

Oracle® Communications Data Model

Implementation and Operations Guide

Release 11.3.1

E28442-04

September 2012

Copyright © 2011, 2012, Oracle and/or its affiliates. All rights reserved.

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

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

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

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

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

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

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

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

1 Introduction to Oracle Communications Data Model Customization	
What is the Oracle Communications Data Model?	1-1
Components of the Oracle Communications Data Model	1-1
Oracle Products That Make Up Oracle Communications Data Model	1-2
Steps for Implementing an Oracle Communications Data Model Warehouse	1-3
Before You Begin Customizing the Oracle Communications Data Model	1-4
Prerequisite Knowledge for Implementers	1-4
Responsibilities of a Data Warehouse Governance Committee.....	1-5
Managing the Metadata for Oracle Communications Data Model	1-6
Metadata Categories and Standards	1-6
Working with a Metadata Repository	1-7
Browsing the Metadata Repository Supplied With Oracle Communications Data Model	1-7
Using Oracle Warehouse Builder with the Oracle Communications Data Model.....	1-9
Performing Fit-Gap Analysis for Oracle Communications Data Model	1-10
2 Physical Model Customization	
Characteristics of the Default Physical Model	2-1
Customizing the Oracle Communications Data Model Physical Model	2-3
Questions to Answer Before You Customize the Physical Model.....	2-4
Conventions When Customizing the Physical Model.....	2-4
Foundation Layer Customization	2-5
Common Change Scenarios.....	2-6
General Recommendations When Designing Physical Structures	2-7
Tablespaces in the Oracle Communications Data Model	2-7
Data Compression in the Oracle Communications Data Model.....	2-8
Types of Data Compression Available	2-8
Basic or Standard Compression	2-8
OLTP Compression	2-9
Hybrid Columnar Compression (HCC)	2-9
Tables for Supertype and Subtype Entities in Oracle Communications Data Model.....	2-10
Surrogate Keys in the Physical Model	2-10
Integrity Constraints in Oracle Communications Data Model	2-11
Indexes and Partitioned Indexes in the Oracle Communications Data Model.....	2-11
Partitioned Tables in the Oracle Communications Data Model	2-12
Partitioning the Oracle Communications Data Model for Manageability	2-13

Partitioning the Oracle Communications Data Model for Easier Data Access.....	2-13
Partitioning the Oracle Communications Data Model for Join Performance	2-13
Parallel Execution in the Oracle Communications Data Model.....	2-14
Enabling Parallel Execution for a Session	2-16
Enabling Parallel Execution of DML Operations	2-16
Enabling Parallel Execution at the Table Level	2-17

3 Access Layer Customization

Introduction to Customizing the Access Layer of Oracle Communications Data Model.....	3-1
Derived Tables in the Oracle Communications Data Model.....	3-2
Creating New Derived Tables for Calculated Data	3-2
Customizing Oracle Communications Data Model Data Mining Models	3-2
Creating a New Data Mining Model for Oracle Communications Data Model.....	3-2
Modifying Oracle Communications Data Model Data Mining Models	3-3
Tutorial: Customizing the Churn Prediction Model.....	3-4
Tutorial Prerequisites	3-4
Preparing Your Environment	3-5
Generating the Churn Prediction Model.....	3-7
Checking the Result.....	3-8
Aggregate Tables in the Oracle Communications Data Model	3-9
Dimensional Components in the Oracle Communications Data Model	3-10
Characteristics of a Dimensional Model	3-10
Characteristics of Relational Star and Snowflake Tables	3-12
Declaring Relational Dimension Tables	3-13
Validating Relational Dimension Tables	3-13
Characteristics of the OLAP Dimensional Model	3-13
Oracle OLAP Cube Views	3-14
Cube Materialized Views.....	3-15
Characteristics of the OLAP Cubes in Oracle Communications Data Model.....	3-16
Defining New Oracle OLAP Cubes for Oracle Communications Data Model.....	3-17
Changing an Oracle OLAP Cube in Oracle Communications Data Model	3-18
Creating a Forecast Cube for Oracle Communications Data Model	3-18
Choosing a Cube Partitioning Strategy for Oracle Communications Data Model.....	3-18
Choosing a Cube Data Maintenance Method for Oracle Communications Data Model	3-19
Materialized Views in the Oracle Communications Data Model.....	3-20
Types of Materialized Views and Refresh options.....	3-21
Refresh Options for Materialized Views with Aggregates	3-21
Refresh Options for Materialized Views Containing Only Joins.....	3-22
Refresh Options for Nested Materialized Views.....	3-23
Choosing Indexes for Materialized Views	3-23
Partitioning and Materialized Views	3-23
Compressing Materialized Views.....	3-25

4 ETL Implementation and Customization

The Role of ETL in the Oracle Communications Data Model	4-1
ETL for the Foundation Layer of an Oracle Communications Data Model Warehouse.....	4-2
Using an Application Adapter to Populate the Foundation Layer	4-2

Writing Your Own Source-ETL.....	4-3
Source-ETL Design Considerations.....	4-3
ETL Architecture for Oracle Communications Data Model Source-ETL	4-4
Creating a Source to Target Mapping Document for the Source-ETL	4-5
Designing a Plan for Rectifying Source-ETL Data Quality Problems	4-5
Designing Source-ETL Workflow and Jobs Control.....	4-5
Designing Source-ETL Exception Handling	4-6
Writing Source-ETL that Loads Efficiently	4-6
Using a Staging Area for Flat Files.....	4-6
Preparing Raw Data Files for Source-ETL	4-6
Source-ETL Data Loading Options	4-7
Parallel Direct Path Load Source-ETL	4-8
Partition Exchange Load for Oracle Communications Data Model Source-ETL	4-8
Customizing Intra-ETL for the Oracle Communications Data Model	4-9
Performing an Initial Load of an Oracle Communications Data Model Warehouse.....	4-11
Performing an Initial Load of the Foundation Layer.....	4-11
Performing an Initial Load of the Access Layer	4-11
Executing the Default Oracle Communications Data Model Intra-ETL.....	4-13
Executing the INTRA_ETL_FLW Workflow from Oracle Warehouse Builder.....	4-13
Manually Executing the ocdm_execute_wf.sh Script.....	4-13
Refreshing the Data in an Oracle Communications Data Model Warehouse.....	4-14
Refreshing the Foundation Layer of Oracle Communications Data Model Warehouse.....	4-14
Refreshing the Access Layer of an Oracle Communications Data Model Warehouse	4-15
Refreshing Oracle Communications Data Model Derived and Aggregate Tables	4-15
Refreshing Oracle Communications Data Model OLAP Cubes	4-15
Refreshing Oracle Communications Data Model Data Mining Models.....	4-16
Managing Errors During Oracle Communications Data Model Intra-ETL Execution	4-17
Monitoring the Execution of the Intra-ETL Process.....	4-17
Recovering an Intra ETL Process	4-18
Troubleshooting Intra-ETL Performance.....	4-18
Checking the Execution Plan.....	4-18
Monitoring PARALLEL DML Executions.....	4-19
Troubleshooting Data Mining Model Creation.....	4-19

5 Report and Query Customization

Reporting Approaches in Oracle Communications Data Model.....	5-1
Customizing Oracle Communications Data Model Sample Reports	5-2
Writing Your Own Queries and Reports.....	5-3
Optimizing Star Queries	5-4
Troubleshooting Oracle Communications Data Model Report Performance	5-6
Writing As Is and As Was Queries	5-7
Characteristics of an As Is Query.....	5-8
Characteristics of an As Was Query	5-8
Examples: As Is and As Was Queries Against Oracle Communications Data Model.....	5-8
Tutorial: Creating a New Oracle Communications Data Model Dashboard	5-13
Tutorial: Creating a New Oracle Communications Data Model Report	5-16

A	Sizing and Configuring an Oracle Communications Data Model Warehouse	
	Sizing an Oracle Communications Data Model Warehouse.....	A-1
	Configuring a Balanced System for Oracle Communications Data Model.....	A-3
	Maintaining High Throughput in an Oracle Communications Data Model Warehouse	A-3
	Configuring I/O in an Oracle Communications Data Model for Bandwidth not Capacity..	A-3
	Planning for Growth of Your Oracle Communications Data Model.....	A-4
	Testing the I/O System Before Building the Warehouse	A-4
	Balanced Hardware Configuration Guidelines for Oracle Communications Data Model	A-4
B	Working with User Privileges in Oracle Communications Data Model	
	Accounts Created for Oracle Communications Data Model	B-1
	When You Must Consider User Privileges in an Oracle Communications Data Model.....	B-2
	Granting Only Select Privileges to Users of the Sample Reports	B-2
C	Using the NCC Adapter for Oracle Communications Data Model	
	What is the NCC Adapter for Oracle Communications Data Model?	C-1
	Database Schemas Used by the NCC Adapter for Oracle Communications Data Model	C-2
	Oracle Data Integrator.....	C-2
	Oracle GoldenGate.....	C-2
	Populating a Warehouse Using the NCC Adapter for Oracle Communications Data Model..	C-3
	Initial Loading Using the NCC Adapter for Oracle Communications Data Model.....	C-3
	Refreshing the Data Using the NCC Adapter for Oracle Communications Data Model.....	C-4
	Refreshing the Foundation Layer on a Scheduled Basis	C-4
	Refreshing the Foundation Layer with Real-Time Data	C-4
	NCC Adapter for Oracle Communications Data Model Execution Flows.....	C-6
	Execution Flow When Using the Adapter Oracle GoldenGate and ODI	C-6
	Execution Flow When Using the Adapter with only ODI.....	C-7
	Components of the NCC Adapter for Oracle Communications Data Model.....	C-8
	NCC Source Schema Objects	C-8
	Schema Definitions Added by the NCC Adapter for Oracle Communications Data Model	C-8
	ncc_stg Schema Objects.....	C-8
	Definitions Added to the ocdm_sys Schema	C-9
	The NCC_OCDM Project	C-10
	SRC_STG_NONAGG Folder.....	C-10
	SRC_STG_LOAD_NON_OGG_PKG Package	C-10
	SRC_STG_EDR_LOAD_INCREMENTAL_PKG Package.....	C-10
	SRC_STG_EDR_LOAD_INITIAL_PKG Package.....	C-10
	STG_OCDM Folder	C-10
	STG_OCDM_INITIAL_LOAD_PKG Package.....	C-10
	STG_OCDM_EDR_PKG Package.....	C-11
	STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG Package	C-11
	Oracle Communications Data Model Tables and Columns Filled by NCC Adapter	C-12
D	Using the Oracle Communications Billing and Revenue Management Adapter	
	What is the BRM Adapter for Oracle Communications Data Model?.....	D-1
	Components of BRM Adapter for Oracle Communications Data Model	D-1

BRM Source Schema Tables.....	D-2
BRM Adapter for Oracle Communications Data Model Database Schemas	D-3
Schema Definitions Added for the BRM Adapter for Oracle Communications Data Model	D-4
BRM Adapter for Oracle Communications Data Model Staging Schema Objects.....	D-4
BRM Adapter for Oracle Communications Data Model OCDM_SYS Objects	D-5
Oracle Data Integrator.....	D-5
Oracle GoldenGate Replication.....	D-6
BRM Adapter for Oracle Communications Data Model Execution Flows	D-6
Execution Flow Using BRM Adapter: ODI Pull Version (without Oracle GoldenGate).....	D-6
Execution Flow Using BRM Adapter: with Oracle GoldenGate and ODI.....	D-7
Populating a Warehouse Using the BRM Adapter for Oracle Communications Data Model .	D-8
Initial Loading Using the BRM Adapter for Oracle Communications Data Model.....	D-8
Refreshing the Data Using the BRM Adapter for Oracle Communications Data Model (Change Data Capture: CDC) D-9	
Refreshing the Foundation Layer on a Scheduled Basis	D-9
Refreshing the Foundation Layer with Real-Time Data	D-10
ODI BRM_OCDM Project	D-11
Objects in SRC_STG_NONOGG Folder of BRM_OCDM Project.....	D-12
SRC_STG_LOAD_NON_OGG_PKG Package	D-14
Objects in STG_OCDM Folder of BRM_OCDM.....	D-14
STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG Package	D-15
Oracle GoldenGate Process Checking	D-16

E Upgrading Oracle Communications Data Model

Considerations for Upgrading Oracle Communications Data Model	E-1
Upgrade Paths for Oracle Communications Data Model.....	E-2
Regression Testing for Oracle Communications Data Model	E-2

Index

List of Examples

2-1	Creating a Compressed Table for Oracle Communications Data Model	2-9
3-1	Adding a New Column to a Mining Model in Oracle Communications Data Model	3-3
3-2	Refreshing Oracle Communications Data Model Nested Materialized Views	3-23
4-1	Using Exchange Partition Statement with a Partitioned Table	4-9
4-2	Troubleshooting an ORA-20991 Error for Oracle Communications Data Model	4-20
4-3	Troubleshooting the "Message not available ... [Language=ZHS]" Error.....	4-20
4-4	Troubleshooting an ORA-40113 Error for Oracle Communications Data Model	4-20
4-5	Troubleshooting an ORA-40112 Error for Oracle Communications Data Model	4-21
4-6	Troubleshooting an ORG-11130 Error for Oracle Communications Data Model	4-21
5-1	Creating a Relational Query for Oracle Communications Data Model.....	5-3
5-2	Star Transformation.....	5-5
5-3	Troubleshooting an Oracle Communications Data Model Report.....	5-7
5-4	Troubleshooting a Report: A Table Does Not Exist.....	5-7
5-5	Troubleshooting a Report: When the Database is Not Connected	5-7
5-6	As Is Query for Tax Collection Split by Marital Status	5-10
5-7	As Was Queries for Tax Collection Split by Marital Status	5-10
5-8	As Is Query for Tax Collection Data Split by County	5-11
5-9	As Was Queries for Tax Collection Data Split by County	5-12

List of Figures

2-1	Layers of an Oracle Communications Data Model Warehouse.....	2-2
2-2	Partitioning for Join Performance.....	2-14
2-3	Parallel Execution of a Full Partition-Wise Join Between Two Tables.....	2-15
2-4	Partial Partition-Wise Join	2-16
3-1	Star Schema Diagram	3-12
3-2	Diagram of the OLAP Dimensional Model.....	3-14
4-1	ETL Flow Diagram.....	4-2
C-1	Execution Flow When Using the Adapter with Oracle GoldenGate	C-6
C-2	Execution Flow When Using the Adapter without Oracle GoldenGate.....	C-7
D-1	Execution Flow When Using the Adapter without Oracle GoldenGate.....	D-6
D-2	Execution Flow When Using the BRM Adapter with Oracle GoldenGate.....	D-7

Preface

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

This preface contains the following topics:

- [Audience](#)
- [Documentation Accessibility](#)
- [Related Oracle Resources](#)
- [Conventions](#)

Audience

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

Oracle Communications Data Model Documentation Set

For more information on Oracle Communications Data Model, see the following documents in the Oracle Communications Data Model Release 11g documentation set:

- *Oracle Communications Data Model Installation Guide*
- *Oracle Communications Data Model Reference*
- *Oracle Communications 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

www.oracle.com/technetwork/indexes/documentation/index.html

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.

For example, search within "All" for "OCDMTutorial" to go to the tutorial for Oracle Communications Data Model.

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.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Introduction to Oracle Communications Data Model Customization

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

- [What is the Oracle Communications Data Model?](#)
- [Steps for Implementing an Oracle Communications Data Model Warehouse](#)
- [Before You Begin Customizing the Oracle Communications Data Model](#)
- [Managing the Metadata for Oracle Communications Data Model](#)
- [Performing Fit-Gap Analysis for Oracle Communications Data Model](#)

What is the Oracle Communications Data Model?

Oracle Communications Data Model is a standards-based, pre-built approach to communications data warehousing enabling a communications company to realize the power of insight more quickly. Oracle Communications Data Model reduces costs for both immediate and on-going operations by leveraging out-of-box Oracle based data warehouse and business intelligence solutions, making world-class database and business intelligence technology solutions available with a communications specific data model. You can use Oracle Communications Data Model in any application environment. Also, you can easily extend the model.

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

Oracle Communications Data Model provides much of the data modeling work that you must do for a communications business intelligence solution. The Oracle Communications Data Model logical and physical data models were designed following best practices for communications service providers. Oracle Communications Data Model is conformance certified with TM Forum's Information Framework (SID) version 8.1 as discussed in *Oracle Communications Data Model Reference*.

- [Components of the Oracle Communications Data Model](#)
- [Oracle Products That Make Up Oracle Communications Data Model](#)

Components of the Oracle Communications Data Model

Oracle Communications Data Model includes the following components:

- Logical model which is a third normal form (3NF) entity-object standards-based model. The logical model is described in *Oracle Communications Data Model Reference*.
- Physical model defined as two Oracle Database schemas. One schema defines all the relational and OLAP physical components of Oracle Communications Data Model, and the tables that hold the results of the data mining models. The other schema defines the actual data mining models.
- Intra-ETL database packages and SQL scripts to extract, transform, and load (ETL) data from the Oracle Communications Data Model 3NF physical tables to the derived and aggregate tables in Oracle Communications Data Model.
- Sample reports and dashboards developed using Oracle Business Intelligence Suite Enterprise Edition.
- DDL and installation scripts

Note: When you use the Oracle Installer to install Oracle Communications Data Model, you have the choice of performing two different types of installations:

- Installation of the Oracle Communications Data Model component, itself
- Installation of sample reports (and schemas)

See *Oracle Communications Data Model Installation Guide* for detailed information on the different types of installation

See: *Oracle Communications Data Model Reference* for detailed descriptions of the components.

Oracle Products That Make Up Oracle Communications Data Model

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

- [Oracle Database with OLAP, Data Mining and Partitioning Option](#)
- [Oracle Communications Data Model Application Adapters](#)
- [Oracle Development Tools](#)
- [Oracle Business Intelligence Suite Enterprise Edition Presentation Tools](#)

Oracle Database with OLAP, Data Mining and Partitioning Option

Oracle Communications 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 Communications Data Model Application Adapters

The Oracle Communications Network Charging and Control Adapter for Oracle Communications Data Model is an option of the Oracle Communications Data Model. You can use this application adapter to populate the foundation layer of an Oracle Communications Data Model warehouse with data from an Oracle Communications Network Charging and Control (NCC) system.

Oracle Development Tools

You can use the following Oracle tools to customize the predefined 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
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 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 Communications Data Model.

See: ["Reporting Approaches in Oracle Communications Data Model"](#) on page 5-1.

Steps for Implementing an Oracle Communications Data Model Warehouse

Although Oracle Communications Data Model was designed following best practices for communications service providers, usually the model requires some customization to meet your business needs.

The reasons that you might customize Oracle Communications Data Model include: your business does not have a business area that is in the logical model of Oracle Communications 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 Communications Data Model warehouse, take the following steps:

1. Perform the organizational tasks outlined in ["Before You Begin Customizing the Oracle Communications Data Model"](#) on page 1-4.
2. Create a fit-gap analysis report by following the process outlined ["Performing Fit-Gap Analysis for Oracle Communications Data Model"](#) on page 1-10.
3. In a development environment, install a copy of the Oracle Communications Data Model.
4. Customize Oracle Communications 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 that layer. When customizing the physical objects, follow the guidelines in ["Foundation Layer Customization"](#) on page 2-5. When writing the ETL, follow the guidelines in ["ETL for the Foundation Layer of an Oracle Communications Data Model Warehouse"](#) on page 4-2.

- b. Access layer of the physical model and the ETL to populate that layer. When designing the physical objects, follow the guidelines in [Chapter 3, "Access Layer Customization."](#) When writing the ETL, follow the guidelines in ["Customizing Intra-ETL for the Oracle Communications Data Model"](#) on page 4-9.
5. In a test environment, make a copy of your customized version of Oracle Communications Data Model. Then, following the documentation you created in Step 2, test the customized version of Oracle Communications Data Model
6. Following your typical procedures, roll the tested customized version of Oracle Communications Data Model out into pre-production and, then, production.

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

Before You Begin Customizing the Oracle Communications Data Model

Before you begin customizing Oracle Communications Data Model, ensure the following teams and committees exist:

- Data warehouse governance steering committee. This steering committee has the responsibilities outlined in ["Responsibilities of a Data Warehouse Governance Committee"](#) on page 1-5.
- Implementation team. This team consists of IT engineers who have the expertise outlined in ["Prerequisite Knowledge for Implementers"](#) on page 1-4. This team has the responsibilities outlined in ["Steps for Implementing an Oracle Communications Data Model Warehouse"](#) on page 1-3.
- Fit-gap analysis team. This team consists of business analysts who can identify the business requirements and scope of the Oracle Communications 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. This team has the responsibilities outlined in ["Performing Fit-Gap Analysis for Oracle Communications Data Model"](#) on page 1-10.

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 Implementers

As outlined in ["Oracle Products That Make Up Oracle Communications Data Model"](#) on page 1-2, the Oracle Communications Data Model uses much of the Oracle stack. Consequently, to successfully implement the Oracle Communications Data Model, the implementation team needs:

- 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, Warehouse Builder, Oracle Business Intelligence Suite Enterprise Edition)
- Hands-on experience using:
 - Oracle Database
 - PL/SQL
 - SQL DDL and DML syntax
 - Analytic Workspace Manager
 - Oracle SQL Developer
 - Oracle Business Intelligence Suite Enterprise Edition Administrator, Answers, and Dashboards

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 Communications 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 as discussed in ["Managing the Metadata for Oracle Communications Data Model"](#) on page 1-6' 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.

Managing the Metadata for Oracle Communications 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 without knowing the context of the data or what the data represents.

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 Communications Data Model so that you can do impact analysis to ensure that changes do not adversely impact data integrity anywhere in your data warehouse.

- [Metadata Categories and Standards](#)
- [Working with a Metadata Repository](#)
- [Browsing the Metadata Repository Supplied With Oracle Communications Data Model](#)
- [Using Oracle Warehouse Builder with the Oracle Communications Data Model](#)

Metadata Categories and Standards

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. Gather this mapping information and business definition and rules information into business metadata.
- **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.

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. A Physical layer. This metadata layer identifies the source data.
2. A 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. The 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 Communications 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.

Browsing the Metadata Repository Supplied With Oracle Communications Data Model

To customize the Oracle Communications Data Model model, you must understand the dependencies among Oracle Communications Data Model components -- especially how the report KPIs are mapped to the physical tables and columns.

Oracle Communications Data Model provides a tool called the "OCDM Metadata" browser that helps you discover these dependencies. When you install Oracle Communications Data Model with its sample reports, the metadata browser is delivered as a sample Dashboard in the webcat.

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

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 `ocdm`, 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.

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

- **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.

For example, you take the following steps to learn more about an entity:

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

- **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.

For example, to see the reports that use the a particular table, take 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).
3. Click **OK**.

4. Select **GO**.

The reports for the selected table are displayed.

Using Oracle Warehouse Builder with the Oracle Communications Data Model

Oracle Warehouse Builder provides excellent metadata management features that you can use with Oracle Communications Data Model.

Before you import source metadata into Oracle Warehouse Builder, you create a module to contain metadata definitions. The type of module you create depends on the source from which you are importing metadata into your Metadata Repository. Oracle Warehouse Builder supports many different sources and targets as discussed in *Oracle Warehouse Builder Sources and Targets Guide*. For example, when connecting to an Oracle database, Oracle Warehouse Builder queries the database dictionary to extract all needed metadata on tables, views, sequences, dimensions, cubes, data types, PL/SQL packages, and so on.

Oracle Warehouse Builder also provides an interface tool called the Metadata Dependency Manager through which you can explore dependencies among data objects, as represented by the metadata in your Oracle Warehouse Builder repository. The Metadata Dependency Manager presents dependencies in the form of interactive lineage and impact diagrams. A lineage diagram traces the data flows for an object back to the sources and displays all objects along those paths. An impact diagram identifies all the objects that are derived from the selected object.

Information provided by the Metadata Dependency Manager can help you in many circumstances. For example:

1. Starting from a target object, such as a dimension, cube, or business intelligence tool report, you can identify the columns in each data source that are used in computing the results in the target.
2. You can assess the impact of design changes in an object such as a source table or a pluggable mapping that is used throughout a project.
3. You can propagate a design change, such as a change to the data type of a source table column, to downstream objects in the design.

Using end-to-end data lineage and impact analysis reduces project risk by allowing better planning for design changes, faster identification of unanticipated impacts when source systems change, and enabling more effective auditing of your business intelligence results, master data or other data integration processes.

You can also define and use SQL- or XML-based custom metadata stores to retrieve definitions of source and target objects such as tables and views.

For data files extracted from some mainframe sources, Oracle Warehouse Builder can interpret Cobol Copybook files that describes the structure of the data file, and create its source metadata based on that.

Oracle Warehouse Builder application adapters or application connectors provide additional metadata about ERP and CRM application sources.

Oracle Warehouse Builder can deploy or execute process flows and schedules to Oracle Enterprise Manager and Oracle Workflow. In general, you can deploy a schedule in any Oracle Database location, release 10g or later.

Performing Fit-Gap Analysis for Oracle Communications Data Model

Fit-gap analysis is where you compare your information needs and communications business requirements with the structure that is available with Oracle Communications 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 Communications Data Model to fit your communications environment.

The fit-gap analysis team writes the customization report by taking 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 Communications Data Model logical model and schema:
 - Starting from business requirements, questions, and rules, identify any entities and attributes that are *not* in the Oracle Communications Data Model.
 - Compare the Oracle Communications Data Model to your existing application model if have one.
 - Compare the Oracle Communications Data Model to the OLTP data that you are using as a data source to the Oracle Communications Data Model warehouse.
3. Determine the differences between your needs and Oracle Communications 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 Communications Data Model Warehouse. Answer the following questions about the differences you find:
 - Which differences you can live with, and which 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 Communications Data Model to create the customized warehouse. Identify these changes in the following order:
 - a. Physical model. Follow the guidelines outlined in [Chapter 2, "Physical Model Customization"](#).
 - b. ETL mapping. Follow the guidelines outlined in [Chapter 4, "ETL Implementation and Customization"](#) to identify and design the source-ETL that you must create and to identify and make any changes to the intra-ETL provided with Oracle Communications 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 Communications 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 and perform the steps outlined in "[Steps for Implementing an Oracle Communications Data Model Warehouse](#)" on page 1-3.

