Oracle® Retail Analytics

Implementation Guide

Release 13.3

E35141-01

June 2012

ORACLE®

Oracle Retail Analytics Implementation Guide, Release 13.3

Primary Author: Nathan Young

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 **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.

(ii) the **Wavelink** component developed and licensed by Wavelink Corporation (Wavelink) of Kirkland, Washington, to Oracle and imbedded in Oracle Retail Mobile Store Inventory Management.

(iii) 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.

(iv) 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.

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

5 Compression and Partitioning

6 Performance

7 Frequently Asked Questions

Send Us Your Comments

Oracle Retail Analytics Implementation Guide, Release 13.3

Oracle welcomes customers' comments and suggestions on the quality and usefulness of this document.

Your feedback is important, and helps us to best meet your needs as a user of our products. For example:

- Are the implementation steps correct and complete?
- Did you understand the context of the procedures?
- Did you find any errors in the information?
- Does the structure of the information help you with your tasks?
- Do you need different information or graphics? If so, where, and in what format?
- Are the examples correct? Do you need more examples?

If you find any errors or have any other suggestions for improvement, then please tell us your name, the name of the company who has licensed our products, the title and part number of the documentation and the chapter, section, and page number (if available).

> **Note:** Before sending us your comments, you might like to check that you have the latest version of the document and if any concerns are already addressed. To do this, access the Online Documentation available on the Oracle Technology Network Web site. It contains the most current Documentation Library plus all documents revised or released recently.

Send your comments to us using the electronic mail address: retail-doc_us@oracle.com

Please give your name, address, electronic mail address, and telephone number (optional).

If you need assistance with Oracle software, then please contact your support representative or Oracle Support Services.

If you require training or instruction in using Oracle software, then please contact your Oracle local office and inquire about our Oracle University offerings. A list of Oracle offices is available on our Web site at http://www.oracle.com.

Preface

The Oracle Retail Analytics 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 Analytics application integrators and implementation staff.

Related Documents

For more information, see the following documents in the Oracle Retail Analytics Release 13.3 documentation set:

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

Customer Support

To contact Oracle Customer Support, access My Oracle Support at the following URL:

`https://support.oracle.com`

When contacting Customer Support, please provide the following:

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

Review Patch Documentation

When you install the application for the first time, you install either a base release (for example, 13.2) or a later patch release (for example, 13.2.1). If you are installing the base release, additional patch, and bundled hot fix releases, read the documentation for all releases that have occurred since the base release before you begin installation. Documentation for patch and bundled hot fix releases can contain critical information related to the base release, as well as information about code changes since the base release.

Oracle Retail Documentation on the Oracle Technology Network

Documentation is packaged with each Oracle Retail product release. Oracle Retail product documentation is also available on the following Web site:

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

(Data Model documents are not available through Oracle Technology Network. These documents are packaged with released code, or you can obtain them through My Oracle Support.)

Documentation should be available on this Web site within a month after a product release.

Conventions

The following text conventions are used in this document:

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

1
Introduction

Retail Analytics offers a rich business intelligence solution to retail industry users. Retail Analytics is built on top of the latest Oracle technology stack and utilizes Oracle Data Integrator (ODI) for extracting, transforming, and loading (ETL) the data and Oracle Business Intelligence Enterprise Edition (BI EE) for end user reporting and analysis needs.

Retail Analytics architecture is designed to meet the retail industry's business intelligence needs in both program and report performance.

The main characteristics of the Retail Analytics product are:

- Rich Reporting Capabilities: Retail Analytics offers report creation capabilities in three different flavors: Historical (As Was), Current (As Is) and Point-In-Time (PIT) in same environment. Packaged reports are provided as reference examples for users to create their own customized reports according to their needs.
- Comprehensive Solution: Retail Analytics includes an end-to-end solution for reporting and BI needs of the retailer by providing data integration with source applications, transforming and loading the fact and dimension data, rolling up the data for improved query performance, Web-based graphical user interface (GUI) for report creation, shell scripts for setting up the batch schedule, and an automated installer by following business intelligence best practices.
- Performant ETL Code: Retail Analytics data processing tool, ODI, offers high performance for the database batch processes on Oracle database.
- Extensibility: Retail Analytics ETL code can be customized and extended for client specific needs.
- Flexibility: Retail Analytics ODI and Oracle BI EE code promote flexibility during implementation based on client specific needs and help in improving batch and report performance.
- Performant Reports: Retail Analytics metadata is built using Oracle BI EE and are designed to work in complex reporting scenarios.
- Robust Data Model: Retail Analytics data model is designed for supporting a retailers' data needs in a business intelligence environment. Data model elements are designed to work with Oracle BI EE architecture.

Business Intelligence and Retail Analytics

This section briefly explains the fundamentals of business intelligence and data warehousing in general. It is important to understand the overall architecture and data flow for implementing Retail Analytics.

Business intelligence includes the processes, methods, and technologies adopted by organizations to answer complex business questions and for building comprehensive decision support systems. These systems help organizations in maintaining secure, conformed, and highly available data for all levels of users from top executives who make decisions based on corporate level information to managers/analysts who analyze their area and take actions based on the information.

Business intelligence is built using several processes and applications that maintain these processes by adopting latest tools and technologies. One of the main components of business intelligence is a data warehouse. A data warehouse is the repository that stores the data extracted from several source systems and modelled to perform for data loading, reporting, and ad-hoc analysis needs.

Retail Analytics has several integrated data sources, including Oracle Retail Merchandising System (RMS) and Oracle Retail Price Management System (RPM). Data from these sources is extracted, loaded and transformed to the Retail Analytics data model to support report requirements. The first step after installing the Retail Analytics application is to load the seed data into the data warehouse table using pre-packaged Oracle Data Integrator ETL programs.

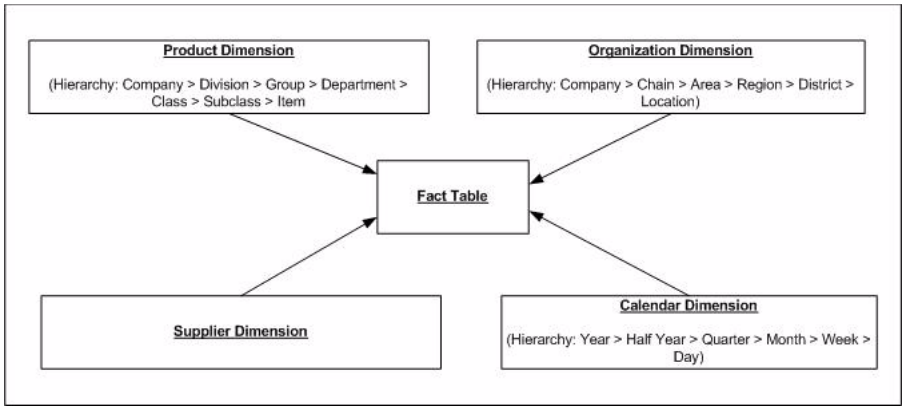
Retail Analytics uses sophisticated techniques to populate the data warehouse. Explained in greater detail throughout this guide, these techniques include taking the data provided by source systems (such as RMS) and then rapidly transforming that data and loading it into the data warehouse. Techniques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

There are several fact and dimension tables in the subject areas available in Retail Analytics. Some examples of subject areas that exist in Retail Analytics include Sales, Inventory Position, and Base Cost. Each subject area has its own data mart to support reporting and analytic needs. At the center of each data mart is fact data (note that fact data here corresponds to both base fact data and aggregated data). Facts are the transactions that occur in your data warehouse's source systems, such as RMS. You may want to look at sales transaction facts, inventory stock count facts at stores or warehouses, or inventory movement facts.

Facts have little meaning by themselves because they are usually just values (for example, six sales at a store, 15 items left at a warehouse, or 300 items transferred). What gives fact data true meaning 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 following diagram illustrates data elements of a generic data mart and their inter-relationships:

Figure 1–1 Data Element Relationships



2

Setup and Configuration

Sizing Information

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

There are two major hardware components that make up the Retail Analytics physical environment:

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

Sizing is customer-specific. The sizing of the Retail Analytics application is sensitive to a wide variety of factors. 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 Oracle database is the most critical performance and sizing component of Retail Analytics. As with any database installation, regularly monitoring database performance and activity levels and regularly tuning the database operation are essential for optimal performance.

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 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 Retail Analytics 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 section in Oracle Business Intelligence chapter of the *Oracle Business Intelligence Enterprise Edition Deployment Guide*.

