model and the ETL to populate the foundation layer.

b. Access layer of the physical model and the ETL to populate the access layer.

5. In a test environment, make a copy of your customized version of Oracle Utilities Data Model. Then, following the documentation you created in Step 2, test the customized version of Oracle Utilities Data Model.

6. Following your typical procedures, roll the tested customized version of Oracle Utilities Data Model out into pre-production and, then, production.

> **Tip:**
>
> Keep 'clean' copies of the components delivered with Oracle Utilities Data Model components. This is important when upgrading to later versions of Oracle Utilities Data Model.

Related Topics:

- Before You Begin Customizing Oracle Utilities Data Model (page 1-5)
  Before you begin customizing Oracle Utilities Data Model, ensure that certain teams and committees exist.

1.3 Before You Begin Customizing Oracle Utilities Data Model

Before you begin customizing Oracle Utilities Data Model, ensure that certain teams and committees exist.

- Data warehouse governance steering committee.
- Implementation team.
- Fit-gap analysis team. This team consists of business analysts who can identify the business requirements and scope of the Oracle Utilities Data Model and at least some engineers in the Implementation team. Business members of this team must understand logical data modeling so that they can evaluate what changes must be made to the foundation and access layers of the physical model.

After these teams and committees are formed:

- Discuss the approach and determine the involvement and roles of every party involved in the customization (for example, business and IT).
- Agree on the scope of the project (that is, agree on what new data must be in the data warehouse and why it is needed).
- Agree on the timing and the working arrangements.
- Prerequisite Knowledge for Implementors (page 1-6)
  The Oracle Utilities Data Model uses much of the Oracle stack. Consequently, to successfully implement the Oracle Utilities Data Model, the implementation team needs certain prerequisite knowledge.

- Responsibilities of a Data Warehouse Governance Committee (page 1-6) Governance is of concern to any enterprise, executive team or individual with an interest in the processes, standards, and compliance. It is even more important to organizations that have invested in data warehousing. Data warehouse governance occurs within the context of overall IT governance. It provides the necessary policies, process and procedures, which must be clearly communicated to the entire corporation, from the IT employees to the front-end operational personnel.

1.3.1 Prerequisite Knowledge for Implementors

The Oracle Utilities Data Model uses much of the Oracle stack. Consequently, to successfully implement the Oracle Utilities Data Model, the implementation team needs certain prerequisite knowledge.

- Experience performing information and data analysis and data modeling. (Experience using Oracle SQL Data Modeler, is a plus.)
- An understanding of the Oracle technology stack, especially data warehouse (Database, Data Warehouse, OLAP, Data Mining, Oracle Business Intelligence Suite Extended Edition)
- Hands-on experience using:
  - Oracle Database
  - PL/SQL
  - SQL DDL and DML syntax
  - Analytic Workspace Manager
  - Oracle SQL Developer
  - Oracle Business Intelligence Suite Extended Edition Administrator, Answers, and Dashboards

1.3.2 Responsibilities of a Data Warehouse Governance Committee

Governance is of concern to any enterprise, executive team or individual with an interest in the processes, standards, and compliance. It is even more important to organizations that have invested in data warehousing. Data warehouse governance occurs within the context of overall IT governance. It provides the necessary policies, process and procedures, which must be clearly communicated to the entire corporation, from the IT employees to the front-end operational personnel.

Before you customize Oracle Utilities Data Model, setup a data warehouse governance steering committee if one does not exist. The role of this steering committee is to oversee the data warehouse to provide an environment that reaches across the enterprise and drives the best business value.

Data Warehouse Governance Committee: Overall Responsibilities

The data warehouse governance steering committee sets direction and response for the governance framework and covers the follow areas:

- The entire data warehouse life cycle.

- Agree on the data to process and make available to end-users.
- Determine what is the minimum quality criteria for the data that is available to end users and determine how to measure and analyze these criteria against the quality of the data that is the source data for the data warehouse.
- The business goals of the organization to apply core information from data warehouse.
- The policies, procedures and standards for data resource and data access.
- The life cycle of data warehouse component management.

Data Warehouse Governance Committee: Data Governance Responsibilities

The more detailed focus in data warehouse governance is data governance. Data governance tasks include:

- Approving the data modeling standards, metadata standards and other related standards. This includes determining a metadata strategy and identifying the data modeling tools to use that support these standards.
- Determining the data retention policy.
- Designing a data access policy based on legal restrictions and data security rules.
- Designing a data backup strategy that aligns with the impact analysis to the business unit.
- Monitoring and reporting on data usage, activity, and alerts.

Related Topics:

- Overview of Managing Metadata for Oracle Utilities Data Model (page 6-1)
  Metadata is any data about data and, as such, is an important aspect of the data warehouse environment. Metadata allows the end user and the business analyst to navigate through the possibilities at a higher business object level.

1.4 Performing Fit-Gap Analysis for Oracle Utilities Data Model

Fit-gap analysis is where you compare your information needs and business requirements with the structure that is available with Oracle Utilities Data Model. You identify any required functionality that is not included in the logical model and the default schema, as well as other modifications that are necessary to meet your requirements. The result of your fit-gap analysis is a customization report which is a brief explanation of the adaptations and adjustments required to customize Oracle Utilities Data Model to fit your environment.

The fit-gap analysis team writes the customization report by performing the following steps:

1. If you have performed previous evaluations, review the documentation from the previous phases, and if necessary add team members with the required business and technical expertise.
2. Review the data and map your logical entities and data structure with the Oracle Utilities Data Model logical model and schema:

- Starting from business requirements, questions, and rules, identify any entities and attributes that are *not* in the Oracle Utilities Data Model.
- Compare the Oracle Utilities Data Model to your existing application model if have one.
- Compare the Oracle Utilities Data Model to the OLTP data that you are using as a data source to the Oracle Utilities Data Model warehouse.

3. Determine the differences between your needs and Oracle Utilities Data Model schema. To help you with this task, produce a list of actions people may take with the system (examples rather than models), and create use cases for appraising the functionality of the Oracle Utilities Data Model Warehouse. Answer the following questions about the differences you find:
   - What differences you can live with, and what must be reconciled?
   - What can you do about the differences you cannot live with?
4. Identify the changes you must make to the default design of Oracle Utilities Data Model to create the customized warehouse. Identify these changes in the following order:
   a. Physical model.
   b. ETL mapping. Identify and design the source-ETL that you must create and identify and make any changes to the intra-ETL provided with Oracle Utilities Data Model.

> **Tip:**
>
> When identifying changes, ensure that the changes meet your security and metadata requirements.

5. Write the customization report, detailing what changes are required to make the Oracle Utilities Data Model match your business needs. This includes any additions and changes to interfaces to existing systems.
6. Based on the customization report update the project plan.

1.5 Data Encryption and Security for Oracle Utilities Data Model

To comply with privacy and data protection requirements, Oracle Utilities Data Model is certified with Transparent Data Encryption and Oracle Database Vault.

For more information on these topics, see:

Transparent Data Encryption

`http://www.oracle.com/technetwork/database/options/database-vault/index-085211.html`

See Also:

- *Oracle Database Administrator's Guide*
- *Oracle Database Vault Administrator's Guide* for information on using Oracle Database Vault

2
Physical Model Customization

Provides general information about customizing the physical model of Oracle Utilities Data Model and more detailed information about customizing the foundation layer of the physical model.

- Characteristics of the Default Physical Model (page 2-1)
  The default physical model of Oracle Utilities Data Model defines specified objects.
- Customizing the Oracle Utilities Data Model Physical Model (page 2-4)
  The starting point for the Oracle Utilities Data Model physical data model is the 3NF logical data model. The physical data model mirrors the logical model as much as possible, (although some changes in the structure of the tables or columns may be necessary) and defines database objects (such as tables, cubes, and views).
- Foundation Layer Customization (page 2-6)
  The first step in customizing the physical model of Oracle Utilities Data Model is customizing the foundation layer of the physical data model.
- General Recommendations When Designing Physical Structures (page 2-8)
  The `oudm_sys` schema delivered with Oracle Utilities Data Model was designed and defined following best practices for data access and performance.

Related Topics:

- Access Layer Customization (page 3-1)

2.1 Characteristics of the Default Physical Model

The default physical model of Oracle Utilities Data Model defines specified objects.

The default physical model of Oracle Utilities Data Model defines:

670+ tables and 4,800+ columns
1,300+ industry-specific measures and 80 KPIs
pre-built OLAP cubes and Analytical Models

The default physical model of the Oracle Utilities Data Model shares characteristics of a multischema "traditional" data warehouse, as described in "Layers in a "Traditional" Data Warehouse (page 2-1)", but defines all data structures in a single schema as described in "Layers in the Default Oracle Utilities Data Model Warehouse (page 2-2)".

Layers in a "Traditional" Data Warehouse

Historically, three layers are defined for a data warehouse environment:

- **Staging layer.** This layer is used when moving data from the operational system and other data sources into the data warehouse itself. It consists of temporary loading structures and rejected data. Having a staging layer enables the speedy extraction, transformation and loading (ETL) of data from your operational systems into data warehouse without disturbing any of the business users. It is in this layer

the much of the complex data transformation and data quality processing occurs. The most basic approach for the design of the staging layer is as a schema identical to the one that exists in the source operational system.

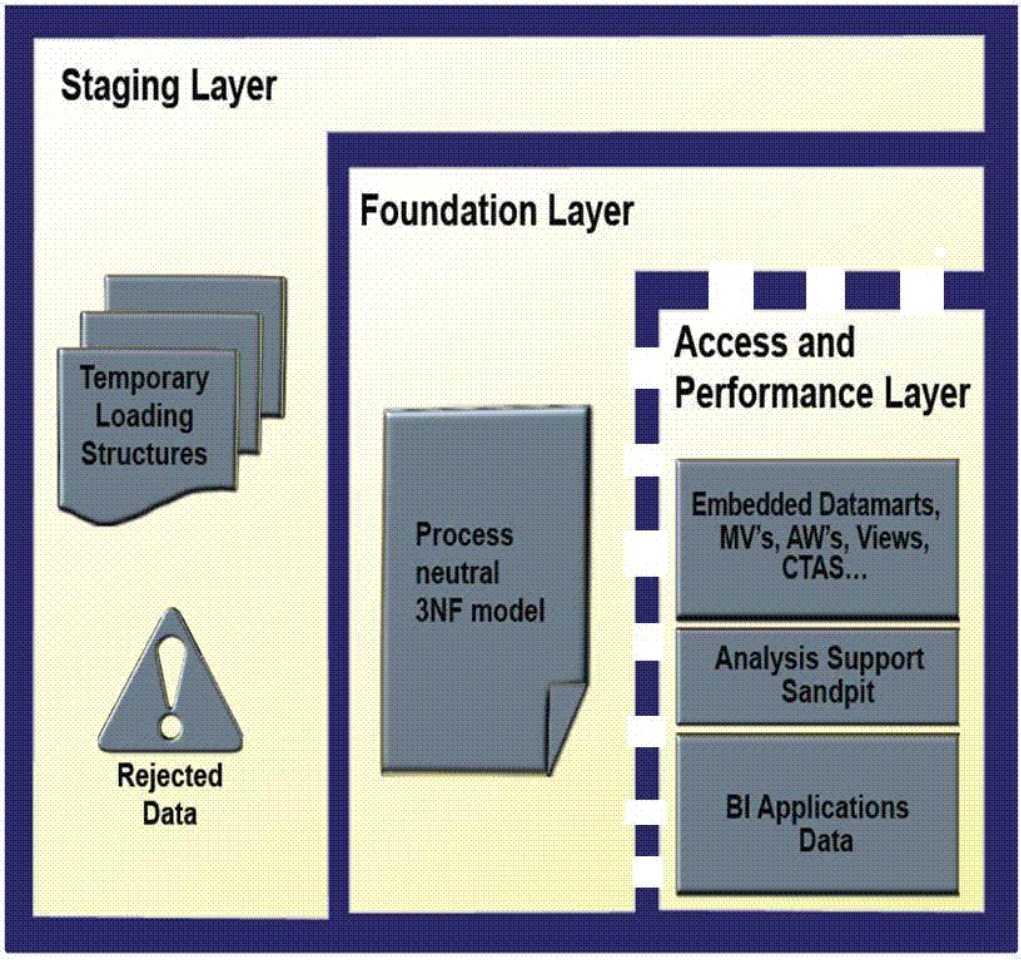
> **Note:**
>
> In some implementations this layer is not necessary, because all data transformation processing is done as needed as data is extracted from the source system before it is inserted directly into the foundation layer.

- **Foundation or integration layer.** This layer is traditionally implemented as a Third Normal Form (3NF) schema. A 3NF schema is a neutral schema design independent of any application, and typically has many tables. It preserves a detailed record of each transaction without any data redundancy and allows for rich encoding of attributes and all relationships between data elements. Users typically require a solid understanding of the data to navigate the more elaborate structure reliably. In this layer data begins to take shape and it is not uncommon to have some end-user application access data from this layer especially if they are time sensitive, as data becomes available here before it is transformed into the Access and Performance layer.
- **Access layer.** This layer is traditionally defined as a snowflake or star schema that describes a "flattened" or dimensional view of the data.

Layers in the Default Oracle Utilities Data Model Warehouse

Oracle Utilities Data Model warehouse environment also consists of three layers However, as indicated by the dotted line in Figure 2-1 (page 2-3), in the Oracle Utilities Data Model the definitions of the foundation and access layers are combined in a single schema.

Figure 2-1 Layers of an Oracle Utilities Data Model Warehouse



The layers in the Oracle Utilities Data Model warehouse are:

- **Staging layer**. As in a "traditional" data warehouse environment, an Oracle Utilities Data Model warehouse environment can have a staging layer. Because the definition of this layer varies by customer, a definition of this area is not provided as part of Oracle Utilities Data Model.
- **Foundation and Access layers.** The physical objects for these layers are defined in a single schema, the `oudm_sys` schema:
  - **Foundation layer.** The foundation layer of the Oracle Utilities Data Model is defined by base tables that present the data in 3NF (that is, tables that have the `DWB_` prefix). This layer also includes reference, lookup, and control tables defined in the `oudm_sys` schema (that is, the tables that have the `DWR_` , `DWL_`, `DWC_` prefixes).
  - **Access layer.** The access layer of Oracle Utilities Data Model is defined by derived and aggregate tables (defined with `DWD_` and `DWA_` prefixes), cubes (defined with a `CB$` prefix), and views (that is, views defined with the `DWV_` prefix), and cube views (defined with `_VIEW` suffix). These structures provide a summarized or "flattened" perspectives of the data in the foundation layer.

This layer also contains the results of the data mining models which are stored in derived (DWD_) tables. The access layer also includes the tables with prefixes as shown in Table 2-1 (page 2-6).

Related Topics:

- *Oracle Utilities Data Model Reference*

2.2 Customizing the Oracle Utilities Data Model Physical Model

The starting point for the Oracle Utilities Data Model physical data model is the 3NF logical data model. The physical data model mirrors the logical model as much as possible, (although some changes in the structure of the tables or columns may be necessary) and defines database objects (such as tables, cubes, and views).

To customize the default physical model of the Oracle Utilities Data Model perform the following steps:

1. Answer the physical customization questions.
2. Familiarize yourself with the characteristics of the logical and physical model of Oracle Utilities Data Model
3. Modify the foundation level of your physical model of Oracle Utilities Data Model, as needed.

   When defining physical structures:

   - Keep the foundation layer in 3NF form.
   - Follow the conventions used when creating the default physical model of Oracle Utilities Data Model.

> **Tip:**
>
> Package the changes you make to the physical data model as a patch to the oudm_sys schema.

4. Modify the access layer of your physical model of Oracle Utilities Data Model.

- Questions to Answer Before You Customize the Physical Model (page 2-5)
  When designing the physical model, remember that the logical data model is not one-to-one with the physical data model. Consider the load, query, and maintenance requirements when you convert the logical data model into the physical layer.
- Conventions When Customizing the Physical Model (page 2-5)
  When developing the physical model for Oracle Utilities Data Model, the naming conventions outlined were followed. Continue to follow these conventions as you customize the physical model.

Related Topics:

- Characteristics of the Default Physical Model (page 2-1)
  The default physical model of Oracle Utilities Data Model defines specified objects.

- Common Change Scenarios (page 2-7)
  There are several common change scenarios when customizing the foundation layer of the physical data model.
- Access Layer Customization (page 3-1)
- ETL Implementation and Customization (page 4-1)

2.2.1 Questions to Answer Before You Customize the Physical Model

When designing the physical model, remember that the logical data model is not one-to-one with the physical data model. Consider the load, query, and maintenance requirements when you convert the logical data model into the physical layer.

For example, answer the following questions before you design the physical data model:

- Do you need the physical data model to cover the full scope of the logical data model, or only part of the scope?

  Common Change Scenarios provides an overview discussion of making physical data model changes when your business needs do not result in a logical model that is the same as the Oracle Utilities Data Model logical model.
- What is the result of the source data profile?
- What is the data load frequency for each table?
- How many large tables are there and which tables are these?
- How will the tables and columns be accessed? What are the common joins?
- What is your data backup strategy?

2.2.2 Conventions When Customizing the Physical Model

When developing the physical model for Oracle Utilities Data Model, the naming conventions outlined were followed. Continue to follow these conventions as you customize the physical model.

General Naming Conventions for Physical Objects

Follow these guidelines for naming physical objects that you define:

- When naming the physical objects follow the naming guidelines for naming objects within an Oracle Database schema. For example:
  - Table and column names must start with a letter, can use only 30 alphanumeric characters or less, cannot contain spaces or some special characters such as "!" and cannot use reserved words.
  - Table names must be unique within a schema that is shared with views and synonyms.
  - Column names must be unique within a table.
- Although it is common to use abbreviations in the physical modeling stage, as much as possible, use names for the physical objects that correspond to the names of the entities in the logical model. Use consistent abbreviations to avoid programmer and user confusion.

- When naming columns, use short names if possible. Short column names reduce the time required for SQL command parsing.
- The `oudm_sys` schema delivered with Oracle Utilities Data Model uses the prefixes and suffixes shown Table 2-1 (page 2-6) to identify object types.

Table 2-1 Default Physical Object Prefixes and Suffixes in Oracle Utilities Data Model

Prefix or Suffix	Used for Name of These Objects
`_VIEW`	A relational view of an OLAP cube, dimension, or hierarchy.
`CCB_`	Customized OLAP cubes.
`CUBE`	Created when OLAP cubes are built. Used to store logs and results.
`DM$`	Created when the mining models are trained. Used to store trained model and logs.
`DR$`	Created when the mining models are trained. Used to store trained model and logs.
`DWA_`	Aggregate tables which are materialized views.
`DWB_`	Base transaction data (3NF) tables.
`DWC_`	Control tables.
`DWD_`	Derived tables -- including data mining result tables.
`DWL_`	Lookup tables.
`DWR_`	Reference data tables.
`DWV_`	Relational view of time dimension

Domain Definition Standards

A domain is a set of values allowed for a column. The domain can be enforced by a foreign key, check constraints, or the application on top of the database. Define the standards for each domain across the model such as:

- Date and time type, such as `'YYYY-MM-DD'`. For example, be aware that most date columns (abbreviation DT) in Oracle Utilities Data Model may contain the time, such as `EVT_STRT_DT`. There is no separate `TIME` column.
- Numeric value in different situations. For example, all columns of type `COUNT` are `NUMBER(9,0)` while all monetary-like columns (`AMOUNT`) are `NUMBER(16,5)`.
- Character string length in different situations. For example, all Code columns are `VARCHAR2(100)`, Name (`NAME`) and Description columns (`DSCR`) are respectively 500 and 1000 characters long (with some exceptions). Indicator columns (`IND`) are `CHAR(1)`.
- Coded value definition such as key or description. For example, all "Key" columns are `NUMBER(30)`.

Related Topics:

- *Oracle Utilities Data Model Reference*

2.3 Foundation Layer Customization

The first step in customizing the physical model of Oracle Utilities Data Model is customizing the foundation layer of the physical data model.

The foundation layer of the physical model mirrors the 3NF logical model of Oracle Utilities Data Model, you might choose to customize the foundation layer to reflect differences between your logical model needs and the default logical model of Oracle Utilities Data Model. Additionally, you might need to customize the physical objects in the foundation layer to improve performance (for example, you might choose to compress some foundation layer tables).

When making changes to the foundation layer, keep the following points in mind:

- When changing the foundation layer objects to reflect your logical model design, make as few changes as possible. "" outlines the most common customization changes you will make in this regard.
- When defining new foundation layer objects or when redesigning existing foundation layer objects for improved performance, follow the and "".
- Remember that changes to the foundation layer objects can also impact the access layer objects.

> **Note:**
>
> Approach any attempt to change the Oracle Utilities Data Model with caution. The foundation layer of the physical model of the Oracle Utilities Data Model has (at its core) a set of generic structures that allow it to be flexible and extensible. You may disable temporarily Foreign Keys and constraints if required but do not forget to set them back when the entities are back in use. Before making extensive additions, deletions, or changes, ensure that you understand the full range of capabilities of Oracle Utilities Data Model and that you cannot handle your requirements using the default objects in the foundation layer.

- Common Change Scenarios (page 2-7)
  There are several common change scenarios when customizing the foundation layer of the physical data model.

Related Topics:

- General Recommendations When Designing Physical Structures (page 2-8)
  The `oudm_sys` schema delivered with Oracle Utilities Data Model was designed and defined following best practices for data access and performance.
- Conventions When Customizing the Physical Model (page 2-5)
  When developing the physical model for Oracle Utilities Data Model, the naming conventions outlined were followed. Continue to follow these conventions as you customize the physical model.

2.3.1 Common Change Scenarios

There are several common change scenarios when customizing the foundation layer of the physical data model.

- **Additions to Existing Structures**

  If you identify business areas or processes that are not supported in the default foundation layer of the physical data model of Oracle Utilities Data Model, add new tables and columns.

Carefully study the default foundation layer of the physical data model of Oracle Utilities Data Model (and the underlying logical data model) to avoid building redundant structures when making additions. If these additions add high value to your business value, communicate the additions back to the Oracle Utilities Data Model Development Team for possible inclusion in future releases of Oracle Utilities Data Model.

- **Deletions of Existing Structures**

  If there are areas of the model that cannot be matched to any of the business requirements of your legacy systems, it is safer to keep these structures and not populate that part of the warehouse.

  Deleting a table in the foundation layer of the physical data model can destroy relationships needed in other parts of the model or by applications based on the it. Some tables may not be needed during the initial implementation, but you may want to use these structures at a later time. If this is a possibility, keeping the structures now saves re-work later. If tables are deleted, perform a thorough analysis to identify all relationships originating from that entity.

- **Changes to Existing Structures**

  In some situations some structures in the foundation layer of the physical data model of the Oracle Utilities Data Model may not exactly match the corresponding structures that you use.

  Before implementing changes, identify the impact that the changes would have on the database design of Oracle Utilities Data Model. Also identify the impact on any applications based on the new design.

2.4 General Recommendations When Designing Physical Structures

The `oudm_sys` schema delivered with Oracle Utilities Data Model was designed and defined following best practices for data access and performance.

Continue to use these practices when you add new physical objects. This section provides information about how decisions about the following physical design aspects were made to the default Oracle Utilities Data Model:

- Tablespaces in Oracle Utilities Data Model (page 2-9)
  A tablespace consists of one or more data files, which are physical structures within the operating system you are using.
- Data Compression in Oracle Utilities Data Model (page 2-10)
  A key decision that you must make is whether to compress your data. Using table compression reduces disk and memory usage, often resulting in a better scale-up performance for read-only operations.
- Tables for Supertype and Subtype Entities in Oracle Utilities Data Model (page 2-12)
  Create separate tables for the super type and all sub type entities in Oracle Utilities Data Model.
- Surrogate Keys in the Physical Model (page 2-12)
  The surrogate key method for primary key construction involves taking the natural key components from the source systems and mapping them through a process of

assigning a unique key value to each unique combination of natural key components (including source system identifier).

- Integrity Constraints in Oracle Utilities Data Model (page 2-13)
  Integrity constraints are used to enforce business rules associated with your database and to prevent invalid information in the tables. The most common types of constraints include:
- Indexes and Partitioned Indexes in Oracle Utilities Data Model (page 2-13)
  Indexes are optional structures associated with tables or clusters. In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.
- Partitioned Tables in Oracle Utilities Data Model (page 2-14)
  Partitioning allows a table, index or index-organized table to be subdivided into smaller pieces. Each piece of the database object is called a partition.
- Parallel Execution in Oracle Utilities Data Model (page 2-16)
  Parallel Execution enables a database task to be parallelized or divided into smaller units of work, thus allowing multiple processes to work concurrently. By using parallelism, a terabyte of data can be scanned and processed in minutes or less, not hours or days.

2.4.1 Tablespaces in Oracle Utilities Data Model

A tablespace consists of one or more data files, which are physical structures within the operating system you are using.

Recommendations: Defining Tablespaces

If possible, define tablespaces so that they represent logical business units.

Use ultra large data files for a significant improvement in very large Oracle Utilities Data Model warehouse.

Changing the Tablespace and Partitions Used by Tables

You can change the tablespace and partitions used by Oracle Utilities Data Model tables. What you do depends on whether the Oracle Utilities Data Model table has partitions:

- For tables that do not have partitions (that is, lookup tables and reference tables), you can change the existing tablespace for a table.

  By default, Oracle Utilities Data Model defines the partitioned tables as interval partitioning, which means the partitions are created only when new data arrives.

  Consequently, for Oracle Utilities Data Model tables that have partitions (that is, Base, Derived, and Aggregate tables), for the new interval partitions to be generated in new tablespaces rather than current ones, issue the following statements:

  ```
  ALTER TABLE table_name MODIFY DEFAULT ATTRIBUTES
  TABLESPACE new_tablespace_name;
  ```

  When new data is inserted in the table specified by *table_name*, a new partition is automatically created in the tablespace specified by *tablespace new_tablespace_name*.

- For tables that have partitions (that is, base, derived, and aggregate tables), you can specify that new interval partitions be generated into new tablespaces.

  For Oracle Utilities Data Model tables that do not have partitions (that is, lookup tables and reference tables), to change the existing tablespace for a table then issue the following statement:

  ```
  ALTER TABLE table_name MOVE TABLESPACE new_tablespace_name;
  ```

2.4.2 Data Compression in Oracle Utilities Data Model

A key decision that you must make is whether to compress your data. Using table compression reduces disk and memory usage, often resulting in a better scale-up performance for read-only operations.

Recommendations: Data Compression

Table compression can also speed up query execution by minimizing the number of round trips required to retrieve data from the disks. Compressing data however imposes a performance penalty on the load speed of the data. Most of the base tables in the Oracle Utilities Data Model are compressed tables.

In general, choose to compress the data. The overall performance gain typically outweighs the cost of compression.

If you decide to use compression, consider sorting your data before loading it to achieve the best possible compression rate. The easiest way to sort incoming data is to load it using an `ORDER BY` clause on either your `CTAS` or `IAS` statement. Specify an `ORDER BY` a `NOT NULL` column (ideally non numeric) that has many distinct values (1,000 to 10,000).

- Types of Data Compression Available (page 2-10)
  Oracle Database offers the following types of compression:

2.4.2.1 Types of Data Compression Available

Oracle Database offers the following types of compression:

- Basic or Standard Compression (page 2-10)
- Advanced Row Compression (page 2-11)
- Hybrid Columnar Compression (HCC) (page 2-11)

Basic or Standard Compression

With standard compression Oracle Database compresses data by eliminating duplicate values in a database block. Standard compression only works for direct path operations (CTAS or IAS). If the data is modified using any kind of conventional DML operation (for example updates), the data within that database block is uncompressed to make the modifications and is written back to the disk uncompressed.

By using a compression algorithm specifically designed for relational data, Oracle Database can compress data effectively and in such a way that Oracle Database incurs virtually no performance penalty for SQL queries accessing compressed tables.

Oracle Utilities Data Model leverages the compress feature for all base, derived, and aggregate tables which reduces the amount of data being stored, reduces memory usage (more data per memory block), and increases query performance.

You can specify table compression by using the `COMPRESS` clause of the `CREATE TABLE` statement or you can enable compression for an existing table by using `ALTER TABLE` statement as shown below.

```
alter table <tablename> move compress;
```

Advanced Row Compression

Advanced row compression is a component of the Advanced Compression option. With advanced row compression, just like standard compression, Oracle Database compresses data by eliminating duplicate values in a database block. But unlike standard compression advanced row compression allows data to remain compressed during all types of data manipulation operations, including conventional DML such as `INSERT` and `UPDATE`.

> **See Also:**
>
> For information about Oracle Advanced Compression, see the "Using Table Compression to Save Storage Costs" OBE tutorial.
>
> To access the tutorial, open the Oracle Learning Library in your browser at Oracle Learning Library and, then, search for the tutorial by name.

Hybrid Columnar Compression (HCC)

HCC is available with some storage formats and achieves its compression using a logical construct called the compression unit which is used to store a set of hybrid columnar-compressed rows. When data is loaded, a set of rows is pivoted into a columnar representation and compressed. After the column data for a set of rows has been compressed, it is fit into the compression unit. If a conventional DML is issued against a table with HCC, the necessary data is uncompressed to do the modification and then written back to the disk using a block-level compression algorithm.

> **Tip:**
>
> If your data set is frequently modified using conventional DML, then the use of HCC is not recommended; instead, the use of advanced row compression is recommended.

HCC provides different levels of compression, focusing on query performance or compression ratio respectively. With HCC optimized for query, fewer compression algorithms are applied to the data to achieve good compression with little to no performance impact. However, compression for archive tries to optimize the compression on disk, irrespective of its potential impact on the query performance.

Related Topics:

- *Oracle Database Administrator's Guide*
- *Oracle Database Concepts*

2.4.3 Tables for Supertype and Subtype Entities in Oracle Utilities Data Model

Create separate tables for the super type and all sub type entities in Oracle Utilities Data Model.

A supertype is a generic entity type that has a relationship with one or more subtypes.

A subtype is a sub-grouping of the entities in an entity type that is meaningful to the organization and that shares common attributes or relationships distinct from other subgroups.

- Subtypes inherit all supertype attributes
- Subtypes have attributes that are different from other subtypes

Recommendations: Tables for Supertype and Subtype Entities

Create separate tables for the super type and all sub type entities for the following reasons:

- Data integrity enforced at database level. (using `NOT NULL` column constraints)
- Relationships can be accurately modeled and enforced including those which apply to only one subtype
- Physical model closely resembles the logical data model.
- It is easier to correlate the logical data model with the physical data model and support the logical data model enhancements and changes.
- Physical data model reflects true business rules (for example, if there are some attributes or relationships mandatory for only one subtype.)

2.4.4 Surrogate Keys in the Physical Model

The surrogate key method for primary key construction involves taking the natural key components from the source systems and mapping them through a process of assigning a unique key value to each unique combination of natural key components (including source system identifier).

The resulting primary key value is completely non-intelligent and is typically a numeric data type for maximum performance and storage efficiency.

Advantages of Surrogate keys include:

- Ensure uniqueness: data distribution
- Independent of source systems
- Re-numbering
- Overlapping ranges
- Uses the numeric data type which is the most performant data type for primary keys and joins

Disadvantages of Surrogate keys:

- Have to allocate during ETL

- Complex and expensive re-processing and data quality correction
- Not used in queries – performance impact
- The operational business intelligence requires natural keys to join to operational systems

2.4.5 Integrity Constraints in Oracle Utilities Data Model

Integrity constraints are used to enforce business rules associated with your database and to prevent invalid information in the tables. The most common types of constraints include:

- `PRIMARY KEY` constraints, this is usually defined on the surrogate key column to ensure uniqueness of the record identifiers. In general, it is recommended that you specify the `ENFORCED ENABLED RELY` mode.
- `UNIQUE` constraints, to ensure that a given column (or set of columns) is unique. For slowly changing dimensions, it is recommended that you add a unique constraint on the Business Key and the Effective From Date columns to allow tracking multiple versions (based on surrogate key) of the same Business Key record.
- `NOT NULL` constraints, to ensure that no null values are allowed. For query rewrite scenarios, it is recommended that you have an inline explicit `NOT NULL` constraint on the primary key column in addition to the primary key constraint.
- `FOREIGN KEY` constraints, to ensure that relation between tables are being honored by the data. Usually in data warehousing environments, the foreign key constraint is present in `RELY DISABLE NOVALIDATE` mode.

The Oracle Database uses constraints when optimizing SQL queries. Although constraints can be useful in many aspects of query optimization, constraints are particularly important for query rewrite of materialized views. Under some specific circumstances, constraints need space in the database. These constraints are in the form of the underlying unique index.

Unlike data in many relational database environments, data in a data warehouse is typically added or modified under controlled circumstances during the extraction, transformation, and loading (ETL) process.

2.4.6 Indexes and Partitioned Indexes in Oracle Utilities Data Model

Indexes are optional structures associated with tables or clusters. In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.

- Bitmap indexes are optimized index structures for set-oriented operations. Additionally, they are necessary for some optimized data access methods such as star transformations. Bitmap indexes are typically only a fraction of the size of the indexed data in the table.
- B-tree indexes are most effective for high-cardinality data: that is, for data with many possible values, such as customer name or phone number. However, fully indexing a large table with a traditional B-tree index can be prohibitively expensive in terms of disk space because the indexes can be several times larger than the data in the table. B-tree indexes can be stored specifically in a compressed manner to enable huge space savings, storing more keys in each index block, which also leads to less I/O and better performance.

Recommendations: Indexes and Partitioned Indexes

Make the majority of the indexes in your customized Oracle Utilities Data Model bitmap indexes.

Use B-tree indexes only for unique columns or other columns with very high cardinalities (that is, columns that are almost unique). Store the B-tree indexes in a compressed manner.

Partition the indexes. Indexes are just like tables in that you can partition them, although the partitioning strategy is not dependent upon the table structure. Partitioning indexes makes it easier to manage the data warehouse during refresh and improves query performance.

Typically, specify the index on a partitioned table as local. Bitmap indexes on partitioned tables must always be local. B-tree indexes on partitioned tables can be global or local. However, in a data warehouse environment, local indexes are more common than global indexes. Use global indexes only when there is a specific requirement which cannot be met by local indexes (for example, a unique index on a nonpartitioning key, or a performance requirement).

2.4.7 Partitioned Tables in Oracle Utilities Data Model

Partitioning allows a table, index or index-organized table to be subdivided into smaller pieces. Each piece of the database object is called a partition.

Each partition has its own name, and may optionally have its own storage characteristics. From the perspective of a database administrator, a partitioned object has multiple pieces that can be managed either collectively or individually. This gives the administrator considerable flexibility in managing partitioned objects. However, from the perspective of the application, a partitioned table is identical to a nonpartitioned table. No modifications are necessary when accessing a partitioned table using SQL DML commands.

As discussed in the following topics, partitioning can provide tremendous benefits to a wide variety of applications by improving manageability, availability, and performance:

- Partitioning the Oracle Utilities Data Model for Manageability (page 2-15)
  Range partitioning helps improve the manageability and availability of large volumes of data.
- Partitioning the Oracle Utilities Data Model for Easier Data Access (page 2-15)
  Range partitioning also helps ensure that only the necessary data to answer a query is scanned. Consider the case where business users predominately access the sales data on a weekly basis (for example, total sales per week) then range partitioning this table by day ensures that the data is accessed in the most efficient manner, as only seven partitions must be scanned to answer the business users query instead of the entire table. The ability to avoid scanning irrelevant partitions is known as partition pruning.
- Partitioning the Oracle Utilities Data Model for Join Performance (page 2-15)
  Sub-partitioning by hash is used predominately for performance reasons. Oracle Database uses a linear hashing algorithm to create sub-partitions.

> **See Also:**
>
> To understand the various partitioning techniques in Oracle Database, see the "Manipulating Partitions in Oracle Database 11*g*" OBE tutorial.
>
> To access the tutorial, open the Oracle Learning Library in your browser at Oracle Learning Library and, then, search for the tutorial by name.

2.4.7.1 Partitioning the Oracle Utilities Data Model for Manageability

Range partitioning helps improve the manageability and availability of large volumes of data.

Consider the case where two year's worth of sales data or 100 terabytes (TB) is stored in a table. At the end of each day a new batch of data must be to loaded into the table and the oldest days worth of data must be removed. If the `Sales` table is range partitioned by day then the new data can be loaded using a partition exchange load. This is a sub-second operation that has little or no impact on end user queries.

2.4.7.2 Partitioning the Oracle Utilities Data Model for Easier Data Access

Range partitioning also helps ensure that only the necessary data to answer a query is scanned. Consider the case where business users predominately access the sales data on a weekly basis (for example, total sales per week) then range partitioning this table by day ensures that the data is accessed in the most efficient manner, as only seven partitions must be scanned to answer the business users query instead of the entire table. The ability to avoid scanning irrelevant partitions is known as partition pruning.

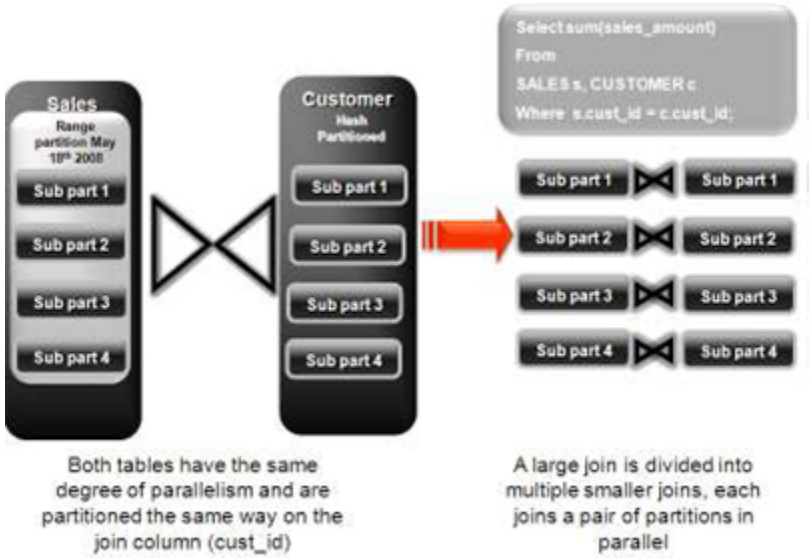
2.4.7.3 Partitioning the Oracle Utilities Data Model for Join Performance

Sub-partitioning by hash is used predominately for performance reasons. Oracle Database uses a linear hashing algorithm to create sub-partitions.

A major performance benefit of hash partitioning is partition-wise joins. Partition-wise joins reduce query response time by minimizing the amount of data exchanged among parallel execution servers when joins execute in parallel. This significantly reduces response time and improves both CPU and memory resource usage. In a clustered data warehouse, this significantly reduces response times by limiting the data traffic over the interconnect (IPC), which is the key to achieving good scalability for massive join operations. Partition-wise joins can be full or partial, depending on the partitioning scheme of the tables to be joined.

As illustrated a full partition-wise join divides a join between two large tables into multiple smaller joins. Each smaller join, performs a joins on a pair of partitions, one for each of the tables being joined. For the optimizer to choose the full partition-wise join method, both tables must be equi-partitioned on their join keys. That is, they have to be partitioned on the same column with the same partitioning method. Parallel execution of a full partition-wise join is similar to its serial execution, except that instead of joining one partition pair at a time, multiple partition pairs are joined in parallel by multiple parallel query servers. The number of partitions joined in parallel is determined by the Degree of Parallelism (DOP).

Figure 2-2 Partitioning for Join Performance



Recommendations: Number of Hash Partitions

To ensure that the data gets evenly distributed among the hash partitions, it is highly recommended that the number of hash partitions is a power of 2 (for example, 2, 4, 8, and so on). A good rule of thumb to follow when deciding the number of hash partitions a table should have is `2 X # of CPUs` rounded to up to the nearest power of 2.

If your system has 12 CPUs, then 32 can be a good number of hash partitions. On a clustered system the same rules apply. If you have 3 nodes each with 4 CPUs, then 32 can still be a good number of hash partitions. However, ensure that each hash partition is at least 16MB in size. Many small partitions do not have efficient scan rates with parallel query. Consequently, if using the number of CPUs makes the size of the hash partitions too small, use the number of Oracle RAC nodes in the environment (rounded to the nearest power of 2) instead.

