Oracle® Communications Data Model

Operations Guide

11*g* Release 1 (11.2)

E15883-03

February 2011

ORACLE®

Oracle Communications Data Model Operations Guide, 11*g* Release 1 (11.2)

E15883-03

Contents

4 Working with an Oracle Communications Data Model Warehouse

Index

Preface

The *Oracle Communications Data Model Operations Guide* describes the tasks and procedures that must be performed after Oracle Communications Data Model is installed, and periodically afterwards to maintain useful performance. Since the needs of each Oracle Communications Data Model environment are unique, Oracle Communications Data Model is configurable so it can be modified to address each customer's needs.

This preface contains the following topics:

Audience

The audience for the *Oracle Communications Data Model Operations Guide* includes the following:

- IT specialists, who maintain and adjust Oracle Communications Data Model. They are assumed to have a strong foundation in Oracle Database and PL/SQL, Oracle Warehouse Builder, or OWB, which generates the data warehouse, AWM, and BIEE.
- Database administrators, who will administer the data warehouse and the database objects that store the data. They are assumed to understand Intra-ETL, which is used to transfer data from one format to another; Oracle Warehouse Builder, which generates the data warehouse, as well as PL/SQL and the Oracle Database.

This document is also intended for business analysts, data modelers, data warehouse administrators, IT staff, and ETL developers.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/us/corporate/accessibility/index.html`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/support/contact.html or visit http://www.oracle.com/accessibility/support.html if you are hearing impaired.

Related Documents

For more information about Oracle Oracle Communications Data Model, see the following documents in the Oracle Oracle Communications Data Model documentation set:

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

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 Communications Data Model

This chapter introduces the Oracle Communications Data Model, which is a start-up kit for implementing a Oracle Communications Data Model data warehouse solution:

- What is Oracle Communications Data Model?
- Oracle Communications Data Model Logical and Physical Models
- Oracle Products That Make Up Oracle Communications Data Model
- Where Oracle Communications Data Model Fits in a Data Warehousing Project
- Prerequisite Knowledge for Implementors

What is Oracle Communications Data Model?

You use Oracle Communications Data Model to jump-start the design and implementation of a Oracle Communications Data Model data warehouse to quickly achieve a positive ROI for your data warehousing and business intelligence project with a predictable implementation effort.

The Oracle Communications Data Model consists of logical and physical data models, intra-ETL that map data from the Oracle Communications Data Model 3NF level to the aggregate level, sample reports, and the Oracle Interactive Dashboard using features of Oracle Business Intelligence Suite Enterprise Edition.

Oracle Communications Data Model includes the following components:

- Logical model

  The logical model is introduced in "Logical Model" on page 1-2 and described in detail in *Oracle Communication Data Model Reference*.

- Physical model

  The physical model is introduced in "Physical Model" on page 1-2 and described in detail in *Oracle Communication Data Model Reference*.

- Intra-ETL database packages and SQL scripts to extract, transform, and load (ETL) data from one layer of Oracle Communications Data Model to another.

  The intra-ETL packages and SQL scripts are described in detail in *Oracle Communication Data Model Reference*. How to use these packages and scripts to populate a data warehouse based on the Oracle Communications Data Model is discussed in Chapter 3, "Populating an Oracle Communications Data Model Warehouse."

- Pre-defined data mining models.

  These models are described in detail in *Oracle Communication Data Model Reference*. Working with these data mining models is discussed in "Refreshing Data Mining Models" on page 3-10 and "Modifying Data Mining Models" on page 4-5.

- Sample reports and dashboards using OBIEE.

  These reports are described in detail in *Oracle Communication Data Model Reference*. Working with these reports are discussed in "Customizing the Reports Delivered with Oracle Communications Data Model" on page 4-1.

- DDL and installation scripts

Oracle Communications Data Model Logical and Physical Models

The logical and physical models of Oracle Communications Data Model have the following characteristics:

- Communication industry-specific, 3rd Normal Form logical and physical relational models
- Physical data model with:

  entity number:
  1300+ tables and 11,000+ columns
  1380+ industry-specific measures and KPIs
  6 pre-built data mining models
  20 pre-built OLAP cubes

- 12 best-practice sample reports and dashboards

Logical Model

Oracle Communications Data Model provides a predefined logical model. The logical data model defines the business entities and their relationships in order provide a clear understanding of the business and data requirements for the data warehouse.

The logical data model is described in detail in *Oracle Communication Data Model Reference*.

Physical Model

Oracle Communications Data Model provides a predefined physical data model.

The physical data model includes the following types of tables and views:

- Reference tables
- Lookup tables
- Database sequences
- Base tables
- Derived tables
- Aggregate tables
- Cube materialized views and views of these dimensions, hierarchies and measures of these cubes.

The physical data model is described in detail in *Oracle Communication Data Model Reference*.

Notes: When examining the predefined physical model, keep in mind the naming convention using DW (Data Warehouse) prefixes and suffixes to identify the types of tables and views:

DWR_ : Reference data tables

DWL_ : Lookup tables

DWB_ : Base transaction data (3NF) tables

DWD_ : Derived tables (including data mining) tables)

DWA_ : Aggregate tables

DWC_: Control tables

CB$: Cube materialized views

_VIEW : View

Oracle Products That Make Up Oracle Communications Data Model

Several Oracle technologies are involved in building the infrastructure for communication business intelligence:

- Oracle Database with OLAP, Data Mining and Partitioning Option
- Oracle Development Tools
- Oracle BI EE Presentation Tools

Oracle Database with OLAP, Data Mining and Partitioning Option

Oracle Communications Data Model utilizes 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

The following Oracle tools can be used to customize the predefined logical and physical models provided with Oracle Communications Data Model, or to populate the target relational tables and materialized cube views.

Table 1–1 Oracle Development Tools Used with Oracle Communications Data Model

Name	Use
Oracle SQL Data Modeler	To modify, customize, and extend the logical model
SQL Developer or SQL*Plus	To modify, customize, and extend database objects
Oracle Warehouse Builder	For the process control of the intra ETL process
Analytic Workspace Manager	To view, create, develop, and manage OLAP dimensional objects.

Oracle BI EE 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 BI EE Answers and Dashboard presentation tools to customize the

predefined sample dashboard reports that are provided with Oracle Communications Data Model.

See: "Customizing the Reports Delivered with Oracle Communications Data Model" on page 4-1.

Where Oracle Communications Data Model Fits in a Data Warehousing Project

Oracle Communications Data Model provides much of the data modeling work that you must do for a communication business intelligence solution. The Foundation Layer provides a solid basis for a communication data warehouse. The Derived and Aggregate Layers provide the infrastructure for creating business intelligence reports.

Oracle Communications Data Model comes with logical and physical data models that have been designed following best practices for communications service providers. However, each communication service provider is unique and you might find that the schem so that you can apply your specific business rules and policies. Typically, the types of modifications that need to be made include, adding, deleting, modifying, or renaming tables and columns; or altering foreign keys, constraints, or indexes.

After the foundation layer of the data warehouse is populated using ETL that you write, you populate the derived and aggregate layers using ETL provided with Oracle Communications Data Model. The derived and aggregate layers support the reporting requirements. Oracle Communications Data Model includes a solid infrastructure for a range of reports.

Prerequisite Knowledge for Implementors

As discussed in "Oracle Products That Make Up Oracle Communications Data Model" on page 1-3, Oracle Communications Data Model uses much of the Oracle stack. Consequently, to successfully customize Oracle Communications Data Model, you need:

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

2

Customizing the Oracle Communications Data Model

This chapter provides an overview of how you customize Oracle Communications Data Model and provides an example of modifying the logical model. It contains the following topics:

- Overview: Customization Steps
- Performing Fit-Gap Analysis
- Discovering the Oracle Communications Data Model Metadata
- Example: Modifying the Logical Model Based on Fit-Gap Analysis

> **See also:** Business use cases in *Oracle Communication Data Model Reference*.