Physical Model Customization

This chapter provides general information about customizing the physical model of Oracle Communications Data Model and more detailed information about customizing the foundation layer of the physical model. This chapter contains the following topics:

- [Characteristics of the Default Physical Model](#)
- [Customizing the Oracle Communications Data Model Physical Model](#)
- [Foundation Layer Customization](#)
- [General Recommendations When Designing Physical Structures](#)

See also: [Chapter 3, "Access Layer Customization"](#)

Characteristics of the Default Physical Model

The default physical of Oracle Communications Data Model defines:

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

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

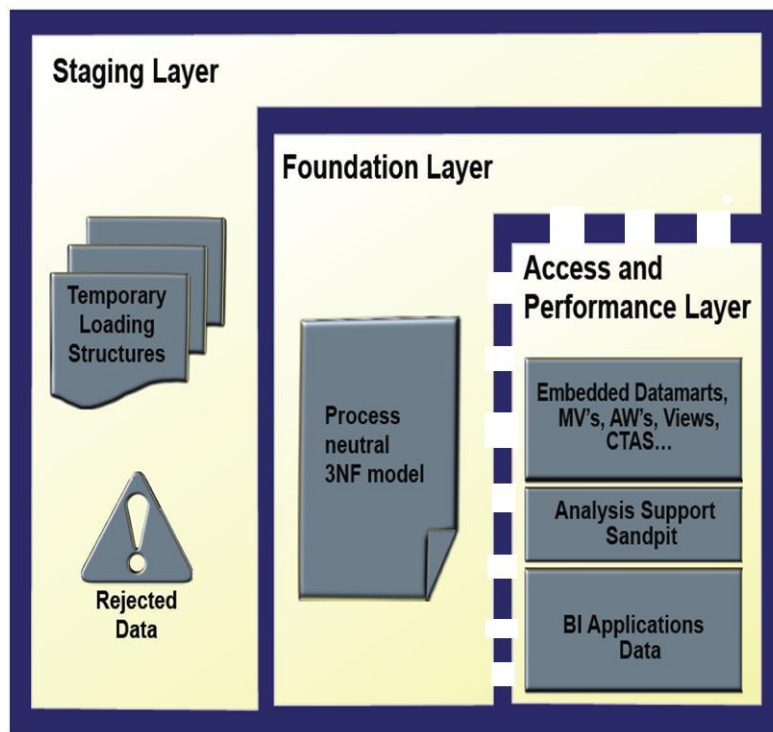
Note: In some implementations this layer is not necessary, because all data transformation processing is done "on the fly" as data is extracted from the source system before it is inserted directly into the foundation layer.

- **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 a large number of 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 Communications Data Model Warehouse

Oracle Communications Data Model warehouse environment also consists of three layers. However, as indicated by the dotted line in [Figure 2-1, "Layers of an Oracle Communications Data Model Warehouse"](#) on page 2-2, in the Oracle Communications Data Model the definitions of the foundation and access layers are combined in a single schema.

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



The layers in the Oracle Communications Data Model warehouse are:

- **Staging layer.** As in a "traditional" data warehouse environment, an Oracle Communications 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 Communications Data Model.

Note: If you are using an application adapter for Oracle Communications Data Model to populate the foundation layer of Oracle Communications Data Model, then that adapter defines and populates an Oracle Communications Data Model staging layer.

- **Foundation and Access layers.** The physical objects for these layers are defined in a single schema, the `ocdm_sys` schema:
 - **Foundation layer.** The foundation layer of the Oracle Communications 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 `ocdm_sys` schema (that is, the tables that have the `DWR_`, `DWL_`, `DWC_` prefixes).
 - **Access layer.** The Access layer of Oracle Communications 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 `_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 models themselves are defined in the `ocdm_mining` schema.

See: *Oracle Communications Data Model Reference* for detailed information on the `ocdm_sys` schema.

Customizing the Oracle Communications Data Model Physical Model

The starting point for the Oracle Communications 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, views).

To customize the default physical model of the Oracle Communications Data Model take the following steps:

1. Answers the questions outlined in "[Questions to Answer Before You Customize the Physical Model](#)" on page 2-4.
2. Familiarize yourself with the characteristics of the logical and physical model of Oracle Communications Data Model as outlined in "[Characteristics of the Default Physical Model](#)" on page 2-1 and presented in detail in *Oracle Communications Data Model Reference*.
3. Modify the foundation level of your physical model of Oracle Communications Data Model, as needed. See "[Common Change Scenarios](#)" on page 2-6 for a discussion of when customization might be necessary.

When defining physical structures:

- Keep the foundation layer in 3NF form.

- Use the information presented in "[General Recommendations When Designing Physical Structures](#)" on page 2-7 to guide you when designing the physical objects.
 - Follow the conventions used when creating the default physical model of Oracle Communications Data Model as outlined in "[Conventions When Customizing the Physical Model](#)" on page 2-4.
- Tip:** Package the changes you make to the physical data model as a patch to the `ocdm_sys` schema.
4. Modify the access layer of your physical model of Oracle Communications Data Model as discussed in [Chapter 3, "Access Layer Customization"](#).

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](#)" on page 2-6 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 Communications 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?

Conventions When Customizing the Physical Model

When developing the physical model for Oracle Communications Data Model, the conventions outlined below 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 `ocdm_sys` schema delivered with Oracle Communications Data Model uses the prefixes and suffixes shown in the following table to identify object types.

Prefix or Suffix	Used for Name of These Objects
CB\$	Materialized view of an OLAP cube. This materialized view is automatically created by the OLAP server. Note: Do not report or query against this object. Instead access the corresponding <code>_VIEW</code> object.
DMV_	Materialized view created for performance reasons (that is, <i>not</i> an aggregate table or an OLAP cube).
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.
DWV_	Relational view of time dimension
DWR_	Reference data tables.
_VIEW	A relational view of an OLAP cube, dimension, or hierarchy.

Use these prefixes and suffixes for any new tables, views, and cubes that you define.

See: *Oracle Communications Data Model Reference* for detailed information about the objects in the default Oracle Communications Data Model.

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'.
- Numeric value in different situations.
- Character string length in different situations.
- Coded value definition such as key or description.

Foundation Layer Customization

The first step in customizing the physical model of Oracle Communications Data Model is customizing the foundation layer of the physical data model. Since, as mentioned in "[Layers in the Default Oracle Communications Data Model Warehouse](#)" on page 2-2, the foundation layer of the physical model mirrors the 3NF logical model

of Oracle Communications 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 Communications 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. "[Common Change Scenarios](#)" on page 2-6 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 "[General Recommendations When Designing Physical Structures](#)" on page 2-7 and "[Conventions When Customizing the Physical Model](#)" on page 2-4.
- Remember that changes to the foundation layer objects can also impact the access layer objects.

Note: Approach any attempt to change the Oracle Communications Data Model with caution. The foundation layer of the physical model of the Oracle Communications Data Model has (at its core) a set of generic structures that allow it to be flexible and extensible. Before making extensive additions, deletions, or changes, ensure that you understand the full range of capabilities of Oracle Communications Data Model and that you cannot handle your requirements using the default objects in the foundation layer.

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 Communications Data Model, add new tables and columns.

Carefully study the default foundation layer of the physical data model of Oracle Communications 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 Communications Data Model Development Team for possible inclusion in future releases of Oracle Communications 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 Communications 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 Communications Data Model. Also identify the impact on any applications based on the new design.

General Recommendations When Designing Physical Structures

The `ocdm_sys` schema delivered with Oracle Communications 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 Communications Data Model:

- [Tablespaces in the Oracle Communications Data Model](#)
- [Data Compression in the Oracle Communications Data Model](#)
- [Tables for Supertype and Subtype Entities in Oracle Communications Data Model](#)
- [Surrogate Keys in the Physical Model](#)
- [Integrity Constraints in Oracle Communications Data Model](#)
- [Indexes and Partitioned Indexes in the Oracle Communications Data Model](#)
- [Partitioned Tables in the Oracle Communications Data Model](#)
- [Parallel Execution in the Oracle Communications Data Model](#)

Tablespaces in the Oracle Communications 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 Communications Data Model warehouse.

Changing the Tablespace and Partitions Used by Tables

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

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), 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 Communications 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;
```

Data Compression in the Oracle Communications 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 better scale-up performance for read-only operations. 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.

Recommendations: Data Compression

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 a large number of distinct values (1,000 to 10,000).

See also: ["Types of Data Compression Available"](#) on page 2-8 and ["Compressing Materialized Views"](#) on page 3-25.

Types of Data Compression Available

Oracle Database offers the following types of compression:

- [Basic or Standard Compression](#)
- [OLTP Compression](#)
- [Hybrid Columnar Compression \(HCC\)](#)

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 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 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;
```

Example 2–1 Creating a Compressed Table for Oracle Communications Data Model

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 ;
```

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

See: *Oracle Database Administrator's Guide* for more information on OLTP table compression features.

Oracle by Example: For more information on 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 by following the instructions in "[Oracle Technology Network](#)" on page xii; 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 conventional DML is issued against a table with HCC, the necessary data is uncompressed to do the modification and then written back to 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 OLTP 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.

See also: The discussion on HCC in *Oracle Database Concepts*.

Tables for Supertype and Subtype Entities in Oracle Communications 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.)

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

Integrity Constraints in Oracle Communications Data Model

Integrity constraints are used to enforce business rules associated with your database and to prevent having 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.

Indexes and Partitioned Indexes in the Oracle Communications 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 Communications 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 non-partitioning key, or a performance requirement).

See also: ["Partitioned Tables in the Oracle Communications Data Model"](#) on page 2-12, ["Choosing Indexes for Materialized Views"](#) on page 3-23, ["Choosing a Cube Partitioning Strategy for Oracle Communications Data Model"](#) on page 3-18, and ["Partitioning and Materialized Views"](#) on page 3-23.

Partitioned Tables in the Oracle Communications 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 non-partitioned 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 Communications Data Model for Manageability](#)
- [Partitioning the Oracle Communications Data Model for Easier Data Access](#)
- [Partitioning the Oracle Communications Data Model for Join Performance](#)

Oracle by Example: To understand the various partitioning techniques in Oracle Database, see the "Manipulating Partitions in Oracle Database 11g" OBE tutorial.

To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in ["Oracle Technology Network"](#) on page xii; and, then, search for the tutorial by name.

See also: ["Indexes and Partitioned Indexes in the Oracle Communications Data Model"](#) on page 2-11, ["Choosing a Cube Partitioning Strategy for Oracle Communications Data Model"](#) on page 3-18, and ["Partitioning and Materialized Views"](#) on page 3-23.

Partitioning the Oracle Communications 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 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.

Partitioning the Oracle Communications Data Model for Easier Data Access

Range partitioning also helps ensure that only the necessary data to answer a query is scanned. Assuming that the business users predominately accesses 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.

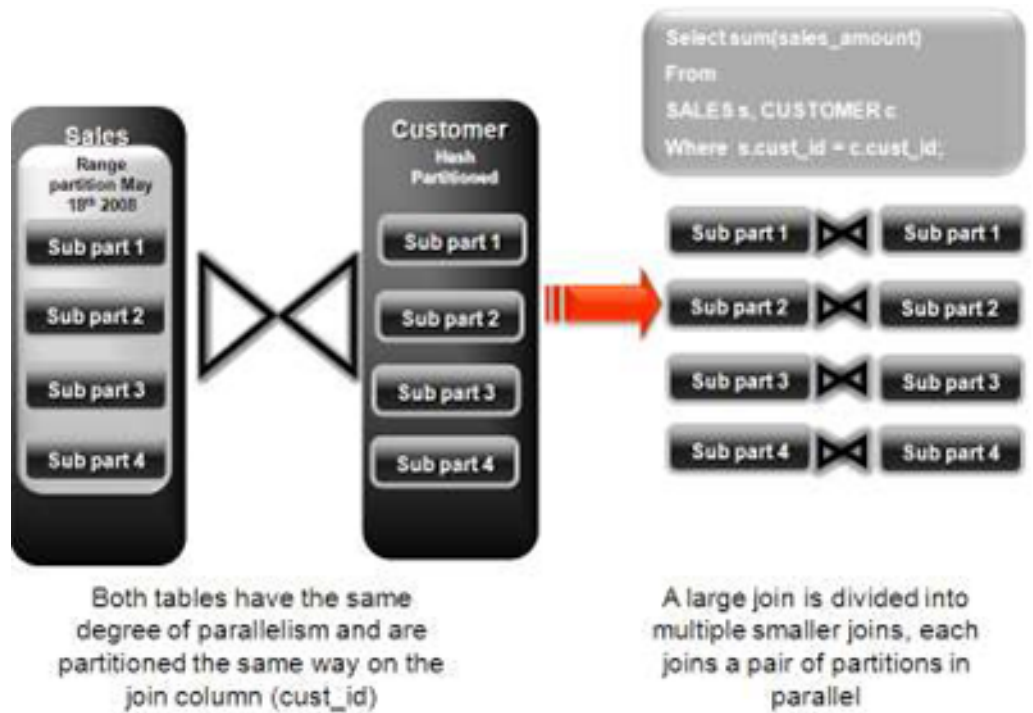
Partitioning the Oracle Communications Data Model for Join Performance

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.

[Figure 2–2](#) shows how 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

In order 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 $2 \times \# \text{ of CPUs}$ rounded to up to the nearest power of 2.

If your system has 12 CPUs, then 32 would 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 would 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.

Parallel Execution in the Oracle Communications 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 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 you would see for this join.

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

Partition Hash All above the join &
single PQ set indicate partition-wise join

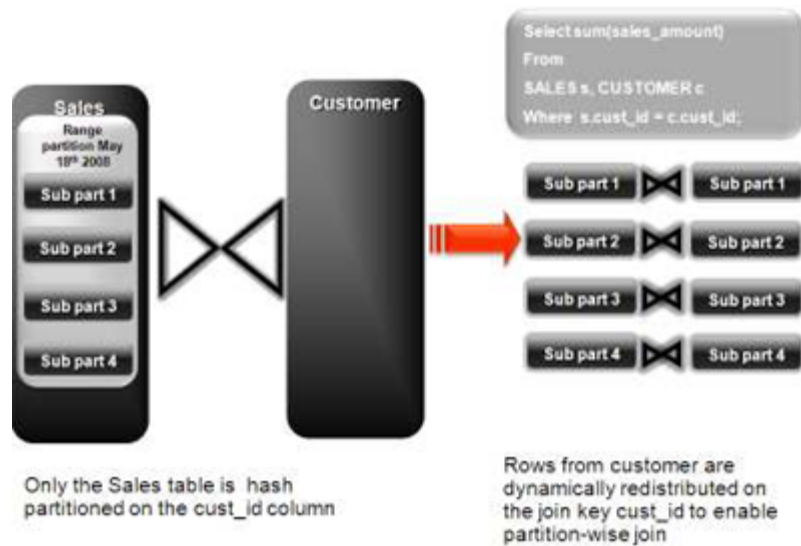
ID	Operation	Name	Pstart	Pstop	TQ	PQ Distrib
0	SELECT STATEMENT					
1	PX COORDINATOR					
2	PX SEND QC (RANDOM)	.TQ10001			Q1,01	QC (RAND)
3	SORT GROUP BY				Q1,01	
4	PX RECEIVE				Q1,01	
5	PX SEND HASH	.TQ10000			Q1,00	HASH
6	SORT GROUP BY				Q1,00	
7	PX PARTITION HASH ALL		1	128	Q1,00	
8	HASH JOIN				Q1,00	
9	TABLE ACCESS FULL	Customers	1	128	Q1,00	
10	TABLE ACCESS FULL	Sales	1	128	Q1,00	

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 parallel servers, each parallel server is given one pair of partitions to join, when the parallel server completes that join, it requests another pair of partitions to join. This process repeats until all pairs have been 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 could pick 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 illustrates a partial partition-wise join. It uses the same example as in Figure 2–3, 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.

Figure 2–4 Partial Partition-Wise Join

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;
```

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 non-partitioned 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.

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;
```

Access Layer Customization

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

- [Introduction to Customizing the Access Layer of Oracle Communications Data Model](#)
- [Derived Tables in the Oracle Communications Data Model](#)
- [Aggregate Tables in the Oracle Communications Data Model](#)
- [Dimensional Components in the Oracle Communications Data Model](#)
- [Materialized Views in the Oracle Communications Data Model](#)

See also: [Chapter 2, "Physical Model Customization"](#)

Introduction to Customizing the Access Layer of Oracle Communications Data Model

The access layer of Oracle Communications 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 `ocdm_sys` schema include: derived and aggregate tables, OLAP cubes, and materialized views. This layer also contains data mining models. The results of these models are stored in derived tables. The models themselves are defined in the `ocdm_mining` schema.

When designing and customizing access layer objects:

- Follow the general guidelines for customizing physical objects given in "[General Recommendations When Designing Physical Structures](#)" on page 2-7.
- Design the access layer objects to support the business intelligence reports and queries that your site makes. See [Chapter 5, "Report and Query Customization."](#)

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

- [Derived Tables in the Oracle Communications Data Model](#)
- [Aggregate Tables in the Oracle Communications Data Model](#)
- [Dimensional Components in the Oracle Communications Data Model](#)
- [Materialized Views in the Oracle Communications Data Model](#)

Derived Tables in the Oracle Communications Data Model

Derived tables are tables that have as values the result of a non-aggregate calculation against the data in the foundation layer tables. Derived tables have a `DWD_` prefix.

There are two main types of derived tables in the default Oracle Communications Data Model and the way you customize these tables varies by type:

- Tables that hold the results of a calculation such as the `DWD_BER_FER_ERR_RATIO_DAY` table that contains values that are the daily BER (Bit Error Rate) and FER (Frame Error Rate) statistics about the network elements. For information on customizing these tables, see ["Creating New Derived Tables for Calculated Data"](#) on page 3-2.
- Result tables for the data mining models (for example, `DWD_CUST_MNNG`). For information on customizing data mining models, see ["Customizing Oracle Communications Data Model Data Mining Models"](#) on page 3-2.

See: The Derived Tables topic in *Oracle Communications Data Model Reference* for a list of all of the derived tables in the default Oracle Communications Data Model. For a list of only those derived tables that are results tables for the data mining models, see the chapter on Data Mining Models in *Oracle Communications Data Model Reference*.

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. When designing these tables, name the tables following the convention of using the `DWD_` prefix for derived tables.

Customizing Oracle Communications Data Model Data Mining Models

Some derived (`DWD_`) tables in the default `ocdm_sys` schema are the results of data mining models defined in the default Oracle Communications Data Model. Those models are defined in the default `ocdm_mining` schema that also comes with Oracle Communications Data Model.

All Oracle Communications Data Model mining models use materialized views defined in the `ocdm_mining` schema 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.

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

- Create a new model as discussed in ["Creating a New Data Mining Model for Oracle Communications Data Model"](#) on page 3-2.
- Modify an existing model as discussed in ["Modifying Oracle Communications Data Model Data Mining Models"](#) on page 3-3.

See also: ["Tutorial: Customizing the Churn Prediction Model"](#) on page 3-4.

Creating a New Data Mining Model for Oracle Communications Data Model

To write a new data mining model:

1. Ensure that the `ocdm_mining` schema includes a definition for a materialized view that you can use as input to the model. Define a new materialized view, if necessary.
2. Create the model as you would any data mining model. Follow the instructions given in *Oracle Data Mining Concepts*. Add the model to the `ocdm_mining` schema.
3. Add any physical tables needed by the model into the `ocdm_sys` schema. Follow the naming conventions outlined in ["Conventions When Customizing the Physical Model"](#) on page 2-4 and use a `DWD_` prefix for results tables.
4. In the `ocdm_mining` schema, grant `SELECT` privileges to the results tables created in Step 3.
5. Modify the intra-ETL to support the use of the data mining model.

Modifying Oracle Communications Data Model 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.
2. Train the model again by calling Oracle Communications Data Model mining package.
3. Ensure that the model reflects the new definition (for example, that a new column has been added).

Example 3-1 Adding a New Column to a Mining Model in Oracle Communications Data Model

To add a new column to `create_churn_svm_model`, take the following steps:

1. Add the new column the following materialized views that are used as input to `create_churn_svm_model`.

```
DMV_CUST_CHRN_SRC_SRC
DMV_CUST_CHRN_SRC_PRD
DMV_CUST_CHRN_SRC_TST
DMV_CUST_CHRN_APPLY_ALL
```

2. Train the model by issuing the following statement.

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

3. Execute 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(Oracle Communications Data Model
  SELECT ATTRIBUTE_SET FROM TABLE(DBMS_DATA_MINING.GET_MODEL_DETAILS_
SVM('OCDM_CHURN_SVM'))
WHERE CLASS='1' );
```

See also: ["Refreshing the Data in an Oracle Communications Data Model Warehouse"](#) on page 4-14, ["Troubleshooting Data Mining Model Creation"](#) on page 4-19, and [Appendix B, "Working with User Privileges in Oracle Communications Data Model."](#)

Tutorial: Customizing the Churn Prediction Model

After you have populated Oracle Communications Data Model foundation layer and the derived tables, you can leverage the prebuilt Oracle Communications Data Model mining model for some more advanced analysis and predictions.

This tutorial shows you how to predict the customers who will terminate the service in next 3 months (churners) based on the populated Oracle Communications Data Model warehouse. Using prebuilt Oracle Communications Data Model Mining models, you can easily and very quickly see the prediction result of your customers, without having to go through all of the data preparation, training, testing and applying process that you must perform in a traditional from-scratch mining project.

See: *Oracle Data Mining Concepts* for more information about the Oracle Database mining training and scoring (applying) process.

After the initially generating a mining model, as time goes by, the customer information and their behavior change. Consequently, you must refresh the previous trained mining models based on the latest customer and usage data. You can follow the process in this tutorial to refresh the mining models to acquire predictions on latest customer information.

This tutorial shows you how to generate the Churn Prediction model through Oracle Communications Data Model Mining APIs. To use different parameters in the training process, or customize the model in more advanced fashion, you can also use Oracle Data Miner to do the same work.

This tutorial consists of the following:

- [Tutorial Prerequisites](#)
- [Preparing Your Environment](#)
- [Generating the Churn Prediction Model](#)
- [Checking the Result](#)

Tutorial Prerequisites Before starting this tutorial:

1. Review the Oracle by Example (OBE) tutorial "Using Oracle Data Miner 11g Release 2." To access this tutorial, open the Oracle Learning Library in your browser by following the instructions in "[Oracle Technology Network](#)" on page xii; and, then, search for the tutorial by name.
2. Install Oracle Communications Data Model.
3. Populate the base, reference and lookup tables.
4. Execute the intra-ETL.

Ensure that at least the following tables contain valid data:

```
DWD_ACCT_STTSTC
DWD_VAS_SBRP_QCK_SUMM
DWR_BSNS_MO
DWR_CUST
DWR_HH
DWR_JB
```

Note: If you have not populated the real customer data, and only want to learn the Oracle Communications Data Model mining model, you can use the sample data by taking the following steps:

1. Ensure that during the install, you generated the calendar data covering range of 2007~2008. For example, the parameters of starting from "20070101" for 5 years satisfy this condition
2. Download the sample data(ocdm.4.mining.dmp.gz) and import that data into your new ocdm_sys schema.

Preparing Your Environment This tutorial requires a valid, populated Oracle Communications Data Model warehouse.

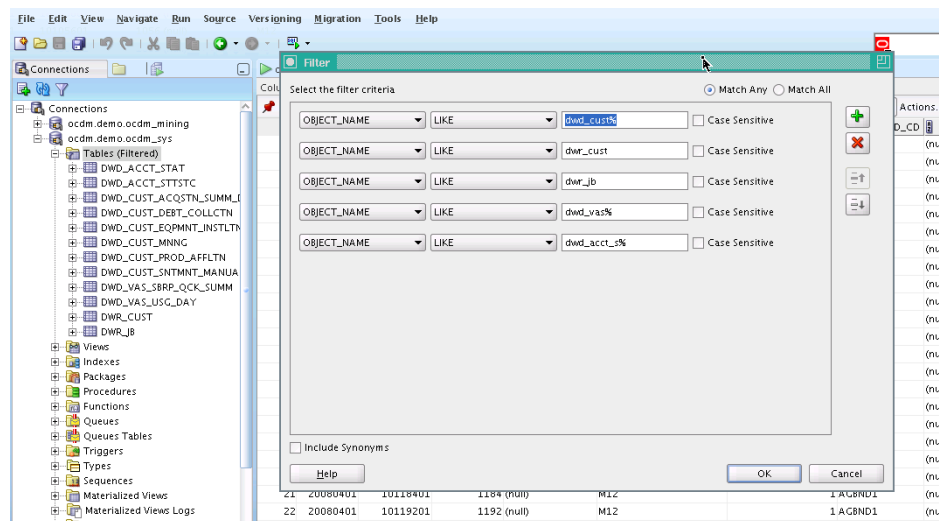
Oracle by Example: For more information about using SQL Developer, refer to tutorial "Getting Started with Oracle SQL Developer 3.0". To access this tutorial, open the Oracle Learning Library in your browser by following the instructions in "[Oracle Technology Network](#)" on page xii; and, then, search for the tutorial by name.

To prepare the environment, take the following steps:

1. In SQL Developer, connect to the ocdm_sys and the ocdm_mining schemas.

Tip: SQL Developer can be found on any Oracle Database Installation under \$ORACLE_HOME/sqldeveloper.