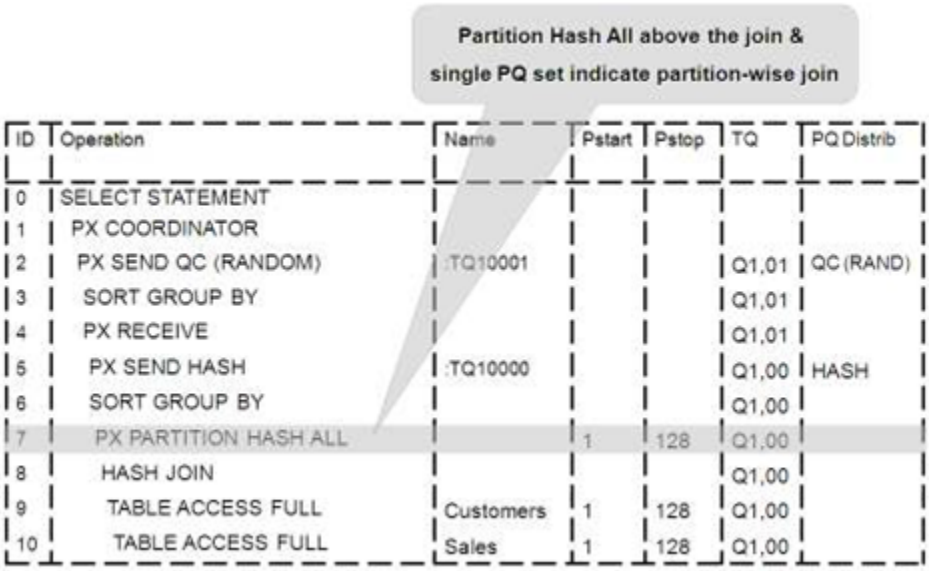
2.4.8 Parallel Execution in Oracle Utilities Data Model

Parallel Execution enables a database task to be parallelized or divided into smaller units of work, thus allowing multiple processes to work concurrently. By using parallelism, a terabyte of data can be scanned and processed in minutes or less, not hours or days.

Figure 2-3 (page 2-17) illustrates the parallel execution of a full partition-wise join between two tables, Sales and Customers. Both tables have the same degree of parallelism and the same number of partitions. They are range partitioned on a date field and sub partitioned by hash on the `cust_id` field. As illustrated in the picture, each partition pair is read from the database and joined directly.

There is no data redistribution necessary, thus minimizing IPC communication, especially across nodes. Below figure shows the execution plan for this join.

Figure 2-3 Parallel Execution of a Full Partition-Wise Join Between Two Tables



To ensure that you get optimal performance when executing a partition-wise join in parallel, specify a number for the partitions in each of the tables that is larger than the degree of parallelism used for the join. If there are more partitions than cluster database servers, each cluster database is given one pair of partitions to join, when the cluster database completes that join, it requests another pair of partitions to join. This process repeats until all pairs are processed. This method enables the load to be balanced dynamically (for example, 128 partitions with a degree of parallelism of 32).

What happens if only one table that you are joining is partitioned? In this case the optimizer picks a partial partition-wise join. Unlike full partition-wise joins, partial partition-wise joins can be applied if only one table is partitioned on the join key. Hence, partial partition-wise joins are more common than full partition-wise joins. To execute a partial partition-wise join, Oracle Database dynamically repartitions the other table based on the partitioning strategy of the partitioned table.

After the other table is repartitioned, the execution is similar to a full partition-wise join. The redistribution operation involves exchanging rows between parallel execution servers. This operation leads to interconnect traffic in Oracle RAC environments, since data must be repartitioned across node boundaries.

Figure 2-4 Partial Partition-Wise Join

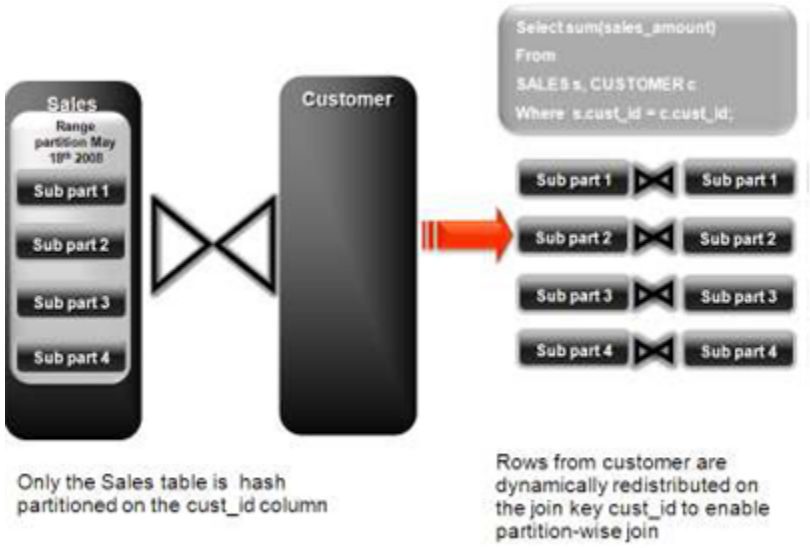


Figure 2-3 (page 2-17) illustrates a partial partition-wise join. It uses the same example as in Figure 2-4 (page 2-18), except that the customer table is not partitioned. Before the join operation is executed, the rows from the customers table are dynamically redistributed on the join key.

- Enabling Parallel Execution for a Session (page 2-18)
- Enabling Parallel Execution of DML Operations (page 2-18)
- Enabling Parallel Execution at the Table Level (page 2-19)
  The setting of parallelism for a table influences the optimizer.

2.4.8.1 Enabling Parallel Execution for a Session

Parallel query is the most commonly used parallel execution feature in Oracle Database. Parallel execution can significantly reduce the elapsed time for large queries. To enable parallelization for an entire session, execute the following statement:

```
alter session enable parallel query;
```

> **Note:**
>
> It is usually suggested to set at session level rather than at the system level.

2.4.8.2 Enabling Parallel Execution of DML Operations

Data Manipulation Language (DML) operations such as `INSERT`, `UPDATE`, and `DELETE` can be parallelized by Oracle Database. Parallel execution can speed up large DML operations and is particularly advantageous in data warehousing environments. To enable parallelization of DML statements, execute the following statement:

```
alter session enable parallel dml;
```

When you issue a DML statement such as an INSERT, UPDATE, or DELETE, Oracle Database applies a set of rules to determine whether that statement can be parallelized. The rules vary depending on whether the statement is a DML INSERT statement, or a DML UPDATE or DELETE statement.

- The following rules apply when determining how to parallelize DML UPDATE and DELETE statements:
  - Oracle Database can parallelize UPDATE and DELETE statements on partitioned tables, but only when multiple partitions are involved.
  - You cannot parallelize UPDATE or DELETE operations on a nonpartitioned table or when such operations affect only a single partition.
- The following rules apply when determining how to parallelize DML INSERT statements:
  - Standard INSERT statements using a VALUES clause cannot be parallelized.
  - Oracle Database can parallelize only INSERT . . . SELECT . . . FROM statements.

2.4.8.3 Enabling Parallel Execution at the Table Level

The setting of parallelism for a table influences the optimizer.

When using parallel query, also enable parallelism at the table level by issuing the following statement:

```
alter table <table_name> parallel 32;
```

3
Access Layer Customization

This chapter provides information about customizing the access layer of Oracle Utilities Data Model. It includes the following topics:

- Introduction to Customizing the Access Layer of Oracle Utilities Data Model (page 3-1)
  The access layer of Oracle Utilities Data Model provides the calculated and summarized ("flattened") perspectives of the data needed by business intelligence tools. Access layer objects are populated using the data from the foundation layer 3NF objects.
- Derived Tables in Oracle Utilities Data Model (page 3-2)
  Derived tables have a `DWD_` prefix.
- Aggregate Tables in Oracle Utilities Data Model (page 3-5)
  Aggregate tables are tables that aggregate or "roll up" the data to one level higher than a base or derived table (and other functions can also be in the aggregate tables such as average, count, min, max, and others).
- Dimensional Components in Oracle Utilities Data Model (page 3-6)
  There is often much discussion regarding the 'best' modeling approach to take for any given data warehouse with each style, classic 3NF and dimensional having their own strengths and weaknesses.
- Materialized Views in Oracle Utilities Data Model (page 3-19)
  Materialized views are query results that have been stored or "materialized" in advance as schema objects. From a physical design point of view, materialized views resemble tables or partitioned tables and behave like indexes in that they are used transparently and can improve performance.

3.1 Introduction to Customizing the Access Layer of Oracle Utilities Data Model

The access layer of Oracle Utilities Data Model provides the calculated and summarized ("flattened") perspectives of the data needed by business intelligence tools. Access layer objects are populated using the data from the foundation layer 3NF objects.

The access layer objects in the `oudm_sys` schema include: derived and aggregate tables and OLAP cube views. This layer also contains data mining models. The results of these models are stored in derived tables. The models themselves are also defined in the `oudm_sys` schema.

When designing and customizing access layer objects:

- Follow the general guidelines for customizing physical objects.
- Design access layer objects to support business intelligence reports and queries that your business end-users require.

The following topics provide specialized information about designing and customizing access layer objects:

3.2 Derived Tables in Oracle Utilities Data Model

Derived tables have a `DWD_` prefix.

Derived tables are tables where one of the following apply:

- Have values as the result of a non-aggregate calculation against the data in the foundation layer tables.
- Have some minimal level of aggregation, typically at the day level (for example, `DWD_ACCT_DEBT_DAY`).
- Summarize, at day or month level, all activities over the period of specific processes, split in various attributes (for example, `DWD_ACCT_BAL_MO`).

Some derived tables leverage, and hence are dependent on, other derived tables (for example `DWD_CUST_DR_PROG_PROFILE`).

Depending on the type of derived table you customize derived tables as follows:

- Tables that hold the results of a calculation such as `DWD_ACCT_DEBT_DAY` table that contains values that are daily `DEBT_CNT` (Debt Count) and `DEBT_AMT` (Debt Amount) statistics about debt.
- Result tables for the data mining models (for example, `DWD_CUST_DR_PROG_PROFILE`).
- Creating New Derived Tables for Calculated Data (page 3-2)
  If, during fit-gap analysis, you identified a need for calculated data that is not provided by the default derived tables, you can meet this need by defining new tables or, alternatively, by adding missing dimensions and measures to existing derived tables.
- Customizing Oracle Utilities Data Model Data Mining Models (page 3-3)
  Some reference (`DWR_`) tables in the default `oudm_sys` schema are the results of data mining models defined in the default Oracle Utilities Data Model.

Related Topics:

- *Oracle Utilities Data Model Reference*

3.2.1 Creating New Derived Tables for Calculated Data

If, during fit-gap analysis, you identified a need for calculated data that is not provided by the default derived tables, you can meet this need by defining new tables or, alternatively, by adding missing dimensions and measures to existing derived tables.

When designing these tables, name the tables following the convention of using the `CWD_` prefix (for Customized Warehouse Derived) or `DWD_` (for Data Warehouse Derived). Make sure all the main dimensions are put first and have Foreign Keys to their corresponding reference or lookup tables. Attributes that add information only, avoiding costly joins, should not be part of the Primary Key of the derived table. Some dimensions that are part of a hierarchy do not necessarily need to be part of the Primary Key. All measures should be put afterward, grouped if possible by similar meaning.

Related Topics:

- *Oracle Utilities Data Model Reference*

3.2.2 Customizing Oracle Utilities Data Model Data Mining Models

Some reference (DWR_) tables in the default oudm_sys schema are the results of data mining models defined in the default Oracle Utilities Data Model.

Those models are defined in the default oudm_sys schema that also comes with Oracle Utilities Data Model For *Customer Savings and Customer Profile by DR program* data mining model a separate derived table, DWD_CUST_DR_PROG_PROFILE, is created. Mining code includes a script to populate this table from the following source tables:

- DWD_MTR_RDNG_DAY
- DWR_CUST
- DWR_DEMAND_RESPN_PROG
- DWR_USG_PNT_GRP_DR_PROG_ASGN
- DWR_USG_PNT_GRP
- DWR_USG_PNT_GRP_ASGN
- DWR_USG_PNT
- DWR_ACCT
- DWR_CUST_ACCT_ASGN

Oracle Utilities Data Model data mining models get source data from views defined on the following derived tables (DWD_):

- DWD_CUST_DR_PROG_PROFILE

Data mining models prediction results and model rules are stored in derived tables (DWD_) and reference tables (DWR_). They are:

- DWD_CUST_DR_PROG_PROFILE
- DWR_CUST_SGMNT
- DWR_CUST_SGMNT_DTL

All mining source scripts are copied to $ORACLE_HOME/oudm/pdm/mining directory when Oracle Utilities Data Model is installed. Table 3-1 (page 3-3) shows the Oracle Utilities Data Model mining scripts.

Table 3-1 Oracle Utilities Data Model Mining Scripts

Script Name	Description
oudm_mining_init.sql	Initializes mining environment and executes the other three mining scripts.
pkg_mining_etl.sql	Defines views, which have training/apply data, on source tables.
pkg_oudm_mining.sql	Core mining package that has a procedure for each model. Each procedure drops, creates mining model and scores mining model.
pkg_dwd_cust_dr_prog_profile.sql	A PL/SQL package to load data into mining source derived table DWD_CUST_DR_PROG_PROFILE.

When creating a customized Oracle Utilities Data Model warehouse, you can customize the data mining model in the following ways:

- Creating a New Data Mining Model for Oracle Utilities Data Model (page 3-4)
- Modifying Oracle Utilities Data Model Data Mining Models (page 3-4)

3.2.2.1 Creating a New Data Mining Model for Oracle Utilities Data Model

To create a data mining model:

1. Define the problem and identify input attributes. Also identify target attribute if the mining problem is supervised.
2. Check if the existing mining source views defined in `pkg_mining_etl.sql` script support the requirement of your problem. Modify the definition of views to support your requirement. Do not remove any columns from view definition unless you are sure that those columns do not make any sense.
3. If the existing mining source views do not support required fields, create a new table or view to support your requirements. Add the new table to `pkg_mining_etl.sql` PL/SQL package. Follow the naming conventions and use a `DWD_` prefix for results tables. Modify the intra-ETL programs to support your mining problem requirements.
4. For each mining problem that Oracle Data Mining supports, there is more than one algorithm. Create a setting table for your mining problem and follow the naming convention. The prefix for a setting table is "`DM_`". Add the definition of new setting table to `oudm_mining_init.sql` script.
5. Add a procedure for your mining problem to `pkg_oudm_mining` PL/SQL package. This procedure should create mining model and score the trained mining model on apply data. Compile the package. To create the mining model for your problem, invoke the newly added procedure. Make sure your new procedure works according to your expectations. Check `user_mining_models` data dictionary view for trained model. There are few more data dictionary views that give more information on the trained models.

Related Topics:

- *Oracle Data Mining User's Guide*
- Conventions When Customizing the Physical Model (page 2-5)
  When developing the physical model for Oracle Utilities Data Model, the naming conventions outlined were followed. Continue to follow these conventions as you customize the physical model.

3.2.2.2 Modifying Oracle Utilities Data Model Data Mining Models

To customize Oracle Utilities Data Model mining models, take the following steps:

1. Change the definition of source views used as input to the mining model.
2. If required, change the definition of source derived table, `DWD_CUST_DR_PROG_PROFILE`. Do not remove any existing columns. Only add new columns with NULL enable.
3. Modify the PL/SQL package of `DWD_CUST_DR_PROG_PROFILE` table. Execute the package to load data into `DWD_CUST_DR_PROG_PROFILE` table.

4. Refresh mining views by executing following statement. You need to pass demand response program key:

```
SQL> exec PKG_MINING_ETL.refresh_mining_views(l_dr_prog_key);
```

5. Train the model again by calling Oracle Utilities Data Model mining package.
6. Ensure that the model reflects the new definition (for example, that a new column has been added).

To add a new column to Customer Savings and Customer Profile by DR Program, take the following steps:

1. Add the new column to views that are used in the mining model.
   - `DWV_FRST_STEP_CUST_SGMNT_SRC`
2. Train the model by issuing the following statement:

```
pkg_oudm_mining. crt_frst_step_cust_sgmnt_model(l_dr_prog_key);
```

3. Execute the following statement to query model details table and ensure the new column name is included in the query result:

```
SQL> SELECT attribute_name
       FROM user_mining_model_attributes
       WHERE model_name = 'OUDM_PROFILE_KMEANS_' || l_dr_prog_key;
```

Related Topics:

- Refreshing the Data in an Oracle Utilities Data Model Warehouse (page 4-16)
  After the initial load, you must load new data into the Oracle Utilities Data Model data warehouse regularly so that it can serve its purpose of facilitating business analysis.

3.3 Aggregate Tables in Oracle Utilities Data Model

Aggregate tables are tables that aggregate or "roll up" the data to one level higher than a base or derived table (and other functions can also be in the aggregate tables such as average, count, min, max, and others).

The aggregate tables in the default Oracle Utilities Data Model are actually materialized views and have a `DWA_` prefix. These aggregate tables provide a view of the data similar to the view provided by a fact table in a snowflake schema.

The default Oracle Utilities Data Model defines several aggregate tables. For example, the `DWA_END_DVC_EVT_DVC_MO` table aggregates the values of the `DWD_END_DVC_EVT_DVC_DAY` table to the month level.

If, during fit-gap analysis, you identified a need for simple aggregated data that is not provided by the default aggregate tables, you can define new materialized views. When designing these tables, keep the following points in mind:

- Create a query for the materialized view that aggregates up only a single level. For example, if aggregating over time, then aggregate only from day to month.

> **Note:**
>
> When you must aggregate up many levels (for example in time, month, quarter, and year) or different hierarchies (for example, the fiscal and calendar hierarchies for a time dimension), do not define a `DWA_` table; instead, define the aggregations by creating OLAP cubes.

- Name the tables following the conventions and use a `DWA_` prefix.

Related Topics:

- Defining New Oracle OLAP Cubes for Oracle Utilities Data Model (page 3-14)
  You can add new OLAP cubes to the `oudm_sys` schema.
- Materialized Views in Oracle Utilities Data Model (page 3-19)
  Materialized views are query results that have been stored or "materialized" in advance as schema objects. From a physical design point of view, materialized views resemble tables or partitioned tables and behave like indexes in that they are used transparently and can improve performance.
- Conventions When Customizing the Physical Model (page 2-5)
  When developing the physical model for Oracle Utilities Data Model, the naming conventions outlined were followed. Continue to follow these conventions as you customize the physical model.
- *Oracle Utilities Data Model Reference*

3.4 Dimensional Components in Oracle Utilities Data Model

There is often much discussion regarding the 'best' modeling approach to take for any given data warehouse with each style, classic 3NF and dimensional having their own strengths and weaknesses.

It is likely that data warehouses must do more to embrace the benefits of each model type rather than rely on just one - this is the approach that was adopted in designing the Oracle Utilities Data Model. The foundation layer of the Oracle Utilities Data Model is a 3NF model. The default Oracle Utilities Data Model also provides a dimensional model of the data. This dimensional model of the data is a perspective that summarizes and aggregates data, rather than preserving detailed transaction information.

Familiarize yourself with dimensional modeling by reading the following topics before you begin to customize the dimensional model of the default Oracle Utilities Data Model

- Characteristics of a Dimensional Model (page 3-7)
  The simplicity of a dimensional model is inherent because it defines objects that represent real-world business entities. Analysts know which business measures they are interested in examining, which dimensions and attributes make the data meaningful, and how the dimensions of their business are organized into levels and hierarchies.
- Characteristics of Relational Star and Snowflake Tables (page 3-8)
  In the case of relational tables, the dimensional model has historically been implemented as a star or snowflake schema.
- Characteristics of the OLAP Dimensional Model (page 3-10)

- Characteristics of the OLAP Cubes in Oracle Utilities Data Model (page 3-14)
  The default access layer of Oracle Utilities Data Model provides a dimensional perspective of the data using Oracle OLAP cubes.
- Defining New Oracle OLAP Cubes for Oracle Utilities Data Model (page 3-14)
  You can add new OLAP cubes to the `oudm_sys` schema.
- Changing an Oracle OLAP Cube in Oracle Utilities Data Model (page 3-16)
  Common customizations to Oracle Utilities Data Model cubes are changing the dimensions or the measures of a cube.
- Creating a Forecast Cube for Oracle Utilities Data Model (page 3-16)
- Choosing a Cube Partitioning Strategy for Oracle Utilities Data Model (page 3-16)
- Choosing a Cube Data Maintenance Method for Oracle Utilities Data Model (page 3-18)
  While developing a dimensional model of your data, it is a good idea to map and load each object immediately after you create it so that you can immediately detect and correct any errors that you made to the object definition or the mapping.

3.4.1 Characteristics of a Dimensional Model

The simplicity of a dimensional model is inherent because it defines objects that represent real-world business entities. Analysts know which business measures they are interested in examining, which dimensions and attributes make the data meaningful, and how the dimensions of their business are organized into levels and hierarchies.

In the simplest terms, a dimensional model identifies the following objects:

- **Measures.** Measures store quantifiable business data (such as sales, expenses, and inventory). Measures are also called "facts". Measures are organized by one or more dimensions and may be stored or calculated at query time:
  - **Stored Measures**. Stored measures are loaded and stored at the leaf level. Commonly, there is also a percentage of summary data that is stored. Summary data that is not stored is dynamically aggregated when queried.
  - **Calculated Measures.** Calculated measures are measures whose values are calculated dynamically at query time. Only the calculation rules are stored in the database. Common calculations include measures such as ratios, differences, totals and moving averages. Calculations do not require disk storage space, and they do not extend the processing time required for data maintenance.
- **Dimensions.** A dimension is a structure that categorizes data to enable users to answer business questions. Commonly used dimensions are Customers, Products, and Time. A dimension's structure is organized hierarchically based on parent-child relationships. These relationships enable:
  - Navigation between levels.

    Hierarchies on dimensions enable drilling down to lower levels or navigation to higher levels (rolling up). Drilling down on the Time dimension member 2012 typically navigates you to the quarters Q1 2012 through Q4 2012. In a calendar year hierarchy for 2012, drilling down on Q1 2012 would navigate you to the months, January 12 through March 12. These kinds of relationships make it easy for users to navigate through large volumes of multidimensional data.

- Aggregation from child values to parent values.

  The parent represents the aggregation of its children. Data values at lower levels aggregate into data values at higher levels. Dimensions are structured hierarchically so that data at different levels of aggregation are manipulated efficiently for analysis and display.

- Allocation from parent values to child values.

  The reverse of aggregation is allocation and is heavily used by planning budgeting, and similar applications. Here, the role of the hierarchy is to identify the children and descendants of particular dimension members of "top-down" allocation of budgets (among other uses).

- Grouping of members for calculations.

  Share and index calculations take advantage of hierarchical relationships (for example, the percentage of total profit contributed by each product, or the percentage share of product revenue for a certain category, or costs as a percentage of the geographical region for a retail location).

A dimension object helps to organize and group dimensional information into hierarchies. This represents natural 1:n relationships between columns or column groups (the levels of a hierarchy) that cannot be represented with constraint conditions. Going up a level in the hierarchy is called rolling up the data and going down a level in the hierarchy is called drilling down the data.

There are two ways that you can implement a dimensional model:

- **Relational tables in a star schema configuration**.
- **Oracle OLAP Cubes**. The physical model provided with Oracle Utilities Data Model provides a dimensional perspective of the data using Oracle OLAP cubes.

Related Topics:

- Characteristics of Relational Star and Snowflake Tables (page 3-8)
  In the case of relational tables, the dimensional model has historically been implemented as a star or snowflake schema.
- Characteristics of the OLAP Dimensional Model (page 3-10)

3.4.2 Characteristics of Relational Star and Snowflake Tables

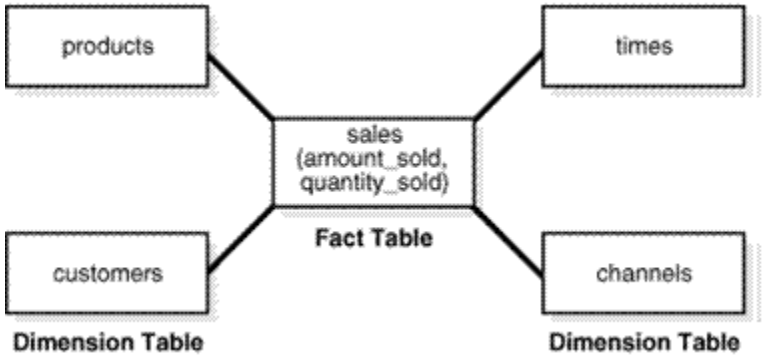
In the case of relational tables, the dimensional model has historically been implemented as a star or snowflake schema.

Dimension tables (which contain information about hierarchies, levels, and attributes) join to one or more fact tables. Fact tables are the large tables that store quantifiable business measurements (such as sales, expenses, and inventory) and typically have foreign keys to the dimension tables. Dimension tables, also known as lookup or reference tables, contain the relatively static or descriptive data in the data warehouse.

A star schema is a relational schema whose design represents a multidimensional data model. The star schema consists of one or more fact tables and one or more dimension tables that are related through foreign keys. This allows drill paths, hierarchy and query profile to be embedded in the data model itself rather than the data. This in part at least, is what makes navigation of the model so straightforward for end users. Star schemas usually have a large fact table surrounded by smaller dimension tables. Dimension tables do not change very much. Most of the information

that the users need are in the fact tables. Therefore, star schemas have fewer table joins than do 3NF models.

A star schema is so called because the diagram resembles a star, with points radiating from a center. The center of the star consists of one or more fact tables and the points of the star are the dimension tables.



Snowflake schemas are slight variants of a simple star schema where the dimension tables are further normalized and broken down into multiple tables. The snowflake aspect only affects the dimensions and not the fact table and is therefore considered conceptually equivalent to star schemas. Snowflake dimensions are useful and indeed necessary when there are fact tables of differing granularity. A month-level derived or aggregate table (or materialized view) must be associated with a month level snowflake dimension table rather than the default (lower) Day level star dimension table.

- Declaring Relational Dimension Tables (page 3-9)
  When a relational table acts as a dimension to a fact table, it is recommended that you declare that table as a dimension (even though it is not necessary).
- Validating Relational Dimension Tables (page 3-9)
  To improve the data quality of the dimension data in the data warehouse, it is recommended that you validate the declarative information about the relationships between the dimension members after any modification to the dimension data.

3.4.2.1 Declaring Relational Dimension Tables

When a relational table acts as a dimension to a fact table, it is recommended that you declare that table as a dimension (even though it is not necessary).

Defined dimensions can yield significant performance benefits, and support the use of more complex types of rewrite.

To define and declare the structure of the dimension use the `CREATE DIMENSION` command. Use the `LEVEL` clause to identify the names of the dimension levels.

3.4.2.2 Validating Relational Dimension Tables

To improve the data quality of the dimension data in the data warehouse, it is recommended that you validate the declarative information about the relationships between the dimension members after any modification to the dimension data.

To perform this validation, use the `VALIDATE_DIMENSION` procedure of the `DBMS_DIMENSION` package. When the `VALIDATE_DIMENSION` procedure encounters any errors, the procedure places the errors into the `DIMENSION_EXCEPTIONS` table. To find the exceptions identified by the `VALIDATE_DIMENSION` procedure, query the `DIMENSION_EXCEPTIONS` table.

You can schedule a call to the `VALIDATE_DIMENSION` procedure as a post-process step to the regular Incremental Dimension load script. This can be done before the call to refresh the derived or aggregate tables of the data model through materialized view refresh, intra-ETL package calls.

3.4.3 Characteristics of the OLAP Dimensional Model

Oracle OLAP Cubes logically represent data similar to relational star tables, although the data is actually stored in multidimensional arrays. Like dimension tables, cube dimensions organize members into hierarchies, levels, and attributes. The cube stores the measure (fact) data. The dimensions form the edges of the cube.

Oracle OLAP is an OLAP server embedded in the Oracle Database. Oracle OLAP provides native multidimensional storage and speed-of-thought response times when analyzing data across multiple dimensions. The database provides rich support for analytics such as time series calculations, forecasting, advanced aggregation with additive and nonadditive operators, and allocation operations.

By integrating multidimensional objects and analytics into the database, Oracle Database provides the best of both worlds: the power of multidimensional analysis along with the reliability, availability, security, and scalability of the Oracle database.

Oracle OLAP is fully integrated into Oracle Database. At a technical level, this means:

- The OLAP engine runs within the kernel of Oracle Database.
- Dimensional objects are stored in Oracle Database in their native multidimensional format.
- Cubes and other dimensional objects are first class data objects represented in the Oracle data dictionary.
- Data security is administered in the standard way, by granting and revoking privileges to Oracle Database users and roles.
- OLAP cubes, dimensions, and hierarchies are exposed to applications as relational views. Consequently, applications can query OLAP objects using SQL.
- Oracle OLAP cubes can be enhanced so that they are materialized views.

Benefits of Using Oracle OLAP

The benefits of using Oracle OLAP are significant; Oracle OLAP offers the power of simplicity and provides: One database, standard administration and security, standard interfaces and development tools.

The Oracle OLAP dimensional data model is highly structured. Structure implies rules that govern the relationships among the data and control how the data can be queried. Cubes are the physical implementation of the dimensional model, and thus are highly optimized for dimensional queries. The OLAP engine leverages this innate dimensionality in performing highly efficient cross-cube joins for inter-row calculations, outer joins for time series analysis, and indexing. Dimensions are pre-joined to the measures. The technology that underlies cubes is based on an indexed multidimensional array model, which provides direct cell access.

The OLAP engine manipulates dimensional objects in the same way that the SQL engine manipulates relational objects. However, because the OLAP engine is optimized to calculate analytic functions, and dimensional objects are optimized for analysis, analytic and row functions can be calculated much faster in OLAP than in SQL.

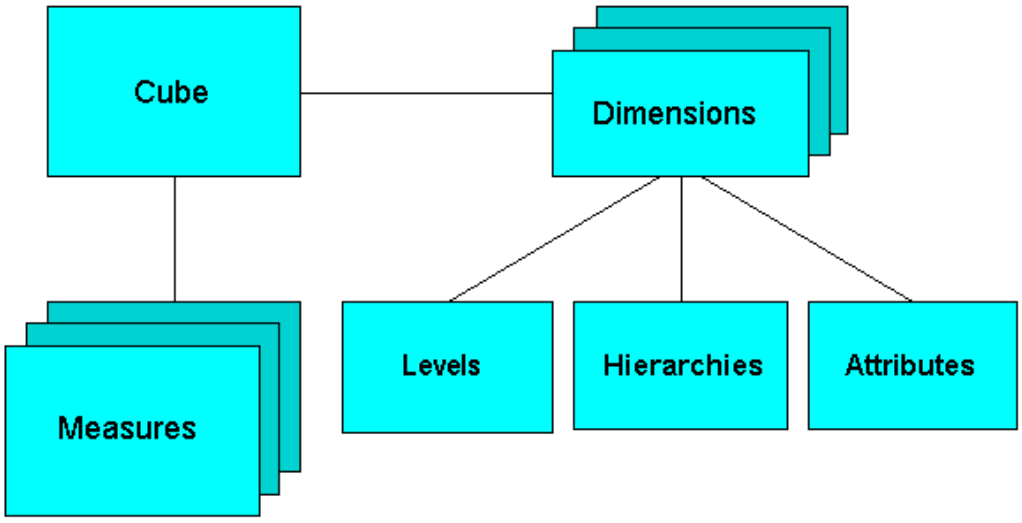
The dimensional model enables Oracle OLAP to support high-end business intelligence tools and applications such as OracleBI Discoverer Plus OLAP, OracleBI Spreadsheet Add-In, Oracle Business Intelligence Suite Enterprise Edition, BusinessObjects Enterprise, and Cognos ReportNet.

Oracle OLAP Dimensional Objects

Oracle OLAP dimensional objects include cubes, measures, dimensions, hierarchies, levels and attributes. Figure 3-1 (page 3-11) shows the general relationships among the objects.

> **See Also:**
>
> The OLAP dimensional objects are described in detail in *Oracle OLAP User's Guide*

Figure 3-1 Diagram of the OLAP Dimensional Model



- Oracle OLAP Cube Views (page 3-12)
- Cube Materialized Views (page 3-12)
  Oracle OLAP cubes can be enhanced so that they are materialized views. A cube that is enhanced in this way is called a cube materialized view and has a `CB$` prefix. Cube materialized views are incrementally refreshed through the Oracle Database materialized view subsystem, and serve as targets for transparent rewrite of queries against the source tables.

Related Topics:

- *Oracle OLAP User's Guide*
- Characteristics of the OLAP Cubes in Oracle Utilities Data Model (page 3-14) The default access layer of Oracle Utilities Data Model provides a dimensional perspective of the data using Oracle OLAP cubes.
- Oracle OLAP Cube Views (page 3-12)
- Report and Query Customization (page 5-1)

3.4.3.1 Oracle OLAP Cube Views

When you define an OLAP cube, Oracle OLAP automatically generates a set of relational views on the cube and its dimensions and hierarchies

- Cube view. Each cube has a cube view that presents the data for all the measures and calculated measures in the cube. You can use a cube view like a fact table in a star or snowflake schema. However, the cube view contains all the summary data in addition to the detail level data. The default name of a cube view is *cube*_VIEW.
- Dimension and hierarchy views. Each dimension has one dimension view plus a hierarchy view for each hierarchy associated with the dimension. The default name for a dimension view is *dimension*_VIEW. For a hierarchy view, the default name is *dimension_hierarchy*_VIEW.

These views are related in the same way as fact and dimension tables in a star schema. Cube views serve the same function as fact tables, and hierarchy views and dimension views serve the same function as dimension tables. Typical queries join a cube view with either a hierarchy view or a dimension view.

SQL applications query these views to display the information-rich contents of these objects to analysts and decision makers. You can also create custom views that follow the structure expected by your applications, using the system-generated views like base tables.

Related Topics:

- Report and Query Customization (page 5-1)

> **See Also:**
>
> The discussion on querying dimensional objects in *Oracle OLAP User's Guide*

3.4.3.2 Cube Materialized Views

Oracle OLAP cubes can be enhanced so that they are materialized views. A cube that is enhanced in this way is called a cube materialized view and has a CB$ prefix. Cube materialized views are incrementally refreshed through the Oracle Database materialized view subsystem, and serve as targets for transparent rewrite of queries against the source tables.

The OLAP dimensions associated with a cube materialized view are also defined with materialized view capabilities.

Necessary Cube Characteristics for Cube Materialized Views

A cube must conform to these requirements, before it can be designated as a cube materialized view:

- All dimensions of the cube have at least one level and one level-based hierarchy. Ragged and skip-level hierarchies are not supported. The dimensions must be mapped.
- All dimensions of the cube use the same aggregation operator, which is either `SUM`, `MIN`, or `MAX`.
- The cube has one or more dimensions and one or more measures.
- The cube is fully defined and mapped. For example, if the cube has five measures, then all five are mapped to the source tables.
- The data type of the cube is `NUMBER`, `VARCHAR2`, `NVARCHAR2`, or `DATE`.
- The source detail tables support dimension and rely constraints. If they are not defined, then use the Relational Schema Advisor to generate a script that defines them on the detail tables.
- The cube is compressed.
- The cube can be enriched with calculated measures, but it cannot support more advanced analytics in a cube script.

Adding Materialized View Capabilities

To add materialized view capabilities to an OLAP cube, perform the following steps:

1. In the Analytic Workspace Manager, connect to the `ocdm_sys` schema.
2. From the cube list, select the cube which you want to enable.
3. In the right pane, select the **Materialized Views** tab.
4. Select **Enable Materialized View Refresh of the Cube**. then click **Apply**.

> **Note:**
>
> You cannot enable the cube materialized view for a forecast cube.

> **See Also:**
>
> For more information on working with OLAP cubes, see the following OBE tutorials:
>
> - "Querying OLAP 11*g* Cubes"
> - "Using Oracle OLAP 11*g* With Oracle BI Enterprise Edition"
>
> To access the tutorial, open the Oracle Learning Library in your browser at Oracle Learning Library and, then, search for the tutorial by name.

Related Topics:

- *Oracle OLAP User's Guide*

3.4.4 Characteristics of the OLAP Cubes in Oracle Utilities Data Model

The default access layer of Oracle Utilities Data Model provides a dimensional perspective of the data using Oracle OLAP cubes.

There are OLAP cubes defined in the default `oudm_sys` schema. These cubes have the general characteristics described in "Characteristics of the OLAP Dimensional Model (page 3-10)". Specifically, OLAP cubes in the Oracle Utilities Data Model have the following characteristics:

- All of the default OLAP cubes are loaded with data from `DWA_` tables that are materialized views.
- The cubes were defined and built using the Analytical Workspace Manager (AWM) client tool.
- A relational view (with a `_VIEW` suffix) is defined over each of the OLAP cubes.
- All of the OLAP cubes in the Oracle Utilities Data Model are cube materialized views (that is, `CB$` objects).

> **Note:**
>
> Immediately after installation, all materialized views underlying the OLAP cubes are disabled by default.

For information on the using OLAP cubes in your customized version of Oracle Utilities Data Model, see *Oracle OLAP User's Guide* and the following topics:

- Defining New Oracle OLAP Cubes for Oracle Utilities Data Model (page 3-14)
- Changing an Oracle OLAP Cube in Oracle Utilities Data Model (page 3-16)
- Creating a Forecast Cube for Oracle Utilities Data Model (page 3-16)
- Choosing a Cube Partitioning Strategy for Oracle Utilities Data Model (page 3-16)
- Choosing a Cube Data Maintenance Method for Oracle Utilities Data Model (page 3-18)

3.4.5 Defining New Oracle OLAP Cubes for Oracle Utilities Data Model

You can add new OLAP cubes to the `oudm_sys` schema.

Take the following steps to define new cubes:

1. Ensure that there is an aggregate table (DWA_) to use as the "lowest leaf" data for the cube.
2. Use the AWM to define new Cubes for a customized version of Oracle Utilities Data Model. Follow the instructions given for creating cubes and dimensions in *Oracle OLAP User's Guide*.

Use the information provided in the *Oracle OLAP User's Guide* to guide you when you design and define new OLAP cubes. Also, if you are familiar with a relational star schema design as outlined in "Characteristics of Relational Star and Snowflake Tables (page 3-8)", then you can use this understanding to help you design an OLAP Cube:

- Fact tables correspond to cubes.
- Data columns in the fact tables correspond to measures.
- Foreign key constraints in the fact tables identify the dimension tables.
- Dimension tables identify the dimensions.
- Primary keys in the dimension tables identify the base-level dimension members.
- Parent columns in the dimension tables identify the higher level dimension members.
- Columns in the dimension tables containing descriptions and characteristics of the dimension members identify the attributes.

You can also get insights into the dimensional model by looking at the sample reports included with Oracle Utilities Data Model.

See:

Oracle Utilities Data Model Installation Guide for more information on installing the sample reports and deploying the Oracle Utilities Data Model RPD and webcat on the Business Intelligence Suite Enterprise Edition instance.

Tip:

While investigating your source data, you may decide to create relational views that more closely match the dimensional model that you plan to create.

3. Add materialized view capabilities to the OLAP cubes.

Note:

For more information on creating OLAP cubes, see the "Building OLAP 11*g* Cubes" OBE tutorial.

To access the tutorial, open the Oracle Learning Library in your browser at Oracle Learning Library and, then, search for the tutorial by name.

Related Topics:

- *Oracle OLAP User's Guide*

- Aggregate Tables in Oracle Utilities Data Model (page 3-5)
  Aggregate tables are tables that aggregate or "roll up" the data to one level higher than a base or derived table (and other functions can also be in the aggregate tables such as average, count, min, max, and others).
- Characteristics of the OLAP Dimensional Model (page 3-10)
- Characteristics of Relational Star and Snowflake Tables (page 3-8)
  In the case of relational tables, the dimensional model has historically been implemented as a star or snowflake schema.

3.4.6 Changing an Oracle OLAP Cube in Oracle Utilities Data Model

Common customizations to Oracle Utilities Data Model cubes are changing the dimensions or the measures of a cube.

All Oracle Utilities Data Model cubes load data from tables with the `DWA_` prefix, to change the measures or dimensions of one cube, you must take the following steps:

1. Use the information in Identify the DWA_ table from which the OLAP cube is populated.
2. Change the structure of the `DWA_` table identified in Step 1.
3. Change the OLAP cube and cube materialized views to reflect the new structure.

Related Topics:

- *Oracle Utilities Data Model Reference*

3.4.7 Creating a Forecast Cube for Oracle Utilities Data Model

To create a forecast cube for Oracle Utilities Data Model:

1. Create a cube to contain the results of the forecast.

> **Note:**
>
> You cannot enable materialized views for an Oracle Utilities Data Model forecast cube.