Overview: Customization Steps

Customizing Oracle Communications Data Model involves the following tasks:

1. Perform fit-gap analysis as described in "Performing Fit-Gap Analysis" on page 2-2.
2. In a development environment, install an Oracle Communications Data Model instance.
3. Working in the copy you created in Step 2 and following the documentation you created when performing your fit-gap analysis, customize Oracle Communications Data Model by making changes to its components. Document all of your changes. Make the changes in the following order:
   a. Logical model
   b. Logical to physical mappings
   c. Physical model. To determine which changes to make and the order in which to make these changes, determine dependences among objects by using the Metadata Browser.
   d. Intra-ETL. Keep in mind the issues discussed in Chapter 3, "Populating an Oracle Communications Data Model Warehouse".
4. In a test environment, make a copy of your customized version of Oracle Communications Data Model.
5. In the test environment, following the documentation you created in Step 3, test the customized version of Oracle Communications Data Model.

6. Roll the final customized version of Oracle Communications Data Model out into production.

Performing Fit-Gap Analysis

Fit-gap analysis is where you compare your information needs and communication business requirements with the structure that is available "out of the box" with Oracle Communications Data Model. You identify any required functionality that is not included in 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 Communications Data Model to fit your communication environment.

Note: Fit-gap analysis is a major undertaking, and normally requires a team performing multiple evaluations.

To perform the actual analysis your evaluation team takes the following steps:

1. If previous evaluations have been performed, review the documentation from the previous phases, and if necessary add team members with the needed business and technical expertise.
2. Meet to review the data and map it into Oracle Communications Data Model's schema. In other words, perform a comprehensive analysis of all of the source data in your OLTP systems and then compare your business with the Oracle Communications Data Model logical model to see how it maps to the Oracle Communications Data Model base and derived layer.
3. Produce a list of what people are going to try to do with the system (examples rather than models), and create use cases for appraising the functionality of Oracle Communications Data Model.

   Procedures are written based on the use cases. Keep in mind that deviations from the procedure can be useful, provided that functionality is not skipped.
4. Check security requirements.
5. Determine the differences between your needs and Oracle Communications Data Model's schema. Answer the following questions:
   - Which differences can you live with, and which must be reconciled?
   - What can you do about the differences you can't live with?
6. Write the customization report, detailing what changes will be required to make Oracle Communications Data Model's schema match your business needs. This includes any interfaces to existing systems, and additions and changes to Oracle Communications Data Model.
7. Based on the customization report, update the Project Plan, and complete a phase section for the Logical Design phase.

Discovering the Oracle Communications Data Model Metadata

To customize the Oracle Communications Data Model model, you need to understand the dependency between each Oracle Communications Data Model component -- especially how the report KPIs are mapped to the physical tables and columns.

Oracle Communications Data Model provided a tool called "Metadata Browser" that helps you discover this. The metadata browser is delivered as one Dashboard in Oracle Business Intelligence Suite Enterprise Edition.

There are four tabs in the Oracle Business Intelligence Suite Enterprise Edition report that is the Oracle Communications Data Model Metadata Browser:

- Measure-Entity tab
- Entity-Measure tab
- Program-Table tab
- Table-Program tab

Measure-Entity tab

As shown below, 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.

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

Entity-Measure tab

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

For example, you take the following steps to learn more about COST ORIGINAL MONTH AGGR:

1. Select the entity COST ORIGINAL MONTH AGGR.
2. Click GO.

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:

1. Select prompt program type (that is, Intra-ETL or report) and program name for sowing particular report or intra_ETL information.
2. Select GO.

Table-Program tab

Table-Program:

By default when you go to the Table-Program tab you see all 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.

For example, to see the reports that use the DWA_CALL_CENTR_CALL_MO table, take the following steps:

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

2. Move it to the Selected list on the left by clicking on <, and click **OK**.
3. Select GO and the reports for DWA_CALL_CENTR_CALLMO table are displayed.

Example: Modifying the Logical Model Based on Fit-Gap Analysis

Application stores are the latest battleground for mobile phone companies seeking to differentiate their products in an increasingly crowded and competitive marketplace. Users will be able to find the applications by searching using keywords or by browsing through categories and top downloads - so they can see what everyone else is downloading - in addition to the front page, which will showcase a selection of featured applications. The 'App World' can be accessed directly through the handset, enabling users to download applications and software on the go. Some applications will be free, others will cost a small fee.

Now, as an example, let's take a look how Oracle Communications Data Model supports the application store business, what a sample customer might discover during fit-gap analysis, and how that customer might extend Oracle Communications Data Model to fit the discovered gaps.

> **See also:** Business use cases in *Oracle Communication Data Model Reference* for other examples.

The supplied entities that support application stores

The entities provided with the logical model of Oracle Communications Data Model that support the Application stores are:

- CONTENT PROVIDER: This entity provides the actual content that will ultimately be consumed by the end user. The contents could be video or audio clips or text content. In the case of Quick Time 7, the content provider is Apple.
- CONTENT TYPE: This entity will store the type of content. For example, for Quick Time 7, Content Type is an application.
- CONTENT: This entity keeps all contents provided by the content provider. For example, Quick Time 7 is Content. In this entity we can store the information about the content; for example, whether the content is still active or not, popularity of the content which will be calculated by the number of downloads, and so on.
- CONTENT PRICE: This entity contains the price for downloading/ordering the content. This price is for individual content clip. There might be other contents priced as a flat rate rather than different price for each content. In this case, the pricing information should be in product rating plan. For example, Quick Time 7 Pro is $30
- CONTENT PRICING TYPE: This entity is type of content pricing. For example, The charge per downloading, monthly fixed rate, and so on.
- CONTENT DELIVERY EVENT: This entity is an event to track downloads of contents. This entity captures the details of the download like the size of the download, duration, date and so on.

The differences discovered during fit-gap analysis

Assume that during the fit-gap analysis, you discover the following needs that are not supported by the logical model delivered with Oracle Communications Data Model:

- Your company saves more than two levels of data for content. In other words, you discover that you track not only Content and Content Type as represented in the standard logical model of Oracle Communications Data Model, but also a level

under Content Type (that is, a content subtype). For example for the value of Games in Content, you have Content Types of Fun Games, Action, and Racing. For the value of Fun Games in Content Types, you have subtype values of Packman, Mummy, and NFS.

- In the Application Store, the operator must maintain a history of Customers who purchased which application so that when customer tries to reinstall, they are not charged again.

Extending the logical model to support the differences

To support the differences, you need to extend the logical model in the following ways:

- To store details about type of the game you will need to modify the logical model. The classic way to do this is to add another entity to hold another level of information. For example, you can add an entity named CONTENT SUBTYPE that has a 1:M relationship with the CONTENT entity. The specific steps to perform these actions in Oracle SQL Developer Data Modeler are given in "Steps: Extending the Logical Model to Support Multiple Levels for CONTENT" on page 2-5.
- To maintain the history of Customers who have purchased an application, you can change the logical model to support this need by creating an entity named APPLICATION SUBSCRIPTION that has a 1:M relationship with `CONTENT`. The specific steps to perform these actions in Oracle SQL Developer Data Modeler are given in "Steps: Modifying the Logical Model" on page 2-6.

Steps: Extending the Logical Model to Support Multiple Levels for CONTENT

Take the following steps in Oracle SQL Developer Data Modeler to extend the logical model to support multiple levels for `CONTENT`:

1. Open the Oracle Communications Data Model in SQL Developer Data Modeler.
2. Create the `CONTENT SUBTYPE` entity by taking the following steps:
   a. In the right-hand pane (the main area) of the SQL Developer Data Modeler window, select the Logical tab.
   b. Click the New Entity icon.
   c. Click in the logical model pane and draw an entity box.

