

Oracle® Retail Data Warehouse

Implementation Guide

Release 13.1

June 2009

The Programs (which include both the software and documentation) contain proprietary information; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. This document is not warranted to be error-free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose.

If the Programs are delivered to the United States Government or anyone licensing or using the Programs on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and we disclaim liability for any damages caused by such use of the Programs.

Oracle, JD Edwards, PeopleSoft, and Siebel are registered trademarks of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Value-Added Reseller (VAR) Language

Oracle Retail VAR Applications

The following restrictions and provisions only apply to the programs referred to in this section and licensed to you. You acknowledge that the programs may contain third party software (VAR applications) licensed to Oracle. Depending upon your product and its version number, the VAR applications may include:

- (i) the software component known as **ACUMATE** developed and licensed by Lucent Technologies Inc. of Murray Hill, New Jersey, to Oracle and imbedded in the Oracle Retail Predictive Application Server - Enterprise Engine, Oracle Retail Category Management, Oracle Retail Item Planning, Oracle Retail Merchandise Financial Planning, Oracle Retail Advanced Inventory Planning, Oracle Retail Demand Forecasting, Oracle Retail Regular Price Optimization, Oracle Retail Size Profile Optimization, Oracle Retail Replenishment Optimization applications.
- (ii) the **MicroStrategy** Components developed and licensed by MicroStrategy Services Corporation (MicroStrategy) of McLean, Virginia to Oracle and imbedded in the MicroStrategy for Oracle Retail Data Warehouse and MicroStrategy for Oracle Retail Planning & Optimization applications.
- (iii) the **SeeBeyond** component developed and licensed by Sun Microsystems, Inc. (Sun) of Santa Clara, California, to Oracle and imbedded in the Oracle Retail Integration Bus application.
- (iv) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.
- (v) the software component known as **Crystal Enterprise Professional and/or Crystal Reports Professional** licensed by SAP and imbedded in Oracle Retail Store Inventory Management.
- (vi) the software component known as **Access Via™** licensed by Access Via of Seattle, Washington, and imbedded in Oracle Retail Signs and Oracle Retail Labels and Tags.
- (vii) the software component known as **Adobe Flex™** licensed by Adobe Systems Incorporated of San Jose, California, and imbedded in Oracle Retail Promotion Planning & Optimization application.
- (viii) the software component known as **Style Report™** developed and licensed by InetSoft Technology Corp. of Piscataway, New Jersey, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.
- (ix) the software component known as **DataBeacon™** developed and licensed by Cognos Incorporated of Ottawa, Ontario, Canada, to Oracle and imbedded in the Oracle Retail Value Chain Collaboration application.

You acknowledge and confirm that Oracle grants you use of only the object code of the VAR Applications. Oracle will not deliver source code to the VAR Applications to you. Notwithstanding any other term or condition of the agreement and this ordering document, you shall not cause or permit alteration of any VAR Applications. For purposes of this section, "alteration" refers to all alterations, translations, upgrades, enhancements, customizations or modifications of all or any portion of the VAR Applications including all reconfigurations, reassembly or reverse assembly, re-engineering or reverse engineering and recompilations or reverse compilations of the VAR Applications or any derivatives of the VAR Applications. You acknowledge that it shall be a breach of the agreement to utilize the relationship, and/or confidential information of the VAR Applications for purposes of competitive discovery.

The VAR Applications contain trade secrets of Oracle and Oracle's licensors and Customer shall not attempt, cause, or permit the alteration, decompilation, reverse engineering, disassembly or other reduction of the VAR Applications to a human perceivable form. Oracle reserves the right to replace, with functional equivalent software, any of the VAR Applications in future releases of the applicable program.

Contents

Preface	ix
Audience	ix
Related Documents	ix
Customer Support	ix
Review Patch Documentation	x
Oracle Retail Documentation on the Oracle Technology Network	x
Conventions	x
 1 Introduction	
What is RDW and Data Warehousing?	1-1
 2 Technical Architecture	
Oracle Retail Data Warehouse Architecture Diagram and Descriptions	2-2
Dimension Processing	2-3
Fact Processing	2-3
Process to Update Records in RDW	2-4
Normal Update Description	2-4
Incremental Update Description (Applicable to Fact Processing Only)	2-5
Repository and Catalog	2-5
 3 Setup and Configuration	
RDW Time	3-1
Time Calendar (4-5-4)	3-1
Time Calendar (4-5-4/Gregorian)	3-1
Time Calendar (13-Period)	3-2
Differentiators	3-2
Diff Attributes	3-2
Creating New Diff Types	3-3
Modifying Diff Types	3-3
RETL Setup	3-3
Environment Variables	3-3
Business Date	3-4
rdw_config.env Settings	3-4
Sizing information	3-6
Overview	3-6

Factors to Consider	3-6
Data Seeding of Positional Facts	3-8
Data Migration from a Legacy Data Warehouse System	3-9
4 Security	
5 Compression and Partitioning	
Overview of Compression	5-1
What Compression Does.....	5-1
The Mechanics of Compression	5-2
Compressed Tables and CUR Tables	5-2
Coping with Major Changes.....	5-3
Factclosedm.ksh	5-3
Factopendm.ksh	5-3
Overview of Partitioning Strategies	5-4
Implementing RDW Partitioning.....	5-5
An Example of Setup and Maintenance for Partitioning RDW Compressed Inventory Tables Using Warehouse Partitioning	5-5
Implementing Partitioning for Compressed Inventory Tables	5-6
How Oracle Implements Partitions	5-9
Summary	5-10
6 RDW Integration with Other Applications	
Oracle Retail Merchandising System	6-2
Dimension Data	6-2
Fact Data	6-2
A Note About Local Currency and Facts	6-3
Oracle Retail Invoice Matching (ReIM)	6-3
Oracle Retail Price Management (RPM)	6-3
Client-Supplied Data.....	6-3
RDW Integration with Oracle Retail Workspace	6-4
SSO Launch from ORW.....	6-4
Listing and Launching RDW Reports and Dashboards from ORW.....	6-5
Displaying RDW Reports and Dashboards in ORW Dashboards	6-5
7 Batch Scheduling	
Identifying RETL Programs by Application.....	7-1
Identifying Source Files for RETL Programs and their Corresponding Target Tables.....	7-1
Removing RETL Programs from RDW Batch Schedule	7-1
Setting up the Batch Schedule	7-2
Batch Schedule Dependencies	7-2
Batch Error Notification Setup	7-2
8 RDW Performance	
Key Factors in Performance	8-1
Purging and Archiving Strategy	8-1

Aggregations.....	8-2
Report Design	8-3
ETL Batch Process	8-3
Decompression View Performance	8-4
Creating and Using the Oracle BI Decompressed View	8-4
Using the Oracle BI Decompression Views for the Price Analysis (A) Report.....	8-9
Partitioning Strategy	8-12
Database Configuration	8-12
Adequate Hardware Resources	8-12
Additional Factors.....	8-12
Caching Considerations for RDW	8-13
Leading Practices	8-13
Customizations.....	8-13
RETL Best Practices.....	8-14
Oracle BI EE or Oracle BI SE One Best Practices	8-14
Batch Schedule Best Practices.....	8-14
Automation	8-14
Recoverability	8-14
High Availability	8-14
Batch Efficiency	8-15

9 Frequently Asked Questions

Preface

The Oracle Retail Data Warehouse Implementation Guide provides detailed information useful for implementing the application. It helps you to view and understand the behind-the-scenes processing of the application.

Audience

The Implementation Guide is intended for Oracle Retail Data Warehouse application integrators and implementation staff.

Related Documents

For more information, see the following documents in the Oracle Retail Data Warehouse 13.1 documentation set:

- *Oracle Retail Data Warehouse Data Model*
- *Oracle Retail Data Warehouse Installation Guide*
- *Oracle Retail Data Warehouse Release Notes*
- *Oracle Retail Data Warehouse Operations Guide*
- *Oracle Retail Data Warehouse User Guide*

See also:

- *Oracle Retail Merchandising Batch Schedule*
- Oracle Retail Extract, Transform, and Load documentation

Customer Support

- <https://metalink.oracle.com>

When contacting Customer Support, please provide:

- Product version and program/module name
- Functional and technical description of the problem (include business impact)
- Detailed step-by-step instructions to recreate
- Exact error message received
- Screen shots of each step you take

Review Patch Documentation

If you are installing the application for the first time, you install either a base release (for example, 13.0) or a later patch release (for example, 13.0.2). If you are installing a software version other than the base release, be sure to read the documentation for each patch release (since the base release) before you begin installation. Patch documentation can contain critical information related to the base release and code changes that have been made since the base release.

Oracle Retail Documentation on the Oracle Technology Network

In addition to being packaged with each product release (on the base or patch level), all Oracle Retail documentation is available on the following Web site (with the exception of the Data Model which is only available with the release packaged code):

http://www.oracle.com/technology/documentation/oracle_retail.html

Documentation should be available on this Web site within a month after a product release. Note that documentation is always available with the packaged code on the release date.

Conventions

The following text conventions are used in this document:

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

Introduction

This RDW version contains enhanced localization functionality and works in conjunction with the Oracle Retail Extract Transform and Load (RETL) 13.0 framework. This architecture optimizes a high performance data processing tool that lets database batch processes take advantage of parallel processing capabilities.

With the implementation of RETL, the RDW client benefits from the following capabilities:

- Parallel computing technology
 - Promotes the flexibility of a stand-alone solution
 - Lets database batch processes take full advantage of parallel processing capabilities
 - Increases scalability, leveraging parallel processing of both the system and database server (reads, writes, performs transformations and aggregations)
- Expanded use of Application Programming Interfaces (API): Allows for easier customization
- Elimination of table triggers: Reduces the burden on the source system
- Extensible Markup Language (XML) scripts: Facilitate the framework's ability to process fact and dimension data by using valid operators
- Streamlined ETL Code: Provides for less data storage, easier implementation, and reduced maintenance requirements through decreased code volume and complexity

What is RDW and Data Warehousing?

A data warehouse is a physical place, a database, where you can place data from a transactional system, such as Oracle Retail Merchandising System (RMS), for the purpose of querying that data. In order to work with RDW, you start by populating it with existing data from source systems such as RMS, Oracle Retail Price Management (RPM), and Oracle Retail Invoice Matching (ReIM).

RDW uses sophisticated techniques to populate the data warehouse. Explained in greater detail throughout this guide, these techniques include taking the data provided by source systems (such as RMS) and then rapidly transforming that data and loading it into the data warehouse. Techniques used to load data into the warehouse vary depending upon whether the data consists of 'facts' or 'dimensions'.

Understanding the differences between fact and dimension data depends first upon understanding data processing in a data warehouse. RDW uses an online analytical processing (OLAP) application that serves as an interface to your data, giving it

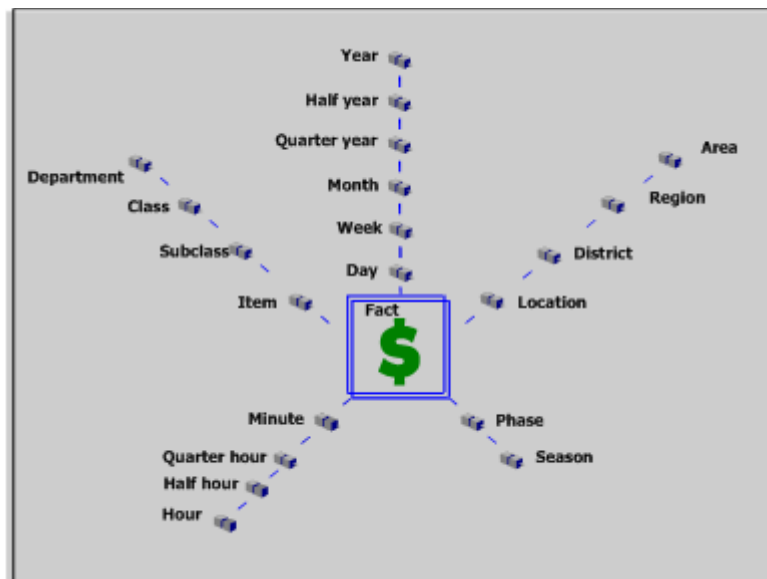
meaning through pre-designed and custom queries and reports. The data warehouse itself supports these queries by structuring data in a useful schema. Note that the word 'schema' in this context is an industry-standard term that refers to the way in which data is modeled and organized throughout a data warehouse and should not be confused with the 'schema files' that are described later in this document. (For more information about schema files, see the latest *Oracle Retail Extract, Transform, and Load Programmer's Guide*.)

At the center of this schema is fact data. Facts are the transactions that occur in your data warehouse's source systems, such as RMS. You might want to look at sales transaction facts, or inventory stock count facts at stores or warehouses, or inventory movement facts.

Facts have little meaning by themselves because they are usually just values, that is: six sales at a store, 15 items left at a warehouse, or 300 items transferred. What gives fact data true meaning in RDW is the intersection of dimensions in which facts exist. In other words, six sales on Wednesday at store B, or 15 dishwashers in stock last Monday at the Chicago warehouse, or 300 blouses transferred during the last week in February from the St. Louis warehouse to the Denver warehouse. Dimension data, therefore, exists in the data warehouse to serve as reference data to facts.

The schema of a data warehouse illustrates its data elements and their inter-relationships. The following graphic describes the schema used in RDW:

Figure 1–1 Snowflake Schema in RDW



The RDW schema, the 'snowflake schema', starts out as a star with a fact in the middle surrounded by rays pointing out from the center. These points are the dimension data that give meaning to the fact by serving as points of reference.

RDW contains far greater volumes of fact data than it does dimension data. Besides being more abundant than dimensions, facts change constantly as new data enters the database. Dimension data, on the other hand, changes much less frequently. New stores need to be added into the data warehouse much less frequently than new sales transactions (fact data) that need to be processed daily. Because of the different natures of fact and dimension data, RDW employs different techniques to load and manipulate the data.