2. Write an OLAP DML forecasting context program.

Related Topics:

- Defining New Oracle OLAP Cubes for Oracle Utilities Data Model (page 3-14)
  You can add new OLAP cubes to the `oudm_sys` schema.
- *Oracle OLAP DML Reference*

3.4.8 Choosing a Cube Partitioning Strategy for Oracle Utilities Data Model

Partitioning is a method of physically storing the contents of a cube. It improves the performance of large cubes in the following ways:

- Improves scalability by keeping data structures small. Each partition functions like a smaller measure.
- Keeps the working set of data smaller both for queries and maintenance, since the relevant data is stored together.
- Enables parallel aggregation during data maintenance. Each partition can be aggregated by a separate process.
- Simplifies removal of old data from storage. Old partitions can be dropped, and new partitions can be added.

The number of partitions affects the database resources that can be allocated to loading and aggregating the data in a cube. Partitions can be aggregated simultaneously when sufficient resources have been allocated.

The Cube Partitioning Advisor analyzes the source tables and develops a partitioning strategy. You can accept the recommendations of the Cube Partitioning Advisor, or you can make your own decisions about partitioning.

If your partitioning strategy is driven primarily by life-cycle management considerations, then you should partition the cube on the Time dimension. Old time periods can then be dropped as a unit, and new time periods added as a new partition. The Cube Partitioning Advisor has a Time option, which recommends a hierarchy and a level in the Time dimension for partitioning.

The level on which to partition a cube is determined based on a trade off between load performance and query performance.

Typically, you do not want to partition on too low a level (for example, on the DAY level of a TIME dimension) because if you do then too many partitions must be defined at load time which slows down an initial or historical load. Also, a large number of partitions can result in unusually long Analytic Workspace attach times and slows down the Time Series-based calculations. Also, a Quarterly Cumulative measure (Quarter to Date Measure) needs to access 90 or 91 partitions to calculate a value for one Customer and Organization. All dimension members above the partition level of partition dimension (including those belonging to nondefault hierarchies) would be present in a single default template. Day level partitioning makes this very heavy since all higher level members are stored in default template. However, the advantage of partitioning `DAY` if the OLAP Cube load frequency is daily then there you must only load from a new partition in fact table into a single partition in the OLAP cube every day. This greatly improves the load performance since percentage-based refresh can be enabled if the cube is materialized-view enabled and has materialized-view logs.

Recommendations: Cube Partitioning Strategy

Usually a good compromise between the differing load and query performance requirements is to use an intermediate level like `MONTH` as the partition level. Time series calculations within a month (week to date, month to date, and so on) are fast and higher level calculations like year to date needs to refer to 12 partitions at most. Also this way the monthly partition is defined and created only one time (that is during the initial load on first of each month) and is then reused for each subsequent load that month. The aggregation process may be triggered off at the month level (instead of specific day level) and some redundant aggregations (of previously loaded dates of current month) may occur each time but it should result in satisfactory load and query performance.

Related Topics:

- Partitioning and Materialized Views (page 3-22)

- Indexes and Partitioned Indexes in Oracle Utilities Data Model (page 2-13) Indexes are optional structures associated with tables or clusters. In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.
- *Oracle OLAP User's Guide*

3.4.9 Choosing a Cube Data Maintenance Method for Oracle Utilities Data Model

While developing a dimensional model of your data, it is a good idea to map and load each object immediately after you create it so that you can immediately detect and correct any errors that you made to the object definition or the mapping.

However, in a production environment, you want to perform routine maintenance as quickly and easily as possible. For this stage, you can choose among data maintenance methods. You can refresh all cubes using the Maintenance Wizard. This wizard enables you to refresh a cube immediately, or submit the refresh as a job to the Oracle job queue, or generate a PL/SQL script. You can run the script manually or using a scheduling utility, such as Oracle Enterprise Manager Scheduler or the `DBMS_SCHEDULER` PL/SQL package. The generated script calls the `BUILD` procedure of the `DBMS_CUBE` PL/SQL package. You can modify this script or develop one from the start using this package.

The data for a partitioned cube is loaded and aggregated in parallel when multiple processes have been allocated to the build. You are able to see this in the build log.

In addition, each cube can support these data maintenance methods:

- Custom cube scripts
- Cube materialized views

If you are defining cubes to replace existing materialized views, then you use the materialized views as an integral part of data maintenance. Note, however, that materialized view capabilities restrict the types of analytics that can be performed by a custom cube script.

> **Note:**
>
> See the following OBE tutorial for an example of how Oracle uses cube materialized views for transparent access to a relational star schema:
>
> - "Querying OLAP 11*g* Cubes"
>
> To access the tutorial, open the Oracle Learning Library in your browser at Oracle Learning Library and, then, search for the tutorial by name.

Related Topics:

- Types of Materialized Views and Refresh Options (page 3-20)
- *Oracle OLAP User's Guide*

3.5 Materialized Views in Oracle Utilities Data Model

Materialized views are query results that have been stored or "materialized" in advance as schema objects. From a physical design point of view, materialized views resemble tables or partitioned tables and behave like indexes in that they are used transparently and can improve performance.

In the past, organizations using summaries spent a significant amount of time and effort creating summaries manually, identifying which summaries to create, indexing the summaries, updating them, and advising their users on which ones to use. With the advent of materialized views, a database administrator creates one or more materialized views, which are the equivalent of a summary. Thus, the workload of the database administrator is eased and the user no longer needed to be aware of the summaries that had been defined. Instead, the end user queries the tables and views at the detail data level. The query rewrite mechanism in the Oracle server automatically rewrites the SQL query to use the summary tables and reduces response time for returning results from the query.

Materialized views improve query performance by precalculating expensive join and aggregation operations on the database before executing and storing the results in the database. The query optimizer automatically recognizes when an existing materialized view can and should be used to satisfy a request.

The default Oracle Utilities Data Model defines many materialized views. In the default `oudm_sys` schema, you can identify these materialized views by looking at objects with the prefixes listed in the following table:

Prefix	Description
`DWA_`	Aggregate tables which are materialized views. See "Aggregate Tables in Oracle Utilities Data Model (page 3-5)" for more information on customizing these objects.
`CB$`	An OLAP cube enhanced with materialized view capabilities. See "Characteristics of the OLAP Cubes in Oracle Utilities Data Model (page 3-14)" for information on OLAP cubes. **Note:** Do `not` report or query against this object. Instead access the relational view of an OLAP cube (that is, the object with the `_VIEW` suffix).

The following topics provide more information on using and creating materialized views in your customized Oracle Utilities Data Model:

- Types of Materialized Views and Refresh Options (page 3-20)
- Choosing Indexes for Materialized Views (page 3-21)
- Partitioning and Materialized Views (page 3-22)
- Compressing Materialized Views (page 3-23)
  Using data compression for a materialized view can provide dramatic performance improvement.

Related Topics:

- *Oracle Utilities Data Model Reference*

3.5.1 Types of Materialized Views and Refresh Options

Refresh options vary by the type of materialized view:

Refresh Options for Materialized Views with Aggregates

In data warehouses, materialized views normally contain aggregates. The `DWA_` tables in the default Oracle Utilities Data Model are this type of materialized view.

For a materialized view with aggregates, for fast refresh to be possible:

- The `SELECT` list must contain all of the `GROUP BY` columns (if present).
- There must be a `COUNT(*)` and a `COUNT(`*`column`*`)` on any aggregated columns.
- Materialized view logs must be present on all tables referenced in the query that defines the materialized view. The valid aggregate functions are: `SUM`, `COUNT(`*`x`*`)`, `COUNT(*)`, `AVG`, `VARIANCE`, `STDDEV`, `MIN`, and `MAX`, and the expression to be aggregated can be any SQL value expression.

Fast refresh for a materialized view containing joins and aggregates is possible after any type of DML to the base tables (direct load or conventional `INSERT`, `UPDATE`, or `DELETE`).

You can define that the materialized view be refreshed `ON COMMIT` or `ON DEMAND`. A `REFRESH ON COMMIT` materialized view is automatically refreshed when a transaction that does DML to a materialized view's detail tables commits.

When you specify `REFRESH ON COMMIT`, the table commit can take more time than if you have not. This is because the refresh operation is performed as part of the commit process. Therefore, this method may not be suitable if many users are concurrently changing the tables upon which the materialized view is based.

Refresh Options for Materialized Views Containing Only Joins

Some materialized views contain only joins and no aggregates (for example, when a materialized view is created that joins the sales table to the times and customers tables).

The advantage of creating this type of materialized view is that expensive joins are precalculated.

Fast refresh for a materialized view containing only joins is possible after any type of DML to the base tables (direct-path or conventional `INSERT`, `UPDATE`, or `DELETE`).

A materialized view containing only joins can be defined to be refreshed `ON COMMIT` or `ON DEMAND`. If it is `ON COMMIT`, the refresh is performed at commit time of the transaction that does DML on the materialized view's detail table.

If you specify `REFRESH FAST`, Oracle Database performs further verification of the query definition to ensure that fast refresh can be performed if any of the detail tables change. These additional checks are:

- A materialized view log must be present for each detail table unless the table supports partition change tracking. Also, when a materialized view log is required, the `ROWID` column must be present in each materialized view log.
- The rowids of all the detail tables must appear in the `SELECT` list of the materialized view query definition.

If some of these restrictions are not met, you can create the materialized view as `REFRESH FORCE` to take advantage of fast refresh when it is possible. If one table does not meet all of the criteria, but the other tables do the materialized view is still fast refreshable with respect to the other tables for which all the criteria are met.

To achieve an optimally efficient refresh:

- Ensure that the defining query does not use an outer join that behaves like an inner join. If the defining query contains such a join, consider rewriting the defining query to contain an inner join.
- If the materialized view contains *only* joins, the `ROWID` columns for each table (and each instance of a table that occurs multiple times in the `FROM` list) must be present in the `SELECT` list of the materialized view.
- If the materialized view has remote tables in the `FROM` clause, all tables in the `FROM` clause must be located on that same site. Further, `ON COMMIT` refresh is not supported for materialized view with remote tables. Except for SCN-based materialized view logs, materialized view logs must be present on the remote site for each detail table of the materialized view and `ROWID` columns must be present in the`SELECT` list of the materialized view.

Refresh Options for Nested Materialized Views

A nested materialized view is a materialized view whose definition is based on another materialized view. A nested materialized view can reference other relations in the database in addition to referencing materialized views.

In a data warehouse, you typically create many aggregate views on a single join (for example, rollups along different dimensions). Incrementally maintaining these distinct materialized aggregate views can take a long time, because the underlying join has to be performed many times.

Using nested materialized views, you can create multiple single-table materialized views based on a joins-only materialized view and the join is performed just one time. In addition, optimizations can be performed for this class of single-table aggregate materialized view and thus refresh is very efficient.

Some types of nested materialized views cannot be fast refreshed. Use `EXPLAIN_MVIEW` to identify those types of materialized views.

You can refresh a tree of nested materialized views in the appropriate dependency order by specifying the `nested =TRUE` parameter with the `DBMS_MVIEW.REFRESH` parameter.

Related Topics:

- *Oracle OLAP User's Guide*

3.5.2 Choosing Indexes for Materialized Views

The two most common operations on a materialized view are query execution and fast refresh, and each operation has different performance requirements:

- Query execution might need to access any subset of the materialized view key columns, and might need to join and aggregate over a subset of those columns. Consequently, for best performance, create a single-column bitmap index on each materialized view key column.

- In the case of materialized views containing only joins using fast refresh, create indexes on the columns that contain the rowids to improve the performance of the refresh operation.
- If a materialized view using aggregates is fast refreshable, then an index appropriate for the fast refresh procedure is created unless `USING NO INDEX` is specified in the `CREATE MATERIALIZED VIEW` statement.

Related Topics:

- Indexes and Partitioned Indexes in Oracle Utilities Data Model (page 2-13)
  Indexes are optional structures associated with tables or clusters. In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.

3.5.3 Partitioning and Materialized Views

Because of the large volume of data held in a data warehouse, partitioning is an extremely useful option when designing a database. Partitioning the fact tables improves scalability, simplifies system administration, and makes it possible to define local indexes that can be efficiently rebuilt. Partitioning the fact tables also improves the opportunity of fast refreshing the materialized view because this may enable partition change tracking refresh on the materialized view.

Partitioning a materialized view has the same benefits as partitioning fact tables. When a materialized view is partitioned a refresh procedure can use parallel DML in more scenarios and partition change tracking-based refresh can use truncate partition to efficiently maintain the materialized view.

Using Partition Change Tracking

It is possible and advantageous to track freshness to a finer grain than the entire materialized view. The ability to identify which rows in a materialized view are affected by a certain detail table partition, is known as partition change tracking. When one or more of the detail tables are partitioned, it may be possible to identify the specific rows in the materialized view that correspond to a modified detail partition(s). those rows become stale when a partition is modified while all other rows remain fresh.

You can use partition change tracking to identify which materialized view rows correspond to a particular partition. Partition change tracking is also used to support fast refresh after partition maintenance operations on detail tables. For instance, if a detail table partition is truncated or dropped, the affected rows in the materialized view are identified and deleted. Identifying which materialized view rows are fresh or stale, rather than considering the entire materialized view as stale, allows query rewrite to use those rows that refresh while in `QUERY_REWRITE_INTEGRITY = ENFORCED` or `TRUSTED` modes.

Several views, such as `DBA_MVIEW_DETAIL_PARTITION`, detail which partitions are stale or fresh. Oracle does not rewrite against partial stale materialized views if partition change tracking on the changed table is enabled by the presence of join dependent expression in the materialized view.

To support partition change tracking, a materialized view must satisfy the following requirements:

- At least one detail table referenced by the materialized view must be partitioned.
- Partitioned tables must use either range, list or composite partitioning.

- The top level partition key must consist of only a single column.
- The materialized view must contain either the partition key column or a partition marker or `ROWID` or join dependent expression of the detail table.
- If you use a `GROUP BY` clause, the partition key column or the partition marker or `ROWID` or join dependent expression must be present in the `GROUP BY` clause.
- If you use an analytic window function or the `MODEL` clause, the partition key column or the partition marker or `ROWID` or join dependent expression must be present in their respective `PARTITION BY` subclauses.
- Data modifications can only occur on the partitioned table. If partition change tracking refresh is being done for a table which has join dependent expression in the materialized view, then data modifications should not have occurred in any of the join dependent tables.
- The `COMPATIBILITY` initialization parameter must be a minimum of `9.0.0.0.0`.
- Partition change tracking is not supported for a materialized view that refers to views, remote tables, or outer joins.

Related Topics:

- Indexes and Partitioned Indexes in Oracle Utilities Data Model (page 2-13)
  Indexes are optional structures associated with tables or clusters. In addition to the classical B-tree indexes, bitmap indexes are very common in data warehousing environments.
- Partitioned Tables in Oracle Utilities Data Model (page 2-14)
  Partitioning allows a table, index or index-organized table to be subdivided into smaller pieces. Each piece of the database object is called a partition.
- Choosing a Cube Partitioning Strategy for Oracle Utilities Data Model (page 3-16)
- *Oracle Database VLDB and Partitioning Guide*

3.5.4 Compressing Materialized Views

Using data compression for a materialized view can provide dramatic performance improvement.

Consider data compression when using highly redundant data, such as tables with many foreign keys. In particular, likely candidates are materialized views created with the `ROLLUP` clause.

Related Topics:

- Data Compression in Oracle Utilities Data Model (page 2-10)
  A key decision that you must make is whether to compress your data. Using table compression reduces disk and memory usage, often resulting in a better scale-up performance for read-only operations.
- Aggregate Tables in Oracle Utilities Data Model (page 3-5)
  Aggregate tables are tables that aggregate or "roll up" the data to one level higher than a base or derived table (and other functions can also be in the aggregate tables such as average, count, min, max, and others).

4
ETL Implementation and Customization

This chapter discusses the ETL (extraction, transformation and loading) programs you use to populate an Oracle Utilities Data Model warehouse. It includes the following topics:

- The Role of ETL in Oracle Utilities Data Model (page 4-1)
  Describes source-ETL and Intra-ETL for Oracle Utilities Data Model.
- ETL for the Foundation Layer of an Oracle Utilities Data Model Warehouse (page 4-3)
  ETL that populates the foundation layer of an Oracle Utilities Data Model warehouse (that is, the base, reference, and lookup tables) with data from an operational system is known as source-ETL.
- Customizing Intra-ETL for Oracle Utilities Data Model (page 4-11)
  The Oracle Utilities Data Model uses workflow implemented using PL/SQL packages to execute the intra-ETL process.
- Performing an Initial Load of an Oracle Utilities Data Model Warehouse (page 4-13)
  Performing an initial load of an Oracle Utilities Data Model is a multistep process with steps: :loading the foundation layer and loading the access layer.
- Refreshing the Data in an Oracle Utilities Data Model Warehouse (page 4-16)
  After the initial load, you must load new data into the Oracle Utilities Data Model data warehouse regularly so that it can serve its purpose of facilitating business analysis.
- Managing Errors During Oracle Utilities Data Model Intra-ETL Execution (page 4-23)
  Describes the steps to identify and manage errors during intra-ETL execution.

4.1 The Role of ETL in Oracle Utilities Data Model

Describes source-ETL and Intra-ETL for Oracle Utilities Data Model.

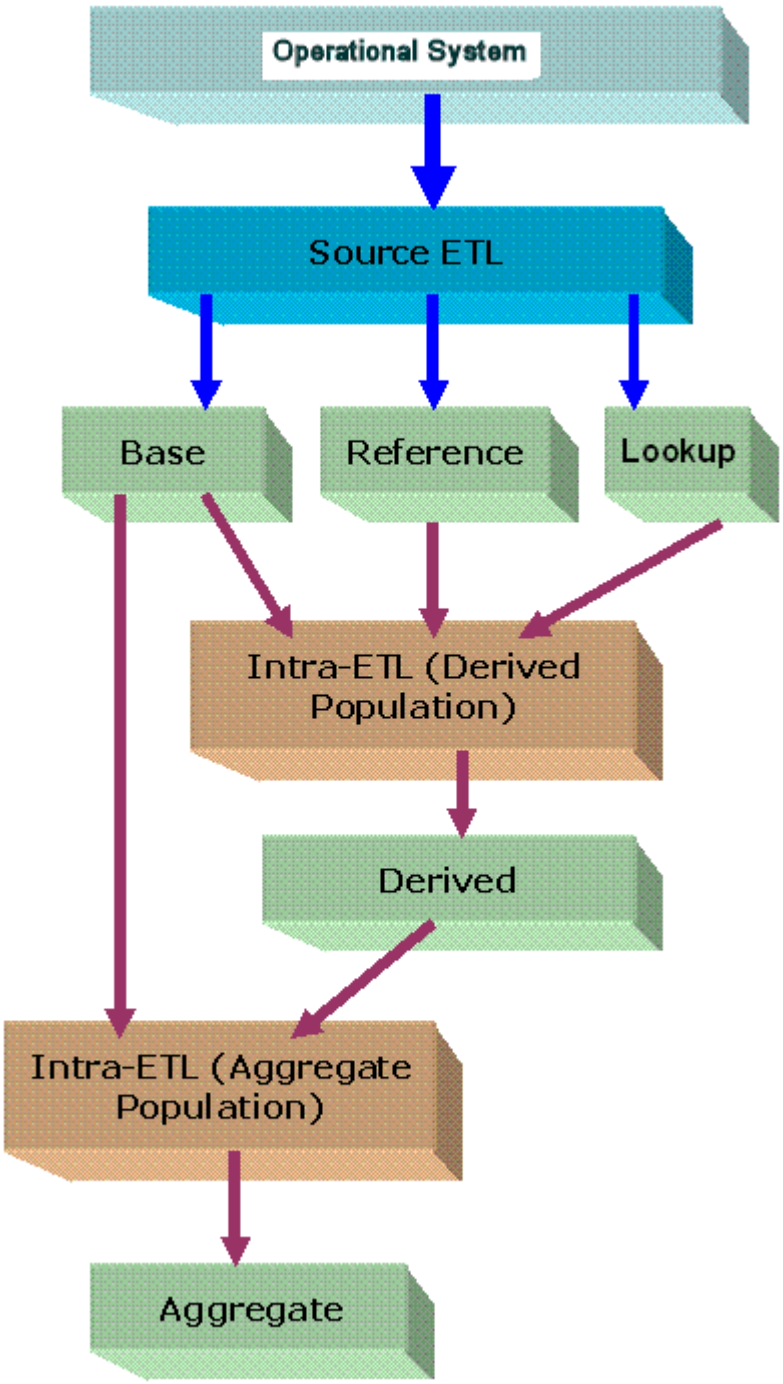
As illustrated in Figure 4-1 (page 4-2), you use two types of ETL (extraction, transformation and loading) to populate the staging layer, the foundation layer, and the access layer.

- **Source-ETL**. ETL that populates the staging layer (if any) and the foundation layer (that is, the base, reference, and lookup tables) with data from the operational system is known as source ETL.

  Oracle Utilities Data Model does *not* include source-ETL scripts. Unless you are using an application adapter for Oracle Utilities Data Model, you must create source-ETL yourself using your understanding of your operational and other source systems and your customized Oracle Utilities Data Model.
- **Intra-ETL**. ETL that populates the access layer (that is, the derived tables, aggregate tables, materialized views, OLAP cubes, and data mining models) using the data in the foundation layer is known as intra-ETL.

Oracle Utilities Data Model does include intra-ETL. You can modify the default intra-ETL to populate a customized access layer from a customized foundation layer.

Figure 4-1 ETL Flow Diagram



Related Topics:

- ETL for the Foundation Layer of an Oracle Utilities Data Model Warehouse (page 4-3)
  ETL that populates the foundation layer of an Oracle Utilities Data Model warehouse (that is, the base, reference, and lookup tables) with data from an operational system is known as source-ETL.
- Customizing Intra-ETL for Oracle Utilities Data Model (page 4-11)
  The Oracle Utilities Data Model uses workflow implemented using PL/SQL packages to execute the intra-ETL process.

4.2 ETL for the Foundation Layer of an Oracle Utilities Data Model Warehouse

ETL that populates the foundation layer of an Oracle Utilities Data Model warehouse (that is, the base, reference, and lookup tables) with data from an operational system is known as source-ETL.

You can populate the foundation layer of an Oracle Utilities Data Model warehouse in the following ways:

- Write your own source-ETL scripts using Oracle Data Integrator or another ETL tool and then use those scripts to populate the foundation layer.
- Writing Your Own Source-ETL (page 4-3)
- Source-ETL Design Considerations (page 4-4)
  Keep specific design features in mind when you design and write source-ETL for Oracle Utilities Data Model.
- ETL Architecture for Oracle Utilities Data Model Source-ETL (page 4-4)
- Creating a Source to Target Mapping Document for the Source-ETL (page 4-5)
- Designing a Plan for Rectifying Source-ETL Data Quality Problems (page 4-5)
- Designing Source-ETL Workflow and Jobs Control (page 4-6)
  All data movement among ETL processes are composed of jobs. An ETL workflow executes these jobs in the proper sequence and with the necessary dependencies. General ETL tools, such as Oracle Warehouse Builder, support this kind of workflow, job design, and execution control.
- Designing Source-ETL Exception Handling (page 4-7)
  As a general principle, all ETL logs status and errors into a table. You monitor execution status using an ETL tool or by querying this log table directly.
- Writing Source-ETL that Loads Efficiently (page 4-7)
  Whether you are developing mapping scripts and loading into a staging layer or directly into the foundation layer the goal is to get the data into the warehouse in the most expedient manner. In order to achieve good performance during the load you must begin by focusing on where the data to be loaded resides and how you load it into the database.

4.2.1 Writing Your Own Source-ETL

Using Oracle Utilities Data Model you must write your own source-ETL scripts using Oracle Data Integrator or another ETL tool or mapping tool.

The following topics provide general information about writing source-ETL:

Related Topics:

- *Developing Integration Projects with Oracle Data Integrator*

4.2.2 Source-ETL Design Considerations

Keep specific design features in mind when you design and write source-ETL for Oracle Utilities Data Model.

- You can populate the calendar data by using the calendar population scripts provided with Oracle Utilities Data Model
- Populate the tables in the following order:
  1. Lookup tables
  2. Reference tables
  3. Base tables
- Analyze the tables in one category before loading the tables in the next category (for example, analyze the reference tables before loading the lookup tables). Additionally, you must analyze all of the tables loaded by the source-ETL process before executing the intra-ETL processes).

> **See:**
>
> The topic about analyzing tables, indexes, and clusters in *Oracle Database Administrator's Guide*.

Related Topics:

- *Oracle Utilities Data Model Reference*

4.2.3 ETL Architecture for Oracle Utilities Data Model Source-ETL

ETL (or EL-T, that is, Extract, Load and Transform) first extracts data from the original sources, assures the quality of the data, cleans the data, and makes the data consistent across the original sources. ETL then populates the physical objects with the "clean" data so that query tools, report writers, dashboards and so on can access the data.

The fundamental services upon which data acquisition is constructed are as follows:

- Data sourcing
- Data movement
- Data transformation
- Data loading

From a logical architecture perspective, there are many different ways to configure these building blocks for delivering data acquisition services. The major architectural styles available that cover a range of options to be targeted within a data warehousing architecture include:

- **Batch Extract, Transform, and Load** and **Batch Extract, Load, Transform, Load**

  Batch Extract, Transform and Load (ETL) and Batch Extract, Load, Transform, Load (ELTL) are the traditional architecture's in a data warehouse implementation. The difference between them is where the transformation proceed in or out of the database.

- **Batch Hybrid Extract, Transform, Load, Transform, Load**

  Batch Hybrid Extract, Transform, Load, Transform, Load (ETLTL) is a hybrid strategy. This strategy provides the most flexibility to remove hand coding approaches to transformation design, apply a metadata-driven approach, and still be able to leverage the data processing capabilities of the enterprise warehouse. In this targeted design, the transformation processing is first performed outside the warehouse as a pre-processing step before loading the staging tables, and then further transformation processing is performed within the data warehouse before the final load into the target tables.

- **Real-time Extract, Transform, Load**

  Real-time Extract, Transform, Load (rETL) is appropriate when service levels for data freshness demand more up-to-date information in the data warehousing environment. In this approach, the OLTP system must actively publish events of interest so that the rETL processes can extract them from a message bus (queue) on a timely basis. A message-based paradigm is used with publish and subscribe message bus structures or point-to-point messaging with reliable queues. In such cases, the staging area can be used as a real-time Operational Data Store, at least for the source concerned, and aggregation could run directly from the Operational Data Store (operational system) to the Access layer, or to the presentation layer in specific cases.

When designing source-ETL for Oracle Utilities Data Model, use the architecture that best meets your business needs.

4.2.4 Creating a Source to Target Mapping Document for the Source-ETL

Before you begin building your extract systems, create a logical data interface document that maps the relationship between original source fields and target destination fields in the tables. This document ties the very beginning of the ETL system to the very end.

Columns in the data mapping document are sometimes combined. For example, the source database, table name, and column name could be combined into a single target column. The information within the concatenated column would be delimited with a period. Regardless of the format, the content of the logical data mapping document has been proven to be the critical element required to sufficiently plan ETL processes.

4.2.5 Designing a Plan for Rectifying Source-ETL Data Quality Problems

Data cleaning consists of all the steps required to clean and validate the data feeding a table and to apply known business rules to make the data consistent. The

perspectives of the cleaning and conforming steps are less about the upside potential of the data and more about containment and control.

There are several potential data quality issues, related to each other, that the staging area needs to handle:

- Data Validity: Is the data content and type sufficient to be usable, and as expected (and "profile" in case one uses this advanced option)?
- Data Accuracy: correct addresses, correct with respect some "true" standard (or as such defined).
- Data Completeness: is all the required data there? What to do when data is missing? What represents the minimum set of required data?
- Data Consistency: that is, consistency of the data between the various sources and what rules one applies for inconsistencies.
- Data Latency: A sub-part of data consistency, but treated separately because of its importance: when does data arrive, over which period and in which one can we combine, which one not?
- Data Reasoning: This is more at reporting level but can be applied at the staging level: Does the data I see make sense from a business perspective? Can I really combine the data as an end-user would expect?

As a consequence, a multi-layer staging is generally required or expected.

If there are data quality problems, then build a plan, in agreement with IT and business users, for how to rectify these problems.

Answer the following questions:

- Is data missing?
- Is the data wrong or inconsistent?
- Should the problem be fixed in the source systems?
- Set up the data quality reporting and action program and people responsibility.

Set up the following processes and programs:

- Set up a data quality measurement process.
- Set up the data quality reporting and action program and people responsibility.

4.2.6 Designing Source-ETL Workflow and Jobs Control

All data movement among ETL processes are composed of jobs. An ETL workflow executes these jobs in the proper sequence and with the necessary dependencies. General ETL tools, such as Oracle Warehouse Builder, support this kind of workflow, job design, and execution control.

Tips for designing ETL jobs and workflow:

- Use common structure across all jobs (source system to transformer to target data warehouse).
- Have a one-to-one mapping from source to target.
- Define one job per Source table.

- Apply generic job structure and template jobs to allow for rapid development and consistency.
- Use an optimized job design to leverage Oracle load performance based on data volumes.
- Design parameterized job to allow for greater control over job performance and behavior.
- Maximize Jobs parallelism execution.

4.2.7 Designing Source-ETL Exception Handling

As a general principle, all ETL logs status and errors into a table. You monitor execution status using an ETL tool or by querying this log table directly.

Your ETL tool or your developed mapping scripts generate status and error handling tables.

As a general principle, all ETL logs status and errors into a table. You monitor execution status using an ETL tool or by querying this log table directly.

4.2.8 Writing Source-ETL that Loads Efficiently

Whether you are developing mapping scripts and loading into a staging layer or directly into the foundation layer the goal is to get the data into the warehouse in the most expedient manner. In order to achieve good performance during the load you must begin by focusing on where the data to be loaded resides and how you load it into the database.

For example, you should not use a serial database link or a single JDBC connection to move large volumes of data. The most common and preferred mechanism for loading large volumes of data is loading from flat files.

The following topics discuss best practices for ensuring your source-ETL loads efficiently:

- Using a Staging Area for Flat Files (page 4-8)
  The area where flat files are stored before being loaded into the staging layer of a data warehouse system is commonly known as staging area.
- Preparing Raw Data Files for Source-ETL (page 4-8)
  In order to parallelize the data load Oracle Database must be able to logically break up the raw data files into chunks, known as granules. To ensure balanced parallel processing, the number of granules is typically much higher than the number of parallel server processes. At any given point in time, a parallel server process is allocated one granule to work on. After a parallel server process completes working on its granule, another granule is allocated until all of the granules are processed and the data is loaded.
- Source-ETL Data Loading Options (page 4-9)
  Oracle offers several Source-ETL data loading options.
- Parallel Direct Path Load Source-ETL (page 4-9)
  A direct path load parses the input data according to the description given in the external table definition, converts the data for each input field to its corresponding Oracle data type, then builds a column array structure for the data. These column array structures are used to format Oracle data blocks and build index keys.

- Partition Exchange Load for Oracle Utilities Data Model Source-ETL (page 4-10) A benefit of partitioning is the ability to load data quickly and easily with minimal impact on the business users by using the `EXCHANGE PARTITION` command.

Related Topics:

- *Oracle Database SecureFiles and Large Objects Developer's Guide*

4.2.8.1 Using a Staging Area for Flat Files

The area where flat files are stored before being loaded into the staging layer of a data warehouse system is commonly known as staging area.

The overall speed of your load is determined by:

- How quickly the raw data can be read from staging area.
- How quickly the raw data can be processed and inserted into the database.

Recommendations: Using a Staging Area

Stage the raw data across as many physical disks as possible to ensure that reading it is not a bottleneck during the load.

Also, if you are using the Exadata Database Machine, the best place to stage the data is in an Oracle Database File System (DBFS) stored on the Exadata storage cells. DBFS creates a mountable cluster file system which can you can use to access files stored in the database. Create the DBFS in a separate database on the Database Machine. This allows the DBFS to be managed and maintained separately from the data warehouse.

Mount the file system using the `DIRECT_IO` option to avoid thrashing the system page cache while moving the raw data files in and out of the file system.

4.2.8.2 Preparing Raw Data Files for Source-ETL

In order to parallelize the data load Oracle Database must be able to logically break up the raw data files into chunks, known as granules. To ensure balanced parallel processing, the number of granules is typically much higher than the number of parallel server processes. At any given point in time, a parallel server process is allocated one granule to work on. After a parallel server process completes working on its granule, another granule is allocated until all of the granules are processed and the data is loaded.

Recommendations: Preparing Raw Data Files for Source-ETL

Follow these recommendations:

- Deliminate each row using a known character such as a new line or a semicolon. This ensures that Oracle can look inside the raw data file and determine where each row of data begins and ends in order to create multiple granules within a single file.
- If a file is not position-able and seek-able (for example the file is compressed or zip file), then the files cannot be broken up into granules and the whole file is treated as a single granule. In this case, only one parallel server process can work on the entire file. In order to parallelize the loading of compressed data files, use multiple compressed data files. The number of compressed data files used determines the maximum parallel degree used by the load.

- When loading multiple data files (compressed or uncompressed):
  - Use a single external table, if at all possible
  - Make the files similar in size
  - Make the size of the files a multiple of 10 MB
- If you must have files of different sizes, list the files from largest to smallest. By default, Oracle assumes that the flat file has the same character set as the database. If this is not the case, specify the character set of the flat file in the external table definition to ensure the proper character set conversions can take place.

4.2.8.3 Source-ETL Data Loading Options

Oracle offers several Source-ETL data loading options.

Oracle offers several data loading options

- External table or SQL*Loader
- Oracle Data Pump (import and export)
- Change Data Capture and Trickle feed mechanisms (such as Oracle GoldenGate)
- Oracle Database Gateways to open systems and mainframes
- Generic Connectivity (ODBC and JDBC)

The approach that you take depends on the source and format of the data you receive.

Recommendations: Loading Flat Files

If you are loading from files into Oracle you have two options: SQL*Loader or external tables.

Using external tables offers the following advantages:

- Allows transparent parallelization inside the database.You can avoid staging data and apply transformations directly on the file data using arbitrary SQL or PL/SQL constructs when accessing external tables. SQL Loader requires you to load the data as-is into the database first.
- Parallelizing loads with external tables enables a more efficient space management compared to SQL*Loader, where each individual parallel loader is an independent database sessions with its own transaction. For highly partitioned tables this could potentially lead to a lot of wasted space.

You can create an external table using the standard `CREATE TABLE` statement. However, to load from flat files the statement must include information about where the flat files reside outside the database. The most common approach when loading data from an external table is to issue a `CREATE TABLE AS SELECT (CTAS)` statement or an `INSERT AS SELECT (IAS)` statement into an existing table.

4.2.8.4 Parallel Direct Path Load Source-ETL

A direct path load parses the input data according to the description given in the external table definition, converts the data for each input field to its corresponding Oracle data type, then builds a column array structure for the data. These column array structures are used to format Oracle data blocks and build index keys.

The newly formatted database blocks are then written directly to the database, bypassing the standard SQL processing engine and the database buffer cache.

The key to good load performance is to use direct path loads wherever possible:

- A `CREATE TABLE AS SELECT (CTAS)` statement always uses direct path load.
- A simple `INSERT AS SELECT (IAS)` statement does *not* use direct path load. In order to achieve direct path load with an IAS statement you must add the `APPEND` hint to the command.

Direct path loads can also run in parallel. To set the parallel degree for a direct path load, either:

- Add the `PARALLEL` hint to the CTAS statement or an IAS statement.
- Set the `PARALLEL` clause on both the external table and the table into which the data is loaded.

  After the parallel degree is set:

  – A `CTAS` statement automatically performs a direct path load in parallel.

  – An `IAS` statement does not automatically perform a direct path load in parallel. In order to enable an `IAS` statement to perform direct path load in parallel, you must alter the session to enable parallel DML by executing the following statement:

  ```
  alter session enable parallel DML;
  ```

4.2.8.5 Partition Exchange Load for Oracle Utilities Data Model Source-ETL

A benefit of partitioning is the ability to load data quickly and easily with minimal impact on the business users by using the `EXCHANGE PARTITION` command.

The `EXCHANGE PARTITION` command enables swapping the data in a nonpartitioned table into a particular partition in your partitioned table. The `EXCHANGE PARTITION` command does not physically move data, instead it updates the data dictionary to exchange a pointer from the partition to the table and vice versa.

Because there is no physical movement of data, an exchange does not generate redo and undo. In other words, an exchange is a sub-second operation and far less likely to impact performance than any traditional data-movement approaches such as `INSERT`.

Recommendations: Partitioning Tables

Partition the larger tables and fact tables in the Oracle Utilities Data Model warehouse.

Example 4-1 Using Exchange Partition Statement with a Partitioned Table

Assume that there is a large table called `Sales`, which is range partitioned by day. At the end of each business day, data from the online sales system is loaded into the `Sales` table in the warehouse.

The following steps ensure the daily data gets loaded into the correct partition with minimal impact to the business users of the data warehouse and optimal speed:

1. Create external table for the flat file data coming from the online system
2. Using a `CTAS` statement, create a nonpartitioned table called `tmp_sales` that has the same column structure as `Sales` table

3. Build any indexes that are on the `Sales` table on the `tmp_sales` table
4. Issue the `EXCHANGE PARTITION` command.

```
Alter table Sales exchange partition p2 with
    table top_sales including indexes without validation;
```

5. Gather optimizer statistics on the newly exchanged partition using incremental statistics.

The `EXCHANGE PARTITION` command in this example, swaps the definitions of the named partition and the `tmp_sales` table, so the data instantaneously exists in the right place in the partitioned table. Moreover, with the inclusion of the `INCLUDING INDEXES` and `WITHOUT VALIDATION` clauses, Oracle swaps index definitions and does not check whether the data actually belongs in the partition - so the exchange is very quick.

> **Note:**
>
> The assumption being made in this example is that the data integrity was verified at date extraction time. If you are unsure about the data integrity, omit the `WITHOUT VALIDATION` clause so that the Database checks the validity of the data.

4.3 Customizing Intra-ETL for Oracle Utilities Data Model

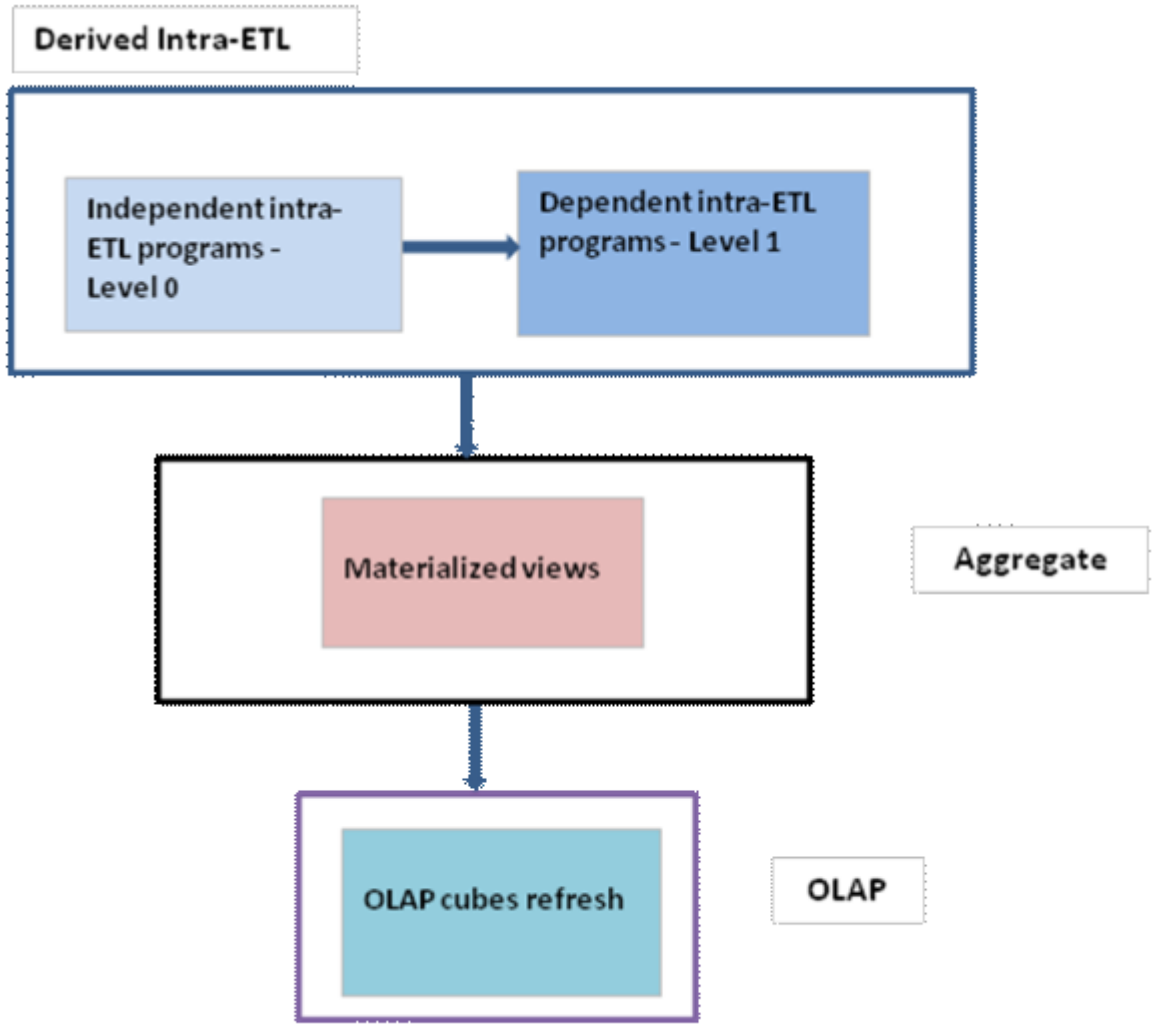
The Oracle Utilities Data Model uses workflow implemented using PL/SQL packages to execute the intra-ETL process.