      The **Entity Properties** window is displayed.
   d. In the left-hand pane of Entity Properties, select **General**. For **Name**, specify, `CONTENT SUBTYPE`. (Note the value of `Entity_`*xxx* where *xxx* is the number for **Long Name** and `Single Table` for **FWD Engineer Strategy**.)
   e. In the left-hand pane, select **Attributes**. Using the Add (+) icon, add the following attributes, one at a time.:

      `CONTENT SUBTYPE CODE` with a data type of `CODE`.

      `CONTENT SUBTYPE NAME` with a data type of `NAME`.

      `CONTENT SUBTYPE DESC` with a data type of `DESCRIPTION`.
   f. In the left-hand pane, select **Unique Identifiers**. Using the Add (+) icon, add a unique, primary key to CONTENT SUBTYPE.
   g. Select `key_1` (which represents the unique, primary key), and use Properties icon to add the properties to the key.

h. In the left-hand pane of **Key Properties**, select **General**. For key properties, specify `CST_PK` for **Name** and `Primary Key` for **State**. (Note that the **Long Name** is `CONTENT SUBTYPE.CST_PK`.

i. Select **Attributes and Relations**, and then select the column for the primary key (that is, CONTENT SUBTYPE CODE).

j. Click **Apply**. The entity CONTENT SUBTYPE is created.

3. Add a CONTENT SUBTYPE CODE column to the `CONTENT` entity by taking the following steps:

   a. Select the `CONTENT` entity.

   b. Right click and select **Properties**.

   c. In the left-hand pane, click **Attributes**.

   d. Use the Add (+) icon to add an attribute with the Name of `CONTENT SUBTYPE CODE` and a Data type of `CODE`.

4. Add a relationship between `CONTENT` and CONTENT SUBTYPE by taking the following steps:

   a. Select the **New 1: N Relation** icon.

   b. Select the CONTENT SUBTYPE entity.

   c. Drag the CONTENT SUBTYPE entity over to the `CONTENT` entity and drop the cursor.

Steps: Modifying the Logical Model

Take the following steps in Oracle SQL Developer Data Modeler to extend the logical model to support maintaining a history of Customers who purchased which application:

1. Open Oracle Communications Data Model in SQL Developer Data Modeler.

2. Create an `APPLICATION SUBSCRIPTION` entity by taking the following steps:

   a. In the right-hand pane (main area) of the SQL Developer Data Modeler window, select the Logical tab.

   b. Click the New Entity icon.

   c. Click in the logical model pane in the main area; and in the Logical pane press, diagonally drag, and release the mouse button to draw an entity box.

      The **Entity Properties** window is displayed.

   d. In the left-hand pane of **Entity Properties**, select **General**. Specify `APPLICATION SUBSCRIPTION` for **Name**, and `SUBSCRIPTION` for **Super Type**. (Note that the value of **FWD Engineer Strategy** is Single Table.)

   e. In the left-hand pane, select **Attributes**. Using the Add (+) icon, add the following

      Attribute APPLICATION SUBSCRIPTION CODE and datatype is CODE. As `APPLICATION SUBSCRIPTION` entity is of SUBSCRIPTION, attributes are inherited from SUBSCRIPTION entity.

   f. In the left-hand pane, select **Unique Identifiers**. Use the Add (+) icon to add the Primary Key

g. Select **Attributes and Relations**, and then select the column for the unique, primary key.

h. Click **Apply**. The `APPLICATION SUBSCRIPTION` entity will be created.

3. Add APPLICATION SUBSCRIPTION CODE to CONTENT by taking the following steps:

   a. Select the CONTENT entity. Right click and select Properties.

   b. In the left-hand pane, select **Attributes**. Use the Add (+) icon to add attributes, one at a time.

4. Add a relationship to CONTENT with APPLICATION SUBSCRIPTION by taking the following steps:

   a. Select the New 1: N Relation icon.

   b. Select the APPLICATION SUBSCRIPTION entity.

   c. Drag the APPLICATION SUBSCRIPTION entity over to the CONTENT entity and drop the cursor.

3

Populating an Oracle Communications Data Model Warehouse

This chapter describes how to populate an Oracle Communications Data Model warehouse.

This chapter discusses the following topics:

- Pre-population Tasks
- Overview: The ETL for an Oracle Communications Data Model Warehouse
- Estimating Space for Oracle Communications Data Model
- Performing an Initial Load of the Warehouse
- Performing Incremental Data Loading of the Warehouse
- Managing Errors During Intra-ETL Execution

> **Note:** The instructions in this chapter assume that after doing the fit-gap analysis described in Performing Fit-Gap Analysis, you have not identified or made any changes to the Oracle Communications Data Model logical or physical model. If you have made changes, you need to modify the ETL accordingly.

Pre-population Tasks

Before you populate your Oracle Communications Data Model warehouse for the first time, take the following steps:

1. Perform all of the post-installation tasks described in *Oracle Communication Data Model Installation Guide*.
2. Familiarize yourself with the ETL you use to populate an Oracle Communications Data Model warehouse as described in "Overview: The ETL for an Oracle Communications Data Model Warehouse" on page 3-2.
3. Determine if you will be doing an initial load of your warehouse as described in "Performing an Initial Load of the Warehouse" on page 3-4 or an incremental load as described in "Performing Incremental Data Loading of the Warehouse" on page 3-7. Familiarize yourself with the process of performing which ever type of load you will be performing.
4. Familiarize yourself with how you can manage any errors that might occur during that initial load as described in "Managing Errors During Intra-ETL Execution" on page 3-11.

5. Estimate the size of your Oracle Communications Data Model warehouse as described in "Estimating Space for Oracle Communications Data Model" on page 3-3.

If you have already performed an initial load of your Oracle Communications Data Model warehouse and are now performing an incremental load of the warehouse, familiarize yourself with the incremental load issues discussed in "Performing Incremental Data Loading of the Warehouse" on page 3-7.

Overview: The ETL for an Oracle Communications Data Model Warehouse

In the Oracle Communications Data Model relational model, reference and lookup tables store master, reference, and dimensional data; while base, derived, and aggregate tables store transaction and fact data at different granularities. Base tables store the transaction data at the lowest level of granularity, while derived and aggregate tables store consolidated and summary transaction data.

As with any data warehouse, you use Extract, Transform, and Load (ETL) operations to populate an Oracle Communications Data Model data warehouse. You perform ETL operations using different types of ETL in the following order:

1. Source-ETL processes that extract data from the source On-Line Transaction Processing (OTLP) system, transform that data, and loads the reference, lookup, and base tables Oracle Communications Data Model warehouse. Source-ETL is *not* provided with Oracle Communications Data Model. For information about creating source-ETL, see "Source-ETL in Oracle Communications Data Model" on page 3-2.
2. Intra-ETL processes that populate the remaining structures in an Oracle Communications Data Model warehouse. Intra-ETL does not access the OLTP data at all. All of the data that intra-ETL extracts and transforms is located within the Oracle Communications Data Model warehouse. Intra-ETL is provided with the Oracle Communications Data Model. For information about the intra-ETL, see "Intra-ETL in Oracle Communications Data Model" on page 3-3.

Source-ETL in Oracle Communications Data Model

Source-ETL processes extract data from the source On-Line Transaction Processing (OTLP) system, transform that data, and load the transformed data into the lookup, reference, and base tables of an Oracle Communications Data Model warehouse.

Note: *You must design and write source-ETL processes yourself.* Source-ETL is *not* provided with Oracle Communications Data Model.