- Back End Database

  Functions Used Determine Tables to be populated - Retail Analytics is designed to be a functional system so that some functions (such as 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. To identify the database objects necessary for the selected functions, refer to the Data Model.

  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, such as inventory data "as of 9:00AM this morning") can result in very large tables. The Retail Analytics concept of data compression (not to be confused with database table compression) is important in controlling the disk space requirement. For more information, see Chapter 5, "Compression and Partitioning".

  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 - 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, 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 - The initialization parameter settings enable you to optimize the daily data loading and report query processing.

  Data Storage Density - As is the case with many data warehouses, the data stored in the Retail Analytics 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 speed both daily data loading and report query processing in the database. The application server needs fewer resources than the database.

  Memory - More memory enables a larger SGA in the 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 very critical to optimal performance. Selection of disk drives in the disk array should focus on speed. For example, faster RPMs, more spindles, larger cache, fiber optic 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.

Data Seeding of Positional Facts

For base level positional fact data, Retail Analytics uses a compression approach to reduce the data volume. Compression in Retail Analytics 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 Retail Analytics user interface, data seeding is required if clients launch Retail Analytics later than RMS, which is the source system of Retail Analytics. For performance reasons, it is recommended that all date range partitioned positional fact tables must seed data on the first date or first week of each partition. This avoids searching the data across partitions.Data used for seeding can come from RMS or from client legacy systems. The following are some recommendations to seed data:

- If seeding data is for new added partition, you can run the Retail Analytics script retailpartseedfactplp.ksh to seed new partition. This script moves seed data from Retail Analytics CUR tables to new partitions.
- If seeding data is for new tables, you may need to provide snapshots of your positional fact data. See "Data Initial Load from RMS" on page 2-6 for how to provide initial snapshots of positional fact data.

Data Migration from a Legacy Data Warehouse System

Retail Analytics 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 higher or lower than what the Retail Analytics data model supports.

- If the granularity of client history data is lower than what the Retail Analytics data model supports, the client can aggregate data to the same level that the Retail Analytics data model supports, and then populate the Retail Analytics base table.

- If the granularity of client history is higher than what the Retail Analytics data model supports, the client can aggregate data to the Retail Analytics aggregation tables (if they are available). This could cause inconsistencies between the Retail Analytics base tables and Retail Analytics 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.
- Retail Analytics provided APIs can be used for designing and developing the data extraction programs from legacy system. For more information on the APIs, refer to the *Oracle Retail Analytics Operations Guide*.

Reporting Scenarios

By default, Retail Analytics provides the features to use the following types of BI reporting scenarios:

- As-Was
- As-Is
- Point in Time

For more information on the reporting scenarios, refer to the *Oracle Retail Analytics User Guide*.

Based on business needs, you can configure to have one or all of these scenarios, or the combination. These configuration changes will be in ODI (the change is only in the batch scheduler, which is not available by default with Retail Analytics), Oracle BI EE, and the *Oracle Retail Analytics Data Model.*

If the business requirement is to see the history as it happened all the time, which is the As-Was scenario, then it is recommended that you disable all ODI jobs related to As-Is and vice versa. The reason for this is to reduce the load and avoid unnecessary jobs to improve the batch time. For more information on identifying these jobs, refer to chapter 6 ODI Program Dependency in the *Oracle Retail Analytics Operations Guide*.

Once the ODI jobs are disabled, the appropriate tables/objects must be disabled in Oracle BI EE and the data model.

Lets take an example of Inventory Receipts fact and following are the required steps if As-Is is not needed.

ODI

Disable the jobs related to the following tables:

```
W_RTL_INVRC_SC_DY_CUR_A
W_RTL_INVRC_SC_LC_WK_CUR_A
W_RTL_INVRC_SC_WK_CUR_A
```

This information can be found in the *Oracle Retail Analytics Operations Guide*.

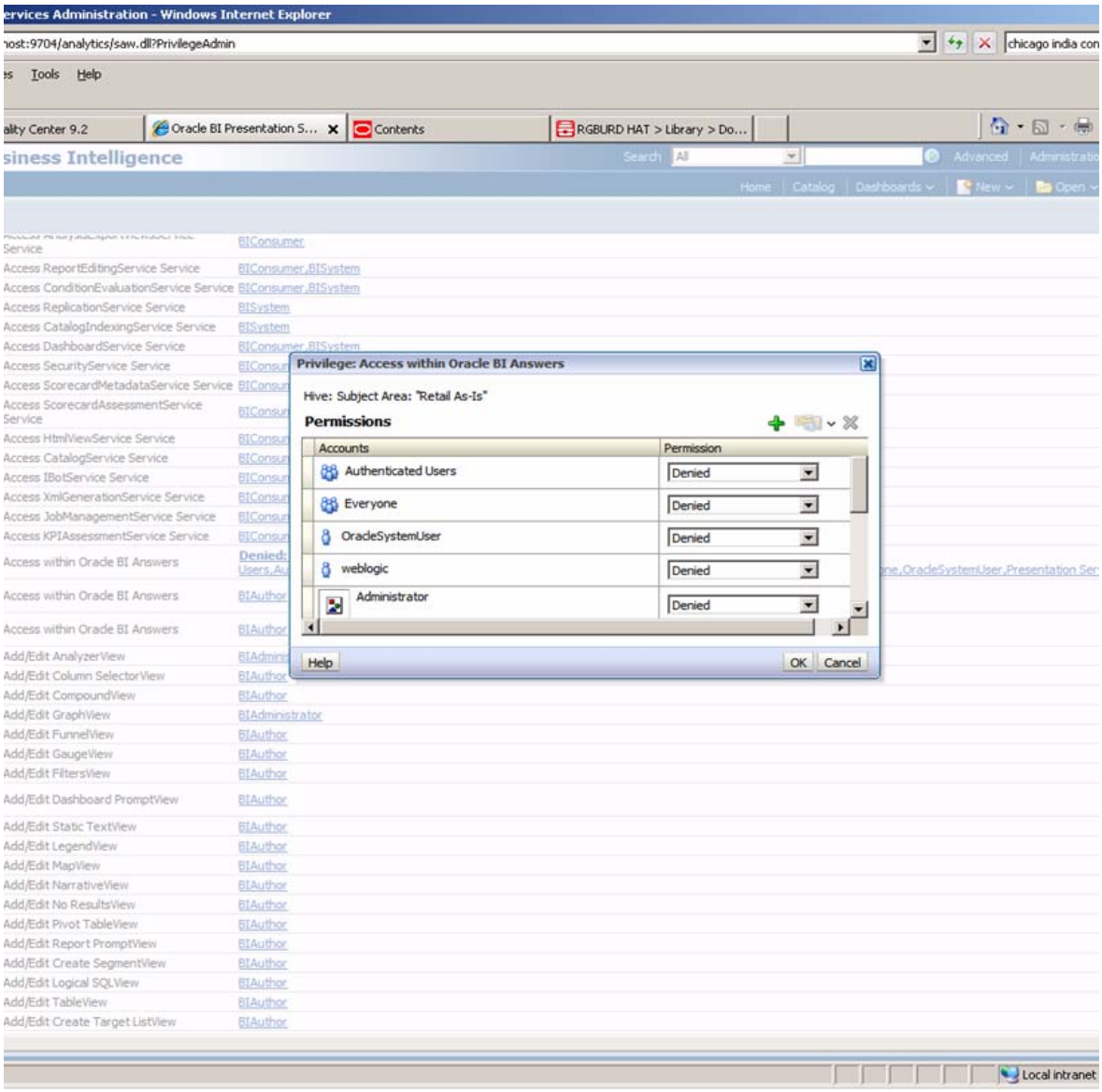
Oracle BI EE

In the "Fact - Retail Inventory Receipts" logical table, disable the following sources:

```
Fact_W_RTL_INVRC_SC_DY_CUR_A
Fact_W_RTL_INVRC_SC_LC_WK_CUR_A
Fact_W_RTL_INVRC_SC_WK_CUR_A
```

With this change, these tables can never be accessed and As-Is reporting cannot be done for inventory receipts. The same changes must be done for all the other fact areas as well. Since Retail Analytics has three subject areas (Retail As-Was, Retail As-Is and Retail Point in Time), all three are available to the user. Since As-Is components should be disabled, go to the Administration -> Manage Privileges on Oracle BI EE web and, for all the users/roles, change the permission setting to "Denied" for the As-Is subject area. This way, the As-Is subject area will not be available for reporting. The following figure displays the Administration screen.

Figure 2–1 Administration Screen



Data Model

All the unused tables can still be in the schema and will be not used by ODI and Oracle BI EE programs. It can be maintained for future changes.