The workflow consists of three major components:

1. **Executing Derived Intra-ETL Programs**
   a. Independent Derived intra-ETL programs - Level 0
   b. First level dependent Derived intra-ETL programs - Level 1
2. **Refreshing Aggregate Materialized Views**:
3. **Refreshing OLAP Cubes**

Figure 4-2 (page 4-12) illustrates the Oracle Utilities Data Model intra-ETL workflow.

Figure 4-2 Oracle Utilities Data Model Intra-ETL Workflow



- Executing Derived Intra-ETL Programs (page 4-12)
- Refreshing Aggregate Materialized Views (page 4-13)
- Refreshing OLAP Cubes (page 4-13)
- Executing Intra-ETL Workflow (page 4-13)
  Oracle Utilities Data Model intra-ETL workflow is implemented using a PL/SQL package, `PKG_INTRA_ETL_PROCESS`.

4.3.1 Executing Derived Intra-ETL Programs

The first component of the workflow to execute the intra-ETL process is the **Derived intra-ETL programs**. This component has two subcomponents to deal with the dependency among Derived intra-ETL programs:

- The first subcomponent, *Independent Derived intra-ETL programs*, has Derived intra-ETL programs that get data from foundation layer tables, that is, base, lookup, and reference tables.
- The second subcomponent has Derived intra-ETL programs that depend on the first subcomponent, *Independent Derived intra-ETL programs*. The second subcomponent intra-ETL programs get data from foundation layer tables, that is, base, lookup, and reference tables and also from derived tables that have intra-ETL programs in first subcomponent.

The Intra-ETL programs in the two subcomponents are implemented using PL/SQL packages. All Intra-ETL packages insert data for the ETL period mentioned in `DWC_ETL_PARAMTER` table for "`OUDM-INTRA-ETL`" process. Modify the ETL period of the process as per the data load requirements. If you are trying to load data for ETL period, for which data is already loaded, intra-ETL program first truncates the partitions existing for the ETL period, and then loads data into the target derived table.

4.3.2 Refreshing Aggregate Materialized Views

This is the second component of the workflow. This component depends on the first component, *Executing Derived intra-ETL programs*. The execution of this component happens only when the execution of the first component completes successfully.

4.3.3 Refreshing OLAP Cubes

This is the third component of the workflow. This component depends on the second component, *Refreshing Aggregate materialized views*, which in turn depends on the first component, *Executing Derived intra-ETL programs*. The execution of this component happens only when the execution of the second component completes successfully. This component refreshes data in OLAP cubes and dimensions based on the parameters given in `DWC_OLAP_ETL_PARAMETER` table.

4.3.4 Executing Intra-ETL Workflow

Oracle Utilities Data Model intra-ETL workflow is implemented using a PL/SQL package, `PKG_INTRA_ETL_PROCESS`.

Each component and their subcomponents of intra-ETL workflow have one procedure each. All these procedures are private to the package. The package has only one public procedure, which invokes all private procedures as depicted in the workflow figure. Before executing the workflow, ensure that you set all ETL parameters in `DWC_OLAP_PARAMTER` and `DWC_OLAP_ETL_PARAMETER` tables. Invoking `PKG_INTRA_ETL_PROCESS.RUN` procedure starts the workflow execution.

4.4 Performing an Initial Load of an Oracle Utilities Data Model Warehouse

Performing an initial load of an Oracle Utilities Data Model is a multistep process with steps: :loading the foundation layer and loading the access layer.

Performing an initial load of an Oracle Utilities Data Model is a multistep process:

1. Load the foundation layer of the Oracle Utilities Data Model warehouse (that is, the reference, lookup, and base tables) .

2. Load the access layer of the Oracle Utilities Data Model warehouse (that is, the derived and aggregate tables, materialized views, OLAP cubes, and data mining models).

- Performing an Initial Load of the Foundation Layer (page 4-14)
- Performing an Initial Load of the Access Layer (page 4-14)
- Executing the Default Oracle Utilities Data Model Intra-ETL (page 4-15)
  Intra-ETL workflow is implemented using PL/SQL package, `PKG_INTRA_ETL_PROCESS`. This package has a public procedure, `Run`, and also has private procedures for executing derived intra-ETL programs, refreshing aggregate materialized views, and refreshing OLAP cubes.

4.4.1 Performing an Initial Load of the Foundation Layer

Perform initial load of the foundation layer using source-ETL that you create.

Related Topics:

- Writing Your Own Source-ETL (page 4-3)

4.4.2 Performing an Initial Load of the Access Layer

To perform an initial load of access layer of the Oracle Utilities Data Model warehouse (that is, the derived and aggregate tables, materialized views, OLAP cubes) take the following steps:

1. Update the parameters in `DWC_ETL_PARAMETER` control table in the `oudm_sys` schema for different processes so that the ETL can use this information (that is, the beginning and end date of the ETL period) when loading the derived and aggregate tables and views.

   For an initial load of an Oracle Utilities Data Model warehouse, specify the values shown in the following tables:

   For *OUDM-INTRA-ETL* process:

   Table 4-1 Initial Load Values for DWC_ETL_PARAMETER Table

Columns	Value
`PROCESS_NAME`	`'OUDM-INTRA-ETL'`
`FROM_DATE_ETL`	The beginning date of the ETL period.
`TO_DATE_ETL`	The ending date of the ETL period.

2. Update the Oracle Utilities Data Model OLAP ETL parameters in `DWC_OLAP_ETL_PARAMETER` control table in the `oudm_sys` schema to specify the build method and other build characteristics so that the ETL can use this information when loading the OLAP cube data.

   For an initial load of the analytic workspace, specify values following the guidelines in Table 4-2 (page 4-15).

Table 4-2 Values of OLAP ETL Parameters in the DWC_OLAP_ETL_PARAMETER table for Initial Load

Column Name	Value
PROCESS_NAME	'OUDM-OLAP-ETL'
BUILD_METHOD	C which specifies a complete refresh which clears all dimension values before loading.
CUBENAME	One of the following values that specifies the cubes you want to build: • ALL specifies a build of the cubes in the Oracle Utilities Data Model analytic workspace. • cubename[[|cubename]...] specifies one or more cubes to build.
MAXJOBQUEUES	A decimal value that specifies the number of parallel processes to allocate to this job. (Default value is 4.) The value that you specify varies depending on the setting of the JOB_QUEUE_PROCESSES database initialization parameter.
CALC_FCST	One of the following values depending on whether you want to calculate forecast cubes: • Y specifies calculate forecast cubes. • N specifies do not calculate forecast cubes.
NO_FCST_YRS	If the value for the CALC_FCST column is Y, specify a decimal value that specifies how many years forecast data you want to calculate; otherwise, specify NULL.
FCST_MTHD	If the value for the CALC_FCST column is Y, then specify AUTO; otherwise, specify NULL.
FCST_ST_YR	If the value for the CALC_FCST column is Y, then specify value specified as 'BY YYYY' which is the "start business year" of a historical period; otherwise, specify NULL.
FCST_END_YR	If the value for the CALC_FCST column is Y, then specify value specified as 'BY YYYY' which is the "end business year" of a historical period; otherwise, specify NULL.
OTHER1	Specify NULL.
OTHER2	Specify NULL.

3. Execute the intra-ETL.

Related Topics:

- *Oracle Utilities Data Model Reference*
- Executing the Default Oracle Utilities Data Model Intra-ETL (page 4-15)
  Intra-ETL workflow is implemented using PL/SQL package, PKG_INTRA_ETL_PROCESS. This package has a public procedure, Run, and also has private procedures for executing derived intra-ETL programs, refreshing aggregate materialized views, and refreshing OLAP cubes.

4.4.3 Executing the Default Oracle Utilities Data Model Intra-ETL

Intra-ETL workflow is implemented using PL/SQL package, PKG_INTRA_ETL_PROCESS. This package has a public procedure, Run, and also has private procedures for executing derived intra-ETL programs, refreshing aggregate materialized views, and refreshing OLAP cubes.

The public procedure, Run, invokes all private procedures.

Before executing intra-ETL workflow, update ETL parameters in DWC_ETL_PARAMETER and DWC_OLAP_ETL_PARAMETER tables. It is suggested to use oudm_user user to update ETL parameter tables and executing intra-ETL workflow. Oracle Utilities Data Model installation creates oudm_user and grants roles and privileges required for updating ETL parameter tables and executing intra-ETL workflow.

```
sqlplus oudm_user/oudm_user@SID
```

Update ETL parameter tables:

```
SQL> UPDATE DWC_ETL_PARAMETER
SET from_date_etl = < The beginning date of the ETL period >,
    to_date_etl   = < The ending date of the ETL period >
WHERE process_name = 'OUDM-INTRA-ETL'
;
/
SQL> commit;

SQL> UPDATE DWC_OLAP_ETL_PARAMETER
SET build_method = <>,
    cubename   = <>,
      .
  .
  .
  .
fcst_st_yr = <>,
fcst_end_yr = <>
;
/
SQL> commit;
```

Run the following command to execute intra-ETL workflow:

```
SQL> BEGIN
OUDM_SYS.PKG_INTRA_ETL_PROCESS.Run;
END;
/
```

The status of each activity is tracked using `DWC_INTRA_ETL_ACTIVITY` table. The status of each cube data loading is tracked using `DWC_OLAP_ACTIVITY` table. The status of the entire intra-ETL workflow process is tracked using `DWC_INTRA_ETL_PROCESS` table.

Related Topics:

- Accounts Created for Oracle Utilities Data Model (page 7-1)
  Installing the Oracle Utilities Data Model component creates the accounts: `oudm_sys`, `oudm_user`, and `oudm_report`. Installing the Oracle Utilities Data Model sample reports creates the `oudm_sample` account. Oracle Utilities Data Model installation creates database accounts with UNLOCK and PASSWORD EXPIRE. Ensure that you unlock these accounts and set new passwords following the post-installation steps.
- Monitoring the Execution of the Intra-ETL Process (page 4-23)
  Three `oudm_sys` schema control tables, `DWC_INTRA_ETL_PROCESS`, `DWC_INTRA_ETL_ACTIVITY`, `DWC_OLAP_ACTIVITY` monitor the execution of the intra-ETL process.

4.5 Refreshing the Data in an Oracle Utilities Data Model Warehouse

After the initial load, you must load new data into the Oracle Utilities Data Model data warehouse regularly so that it can serve its purpose of facilitating business analysis.

To load new data into Oracle Utilities Data Model warehouse, you extract the data from one or more operational systems and copy that data into the warehouse. The challenge in data warehouse environments is to integrate, rearrange and consolidate large volumes of data over many systems, thereby providing a new unified information base for business intelligence.

The successive loads and transformations must be scheduled and processed in a specific order that is determined by your business needs. Depending on the success or failure of the operation or parts of it, the result must be tracked and subsequent, alternative processes might be started.

You can do a full incremental load of the Oracle Utilities Data Model warehouse, or you can refresh the data sequentially.

- Refreshing the Foundation Layer of Oracle Utilities Data Model Warehouse (page 4-17)
- Refreshing the Access Layer of an Oracle Utilities Data Model Warehouse (page 4-18)
- Refreshing Oracle Utilities Data Model Derived Tables (page 4-18)
- Refreshing Oracle Utilities Data Model Aggregate Materialized Views (page 4-19)
- Refreshing Oracle Utilities Data Model OLAP Cubes (page 4-20)
  On a scheduled basis you must update the OLAP cube data with the relational data that has been added to the Oracle Utilities Data Model data warehouse since the initial load of the OLAP cubes.
- Refreshing Oracle Utilities Data Model Data Mining Models (page 4-22)
  Refreshing of data mining models is *not* integrated into intra-ETL workflow. There is only one data mining model, Customer Savings and Customer Profile by DR Program.

Related Topics:

- Performing an Initial Load of an Oracle Utilities Data Model Warehouse (page 4-13)
  Performing an initial load of an Oracle Utilities Data Model is a multistep process with steps: :loading the foundation layer and loading the access layer.
- **Managing Errors During Oracle Utilities Data Model Intra-ETL Execution** (page 4-23)
  Describes the steps to identify and manage errors during intra-ETL execution.

4.5.1 Refreshing the Foundation Layer of Oracle Utilities Data Model Warehouse

You can refresh the foundation layer of an Oracle Utilities Data Model warehouse (that is, the reference, lookup, and base tables) in the following ways:

- You can refresh the foundation layer using source-ETL scripts that you wrote using Oracle Warehouse Builder or another ETL tool. For more information on creating source-ETL, see "Writing Your Own Source-ETL (page 4-3)".

4.5.2 Refreshing the Access Layer of an Oracle Utilities Data Model Warehouse

Refreshing the access layer of an Oracle Utilities Data Model is a multi-step process. You can do a full incremental load of the access layer all at one time, or you can refresh the data sequentially, as follows:

1. Refreshing Oracle Utilities Data Model Derived Tables
2. Refreshing Oracle Utilities Data Model Aggregate Materialized Views
3. Refreshing Oracle Utilities Data Model OLAP Cubes

Related Topics:

- Refreshing Oracle Utilities Data Model Derived Tables (page 4-18)
- Refreshing Oracle Utilities Data Model Aggregate Materialized Views (page 4-19)
- Refreshing OLAP Cubes (page 4-13)
- **Managing Errors During Oracle Utilities Data Model Intra-ETL Execution** (page 4-23)
  Describes the steps to identify and manage errors during intra-ETL execution.

4.5.3 Refreshing Oracle Utilities Data Model Derived Tables

Refreshing the relational tables in an Oracle Utilities Data Model is a multi-step process:

1. Refresh the foundation layer of the Oracle Utilities Data Model warehouse (that is, the reference, lookup, and base tables) with operational system data by executing the source-ETL that you have written.
2. Update the parameters of the `DWC_ETL_PARAMETER` control table for `'OUDM-INTRA-ETL'` process. For an incremental load of an Oracle Utilities Data Model warehouse, specify the values shown in the following table (that is, the beginning and end date of the ETL period) for all three processes.

Table 4-3 DWC_ETL_PARAMETER Table Parameter Descriptions

Columns	Value
FROM_DATE_ETL	The beginning date of the ETL period.
TO_DATE_ETL	The ending date of the ETL period.

3. Create a session by connecting `oudm_user` user through SQLPLUS. Then, start an intra-ETL process. Make sure the previous process ended with `'COMPLETED-SUCCESS'` status before starting a new process:

```
sqlplus oudm_user/oudm_user@SID

SQL> DECLARE
  l_process_type  OUDM_SYS.DWC_INTRA_ETL_PROCESS.PROCESS_TYPE%TYPE;
  l_error_text    OUDM_SYS.DWC_MESSAGE.MESSAGE_TEXT%TYPE;
  l_process_no    NUMBER;
BEGIN
  l_process_no := OUDM_SYS.PKG_INTRA_ETL_UTIL.Start_Process(l_process_type,l_error_text);
```

```
    END;
    /
```

4. Refresh Oracle Utilities Data Model derived tables by executing following commands:

```
SQL> DECLARE
  p_process_no    NUMBER;
  l_status        VARCHAR2(20);
BEGIN
  l_status :=
    OUDM_SYS.PKG_DWD_ACCT_ARRER_MO.Load('DWD_ACCT_ARRER_MO',p_process_no);
  l_status :=
    OUDM_SYS.PKG_DWD_ACCT_BAL_MO.Load('DWD_ACCT_BAL_MO',p_process_no);
  l_status :=
    OUDM_SYS.PKG_DWD_ACCT_DEBT_DAY.Load('DWD_ACCT_DEBT_DAY',p_process_no);
  l_status :=
OUDM_SYS.PKG_DWD_ACCT_PMT_MTD_STAT_HST.Load('DWD_ACCT_PYMT_MTHD_STAT_HIST',p_process_no);
  l_status :=
    OUDM_SYS.PKG_DWD_ACCT_PYMT_DAY.Load('DWD_ACCT_PYMT_DAY',p_process_no);
  l_status := OUDM_SYS.PKG_DWD_END_DVC_EVT_CUST_DAY.Load('DWD_END_DVC_EVT_CUST_DAY',p_process_no);
l_status :=   OUDM_SYS.PKG_DWD_END_DVC_EVT_DVC_DAY.Load('DWD_END_DVC_EVT_DVC_DAY',p_process_no);
  l_status :=
    OUDM_SYS.PKG_DWD_MTR_RDNG_DAY.Load('DWD_MTR_RDNG_DAY',p_process_no);
  l_status :=
    OUDM_SYS.PKG_DWD_MTR_RDNG_HR.Load('DWD_MTR_RDNG_HR',p_process_no);
  l_status :=
    OUDM_SYS.PKG_DWD_OUTG_DAY.Load('DWD_OUTG_DAY',p_process_no);
  l_status :=
    OUDM_SYS.PKG_DWD_OUTG_USG_PNT.Load('DWD_OUTG_USG_PNT',p_process_no);
END;
/

SQL> DECLARE
  p_process_no    NUMBER;
  l_status        VARCHAR2(20);
BEGIN
l_status := OUDM_SYS.PKG_DR_PROG_LD_RDCTN_RGN_DAY.Load ('DWD_DR_PROG_LD_RDCTN_RGN_DAY',p_process_no);
l_status := OUDM_SYS.PKG_DWD_RLBLTY_IND_CITY_MO.Load('DWD_RLBLTY_IND_CITY_MO',p_process_no);
l_status := OUDM_SYS.PKG_DWD_RLBLTY_IND_FEDR_MO.Load('DWD_RLBLTY_IND_FEDR_MO',p_process_no);
END;
/
```

Related Topics:

- Performing an Initial Load of an Oracle Utilities Data Model Warehouse (page 4-13)
  Performing an initial load of an Oracle Utilities Data Model is a multistep process with steps: :loading the foundation layer and loading the access layer.
- *Oracle Utilities Data Model Reference*

4.5.4 Refreshing Oracle Utilities Data Model Aggregate Materialized Views

Refreshing the Aggregate Materialized Views in an Oracle Utilities Data Model is a multi-step process:

1. Refresh the foundation layer of the Oracle Utilities Data Model warehouse (that is, the reference, lookup, and base tables) with operational system data by executing the source-ETL that you have written.
2. Refresh Oracle Utilities Data Model derived tables.
3. Create a session by connecting `oudm_user` user through SQLPLUS. An intra-ETL process must be in 'RUNNING' status now:

   ```
   sqlplus oudm_user/oudm_user@SID
   ```

4. Refresh Oracle Utilities Data Model aggregate materialized views by executing following commands:

```
SQL> DECLARE
  p_process_no    NUMBER;
  l_status        VARCHAR2(20);
BEGIN
l_status := OUDM_SYS.PKG_INTRA_ETL_UTIL.Refresh_MV('DWA_END_DVC_EVT_DVC_MO',p_process_no);
l_status := OUDM_SYS.PKG_INTRA_ETL_UTIL.Refresh_MV('DWA_END_DVC_EVT_CUST_MO',p_process_no);
l_status :=   OUDM_SYS.PKG_INTRA_ETL_UTIL.Refresh_MV('DWA_MTR_RDNG_MO',p_process_no);
l_status := OUDM_SYS.PKG_INTRA_ETL_UTIL.Refresh_MV('DWA_MTR_RDNG_TOU_MO',p_process_no);
l_status := OUDM_SYS.PKG_INTRA_ETL_UTIL.Refresh_MV('DWA_OUTG_MO',p_process_no);
l_status := OUDM_SYS.PKG_INTRA_ETL_UTIL.Refresh_MV('DWA_MTR_RDNG_MO_ACCT',p_process_no);
l_status := OUDM_SYS.PKG_INTRA_ETL_UTIL.Refresh_MV('DWA_MTR_RDNG_MO_CUST',p_process_no);
l_status := OUDM_SYS.PKG_INTRA_ETL_UTIL.Refresh_MV('DWA_MTR_RDNG_MO_UP',p_process_no);
END;
/
```

Related Topics:

- Refreshing Oracle Utilities Data Model Derived Tables (page 4-18)

4.5.5 Refreshing Oracle Utilities Data Model OLAP Cubes

On a scheduled basis you must update the OLAP cube data with the relational data that has been added to the Oracle Utilities Data Model data warehouse since the initial load of the OLAP cubes.

Refreshing the OLAP Cubes in an Oracle Utilities Data Model is a multi-step process:

1. Refresh the foundation layer of the Oracle Utilities Data Model warehouse (that is, the reference, lookup, and base tables) with operational system data by executing the source-ETL that you have written.
2. Refresh Oracle Utilities Data Model derived tables.
3. Refresh Oracle Utilities Data Model aggregate materialized views.
4. Update the parameters of the `DWC_OLAP_ETL_PARAMETER` control table.

> **See Also:**
>
> For more information on `DWC_OLAP_ETL_PARAMETER` control table, see *Oracle Utilities Data Model Reference*.

5. Create a session by connecting `oudm_user` user through SQLPLUS. An intra-ETL process created in "Refreshing Oracle Utilities Data Model Derived Tables (page 4-18)" must be in `'RUNNING'` status:

```
sqlplus oudm_user/oudm_user@SID
```

6. Refresh Oracle Utilities Data Model OLAP cubes by executing following commands:

```
SQL> DECLARE
  l_build_methd OUDM_SYS.DWC_OLAP_ETL_PARAMETER.BUILD_METHOD%TYPE;
  l_cube_nm OUDM_SYS.DWC_OLAP_ETL_PARAMETER.CUBENAME%TYPE;
  l_maxjobques OUDM_SYS.DWC_OLAP_ETL_PARAMETER.MAXJOBQUEUES%TYPE;
  l_calc_fcst  OUDM_SYS.DWC_OLAP_ETL_PARAMETER.CALC_FCST%TYPE;
  l_no_fcst_yrs OUDM_SYS.DWC_OLAP_ETL_PARAMETER.NO_FCST_YRS%TYPE;
  l_fcst_mthd  OUDM_SYS.DWC_OLAP_ETL_PARAMETER.FCST_MTHD%TYPE;
  l_fcst_st_yr OUDM_SYS.DWC_OLAP_ETL_PARAMETER.FCST_ST_YR%TYPE;
  l_fcst_end_yr OUDM_SYS.DWC_OLAP_ETL_PARAMETER.FCST_END_YR%TYPE;
  l_status VARCHAR2(20);
BEGIN
  /***************  Fetching the values of the OLAP ETL parameters variable used in this
procedure  ****************/
      SELECT
         BUILD_METHOD l_build_methd,
        CUBENAME l_cube_nm,
        MAXJOBQUEUES l_maxjobques,
        CALC_FCST l_calc_fcst,
        NO_FCST_YRS l_no_fcst_yrs,
        FCST_MTHD l_fcst_mthd,
        FCST_ST_YR l_fcst_st_yr,
        FCST_END_YR l_fcst_end_yr
      INTO
        l_build_methd,
        l_cube_nm,
        l_maxjobques,
        l_calc_fcst,
        l_no_fcst_yrs,
        l_fcst_mthd,
        l_fcst_st_yr,
        l_fcst_end_yr
      FROM
      OUDM_SYS.DWC_OLAP_ETL_PARAMETER;
   l_status :=
OUDM_SYS.PKG_OUDM_OLAP_ETL_AW_LOAD.olap_etl_aw_build(l_build_methd,l_cube_nm,l_maxjobques,l_calc_f
cst,l_no_fcst_yrs,l_fcst_mthd,l_fcst_st_yr,l_fcst_end_yr,null,null);
END;
/
```

7. If there is requirement to refresh only Oracle Utilities Data Model OLAP cubes, the same can be achieved with step 6, but before that make sure an intra-ETL process is already running. If no intra-ETL process is running, start one:

```
sqlplus oudm_user/oudm_user@SID

SQL> DECLARE
  l_process_type  OUDM_SYS.DWC_INTRA_ETL_PROCESS.PROCESS_TYPE%TYPE;
  l_error_text    OUDM_SYS.DWC_MESSAGE.MESSAGE_TEXT%TYPE;
  l_process_no    NUMBER;
BEGIN
  l_process_no := OUDM_SYS.PKG_INTRA_ETL_UTIL.Start_Process(l_process_type,l_error_text);
END;
/
```

Related Topics:

- Performing an Initial Load of an Oracle Utilities Data Model Warehouse (page 4-13)
  Performing an initial load of an Oracle Utilities Data Model is a multistep process with steps: :loading the foundation layer and loading the access layer.
- Refreshing Oracle Utilities Data Model Derived Tables (page 4-18)

4.5.6 Refreshing Oracle Utilities Data Model Data Mining Models

Refreshing of data mining models is *not* integrated into intra-ETL workflow. There is only one data mining model, Customer Savings and Customer Profile by DR Program.

For each DR program, a mining model is trained. If you want to rebuild the mining models for all DR programs, perform the following steps:

1. Make sure an intra-ETL process is running. If not, start an intra-ETL process:

```
sqlplus oudm_user/oudm_user@SID

SQL> DECLARE
  l_process_type  OUDM_SYS.DWC_INTRA_ETL_PROCESS.PROCESS_TYPE%TYPE;
  l_error_text    OUDM_SYS.DWC_MESSAGE.MESSAGE_TEXT%TYPE;
  l_process_no    NUMBER;
BEGIN
  l_process_no :=
OUDM_SYS.PKG_INTRA_ETL_UTIL.Start_Process(l_process_type,l_error_text);
END;
/
```

2. Build (rebuild in case already built) mining model by executing the following commands:

```
sqlplus oudm_user/oudm_user@SID

DECLARE
--Indicates whether a model already built for a DR program can be rebuilt
l_model_refresh_ind CHAR(1) := 'Y';
l_model_exist_ind   CHAR(1);
BEGIN
  FOR rec IN (SELECT DEMAND_RESPN_PROG_KEY FROM OUDM_SYS.DWR_DEMAND_RESPN_PROG
WHERE END_DT < OUDM_SYS.PKG_INTRA_ETL_UTIL.Get_End_Date)
  LOOP
    SELECT COUNT(*) INTO l_model_exist_ind
    FROM OUDM_SYS.USER_MINING_MODELS
    WHERE MODEL_NAME = 'OUDM_PROFILE_KMEANS_' || rec.DEMAND_RESPN_PROG_KEY
    ;

IF l_model_exist_ind = 1
THEN
     IF l_model_refresh_ind = 'Y'
     THEN

OUDM_SYS.PKG_DWD_CUST_DR_PROG_PROFILE.loaddata(rec.DEMAND_RESPN_PROG_KEY);

OUDM_SYS.PKG_MINING_ETL.crt_cust_sgmnt_src_view(rec.DEMAND_RESPN_PROG_KEY);

OUDM_SYS.PKG_OUDM_MINING.crt_frst_step_cust_sgmnt_model(rec.DEMAND_RESPN_PROG_KEY
);
     ELSE
```

```
DBMS_OUTPUT.PUT_LINE('NOTE:Chosen to not refresh the existing model :: ' ||
'OUDM_PROFILE_KMEANS_' || rec.DEMAND_RESPN_PROG_KEY || ' :: set
l_model_refresh_ind to ''Y'' to refresh existing models');
  END IF;

    ELSE
OUDM_SYS.PKG_DWD_CUST_DR_PROG_PROFILE.loaddata(rec.DEMAND_RESPN_PROG_KEY);
OUDM_SYS.PKG_MINING_ETL.crt_cust_sgmnt_src_view(rec.DEMAND_RESPN_PROG_KEY);
OUDM_SYS.PKG_OUDM_MINING.crt_frst_step_cust_sgmnt_model(rec.DEMAND_RESPN_PROG_KEY
);

    END IF;
    END LOOP;
  END;
  /
```

4.6 Managing Errors During Oracle Utilities Data Model Intra-ETL Execution

Describes the steps to identify and manage errors during intra-ETL execution.

- Monitoring the Execution of the Intra-ETL Process (page 4-23)
  Three `oudm_sys` schema control tables, `DWC_INTRA_ETL_PROCESS`, `DWC_INTRA_ETL_ACTIVITY`, `DWC_OLAP_ACTIVITY` monitor the execution of the intra-ETL process.
- Recovering an Intra ETL Process (page 4-24)
  Describes the process to recover an intra-ETL process.

4.6.1 Monitoring the Execution of the Intra-ETL Process

Three `oudm_sys` schema control tables, `DWC_INTRA_ETL_PROCESS`, `DWC_INTRA_ETL_ACTIVITY`, `DWC_OLAP_ACTIVITY` monitor the execution of the intra-ETL process.

You can access these three tables from `oudm_user` user.

Each normal run (as opposed to an error-recovery run) of a separate intra-ETL execution performs the following steps:

1. Inserts a record into the `DWC_INTRA_ETL_PROCESS` table with a monotonically increasing system generated unique process key, `SYSDATE` as process start time, `RUNNING` as the process status, and an input date range in the `FROM_DATE_ETL` and `TO_DATE_ETL` columns.
2. Invokes each of the individual intra-ETL programs in the appropriate order of dependency. Before the invocation of each program, the procedure inserts a record into the intra-ETL Activity detail table, `DWC_INTRA_ETL_ACTIVITY`, with values for:

   `ACTIVITY_KEY`, a system generated unique activity key.

   `PROCESS_KEY`, the process key value corresponding to the intra-ETL process.

   `ACTIVITY_NAME`, an individual program name.

   `ACTIVITY_DESC`, a suitable activity description.

   `ACTIVITY_START_TIME`, the value of `SYSDATE`.

ACTIVITY_STATUS, the value of RUNNING.

3. Updates the corresponding record in the DWC_INTRA_ETL_ACTIVITY table for the activity end time and activity status after the completion of each individual ETL program (either successfully or with errors). For successful completion of the activity, the procedure updates the status as 'COMPLETED-SUCCESS'. When an error occurs, the procedure updates the activity status as 'COMPLETED-ERROR', and also updates the corresponding error detail in the ERROR_DTL column.
4. Updates the record corresponding to the process in the DWC_INTRA_ETL_ PROCESS table for the process end time and status, after the completion of all individual intra-ETL programs. When all the individual programs succeed, the procedure updates the status to 'COMPLETED-SUCCESS'; otherwise it updates the status to 'COMPLETED-ERROR'.
5. For OLAP cubes loading, a record is inserted into DWC_OLAP_ACTIVITY table with CUBENAME as cube name, status as 'RUNNING', and LOAD_START_DT as SYSDATE for each cube. It updates the record upon the completion of cube loading. It updates STATUS column to 'COMPLETED-SUCCESS' if cube loading is successful, otherwise 'COMPLETE-ERROR' and updates LOAD_END_DT column to SYSDATE. In case of 'COMPLETED-ERROR' cubes, it also updates ERROR_DTL column with error details.

You can monitor the execution state of the intra-ETL, including current process progress, time taken by individual programs, or the complete process, by viewing the contents of the DWC_INTRA_ETL_PROCESS, DWC_INTRA_ETL_ACTIVITY, and DWC_OLAP_ACTIVITY tables. In DWC_INTRA_ETL_ACTIVITY table, see the records of currently running processes. Monitoring can be done both during and after the execution of the intra-ETL procedure.

Related Topics:

- *Oracle Utilities Data Model Reference*

4.6.2 Recovering an Intra ETL Process

Describes the process to recover an intra-ETL process.

To recover an intra-ETL process:

1. Identify the errors by looking at the corresponding error details that are tracked against the individual programs in the DWC_INTRA_ETL_ACTIVITY table.
2. Identify errors of OLAP cubes loading for individual cubes in DWC_OLAP_ACTIVITY table.
3. Correct the causes of the errors.
4. Re-invoke the intra-ETL process.

The intra-ETL workflow process identifies whether it is a normal run or recovery run by referring the DWC_INTRA_ETL_ACTIVITY table. During a recovery run, the intra-ETL workflow executes only the necessary programs. For example, for a derived population error as a part of the previous run, this recovery run executes the individual derived population programs which produced errors in the previous run. After their successful completion, the run refreshes aggregate materialized views in the appropriate order.

In this way, the intra-ETL error recovery is almost transparent, without involving the data warehouse or ETL administrator. The administrator must only correct the causes

of the errors and re-invoke the intra-ETL process. The intra-ETL process identifies and executes the programs that generated errors.

5
Report and Query Customization

This chapter provides information about creating reports, queries, and dashboards against the data in an Oracle Utilities Data Model warehouse. It contains the following topics:

- Reporting Approaches in Oracle Utilities Data Model (page 5-1)
  There are two main approaches to creating reports from data in an Oracle Utilities Data Model warehouse: Relational Reporting and OLAP Reporting.
- Customizing Oracle Utilities Data Model Sample Reports (page 5-3)
  Sample reports and dashboards are delivered with Oracle Utilities Data Model. These sample reports illustrate the analytic capabilities provided with Oracle Utilities Data Model -- including the OLAP and data mining capabilities.
- Writing Your Own Queries and Reports (page 5-3)
  The `oudm_sys` schema defines the relational tables and views in Oracle Utilities Data Model. You can use any SQL reporting tool to query and report on these tables and views.
- Optimizing Star Queries (page 5-5)
  A typical query in the access layer is a join between the fact table and some number of dimension tables and is often referred to as a star query.
- Troubleshooting Oracle Utilities Data Model Report Performance (page 5-7)
- Writing As Is and As Was Queries (page 5-8)
- Tutorial: Creating a New Oracle Utilities Data Model Dashboard (page 5-14)
  This tutorial explains how to create a dashboard based on the Oracle Utilities Data Model webcat included with the sample reports delivered with Oracle Utilities Data Model.
- Tutorial: Creating a New Oracle Utilities Data Model Report (page 5-17)
  Describes how to create a report based on the Oracle Utilities Data Model webcat included with the sample reports delivered with Oracle Utilities Data Model.

5.1 Reporting Approaches in Oracle Utilities Data Model

There are two main approaches to creating reports from data in an Oracle Utilities Data Model warehouse: Relational Reporting and OLAP Reporting.

Relational Reporting

With relational reporting, you create reports against the analytical layer entities using the fact entities as the center of the star with the reference entities (that is, `DWR_` and `DWL_` tables) acting as the dimensions of the star. Typically the fact entities include the derived and aggregate entities (that is, `DWD_` and `DWA_` tables). However in some cases, you may need to use the base entities (that is, `DWB_` tables) along with the reference tables to generate more detailed reports.

The reference tables (that is, DWR_ tables) typically represent dimensions which contain a business hierarchy and are present in the form of snowflake entities

containing a table for each level of the hierarchy. This allows us to attach the appropriate set of reference entities for the multiple subject area and fact entities composed of differing granularity.

For example, you can use the set of tables comprising DWR_DAY tables to query against a DAY level meter reading entity such as DWD_MTR_RDNG_DAY. On the other hand, you need to use the higher level snowflakes at Month level and above such as DWR_CLNDR_MO, DWR_CLNDR_QTR, DWR_CLNDR_YR in order to query against the MONTH level meter reading entity such as DWA_MTR_RDNG_MO.

The lookup tables (that is tables, with the DWL_ prefix) represent the simpler dimensions comprising a single level containing a flat list of values. Typically, most reporting tools add a superficial top level to the dimension.

> **Note:**
>
> The use of numbers as text in Lookup code allows you to group them by using only the first character of the lookup value code. This could provide an artificial hierarchy level.

OLAP Reporting

With OLAP reporting, you access Oracle OLAP cubes using SQL against the dimension and cube (fact) views. Cubes and dimensions are represented using a star schema design. Dimension views form a constellation around the cube (or fact) view. The dimension and cube views are relational views with names ending with _VIEW. Typically, the dimension view used in the reports is named *dimension_hierarchy*_VIEW and the cube view is named *cube*_VIEW.

Unlike the corresponding relational dimension entities stored in DWR_ tables, the OLAP dimension views contains information relating to the whole dimension including all the levels of the hierarchy logically partitioned on the basis of a level column (identified as level_name). On a similar note, the cube views also contain the facts pertaining to the cross-combination of the levels of individual dimensions which are part of the cube definition. Also the join from the cube view and the dimension views are based on the dimension keys along with required dimension level filters.

Although the OLAP views are also modeled as a star schema, there are certain unique features to the OLAP reporting methodology which requires special modeling techniques in Oracle Business Intelligence Suite Enterprise Edition.

> **See Also:**
>
> The Oracle By Example tutorial, entitled "Using Oracle OLAP 11*g* With Oracle BI Enterprise Edition".
>
> To access the tutorial, open the Oracle Learning Library in your browser at Oracle Learning Library and, then, search for the tutorial by name.

The rest of this chapter explains how to create Oracle Utilities Data Model reports. For examples of Oracle Utilities Data Model reports, see:

- Writing As Is and As Was Queries (page 5-8)
- Tutorial: Creating a New Oracle Utilities Data Model Dashboard (page 5-14)
- Tutorial: Creating a New Oracle Utilities Data Model Report (page 5-17)
- The sample reports provided with Oracle Utilities Data Model.

5.2 Customizing Oracle Utilities Data Model Sample Reports

Sample reports and dashboards are delivered with Oracle Utilities Data Model. These sample reports illustrate the analytic capabilities provided with Oracle Utilities Data Model -- including the OLAP and data mining capabilities.

> **See:**
>
> *Oracle Utilities Data Model Installation Guide* for more information on installing the sample reports and deploying the Oracle Utilities Data Model RPD and webcat on the Business Intelligence Suite Enterprise Edition instance.

The sample reports were developed using Oracle Business Intelligence Suite Enterprise Edition which is a comprehensive suite of enterprise business intelligence products that delivers a full range of analysis and reporting capabilities. Thus, the reports also illustrate the ease with which you can use Oracle Business Intelligence Suite Enterprise Edition Answers and Dashboard presentation tools to create useful reports.

You can use Oracle Business Intelligence Suite Enterprise Edition Answers and Dashboard presentation tools to customize the predefined sample dashboard reports:

- **Oracle BI Answers**. Provides end user ad hoc capabilities in a pure Web architecture. Users interact with a logical view of the information -- completely hidden from data structure complexity while simultaneously preventing runaway queries. Users can easily create charts, pivot tables, reports, and visually appealing dashboards.
- **Oracle BI Interactive Dashboards**. Provide any knowledge worker with intuitive, interactive access to information. The end user can be working with live reports, prompts, charts, tables, pivot tables, graphics, and tickers. The user has full capability for drilling, navigating, modifying, and interacting with these results.

Related Topics:

- *Oracle Utilities Data Model Reference*

5.3 Writing Your Own Queries and Reports

The `oudm_sys` schema defines the relational tables and views in Oracle Utilities Data Model. You can use any SQL reporting tool to query and report on these tables and views.

Oracle Utilities Data Model also supports On Line Analytic processing (OLAP) reporting using OLAP cubes defined in the `oudm_sys` schema. You can query and write reports on OLAP cubes by using SQL tools to query the views that are defined for the cubes or by using OLAP tools to directly query the OLAP components.

> **See Also:**
>
> The discussion on querying dimensional objects in *Oracle OLAP User's Guide*

Example 5-1 Creating a Relational Query for Oracle Utilities Data Model

For example, assume that you want to know the total kilowatt usage details for the top ten customers for March 2006. To answer this question, you might have to query the tables described in the following table:

Entity Name	Table Name	Description
METER READING BY DAY	DWD_MTR_RDNG_DAY	Derived fact table on daily meter reading
CUSTOMER	DWR_CUST	All the customers, including individual and organization customers
DAY	DWR_DAY	Calendar day in the day dimension.