Keep the following points in mind when designing and writing source-ETL:

- You can populate the calendar data by using the calendar population scripts provided with Oracle Communications Data Model and described in *Oracle Communication Data Model Reference*.
- 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 need to analyze all of the tables loaded by the source-ETL process before executing the intra-ETL processes).

Intra-ETL in Oracle Communications Data Model

One component delivered with the Oracle Communications Data Model is the `INTRA_ETL_FLW` process flow which is designed using the Oracle Warehouse Builder Workflow component.

The `INTRA_ETL_FLW` process flow uses the data in the Oracle Communications Data Model base, reference, and lookup tables to populate all of the other structures of the Oracle Communications Data Model.

The `INTRA_ETL_FLW` process flow consists of the following subprocesses and includes the dependency of individual sub-process flows and executes them in the proper order:

1. **DRVD_FLW**

   This sub-process flow contains all the Oracle Warehouse Builder mappings for populating derived tables based on the content of the base, reference, and lookup tables. This process flow allows mappings to be executed concurrently.

2. **AGGR_FLW**

   This sub-process flow contains PL/SQL code that uses the partitions change tracking strategy to refresh all the aggregate tables which are materialized views in the Oracle Communications Data Model.

3. **MINING_FLW**

   This sub-process flow triggers the data mining model refreshment.

The `INTRA_ETL_FLW` process flow also includes the OLAP_MAP mapping. OLAP_MAP invokes the analytic workspace build function of the `PKG_OCDM_OLAP_ETL_AW_LOAD` package to load data from Oracle Communications Data Model aggregate materialized views into the Oracle Communications Data Model analytical workspace and calculates the forecast data. The `PKG_OCDM_OLAP_ETL_AW_LOAD` package reads OLAP ETL parameters from the `DWC_OLAP_ETL_PARAMETER` control table in the `ocdm_sys` schema to determine the specifics of how to load the data and perform the forecasts.

Note: The shell script `ocdm_execute_wf.sh` delivered with Oracle Communications Data Model performs the same function as Oracle Warehouse Builder Workflow `INTRA_ETL_FLW`, and it does not require Oracle Warehouse Builder workflow. For more information about working with this script, see "Manually Executing the Intra-ETL" on page 3-6.

See: *Oracle Communication Data Model Reference* for detailed information about the Oracle Communications Data Model Intra-ETL.

Estimating Space for Oracle Communications Data Model

The size of Oracle Communications Data Model warehouse largely depends on size of network event (or CDRs). You can calculate the approximate overall size of your Oracle Communications Data Model warehouse by using the following formula.

OCDM_Size = 150 * *NAcct* * *NCall* * 365 * *NYear* * 4

where:

- *NAcct* is the number of accounts or subscriptions
- *NCall* is the average number of call records per customer every day, including all relevant usages operator that you want to save, (for example, voice call, IP usage, and so on).
- *NYear* is how many years data you want the Oracle Communications Data Model warehouse to hold

For example, assume for an operator with only the wireless business has 350,000 accounts, every customer make 2 calls each year, the operator wants to save data for 5 years, then the size can be estimated as 766GB.

Note: Note: This is a very preliminary estimation and to plan your data warehouse more accurately, you should do some experiments by loading some sample data into Oracle Communications Data Model schema and measure the actual storage size.

Performing an Initial Load of the Warehouse

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

1. Load the reference, lookup, and base tables Oracle Communications Data Model warehouse by executing the source-ETL that you have written using the guidelines given in "Source-ETL in Oracle Communications Data Model" on page 3-2.
2. Load the remaining structures in the Oracle Communications Data Model, by taking the following steps:
   a. Update the parameters in `DWC_ETL_PARAMETER` control table in the `ocdm_sys` schema so that this information (that is, the beginning and end date of the ETL period) can be used when loading the derived and aggregate tables and views. See "Updating the Parameters of the DWC_ETL_PARAMETER Table for an Initial Load" on page 3-4 for more information.
   b. Update the Oracle Communications Data Model OLAP ETL parameters in `DWC_OLAP_ETL_PARAMETER` control table in the `ocdm_sys` schema to specify the build method and other build characteristics so that this information can be used when loading the OLAP cube data. See "Updating DWC_OLAP_ETL_PARAMETER Table for an Initial Load" on page 3-5 for more information.
   c. Execute the Intra-ETL in one of the ways described in "Executing the Intra-ETL for Oracle Communications Data Model" on page 3-6.

Updating the Parameters of the DWC_ETL_PARAMETER Table for an Initial Load

Before you execute the `INTRA_ETL_FLW` process to load the derived and aggregate relational tables and views, you must update the parameters of the `DWC_ETL_PARAMETER` control table in the `ocdm_sys` schema.

For an initial load of an Oracle Communications Data Model warehouse, specify the values shown in the following table.

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

See: *Oracle Communication Data Model Reference* for more information on the DWC_ETL_PARAMETER control table.

Updating DWC_OLAP_ETL_PARAMETER Table for an Initial Load

Before you can load OLAP cube data, you need to update the Oracle Communications Data Model OLAP ETL parameters in DWC_OLAP_ETL_PARAMETER control table in the ocdm_sys schema.

For an initial load of the analytic workspace, specify values following the guidelines in Table 3–1.

Table 3–1 Values of Oracle Communications Data Model OLAP ETL Parameters in the DWC_OLAP_ETL_PARAMETER Table for Initial Load

Column Name	Value
PROCESS_NAME	'OCDM-INTRA-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 that you want to build all of the cubes in the Oracle Communications Data Model analytic workspace. ■ *cubename*[[|*cubename*]...] specifies one or more cubes that you want 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 or not to calculate forecast cubes: ■ Y specifies calculate forecast cubes. ■ N specifies do not calculate forecast cubes.
NO_FCST_YRS	If the value of CALC_FCST 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 of CALC_FCST is Y, then specify AUTO; otherwise, specify NULL.
FCST_ST_YR	If the value of CALC_FCST is Y, then specify value specified as *yyyy* which is the "start business year" of a historical period; otherwise, specify NULL.
FCST_END_YR	If the value of CALC_FCST is Y, then specify value specified as *yyyy* which is the "end business year" of a historical period; otherwise, specify NULL.
OTHER1	Specify NULL.
OTHER2	Specify NULL.

> **See:** "Updating DWC_OLAP_ETL_PARAMETER Table for an Incremental Load" on page 3-9 for information on the values to specify for an incremental load of OLAP cube data, and *Oracle Communication Data Model Reference*. for more detailed information on all possible values for the `DWC_OLAP_ETL_PARAMETER` control table in the `ocdm_sys` schema.

Executing the Intra-ETL for Oracle Communications Data Model

You can execute the Intra-ETL packages provided with Oracle Communications Data Model in the following ways:

- As a Workflow within Oracle Warehouse Builder as described in "Executing the INTRA_ETL_FLW Workflow Within Oracle Warehouse Builder" on page 3-6.
- Without using Oracle Warehouse Builder Workflow as described in "Manually Executing the Intra-ETL" on page 3-6.

In either case, you can monitor the execution of the Intra-ETL, recover and troubleshoot errors as described in "Monitoring the Execution of the Intra-ETL Process" on page 3-11, "Recovering an Intra ETL Process" on page 3-11, and "Troubleshooting Intra-ETL Performance" on page 3-12.

Executing the INTRA_ETL_FLW Workflow Within Oracle Warehouse Builder

To execute the INTRA_ETL_FLW process flow from Oracle Warehouse Builder, take the following steps:

> **See:** *Oracle Warehouse Builder User's Guide* for information about Oracle Warehouse Builder.