2. After you connect to the ocdm_sys schema, you can see all the tables in that schema. You can narrow down the list by right clicking the "table" and then applying filters:



3. (Optional) As mentioned in the "[Preparing Your Environment](#)" on page 3-5, if you have not populated those tables with your own data, you can try with some sample data. After you download the sample data, follow those steps to import the data:
 - a. Grant dba to ocdm_sys by issuing the following statement:

```
grant dba to ocdm_sys.
```

b. Disable all foreign keys on those tables used by the tutorial:

```
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_CS5_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_POT_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_DLCS_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_SJ_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_SEG_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_DLCT_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_LAN_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_AL1_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_PT_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_BLC_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_DLMS_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_DLJ_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_DLG_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_HOU1_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_DLE_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_DLN_FK".
ALTER TABLE "DWR_CUST" DISABLE CONSTRAINT "CI_OI_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_CI_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_BM_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_RSCR_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_AL1_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_DAB_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_DLCRB_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_DLCC_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_DAS_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_PM_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_DOB_U_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_CR1_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_LAB_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_ACCT_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_CHR_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_DLAON_FK".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_DLCT_FK1".
ALTER TABLE "DWD_ACCT_STTSTC" DISABLE CONSTRAINT "ACS_GEOC_FK1".
ALTER TABLE "DWD_VAS_SBRP_QCK_SUMM" DISABLE CONSTRAINT "VS_AM_FK".
ALTER TABLE "DWD_VAS_SBRP_QCK_SUMM" DISABLE CONSTRAINT "VS_CI_FK".
ALTER TABLE "DWD_VAS_SBRP_QCK_SUMM" DISABLE CONSTRAINT "VS_BM_FK".
ALTER TABLE "DWD_VAS_SBRP_QCK_SUMM" DISABLE CONSTRAINT "VS_DLCT_FK1".
```

c. Import the sample dump into ocdm_sys schema by issuing the following statement.

```
imp ocdm_sys/ocdm_sys@tnsname file=ocdm.4.mining.dmp full=y ignore=y
```

4. Review tables to ensure they contain valid data, either from your own customer data, or from the sample data.

	MO_KEY	ACCT_KEY	CUST_KEY	CHRNRSN_CD	ACE_ON_NET_BND_CD	CRDT_CTGRY_KEY	AGE
1	20080401	10105001	1050	(null)	M12		1.AGEND1
2	20080401	10105601	1056	(null)	M12		1.AGEND1
3	20080401	10106301	1063	(null)	M12		4.AGEND2
4	20080401	10107201	1072	(null)	M12		1.AGEND1
5	20080401	10109001	1090	(null)	M12		1.AGEND2
6	20080401	10109201	1092	(null)	M12		1.AGEND2
7	20080401	10109801	1098	(null)	M12		1.AGEND2
8	20080401	10110201	1102	(null)	M12		1.AGEND2
9	20080401	10110401	1104	(null)	M12		1.AGEND1
10	20080401	10111101	1111	(null)	M12		4.AGEND2
11	20080401	10111501	1115	(null)	M12		4.AGEND1
12	20080401	10111901	1119	(null)	M12		2.AGEND1
13	20080401	10112601	1126	(null)	M12		1.AGEND1
14	20080401	10113501	1135	(null)	M12		4.AGEND2
15	20080401	10113701	1137	(null)	M12		2.AGEND1
16	20080401	10114001	1140	(null)	M12		1.AGEND1
17	20080401	10114101	1141	(null)	M12		3.AGEND2
18	20080401	10114401	1144	(null)	M12		1.AGEND2
19	20080401	10116001	1160	(null)	M12		1.AGEND1
20	20080401	10117701	1177	(null)	M12		4.AGEND2
21	20080401	10118401	1184	(null)	M12		1.AGEND1
22	20080401	10119201	1192	(null)	M12		1.AGEND1

5. Review the DWR_CUST table.

	CUST_KEY	CUST_CD	PRTY_KEY	PRTY_TYP_CD	BSNS_LEGAL
1	3744	CUST-3744	53945	CUST	(null)
2	2079	CUST-2079	52280	CUST	(null)
3	10522	CUST-10522	60723	CUST	(null)
4	806	CUST-806	51007	CUST	(null)
5	9080	CUST-9080	59281	CUST	(null)
6	3125	CUST-3125	53326	CUST	(null)
7	3137	CUST-3137	53338	CUST	(null)
8	6623	CUST-6623	56824	CUST	(null)
9	9726	CUST-9726	59927	CUST	(null)
10	9041	CUST-9041	59242	CUST	(null)
11	2586	CUST-2586	52787	CUST	(null)
12	6687	CUST-6687	56888	CUST	(null)
13	6059	CUST-6059	56260	CUST	(null)
14	537	CUST-537	50738	CUST	(null)
15	8984	CUST-8984	59185	CUST	(null)
16	8969	CUST-8969	59170	CUST	(null)
17	6161	CUST-6161	56362	CUST	(null)
18	4562	CUST-4562	54763	CUST	(null)
19	9188	CUST-9188	59389	CUST	(null)
20	708	CUST-708	50909	CUST	(null)
21	8976	CUST-8976	59177	CUST	(null)

6. Click each of the following tables to ensure that the table is properly populated:

- DWD_ACCT_STTSTC
- DWD_VAS_SBRP_QCK_SUMM
- DWR_BSNS_MO
- DWR_CUST
- DWR_HH
- DWR_JB

7. Check that the DWD_CUST_MNNG and DWD_CUST_PROD_AFFLTN results tables are empty before running the model generation function as described in "Generating the Churn Prediction Model" on page 3-7.

Generating the Churn Prediction Model This tutorial uses two procedures from Oracle Communications Data Model Mining APIs:

- pkg_ocdm_mining.refresh_mining_source that populates some mining source tables from the ocdm_sys schema tables.
- pkg_ocdm_mining.create_churn_svm_model that trains the model, and generates the result of prediction.

Take the following steps to use the procedures:

1. Refresh the Oracle Communications Data Model mining source tables by executing the following SQL statements.

```
select count (*) from dmvcust_chrn_src_all;
exec pkg_ocdm_mining.refresh_mining_source;
select count (*) from dmvcust_chrn_src_all;
```

These statements display the number of tables before refreshing.

Refresh the tables and reexecute the statements to display the number of tables after refreshing.

2. Generate the Churn Prediction model by executing the following SQL statements.

```
select count(*) from dwd_cust_mnng;
select count(*) from dwd_chrn_svm_factor;
exec pkg_ocdm_mining.create_churn_svm_model;
select count(*) from dwd_cust_mnng;
```

These statements:

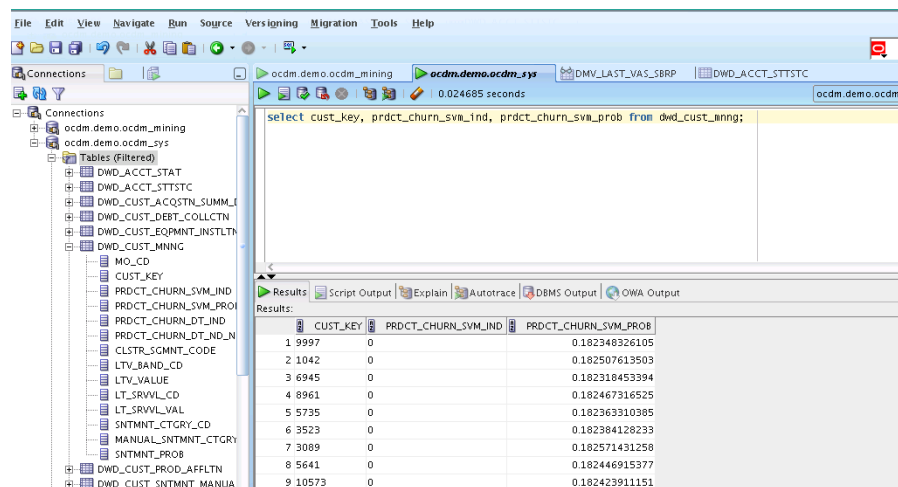
- Show the record counts before generation.
- Generate the mode.
- Show the count after generation.

Note: This tutorial does not refresh all models. It only refreshes one churn prediction model. To refresh all of the default mining models based on latest customer data, follow the instructions in ["Refreshing the Data in an Oracle Communications Data Model Warehouse"](#) on page 4-14.

Checking the Result After you have refreshed, trained, and generated the model, check the DWD_CUST_MNNG results table in ocdm_sys schema by taking the following steps:

1. Issue the following query.

```
Select cust_key, prdct_chrn_svm_ind, prdct_chrn_svm_prob From dwd_cust_mnng
where rownum < 10;
```



2. For each customer identified by CUST_KEY, the PRDCT_CHRN_SVM_IND column gives a Boolean prediction of whether a customer will churn in next 3 months.

Zero (0) stands for non-churner, while one (1) stands for churner. The PRDCT_CHURN_SVM_PROB column provides a more detailed probability (0~1) of how likely a customer is going to churn.

- (Optional) If you have also installed the Oracle Communications Data Model sample reports in Oracle Business Intelligence Suite Enterprise Edition 11g, you can also view the results as an Oracle Business Intelligence Suite Enterprise Edition report.

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

The screenshot shows a report titled "9-Churn Prediction (by SVM)-Cross tab Rpt" in the Oracle Business Intelligence Suite Enterprise Edition interface. The report displays a table with columns for ARPU Band Name, Customer ID, Customer Name, Phone Number, Contract ARPU, Actual ARPU, and Debt Value. The data is filtered for the ARPU1000 BAND.

ARPU Band Name	Customer ID	Customer Name	Phone Number	Contract ARPU	Actual ARPU	Debt Value
ARPU1000 BAND	CUST-7200	Heidi Newark	9985007202	\$1,476.00	\$812.90	\$147.60
	CUST-7203	Isaac Yarnus	9985007205	\$158.00	\$164.32	\$213.20
	CUST-7206	Terrie Washington	9985007208	\$168.00	\$172.32	\$213.20
	CUST-7207	Wynnee Zimmer	9985007209	\$164.00	\$171.32	\$213.20
	CUST-7212	Wylie Stockman	9985007214	\$152.00	\$162.10	\$364.08
	CUST-7214	Gideon Roche	9985007216	\$148.00	\$151.20	\$213.20
	CUST-7226	Gale Wright	9985007228	\$128.00	\$132.87	\$213.20
	CUST-7227	Hamilton Alexander	9985007229	\$136.00	\$138.65	\$147.60
	CUST-7228	Page Rudder	9985007230	\$1,148.00	\$564.52	\$213.20
	CUST-7230	Sarah Moy	9985007232	\$132.00	\$136.25	\$114.80

In the Oracle Business Intelligence Suite Enterprise Edition report, the customer revenue information such as the contract ARPU, debt value is shown combined with the prediction results.

Aggregate Tables in the Oracle Communications Data Model

Aggregate tables are tables that aggregate or "roll up" the data to one level higher than a base or derived table. The aggregate tables in the default Oracle Communications 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 Communications Data Model defines several aggregate tables. For example, the DWA_BER_FER_ERR_RATIO_MONTH table aggregates the values of the DWD_BER_FER_ERR_RATIO_DAY table to the month level.

See: The "Aggregate Tables" topic in *Oracle Communications Data Model Reference* for a list of the aggregate tables in the default Oracle Communications Data Model.

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.

See also: ["Materialized Views in the Oracle Communications Data Model"](#) on page 3-20 and ["Defining New Oracle OLAP Cubes for Oracle Communications Data Model"](#) on page 3-17.

- Name the tables following the conventions outlined in ["General Naming Conventions for Physical Objects"](#) on page 2-4 and use a `DWA_` prefix.

Dimensional Components in the Oracle Communications 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 Communications Data Model. The foundation layer of the Oracle Communications Data Model is a 3NF model. The default Oracle Communications 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 Communications Data Model:

- [Characteristics of a Dimensional Model](#)
- [Characteristics of Relational Star and Snowflake Tables](#)
- [Characteristics of the OLAP Dimensional Model](#)
- [Characteristics of the OLAP Cubes in Oracle Communications Data Model](#)
- [Defining New Oracle OLAP Cubes for Oracle Communications Data Model](#)
- [Changing an Oracle OLAP Cube in Oracle Communications Data Model](#)
- [Creating a Forecast Cube for Oracle Communications Data Model](#)
- [Choosing a Cube Partitioning Strategy for Oracle Communications Data Model](#)
- [Choosing a Cube Data Maintenance Method for Oracle Communications Data Model](#)

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 sometimes 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, moving totals, and 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 (rolling up) to higher levels. Drilling down on the Time dimension member 2005 typically navigates you to the quarters Q1 2005 through Q4 2005. In a calendar year hierarchy, drilling down on Q1 2005 would navigate you to the months, January 05 through March 05. These kinds of relationships make it easy for users to navigate 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.** This traditional method of implementing a dimensional model is discussed in "[Characteristics of Relational Star and Snowflake Tables](#)" on page 3-12.

- Oracle OLAP Cubes.** The physical model provided with Oracle Communications Data Model provides a dimensional perspective of the data using Oracle OLAP cubes. This dimensional model is discussed in ["Characteristics of the OLAP Dimensional Model"](#) on page 3-13.

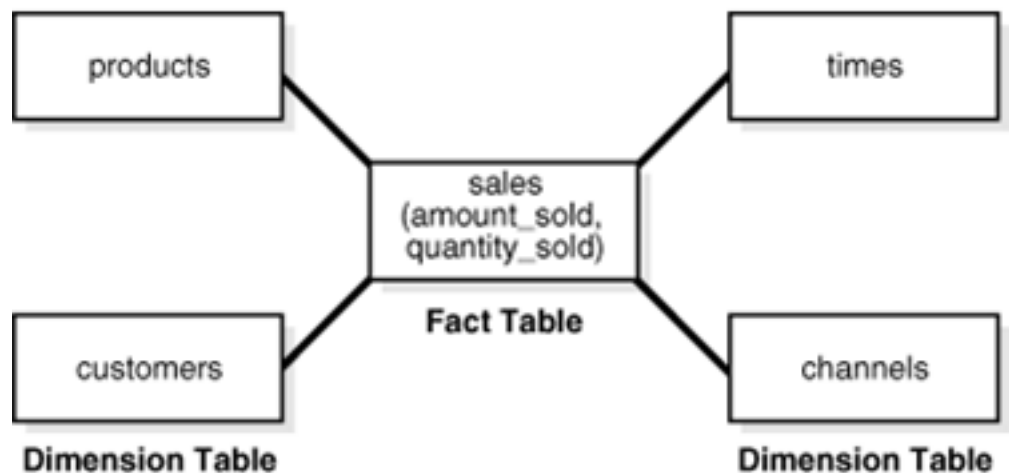
Characteristics of Relational Star and Snowflake Tables

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 borders on a physical model, as drill paths, hierarchy and query profile are 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.

Figure 3–1 Star Schema Diagram



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.

See also: ["Declaring Relational Dimension Tables"](#) on page 3-13 and ["Validating Relational Dimension Tables"](#) on page 3-13.

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.

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.

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 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 as described in ["Oracle OLAP Cube Views"](#) on page 3-14 and [Chapter 5, "Report and Query Customization."](#)

- Oracle OLAP cubes can be enhanced so that they are materialized views as described in ["Cube Materialized Views"](#) on page 3-15.

See also: *Oracle OLAP User's Guide* and ["Characteristics of the OLAP Cubes in Oracle Communications Data Model"](#) on page 3-16.

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.

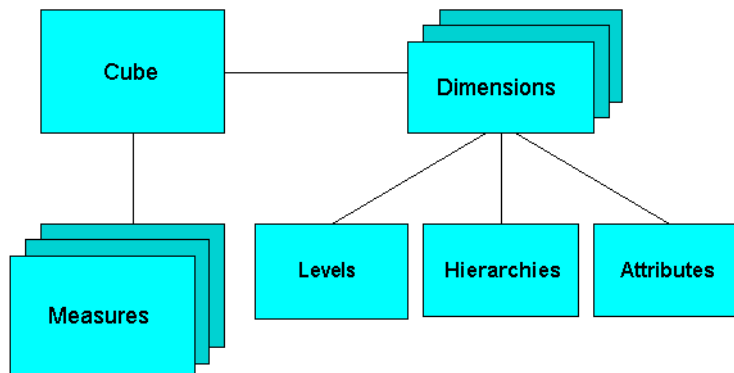
The OLAP engine manipulates dimensional objects in the same way that the SQL engine manipulates relational objects. However, because the OLAP engine is optimized to calculate analytic functions, and dimensional objects are optimized for analysis, analytic and row functions can be calculated much faster 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. The OLAP dimensional objects are described in detail in *Oracle OLAP User's Guide*. [Figure 3-2](#) shows the general relationships among the objects.

Figure 3-2 *Diagram of the OLAP Dimensional Model*



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.

See also: The discussion on querying dimensional objects in *Oracle OLAP User's Guide* and [Chapter 5, "Report and Query Customization."](#)

Cube Materialized Views

Oracle OLAP cubes can be enhanced so that they are materialized views. A cube that has been enhanced in this way is called a cube materialized view and has a CB\$ prefix. Cube materialized views can be incrementally refreshed through the Oracle Database materialized view subsystem, and they can 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 the following 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 have not been 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, 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.

Oracle by Example: For more information on working with OLAP cubes, see the following OBE tutorials:

- "Querying OLAP 11g Cubes"
- "Using Oracle OLAP 11g With Oracle BI Enterprise Edition"

To access the tutorials, open the Oracle Learning Library in your browser by following the instructions in "[Oracle Technology Network](#)" on page xii; and, then, search for the tutorials by name.

See also: *Oracle OLAP User's Guide*

Characteristics of the OLAP Cubes in Oracle Communications Data Model

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

There are OLAP cubes defined in the default `ocdm_sys` schema. These cubes have the general characteristics described in "[Characteristics of the OLAP Dimensional Model](#)" on page 3-13. Specifically, OLAP cubes in the Oracle Communications 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 Communications 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. To enable the cube materialized views, you must follow the steps outlined in "[Adding Materialized View Capabilities](#)" on page 3-16.

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

- [Defining New Oracle OLAP Cubes for Oracle Communications Data Model](#)

- [Changing an Oracle OLAP Cube in Oracle Communications Data Model](#)
- [Creating a Forecast Cube for Oracle Communications Data Model](#)
- [Choosing a Cube Partitioning Strategy for Oracle Communications Data Model](#)
- [Choosing a Cube Data Maintenance Method for Oracle Communications Data Model](#)

Defining New Oracle OLAP Cubes for Oracle Communications Data Model

You can add new OLAP cubes to the `ocdm_sys` schema. For consistency's sake, design and define these new cubes as described in "[Characteristics of the OLAP Cubes in Oracle Communications Data Model](#)" on page 3-16.

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. See "[Aggregate Tables in the Oracle Communications Data Model](#)" on page 3-9 for information on creating new tables.
2. Use the AWM to define new Cubes for a customized version of Oracle Communications Data Model. Follow the instructions given for creating cubes and dimensions in *Oracle OLAP User's Guide*.

Use the information provided in "[Characteristics of the OLAP Dimensional Model](#)" on page 3-13, and 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](#)" on page 3-12, 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 Communications Data Model.

See: *Oracle Communications Data Model Installation Guide* for more information on installing the sample reports and deploying the Oracle Communications 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 as described in "[Adding Materialized View Capabilities](#)" on page 3-16.

See also: *Oracle OLAP User's Guide*, "[Defining New Oracle OLAP Cubes for Oracle Communications Data Model](#)" on page 3-17, and the sample reports in *Oracle Communications Data Model Reference*.

Oracle by Example: For more information on creating OLAP cubes, see the "Building OLAP 11g Cubes" OBE tutorial.

To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "[Oracle Technology Network](#)" on page xii; and, then, search for the tutorial by name.

Changing an Oracle OLAP Cube in Oracle Communications Data Model

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

Since all Oracle Communications 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 *Oracle Communications 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.

Creating a Forecast Cube for Oracle Communications Data Model

To create a forecast cube for Oracle Communications Data Model:

1. Create a cube to contain the results of the forecast as described in "[Defining New Oracle OLAP Cubes for Oracle Communications Data Model](#)" on page 3-17.

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

2. Write an OLAP DML forecasting context program as described in *Oracle OLAP DML Reference*.

Choosing a Cube Partitioning Strategy for Oracle Communications 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 specific 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 calculation 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.

See also: "The discussion on choosing a partition strategy in *Oracle OLAP User's Guide*, "[Indexes and Partitioned Indexes in the Oracle Communications Data Model](#)" on page 2-11, and "[Partitioning and Materialized Views](#)" on page 3-23.

Choosing a Cube Data Maintenance Method for Oracle Communications 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.

See also: *Oracle OLAP User's Guide* and "[Types of Materialized Views and Refresh options](#)" on page 3-21

Oracle by Example: 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 11g Cubes"

To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "[Oracle Technology Network](#)" on page xii; and, then, search for the tutorial by name.

Materialized Views in the Oracle Communications 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 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 Communications Data Model defines many materialized views. In the default `ocdm_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 <i>Oracle Communications Data Model Reference</i> for a list of these objects in the default data model. "Aggregate Tables in the Oracle Communications Data Model" on page 3-9 for more information on customizing these objects,.
CB\$	An OLAP cube enhanced with materialized view capabilities. See: OLAP cube materialized views in <i>Oracle Communications Data Model Reference</i> for a list of these objects in the default data model. "Characteristics of the OLAP Cubes in Oracle Communications Data Model" on page 3-16 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 <code>_VIEW</code> suffix).
DMV_	Materialized views created for performance reasons (that is, <i>not</i> an aggregate table or a cube materialized view). See: <i>Oracle Communications Data Model Reference</i> to identify these objects in the default data model.

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

- [Types of Materialized Views and Refresh options](#)
- [Choosing Indexes for Materialized Views](#)
- [Partitioning and Materialized Views](#)
- [Compressing Materialized Views](#)

Types of Materialized Views and Refresh options

Refresh option vary by the type of materialized view:

- [Refresh Options for Materialized Views with Aggregates](#)
- [Refresh Options for Materialized Views Containing Only Joins](#)
- [Refresh Options for Nested Materialized Views](#)

See: *Oracle OLAP User's Guide* for a discussion of creating materialized views of Oracle OLAP cubes.

Refresh Options for Materialized Views with Aggregates

In data warehouses, materialized views normally contain aggregates. The `_DWA` tables in the default Oracle Communications 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 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.

Example 3–2 Refreshing Oracle Communications Data Model Nested Materialized Views

For example, if you call `DBMS_MVIEW.REFRESH ('SUM_SALES_CUST_TIME', nested => TRUE)`, the `REFRESH` procedure first refreshes the `join_sales_cust_time` materialized view, and then refreshes the `sum_sales_cust_time` materialized view.

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.

See also: ["Indexes and Partitioned Indexes in the Oracle Communications Data Model"](#) on page 2-11

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.

See also: *Oracle Database VLDB and Partitioning Guide*, "[Partitioned Tables in the Oracle Communications Data Model](#)" on page 2-12, "[Indexes and Partitioned Indexes in the Oracle Communications Data Model](#)" on page 2-11, and "[Choosing a Cube Partitioning Strategy for Oracle Communications Data Model](#)" on page 3-18

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.

Compressing Materialized Views

Using data compression for a materialized view brings you an additional 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.

See also: ["Data Compression in the Oracle Communications Data Model"](#) on page 2-8, and ["Aggregate Tables in the Oracle Communications Data Model"](#) on page 3-9.

ETL Implementation and Customization

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

- [The Role of ETL in the Oracle Communications Data Model](#)
- [ETL for the Foundation Layer of an Oracle Communications Data Model Warehouse](#)
- [Customizing Intra-ETL for the Oracle Communications Data Model](#)
- [Performing an Initial Load of an Oracle Communications Data Model Warehouse](#)
- [Refreshing the Data in an Oracle Communications Data Model Warehouse](#)
- [Managing Errors During Oracle Communications Data Model Intra-ETL Execution](#)

The Role of ETL in the Oracle Communications Data Model

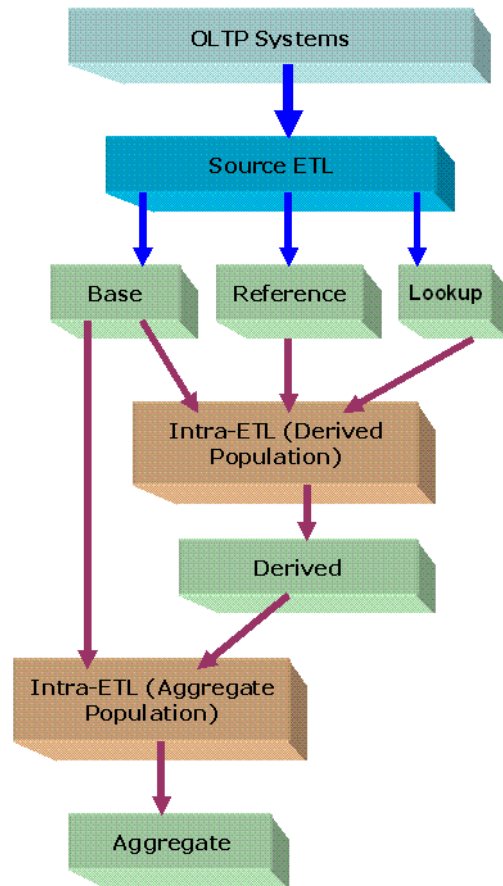
[Figure 2-1, "Layers of an Oracle Communications Data Model Warehouse"](#) on page 2-2 illustrated the three layers in Oracle Communications Data Model warehouse environment: the optional staging layer, the foundation layer, and the access layer. As illustrated by [Figure 4-1](#), you use two types of ETL (extraction, transformation and loading) to populate these layers:

- **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 OLTP system is known as source ETL.

Oracle Communications Data Model does *not* include source-ETL scripts. Unless you are using an application adapter for Oracle Communications Data Model, you must create source-ETL yourself using your understanding of your OLTP system and your customized Oracle Communications Data Model. See ["ETL for the Foundation Layer of an Oracle Communications Data Model Warehouse"](#) on page 4-2 for more information on creating source-ETL.

- **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 Communications Data Model *does* include intra-ETL. You can modify the default intra-ETL to populate a customized access layer from a customized foundation layer. See ["Customizing Intra-ETL for the Oracle Communications Data Model"](#) on page 4-9 for more information on the intra-ETL.

Figure 4–1 ETL Flow Diagram

ETL for the Foundation Layer of an Oracle Communications Data Model Warehouse

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

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

- If an application adapter for Oracle Communications Data Model is available for the system from which you want to populate the foundation layer of an Oracle Communications Data Model warehouse, you can use that adapter to populate the foundation layer. For more information, see ["Using an Application Adapter to Populate the Foundation Layer"](#) on page 4-2.
- Write your own source-ETL scripts using Oracle Warehouse Builder or another ETL tool and then use those scripts to populate the foundation layer. For more information, see ["Writing Your Own Source-ETL"](#) on page 4-3.

Using an Application Adapter to Populate the Foundation Layer

If an Application Adapter for Oracle Communications Data Model is available for the application that populates your OLTP system, you use that adapter to populate the foundation layer of your Oracle Communications Data Model warehouse.

For example, if want to populate the foundation layer of your Oracle Communications Data Model warehouse with data from your Oracle Communications Network Charging and Control system, then you can use the Oracle Communications Network Charging and Control Adapter for Oracle Communications Data Model. See [Appendix C, "Using the NCC Adapter for Oracle Communications Data Model"](#) for more information

Writing Your Own Source-ETL

If you are not using an application adapter for Oracle Communications Data Model, you must write your own source-ETL scripts using Oracle Warehouse Builder or another ETL tool or mapping tool.

See also: *Oracle Warehouse Builder Data Modeling, ETL, and Data Quality Guide.*

Oracle By Example: See the following OBE tutorials for more information on Oracle Warehouse Builder:

- "Setting Up the Oracle Warehouse Builder 11g Release 2 Environment"
- "Improved User Interface, Usability, and Productivity With Oracle Warehouse Builder 11g R2"
- "Using Data Transformation Operators with Source and Target Operators"
- "Working with Pluggable Mappings"
- "Examining Source Data Using Data Profiling with Database 11g Release 2"

To access the tutorials, open the Oracle Learning Library in your browser by following the instructions in "[Oracle Technology Network](#)" on page xii; and, then, search for the tutorials by name.