To perform this query, you execute the following SQL statement:

```
SELECT *
FROM
  (SELECT c.prty_dscr as Customer,
    SUM( m.tot_kwh ) as Total_KWH
  FROM dwd_mtr_rdng_day m,
    dwr_day d,
    dwr_cust c
  WHERE m.day_key = d.day_key
  AND m.cust_key  = c.cust_key
  AND TO_CHAR(to_date(m.clndr_mo_key, 'yyyymmdd'), 'MON-YY') LIKE 'MAR-06'
  GROUP BY c.prty_dscr,
    TO_CHAR(to_date(m.clndr_mo_key, 'yyyymmdd'), 'MON-YY')
  ORDER BY 3 DESC
  )
WHERE ROWNUM <= 10;
```

The result of this query:

```
CUSTOMER            TOTAL_KWH
Brady Bakker        1152.58
Deb Abbassi         1120.201
Charli Eddisson     1119.701
Jasmine Seto        1111.391
Mason Murray        1109.741
Reuben Zanth        1044.612
Radley Whitehead    1043.892
Bert Faimon          594.49
Lolita Barkley       591.99
Denise Mulholland    590.69
```

Related Topics:

- Reporting Approaches in Oracle Utilities Data Model (page 5-1)
  There are two main approaches to creating reports from data in an Oracle Utilities Data Model warehouse: Relational Reporting and OLAP Reporting.

- Oracle OLAP Cube Views (page 3-12)

5.4 Optimizing Star Queries

A typical query in the access layer is a join between the fact table and some number of dimension tables and is often referred to as a star query.

In a star query each dimension table is joined to the fact table using a primary key to foreign key join. Normally the dimension tables do not join to each other.

Typically, in this kind of query all of the `WHERE` clause predicates are on the dimension tables and the fact table. Optimizing this type of query is very straight forward.

To optimize this query, do the following:

- Create a bitmap index on each of the foreign key columns in the fact table or tables
- Set the initialization parameter `STAR_TRANSFORMATION_ENABLED` to `TRUE`.

This enables the optimizer feature for star queries which is off by default for backward compatibility.

If your environment meets these two criteria, your star queries should use a powerful optimization technique that rewrites or transforms your SQL called star transformation. Star transformation executes the query in two phases:

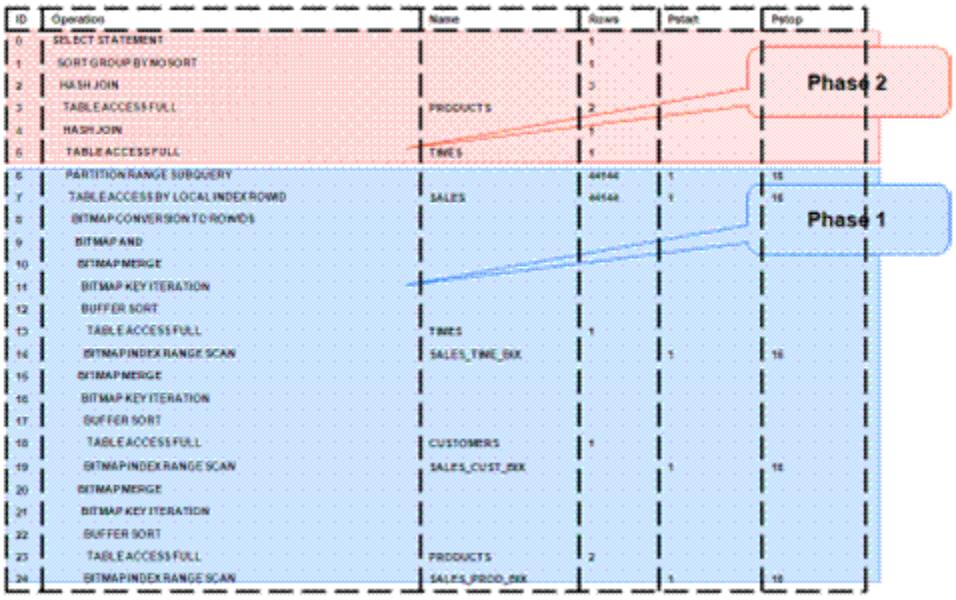
1. Retrieves the necessary rows from the fact table (row set).
2. Joins this row set to the dimension tables.

The rows from the fact table are retrieved by using bitmap joins between the bitmap indexes on all of the foreign key columns. The end user never needs to know any of the details of `STAR_TRANSFORMATION`, as the optimizer automatically chooses `STAR_TRANSFORMATION` when it is appropriate.

Example 5-2 (page 5-6) gives the step by step process to use `STAR_TRANSFORMATION` to optimize a star query.

The following figure shows the typical execution plan for a star query when `STAR_TRANSFORMATION` has kicked in. The execution plan may not look exactly as you expected. There is no join back to the customer table after the rows have been successfully retrieved from the `Sales` table. If you look closely at the select list, you can see that there is not anything actually selected from the `Customers` table so the optimizer knows not to bother joining back to that dimension table. You may also notice that for some queries even if `STAR_TRANSFORMATION` does kick in it may not use all of the bitmap indexes on the fact table. The optimizer decides how many of the bitmap indexes are required to retrieve the necessary rows from the fact table. If an additional bitmap index would not improve the selectivity, the optimizer does not use it. The only time you see the dimension table that corresponds to the excluded bitmap in the execution plan is during the second phase or the join back phase.

Figure 5-1 Star Schema Transformation



Example 5-2 Star Transformation

A business question that could be asked against the star schema in Figure 5-1 (page 5-6) would be "What was the total number of umbrellas sold in Boston during the month of May 2008?"

1. The original query.

```
select SUM(quantity_sold) total_umbrellas_sold_in_Boston
From Sales s, Customers c, Products p, Times t
Where s.cust_id=cust_id
And s.prod_id = p.prod_id
And s.time_id=t.time_id
And c.cust_city='BOSTON'
And p.product='UMBRELLA'
And t.month='MAY'
And t.year=2012;
```

   As you can see all of the where clause predicates are on the dimension tables and the fact table (`Sales`) is joined to each of the dimensions using their foreign key, primary key relationship.

2. Take the following actions:

   a. Create a bitmap index on each of the foreign key columns in the fact table or tables.

   b. Set the initialization parameter `STAR_TRANSFORMATION_ENABLED` to `TRUE`.

3. The rewritten query. Oracle rewrites and transfers the query to retrieve only the necessary rows from the fact table using bitmap indexes on the foreign key columns

```
select SUM(quantity_sold
From Sales
Where cust_id IN
```

```
(select c.cust_id From Customers c Where c.cust_city='BOSTON')
And s.prod_id IN
(select p.prod_id From Products p Where  p.product='UMBRELLA')
And s.time_id IN
(select t.time_id From Times(Where t.month='MAY' And t.year=2012);
```

By rewriting the query in this fashion you can now leverage the strengths of bitmap indexes. Bitmap indexes provide set based processing within the database, allowing you to use various fact methods for set operations such as `AND`, `OR`, `MINUS`, and `COUNT`. So, you use the bitmap index on time_id to identify the set of rows in the fact table corresponding to sales in May 2008. In the bitmap the set of rows are actually represented as a string of 1's and 0's. A similar bitmap is retrieved for the fact table rows corresponding to the sale of umbrellas and another is accessed for sales made in Boston. At this point there are three bitmaps, each representing a set of rows in the fact table that satisfy an individual dimension constraint. The three bitmaps are then combined using a bitmap `AND` operation and this newly created final bitmap is used to extract the rows from the fact table needed to evaluate the query.

4. Using the rewritten query, Oracle joins the rows from fact tables to the dimension tables.

   The join back to the dimension tables is normally done using a hash join, but the Oracle Optimizer selects the most efficient join method depending on the size of the dimension tables.

5.5 Troubleshooting Oracle Utilities Data Model Report Performance

Take the following actions to identify problems generating a report created using Oracle Business Intelligence Suite Enterprise Edition:

1. In the (Online) Oracle BI Administrator Tool, select **Manage,** then **Security**, then **Users**, and then **oudm**.

   Ensure that the value for **Logging level** is `7`.

2. Open the Oracle Utilities Data Model Repository, select **Manage**, and then **Cache**.
3. In the right-hand pane of the Cache Manager window, select all of the records, then right-click and select **Purge**.
4. Run the report or query that you want to track using the SQL log.
5. Open the query log file (`NQQuery.log`) under `OracleBI\server\Log`.

   The last query SQL is the log of the report you have just run. If an error was returned in your last accessed report, there is an error at the end of this log.

Example 5-3 Troubleshooting an Oracle Utilities Data Model Report

Assume the log file contains the following error:

```
Query Status: Query Failed: [nQSError: 15018] Incorrectly defined logical table
source (for fact table Customer Mining) does not contain mapping for
[Customer.Address Location Type, Customer.Customer Segment Name, Customer.Party
Name].
```

This error occurs when there is a problem in the Business layer in your Oracle Business Intelligence Suite Enterprise Edition repository.

In this case, you need to check the mapping for `Customer.Address Location Type`, `Customer.Customer Segment Name`, and `Customer.Party Name`.

Example 5-4 Troubleshooting a Report: A Table Does Not Exist

Assume the log file contains the following error:

```
Query Status: Query Failed: [encloser: 17001] Oracle Error code: 942, message:
ORA-00942: table or view does not exist.
```

This error occurs when the physical layer in your Oracle Business Intelligence Suite Enterprise Edition repository has the table which actually does not exist in the Database.

To find out which table has problem:

1. Copy the SQL query to database environment.
2. Execute the query.

The table which does not exist is marked out by the database client.

Example 5-5 Troubleshooting a Report: When the Database is Not Connected

Assume the log file contains the following error:

Error: `Query Status: Query Failed: [nQSError: 17001] Oracle Error code: 12545, message: ORA-12545: connect failed because target host or object does not exist.`

Meaning: This error occurs when the Database is not connected.

Action: Check connecting information in physical layer and ODBC connection to ensure that the repository is connecting to the correct database.

The following examples illustrate how to use these error messages:

- Example 5-3 (page 5-7)
- Example 5-4 (page 5-8)
- Example 5-5 (page 5-8)

5.6 Writing As Is and As Was Queries

Two common query techniques are "as is" and "as was" queries:

- Characteristics of an As Is Query (page 5-8)
- Characteristics of an As Was Query (page 5-9)
  Describes an An As Was query (also known as point-in-time analysis).
- Examples: As Is and As Was Queries Against Oracle Utilities Data Model (page 5-9)

5.6.1 Characteristics of an As Is Query

An As Is query has the following characteristics:

- The resulting report shows the data as it happened.

- The snowflake dimension tables are also joined using the surrogate key columns (that is the primary key and foreign key columns).
- The fact table is joined with the dimension tables (at leaf level) using the surrogate key column.
- Slowly-changing data in the dimensions are joined with their corresponding fact records and are presented individually.
- It is possible to add up the components if the different versions share similar characteristics.

5.6.2 Characteristics of an As Was Query

Describes an An As Was query (also known as point-in-time analysis).

An As Was query has the following characteristics:

- The resulting report shows the data that would result from freezing the dimensions and dimension hierarchy at a specific point in time.
- Each snowflake table is initially filtered by applying a point-in-time date filter which selects the records or versions which are valid as of the analysis date. This structure is called the point-in-time version of the snowflake.
- The filtered snowflake is joined with an unfiltered version of itself by using the natural key. All of the snowflake attributes are taken from the point-in-time version alias. The resulting structure is called the composite snowflake.
- A composite dimension is formed by joining the individual snowflakes on the surrogate key.
- The fact table is joined with the composite dimension table at the leaf level using the surrogate key column.
- The point-in-time version is super-imposed on all other possible SCD versions of the same business entity -- both backward and forward in time. Joining in this fashion gives the impression that the dimension is composed of only the specific point-in-time records.
- All of the fact components for various versions add up correctly due to the super-imposition of point-in-time attributes within the dimensions.

5.6.3 Examples: As Is and As Was Queries Against Oracle Utilities Data Model

Based on the "Data used for the examples (page 5-9)", the following examples illustrate the characteristics of As Is and As Was queries:

- Example 5-6 (page 5-11)
- Example 5-7 (page 5-11)
- Example 5-8 (page 5-12)
- Example 5-9 (page 5-13)

Data used for the examples

Assume that your data warehouse has a `Customer` table, a `County`, and a `TaxPaid` fact table. As of January 1, 2012, these tables include the values shown:

Customer Table

Cust Id	Cust Cd	Cust Nm	Gender	M Status	County Id	County Cd	Country Nm	...	Eff Frm	Eff To
101	JoD	John Doe	Male	Single	5001	SV	Sunnyvale	...	1-Jan-12	31-Dec-99
102	JaD	Jane Doe	Female	Single	5001	SV	Sunnyvale	...	1-Jan-12	31-Dec-99
103	JiD	Jim Doe	Male	Married	5002	CU	Cupertino	...	1-Jan-12	31-Dec-99

County Table

County Id	County CD	County Nm	Population	...	Eff Frm	Eff To
5001	SV	Sunnyvale	Very High	...	1-Jan-12	31-Dec-99
5002	CU	Cupertino	High	...	1-Jan-12	31-Dec-99

TaxPaid Table

Cust Id	Day	Tax Type	Tax
101	1-Jan-12	Professional Tax	100
102	1-Jan-12	Professional Tax	100
103	1-Jan-12	Professional Tax	100

Assume that the following events occurred in January 2012:

- On January 20, 2012, Jane Doe marries.
- On Jan 29, 2012, John Doe moves from Sunnyvale to Cupertino.

Consequently, as shown, on February 1, 2012, the `Customer` and `TaxPaid` tables have new data while the values in the `County` table stay the same.

Customer table

Cust Id	Cust Cd	Cust Nm	Gender	M Status	County Id	County Cd	Country Nm	...	Eff Frm	Eff To
101	JoD	John Doe	Male	Single	5001	SV	Sunnyvale	...	1-Jan-12	29-Jan-12
102	JaD	Jane Doe	Female	Single	5001	SV	Sunnyvale	...	1-Jan-12	20-Jan-12
103	JiD	Jim Doe	Male	Married	5002	CU	Cupertino	...	1-Jan-12	31-Dec-99
104	JaD	Jane Doe	Female	Married	5001	SV	Sunnyvale	...	21-Jan-12	31-Dec-99
105	JoD	John Doe	Male	Single	5002	CD	Cupertino	...	30-Jan-12	31-Dec-99

County table

County Id	County CD	County Nm	Population	...	Eff Frm	Eff To
5001	SV	Sunnyvale	Very High	...	1-Jan-12	31-Dec-99
5002	CU	Cupertino	High	...	1-Jan-12	31-Dec-99

TaxPaid Table

Cust Id	Day	Tax Type	Tax
101	1-Jan-12	Professional Tax	100
102	1-Jan-12	Professional Tax	100
103	1-Jan-12	Professional Tax	100
105	1-Feb-12	Professional Tax	100
104	1-Feb-12	Professional Tax	100
103	1-Feb-12	Professional Tax	100

Example 5-6 As Is Query for Tax Collection Split by Marital Status

Assuming the "Data used for the examples (page 5-9)", to show the tax collection data split by martial status, the following SQL statement that joins the `TaxPaid` fact table and the `Customer` dimension table on the `cust_id` surrogate key and the `Customer` and `County` snowflakes on the `cnty_id` surrogate key.

```
SELECT cust.cust_nm, cust.m_status, SUM(fct.tx)
FROM taxpaid fct, customer cust, county cnty
WHERE fct.cust_id = cust.cust_id
AND cust.cnty_id = cnt.cnt_id
GROUP BY cust.cust_nm, cust.m_status
ORDER BY 1,2,3;
```

The results of this query are shown. Note that there are two rows for Jane Doe; one row for a marital status of Married and another for a marital status of Single.

Cust Nm	M Status	Tax
Jane Doe	Married	100
Jane Doe	Single	100
Jim Doe	Married	200
John Doe	Single	200

Example 5-7 As Was Queries for Tax Collection Split by Marital Status

Assuming the "Data used for the examples (page 5-9)", issue the following SQL statement to show the tax collection data split by marital status using an analysis date of January 15, 2012.

```
select
   cust.cust_nm, cust.m_status, sum(fct.tax)
from
   TaxPaid fct,
   (
      select
         cust_act.cust_id, cust_pit.cust_cd, cust_pit.cust_nm,
```

```
         cust_pit.m_status, cust_pit.gender,
         cust_pit.cnty_id, cust_pit.cnty_cd, cust_pit.cnty_nm
      from Customer cust_act
      inner join (
         select
            cust_id, cust_cd, cust_nm,
            m_status, gender,
            cnty_id, cnty_cd, cnty_nm
         from Customer cust_all
         where to_date('15-JAN-2012', 'DD-MON-YYYY') between eff_from and eff_to
      ) cust_pit
      on (cust_act.cust_cd = cust_pit.cust_cd)
   ) cust,
   (
      select
         cnty_act.cnty_id, cnty_pit.cnty_cd, cnty_pit.cnty_nm
      from County cnty_act
      inner join (
         select
            cnty_id, cnty_cd, cnty_nm
         from County cnty_all
         where to_date('15-JAN-2012', 'DD-MON-YYYY') between eff_from and eff_to
      ) cnty_pit
      on (cnty_act.cnty_cd = cnty_pit.cnty_cd)
   ) cnty
where fct.cust_id = cust.cust_id
and cust.cnty_id = cnty.cnty_id
GROUP BY cust.cust_nm, cust.m_status
order by 1,2,3;
```

The results of this query are shown. Since Jane Doe was single on January 15, 2012 (the analysis date), all tax for Jane Doe is accounted under her Single status.

Cust Nm	M Status	Tax
Jane Doe	Single	200
Jim Doe	Married	200
John Doe	Single	200

Assume instead that you issued the exact same query except that for the `to_date` phrase you specify `09-FEB-2012` rather than `15-JAN-2012`. Since Jane Doe was married on February 9, 2012, then, as shown, all tax for Jane Doe would be accounted under her Married status.

Cust Nm	M Status	Tax
Jane Doe	Married	200
Jim Doe	Married	200
John Doe	Single	200

Example 5-8 As Is Query for Tax Collection Data Split by County

Assuming the "Data used for the examples (page 5-9)", issue the following SQL statement to show the tax collection data split by county.

```
SELECT cust.cust_nm, cnty.cnty_nm, SUM(fct.tax)
FROM TaxPaid fct, customer cust, county cnty
```

```
WHERE fct.cust_id = cust.cust_id
AND cust.cnty_id = cnty.cnty_ID
GROUP BY cut.cust_nm, cnty.cnty_nm
ORDER BY 1,2,3;
```

The results of this query are shown. Note that since John Doe lived in two different counties, there are two rows of data for John Doe.

Cust Nm	County Nm	Tax
Jane Doe	Sunnyvale	200
Jim Doe	Cupertino	200
John Doe	Cupertino	100
John Doe	Sunnyvale	100

Example 5-9 As Was Queries for Tax Collection Data Split by County

Assuming the "Data used for the examples (page 5-9)", issue the following SQL statement to show the tax collection data split by county using an analysis date of January 15, 2012.

```
select
   cust.cust_nm, cnty.cnty_nm, sum(fct.tax)
from
  TaxPaid fct,
   (
      select
         cust_act.cust_id, cust_pit.cust_cd, cust_pit.cust_nm,
         cust_pit.m_status, cust_pit.gender,
         cust_pit.cnty_id, cust_pit.cnty_cd, cust_pit.cnty_nm
      from Customer cust_act
      inner join (
         select
            cust_id, cust_cd, cust_nm,
            m_status, gender,
            cnty_id, cnty_cd, cnty_nm
         from Customer cust_all
         where to_date('15-JAN-2012', 'DD-MON-YYYY') between eff_from and eff_to
      ) cust_pit
      on (cust_act.cust_cd = cust_pit.cust_cd
   ) cust,
   (
      select
         cnty_act.cnty_id, cnty_pit.cnty_cd, cnty_pit.cnty_nm
      from County cnty_act
      inner join (
         select
            cnty_id, cnty_cd, cnty_nm
         from County cnty_all
         where to_date('15-JAN-2012', 'DD-MON-YYYY') between eff_from and eff_to
      ) cnty_pit
      on (cnty_act.cnty_cd = cnty_pit.cnty_cd)
   ) cnty
where fct.cust_id = cust.cust_id
and cust.cnty_id = cnty.cnty_id
GROUP BY cust.cust_nm, cnty.cnty_nm
order by 1,2,3;
```

The results of this query are shown. Note that since John Doe was in Sunnyvale as of the analysis date of January 15, 2012, all tax for John Doe is accounted for under the Sunnyvale county.

Cust Nm	County Nm	Tax
Jane Doe	Sunnyvale	200
Jim Doe	Cupertino	200
John Doe	Sunnyvale	200

Assume instead that you issued the exact same query except that for the `to_date` phrase you specify `09-FEB-2012` rather than `15-JAN-2012`. Since John Doe lived in Cupertino on February 9, 2012, then, as shown, all tax for John Doe would be accounted under Cupertino.

Cust Nm	County Nm	Tax
Jane Doe	Sunnyvale	200
Jim Doe	Cupertino	200
John Doe	Cupertino	200

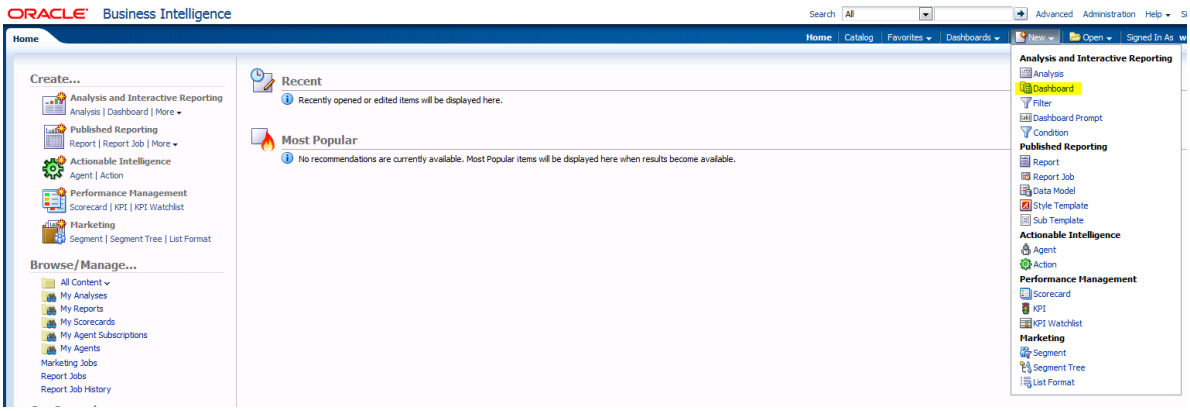
5.7 Tutorial: Creating a New Oracle Utilities Data Model Dashboard

This tutorial explains how to create a dashboard based on the Oracle Utilities Data Model webcat included with the sample reports delivered with Oracle Utilities Data Model.

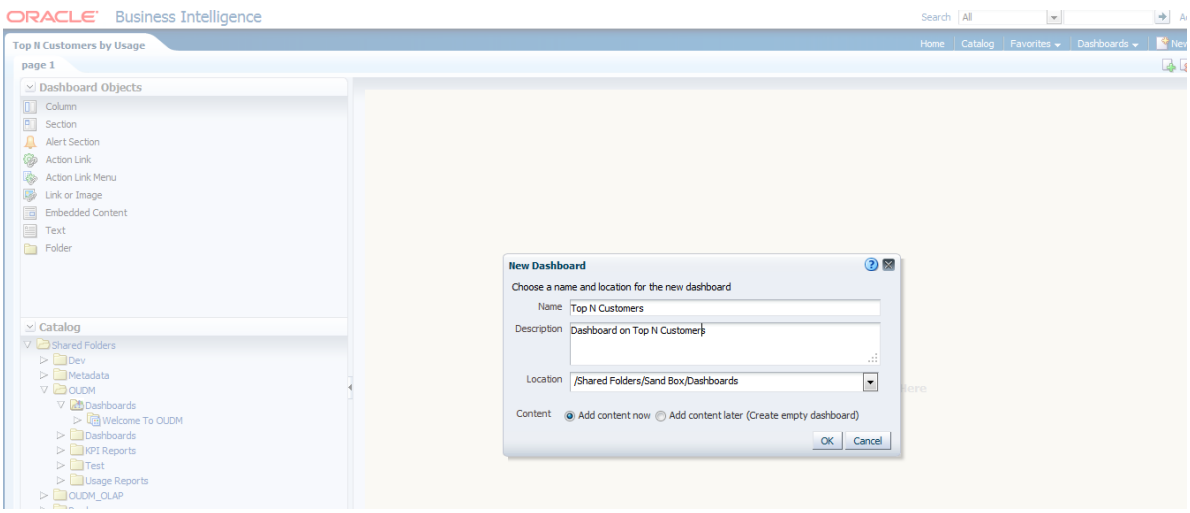
To create a dashboard, take the following steps:

1. In the browser, open the login page at `http://servername:7001/analytics` where *`servername`* is the server on which the webcat is installed.
2. Login with username of `oudm`, and provide the password.

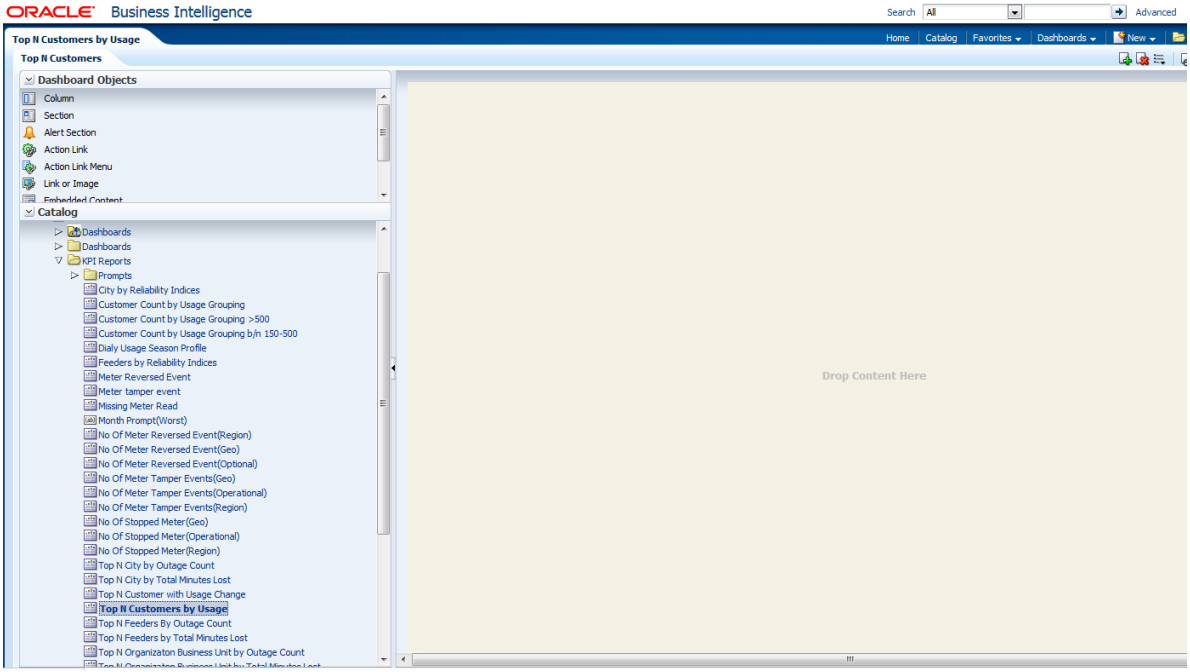
   Then, click **newDashboard** to create an Oracle Business Intelligence Suite Enterprise Edition dashboard.



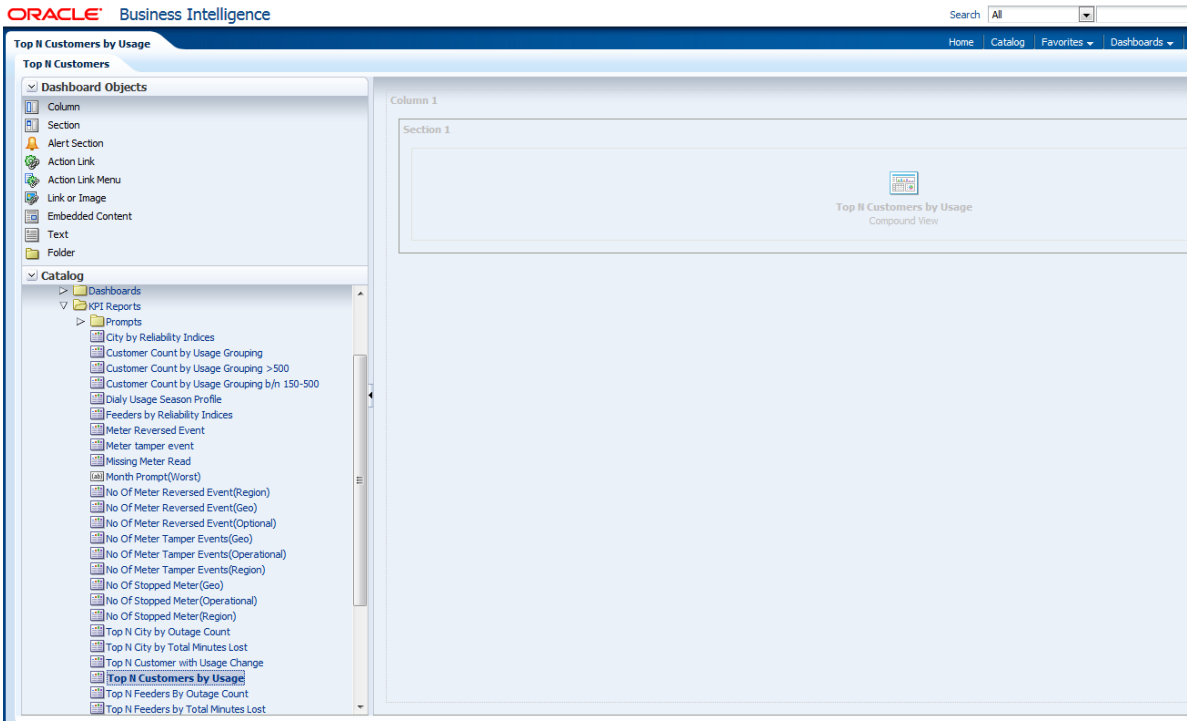
3. Input name and description, save it to the Sandbox folder. Click **OK**.



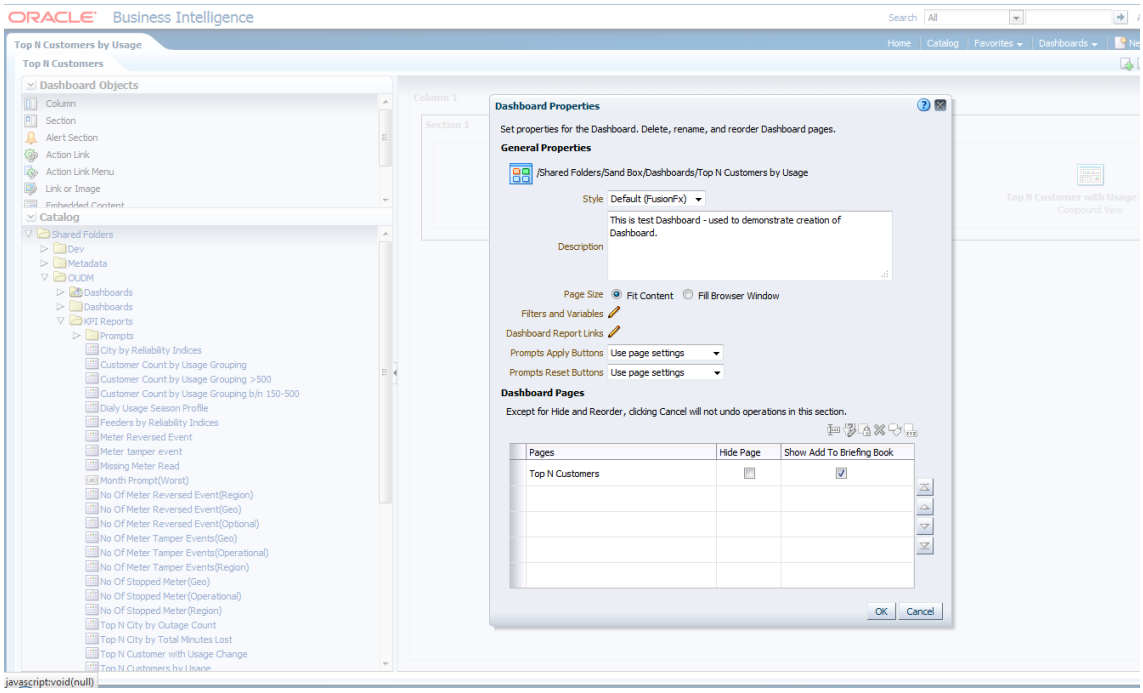
4. In the Catalog view, expand the OUDM folder and KPI Reports folder. You can see Top N Customers by Usage.



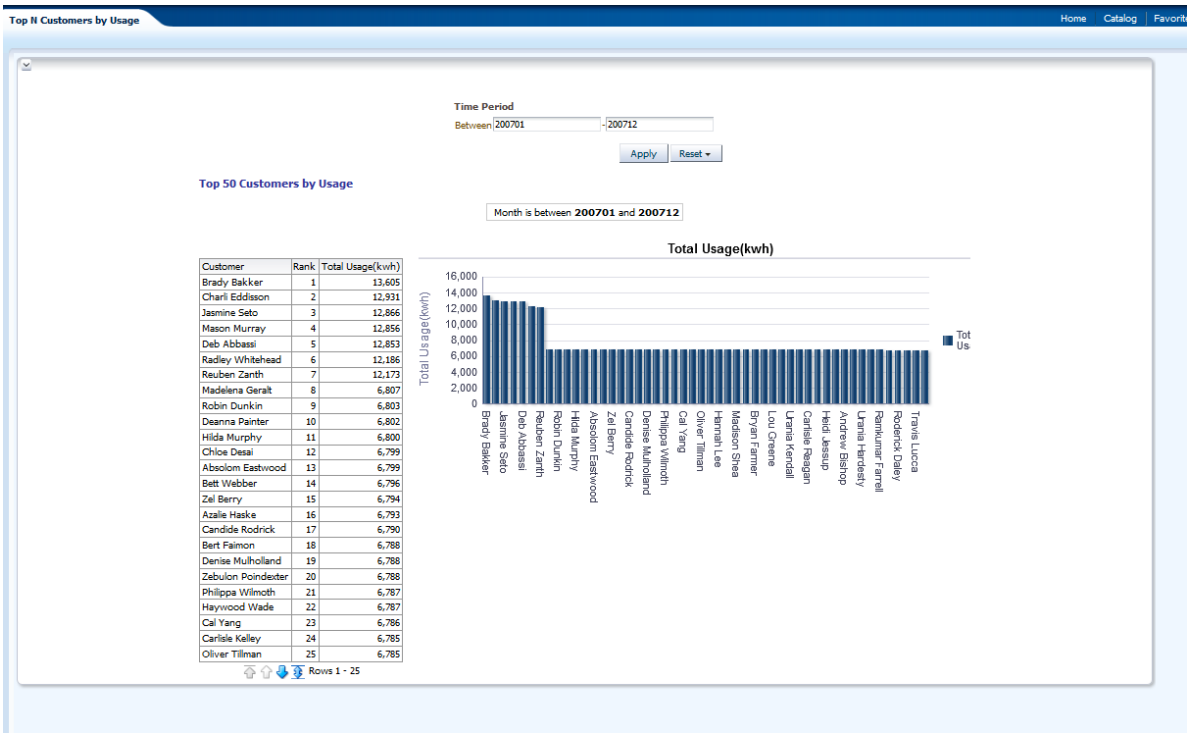
5. Drag the Top N Customers by Usage report into the right panel:



6. To change the page name:
   a. Select the Dashboard.
   b. In Dashboard Properties window, click **Change Name**.



   c. Change the name to "Top N Customers", then click **OK.**
7. Click **Save** on the top of the dashboard. Now you have a new dashboard.



> **Note:**
>
> For more information on creating dashboards see the "Creating Analyses and Dashboards 11*g*" OBE tutorial.
>
> To access the tutorial, open the Oracle Learning Library in your browser at Oracle Learning Library and, then, search for the tutorial by name.

Related Topics:

- *Oracle Utilities Data Model Installation Guide*

5.8 Tutorial: Creating a New Oracle Utilities Data Model Report

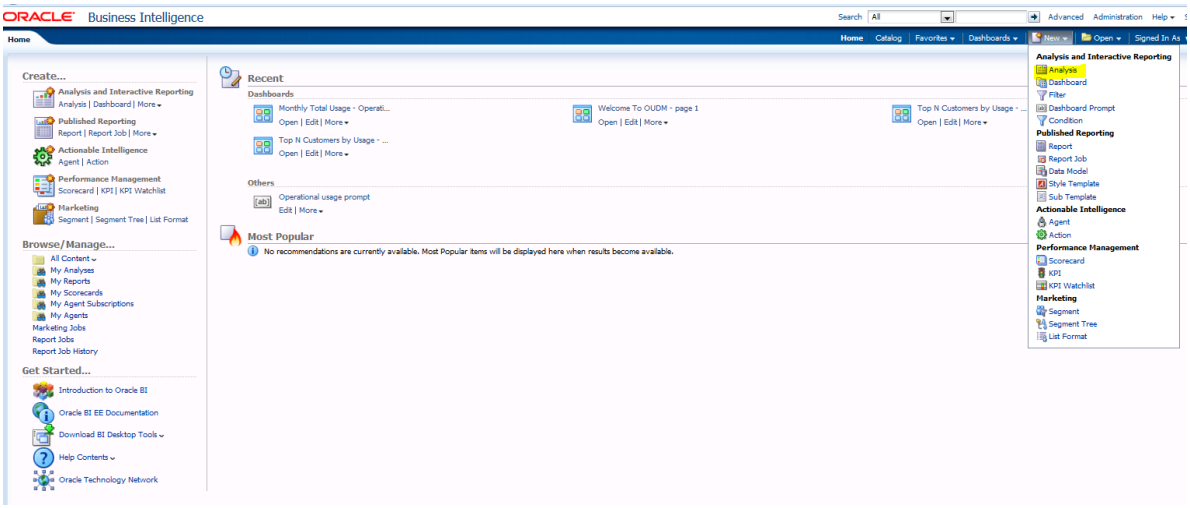
Describes how to create a report based on the Oracle Utilities Data Model webcat included with the sample reports delivered with Oracle Utilities Data Model.

In this example, assume that you want to create a report named "Monthly Total Usage".

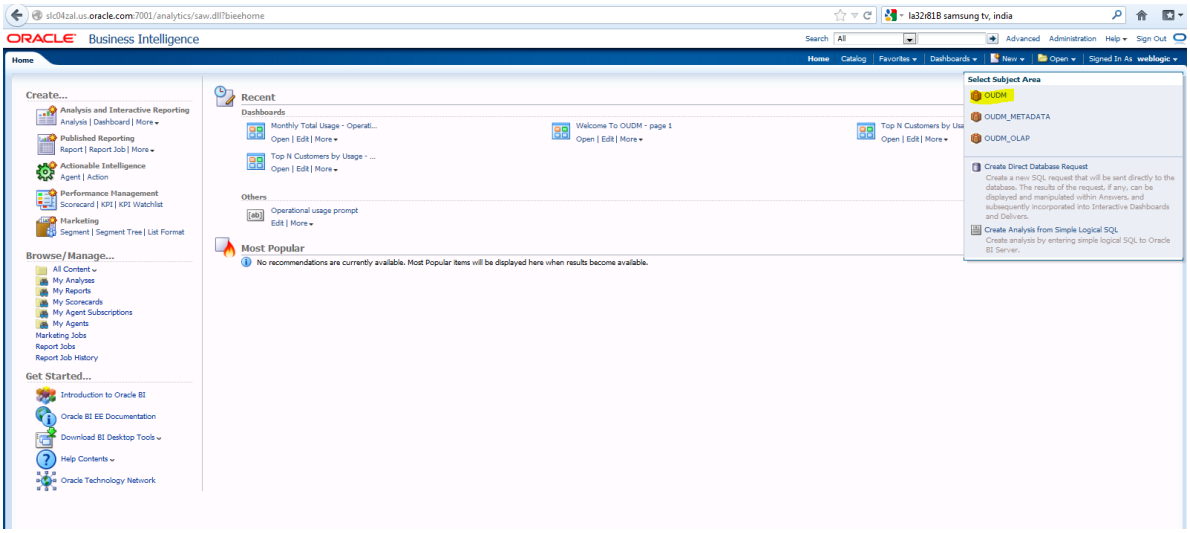
To create a this new report, take the following steps:

1. In the browser, open the login page at `http://servername:7001/analytics` where *`servername`* is the server on which the webcat is installed.
2. Login with username of `oudm`, and provide the password.