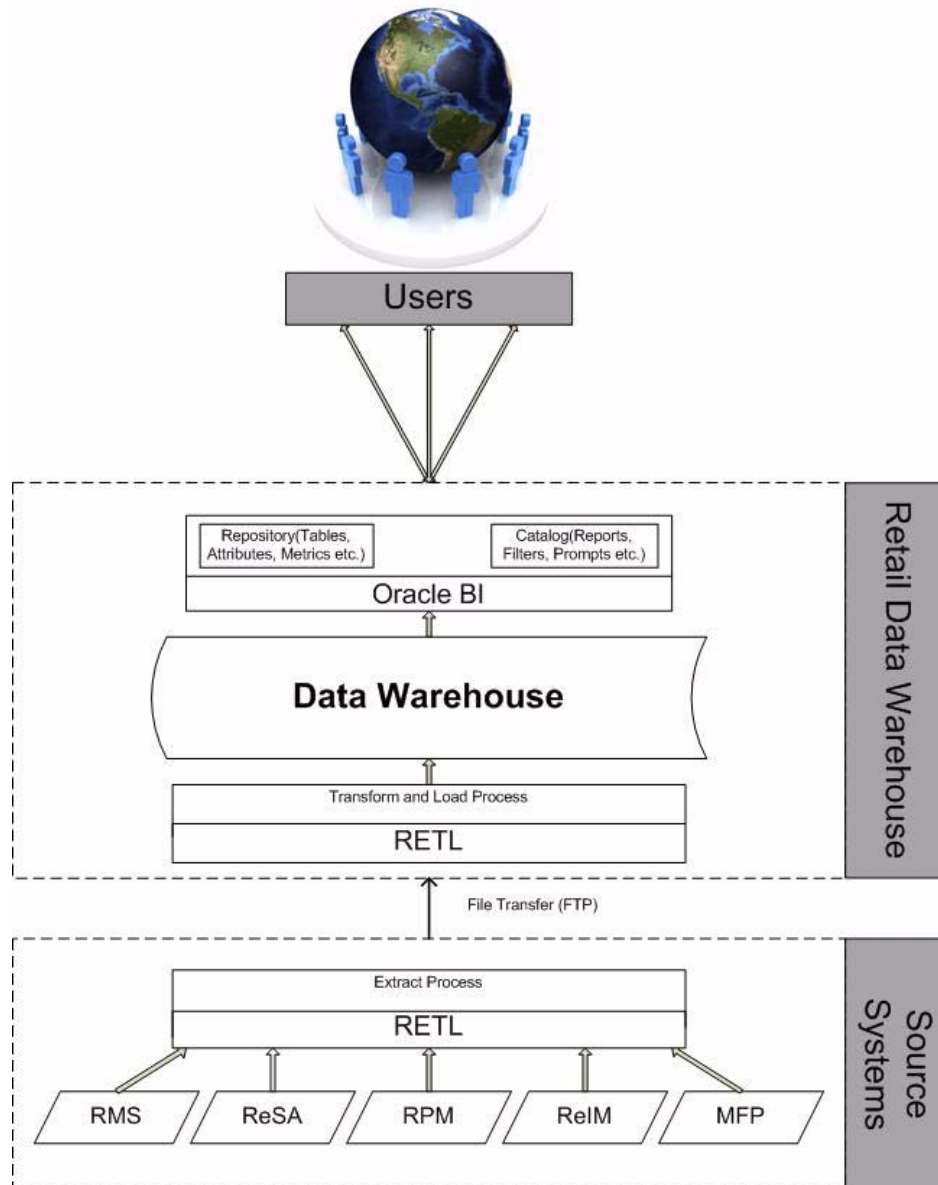
Technical Architecture

This chapter describes the overall software architecture for RDW. A high-level discussion of the general structure of the system is included. From this content, integrators can learn about the parts of the application and the interaction between the parts.

Oracle Retail Data Warehouse Architecture Diagram and Descriptions

The following diagram explains the overall architecture of RDW.

Figure 2–1 RDW Architecture Diagram



RDW uses Oracle Business Intelligence (BI) as a reporting tool and Oracle Retail Extract, Transform, and Load (RETL) as ETL tool.

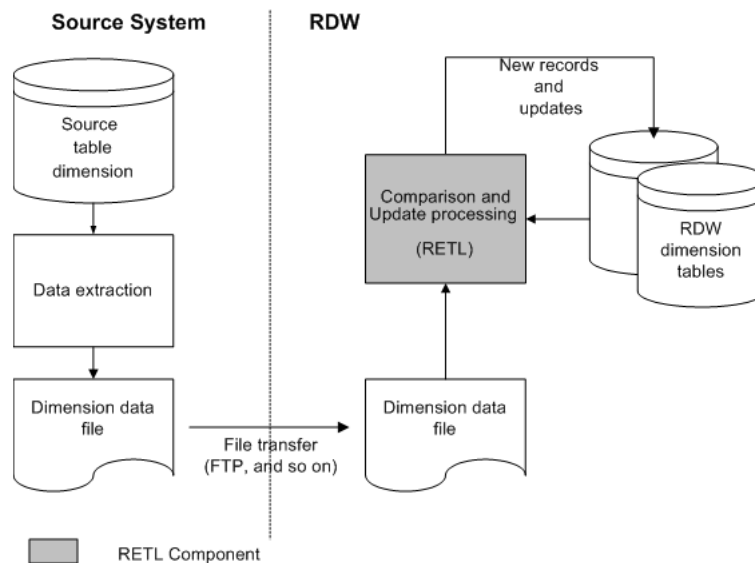
In this section, processing of RDW core components are described, which includes: dimension processing, fact processing, process to update records and repository/catalog.

Dimension Processing

The following diagram illustrates the dimension processing architecture that is employed in RDW. The process involves comparing a dimension data file that contains the current snapshot of the applicable source system table with the historical data in RDW. This comparison eliminates the need to capture frequent dimension changes as they occur in the source system over the course of the day. The comparison is performed on the RETL framework and written back directly to the data mart tables in RDW. The source dimension data file can be created for retailers that have the Oracle Retail source applications by running the data extraction programs that are packaged with the applications. Retailers without these source applications must provide the data files from their source system in the format recognized by RDW (see the *RDW Operation Guide*).

Note: Dimension data extraction programs are available for retailers with Oracle Retail source applications (that is, RMS and RPM). These programs are packaged with the applications.

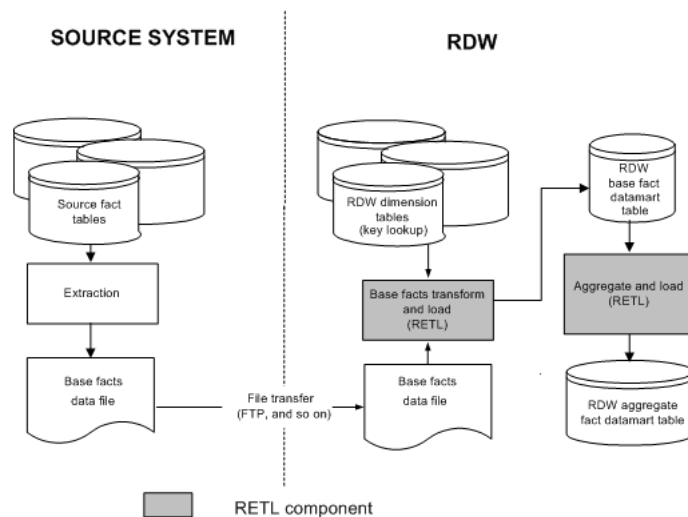
Figure 2–2 Dimension Processing in RDW



Fact Processing

Fact data provided via a base fact data file is transformed, aggregated, and loaded directly into RDW data mart tables without the need of staging tables. The source fact data file can be created for retailers that have Oracle Retail source applications by running the data extraction programs that are packaged with these applications. Retailers without the Oracle Retail source applications must provide data files from their source system in the format recognized by RDW.

Note: Fact data extraction and transformation programs are available for retailers with Oracle Retail source applications (that is, RMS and ReIM).

Figure 2–3 Fact Processing in RDW

Process to Update Records in RDW

Updates into the database are accomplished through one of two processes, depending upon whether a normal update or an incremental update is occurring. An incremental update (applicable to fact processing only) is one that sums the incoming records with the old records in the target table and replaces those old records with the new summed records. A normal update is one that uses incoming records to replace old records in the target table. There are two ways to perform normal updates. The first way is 'delete, insert' logic. The second way is using the Oracle 'merge' statement. Oracle 'merge' is used on larger volume tables to improve performance.

Note: The temporary tables that are mentioned throughout this operations guide are always dropped by the batch code every day, after the various batch processes that use the temporary tables complete.

Normal Update Description

Update using 'delete, insert' logic:

1. The dataset (containing the new records) is written into a temporary table.
2. This temporary table is used to determine which of the old update records in the target table should be deleted.
3. The old records are deleted from the target table.
4. The new records are inserted into the target table.

Update using 'merge' logic:

1. The dataset (containing the new records) is written into a temporary table.
2. This temporary table is used to determine which of the old update records in the target table should be merged (updated).
3. The old records are updated and new records are inserted into the target table by using merge statement.

Incremental Update Description (Applicable to Fact Processing Only)

1. The dataset (containing the new records) is written to a temporary table.
2. The records to be updated are read from the target table and a second temporary table (temporary table 2) is created.
3. The temporary table 2 is used to determine which of the old update records in the target table should be deleted, and those records are deleted.
4. The records in the temporary table and in temporary table 2 are combined to form a new dataset.
5. The new dataset is grouped by the primary keys of the target table to sum up the required fact fields.
6. The resulting dataset is written to the target table (that is, the records are inserted into the target table).

Repository and Catalog

Warehouse is the single source for the Front End Metadata. All the tables/views from the warehouse are imported to Oracle BI and properly joined according to the data model. Once the physical tables are joined, a Business model is defined where attributes, metrics and hierarchies are created, which are then visible to the end user via Presentation model. RDW Metadata exists in the form of two files.

- Repository
 - This file contains Physical tables, their joins, Business model, Attributes, Dimensions (hierarchies), Metrics and Presentation layer.
 - Repository is accessed via the Oracle BI Administration tool, which is mostly used by the power users or the developers to define the Physical, Business and Presentation models. Example modifying/creating physical joins or metrics or attributes etc. For more details about this interface refer to *Oracle Business Intelligence Server Administration Guide*.
- Catalog
 - This file contains Answers, Dashboards, Dashboard prompts, Filters etc.
 - Catalog is accessed via Oracle BI Presentation Service, which is used by the end users like the Analysts to execute/modify/create Dashboards, Answers, or Filters etc through the web browser. For more details about this interface refer to *Oracle Business Intelligence Presentation Services Administration Guide*.

Setup and Configuration

RDW Time

Time Calendar (4-5-4)

RDW provides support for the retail 4-5-4 calendar. The fiscal 4-5-4 calendar is the calendar supported by RMS and other Oracle Retail applications and is populated in RDW via an extract from RMS. The 4-5-4 calendar is the default calendar used when viewing the time dimension via the Oracle BI middle tier layer of RDW.

Note: Even if 4-5-4 is the default calendar for RDW, you can still see the Gregorian Time attributes and transformations from within Oracle BI. However, you can only utilize these objects if you have opted for Gregorian calendar during database installation.

Time Calendar (4-5-4/Gregorian)

RDW provides support for the combined 4-5-4 calendar/Gregorian calendar. If a user chooses to use the combined 4-5-4/Gregorian calendar, the user must execute the batch program that generates the text files needed to populate the Gregorian time dimension.

Note: In order to update RDW to report in combined 4-5-4/Gregorian calendar, see the Create Time in RDW appendix in the *RDW Installation Guide* for specific instructions. Nothing needs to be modified within Oracle BI.

Time Calendar (13-Period)

The 13-Period calendar can also be used, but RMS does not support it. If a user chooses to use the 13-Period calendar, the user can either provide a flat file with its 13-period time, or utilize a sample 13-period time flat file, and then ETL scripts populate the time dimension with this file during RDW installation. Within the middle-tier application, all references to Month must be manually updated to Period, to comply with the 13 period nomenclature.

Use the following procedure to update RDW to report in 13 period time:

1. Update the time dimension tables. (For complete instructions, see the *RDW Installation Guide*.)
2. Within the repository, delete the Half Year attribute, and re-name the objects referencing month.
 - Highlighting the object in the Business Layer and pressing the delete key deletes the Half Year attribute. All relationships and hierarchies are automatically updated. The TIME_HALF_DM table and its aliases can also be deleted the same way in the physical layer. Global Consistency check is recommended after the change.
 - Each repository object referencing Month can be re-named by selecting the object and pressing F2. After all the changes, Global Consistency check is recommended before saving the repository.

Differentiators

Diff Attributes

As part of the RDW install, the first five diff types are hard-coded in both the DIFF_TYPE_DM look-up table and in the names of the attributes that point to them. This matches the initial set of pre-defined diff types that are provided within the Oracle Retail Merchandising System (RMS). This leaves the remaining diff types, and their corresponding attributes, to be defined at the user's discretion at a later date.

The following is a list of differentiators:

Table 3–1

DIFF_TYPE_KEY	Attribute Name
1	Size
2	Color
3	Flavor
4	Scent
5	Pattern

Creating New Diff Types

With each additional diff type that is added to the merchandising system, a new attribute, within RDW, should be created to suitably match the name of the new diff type. Perform the following procedure to create new Diff Type attributes:

1. Open the RDW.rpd.
2. Create a new logical column in the Dim Product Differentiator Logical table of the RDW.rpd business layer. The first time, this will be Diff 6 desc.
3. Once completed, save the repository.

Modifying Diff Types

Functionality exists within the merchandising system to modify existing diff types. This can be done by either changing the description of the diff type (that is, change the diff type "Size" to "Fashion Size") or by deleting the diff type altogether.

When the description is changed within the merchandising system, the batch process updates the description column within the PROD_DIFF_TYPE_DM table to match this new description. The attribute name within the repository must be updated manually by changing the attribute name.