Point in Time reporting can be done with As-Was, As-Is, or with both. There are no separate ODI processes for PIT. PIT can be derived from either As-Is or As-Was data and the processing will happen during report execution. Only difference is, PIT reports cannot use aggregate tables and will always be reported from the base fact tables. There are some limitations to do this reporting in some fact areas owing to performance. For example, with the Positional facts, PIT is possible only for Product hierarchy because there are corporate aggregates for product which have the decompressed data.

The real differentiation of As-Is and As-Was in the data happens above the base fact table or when reporting is done at the parent level. Otherwise if the reporting is done at the lowest grain level, the result set will be the same for both.

Data Initial Load from RMS

In order to report Retail Analytics positional data correctly, all Retail Analytics positional compressed tables need to be seeded with source data (RMS) correctly before they can be loaded using Retail Analytics batch ETL with daily data. This seeding process is to load positional fact data for each item location combination available from RMS to Retail Analytics as initial data. This can be done by using following recommended approach. This approach assumes that user uses RMS as Retail Analytics source system and the required data are available from RMS.

Inventory Position Initial Loading

This initial inventory position data loading includes loading seeding date from RMS to RA W_RTL_INV_IT_LC_DY_F, W_RTL_INV_IT_LC_WK_A, and W_RTL_INV_IT_LC_G tables. Perform the following steps:

1. In RMS, set the RMS vdate to the date that the seeding data will be used for.
2. Execute the Retail Analytics SDE script etlrefreshgensde.ksh to load RMS system data (including vdate) to the Retail Analytics temporary table RA_SRC_CURR_PARAM_G which is under Retail Analytics RMS batch user schema.
3. Populate the RMS IF_TRAN_DATA table with all combinations of item and location that have SOH. This can be done by using the data from the RMS ITEM_LOC_SOH table. Only columns ITEM, LOCATION, and LOC_TYPE on the IF_TRAN_DATE table will be used by the Retail Analytics SDE program and other columns on this table can be given any dummy value for this seeding purpose.
4. Execute the Retail Analytics SDE script invildsde.ksh to populate the Retail Analytics inventory staging table W_RTL_INV_IT_LC_DY_FS.
5. Make sure the Retail Analytics table W_RTL_CURR_MCAL_G has the business date and week for the current Retail Analytics business date. This is used as the date for seeding data. This date should match the RMS vdate set for the SDE program.
6. Execute the Retail Analytics SIL script invildsil.ksh to load the inventory seeding data from the staging table to the Retail Analytics base fact table W_RTL_INV_IT_LC_DY_F and W_RTL_INV_IT_LC_G.
7. Execute the Retail Analytics PLP script invildwplp.ksh to load inventory seeding data to the Retail Analytics table W_RTL_INV_IT_LC_WK_A.

8. Execute other inventory PLP scripts to populate the Retail Analytics inventory aggregation tables. These are chosen by the client for reporting purposes.

Pricing Initial Loading

This initial Pricing data loading includes loading seeding date from RMS to Retail Analytics W_RTL_PRICE_IT_LC_DY_F and W_RTL_PRICE_IT_LC_G tables. Perform the following steps:

1. In RMS, set the RMS vdate to the date that the seeding data will be used for.
2. Execute the Retail Analytics SDE script etlrefreshgensde.ksh to load the RMS system data (including vdate) to the Retail Analytics temporary table RA_SRC_CURR_PARAM_G. This is under the Retail Analytics RMS batch user schema.
3. Modify the Retail Analytics SDE ODI program as follows:
   a. In the ODI SDE interface SDE_RetailPriceLoad, modify the filter on PRICE_HIST table to change the filter condition from PRICE_HIST.ACTION_DATE = TO_DATE('#RA_SRC_BUSINESS_CURRENT_DT','YYYY-MM-DD') to PRICE_HIST.ACTION_DATE <= TO_DATE('#RA_SRC_BUSINESS_CURRENT_DT','YYYY-MM-DD').
   b. Regenerate the SDE_RetailPriceFact and MASTER_ SDE_RetailPriceFact ODI scenarios.
4. Execute the Retail Analytics SDE script prcildsde.ksh to populate the Retail Analytics Price staging table W_RTL_PRICE_IT_LC_DY_FS.
5. In the Price staging table, only keep the records with the same PROD_IT_NUM, ORG_NUM, MULTI_UNIT_QYT, but maximum DAY_DT. This ensures that the staging table only contains the latest price for each combination of item, location, and multi unit quantity.
6. Update the Pricing staging table to replace DAY_DT with the current business date that will be used for seeding data.
7. Make sure the Retail Analytics table W_RTL_CURR_MCAL_G has the business date and week for the current Retail Analytics business date. This is used as the date for seeding data. This date should match RMS vdate set for the SDE program.
8. Execute the Retail Analytics SIL script prcilsil.ksh to load pricing seeding data from the staging table to the Retail Analytics base fact tables W_RTL_PRICE_IT_LC_DY_F and W_RTL_PRICE_IT_LC_G.
9. Execute the other Price PLP scripts to populate the Retail Analytics Price aggregation tables. These are chosen by the client for reporting purposes.
10. When the initial loading is complete, change the filter condition back and regenerate the two scenarios.

Net Cost Initial Loading

This initial Net Cost data loading includes loading seeding date from RMS to the Retail Analytics W_RTL_NCOST_IT_LC_DY_F and W_RTL_NCOST_IT_LC_G tables. Perform the following steps:

1. In RMS, set the RMS vdate to the date that the seeding data will be used for.
2. Execute the Retail Analytics SDE script etlrefreshgensde.ksh to load the RMS system data (including vdate) to the Retail Analytics temporary table RA_SRC_CURR_PARAM_G. This is under the Retail Analytics RMS batch user schema.

3. Modify the Retail Analytics SDE ODI program as follows:
   a. In the ODI SDE interface SDE_RetailNetCostTempLoad, modify the filter on FUTURE_COST table to change the filter condition from FUTURE_COST.ACTIVE_DATE = to_date('#RA_SRC_BUSINESS_CURRENT_DT','YYYY-MM-DD') to FUTURE_COST.ACTIVE_DATE <= to_date('#RA_SRC_BUSINESS_CURRENT_DT','YYYY-MM-DD').
   b. Regenerate the SDE_RETAILNETCOSTFACT and MASTER_SDE_RETAILNETCOSTFACT ODI scenarios.
4. Execute the Retail Analytics SDE script ncstildsde.ksh to populate the Retail Analytics Net Cost staging table W_RTL_NCOST_IT_LC_DY_FS.
5. In the Net Cost staging table, only keep the records with the same PROD_IT_NUM, ORG_NUM, SUPPLIER_NUM, but maximum DAY_DT. This ensures that the staging table only contains the latest net cost for each combination of item, location and supplier.
6. Update the Net Cost staging table to replace DAY_DT with the current business date that will be used for seeding data.
7. Make sure the Retail Analytics table W_RTL_CURR_MCAL_G has the business date and week for the current Retail Analytics business date. This is used as the date for seeding data. This date should match the RMS vdate set for the SDE program.
8. Execute the Retail Analytics SIL script ncstildsil.ksh to load Net Cost seeding data from the staging table to the Retail Analytics base fact table W_RTL_NCOST_IT_LC_DY_F and W_RTL_NCOST_IT_LC_G.
9. Execute other Net Cost PLP scripts to populate the Retail Analytics Net Cost aggregation tables. These are chosen by the client for reporting purpose.
10. When the initial loading is complete, change the filter condition back and regenerate the two scenarios.

Base Cost Initial Loading

This initial Base Cost data loading includes loading seeding date from RMS to Retail Analytics W_RTL_BCOST_IT_LC_DY_F and W_RTL_BCOST_IT_LC_G tables. Perform the following steps:

1. In RMS, set the RMS vdate to the date that the seeding data will be used for.
2. Execute the Retail Analytics SDE script etlrefreshgensde.ksh to load the RMS system data (including vdate) to the Retail Analytics temporary table RA_SRC_CURR_PARAM_G. This is under the Retail Analytics RMS batch user schema.
3. Modify the Retail Analytics SDE ODI program as follows:
   a. In the ODI SDE interface SDE_RetailBaseCostTempLoad, modify the filter on the PRICE_HIST table to change the filter condition from PRICE_HIST.POST_DATE = TO_DATE('#RA_SRC_BUSINESS_CURRENT_DT','YYYY-MM-DD') to PRICE_HIST.POST_DATE <= TO_DATE('#RA_SRC_BUSINESS_CURRENT_DT','YYYY-MM-DD').
   b. Regenerate the SDE_RETAILBASECOSTFACT and MASTER_SDE_RETAILBASECOSTFACT ODI scenarios.
4. Execute the Retail Analytics SDE script cstisldsde.ksh to populate the Retail Analytics Base Cost staging table W_RTL_BCOST_IT_LC_DY_FS.