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

- [Source-ETL Design Considerations](#)
- [ETL Architecture for Oracle Communications Data Model Source-ETL](#)
- [Creating a Source to Target Mapping Document for the Source-ETL](#)
- [Designing a Plan for Rectifying Source-ETL Data Quality Problems](#)
- [Designing Source-ETL Workflow and Jobs Control](#)
- [Designing Source-ETL Exception Handling](#)
- [Writing Source-ETL that Loads Efficiently](#)

Source-ETL Design Considerations

Keep the following points in mind when designing and writing source-ETL for Oracle Communications Data Model:

- You can populate the calendar data by using the calendar population scripts provided with Oracle Communications Data Model and described in *Oracle Communications 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 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*.

ETL Architecture for Oracle Communications Data Model Source-ETL

ETL 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 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.

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

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.

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.

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.

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 regard to the necessary dependencies. General ETL tools, such as Oracle Warehouse Builder, support this kind of workflow, job design, and execution control.

Below are some tips when you design 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.

Designing Source-ETL Exception Handling

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.

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](#)
- [Preparing Raw Data Files for Source-ETL](#)
- [Source-ETL Data Loading Options](#)
- [Parallel Direct Path Load Source-ETL](#)
- [Partition Exchange Load for Oracle Communications Data Model Source-ETL](#)

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 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.

See: *Oracle Database SecureFiles and Large Objects Developer's Guide* for more information on setting up DBFS.

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:

- Delimitate 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.

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.

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;
```

Partition Exchange Load for Oracle Communications 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 Communications 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.

Customizing Intra-ETL for the Oracle Communications Data Model

The Oracle Communications Data Model supports the use of ETL tools such as Oracle Warehouse Builder to define the workflow to execute the intra-ETL process. You can, of course, write your own intra-ETL. However, an intra-ETL component is delivered with Oracle Communications Data Model that is a process flow designed using the Oracle Warehouse Builder Workflow component. This process flow is named `INTRA_ETL_FLW`.

As illustrated by [Figure 4–1, "ETL Flow Diagram"](#) on page 4-2, 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 Oracle Communications Data Model structures. Within this package the dependency of each individual program is implemented and enforced so that each program executes in the proper order.

You can change the original intra-ETL script for your specific requirements. However, perform a complete impact analysis before you make the change. Package the changes as a patch to the original Oracle Communications Data Model intra-ETL mapping.

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.

This sub-process uses the following technologies:

- **MERGE clause.** This statement is used for initial load and for incremental load. It determines if the data to be inserted or updated into target table based on checking key columns with incoming data and provides a much superior performance compared to an `INSERT` and `UPDATE` combination.
- **TABLE function.** This function is used mainly for derived tables (and some source data) in order to avoid needing a separate staging table. The absence of a staging table provides a performance boost in the ETL operation and simplifies database maintenance
- **Pipeline clause.** This clause is used in the `TABLE` function for incrementally returning the result set for further processing as soon as they are created. Therefore, the use of this clause avoids bulk return which makes the complete process much faster than using a normal return.

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.

This subprocess uses the following technologies:

- **Materialized View.** A materialized view is used to hold the aggregation data. Whenever this is refreshed then the modified data get reflected in the corresponding aggregate table and this leads to a significant increase in the query execution performance. Moreover usage of materialized view allows Oracle Communications Data Model to make use of the Oracle Query Rewrite feature for better SQL optimization and hence improved performance.
- **FAST Refresh.** This refresh type is used to refresh the aggregates with only the incremental data (inserted and modified) in base and derived tables after the immediately previous refresh and this incremental refresh leads to much better performance and hence shorter intra-ETL window.

3. MINING_FLW

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

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`. This shell script does not require Oracle Warehouse Builder workflow. For more information about working with this script, see "[Manually Executing the `ocdm_execute_wf.sh` Script](#)" on page 4-13.

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

Performing an Initial Load of an Oracle Communications Data Model Warehouse

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

1. Load the foundation layer of the Oracle Communications Data Model warehouse (that is, the reference, lookup, and base tables) as described in "[Performing an Initial Load of the Foundation Layer](#)" on page 4-11.
2. Load the access layer of the Oracle Communications Data Model warehouse (that is, the derived and aggregate tables, materialized views, OLAP cubes, and data mining models) as described in "[Performing an Initial Load of the Access Layer](#)" on page 4-11.

Performing an Initial Load of the Foundation Layer

The manner in which you perform an initial load of the foundation layer of an Oracle Communications Data Model warehouse (that is, the reference, lookup, and base tables) varies depending on whether or not you are using an application adapter for Oracle Communications Data Model:

- If you are using an application adapter for Oracle Communications Data Model, then you use that adapter to load the foundation layer. For example, you can use the NCC Adapter for Oracle Communications Data Model to populate the foundation layer of an Oracle Communications Data Model warehouse with data from an Oracle Communications Network Charging and Control system as described in "[Initial Loading Using the NCC Adapter for Oracle Communications Data Model](#)" on page C-3.
- If you are not using an application adapter for, then you perform the initial load of the foundation layer using source-ETL that you create. See "[Writing Your Own Source-ETL](#)" on page 4-3 for more information on creating this ETL.

Performing an Initial Load of the Access Layer

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

1. Update the parameters in `DWC_ETL_PARAMETER` control table in the `ocdm_sys` schema 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 Communications Data Model warehouse, specify the values shown in the following table.

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

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

- 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 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 4-1](#).

Table 4-1 Values of Oracle Communications Data Model OLAP ETL Parameters in the `DWC_OLAP_ETL_PARAMETER` Table for Initial Load

Column Name	Value
<code>PROCESS_NAME</code>	'OCDM-INTRA-ETL'
<code>BUILD_METHOD</code>	C which specifies a complete refresh which clears all dimension values before loading.
<code>CUBENAME</code>	One of the following values that specifies the cubes you want to build: <ul style="list-style-type: none"> ▪ <code>ALL</code> specifies a build of the cubes in the Oracle Communications Data Model analytic workspace. ▪ <code>cubename[cubename]...</code> specifies one or more cubesto build.
<code>MAXJOBQUEUES</code>	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 <code>JOB_QUEUE_PROCESSES</code> database initialization parameter.
<code>CALC_FCST</code>	One of the following values depending on whether you want to calculate forecast cubes: <ul style="list-style-type: none"> ▪ <code>Y</code> specifies calculate forecast cubes. ▪ <code>N</code> specifies do not calculate forecast cubes.
<code>NO_FCST_YRS</code>	If the value for the <code>CALC_FCST</code> column is <code>Y</code> , specify a decimal value that specifies how many years forecast data you want to calculate; otherwise, specify <code>NULL</code> .
<code>FCST_MTHD</code>	If the value for the <code>CALC_FCST</code> column is <code>Y</code> , then specify <code>AUTO</code> ; otherwise, specify <code>NULL</code> .
<code>FCST_ST_YR</code>	If the value for the <code>CALC_FCST</code> column is <code>Y</code> , then specify value specified as <code>yyyy</code> which is the "start business year" of a historical period; otherwise, specify <code>NULL</code> .
<code>FCST_END_YR</code>	If the value for the <code>CALC_FCST</code> column is <code>Y</code> , then specify value specified as <code>yyyy</code> which is the "end business year" of a historical period; otherwise, specify <code>NULL</code> .
<code>OTHER1</code>	Specify <code>NULL</code> .
<code>OTHER2</code>	Specify <code>NULL</code> .

- Execute the intra-ETL in one of the ways described in ["Executing the Default Oracle Communications Data Model Intra-ETL"](#) on page 4-13.

See also: ["Refreshing the Data in an Oracle Communications Data Model Warehouse"](#) on page 4-14

Executing the Default Oracle Communications Data Model Intra-ETL

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 from Oracle Warehouse Builder"](#) on page 4-13.
- Without using Oracle Warehouse Builder Workflow as described in ["Manually Executing the ocdm_execute_wf.sh Script"](#) on page 4-13.

In either case, you can execute the intra-ETL explicitly or invoke its execution in some other program or process (for example, the source-ETL process after its successful execution) or through a predefined schedule (for example, using Oracle Job Scheduling feature and so on).

See: ["Monitoring the Execution of the Intra-ETL Process"](#) on page 4-17, ["Recovering an Intra ETL Process"](#) on page 4-18, and ["Troubleshooting Intra-ETL Performance"](#) on page 4-18.

Executing the INTRA_ETL_FLW Workflow from Oracle Warehouse Builder You can execute the INTRA_ETL_FLW process from within Oracle Warehouse Builder after you deploy it.

To deploy the INTRA_ETL_FLW process flow, take the following steps:

1. Confirm that Oracle Warehouse Builder Workflow has been installed as described in *Oracle Communications 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**.
 - If this is a later deployment, set action to **Replace**.

Deploy the process flow.

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

See: *Oracle Warehouse Builder Sources and Targets Guide* for information about Oracle Warehouse Builder.

Manually Executing the ocdm_execute_wf.sh Script Oracle Communications Data Model provides a script that you can use to populate the intra-ETL without using an Oracle Warehouse Builder Workflow. This shell script is named `ocdm_execute_wf.sh`.

The `ocdm_execute_wf.sh` script performs the same function as Oracle Warehouse Builder Workflow INTRA_ETL_FLW. The script 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.

Variable	Description
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 4-17.

Refreshing the Data in an Oracle Communications Data Model Warehouse

["Performing an Initial Load of the Access Layer"](#) on page 4-11 describes how to perform an initial load of an Oracle Communications Data Model data warehouse. After this initial load, you must 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 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 Communications Data Model warehouse, or you can refresh the data sequentially, as follows:

1. [Refreshing the Foundation Layer of Oracle Communications Data Model Warehouse](#)
2. [Refreshing the Access Layer of an Oracle Communications Data Model Warehouse](#)

In either case, you can manage errors during the execution of the intra-ETL as described in ["Managing Errors During Oracle Communications Data Model Intra-ETL Execution"](#) on page 4-17.

Refreshing the Foundation Layer of Oracle Communications Data Model Warehouse

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

- If an application adapter for Oracle Communications Data Model is available for the system from which you want to refresh the foundation layer of an Oracle Communications Data Model warehouse, you can use that adapter to refresh the foundation layer.

For example, you can use the NCC Adapter for Oracle Communications Data Model to refresh the foundation layer of an Oracle Communications Data Model warehouse with data from an Oracle Communications Network Charging and Control system as described in ["Refreshing the Data Using the NCC Adapter for Oracle Communications Data Model"](#) on page C-4.

- 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"](#) on page 4-3.

Refreshing the Access Layer of an Oracle Communications Data Model Warehouse

Refreshing the access layer of an Oracle Communications 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 Communications Data Model Derived and Aggregate Tables](#)
2. [Refreshing Oracle Communications Data Model OLAP Cubes](#)
3. [Refreshing Oracle Communications Data Model Data Mining Models](#)

In either case, you can manage errors during the execution of the intra-ETL as described in ["Managing Errors During Oracle Communications Data Model Intra-ETL Execution"](#) on page 4-17.

Refreshing Oracle Communications Data Model Derived and Aggregate Tables

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

1. Refresh the foundation layer of the Oracle Communications Data Model warehouse (that is, the reference, lookup, and base tables) with OLTP data by executing the source-ETL that you have written.
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
<code>PROCESS_NAME</code>	'OCDM-INTRA-ETL'
<code>FROM_DATE_ETL</code>	The beginning date of the ETL period.
<code>TO_DATE_ETL</code>	The ending date of the ETL period.

See: *Oracle Communications 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 Default Oracle Communications Data Model Intra-ETL"](#) on page 4-13 for more information.

See also: *Oracle Warehouse Builder Sources and Targets Guide*

Refreshing Oracle Communications 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 Communications Data Model data warehouse since the initial load of the OLAP cubes.

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 Default Oracle Communications Data Model Intra-ETL](#)" on page 4-13.

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, refer to *Oracle Communications Data Model Reference*.)

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 "[Refreshing Oracle Communications Data Model Derived and Aggregate Tables](#)" on page 4-15 for more information.
2. Execute the intra-ETL to load the cube data in one of the ways described in "[Executing the Default Oracle Communications Data Model Intra-ETL](#)" on page 4-13.
3. If necessary, to recover from errors during the execution of `OLAP_MAP` take the following steps.
 - a. Change the value of the `BUILD_METHOD` column of the `ocdm_sys.DWC_OLAP_ETL_PARAMETER` table to "C".
 - b. In Oracle Warehouse Builder, rerun the `OLAP_MAP` map.

Refreshing Oracle Communications Data Model 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.

The way you refresh a data mining model varies depending on whether you want to refresh all of the models or only one model:

- To manually refresh *all* mining models, call the following procedure.
`ocdm_mining.PKG_ocdm_mining.REFRESH_MODEL(MONTH_CODE, P_PROCESS_NO)`

This procedure performs the following tasks for each model:

1. Refreshes the mining source materialized views based on the latest data from `ocdm_sys` schema.
 2. Trains each model on the new training data.
 3. Applies each model onto the new apply data set.
- To manually re-create only one mining model, you can call the corresponding procedure. For example, to re-create the churn SVM model, you can call the following procedure.

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

["Tutorial: Customizing the Churn Prediction Model"](#) on page 3-4 provides detailed instructions for refreshing a single data mining model.

See also: ["Troubleshooting Data Mining Model Creation"](#) on page 4-19

Managing Errors During Oracle Communications Data Model 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](#)
- [Recovering an Intra ETL Process](#)
- [Troubleshooting Intra-ETL Performance](#)

Monitoring the Execution of the Intra-ETL Process

Two `ocdm_sys` schema control tables, `DWC_INTRA_ETL_PROCESS` and `DWC_INTRA_ETL_ACTIVITY`, monitor the execution of the intra-ETL process. These tables are documented in *Oracle Communications 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 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'`.

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 that 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.

The `INTRA_ETL_FLW` process identifies 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 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.

Troubleshooting Intra-ETL Performance

To troubleshoot the performance of the intra-ETL:

- Check the execution plan as described in ["Checking the Execution Plan"](#) on page 4-18.
- Monitor parallel DML executions as described in ["Monitoring PARALLEL DML Executions"](#) on page 4-19.
- Check that data mining models were created correctly as described in ["Troubleshooting Data Mining Model Creation"](#) on page 4-19.

Checking the Execution Plan

Use 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 a map:

1. Copy out the main query statement from code viewer.
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 statement to turn on the parallel DML:

```
Alter session enable parallel dml;
```
3. Paste the main query statement into another SQL Developer worksheet and view the execution plan by clicking F6.

Carefully examine the execution plan to make the mapping runs according to a 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 run mapping in parallel mode, you should see "DML statements parallelized" increased by 1 (one) 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 parallelized.

See also: ["Parallel Execution in the Oracle Communications Data Model"](#) on page 2-14

Troubleshooting Data Mining Model Creation

After 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` is all "COMPLETED-SUCCESS". If the `activity_status` is "COMPLETED-ERROR" for a certain step, check the `ERROR_DTL` column, and fix the problem accordingly.

The following examples illustrate how to troubleshoot some common error messages returned in `ERROR_DTL` and `ACTIVITY_NAME` when working with Oracle Communications Data Model:

- [Example 4-2, "Troubleshooting an ORA-20991 Error for Oracle Communications Data Model"](#)
- [Example 4-3, "Troubleshooting the "Message not available ... \[Language=ZHS\]" Error"](#)
- [Example 4-4, "Troubleshooting an ORA-40113 Error for Oracle Communications Data Model"](#)
- [Example 4-5, "Troubleshooting an ORA-40112 Error for Oracle Communications Data Model"](#)
- [Example 4-6, "Troubleshooting an ORG-11130 Error for Oracle Communications Data Model"](#)

Example 4–2 Troubleshooting an ORA-20991 Error for Oracle Communications Data Model

Assume that the returned error is ORA-20991: Message not available ... [Language=ZHS]CURRENT_MONTH_KEY.

This error may happen when there is 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);
```

See *Oracle Communications Data Model Reference* for information on the calendar population utility script.

Example 4–3 Troubleshooting the "Message not available ... [Language=ZHS]" Error

Assume that the returned error is 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').

Example 4–4 Troubleshooting an ORA-40113 Error for Oracle Communications Data Model

Assume that the returned error is 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 ensure 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.

- Execute the following statement to refresh the mining source materialized views in the `ocdm_mining` schema:

```
exec pkg_ocdm_mining.refresh_mining_source;
```

Example 4-5 Troubleshooting an ORA-40112 Error for Oracle Communications Data Model

Assume that the returned error is ORA-40112:insufficient number of valid data rows, for "create_ltv_glmr_model"

For this model, the target column is `ocdm_mining.DMV_CUST_LTV_PRDCT_SRC.TOT_PYMT_RVN`.

To troubleshoot this error:

- Execute the following SQL statement:

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

- 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 PARALLEL DML Executions"](#) on page 4-19.

Example 4-6 Troubleshooting an ORG-11130 Error for Oracle Communications Data Model

Assume that the returned error is ORG-11130:no data found in the collection, for "create_sentiment_svm_model".

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

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

- Issue the following SQL statement.

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

- Check the number of text comments from the customer interaction table `DWB_EVT_PRTY_INTRACN`.
- 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.

Report and Query Customization

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

- [Reporting Approaches in Oracle Communications Data Model](#)
- [Customizing Oracle Communications Data Model Sample Reports](#)
- [Writing Your Own Queries and Reports](#)
- [Optimizing Star Queries](#)
- [Troubleshooting Oracle Communications Data Model Report Performance](#)
- [Writing As Is and As Was Queries](#)
- [Tutorial: Creating a New Oracle Communications Data Model Dashboard](#)
- [Tutorial: Creating a New Oracle Communications Data Model Report](#)

Reporting Approaches in Oracle Communications Data Model

There are two main approaches to creating reports from data in an Oracle Communications Data Model warehouse:

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 as well as DWR_BSNS_MO, DWR_BSNS_QTR, DWR_BSNS_YR tables to query against a DAY level CDR Wireless entity such as DWD_VOI_CALL_DAY. On the other hand, you need to use the higher level snowflakes at Month level and above such as DWR_BSNS_MO, DWR_BSNS_QTR, DWR_BSNS_YR in order to query against the MONTH level CDR Wireless entity such as DWA_VOI_CALL_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. These could be individual tables starting with `DWL_` or views (also named `DWL_`) on `DWL_CODE_MASTER` that break out different code types into separate dimensions.

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 11g With Oracle BI Enterprise Edition". To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "[Oracle Technology Network](#)" on page xii; and, then, search for the tutorials by name.

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

- "[Writing As Is and As Was Queries](#)" on page 5-7
- "[Tutorial: Creating a New Oracle Communications Data Model Dashboard](#)" on page 5-13
- "[Tutorial: Creating a New Oracle Communications Data Model Report](#)" on page 5-16
- The sample reports provided with Oracle Communications Data Model that are documented in *Oracle Communications Data Model Reference*.

Customizing Oracle Communications Data Model Sample Reports

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.

See: *Oracle Communications Data Model Installation Guide* for more information on installing the sample reports and deploying the Oracle Communications 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.

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

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.

Oracle Communications Data Model also supports On Line Analytic processing (OLAP) reporting using 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 also: ["Reporting Approaches in Oracle Communications Data Model"](#) on page 5-1, ["Oracle OLAP Cube Views"](#) on page 3-14, and the discussion on querying dimensional objects in *Oracle OLAP User's Guide*.

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

For example, assume that you want to know the total call minutes for the top ten customers in the San Francisco area for March 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 perform this 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-09','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

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, "Star Transformation"](#) gives the step by step process to use STAR_TRANSFORMATION to optimize a star query.

Example 5-2 Star Transformation

A business question that could be asked against the star schema in [Figure 3-1, "Star Schema Diagram"](#) on page 3-12 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=2008;
```

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 tyear=2008);
```

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.

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.

ID	Operation	Name	Rows	Cost	Phase
0	SELECT STATEMENT		1		
1	SORT GROUP BY NO SORT		3		
2	HASH JOIN		3		
3	TABLE ACCESS FULL	PRODUCTS	2		
4	HASH JOIN		3		
5	TABLE ACCESS FULL	TREES	1		
6	PARTITION RANGE SUBQUERY		44544	1	15
7	TABLE ACCESS BY LOCAL INDEX ROWID	SALES	44544	1	15
8	BITMAP CONVERSION TO ROWS				
9	BITMAP AND				
10	BITMAP MERGE				
11	BITMAP KEY ITERATION				
12	BUFFER SORT				
13	TABLE ACCESS FULL	TREES	1		
14	BITMAP INDEX RANGE SCAN	SALES_TIME_IDX		1	15
15	BITMAP MERGE				
16	BITMAP KEY ITERATION				
17	BUFFER SORT				
18	TABLE ACCESS FULL	CUSTOMERS	1		
19	BITMAP INDEX RANGE SCAN	SALES_CUST_IDX		1	15
20	BITMAP MERGE				
21	BITMAP KEY ITERATION				
22	BUFFER SORT				
23	TABLE ACCESS FULL	PRODUCTS	2		
24	BITMAP INDEX RANGE SCAN	SALES_PROD_IDX		1	15

Troubleshooting Oracle Communications 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 **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 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.

The following examples illustrate how to use these error messages:

- [Example 5-3, "Troubleshooting an Oracle Communications Data Model Report"](#)
- [Example 5-4, "Troubleshooting a Report: A Table Does Not Exist"](#)
- [Example 5-5, "Troubleshooting a Report: When the Database is Not Connected"](#)

Example 5-3 Troubleshooting an Oracle Communications 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.Cell Phone Number,
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.Cell Phone Number`, `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.

Writing As Is and As Was Queries

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

- [Characteristics of an As Is Query](#)
- [Characteristics of an As Was Query](#)
- [Examples: As Is and As Was Queries Against Oracle Communications Data Model](#)

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.

Characteristics of an As Was Query

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

Examples: As Is and As Was Queries Against Oracle Communications Data Model

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

- [Example 5-6, "As Is Query for Tax Collection Split by Marital Status"](#)
- [Example 5-7, "As Was Queries for Tax Collection Split by Marital Status"](#)
- [Example 5-8, "As Is Query for Tax Collection Data Split by County"](#)
- [Example 5-9, "As Was Queries for Tax Collection Data Split by County"](#)

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, 2011, these tables include the values shown below.

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-11	31-Dec-99
102	JaD	Jane Doe	Female	Single	5001	SV	Sunnyvale	...	1-Jan-11	31-Dec-99
103	JiD	Jim Doe	Male	Married	5002	CU	Cupertino	...	1-Jan-11	31-Dec-99

County Table

County Id	County CD	County Nm	Population	...	Eff Frm	Eff To
5001	SV	Sunnyvale	Very High	...	1-Jan-11	31-Dec-99
5002	CU	Cupertino	High	...	1-Jan-11	31-Dec-99

TaxPaid Table

Cust Id	Day	Tax Type	Tax
101	1-Jan-11	Professional Tax	100
102	1-Jan-11	Professional Tax	100
103	1-Jan-11	Professional Tax	100

Assume that the following events occurred in January 2011:

- On January 20, 2011, Jane Doe marries.
- On Jan 29, 2011, John Doe moves from Sunnyvale to Cupertino.

Consequently, as shown below, on February 1, 2011, 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-11	29-Jan-11
102	JaD	Jane Doe	Female	Single	5001	SV	Sunnyvale	...	1-Jan-11	20-Jan-11
103	JiD	Jim Doe	Male	Married	5002	CU	Cupertino	...	1-Jan-11	31-Dec-99
104	JaD	Jane Doe	Female	Married	5001	SV	Sunnyvale	...	21-Jan-11	31-Dec-99
105	JoD	John Doe	Male	Single	5002	CD	Cupertino	...	30-Jan-11	31-Dec-99

County table

County Id	County CD	County Nm	Population	...	Eff Frm	Eff To
5001	SV	Sunnyvale	Very High	...	1-Jan-11	31-Dec-99
5002	CU	Cupertino	High	...	1-Jan-11	31-Dec-99

TaxPaid Table

Cust Id	Day	Tax Type	Tax
101	1-Jan-11	Professional Tax	100
102	1-Jan-11	Professional Tax	100
103	1-Jan-11	Professional Tax	100
105	1-Feb-11	Professional Tax	100
104	1-Feb-11	Professional Tax	100
103	1-Feb-11	Professional Tax	100

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

Assuming the [Data used for the examples](#) on page 5-9, to show the tax collection data split by marital 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.cnty_id
GROUP BY cust.cust_nm, cust.m_status
ORDER BY 1,2,3;
```

The results of this query are shown below. 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](#) on 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, 2011.