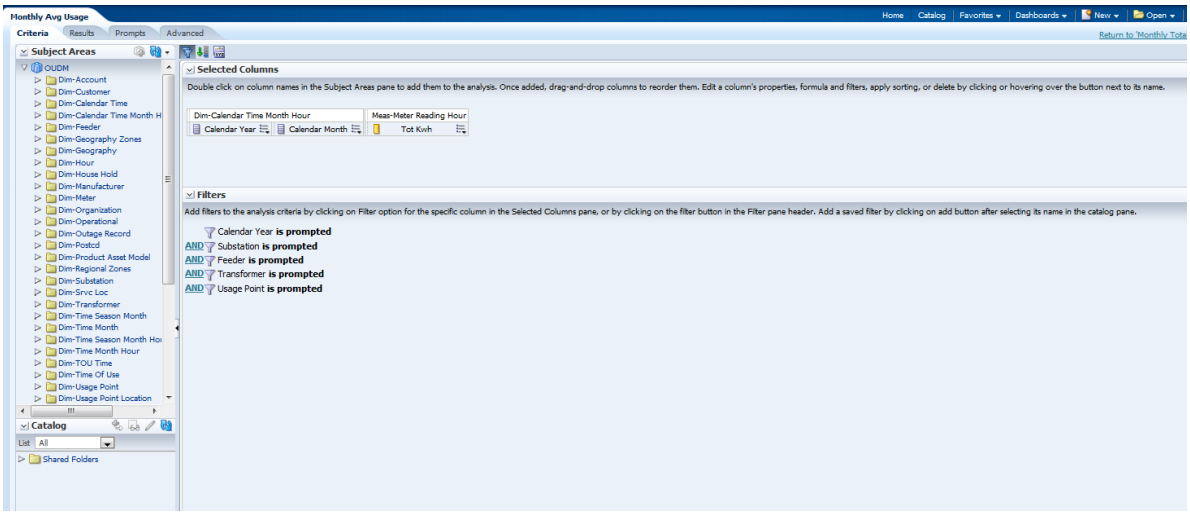
Then, click **newAnalysis** to create an Oracle Business Intelligence Suite Enterprise Edition report.



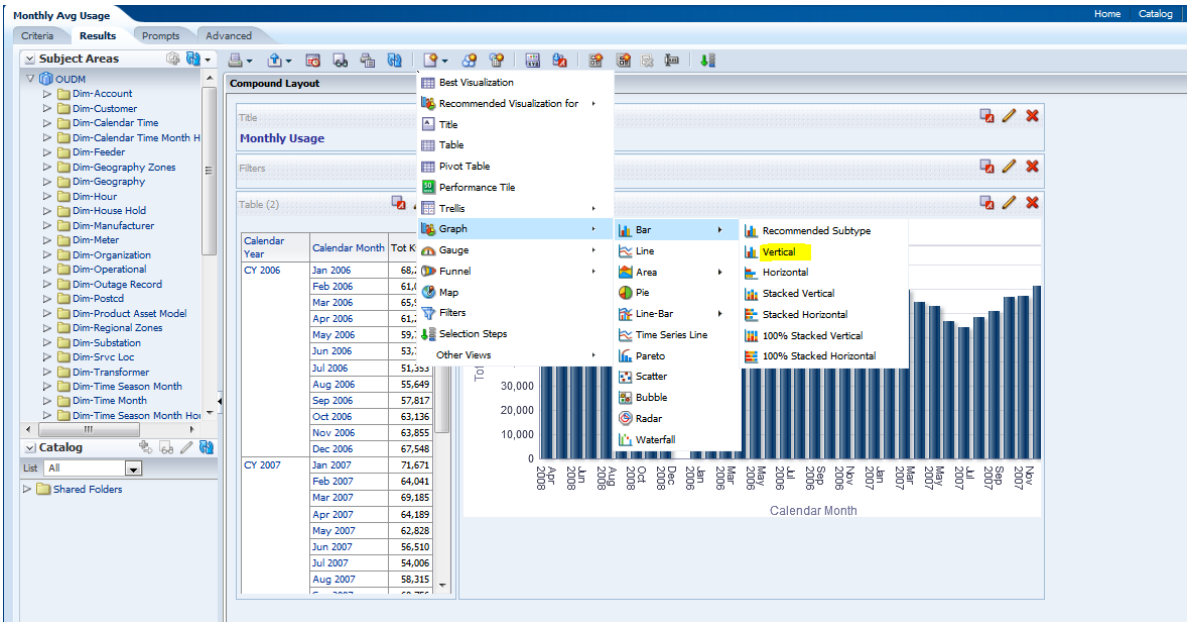
3. Select **Subject Area**, then select **OUDM** to create a relational report.



4. Drag and put the dimension and fact columns into the Select Columns panel.



5. Select the Results tab to view the report. Click **New View** to add a chart into the report:



6. Click **Save** to save this report into one of the desired folder.

> **Note:**
>
> For more information on creating a report, see the "Creating Analyses and Dashboards 11*g*" OBE tutorial.
>
> To access the tutorial, open the Oracle Learning Library in your browser at Oracle Learning Library and, then, search for the tutorial by name.

Related Topics:

- *Oracle Utilities Data Model Installation Guide*

6
Metadata Collection and Reports

This chapter includes the following sections:

6.1 Overview of Managing Metadata for Oracle Utilities Data Model

Metadata is any data about data and, as such, is an important aspect of the data warehouse environment. Metadata allows the end user and the business analyst to navigate through the possibilities at a higher business object level.

Metadata management is a comprehensive, ongoing process of overseeing and actively managing metadata in a central environment which helps an enterprise to identify how data is constructed, what data exists, and what the data means. It is particularly helpful to have good metadata management when customizing Oracle Utilities Data Model so that you can do impact analysis to ensure that changes do not adversely impact data integrity anywhere in your data warehouse.

6.1.1 Metadata Categories and Standards

Metadata is organized into three major categories: **Business metadata**, **Technical metadata**, **Process execution metadata**.

Metadata is organized into three major categories:

- **Business metadata** describes the meaning of data in a business sense. The business interpretation of data elements in the data warehouse is based on the actual table and column names in the database. Business metadata gathers this mapping information, business definitions, and rules information.

- **Technical metadata** represents the technical aspects of data, including attributes such as data types, lengths, lineage, results from data profiling, and so on.
- **Process execution metadata** presents statistics on the results of running the ETL process itself, including measures such as rows loaded successfully, rows rejected, amount of time to load, and so on.

Since metadata is so important in information management, many organizations attempt to standardize metadata at various levels, such as:

- Metadata Encoding and Transmission Standard (METS). A standard for encoding descriptive, administrative, and structural metadata regarding objects within a digital library.
- American National Standards Institute (ANSI). The organization that coordinates the U.S. voluntary standardization and conformity-assessment systems.
- International Organization for Standardization (ISO). The body that establishes, develops, and promotes standards for international exchange.
- Common Warehouse Metamodel (CWM). A specification, released and owned by the Object Management Group, for modeling metadata for relational, non-relational, multi-dimensional, and most other objects found in a data warehousing environment.

When you implement your metadata management solution, reference your data warehouse infrastructure environment and make the decision which standard to follow.

6.1.2 Working with a Metadata Repository

You manage metadata using a Metadata Repository. At the highest level, a Metadata Repository includes three layers of information. The layers are defined in the following order:

1. Physical layer: this metadata layer identifies the source data.
2. Business Model and Mapping layer: this metadata layer organizes the physical layer into logical categories and records the appropriate metadata for access to the source data.
3. Presentation layer: this metadata layer exposes the business model entities for end-user access.

The first step in creating a Metadata Repository is to scope your metadata management needs by:

- Identifying the metadata consumers. Typically, there are business consumers and technical consumers.
- Determine the business and technical metadata requirements.
- Aligning metadata requirements to specific data elements and logical data flows.

Then:

- Decide how important each part is.
- Assign responsibility to someone for each piece.
- Decide what constitutes a consistent and working set of metadata
- Where to store, backup, and recover the metadata.
- Ensure that each piece of metadata is available only to those people who need it.

- Quality-assure the metadata and ensure that it is complete and up to date.
- Identify the Metadata Repository to use and how to control that repository from one place

After creating the metadata definitions, review your data architecture to ensure you can acquire, integrate, and maintain the metadata.

As the data keeps on changing in your data warehouse day by day, update the Metadata Repository. When you want to change business rules, definitions, formulas or process (especially when customizing the Oracle Utilities Data Model), your first step is to survey the metadata and do an impact analysis to list all of the attributes in the data warehouse environment that would be affected by a proposed change.

6.2 Browsing Metadata Reports and Dashboard

To customize the Oracle Utilities Data Model model, you must understand the dependencies among Oracle Utilities Data Model components, especially how the report KPIs are mapped to the physical tables and columns. Oracle Utilities Data Model provides a tool, the OUDM Metadata browser that helps you discover these dependencies. When you install Oracle Utilities Data Model with its sample reports, the metadata browser is delivered as a sample Dashboard in the webcat.

There are four tabs (reports) in the Oracle Utilities Data Model Metadata browser. To browse the metadata repository:

1. In the browser, open the login page at `http://servername:9704/analytics` where *`servername`* is the server on which the webcat is installed.
2. Login with username of `oudm`, and provide the password.
3. Select the Metadata Browser dashboard.
4. Use the tabs in the Metadata browser to explore the metadata.
    - **Measure-Entity tab**

      On the Measure-Entity tab you can see the business areas (relational, OLAP, mining), the measures description, corresponding formula, responsible entities, and attributes for the measure.
    - **Entity-Measure tab**

      Using the Entity-Measure tab, you can discover the mappings between entities, attributes, supported measures, and calculations of the measures. You can discover information about particular entities and attributes.
    - **Program-Table tab**

      Using the Program-Table tab you can browse for information on the intra-ETL mappings and report information. Take the following steps:
    - **Table-Program tab**

      By default when you go to the Table-Program tab you see all of the tables used for all the reports.

      To discover what reports use a particular table, you must move a particular table from the right pane to the left (Selected) pane.

- Using the Measure-Entity Tab Business Areas and Measures Attributes and Entities (page 6-4)

6.2.1 Using the Measure-Entity Tab Business Areas and Measures Attributes and Entities

The **Measure-Entity** tab provides information on the measure descriptions, computational formulas with physical columns, physical tables, and corresponding entities by Business Area.

To browse the **Measure-Entity** data, select the business area and measure description that you are interested in.

6.2.2 Using the Entity-Measure Tab Entity to Attribute Measures

The **Entity-Measure** tab displays the measures supported by the entities and how they are calculated. You can discover information about particular entities and attributes.

To view the **Entity-Measure** tab perform the following steps to learn more about an entity:

1. Select the entity.
2. Click **GO**.

6.2.3 Using the Program-Table Tab

The **Program-Table** tab displays the input and output tables used in the selected programs.

To use the Program-Table tab, perform the following steps to learn more about intra-ETL mappings:

1. Select the program type (that is, intra-ETL or report) and program name for showing particular report or intra-ETL information.
2. Select **GO**.

6.2.4 Using the Table-Program Tab

The **Table-Program** tab lists the Programs used by a given table and whether that table is an input or output, or both, of that program.

To discover what reports use a particular table, move a particular table from the right pane to the left (Selected) pane.

To see the reports that use a particular table, perform the following steps:

1. In the right pane of the **Table-Program** tab, select the table.

2. Move the table to the Selected list on the left by clicking on **<** (left arrow), and click **OK**.
3. Select **GO**.

   The reports for the selected table are displayed.

6.3 Collecting and Populating Metadata

The Oracle Utilities Data Model metadata browser generation packages generate and update the Oracle Utilities Data Model metadata. The metadata generation package contains four main tables and several staging tables and views. The metadata generation tables are:

- MD_ENTY
- MD_PRG
- MD_KPI
- MD_REF_ENTY_KPI

Use the following steps to collect and populate the metadata.

1. Collect LDM Metadata:

   Extract the Logical Data Model repository metadata from Oracle SQL Developer Data Modeler (OSDM) into a database schema. Use manual steps to generate Logical Data Model repository tables in the database with Oracle SQL Developer Data Modeler.

   a. Start Oracle SQL Developer Data Modeler
   b. Open Logical Data Model
   c. Select **File**.
   d. Select **Export**.
   e. Select **To Reporting Schema**.

2. Collect Sample Dashboard Metadata:

   Extract the BIEE dashboard metadata from webcat to csv file.

   Using OBIEE catalog manager open the SQL Developer sample report webcat:

   Tools -> create Report -> Select type to report on -> select dashboard

   Select columns one by one as shown in the `md_dashboard.ldr` specified in the meta_data folder, then save as a csv format file, `md_dashboard.csv`.

   Put this file in the meta_data folder.

   Column Sequence:

   a. Name
   b. Description
   c. Path
   d. Folder
   e. Analysis Path
   f. Analysis Name

g. Analysis Description

h. Dashboard Page Description

i. Dashboard Page Name

j. Dashboard Page Path

k. Owner

3. Collect Sample Report Metadata:

   Extract BIEE report metadata from webcat to csv file. Use OBIEE catalog manager to open Oracle Utilities Data Model sample report webcat.

   - Tools -> create Report -> Select type to report on -> select Analysis -> select columns one by one as shown in the `md_dashboard.ldr` specified in the meta_data folder.
   - Save the file as csv format, `md_dashboard.csv`. Put the file under meta_data folder

   Column Sequence:

   a. NAME

   b. DESCRIPTION

   c. TABLE_NAME

   d. COLUMN_NAME

   e. FOLDER

   f. PATH

   g. SUBJECT_AREA

   h. FORMULA

4. Collect Sample RPD Metadata:

   Extract BIEE RPD metadata from RPD to csv file. Use Administrator Tool to open Oracle Utilities Data Model sample report RPD:

   - Tools -> Utilities -> Repository Documentation -> Execute -> select location -> set xls file name as `md_rpd`.
   - Save as csv format `md_rpd.csv` and put under meta_data folder.

5. Load Naming Convention Information:

   Load Oracle Utilities Data Model Physical Data Model naming convention information from csv into a staging table. Use `sqlloader` to load data from `name_conversion.csv` into MD_NAME_CONVERSION table. The `sqlloader` format file: `Name_conversion.ldr`

```
Name_conversion.ldr:
OPTIONS (SKIP=1)
LOAD DATA
INFILE      'name_conversion.csv'
BADFILE     'name_conversion.csv.bad'
DISCARDFILE 'name_conversion.csv.dsc'
truncate
INTO TABLE MD_NAME_CONVERSION
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
TRAILING NULLCOLS
(
```

```
ABBREVIATION    ,
FULL_NAME
)
```

6. Load Sample Dashboard Metadata:

   Load sample dashboard metadata from csv into a staging table. Use `sqlloader` to load data from `md_dashboard.csv` into MD_DASHBOARD table. The `sqlloader` format file: `md_dashboard.ldr`.

   ```
   Md_dashboard.ldr:

   OPTIONS (SKIP=1)
   LOAD DATA
   INFILE      'md_dashboard.csv'
   BADFILE     'md_dashboard.csv.bad'
   DISCARDFILE 'md_dashboard.csv.dsc'
   truncate
   INTO TABLE MD_DASHBOARD
   FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
   TRAILING NULLCOLS
   (
   NAME char(2000),
   DESCRIPTION char(2000),
   PATH char(2000),
   FOLDER char(2000),
   ANALYSIS_PATH char(2000),
   ANALYSIS_NAME char(2000),
   ANALYSIS_DESCRIPTION char(2000),
   DASHBOARD_PAGE_DESCRIPTION char(2000),
   DASHBOARD_PAGE_NAME char(2000),
   DASHBOARD_PAGE_PATH char(2000),
   OWNER char(2000)
   )
   ```

7. Load Sample Report Metadata

   Load sample report metadata from csv into a staging table. Use `sqlloader` to load data from `md_report.csv` into MD_REPORT table. The `sqlloader` format file: `md_report.ldr`.

   ```
   Md_dashboard.ldr:

   OPTIONS (SKIP=1)
   LOAD DATA
   INFILE      'md_dashboard.csv'
   BADFILE     'md_dashboard.csv.bad'
   DISCARDFILE 'md_dashboard.csv.dsc'
   truncate
   INTO TABLE MD_DASHBOARD
   FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
   TRAILING NULLCOLS
   (
   NAME char(2000),
   DESCRIPTION char(2000),
   PATH char(2000),
   FOLDER char(2000),
   ANALYSIS_PATH char(2000),
   ANALYSIS_NAME char(2000),
   ANALYSIS_DESCRIPTION char(2000),
   DASHBOARD_PAGE_DESCRIPTION char(2000),
   DASHBOARD_PAGE_NAME char(2000),
   DASHBOARD_PAGE_PATH char(2000),
   ```

```
OWNER char(2000)
)
```

8. Load Sample RPD Metadata:

   Load sample RPD metadata from csv into a staging table.

> **Note:**
>
> If the OLAP part of the RPD is populated by the BIEE native OLAP import. Then the metadata of this part will not be shown in `md_rpd.csv`. You need to manually populate this part of metadata from the RPD.

   Use `sqlloader` to load data from `md_rpd.csv` into MD_RPD table. The `sqlloader` format file: `md_rpd.ldr`.

```
Md_rpd.ldr:

OPTIONS (SKIP=0)
LOAD DATA
INFILE      'md_rpd.csv'
BADFILE     'md_rpd.csv.bad'
DISCARDFILE 'md_rpd.csv.dsc'
truncate
INTO TABLE MD_RPD
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
TRAILING NULLCOLS
(
 SUBJECT_AREA
,PRESENTATION_TABLE
,PRESENTATION_COLUMN char(500)
,DESC_PRESENTATION_COLUMN
,BUSINESS_MODEL
,DERIVED_LOGICAL_TABLE
,DERIVED_LOGICAL_COLUMN
,DESC_DERIVED_LOGICAL_COLUMN
,EXPRESSION char(1000)
,LOGICAL_TABLE
,LOGICAL_COLUMN
,DESC_LOGICAL_COLUMN
,LOGICAL_TABLE_SOURCE
,EXPRESSION_1 char(1000)
,INITIALIZATION_BLOCK
,VARIABLE
,DATABASE
,PHYSICAL_CATALOG
,PHYSICAL_SCHEMA
,PHYSICAL_TABLE
,ALIAS
,PHYSICAL_COLUMN
,DESC_PHYSICAL_COLUMN
)
```

9. Load LDM/PDM Metadata (Table MD_ENTY):

   Load LDM/PDM mapping and related information into table MD_ENTY.

10. Load Program (Intra-ETL) Metadata (Table MD_PRG):

    Load Intra-ETL program input/output and related information into table MD_PRG.

11. Load Reports and KPI Metadata (Table - MD_KPI and MD_REF_ENTY_KPI)

    Load sample report metadata into MD_KPI and load report/PDM/LDM mapping related information into table MD_REF_ENTY_KPI.

6.3.1 Load LDM/PDM Metadata (Table MD_ENTY)

If you want to get the mapping between a business area and an entity, you have to manually populate this information. You can only get this information from the metadata report for those entities which are used in the report, for those entities which are not used in report, you have to manually map them to the correct business area.

Source Tables Required

Source Table Name	Description
DMRS_ATTRIBUTES	Containing attributes of the particular entity
DMRS_ENTITIES	Containing entity name with unique id
MD_NAME_CONVERSION	Containing full name and abbreviation of the distinct word used in the LDM

Staging Tables/Views

Staging Table/View Name	Description
MD_OIDM_ATTR_COL_NAME_MAP	Used to store abbreviate the column names based on the standard abbreviation used in the project.
MD_DM_ALL_ENT_ATTR	Used to generate and keep the entity description.

Loading MD_ENTY (MD_ENTY_POP.SQL)

6.3.1.1 GIVE_ABBRV

This database function GIVE_ABBRV provides the abbreviation for a named token from the table MD_NAME_CONVERSION.

Type: Function

This database function GIVE_ABBRV provides the abbreviation for a named token from the table MD_NAME_CONVERSION.

Source Table

MD_NAME_CONVERSION

Columns: ABBREVIATION

Target

Table: MD_OIDM_ATTR_COL_NAME_MAP

Columns: column_name_abbr

6.3.1.2 MD_DM_ALL_ENT_ATTR

Type: View

This database view provides the description of each entity.

Source Table	Target View
DMRS_ENTITIES	MD_DM_ALL_ENT_ATTR

6.3.1.3 PL/SQL Program to Update Column Name

Type: PL/SQL Program

This program updates the column name based on the result of function GIVE_ABBRV.

Source Tables	Target Table
MD_OIDM_ATTR_COL_NAME_MAP DMRS_ATTRIBUTES	MD_OIDM_ATTR_COL_NAME_MAP Column: column_name_abbr

6.3.1.4 PL/SQL program to insert initial data into MD_OIDM_ATTR_COL_NAM

Type: PL/SQL Program

Provides initial loading for table MD_OIDM_ATTR_COL_NAME_MAP

Source Tables	Target Table
MD_DM_ALL_ENT_ATTR DMRS_ENTITIES	MD_OIDM_ATTR_COL_NAME_MAP

6.3.1.5 PL/SQL program to load data into MD_ENTY

Type: PL/SQL Program

Loads data into MD_ENTY from all the staging tables.

Source Table	Target Table
MD_OIDM_ATTR_COL_NAME_MAP	MD_ENTY

6.3.2 Load Program (Intra-ETL) Metadata (Table MD_PRG)

Source Tables Required

Source Table Name	Description
USER_DEPENDENCIES	This database view describes dependencies between procedures, packages, functions, package bodies, and triggers owned by the current user, including dependencies on views created without any database links.
MD_RPD_RPT	This table contains the sample report related information.

Staging Tables/Views

Staging Table/View Name	Description
MD_INTRA_ETL	Used to generate and keep the relational/OLAP ETL program metadata information.
MD_MINING	Used to generate and keep the data mining ETL program metadata information.

Loading MD_PRG (MD_PRG_POP.SQL, MD_MIN_PRG_POP.SQL)

Program: MD_INTRA_ETL

Type: View

This view extracts information for relational and OLAP Intra-ETL packages. The structure is the same as MD_PRG.

Source View	Target View
USER_DEPENDENCIES	MD_INTRA_ETL

Program: MD_MINING

Type: View

This view extracts information for the data mining Intra-ETL packages. The structure of the view same as MD_PRG.

Source View	Target View
USER_DEPENDENCIES	MD_MINING

Program: PL/SQL program to load ETL mapping data into MD_PRG.

Type: PL/SQL Program

Load ETL program data into MD_PRG from all the staging views

Source Views	Target Table
MD_INTRA_ETL MD_MINING	MD_PRG

Program: PL/SQL program insert report data into MD_PRG

Type: PL/SQL Program

Load report data into MD_PRG from report staging table.

Source Table	Target Table
MD_RPD_RPT	MD_PRG

6.3.3 Load Reports and KPI Metadata (Table MD_KPI and MD_REF_ENTY_KPI):

Source Tables Required

Source Table Name	Description
MD_RPD	This tables stores all the RPD metadata information, it is directly loaded from md_rpd.csv
MD_REPORT	This tables stores all the report (analysis) metadata information, it is directly loaded from md_report.csv
MD_DASHBOARD	This tables stores all the sample report dashboard metadata information, it's directly loaded from md_dashboard.csv

Staging Tables/Views

Staging Table/View Name	Description
MD_RPD_CALC_PHY	Stores the missing physical tables and columns for derived measures. Wrote a query to find out missing Physical tables and columns for derived measures.
MD_REPORT1	MD_REPORT1 has the same structure of MD_RPT, it is used to store comma separated tables and columns to the new row, by that it can directly join with physical tables and columns from MD_RPD_CALC_PHY.
MD_RPT_DASH	Contains all mappings information between RPD and reports.
MD_RPD_RPT_DASH	Stores all the mappings information of Report, RPD and Dashboard.

Loading MD_KPI and MD_REF_ENTY_KPI (SAMPLE_REP_POP.SQL)

Program: PL/SQL program Insert non calculated columns Data Into MD_RPD_CALC_PHY

Type: PL/SQL Program

This program extracts those base KPIs or non calculated column information and inserts into MD_RPD_CALC_PHY.

Source Table	Target Table
MD_RPD	MD_RPD_CALC_PHY

Program: PROCEDURE Proc_DelmValuePopulate2

Type: Procedure

This procedure loads comma separated data to new row of the MD_REPORT1 table.

Source Table	Target Table
MD_REPORT	MD_REPORT1

Program: PL/SQL program to create and perform initial load of data into MD_RPD_RPT

Type: PL/SQL Program

This program creates and performs initial load of data for the table MD_RPD_RPT.

Source Tables	Target Table
MD_RPD_CALC_PHY MD_REPORT1	MD_RPD_RPT

Program: PL/SQL program to create and initial load data into MD_RPD_RPT_DASH.

Type: PL/SQL Program

This program creates and performs initial load of data for table MD_RPD_RPT_DASH.

Source Tables	Target Table
MD_RPD_CALC_PHY MD_RPT_DASH MD_RPD_RPT_DASH	MD_RPD_RPT_DASH

Program: PL/SQL program to create and initial load data into MD_RPD_RPT.

Type: PL/SQL Program

This program creates performs initial load of data for table MD_RPD_RPT.

Source Tables	Target Table
MD_RPD_CALC_PHY MD_REPORT1	MD_RPD_RPT

Program: MD_DRVD_KP

Type: View

This view extracts and keeps the information for all the calculated KPIs.

Source Table	Target Table
MD_RPD_RPT_DASH	MD_DRVD_KPI

Program: PL/SQL program to create and performs initial load of data into MD_KPI.

Type: PL/SQL Program

This program creates and performs initial load of data for table MD_KPI.

Source Table	Target Table
MD_RPD_RPT_DASH	MD_KPI

Program: PL/SQL program to create and initial load data into MD_REF_ENTY_KPI.

Type: PL/SQL Program

This program creates and performs the initial load of data for table MD_REF_ENTY_KPI.

Source Table	Target Table
MD_RPD_RPT_DASHI	MD_REF_ENTY_KPI

7

Working with User Roles and Privileges in Oracle Utilities Data Model

This chapter provides information about managing user roles and privileges in Oracle Utilities Data Model.

- Accounts Created for Oracle Utilities Data Model (page 7-1)
  Installing the Oracle Utilities Data Model component creates the accounts: `oudm_sys`, `oudm_user`, and `oudm_report`. Installing the Oracle Utilities Data Model sample reports creates the `oudm_sample` account. Oracle Utilities Data Model installation creates database accounts with UNLOCK and PASSWORD EXPIRE. Ensure that you unlock these accounts and set new passwords following the post-installation steps.
- Roles and Privileges in Oracle Utilities Data Model (page 7-2)
  The installation process grants the necessary roles and privileges required for users of the default accounts.
- When You Must Consider User Privileges in an Oracle Utilities Data Model (page 7-3)
  The installation process grants the necessary privileges required for users of the default accounts (`oudm_sys` and `oudm_sample`).

7.1 Accounts Created for Oracle Utilities Data Model

Installing the Oracle Utilities Data Model component creates the accounts: `oudm_sys`, `oudm_user`, and `oudm_report`. Installing the Oracle Utilities Data Model sample reports creates the `oudm_sample` account. Oracle Utilities Data Model installation creates database accounts with UNLOCK and PASSWORD EXPIRE. Ensure that you unlock these accounts and set new passwords following the post-installation steps.

Oracle Utilities Data Model includes the following:

- `oudm_sys` is the main schema for Oracle Utilities Data Model. This schema contains all the relational and OLAP components of Oracle Utilities Data Model. This schema is the owner of Oracle Utilities Data Model database objects.

  The Oracle Utilities Data Model data mining tables are also in this schema.
- `oudm_sample` is schema owner of Oracle Utilities Data Model database objects with sample data.
- `oudm_user` a database user to invoke Intra-ETL packages and for OLAP data loading and for Data Mining models building.
- `oudm_report` database user for OBIEE services to query data from `oudm_sys` schema and return query results back to OBIEE services.

Related Topics:

- *Oracle Utilities Data Model Installation Guide*
- *Oracle Utilities Data Model Installation Guide*

7.2 Roles and Privileges in Oracle Utilities Data Model

The installation process grants the necessary roles and privileges required for users of the default accounts.

Table 7-1 Default Privileges Granted to OUDM_SYS and OUDM_SAMPLE

Privilege	Justification
create materialized view	To create materialized view in OUDM_SYS and OUDM_SAMPLE schemas.
create procedure	To create procedure in OUDM_SYS and OUDM_SAMPLE schemas.
create sequence	To create sequence in OUDM_SYS and OUDM_SAMPLE schemas.
create session	To create session to execute SQL,PL/SQL scripts as OUDM_SYS and OUDM_SAMPLE users.
create synonym	Synonyms are used as alternative table names.
create table	To create tables in OUDM_SYS and OUDM_SAMPLE schemas.
create type	To create type in OUDM_SYS and OUDM_SAMPLE schemas.
create view	To create view in OUDM_SYS and OUDM_SAMPLE schemas.
create mining model	To create mining model in OUDM_SYS and OUDM_SAMPLE schemas.
execute on ctxsys.ctx_ddl	To use Oracle Text for customer sentiment analysis in OUDM_SYS and OUDM_SAMPLE schemas.
olap_user	To create Analytic Workspace, cubes, and cube dimensions in OUDM_SYS and OUDM_SAMPLE schemas.
create dimension	To create dimensions in OUDM_SYS and OUDM_SAMPLE schemas.
create job	Available for oudm_user to run olap packages.

Table 7-2 Default Privileges Granted to OUDM_USER

Privilege	Justification
create session	To create session to invoke Intra-ETL packages and OLAP data loading.
execute on	Privilege to invoke Intra-ETL packages and OLAP package owned by `OUDM_SYS` schema. Select privilege on `OUDM_SYS` tables, views, cubes, and cube dimensions.
read, update, delete	READ, UPDATE, and DELETE on intra-ETL OUDM configuration tables to run different loads.

Table 7-3 Default Privileges Granted to OUDM_REPORT

Privilege	Justification
create session	To create session to query data from `OUDM_SYS` schema.
read	READ privilege on `OUDM_SYS` tables, views, and materialized views.
select	SELECT privilege on `OUDM_SYS` cubes and cube dimensions.

7.3 When You Must Consider User Privileges in an Oracle Utilities Data Model

The installation process grants the necessary privileges required for users of the default accounts (`oudm_sys` and `oudm_sample`).

After installing the product, you only need to consider user privileges for the following:

- The intra-ETL programs run inside the `oudm_sys` schema, therefore, these programs require the full access to the `oudm_sys` schema. By default, the PL/SQL intra-ETL packages for Oracle Utilities Data Model connect to the `oudm_sys` schema for intra-ETL execution. For security reasons, you may want to grant different privileges, for different purposes, to users of the `oudm_sys` schema.
- By default, the Oracle Utilities Data Model sample reports connect to the `oudm_sys` schema directly. For security reasons, you may want to grant only select privileges to users of the sample reports.
- By default, you connect as `oudm` in OBIEE to access the reports. For security reasons, you may want to create different users in OBIEE for different purposes.
- Granting Only Required Privileges to Database Users of OUDM_SYS (page 7-3)
  Describes the steps to grant only select privileges to users of the `oudm_sys` schema.
- Granting Permission Privileges for Oracle Business Intelligence Suite Extended Edition Reports to Users and Roles (page 7-4)
  Describes the steps to perform to grant permission privileges to users of the Oracle Business Intelligence Suite Extended Edition reports,

7.3.1 Granting Only Required Privileges to Database Users of OUDM_SYS

Describes the steps to grant only select privileges to users of the `oudm_sys` schema.

- Create another role for a different purpose (for example, `OUDM_developer` for Oracle Utilities Data Model customization for a developer who can execute packages and perform dml/ddl operations. Create `OUDM_Viewer` for a report viewer who wants to view data but cannot modify and object or data. Then create the user and grant proper roles.).
- Grant required privilege to different roles (For example, `OUDM_developer` needs execute privilege on etl packages but `oudm_viewer` does not).
- Create users and grant required roles.
- Create a view (or synonym) in user schema that points to the `oudm_sys` tables.

7.3.2 Granting Permission Privileges for Oracle Business Intelligence Suite Extended Edition Reports to Users and Roles

Describes the steps to perform to grant permission privileges to users of the Oracle Business Intelligence Suite Extended Edition reports,

- Create a dedicated report user (for example, `market_manager`).
- Grant required group membership for user `market_manager`.
- Create a role or manage the existing roles and add the user `market_manager` in referenced roles.
- Configure permission privileges of the related reports or dashboards to user `market_manager` or the referenced roles.
- Apply and refresh the Oracle Business Intelligence Suite Extended Edition server.

8

Oracle Utilities Data Model Security

Provides information on Oracle Utilities Data Model security.

- Oracle Utilities Data Model Security Overview (page 8-2)
  Provides an overview of Oracle Utilities Data Model security. When dealing with security, Oracle Utilities Data Model does not provide specific security features but assumes that you leverage the security features of its dependent Oracle products, including Oracle Database and WebLogic Server.
- Architecture Analysis for Security (page 8-6)
  This topic describes areas in Oracle Utilities Data Model that are potential security risks, in order to first be aware of them and second, to secure them.
- Security by Component (page 8-9)
  Outlines security by component. To obtain the highest level of security you must leverage all the possible security options you can within your budget and resources. The default security provided is better than no security, but you should provide enough resources to obtain the highest possible security standard.
- Performing a Secure Oracle Utilities Data Model Installation (page 8-10)
  Presents information to plan for a secure Oracle Utilities Data Model installation. Outlines the steps required to perform a secure installation.
- Implementing Data Model Security (page 8-11)
  Explains the Oracle Utilities Data Model security features for tables and entities. This topic lists security sensitivity at various levels for Oracle Utilities Data Model entities.
- Security Maintenance, Monitoring and Control (page 8-11)
  Provides information on controls for maintaining security for Oracle Business Intelligence Suite Extended Edition and the Oracle Database. You must ensure that you remove or at least lock all unused users at the Oracle Business Intelligence Suite Extended Edition and database level, and perform regular monitoring of user permissions and usage.
- Security Considerations for Developers (page 8-12)
  Provides information for developers about how to create secure applications for Oracle Utilities Data Model, and how to extend Oracle Utilities Data Model without compromising security.
- Dealing with Data Privacy, Data Retention, and other Data Related Rules and Laws (page 8-13)
  Describes regulations and covers dealing with data privacy, other data related rules and laws.
- Database Vault on Oracle Utilities Data Model (page 8-13)
  Describes how to use Database Vault with Oracle Utilities Data Model.
- Data Masking on Oracle Utilities Data Model (page 8-20)
  Describes how to use Data Masking with Oracle Utilities Data Model.

- Transparent Data Encryption in Oracle Utilities Data Model (page 8-22) Describes how to configure Transparent Data Encryption (TDE), and demonstrates using TDE (making one encrypted column and one encrypted tablespace).

8.1 Oracle Utilities Data Model Security Overview

Provides an overview of Oracle Utilities Data Model security. When dealing with security, Oracle Utilities Data Model does not provide specific security features but assumes that you leverage the security features of its dependent Oracle products, including Oracle Database and WebLogic Server.

Oracle Utilities Data Model Security

Oracle Utilities Data Model has no specific security features enabled by default.

Oracle Utilities Data Model is a “normal” data warehouse implemented on top of an Oracle Database (although the data warehouse may only include industry information). Oracle Utilities Data Model is an addition to the Oracle Database and includes all the Oracle Database standard security features, or in some cases Oracle Advanced Security features that should be leveraged (as for any data warehouse containing sensitive data). Oracle Database provides privileged user controls, multi-factor authorization, transparent data encryption, auditing, configuration scanning, and SQL monitoring, and other industry leading security controls.

Read the following documents and apply the security solutions where relevant:

- *Oracle Database Security Guide*
- *Oracle Database Advanced Security Guide*
- *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server* (especially relevant when either WebLogic only or full Oracle Business Intelligence Suite Extended Edition suite is used in combination with Oracle Utilities Data Model).
- *Oracle Application Server Security Guide* (only if used)
- *Oracle Application Server Administrator's Guide* (only if used)

All Oracle documentation is available from Oracle Help Center:

http://docs.oracle.com

There are several areas in Oracle Utilities Data Model where you must take specific care and note the security considerations, including:

- When the sensitivity of the data Oracle Utilities Data Model deals with is an important consideration.
- When the sensitivity of the access granted for some (power) users is an important consideration. For example, for anyone accessing critical application areas (for example campaign management).

Oracle Utilities Data Model Basic Security Considerations

The following principles are fundamental to using any application securely:

- **Keep software up to date.** This includes using the latest product release and any patches that apply to it.

- **Limit privileges as much as possible.** Users should be given only the access necessary to perform their work. User privileges should be reviewed periodically to determine relevance to current work requirements. This also means that no user should have direct access to OUDM_SYS (at least for production).
- **Monitor system activity.** Establish who should access which system components, and how often, and monitor those components. This means that each user should have their own user ID (no generic login). It also means that roles and data access should be restricted and monitored, especially when accessing sensitive or very sensitive data.
- **Install software securely.** For example, use firewalls, secure protocols using TLS (SSL), and secure passwords.
- **Learn about and use the Oracle Utilities Data Model security features.**

> **Note:**
>
> Using Oracle Utilities Data Model the Oracle Database DBA will leverage the standard or advanced security database options to protect Oracle Utilities Data Model content.

- **Use secure development practices.** For example, take advantage of existing database security functionality instead of creating your own application security.
- **Keep up to date on security information.** Oracle regularly issues security-related patch updates and security alerts. You must install all security patches as soon as possible. See the Critical Patch Updates and Security Alerts Web site:

  Critical Patch Updates and Security Alerts

Understanding the Oracle Utilities Data Model Environment

When planning your Oracle Utilities Data Model implementation, consider the following:

- **Which resources must be protected?**
  - You must protect employee and customer personal data, such as credit-card numbers, social security numbers and login, PIN, and so on.
  - You must protect internal data, such as proprietary source code, custom development, and so on. This is not specific to Oracle Utilities Data Model, but you must make sure that any party able to access Oracle Utilities Data Model and its customer paying documentation should have an NDA in place, to avoid being liable for dispatch or reuse of information by 3rd party.
  - You must protect system components from being disabled by external attacks or intentional system overloads. This is not specific to Oracle Utilities Data Model.
- **Who Are You Protecting Data From?**

  The list of sources of possible threats includes a wide range of people and organizations. Threats depend on the data stored. Individuals include employees or an external hacker.

  For example, you must protect your subscribers' data from other subscribers, but someone in your organization might need to access that data to manage it. You

should analyze your workflows to determine who needs access to the data and what for; it is possible that a system administrator can manage your system components without needing to access the system data. Also, hackers continue to target privileged accounts inside and outside of databases.

The following list provides some examples of threats and the possible financial impact:

- **Insider**: typically an employee or a contractor, who tries to abuse the normal rights for “revenge” or for their own benefit. In this case the possible damage is highly dependent on the breadth of access the person has and how malevolent the actions are. The damage can range from self adjustment (of one’s own bills) up to destruction or re-selling of data.
- **Individual**: individual persons outside the organization, not including employees or contractors, that try to get into the organization’s network to test themselves or to test the security for “fun”. These threats can cause significant damage.
- **Criminal organizations**: such organizations target the organization for subscriber data, in particular bank account and credit card data. They may also target the network to take control and blackmail the organization.
- **State-like organizations**: their aim is usually to gather information on their own target(s). organization data is a treasure if they want this data and they do not or cannot go through a standard judicial process to obtain the data.
- **Research organizations**: these organizations try to test security and look for breaches. Their goal is often to publish their finding but such organizations sometimes notify the organization and provide the opportunity (time) to change the system before they publish security vulnerabilities. Such threats, when identified may have a small financial impact if the vulnerability stays under the organization’s control.

The organization has to protect Oracle Utilities Data Model with the highest security standard possible with respect to the possible wide range of threats.

- **What happens if protections on a strategic resource fail?**

  In some cases, a fault in your security scheme is nothing more than an inconvenience. In other cases, a fault might cause great damage to the organization and to your customers. Understanding the security ramifications of each resource allows you to properly protect it.

Oracle Utilities Data Model is an option of the Oracle Database and because it is a physicalized Enterprise Data Warehouse, the security principles of Oracle Utilities Data Model follow the standard security principles valid for any Oracle Database. Additionally, you must apply the security policies associated with the applicable reporting tools. By default, Oracle Utilities Data Model uses the repository and reports with Oracle Business Intelligence Suite Extended Edition(OBIEE), which leverages WebLogic server. Hence, the security standards applied to the Oracle Utilities Data Model reporting layer should be set up as security for WebLogic server (and OBIEE).