5. In the Base Cost staging table, only keep the records with the same PROD_IT_NUM, ORG_NUM but maximum DAY_DT. This ensures that the staging table only contains the latest base cost for each combination of item and location.
6. Update the Base Cost staging table to replace DAY_DT with the current business date that will be used for seeding data.
7. Make sure the Retail Analytics table W_RTL_CURR_MCAL_G has the business date and week for the current Retail Analytics business date. This is used as the date for seeding data. This date should match the RMS vdate set for the SDE program.
8. Execute the Retail Analytics SIL script cstildsil.ksh to load Base Cost seeding data from staging table to the Retail Analytics base fact table W_RTL_BCOST_IT_LC_DY_F and W_RTL_BCOST_IT_LC_G.
9. Execute the other Base Cost PLP scripts to populate the Retail Analytics Base Cost aggregation tables. These are chosen by the client for reporting purposes.
10. When the initial loading is complete, change the filter condition back and regenerate the two scenarios.

3

Security

Security Integration with Oracle Business Intelligence

Oracle Retail Analytics integrates tightly with Oracle Business Intelligence Enterprise Edition (BI EE) to allow the right content to be shown to the right user.

All components of Oracle Business Intelligence Enterprise Edition are fully integrated with Oracle Fusion Middleware security architecture. Oracle BI EE authenticates users using an Oracle WebLogic Server authentication provider against user information held in an identity store. User and group information is held within the Oracle WebLogic Server embedded directory server, which is the default identity store.

Another security feature available in Oracle BI EE is Single Sign-On. The Oracle BI EE is integrated and certified with Oracle Single Sign-On (OSSO). Once OSSO is enabled in Oracle BI, user authentication happens on the OSSO server. For more details, refer to the Enabling SSO Authentication chapter of the *Oracle Business Intelligence Enterprise Edition Security Guide*.

Ensure that you are familiar with the security features of Oracle Business Intelligence Enterprise Edition before you begin working with Oracle BI Applications.

Security settings for Oracle Business Intelligence Enterprise Edition are made in the following Oracle Business Intelligence components. See the *Oracle Business Intelligence Enterprise Edition Security Guide* for more details.

- Oracle WebLogic Server Administration Console
- Oracle Fusion Middleware Control
- Oracle BI Administration Tool
- Administration Page in Oracle BI Presentation Catalog

Security Types

Security in Oracle Retail Analytics can be classified into the following types. By default, Retail Analytics does not provide these security features. You can choose to implement it based on the implementation requirements:

- **Data-level security** - controls the visibility of data (content rendered in subject areas, dashboards, Oracle BI answers, and so on) based on the user's association to data in the transactional system.
- **Object-level security** - controls the visibility to business logical objects based on a user's role. You can set up object-level security for metadata repository objects, such as subject areas and presentation folders, and for web objects, such as dashboards and dashboard pages, which are defined in the presentation catalog.

Object-Level Security in Retail Analytics

This section describes the object-level security features in Retail Analytics. It contains the following topics:

Metadata Object-Level Security (Application Roles)

Application roles control access to metadata objects, such as subject areas, tables, and columns. For example, certain Retail Analytics roles may not have access to view certain presentation tables. Metadata object security is configured in the Oracle BI Repository, using the Oracle BI Administration Tool. The Everyone user group is denied access to some of the presentation tables and only related roles have explicit read access. This access can be extended to subject areas and columns.

Note: By default in Oracle BI Retail Analytics, only permissions at the presentation tables and dashboard level have been configured.

Below are the list of Retail Analytics roles and the associated groups. You have to create these groups in your authentication provider. For more information on how to set-up groups, refer to the *Oracle® Fusion Middleware - Security Guide for Oracle Business Intelligence Enterprise Edition*.

Table 3–1

Groups	Roles
RetailAnalysts	RetailAnalyst
RetailBuyers	RetailBuyer
RetailBuyerAnalysts	RetailBuyerAnalyst
RetailInventoryAnalysts	RetailInventoryAnalyst
RetailInventoryManagers	RetailInventoryManager
RetailMerchandiseExecutives	RetailMerchandiseExecutive
RetailMerchandiseFinancialPlanners	RetailMerchandiseFinancialPlanner
RetailPlanningExecutives	RetailPlanningExecutive
RetailPricingAnalysts	RetailPricingAnalyst
RetailPromotionalPlanners	RetailPromotionalPlanner

Table 3–2

Subject Area	Presentation Table(s)	Retail Analyst	Retail Buyer	Retail Buyer Analyst	Retail Inventory Analyst	Retail Inventory Manager	Retail Merchandise Executive	Retail Merchandise Financial Planner	Retail Planning Executive	Retail Pricing Analyst	Retail Promotional Planner
Retail Customer Analytics As-Is/As-Was/PIT	Promotion Baseline	X	-	-	-	-	-	-	-	-	X
	Promotion Budget	X	-	-	-	-	-	-	-	-	X
	Promotion Actuals	X	-	-	-	-	-	-	-	-	X
	Promotion Forecast	X	-	-	-	-	-	-	-	-	X
	Sales	X	X	X	X	X	X	X	X	X	X
	Sales Promotion	X	-	-	X	X	X	X	X	X	X
	Trial & Repeat	X	X	-	X	X	X	-	X	X	X

Table 3–2 (Cont.)

Subject Area	Presentation Table(s)	Retail Analyst	Retail Buyer	Retail Buyer Analyst	Retail Inventory Analyst	Retail Inventory Manager	Retail Merchandise Executive	Retail Merchandise Financial Planner	Retail Planning Executive	Retail Pricing Analyst	Retail Promotional Planner
Retail Merchandising Analytics As-Is/As-Was/PIT	Inventory Receipts	X	-	-	X	X	X	X	X	-	-
	Markdown	X	-	-	X	X	X	X	X	X	X
	Net Cost	X	X	X	-	-	X	X	X	X	-
	Net Profit	X	X	X	-	-	X	X	X	X	-
	Pricing	X	-	-	X	X	X	X	X	X	X
	Sales	X	X	X	X	X	X	X	X	X	X
	Sales Forecast	X	-	-	X	X	X	X	X	-	-
	Sales Pack	X	X	X	X	X	X	X	X	X	-
	Supplier invoice	X	X	X	-	X	X	-	X	-	-
	Unit Cost	X	X	X	-	-	X	X	X	X	-
	Supplier Compliance	X	X	X	-	-	X	-	X	-	-
	Wholesale Franchisee	X	-	-	-	X	X	X	X	-	X
	Planning	X	-	-	X	X	X	X	X	X	X
	Stock Ledger	X	-	-	-	X	X	-	X	-	-
	Inventory Positions	X	X	X	X	X	X	X	X	X	X
Retail Customer Analytics Data Mining	All 5 Affinities Tables	X	X	-	X	X	X	X	X	-	X

Metadata Object-Level Security (Presentation Services)

Oracle BI Presentation Services objects are controlled using Presentation Services groups. Access to these objects, such as dashboards and pages, reports, and Web folders, is controlled using the Presentation Services groups. Presentation Services groups are customized in the Oracle BI Presentation Services interface. For detailed information about Presentation Services groups, see the *Oracle Business Intelligence Presentation Services Administration Guide*.

Table 3–3

Dashboard	Retail Analyst	Retail Buyer	Retail Buyer Analyst	Retail Inventory Analyst	Retail Inventory Manager	Retail Merchandise Executive	Retail Merchandise Financial Planner	Retail Planning Executive	Retail Pricing Analyst	Retail Promotional Planner
Customer Analysis Dashboard	X	X	X	X	X	X	X	X	X	X
Promotions	X	-	-	-	-	-	-	-	-	X
Market Basket Analysis	X	X	-	X	X	X	X	X	-	X
Inventory Alerts	X	X	X	X	X	X	X	X	X	X
Inventory Performance	X	X	X	X	X	X	X	X	X	X
Markdowns	X	-	-	X	X	X	X	X	X	X
Merchandise Pack Performance	X	X	X	X	X	X	X	X	X	X
Merchandise Performance	X	X	X	X	X	X	X	X	X	X
Merchandise Sales and Profit	X	X	X	X	X	X	X	X	X	X
Merchandising Location Analysis	X	X	X	X	X	X	X	X	X	X
Out Of Stock Analysis	X	X	X	X	X	X	X	X	X	X

Table 3–3 (Cont.)