```
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-2011', '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-2011', '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 below. Since Jane Doe was single on January 15, 2011 (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-2011` rather than `15-JAN-2011`. Since Jane Doe was married on February 9, 2011, then, as shown below 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](#) on 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 below. 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](#) on page 5-9, issue the following SQL statement to show the tax collection data split by county using an analysis date of January 15, 2011.

```
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-2011', '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-2011', '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 below. Note that since John Doe was in Sunnyvale as of the analysis date of January 15, 2011, all tax for John Doe is accounted for under the Sunnyvale county.

Cust Nm	County Nm	Tax
Jane Doe	Sunnyvale	200

Cust Nm	County Nm	Tax
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-2011` rather than `15-JAN-2011`. Since John Doe lived in Cupertino on February 9, 2011, then, as shown below 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

Tutorial: Creating a New Oracle Communications Data Model Dashboard

This tutorial explains how to create a new dashboard based on the Oracle Communications Data Model webcat included with the sample Oracle Business Intelligence Suite Enterprise Edition reports delivered with Oracle Communications Data Model.

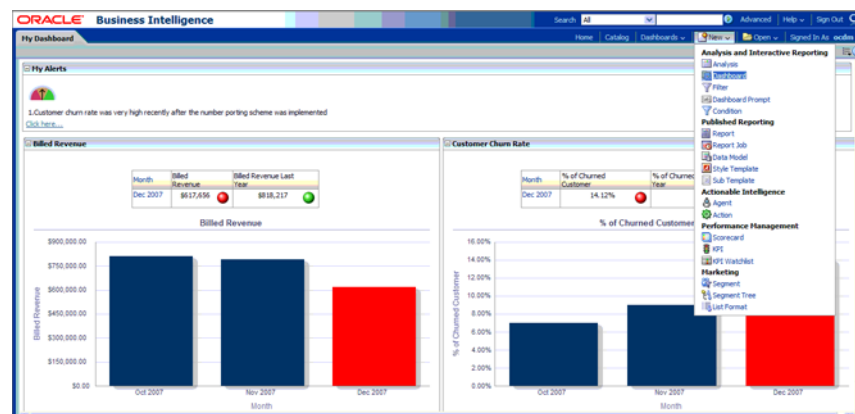
See: *Oracle Communications Data Model Installation Guide* for more information on installing the sample reports and deploying the Oracle Communications Data Model RPD and webcat on the Business Intelligence Suite Enterprise Edition instance.

In this example assume that you want to create a dashboard named "Dropped call and Failed Call", and put both "Dropped Call Rate Report" and "Failed Call Rate Report" into this new dashboard.

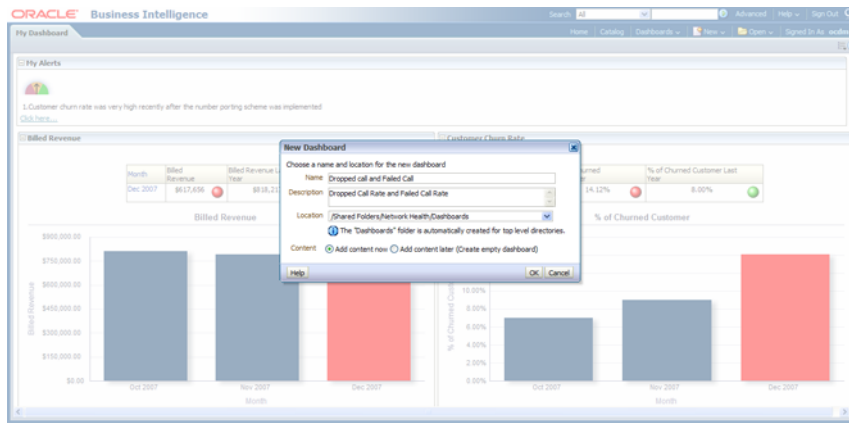
To create a dashboard, take the following steps:

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 `ocdm`, and provide the password.

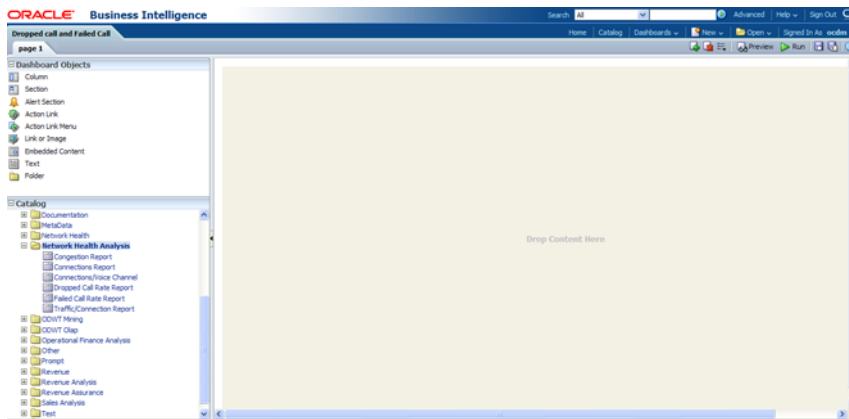
Then, click **newDashboard** to create an Oracle Business Intelligence Suite Enterprise Edition dashboard.



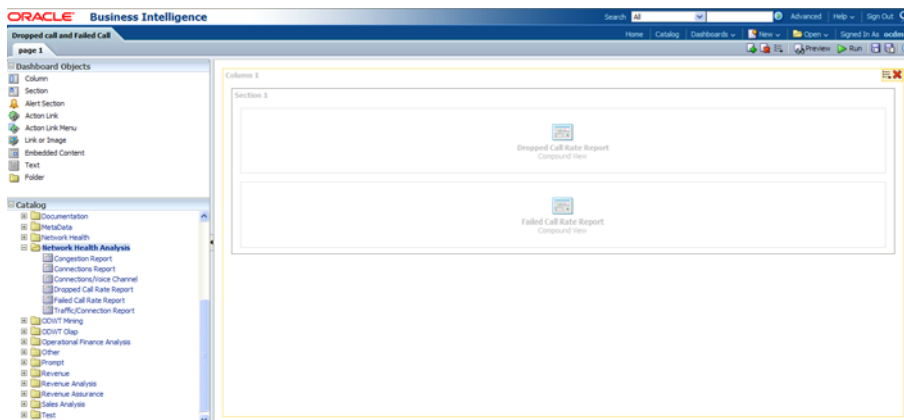
- Input name and description, save it to the Network Health folder. Click OK.



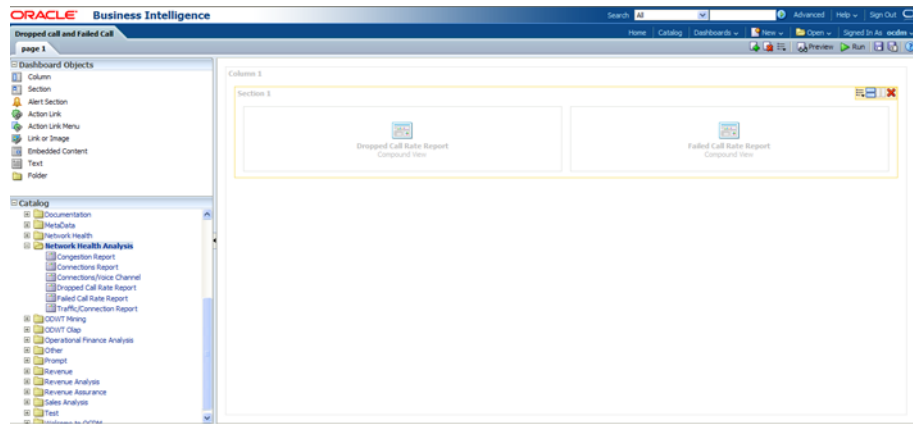
- In the Catalog view, expand the Network Health Analysis folder. You can see the Failed Call Rate Report and Dropped Call Rate Report.



- Drag the Failed Call Rate Report and Dropped Call Rate Report from the Catalog view into the right panel.

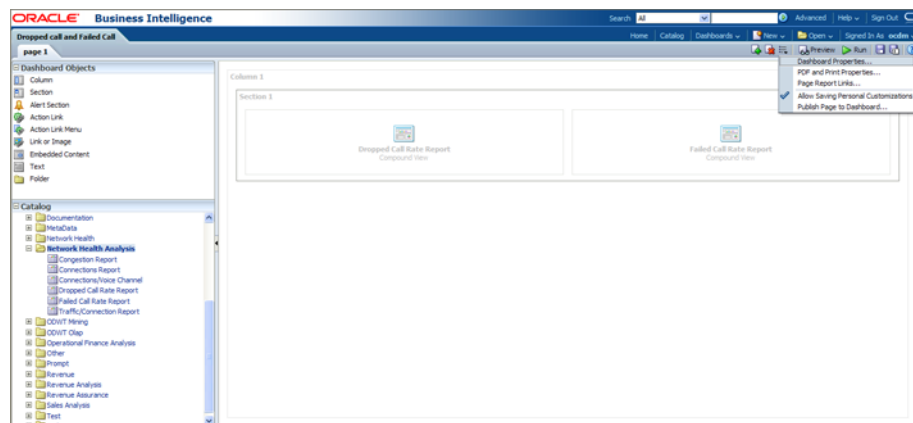


- You can change the layout of this section to organize the two reports by horizontal or vertical.

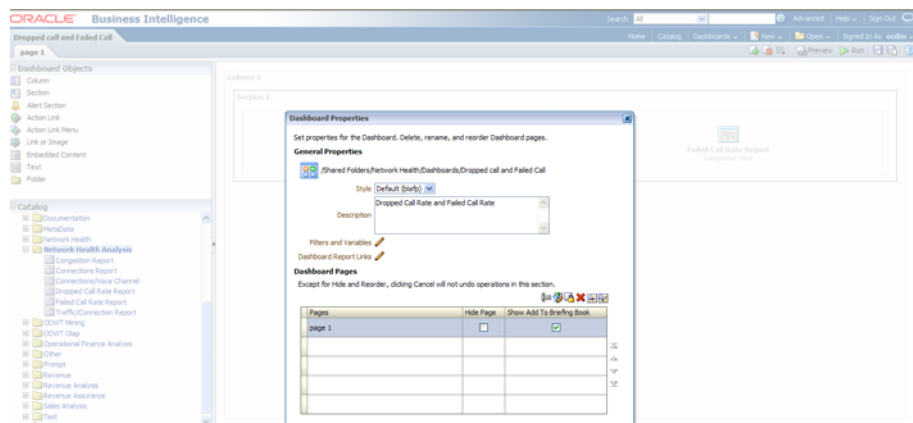


Note that the page name is still "Page1" so you must change it.

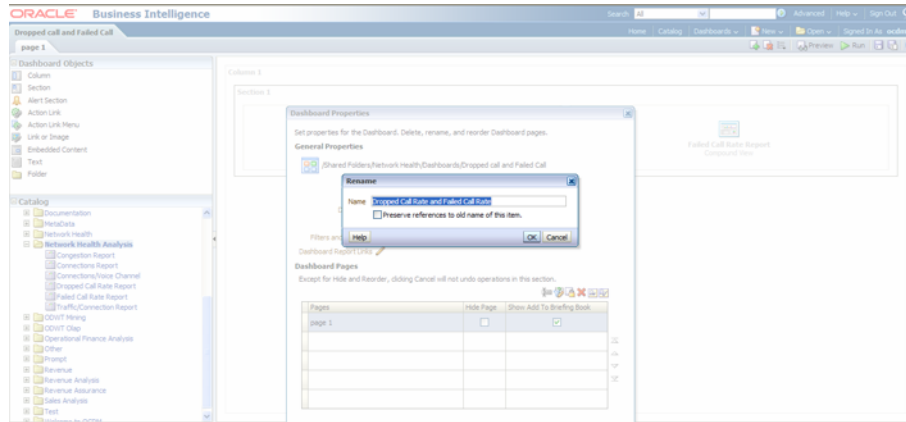
7. To change the page name:
 - a. Select the Dashboard.



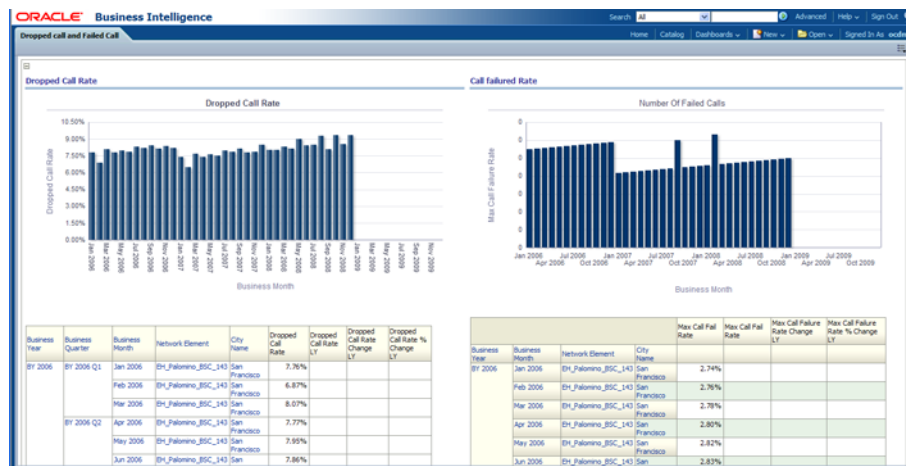
- b. In Dashboard Properties window, click **Change Name**.



- c. Change the name to "Dropped Call Rate and Failed Call Rate", then click **OK**.



8. Click **Save** on the top of the dashboard. Now you have a new dashboard.



Oracle by Example: For more information on creating dashboards see the "Creating Analyses and Dashboards 11g" OBE tutorial.

To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "[Oracle Technology Network](#)" on page xii; and, then, search for the tutorial by name.

Tutorial: Creating a New Oracle Communications Data Model Report

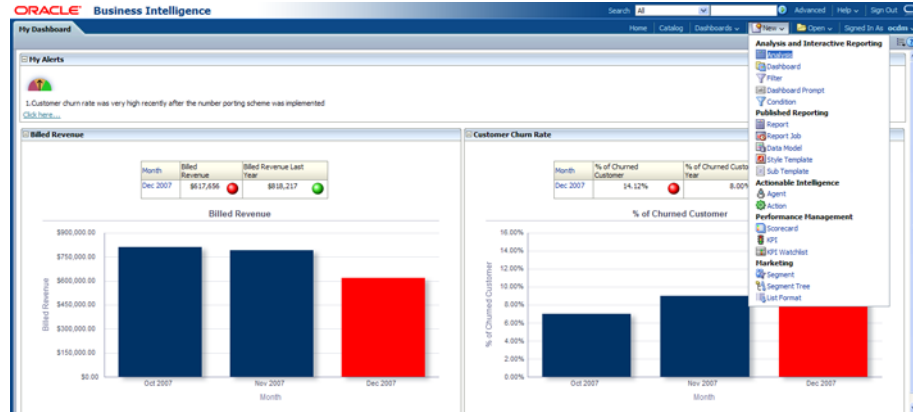
This tutorial explains how to create a new report based on the Oracle Communications Data Model webcat included with the sample Oracle Business Intelligence Suite Enterprise Edition reports delivered with Oracle Communications Data Model.

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

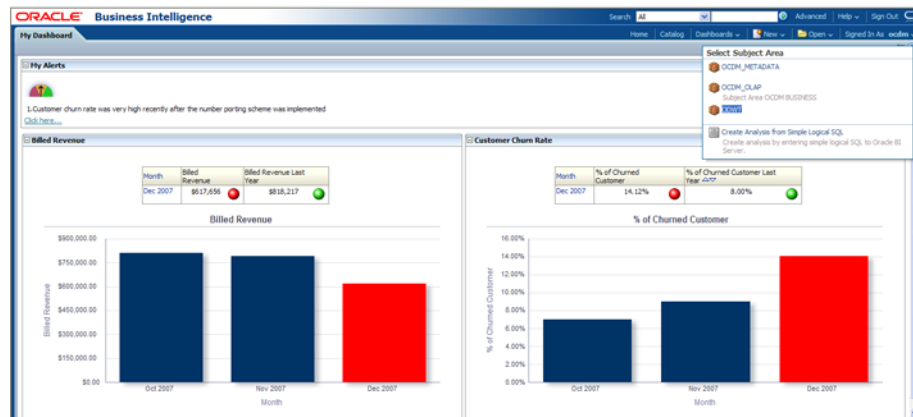
In this example, assume that you want to create a report named "Dropped call vs. Failed Call" to put both dropped call rate and failed call rate in one report.

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

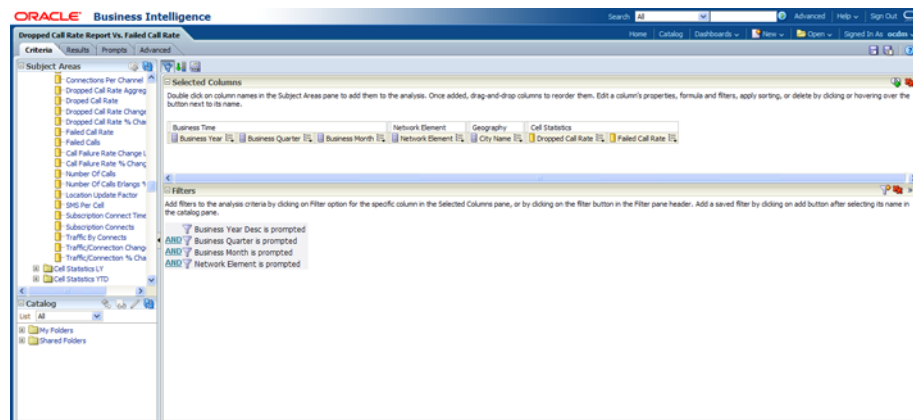
1. In the browser, open the login page at `http://servername:9704/analytics` where `servername` is the server on which the webcat is installed.
2. Login with username of `ocdm`, and provide the password.
Then, click **newAnalysis** to create an Oracle Business Intelligence Suite Enterprise Edition report.



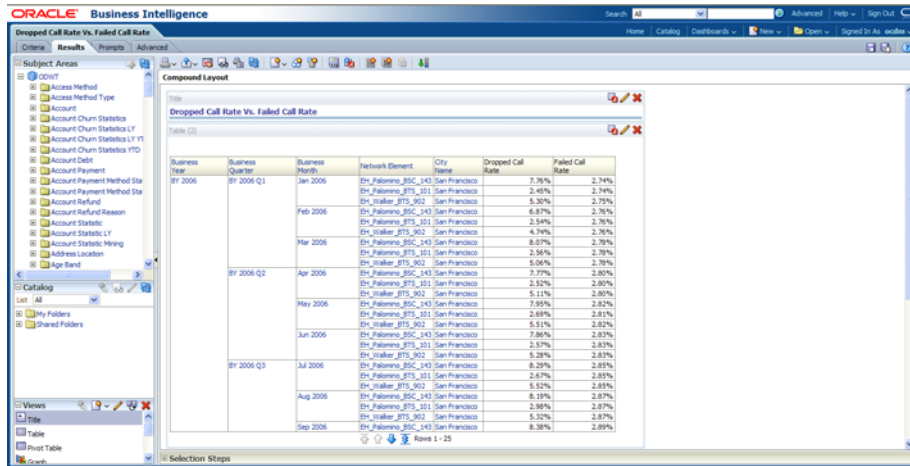
3. Select **Subject Area**, then select **ODWT** to create a relational report.



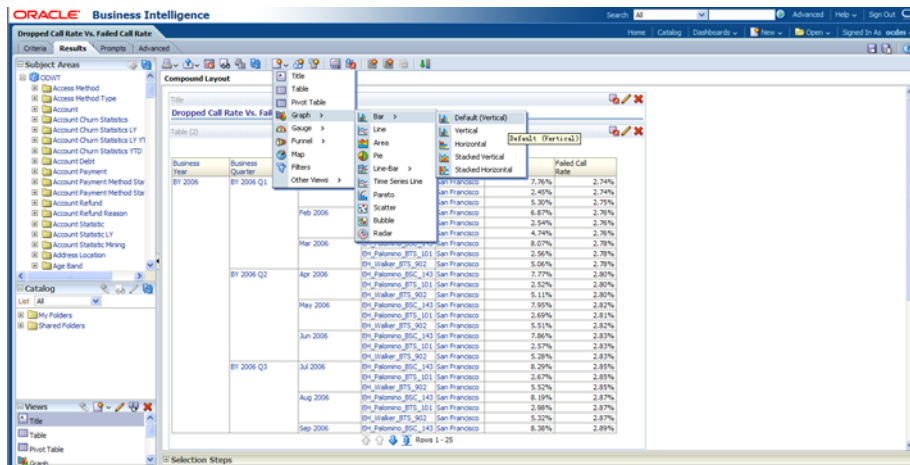
4. Drag and put the dimension and fact columns into the Select Columns panel.



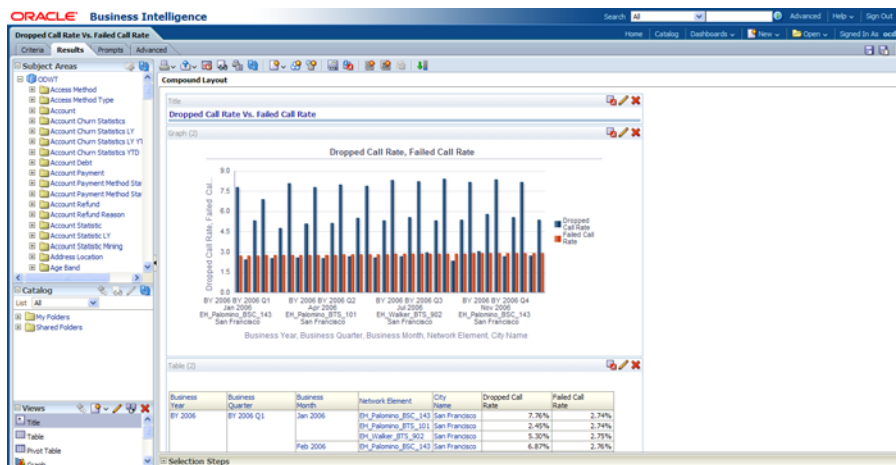
5. Select the **Results** tab to view the report



6. Click New View to add a chart into report.



7. Click Save to save this report to the Network Health Analysis folder



Oracle by Example: For more information on creating a report, see the "Creating Analyses and Dashboards 11g" OBE tutorial.

To access the tutorial, open the Oracle Learning Library in your browser by following the instructions in "[Oracle Technology Network](#)" on page xii; and, then, search for the tutorial by name.

Sizing and Configuring an Oracle Communications Data Model Warehouse

This appendix provides information about sizing and configuring an Oracle Communications Data Model warehouse. It contains the following topics:

- [Sizing an Oracle Communications Data Model Warehouse](#)
- [Configuring a Balanced System for Oracle Communications Data Model](#)

Sizing an Oracle Communications 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. Regardless of the design or implementation of a data warehouse the initial key to good performance lies in the hardware configuration used. This has never been more evident than with the recent increase in the number of data warehouse appliances in the market.

But how do you go about sizing such a 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 Communications 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 Communications 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 Communications Data Model Warehouse

Use the following formula, based on the factors outlined in ["Calculation Factors When Making a Data Volume Calculation for an Oracle Communications Data Model Warehouse"](#) on page A-1, to determine the minimum disk space requirements for an Oracle Communications Data Model warehouse.

Disk Space Minimum Requirements = Raw data size * Database space factor * (1+GrthperY)ⁿ*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
- 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.

Configuring a Balanced System for Oracle Communications 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 Communications Data Model:

- [Maintaining High Throughput in an Oracle Communications Data Model Warehouse](#)
- [Configuring I/O in an Oracle Communications Data Model for Bandwidth not Capacity](#)
- [Planning for Growth of Your Oracle Communications Data Model](#)
- [Testing the I/O System Before Building the Warehouse](#)
- [Balanced Hardware Configuration Guidelines for Oracle Communications Data Model](#)

Maintaining High Throughput in an Oracle Communications 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

Configuring I/O in an Oracle Communications 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 200MB 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.

Planning for Growth of Your Oracle Communications 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.

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.

Balanced Hardware Configuration Guidelines for Oracle Communications 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).

Oracle now provides the Oracle Database Machine, Exadata which combines industry-standard hardware from Oracle, Oracle Database 11g Release 2, and Oracle

Exadata Storage Server Software to create a faster, more versatile database machine. It's a completely scalable and fault tolerant package for all data management, especially for data warehousing.

Oracle also has a series of Optimized Warehouse Reference configurations that help customers take the risk out of designing and deploying Oracle data warehouses. Using extensive field experience and technical knowledge, Oracle and its hardware partners have developed a choice of data warehouse reference configurations that can support various sizes, user populations and workloads. These configurations are fast, reliable and can easily scale from 500 GB to over 100 TB on single and clustered servers to support tens to thousands of users.

Working with User Privileges in Oracle Communications Data Model

This appendix provides information about managing user privileges in Oracle Communications Data Model. It contains the following topics:

- [Accounts Created for Oracle Communications Data Model](#)
- [When You Must Consider User Privileges in an Oracle Communications Data Model](#)
- [Granting Only Select Privileges to Users of the Sample Reports](#)

Accounts Created for Oracle Communications Data Model

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` account. Ensure you unlock these accounts with new passwords following the post-installation steps provided in *Oracle Communications Data Model Installation Guide*.

See: *Oracle Communications 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 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.

When You Must Consider User Privileges in an Oracle Communications Data Model

The installation process grants the necessary privileges required for users of both accounts. After installing 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 require 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 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 users of the sample reports by following the steps outlined in ["Granting Only Select Privileges to Users of the Sample Reports"](#) on page B-2.

Granting Only Select Privileges to Users of the Sample Reports

To grant only select privileges to users of the sample reports, 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 select privileges for these tables 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.

Using the NCC Adapter for Oracle Communications Data Model

This appendix provides information about populating an Oracle Communications Data Model warehouse using the NCC Adapter for Oracle Communications Data Model.

It contains the following topics:

- [What is the NCC Adapter for Oracle Communications Data Model?](#)
- [Populating a Warehouse Using the NCC Adapter for Oracle Communications Data Model](#)
- [NCC Adapter for Oracle Communications Data Model Execution Flows](#)
- [Components of the NCC Adapter for Oracle Communications Data Model](#)
- [Oracle Communications Data Model Tables and Columns Filled by NCC Adapter](#)

For information on installing and configuring the NCC Adapter for Oracle Communications Data Model, see *Oracle Communications Data Model Installation Guide*.

For detailed information about the objects and ETL associated with the NCC Adapter for Oracle Communications Data Model, see *Oracle Communications Data Model Reference*.

What is the NCC Adapter for Oracle Communications Data Model?

The Oracle Communications Network Charging and Control Adapter for Oracle Communications Data Model (sometimes called the NCC Adapter for Oracle Communications Data Model) is an option of the Oracle Communications Data Model. You can use the NCC Adapter for Oracle Communications Data Model to populate the foundation layer (that is, the base, reference, and lookup tables) of an Oracle Communications Data Model warehouse with data from an Oracle Communications Network Charging and Control (NCC) system.

The NCC Adapter for Oracle Communications Data Model uses the following components:

- [Database Schemas Used by the NCC Adapter for Oracle Communications Data Model](#)
- [Oracle Data Integrator](#)
- [Oracle GoldenGate](#)

Database Schemas Used by the NCC Adapter for Oracle Communications Data Model

The NCC Adapter for Oracle Communications Data Model uses the following Database schemas:

- **Source schemas** The source of the NCC Adapter for Oracle Communications Data Model is an NCC system as defined by the NCC BE and CSS schemas.

The specific source objects are described in "[NCC Source Schema Objects](#)" on page C-8.

- **Staging layer schema.** The NCC Adapter for Oracle Communications Data Model uses a staging area when loading data from the NCC system to foundation layer objects of an Oracle Communications Data Model warehouse. This staging area is used for changed data and historical data storage and as a source schema when loading the foundation layer objects of the Oracle Communications Data Model. The NCC Adapter for Oracle Communications Data Model defines the staging as a database schema named `ncc_stg`.

The `ncc_stg` schema is described in "[ncc_stg Schema Objects](#)" on page C-8.

- **Target schema.** The target of the NCC Adapter for Oracle Communications Data Model is an Oracle Communications Data Model warehouse. The specific target objects are the foundation layer objects of the Oracle Communications Data Model. These objects are defined in the `ocdm_sys` schema which is delivered with Oracle Communications Data Model and documented in Oracle Communications Data Model Reference.

The NCC Adapter for Oracle Communications Data Model defines additional procedures in the `ocdm_sys` schema. These procedures are outlined in "[Definitions Added to the ocdm_sys Schema](#)" on page C-9.

Oracle Data Integrator

The NCC Adapter for Oracle Communications Data Model uses ETL created by Oracle Data Integrator (ODI) as the ETL for performing an initial load of the foundation layer objects of the Oracle Communications Data Model (that is, the base, reference, and lookup tables). You can also use ODI ETL to perform a scheduled refresh of the staging database.

For more information on the use of ODI with the NCC Adapter for Oracle Communications Data Model, see the following topics:

- "[Initial Loading Using the NCC Adapter for Oracle Communications Data Model](#)" on page C-3.
- "[Execution Flow When Using the Adapter with only ODI](#)" on page C-7
- "[Refreshing the Foundation Layer with Real-Time Data](#)" on page C-4,
- "[The NCC_OCDM Project](#)" on page C-10.

Oracle GoldenGate

After you have performed an initial load of the foundation layer objects of the Oracle Communications Data Model using ODI, you can use Oracle GoldenGate for to perform real-time feed of NCC source data to the staging database.

For more information on the use of Oracle GoldenGate with the NCC Adapter for Oracle Communications Data Model, see the following topics:

- "[Refreshing the Foundation Layer with Real-Time Data](#)" on page C-4.

- ["Execution Flow When Using the Adapter Oracle GoldenGate and ODI"](#) on page C-6.

Populating a Warehouse Using the NCC Adapter for Oracle Communications Data Model

You use the NCC Adapter for Oracle Communications Data Model to populate or refresh the foundation layer of the Oracle Communications Data Model warehouse (that is, the base, reference, and lookup tables defined in the `ocdm_sys` schema). For more information, see:

- [Initial Loading Using the NCC Adapter for Oracle Communications Data Model](#)
- [Refreshing the Data Using the NCC Adapter for Oracle Communications Data Model](#)

After you use the NCC Adapter for Oracle Communications Data Model to populate the foundation layer, you populate or refresh the access layer of the Oracle Communications Data Model (that is, the derived tables, aggregate tables, OLAP cubes, and data mining models defined in the `ocdm_sys` schema) in the same manner as you would if you were *not* using an application adapter for Oracle Communications Data Model to populate the foundation layer. For more information, see ["Performing an Initial Load of the Access Layer"](#) on page 4-11 and ["Refreshing the Access Layer of an Oracle Communications Data Model Warehouse"](#) on page 4-15.

Initial Loading Using the NCC Adapter for Oracle Communications Data Model

To perform an initial load of the foundation layer of an Oracle Communications Data Model warehouse using the NCC Adapter for Oracle Communications Data Model, you use Oracle Data Integrator (ODI), take the following steps:

1. Verify that the installation and configuration created the schema objects described in ["Schema Definitions Added by the NCC Adapter for Oracle Communications Data Model"](#) on page C-8.

See: Oracle Communications Data Model Installation Guide for instructions on installing and configuring the NCC Adapter for Oracle Communications Data Model for use by ODI.

2. From ODI Studio Designer, go to the `NCC_OCDM` project, then take the following steps:
 - a. Select `SRC_STG_NONOGG`, then **Packages**, and then, select and execute the following packages:
 - `SRC_STG_LOAD_NON_OGG_PKG`
 - `SRC_STG_EDR_LOAD_INITIAL_PKG`
 - b. Select `STG_OCDM`, then **Packages**, and then, select and execute the following packages:
 - `STG_OCDM_INITIAL_LOAD_PKG`
 - `STG_OCDM_EDR_PKG`

See: ["The NCC_OCDM Project"](#) on page C-10 for more information on the packages and ["Execution Flow When Using the Adapter with only ODI"](#) on page C-7 for a description of the execution flow for the initial load.

Refreshing the Data Using the NCC Adapter for Oracle Communications Data Model

Once you have performed an initial data load of an Oracle Communications Data Model warehouse, you need to refresh the data

You use the NCC Adapter for Oracle Communications Data Model to refresh the data in the foundation layer of an Oracle Communications Data Model warehouse. There are two different ways that you can refresh the data in the foundation layer:

- Use Oracle Data Interface to refresh all of the data in the staging area and in the foundation layer of the Oracle Communications Data Model warehouse. This steps for performing this type of refresh are given in ["Refreshing the Foundation Layer on a Scheduled Basis"](#) on page C-4. The execution flow for this type of refresh is described in ["Execution Flow When Using the Adapter with only ODI"](#) on page C-7.
- Use Oracle GoldenGate to populate the staging area with real-time data and, then, use Oracle Data Interface to refresh the data in the Oracle Communications Data Model warehouse. This steps for performing this type of refresh are given in ["Refreshing the Foundation Layer with Real-Time Data"](#) on page C-4. The execution flow for this type of refresh is described in ["Execution Flow When Using the Adapter Oracle GoldenGate and ODI"](#) on page C-6

Refreshing the Foundation Layer on a Scheduled Basis

If you are not using the NCC Adapter for Oracle Communications Data Model with Oracle GoldenGate to populate the foundation layer of an Oracle Communications Data Model warehouse (that is, the base, reference, and lookup tables) with real-time data, you refresh the foundation layer on a scheduled basis using Oracle Data Integrator (ODI).

To refresh the data in the foundation layer using only ODI, take the following steps:

1. Verify that the installation and configuration created the schema objects described in ["Schema Definitions Added by the NCC Adapter for Oracle Communications Data Model"](#) on page C-8.
 - See:** Oracle Communications Data Model Installation Guide for instructions on installing and configuring the NCC Adapter for Oracle Communications Data Model for use by ODI.
2. From ODI Studio Designer, go to the NCC_OCDM project, then take the following steps:
 - a. Select **SRC_STG_NONOGG**, then **Packages**, and then, select and execute the following packages:
 - SRC_STG_LOAD_NON_OGG_PKG
 - SRC_STG_EDR_LOAD_INCREMENTAL_PKG
 - b. Select **STG_OCDM**, then **Packages**, and then, select and execute the following packages:
 - STG_OCDM_INITIAL_LOAD_PKG
 - STG_OCDM_EDR_PKG

Refreshing the Foundation Layer with Real-Time Data

After performing an initial load of the Oracle Communications Data Model warehouse as described in ["Initial Loading Using the NCC Adapter for Oracle Communications](#)

[Data Model](#)" on page C-3, you can refresh the data in the foundation layer of an Oracle Communications Data Model warehouse on a real-time basis.

Take the followings steps to ensure that to the data in the foundation layer of an Oracle Communications Data Model warehouse in this manner:

1. Install the NCC Adapter for Oracle Communications Data Model and configure it for use by both Oracle GoldenGate and Oracle Data Integrator (ODI) and as described in Oracle Communications Data Model Installation Guide.

Verify that the installation and configuration created the schema objects described in "[Schema Definitions Added by the NCC Adapter for Oracle Communications Data Model](#)" on page C-8.

2. From the GGSCI prompt, issue `info all` commands to verify that the Oracle GoldenGate processes needed by the NCC Adapter for Oracle Communications Data Model (listed below) are running.

Source System Processes	Staging System Processes
Manager process	Manager process
Extract process (EXTNCC)	Replicate process (REPNC)
Extract Pump process (EXTPNCC)	

The following code shows the issuing of these commands with successful results.

