Oracle® Retail Data Model

Operations Guide

10*g* Release 2 (10.2)

E14480-02

August 2009

ORACLE®

Oracle Retail Data Model Operations Guide, 10*g* Release 2 (10.2)

E14480-02

Oracle Retail Data Model is based on the ARTS 5.0 standard.

Contents

Preface v

Audience v
Documentation Accessibility v
Related Documents vi
Conventions vi

1 Introducing Oracle Retail Data Model

What is Oracle Retail Data Model? 1-1
Scope of Retail Organizations That Oracle Retail Data Model Supports 1-2
Oracle Products That Make Up Oracle Retail Data Model 1-2
What Are the Components of Oracle Retail Data Model 1-3
Oracle Retail Data Model Logical and Physical Models 1-4
Logical Model 1-5
Physical Model 1-5
Where Oracle Retail Data Model Fits in a Data Warehousing Project 1-5

2 Introduction to Customizing Oracle Retail Data Model

Prerequisite Knowledge for Customizers 2-1
Performing Fit-Gap Analysis 2-1
Overview: Customization Steps 2-2
Dependencies When Customizing the Physical Model 2-3

3 Populating the Oracle Retail Data Model Warehouse

Overview: Populating an Oracle Retail Data Model Warehouse 3-1
Populating Reference, Lookup, and Base Relational Tables 3-2
Populating Derived and Aggregate Relational Tables and Views 3-2
Executing the Intra-ETL for Oracle Retail Data Model 3-3
Executing the Intra-ETL in Oracle Warehouse Builder 3-3
Import the ORDM_INTRA_ETL Project 3-4
Configure the ORDM_INTRA_ETL Project 3-4
Prepare to Execute the Project 3-5
Deploy and Execute the Project 3-5
Explicitly Executing the Intra-ETL Package 3-6
Monitoring the Execution of the Intra-ETL Process 3-6
Recovering an Intra_ETL Process 3-8

Implementing Oracle Retail Data Model Data Mining Models 3-9
Populating the Data Mining Source Tables 3-9
Creating the Data Mining Models 3-11
Populating OLAP Cubes 3-13

4 Analysis and Reporting in Oracle Retail Data Model

Reports Delivered with Oracle Retail Data Model 4-1
Sample Associate Basket Analysis Model Report 4-2
Sample Associate Loss Analysis Model Reports 4-3
Sample Associate Sales Analysis Model Report 4-5
Sample Customer Product Category Mix Analysis Model Report 4-6
Sample Customer Loyalty Analysis Model Report 4-7
Sample Item Basket Analysis Model Report 4-8
Sample Item Point of Sale (POS) Loss Analysis Model Report 4-9
Customizing the Reports Delivered with Oracle Retail Data Model 4-10
Writing Your Own Queries and Reports on Relational Data 4-11
Writing Your Own Queries and Reports on OLAP Cube Data 4-11
Writing Reports on Time-Series and Ranking Analysis 4-12
Writing Forecasting Reports 4-12

5 Maintaining an Oracle Retail Data Model Warehouse

Overview: Maintaining an Oracle Retail Data Model 5-1
Maintaining Relational Tables and Views 5-1
Refreshing OLAP Cube Data 5-2
Sample Incremental Load 5-2
OLAP Incremental Load Recovery 5-3
Updating Forecast Cubes 5-7
Creating a New Forecast 5-7

A Operations Scripts

Calendar Population Script A-1
Bitmap Index for Fact Tables Script A-2
Partition Append Scripts A-2
Create Dimensions Script A-3
Foreign Key Manipulation Scripts A-3
Lookup Value Population Scripts A-4
The Out of Stock Script A-4
RFMP Calculation Script A-5

Index

Preface

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

Audience

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

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

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

Documentation Accessibility

Accessibility of Links to External Web Sites in Documentation

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

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at `http://www.fcc.gov/cgb/consumerfacts/trs.html`, and a list of phone numbers is available at `http://www.fcc.gov/cgb/dro/trsphonebk.html`.

Related Documents

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

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

Conventions

The following text conventions are used in this document:

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

1 Introducing Oracle Retail Data Model

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

- What is Oracle Retail Data Model?
- Scope of Retail Organizations That Oracle Retail Data Model Supports
- Oracle Products That Make Up Oracle Retail Data Model
- What Are the Components of Oracle Retail Data Model
- Where Oracle Retail Data Model Fits in a Data Warehousing Project

What is Oracle Retail Data Model?

Oracle Retail Data Model is a startup kit for implementing a retail business intelligence solution. It is a standards-based data model, designed and pre-tuned for Oracle data warehouses, including the HP Oracle Database Machine.

The Oracle Retail Data Model for Retail offers a single-vendor solution package that is tightly integrated with the business intelligence platform. With pre-built data mining, On-line Analytical Processing (OLAP) and dimensional models, Oracle Retail Data Model provides you with industry-specific metrics and insights that you can act on immediately to improve your bottom line. These BI solution offerings take advantage of Oracle's scalability and reliability, using Oracle's familiar optimization, parallelism, and performance engineering within the database.

Oracle Retail Data Model can be used in any application environment and is easily extendable.

By leveraging Oracle’s strong retail domain expertise, Oracle Retail Data Model provides an industry standard compliant foundation schema that is modern, relevant, topical, and addresses needs of most retail segments. This normalized foundation schema serves as a detailed and structured representation of the retail business, providing an integrated base for business information with fully defined entities and relationships. Oracle Retail Data Model includes an exhaustive set of embedded advanced analytics, using Oracle's OLAP and data mining technology. You can take advantage of pre-built and pre-tested solution sets designed by industry experts that deliver relevant insights, are actionable, and aimed at improving both top-line and bottom-line results. You can see summarized, aggregated information or quickly navigate to drill-down transaction details to better understand business issues. For example, with Oracle Retail Data Model’s out-of-the-box sample reports, merchandisers gain improved insight into product affinities; loss prevention specialists gain improved visibility; and marketing analysts gain improved understanding of promotional effectiveness and customer segmentation. You can add your own reports as well. Oracle Retail Data Model, combined with Oracle technology, provides all of the components required for a complete and extendable Retail Data Warehouse and Business Intelligence framework in order to eliminate complex and costly integration requirements, all designed to reduce your total cost of ownership.

Oracle Retail Data Model is a pre-built, pre-tested solution designed by industry experts to help retailers maximize the value of their Oracle data warehouse. Using sophisticated trending and data mining capabilities based on Oracle's OLAP and data mining technology, retailers - including grocery stores, department stores, specialty store chains, mass merchants, convenience stores, and multi-channel retailers - now have the data analysis capabilities to develop retail-specific insights that are relevant, actionable, and can improve both top-line and bottom-line results.

With Oracle Retail Data Model, you can jump-start the design and implementation of a retail data warehouse to quickly achieve a positive ROI for your data warehousing and business intelligence project with a predictable implementation effort

Scope of Retail Organizations That Oracle Retail Data Model Supports

Oracle Retail Data Model provides a broad base for supporting retail operations. It is geared especially for general merchandise and grocery, but it can also support other types of retail, such as hard lines, soft lines, and multi-channel retailers.

Oracle Retail Data Model provides a data architecture and data model along with reports and key performance indicators to support multiple business areas in retail organizations:

- Store Operations
- Point-of-Sale (POS)
- Loss Prevention
- Merchandising
- Inventory
- Workforce Management
- Order Management
- Customer
- Category Management
- Promotion

Oracle Products That Make Up Oracle Retail Data Model

Several Oracle technologies are involved in building the infrastructure for retail business intelligence.

Oracle Database with OLAP, Data Mining and Partitioning Option

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

> **Tip:** To save some money, you can consider using RAC and commodity hardware.

Oracle Development Tools

The following Oracle tools can be used to customize the predefined logical and physical models provided with ORDM, or to populate the target relational tables,mmaterialized views, or OLAP cubes.

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

Name	Use
Designer	To create the logical model
SQL Developer or SQL*Plus	To create or modify database objects
Oracle Warehouse Builder	For the process control of the intra ETL process
Analytic Workspace Manager	To populate the target OLAP cubes

Oracle BI EE Presentation Tools

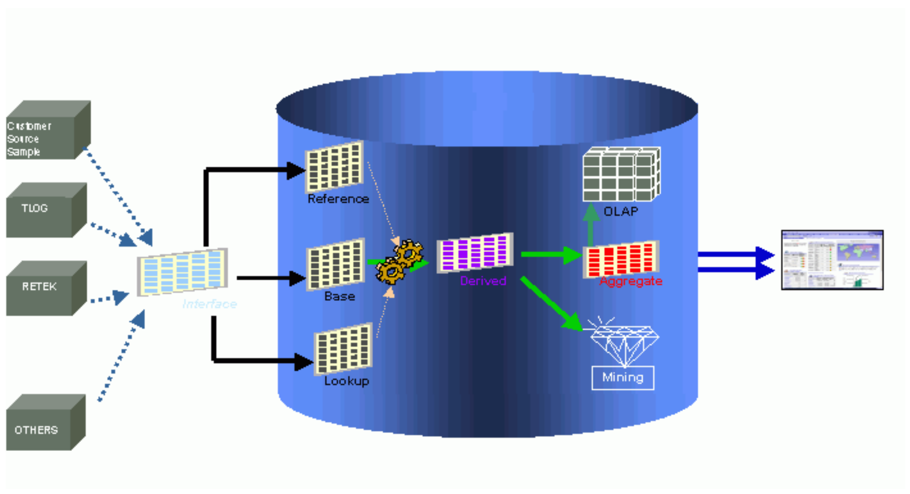
Oracle Business Intelligence Suite Enterprise Edition (Oracle BI EE) is a comprehensive suite of enterprise BI products that delivers a full range of analysis and reporting capabilities. You can use Oracle BI EE Answers and Dashboard presentation tools to customize the predefined sample dashboard reports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

> **See:** Chapter 4, "Analysis and Reporting in Oracle Retail Data Model".

What Are the Components of Oracle Retail Data Model

The Oracle Retail Data Model consists of logical and physical data models, intra-ETL that maps your OLTP tables to the Oracle Retail Data Model tables and views, sample reports, and the Oracle Interactive Dashboard using features of Oracle Business Intelligence Suite Enterprise Edition (OBIEE).

Figure 1–1 Oracle Retail Data Model



ORDM includes the following components:

- Logical model

The logical model is introduced in "Oracle Retail Data Model Logical and Physical Models" on page 1-4 and described in detail in *Oracle Retail Data Model Reference*.

- Physical model

  The physical model is introduced in "Oracle Retail Data Model Logical and Physical Models" on page 1-4 and described in detail in *Oracle Retail Data Model Reference*.

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

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

- Pre-defined data mining models.

  These models are described in detail in *Oracle Retail Data Model Reference*. How to create these models is discussed in "Implementing Oracle Retail Data Model Data Mining Models" on page 3-9.

- Sample reports and dashboards using OBIEE.

  These reports are discussed in "Reports Delivered with Oracle Retail Data Model" on page 4-1.

- DDL and installation scripts

Oracle Retail Data Model Logical and Physical Models

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

- Retail industry-specific, 3rd Normal Form logical and physical relational models
- Physical Data model with 650+ tables and 10500+ attributes
- Data warehouse models (based on Association for Retails Technology Standards (ARTS)
- Industry specific Measures
- Pre-built OLAP cubes

consist of multiple layers, with the details stored in the Base and Reference layers, and summary data stored in the Aggregate Layer and Analytical Workspace. The Derived layer stores information derived from the Base and Reference layers. Lookup tables are also included to store frequently used descriptive data.

Oracle Retail Data Model consists of enterprise data warehouse logical and physical components:

- Base models (in 3rd Normal Form, transaction detail storage)
- Reference models (describing the people, places and things within the retail organization)
- Derived models (variation of the base model; could have minor aggregations, which store state or status information, or a combination of objects from the base model and data mining objects)
- Aggregate models (aggregations of base and derived data for analytical purposes, including relational materialized views and OLAP). Aggregate data is stored in

the form of relational materialized views. These materialized views are aggregates of the Base and Derived layers

Logical Model

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

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

Physical Model

Oracle Retail Data Model provides a predefined physical data model.

The physical data model of the Oracle Retail Data Model is the physical manifestation of the logical data model into database tables and relationships (or foreign key constraints). Partitions, indexes, and materialized views have been added to aid performance.

The physical data model includes the following:

- Reference tables
- Lookup tables
- Database sequences
- Base tables
- Derived tables
- Aggregate tables and relational materialized views
- (When the optional OLAP component is installed) Multi--dimensional OLAP cubes

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

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

DWR_ : Reference data tables

DWL_ : Lookup tables

DWB_ : Base transaction data (3NF) tables

DWD_ : Derived (data mining) tables

DWA_ : Aggregate (ROLAP and MOLAP) tables

_MV : Materialized view

Where Oracle Retail Data Model Fits in a Data Warehousing Project

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

Each retail operation is unique, and therefore the structure of the data warehouse will need to be different in order to match the needs of that unique retail environments.

Oracle Retail Data Model comes with a generic schema that requires modification. These modifications include, adding, deleting, modifying, or renaming tables and columns; or altering foreign keys, constraints, or indexes. These changes must be made to the foundation data warehouse, not in the Oracle Retail Data Model schema itself.

After the data warehouse is populated, populate the derived and aggregate layers to support the reporting requirements. Oracle Retail Data Model includes a solid infrastructure for a range of reports.

2 Introduction to Customizing Oracle Retail Data Model

This chapter provides an overview of how you customize Oracle Retail Data Model. It contains the following topics:

- Prerequisite Knowledge for Customizers
- Performing Fit-Gap Analysis
- Overview: Customization Steps
- Dependencies When Customizing the Physical Model

Prerequisite Knowledge for Customizers

As discussed in "Oracle Products That Make Up Oracle Retail Data Model" on page 1-2, Oracle Retail Data Model uses much of the Oracle stack. Consequently, to successfully customize Oracle Retail Data Model, you need:

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

Performing Fit-Gap Analysis

The Fit-Gap analysis is where you compare your information needs and retail business requirements with the structure that is available "out of the box" with Oracle Retail Data Model. You identify any required functionality that is not included in the default schema, as well as other modifications that are necessary to meet your requirements.

The result of your fit-gap analysis is a customization report which is a brief explanation of the adaptations and adjustments required to customize Oracle Retail Data Model to fit your retail environment.

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

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

1. If previous evaluations have been performed, review the documentation from the previous phases, and if necessary add team members with the needed business and technical expertise.
2. Meet to review the data and maps your data structure data into Oracle Retail Data Model's schema.
3. Produce a list of what people are going to try to do with the system (examples rather than models), and create use cases for appraising the functionality of Oracle Retail Data Model.

   Procedures are written based on the use cases. Keep in mind that deviations from the procedure can be useful, provided that functionality is not skipped.
4. Map your business procedures against Oracle Retail Data Model functions, noting which processes are not available in Oracle Retail Data Model, or work differently in it. Be sure to check security requirements.
5. Determine the differences are between your needs and Oracle Retail Data Model's schema and discusses the following:
   - Which differences can you live with, and which must be reconciled?
   - What can you do about the differences you can't live with?
6. Based on the preceding steps, update the business process models, activity flow diagrams, entity object model, and object life cycle models to reflect the customized system.
7. Write the customization report, detailing what changes will be required to make Oracle Retail Data Model's schema match your business needs. This includes any interfaces to existing systems, and additions and changes to Oracle Retail Data Model.
8. Based on the customization report, update the Project Plan, and complete a phase section for the Logical Design phase.

Overview: Customization Steps

Customizing Oracle Retail Data Model involves the following tasks:

1. Perform fit-gap analysis as described in "Performing Fit-Gap Analysis" on page 2-1.
2. In a development environment, make a copy of Oracle Retail Data Model.
3. Working in the copy you created in Step 2, make changes to the Oracle Retail Data Model components. Document all of your changes. Make the changes in the following order:
   a. Logical model
   b. Logical to physical mappings
   c. Physical model. Keep in mind the issues discussed in "Dependencies When Customizing the Physical Model" on page 2-3.
   d. ETL. Keep in mind the issues discussed in Chapter 3, "Populating the Oracle Retail Data Model Warehouse".
4. In a test environment, make a copy of Oracle Retail Data Model.
5. Following the documentation you created in when performing your fit-gap analysis, customize Oracle Retail Data Model and test the customized version.

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

Dependencies When Customizing the Physical Model

The physical model of Oracle Retail Data Model is implemented as layered components, where the structure and data of one component is dependent on another. Consequently, make your changes in the following order:

1. Base tables
2. Reference tables
3. Lookup tables
4. Derived tables
5. Aggregate tables
6. Materialized views
7. Analytic workspace
    a. Dimensions and levels
    b. Measures
    c. Calculations (including aggregations and forecasts)

3 Populating the Oracle Retail Data Model Warehouse

This chapter describes how Extract, Transform, and Load (ETL) operations populate Oracle Retail Data Model with data.