To use Oracle Communications Data Model Intra-ETL as a Oracle Warehouse Builder Workflow, follow these steps:

1. Confirm that Oracle Warehouse Builder Workflow has been installed as described in *Oracle Communication Data Model Installation Guide*.
2. Within Oracle Warehouse Builder, go to the **Control Center Manager**.
3. Select OLAP_PFLW, then select AGR_PFLW, then select the main process flow INTRA_ETL_FLW.
4. Right-click INTRA_ETL_FLW and select **set action**. If this is the first deployment, set action to **Create**; for deployment after the first, set action to **Replace**. Deploy the process flow.

   After the deployment finishes successfully, INTRA_ETL_FLW is ready to execute.

Manually Executing the Intra-ETL

Oracle Communications Data Model provides you with a script that you can use to populate the intra-ETL without using Oracle Warehouse Builder Workflow. This shell script is named `ocdm_execute_wf.sh` . It performs the same function as Oracle Warehouse Builder Workflow `INTRA_ETL_FLW`. It can be invoked by another process such as Source-ETL, or according to a predefined schedule such as Oracle Job Scheduling.

1. The `ocdm_execute_wf.sh` program prompts you to enter the environmental variables described in the following table.

Variable	Description
TSNAME	The tnsname of target database.
SYSTEM password	SYSTEM account password of target database.
ocdm_sys password	The password of the ocdm_sys account.
ORACLE HOME	Oracle Database home(that is, the full path without ending "/").

2. Reads the values from the DWC_ETL_PARAMETER and DWC_OLAP_ETL_PARAMETER control tables in the ocdm_sys schema before executing the mappings in the correct order
3. The result of each table loading are tracked in the DWC_INTRA_ETL_PROCESS and DWC_INTRA_ETL_ACTIVITY control tables as described in "Monitoring the Execution of the Intra-ETL Process" on page 3-11.

> **See:** "Updating the Parameters of the DWC_ETL_PARAMETER Table for an Initial Load" on page 3-4 and "Updating DWC_OLAP_ETL_PARAMETER Table for an Initial Load" on page 3-5 and "Incremental Loading of Relational Tables and Views" on page 3-8 and "Updating DWC_OLAP_ETL_PARAMETER Table for an Incremental Load" on page 3-9 for values to specify for the DWC_ETL_PARAMETER and DWC_OLAP_ETL_PARAMETER control table in the ocdm_sys schema.

Performing Incremental Data Loading of the Warehouse

"Performing an Initial Load of the Warehouse" on page 3-4 describes how to perform an intial load of an Oracle Communications Data Model data warehouse. After this initial load, you need to load new data into your Oracle Communications Data Model data warehouse regularly so that it can serve its purpose of facilitating business analysis.

To load new data into your Oracle Communications 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 and will be 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 relational tables and views, OLAP cubes, and data mining models all at once, or you can refresh the the data sequentially:

1. Incremental Loading of Relational Tables and Views
2. Incremental Loading of OLAP Cubes
3. Refreshing Data Mining Models

In either case, you can manage errors during the execution of the Intra-ETL as described in "Managing Errors During Intra-ETL Execution" on page 3-11

Incremental Loading of Relational Tables and Views

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

1. Refresh the reference, lookup, and base tables Oracle Communications Data Model warehouse with OLTP data by executing the source-ETL that you have written using the guidelines given in "Source-ETL in Oracle Communications Data Model" on page 3-2 .
2. Update the parameters of the `DWC_ETL_PARAMETER` control table in the `ocdm_sys` schema. For an incremental load of an Oracle Communications Data Model warehouse, specify the values shown in the following table (that is, the beginning and end date of the ETL period).

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

See: *Oracle Communication Data Model Reference* for more information on the `DWC_ETL_PARAMETER` control table.

3. Refresh the derived tables and aggregate tables which are materialized views in Oracle Communications Data Model by executing the DRVD_FLOW and AGGR_FLOW subprocess of the INTRA_ETL_FLW process flow. See "Executing the Intra-ETL for Oracle Communications Data Model" on page 3-6 for more information.

See also: *Oracle Warehouse Builder User's Guide*

Incremental Loading of OLAP Cubes

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

Take these steps to perform an incremental load of the analytic workspace that is part of the Oracle Communications Data Model warehouse:

1. Update the aggregate tables which are materialized views in Oracle Communications Data Model. See "Incremental Loading of Relational Tables and Views" on page 3-8 for more information.
2. Update the OCDM OLAP ETL parameters in `DWC_OLAP_ETL_PARAMETER` control table in the `ocdm_sys` schema so that this information (that is, the build method and other build characteristics) can be used when loading the OLAP cube data. See "Updating DWC_OLAP_ETL_PARAMETER Table for an Incremental Load" on page 3-9 for more information.
3. Execute the Intra-ETL to load the cube data in one of the ways described in "Executing the Intra-ETL for Oracle Communications Data Model" on page 3-6.
4. If necessary, recover from errors that happen during the execution of OLAP_MAP by taking the steps outlined in "Recovering from Errors During an Incremental Load of OLAP Cubes" on page 3-10.

Updating DWC_OLAP_ETL_PARAMETER Table for an Incremental Load

Before you can load load OLAP cube data, you need to specify appropriate Oracle Communications Data Model OLAP ETL parameters values in the DWC_OLAP_ETL_PARAMETER control table in the ocdm_sys schema.

For an incremental load of the analytic workspace, specify values following the guidelines in Table 3–2, " Values for Oracle Communications Data Model OLAP ETL Parameters in the DWC_OLAP_ETL_PARAMETER Table for an Incremental Load". You can either run this mapping in Oracle Warehouse Builder or use OMBPlus (Oracle Warehouse Builder Scripting Language) to execute the $ORACLE_HOME/ocdm/pdm/relational/intra_etl/owb_exec/todcolapetl.tcl in command line.

Table 3–2 Values for Oracle Communications Data Model OLAP ETL Parameters in the DWC_OLAP_ETL_PARAMETER Table for an Incremental Load

Column Name	Value
PROCESS_NAME	'OCDM-INTRA-ETL'
BUILD_METHOD	? which specifies a fast refresh if possible; otherwise a complete refresh. (Default).
CUBENAME	One of the following values that specifies the cubes you want to build: ■ ALL specifies that you want to build all of the cubes in the Oracle Communications Data Model analytic workspace. ■ *cubename*[[|*cubename*]...] specifies one or more cubes that you want 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 or not to calculate forecast cubes: ■ Y specifies calculate forecast cubes. ■ N specifies do not calculate forecast cubes.
NO_FCST_YRS	If the value of CALC_FCST is Y, a decimal value that specifies how many years forecast data you want to calculate; otherwise, specify NULL.
FCST_MTHD	If the value of CALC_FCST is Y, then specify AUTO; otherwise, specify NULL.
FCST_ST_YR	If the value of CALC_FCST is Y, then specify value specified as *yyyy* which is the "start business year" of a historical period; otherwise, specify NULL.
FCST_END_YR	If the value of CALC_FCST is Y, then specify value specified as *yyyy* which is the "end business year" of a historical period; otherwise, specify NULL.
OTHER1	Specify NULL.
OTHER2	Specify NULL.

See: Table 3–1, " Values of Oracle Communications Data Model OLAP ETL Parameters in the DWC_OLAP_ETL_PARAMETER Table for Initial Load" on page 3-5 for information on the values to specify for an inital load of OLAP cube data, and *Oracle Communication Data Model Reference* for more detailed information on all possible values for the `DWC_OLAP_ETL_PARAMETER` control table in the `ocdm_sys` schema.

Executing the Oracle Communications Data Model OLAP ETL Mapping

You can execute the Oracle Communications Data Model ETL to update the OLAP cubes in the following ways

- Refresh *all* of the data in the warehouse by executing the Oracle Warehouse Builder Workflow INTRA_ETL_FLW in one of the ways that are described in "Executing the Intra-ETL for Oracle Communications Data Model" on page 3-6.