```
GGSCI> (mypc1) 5> info all
```

```

Program      Status      Group      Lag          Time Since Chkpt
-----
MANAGER      RUNNING
EXTRACT      RUNNING    EXTNCC     00:00:00     00:00:02
EXTRACT      RUNNING    EXTPNCC    00:00:00     00:00:06
    
```

```
GGSCI> (ocdm01) 2> info all
```

```

Program      Status      Group      Lag          Time Since Chkpt
-----
MANAGER      RUNNING
REPLICAT     RUNNING    REPNC      00:00:00     00:03:09
    
```

Tips: If you have two source systems, check process status on both source systems.

For commands to manage Oracle GoldenGate processes, see *Oracle Communications Data Model Installation Guide*.

3. From ODI Studio Designer, go to the NCC_OCDM project, then take the following steps:
 - a. Select **SRC_STG_NONOGG**, then select **Procedures**, and then select and execute the **EXEC_NORMAL_TABLES_PKG**.
 - b. Select **STG_OCDM**, then select **Packages**, and then, select and execute the **STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG**.

See: ["The NCC_OCDM Project"](#) on page C-10 for more information on the packages; and ["Execution Flow When Using the Adapter Oracle GoldenGate and ODI"](#) on page C-6 for an explanation of the execution flow.

NCC Adapter for Oracle Communications Data Model Execution Flows

The NCC Adapter for Oracle Communications Data Model uses a slightly different execution flow depending on whether or not you are using the adapter with Oracle GoldenGate:

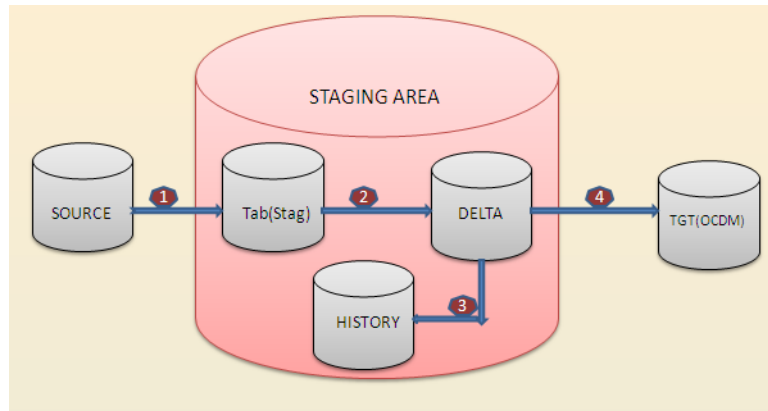
- [Execution Flow When Using the Adapter Oracle GoldenGate and ODI](#)
- [Execution Flow When Using the Adapter with only ODI](#)

Execution Flow When Using the Adapter Oracle GoldenGate and ODI

When you use the NCC Adapter for Oracle Communications Data Model with Oracle GoldenGate to refresh the foundation layer of the Oracle Communications Data Model warehouse as described in ["Refreshing the Foundation Layer with Real-Time Data"](#) on page C-4, the foundation layer data is always synchronized with the NCC source data.

[Figure C-1](#) illustrates the NCC Adapter for Oracle Communications Data Model execution flow when using the adapter with Oracle GoldenGate.

Figure C-1 Execution Flow When Using the Adapter with Oracle GoldenGate



When using Oracle GoldenGate with the NCC Adapter for Oracle Communications Data Model, the execution flow is as follows:

1. Assuming that all Oracle GoldenGate processes for the NCC Adapter for Oracle Communications Data Model are in running mode, whenever new data comes into the source database, the GoldenGate parameter files extract that new incoming data and use it to populate the normal staging tables.
2. (Steps 2 and 3 in the diagram) The `pre_ocdm_load` procedure executes and loads the delta tables in the staging area with data from the normal staging tables and the the delta history tables in the staging area with data from the delta tables.

Note: Only the data in the delta history tables remains forever and always in sync with source data. Once the data in the delta staging tables is loaded into the foundation layer of the Oracle Communications Data Model warehouse, the delta tables are truncated.

- (Step 4 in the diagram) ODI loads data from the delta tables in the staging layer into the foundation layer of the Oracle Communications Data Model warehouse when the data in the staging table changes.

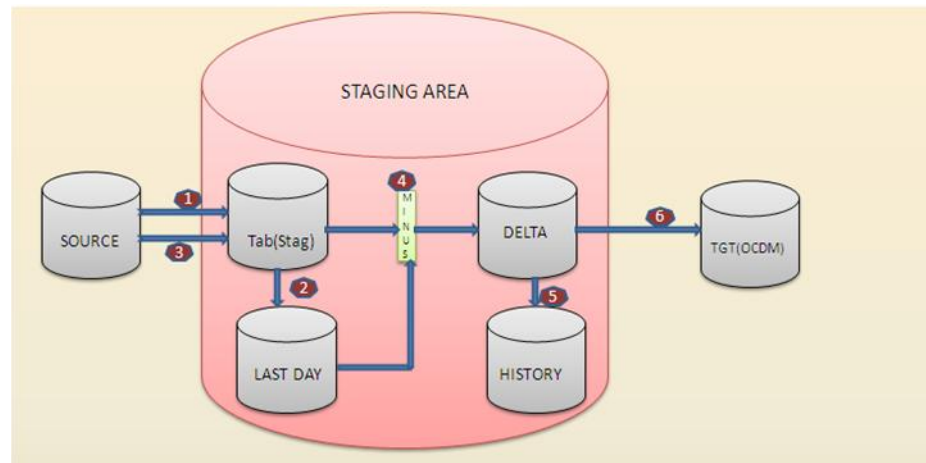
Execution Flow When Using the Adapter with only ODI

You can use the NCC Adapter for Oracle Communications Data Model with Oracle Data Integrator (ODI) alone. This method of using the NCC Adapter for Oracle Communications Data Model is sometimes called the ODI-pull version of the NCC Adapter for Oracle Communications Data Model.

You always use the ODI-pull version of NCC adapter for the initial load you perform right after installation as described in ["Initial Loading Using the NCC Adapter for Oracle Communications Data Model"](#) on page C-3. Later, you can also use it to perform scheduled data refresh as described in ["Refreshing the Foundation Layer on a Scheduled Basis"](#) on page C-4.

[Figure C-2](#) illustrates the NCC Adapter for Oracle Communications Data Model execution flow when using the adapter *only* with ODI.

Figure C-2 Execution Flow When Using the Adapter without Oracle GoldenGate



When using this approach, the execution process is as follows:

- (Steps 1 and 3 in the diagram) The adapter, using ODI, loads the data from the source database into stage database.
- (Steps 2 and 5 in the diagram) Using pre and post procedures, the adapter loads data into several tables stage database tables; for example, lastday, delta, and history tables.
- (Step 4 in the diagram) The adapter loads the delta tables in the staging area using normal table 'MINUS' last day table.

4. (Step 6 in diagram) The adapter loads the data from the delta tables into foundation layer of the Oracle Communications Data Model warehouse.

Components of the NCC Adapter for Oracle Communications Data Model

The NCC Adapter for Oracle Communications Data Model uses the following components:

- [NCC Source Schema Objects](#)
- [Schema Definitions Added by the NCC Adapter for Oracle Communications Data Model](#)
- [STG_OCDM Folder](#)

NCC Source Schema Objects

The NCC Adapter for Oracle Communications Data Model uses the data in the following NCC tables as source data:

BE_BALANCE
BE_BUCKET
BE_WALLET
CCS_ACCT
CCS_ACCT_ACCT_REFERENCES
CCS_ACCT_REFERENCE
CCS_ACCT_TYPE
CCS_BALANCE_TYPE
CCS_BALANCE_UNIT
CCS_BE_CDR
CCS_MB_RECHARGE
CCS_VOUCHER_BATCH
CCS_VOUCHER_REFERENCE
CCS_WALLET_TYPE

Additionally, the NCC Adapter for Oracle Communications Data Model uses the `CCS_RECHARGE_TYPE` view as defined in the NCC `css` Database.

Schema Definitions Added by the NCC Adapter for Oracle Communications Data Model

Installing and configuring the NCC Adapter for Oracle Communications Data Model as described in *Oracle Communications Data Model Installation Guide* adds the following definitions:

- A database named `ncc_stg` that defines a staging area for use by the adapter. For information on the definitions in this database, see "[ncc_stg Schema Objects](#)" on page C-8.
- Procedure definitions that are used to populate the foundation layer of an Oracle Communications Data Model warehouse. These definitions are added to the `ocdm_sys` schema delivered with Oracle Communications Data Model. For more information on these procedures, see "[Definitions Added to the ocdm_sys Schema](#)" on page C-9.

`ncc_stg` Schema Objects

The `ncc_stg` schema defines the staging area used by the NCC Adapter for Oracle Communications Data Model. The schema contains the definitions outlined in [Table C-1](#).

Table C-1 *Definitions in the ncc_stg Schema*

Object Type	Object Name	Description and/or Usage
Type	CDR_T	Supports the population of Event Detail Record (EDR) data.
Type	CDRSET_T	Supports for the population of EDR data.
Table	CCS_BE_CDR	Event detail record table.
Table	<i>source_table_name</i>	Normal staging tables. There is one table definition for each of NCC source tables listed in "NCC Source Schema Objects" on page C-8.
Table	<i>source_table_name_LD</i>	Previous day tables. There is one table definition for each of NCC source tables outlined in "NCC Source Schema Objects" on page C-8.
Table	<i>source_table_name_DELTA</i>	Delta tables. There is one table definition for each of NCC source tables outlined in "NCC Source Schema Objects" on page C-8.
Table	<i>source_table_name_DELTA_H</i>	Delta history tables. There is one table definition for each of NCC source tables outlined in "NCC Source Schema Objects" on page C-8.
View	CCS_RECHARGE_TYPE	This is a view in the staging area that corresponds to a view of the same name in the <code>ncc_ccs</code> database.
View	CDR_FREEFORM_RECHARGE	Supports the population of EDR data.
View	CDR_PREPAID	Supports the population of EDR data.
View	CDR_SMS	Supports the population of EDR data.
View	CDR_VOICE_CALL	Supports the population of EDR data.
View	VOUCHER_RECHARGE	Supports the population of EDR data.
View	VOUCHER_RECHARGE_BALANCE_TYPE	Supports the population of EDR data.
Package	EDR	Supports the population of EDR data.
Procedure	POST_STAGING_LOAD	Maintains the everyday source data changes and history of the data changes. Used by the Oracle GoldenGate option.
Procedure	PRE_STAGING_LOAD	Maintains the everyday source data changes and history of the data changes. Used by the Oracle GoldenGate option.
Procedure	PRE_OCDM_LOAD	Maintains the everyday source data changes and history of the data changes. Used by the ODI-only option.

Definitions Added to the ocdm_sys Schema

Installing and configuring the adds the following procedure definitions to the `ocdm_sys` schema delivered with Oracle Communications Data Model.