This chapter discusses the following topics:

- Overview: Populating an Oracle Retail Data Model Warehouse
- Populating Reference, Lookup, and Base Relational Tables
- Populating Derived and Aggregate Relational Tables and Views
- Implementing Oracle Retail Data Model Data Mining Models
- Populating OLAP Cubes

Once you have performed ETL operations to implement an Oracle Retail Data Model data warehouse, you need to update with new data from your OLTP system. The process for doing this is discussed in Chapter 5, "Maintaining an Oracle Retail Data Model Warehouse".

Note: The instructions in this chapter assume that after doing the fit-gap analysis described in "Performing Fit-Gap Analysis" on page 2-1, you have not identified or made any changes to the Oracle Retail Data Model logical or physical model. If you have made changes, you need to modify the ETL accordingly.

Overview: Populating an Oracle Retail Data Model Warehouse

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

As with any data warehouse, you use Extract, Transform, and Load (ETL) operations to populate an Oracle Retail Data Model data warehouse. You perform ETL operations as three separate steps using three different types of ETL:

1. Source-ETL processes that extract data from the source On-Line Transaction Processing (OTLP) system, transform that data, and loads the reference, lookup, and base tables Oracle Retail Data Model warehouse. Source-ETL is *not* provided with Oracle Retail Data Model. *You must write source-ETL processes yourself.* For

information about creating source-ETL, see "Populating Reference, Lookup, and Base Relational Tables" on page 3-2.

2. Intra-ETL processes that populate the remaining Oracle Retail Data Model warehouse relational data structures. Intra-ETL does not access the OLTP data at all. All of the data that intra-ETL extracts and transforms is located within the Oracle Retail Data Model warehouse. Intra-ETL is provided with the Oracle Retail Data Model and is executed in the following order:

    a. Intra-ETL that populates the derived and aggregate tables and materialized views with data from the base, reference, and lookup tables. For information about using this intra-ETL, see "Populating Derived and Aggregate Relational Tables and Views" on page 3-2.

    b. Intra-ETL that populates the tables used for the data mining models. For more information on using this intra-ETL, see "Implementing Oracle Retail Data Model Data Mining Models" on page 3-9

3. SQL scripts that populate the OLAP cubes provided with Oracle Retail Data Model. These scripts define the OLAP cubes and populate these cubes with data extracted from the Oracle Retail Data Model relational tables and views. For more information on populating OLAP cubes in a Oracle Retail Data Model warehouse, see "Populating OLAP Cubes" on page 3-13.

Populating Reference, Lookup, and Base Relational Tables

You populate the reference, lookup, and base tables with data from the source On-Line Transaction Processing (OTLP) applications using source-ETL.

Source-ETL is *not* provided with Oracle Retail Data Model. You must design and write the source-ETL processes yourself. When writing these ETL processes, populate the tables in the following order:

1. Reference tables
2. Lookup tables
3. Base tables

> **See:** Appendix A, "Operations Scripts" for information about scripts provided with Oracle Retail Data Model that can help you implement your physical data model. For example, "Lookup Value Population Scripts" on page A-4 provides instructions for using two SQL scripts provided with Oracle Retail Data Model to seed values into physical lookup tables.

Populating Derived and Aggregate Relational Tables and Views

One component of Oracle Retail Data Model is a database package named `PKG_INTRA_ETL_PROCESS` which is a complete Intra-ETL process composed of Intra-ETL scripts operations that populate the derived and aggregate tables and relational materialized views with the data from the base, reference, and lookup table. This package respects the dependency of each individual program. It executes the programs in the proper order.

> **See also:** The Intra-ETL scripts are discussed in detail in *Oracle Retail Data Model Reference*.

There are two categories of Intra-ETL scripts:

- Derived Population - These are a set of database packages that populate the derived tables based on the content of the base, reference, and lookup tables. Derived tables are implemented using Oracle tables.
- Aggregate Population - These are a set of database packages that populate the aggregate tables that are mainly implemented as materialized views based on the content of the previously populated Oracle tables.

Note: The Intra-ETL scripts provided with Oracle Retail Data Model assume that there is no data in the derived tables and aggregate tables and views. Typically, you perform this type of load only when you first create your data warehouse. Later, you need to add additional data to the tables and refresh your views. In this case, you perform an incremental load as described in "Maintaining Relational Tables and Views" on page 5-1.

Using the Intra-ETL involves the following tasks:

- Executing the Intra-ETL for Oracle Retail Data Model
- Monitoring the Execution of the Intra-ETL Process
- Recovering an Intra_ETL Process

Executing the Intra-ETL for Oracle Retail Data Model

There are two ways that you can execute the Intra-ETL packages provided with Oracle Retail Data Model. The method you use depends on whether you answered "yes" or "no" to the question "Indicate if this installation will be used to store transaction level history" when you installed Oracle Retail Data Model:

- If you selected "yes" during installation, then Level0 is MV and you can execute the Intra-ETL using Oracle Warehouse Builder as discussed in "Executing the Intra-ETL in Oracle Warehouse Builder" on page 3-3.
- If you selected "no" during installation, then Level0 is Table and you must explicate execute the Intra-ETL package as described in "Explicitly Executing the Intra-ETL Package" on page 3-6.

Executing the Intra-ETL in Oracle Warehouse Builder

You can install Oracle Retail Data Model Intra-ETL as a project in Oracle Warehouse Builder (OWB). Once installed, you can execute the intra-ETL from OWB.

See: For information about Oracle Warehouse Builder, see *Oracle Warehouse Builder User's Guide*.

To use Oracle Retail Data Model Intra-ETL in Oracle Warehouse Builder (OWB), follow these steps:

1. Import the ORDM_INTRA_ETL Project.
2. Configure the ORDM_INTRA_ETL Project.
3. Prepare to Execute the Project.
4. Deploy and Execute the Project.

This installation requires Oracle Warehouse Builder 10.2.0.1.0.

Import the ORDM_INTRA_ETL Project Follow these steps to import the ORDM_INTRA_ETL project:

1. Log in as a repository user to the to the OWB design center where you want to import ORDM_INTRA_ETL. In this example, the repository user is RBIA_ETL.
2. Click the **Design** menu, and select **Import**.
3. Click the submenu **Warehouse Builder Metadata**. The Metadata Import window opens.
4. Enter the name and location of the exported metadata loader file (`.mdl`), `Intra_ETL_OWB_ProcessFlow.mdl` in the folder `*ORACLE_HOME*/ORDM/ORDM/PDM/Relational/Intra_ETL/OWB`.
5. Enter a log file name and location. The log file enables monitoring of import operations.
6. Don't change any other options.
7. Click **Import**.
8. After the import finishes, ORDM_INTRA_ETL appears in the OWB Design Center in the Project Explorer.

Configure the ORDM_INTRA_ETL Project Follow these steps to configure the imported ORDM_INTRA_ETL project:

1. Create a metadata location in OWB for which the corresponding schema is the database schema where all Oracle Retail Data Model-related objects are available and installed. For example, create a metadata location named ETL_DEMO_LOC under the ETL_DEMO schema where all Oracle Retail Data Model-related objects are available.
2. Right-click the data module ORDM_DERIVE_AGGREGATE, and select **Open Editor**.
3. Change the **Metadata Location** and the **Data Location** of the data module of the imported project to the location defined in the first step. In this example, change the location to ETL_DEMO_LOC.
4. Right-click the data module ORDM_DERIVE_AGGREGATE and select **Configure**.
5. In the **Identification** property, change **Location** and **Streams Administrator location** to the location created in the first step. In this example, change both items to ETL_DEMO_LOC.
6. Create an Oracle Workflow location in OWB in a workflow schema. For example, create the Oracle Workflow location OWF_ET_DEMO_LOC in the OWF_MGR schema.
7. Right-click the process flow module (in this example, ORDM_INTRA_ETL), and select **Open Editor**.
8. In the editor, change the **Data Location** of the process flow module to the new location created in step 6. In this example, change the **Data Location** for ORDM_INTRA_ETL to OWF_ET_DEMO_LOC.
9. Right-click the process flow module, and select **Configure**. In this example, right-click ORDM_INTRA_ETL.
10. In the Configuration Properties,
    - In **Execution**, change **Evaluation Location** to the new location created in step 6.

- In **Identification**, change **Location** to the new location created in step 6.

  In this example, change both values to OWF_ET_DEMO_LOC.

11. Log in to Design Center as OWB owner user. In **Global Explorer** -> **Security** -> **Users**, right- click **Users** and click **new**. Select the two users corresponding to the new locations. Click **Next** and then **Finish**.
12. Save the project.

Prepare to Execute the Project You are now almost ready to execute the project. Before you can execute the project, ensure that the repository user (in this example, RBIA_ETL) has the EXECUTE privilege for the following packages:

- PKG_AGGREGATE_ALL
- PKG_DWD_CTLG_RQST_BY_DAY
- PKG_DWD_POS_RTL
- PKG_DWD_POS_CNTRL
- PKG_DWD_POS_STORE_FINCL
- PKG_DWD_CUST_EMP_RLTNSHP_DAY
- PKG_DWD_INV_POSN_BY_ITEM_DAY
- PKG_DWD_CERTIFICATE_ACTVTY_TRX
- PKG_DWD_CUST_ORDR_LI_STATE
- PKG_DWD_RTV_ITEM_DAY
- PKG_DWD_CUST_ORDR_ITEM_DAY
- PKG_DWD_CUST_SKU_SL_RETRN_DAY
- PKG_DWD_INV_ADJ_BY_ITEM_DAY
- PKG_DWD_INV_UNAVL_BY_ITEM_DAY
- PKG_DWD_SPACE_UTLZTN_ITEM_DAY
- PKG_DWD_POS_TNDR_FLOW
- PKG_DWD_RTL_SL_RETRN_ITEM_DAY
- PKG_INTRA_ETL_UTIL

These packages are listed on the Transformation node of the data module. In this example, they are listed on the Transformation node of ORDM_DERIVE_AGGREGATE.

Deploy and Execute the Project Follow these steps to deploy and execute the main process flow:

1. Go to the **Control Center Manager**.
2. Select the Oracle Workflow location that was created in Configure the ORDM_INTRA_ETL Project. In this example, the location is OWF_ET_DEMO_LOC.
3. Select the main process flow RBIA_INTRA_ETL_FLW. Right-click and select **set action**. If this is the first deployment, set action to **Create**; for deployment after the first, set action to **Replace**. Deploy the process flow.
4. After the deployment finishes successfully, RBIA_INTRA_ETL_FLW is ready to execute.

Tips

Keep the following in mind:

- Insure that you specify the date ranges in the `DWC_ETL_PARAMETER` table for `PROCESS_NAME 'RBIA-INTRA-ETL'` before you trigger the ETL process.
- Insure that time partitions for the load period have been created:
  - To generate partition for L0 MV, use `ORACLE_HOME/ORDM/PDM/Relational/Intra_ETL/L0_MV/generate_add_partition.sql`
  - To generate partition for L0 Tables, use `ORACLE_HOME/ORDM/PDM/Relational/Intra_ETL/L0_Table/generate_add_partition.sql`

Explicitly Executing the Intra-ETL Package

Oracle Retail Data Model provides you with a `PKG_INTRA_ETL_PROCESS.RUN` procedure which starts the Oracle Retail Data Model Intra-ETL process. This procedure can be invoked manually, by another process such as Source-ETL, or according to a predefined schedule such as Oracle Job Scheduling.

`PKG_INTRA_ETL_PROCESS.RUN` does not accept parameters. This procedure calls other programs in the correct order to load the data for current day (according to the Oracle system date). The result of each table loading are tracked in `DWC_` control tables.

Tips

Keep the following in mind:

- Insure that you specify the date ranges in the `DWC_ETL_PARAMETER` table for `PROCESS_NAME 'RBIA-INTRA-ETL'` before you trigger the ETL process.
- Insure that time partitions for the load period have been created:
  - To generate partition for L0 MV, use `ORACLE_HOME/ORDM/PDM/Relational/Intra_ETL/L0_MV/generate_add_partition.sql`
  - To generate partition for L0 Tables, use `ORACLE_HOME/ORDM/PDM/Relational/Intra_ETL/L0_Table/generate_add_partition.sql`

Monitoring the Execution of the Intra-ETL Process

Two control tables, `DWC_INTRA_ETL_PROCESS` and `DWC_INTRA_ETL_ACTIVITY`, monitor the execution of the Intra-ETL process.

- Table 3–1 on page 3-6 contains column information for DWC_INTRA_ETL_PROCESS.
- Table 3–2 on page 3-7 contains column information for DWC_INTRA_ETL_ACTIVITY.

Table 3–1 DWC_INTRA_ETL_PROCESS Columns

Column Name	Data Type and Size	Not NULL?	Remarks
PROCESS_KEY	NUMBER(30)	Yes	Primary Key. System Generated Unique Identifier

Table 3–1 (Cont.) DWC_INTRA_ETL_PROCESS Columns

Column Name	Data Type and Size	Not NULL?	Remarks
PROCESS_START_ TIME	DATE	Yes	ETL Process Start Date and Time
PROCESS_END_ TIME	DATE	ETL Process End Date and Time	
PROCESS_STATUS	VARCHAR2(30)	Yes	Current Status of the process
FROM_DATE_ETL	DATE	Start Date (ETL) - From Date of the ETL date range	
TO_DATE_ETL	DATE	End Date (ETL) - To Date of the ETL date range	
LOAD_DT	DATE	Record Load Date - Audit Field	
LAST_UPDT_DT	DATE	Last Update Date and Time - Audit Field	
LAST_UPDT_BY	VARCHAR2(30)	Last Update By - Audit Field	

Table 3–2 DWC_INTRA_ETL_ACTIVITY Columns

Column Name	Data Type and Size	Not NULL?	Remarks
ACTIVITY_KEY	NUMBER(30)	Yes	Primary Key. System Generated Unique Identifier
PROCESS_KEY	NUMBER(30)	Yes	Process Key. FK to DWC_INTRA_ETL_ PROCESS table
ACTIVITY_NAME	VARCHAR2(50)	Yes	Activity Name or Intra ETL Program Name
ACTIVITY_DESC	VARCHAR2(500)	Activity Description	
ACTIVITY_START_ TIME	DATE	Yes	Intra-ETL Program Execution Start Time
ACTIVITY_END_ TIME	DATE	Intra-ETL Program Execution End Time	
ACTIVITY_STATUS	VACHAR2(30)	Yes	Current Status of the individual program
ERROR_DTL	VARCHAR2(2000)	Error details, if any	
LOAD_DT	DATE	Record Load Date - Audit Field	
LAST_UPDT_DT	DATE	Last Update Date and Time - Audit Field	
LAST_UPDT_BY	VARCHAR2(30)	Last Update By - Audit Field	

At the top level, the complete Intra-ETL process is divided into two groups: 1) Derived Population and, 2) Aggregate Population. The programs are executed in that order (that is, Aggregate Population is invoked only when *all* of the individual programs that make up the Derived Population complete with a status of COMPLETED-SUCCESS).

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 table `DWC_INTRA_ETL_PROCESS` with SYSDATE (a unique, monotonically-increasing, system-generated process key) as process start time, `RUNNING` as the process status, and input date range in the fields `FROM_DATE_ETL` and `TO_DATE_ETL`.
2. Invokes each of the individual Intra-ETL programs in the appropriate order of dependency. Before the invocation of each program, the procedure inserts a record into the Intra-ETL Activity detail table `DWC_INTRA_ETL_ACTIVITY` with a system generated unique activity key, the process key value corresponding to the Intra-ETL process, individual program name as the `Activity Name`, a suitable activity description, `SYSDATE` as activity start time, `RUNNING` as the activity status.
3. Updates the corresponding record in the `DWC_INTRA_ETL_ACTIVITY` table for the activity end time and activity status after the completion of each individual ETL program (either successfully or with errors. For successful completion of the activity, the procedure updates the status as `COMPLETED-SUCCESS`. If an error occurs, the procedure updates the activity status as `COMPLETED-ERROR`, and also updates the corresponding error detail in the `ERROR_DTL` column.

   In case of an error, all of the DML operations performed as part of the individual ETL programs are rolled back and the corresponding error details are updated into the activity table record, along with the status indicating that an error occurred. Within a group, an error in one program does *not* stop the execution of the other subsequent independent programs; only the programs which are dependent of the program which has encountered error are skipped. However, as explained earlier, since Aggregate Population will only start after the successful completion of the entire Derived Population, an error in any individual program in the Derived Population group results in the skipping of the execution of the Aggregate Population set of programs and MV refresh
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. If 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. This can done both during and after the execution of the Intra-ETL procedure.

Recovering an Intra_ETL Process

When an error occurs, the corresponding error details are tracked against the individual programs in the `DWC_INTRA_ETL_ACTIVITY` table.

> **See:** "Monitoring the Execution of the Intra-ETL Process" on page 3-6

In order to restart the Intra-ETL operation, you must:

1. Check and correct (either for data correction or database-related action) each of these errors (which may be related to data or database size).
2. Invoke the Intra-ETL process as an error-recovery run using the `PKG_INTRA_ETL_PROCESS.RECOVERYRUN` procedure instead of the database package used for a normal run.