Also, security features must protect the servers and their operating systems on which the database for Oracle Utilities Data Model runs, as well as any interfaces and configurations for supplying and accessing the systems or for reports. In particular:

- ETL/ELT servers and ETL GUI (staging area and more)
- Direct feed to Oracle Utilities Data Model (whether with GUI or not)
- Access to the database management (DBA GUI)

- Access to/from Oracle Utilities Data Model
  - For Development Tools: SQL developer, SQL Data Modeler
  - For research and discovery: Oracle Data Mining GUI, Oracle Data Miner (including SNA), Oracle R
  - For cubes management: Oracle Analytical Workspace Manager
  - External feeds to other applications (from Oracle Utilities Data Model directly, for example through a trigger)
  - Configuration Files
- Oracle Utilities Data Model Content
  - Users Traceability and Audit-ability (especially for sensitive data)
  - Encryption and masking (especially for sensitive data and the backups for sensitive data)
  - User access rights (through roles)
- Backup and Disaster Recovery Systems (and high availability architecture).
- BI systems and applications on top of Oracle Utilities Data Model, leveraging its data
  - OBIEE
  - Any other system (for example a campaign management tool)

Depending on which part of Oracle Utilities Data Model has been implemented and filled with data, some or all of the security steps should be applied. Additionally, any customization must be protected as part of the global security processes.

Recommended Deployment Configurations

Typically, as for any data warehouse, there are several environments on which you install Oracle Utilities Data Model:

- **The Test and Development Environment (R&D)**: this area is for testing, development, research, and unit testing with a subset of data. The test and development environment is typically where most customization takes place before being moved to other environments. The test and development area usually contains a full end-to-end implementation (from staging area up to BI reporting). Despite the test and development apparently temporary presence, this environment should be kept available for debugging and for work on extensions to Oracle Utilities Data Model (this area could be reduced when there are no planned project extensions).

> **Note:**
>
> The staging, data warehouse, and BI server do not need to be installed on the same platform and they can have their own test or pre-production and production environment.

- **The Pre-production Environment**: this area is typically provided for scalability testing (load and performance) and global integration testing. The pre-production environment should contain a representative subset of the data set (for example,

containing a few months of history), for a relevant test and for final user acceptance.

- **The Production Environment**: this area is the final end user environment, providing the required performance and capacity to store the complete data set and any required history.

All these environments must be protected equally since they likely will contain copies of similar sensitive data, whatever subset of data you use for the different testing and development or pre-production environments. Additionally, the backup of each of these environments, if any, should be protected as if it was production.

8.2 Architecture Analysis for Security

This topic describes areas in Oracle Utilities Data Model that are potential security risks, in order to first be aware of them and second, to secure them.

Security Boundaries of Trust

Within each boundary shown in Figure 8-1 (page 8-6), assume that whoever accesses the data can be trusted (with “recognized” users or roles). Outside those boundaries, anyone is a potential threat (this is critical from a security analysis perspective).

Figure 8-1 Oracle Utilities Data Model Security Boundaries of Trust

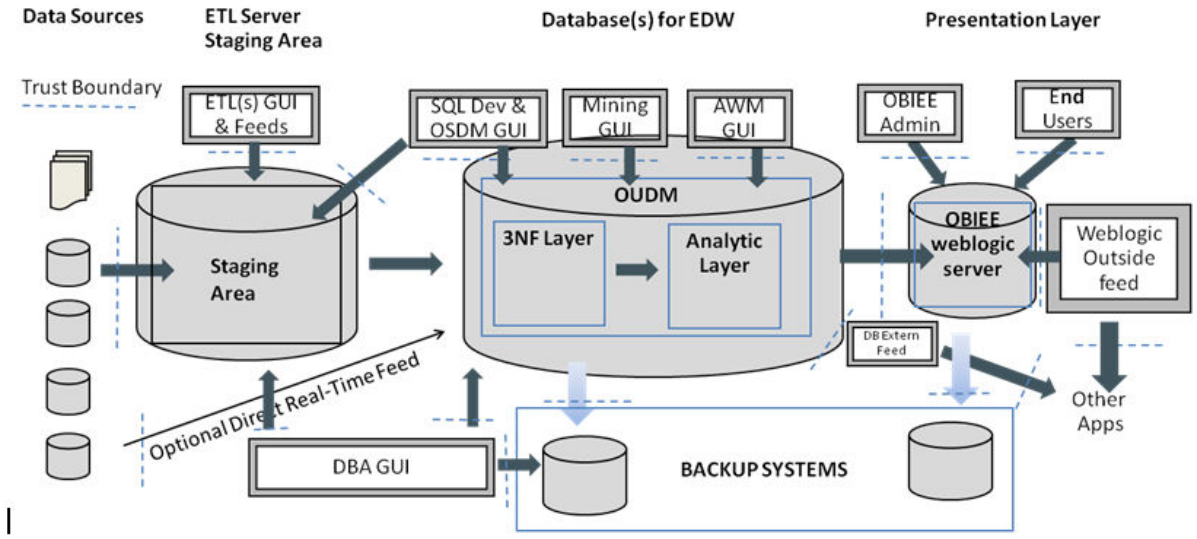


Figure 8-1 (page 8-6) identifies points of interactions at risk, trust boundaries, through which an attack could occur. The interactions at risk are:

1. ETL GUI: Whatever tools and GUI you use for the Extraction, Transform and Load tools that manage the data movement and transformation from the source to staging area, there is clearly a risk of attack:

- Because the tool has access to the raw data source as well as all its transformation rules (Database login data, data source visibility, metadata and transformation rules, mapping).
- Because someone could change the rules (and place unexpected code into Oracle Utilities Data Model).

> **Note:**
>
> If you use Big Data this is also considered part of the “Staging area” even if it also performs operational analysis and reporting. Adding a Big Data environment into the architecture thus includes additional interfaces, and additional boundaries of trust that need to be considered for security and analyzed.

2. SQL developer and Oracle Data Modeler GUI (or any GUI that can directly access the database of either the staging area or the Oracle Utilities Data Model area – hence two accesses at risk). The access at this boundary needs to be restricted (leveraging roles) and monitored (traceability). Additionally, the access string must be protected (do not allow the `tnsnames.ora` to be saved with an unencryption schema password) and the schema password should never be saved with the applications.
3. Data Source feed: when some data from the source system could cause a security breach (through unexpected code or corrupted data, intentional or not, that could break the process). The best way to protect from such an attack is to set up a data profiling check (for example with ODI Data Profiling option or manually by verifying that the input data have the content you expect). Additionally, the schema name and password for the connection should be protected as for 1 and 2 (if not done in the respective GUI).
4. Direct Data Source Feed: this case is similar to case 3, but a direct data source feed assumes a real-time or near real-time direct feed to Oracle Utilities Data Model (to the foundation or analytic layer). The protection should however be similar to 3.
5. DBA GUI access: Due to its rights, the DBA is clearly a role with risk, whatever GUI this person uses (Oracle Enterprise Manager, or any other tools of choice). Leverage the suggestions in *Oracle Database Security Guide* to split the various DBA roles and data access.
6. DBA Backup process and access GUI: This is similar to 5, but for the backup process. Refer to the *Oracle Database Security Guide* for this area, in particular for backup encryption and splitting the backup role from the standard DBA role.
7. Backup process itself: The backup process must be as secure as the Oracle database it is supposed to backup. Configuration files must be encrypted to avoid malevolent use (for example running an unwanted “Restore” or changing the target for a backup). Refer to the security guide of the backup software for such risk analysis.
8. Oracle Advanced Analytics Mining GUI (or any other mining tools GUI): Power users that have access to mining normally also have access to most of the data in clear text (to be able to run the mining and to build the models). They should be treated correspondingly from a data access perspective with much care, as their power, unless explicitly limited within their role, is similar to the DBA. Such power users may also use significant resources (for example CPUs) to run their models,

possibly unwillingly deploying a self-developed model to the full customer base on a production system at an unplanned time, making the data warehouse slow down such that it becomes unusable. On top of a secure access to the database (with hopefully encrypted configuration files), the GUI for mining users should be protected (and access traced), as for the DBA GUI (5).

9. Oracle Analytics Workspace Manager GUI (or any GUI accessing the database OLAP cubes or creating M/R/HOLAP cubes in Oracle Utilities Data Model or accessing data from Oracle Utilities Data Model): Security risk and mitigations should be as for the mining GUI (7).
10. WebLogic or BI server Access to Oracle Utilities Data Model: The BI server that serves as “BI windows” to Oracle Utilities Data Model end users is a boundary of trust, independent of the BI tool used (including WebLogic or other tools) The configuration file and the connection to access Oracle Utilities Data Model should be protected according to the security guidelines for the corresponding tool. For more information for WebLogic guidelines see *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server* .
11. External Data feed (outgoing): When information is directly picked up from Oracle Utilities Data Model database by an external application (campaign management or monitoring tools for example, to obtain the churn probability and to generate the best offer for a customer) or when Oracle Utilities Data Model itself is setup to export on a regular basis directly to a file, using a database trigger or Spool scripts for example, the data exported should be correspondingly:

    - Anonymized (as part of the export process) or encrypted (before being transmitted) according to the sensitivity of the content.
    - Stored in a place that is only accessible for the purpose required, and not changeable, possibly encrypted or unreadable except by the targeted application.
    - In a format and with the data expected (quality check).
    - Removed after use by the target application.

    Check the appropriate database security guide to protect the external feed and the security guide of the application when direct load or query to the database.
12. OBIEE Administrator GUI: The administrator is usually accessing the BI tools and manages the access rights and data accessed as well as the way they are organized and presented to the end-users. With OBIEE, the repository (rpd – default in Oracle Utilities Data Model is `oudm.rpd`) and the webcat files are the most important to protect. See the *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server* and apply the suggestions according to your environment and needs.

    The OBIEE admin should remove or restrict access to OUDM_SAMPLE after it is no longer required, change default password of OUDM RPD, and define all the users (individually), associated with database Users. Additionally, the OBIEE administrator should define various groups with limited rights (at OBIEE level) and associate each user with one of those groups. It is important to implement a 2 level security: .

    - Database level (I can only access the data I am entitled to – independently of what rights I am given at BI level)
    - OBIEE level (I can only access the reports and create new reports or leverage OBIEE features on data I am entitled to see and/or use).

13. OBIEE end-users: The end-users usually access their data through their browser on their mobile device or from their desktop. See the *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server* for more information.
14. WebLogic Outside or Others Incoming feed: this corresponds to the loading of external data to the WebLogic server to combine them with Oracle Utilities Data Model output. It is proper to the WebLogic server and should be handled through *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server* instructions. An example of load is an end-users loading an Excel file with 3rd party data (weather data, market share data, and so on) to enrich its own reports. If this file is corrupted or prepared with a malevolent intention, it can represent a risk for the complete architecture.
15. WebLogic Outside or other (outgoing) feeds: This is exported from WebLogic server to the external world (as MS object or flat file, SMS or email, web agent, and so on). This can become particularly sensitive depending on the information sent or its form of management. Such external feeds must be handled as referred to in *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server*.
16. BI server backup: this is similar to the Oracle Utilities Data Model backup, the BI server backup is to be secured in a similar way as for the Database. Refer to both the Oracle database and *Oracle Fusion Middleware Securing a Production Environment for Oracle WebLogic Server* to securely handle backups.
17. Application GUI that manages the access and data loaded directly from the Oracle Utilities Data Model database: Such applications should be similarly protected (encrypted where possible), monitored and limited (from a data access) as for a normal end-user. Avoid general application (generic) user accounts unless the information is not accessible by anyone else but the application itself (no external communications – internal processes only). Always use the “user ID” that logs into the application and do not use a generic user account.

8.3 Security by Component

Outlines security by component. To obtain the highest level of security you must leverage all the possible security options you can within your budget and resources. The default security provided is better than no security, but you should provide enough resources to obtain the highest possible security standard.

Security Component by Component

- Operating System Security

  This area is not Oracle Utilities Data Model specific. For example, if the Linux OS is used, see the following documents:

  – Guide to the Secure Configuration of Red Hat Enterprise Linux 5

  – Hardening Tips for the Red Hat Enterprise Linux 5

- Oracle Database Security and WebLogic Server Security

  Not Oracle Utilities Data Model specific.

8.4 Performing a Secure Oracle Utilities Data Model Installation

Presents information to plan for a secure Oracle Utilities Data Model installation. Outlines the steps required to perform a secure installation.

Pre-Installation Configuration

On the Oracle Database, after installation of partitioning, Database OLAP, and mining, the simplest recommendation is first to change all the default passwords (in particular of SYS and SYSTEM) and lock all accounts and schema that are not used.

Installing Oracle Utilities Data Model Securely

You can perform a custom installation or a typical installation. Perform a custom installation to avoid installing options and products you do not need. If you perform a typical installation, remove or disable features that you do not need after the installation.

The following are areas with passwords you need to set after installation:

- OUDM_SYS
- OUDM_SAMPLE (to be removed at least for pre-production and production).
- OUDM_USER
- RPD Admin default password
- OUDM _REPORT user Oracle Business Intelligence Suite Extended Edition password

Define the list of people and applications that must access Oracle Utilities Data Model, then determine the data they require the constraints to their access (the complete data set or only a subset), and for each type of profile define a role at the database level with corresponding privileges. (`CREATE ROLE`command, and `GRANT`*ROLE, PRIVILEGEs* `TO` *ROLE / USER* command).

Create as many schemas as you need for each end-user that accesses Oracle Utilities Data Model and specify the correct roles to limit access as required.

When you use Oracle Business Intelligence Suite Extended Edition for reporting, change the passwords of the RPD and also change the user password. Create a user for each end-user, with groups, rights, and roles as you did for the Database but at the Oracle Business Intelligence Suite Extended Edition level.

Post-Installation Configuration

There is nothing specific for this area except that you must encrypt any custom scripts that you create and use with Oracle Utilities Data Model.

Related Topics:

- *Oracle Utilities Data Model Installation Guide*

8.5 Implementing Data Model Security

Explains the Oracle Utilities Data Model security features for tables and entities. This topic lists security sensitivity at various levels for Oracle Utilities Data Model entities.

Implementing Data Model Security

Oracle Utilities Data Model is an option of the database and it does not contain product specific security features. However to secure Oracle Utilities Data Model you must apply the rules of a secured data warehouse as explained in the *Oracle Database Security Guide*. The security rules must be applied and enforced.

The Oracle Utilities Data Model includes entities and some entities must be protected at various levels of security based on the sensitivity of the data they store.

The range for secure entities in the listing ranges from 1 (very sensitive) to 5 (public data).

Table 8-1 Entity Sensitivity Levels

Level	Type of data	Actions recommended
1 Highly sensitive	payment data, and so on.	Encrypt by default (at least the sensitive fields). Audit, do not allow backup (without encryption), limit roles accessing them
2 Sensitive	address, age, resource, and so on.	Mask by default (at least sensitive fields), audit from time to time, limit roles accessing them
3 Somewhat sensitive	No value	Limit access
4 Not sensitive	No value	No value
5 Public	Publicly available data (weather, and so on.)	No value

> **Note:**
>
> By default, any information in Oracle Utilities Data Model is somewhat sensitive. Even if no data is stored in an entity, the entity should be protected from access.

8.6 Security Maintenance, Monitoring and Control

Provides information on controls for maintaining security for Oracle Business Intelligence Suite Extended Edition and the Oracle Database. You must ensure that you remove or at least lock all unused users at the Oracle Business Intelligence Suite

Extended Edition and database level, and perform regular monitoring of user permissions and usage.

Users and Schema Management at Database and Oracle Business Intelligence Suite Extended Edition Level

Remove (or at least lock) all unused users from both the Oracle Business Intelligence Suite Extended Edition and the database level.

Monitor and automatically lock users without activity for the last *N* or 60 days (on Oracle Business Intelligence Suite Extended Edition and on Oracle Utilities Data Model). Providing a regular schedule for locking or resetting for the password for accounts every 120 days, to force a password change, is also a good practice.

8.7 Security Considerations for Developers

Provides information for developers about how to create secure applications for Oracle Utilities Data Model, and how to extend Oracle Utilities Data Model without compromising security.

Security Considerations for Developers

Oracle Utilities Data Model developers work on the following levels:

- **Database level**: In Oracle Utilities Data Model itself, typically for extensions and Customization, from tables to Intra-ETLs up to OLAP cubes or Mining models.
- **ETL level**: In the staging area typically, to move data from source to Oracle Utilities Data Model, while taking care of quality (correctness of content and quantity, timely arrival, meaningfulness for the scope), coherence, latency (or time coherence) and correctness (of mapping to Oracle Utilities Data Model).
- **Oracle Business Intelligence Suite Extended Edition level**: Development (extensions, customizations) of dashboards, reports, alerts, web agents, guided analytics, users and user interfaces.

Each developer should work only on the test environment (except those who test performance in a pre-production environment). Developers should have access to a limited extract of the data that only concerns their scope of work. If possible, provide a dummy set of sample data, as similar to a real data set as possible, for unit testing before testing the system with real data.

If it is not possible to use sample data, in most cases, you must encrypt sensitive data on your site to meet a regulatory compliance for customer and payment information. Also, consider encrypting the source files or input files if sensitive information is stored there.

8.8 Dealing with Data Privacy, Data Retention, and other Data Related Rules and Laws

Describes regulations and covers dealing with data privacy, other data related rules and laws.

Dealing with Data Privacy, Data Retention, and other Data Related Rules and Laws

Each country has its own regulations with respect to data privacy, data allowed and forbidden to be stored and used, and the time for which it must be stored and available for criminal or state inquiry.

For example, in Europe, you must consider the laws at least for data retention, that includes:

- **DIRECTIVE 2006/24/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL - 15 March 2006,** which amended the **Directive 2002/58/EC** for Data privacy of Electronic communications.

These items specify rules for data for quality, security, and confidentiality.

Another concrete example: some fields need to be removed from Oracle Utilities Data Model(or not used at least) in Europe: ETHNICITY field in PARTY is not allowed. The Opt-in and Opt out option of customers must be leveraged by campaign management systems. The Op-in and Opt out options are available in the Social Network Analytics option of Oracle Utilities Data Model.

Because it is locally specific, you should review the Oracle Utilities Data Model content for the project and data you are considering using with your data protection staff (or review the law currently applicable).

8.9 Database Vault on Oracle Utilities Data Model

Describes how to use Database Vault with Oracle Utilities Data Model.

DBA Privileges

It is known that the user who has DBA privilege can see data in OUDM_SYS schema, Oracle Database Vault provides powerful security controls to help protect application data from unauthorized access, and comply with privacy and regulatory requirements.

- Registering Oracle Database Vault with an Oracle Database (page 8-14)
  Describes how to register Oracle Database Vault from SQL*Plus in a non-multitenant environment.
- Verifying That Oracle Database Vault Is Configured and Enabled (page 8-15)
  You can query the V$OPTION dynamic view to verify if Oracle Database Vault is configured and enabled.
- Logging into Oracle Database Vault (page 8-15)
  From Oracle Enterprise Manager Cloud Control (Cloud Control), you can use the Oracle Database Vault pages to administer and monitor Database Vault-protected databases from a centralized console, automate alerts, view Database Vault

reports, and propagate Database Vault policies to other Database Vault-protected databases.

- Securing OUDM_SYS Schema from DBA Access (page 8-16)
  Describes how to secure the OUDM_SYS Schema.

8.9.1 Registering Oracle Database Vault with an Oracle Database

Describes how to register Oracle Database Vault from SQL*Plus in a non-multitenant environment.

Registering Oracle Database Vault with an Oracle Database

You can register Oracle Database Vault from SQL*Plus in a non-multitenant environment.

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

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

2. As user SYS with the SYSDBA administrative privilege, configure the Database Vault user accounts:

```
SQL>CREATE USER dbv_owner IDENTIFIED BY dbv_owner;
SQL>CREATE USER dbv_acctmgr IDENTIFIED BY dbv_acctmgr;
SQL>GRANT CREATE SESSION TO dbv_owner, dbv_acctmgr;
SQL>GRANT SELECT_CATALOG_ROLE to dbv_owner;
```

3. Configure the Database Vault user accounts:

```
SQL>BEGIN
     DVSYS.CONFIGURE_DV (
                         dvowner_uname => 'dbv_owner',
                         dvacctmgr_uname => 'dbv_acctmgr'
                         );
END;
/
```

4. Run the `utlrp.sql` script to recompile invalidated objects:

```
SQL>@$ORACLE_HOME/rdbms/admin/utlrp.sql
```

> **Note:**
>
> If the script gives you any instructions, then follow the instructions and then run the script again. If the script terminates abnormally without giving any instructions, then run the script again.

5. Connect as the Database Vault Owner user that you just configured:

```
SQL> CONNECT dbv_owner
Enter password: password
```

6. Enable Oracle Database Vault:

```
SQL>EXEC DBMS_MACADM.ENABLE_DV;
```

7. Connect with the SYSDBA administrative privilege:

```
SQL>CONNECT / AS SYSDBA
```

8. Restart the database:

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

8.9.2 Verifying That Oracle Database Vault Is Configured and Enabled

You can query the V$OPTION dynamic view to verify if Oracle Database Vault is configured and enabled.

To find if Oracle Database Vault is configured and enabled, run the following query, which should show the VALUE setting as TRUE:

1. Query the V$OPTION dynamic view as follows:

```
SQL>SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Oracle Database Vault';
```

2. To check if Oracle Label Security is enabled, query V$OPTION as follows:

```
SQL>SELECT VALUE FROM V$OPTION WHERE PARAMETER = 'Oracle Label Security';
```

8.9.3 Logging into Oracle Database Vault

From Oracle Enterprise Manager Cloud Control (Cloud Control), you can use the Oracle Database Vault pages to administer and monitor Database Vault-protected databases from a centralized console, automate alerts, view Database Vault reports, and propagate Database Vault policies to other Database Vault-protected databases.

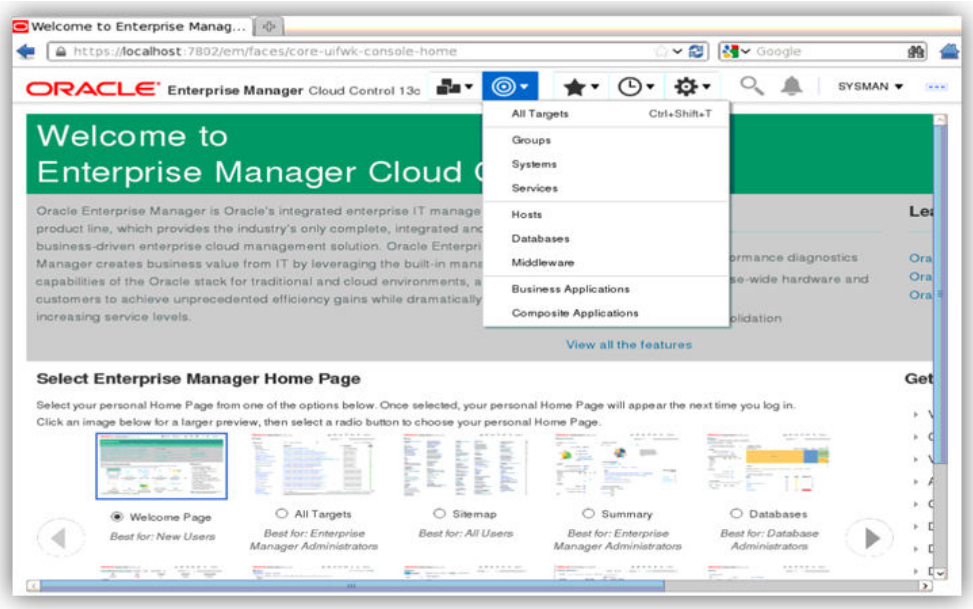
1. Ensure that you have configured the Cloud Control target databases that you plan to use with Database Vault.
2. If necessary, register Oracle Database Vault.
3. Start Cloud Control.

   For example:

   ```
   https://myserver.example.com:7799/em
   ```

4. Log into Cloud Control as user SYSMAN.
5. In the Cloud Control home page, from the **Targets** menu, select **Databases**.

Figure 8-2 Enterprise Manager Cloud Control Console Targets Menu



6. In the Databases page, select the link for the Oracle Database Vault-protected database to which you want to connect. The Database home page appears.
7. From the **Security** menu, select **Database Vault**. The Database Login page appears.
8. Enter the following information:

   Username: Enter the name of a of a user who has been granted the appropriate Oracle Database Vault role(dbv_owner).

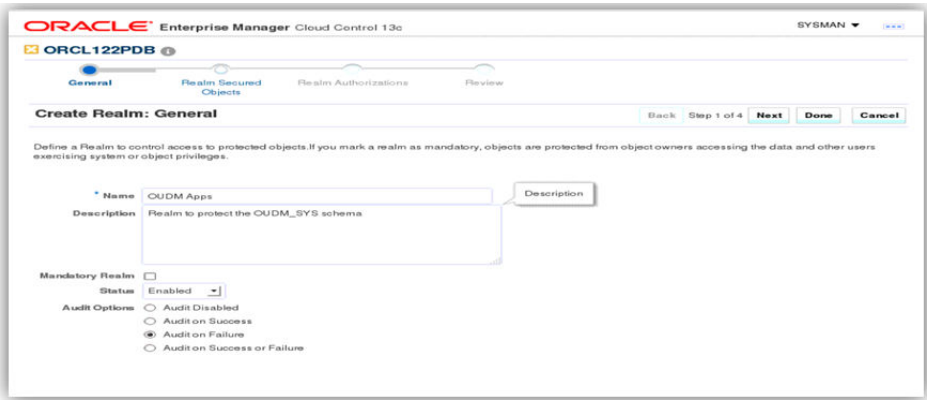
   The Database Vault home page appears.

8.9.4 Securing OUDM_SYS Schema from DBA Access

Describes how to secure the OUDM_SYS Schema.

1. Log On as SYSTEM to access the OUDM_SYS Schema.
2. Create a Realm.
   a. Log into Oracle Database Vault Administrator from Cloud Control as a user who has been granted the `DV_OWNER` or `DV_ADMIN` role and the `SELECT ANY DICTIONARY` privilege.
   b. In the Administration page, under Database Vault Components, click **Realms**. (It should be selected by default.)
   c. In the Realms page of Oracle Database Vault Administrator, click **Create**.
   d. In the Create Realm page, under General, enter OUDM Apps after Name.
   e. In the Description field, enter Realm to protect the OUDM _SYS schema.

Figure 8-3 Create Realm Page



f. After **Status** , ensure that **Enabled** is selected so that the realm can be used.

g. Under **Audit Options**, ensure that **Audit On Failure** is selected so that you can create an audit trial later on.

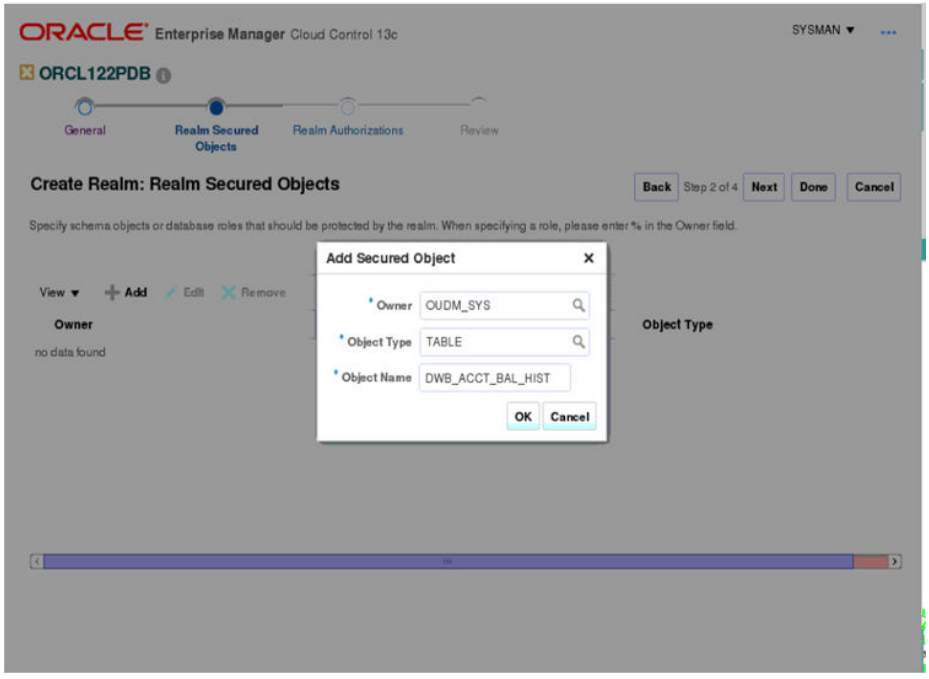
h. Click **Next** to display the Realm secured objects page.

i. Click **Add** and in the Add Secured Object dialog box, enter the following information:

Owner: Enter OUDM_SYS to select the OUDM_SYS schema.
Object Type: Enter TABLE.

Object Name: Enter DWB_ACCT_BAL_HIST.

Figure 8-4 Cloud Control Create Realm Add Secured Object Page



j. Click **OK**.

The OUDM_SYS.DWB_ACCT_BAL_HIST table is added to the Create Realm : Realm Secured Objects page.

k. Click **Done**, and then click **Finish**.

3. Create the OUDM Manager User Account.

   At this stage, there are no database accounts or roles authorized to access or otherwise manipulate the database objects the realm protects. The next step is to authorize database accounts or database roles so that they can have access to the schemas within the realm.

   Create the OUDM_MGR User Account.

   a. In SQL*Plus, connect as the Database Vault Account Manager, with the DV_ACCTMGR role, and create the local user OUDM_MGR:

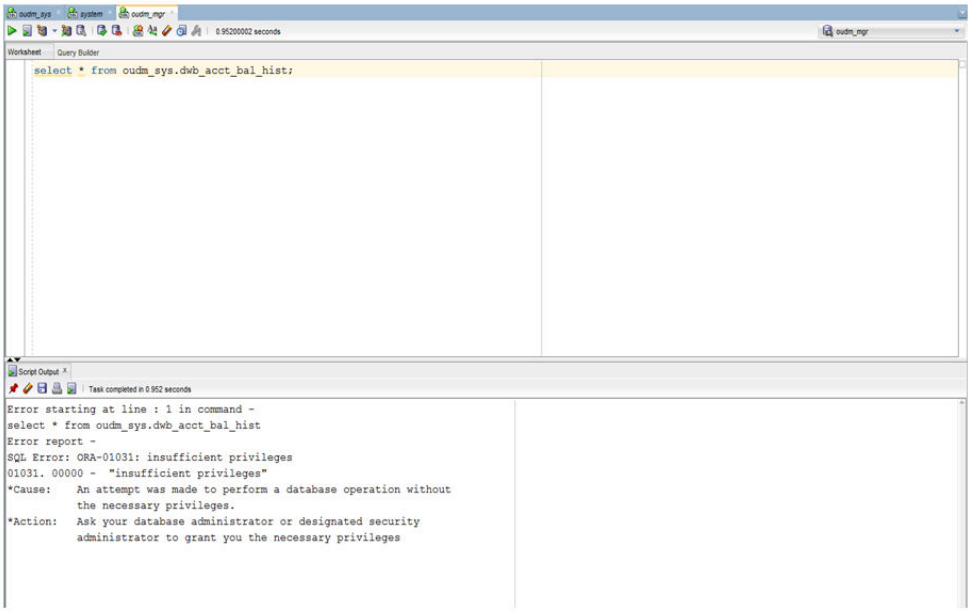
      ```
      SQL>conn dbv_acctmgr/dbv_acctmgr;
      SQL>GRANT CREATE SESSION TO OUDM_MGR identified by OUDM_MGR;
      ```

   b. Connect as SYS with the SYSDBA privilege, and then grant OUDM_MGR the following additional privilege:

      ```
      SQL>conn / as sysdba;
      SQL>GRANT READ ANY TABLE TO OUDM_MGR;
      ```

4. Create an Authorization for the Realm.

   At this stage, even though OUDM_MGR has the SELECT ANY TABLE privilege, he cannot select from the OUDM_SYS.DWB_ACCT_BAL_HIST table because it is protected by a realm.



Next, authorize user OUDM_MGR to access the OUDM Apps realm as follows:

a. In the Realms page of Database Vault Administrator, select the OUDM Apps in the list of realms, and click **Edit**.

b. Click **Next** until you reach the Realm authorizations page.

c. Click **Add** and enter the following information in the Add Authorizations dialog box:

- **Realm Authorization Grantee:** Enter OUDM_MGR.
- **Realm Authorization Type:** Select Participant from the list.
- **Realm Authorization Ruleset:** Leave this field blank.

d. Click **OK**.

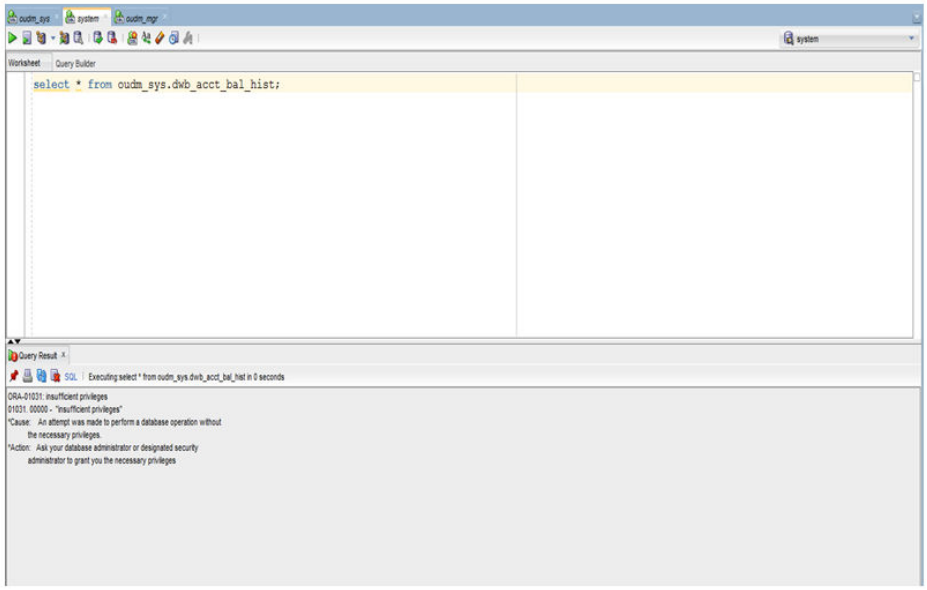
e. Click **Done**, and click **Finish**.

5. Test the Realm.

To test the realm, access the DWB_ACCT_BAL_HIST table as a user other than OUDM_SYS. The SYSTEM account normally has access to all objects in the OUDM schema, but now that you have safeguarded the DWB_ACCT_BAL_HIST table with Oracle Database Vault, this is no longer the case.

a. Connect as SYSTEM, and then access the account balance information in the DWB_ACCT_BAL_HIST table again:

Figure 8-5 Data Vault Access to DWB_ACCT_BAL_HIST with SYSTEM



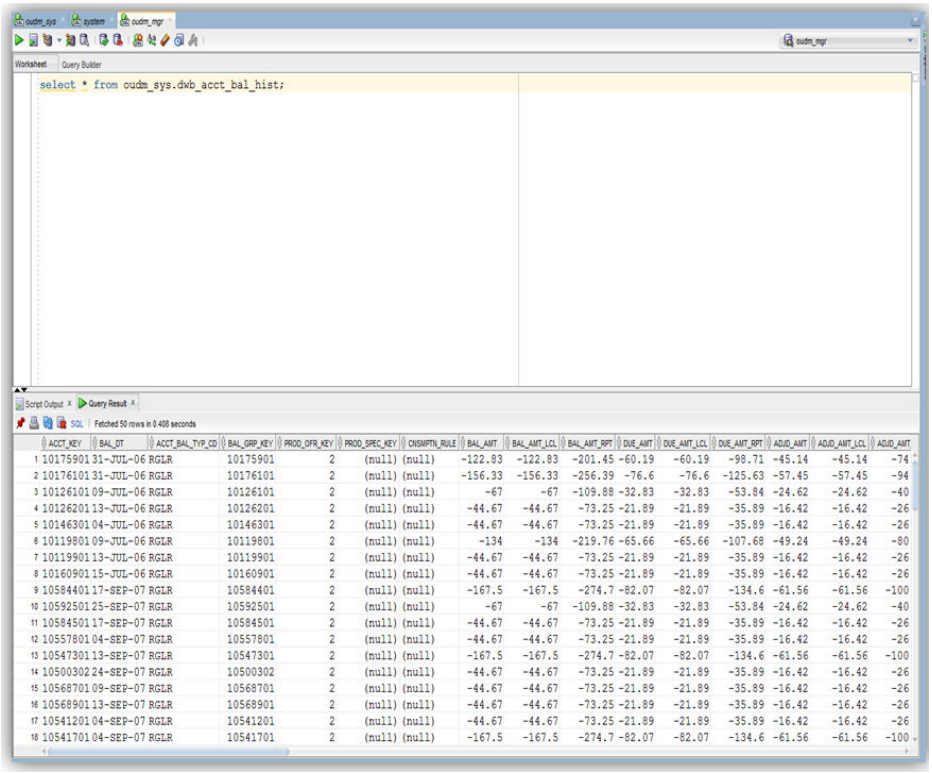
The following output should appear:

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

b. Now, connect as OUDM_MGR, and then try accessing the account balance information in the DWB_ACCT_BAL_HIST table again:

```
SQL>CONNECT OUDM_MGR
Enter password: password
```

Figure 8-6 Data Vault Access to DWB_ACCT_BAL_HIST with OUDM_MGR



Query data in DWB_ACCT_BAL_HIST, data appears.

Related Topics:

- Logging into Oracle Database Vault (page 8-15)
  From Oracle Enterprise Manager Cloud Control (Cloud Control), you can use the Oracle Database Vault pages to administer and monitor Database Vault-protected databases from a centralized console, automate alerts, view Database Vault reports, and propagate Database Vault policies to other Database Vault-protected databases.

8.10 Data Masking on Oracle Utilities Data Model

Describes how to use Data Masking with Oracle Utilities Data Model.

Data Masking Overview

When performing real-world testing, there is the risk of exposing sensitive data to non-production users in a test environment. The Data Masking feature of the Enterprise Manager for Oracle Database Plug-in enables you to securely manage test data.

- Using Data Masking (page 8-21)
  Describes steps for using data masking with Oracle Utilities Data Model.

8.10.1 Using Data Masking

Describes steps for using data masking with Oracle Utilities Data Model.

Assume there is sensitive data showing bank card number information where masking is required before sending the data to a test environment.

1. Go to EM cloud control, Security->Application Data Model.
2. Create Application Data Model.
3. Enter Application Data Model name and select source database.
4. Enter the database credentials and select OUDM_SYS schema.
5. Enter the job name.
6. Job submitted successfully.
7. Job done.
8. Edit the Application Data Model.
9. Add sensitive column.
10. Select column BNK_CARD_NBR in table DWR_BNK_CARD.
11. Sensitive column added.
12. Set sensitive column type.
13. Set sensitive column type to ‘CREDIT_CARD_NUMBER’.
14. Sensitive column type updated.
15. Go to Security->Data Masking and Subsetting->Data Masking Definitions.
16. Data masking definition.
17. Create data masking definition.
18. Add column BNK_CARD_NBR.
19. Set format entry to random number.
20. Set the random number from 100000000 to 999999999.
21. Data masking definition created.
22. Generate data masking script.
23. Schedule data masking script job.
24. Generating masking script.
25. Masking script generating job succeeded.
26. Masking script generated.
27. Schedule data masking job.
28. Data masking job submitted successfully.
29. Data masking job succeeded.
30. Now, check BNK_CARD_NBR, it is masked.

8.11 Transparent Data Encryption in Oracle Utilities Data Model

Describes how to configure Transparent Data Encryption (TDE), and demonstrates using TDE (making one encrypted column and one encrypted tablespace).

Transparent Data Encryption (TDE) stops would-be attackers from bypassing the database and reading sensitive information from storage by enforcing data-at-rest encryption in the database layer. Applications and users authenticated to the database continue to have access to application data transparently (no application code or configuration changes are required), while attacks from OS users attempting to read sensitive data from tablespace files and attacks from thieves attempting to read information from acquired disks or backups are denied access to the clear text data.

- Configuring a Software Keystore (page 8-22)
  Describes the steps required to configure a Software Keystore.
- Demonstration of Oracle Utilities Data Model Working with TDE (page 8-23)
  Demonstrates TDE with one Intra-ETL package (PKG_DWD_ACCT_BAL_MO) as a TDE example.

8.11.1 Configuring a Software Keystore

Describes the steps required to configure a Software Keystore.

Steps show how to configure TDE, and demonstrate after using TDE, making one encrypted column and one encrypted tablespace, how Oracle Utilities Data Model can work transparently.