When deleting a diff type within the merchandising system, the batch process updates the description column within the PROD_DIFF_TYPE_DM table to 'NULL'. The attribute name within the repository must be updated manually to the generic name corresponding to the DIFF_TYPE_KEY, by renaming. That is, if Color was deleted, the system updates the name of the Color attribute to DIFF 2.

RETL Setup

- Ensure that RETL is configured per the RETL documentation. The 'retl -v' command should be run at the UNIX prompt to confirm that you are pointing to the proper RETL executable version.
- Verify that the RETL executable is in the path of your UNIX session by typing the following:

```
%which retl
```

Environment Variables

See the *RETL Programmer's Guide* for RETL environment variables that must be set up for your version of RETL. You need to set MMHOME to your base directory for RDW RETL. This is the top level directory that you selected during the installation process. MMUSER, PASSWORD, and ORACLE_SID should also be set up. In your profile script, you should add lines such as the following:

```
export MMHOME=<base directory for RMS ETL>\rdw13.0\dev
export MMUSER=rdw13dev
export ORACLE_SID=rdw13
```

Business Date

- The CURR_LOAD_DT column in the MAINT_LOAD_DT_DM table should be modified to the date before the first dimension/fact loading (that is, if you plan to load data and have all the items on the first day of history have a dm_recld_load_dt of '20000101', then the curr_load_dt should be updated to '19991231').
- Run mt_prime.ksh before the dimension and fact modules to update the CURR_LOAD_DT column in the MAINT_LOAD_DT_DM table to the intended dimension/fact load date (that is, the curr_load_dt before mt_prime.ksh is '19991231', and then after mt_prime.ksh the curr_load_dt is updated to '20000101').

rdw_config.env Settings

The rdw_config.env is an environment configuration script that is used by RDW programs. There are a number of settings within this script, some of which are client configurable, some of which are not. Listed below are the configurable variables which a client must set prior to running the RDW batch.

Note: Clients should weigh the benefits of these settings. Clients may wish to test different settings to find the best solution for their unique environment and needs.

- DBNAME refers to the name of the database (ORACLE_SID).
- DB_OWNER refers to the owner of all tables used by RDW.
- BA_OWNER refers to Oracle batch user.
- ORACLE_PORT refers to the port number the database server is using. RETL uses this variable to find a correct database.
- ORACLE_HOST refers to which server the database is on. RETL uses this variable to find a correct database.
- LOAD_TYPE refers to the SQL*Loader method used to load data to the database. Direct load can be used when you have a large amount of data to load quickly. A direct path load can quickly load and index large amounts of data. It can also load data into either an empty or non-empty table. Conventional, not direct load, is the Oracle and RDW default. There are restrictions on when direct load can be used. Before using direct load, clients should research direct vs conventional load, and the restrictions to using direct load.
 - **LOAD_TYPE=conventional:** loads the data using the conventional SQL-loader method for Oracle.
 - **LOAD_TYPE=direct:** loads the data using the direct SQL_loader method for Oracle.
- CONN_TYPE refers to the Database connection method. It is either Oracle Real Application Clusters (RAC) enabled or disabled. The default is RAC disabled.
 - **CONN_TYPE=thin:** Real Application Cluster disabled
 - **CONN_TYPE=oci:** Real Application Cluster enabled
- IM_PROCESS refers to the method used to load item dimensional data. A number of factors determine which method performs best, including the volume of items a client has and how often changes are made to items. The higher the ratio of item changes to total items, the more likely one would benefit from the full snapshot

method. Clients should test the methods to see what performs best for them. The default value for this setting is full snapshot.

- **IM_PROCESS=D**: loads the data using delta mode, which processes item data based on daily changed data only.
- **IM_PROCESS=F**: loads the data using snapshot mode, which processes items based on a full snapshot of all data.
- **LOOK_TYPE** refers to the method that RETL uses to look up item key. This setup depends on the number of items that clients have, the hardware capacity, as well as a number of other variables. Clients should test the two methods in their environment to see which method performs best. As a rule, 'lookup' typically performs better for clients with a smaller number of items (less than 1M) and 'innerjoin' typically Data from Undefined Sources performs better for clients with a larger number of items (more than 1M). The default value for this setting is lookup.
 - **LOOK_TYPE=innerjoin**: looks up item key using RETL 'innerjoin' operator.
 - **LOOK_TYPE=lookup**: looks up item key using RETL 'lookup' operator.
- **PARA_NUM** refers to the parallel degree used in SQL statement. The parallel degree can affect the performance of a module. Currently it is only used for inventory aggregation module invblddm.ksh. The setup of this value depends on client system capacity. Clients should test it different values to select the best. It is suggested this value be half the number of CPUs available on the server if this server is only used for RDW. To disable parallelism, set this setting to a value of '0'. The default value '0'.
- **LANGUAGE** refers to which language is to be used. The default is set to 'en' for English.
 - **LANGUAGE=pt_br**: Brazilian Portuguese
 - **LANGUAGE=zh**: Chinese (Simplified)
 - **LANGUAGE=zh_tw**: Chinese (Traditional)
 - **LANGUAGE=en**: English
 - **LANGUAGE=fr**: French
 - **LANGUAGE=de**: German
 - **LANGUAGE=it**: Italian
 - **LANGUAGE=ja**: Japanese
 - **LANGUAGE=ko**: Korean
 - **LANGUAGE=ru**: Russian
 - **LANGUAGE=es**: Spanish
 - **MILITARY** refers to if military or non-military time is loaded to RDW time dimension tables. The default is set to 'NON_MILITARY'.
 - * **MILITARY=MILITARY**: Military time is loaded to RDW time dimension tables.
 - * **MILITARY=NON_MILITARY**: Non-military time is loaded to RDW time dimension tables.
- **NUMOPENDAYS** refers to the number of days a store must be open before becoming comparable.

Sizing information

Sizing Information - This section provides a list of factors that should be taken into account when making sizing plans.

Overview

RDW13 Components - There are two major hardware components that make up the RDW13 physical environment:

Middle Tier Application Server - The middle tier application server hosts software components Oracle Application Server (OAS) and Oracle Business Intelligence - Enterprise Edition (EE) or Oracle Business Intelligence Standard Edition One (SE One).

Back End Database - The back end Oracle Database stores large amounts of data that are queried in generating Oracle BI reports. The daily data loading process and report query processing process are both heavily dependent on the hardware sizing decision.

Factors to Consider

- Application Server

Report Complexity - Reports processed through Oracle BI can range from very simple one-table reports to very complex reports with multiple-table joins and in-line nested queries. The Application Server receives data from the back-end database and converts it into report screens. The mix of reports that will be run will heavily influence the sizing decision.

Number of Concurrent Users - The RDW13 application is designed to be a multiple concurrent use system. When more users are running reports simultaneously, more resources are necessary to handle the reporting workload. For more details on Clustering and Load Balancing refer to the Clustering, Load Balancing and Failover in Oracle Business Intelligence chapter of the *Oracle Business Intelligence Enterprise Edition Deployment Guide*.

- Back End Database

Functions Used Determine Tables to be populated - RDW13 is designed to be a functional system so that some functions (like supplier compliance or order processing) that are available do not have to be used. To the extent that some functions are not used, the amount of resources may be reduced correspondingly.

Fact Tables and Indexes - Disk space is required for tables and indexes. Please refer to the data model to identify the database objects necessary for the functions selected.

Dimension Tables and Indexes - Dimension tables and indexes also require space and generally indicate the size of the data to be stored. Disk space must be planned on the basis of record counts in the dimension tables.

Data Purging Plan - How Much Data to be Stored - The number of years of data to be stored also contributes to the amount of disk space required. Disk space to store fact data is generally linear with the number of years to data to be stored.

Database Backup and Recovery - The importance of the data and the urgency with which a recovery must be made will drive the backup and recovery plan. The backup and recovery plan may have a significant impact on disk space requirements.

- Data Storage Requirements

Transaction Volume - Sales - the higher the number of sales records, the higher the disk storage requirements and the higher the resource requirements to process queries against sales-oriented tables and indexes

Positional Data - Inventory, Price, Cost - Positional data (data that is a snapshot at a specific point in time, like inventory data "as of 9:00AM this morning") can result in very large tables. The RDW concept of data compression (not to be confused with database table compression) is important in controlling the disk space requirement. Review [Chapter 5, "Compression and Partitioning"](#) for more information.

Extract, Transform, Load - Daily Processing - The daily loading process is a batch process of loading data from external files into various temporary tables, then into permanent tables, and finally into aggregation tables. Disk space and other resources are necessary to support the ETL process.

Data Reclassification Requirements - More frequent hierarchy reclassification impacts resources.

Processing Report Queries - Report queries submitted to the back-end database have the potential to be large and complex. The size of the temporary tablespace and other resources are driven by the nature of the queries being processed.

- Configuration issues

Archivelog mode - If the database is being operated in archivelog mode, then additional disk space is required to store archived redo logs.

SGA and PGA sizing - the sizing of these memory structures is critical to efficient processing of report queries, particularly when there are multiple queries running simultaneously.

Initialization Parameters - Particular attention should be made to initialization parameter settings to optimize both daily data loading and report query processing

Data Storage Density - As is the case with many data warehouses, the data stored in the RDW13 back-end database is relatively static and dense storage of data in database data blocks results in more efficient report query processing.

- Hardware Architecture

Number and Speed of Processors - More and faster processors speeds both daily data loading and report query processing in the back-end database. The application server needs fewer resources than the back-end database.

Memory - More memory enables a larger SGA in the back-end database and will reduce the processing time for report queries.

Network - Since the data from the report queries needs to go from the back-end database to the application server, a faster network is better than a slower network, but this is a relatively low priority.

Disk - RPMs, spindles, cache, cabling, JFS - I/O considerations are most critical to optimal performance. Selection of disk drives in the disk array should focus on speed. Faster RPM's, more spindles, larger cache, fiberoptic cabling, JFS2 or equivalent.

RAID - The selection of a RAID configuration is also a critical decision. In particular, RAID5 involves computations that slows Disk I/O. The key is to select

the RAID configuration that maximizes I/O while meeting redundancy requirements for data protection.

Backup and Recovery - The backup and recovery strategy drives disk configuration, disk size, and possibly the number of servers if Dataguard or Real Application Clusters are used.

Sizing is customer-specific. The sizing of the RDW13 application is sensitive to a wide variety of factors so therefore sizing must be determined on an individual installation basis.

Testing is essential. As with any large application, extensive testing is essential for determining the best configuration of hardware.

Database tuning is essential, just like any other database. The back-end Oracle database is the most critical performance and sizing component of RDW13. As with any database installation, continuously monitoring database performance and activity levels and continuously tuning the database operation on an ongoing basis are essential for optimal performance.

Data Seeding of Positional Facts

For base level positional fact data, RDW uses compression approach to reduce the data volume. Compression in RDW refers to storing physical data that only reflects changes to the underlying data source, and filling in the gaps between actual data records through the use of database views. For detailed information about compression, refer to [Chapter 5, "Compression and Partitioning"](#).

To report positional data correctly in the RDW front end, data seeding is required if clients launch RDW later than RMS, which is the source system of RDW. For performance reasons, all date range partitioned positional fact tables are also highly recommended to seed data on the first date or first week of each partition to avoid searching data acrossing partitions. Data used for seeding can either come from RMS or from client legacy systems. The following is some recommendations for clients to seed data:

- If seeding data is for new added partition, clients can run RDW script `orapartseed.ksh` to seed new partition. This script moves seed data from RDW CUR tables to new partitions.
- If seeding data is for new tables, clients may need to provide snapshots of their positional fact data. This data is used as base loading source files. With these source files, clients can run base loading programs to populate and seed positional fact tables.
- Some of the snapshots of positional data can come from RMS tables if RMS has history data for them. Pricing is one of example for this. Client can get pricing snapshot data for a certain date from RMS PRICE_HIST table.
- Some of the snapshots of positional data may not be available from RMS tables (such as Inventory stock on hand, in transit) as RMS does not hold history information for them. To get this kind of data, clients can use daily transaction data from RMS TRAN_DATA_HISTORY to calculate.

Data Migration from a Legacy Data Warehouse System

RDW fact tables may not have data at the same granularity as a client has in its legacy system. The granularity of client history data can be either higher or lower than what RDW data model supports.

- If the granularity of client history data is lower than what the RDW data model supports, the client can aggregate data to the same level that the RDW data model supports and then populate the RDW base table.
- If the granularity of client history is higher than what the RDW data model supports, the client can aggregate data to the RDW aggregation tables if they are available. This could cause inconsistencies between RDW base tables and RDW aggregation tables within the legacy time period. When the client reports data on those time periods, the client has to be aware of the inconsistencies between base level and high aggregation level.
- RDW provided APIs can be used for designing and developing the data extraction programs from legacy system. Detailed information of APIs can be found in the *RDW Operations Guide*.

The Oracle BI server provides secure access control at many levels. The Oracle BI Security Manager displays all security information for a repository. Oracle BI Security Manager allows you to set up users and groups, synchronize LDAP users and groups, set access privileges for objects such as metrics, logical tables and columns, set filters on data, and set up a managed query environment which can even have a control over when users can access data. RDW recommends implementing the security features available within Oracle BI which can prevent unauthorized users accessing certain data or objects. For details on security features and benefits refer to the Security in Oracle BI chapter of the *Oracle Business Intelligence Server Administration Guide*.

Another security feature available in Oracle BI is Single Sign-On. Oracle BI is integrated and certified with Oracle Single Sign-On (OSSO). Once OSSO is enabled in Oracle BI, user authentication happens on the Oracle SSO server. For additional details, refer to the Configuring RDW and Oracle BI for Single Sign-on appendix of the *Oracle Retail Data Warehouse Installation Guide*.

Compression and Partitioning

This chapter describes how RDW implements compression, and offers a discussion of Oracle partitioning.

Overview of Compression

Although data warehouses are often very large, the amount of detail generated in some RDW tables is enormous even by usual standards. That is, a retailer with 500,000 items and 500 locations would generate 250,000,000 new rows each day. Storing this amount of uncompressed data is impractical from a disk storage perspective, both in the cost to store the rows and in the cost to perform backups and other database maintenance operations.