   The `PKG_INTRA_ETL_PROCESS.RECOVERYRUN` procedure identifies the programs that failed during the previous execution based on the content of the `DWC_INTRA_ETL_ACTIVITY` table, and will execute only those programs as a part of the recovery run. In the case of Derived Population error as a part of the previous run, this recovery run executes the individual derived population programs which produced errors in the previous run. After their successful completion, the run executes the Aggregate Population programs and materialized view refresh in the appropriate order. In this way, the Intra-ETL error recovery is almost transparent, without involving the Data Warehouse or ETL administrator. The administrator only needs to take correct the causes of the errors and re-invoke the Intra-ETL process once more. The Intra-ETL process identifies and executes the programs that generated errors.

Implementing Oracle Retail Data Model Data Mining Models

Oracle Retail Data Model provides an optional data mining component. This data mining component extends the core extends the core functionality of Oracle Retail Data Model by adding data mining models.

The tables used by the data mining models are defined and populated by Intra-ETL process provided with Oracle Retail Data Model.

The the steps required to implement the data mining models are:

1. Populating the Data Mining Source Tables
2. Creating the Data Mining Models

Populating the Data Mining Source Tables

When you install the Data Mining component of Oracle Retail Data Model, the data mining ETL packages are also installed. The Mining ETL packages have names of the form `PKG_POP_DM_*` and are listed in *Oracle Retail Data Model Reference*.

You use these Mining ETL packages to populate data from base or derived tables in the `bia_rtl` schema to tables named `*_SRC` in the `bia_rtl_mining` schema. These `*_SRC` contain source input data for the data mining models.

To populate the *_SRC tables:

1. Ensure that the corresponding base and derived tables in the `bia_rtl` schema are populated.

   Note: For testing purposes, you can create populated tables by importing the `*_SRC` tables from `bia_rtl_mining.dmp.zip` which is installed during the Oracle Retail Data Model install option 3 "Sample Schema and Reports." The zip file is located in *ORACLE_HOME*`/ORDM/Sample/Schema_Dump`.

2. Execute the `PKG_POP_DM_*.LOADDATA(`*`p_yearmonth`*`)` procedures.

The parameter `p_yearmonth` is the Business Month that you want to analyze. All Business Months are stored in `bia_rtl.DWR_BSNS_MO` . The input of *`p_yearmonth`* must be in `bia_rtl.DWR_BSNS_MO.MO_KEY`.

Note: The programs that load the data also perform required data preparation.

Example 3–1 Populating Source input Data Tables for the Data Mining Models

This example populates data for Business Year 2007 Month 2:

1. Log in to the database as the BIA_RTL_MINING user:

```
$ sqlplus bia_rtl_mining/bia_rtl_mining
```

2. Execute packages to load data from base and derived table in the `bia_rtl` schema to the mining source tables in the `bia_rtl_mining` schema:

```
--Populate ASSOCIATE_BASKET_SRC
SQL>exec pkg_pop_dm_assbas.loaddata(20070219);

--Populate ASSOCIATE_LOSS_SRC
SQL>exec pkg_pop_dm_assloss.loaddata(20070219);

--Populate ASSOCIATE_SALES_SRC
SQL>exec pkg_pop_dm_asssls.loaddata(20070219);

--Populate STR_CTGRY_DTLS_SRC
SQL>exec pkg_pop_dm_custcatgmix.loaddata(20070219);

--Populate CUSTOMER_LOYALTY_SRC
SQL>exec pkg_pop_dm_custlty.loaddata(20070219);

--Populate FS_CATEGORY_MIX_SRC
SQL>exec pkg_pop_dm_fscatgmix.loaddata(20070219);

--Populate ITEM_BASKET_SRC
SQL>exec pkg_pop_dm_itmbas.loaddata(20070219);

--PopulateI ITEM_POS_LOSS_SRC
SQL>exec pkg_pop_dm_itmposloss.loaddata(20070219);

--Populate for POS_FLOW_SRC
SQL>exec pkg_pop_dm_posflow.loaddata(20070219);

--Popule STORE_LOSS_SRC
SQL>exec pkg_pop_dm_strloss.loaddata(20070219);
```

3. Check source table to see if data was loaded:

```
select 'ASSOCIATE_BASKET_SRC',count(*) from ASSOCIATE_BASKET_SRC where month =
'BY 2007 M2' union
select 'ASSOCIATE_LOSS_SRC  ',count(*) from ASSOCIATE_LOSS_SRC where month =
'BY 2007 M2'  union
select 'ASSOCIATE_SALES_SRC ',count(*) from ASSOCIATE_SALES_SRC where month =
'BY 2007 M2' union
select 'STR_CTGRY_DTLS_SRC  ',count(*) from STR_CTGRY_DTLS_SRC where month =
'BY 2007 M2'  union
select 'CUSTOMER_LOYALTY_SRC',count(*) from CUSTOMER_LOYALTY_SRC where month =
'BY 2007 M2' union
```

```
select 'FS_CATEGORY_MIX_SRC ',count(*) from FS_CATEGORY_MIX_SRC where month =
'BY 2007 M2'  union
select 'ITEM_BASKET_SRC     ',count(*) from ITEM_BASKET_SRC where month = 'BY
2007 M2'     union
select 'ITEM_POS_LOSS_SRC   ',count(*) from ITEM_POS_LOSS_SRC where month = 'BY
2007 M2'   union
select 'POS_FLOW_SRC        ',count(*) from POS_FLOW_SRC where month = 'BY 2007
M2'        union
select 'STORE_LOSS_SRC      ',count(*) from STORE_LOSS_SRC where month = 'BY
2007 M2'     ;
```

Check the result to ensure that the source tables for which you will create mining model contains data.

Creating the Data Mining Models

Oracle Retail Data Model creates mining models using the following three Oracle Data Mining algorithms: Adaptive Bayes (ABN), Decision Tree (DT), and Apriori Association (APASS). These algorithms all build models that have rules.

Note: In Oracle Data Mining, "Association Rules" is abbreviated as AR.

The mining models are defined and populated using packages that are named `PKG_RBIW_DM_*` provided with Oracle Retail Data Model. Each package (analysis) builds models using one or two of the algorithms. The models built depend on the analysis being performed. The output of the model build is a view containing rules generated by the model.

To create a data mining model:

1. Log in to the database as the `BIA_RTL_MINING` user.
2. Execute the appropriate `PKG_RBIW_DM_*` package with the appropriate build procedure to create the mining model of the model type(s) you want.

   See: For a complete list of the data mining model creation packages, the build procedures and build parameters for creating different models of different types, see "Data Mining Component ETL" in *Oracle Retail Data Model Reference*.

3. Check that `RBIW_DM_MODEL_SIGN` and `RBIW_DM_RULES` were populated with new rules with the following query.
4. Check that a model-specific view was created with the new rules for the specific model you created.

   See: For a complete list of these views, see "Physical Data Model of the Data Mining Component" in *Oracle Retail Data Model Reference*.

Example 3–2 Creating a Data Mining Model

Before you build models, ensure that the `*_SRC` tables are populated. For testing purposes, you can create populated tables by importing the `*_SRC` tables from `bia_rtl_mining.dmp.zip`. The zip file `bia_rtl_mining.dmp.zip` is installed during the Oracle Retail Data Model install option 3 "Sample Schema and Reports." The zip file is located in the directory

```
ORACLE_HOME/ORDM/Sample/Schema_Dump
```

These steps show how to run the Mining Model Creation packages. In this example, we create an Associate Basket Analysis model for Business Year 2007 Month 2.

1. Log in to the database as the BIA_RTL_MINING user:

   ```
   $ sqlplus bia_rtl_mining/bia_rtl_mining
   ```

2. Execute packages to create the Associate Basket Analysis Model for the two model types ABN and DT:

   ```
   --For the model type Adaptive Bayes Network (ABN)
   SQL> set serveroutput on size 1000000
   spool ASSBAS_ABN.txt
   BEGIN
       PKG_RBIW_DM_ASSBAS.PRC_RUNALL_ABN('ASSOCIATE_BASKET_SRC',
                 'ASSBAS_MDL_ABN',
                 2007,
                 'BY 2007 M2',
                 TRUE
       );
   END;
   /

   spool off
   --For the model type Decision Tree (DT)
   SQL> set serveroutput on size 1000000
   spool ASSBAS_DT.txt
   BEGIN
       PKG_RBIW_DM_ASSBAS.PRC_RUNALL_DT (
                 'ASSOCIATE_BASKET_SRC',
                 'ASSBAS_MDL_DT',
                 2007,
                 'BY 2007 M2',
                 TRUE
       );
   END;
   /
   spool off
   ```

3. Check that RBIW_DM_MODEL_SIGN and RBIW_DM_RULES were populated with new rules with the following query:

   ```
   SQL>
   Select 'SIGN' TYPE, analysis_desc, model_type, count(*) COUNT from RBIW_DM_
   MODEL_SIGN where analysis_name = 'ASSOCIATE_BASKET' group by analysis_desc,
   model_type
   union
   Select 'RULES' TYPE, analysis_desc, model_type, count(*) COUNT from RBIW_DM_
   RULES where analysis_name = 'ASSOCIATE_BASKET' group by analysis_desc, model_
   type
   order by 2,3;
   ```

   The query should return information like the following:

TYPE	ANALYSIS_DESC	MODEL_TYPE	COUNT
RULES	Associate Basket Analysis	ABN	61
SIGN	Associate Basket Analysis	ABN	18

TYPE	ANALYSIS_DESC	MODEL_TYPE	COUNT
RULES	Associate Basket Analysis	DT	76
SIGN	Associate Basket Analysis	DT	39

The actual COUNT values may be different from the ones shown; however, if the model was created successfully, the COUNT should always be greater than 0.

Populating OLAP Cubes

The OLAP component is an option of Oracle Retail Data Model. This OLAP component extends the core functionality of Oracle Retail Data Model by adding OLAP cubes for OLAP analysis and forecasting.

Oracle Retail Data Model OLAP cubes are *not* populated using a formal Extract, Transform, and Load workflow process. Instead, OLAP cubes are populated through SQL scripts that use the `RBIA_OLAP_ETL_AW_LOAD` package that is provided with the OLAP component.

> **See:** The `RBIA_OLAP_ETL_AW_LOAD` package is documented in *Oracle Retail Data Model Reference*.

OLAP cubes are populated at the following times:

1. During the initial load of the OLAP cubes.

   This load is performed by a SQL script (sometimes called the "OLAP cube initial load script") that is delivered with the Oracle Retail Data Model OLAP component. The actual script that performs the OLAP cube initial load varies depending on Oracle Database release:

   - For Oracle Database 10*g*, the script is `ordm_olap_install_scr.sql`
   - For Oracle Database 11*g*, the script is `ordm_olap_11g_install_scr.sql`

   When the relational data exists in the Oracle Retail Data Model data warehouse, the OLAP cube initial load script loads relational table data (from a specified start date to the present time) into the OLAP cubes. It also performs the default OLAP forecasts. (For detailed information about the behavior of the OLAP cube initial load script, see the discussion on the OLAP component installation scripts in *Oracle Retail Data Model Reference*.)

   You can execute the OLAP cube initial load SQL script in the following ways:

   - Implicitly, by installing the Oracle Retail Data Model OLAP component *after* you have loaded data into the Oracle Retail Data Model relational tables. For instructions on how to install the Oracle Retail Data Model OLAP component, see *Oracle Retail Data Model Installation Guide*.
   - Explicitly, after you install the Oracle Retail Data Model OLAP component and populate the relational tables. In this case, you execute the OLAP cube initial load SQL program as you execute any other SQL program.

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

   This type of load (sometimes referred to as an "intermittent" or "refresh" load) merely adds relational data from a specified time period to the data in the Sales and Inventory cubes; it does not change the data in the Sales Forecast and Inventory Forecast cubes.

Oracle Retail Data Model does *not* provide an OLAP intermittent load cube script. You must write your own OLAP intermittent load cube script using the `RBIA_OLAP_ETL_AW_LOAD` package. For information on writing your own intermittent OLAP cube program and for updating the data in the OLAP forecast cubes, see "Refreshing OLAP Cube Data" on page 5-2.

4

Analysis and Reporting in Oracle Retail Data Model

This chapter introduces the analytic and reporting capabilities of Oracle Retail Data Model.

This chapter contains the following topics:

- Reports Delivered with Oracle Retail Data Model
- Customizing the Reports Delivered with Oracle Retail Data Model
- Writing Your Own Queries and Reports on Relational Data
- Writing Your Own Queries and Reports on OLAP Cube Data

Reports Delivered with Oracle Retail Data Model

Sample reports and dashboards are delivered with Oracle Retail Data Model. They are listed in a spreadsheet that resides in the following directory.

```
ORACLE_HOME/ORDM/REPORT/INSTALL
```

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

These sample reports illustrate the analytic capabilities provided with Oracle Retail Data Model -- including the OLAP and data mining capabilities.

The sample reports were developed using Oracle Business Intelligence Suite Enterprise Edition (Oracle BI EE) using the sample repository defined by the `RBIAII.rpd` file. Oracle BI EE is a comprehensive suite of enterprise BI products that delivers a full range of analysis and reporting capabilities. Thus, the reports also illustrated the ease with which you can use Oracle BI EE Answers and Dashboard presentation tools to create useful reports.

The sample reports delivered with Oracle Retail Data Model include:

- Sample Associate Basket Analysis Model Report
- Sample Associate Loss Analysis Model Reports
- Sample Associate Sales Analysis Model Report
- Sample Customer Product Category Mix Analysis Model Report

- Sample Customer Loyalty Analysis Model Report
- Sample Item Basket Analysis Model Report
- Sample Item Point of Sale (POS) Loss Analysis Model Report

Sample Associate Basket Analysis Model Report

In a retail environment, "basket" refers to the items a customer purchases in one transaction, as with a shopping cart of groceries or a car and a number of options. The size, context, and number of baskets sold are all valuable pieces of information.

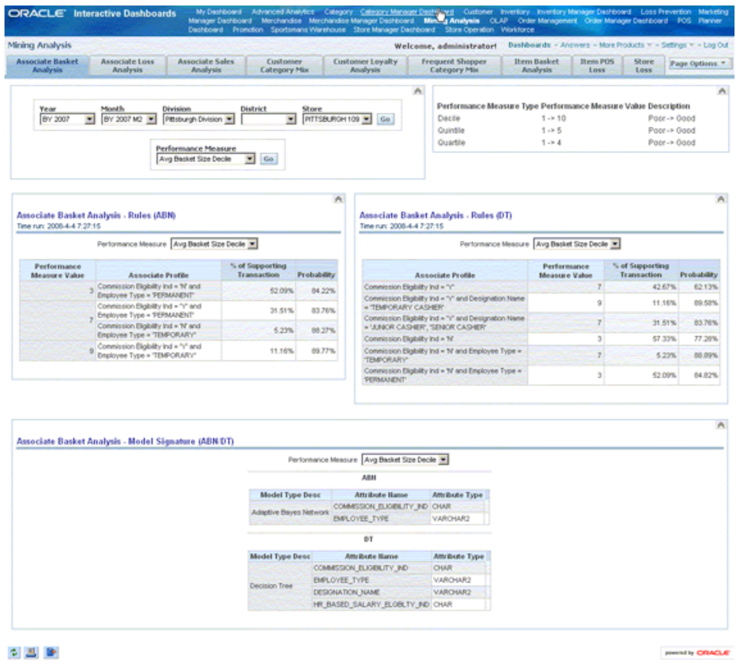
This model addresses the business problem of building a profile of associates to explain their basket Key Performance Indicators (KPIs), such as Total Baskets, Average Basket Value, and other statistics.

An Associate Basket model analysis identifies which key attributes of an associate influence his or her number of baskets sold, average basket value, and size. This model mines the various attributes of associates. It takes the binned variables one at a time for the Total Basket Count, Average Basket Value, and Average Basket Size as the target variable of an Adaptive Bayes (ABN) and Decision Tree (DT) model with a single feature and discovers rules described in terms of associate attributes.

> **See:** For detailed information about the Associate Basket Analysis Model delivered with Oracle Retail Data Model, see the discussion of that model in *Oracle Retail Data Model Reference*.

For this sample report, the presentation of the Rules has been changed to better reflect the Mining Model that the rules are based on. Decision Tree (DT) Rules make sense when displayed in their entirety in the order of the hierarchy (or in the order of the parent or child nodes making up the DT model). Adaptive Bayes (ABN) Rules are independent of each other and do not have a natural display order and can be customized to suit the presentation layer (front-end report) needs.

New report layouts (OBIEE or Oracle Answers reports) displaying the Associate Profile Rules or Model Signature corresponding to the 3 target variables.