Dashboard	Retail Analyst	Retail Buyer	Retail Buyer Analyst	Retail Inventory Analyst	Retail Inventory Manager	Retail Merchandise Executive	Retail Merchandise Financial Planner	Retail Planning Executive	Retail Pricing Analyst	Retail Promotional Planner
Retail Merchandising Analytics Overview	X	X	X	X	X	X	X	X	X	X
Suuplier Cost Analysis	X	X	X	X	X	X	X	X	X	X
Supplier Performance	X	X	X	X	X	X	X	X	X	X

4

Internationalization

Internationalization is the process of creating software that is able to be translated more easily. Changes to the code are not specific to any particular market. Retail Analytics has been internationalized to support multiple languages.

This section describes configuration settings and features of the software that ensure that the base application can handle multiple languages.

Translation

Translation is the process of interpreting and adapting text from one language into another. Although the code itself is not translated, components of the application that are translated may include the following:

- Graphical user interface (GUI)
- Error messages
- Reports

The following components are not translated:

- Documentation (online help, release notes, installation guide, user guide, operations guide)
- Batch programs and messages
- Log files
- Configuration tools
- Demonstration data
- Training materials

The user interface for Retail Analytics has been translated into:

- Chinese (simplified)
- Chinese (traditional)
- Croatian
- Dutch
- French
- German
- Greek
- Hungarian

- Italian
- Japanese
- Korean
- Polish
- Portuguese (Brazilian)
- Russian
- Spanish
- Swedish
- Turkish

Multi-Language Setup

Retail Analytics data is supported in 18 languages. This section provides details of various scenarios that may come across during implementation. See "Translation" on page 4-1 for a list of supported languages.

Since multi-language data support in Retail Analytics is dependent on the availability of the multi-language data in the source system, it is important to understand various scenarios the user may encounter. Before proceeding review the following facts about multi-language support:

- Retail Analytics programs extracts multi-language data from source systems.
- A list of languages for multi-language data support can be chosen during the installation process. Please refer to the *Oracle Retail Analytics Installation Guide* for more details.
- Depending on the implementation, the source system may or may not have data for particular supported language(s). For example, RMS supports Item Descriptions in multiple languages but the item's description may not be available in the translated languages.
- For source system released languages, please refer to source system Operations Guides.
- You must select a Retail Analytics primary language for data purposes to be supported within the source system.

Scenario 1

All the supported languages are implemented in Retail Analytics and the same set of languages are supported in the source system as well.

Multi-lingual data sets are enabled in both Retail Analytics and the source system.

Data Scenario 1a

Translated data exists for all records in Source System: This is an ideal scenario where the source system supports data for the same set of languages as Retail Analytics and data for the required column exists in all the languages in the source system.

In this scenario the attributes that are supported for multi-languages will get all the multi-language data in Retail Analytics.

Data Scenario 1b

Translated data does not exist for some of the records in the source system.

For the attributes for which data is not available in the source system, Retail Analytics will display the attribute in source system primary language. For example, Retail Analytics requests data in German and English languages. In RMS the Item attribute description is not available in the German language but is available in English language.

Retail Analytics will display Item description in English to a user who is logged into Oracle BI EE (assuming English is the primary language of RMS for that implementation).

Scenario 2

All or a subset of languages are implemented in Retail Analytics and some of these are not supported in the source system:

Data Scenario 2a

Translated data does not exist for some of the languages in the source system. In this case, the data is displayed in Retail Analytics' primary language.

Scenario 3

Source system supports more languages than are supported for Retail Analytics. In this case Retail Analytics filters out the additional languages' data. This data will not be loaded into Retail Analytics tables and cannot be used for reporting.

5

Compression and Partitioning

This chapter describes how Retail Analytics 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 Retail Analytics 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, in the cost to store the rows, the cost to perform backups, and other database maintenance operations.

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

- What compression does
- Mechanics of compression
- Which tables are currently compressed
- 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 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 and a 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.

Retail Analytics compression is different with Oracle DB table compression. Oracle DB table compression compress data by eliminating duplicate values within a data block. Any repetitive occurrence of a value in a block is replaced by a symbol entry in a "symbol table" within the data block. So for example DEPT_NUM=10 is repeated five times within a data block, it will be only stored once and for the other four times a symbol entry will be stored in symbol table. Oracle database table compression can also significantly reduce disk and buffer cache requirements for database tables while improving query performance. Oracle database compressed tables use fewer data blocks on disk, reducing disk space requirement.

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. Retail Analytics compressed tables use FROM_DT_WID and TO_DT_WID columns to indicate the time range in which records are valid.

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. A decompression view compares FROM_DT_WID and TO_DT_WID of records with FROM_VALUE and TO_VALUE on partition mapping table W_RTL_PARTITION_MAP_G to make sure that a right partition is used by the decompression view.

To illustrate how the decompression views actually work, assume the following:

- The user is 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 latest 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 will 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, you may take advantage of compressed table partition seeding (see "Overview of Partitioning Strategies" on page 5-4).

This partition seeding utilizes the latest position status tables (also known as 'current' tables). An example is the W_RTL_INV_IT_LC_G table, which holds the current decompressed position for every item and location on the W_RTL_INV_IT_LC_DY_F

table. This position can be used as a partition seed. This position is also utilized by base Retail Analytics code during major change fact seeding.

Compressed Tables and 'CURRENT' Tables

The table below illustrates the compressed tables within Retail Analytics, along with their corresponding 'CURRENT' tables.

Table 5–1

Compressed Tables	Current Tables
W_RTL_INV_IT_LC_DY_F	W_RTL_INV_IT_LC_G
W_RTL_INV_IT_LC_WK_A	W_RTL_INV_IT_LC_G
W_RTL_PRICE_IT_LC_DY_F	W_RTL_PRICE_IT_LC_G
W_RTL_BCOST_IT_LC_DY_F	W_RTL_BCOST_IT_LC_G
W_RTL_NCOST_IT_LC_DY_F	W_RTL_NCOST_IT_LC_G

Coping with Slowly Changing Dimension Type 2

Fact Close Program (factcloseplp.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 to ensure that a fact record appears for each item-location-day combination in the user interface. However, when an item, 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, scenario PLP_RetailFactCloseFact (called by factcloseplp.ksh) first queries the W_RTL_PROD_RECLASS_TMP and W_RTL_ORG_RECLASS_TMP tables to determine the compressed item-location facts that need to be closed today. The PLP_RetailFactCloseFact scenario then updates TO_DT_WID to the current date WID to stop the record. The decompression view fills in records up to the day that is in the range between FROM_DT_WID and TO_DT_WID.

Fact Open Program (factopenplp.ksh)

Retail Analytics Data Compression tables require seeding when a major change in the product and 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.

To accomplish this seeding scenario, PLP_RetailFactOpenFact (called by factopenplp.ksh) first queries the W_RTL_PROD_RECLASS_TMP and W_RTL_ORG_RECLASS_TMP tables to determine what compressed item-location facts need to be closed today. The PLP_RetailFactOpenFact scenario then inserts seeded (closed) records for tomorrow's FROM_DT_WID, indicating that the closed fact records are no longer valid beginning tomorrow, when the newly seeded records (from PLP_RetailFactOpenFact) become active. In the case of the compressed week table, W_RTL_INV_IT_LC_WK_A, PLP_RetailFactOpenFact inserts seeded records with next week's warehouse ID.

Oracle Table Compression

Oracle table compression not only helps customers save disk space, it also helps to increase cache efficiency since more blocks can fit in the memory. Advanced Compression is available for Oracle 11g Enterprise Edition and Hybrid Columnar Compression is available for Exadata only.

Since compression could cause contention when tables get updated, it is suggested users only compress non-current partitions and leave the current partition uncompressed. This partial compression approach has proven to be a valuable implementation option.

Overview of Partitioning Strategies

This section describes partitioning strategies for Retail Analytics data marts. Although optional, partitioning provides powerful performance benefits, and therefore is highly recommended. Tables in the RA_partitioned_tables.xls spreadsheet (see the *Oracle Retail Analytics Installation Guide*) are highly recommended to be partitioned. If a report runs slowly and a fact table in the query is not partitioned, that fact table may be a good candidate for partitioning. For large tables, such as 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, the Oracle database 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 will 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.