One approach that RDW uses to reduce the data volume is compression. This chapter describes:

- What compression does
- The mechanics of compression
- Which tables are currently compressed
- The Oracle features that are related to compression
- Strategies for implementing compressed tables

What Compression Does

Compression refers to storing physical data that only reflects changes to the underlying data source, and filling in the gaps between actual data records through the use of database views. This method is engaged primarily for subject areas that are perpetual, such as inventory. That is, when querying sales data, a valid sale record either exists (a sale occurred) or a record does not exist (no sale occurred). However, when querying for on-hand inventory, even if no change occurred to the inventory on the date desired, a valid value is still required. One way to resolve this discrepancy is to store a record for every day for every valid item-location combination as mentioned above. Another method, compression, allows for the storage of only changes to the inventory position. The query is resolved by looking backward through time from the desired date (if no change record exists on that date) until an actual change record is found. This method returns the correct current data with the minimum requirements necessary for processing and storing data.

The Mechanics of Compression

The purpose of decompression views is to give the application the illusion that there is a record for each possible combination (that is, an item-location-day record for each permutation) when in fact there is not. Thus, the fact of whether a table is compressed or not should not be visible to the application that queries data from that table.

A compressed table is made up of two distinct parts: a 'seed' that consists of all existing combinations at a point in time (typically the first day or week of the table or partition) and the changed data since that time.

When resolving a query for a particular record, the decompression view provides the latest record for the requested item and location with the maximum day that is less than or equal to the requested day. A decompression view needs to encompass both the seed and all of the changed data since that seed.

To illustrate how the decompression views actually work, assume the following: I am interested in the inventory position of item 10 at location 10 on 1/23/02. The seed was done on 1/1/02. Changes were posted on 1/4/02, 1/15/02, and 1/30/02. The row that is presented to the application by the decompression view is the row on 1/15/02, because it is the greatest date that is less than or equal to the requested date. As a second example, assume that the inventory position of item 10, location 10, day 1/3/02 was desired. Because there was no change record less than or equal to the desired date, the seed record from 1/1/02 would be presented to the application.

Compression's performance is excellent when the user is querying for a single day (as in the example above). When querying over a group of days, however (that is, all of the inventory positions at a given location on a given day), the performance can be unacceptable. Even though the user is requesting a group of information back and in most cases the database can process groups of information efficiently, each individual row must be evaluated individually by the decompression view and cannot be processed as a group. To counteract the slow performance of these summary operations, clients may take advantage of compressed table partition seeding (see the passage, "[Overview of Partitioning Strategies](#)" later in this chapter).

This partition seeding utilizes CUR tables (also known as 'current' tables). An example is the INV_IL_CUR_DM table, which holds the current decompressed position for every item and location on the INV_ITEM_LD_DM table. This position can be used as a partition seed. This position is also utilized by base RDW code during major change fact seeding (see the passage, "[Factopendm.ksh](#)" later in this chapter).

Compressed Tables and CUR Tables

The table below illustrates the compressed tables within RDW, along with their corresponding CUR tables.

Compressed Tables	CUR Tables
CMPTR_PRICING_ITEM_LD_DM	CMPTR_PRICING_IL_CUR_DM
COST_ITEM_SUPP_LD_DM	COST_ISL_CUR_DM
EXCHNG_RATE_CRNCY_DAY_DM	(No CUR table)
INV_ITEM_LD_DM	INV_IL_CUR_DM
INV_ITEM_LW_DM	(No CUR table)
INV_UNAVL_ITEM_LD_DM	INV_UNAVL_IL_CUR_DM
NET_COST_SUPP_ITEM_LD_DM	NET_COST_SIL_CUR_DM

Compressed Tables	CUR Tables
ORG_LOC_WK_MTX_DM	(No CUR table)
PRICING_ITEM_LD_DM	PRICING_IL_CUR_DM
SPACE_ALLOC_DEPT_LD_DM	SPACE_ALLOC_DL_CUR_DM
SPACE_ALLOC_ITEM_LD_DM	SPACE_ALLOC_IL_CUR_DM
SUPP_AVAIL_ITEM_DAY_DM	SUPP_AVAIL_ITEM_CUR_DM
SUPP_CNTRCT_ITEM_DAY_DM	SUPP_CNTRCT_I_CUR_DM

Coping with Major Changes

Factclosedm.ksh

On a compressed fact table, a record is only posted to the table when there is a change in one of the fact attributes. If there is no activity, no record is posted. Decompression views then fill in the gaps between physically posted records so the front end can see a fact record for each item-location-day combination. However, when an item or location or department is closed or major-changed, any fact record with those dimensions becomes inactive. The decompression views need to be informed to stop filling in the gap after the last record was posted. To accomplish this instruction, factclosedm first queries the PROD_ITEM_RECLASS_DM, ORG_LOC_RECLASS_DM, and PROD_DEPT_RECLASS_DM tables (see the section, "Factopendm.ksh" later in this chapter) to determine what compressed item-location facts need to be closed today. Factclosedm then inserts a 'stop record' that has a DM_RECD_STATUS_CDE = 'X'. The decompression view fills in records up to the day that a status code of 'X' is posted. The close record is inserted for tomorrow's DAY_IDNT, indicating that the fact record is no longer valid beginning tomorrow, when the newly seeded record (from factopendm.ksh) becomes active. In the case of the one compressed week table, INV_ITEM_LW_DM, factclosedm inserts close records for next week's WK_IDNT.

Factopendm.ksh

RDW Data Compression tables require seeding when a major change in the product and/or organization dimension causes new surrogate keys to be created for items or locations. Seeding the compressed tables is required because the new key represents a new hierarchy relationship. If the new key is not represented on the compressed table, the compression view does not pick up any data from the day the old dimensions were closed to the day a record with the new dimensions is posted to the compressed fact tables. This missed data causes inaccuracy in query results and incorrect data aggregation.

There are two steps to this seeding process. First, the programs prditmdm.ksh, prddepdm.ksh, and orglocdm.ksh run as part of the dimension loading process to populate temporary tables PROD_ITEM_RECLASS_DM, PROD_DEPT_RECLASS_DM, and ORG_LOC_RECLASS_DM. Next, the program factopendm.ksh looks at the tables for reclassified items, departments, or locations. It seeds all of the compressed tables with records that contain the reclassified ITEM_KEYS, DEPT_KEYS and/or LOC_KEYS.

Overview of Partitioning Strategies

This section describes partitioning strategies for RDW data marts. Although optional, partitioning provides powerful performance benefits, and therefore is very highly recommended. Tables in the RDW13_partitioned_tables.xls spreadsheet (see the *Oracle Retail Data Warehouse Installation Guide*) are highly recommended to be partitioned. If a report runs slowly and a fact table in the query is not partitioned, this fact table may be a good candidate for partitioning. For large tables, for instance the inventory, pricing, cost and sales tables, splitting them into table partitions can provide the following benefits:

- Partitions are smaller and therefore easier to manage
- Management operations on multiple partitions can occur in parallel
- Partition maintenance operations (such as index rebuilds) are faster than full table operations
- Partition availability is higher than table availability (that is, when recovering a particular partition, users may access all other partitions of the table at the same time)
- The optimizer can prune queries to access data in only the partition of interest, not the entire table (that is, if you are interested only in February's data, you do not need to look at any of the table's data outside of the February partition)
- Partitions are separate database objects, and can be managed accordingly (that is, if December sales are frequently accessed throughout the year whereas other months are not, the December sales partition could be located in a special tablespace that allows for faster disk access)
- In some situations, Oracle can create parallel operations on partitions that it cannot on tables; an example is joining between two different tables if they are partitioned on the same key (this feature is called a 'parallel partition-wise join')

Indexes, as well as tables, can be partitioned. Index partitions can be global (one index over the table, regardless of whether the table is partitioned or not) or local (there is a one-to-one correspondence between index partitions and table partitions). In general, when tables are partitioned, local indexes should be preferred to global indexes for the following reasons:

- Maintenance operations involve only one index partition instead of the entire index (that is, if the oldest table partition is aged out, a local index partition can be dropped along with its corresponding table partition, whereas an entire global index would need to be rebuilt after it becomes unusable when a table partition is dropped).
- The optimizer can generate better query access plans that use only an individual partition.
- When multiple index partitions are accessed, the optimizer may choose to use multiple parallel processes rather than just one.

Implementing RDW Partitioning

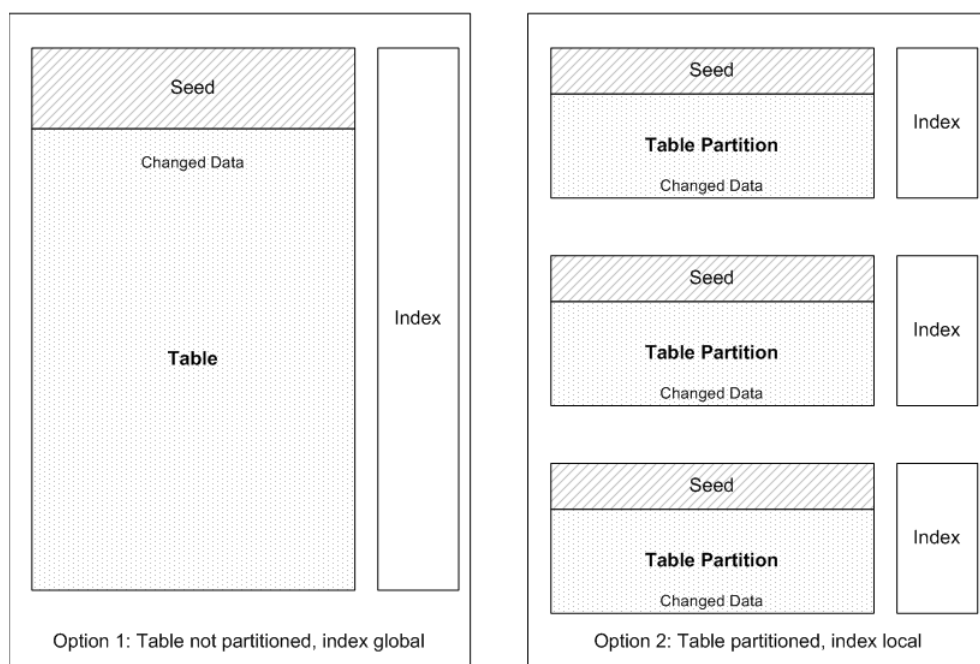
For those clients who choose to partition a fact table, the diagrams on the following page illustrate some of the possibilities for table and index layout.

In general, option 2 is the preferred solution for large regular or compressed tables (for example, the INV_ITEM_LD_DM and INV_ITEM_LW_DM tables). It uses table partitions and local indexes, thus minimizing the impact of index maintenance and the deletion of old table partitions. Global index on partitioned tables are not recommended.

Option 1 is viable for smaller compressed tables (typically, all other compressed tables besides INV). The disadvantage is that, functionally, there is no way to delete historical data and the table continues to grow.

See the *Oracle Retail Data Warehouse Installation Guide* for details on how to implement RDW partitioning.

Figure 5–1



An Example of Setup and Maintenance for Partitioning RDW Compressed Inventory Tables Using Warehouse Partitioning

1. Make the following determinations, among others (see the *Oracle Retail Data Warehouse Installation Guide* for details):
 - Your partitioning strategy
 - The time period your partitions will use
 - The 'values less than' boundaries according to your calendar
 - How many partitions are to be used
 - The partition naming standard

2. On the database, create the partitions and indexes for the tables you want to partition. See the *Oracle Retail Data Warehouse Installation Guide* for details.
3. Verify you have populated the Time Calendar Dimension. See the *Oracle Retail Data Warehouse Installation Guide* for details.
4. Perform step numbers 2 and 3 whenever any of the following events occur:
 - Records are added to or deleted from the Time Calendar tables TIME_DAY_DM or TIME_WK_DM (extending time calendar for new time period).
 - Partitions are added to the Inventory Position tables INV_ITEM_LD_DM or INV_ITEM_LW_DM.
5. Other maintenance activities will include archive and remove of partitions.

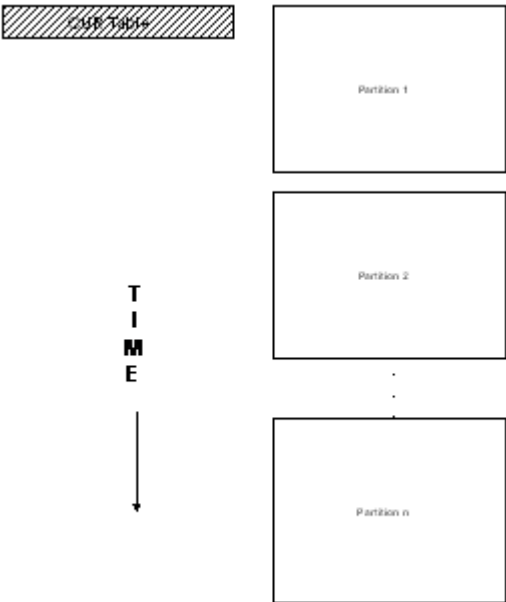
Implementing Partitioning for Compressed Inventory Tables

Once the tables (including partitions) and indexes have been created, the data must be loaded. For tables that have a corresponding CUR table (such as INV_ITEM_LD_DM and INV_IL_CUR_DM), the recommended steps are described in the following text and shown in the diagram on the following page:

1. Populate the INV_IL_CUR_DM table with data. (This population is accomplished the first time that invliddm.ksh runs and an inv position record is processed.)
2. Copy this INV_IL_CUR_DM table as the seed to the first partition. (This step is performed automatically by the orapartseed.ksh program.)
3. At this point, only the changed records are added to the INV_ITEM_LD_DM table, whereas the INV_IL_CUR_DM table is a full, uncompressed version that holds the current inventory position as of the last time period.
4. When a partition boundary is crossed, the INV_IL_CUR_DM table is copied as the seed to the new partition, via the orapartseed.ksh program.