ORACLE Interactive Dashboards
My Dashboard Advanced Analytics Category Category Manager Dashboard Customer Inventory Inventory Manager Dashboard Loss Prevention Marketing Manager Dashboard Merchandise Merchandise Manager Dashboard Mining Analysis OLAP Order Management Order Manager Dashboard POS Planner Dashboard Promotion Sportsmans Warehouse Store Manager Dashboard Store Operation Workforce
Mining Analysis
Welcome, administrator!
Dashboards - Answers - More Products - Settings - Log Out
Associate Basket Analysis
Associate Loss Analysis
Associate Sales Analysis
Customer Category Mix
Customer Loyalty Analysis
Frequent Shopper Category Mix
Item Basket Analysis
Item POS Loss
Store Loss
Page Options
Go
powered by ORACLE

Year: BY 2007 | Month: BY 2007 M2 | Division: Pittsburgh Division | District: | Store: PITTSBURGH 109

Performance Measure: Avg Basket Size Decile

Performance Measure Type	Performance Measure Value	Description
Decile	1 -> 10	Poor -> Good
Quintile	1 -> 5	Poor -> Good
Quartile	1 -> 4	Poor -> Good

Associate Basket Analysis - Rules (ABN)
Time run: 2008-4-4 7:27:15

Performance Measure: Avg Basket Size Decile

Performance Measure Value	Associate Profile	% of Supporting Transaction	Probability
3	Commission Eligibility Ind = 'N' and Employee Type = 'PERMANENT'	52.09%	84.22%
7	Commission Eligibility Ind = 'Y' and Employee Type = 'PERMANENT'	31.51%	83.76%
	Commission Eligibility Ind = 'N' and Employee Type = 'TEMPORARY'	5.23%	88.27%
9	Commission Eligibility Ind = 'Y' and Employee Type = 'TEMPORARY'	11.16%	89.77%

Associate Basket Analysis - Rules (DT)
Time run: 2008-4-4 7:27:15

Performance Measure: Avg Basket Size Decile

Associate Profile	Performance Measure Value	% of Supporting Transaction	Probability
Commission Eligibility Ind = 'Y'	7	42.67%	62.13%
Commission Eligibility Ind = 'Y' and Designation Name = 'TEMPORARY CASHIER'	9	11.16%	89.58%
Commission Eligibility Ind = 'Y' and Designation Name = 'JUNIOR CASHIER', 'SENIOR CASHIER'	7	31.51%	83.76%
Commission Eligibility Ind = 'N'	3	57.33%	77.28%
Commission Eligibility Ind = 'N' and Employee Type = 'TEMPORARY'	7	5.23%	88.89%
Commission Eligibility Ind = 'N' and Employee Type = 'PERMANENT'	3	52.09%	84.82%

Associate Basket Analysis - Model Signature (ABN/DT)

Performance Measure: Avg Basket Size Decile

ABN

Model Type Desc	Attribute Name	Attribute Type
Adaptive Bayes Network	COMMISSION_ELIGIBILITY_IND	CHAR
	EMPLOYEE_TYPE	VARCHAR2

DT

Model Type Desc	Attribute Name	Attribute Type
Decision Tree	COMMISSION_ELIGIBILITY_IND	CHAR
	EMPLOYEE_TYPE	VARCHAR2
	DESIGNATION_NAME	VARCHAR2
	HR_BASED_SALARY_ELGBLTY_IND	CHAR

Sample Associate Loss Analysis Model Reports

The Associate Loss Analysis model addresses the business problem of correlating associate characteristics to shrink and theft. (In a retail environment, shrink refers to merchandise that, for unknown reasons, is unaccounted for.)

The KPIs are converted into categorical variables using standard database binning operations. The categorical variables are modeled as a classification model in order to identify or predict the impact of various independent variables (attributes) on the dependent target variable (KPI - categorical).

The Associate Loss Analysis model mines the Total Shrink Count, Total Shrink Amount, Shrink as a percentage of Sales, Total Theft Count, Total Theft Amount and Theft as a percentage of Sales of individual associates to identify which of their key attributes influence their shrinkage and theft. This model takes the binned variables one at a time for Total Shrinkage and Theft Count or Value or percentage of Sales as the target variable of an Adaptive Bayes (ABN) and Decision Tree (DT) model and discovers rules described in terms of associate attributes.

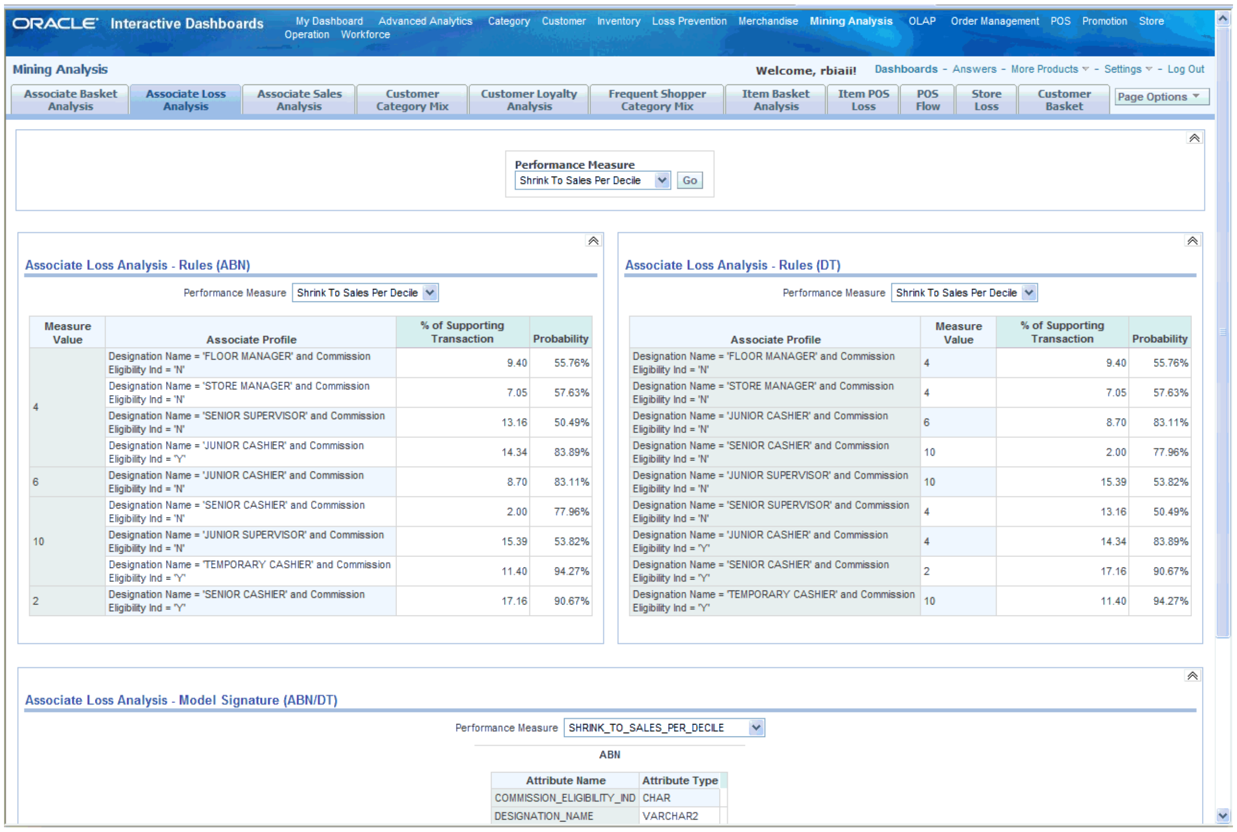
> **See:** For detailed information about the Associate Loss Analysis Model delivered with Oracle Retail Data Model, see the discussion of that model in *Oracle Retail Data Model Reference*.

For this sample report, the presentation of the Rules has been changed to better reflect the Mining Model that the rules are based on. Decision Tree (DT) Rules make sense when displayed in their entirety in the order of the hierarchy (or in the order of the parent or child nodes making up the DT model). Adaptive Bayes (ABN) Rules are independent of each other and do not have a natural display order and can be customized to suit the presentation layer (front-end report) needs.

New report layouts (OBIEE or Oracle Answers reports) displaying the Associate Profile Rules or Model Signature corresponding to the 3 target variables.

This sample OBIEE report shows Associate Shrink to Sales Rules:

Figure 4–1 Shrink to Sales Data Mining Report



Performance Measure: Shrink To Sales Per Decile

Associate Loss Analysis - Rules (ABN)

Performance Measure: Shrink To Sales Per Decile

Measure Value	Associate Profile	% of Supporting Transaction	Probability
4	Designation Name = 'FLOOR MANAGER' and Commission Eligibility Ind = 'N'	9.40	55.76%
	Designation Name = 'STORE MANAGER' and Commission Eligibility Ind = 'N'	7.05	57.63%
	Designation Name = 'SENIOR SUPERVISOR' and Commission Eligibility Ind = 'N'	13.16	50.49%
	Designation Name = 'JUNIOR CASHIER' and Commission Eligibility Ind = 'Y'	14.34	83.89%
6	Designation Name = 'JUNIOR CASHIER' and Commission Eligibility Ind = 'N'	8.70	83.11%
10	Designation Name = 'SENIOR CASHIER' and Commission Eligibility Ind = 'N'	2.00	77.96%
	Designation Name = 'JUNIOR SUPERVISOR' and Commission Eligibility Ind = 'N'	15.39	53.82%
	Designation Name = 'TEMPORARY CASHIER' and Commission Eligibility Ind = 'Y'	11.40	94.27%
2	Designation Name = 'SENIOR CASHIER' and Commission Eligibility Ind = 'Y'	17.16	90.67%

Associate Loss Analysis - Rules (DT)

Performance Measure: Shrink To Sales Per Decile

Associate Profile	Measure Value	% of Supporting Transaction	Probability
Designation Name = 'FLOOR MANAGER' and Commission Eligibility Ind = 'N'	4	9.40	55.76%
Designation Name = 'STORE MANAGER' and Commission Eligibility Ind = 'N'	4	7.05	57.63%
Designation Name = 'JUNIOR CASHIER' and Commission Eligibility Ind = 'N'	6	8.70	83.11%
Designation Name = 'SENIOR CASHIER' and Commission Eligibility Ind = 'N'	10	2.00	77.96%
Designation Name = 'JUNIOR SUPERVISOR' and Commission Eligibility Ind = 'N'	10	15.39	53.82%
Designation Name = 'SENIOR SUPERVISOR' and Commission Eligibility Ind = 'N'	4	13.16	50.49%
Designation Name = 'JUNIOR CASHIER' and Commission Eligibility Ind = 'Y'	4	14.34	83.89%
Designation Name = 'SENIOR CASHIER' and Commission Eligibility Ind = 'Y'	2	17.16	90.67%
Designation Name = 'TEMPORARY CASHIER' and Commission Eligibility Ind = 'Y'	10	11.40	94.27%

Associate Loss Analysis - Model Signature (ABN/DT)

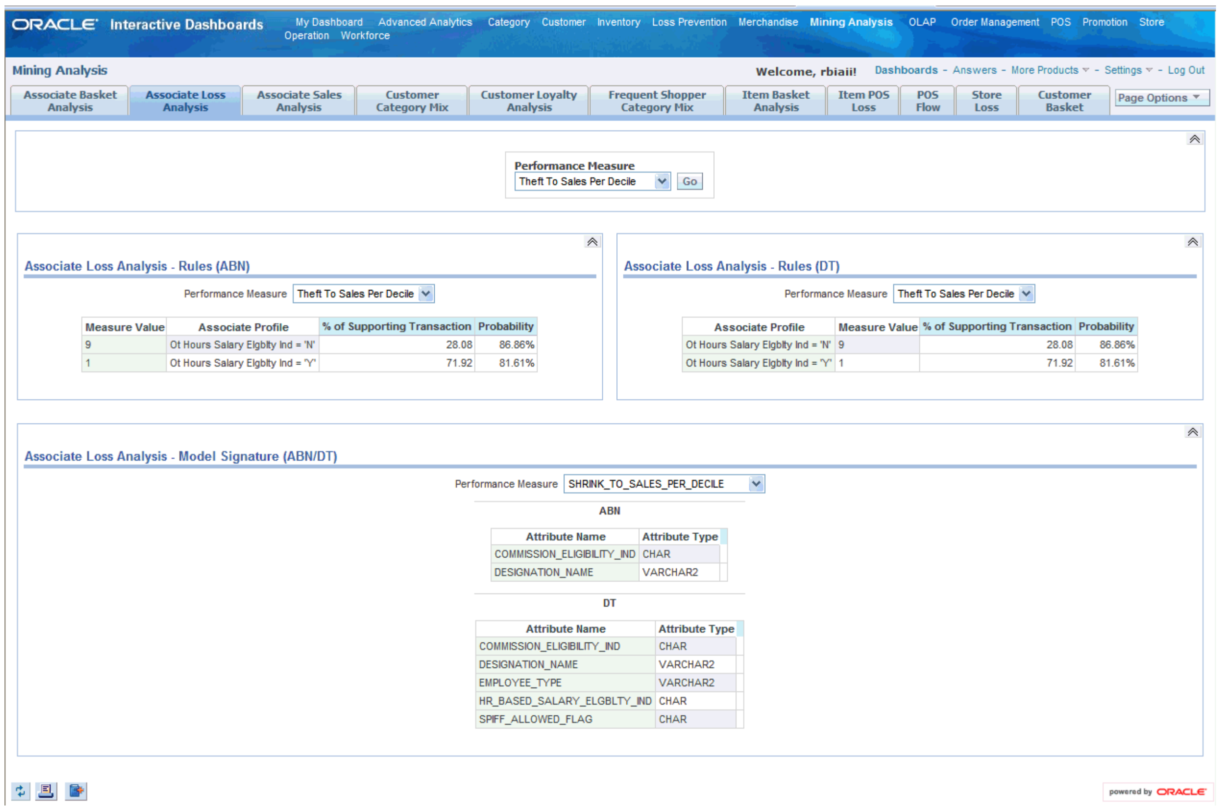
Performance Measure: SHRINK_TO_SALES_PER_DECILE

ABN

Attribute Name	Attribute Type
COMMISSION_ELIGIBILITY_IND	CHAR
DESIGNATION_NAME	VARCHAR2

This sample OBIEE report shows Associate Theft to Sales Rules:

Figure 4–2 Theft to Sales Data Mining Report



Performance Measure: Theft To Sales Per Decile

Associate Loss Analysis - Rules (ABN)

Performance Measure: Theft To Sales Per Decile

Measure Value	Associate Profile	% of Supporting Transaction	Probability
9	Ot Hours Salary Elgblty Ind = 'N'	28.08	86.86%
1	Ot Hours Salary Elgblty Ind = 'Y'	71.92	81.61%

Associate Loss Analysis - Rules (DT)

Performance Measure: Theft To Sales Per Decile

Associate Profile	Measure Value	% of Supporting Transaction	Probability
Ot Hours Salary Elgblty Ind = 'N'	9	28.08	86.86%
Ot Hours Salary Elgblty Ind = 'Y'	1	71.92	81.61%

Associate Loss Analysis - Model Signature (ABN/DT)

Performance Measure: SHRINK_TO_SALES_PER_DECILE

ABN

Attribute Name	Attribute Type
COMMISSION_ELIGIBILITY_IND	CHAR
DESIGNATION_NAME	VARCHAR2

DT

Attribute Name	Attribute Type
COMMISSION_ELIGIBILITY_IND	CHAR
DESIGNATION_NAME	VARCHAR2
EMPLOYEE_TYPE	VARCHAR2
HR_BASED_SALARY_ELGBLTY_IND	CHAR
SPIFF_ALLOWED_FLAG	CHAR

Sample Associate Sales Analysis Model Report

The Associate Sales Analysis model addresses the business problem of profiling associate characteristics to sales, cost, and profit patterns.

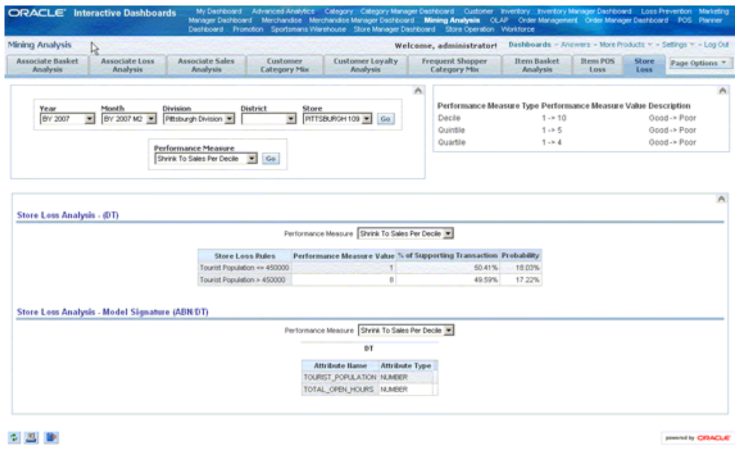
The KPIs are converted into categorical variables using standard database binning operations. The categorical variables are modeled as a classification model in order to identify or predict the impact of various independent variables (attributes) on the dependent target variable (KPI - categorical).