1. Set the Software Keystore Location in the `sqlnet.ora` file.

```
NAMES.DIRECTORY_PATH (TNSNAME, EXCONNECT)
  ENCRYPTION_WALLET_LOCATION =
    (SOURCE=(METHOD=FILE)(MTHOD_DATA=
     (DIRECTORY=/biaora/home/app/biaora/admin/orcl12102/wallet)))
```

2. Create the Software Keystore:

```
SQL>ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/biaora/home/app/biaora/admin/
orcl12102/wallet' IDENTIFIED BY "password";

keystore altered
```

3. Open the Software Keystore:

```
SQL>ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password";

keystore altered
```

4. Set the master key

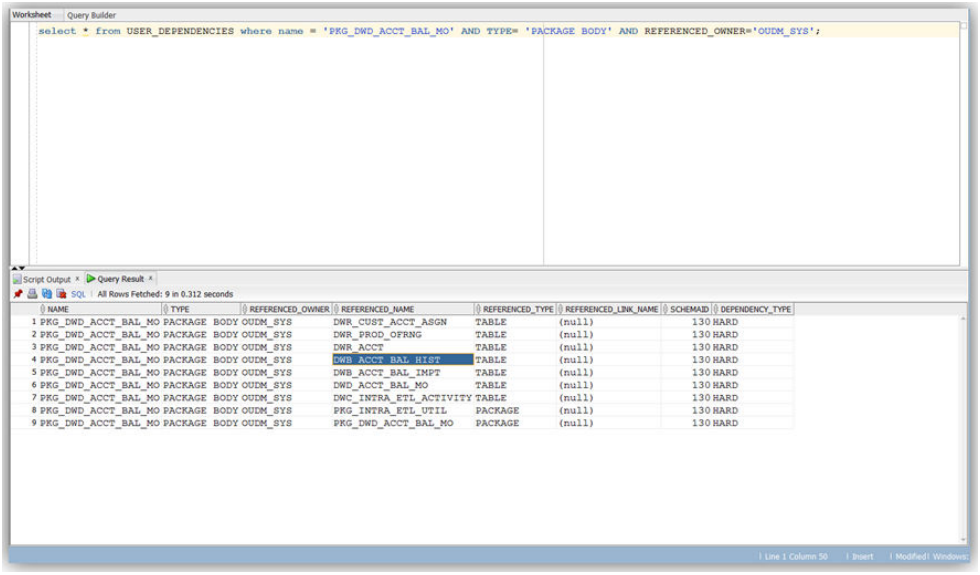
```
 SQL> ADMINISTER KEY MANAGEMENT SET KEY IDENTIFIED BY "password" with backup;
```

8.11.2 Demonstration of Oracle Utilities Data Model Working with TDE

Demonstrates TDE with one Intra-ETL package (PKG_DWD_ACCT_BAL_MO) as a TDE example.

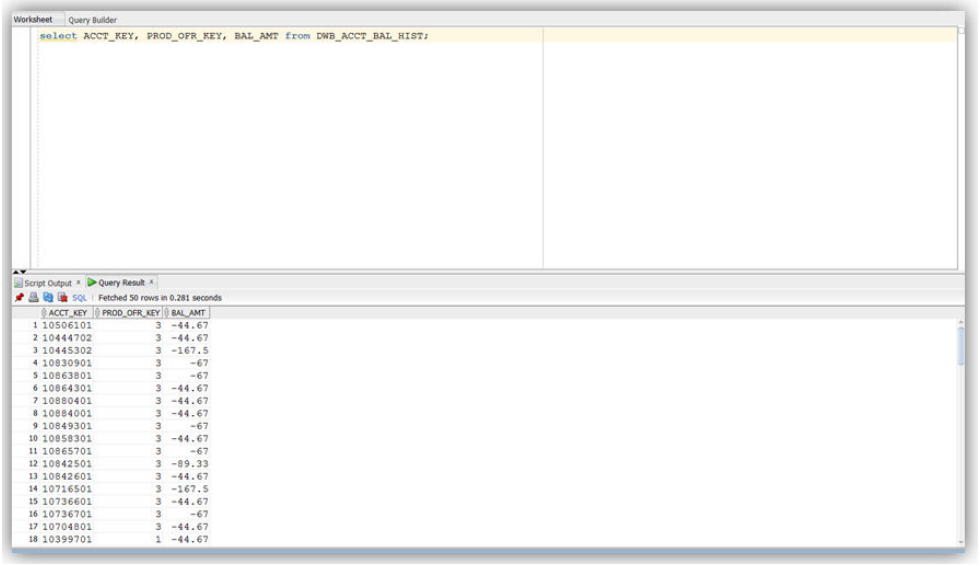
1. Query that shows dependent tables for PKG_DWD_ACCT_BAL_MO.

Figure 8-7 Query Listing Tables for PKG_DWD_ACCT_BAL_MO



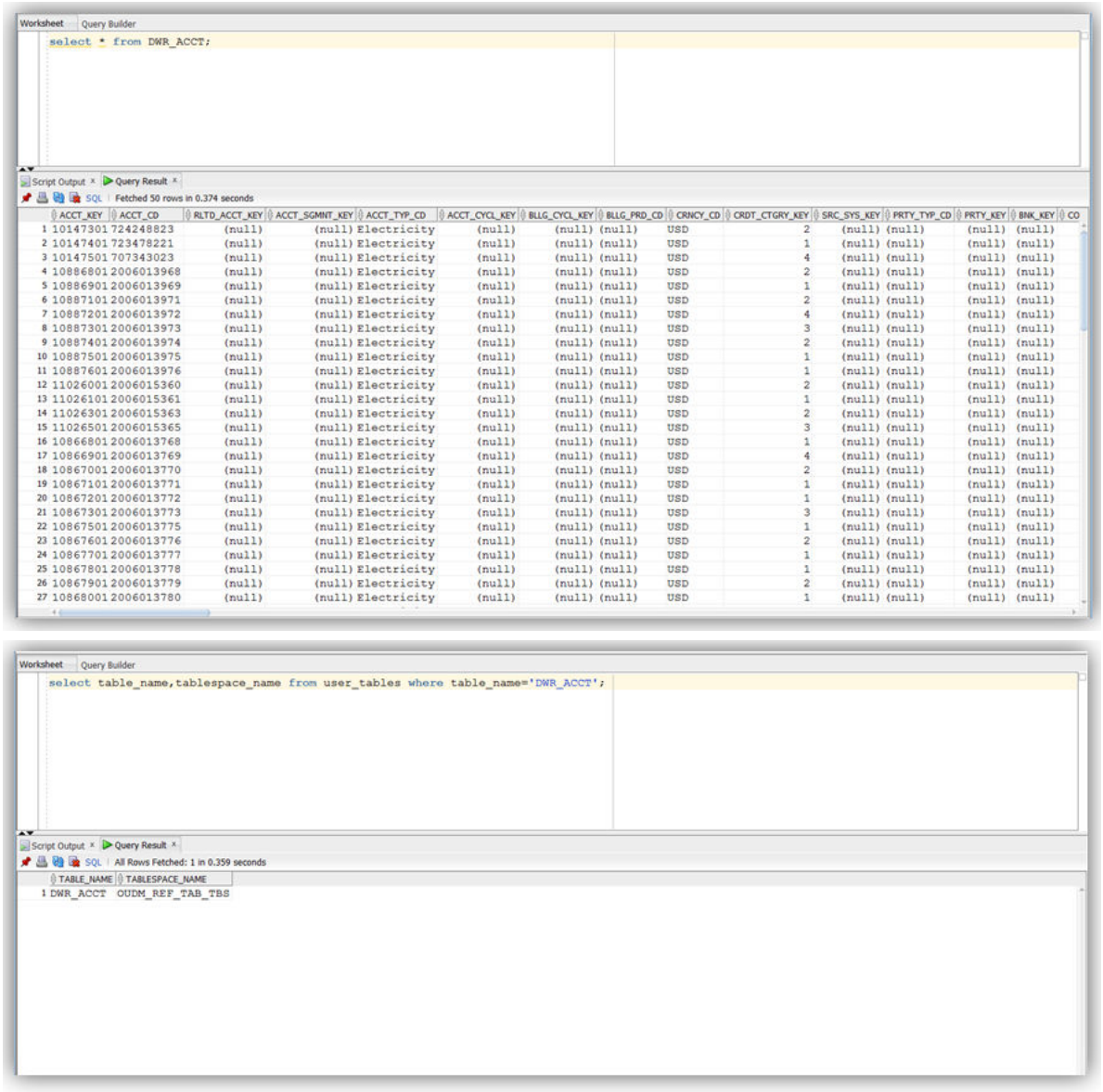
2. Query shows part of data in table DWB_ACCT_BAL_HIST including the column BAL_AMT that is encrypted in a later step.

Figure 8-8 Shows Query for DWB_ACCT_BAL_HIST with BAL_AMT Column



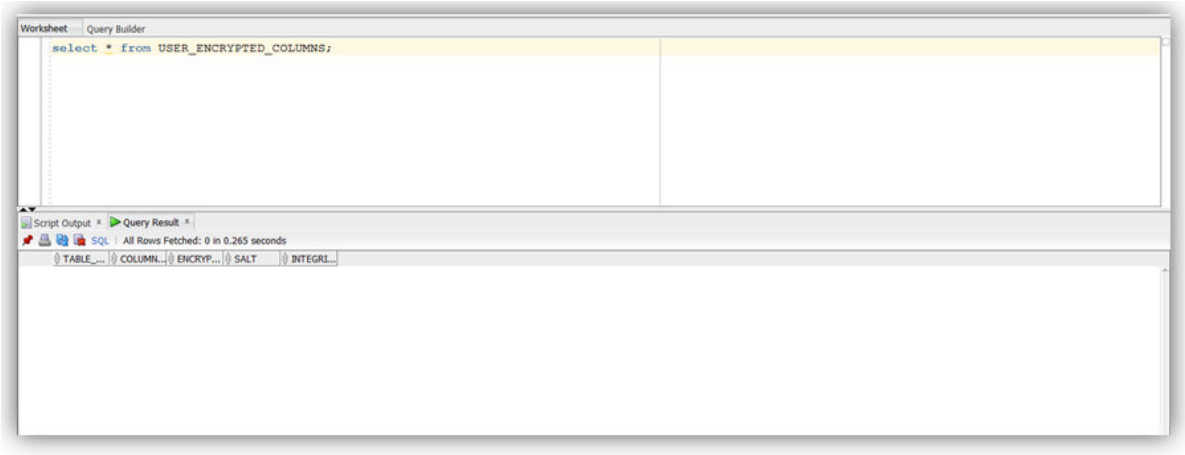
3. Query shows data and current tablespace of table DWR_ACCT, later we will move it to one encrypted tablespace.

Figure 8-9 Data and Table for DWR_ACCT



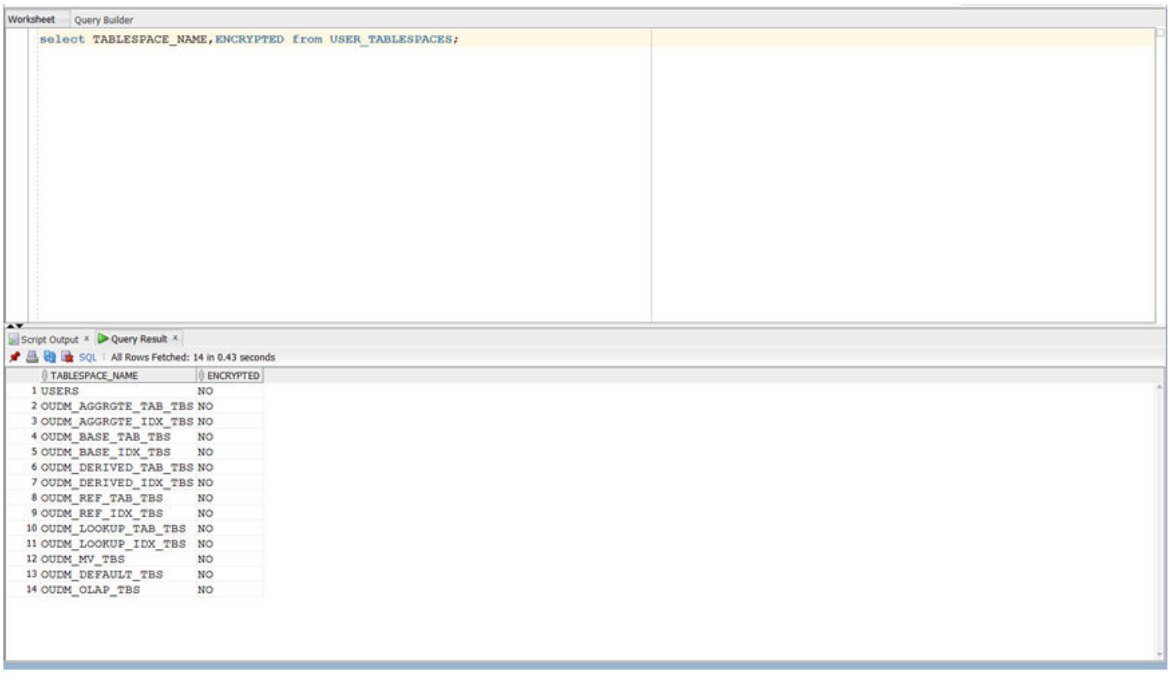
4. Query shows encrypted columns (no columns are encrypted).

Figure 8-10 Query to Show Encrypted Columns



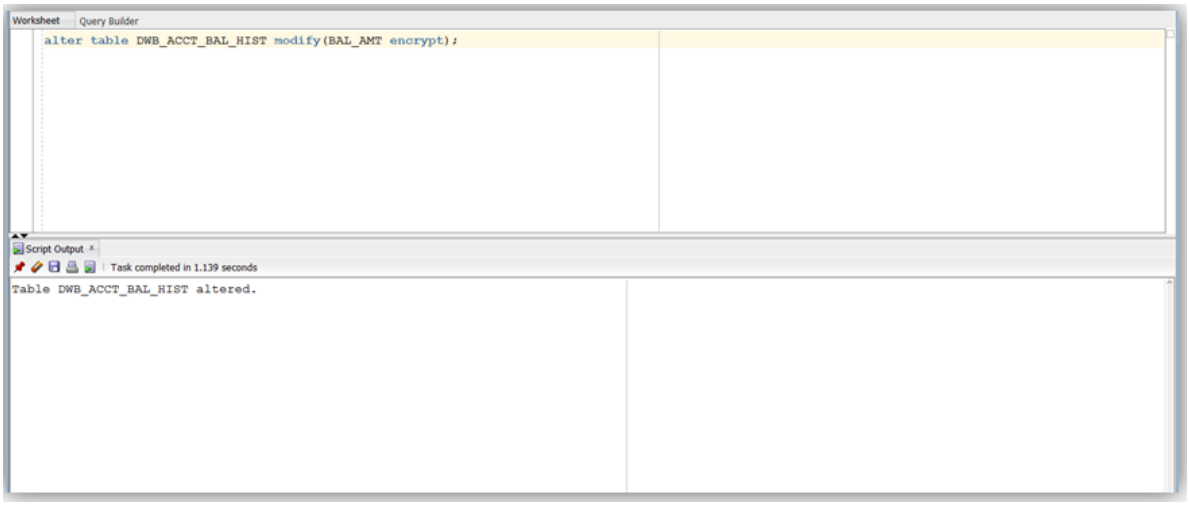
5. Showing current encryption property for each tablespace.

Figure 8-11 Showing Encryption Property for Tablespaces



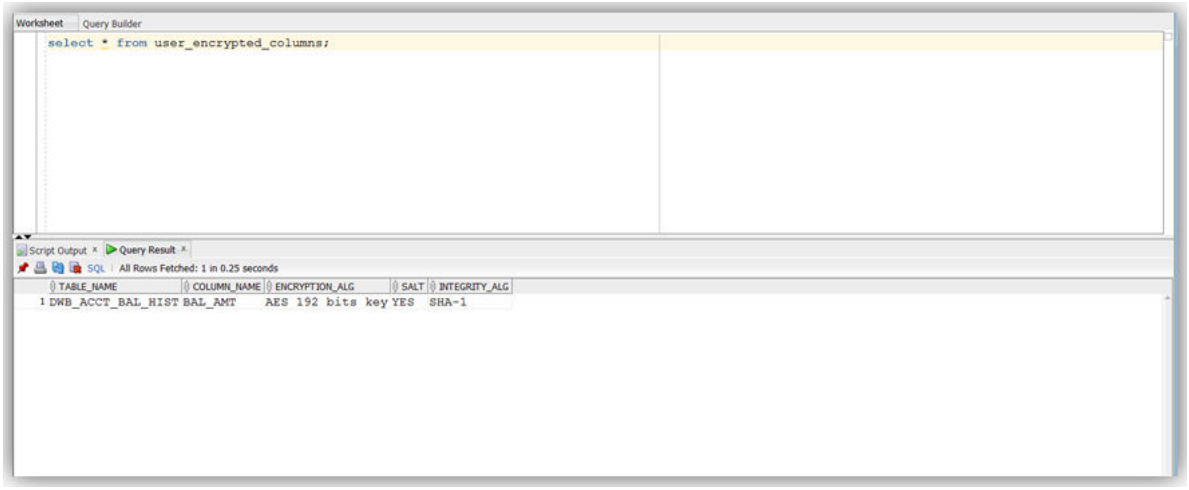
6. Encrypt column BAL_AMT.

Figure 8-12 Alter DWB_ACCT_BAL_HIST and Encrypt Column BAL_AMT



7. Now, column BAL_AMT is encrypted.

Figure 8-13 List Encrypted Columns



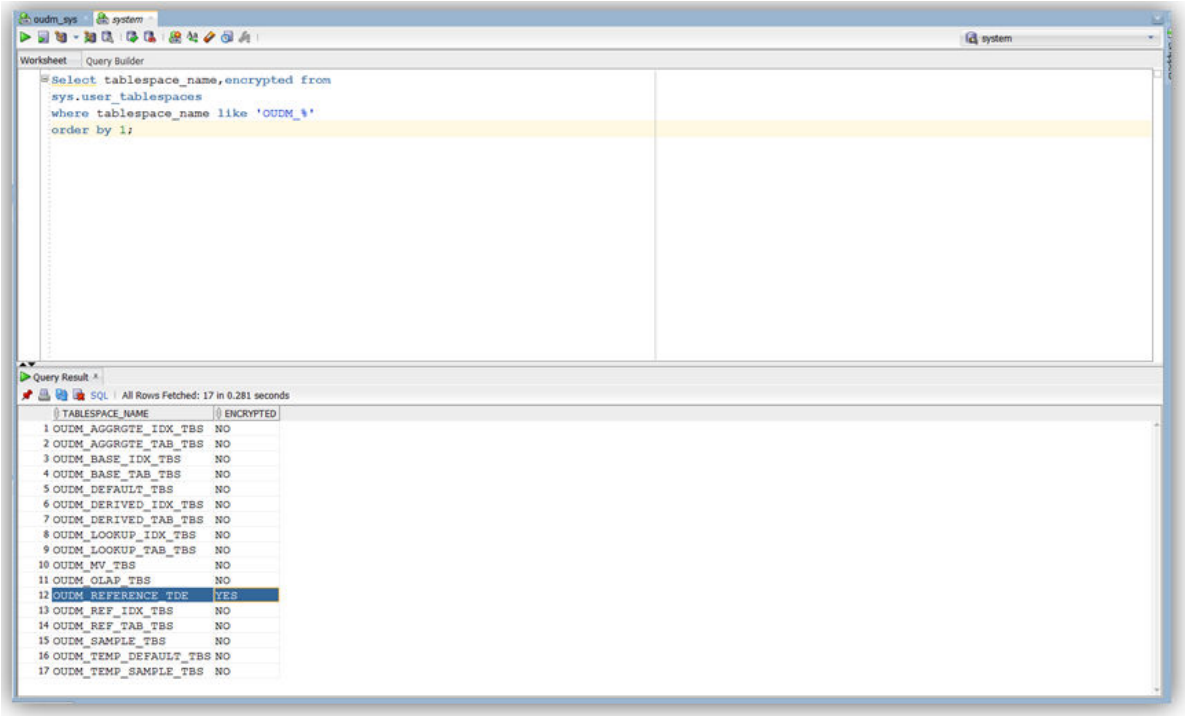
8. Shows the command to create the encrypted tablespace:

```
SQL> CREATE SMALLFILE TABLESPACE "OUDM_REFERENCE_TDE" DATAFILE '/scratch/
biaora/app/122/oradata/orcl122/orcl122pdb/OUDM_REFERENCE_TDE' SIZE 64M LOGGING
EXTENT MANAGEMENT LOCAL SEGMENT SPACE MANAGEMENT AUTO DEFAULT STORAGE(ENCRYPT)
ENCRYPTION USING 'AES256';
```

9. List the encrypted value for tablespaces, including: OUDM_REFERENCE_TDE:

```
Select tablespace_name,encrypted from sys.user_tablespaces where tablespace_name
like ‘OUDM_%’;
```

Figure 8-14 Show the OUDM_REFERENCE_TDE Tablespace



10. Shows the commands to grant privileges on the new tablespace:

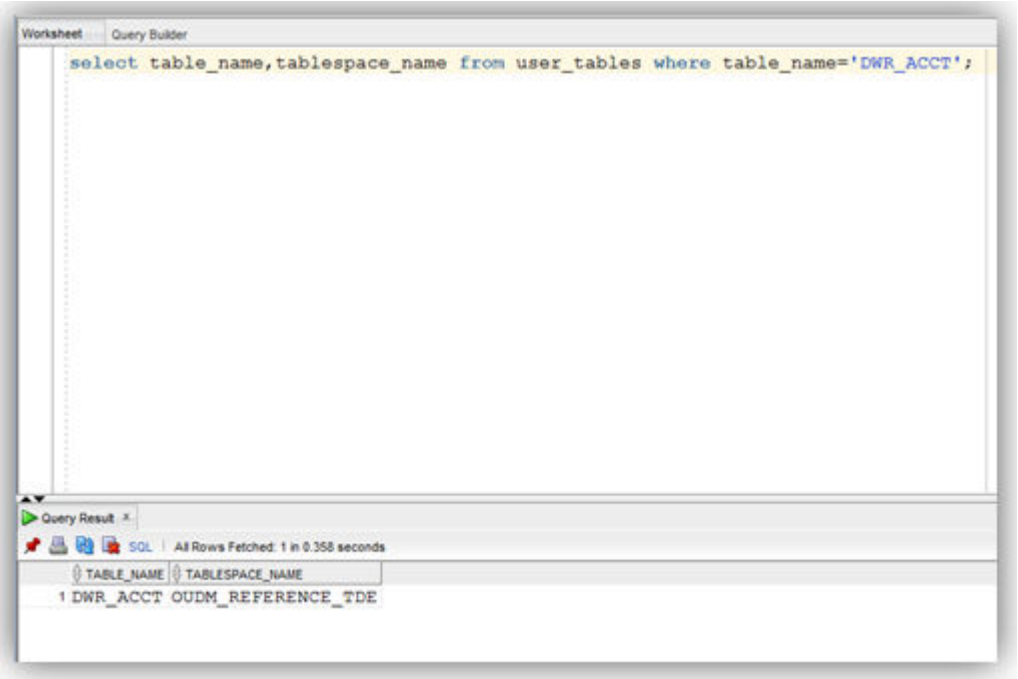
```
SQL> ALTER USER OUDM_SYS  QUOTA UNLIMITED ON OUDM_REFERENCE_TDE;

user altered.
```

11. Shows the commands to move the table DWR_ACCT to the encrypted tablespace:

```
SQL>ALTER TABLE OUDM_SYS.DWR_ACCT MOVE TABLESPACE "OUDM_REFERENCE_TDE";
```

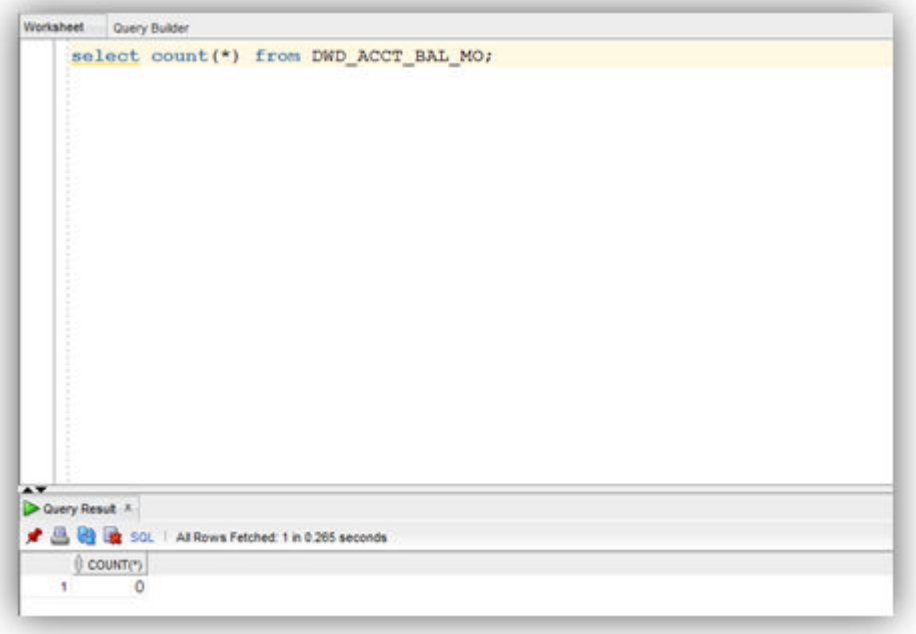
12. Shows the query that lists the table DWR_ACCT in the encrypted OUDM_REFERENCE_TDE tablespace.

Figure 8-15 Shows the Table Name and Tablespace Name for DWR_ACCT



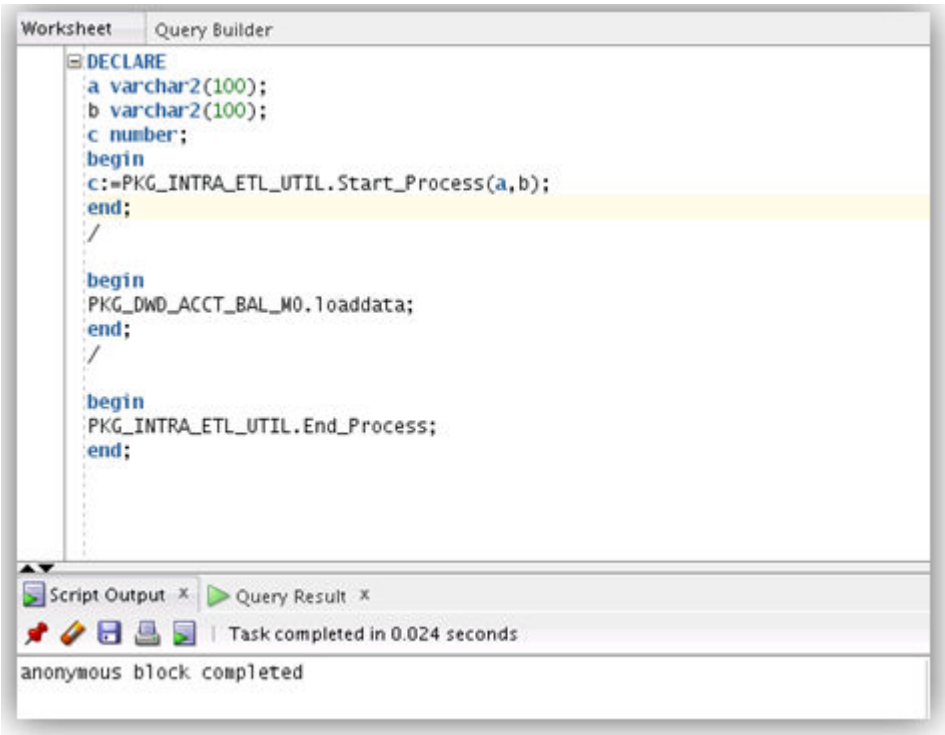
13. Shows the content of the table DWD_ACCT_BAL_MO before running the Intra-ETL package.

Figure 8-16 Shows the Empty DWD_ACCT_BAL_MO Count



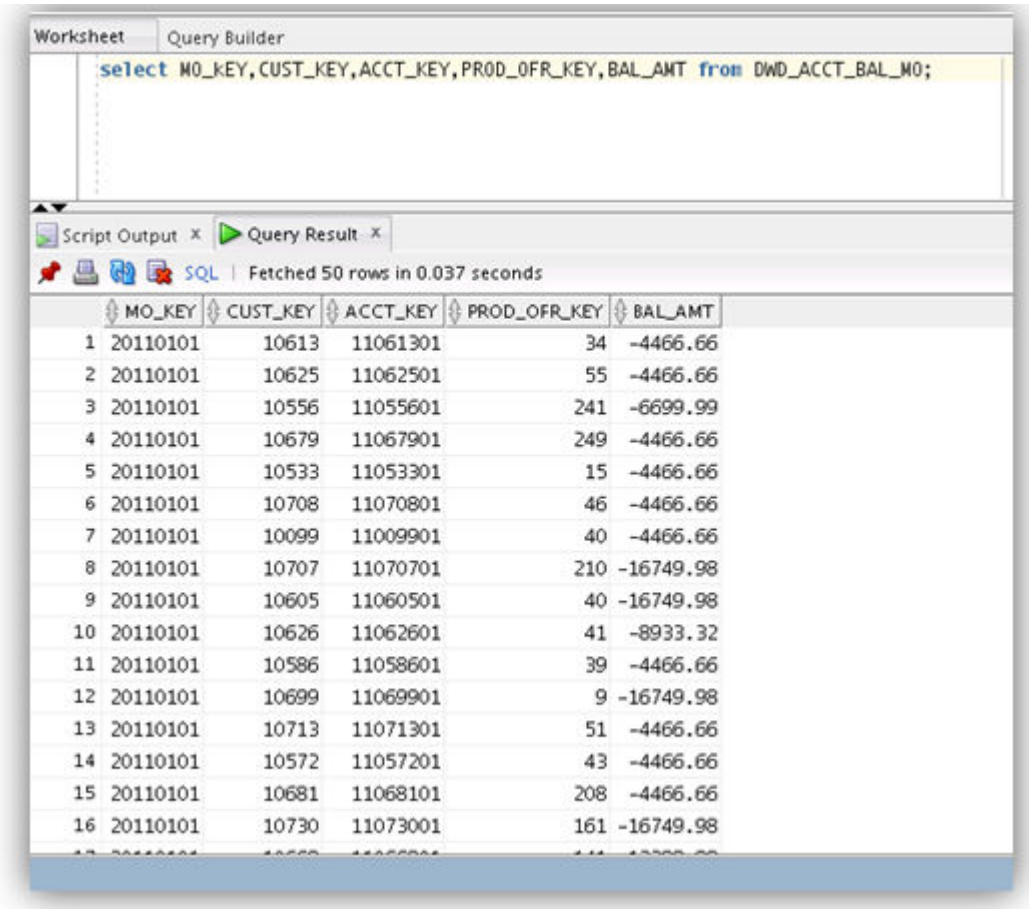
14. Run the IETL process to populate the DWD_ACCT_BAL_MO table:

Figure 8-17 Shows a Run of the IETL Process



15. Now the target table has data.

Figure 8-18 Shows a Nonzero Count for DWD_ACCT_BAL_MO



	MO_KEY	CUST_KEY	ACCT_KEY	PROD_OFR_KEY	BAL_AMT
1	20110101	10613	11061301	34	-4466.66
2	20110101	10625	11062501	55	-4466.66
3	20110101	10556	11055601	241	-6699.99
4	20110101	10679	11067901	249	-4466.66
5	20110101	10533	11053301	15	-4466.66
6	20110101	10708	11070801	46	-4466.66
7	20110101	10099	11009901	40	-4466.66
8	20110101	10707	11070701	210	-16749.98
9	20110101	10605	11060501	40	-16749.98
10	20110101	10626	11062601	41	-8933.32
11	20110101	10586	11058601	39	-4466.66
12	20110101	10699	11069901	9	-16749.98
13	20110101	10713	11071301	51	-4466.66
14	20110101	10572	11057201	43	-4466.66
15	20110101	10681	11068101	208	-4466.66
16	20110101	10730	11073001	161	-16749.98

Although the column DWB_ACCT_BAL_HIST.BAL_AMT is encrypted and the table DWR_ACCT is moved to the encrypted tablespace, this is transparent to Oracle Utilities Data Model.

A

Sizing and Configuring an Oracle Utilities Data Model Warehouse

This appendix provides information about sizing and configuring an Oracle Utilities Data Model warehouse. It contains the following topics:

- Sizing an Oracle Utilities Data Model Warehouse (page A-1)
  Businesses now demand more information sooner and are delivering analytics from their Enterprise Data Warehouse (EDW) to an ever-widening set of users and applications. In order to keep up with this increase in demand the EDW must now be near real-time and be highly available.
- Configuring a Balanced System for Oracle Utilities Data Model (page A-3)
  Many data warehouse operations are based upon large table scans and other I/O-intensive operations, which perform vast quantities of random I/Os. In order to achieve optimal performance the hardware configuration must be sized end to end to sustain this level of throughput.

A.1 Sizing an Oracle Utilities Data Model Warehouse

Businesses now demand more information sooner and are delivering analytics from their Enterprise Data Warehouse (EDW) to an ever-widening set of users and applications. In order to keep up with this increase in demand the EDW must now be near real-time and be highly available.

But how do you go about sizing your system? You must first understand how much throughput capacity is required for your system and how much throughput each individual CPU or core in your configuration can drive, thus the number one task is to calculate the database space requirement in your data warehouse.

There are two data volume estimate resources in a data warehouse environment:

- The estimated raw data extract from source systems. This estimate affects the ETL system configuration and the stage layer database space in data warehouse system. Because this value is determined by your unique OLTP system, you must calculate this information yourself.
- The space needed for data stored to support the objects defined in the default Oracle Utilities Data Model schema. This appendix provides information you can use to make this calculation.

Calculation Factors When Making a Data Volume Calculation for an Oracle Utilities Data Model Warehouse

Consider the following calculation factors when making a data volume calculation:

- Calculates data unit volume within different type:
- Reference and lookup tables data. Assume this data is permanently stored.
- Base tables data (transaction data). Assume that this data is stored within its life cycle.

- Star schema (derived and summary). Assume that this data is stored within its life cycle.
- Calculate each type of data retention.
- Define how many months or years of each type of tables to retain.
- Calculate data growth.
- Assume that annual growth rate: applies to both transaction and reference data and data in the star schema.
- Assume that annual change rate applies only to reference data.
- Calculate Staging Area data requirements, if proposed.

> **Tip:**
>
> Multiply ETL volume by day by number of days held for problem resolution and re-run of transform with new extract from source systems.

- Calculate data volume for indexes, temporary tables, and transaction logs.
- Calculate the space requirement for business intelligence tools, such as cubes, and data mining.
- Consider the redo log and Oracle ASM space requirement.
- Consider the RAID architecture [RAID 1, 0+1, 5]
- Consider the backup strategy.
- Consider the compress factor if applied.
- Consider the OS and file system disk space requirements.

Formula to Determine Minimum Disk Space Requirements for an Oracle Utilities Data Model Warehouse

Use the following formula, based on the factors outlined in "Calculation Factors When Making a Data Volume Calculation for an Oracle Utilities Data Model Warehouse (page A-1)", to determine the minimum disk space requirements for an Oracle Utilities Data Model warehouse.[1]

```
Disk Space Minimum Requirements = Raw data size * Database space factor *
(1+GrthperY)nY*OS and File system factor * Compress Factor * Storage Redundant
factor
```

where:

- `Raw data size = (reference and lookup data per year + base/transaction data per year + derived and summary data per year +staging data +other data(OLAP/ Data Mining))`
- `Database space factor = Indexes + Temporary Tables + Logs]`
- `GrthperY` = growth rate per year

[1] Carefully review whether these factors apply in your environment. These factors may not apply or may change in your environment, especially when using pretuned Exadata hardware.)

- `OS and File system factor` is the install and configuration and maintain space for OS and DB
- `Redundant factor= ASM disk space and RAID factor. [ RAID 1=2, RAID 5=1.25 or 1.33 ]`
- `Compress factor` depends how you apply the compress function. If you are executing on an Exadata Database machine, it has a huge savings in disk space by using compression.

A.2 Configuring a Balanced System for Oracle Utilities Data Model

Many data warehouse operations are based upon large table scans and other I/O-intensive operations, which perform vast quantities of random I/Os. In order to achieve optimal performance the hardware configuration must be sized end to end to sustain this level of throughput.

This type of hardware configuration is called a balanced system. In a balanced system all components - from the CPU to the disks - are orchestrated to work together to guarantee the maximum possible I/O throughput. I/O performance is always a key consideration for data warehouse designers and administrators. The typical workload in a data warehouse is especially I/O intensive, with operations such as large data loads and index builds, creation of materialized views, and queries over large volumes of data. Design the underlying I/O system for a data warehouse to meet these heavy requirements.

To create a balanced system, answer the following questions:

- How many CPUs are required? What speed is required?
- What amount of memory is required? Data warehouse do not have the same memory requirements as mission-critical OLTP applications?
- How many I/O bandwidth components are required? What is the desired I/O speed?

Each component must be able to provide sufficient I/O bandwidth to ensure a well-balanced I/O system.

The following topics provide more information about configuring a balanced system for Oracle Utilities Data Model:

- Maintaining High Throughput in an Oracle Utilities Data Model Warehouse (page A-4)
- Configuring I/O in an Oracle Utilities Data Model for Bandwidth not Capacity (page A-4)
- Planning for Growth of Your Oracle Utilities Data Model (page A-4)
- Testing the I/O System Before Building the Warehouse (page A-5)
- Balanced Hardware Configuration Guidelines for Oracle Utilities Data Model (page A-5)

A.2.1 Maintaining High Throughput in an Oracle Utilities Data Model Warehouse

The hardware configuration and data throughput requirements for a data warehouse are unique mainly because of the sheer size and volume of data. Before you begin sizing the hardware configuration for your data warehouse, estimate the highest throughput requirement to determine whether current or proposed hardware configuration can deliver the necessary performance. When estimating throughput, use the following criteria:

- The amount of data accessed by queries during peak time, and the acceptable response time
- The amount of data that is loaded within a window of time

A.2.2 Configuring I/O in an Oracle Utilities Data Model for Bandwidth not Capacity

Based on the data volume calculated and the highest throughput requirement, you can estimate the I/O throughput along with back-end ETL process and front end business intelligence applications by time unit. Typically, a value of approximately 200 MB per second I/O throughput per core is a good planning number for designing a balanced system. All subsequent critical components on the I/O path - the Host Bus Adapters, fiber channel connections, the switch, the controller, and the disks - have to be sized appropriately.

When running a data warehouse on an Oracle Real Application Cluster (Oracle RAC) it is just as important to size the cluster interconnect with the same care and caution you would use for the I/O subsystem throughput.

When configuring the storage subsystem for a data warehouse, it should be simple, efficient, highly available and very scalable. An easy way to achieve this is to apply the S.A.M.E. methodology (Stripe and Mirror Everything). S.A.M.E. can be implemented at the hardware level or by using Oracle ASM (Automatic Storage Management) or by using a combination of both. There are many variables in sizing the I/O systems, but one basic rule of thumb is that the data warehouse system has multiple disks for each CPU (at least two disks for each CPU at a bare minimum) to achieve optimal performance.

A.2.3 Planning for Growth of Your Oracle Utilities Data Model

A data warehouse designer plans for future growth of a data warehouse. There are several approaches to handling the growth in a system, and the key consideration is to be able to grow the I/O system without compromising on the I/O bandwidth. You cannot, for example, add four disks to an existing system of 20 disks, and grow the database by adding a new tablespace striped across only the four new disks. A better solution would be to add new tablespaces striped across all 24 disks, and over time also convert the existing tablespaces striped across 20 disks to be striped across all 24 disks.

A.2.4 Testing the I/O System Before Building the Warehouse

When creating a data warehouse on a new system, test the I/O bandwidth before creating all of the database data files to validate that the expected I/O levels are being achieved. On most operating systems, you can perform the test using simple scripts to measure the performance of reading and writing large test files.

A.2.5 Balanced Hardware Configuration Guidelines for Oracle Utilities Data Model

You can reference the follow tips for a balanced hardware configuration:

- Total throughput = `#cores X 100-200MB` (depends on the chip set)
- Total host bus adapter (HBA) throughput = Total core throughput

> **Note:**
>
> If total core throughput is 1.6 GB, you need four 4 Gbit HBAs.

- Use one disk controller per HBA port (throughput capacity must be equal).
- Switches must have the capacity as HBAs and disk controllers.
- Use a maximum of ten physical disk per controller (that is, use smaller drives: 146 or 300 GB).
- Use a minimum of 4 GB of memory per core (8 GB if using compress).
- Interconnect bandwidth equals I/O bandwidth (InfiniBand).

Index

K

M

N

O

P

Q

R

S

T

U