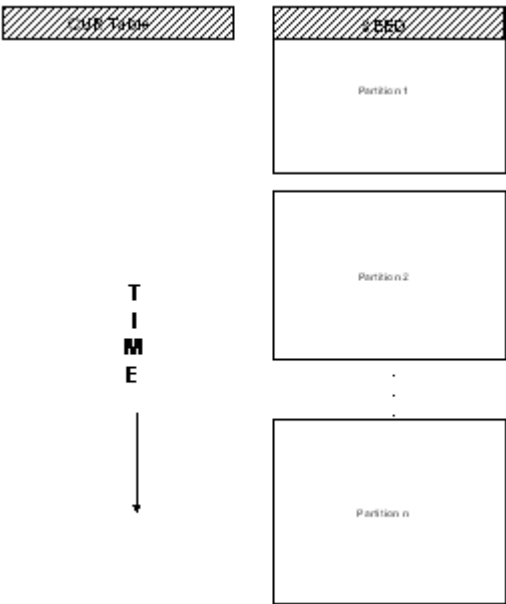
If you have questions about how to implement partitioning with compression or would like assistance implementing partitioning, contact Oracle Customer Support or Oracle Retail Services.

Figure 5-2



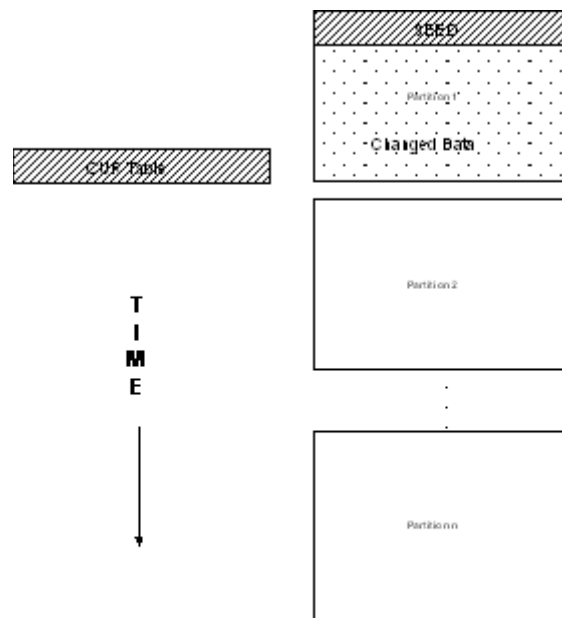
Step 1. Populate the CUR table.

Figure 5-3



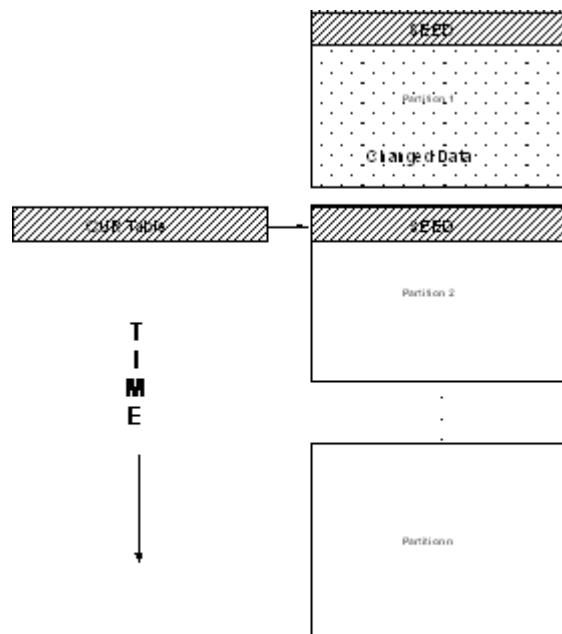
Step 2. Copy the CUR table to the end of the new partition.

Figure 5–4



Step 3. As time period goes by, the CUR table is maintained while the partition only has changes.

Figure 5–5



Step 4. As a partition boundary is crossed, copy the CUR table to the seed of the new partition.

How Oracle Implements Partitions

Note: Refer to the Partitioning in Data Warehouses chapter of the *Oracle10g Data Warehousing Guide* for further details regarding partitioning concepts.

Oracle range partitions are split by a range of values on the partition key. Examples include partitions by month, partitions by department number, and partitions by item range. Oracle expanded the partitioning options to include hash partitions (spreading the rows across a fixed number of partitions by applying a hash function to the partition key), as well as composite partitioning (a combination of range partitioning and hash partitioning). Oracle Retail recommends that clients partition their tables using range partitioning. Oracle Retail also recommends that the partition key be the date field in the primary key to allow partitions to be aged out when no longer needed.

The Oracle Retail general guideline is that partitioning should be considered for tables in the RDW13_partitioned_tables.xls spreadsheet (see *Oracle Retail Data Warehouse Installation Guide*) and any fact tables in a slow-running query. There is an administrative trade-off between having more partitions to manage and obtaining the benefits of partitioning.

The actual physical layout of partitions varies from site to site. A general approach is to put each partition into its own tablespace and map each tablespace to a separate mount point. This has several advantages:

- Maintenance operations, as well as tablespace recovery, can occur on a partition while other partitions are unaffected.
- If manual performance tuning of the data files is being done, tablespaces and their files can be moved around to achieve optimal performance.
- If partitions are no longer being updated, their tablespaces can be changed to READ ONLY, which greatly reduces backup requirements.
- Separate mountpoint pointing to separate set of physical drives will significantly reduce I/O time.

Partitions are ordered from low values to high values. The partition key value for a partition is a non-inclusive upper bound (high value) for that partition. That is, if the SLS_ITEM_LD_DM table is partitioned by month, the high value for the January, 2000, partition is 01-Feb-2000. A low value can always be inserted into the lowest partition. However, a high value may not be able to be inserted depending on the high value of the highest partition. For instance, if the highest partition has a high value of 01-Feb-2000, and the insertion of a record is attempted with a date of 01-Feb-2000, the row cannot be inserted into the table (remember that the high value of 01-Feb-2000, is a non-inclusive upper bound). For this reason, Oracle allows a special high value partition with a key of MAXVALUE. Oracle Retail recommends that all partitioned tables include a dummy partition with a MAXVALUE high value.

There are special considerations for the partitioning of RDW compressed tables. The following is a brief description of the different partition maintenance commands. Please see the current Oracle database documentation set for more details:

- **ADD PARTITION:** Adds a new partition to the high end of a partitioned table; because Oracle Retail recommends having a MAXVALUE partition, and this is the highest partition, the ADD PARTITION functionality can be achieved by performing a SPLIT of the MAXVALUE partition instead.

- **DROP PARTITION:** Drops the partition; this is the typical method to delete the oldest partitions (those with the lowest values) as they age to maintain a rolling window of data.
- **EXCHANGE PARTITION:** Converts a non-partitioned table into a partitioned table or converts a partitioned table into a non-partitioned table.
- **MERGE PARTITION:** Merges two adjacent partitions into one.
- **MOVE PARTITION:** Moves a partition to another segment; this is used to defragment a partition or to change its storage characteristics.
- **SPLIT PARTITION:** Splits an existing partition by adding a new partition at its low end.
- **TRUNCATE PARTITION:** Removes all rows from the partition.

Oracle database automatically maintains local index partitions in a 1-to-1 correspondence with their underlying table partitions. Any table partition operations, such as ADD PARTITION, also affect its appropriate index partitions.

Summary

Partitions are useful for breaking up large tables into smaller, more manageable pieces. Oracle Retail offers the following recommendations for partitions:

- Consider partitioning tables that are in the RDW13_partitioned_tables.xls spreadsheet (see the *Oracle Retail Data Warehouse Installation Guide*) and fact tables that are in a slow- running query.
- Use the date as the partition key for range partitioning.
- When tables are partitioned, make their indexes local.
- Consider putting each partition in its own tablespace and each tablespace on its own mountpoint.
- After updates on a partition cease, consider changing its tablespace to READ ONLY to reduce backup requirements.
- If partitioning compressed tables, be sure to address these special requirements for seeding.

RDW Integration with Other Applications

This chapter provides a functional summary of data interfaces with RDW.

RDW receives information from various source systems. For the majority of RDW functionality, there are Oracle Retail applications that can serve as these source systems. These Oracle Retail applications are the following:

- Oracle Retail Merchandising System (RMS)
- Oracle Retail Invoice Matching (ReIM)
- Oracle Retail Price Management (RPM)

Packaged with these Oracle Retail applications is the code to extract data from these systems and provide that data in a text file format which RDW can recognize and then load into RDW tables via RDW RETL programs.

Retailers who have not purchased Oracle Retail applications, or have purchased them and simply chosen not to use them as an RDW data source, can use their own source systems and code to provide RDW with the identical text files that the Oracle Retail source applications provide. APIs/schema files associated with this method are illustrated in the Application Programming Interface (API) flat file specifications appendix of the *Oracle Retail Data Warehouse Operations Guide*.

The following interfaces are described in this section:

- Oracle Retail Merchandising System (RMS)
 - Dimension data
 - Fact data
- Oracle Retail Invoice Matching (ReIM)
- Oracle Retail Price Management (RPM)
- Client-supplied data
- Loaded at install: Voucher age dimension and Time like for like transformations

All data comes into RDW as text files. See the Application Programming Interface (API) flat file specifications appendix of the *Oracle Retail Data Warehouse Operations Guide* for a complete list of the RDW API specifications and their business requirements.

RDW and RDW reports can be configured to launch through Oracle Retail Workspace (ORW). ORW provides a single point of access to Oracle Retail applications used by your business. It provides an integrated platform that provides operational and analytical information through dashboards and reports from both internal and external sources.

Oracle Retail Merchandising System

The Oracle Retail Merchandising System (RMS) can be the principle source of the RDW dimension and fact data. RMS is the retail client's central transactional processing system. RMS data feeds to RDW can be broken down into dimensions and facts. General descriptions of each are contained in this section.

Dimension Data

RMS can be the sole source of organization and product dimension data, and it supplies the majority of other dimension data as well.

The dimension data process extracts current dimension data from RMS using RETL scripts that are packaged with RMS. The extracted data is outputted to text files. After these text files are moved to the RDW server, RETL then compares the data in the text file with the historical dimension data in RDW, and thereafter inserts/updates the dimension changes back into RDW. This comparison eliminates the need to capture dimension changes as they occur in the source system during the day.

RMS supplied dimensions include the following: Code Detail, Company, Competitor, Currency Code, *Employee, Item-Location Trait Cross, Item-Supplier-Location Cross, Organization, Product (including attributes, such as differentiators), Product Season, Reason, Regionality, *Register, Sub-Transaction Type, Supplier, Tender Type, and *Total Type.

For the dimensions labeled with a “*” above, the data comes to RDW from RMS, however the data originates in Oracle Retail Sales Audit (ReSA). ReSA is a flow through application that accepts ‘raw’ point of sale information and provides ‘clean’ data to downstream applications, such as the RDW.

RMS can be the source of two of the three types of time that RDW supports: fiscal 454 time and fiscal 454 time with a Gregorian time calendar. Retailers can supply the other RDW-supported time functionality: 13 period time. For more information on how time is loaded to RDW, see the *Oracle Retail Data Warehouse Installation Guide*.

Fact Data

The RDW fact data process extracts fact data from RMS using RETL scripts that are packaged with RMS. The extracted data is outputted to text files. After these text files are moved to the RDW server, RETL takes the data in the text files and performs the appropriate transformation, inserts and updates to the fact datamart tables.

RMS supplied facts include the following: Competitor Pricing, Cost, *Escheated Vouchers, Exchange Rates, Inventory Adjustments, Inventory Position, Inventory Receipts, Inventory Transfers, *Loss Prevention Transactions, *Loss Prevention Totals (Tender Transactions, Cashier Over or Short, User-Defined Totals), Markdowns, Net Cost, Pricing, *Outstanding Vouchers, Profit on Base Cost, Return to Vendor, *Sales and Return Transactions (including Pack Sales), Sales Forecasts, *Sales Productivity, Stock Ledger, Supplier Availability, Supplier Compliance, Supplier Contract, Unavailable Inventory, Wholesale/Franchise Sales, Wholesale/Franchise Markdowns, and *Voucher Movement.

For the facts labeled with a “*” above, the data comes to RDW from RMS, however the data originates in Oracle Retail Sales Audit (ReSA). ReSA is a flow through application that accepts ‘raw’ point of sale information and provides ‘clean’ data to downstream applications, such as the RDW.

A Note About Local Currency and Facts

Many RDW clients conduct business in a multi-currency environment. While querying sales facts, for instance, a client may want to see the values in the common local currency of a group of stores in one country, or see values aggregated across all their stores from a number of countries. In order to provide clients both accuracy and flexibility in storing currency values, both local and primary currency values are stored in most of the RDW fact tables. A client using multiple currencies would have any fact that is stored by loc_key held in that location's local currency for that day, side-by-side with a column for that fact converted to primary currency. A client using only one currency would have only the primary currency column populated, leaving the local column null. This currency storage strategy is accomplished by either RETL fact extraction code or legacy fact interfaces, which generate text files that include both local and primary currency values for loading into the datamart tables.

Oracle Retail Invoice Matching (ReIM)

ReIM is a solution that provides all of the data necessary to support the invoice verification function, minimizing interface development and maintenance costs. ReIM can serve as the source of invoice cost fact data. ReIM extracts data using code packaged with ReIM. ReIM then writes one text file for RDW called sincilddm.txt. After this text file is moved to the RDW server, it is made available for RETL. RETL uses the data from the text file to perform the applicable inserts and updates to the invoice cost fact datamart table.

Oracle Retail Price Management (RPM)

RPM is a solution that suggests and assists with pricing decisions. RPM can serve as the source of promotion dimensional data. RPM extracts data using code packaged with RPM. RPM then writes three text files for RDW. After these text files are moved to the RDW server, they are made available for RETL to use the data. RETL uses the data from the text file to perform applicable inserts and updates to the promotion dimension datamart table.