This model mines the various attributes of associates. It takes the binned variables one at a time for Sales, Costs, and Profits as the target variable of an Adaptive Bayes (ABN) and Decision Tree (DT) model with a single feature and discovers rules described in terms of associate attributes.

> **See:** For detailed information about the Sample Sales Analysis Model delivered with Oracle Retail Data Model, see the discussion of that model in *Oracle Retail Data Model Reference*.

For this sample report, the presentation of the Rules has been changed to better reflect the Mining Model that the rules are based on. Decision Tree (DT) Rules make sense when displayed in their entirety in the order of the hierarchy (or in the order of the parent or child nodes making up the DT model). Adaptive Bayes (ABN) Rules are independent of each other and do not have a natural display order and can be customized to suit the presentation layer (front-end report) needs.

The following report shows the layout of the Store Loss report.



Year: BY 2007
Month: BY 2007 M2
Division: Pittsburgh Division
District:
Store: PITTSBURGH 109

Performance Measure: Shrink To Sales Per Decile

Performance Measure Type	Performance Measure Value	Description
Decile	1 -> 10	Good -> Poor
Quintile	1 -> 5	Good -> Poor
Quartile	1 -> 4	Good -> Poor

Store Loss Analysis - (DT)

Performance Measure: Shrink To Sales Per Decile

Store Loss Rules	Performance Measure Value	% of Supporting Transaction	Probability
Tourist Population <= 450000	1	50.41%	18.03%
Tourist Population > 450000	8	49.59%	17.22%

Store Loss Analysis - Model Signature (ABN/DT)

Performance Measure: Shrink To Sales Per Decile

DT

Attribute Name	Attribute Type
TOURIST_POPULATION	NUMBER
TOTAL_OPEN_HOURS	NUMBER

Sample Customer Product Category Mix Analysis Model Report

The Customer Category Mix Analysis model addresses the business problem of discovering product categories that are frequently bought together by customers. The model is used to understand the Categories purchased by a Customer in a typical transaction in terms of the components like the Categories in the Basket, Target Category in a Basket and additional information like Basket Significance (Sales Value), Target Category Significance (Sales Value) which are generated from regular Customer Transactional data.

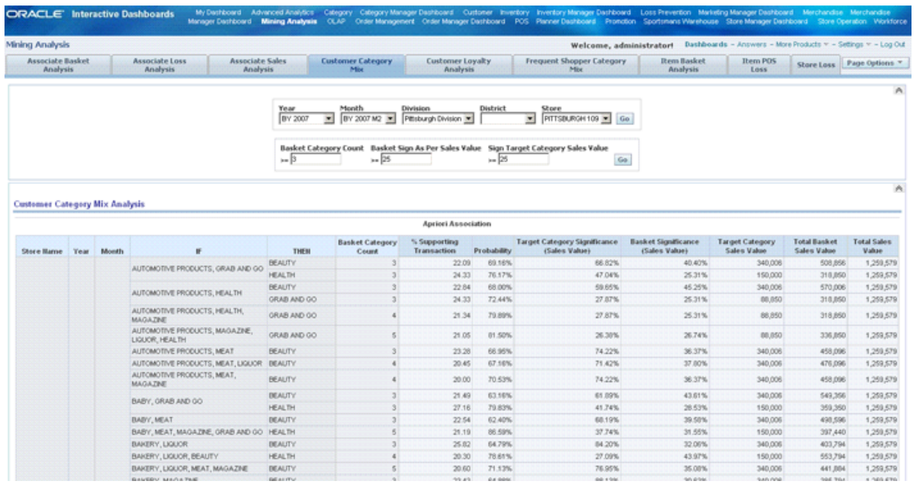
Using Oracle Data Mining, the KPIs are modeled with the APRIORI algorithm utilized by the Association Rules model. The model type used for Association Rules with Apriori Algorithm is APASS. This is an example of Unclassified Learning since the Categories (or Target Category) which make up the Category Basket are not inferred or guided (as part of data preparation) but are generated by the model itself.

This model mines the monthly purchases of individual customers and discovers rules about the categories that are frequently bought in groups by customers.

> **See:** For detailed information about the Customer Category Mix Analysis Model delivered with Oracle Retail Data Model, see the discussion of that model in *Oracle Retail Data Model Reference*.

For this sample report, the presentation of the Rules for Category Mix Analysis has been enhanced by the introduction of an additional Dashboard prompt to filter the Report (Rules) on (a) Number of Categories in the Rule, (b) Category Basket Significance (Sales Value) and (c) Target Category Significance (Sales Value) within the Category Basket.

The following is a portion of the Customer Product Category Mix Analysis report.



Year: BY 2007 | Month: BY 2007 M2 | Division: Pittsburgh Division | District: | Store: PITTSBURGH 109

Basket Category Count >= 3 | Basket Sign As Per Sales Value >= 25 | Sign Target Category Sales Value >= 25

Customer Category Mix Analysis

Apriori Association

Store Name	Year	Month	IF	THEN	Basket Category Count	% Supporting Transaction	Probability	Target Category Significance (Sales Value)	Basket Significance (Sales Value)	Target Category Sales Value	Total Basket Sales Value	Total Sales Value
			AUTOMOTIVE PRODUCTS, GRAB AND GO	BEAUTY	3	22.09	69.16%	66.82%	40.40%	340,006	508,856	1,259,579
				HEALTH	3	24.33	76.17%	47.04%	25.31%	150,000	318,850	1,259,579
			AUTOMOTIVE PRODUCTS, HEALTH	BEAUTY	3	22.84	68.00%	59.65%	45.25%	340,006	570,006	1,259,579
				GRAB AND GO	3	24.33	72.44%	27.87%	25.31%	88,850	318,850	1,259,579
			AUTOMOTIVE PRODUCTS, HEALTH, MAGAZINE	GRAB AND GO	4	21.34	79.89%	27.87%	25.31%	88,850	318,850	1,259,579
			AUTOMOTIVE PRODUCTS, MAGAZINE, LIQUOR, HEALTH	GRAB AND GO	5	21.05	81.50%	26.38%	26.74%	88,850	336,850	1,259,579
			AUTOMOTIVE PRODUCTS, MEAT	BEAUTY	3	23.28	66.95%	74.22%	36.37%	340,006	458,096	1,259,579
			AUTOMOTIVE PRODUCTS, MEAT, LIQUOR	BEAUTY	4	20.45	67.16%	71.42%	37.80%	340,006	476,096	1,259,579
			AUTOMOTIVE PRODUCTS, MEAT, MAGAZINE	BEAUTY	4	20.00	70.53%	74.22%	36.37%	340,006	458,096	1,259,579
			BABY, GRAB AND GO	BEAUTY	3	21.49	63.16%	61.89%	43.61%	340,006	549,356	1,259,579
				HEALTH	3	27.16	79.83%	41.74%	28.53%	150,000	359,350	1,259,579
			BABY, MEAT	BEAUTY	3	22.54	62.40%	68.19%	39.58%	340,006	498,596	1,259,579
			BABY, MEAT, MAGAZINE, GRAB AND GO	HEALTH	5	21.19	86.59%	37.74%	31.55%	150,000	397,440	1,259,579
			BAKERY, LIQUOR	BEAUTY	3	25.82	64.79%	84.20%	32.06%	340,006	403,794	1,259,579
			BAKERY, LIQUOR, BEAUTY	HEALTH	4	20.30	78.61%	27.09%	43.97%	150,000	553,794	1,259,579
			BAKERY, LIQUOR, MEAT, MAGAZINE	BEAUTY	5	20.60	71.13%	76.95%	35.08%	340,006	441,884	1,259,579

Sample Customer Loyalty Analysis Model Report

The Customer Loyalty Analysis model addresses the business problem of discovering the impact of customer characteristics on customers' loyalty to a store.

This model mines the Customer and Account demographic characteristics of Customers to identify the key attribute influencing the Customer Loyalty scores (RFMP Category Value).

The RFMP algorithms provide functionality to group customers into quartiles, deciles, and quintiles. Each customer falls into one of the following five loyalty categories based on the RFMP quintile he or she belongs to in a particular month:

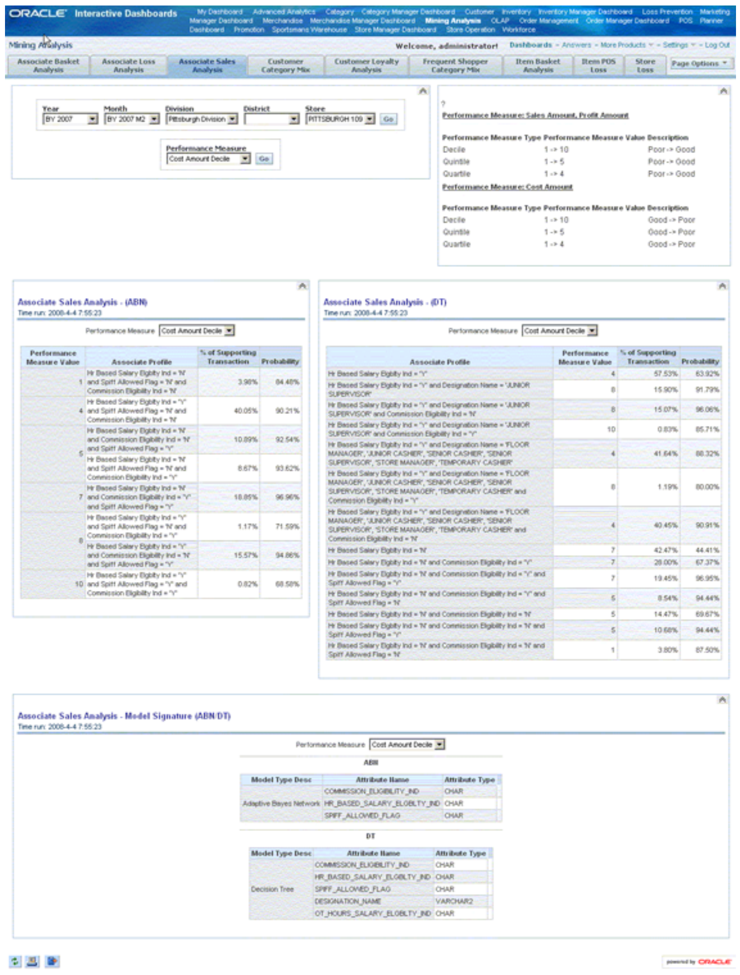
- Group A (RFMP Quintile 5)
- Group B (RFMP Quintile 4)
- Group C (RFMP Quintile 3)
- Group D (RFMP Quintile 2)
- Group E (RFMP Quintile 1)

> **See:** For detailed information about the Customer Loyalty Analysis Model delivered with Oracle Retail Data Model, see the discussion of that model in *Oracle Retail Data Model Reference*.

For this sample report, the presentation of the Rules has been changed to better reflect the Mining Model that the rules are based on. Decision Tree (DT) Rules make sense when displayed in their entirety in the order of the hierarchy (or in the order of the parent or child nodes making up the DT model). Adaptive Bayes (ABN) Rules are independent of each other and do not have a natural display order and can be customized to suit the presentation layer (front-end report) needs.

New report layouts (OBIEE or Oracle Answers reports) displaying the Customer Loyalty Rules or Model Signature.

This sample OBIEE report shows Customer Loyalty Rules:



Year: BY 2007 | Month: BY 2007 M2 | Division: Pittsburgh Division | District: | Store: PITTSBURGH 109

Performance Measure: Cost Amount Decile

Performance Measure: Sales Amount, Profit Amount

Performance Measure Type	Performance Measure Value	Description
Decile	1 -> 10	Poor -> Good
Quintile	1 -> 5	Poor -> Good
Quartile	1 -> 4	Poor -> Good

Performance Measure: Cost Amount

Performance Measure Type	Performance Measure Value	Description
Decile	1 -> 10	Good -> Poor
Quintile	1 -> 5	Good -> Poor
Quartile	1 -> 4	Good -> Poor

Associate Sales Analysis - (ABN)
Time run: 2008-4-4 7:55:23

Performance Measure: Cost Amount Decile

Performance Measure Value	Associate Profile	% of Supporting Transaction	Probability
1	Hr Based Salary Elgblty Ind = 'N' and Spiff Allowed Flag = 'N' and Commission Eligibility Ind = 'N'	3.98%	84.48%
4	Hr Based Salary Elgblty Ind = 'Y' and Spiff Allowed Flag = 'N' and Commission Eligibility Ind = 'N'	40.05%	90.21%
5	Hr Based Salary Elgblty Ind = 'N' and Commission Eligibility Ind = 'N' and Spiff Allowed Flag = 'Y'	10.89%	92.54%
	Hr Based Salary Elgblty Ind = 'N' and Spiff Allowed Flag = 'N' and Commission Eligibility Ind = 'Y'	8.67%	93.62%
7	Hr Based Salary Elgblty Ind = 'N' and Commission Eligibility Ind = 'Y' and Spiff Allowed Flag = 'Y'	18.85%	96.96%
8	Hr Based Salary Elgblty Ind = 'Y' and Spiff Allowed Flag = 'N' and Commission Eligibility Ind = 'Y'	1.17%	71.59%
	Hr Based Salary Elgblty Ind = 'Y' and Commission Eligibility Ind = 'N' and Spiff Allowed Flag = 'Y'	15.57%	94.86%
10	Hr Based Salary Elgblty Ind = 'Y' and Spiff Allowed Flag = 'Y' and Commission Eligibility Ind = 'Y'	0.82%	68.58%

Associate Sales Analysis - (DT)
Time run: 2008-4-4 7:55:23

Performance Measure: Cost Amount Decile

Associate Profile	Performance Measure Value	% of Supporting Transaction	Probability
Hr Based Salary Elgblty Ind = 'Y'	4	57.53%	63.92%
Hr Based Salary Elgblty Ind = 'Y' and Designation Name = 'JUNIOR SUPERVISOR'	8	15.90%	91.79%
Hr Based Salary Elgblty Ind = 'Y' and Designation Name = 'JUNIOR SUPERVISOR' and Commission Eligibility Ind = 'N'	8	15.07%	96.06%
Hr Based Salary Elgblty Ind = 'Y' and Designation Name = 'JUNIOR SUPERVISOR' and Commission Eligibility Ind = 'Y'	10	0.83%	85.71%
Hr Based Salary Elgblty Ind = 'Y' and Designation Name = 'FLOOR MANAGER', 'JUNIOR CASHIER', 'SENIOR CASHIER', 'SENIOR SUPERVISOR', 'STORE MANAGER', 'TEMPORARY CASHIER'	4	41.64%	88.32%
Hr Based Salary Elgblty Ind = 'Y' and Designation Name = 'FLOOR MANAGER', 'JUNIOR CASHIER', 'SENIOR CASHIER', 'SENIOR SUPERVISOR', 'STORE MANAGER', 'TEMPORARY CASHIER' and Commission Eligibility Ind = 'Y'	8	1.19%	80.00%
Hr Based Salary Elgblty Ind = 'Y' and Designation Name = 'FLOOR MANAGER', 'JUNIOR CASHIER', 'SENIOR CASHIER', 'SENIOR SUPERVISOR', 'STORE MANAGER', 'TEMPORARY CASHIER' and Commission Eligibility Ind = 'N'	4	40.45%	90.91%
Hr Based Salary Elgblty Ind = 'N'	7	42.47%	44.41%
Hr Based Salary Elgblty Ind = 'N' and Commission Eligibility Ind = 'Y'	7	28.00%	67.37%
Hr Based Salary Elgblty Ind = 'N' and Commission Eligibility Ind = 'Y' and Spiff Allowed Flag = 'Y'	7	19.45%	96.95%
Hr Based Salary Elgblty Ind = 'N' and Commission Eligibility Ind = 'Y' and Spiff Allowed Flag = 'N'	5	8.54%	94.44%
Hr Based Salary Elgblty Ind = 'N' and Commission Eligibility Ind = 'N'	5	14.47%	69.67%
Hr Based Salary Elgblty Ind = 'N' and Commission Eligibility Ind = 'N' and Spiff Allowed Flag = 'Y'	5	10.68%	94.44%
Hr Based Salary Elgblty Ind = 'N' and Commission Eligibility Ind = 'N' and Spiff Allowed Flag = 'N'	1	3.80%	87.50%

Associate Sales Analysis - Model Signature (ABN/DT)
Time run: 2008-4-4 7:55:23

Performance Measure: Cost Amount Decile

ABN

Model Type Desc	Attribute Name	Attribute Type
Adaptive Bayes Network	COMMISSION_ELIGIBILITY_IND	CHAR
	HR_BASED_SALARY_ELGBLTY_IND	CHAR
	SPIFF_ALLOWED_FLAG	CHAR

DT

Model Type Desc	Attribute Name	Attribute Type
Decision Tree	COMMISSION_ELIGIBILITY_IND	CHAR
	HR_BASED_SALARY_ELGBLTY_IND	CHAR
	SPIFF_ALLOWED_FLAG	CHAR
	DESIGNATION_NAME	VARCHAR2
	OT_HOURS_SALARY_ELGBLTY_IND	CHAR

Sample Item Basket Analysis Model Report

The Sample Item Basket Analysis model addresses the business problem of identifying the extent to which item (product) characteristics influence the items' sales KPIs.