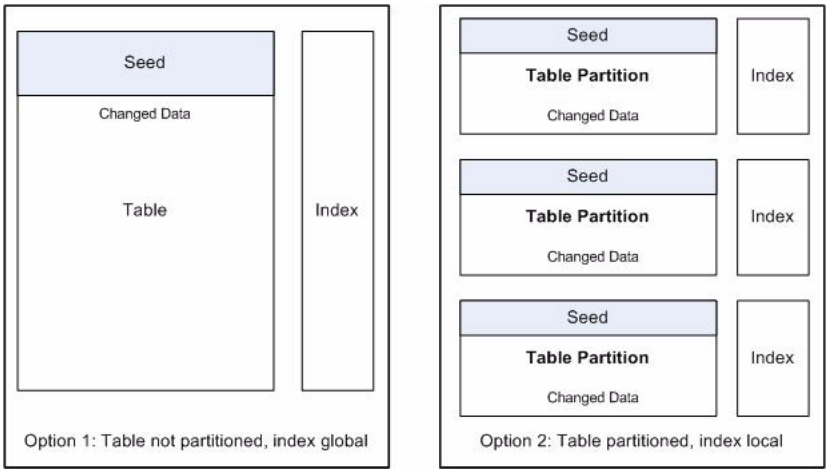
Implementing Retail Analytics Partitioning

For retailers who choose to partition a fact table, the figure 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 W_RTL_INV_IT_LC_DY_F and W_RTL_INV_IT_LC_WK_A tables). It uses table partitions and local indexes, thus minimizing the impact of index maintenance and the deletion of old table partitions. Global indexes on partitioned tables are not recommended.

Option 1 can be used for smaller compressed tables. The disadvantage is that, functionally, there is no way to delete historical data and the table continues to grow.

Figure 5–1 Retail Analytics Partitioning Options



Setup and Maintenance for Partitioning Retail Analytics Compressed Inventory Table

The following procedure describes how to setup and maintain Retail Analytics partitioning of the compressed inventory table (W_RTL_INV_IT_LC_DY_F) using Retail Analytics partitioning:

1. Make the following determinations, among others (see the *Oracle Retail Analytics Installation Guide* for details):
   - Your partitioning strategy.
   - The time period your partitions will use.
   - The 'values less than' boundaries according to your multi business calendar WID values.
   - 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.
3. Verify you have populated the Time Calendar Dimension. See the *Oracle Retail Analytics 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 W_MCAL_DAY_D (extending time calendar for a new time period).
   - Partitions are added to the Inventory Position table W_RTL_INV_IT_LC_DY_F.

Other maintenance activities include archiving and removing of partitions.

Implementing Partitioning for Compressed Inventory Table

Once the tables (including partitions) and indexes have been created, the data must be loaded. For tables that have a corresponding current status table (such as W_RTL_INV_IT_LC_DY_F and W_RTL_INV_IT_LC_G), the following steps are recommended:

> **Note:** All these steps can be performed automatically by the Retail Analytics seeding program PLP_RetailPartSeed.ksh. See the *Oracle Retail Analytics Operations Guide* for detail about how to execute this script.

1. In the partition mapping table W_RTL_PARTITION_MAP_G, update column TO_VALUE with the current business date WID for the latest partition on the target table W_RTL_INV_IT_LC_DY_F.
2. Insert a new record to partition mapping table W_RTL_PARTITION_MAP_G with next business date WID or week WID as FROM_VALUE and dummy value '999999999999999' as TO_VALUE. Column TABLE_NAME must be populated with target table W_RTL_INV_IT_LC_DY_F and column PARTITION_NAME must be populated with 'P_XX'.

> **Note:** XX is the number part of current partition name on the same target table plus 1. This partition name 'P_XX' can be different from the real partition name used in the database.

3. Copy data in W_RTL_INV_IT_LC_G table as the seed to the first partition or a new partition that is going to be used on the next day.

   At this point, only the changed records are added to the W_RTL_INV_IT_LC_DY_F table. Whereas the W_RTL_INV_IT_LC_G table is a full and uncompressed version that holds the current inventory position as of the last time period.

4. When a partition boundary is crossed, the W_RTL_INV_IT_LC_G table is copied as the seed to the new partition, via the PLP_RetailPartSeed.ksh program.

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

How Oracle Implements Partitions

This section highlights how partitions are implemented in an Oracle data warehouse.

For details on partitioning concepts, refer to the chapter Partitioning in Data Warehouses in the *Oracle Database Data Warehousing Guide 11g Release 1 (11.1)*.

Range partitions in the Oracle data warehouse/database 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. Partitioning options also include hash partitions (spreading the rows across a fixed number of partitions by applying a hash function to the partition key), and composite partitioning (a combination of range partitioning and hash partitioning). It is recommended that you partition the 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.

As a general guideline, partitioning must be considered for tables listed in the RA_partitioned_tables.xls spreadsheet (see the *Oracle Retail Analytics 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 significantly reduces backup requirements.
- Separate mount points pointing to a separate set of physical drives significantly reduces 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 W_RTL_SLS_IT_LC_DY_A table is partitioned by month, the high value for January, 2010, partition is 01-Feb-2010. A low value can always be inserted into the lowest partition. However, you may not be able to insert a high value depending on the high value of the highest partition. For instance, if the highest partition has a high value of 01-Feb-2000, and you attempt to insert a record with a date of 01-Feb-2010, the row will not be inserted into the table (the high value of 01-Feb-2010 is a non-inclusive upper bound). For this reason, a special high value partition with a key of

MAXVALUE is available in the Oracle database. It is recommended that all partitioned tables include a dummy partition with a MAXVALUE high value.

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

- ADD PARTITION: Adds a new partition to the high end of a partitioned table. Because it is recommended to have 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 the relevant index partitions.

Summary

Partitions are useful for breaking up large tables into smaller, more manageable pieces. Take note of the following partitioning recommendations:

- Consider partitioning tables that are in the RA_partitioned_tables.xls spreadsheet (see the *Oracle Retail Analytics 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 mount point.
- 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 any special requirements for seeding.

6

Performance

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

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

Oracle Retail Analytics includes ODI for extract, transform and load and Oracle Business Intelligence (BI EE) for analytic reporting purposes. The recommendations in this chapter will focus on both back end (ETL) and front end (Oracle BI EE) components of Retail Analytics.

Key Factors in Performance

Based on the complexity of the report, Oracle BI EE sometimes generates complex SQL, causing the Oracle Database to pick a less than optimized execution plan. In order to avoid this scenario, it is recommended that the "SQL Plan Baseline" functionality of the Oracle 11gR2 be enabled (it is disabled by default). For more details refer to the *Oracle 11gR2 Performance Tuning Guide*.

Purging and Archiving Strategy

With an increased use of the Retail Analytics application, the data volumes will grow and may result in slower performance. The performance impact can be on Retail Analytics batch that loads data to data warehouse tables, Retail Analytics reports, and storage.

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

- Design your archiving and purging strategy as early as possible in the Retail Analytics implementation. This helps in designing the most optimal table partitioning for large tables.
- Ensure that the data is deleted in the most optimal manner. SQL delete statements may not be the most efficient way of removing unnecessary data from Retail Analytics 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 may 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 the 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.
- An important consideration during purging is to make sure that Retail Analytics seed data (where applicable) is not deleted accidentally.

Flexible Aggregates

Retail Analytics, by default, provides several aggregate tables. For the complete list, see "Aggregates List" on page 6-11. These pre-built aggregate tables are selected based on the following:

- General usage patterns of the data
- Reporting needs (As-Is or As-Was or both)
- General aggregation ratio

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 through 1:10 is a good starting point, a ratio of 1:10 through 1:20 is good to aggregate, and a ratio of 1 to more than 20 must be aggregated.

During implementation or before, it is expected for the retailer to identify these scenarios and select the appropriate aggregate tables for best performance and usability. All the aggregate tables which are pre-packaged will have the ODI and Oracle BI EE mappings. It is highly recommended not to use Retail Analytics with all the available aggregates.

Using all these aggregations improves report performance but the improved report performance should be weighed against reduced ETL batch performance and increased storage requirement.

The reason for providing these aggregates is to give flexibility for the customer to pick appropriate levels and doesn't have to invest in customizing the product.

Below are the different groupings of aggregations. See "Aggregates List" on page 6-11 for additional details.