Client-Supplied Data

RDW provides programs and tables for the areas of functionality described in this section. However, there currently is no Oracle Retail source system available to provide data for these functional areas. Clients need to supply the data via text files. These text files will be used as the inputs to process and load data into RDW datamart tables. See the location of client-supplied data in the RDW program schedule. For more business content information regarding the following functional areas of client-supplied data, see the Application Programming Interface (API) Flat File Specifications appendix of the *Oracle Retail Data Warehouse Operations Guide*.

- Catalog and activity requests facts
- Customer dimension
- Customer account dimension
- Customer demographic dimension
- Customer geographic dimension
- Customer and product cluster dimension
- Customer order demand facts

- Customer order line position facts
- Customer order code detail dimension
- Customer order promotion facts
- Customer order returns, replacement, exchange facts
- Customer order value added services facts
- Depiction code dimension
- Plan season dimension
- Market data facts and dimensions
- Media facts and dimensions
- Media location matrix dimensions
- Media transformations dimension
- Selling item dimension
- Space allocation facts
- Store traffic facts
- Two of six supplier compliance facts text files
- Quality control
- Missed schedule deliveries

The tables representing the following areas of functionality are loaded once at installation: voucher age dimension and time like for like transformations. See the *Oracle Retail Data Warehouse Installation Guide* for more information.

RDW Integration with Oracle Retail Workspace

RDW integrates with Oracle Retail Workspace (ORW) in three ways:

- Single Sign-On (SSO) launch of the RDW application from ORW
See the *Oracle Retail Data Warehouse Installation Guide* for additional information about setting up Oracle Single Sign-On.
- Listing and Launching RDW Reports and Dashboards from ORW.
- Displaying RDW reports and dashboards in ORW dashboards

SSO Launch from ORW

The Oracle Retail Workspace installer prompts you to enter the URL for your supported Oracle Retail applications. However, if a client installs a new application after Oracle Retail Workspace is installed, the `retail-workspace-page-config.xml` file needs to be edited to reflect the new application.

The file as supplied comes with all appropriate products configured, but the configurations of non-installed products have been "turned off". Therefore, when "turning on" a product, locate the appropriate entry, set "rendered" to "true", and enter the correct URL and parameters for the new application.

The entry consists of the main URL string plus one parameter named "config". The value of the config parameter is inserted by the installer. Somewhere in the installer

property files there is a value for the properties "deploy.retail.product.rms.url" and "deploy.retail.product.rms.config".

For example, suppose RDW was installed on mycomputer.mycompany.com, port 7777, using a standard install and rms configured with the application name of "rdw121sedevhpsso". If you were to access RDW directly from your browser, you would type in:

`http://mycomputer.mycompany.com:7777/forms/frmservlet?config=rdw121sedevhpsso`

The entry in the retail-workspace-page-config.xml after installation would resemble the following:

```
<secure-work-item id="bieeApp"
  display-string="#{confMsgs.bieeAppTitle}"
  rendered="true"
  launchable="true"
  show-in-content-area="false">
  <url>http://mspdev69:7777/analytics</url>
</secure-work-item>
```

Listing and Launching RDW Reports and Dashboards from ORW

ORW can be configured to show the list of RDW (Oracle BI) reports and dashboards to which a user has access in the Reports worklist of its navigation panel.

The user is then able to click on a report to launch it. For complete details on this integration, refer to the Interfacing with Reports Server section of the Integration Methods and Communication Flow chapter of the *Oracle Retail Workspace Implementation Guide*.

Displaying RDW Reports and Dashboards in ORW Dashboards

ORW dashboards may be configured to display RDW reports and dashboards.

For complete details on this integration refer to the Interfacing with Reports Servers section in the Integration Methods and Communication Flow chapter of the *Oracle Retail Workspace Implementation Guide*.

Note: Due to the space constraints in portlets on a dashboard, reports displayed in these portlets may need to be resized. See the *Oracle Retail Report Resizing Guide* which is available as Metalink Note 559554.1.

Batch Scheduling

Once you have an understanding of how RDW integrates with other applications (see ["RDW Integration with Other Applications"](#)), the information in this chapter will help you streamline your batch scheduling.

- ["Identifying RETL Programs by Application"](#)
- ["Identifying Source Files for RETL Programs and their Corresponding Target Tables"](#)
- ["Removing RETL Programs from RDW Batch Schedule"](#)
- ["Setting up the Batch Schedule"](#)
- ["Batch Schedule Dependencies"](#)
- ["Batch Error Notification Setup"](#)

Identifying RETL Programs by Application

One of the tasks is to identify the programs that need to be scheduled based on the integration with one or more of the following applications: RMS, RPM, or ReIM.

A complete list of all the RDW programs can be found in the Program Reference Lists chapter of the *Oracle Retail Data Warehouse Operations Guide*. The External Data Source column in the program reference list mentions all the data sources.

Identifying Source Files for RETL Programs and their Corresponding Target Tables

A complete list of all the RDW programs can be found in the Program Reference Lists chapter of the *Oracle Retail Data Warehouse Operations Guide*. The Source Table or File column in the program reference list mentions all the sources table names and files.

Removing RETL Programs from RDW Batch Schedule

There can be several factors for removing RETL programs from RDW batch schedule:

- Implementation of RDW is not required to be integrated with one of the source application. This is only true if the client is not integrating RDW with one or more of these source applications: RMS, RPM, or ReIM.
- Aggregate tables provided by RDW are not required for reporting and need to be removed from schedule for improved batch performance.

- If a program is customized and the RETL program delivered by RDW is no longer required.

Use the following procedure to remove a program from RDW batch schedule:

1. Based on the implementation requirements, identify the programs that need to be removed from the RDW batch schedule by referring to *Oracle Retail Data Warehouse Operations Guide*.
2. Put these programs on freeze in your scheduling tool.
3. Identify the corresponding target tables where these programs load the data and drop them from the database schema to free up disk space.
4. Refer to the source system documentation for removing the corresponding data extract program from the source system batch schedule. This will avoid creation of files that are not required by RDW programs.

Setting up the Batch Schedule

For information on setting up the batch schedule, refer to the Program Flow Diagrams chapter of the *Oracle Retail Data Warehouse Operations Guide*.

Batch Schedule Dependencies

All RDW batch dependencies are identified in the Program Flow Diagrams chapter of the *Oracle Retail Data Warehouse Operations Guide*. Setup proper dependencies between RDW and the source applications: RMS, ReIM, RPM that feed RDW.

Validate the batch schedule setup correctness by validating data in dimension tables and fact tables.

If the dependencies between programs are not set up correctly, some programs may fail or will not load any data to the target table.

Batch Error Notification Setup

Each program in RDW returns a code of 'zero' on successful completion which can be used by customer's batch scheduler for checking the program status.

If a program fails, it returns a 'non-zero' value which can be used by customer's batch scheduler.

Every RETL program writes an error to the daily log file when the program fails and a client can find the error by searching the log. In addition to the daily log file, each program also writes its own detail flow and error messages to a separate error file per program. For more details refer to "Message Logging" section of the *Oracle Retail Data Warehouse Operations Guide*.

RDW Performance

RDW is a high performance data warehouse, capable of moving and storing massive amounts of data, as well as providing efficient access to that data via the RDW delivered and custom built reports. With any data warehouse, including RDW, smart decisions on how to implement and run your data warehouse will ensure you are getting the most out of it. This section contains information that will help you get the best performance out of RDW. This section identifies common contributors that can weaken performance, as well as best practices that will ensure RDW is running in the most optimal manner.

It is also important to note that all implementations are unique and the factors that are beneficial for one implementation may not have the same affect for all the implementations. It is a good practice to try out several settings/approaches for the factors/recommendations listed below and use the ones that work best for your environment. The factors listed in this section are the key factors that impact performance but no absolute values or settings can be provided for implementation purposes due to uniqueness of each environment.

Components: Oracle Retail Data Warehouse includes RETL for extract, transform and load and Oracle Business Intelligence (BI) for analytic reporting purposes. The recommendations in this section will focus on both backend (ETL) and front-end (Oracle BI) components of RDW.

Key Factors in Performance

Purging and Archiving Strategy

As RDW grows with time; the data volumes will grow and may result in slower performance. The performance impact can be on RDW batch that loads data to data warehouse tables, RDW reports, and storage.

Adoption of purging and archiving strategy help in reducing data volumes, resulting in better performance. Please consider the following recommendations while implementing these strategies in a data warehouse:

- Design your archiving and purging strategy as early as possible in the RDW implementation. This helps in designing the most optimal table partitioning for large tables.
- Make sure that data is deleted in the most optimal manner. SQL DELETE statements might not be the efficient way of removing un-required data from RDW tables. Consult with your database administrator to discuss purging and archiving techniques.

- Purging and archiving of tables must be carefully designed as it requires a good understanding of analytic reports required by business users or regulatory requirements that require companies to retain certain data for a required duration. For example, in certain cases, aggregated data might be kept longer as compared to the base level fact data because the users are interested in summary level reports as compared to detailed (base level) reports for data older than two years.
- Automation of archiving and purging processes ensures that a consistent approach is being followed in maintaining tables with large data volumes and provides consistent report performance to the users.
- While designing purging programs, make sure that dimensional data is not deleted for which fact data is available or will be available.
- Another important consideration during purging is to make sure that RDW seed data (where applicable) is not deleted accidentally.

Aggregations

Data warehouse reports are widely used for summary level reporting and the users are allowed to drill down to the base fact level data. Summarizing huge volumes of data during report execution is not the preferred way for data warehouse reports and the technique used to improve report performance is to build aggregate tables. Aggregation tables are built only for improved performance.

RDW provides some possible aggregations of base data tables for improved report performance and these serve as an example for building out additional aggregations. Additional aggregations can be built based on the reporting levels like organization hierarchy, product hierarchy, and so on.

Please consider the following recommendations while using aggregations or building additional aggregations:

- While building aggregations improves report performance, the improved report performance should be weighed against reduced ETL batch performance and increased storage requirement.
- The ratio between data in the base fact table versus data in the potential aggregate table should be considered while deciding whether the fact table should be aggregated or not. A ratio of 1:5-10 is a good starting point, a ratio of 1:10-20 is good to aggregate and a ratio of 1 :> 20 must be aggregated.
- Always keep the total number of aggregate tables to as few as possible. Review all the aggregations provided by RDW and remove the ones not important to your business by removing them from RDW batch and deactivate the corresponding aggregate table built in Oracle BI.

An example of inactivating the Oracle BI aggregate table is explained in the ["Decompression View Performance"](#) section of this document

- After it has been decided to build a new aggregate, always refer to the existing data model and make sure to aggregate from the next lower level of aggregation available.

When considering removing the aggregation program from RDW batch refer to the Program Reference Lists chapter of the *Oracle Retail Data Warehouse Operations Guide* for getting the information on program name and the corresponding target table name. In addition to removal of program from RDW batch the corresponding database table can also be dropped from the database schema to free up space in the storage disk. See [Chapter 7, "Batch Scheduling"](#) for additional details on batch schedule and removal of programs.

Report Design

Report design can affect the performance of a report. While creating custom reports, refer to the following guidelines:

- Report developers should be trained in Oracle BI to learn how to design reports in the most optimal manner.
- Design reports at the highest level possible and allow drill down to more detailed levels when required.
- Design reports in a manner that multiple users can utilize a single report output rather than multiple users running the same report. A best practice is to run one report and distribute that report to multiple users. For more information on how to distribute reports, refer to the *Oracle Business Intelligence Answers, Delivers, and Interactive Dashboards User Guide*.
- Do not design reports to request data at a level lower than the minimum level that a metric can be reported. In addition, drilling should not be performed at these levels. This ensures that reports do not produce misleading or invalid results. For example, reports should not be designed to request Planning data at the item level because Planning data is only available at the subclass level and above.
- Evaluate and purge reports periodically to eliminate any outdated or duplicate reports.
- Design reports to use the least amount of fact areas necessary. This reduces the number of fact table joins and in turn reduces the risk of poor report performance. For example, a best practice is not to design a single report with all Sales, Inventory, Pricing and Cost metrics, as this report will perform poorly due to joins on big fact tables. In this type of scenario, try creating separate reports with one or two fact areas on the report at a time and combining the results after these reports have run successfully.
- Design reports with the least number of metrics necessary.
- Schedule reports according to priority. This ensures that critical reports are available when needed. For more information on how to schedule reports refer to the *Oracle Business Intelligence Scheduler Guide*.

ETL Batch Process

- Setup the proper dependencies between the applications to ensure resources are being fully utilized, which helps the nightly batch finish earlier.
- If RDW jobs are scheduled on an event basis (like file watchers), then triggering of RDW jobs do not require all the RMS jobs to finish. As the extract files become available, these can be loaded into RDW tables saving processing time.
- Allocate resources to the most important batch jobs (ones that populate the tables) that support your critical reports (the reports you need first thing in the morning). You can assign job priority in most batch scheduling tools.
- Ensure your source applications batch is optimized. RDW runs towards the end of the nightly batch. RDW jobs are often the last jobs to start due to the dependencies on the source system jobs, so RDW is often the last to finish. Optimizing the source applications batch helps RDW jobs start earlier.

Decompression View Performance

The purpose of decompression view is to give the application the illusion that there is a record for each possible combination (that is, an item-location-day record for each permutation) when in fact there is not. Thus, the fact of whether a table is compressed or not should not be visible to the application that queries data from that table.

When a report is run against the "decompression database view", the response time is very long. The essence of the issue is that the "decompression view" joins the fact table against itself in a correlated sub-query to find the "most recent change record before the date specified in the query" and requires a full table scan. "Join against itself" is a very large and resource consuming task for the back-end database if the customers have a large data set. RDW recommends using the following option for granular data analysis which uses the decompressed views. And this is only for the customers who have the performance issue with the decompressed views.

In this option the decompressed view is built in Oracle BI with bind variables. These bind variables get the values selected by the user from the prompts. This way the decompressed view extracts only the relevant data and then applies the "decompression" logic instead of scanning through the entire fact table.

The only limitation of the new decompressed view is, restricting for one Location, One Department and one Quarter. This means any report which uses this view must have mandatory prompts on one Department, One Location, and One Quarter.