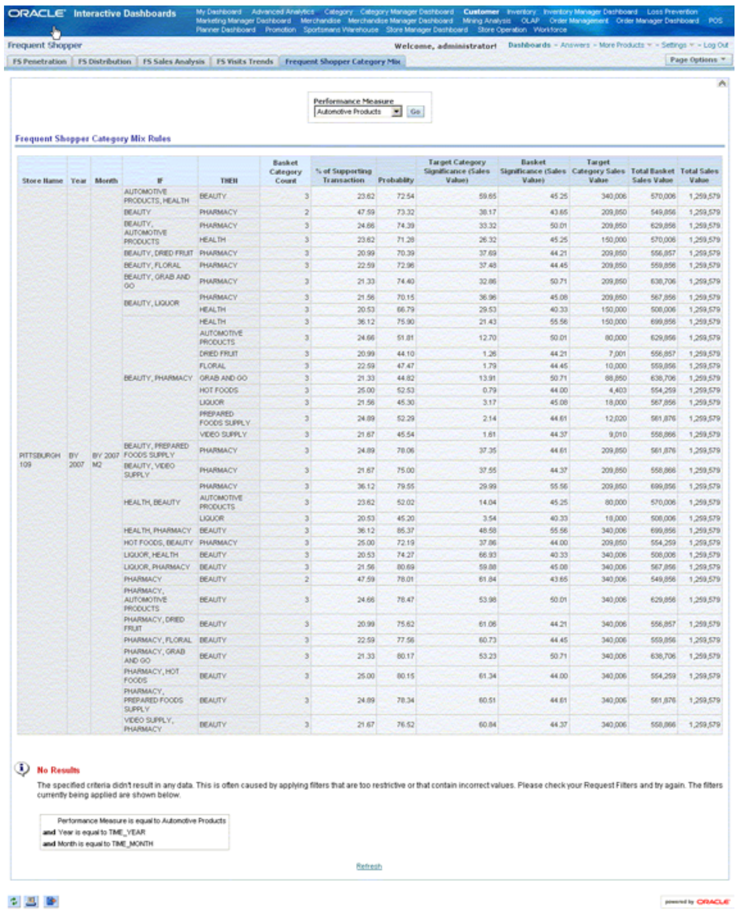
The KPIs are converted into categorical variables using standard database binning operations. The categorical variables are modeled as a classification model in order to identify or predict the impact of various independent variables (attributes) on the dependent target variable (KPI - categorical).

This model identifies which key attributes of an item influence the number of baskets sold, average basket value, and size in a particular store. This model mines the various attributes of items. It takes the binned variables one at a time for Total Basket Count, Average Basket Value, and Average Basket Size as the target variable of an ABN model and DT model with a single feature and discovers rules described in terms of item characteristics.

> **See:** For detailed information about the Item Basket Analysis Model delivered with Oracle Retail Data Model, see the discussion of that model in *Oracle Retail Data Model Reference*.

For this sample report, the presentation of the Rules has been changed to better reflect the Mining Model that the rules are based on. Decision Tree (DT) Rules make sense when displayed in their entirety in the order of the hierarchy (or in the order of the parent or child nodes making up the DT model). Adaptive Bayes (ABN) Rules are independent of each other and do not have a natural display order and can be customized to suit the presentation layer (front-end report) needs.

The layout of the report is shown in the following sample.



Performance Measure: Automotive Products

Frequent Shopper Category Mix Rules

Store Name	Year	Month	IF	THEN	Basket Category Count	% of Supporting Transaction	Probability	Target Category Significance (Sales Value)	Basket Significance (Sales Value)	Target Category Sales Value	Total Basket Sales Value	Total Sales Value
PITTSBURGH 109	BY 2007	BY 2007 M2	AUTOMOTIVE PRODUCTS, HEALTH	BEAUTY	3	23.62	72.54	59.65	45.25	340,006	570,006	1,259,579
			BEAUTY	PHARMACY	2	47.59	73.32	38.17	43.65	209,850	549,856	1,259,579
			BEAUTY, AUTOMOTIVE PRODUCTS	PHARMACY	3	24.66	74.39	33.32	50.01	209,850	629,856	1,259,579
				HEALTH	3	23.62	71.28	26.32	45.25	150,000	570,006	1,259,579
			BEAUTY, DRIED FRUIT	PHARMACY	3	20.99	70.39	37.69	44.21	209,850	556,857	1,259,579
			BEAUTY, FLORAL	PHARMACY	3	22.59	72.96	37.48	44.45	209,850	559,856	1,259,579
			BEAUTY, GRAB AND GO	PHARMACY	3	21.33	74.40	32.86	50.71	209,850	638,706	1,259,579
			BEAUTY, LIQUOR	PHARMACY	3	21.56	70.15	36.96	45.08	209,850	567,856	1,259,579
				HEALTH	3	20.53	66.79	29.53	40.33	150,000	508,006	1,259,579
			BEAUTY, PHARMACY	HEALTH	3	36.12	75.90	21.43	55.56	150,000	699,856	1,259,579
				AUTOMOTIVE PRODUCTS	3	24.66	51.81	12.70	50.01	80,000	629,856	1,259,579
				DRIED FRUIT	3	20.99	44.10	1.26	44.21	7,001	556,857	1,259,579
				FLORAL	3	22.59	47.47	1.79	44.45	10,000	559,856	1,259,579
				GRAB AND GO	3	21.33	44.82	13.91	50.71	88,850	638,706	1,259,579
				HOT FOODS	3	25.00	52.53	0.79	44.00	4,403	554,259	1,259,579
				LIQUOR	3	21.56	45.30	3.17	45.08	18,000	567,856	1,259,579
				PREPARED FOODS SUPPLY	3	24.89	52.29	2.14	44.61	12,020	561,876	1,259,579
				VIDEO SUPPLY	3	21.67	45.54	1.61	44.37	9,010	558,866	1,259,579
			BEAUTY, PREPARED FOODS SUPPLY	PHARMACY	3	24.89	78.06	37.35	44.61	209,850	561,876	1,259,579
			BEAUTY, VIDEO SUPPLY	PHARMACY	3	21.67	75.00	37.55	44.37	209,850	558,866	1,259,579
			HEALTH, BEAUTY	PHARMACY	3	36.12	79.55	29.99	55.56	209,850	699,856	1,259,579
				AUTOMOTIVE PRODUCTS	3	23.62	52.02	14.04	45.25	80,000	570,006	1,259,579
				LIQUOR	3	20.53	45.20	3.54	40.33	18,000	508,006	1,259,579
			HEALTH, PHARMACY	BEAUTY	3	36.12	85.37	48.58	55.56	340,006	699,856	1,259,579
			HOT FOODS, BEAUTY	PHARMACY	3	25.00	72.19	37.86	44.00	209,850	554,259	1,259,579
			LIQUOR, HEALTH	BEAUTY	3	20.53	74.27	66.93	40.33	340,006	508,006	1,259,579
			LIQUOR, PHARMACY	BEAUTY	3	21.56	80.69	59.88	45.08	340,006	567,856	1,259,579
			PHARMACY	BEAUTY	2	47.59	78.01	61.84	43.65	340,006	549,856	1,259,579
			PHARMACY, AUTOMOTIVE PRODUCTS	BEAUTY	3	24.66	78.47	53.98	50.01	340,006	629,856	1,259,579
			PHARMACY, DRIED FRUIT	BEAUTY	3	20.99	75.62	61.06	44.21	340,006	556,857	1,259,579
			PHARMACY, FLORAL	BEAUTY	3	22.59	77.56	60.73	44.45	340,006	559,856	1,259,579
			PHARMACY, GRAB AND GO	BEAUTY	3	21.33	80.17	53.23	50.71	340,006	638,706	1,259,579
			PHARMACY, HOT FOODS	BEAUTY	3	25.00	80.15	61.34	44.00	340,006	554,259	1,259,579
			PHARMACY, PREPARED FOODS SUPPLY	BEAUTY	3	24.89	78.34	60.51	44.61	340,006	561,876	1,259,579
			VIDEO SUPPLY, PHARMACY	BEAUTY	3	21.67	76.52	60.84	44.37	340,006	558,866	1,259,579

No Results

The specified criteria didn't result in any data. This is often caused by applying filters that are too restrictive or that contain incorrect values. Please check your Request Filters and try again. The filters currently being applied are shown below.

Performance Measure is equal to Automotive Products
and Year is equal to TIME_YEAR
and Month is equal to TIME_MONTH

Sample Item Point of Sale (POS) Loss Analysis Model Report

The Item POS Loss Analysis model addresses the business problem of building a profile of item (product) characteristics with regard to POS losses.

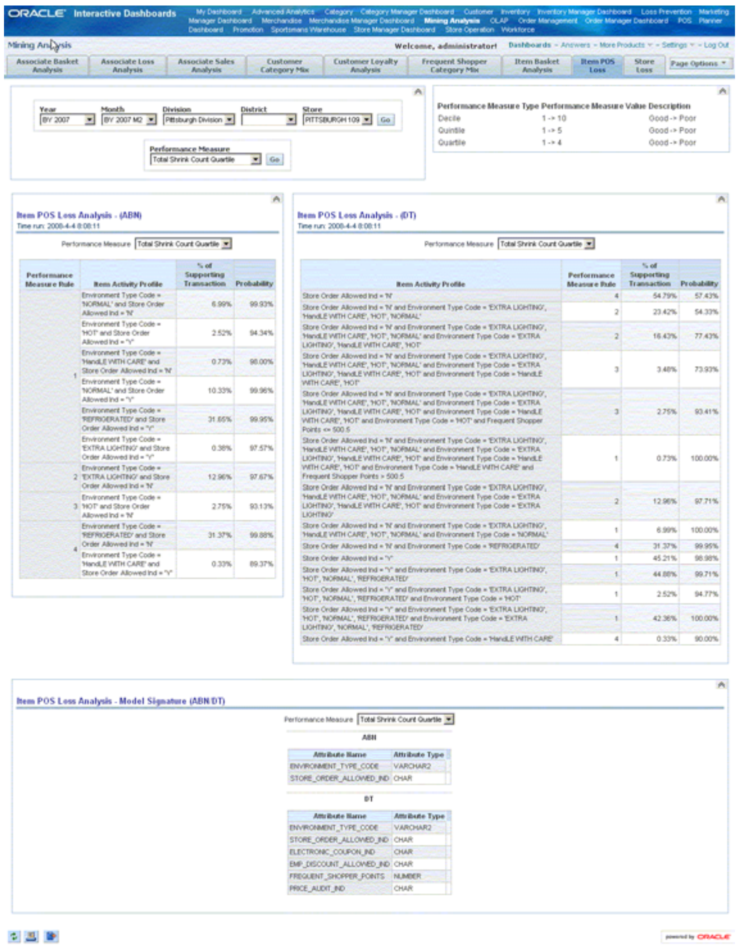
The KPIs are converted into categorical variables using standard database binning operations. The categorical variables are modeled as a classification model in order to identify or predict the impact of various independent variables (attributes) on the dependent target variable (KPI - categorical).

This model mines the POS transactions along with the item attributes to identify their impact on Total Shrink Count, Total Shrink Amount, Shrink as a percentage of Sales, Total Theft Count, Total Theft Amount, and Theft as a percentage of Sales.

> **See:** For detailed information about the Item POS Loss Analysis Model delivered with Oracle Retail Data Model, see the discussion of that model in *Oracle Retail Data Model Reference*.

For this sample report, the presentation of the Rules has been changed to better reflect the Mining Model that the rules are based on. Decision Tree (DT) Rules make sense when displayed in their entirety in the order of the hierarchy (or in the order of the parent or child nodes making up the DT model). Adaptive Bayes (ABN) Rules are independent of each other and do not have a natural display order and can be customized to suit the presentation layer (front-end report) needs.

The following report shows the layout of the report described in this section.



Year: BY 2007 Month: BY 2007 M2 Division: Pittsburgh Division District: Store: PITTSBURGH 109

Performance Measure: Total Shrink Count Quartile

Performance Measure Type	Performance Measure Value	Description
Decile	1 -> 10	Good -> Poor
Quintile	1 -> 5	Good -> Poor
Quartile	1 -> 4	Good -> Poor

Item POS Loss Analysis - (ABN)
Time run: 2008-4-4 8:08:11

Performance Measure: Total Shrink Count Quartile

Performance Measure Rule	Item Activity Profile	% of Supporting Transaction	Probability
1	Environment Type Code = 'NORMAL' and Store Order Allowed Ind = 'N'	6.99%	99.93%
	Environment Type Code = 'HOT' and Store Order Allowed Ind = 'Y'	2.52%	94.34%
	Environment Type Code = 'HandLE WITH CARE' and Store Order Allowed Ind = 'N'	0.73%	98.00%
	Environment Type Code = 'NORMAL' and Store Order Allowed Ind = 'Y'	10.33%	99.96%
	Environment Type Code = 'REFRIGERATED' and Store Order Allowed Ind = 'Y'	31.65%	99.95%
	Environment Type Code = 'EXTRA LIGHTING' and Store Order Allowed Ind = 'Y'	0.38%	97.57%
2	Environment Type Code = 'EXTRA LIGHTING' and Store Order Allowed Ind = 'N'	12.96%	97.67%
3	Environment Type Code = 'HOT' and Store Order Allowed Ind = 'N'	2.75%	93.13%
4	Environment Type Code = 'REFRIGERATED' and Store Order Allowed Ind = 'N'	31.37%	99.88%
	Environment Type Code = 'HandLE WITH CARE' and Store Order Allowed Ind = 'Y'	0.33%	89.37%

Item POS Loss Analysis - (DT)
Time run: 2008-4-4 8:08:11

Performance Measure: Total Shrink Count Quartile

Item Activity Profile	Performance Measure Rule	% of Supporting Transaction	Probability
Store Order Allowed Ind = 'N'	4	54.79%	57.43%
Store Order Allowed Ind = 'N' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT', 'NORMAL'	2	23.42%	54.33%
Store Order Allowed Ind = 'N' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT', 'NORMAL' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT'	2	16.43%	77.43%
Store Order Allowed Ind = 'N' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT', 'NORMAL' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT' and Environment Type Code = 'HandLE WITH CARE', 'HOT'	3	3.48%	73.93%
Store Order Allowed Ind = 'N' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT', 'NORMAL' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT' and Environment Type Code = 'HandLE WITH CARE', 'HOT' and Environment Type Code = 'HOT' and Frequent Shopper Points <= 500.5	3	2.75%	93.41%
Store Order Allowed Ind = 'N' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT', 'NORMAL' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT' and Environment Type Code = 'HandLE WITH CARE', 'HOT' and Environment Type Code = 'HandLE WITH CARE' and Frequent Shopper Points > 500.5	1	0.73%	100.00%
Store Order Allowed Ind = 'N' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT', 'NORMAL' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT' and Environment Type Code = 'EXTRA LIGHTING'	2	12.96%	97.71%
Store Order Allowed Ind = 'N' and Environment Type Code = 'EXTRA LIGHTING', 'HandLE WITH CARE', 'HOT', 'NORMAL' and Environment Type Code = 'NORMAL'	1	6.99%	100.00%
Store Order Allowed Ind = 'N' and Environment Type Code = 'REFRIGERATED'	4	31.37%	99.95%
Store Order Allowed Ind = 'Y'	1	45.21%	98.98%
Store Order Allowed Ind = 'Y' and Environment Type Code = 'EXTRA LIGHTING', 'HOT', 'NORMAL', 'REFRIGERATED'	1	44.88%	99.71%
Store Order Allowed Ind = 'Y' and Environment Type Code = 'EXTRA LIGHTING', 'HOT', 'NORMAL', 'REFRIGERATED' and Environment Type Code = 'HOT'	1	2.52%	94.77%
Store Order Allowed Ind = 'Y' and Environment Type Code = 'EXTRA LIGHTING', 'HOT', 'NORMAL', 'REFRIGERATED' and Environment Type Code = 'EXTRA LIGHTING', 'NORMAL', 'REFRIGERATED'	1	42.36%	100.00%
Store Order Allowed Ind = 'Y' and Environment Type Code = 'HandLE WITH CARE'	4	0.33%	90.00%

Item POS Loss Analysis - Model Signature (ABN DT)

Performance Measure: Total Shrink Count Quartile

ABN

Attribute Name	Attribute Type
ENVIRONMENT_TYPE_CODE	VARCHAR2
STORE_ORDER_ALLOWED_IND	CHAR

DT

Attribute Name	Attribute Type
ENVIRONMENT_TYPE_CODE	VARCHAR2
STORE_ORDER_ALLOWED_IND	CHAR
ELECTRONIC_COUPON_IND	CHAR
EMP_DISCOUNT_ALLOWED_IND	CHAR
FREQUENT_SHOPPER_POINTS	NUMBER
PRICE_AUDIT_IND	CHAR

Customizing the Reports Delivered with Oracle Retail Data Model

You can use Oracle BI EE 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.