  The OLAP Cubes are populated through OLAP_MAP which is a part of Oracle Communications Data Model intra-ETL main workflow INTRA_ETL_FLW.

- Refresh *only* the OLAP cube data by executing the OLAP_MAP Oracle Warehouse Builder mapping in the Oracle Warehouse Builder control center.

Note: You must refresh the corresponding materialized view of the OLAP cubes you are refreshing before you execute OLAP_MAP. (For the mapping between OLAP cube and materialized views, please refer to *Oracle Communication Data Model Reference*.

Recovering from Errors During an Incremental Load of OLAP Cubes

To recover from any errors during the execution of OLAP_MAP during an incremental load of OLAP cubes, take the following steps:

1. Change the value of the BUILD_METHOD column of the `ocdm_sys.DWC_OLAP_ETL_PARAMETER` table to `"C"`.
2. In Oracle Warehouse Builder, rerun the OLAP_MAP map.

Refreshing Data Mining Models

The `MINING_FLW` sub-process flow of the `INTRA_ETL_FLW` process flow triggers the data mining model refreshment. After the initial load of the warehouse, it is recommended to refresh the data mining models monthly. Refreshing the data models is integrated into the `MINING_FLW` sub-process flow. You can also manually refresh the data models.

To manually refresh all mining models, please call the following procedure.

```
ocdm_mining.PKG_OCDM_MINING.REFRESH_MODEL( MONTH_CODE,P_PROCESS_NO)
```

To manually recreate only one mining model, you can call the corresponding procedure. For example, to recreate ther churn SVM model, you can call the following procedure.

```
ocdm_mining.create_churn_svm_model( MONTH_CODE );
```

See also: "Troubleshooting Data Mining Model Creation" on page 3-13

Managing Errors During Intra-ETL Execution

This topic discusses how you can identify and manage errors during Intra-ETL execution. It contains the following topics:

Monitoring the Execution of the Intra-ETL Process

Two control table in the `ocdm_sys` schema, `DWC_INTRA_ETL_PROCESS` and `DWC_INTRA_ETL_ACTIVITY`, monitor the execution of the Intra-ETL process. These tables are documented in *Oracle Communication Data Model Reference*.

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 a system generated unique activity key in ACTIVITY_KEY, the process key value corresponding to the Intra-ETL process in PROCESS_KEY, an individual program name as the ACTIVITY_NAME, a suitable activity description in ACTIVITY_DESC, `SYSDATE` as the value of activity start time, and `RUNNING` as the activity status
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'`.

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` and `DWC_INTRA_ETL_ACTIVITY` tables corresponding to the maximum process key. Monitoring can be done both during and after the execution of the Intra-ETL procedure.

Recovering an Intra ETL Process

To recover an intra-ETL process

1. Identify the errors by looking at the corresponding error details are tracked against the individual programs in the `DWC_INTRA_ETL_ACTIVITY` table.

2. Correct the causes of the errors.
3. Re-invoke the Intra-ETL process.

INTRA_ETL_FLW has the intelligence of identifying whether it is a normal run or recovery run by referring the DWC_INTRA_ETL_ACTIVITY table. During a recovery run, INTRA_ETL_FLW executes only the necessary programs. For example, in the case of 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 executes the aggregate population programs and materialized view refresh 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 only needs to take correct the causes of the errors and re-invoke the Intra-ETL process once more. The Intra-ETL process identifies and executes the programs that generated errors.

See also: "Recovering from Errors During an Incremental Load of OLAP Cubes" on page 3-10.

Troubleshooting Intra-ETL Performance

To troubleshoot the Intra-ETL performance:

- Check the execution plan as described in "Checking the Execution Plan" on page 3-12
- Monitor parallel DML executions as described in "Monitoring PARALLEL DML Executions" on page 3-12

Checking the Execution Plan

Use the SQLDeveloper or other tools to view the package body of the code generated by Oracle Warehouse Builder.

For example, take the following steps to examine DWD_ACCT_PYMT_DAY__MAP .

1. Copy out the main query statement from code viewer.

   To do this, you copy from "CURSOR "AGGREGATOR_c" IS …." to end of the query, which is right above another "CURSOR "AGGREGATOR_c$1" IS".

2. In SQLDeveloper worksheet, issue the following command to turn on the parallel DML:

   ```
   Alter session enable parallel dml;
   ```

3. Paste the main query statement into another SQLDeveloper worksheet and view the execution plan by clicking F6.

   Carefully examine the execution plan to make the mapping runs according to an valid plan.

Monitoring PARALLEL DML Executions

Check that you are running mapping in parallel mode by executing the following SQL statement to count the executed "Parallel DML/Query" statement

```
column name format a50
column value format 999,999
SELECT NAME, VALUE
FROM GV$SYSSTAT
WHERE UPPER (NAME) LIKE '%PARALLEL OPERATIONS%'
```

```
  OR UPPER (NAME) LIKE '%PARALLELIZED%'
  OR UPPER (NAME) LIKE '%PX%'
;
```

If you are running mapping in parallel mode, you should see "DML statements parallelized " increased by 1 every time the mapping was invoked. If not you do not see this increase, then the mapping was not invoked as "parallel DML".

If you see "queries parallelized" increased by 1 (one) instead, then typically it means that the SELECT statement inside of the INSERT was parallelized but that INSERT itself was not.

See also: "Working with Parallelization" on page 4-7

Troubleshooting Data Mining Model Creation

Once the data mining models are created, check the error log in `ocdm_sys.dwc_intra_etl_activity` table. For example, execute the following code.

```
set line 160
col ACTIVITY_NAME format a30
col ACTIVITY_STATUS format a20
col error_dtl format a80
select activity_name, activity_status,  error_dtl from dwc_intra_etl_activity;
```

If all models are created successfully, the activity_status will be all `"COMPLETED-SUCCESS"`. If the activity_status is `"COMPLETED-ERROR"` for a certain step, please check the ERROR_DTL column, and fix the problem accordingly.

Some common error messages from ERROR_DTL and ACTIVITY_NAME are listed below.

ORA-20991: Message not available ... [Language=ZHS]CURRENT_MONTH_KEY

This error may happen when there are not enough data in the `DWR_BSNS_MO` table. For example, if the calendar data is populated with 2004~2009 data, the mining model refresh for Year 2010 may result in this error.

To fix this error, execute the Oracle Communications Data Model calendar utility script again to populate the calendar with sufficient data. For example:

```
Execute Calendar_Population.run('2005-01-01',10);
```

Tip: *Oracle Communication Data Model Reference* for information on the calendar population utility script.

Message not available ... [Language=ZHS]

'ZHS' is a code for a language. The language name it relates to can appear as different name depending on the database environment. This error happens when `ocdm_sys.DWC_MESSAGE.LANGUAGE` does not contain messages for the current language.

Check the values in the `DWC_MESSAGE` table and, if required, update to the language code specified by the Oracle session variable `USERENV('lang')`.

ORA-40113: insufficient number of distinct target values, for "create_churn_svm_model"

This error happens when the target column for the training model contains only one value or no value when it is expecting more than one value.

For example, for the churn svm model, the target column is:

```
ocdm_mining.DMV_CUST_CHRN_SRC_PRD.chrn_ind
```

To troubleshoot this error:

1. Execute a SQL query to check if there are enough values in this column.

   Using the churn svm model as an example, issue the following statement.

   ```
   select chrn_ind, count(*) from DMV_CUST_CHRN_SRC_PRD  group by chrn_ind;
   ```

   The following is a result of the query.