The reason for this restriction is because these decompression views access granular data (which could number into several hundred million) and without any limitations, there will always be performance problems. In order to do granular analysis it is always a good practice to restrict the number of rows.

Creating and Using the Oracle BI Decompressed View

The following example is for the Oracle BI decompressed view equivalent to Pricing_item_ld_dm_v.

1. In the Physical layer, create a new physical table.
2. Give the new table an appropriate name and change the table type to "select". Paste the following SQL as shown in the screenshot and example below.

Figure 8–1 Screen: Physical Table; General Tab

Physical Table - FACT_PRICING

General Columns Keys Foreign Keys

Name: FACT_PRICING

Table Type: Select

☐ Use database specific SQL

Default Initialization String

```
WITH v1 AS
(SELECT /*+ parallel(TBL, 5) index(TBL X_PRICING_ILD_ILD) */TBL.ITEM_KEY ITEM_KEY,
TBL.LOC_KEY,
TBL.DAY_IDNT,
TBL.CHNG_CDE,
TBL.SELLING_UOM_CDE,
TBL.MULTI_SELLING_UOM_CDE,
TBL.F_MULTI_UNIT_QTY,
TBL.F_UNIT_RTL_AMT,
TBL.F_UNIT_RTL_AMT_LCL,
TBL.F_MULTI_UNIT_RTL_AMT,
TBL.F_MULTI_UNIT_RTL_AMT_LCL,
TBL.DM_RECD_STATUS_CDE,
TBL.DM_RECD_LOAD_DT
FROM pricing_item_ld_dm TBL,
prod_item_dm pd
where TBL.ITEM_KEY = pd.ITEM_KEY
and
(
TBL.day_idnt >= (select min(day_idnt) from time_day_dm where qtr_idnt =
substr('VALUEOF(NQ_SESSION.QTR)', 1, 4) || substr('VALUEOF(NQ_SESSION.QTR)',
(INSTR('VALUEOF(NQ_SESSION.QTR)', ' ', -1)+1)))
and TBL.day_idnt <= (select max(day_idnt) from time_day_dm where qtr_
```

☒ Cacheable

☒ Cache never expires

☐ Cache persistence time

Hint:

Description:

OK Cancel Help

```
WITH v1 AS
  (SELECT /*+ parallel(TBL, 5) index(TBL X_PRICING_ILD_ILD) */TBL.ITEM_KEY
ITEM_KEY,
      TBL.LOC_KEY,
      TBL.DAY_IDNT,
      TBL.CHNG_CDE,
      TBL.SELLING_UOM_CDE,
      TBL.MULTI_SELLING_UOM_CDE,
      TBL.F_MULTI_UNIT_QTY,
      TBL.F_UNIT_RTL_AMT,
      TBL.F_UNIT_RTL_AMT_LCL,
      TBL.F_MULTI_UNIT_RTL_AMT,
      TBL.F_MULTI_UNIT_RTL_AMT_LCL,
      TBL.DM_RECD_STATUS_CDE,
      TBL.DM_RECD_LOAD_DT
FROM pricing_item_ld_dm TBL,
      prod_item_dm pd
where TBL.ITEM_KEY = pd.ITEM_KEY
and
  (
    TBL.day_idnt >= (select min(day_idnt) from time_day_dm where qtr_idnt =
substr('VALUEOF(NQ_SESSION.QTR)', 1, 4) || substr('VALUEOF(NQ_SESSION.QTR)',
(INSTR('VALUEOF(NQ_SESSION.QTR)', ' ', -1)+1)))
    and TBL.day_idnt <= (select max(day_idnt) from time_day_dm where qtr_
```

```

idnt = substr('VALUEOF(NQ_SESSION.QTR)', 1, 4) || substr('VALUEOF(NQ_
SESSION.QTR)', (INSTR('VALUEOF(NQ_SESSION.QTR)', ' ', -1)+1)))
or
TBL.day_idnt >= (select min(day_idnt) from time_day_dm where qtr_idnt =
substr('VALUEOF(NQ_SESSION.QTR)', 1, 4)-1 || substr('VALUEOF(NQ_SESSION.QTR)',
(INSTR('VALUEOF(NQ_SESSION.QTR)', ' ', -1)+1)))
and TBL.day_idnt <= (select max(day_idnt) from time_day_dm where qtr_
idnt = substr('VALUEOF(NQ_SESSION.QTR)', 1, 4)-1 || substr('VALUEOF(NQ_
SESSION.QTR)', (INSTR('VALUEOF(NQ_SESSION.QTR)', ' ', -1)+1)))
)
and ( concat(concat(pd.DEPT_DESC, ' '), pd.DEPT_IDNT) = 'VALUEOF(NQ_
SESSION.Dept)')
and TBL.LOC_KEY in (select LOC_KEY from org_loc_dm where
concat(concat(LOC_DESC, ' '), LOC_IDNT) = 'VALUEOF(NQ_SESSION.Loc)')
and TBL.DM_REC'D_STATUS_CDE is NULL
)
SELECT  ITEM_KEY,
        LOC_KEY,
        DAY_IDNT,
        CHNG_CDE,
        SELLING_UOM_CDE,
        MULTI_SELLING_UOM_CDE,
        F_MULTI_UNIT_QTY,
        F_UNIT_RTL_AMT,
        F_UNIT_RTL_AMT_LCL,
        F_MULTI_UNIT_RTL_AMT,
        F_MULTI_UNIT_RTL_AMT_LCL,
        DM_REC'D_STATUS_CDE,
        DM_REC'D_LOAD_DT
FROM
  (SELECT /*+ parallel(TBL2, 5) */TBL2.ITEM_KEY ITEM_KEY
    ,TBL2.LOC_KEY LOC_KEY
    ,t.DAY_IDNT DAY_IDNT
    ,TBL2.CHNG_CDE CHNG_CDE
    ,TBL2.SELLING_UOM_CDE SELLING_UOM_CDE
    ,TBL2.MULTI_SELLING_UOM_CDE MULTI_SELLING_UOM_CDE
    ,TBL2.F_MULTI_UNIT_QTY F_MULTI_UNIT_QTY
    ,TBL2.F_UNIT_RTL_AMT F_UNIT_RTL_AMT
    ,TBL2.F_UNIT_RTL_AMT_LCL F_UNIT_RTL_AMT_LCL
    ,TBL2.F_MULTI_UNIT_RTL_AMT F_MULTI_UNIT_RTL_AMT
    ,TBL2.F_MULTI_UNIT_RTL_AMT_LCL F_MULTI_UNIT_RTL_AMT_LCL
    ,TBL2.DM_REC'D_LOAD_DT DM_REC'D_LOAD_DT
    ,tbl2.dm_rec'd_status_cde
    ,tbl2.day_idnt inv_day_idnt,
    MAX(tbl2.day_idnt) over(PARTITION BY item_key, loc_key, t.day_idnt) inv_
max_day_idnt
  FROM v1 tbl2,
       time_day_dm t
  WHERE tbl2.day_idnt <= t.day_idnt)
WHERE inv_day_idnt = inv_max_day_idnt

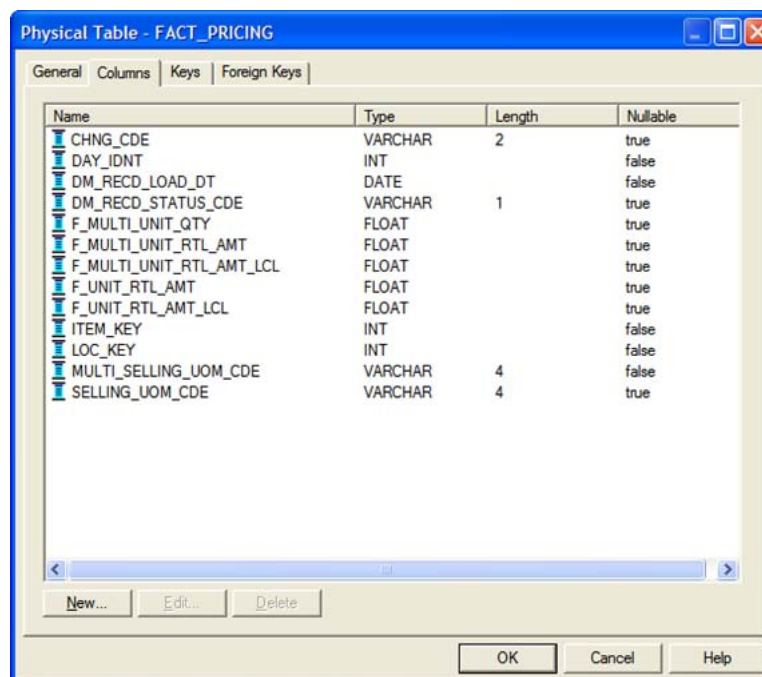
```

Note: The above SQL uses an index (X_PRICING_ILD_ILD) on item location and day. This needs to be changed if the index is created with a different name. RDW recommends having this index.

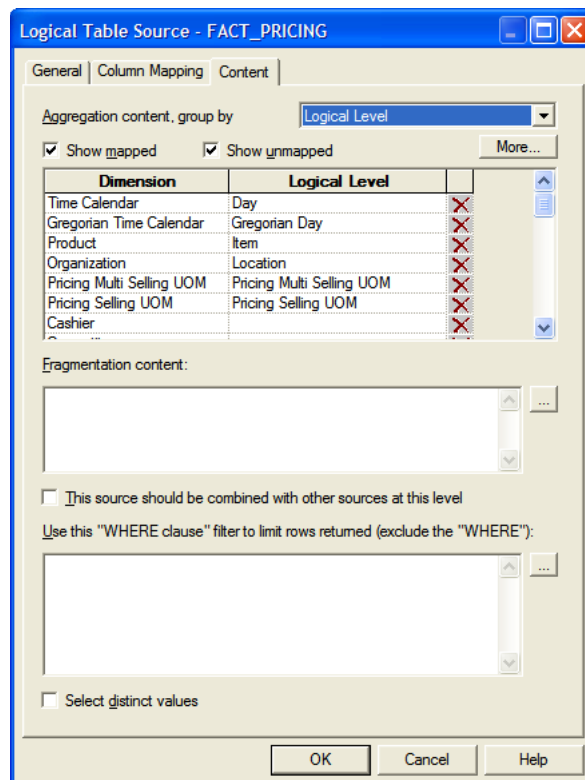
The only difference between this new SQL and the existing DB view SQL is, this view has the bind variables which get the values from the prompts selected by the users, which are mandatory, for one Department, one Location and one Quarter (plus previous quarter for comparison metrics). These session variables (Dept, Loc and QTR) are already built in the RDW.rpd. For more information on Oracle BI variables refer to the Using Variables in Oracle BI Repository chapter of the *Oracle Business Intelligence Server Administration Guide*.

3. Go to the Columns tab and create the columns by clicking the New button. Make sure the column names are the same as the physical table. The following screenshot is an example.

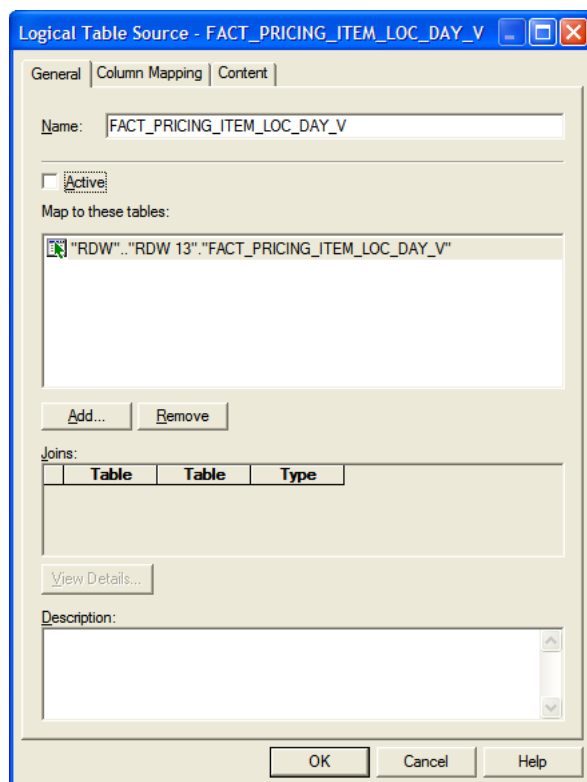
Figure 8–2 Screen: Physical Table; Columns Tab



4. Click OK.
5. Join this new view to all the dimension tables which are joined to the FACT_PRICING_ITEM_LOC_DAY table.
6. Add this new view to the "FACT - Pricing" logical table in the business layer.
7. In the sources section of the "FACT - Pricing", double click on the new view and define the "Logical Levels" in the content tab appropriately as shown in the following screenshot.

Figure 8–3 Screen: Logical Table Source; Content Tab

8. Click OK.
9. Map all the metrics of "FACT - Pricing" to the new view.
10. In order to make use of the new Oracle BI view, the original DB view should be deactivated. To do that, double-click on FACT_PRICING_ITEM_LOC_DAY_V and uncheck the "Active" option as shown in the following screenshot.

Figure 8–4 Screen: Logical Table Source; General Tab

11. Click OK.
12. Do a consistency check and save the repository.
13. In the Presentation Services, all the new and existing reports which use decompressed views should have mandatory prompts with request variables on Department, Location, and Quarter. For more details on creating prompts using request variables refer to the Basics of Working with Requests in Oracle BI Answers chapter of the *Oracle Business Intelligence Answers, Delivers, and Interactive Dashboards User Guide*. A sample prompt is also available in the catalog for a reference. The name of the prompt is "Sample" and it can be found in Answers ' Shared Folder ' Dashboard Prompts.

Note: As an example, two Oracle BI views with complete mappings are packaged with the RDW.rpd. For both: Cost and Pricing. These views can be used for the Price Analysis (A) report of the Merchandising dashboard. This is recommended only if there is a performance issue with the database decompressed views.