Writing Your Own Queries and Reports on Relational Data

The `bia_rtl` and bia_rtl_mining schemas define the relational tables and views in Oracle Retail Data Model. You can use any SQL reporting tool to query and report on these tables and views.

> **See:** For more information on the relational tables and views, see the discussion of the physical model of the Oracle Retail Data Model in *Oracle Retail Data Model Reference*.

Writing Your Own Queries and Reports on OLAP Cube Data

Oracle Retail Data Model supports On Line Analytic Processing (OLAP) reporting through the use of OLAP cubes. The OLAP components of Oracle Retail Data Model are described in "Physical Data Model of the OLAP Component" in *Oracle Retail Data Model Reference*.

You can query and write reports on these OLAP cubes using SQL tools or OLAP tools. Additionally, when you have installed the Oracle Retail Data Model OLAP component for Oracle Database 11*g*, you can create an Oracle Business Intelligence Enterprise Edition (OBIEE) repository for the OLAP cubes.

Querying and Reporting on OLAP Cubes Using SQL Tools

The `bia_rtl_olap` schema that defines the OLAP cubes also defines relational views of the OLAP dimensions and cubes. You can use any SQL reporting tool to query and report on these views. For more information on the relational views of OLAP cube data, see the discussion of the physical model of the OLAP component in *Oracle Retail Data Model Reference*.

> **See also:** "Reports Delivered with Oracle Retail Data Model" on page 4-1.

Querying and Reporting on OLAP Cube Data Using OLAP Tools

When you have the Oracle Retail Data Model OLAP component installed, you can write reports on OLAP cubes using OLAP tools. OLAP tools are designed specifically to locate all cubes and dimensions that are accessible to the current user. They automatically use the implicit relationships among cubes, dimensions, hierarchies, levels, and attributes. For example, drilling is automatically supported, children are clearly identified under their parent values, and description attributes are used as labels instead of dimension keys.

> **See also:** Chapter 5, "Querying Dimensional Objects Using OLAP Tools" in *Oracle OLAP Application Developer's Guide* which discusses how to use the Oracle Business Intelligence Spreadsheet Add-In with an OLAP cube.

Using a OBIEE Repository for OLAP Cubes

When you install the Oracle Retail Data Model OLAP component, a relational view named `bia_rtl_olap.OOS_CUBEVIEW` view is defined. This relational view is a view of all of the data in the OLAP cubes. You can utilize the `bia_rtl_olap.OOS_CUBEVIEW` view to create a physical area for the Oracle Retail Data Model OLAP cubes in an Oracle Business Intelligence Enterprise Edition (OBIEE) repository file (that is, an `rpd` file). Once the OBIEE repository "knows" the OLAP cubes, the OBIEE Server (and therefore any OBIEE client, including as Dashboards, Answers, Delivers and the MS Office Plug-in) to query the ORDM cubes.

Also, when you have installed the Oracle Retail Data Model OLAP component for Oracle Database 11*g*, you can use the OBIEE Plug-in for Analytic Workspace Manager (AWM) with the Oracle Retail Data Model OLAP Cubes. Using this plug-in you can quickly create an OBIEE repository.

Note: The OBIEE Plug-in for AWM is available for download from the Oracle Technology Network Web site at `http://www.oracle.com/technology/index.html`.

Writing Reports on Time-Series and Ranking Analysis

There are two OLAP cubes that hold measures that contain time-series and ranking data for sales and inventory: OOS_SALES and OOS_INV. See *Oracle Retail Data Model Reference* for a list of measures in these cubes.

Writing Forecasting Reports

In Oracle Retail Data Model, OLAP forecasting is performed through OLAP DML programs that call the OLAP DML `FORECAST` command. Two forecast programs, `FORECAST_STOCK_SALES` and `FORECAST_STOCK_INV`, are delivered with Oracle Retail Data Model and reside in the `PSLSINV` analytic workspace. These default forecast programs are executed during an Oracle Retail Data Model historical load.

By default, the forecast programs use two years of Day level data input and generate a forecast for the third year. The results of the forecasts performed by the default forecast programs are stored as measures in two OLAP cubes for sales and inventory: OOS_SALES_FST and OOS_INV_FST. See *Oracle Retail Data Model Reference* for a list of measures in these cubes.

The default forecasting programs produce forecasts of the following types and flavors:

- Moving average method. Computes a series of averages for the values of a dimensioned variable or expression over a specified dimension. For each dimension value in status, MOVINGAVERAGE computes the average of the data in the range specified, relative to the current dimension value. The default forecast programs produce moving average forecasts of the following "flavors":
  - Moving average 500
  - Moving average 3
  - Moving average 10
  - Moving average 10 (weekend days)
  - Moving average 10 (week days)
- Trend method. A straight-line extrapolation of historical data. The default forecast programs produce trend forecasts of the following "flavors":
  - Trend (all days)
  - Trend (weekend days)
  - Trend (week days)
- Exponential method. An extrapolation of historical data using a constant period-to-period percentage growth. The default forecast programs produce exponential forecasts of the following "flavors":
  - Exponential (all days

- Exponential (weekend days)
- Exponential (week days)

- The Holt-Winters method. An extrapolation method that allows for both a linear trend and seasonal fluctuations in the data. OLAP first constructs three statistically related series for each time period of the historical data. Oracle OLAP produces a forecast from the three series for the specified number of periods into the future. The default forecast programs produce Holt-Winters forecasts of the following "flavors":
  - Holt-Wiinters using 364 time periods periodicity (all days)
  - Holt-Winters using 364 time periods periodicity (weekend days)
  - Holt-Winters using 364 time periods periodicity (weekdays)

5 Maintaining an Oracle Retail Data Model Warehouse

This chapter discusses how to refresh an Oracle Retail Data Model data warehouse. It includes the following topics:

- Overview: Maintaining an Oracle Retail Data Model
- Maintaining Relational Tables and Views
- Refreshing OLAP Cube Data

Overview: Maintaining an Oracle Retail Data Model

You need to load your Oracle Retail Data Model data warehouse regularly so that it can serve its purpose of facilitating business analysis. To do this, data from one or more operational systems needs to be extracted and copied into the data 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. 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.

The way you perform these incremental loads varies depending on whether you are maintaining relational tables and views, or OLAP cubes:

- Maintaining Relational Tables and Views
- Refreshing OLAP Cube Data

Maintaining Relational Tables and Views

Once you have implemented an Oracle Retail Data Model data warehouse as described in Chapter 3, "Populating the Oracle Retail Data Model Warehouse", you can administer the relational tables and views in the relational physical model in the same way you administer any other data warehouse.

You perform ETL on a scheduled basis to reflect changes made to the original source system. During this step, you physically insert the new, clean data into the production data warehouse schema, and take all of the other steps necessary (such as building indexes, validating constraints, taking backups) to make this new data available to the end users. Once all of this data has been loaded into the data warehouse, you update the relational materialized views to reflect the latest data.

When you have used OWB for the ETL processes implement the relational physical model, you can use the typical OWB process to perform periodic updates to the relational objects in your Oracle Retail Data Model data warehouse.

Tip: *Oracle Warehouse Builder User's Guide* and Chapter 3, "Populating the Oracle Retail Data Model Warehouse."

Refreshing OLAP Cube Data

Since OLAP cubes do not use the typical ETL workflow process, you cannot use OWB to refresh OLAP cube data.

OLAP cubes are populated through SQL scripts that use the RBIA_OLAP_ETL_AW_LOAD package that is provided with the ORDM OLAP component. The subprograms in the RBIA_OLAP_ETL_AW_LOAD package support two modes of loading OLAP cubes: historical mode and incremental mode. Historical mode is used by the OLAP installation scripts provided with Oracle Retail Data Model. You use incremental mode to refresh OLAP cube data.

Note: An incremental load of OLAP cubes does not recalculate the cubes that hold forecast data. To refresh forecast data, you also need to redesign the forecast process as described in "Updating Forecast Cubes" on page 5-7.

Specifically, to refresh OLAP cubes, you:

1. Write a script that calls the subprograms in RBIA_OLAP_ETL_AW_LOAD package in incremental mode. For an example, see "Sample Incremental Load" on page 5-2.

   See: For detailed information about the RBIA_OLAP_ETL_AW_LOAD package and its subprograms, see *Oracle Retail Data Model Reference*.

2. Execute this script on a regular basis.

   See also : "OLAP Incremental Load Recovery" on page 5-3

Sample Incremental Load

The following code (executed from BIA_RTL_OLAP login) triggers an incremental load of the OLAP cubes.

```
SQL>
set serverout on size 1000000
set linesize 200
set pagesize 0
set timing on
exec cwm2_olap_manager.set_echo_on;
--DATA SETUP Incremental
UPDATE BIA_RTL.DWC_ETL_PARAMETER
SET
FROM_DATE_ETL = TO_DATE('21-JAN-2007', 'DD-MON-YYYY'),
TO_DATE_ETL = TO_DATE('21-JAN-2007', 'DD-MON-YYYY'),
LAST_UPDT_DT = SYSDATE,
LAST_UPDT_BY = USER
WHERE PROCESS_NAME = 'RBIA-INTRA-ETL-OLAP'
```

```
;
COMMIT;
-- Incremental BUILD
declare
aint integer :=100;
begin
  aint := rbia_olap_etl_aw_load.olap_etl_aw_dimbuild('INCREMENTAL', 'EXECUTE', 2);
  aint := rbia_olap_etl_aw_load.olap_etl_aw_build('INCREMENTAL', 'EXECUTE', 2);

  if aint = 0 then
     dbms_output.put_line('Function call build successful');
  else
     dbms_output.put_line('Function call build failed');
  end if;

end;
/
```

OLAP Incremental Load Recovery

When a load fails or is only partially successful, an error occurs. To correct the error, you will need to identify the type of load error and perform the steps necessary to correct it. The following sample scenarios provide examples of common errors and how to correct them.

Note: The scenarios and examples in this section use the OLAP Demo (Sample schema) for Oracle Retail Data Model. The values of the analytic workspace internal partition names, start and end date boundaries for Business Months, and so forth, are dependent on the nature of the Time Calendar scripts installed with Oracle Retail Data Model sample schema installation and are explained from the perspective of the default Time Business Hierarchy which is week based.

A sample scenario where the user or developer needs to perform some clean up activity relates to a partial or failed load. The historical load is always performed on a fresh or empty analytic workspace and hence it can safely be restarted without bothering about the current state of the analytic workspace. One should note that running the Historical Load involves the loss of all data from the analytic workspace and as such it is typically run only once during the implementation life cycle.

The incremental load however runs in an incremental mode as a scheduled process with a definite frequency and only affects the cube for the time range that is being loaded. Hence if an incremental load fails or is partially successful, then there is a need to reload the same after validating the reason for failure.

Cleaning up a partial load is dependent on the load situation and the extent of failure. The sample scenario can illustrate the steps to be taken to re-run the load when it fails or is loaded partially as the following scenarios describe:

- When the incremental load fails completely and none of the records have loaded successfully
- When the incremental load is partly successful, but some records fail to load due to invalid dimension information
- When the incremental load successful, but additional fact information needs to be loaded

- When the incremental load is successful, but an alternate set of data records must be loaded

When the incremental load fails completely and none of the records have loaded successfully

This situation can occur because of missing dimension information. Assuming that missing dimension information has subsequently been made available:

- Possible error due to invalid dimension or foreign key in fact table.
- All Loads usually load dimensions first and then attempt the Facts. Hence the dimension information is missing from the dimension tables and is present in the Fact table. You need to correct the dimension information and include the missing dimension value (wait for the right file, wait for regular intra etl load completion, etc.) and then attempt the load.

When the incremental load is partly successful, but some records fail to load due to invalid dimension information

Assuming that missing dimension information has subsequently been made available:

- Possible error due to invalid dimension or foreign key in fact table.
- Since dimension information has been made available, the incremental load can be attempted once again and the data should get loaded once again. In this case, the successful records would be reloaded once again into the cube. The failed records would get loaded successfully this time.

When the incremental load successful, but additional fact information needs to be loaded

In this case, since the data being loaded is additional (extra records have come in and none of the earlier or existing records are invalid), the incremental load can be attempted once again and the data should get loaded once again.

In this case, the successful records would be reloaded once again into the cube. The additional records would get processed this time and succeed or fail determined by the validity of the data.

When the incremental load is successful, but an alternate set of data records must be loaded

In this case, the earlier set of records loaded into the cube for the incremental load was from an invalid source (wrong file for example) and needs to be undone. This can be due to a single record or multiple records being in error. You need to:

- Undo the effect of the incremental load performed.
- Deleting the faulty records in the source data and reloading the correct set of records will not result in the cube containing only the valid records. The cube will contain all records loaded earlier as well as the current (correct set of) records.
- Data already loaded into the cube remains in its place unless explicitly deleted from the cube.
- To perform the delete action in the cube, execute OLAP DML commands to selectively (partially) clean up the cube of earlier records and to reload the data.
- Using the time range of the incremental data load as the Time dimension boundaries, perform the clean up in the cube.

- For example, assume that 5,000 records have been incrementally loaded into the Sales cube has loaded incrementally for a single day's data (21-JAN-2007), but now you fine that the original data file was invalid. You need to incrementally load a new file of 4,000 records for 21-JAN-2007. (These 4000 records can contain a mixture of modified records from earlier file, new records as well as having earlier records missing entirely from this file). To perform this action, clean up the cube for 21-JAN-2007 and reload the data once again in incremental mode as illustrated in Example 5–1, "Cleaning up and reloading a cube".

Example 5–1 Cleaning up and reloading a cube

Assume that you have populated the PSLSINV analytic workspace using the sample schema data. In Oracle Retail Data Model OLAP analytic workspace PSLSINV, the Sales cube is a cube that is partitioned at the Business Month level for Time dimension, and, when in an Oracle Database 11*g*, has been defined as a cube with compressed composites (11g) and is partitioned at the Business Month level for Time dimension (both 10g and 11g). Hence there is a separate variable that stores all the data of Sales cube for each Business Month. The data for day `21-JAN-2007` resides in the business month level partition which corresponds to Month with the value of `BSNS_MO_20061225`. The partition template for Sales cube (that is, `OOS_SALES_PARTITION_TEMPLATE`), the partition contains the DAY member with the value of `DAY_20070121`.

The solution to reload DAY_20070121 by clearing up previously loaded data is two fold:

1. In the Analytic Workspace Manager, attach the `PSLSINV` analytic workspace in read-write mode, and, then, issue the following OLAP DML commands to clean up the data in Sales cube for partition that contains values dimensioned by DAY_20070121. The actual commands vary depending on the Oracle Database release you are using:

   - In Oracle Database 10*g*, issue the following commands.

   ```
   CLEAR ALL FROM OOS_SALES_PRT_TOPVAR(partition p82)
   UPDATE PSLSINV
   COMMIT
   ```

   - In an Oracle Database 11*g*, issue the following commands.

   ```
   CLEAR ALL FROM OOS_SALES_STORED(partition p246)
   UPDATE PSLSINV
   COMMIT
   ```

2. Reload the data in Sales cube for the time range 25-DEC-2006 and 21-JAN-2007 which corresponds to the Time dimension boundary for the partition which has been purged of all data.

   Login as BIA_RTL_OLAP, then in SQL*Plus issue your commands. The actual commands you issue varies depending on the Oracle Database release you are using:

   - In Oracle Database 10*g*, issue the following commands.