   ```
   C   COUNT(*)
   - ----------
   1       1228
   0       4911
   ```

2. Check the following tables to make sure customer churn indicators are set properly:
   - `ocdm_sys.dwr_cust.chrn_dt` should contain the value for the churned customers.
   - `ocdm_sys.dwd_acct_sttstc` should contain the correct value for each customer in the most recent six months.
3. Execute the following statement to refresh the mining source materialized views in the `OCDM_MINING` schema:

   ```
   exec pkg_ocdm_mining.refresh_mining_source;
   ```

ORA-40112:insufficient number of valid data rows, for "create_ltv_glmr_model"

For this model, target column is `OCDM_MINING.DMV_CUST_LTV_PRDCT_SRC.TOT_PYMT_RVN`.

To troubleshoot this error:

1. Execute the following SQL statement:

   ```
   select count(TOT_PYMT_RVN) from DMV_CUST_LTV_PRDCT_SRC;
   ```

2. Check to see that the value returned by this query is greater than 0 (zero) and similar to number of customers. If the number is 0 or too small, check the Oracle Communications Data Model Intra-ETL execution status as described in "Monitoring the Execution of the Intra-ETL Process" on page 3-11.

ORG-11130:no data found in the collection, for "create_sentiment_svm_model"

This error occur when there is not enough data in the source table for Text sentiment model training: `ocdm_mining.dm_cust_cmmnt.`

To ensure that some text is loaded for customer sentiment analysis:

1. Issue the following SQL statement.

   ```
   Select OVRAL_RSLT_CD, count(CUST_COMMENT) from DWB_EVT_PRTY_INTRACN
   group by OVRAL_RSLT_CD;
   ```

2. Check the number of text comments from the customer interaction table (`DWB_EVT_PRTY_INTRACN`.
3. If there is not enough data in the customer interaction table, check the ETL logic from the source system to the Oracle Communications Data Model.

4

Working with an Oracle Communications Data Model Warehouse

In general, you manage an Oracle Communications Data Model data warehouse in much the same way that you manage any other data warehouse. After the initial data load, you perform incremental data loading as described in "Performing Incremental Data Loading of the Warehouse" on page 3-7.

This chapter discusses considerations other than incremental data loading that are specific to an Oracle Communications Data Model data warehouse. This chapter includes the following topics:

- Customizing the Reports Delivered with Oracle Communications Data Model
- Writing Your Own Queries and Reports
- Modifying and Creating New OLAP Cubes
- Modifying Data Mining Models
- Changing the Tablespaces and Partitions Used by Tables
- Working with Compression
- Working with Parallelization
- Working with User Privileges

Customizing the Reports Delivered with Oracle Communications Data Model

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

The sample reports were developed using Oracle Business Intelligence Suite Enterprise Edition which is a comprehensive suite of enterprise BI 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 use Oracle Business Intelligence Suite Enterprise Edition tools to customize and troubleshoot the execution of reports, as described in:

- "Tools for Customizing Reports" on page 4-2
- "Troubleshooting Reporting Performance" on page 4-2

See: *Oracle Communication Data Model Reference* for detailed information on the sample reports.

Note: The reports and dashboards that are used in examples and delivered with Oracle Communications Data Model are provided only for demonstration purposes. They are not supported by Oracle.

Tools for Customizing 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.

Troubleshooting Reporting 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 **ocdm**.

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

2. Open the Oracle Communications 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 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.I f an error was returned in your last accessed report, there will be an error at the end of this log.

For example:

- **Error:** `Query Status: Query Failed: [nQSError: 15018] Incorrectly defined logical table source (for fact table Customer Mining) does not contain mapping for [Customer.Cell Phone Number, Customer.Customer Segment Name, Customer.Party Name].`

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

  Action: Check the mapping for Customer.Cell Phone Number, Customer.Customer Segment Name, and Customer.Party Name.

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

  Meaning: 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 database.

  Action: To find out which table has problem: Copy the sql query to database environment and execute the query. The table which does not exist will be marked out by the database client.

- **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 because 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.

Writing Your Own Queries and Reports

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

> **See:** *Oracle Communication Data Model Reference* for more information on Oracle Communications Data Model relational tables and views.

Oracle Communications Data Model also supports On Line Analytic Processing (OLAP) reporting through the use of OLAP cubes defined in the `ocdm_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:** *Oracle Communication Data Model Reference* for more information on Oracle Communications Data Model OLAP cubes and the relational views by which you can access them, and *Oracle OLAP Application Developer's Guide* for information on how to create queries and reports on OLAP cube data.

Example 4–1 Creating a Relational Query

For example, assume that you want to know the total call minutes for the top ten customers in the San Francisco area for Mar 2009. To answer this question, you might have to query the tables described in the following table.

Entity Name	Table Name	Description
WIRELESS CALL EVENT	DWB_WRLS_CALL_EVT	Occurrences of the wireless call.
CUSTOMER	DWR_CUST	Individual customers
ADDRESS LOCATION	DWR_ADDR_LOC	All addresses. The table has levels as country, state, city, address, and so on.
GEOGRAPHY CITY	DWR_GEO_CITY	The CITY level of the GEOGRAPHY hierarchy.

To make the query, you execute the following SQL statement.

```
SELECT cust_key, tot_call_min FROM
(select round(sum(call.call_drtn)/60,2) tot_call_min , call.cust_key
from DWB_WRLS_CALL_EVT call,
DWR_CUST     cust,
DWR_ADDR_LOC addr,
DWR_GEO_CITY city
         Where to_date(to_char(call.evt_begin_dt,'MON-YY'),'MON-YY') like
to_date('MAR-07','MON-YY')
and cust.cust_key = call.cust_key
and cust.addr_loc_key = addr.addr_loc_key
and addr.geo_city_key = city.geo_city_key
and initcap(city.geo_city_name)='San Francisco'
group by call.cust_key
order by 1 desc) WHERE ROWNUM < 10;
```

The result of this query is shown below.

```
  CUST_KEY TOT_CALL_MIN
---------- ------------
      3390        101.6
      4304       100.25
      4269        97.37
      4152        93.02
      4230        92.97
      4157        92.95
      3345        91.62
      4115        48.43
      4111        44.48
```

Modifying and Creating New OLAP Cubes

You create new OLAP cubes in Oracle Communications Data Model in much the same way as you would in any other data warehouse. See *Oracle OLAP Application Developer's Guide* for more information.

Additionally, since all OLAP cubes are loaded with data from the materialized views that have a `DWA_` prefix, when working with OLAP cubes in Oracle Communications Data Model, keep the following points in mind:

- Immediately after installation, all materialized views underlying the OLAP cubes are disabled by default. To enable the cube materialized views, you must follow the steps outlined in "Enabling Cube Materialized Views" on page 4-4.
- To change the measures or dimensions of a cube, you must follow the steps outlined in "Changing an Oracle Communications Data Model OLAP Cube" on page 4-5.
- There are special considerations when working with Oracle Communications Data Model cubes for forecasts as discussed in "Working with Forecast Cubes" on page 4-5

Enabling Cube Materialized Views

To enable the cube materialized views, take 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.

Changing an Oracle Communications Data Model OLAP Cube

Since all Oracle Communications Data Model cubes load data from tables with the DWA_ prefix, if you want to change the measures or dimensions of one cube, you must take the following steps:

1. Use the information in *Oracle Communication Data Model Reference*, to 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.

Working with Forecast Cubes

You cannot enable materialized views for an Oracle Communications Data Model forecast cube.

To create a new forecast in Oracle Communications Data Model:

1. Create a new cube to contain the results of the forecast.
2. Write an OLAP DML forecasting context program as described in *Oracle OLAP DML Reference*.