```

DISB_CONS
ENAB_CONS
INSERT_REF
LOOK_UP_LOAD_ONE_TIME
UPDATE_ACCT
UPDATE_CHURN_DATE
UPDATE_CNRT
UPDATE_CUST
UPDATE_PROD
UPDATE_PRPD_VCHR_INSTNC
UPDATE_SBRP

```

The NCC Adapter for Oracle Communications Data Model uses these procedures as it populates the target objects.

The NCC_OCDM Project

When you install and configure the NCC Adapter for Oracle Communications Data Model, an ODI project named NCC_OCDM project is defined. The NCC_OCDM project contains the following folders:

- **SRC_STG_NONAGG Folder** contains packages used to populate the staging area with data from the NCC source system.
- **STG_OCDM Folder** contains packages used to populate the foundation layer of an Oracle Communications Data Model warehouse with data from the NCC Adapter for Oracle Communications Data Model staging area.

SRC_STG_NONAGG Folder

The packages in the STG_STG_MPMAGG folder of the NCC_OCDM project are used to populate the tables in the staging layer of the (that is, the tables defined in the `ncc_stg` database) from data in the NCC source system (that is, the tables defined in the NCC BE and CSS databases).

SRC_STG_LOAD_NON_OGG_PKG Package This package loads data from the NCC source database into the NCC staging database. You use this package during an initial load and when you refresh the foundation layer using only ODI.

SRC_STG_EDR_LOAD_INCREMENTAL_PKG Package This package loads the NCC source Event Detail Record incremental data into the Staging Database EDR Table (that is, the `CCS_BE_CDR` table).

SRC_STG_EDR_LOAD_INITIAL_PKG Package This package loads NCC source Event Detail Record initial data into the Staging Database EDR Table (that is, `CCS_BE_CDR`). You use this package during an initial load of the foundation layer.

STG_OCDM Folder

The packages in the STG_OCDM folder are used to populate the tables in the foundation layer of an Oracle Communications Data Model warehouse (that is, the tables defined in the `ocdm_sys` database) from data in the staging area (that is, the tables defined in the `ncc_stg` database).

The following packages are defined in the STG_OCDM folder of the NCC_OCDM project:

- **STG_OCDM_INITIAL_LOAD_PKG Package**
- **STG_OCDM_EDR_PKG Package**
- **STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG Package**

STG_OCDM_INITIAL_LOAD_PKG Package This package loads data into the foundation layer tables of the Oracle Communications Data Model warehouse from data in the staging area delta tables (other than Event Detail Record objects). You use this package during the initial load and when refreshing the data using only ODI.

This package executes interfaces in the STG_OCDM folder named `ncc_source_table_MAP` (where `ncc_source_table` is the name of an NCC source table listed in "NCC Source Schema Objects" on page C-8) in hierarchical sequence. These interfaces use the data in the staging delta tables (that is, the `ncc_source_table_DELTA` tables) to populate the related foundation layer tables.

The following interfaces are executed by this package:

- BE_BALANCE_MAP
- BE_BUCKET_MAP
- CCS_ACCT_BE_WALLET_MAP
- CCS_ACCT_CNRT_MAP
- CCS_ACCT_REFERENCE_MAP
- CCS_ACCT_SBRP_MAP
- CCS_ACCT_TYPE_MAP
- CCS_BALANCE_TYPE_MAP
- CCS_BALANCE_UNIT_MAP
- CCS_MB_RECHARGE_MAP
- CCS_RECHARGE_TYPE_MAP
- CCS_VOUCHER_BATCH_MAP
- CCS_VOUCHER_REFERENCE_MAP
- CCS_WALLET_TYPE_MAP

STG_OCDM_EDR_PKG Package This package loads tables in the foundation layer of an Oracle Communications Data Model warehouse from Event Detail Record objects in the staging layer. You use this package during the initial load and when refreshing the data using only ODI.

The package executes the following STG_OCDM Event Detail Record interfaces in hierarchical sequence:

- EDR_FREEFROM
- EDR_RECHARGE_BALANCE_TYPE
- EDR_RECHRG
- EDR_CALL_EVT
- EDR_PREPAID
- EDR_SMS

STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG Package The STG_OCDM folder contains one package for each of the NCC source tables. These packages are named *ncc_source_table_PKG* where *ncc_source_table* is the name of an NCC source table listed in "[NCC Source Schema Objects](#)" on page C-8.

When these packages are in running mode, they execute the interfaces that populate the Oracle Communications Data Model foundation layer tables.

You use these packages when using Oracle GoldenGate to refresh the data in the staging layer.

STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG calls STG_OCDM_OGG_CDC_LOAD_PKG which finally calls the following interfaces:

- BE_BALANCE_PKG
- BE_BUCKET_PKG
- CCS_ACCT_BE_WALLET_PKG
- CCS_ACCT_CNRT_PKG

- CCS_ACCT_REFERENCE_PKG
- CCS_ACCT_SBRP_PKG
- CCS_ACCT_TYPE_PKG
- CCS_BALANCE_TYPE_PKG
- CCS_BALANCE_UNIT_PKG
- CCS_MB_RECHARGE_PKG
- CCS_RECHARGE_TYPE_PKG
- CCS_VOUCHER_BATCH_PKG
- CCS_VOUCHER_REFERENCE_PKG
- CCS_WALLET_TYPE_PKG

Oracle Communications Data Model Tables and Columns Filled by NCC Adapter

Table C-2 shows the Oracle Communications Data Model tables and columns filled by the NCC Adapter.

Table C-2 Oracle Communications Data Model Tables and Columns Filled by NCC Adapter

Target Table	Target Column	Remarks
DWB_ACCT_BAL_BUCKET	BUCKET_CD	
DWB_ACCT_BAL_BUCKET	ACCT_BAL_TYP_CD	
DWB_ACCT_BAL_BUCKET	NEVER_EXP_IND	
DWB_ACCT_BAL_BUCKET	EXP_DT	
DWB_ACCT_BAL_BUCKET	BUCKET_AMT	
DWB_ACCT_BAL_BUCKET	NEVER_USED_IND	
DWB_ACCT_BAL_BUCKET	LAST_USE_DT	
DWB_ACCT_BAL_BUCKET	EXTRNL_REF	
DWB_ACCT_BAL_BUCKET	STRT_DT	
DWB_ACCT_BAL_HIST	ACCT_KEY	Do look up, find acct_key by acct_cd
DWB_ACCT_BAL_HIST	ACCT_BAL_TYP_CD	
DWB_ACCT_BAL_HIST	MIN_CRDT_AMT	
DWL_ACCT_BAL_TYP	ACCT_BAL_TYP_CD	
DWL_ACCT_BAL_TYP	ACCT_BAL_TYP_NAME	
DWL_ACCT_BAL_TYP	CRNCY_UNIT	
DWL_ACCT_BAL_TYP	ACCT_BAL_TYP_DESC	
DWL_ACCT_TYP	ACCT_TYP_CD	
DWL_ACCT_TYP	ACCT_TYP_DESC	
DWL_ACCT_TYP	ACCT_TYP_NAME	Concatenate the value of NAME with ncc_acct
DWL_ACCT_TYP	ACCT_TYP_CD	
DWL_ACCT_TYP	LAST_UPDT_DT	

Table C-2 (Cont.) Oracle Communications Data Model Tables and Columns Filled by NCC Adapter

Target Table	Target Column	Remarks
DWL_ACCT_TYP	LAST_UPDT_BY	
DWL_ACCT_TYP	ACCT_TYP_NAME	
DWL_UOM	UNIT_OF_MSR_CD	
DWL_UOM	LAST_UPDT_DT	
DWL_UOM	LAST_UPDT_BY	
DWL_UOM	UNIT_OF_MSR_NAME	
DWR_ACCT	ACCT_CD	"NCCWL_" ID for NCC wallet and prnt_acct_key will be empty
DWR_ACCT	STAT_CD	
DWR_ACCT	BAL_EXP_PRD	
DWR_ACCT	TMNT_DT	
DWR_ACCT	INITIAL_ACTVTN_DT, & crt_dt	Load into both INITIAL_ACTVTN_DT, & crt_dt
DWR_ACCT		
DWR_ACCT	ACCT_Name	
DWR_ACCT	LAST_UPDT_DT	
DWR_ACCT	LAST_UPDT_BY	
DWR_ACCT	CRNCY_CD	
DWR_ACCT	ACCT_TYP_CD	
DWR_ACCT	LAST_UPDT_DT	
DWR_ACCT	LAST_UPDT_BY	
DWR_CNRT	ACCT_KEY	
DWR_CNRT	CUST_KEY	
DWR_CNRT	CNRT_CD	
DWR_CNRT	LAST_UPDT_DT	
DWR_CNRT	LAST_UPDT_BY	
DWR_CNRT	CNRT_CD	Load "NCCWL_" ID
DWR_CUST	CUST_CD	Load same into PRTY_CD
DWR_CUST	CUST_NAME	
DWR_CUST	LAST_UPDT_DT	
DWR_CUST	LAST_UPDT_BY	
DWR_CUST	Source System Identifier	
DWR_CUST	CRT_DT	
DWR_CUST	PRTY_DESC	
DWR_CUST	cust_typ_cd	
DWR_CUST	Chrn_DT	
DWR_PROD	VCHR_CD	Simple product_cd

Table C-2 (Cont.) Oracle Communications Data Model Tables and Columns Filled by NCC Adapter

Target Table	Target Column	Remarks
DWR_PROD	LAST_UPDT_DT	
DWR_PROD	LAST_UPDT_BY	
DWR_PROD	VCHR_DESC & PROD_DESC	
DWR_PROD	VCHR_NAME & PROD_NAME	
DWR_PROD	PRE_USE_EXPRY_PRD	
DWR_PROD	SHORT_NAME	
DWR_PROD	ACCT_EXPRY_PRD	
DWR_PROD	PROD_TYP_CD	Constant value
DWR_PROD	PROD_DESC	
DWR_PROD	PROD_CD	
DWR_PROD	PROD_NAME	
DWR_PRPD_VCHR_BTCH	VCHR_KEY	
DWR_PRPD_VCHR_BTCH	ACTVTN_DT	
DWR_PRPD_VCHR_BTCH	BTCH_CD	
DWR_PRPD_VCHR_BTCH	BTCH_NAME	
DWR_PRPD_VCHR_BTCH	LAST_UPDT_DT	
DWR_PRPD_VCHR_BTCH	LAST_UPDT_BY	
DWR_PRPD_VCHR_BTCH	END_OF_RNG	
DWR_PRPD_VCHR_BTCH	ORGNL_CNT	
DWR_PRPD_VCHR_BTCH	STRT_OF_RNG	
DWR_PRPD_VCHR_BTCH	STATE	
DWR_PRPD_VCHR_BTCH	prod_key	
DWR_PRPD_VCHR_BTCH	STRT_SRL_NBR	
DWR_PRPD_VCHR_BTCH	END_SRL_NBR	
DWR_PRPD_VCHR_INSTNC	PROD_INSTNC_CD	Prepaid voucher instance code
DWR_PRPD_VCHR_INSTNC	LAST_UPDT_DT	
DWR_PRPD_VCHR_INSTNC	LAST_UPDT_BY	
DWR_PRPD_VCHR_INSTNC	ISUD_DT	
DWR_PRPD_VCHR_INSTNC	USED_DT	
DWR_PRPD_VCHR_INSTNC	STAT	
DWR_PRPD_VCHR_INSTNC	BTCH_KEY	
DWR_PRPD_VCHR_INSTNC	VCHR_NBR	
DWR_PRPD_VCHR_INSTNC	VCHR_PIN_NBR	
DWR_PRPD_VCHR_INSTNC	prod_key	Constant value - Use key associated to cd="NCC_VOUCHER"
DWR_PRPD_VCHR_RCHRG_OPTN	VCHR_KEY	
DWR_PRPD_VCHR_RCHRG_OPTN	LAST_UPDT_DT	

Table C-2 (Cont.) Oracle Communications Data Model Tables and Columns Filled by NCC Adapter

Target Table	Target Column	Remarks
DWR_PRPD_VCHR_RCHRG_OPTN	LAST_UPDT_BY	
DWR_PRPD_VCHR_RCHRG_OPTN	ACCT_BAL_TYP_CD	
DWR_PRPD_VCHR_RCHRG_OPTN	RCHRG_OPTN_CD	
DWR_PRPD_VCHR_RCHRG_OPTN	VAL	
DWR_PRPD_VCHR_RCHRG_OPTN	BAL_EXPRY_PRD	
DWR_PRPD_VCHR_RCHRG_OPTN	STRT_DT	
DWR_PRPD_VCHR_RCHRG_OPTN	END_DT	
DWR_PRPD_VCHR_RCHRG_OPTN	NEW_BUCKET_IND	
DWR_PRPD_VCHR_RCHRG_OPTN	BAL_EXPRY_POLICY	
DWR_PRPD_VCHR_RCHRG_OPTN	RPLC_BAL_IND	
DWR_PRPD_VCHR_RCHRG_OPTN	VAL_TYP	
DWR_PRTY	prty_typ_cd	
DWR_SBRP	CNRT_TYP_CD	Lookup prod_key from dwr_prod
DWR_SBRP	BRP_CD	Load "NCCWL_" ID
DWR_SBRP	PROD_KEY	Lookup prod_key from dwr_prod

Using the Oracle Communications Billing and Revenue Management Adapter

This appendix provides information about populating an Oracle Communications Data Model warehouse using the Oracle Communications Billing and Revenue Management Adapter for Oracle Communications Data Model (BRM Adapter).

This appendix contains the following topics:

- [What is the BRM Adapter for Oracle Communications Data Model?](#)
- [Components of BRM Adapter for Oracle Communications Data Model](#)
- [BRM Adapter for Oracle Communications Data Model Execution Flows](#)
- [Populating a Warehouse Using the BRM Adapter for Oracle Communications Data Model](#)
- [Oracle GoldenGate Process Checking](#)

For information on installing and configuring the BRM Adapter for Oracle Communications Data Model, see *Oracle Communications Data Model Installation Guide*.

For detailed information about the objects and ETL associated with the BRM Adapter for Oracle Communications Data Model, see *Oracle Communications Data Model Reference*.

What is the BRM Adapter for Oracle Communications Data Model?

The Oracle Communications Billing and Revenue Management Adapter for Oracle Communications Data Model (BRM Adapter) is an option to the Oracle Communications Data Model.

The BRM Adapter loads data from an Oracle Communications Billing and Revenue Management source system into Oracle Communications Data Model. You can load data in both an initial and an incremental manner. The data from Oracle Communications Billing and Revenue Management populates the Oracle Communications Data Model derived and aggregate tables, reports, and mining models.

Components of BRM Adapter for Oracle Communications Data Model

The BRM Adapter for Oracle Communications Data Model uses the following components:

- [BRM Source Schema Tables](#)
- [BRM Adapter for Oracle Communications Data Model Database Schemas](#)

- [Schema Definitions Added for the BRM Adapter for Oracle Communications Data Model](#)
- [Oracle Data Integrator](#)
- [Oracle GoldenGate Replication](#): used for real time data feed to the staging database
- Oracle Communications Data Model

BRM Source Schema Tables

The BRM Adapter for Oracle Communications Data Model loads data from tables listed in [Table D-1](#) into Oracle Communications Data Model.

Table D-1 BRM Adapter Source Tables

BRM Adapter Source Tables

ACCOUNT_NAMEINFO_T
 ACCOUNT_T
 BAL_GRP_T
 BILL_T
 BILLINFO_T
 CONFIG_BEID_BALANCES_T
 CONFIG_BUSINESS_TYPE_T
 CONFIG_CUR_CONV_RATES_T
 CONFIG_T
 DD_OBJECTS_T
 DEAL_PRODUCTS_T
 DEAL_T
 DISCOUNT_T
 EVENT_ACTIVITY_TLCS_T
 EVENT_ACTV_TLCS_SVC_CODES_T
 EVENT_BAL_IMPACTS_T
 EVENT_BILLING_PAYMENT_CASH_T
 EVENT_BILLING_PAYMENT_CC_T
 EVENT_BILLING_PAYMENT_CHECK_T
 EVENT_BILLING_PAYMENT_DD_T
 EVENT_BILLING_PAYMENT_FAILED_T
 EVENT_BILLING_PAYMENT_PAYORD_T
 EVENT_BILLING_PAYMENT_POST_T
 EVENT_BILLING_PAYMENT_T
 EVENT_BILLING_PAYMENT_WTRAN_T
 EVENT_BROADBAND_USAGE_T
 EVENT_DLAY_ACTV_TLCS_SVC_CDS_T
 EVENT_DLAY_ACTV_TLCS_T
 EVENT_DLAY_SESS_TLCS_SVC_CDS_T
 EVENT_DLAY_SESS_TLCS_T
 EVENT_DLYD_SESSION_TLCO_GPRS_T

Table D-1 (Cont.) BRM Adapter Source Tables**BRM Adapter Source Tables**

EVENT_DLYD_SESSION_TLCO_GSM_T
 EVENT_RUM_MAP_T
 EVENT_SESS_TLCS_SVC_CODES_T
 EVENT_SESSION_DIALUP_T
 EVENT_SESSION_TELCO_GPRS_T
 EVENT_SESSION_TLCO_GSM_T
 EVENT_SESSION_TLCS_T
 EVENT_T
 EVENT_TAX_JURISDICTIONS_T
 IFW_CURRENCY
 IFW_TIMEZONE
 IFW_USAGETYPE
 INVOICE_T
 ITEM_T
 NOTE_T
 PAYINFO_CC_T
 PAYINFO_DD_T
 PAYINFO_INV_T
 PAYINFO_T
 PLAN_SERVICES_T
 PLAN_T
 PRODUCT_T
 PURCHASED_DISCOUNT_T
 PURCHASED_PRODUCT_T
 RATE_PLAN_T
 SERVICE_EMAIL_T
 SERVICE_T
 SERVICE_TELCO_FEATURES_T
 SERVICE_TELCO_GPRS_T
 SERVICE_TELCO_GSM_T

BRM Adapter for Oracle Communications Data Model Database Schemas

The BRM Adapter for Oracle Communications Data Model uses the following Database schemas:

- **Source schemas** the source system is Oracle Communications Billing and Revenue Management and database schemas is BRM_SRC (PIN).
 The specific source objects are described in "[BRM Source Schema Tables](#)" on page D-2.
- **Staging layer schema.** The BRM Adapter for Oracle Communications Data Model uses a staging area when loading data from the BRM system to foundation layer objects of an Oracle Communications Data Model warehouse. The staging layer is used for changed data and historical data storage and as a source schema to load Oracle Communications Data Model tables.

The BRM Adapter for Oracle Communications Data Model defines the staging as a database schema named `brm_stg`.

The `brm_stg` schema is described in "[BRM Adapter for Oracle Communications Data Model Staging Schema Objects](#)" on page D-4.

- **Target schema.** The target of the BRM Adapter for Oracle Communications Data Model is an Oracle Communications Data Model warehouse. The specific target objects are the foundation layer objects of the Oracle Communications Data Model. These objects are defined in the `ocdm_sys` schema which is delivered with Oracle Communications Data Model and documented in Oracle Communications Data Model Reference.

The installation of the BRM Adapter for Oracle Communications Data Model defines additional procedures in the `ocdm_sys` schema. These procedures are outlined in "[BRM Adapter for Oracle Communications Data Model OCDM_SYS Objects](#)" on page D-5.

Schema Definitions Added for the BRM Adapter for Oracle Communications Data Model

Installing and configuring the BRM Adapter for Oracle Communications Data Model as described in *Oracle Communications Data Model Installation Guide* adds the following definitions:

- A database schema named `brm_stg` that defines a staging area for use by the adapter. For information on the definitions in this database, see "[BRM Adapter for Oracle Communications Data Model Staging Schema Objects](#)" on page D-4.
- Procedure definitions that are used to populate the foundation layer of an Oracle Communications Data Model warehouse. These definitions are added to the `ocdm_sys` schema delivered with Oracle Communications Data Model. For more information on these procedures, see "[BRM Adapter for Oracle Communications Data Model OCDM_SYS Objects](#)" on page D-5.

BRM Adapter for Oracle Communications Data Model Staging Schema Objects

After the installation of BRM Adapter for Oracle Communications Data Model, a staging schema is created, (`brm_stg`). The staging schema contains the following tables, views, functions, and procedures.

Verify that these objects are in the respective BRM staging schema (`brm_stg`).

Staging Schema Tables

Corresponding to each Oracle Communications Billing and Revenue Management source table to be loaded, each of the following tables are created or updated in the staging schema:

- Normal Staging Tables (XXX) (the table name is same as source table name)
- Previous Day Tables (XXX_LD)
- Delta Tables (XXX_D)
- Delta History Tables (XXX_D_H)
- Golden Gate Option Input Parameter Table (BRM_MAPPING_TAB)
- Loading Parameter Table BRM_ETL_PARAMETER (having mainly From Date, To Date and Loading date)

Staging Schema Views

For each Oracle Communications Billing and Revenue Management source system view used in the BRM Adapter for Oracle Communications Data Model, two views are created in staging schema (brm_stg), (XXX_vw), with the names:

- EVENT_BILLING_PAYMENT_T_MAP_VW
- SERVICE_T_MAP_VW

Staging Schema Functions

For each Oracle Communications Billing and Revenue Management source system function used in the BRM Adapter for Oracle Communications Data Model (having name UTC_TO_ORACLE) is created in staging schema (brm_stg).

Staging Schema Procedures

To maintain the everyday source data changes and history of the data changes three procedures are created in staging schema (brm_stg) for both Golden Gate and Non-Golden Gate options:

- PRE_STAGING_LOAD (For Oracle Golden Gate Option)
- PRE_OCDM_LOAD (For Non - Oracle Golden Gate Option)
- POST_STAGING_LOAD (For Oracle Golden Gate Option)

BRM Adapter for Oracle Communications Data Model OCDM_SYS Objects

Installing and configuring the BRM Adapter for Oracle Communications Data Model adds the following procedure definitions to the ocdm_sys schema delivered with Oracle Communications Data Model creates procedures in ocdm_sys schema to populate the integrated BRM data.

Verify that these objects are in the Oracle Communications Data Model schema (ocdm_sys).

Procedures

- DISB_CONS
- ENAB_CONS
- INSERT_REF
- SCD2_UPDATE

Oracle Data Integrator

The BRM Adapter for Oracle Communications Data Model uses ETL created by Oracle Data Integrator (ODI) as the ETL for performing an initial load of the foundation layer objects of the Oracle Communications Data Model (that is, the base, reference, and lookup tables). You can also use ODI ETL to perform a scheduled refresh of the staging database.

For more information on the use of ODI with the BRM Adapter for Oracle Communications Data Model, see the following topics:

- ["Execution Flow Using BRM Adapter: ODI Pull Version \(without Oracle GoldenGate\)"](#) on page D-6
- ["Execution Flow Using BRM Adapter: with Oracle GoldenGate and ODI"](#) on page D-7

Oracle GoldenGate Replication

After you have performed an initial load of the foundation layer objects of the Oracle Communications Data Model using ODI, you can use Oracle GoldenGate to perform real-time feed of BRM source data to the staging database.

For more information on the use of Oracle GoldenGate with the BRM Adapter for Oracle Communications Data Model, see the following topics:

- ["Refreshing the Foundation Layer with Real-Time Data"](#) on page D-10.
- ["Execution Flow Using BRM Adapter: ODI Pull Version \(without Oracle GoldenGate\)"](#) on page D-6.

BRM Adapter for Oracle Communications Data Model Execution Flows

The BRM Adapter for Oracle Communications Data Model uses a different execution flow depending on whether you are using the adapter with Oracle GoldenGate:

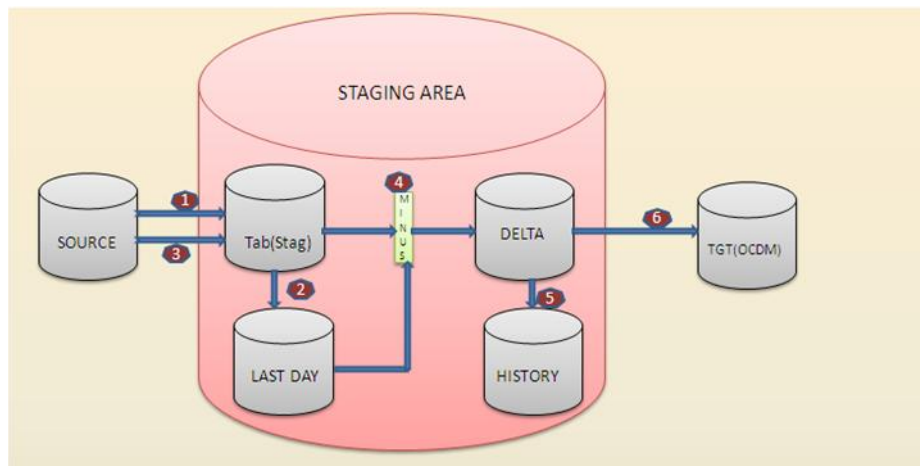
- [Execution Flow Using BRM Adapter: ODI Pull Version \(without Oracle GoldenGate\)](#)
- [Execution Flow Using BRM Adapter: with Oracle GoldenGate and ODI](#)

Execution Flow Using BRM Adapter: ODI Pull Version (without Oracle GoldenGate)

You can use the BRM Adapter for Oracle Communications Data Model with Oracle Data Integrator (ODI) alone. This method of using the BRM Adapter for Oracle Communications Data Model is the ODI-pull version.

[Figure D-1](#) illustrates the BRM Adapter for Oracle Communications Data Model execution flow when using the adapter *only* with ODI (ODI-pull version).

Figure D-1 Execution Flow When Using the Adapter without Oracle GoldenGate



The ODI-pull version gives you the option to load data at regular periods. When using the ODI-pull approach, the execution flow is as follows (as shown in [Figure D-1](#)):

1. (Steps 1 and 3) The BRM Adapter, using ODI, loads the data from the source database into stage database.

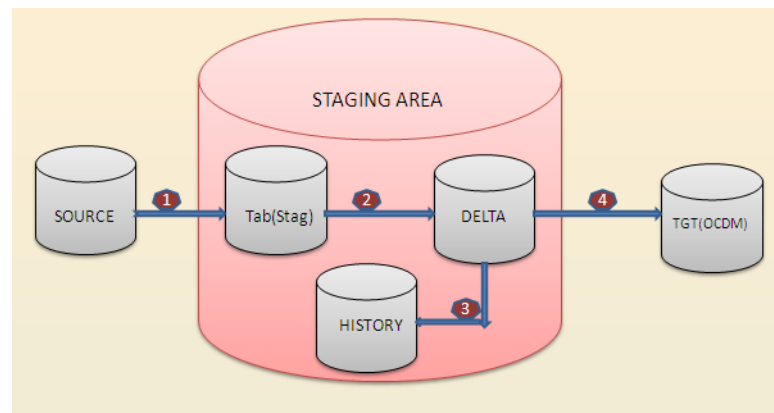
2. (Steps 2 and 5) Using pre and post procedures, the BRM Adapter loads data into several stage database tables; for example, lastday (*xxx_ld*), delta (*_delta*), and history tables.
3. (Step 4 in the diagram) The BRM Adapter loads the delta tables in the staging area using normal table 'MINUS' last day table.
4. (Step 6 in diagram) The BRM Adapter loads the data from the delta tables into foundation layer of the Oracle Communications Data Model warehouse.

Execution Flow Using BRM Adapter: with Oracle GoldenGate and ODI

When you use the BRM Adapter for Oracle Communications Data Model with Oracle GoldenGate to refresh the foundation layer of the Oracle Communications Data Model warehouse, this method allows the foundation layer data to be synchronized with the BRM source data.

Figure D-2 illustrates the BRM Adapter for Oracle Communications Data Model execution flow when using the BRM Adapter with Oracle GoldenGate.

Figure D-2 Execution Flow When Using the BRM Adapter with Oracle GoldenGate



When using Oracle GoldenGate with the BRM Adapter for Oracle Communications Data Model, the execution flow is as follows (as shown in Figure D-2):

1. Assuming that all Oracle GoldenGate processes for the BRM Adapter for Oracle Communications Data Model are running, whenever new data is placed in the source database, Oracle GoldenGate extracts the new incoming data and uses it to populate the staging tables.
2. Steps 2 and 3 in Figure D-2 (Staging area): in this part of the execution process you can hold all the changed data including the current and the historical data. The procedure `STG_OCDM_OGG_CDC_LOAD_SCENERIO_PKG` handles this process; this runs after the normal tables are updated by the Oracle GoldenGate extract process.

Note: Only the data in the delta history tables remains forever and is always in sync with source data. Once the data in the delta staging tables is loaded into the foundation layer of the Oracle Communications Data Model warehouse, the delta tables are truncated.

3. The final step Step 4 in [Figure D-2](#) is to load data from the Delta tables to the foundation layer of Oracle Communications Data Model; to do this use the ODI package `STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG` which is dependent on `STG_OCDM_OGG_CDC_LOAD_PKG`.

After the initial installation, if you are planning to use Oracle GoldenGate for incremental loads, you should first use the ODI-pull procedures described in ["Execution Flow Using BRM Adapter: ODI Pull Version \(without Oracle GoldenGate\)"](#) on page D-6 to load data from the source system to the staging system and from the staging system to Oracle Communications Data Model. After the initial load, you can then setup the incremental load to keep the Oracle Communications Data Model data synchronized with the Oracle Communications Billing and Revenue Management source system, as described in ["Refreshing the Foundation Layer with Real-Time Data"](#) on page D-10.

Populating a Warehouse Using the BRM Adapter for Oracle Communications Data Model

You use the BRM Adapter for Oracle Communications Data Model to populate or refresh the foundation layer of the Oracle Communications Data Model warehouse (that is, the base, reference, and lookup tables defined in the `ocdm_sys` schema). For more information, see:

- [Initial Loading Using the BRM Adapter for Oracle Communications Data Model](#)
- [Refreshing the Data Using the BRM Adapter for Oracle Communications Data Model \(Change Data Capture: CDC\)](#)

After you use the BRM Adapter for Oracle Communications Data Model to populate the foundation layer, you populate or refresh the access layer of the Oracle Communications Data Model (that is, the derived tables, aggregate tables, OLAP cubes, and data mining models defined in the `ocdm_sys` schema) in the same manner as you would if you were *not* using an application adapter for Oracle Communications Data Model to populate the foundation layer. For more information, see ["Performing an Initial Load of the Access Layer"](#) on page 4-11 and ["Refreshing the Access Layer of an Oracle Communications Data Model Warehouse"](#) on page 4-15.

Initial Loading Using the BRM Adapter for Oracle Communications Data Model

To perform an initial load of the foundation layer of an Oracle Communications Data Model warehouse using the BRM Adapter for Oracle Communications Data Model with Oracle Data Integrator (ODI), take the following steps:

1. Verify that the installation and configuration created the schema objects described in ["Schema Definitions Added for the BRM Adapter for Oracle Communications Data Model"](#) on page D-4.

See: *Oracle Communications Data Model Installation Guide* for instructions on installing and configuring the BRM Adapter for Oracle Communications Data Model for use by ODI.

2. Setting the `BRM_ETL_PARAMETER`:
 - a. Connect to the staging schema (for example: `brm_stg`).
 - b. Delete data from the table `BRM_ETL_PARAMETER`, then set the values:

```
FROM_DATE_ETL=01/01/1900;  
TO_DATE_ETL=current time of BRM system;
```

```
LOAD_DT=sysdate
```

For example:

```
delete from brm_etl_parameter;
commit;
Insert into BRM_ETL_PARAMETER
(PROCESS_NAME, FROM_DATE_ETL, TO_DATE_ETL, LOAD_DT)
Values ('BRM-ADAPTER', TO_DATE('01/01/1900 00:00:00', 'MM/DD/YYYY
HH24:MI:SS'), TO_DATE('05/22/2012 10:28:55', 'MM/DD/YYYY HH24:MI:SS'),
sysdate);
commit;
```

Note: you can obtain the current Oracle Communications Billing and Revenue Management time by pvt (pin_virtual_time).

3. From ODI Studio Designer, go to the BRM_OCDM project, then take the following steps:
 - a. Select **SRC_STG_NONOGG**, then **Packages**, and then, select and execute the following package:
 - **SRC_STG_LOAD_NON_OGG_PKG**
 - b. Select **STG_OCDM**, then **Packages**, and then, select and execute the following package:
 - **STG_OCDM_INITIAL_LOAD_PKG**

See: ["Execution Flow Using BRM Adapter: ODI Pull Version \(without Oracle GoldenGate\)"](#) on page D-6 for a description of the execution flow for the initial load.

Refreshing the Data Using the BRM Adapter for Oracle Communications Data Model (Change Data Capture: CDC)

Once you have performed an initial data load of an Oracle Communications Data Model warehouse, when you need to refresh the data, you use ODI to refresh all of the data in the staging area and in the foundation layer of the Oracle Communications Data Model warehouse.

Refreshing the Foundation Layer on a Scheduled Basis

If you are not using the BRM Adapter for Oracle Communications Data Model with Oracle GoldenGate to populate the foundation layer of an Oracle Communications Data Model warehouse, then you refresh the foundation layer on a scheduled basis using Oracle Data Integrator (ODI).

To refresh the data in the foundation layer using only ODI, take the following steps:

1. Verify that the installation and configuration created the schema objects described in ["Schema Definitions Added for the BRM Adapter for Oracle Communications Data Model"](#) on page D-4.

See: Oracle Communications Data Model Installation Guide for instructions on installing and configuring the BRM Adapter for Oracle Communications Data Model for use by ODI.

2. Reset the BRM_ETL_PARAMETER table parameter values:
 - a. Connect to the staging schema (for example: brm_stg).

Source System Processes	Staging System Processes
Manager process	Manager process
Extract process (EXTBRM)	Replicate process (REPBRM)
Extract Pump process (EXTPBRM)	

The following code shows the issuing of these commands with successful results.