- As-Was aggregates
- As-Is aggregates
- As-Was Corporate aggregates
- As-Is Corporate aggregates
- Season aggregates

Even though there are different flavors of aggregations based on As-Is and As-Was, there will be few aggregate tables which will be commonly used for both As-Is and As-Was. That is because, As-Is and As-Was differentiation is only across Product and Organization Hierarchy. If the aggregation for a fact table is based on Time dimension or any dimension other than Product and Organization, then that aggregate table can be used for both As-Is and As-Was. For example, since the W_RTL_SLS_IT_LC_WK_A table is at Item and Location, it can used for either As-Was, As-Is, or both.

When the aggregations happen on any level of the Product or Organization hierarchy, there will be separate aggregates for As-Is and As-Was. For example, the W_RTL_SLS_SC_LC_DY_A and W_RTL_SLS_SC_LC_DY_CUR_A aggregates are on the subclass level of product dimension. The W_RTL_SLS_SC_LC_DY_A aggregate is for As-Was and the W_RTL_SLS_SC_LC_DY_CUR_A aggregate is for As-Is. All the As-Is aggregates are suffixed by 'CUR_A', which means 'Current.'

Note: With the exception of a Corporate aggregate in Sales, Retail Analytics out of the box does not have any aggregates across the Organization Dimension.

When the aggregation is only on a level from Product or Organization dimension then those are referred as Corporate Aggregates. These kinds of aggregates are very useful when reporting is done on any level of Product and Calendar hierarchy or Organization and Calendar hierarchy. For the list of this type of aggregates for every fact area see "Aggregates List" on page 6-11. Corporate aggregates are also classified into As-Is and As-Was because they need to be processed separately to capture the current as opposed to historical parent information.

Season aggregates are useful to do reporting specific to Season dimension. All the aggregates on Season can be used for both As-Is and As-Was.

For each group of aggregations, as mentioned above, there is a mandatory aggregate table that needs to be used for other selections of the aggregates and that can be identified in the FlexAggregates document with the highlighted text.

For example, if the business only needs As-Is, the following points need to be considered:

1. Get the general usage patterns of the data and aggregation ratio. Based on that, select the list of aggregate tables. This may not be accurate for the first time but can always be changed over a period of time based on the usage and the changing data.
2. Ensure that you disable/freeze all the As-Was aggregate jobs and some of the As-Is aggregates which were not selected, in ODI and disable the same in Oracle BI EE as well. See the *Oracle Retail Analytics Operations Guide* for more information.

This section only covers As-Is and As-Was aggregates, but Retail Analytics also offers PIT (Point in Time) reporting, which does not require any special processing of data. There are no special tables or ODI jobs for PIT. In Oracle BI EE there is a separate subject area for PIT reporting. For additional information on PIT see the *Oracle Retail Analytics User Guide*. PIT reporting is always done from the base fact tables or the corporate aggregate tables. If PIT is required along with As-Is, As-Was, or both then choose the corporate aggregates so that all the three reporting scenarios will be benefited.

ETL Programs Performance

Setting ETL Program Multi-threading

Retail Analytics base fact extract and load programs can be configured to run using multiple threads. The default number of threads for these programs is set to one and can be configured based on requirements. For additional information on how multi-threading works, see the Program Overview chapter of the *Oracle Retail Analytics Operations Guide*.

1. Finalize the multi-threading strategy for the base fact extract and load programs.
2. Number of threads for each program may vary based on the data volume that program handles and resource availability.
3. In the C_ODI_PARAM table, update the value of the PARAM_VALUE column to the desired number of threads. This applies to all records with the value 'LOC_NUM_OF_THREAD' in the PARAM_NAME column and the name of the program that requires multi-threading set in the SCENARIO_NAME column. See an example below for scenario named SDE_Test, where the desired number of threads needs to be set to 2 from 1 (default).

```
UPDATE C_ODI_PARAM
SET PARAM_VALUE = 2
WHERE PARAM_NAME = 'LOC_NUM_OF_THREAD'
AND SCENARIO_NAME = 'SDE_Test'
```

4. If the number of thread required is more than 10, you need to modify the DDL for intermediate temp tables used by the ODI scenario. DDL changes require adding extra partitions to hold the data. The number of partitions on the intermediate temp table must be the same or higher than the required number of threads (which is the value for LOC_NUM_OF_THREADS set in the previous step).

ODI Configuration

ODI must be configured prior to implementing Retail Analytics. See the *Oracle Retail Analytics Installation Guide* for details on configuring ODI.

ETL Batch Scheduling

- Set up the proper dependencies between the applications to ensure resources are fully utilized, which helps the nightly batch finish earlier.
- Retail Analytics load programs (SIL programs) must not wait for all the extraction programs (sde) to finish before starting. Some of them can start executing as soon as the corresponding staging table is populated. For more information on setting up dependencies, refer to the *Oracle Retail Analytics Operations Guide*.
- 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 that your source applications batch is optimized. Retail Analytics runs towards the end of the nightly batch. Retail Analytics jobs are often the last jobs to start due to the dependencies on the source system jobs, so Retail Analytics is often the last to finish. Optimizing the source applications batch helps Retail Analytics jobs start earlier.

Additional Considerations

- Sort the W_RTL_INV_IT_LC_G table data after the data is seeded for the first time to improve ETL performance.
- In a production environment, fact tables with large data volume can be created with the No Logging option. This improves the ETL performance and can be implemented on a case by case basis.

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 Delivering Content chapter of the *Oracle Business Intelligence Enterprise Edition User Guide* and the Configuring Oracle BI Scheduler chapter of the *Oracle Business Intelligence System Administrator's 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 must not be performed at these levels. This ensures that reports do not produce misleading or invalid results. For example, reports must not be designed to request planning data at the item level because planning data is only available at the subclass level and above.
- As-Is reporting for all the positional facts such as inventory, cost and so on is only possible at the corporate level aggregates.
- 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 Configuring Oracle BI Scheduler chapter of the *Oracle Business Intelligence System Administrator's Guide*.

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 permits a query to be run one time for multiple runnings of a query and not necessarily every time the query is run. The query cache allows the Oracle BI Server to satisfy many subsequent query requests without having to access back-end data sources (such as Oracle database). This reduction in communication costs can dramatically decrease query response time.

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

For more details on Caching refer to the Managing Performance Tuning and Query Caching chapter in the *Oracle BI EE System Administrator's Guide*.

Partitioning Strategy

Database level table partitioning is very important for ETL batch and report performance. For more information, see Chapter 5, "Compression and Partitioning".

Data Base Configuration

Retail Analytics is built on Oracle Database 11g Release 2 and must be optimized and configured for a retailers' needs. Refer to the Setting up your Data Warehouse System chapter of the *Oracle 11g Data Warehouse Guide*.

Adequate Hardware Resources

ETL program and report performance are highly dependent on the hardware resources. For more information, see Chapter 2, "Setup and Configuration".

Leading Practices

Customizations

Changes and modifications to the Retail Analytics delivered code or development of new code is considered customization. Retail Analytics does not support custom code developed by clients unless the issue related to customization can be recreated using Retail Analytics delivered objects. Listed below are recommendations that will help you in maintaining Retail Analytics code:

- Naming convention: it is recommended that you use a good and consistent naming convention when customizing Retail Analytics delivered code or building new code in the Retail Analytics environment.

  This strategy is helpful in identifying custom code and also helps when merging a retailer's Retail Analytics repository with future releases of the Retail Analytics repository. There is a possibility of losing customizations to Retail Analytics provided ODI scripts or Oracle BI EE repository, if the customized code uses the same object/script names that are used by Retail Analytics.

- As a best practice, keep all the documentation up-to-date for capturing any changes or new code that has been developed at a site. For example, if table structure has been customized, create or update the custom Data Model Guide with these changes.

- While customizing the rpd, do not make any changes directly on the main shipped/original rpd. Make a copy of the original rpd and start the changes on the copied rpd which will be the modified version. This is useful while applying any patches in future releases of Retail Analytics through Oracle BI EE's merge utility. For more details refer to the Managing Oracle BI Repository Files chapter of the *Oracle BI EE Metadata Repository Builder's Guide*.

ODI Best Practices

For customizations to existing ODI code or while creating new ODI code, refer to the *ODI Best Practices Guide* included with your product code.

Oracle BI EE 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 must be set to non-drillable.
- In the presentation layer, fact folders (presentation tables) must contain only metrics and dimension folders (presentation tables) must 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 Analytics Operations Guide*. Any manual intervention should be avoided.

Recoverability

Set up the batch schedule in such a manner that the batch can resume from the point where it failed.

Retail Analytics Loading Batch Execution Catch-Up

Loading batch (sil) execution catch-up can be achieved by backing up the staging table data. The following scenarios explain when users can benefit from this. This approach is considered a customization on Retail Analytics programs and is not supported.