   ```
   set serverout on size 1000000
   set linesize 200
   set pagesize 0
   set timing on
   exec cwm2_olap_manager.set_echo_on;
   --DATA SETUP for incremental load of partition p82 in OOS_SALES cube
   UPDATE BIA_RTL.DWC_ETL_PARAMETER
   ```

```
SET
FROM_DATE_ETL = TO_DATE('25-DEC-2006', 'DD-MON-YYYY'),
TO_DATE_ETL = TO_DATE('21-JAN-2007', 'DD-MON-YYYY'),
LAST_UPDT_DT = SYSDATE,
LAST_UPDT_BY = USER
WHERE PROCESS_NAME = 'RBIA-INTRA-ETL-OLAP'
;
COMMIT;
-- Incremental BUILD for cube: OOS_SALES
declare
aint integer :=100;
begin

  aint := rbia_olap_etl_aw_load.olap_etl_aw_reset_views('Incremental');
  if aint = 0 then
     dbms_output.put_line('Function call resetviews successful');
     aint := rbia_olap_etl_aw_load.olap_etl_aw_dimbuild('INCREMENTAL',
'EXECUTE', 2);

     aint := rbia_olap_etl_aw_load.olap_etl_aw_cubebuild('OOS_SALES.CUBE',
'INCREMENTAL', 'EXECUTE', 2);
     if aint = 0 then
         dbms_output.put_line('Function call cubebuild successful');
     else
         dbms_output.put_line('Function call cubebuild failed');
      end if;

  else
     dbms_output.put_line('Function call resetviews failed');
  end if;

end;
/

*******************
```

- In Oracle Database 11*g*, issue the following commands.

```
set serverout on size 1000000
set linesize 200
set pagesize 0
set timing on

--DATA SETUP for incremental load of partition p246 in OOS_SALES cube

UPDATE BIA_RTL.DWC_ETL_PARAMETER
SET
FROM_DATE_ETL = TO_DATE('25-DEC-2006', 'DD-MON-YYYY'),
TO_DATE_ETL = TO_DATE('21-JAN-2007', 'DD-MON-YYYY'),
LAST_UPDT_DT = SYSDATE,
LAST_UPDT_BY = USER
WHERE PROCESS_NAME = 'RBIA-INTRA-ETL-OLAP'
;
COMMIT;

-- Incremental BUILD for cube: OOS_SALES
declare
aint integer :=100;
begin

  aint := rbia_olap_etl_aw_load.olap_etl_aw_reset_views('INCREMENTAL');
  if aint = 0 then
     dbms_output.put_line('Function call resetviews successful');
     aint := rbia_olap_etl_aw_load.olap_etl_aw_dimbuild('INCREMENTAL',
'EXECUTE', 2);
     aint := rbia_olap_etl_aw_load.olap_etl_aw_cubebuild('OOS_SALES',
'INCREMENTAL', 'EXECUTE', 2);
     if aint = 0 then
        dbms_output.put_line('Function call cubebuild successful');
     else
        dbms_output.put_line('Function call cubebuild failed');
      end if;

  else
     dbms_output.put_line('Function call resetviews failed');

  end if;

end;
/
```

Updating Forecast Cubes

During the initial load of the OLAP cubes, the default Forecasting programs (OLAP DML `FORECAST_STOCK_SALES` and F`ORECAST_STOCK_INV`) populate the Sales Forecast and Inventory Forecast cubes using data from two historical years to forecast one year into the future. To refresh the values in the forecast cubes on an incremental basis then you need to write an incremental load script that executes those programs

When creating forecasts to run intermittently, the main points to consider are the frequency with which you want to run the forecasts and the duration of future Time periods over which you want to forecast. Because forecasting depends on this "yearly" data, typically there is no need to refresh the data in the Sales Forecast and Inventory Forecast cubes as frequently as you refresh the Sales and Inventory cubes. For example, you could schedule the forecasts to execute every month and use the same forecasting Time periods as those used by the historical load script. In this case, you could decide to have the intermittent forecasts overwrite the data in the forecast cubes. On the other hand, you could forecast more frequently, in which case, to avoid overlapping forecasts, you could create new measures to hold the intermittent forecasts-- or even create entirely new forecasts as described in "Creating a New Forecast" on page 5-7.

Note: If you want to replace the data Sales Forecast and Inventory Forecast cubes that was generated during the initial load of the OLAP cubes with new data, you *must* perform another historical or initial load of the OLAP cubes.

Creating a New Forecast

To create a new forecast:

1. Create a new measure to hold the result of the forecast as described in *Oracle OLAP Application Developer's Guide*
2. Open the `PSLSINV` analytic workspace in the Analytic Workspace Manager.

See: Oracle OLAP Application Developer's Guide

3. Within the Analytic Workspace Manager, open the Analytic Workspace Worksheet.
4. Create a new OLAP DML forecast program in one of the following ways:
   - Create an OLAP DML forecasting program that uses the single OLAP DML FORECAST command to perform the forecast. In this case, you can use the forecast programs that are delivered with Oracle Retail Data Model as templates. These programs are named `FORECAST_STOCK_SALES` and `FORECAST_STOCK_INV` and reside in the `PSLSINV` analytic workspace.

     See: `FORECAST` command in *Oracle OLAP DML Reference*

   - Write a new OLAP DML forecasting program with the OLAP DML commands that use a forecasting program.

     See: The discussion on writing a forecasting program in *Oracle OLAP DML Reference*

5. Populate the new forecast by executing the new forecasting program using the OLAP DML CALL command.

   Tip: You can embed OLAP DML commands in a SQL program using the DBMS_AW package.

6. (Optional) Integrate the new forecast program into Oracle Retail Data Model OLAP cube load process.

A Operations Scripts

This appendix provides information scripts that you might find useful when creating your physical data model. It consists of the following topics:

- Calendar Population Script
- Partition Append Scripts
- Bitmap Index for Fact Tables Script
- Create Dimensions Script
- Foreign Key Manipulation Scripts
- Lookup Value Population Scripts
- The Out of Stock Script
- RFMP Calculation Script

Calendar Population Script

The Calendar population scripts consist of a one-time installation script named `calendar_population.sql` that:

1. Prepares some necessary changes on the schema.
2. Creates the `Calendar_Population` package that contains following procedures:
   - `RUN` (the main procedure).
   - `RBIW_Base_Time_Tables_ddl` creates the base table needed to support multiple hierarchies: Business or Calendar.
   - `RBIW_Populate_Time_Hier_Bsns(`*`in_setup_start_date`*`, `*`in_setup_no_years`*` )` sets up the data in base table for the Business hierarchy as specified in setup or install section.
   - `RBIW_Populate_Time_Hier_Clndr(`*`in_setup_start_date`*`, `*`in_setup_no_years`*`)` sets up the data in base table for the Calendar hierarchy as specified in setup or install section.
   - `RBIW_Time_hier_Star` sets up the Time hierarchy reporting layer tables.
   - `RBIW_Time_Views` sets up the Time hierarchy reporting layer views, star and hybrid snowflake views.
   - `RBIW_Populate_Time_Transform` populates the Time transformation tables using the base Time tables or views created above. It populates transformation data for both hierarchies: Business and Calendar.

Executing the Calendar Scripts

To populate calendar data:

1. Go to `ORACLE_HOME`/*ORDM/PDM/Relational/SQL_Scripts/Calendar*.
2. Log in to BIA_RTL user compile.
3. Execute the following SQL statements.

   ```
   @calendar_population.sql
   exec  Calendar_Population.run(date,num_years);
   ```

   where:

   `date` is the start date with which you want to populate calendar data. It is of type CHAR and should be input in the format 'YYYY-MM-DD' (for example, `'2005-05-18'`).

   `num_years` is the number of years to populate calendar data.It should be INTEGER.

Bitmap Index for Fact Tables Script

The `gen_bitmap.sh` script is a Linux or UNIX Shell script. It generates a script that creates the on fact tables (Base, Derived, and Aggregate).

`gen_bitmap.sh` is located in the directory `ORACLE_HOME/ORDM/PDM/Utilities/Bitmap_Generation`.

`gen_bitmap.sh` includes the function `add_bitmap_index`. `add_bitmap_index` takes two parameters: `Table Name` and `Column Name`.

`gen_bitmap.sh` calls `add_bitmap_index` for all fact tables and columns to create bitmap indexes. Once the script is finished, it creates a SQL script named `add_bitmap.sql` in the DDL directory.

Partition Append Scripts

You use the partition scripts to append partitions. The partition scripts consist of the `gen_script_add_partition.sh` which is a shell script that generates two other scripts:

- `add_partition_tbs.sql` that creates tablespaces
- `add_partition.sql` that adds partitions

Executing the Partition Append Scripts

Take the following steps:

1. Go to `ORACLE_HOME/ORDM/PDM/Utilities/Partition_Generation`.
2. Login into BIA_RTL user
3. Invoke the script `GEN_SCRIPT_ADD_PARTITION.SH` using two parameters where the first parameter is the start year and the second parameter is the end year as shown in the following example.

   ```
    [oracle@zeta oracle]$ gen_script_add_partition.sh 1997 2005
   ```

4. After the Partition Appending scripts are generated, run the SQL script in SQL*Plus to add new partitions to each partitioned table as follows:

a. Login into BIA_RTL user.

b. Run the scripts in SQL*Plus as shown in the following code:

```
SQL> @add_partition_tbs.sql
SQL> @add_partition.sql
```

Create Dimensions Script

By default, Oracle Retail Data Model creates three dimensional objects for major dimensions in the BIA_RTL schema: product, time, organization. The dimensional object gives advantages in the query rewrite. Consequently, in the future, reporting tools such as BIEE might import the dimension definition from the database instead of creating them from the beginning each time.

The dimensional object scripts are one-time installation scripts. By default, Oracle Retail Data Model uses three scripts to create hierarchies for three dimensions: organization (`ORG_DIM.sql`), product (`PRO_DIM.sql`) and time (`TIM_DIM.sql`). The script tries first to drop dimensions, and then creates them.

To execute the Dimensional Object Script

To create major dimensions yourself:

1. Go to *`ORACLE_HOME`*`/ORDM/PDM/Relational/SQL_Scripts/Dimensional_Object`
2. Login into BIA_RTL user.
3. Run the `Create_Dimensions` script by issuing the following SQL*Plus command.

   ```
   SQL> @Create_Dimension.sql
   ```

4. Check the `Create_Dimension.spool` for errors after the execution.

Foreign Key Manipulation Scripts

The `generate_fk_script.sql` script, which is generated according to RBIA schema, allows you to enable or drop all foreign keys at the same time. The `generate_fk_script.sql` script generates two other SQL scripts:

- `create_fk_constraint.sql` creates of all the Foreign Key Constraints in the BIA_RTL Schema which is convenient when working with the ETL. Together with the `drop_fk_constraints.sql` script, you can maintain the foreign key; or, if you want to, disable and then enable some of foreign keys (on some tables).
- `drop_fk_constraint.sql` drops the Foreign Key Constraints in the BIA_RTL Schema. If you choose to drop only some Foreign Key constraints on certain tables (for example, when you did the ETL for a table that failed because of reference integrity), you can search the script by table name and run only the script for that one table.

Note: The `generate_fk_script.sql` script is provided “as is”; its output is not guaranteed. This script is useful when you are trying to disable all foreign key constraints in the schema for testing or ETL purpose.

Executing the Scripts to Create and Drop Foreign Keys

To execute these scripts:

1. Go to *`ORACLE_HOME`*`/ORDM/PDM/Relational/SQL_Scripts/Create_Drop_FK_Constraint/`
2. Login into BIA_RTL user.
3. Run the two scripts in SQL*Plus as shown in the following code.

```
SQL> @create_fk_constraints.sql
SQL> @drop_fk_constraints.sql
```

Lookup Value Population Scripts

The Lookup Value population scripts are one-time installation scripts that populate Seed value for lookup tables:

- `insert_LookupOthers_record.sql` inserts values into the physical lookup table.
- `insert_LookupViews_record.sql` inserts values into a master code table, which then provides data for the various Lookup Views.

All lookup tables are divided into two groups:

- Lookup tables with multiple levels. Each of these are implemented as one physical table. The `insert_LookupOthers_record.sql` script inserts seed values into each of the physical lookup tables.
- Lookup tables with only one level that are used purely for lookup. These are implemented into one master code table. There are views that present the value of each table. The script `insert_LookupViews_record.sql` inserts seed values into the master code table.

Executing the Lookup Population Scripts

To insert lookup data, take the following steps:

1. Go to *`ORACLE_HOME`*`/ORDM/PDM/Relational/SQL_Scripts/Lookup_Value_Insert`.
2. Log in to the BIA_RTL Schema.
3. Execute the two scripts as shown in the following code.

```
SQL> @insert_LookupOthers_record.sql
SQL> @insert_LookupViews_record.sql
```

4. Check spool files `insert_LookupOthers_record.spool` and `insert_LookupViews_record.spool` for errors.

The Out of Stock Script

The Out of Stock scripts are:

- `Out_Of_Stock_SQL_Model.SQL` which is a script that creates the materialized view that keeps the forecast value for sales quantity and stock on hand.
- `STOCK_Modified_SQL_MODEL_Query.SQL` which is a version of the model script (which was the first version of the script) that contains comments and modified SQL.

The Out of Stock scripts are located in the 'Out of Stock' folder (*ORACLE_HOME*/ORDM/PDM/Relational/SQL_Scripts).

Included with the Out of Stock scripts is `Out_Of_Stock_SQL_Model.xls` which is a spreadsheet that explains the logic of the forecast. In the most general terms, the Out of Stock files use the following calculation model:

1. Get the average SLS_QTY for around 5 days last year: 'SLS_QTY_LP'
2. Get the average SLS_QTY for around 5 days the year before last year: SLS_QTY_LLP.
3. Use SLS_QTY_LP and increment percentage for last year to get the forecast sales quantity.

   FCST_SLS_QTY=SLS_QTY_LP*(SLS_QTY_LP/SLS_QTY_LLP)
4. Get the forecast stock on hand (FCST_SOH_QTY) in the same way
5. Forecast out of stock=FCST_SOH_QTY-FCST_SLS_QTY By now, the data forecast is for two days: 20070101 and 20070102.

As written, the script forecasts data for two days. (in this case, 20070101 and 20070102). To forecast more data, make changes to the script.

RFMP Calculation Script

The RFMP calculation script is a one-time installation script that is integrated into the Installer and the Intra-ETL packages. The script contains the following:

- `Pop_rfmp` procedure:
  1. Creates two TMP tables for rotated calculation.
  2. Populates tmp_1 with New Month data filtered by Month+Busn_unit
  3. Performs three calculation loops for Recency, Frequency, Monetary. (The fourth calculation loop for Profit is optional.)
  4. Copies the result into DWD_CUST_RFMP_SCR table.
- Following the `pop-rfmp` procedure are two PLSQL blocks which, in turn:
  1. `Populate_DWD_CUST_RFMP_SCR` that calls `pop_rfmp` with all `bsns_unit` code.
  2. Create a job for the `rfmp` procedure to run every day.

Executing the RFMP Script

To run the RFMP script separately, take the following steps:

1. Go to *ORACLE_HOME*/ORDM/PDM/Relational/SQL_Scripts/RFMP.
2. Login to the BIA_RTL Schema.
3. Run the two scripts using the following code:

```
SQL> @RFMP_Population.SQL
SQL> EXEC RFMP_Population.RUN
```

Index

A

Associate Basket Analysis Model, 4-2
Associate Loss Analysis Model, 4-3
Associate Sales Analysis Model, 4-5

B

bitmap index, creating in Oracle Retail Data Model, A-2

C

calendar values, populating, A-1
Customer Category Mix Analysis Model, 4-6
Customer Loyalty Analysis Model, 4-7
customizing Oracle Retail Data Model, 2-2

D

data mining models, Oracle Retail Data Model
- creating, example, 3-11
- creation packages, 3-11
- reports, 4-2, 4-3, 4-5, 4-6, 4-7, 4-8, 4-9
- *See also* names of individual models

data mining source tables, Oracle Retail Data Model
- populating, 3-9

data models, Oracle Retail Data Model
- about, 1-4
- *See also* physical data model, Oracle Retail Data Model

data warehouse, Oracle Retail Data Model, 1-5
- maintaining, 5-1
- populating, 3-1

dimensions, Oracle Retail Data Model, A-3

F

fit-gap analysis, Oracle Retail Data Model, 2-1
forecasting in Oracle Retail Data Model, 4-12

I

installation
- Intra-ETL for Oracle Retail Data Model, 3-3

Intra-ETL process, Oracle Retail Data Model
- error conditions, 3-7
- executing, 3-2
- monitoring, 3-6
- nstallation, 3-3
- recovering, 3-8
- recovery during, 3-7

Item Basket Analysis Model, 4-8
Item POS Loss Analysis Model, 4-9

M

maintaining Oracle Retail Data Model warehouse, 5-1
materialized views, Oracle Retail Data Model, 3-3
modifying Oracle Retail Data Model, 1-6

O

OLAP cubes, Oracle Retail Data Model
- forecast cubes, 4-12, 5-7
- forecasting, 4-12
- incremental load recovery, 5-3
- OBIEE repository for, 4-11
- populating, 3-13
- querying, 4-11
- refreshing, 5-2
- reporting on, 4-11, 4-12

OLAP in Oracle Retail Data Model
- installation and configuration, 3-13

Oracle Retail Data Model
- about, 1-1
- customizing, 2-2
- data warehouse, 1-5
- materialized views, 3-3
- modifying, 1-6
- retail organizations supported, 1-2

Oracle Warehouse Builder, Oracle Retail Data Model project
- configuring, 3-4
- deploying, 3-5
- executing, 3-5
- importing, 3-4
- using, 3-3

P

partitions, Oracle Retail Data Model

appending, A-2
physical data model, Oracle Retail Data Model
customizing, 2-3
populating
tables, Oracle Retail Data Model, 3-2
populating calendar data, A-1

R

reports, Oracle Retail Data Model
customizing, 4-10
delivered with, 4-1
writing, 4-11
retail organizations, supported by Oracle Retail Data Model, 1-2
RFMP script, Oracle Retail Data Model, A-5

S

scripts
operations, A-1

T

tables, Oracle Retail Data Model
aggregate, 3-2
base, 3-2
data mining, 3-9
derived, 3-2
lookup, 3-2, A-4
maintaining, 5-1
populating, 3-2
reference, 3-2