```
GGSCI> (mypc1) 5> info all

Program      Status      Group      Lag          Time Since Chkpt
-----
MANAGER      RUNNING
EXTRACT      RUNNING    EXTBRM     47:29:00    00:00:20
EXTRACT      RUNNING    EXTPBRM    00:00:00    47:29:06

GGSCI> (ocdm01) 2> info all

Program      Status      Group      Lag          Time Since Chkpt
-----
MANAGER      RUNNING
REPLICAT     RUNNING    REPBRM     00:00:00    00:03:09
```

Tips: If you have two source systems, check process status on both source systems.

For commands to manage Oracle GoldenGate processes, see *Oracle Communications Data Model Installation Guide*.

3. To perform Step 2, 3, 4 as shown in [Figure D-2](#), using ODI run the following package: STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG.

From ODI Studio Designer, go to the BRM_OCDM project, then take the following steps:

- a. Select **STG_OCDM**, then select **Packages**, and then, select and execute the **STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG**.

See: See ["Execution Flow Using BRM Adapter: with Oracle GoldenGate and ODI"](#) on page D-7 for an explanation of the execution flow.

Note: The STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG is a scenario based package has a package and a SCD2_UPDATE procedure; this package is based on another packaged named STG_OCDM_OGG_CDC_LOAD_PKG which consists of interfaces and a procedure PRE_OCDM_LOAD. The STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG waits for data to perform data load from the staging system to the target OCDM_SYS

ODI BRM_OCDM Project

This section describes the internal operations performed by each executable object mentioned in ["Initial Loading Using the BRM Adapter for Oracle Communications Data Model"](#) on page D-8 and ["Refreshing the Data Using the BRM Adapter for Oracle Communications Data Model \(Change Data Capture: CDC\)"](#) on page D-9.

Objects in SRC_STG_NONOGG Folder of BRM_OCDM Project

SRC_STG_LOAD_NON_OGG_PKG Package

The SRC_STG_LOAD_NON_OGG_PKG package in SRC_STG_NONOGG folder of BRM_OCDM project is using to load data into BRM staging database from BRM source database. Following steps were included in this process of loading staging database.

1. Execute PRE_STAGING_LOAD procedure to load previous day tables(XXX_LD). This procedure performs following operations:
 - a. Drop Previous Day Tables (XXX_LD)
 - b. Rename normal staging tables (XXX) to previous day tables (XXX_LD)
 - c. Create normal staging tables (XXX) structure from previous day tables (XXX_LD)

(Drop XXX_LD -> rename XXX to XXX_LD ->Create XXX from XXX_LD).
2. Execute Non Golden Gate Interfaces (XXX_NONGG) one by one to load normal staging tables (XXX).

There are 101 Non Golden Gate Interfaces out of which 51 have the naming convention (XXX_NONGG) and the other 51 use the naming convention (XXX_NONGG_IU); these are executed in sequence to load Staging tables form Source tables (src.XXX to stg.XXX).

SRC_STG_NONGG Interfaces:

1. ACCOUNT_NAMEINFO_T_NONGG_IU
2. ACCOUNT_NAMEINFO_T_NONGG
3. ACCOUNT_T_NONGG_IU
4. ACCOUNT_T_NONGG
5. BAL_GRP_T_NONGG_IU
6. BAL_GRP_T_NONGG
7. BILLINFO_T_NONGG_IU
8. BILLINFO_T_NONGG
9. BILL_T_NONGG_IU
10. BILL_T_NONGG
11. CONFIG_T_NONGG_IU
12. CONFIG_T_NONGG
13. CONFIG_BEID_BALANCES_T_NONGG_IU
14. CONFIG_BEID_BALANCES_T_NONGG
15. CONFIG_BUSINESS_TYPE_T_NONGG_IU
16. CONFIG_BUSINESS_TYPE_T_NONGG
17. CONFIG_CUR_CONV_RATES_T_NONGG_IU
18. CONFIG_CUR_CONV_RATES_T_NONGG
19. DD_OBJECTS_T_NONGG_IU
20. DD_OBJECTS_T_NONGG
21. DEAL_PRODUCTS_T_NONGG_IU
22. DEAL_PRODUCTS_T_NONGG
23. DEAL_T_NONGG_IU
24. DEAL_T_NONGG
25. DISCOUNT_T_NONGG_IU
26. DISCOUNT_T_NONGG
27. EVENT_BAL_IMPACTS_T_NONGG_IU
28. EVENT_BAL_IMPACTS_T_NONGG
29. EVENT BILLING PAYMENT CASH_T_NONGG_IU
30. EVENT BILLING PAYMENT CASH_T_NONGG
31. EVENT BILLING PAYMENT_CC_T_NONGG_IU
32. EVENT BILLING PAYMENT_CC_T_NONGG
33. EVENT BILLING PAYMENT CHECK_T_NONGG_IU

34. EVENT_BILLING_PAYMENT_CHECK_T_NONGG
 35. EVENT_BILLING_PAYMENT_DD_T_NONGG_IU
 36. EVENT_BILLING_PAYMENT_DD_T_NONGG
 37. EVENT_BILLING_PAYMENT_FAILED_T_NONGG_IU
 38. EVENT_BILLING_PAYMENT_FAILED_T_NONGG
 39. EVENT_BILLING_PAYMENT_PAYORD_T_NONGG_IU
 40. EVENT_BILLING_PAYMENT_PAYORD_T_NONGG
 41. EVENT_BILLING_PAYMENT_POST_T_NONGG_IU
 42. EVENT_BILLING_PAYMENT_POST_T_NONGG
 43. EVENT_BILLING_PAYMENT_T_NONGG_IU
 44. EVENT_BILLING_PAYMENT_T_NONGG
 45. EVENT_BILLING_PAYMENT_WTRAN_T_NONGG_IU
 46. EVENT_BILLING_PAYMENT_WTRAN_T_NONGG
 47. EVENT_BROADBAND_USAGE_T_NONGG_IU
 48. EVENT_BROADBAND_USAGE_T_NONGG
 49. EVENT_SESSION_DIALUP_T_NONGG_IU
 50. EVENT_SESSION_DIALUP_T_NONGG
 51. EVENT_SESSION_TELCO_GPRS_T_NONGG_IU
 52. EVENT_SESSION_TELCO_GPRS_T_NONGG
 53. EVENT_SESSION_TLCO_GSM_T_NONGG_IU
 54. EVENT_SESSION_TLCO_GSM_T_NONGG
 55. EVENT_SESSION_TLCS_T_NONGG_IU
 56. EVENT_SESSION_TLCS_T_NONGG
 57. EVENT_SESS_TLCS_SVC_CODES_T_NONGG_IU
 58. EVENT_SESS_TLCS_SVC_CODES_T_NONGG
 59. EVENT_T_NONGG_IU
 60. EVENT_T_NONGG
 61. IFW_CURRENCY_NONGG_IU
 62. IFW_CURRENCY_NONGG
 63. IFW_TIMEZONE_NONGG_IU
 64. IFW_TIMEZONE_NONGG
 65. IFW_USAGETYPE_NONGG_IU
 66. IFW_USAGETYPE_NONGG
 67. INVOICE_T_NONGG_IU
 68. INVOICE_T_NONGG
 69. ITEM_T_NONGG_IU
 70. ITEM_T_NONGG
 71. NOTE_T_NONGG_IU
 72. NOTE_T_NONGG
 73. PLAN_SERVICES_T_NONGG_IU
 74. PLAN_SERVICES_T_NONGG
 75. PLAN_T_NONGG_IU
 76. PLAN_T_NONGG
 77. PRODUCT_T_NONGG_IU
 78. PRODUCT_T_NONGG
 79. PURCHASED_DISCOUNT_T_NONGG_IU
 80. PURCHASED_DISCOUNT_T_NONGG
 81. PURCHASED_PRODUCT_T_NONGG_IU
 82. PURCHASED_PRODUCT_T_NONGG
 83. RATE_PLAN_T_NONGG_IU
 84. RATE_PLAN_T_NONGG
 85. SERVICE_EMAIL_T_NONGG_IU
 86. SERVICE_EMAIL_T_NONGG
 87. SERVICE_TELCO_FEATURES_T_NONGG_IU
 88. SERVICE_TELCO_FEATURES_T_NONGG
 89. SERVICE_TELCO_GPRS_T_NONGG_IU
 90. SERVICE_TELCO_GPRS_T_NONGG
 91. SERVICE_TELCO_GSM_T_NONGG_IU
 92. SERVICE_TELCO_GSM_T_NONGG
 93. SERVICE_T_NONGG_IU

- 94. SERVICE_T_NONGG
- 95. PAYINFO_T_NONGG_IU
- 96. PAYINFO_T_NONGG
- 97. PAYINFO_CC_T_NONGG_IU
- 98. PAYINFO_CC_T_NONGG
- 99. PAYINFO_DD_T_NONGG_IU
- 100. PAYINFO_DD_T_NONGG
- 101. PAYINFO_INV_T_NONGG_IU
- 102. PAYINFO_INV_T_NONGG

3. Execute POST_STAGING_LOAD procedure to load the data into Delta Tables (XXX_DELTA) and Delta History Tables (XXX_DELTA_H).
 - a. Truncate delta tables (XXX_DELTA)
 - b. Insert data into delta tables (XXX_DELTA) from normal staging tables (XXX) and previous day tables (XXX_LD).
 - c. Update delta tables (XXX_DELTA) columns as per the source changes (OPTYPE, LAST_UPDATE_TIME & CHANGE_DATE)
 - d. Insert data into delta history tables(XXX_DELTA_H) from delta tables (XXX_DELTA)

(Truncate XXX_DELTA -> Insert into XXX_DELTA from (XXX minus XXX_LD) -> Insert into XXX_DELTA_H from XXX_DELTA)

SRC_STG_LOAD_NON_OGG_PKG Package

The SRC_STG_LOAD_NON_OGG_PKG package in SRC_STG_NONOGG folder of BRM_OCDM project is using to load the staging delta and delta history tables. This package will call the PRE_OCDM_LOAD procedure and execute in an infinite loop. In each loop of iteration, it performs the following steps and loads the data into Delta Tables (XXX_DELTA) and Delta History Tables (XXX_DELTA_H).

1. Insert data into delta history tables(XXX_DELTA_H) from delta tables (XXX_DELTA)
2. Truncate delta tables (XXX_DELTA)
3. Insert data into delta tables (XXX_DELTA) from normal staging tables (XXX)
4. Wait for new data in any XXX_DELTA tables, ODI uses an tool called ODI WAIT FOR DATA which will be in the waiting state; the job of this Tool (ODI WAIT FOR DATA) will be waiting for any data changes in staging layer delta tables (XXX_DELTA).Once it finds any changes it will execute the concerned interface where the data will be populated into related OCDM foundation layer tables

(Insert into XXX_DELTA_H from XXX_DELTA and Truncate XXX_DELTA and Insert into XXX_DELTA from XXX)

Objects in STG_OCDM Folder of BRM_OCDM

STG_OCDM_INITIAL_LOAD_PKG Package

The STG_OCDM_INITIAL_LOAD_PKG package in STG_OCDM folder of BRM_OCDM project loads the OCDM foundation layer tables from BRM Staging Database delta tables (XXX_DELTA). The package STG_OCDM_INITIAL_LOAD_PKG executes 46 STG-OCDM interfaces (XXX_MAP) which are executed in sequence, where the data is populated into related OCDM foundation layer tables form staging delta tables (XXX_DELTA).

STG-OCDM Interfaces:

1. ACCOUNT_ACCT_MAP
2. ACCOUNT_ADDR_LOC_MAP
3. ACCOUNT_CUST_MAP
4. ACCOUNT_GEO_CITY_MAP
5. ACCOUNT_GEO_CNTRY_MAP
6. ACCOUNT_GEO_STATE_MAP
7. ACCOUNT_INVC_DLVRY_MAP
8. ACCOUNT_POSTCD_MAP
9. ACCOUNT_PRTY_CNCT_INFO_MAP
10. ACCOUNT_PRTY_MAP
11. BAL_GRP_GRP_MAP
12. BILLINFO_ACCT_MAP
13. BILLINFO_PYMT_MTHD_TYP_MAP
14. BILL_INVC_MAP
15. CONFIG_BEID_BAL_TYP_MAP
16. CONFIG_BUSINESS_TYPE_T_MAP
17. CONFIG_CUR_CRNCY_EXCHNG_RATE_MAP
18. DD_OBJECTS_SRVC_SPEC_MAP
19. DEAL_PKG_ASGN_MAP
20. DEAL_PROD_MAP
21. EVENT_BAL_IMPACTS_NTWK_EVT_ACCT_BAL_IMPC_MAP
22. EVENT BILLING ACCT_RCHR_MAP
23. EVENT_BROADBAND_USAGE_BRDBND_USG_EVT_MAP
24. EVENT_CALL_EVT_MAP
25. EVENT_DATA_SRVC_EVT_MAP
26. EVENT_SESSION_GPRS_USG_EVT_MAP
27. IFW_CRNCY_MAP
28. IFW_TIMWZONE_TIME_MAP
29. IFW_USAGETYPE_SRVC_TYP_MAP
30. INVOICE_INVC_DLVRY_MAP
31. ITEM_INVC_ITEM_MAP
32. NOTE_EVT_PRTY_INTRACN_MAP
33. PAYINFO_ACCT_PREF_PYMT_MTHD_MAP
34. PAYINFO_CC_PRTY_MAP
35. PAYINFO_DD_PRTY_MAP
36. PAYINFO_INV_T_PRTY_MAP
37. PLAN_MKT_PLN_MAP
38. PLAN_SERVICES_PROD_MKT_PLN_ASGN_MAP
39. PRODUCT_DISCOUNT_PROD_MAP
40. PRODUCT_DISCOUNT_PROD_RLTN_MAP
41. PURCHASED_PRODUCT_PRICE_CHRG_MAP
42. PURCHASED_PRODUCT_SBRP_MAP
43. RATE_PLAN_PROD_RTNG_PLN_MAP
44. SERVICE_CUST_FCNG_SRVC_MAP
45. SERVICE_SRVC_STAT_HIST_MAP
46. SERVICE_TELCO_FEATURES_SRVC_CHTRSTC_MAP

STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG Package

The following mentioned STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG package in STG_OCDM folder of BRM_OCDM project executes the OCDM foundation layer tables population interfaces and these packages needs to be executed and should be in running mode.

Note: STG_OCDM_OGG_CDC_LOAD_SCENARIO_PKG is a package which is having a package STG_OCDM_OGG_CDC_LOAD_PKG and a SCD2_UPDATE procedure, the scenario package is based on STG_OCDM_OGG_CDC_LOAD_PKG which consist of 46 interfaces and a procedure PRE_OCDM_LOAD. The STG_OCDM_

OGG_CDC_LOAD_SCENARIO_PKG will be in an Infinite loop process which will wait for data to perform data load from stage(BRM_STG) to target (OCDM_SYS)

Oracle GoldenGate Process Checking

If you are using Oracle GoldenGate, verify that Oracle GoldenGate is working so that the Adapter can load data into the staging system. The processes listed should be running on both the source systems and the staging system as shown:

Source Processes

- Manager Process
- Extract Process (EXTBRM)
- Extract Pump Process (EXTPBRM)

For example, on the BRM_SRC Source System, the source processes should look as follows:

```
GGSCI> (mypc1) 5> info all
```

Program	Status	Group	Lag	Time Since Chkpt
MANAGER	RUNNING			
EXTRACT	RUNNING	EXTBRM	47:29:00	00:00:20
EXTRACT	RUNNING	EXTPBRM	00:00:00	47:29:06

Staging Processes

```
GGSCI> (ocdm01) 2> info all
```

Program	Status	Group	Lag	Time Since Chkpt
MANAGER	RUNNING			
REPLICAT	RUNNING	REPBRM	00:00:00	00:03:09

Tips: If you have two source systems, check process status on both source systems. For commands to manage Oracle GoldenGate processes, see *Oracle Communications Data Model Installation Guide*.

Upgrading Oracle Communications Data Model

This appendix provides information about how to upgrade Oracle Communications Data Model. It contains the following topics:

- [Considerations for Upgrading Oracle Communications Data Model](#)
- [Upgrade Paths for Oracle Communications Data Model](#)
- [Regression Testing for Oracle Communications Data Model](#)

Note: This appendix provides general information about how to upgrade the Oracle Communications Data Model.

Considerations for Upgrading Oracle Communications Data Model

With the next release of Oracle Communications Data Model, identify the components in Oracle Communications Data Model that must be upgraded and analyze the impact that the upgrade will have on your existing data warehouse architecture and applications. The Oracle Communications Data Model upgrade focuses on Oracle Communications Data Model itself -- it does not include considerations of upgrading a hardware platform or related software.

You should consider the following points before you begin your upgrade:

- Review the System Requirements and Supported Platforms for Oracle Communications Data Model to make sure your database platform versions are supported.
- The upgrade process is not only a technical process. It requires significant planning and involvement from many teams, including development, database administrators, business analysts, QA, and so on.
- You should determine what customization was made to your existing Oracle Communications Data Model system before you begin the upgrade process. It is recommended that you budget sufficient time (in the upgrade project) for detailed review of current customizations and relevance of the customizations for the new version of the Oracle Communications Data Model system to which you plan to upgrade.
- You will also need to analyze the impact of the schema changes on your current custom implementation. The extent of your customizations will have an impact on the length of time required for the upgrade.

- Depending on the release from which you are upgrading, moving the customizations in your existing repositories, reports, and dashboards to the new version may require a multi-step process and may involve manual processes at some stages.
- It is highly recommended that you use side-by-side environments when performing each stage of the upgrade process. Enabling side-by-side instances of the entire Oracle Communications Data Model environment is a critical success factor for upgrade. For some stages of the upgrade, you can upgrade your environments in place. However, for comparison and benchmarking purposes, it is recommended that you upgrade using side-by-side environments.
- For each stage of the upgrade process, you need to allocate a reasonable amount of time to validate the results of that stage and address any problems. In addition, final user acceptance testing must confirm that the entire upgrade process was successful.

Upgrade Paths for Oracle Communications Data Model

There are two different paths to upgrade Oracle Communications Data Model, depending on whether Oracle Communications Data Model has been customized.

Upgrade Path A: Based on an Earlier Default Oracle Communications Data Model

Upgrade Oracle Communications Data Model based on an earlier default Oracle Communications Data Model.

Upgrade Path B: Based on a Customized Oracle Communications Data Model

Take the following steps to upgrade your Oracle Communications Data Model based on a customized Oracle Communications Data Model:

1. Review your customization patch and upgrade it if necessary.
2. Execute the (new) customization patch.
3. Migrate the data.
4. Additions, if necessary, to modify the:
 - Derived, aggregate layer
 - Source-ETL and intra-ETL mapping
 - Workflow
 - Business intelligence applications
 - Metadata Repository

Regression Testing for Oracle Communications Data Model

After you upgrade Oracle Communications Data Model, perform regression testing. The major tasks to perform are:

- Create a test plan.
- Execute a test case.
- Test error handling.
- Create a final report.

Index

A

access layer, 2-2
 customizing, 3-1
 Oracle Communications Data Model, 2-3, 3-1
access layer of Oracle Communications Data Model
 populating, 4-11, 4-15
accounts
 for Oracle Communications Data Warehouse, B-1
AGGR_FLW, 4-10
aggregate tables
 in Oracle Communications Data Model, 3-9
 refreshing, 4-10
application adapters
 for Oracle Communications Data Model, C-1, D-1
 Oracle Communications Data Model, 4-2, 4-14
As Is reports, 5-8
As Was reports, 5-8

B

Billing and Revenue Management Adapter
 Oracle Communications Data Model, D-1
BRM Adapter for Oracle Communications Data
 Model, D-5
 introduction to, D-1
 populating using, D-8
 schemas, D-3
 with ODI, D-5
 with Oracle GoldenGate, D-6
brm_stg schema, D-4

C

Churn Prediction Data Mining Model, 3-4
compression
 in Oracle Communications Data Model, 2-8
 materialized views, 3-25
configuring Oracle Communications Data Model
 warehouse, A-3
conventions
 when customizing physical model, 2-4
cubes
 adding materialized view capabilities to, 3-16
 changing the dimensions of, 3-18
 changing the measures of, 3-18

 customizing, 3-16
 data maintenance methods, 3-19
 forecast, 3-18
 in Oracle Communications Data Model, 3-16,
 3-17
 partitioning, 3-18
customizing
 access layer, 3-1
 cubes, 3-16
 Oracle Communications Data Model, 1-3
 physical data model, 2-1

D

dashboards, Oracle Communications Data
 Model, 5-2, 5-13
data governance committee, responsibilities of, 1-5
data mining models
 customizing, 3-2
 refreshing, 4-10
definitions added to ocdm_sys, C-9, D-5
derived tables
 in Oracle Communications Data Model, 3-2
 refreshing, 4-10
dimensional components, Oracle Communications
 Data Model, 3-10
DRVD_FLW, 4-10

E

error handling
 during intra-ETL execution, 4-17
ETL for Oracle Communications Data Model, 4-1

F

fit-gap analysis for Oracle Communications Data
 Model, 1-10
forecast cube in Oracle Communications Data
 Model, 3-18
foundation layer
 defined, 2-2
 Oracle Communications Data Model, 2-3
foundation layer of Oracle Communications Data
 Model
 common change scenarios, 2-6

populating, 4-11, 4-14

H

HCC, 2-9

hybrid columnar compression
and Oracle Communications Data Model, 2-9

I

implementers of Oracle Communications Data Model
prerequisite knowledge, 1-4

implementing
Oracle Communications Data Model, 1-3

indexes
in Oracle Communications Data Model, 2-11
materialized views, 3-23
partitioning, 2-12

integrity constraints
in Oracle Communications Data Model, 2-11

INTRA_ETL_FLW process flow, 4-9

intra-ETL
customizing, 4-9
executing, 4-13
managing errors, 4-17
monitoring execution of, 4-17
Oracle Communications Data Model, 4-1
recovery, 4-18
troubleshooting, 4-18

J

join performance, improving, 2-13

K

keys, surrogate
in Oracle Communications Data Model, 2-10

L

loading Oracle Communications Data Model
data, 4-11

M

materialized views
compressing, 3-25
in Oracle Communications Data Model, 3-20
indexing, 3-23
partition change tracking, 3-24
partitioning, 3-23
refresh options
refreshing
materialized views, 3-21

Metadata Dependency Manager
with Oracle Communications Data Model, 1-9

metadata management
repository, 1-7
with Oracle Communications Data Model, 1-6
metadata repository, 1-7

browsing, 1-7
with Oracle Communications Data Model, 1-7
MINING_FLW, 4-10

N

naming conventions
for physical model of Oracle Communications
Data Model, 2-4
NCC Adapter for Oracle Communications Data
Model, C-9
components of, C-8
introduction to, C-1
ODI project, C-10
populating using, C-3
schemas, C-2
with ODI, C-2
with Oracle GoldenGate, C-2
NCC source schema, C-8
NCC_OCDM project, C-10
ncc_stg schema, C-8
Network Charging and Control Adapter
Oracle Communications Data Model, C-1
Network Charging and Control Adapter for Oracle
Communications Data Model, 4-2

O

ocdm_sys schema
definitions added by BRM Adapter for Oracle
Communications Data Model, D-5
definitions added by NCC Adapter for Oracle
Communications Data Model, C-9
ODI
BRM Adapter for Oracle Communications Data
Model, D-5
NCC Adapter for Oracle Communications Data
Model, C-2
ODI project
NCC Adapter for Oracle Communications Data
Model, C-10
Oracle Communications Data Model
access layer, 2-3, 3-1
accounts for, B-1
application adapters, 4-2, 4-14
Billing and Revenue Management Adapter, D-1
components of, 1-1
customizing, 1-3
customizing physical model, 2-1, 2-3, 2-4, 2-7
dashboards, 5-2
data governance, 1-5
described, 1-1
dimensional components, 3-10
ETL, 4-1
fit-gap analysis, 1-10
foundation layer, 2-3
implementing, 1-3
intra-ETL, 4-1
loading, 4-11
Metadata Dependency Manager, 1-9

- metadata management, 1-6
- metadata repository, 1-7
- Network Charging and Control Adapter, C-1
- Oracle products used by, 1-2
- Oracle Warehouse Builder, using with, 1-9
- physical layers of, 2-2
- populating using BRM Adapter for Oracle Communications Data Model, D-8
- populating using NCC Adapter for Oracle Communications Data Model, C-3
- pre-implementation tasks, 1-4
- querying, 5-3
- refreshing data, 4-14
- refreshing with real-time data, C-4, C-6, D-9
- refreshing with real-time data for BRM Adapter, D-7
- reporting, 5-1, 5-3
- source-ETL, 4-1, 4-2, 4-4, 4-5
- staging layer, 2-3
- tablespaces, design recommendations, 2-7
- upgrading, E-1
- user privileges, B-1
- Oracle Communications Data Model implementers prerequisite knowledge for, 1-4
- Oracle Communications Data Model warehouse
 - configuring, A-3
 - sizing, A-1
- Oracle data mining models
 - Oracle Communications Data Model, 3-2
- Oracle GoldenGate
 - BRM Adapter for Oracle Communications Data Model, D-6
 - NCC Adapter for Oracle Communications Data Model, C-2
- Oracle Warehouse Builder
 - with Oracle Communications Data Model, 1-9

P

- parallel execution
 - enabling for a session, 2-16
 - enabling for DML operations, 2-16
 - in Oracle Communications Data Model, 2-14
- partition change tracking, 3-24
- partition exchange load, 4-8
- partitioned indexes in Oracle Communications Data Model, 2-11
- partitioning
 - cubes, 3-18
 - for easier data access, 2-13
 - for join performance, 2-13
 - for manageability, 2-13
 - for source-ETL, 4-8
 - indexes, 2-12
 - materialized views, 3-23
 - tables, 2-12
- partitions, changing, 2-7
- physical model of Oracle Communications Data Model
 - characteristics of, 2-1, 2-3, 2-4

- customizing, 2-4
- general recommendations for, 2-7

Q

- querying Oracle Communications Data Model, 5-3

R

- real-time data
 - populating with, C-4, C-6, D-9
 - populating with for BRM Adapter, D-7
- refreshing
 - aggregate tables, 4-10
 - data mining models, 4-10
 - derived tables, 4-10
 - Oracle Communications Data Warehouse, 4-14
- reporting
 - Oracle Communications Data Model, 5-1, 5-3
- reports
 - approaches to, 5-1
 - As Is, 5-8
 - As Was, 5-8
 - troubleshooting performance, 5-6
- reports, Oracle Communications Data Model
 - creating new, 5-16

S

- sample reports
 - customizing
 - Oracle Communications Data Model
 - sample reports, 5-2
- schemas
 - BRM source, D-3
 - brm_stg, D-3, D-4
 - NCC source, C-2, C-8
 - ncc_stg, C-2, C-8
 - ocdm_sys, C-2, D-3
 - staging, C-8, D-4
- sizing
 - Oracle Communications Data Model
 - warehouse, A-1
- source-ETL
 - exception handling, 4-6
 - jobs control, 4-5
 - loading considerations, 4-6
 - Oracle Communications Data Model, 4-1, 4-2, 4-4, 4-5, 4-6
 - parallel direct path load, 4-8
 - partitioning for, 4-8
 - using application adapter to create, 4-2, 4-14
 - workflow, 4-5
- staging layer, 2-1
 - Oracle Communications Data Model, 2-3
- star queries, optimizing, 5-4
- subtypes
 - defining tables for, 2-10
 - physical implementation of, 2-10
- supertypes

- defining tables for, 2-10
- physical implementation of, 2-10
- surrogate keys
 - in Oracle Communications Data Model, 2-10

T

- tables
 - aggregate, 3-9
 - compressing, 2-8
 - derived, 3-2
 - partitioning, 2-12
- tablespace in Oracle Communications Data Model, 2-7

U

- upgrading Oracle Communications Data Warehouse, E-1
- user privileges in Oracle Communications Data Warehouse, B-1