Modifying Data Mining Models

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

1. Change the definition for source materialized views used as input to the mining model.

   All Oracle Communications Data Model mining models use materialized views as source input. Those materialized views are defined in `ocdm_mining_etl.sql` file in `$ORACLE_HOME/ocdm/pdm/mining/src`. Different mining models use different source materialized views.

   For example, the activity `create_churn_svm_model` uses the DMV_CUST_CHRN_SRC_PRD source materialized view. To add a new column into this model, modify the following materialized views to add a new column:

   ```
   DMV_CUST_CHRN_SRC_SRC
   DMV_CUST_CHRN_SRC_PRD
   DMV_CUST_CHRN_SRC_TST
   DMV_CUST_CHRN_APPLY_ALL
   ```

2. Train the model again by calling Oracle Communications Data Model mining package.

   For example, to train the churn svm model, execute the following statement.

   ```
   ocdm_mining.create_churn_svm_model( MONTH_CODE );
   ```

3. Ensure that the new column has been added into the model.

   For example, use the following statement to query the result table and ensure the new column name is included in the query result:

```
  SELECT attribute_name FROM TABLE(
    SELECT ATTRIBUTE_SET ROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_
SVM('OCDM_CHURN_SVM'))
    WHERE CLASS='1' );
```

   See also: "Refreshing Data Mining Models" on page 3-10, "Troubleshooting Data Mining Model Creation" on page 3-13, and "Working with User Privileges" on page 4-8

Changing the Tablespaces and Partitions Used by Tables

You can change the tablespace and partitions used by tables. What you do depends on whether or not the Oracle Communications 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 as described in "Diverting Partitions into New Tablespaces" on page 4-6.
- For tables that have partitions (that is, base, derived, and aggregate tables), you can specify that new interval partitions be generated into new tablespaces as described in "Changing an Existing Tablespace" on page 4-6.

Diverting Partitions into New Tablespaces

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

Consequently, for Oracle Communications Data Model tables that have partitions (that is, Base, Derived, and Aggregate tables), if you want the new interval partitions to be generated in new tablespaces rather than current ones, please 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*.

Changing an Existing Tablespace

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

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

Working with Compression

By using a compression algorithm specifically designed for relational data, Oracle is able to compress data much more effectively than standard compression techniques. More significantly, unlike other compression techniques, Oracle incurs virtually no performance penalty for SQL queries accessing compressed tables.

Oracle Communications 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;
```

For example, you could issue the following statements to create a compressed table named DWB_ACCS_MTHD_PORT_HIST.

```
COMPRESS
Create table DWB_ACCS_MTHD_PORT_HIST
(NP_RQST_HDR_CD                 VARCHAR2(30)
,ACCS_MTHD_KEY                   NUMBER(30)  NOT NULL ENABLE
,EXTRNL_OPRTR_KEY                NUMBER(30)
…..
)
tablespace TBS_BASE
COMPRESS ;
```

Working with Parallelization

Oracle Communications Data Model leverages the parallel query and parallel DML feature of Oracle Database. Parallel operations speed up DML statement execution by dividing the work among multiple child processes. Each child process executes its portion of the work under its own parallel process transaction.

As shown in the following example, all Oracle Communications Data Model base, derived, and aggregate tables are, by default, created with initial parallel degree of `4`.

```
PARALLEL (DEGREE 4)
Create table DWB_ACCS_MTHD_PORT_HIST
(NP_RQST_HDR_CD                 VARCHAR2(30)
,ACCS_MTHD_KEY                   NUMBER(30)  NOT NULL ENABLE
,EXTRNL_OPRTR_KEY               NUMBER(30)
…..
)
tablespace TBS_BASE
PARALLEL (DEGREE 4) ;
```

If you have a bigger machine, (for example 16 CPUs), then you can alter the degree of parallelization for the entire session as descried in "Enabling Parallel Execution for a Session" on page 4-7 or for certain DML operations as described in "Enabling Parallel Execution of DML Operations" on page 4-8.

Regardless at which level you enable prallelism, the setting of parallelism for a table influences the optimizer. Consequently, when using parallel query, also enable parallelism at the table level as described in "Enabling Parallel Execution at the Table Level" on page 4-8.

Enabling Parallel Execution for a Session

Parallel query is the most commonly used of Oracle's parallel execution features. 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;
```

Enabling Parallel Execution of DML Operations

Data Manipulation Language (DML) operations such as INSERT, UPDATE, and DELETE can be parallelized by Oracle. 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 applies a set of rules to determine whether that statement can be parallelized. The rules vary depending on whether or not the statement is a DML INSERT statement, or a DML UPDATE or DELETE statement.

Rules for Parallelizing DML UPDATE and DELETE statements

The following rules apply when determining how to parallelize DML UPDATE and DELETE statements:

- Oracle 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.

Rules for Paralellizing DML INSERT statements

The following rules apply when determining how to parallelize DML INSERT statements:

- Standard INSERT statements using a VALUES clause cannot be parallelized.
- Oracle can parallelize only INSERT . . . SELECT . . . FROM statements.

Enabling Parallel Execution at the Table Level

The setting of parallelism for a table influences the optimizer. Consequently, when using parallel query, also enable parallelism at the table level by issuing the following statement.

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

Working with User Privileges

Installing the Oracle Communications Data Model component creates two accounts: the OCDM_SYS and OCDM_MINING accounts. Installing the Oracle Communications Data Model sample reports create the OCDM_SAMPLE_SYS account. Please make sure you unlock those two accounts with new password following "Post-installation" section.

> **See:** *Oracle Communication Data Model Installation Guide* for information on installing Oracle Communications Data Model and for unlocking the OCDM_SYS and OCDM_MINING accounts

The OCDM_SYS and OCDM_MINING accounts serve different purposes:

- **ocdm_sys** is the main schema for Oracle Communications Data Model. This schema contains all the relational and OLAP components of Oracle Communications Data Model.

  The Oracle Communications Data Model data mining result tables are also in this schema. The final output from the data mining models is saved back into OCDM_SYS schema for easy integration with other components. With mining result tables in OCDM_SYS schema, customer can run report joining relational content with Mining results together much easily. For example, customer can query for revenue sum of customers belonging to a specific segment according to Customer Segmentation model.

- **ocdm_mining** is the data mining schema of Oracle Communications Data Model. This schema contains all the mining components of Oracle Communications Data Model *except* the final result tables.

The installation process grants the necessary privileges required for users of both accounts. Once you have installed the product, you only need to consider user privileges in the following situations:

- The `ocdm_mining` schema requires select privileges to only a small amount of tables in the `ocdm_sys` schema, (that is, only those tables used to hold the results of the data models shipped with Oracle Communications Data Model). If you decide to create additional data mining models that which requires more table access from `ocdm_sys` schema, grant the select privilege explicitly.
- The Intra-ETL programs are executed inside the `ocdm_sys` schema, therefore, they require the full access to the that schema. By default, the Oracle Warehouse Builder Intra-ETL mappings for Oracle Communications Data Model connect to the ocdm_sys schema for intra-ETL execution.
- By default, the Oracle Communications Data Model sample reports connect to the ocdm_sys schema directly. For security reason, you may want to grant only select privileges to those reporting users. To do this, take the following steps:

  1. Create a dedicated reporting user (for example, OCDM_Report).
  2. Grant select privilege for all Oracle Communications Data Model tables required for reporting to OCDM_Report. (The easiest way to do this is to grant all Oracle Communications Data Model tables that start with a prefix of DWA, DWB, DWD, DWR, or DWL.)
  3. Create a view (or synonym) in OCDM_Report schema that points to the OCDM_SYS tables.
  4. In the Oracle Business Intelligence Suite Enterprise Edition repository for Oracle Communications Data Model, change the connection information to point to the new schema.

Index

P

R

S

T