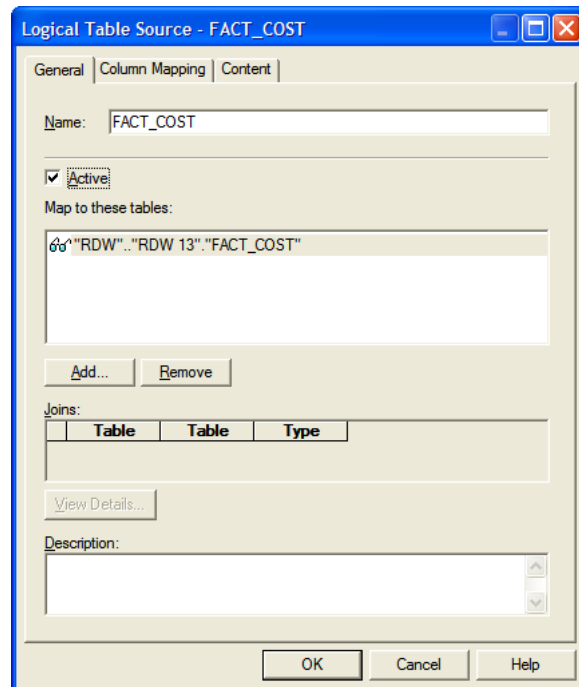
Using the Oracle BI Decompression Views for the Price Analysis (A) Report

The following changes must be made in order to use the FACT_COST and FACT_PRICING views for the Price Analysis (A) report.

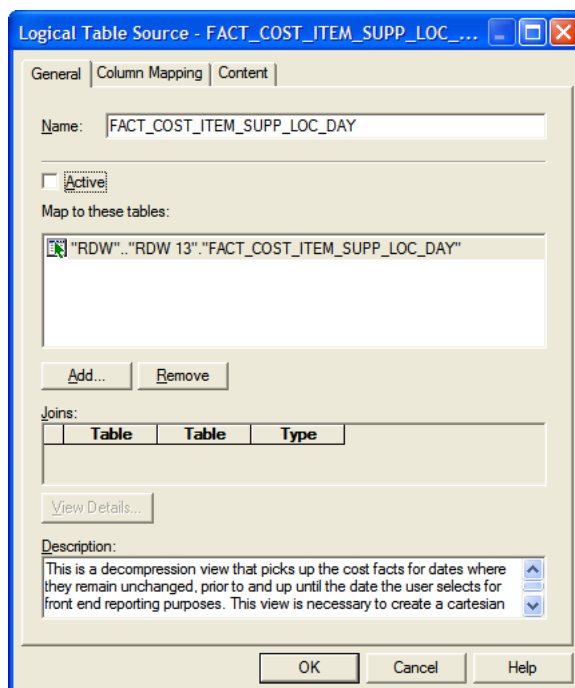
1. Open RDW.rpd in offline mode.

2. In the Business layer go to the "Fact - Cost" logical table and double-click the source "FACT_COST". Make it active by checking the option provided as shown in the following sample screenshot.

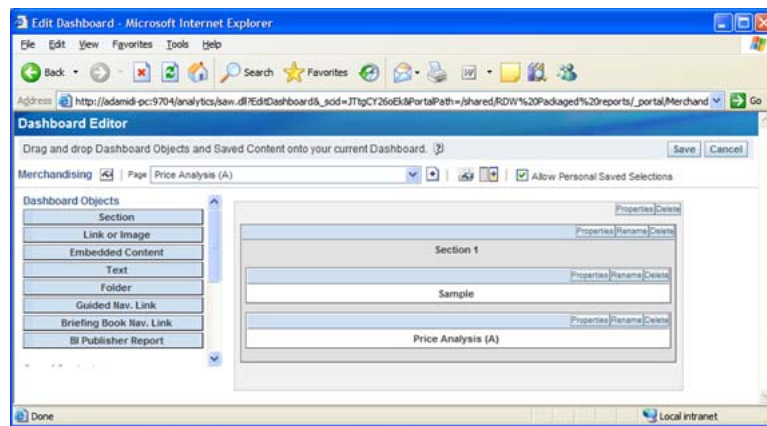
Figure 8–5 Screen: Fact_Cost Logical Table



3. In the same way double-click on "FACT_COST_ITEM_SUPP_LOC_DAY" and make it inactive by un-checking the active option as shown in the following screenshot.

Figure 8–6 Screen: Logical Table Active Option Unchecked

4. Repeat steps 2 and 3 for the "Fact - Pricing" logical table. Make "FACT_PRICING" active and "FACT_PRICING_ITEM_LOC_DAY_V" inactive.
5. Do a consistency check and save the repository.
6. Log on to the Oracle BI Presentation Services.
7. Go to the Merchandising dashboard and click on the Price Analysis (A) page. Click Cancel to stop the report execution.
8. Click on Page Options located at the right corner of the screen and select Edit Dashboard.
9. Replace the existing prompt on the dashboard with the new prompt "Sample" found under "Dashboard Prompts" in the left panel. This "Sample" prompt sends the selected values to the session variables (Dept, Loc and QTR) which are used in the new decompressed Oracle BI views. See below for a sample screenshot.

Figure 8–7 Screen: Dashboard Editor

10. Click Save.
11. Restart the Oracle BI Server and the Oracle Presentation Server services.
12. Log on to Oracle BI Presentation Services and execute the "Price Analysis (A)" report.

Partitioning Strategy

Database level table partitioning is very important for ETL batch and report performance. Refer to [Chapter 5, "Compression and Partitioning"](#) for more information.

Database Configuration

Database configuration plays an important role in overall performance of ETL programs and reports. Refer to [Chapter 3, "Setup and Configuration"](#) for more details.

Adequate Hardware Resources

ETL program and report performance are highly dependent on the hardware resources, refer to [Chapter 3, "Setup and Configuration"](#) for more details.

Additional Factors

Decision support queries sometimes require retrieval of large amounts of data. The Oracle BI server can save the results of a query in cache files and then reuse those results later when a similar query is requested. Using the middle-tier cache thus permits a query to be run one time for multiple runnings of a query and not necessarily every time the query is run.

To summarize, query caching has the following advantages only when the same report is run repeatedly:

- Improvement of query performance
- Less network traffic
- Reduction in database processing and charge back
- Reduction in Oracle BI server processing overhead

Caching Considerations for RDW

- Be sure that the tables and views that need to be cached are Cache Enabled. Verify this in the Oracle BI Administration Tool by ensuring the Cache Enabled box is checked on the General tab in the table or view physical properties.
- Make sure the cache persistence settings defined are appropriate for the environment and report requirements.
- Create a solid cache management strategy. For more info, see the Query Caching in the Oracle BI Server chapter in the *Oracle Business Intelligence Server Administration Guide*.
- Monitor cache usage daily (Oracle BI Administration Tool>Manage>Cache) to be sure enough cache file space has been allocated for the implementation.
- Consider implementing a cache seeding strategy for long-running queries. This would entail selecting normally high volume reports that are also typically poor performers and executing those queries before the business day to preload data to the Oracle BI server cache.

The cache settings are specified in the \$OBI_HOME/server/Config/NQSConfig.INI file on the middle-tier server. The settings should be tuned through testing and analysis in each specific production RDW implementation. Key settings are as follows, but it is highly recommended to run the reports with different values during test runs for the below listed parameters and implement those settings that work best for your environment:

```
MAX_CACHE_ENTRIES
POPULATE_AGGREGATE_ROLLUP_HITS = NO;
USE_ADVANCED_HIT_DETECTION = YES;
ENABLE = YES;
MAX_ROWS_PER_CACHE_ENTRY = 0; // 0 is unlimited size
MAX_CACHE_ENTRY_SIZE
MAX_CACHE_ENTRIES
POPULATE_AGGREGATE_ROLLUP_HITS = NO;
USE_ADVANCED_HIT_DETECTION = YES;
MAX_SUBEXPR_SEARCH_DEPTH
```

Overall, it should be noted that caching has benefits only when the same report is run repeatedly. Enabling caching does not benefit the report that has performance issues and is run for the first time.

Leading Practices

Customizations

Changes and modifications to the RDW delivered code or development of new code is considered customization. RDW does not support custom code developed by clients unless the issue related to customization can be recreated using RDW delivered objects. Below are a few recommendations that will help in maintaining RDW code.

- Naming Convention –It is highly recommended to use a good and consistent naming convention when customizing RDW delivered code or building new code in RDW environment.

This strategy is helpful in identifying custom code and also helps when merging a client's RDW repository with future releases of the RDW repository. There is a possibility of losing customizations to RDW provided RETL scripts or RDW

provided repository if the customized code uses the same object/script names that are used by RDW.

- As a best practice, keep all the documentation up-to-date for capturing any changes or new code that has been developed at a client site. For example if table structure has been customized, a custom data model guide should be created or updated with these changes.

RETL Best Practices

For customizations to existing RETL code or while creating new RETL code, refer to the Best Practices chapter of the *RETL Programmer's Guide*.

Oracle BI EE or Oracle BI SE One Best Practices

- Create aliases for the objects created in the physical layer for usability purposes.
- Do not design the business layer model as a snow-flake model.
- Any level key on ident's should be set to non-drillable.
- In the presentation layer, fact folders (presentation tables) should contain only metrics and dimension folders (presentation tables) should contain only attributes.
- For a development environment, it is recommended to use a multi-user environment. For more information on setting up a multi-user environment refer to the Completing Setup and Managing Oracle BI Repository Files chapter of the *Oracle Business Intelligence Server Administration Guide*.

Batch Schedule Best Practices

Automation

The batch schedule should be automated as per the *Oracle Retail Data Warehouse Operations Guide* and any manual intervention should be avoided.

Recoverability

The batch schedule should be setup in such a way that the batch can be restarted from the point where it failed.

High Availability

Depending on your specific requirements and for facilitating performance improvement, a reporting mirror (exact copy of existing data warehouse) can be created. With this approach, one database can be used for ETL processes and the second database can be used by users for running their reports. There are several ways (database level solutions, operating system level solutions and hardware level solutions) of creating a database mirror. Consult with your IT resources or database administrator for evaluating available options.

If this approach is adopted, the users should run their queries from the reporting mirror area, not from core DW area. Take the following into consideration:

- This approach should be considered for large data warehouse implementations
- Creating data marts can be a good option when implementing mirroring

- User notification mechanism should be built to notify users after the data has been refreshed on the mirror.

Advantages

- High availability of data warehouse. When batch is running the users access the mirror and the only downtime is when data is copied over from the core data warehouse to the mirror.
- There are no conflicts between user queries and ETL batch schedule.

Disadvantages

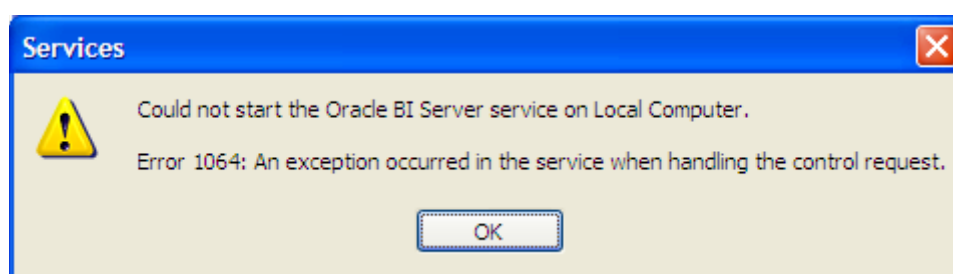
- Storage requirements are increased.
- Additional database maintenance is required.

Batch Efficiency

Keep revisiting the batch timings on a periodic basis to identify the candidates for performance improvements.

Frequently Asked Questions

Q: Why am I getting the following error when trying to start the Oracle BI Server Service?



A: Check the OracleBI\server\Log\ NQServer.log, fix the metadata errors and perform global consistency check, in BI Admin tool.

Q: Why am I getting the login denied error with the following message when tried to run a report Using Oracle BI Presentation Services?

Oracle Error code: 1017, message: ORA-01017: invalid username/password; logon denied

A: Make sure the repository connection pool has the right login credentials in the Oracle BI Administration tool and check the tnsnames.ora file.

Q: I am getting the following error when I performed the "update all row counts" task from the Oracle BI Administration tool.

"Unable to connect database using connection pool"

A: Make sure the repository connection pool has the right login credentials in Oracle BI Administration tool or check TNSNames.ora entry.

Q: Why can't I see query activity at the individual user level in the file NQQuery.log file?

A: Check the Logging level field in the User dialog box, in the User tab. If the logging level is set to 0 (zero), means administrator might have disabled query logging. Contact the Oracle BI administrator to enable query logging.

Q: Batch job failed with an error ORA-12154: TNS:could not resolve the connect identifier specified.

A: Make sure the parameters in the `rdw_config.env` file under `$MMHOMERfx/etc`, are pointing to correct database configuration.

Q: Why is the data not loaded to the fact table, even though I have valid data in the text file?

A: Data may be missing in the corresponding dimension table(s) or the transaction date is not in the active time period of the dimension.

Q: Why is the data not loaded to the Dimension table, even though I have valid data in text file?

A: Parent Data may be missing in the corresponding dimension table(s).

Q: Why is the data not loaded to the fact table, even though I have valid data in the fact text file and all the corresponding keys in the dimensions.

A: Check the `DM_RECD_LOAD_DT` value in the dimension tables. If any of these dimension date values is greater than the fact date value then those will not be loaded to the fact tables as those are the future dimension records.

Q: Description of a subclass is changed from A to B in the input text file but I cannot see both the records in RDW after the loading process?

A: This type of change does not alter the relationship of subclass to any other level of the hierarchy above or below it. The record is simply updated to reflect the description change; and no history will be maintained for the minor changes. For more information please refer to *Oracle Retail Data Warehouse Operations Guide*.

Q: What are the minimum Oracle BI applications that need to be installed for different type of users?

A:

- For the Oracle BI Administrator:
 - Full Oracle BI install (including client and server)
 - Oracle Client.
- For the Oracle BI report User:
 - web browser

For more information, refer to the Installing Oracle BI Infrastructure chapter of the *Oracle Business Intelligence Infrastructure Installation and Configuration Guide*.