- Catch-up: When Retail Analytics is not ready for implementation, users can use history data stored in the staging backup tables (explained later in this section) to catch-up on the data loading once the system is implemented or becomes available.
- Retail Analytics database systems are down: When Retail Analytics database systems are down, users can use history data stored in the staging backup tables (explained later in this section) to load the data once the system becomes available.

The following steps illustrate/show how the Loading Batch Execution catch-up solution works:

1. Create Retail Analytics staging tables for each corresponding staging table using the DDL for the staging tables provided. For more information, see the *Oracle Retail Analytics Installation Guide*.
2. Set up the Retail Analytics ODI Extract (SDE) programs, so they are ready to be executed against source applications and load into the staging tables created in previous setup.
3. Create a one-to-one backup table for each staging table. This backup table uses the same DDL as the staging table along with an additional field (load_date) which can be mapped to source system business date. This date is used as a filter when the backup data is ready to be moved to the staging table.
4. Execute the SDE program to populate the staging tables created in the first step.
5. Move staging table data into backup staging table with the correct business date. By default, Retail Analytics does not provide the backup table DDL or backup data population scripts.
6. Repeat the process of executing the SDE program and taking the backup to the staging backup table periodically (daily, once in two days, weekly, and so on), until the Retail Analytics systems are available. Note that the staging table only contains the current business day's data, while the backup staging table contains data for all the business dates when the program was executed.
7. Once the Retail Analytics systems become available, move the backup staging table data to the staging tables, one day at a time. This can be done by using 'load_date' as a filter on the source backup staging table data.

8. Once one business date data is moved to the staging table, SIL programs and corresponding PLP programs need to be executed for loading data into final data warehouse tables.
9. Repeat the process of moving data from backup staging to staging and executing SIL and PLP programs until all the data for all business dates from backup staging tables is loaded into the fact and dimension tables.

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 instance 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, you must run your queries from the reporting mirror area, not from core data warehouse area. Take the following into consideration:

- Consider this approach for large data warehouse implementations.
- Creating data marts can be a good option when implementing mirroring.
- Build a 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 the 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.

Aggregates List

The table below lists Retail Analytics aggregates grouped by subject area and aggregation type.

Table 6–1 Retail Analytics Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Inventory Position	W_RTL_INV_IT_LC_DY_F	W_RTL_INV_IT_LC_WK_A, W_RTL_INV_SC_LC_DY_A, W_RTL_INV_SC_LC_WK_A, W_RTL_INV_CL_LC_DY_A, W_RTL_INV_CL_LC_WK_A, W_RTL_INV_DP_LC_DY_A, W_RTL_INV_DP_LC_WK_A		W_RTL_INV_IT_DY_A, W_RTL_INV_IT_WK_A, W_RTL_INV_SC_DY_A, W_RTL_INV_SC_WK_A, W_RTL_INV_SC_DY_CUR_A, W_RTL_INV_SC_WK_CUR_A	W_RTL_INV_IT_DY_A, W_RTL_INV_IT_WK_A, W_RTL_INV_SC_DY_CUR_A, W_RTL_INV_SC_WK_CUR_A	
Inventory Receipts	W_RTL_INVRC_IT_LC_DY_F	W_RTL_INVRC_IT_LC_WK_A, W_RTL_INVRC_SC_LC_DY_A, W_RTL_INVRC_SC_LC_WK_A, W_RTL_INVRC_CL_LC_WK_A, W_RTL_INVRC_CL_LC_DY_A, W_RTL_INVRC_DP_LC_DY_A, W_RTL_INVRC_DP_LC_WK_A	W_RTL_INVRC_SC_LC_DY_CUR_A, W_RTL_INVRC_SC_LC_WK_CUR_A	W_RTL_INVRC_IT_DY_A, W_RTL_INVRC_IT_WK_A, W_RTL_INVRC_SC_DY_A, W_RTL_INVRC_SC_WK_A, W_RTL_INVRC_SC_DY_CUR_A, W_RTL_INVRC_SC_WK_CUR_A	W_RTL_INVRC_IT_DY_A, W_RTL_INVRC_IT_WK_A, W_RTL_INVRC_SC_DY_CUR_A, W_RTL_INVRC_SC_WK_CUR_A	
Markdown	W_RTL_MKDN_IT_LC_DY_F	W_RTL_MKDN_IT_LC_WK_A, W_RTL_MKDN_SC_LC_DY_A, W_RTL_MKDN_SC_LC_WK_A, W_RTL_MKDN_CL_LC_DY_A, W_RTL_MKDN_CL_LC_WK_A, W_RTL_MKDN_DP_LC_DY_A, W_RTL_MKDN_DP_LC_WK_A	W_RTL_MKDN_SC_LC_DY_CUR_A, W_RTL_MKDN_SC_LC_WK_CUR_A, W_RTL_MKDN_CL_LC_DY_CURR_A, W_RTL_MKDN_CL_LC_WK_CURR_A, W_RTL_MKDN_DP_LC_DY_CURR_A, W_RTL_MKDN_DP_LC_WK_CURR_A	W_RTL_MKDN_IT_DY_A, W_RTL_MKDN_IT_WK_A, W_RTL_MKDN_SC_DY_A, W_RTL_MKDN_SC_WK_A, W_RTL_MKDN_SC_DY_CURR_A, W_RTL_MKDN_SC_WK_CURR_A	W_RTL_MKDN_IT_DY_A, W_RTL_MKDN_IT_WK_A, W_RTL_MKDN_SC_DY_CURR_A, W_RTL_MKDN_SC_WK_CURR_A	W_RTL_MKDN_IT_LC_DY_SN_A, W_RTL_MKDN_IT_LC_WK_SN_A, W_RTL_MKDN_IT_DY_SN_A, W_RTL_MKDN_IT_WK_SN_A
Net Cost	W_RTL_NCOST_IT_LC_DY_F			W_RTL_NCOST_IT_DY_A	W_RTL_NCOST_IT_DY_A	
Net Profit	W_RTL_NPROF_IT_LC_DY_F	W_RTL_NPROF_IT_LC_WK_A, W_RTL_NPROF_SC_LC_DY_A, W_RTL_NPROF_SC_LC_WK_A, W_RTL_NPROF_CL_LC_DY_A, W_RTL_NPROF_CL_LC_WK_A, W_RTL_NPROF_DP_LC_DY_A, W_RTL_NPROF_DP_LC_WK_A		W_RTL_NPROF_IT_DY_A, W_RTL_NPROF_IT_WK_A, W_RTL_NPROF_SC_DY_A, W_RTL_NPROF_SC_WK_A	W_RTL_NPROF_IT_DY_A, W_RTL_NPROF_IT_WK_A	

Table 6–1 Retail Analytics Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Planning	W_RTL_MFPCPC_SC_CH_WK_F, W_RTL_MFPOPC_SC_CH_WK_F, W_RTL_MFPCPR_SC_CH_WK_F, W_RTL_MFPOPR_SC_CH_WK_F					
Pricing	W_RTL_PRICE_IT_LC_DY_F			W_RTL_PRICE_IT_DY_A	W_RTL_PRICE_IT_DY_A	
Sales Transaction	W_RTL_SLS_TRX_IT_LC_DY_F	W_RTL_SLS_IT_LC_DY_A, W_RTL_SLS_IT_LC_WK_A, W_RTL_SLS_SC_LC_DY_A, W_RTL_SLS_SC_LC_WK_A, W_RTL_SLS_CL_LC_DY_A, W_RTL_SLS_CL_LC_WK_A, W_RTL_SLS_DP_LC_DY_A, W_RTL_SLS_DP_LC_WK_A	W_RTL_SLS_IT_LC_DY_A, W_RTL_SLS_IT_LC_WK_A, W_RTL_SLS_SC_LC_DY_CUR_A, W_RTL_SLS_SC_LC_WK_CUR_A, W_RTL_SLS_CL_LC_DY_CUR_A, W_RTL_SLS_CL_LC_WK_CUR_A, W_RTL_SLS_DP_LC_DY_CUR_A, W_RTL_SLS_DP_LC_WK_CUR_A	W_RTL_SLS_IT_DY_A, W_RTL_SLS_IT_WK_A, W_RTL_SLS_SC_DY_A, W_RTL_SLS_SC_WK_A, W_RTL_SLS_SC_DY_CUR_A, W_RTL_SLS_SC_WK_CUR_A, W_RTL_SLS_LC_DY_A, W_RTL_SLS_LC_WK_A	W_RTL_SLS_IT_DY_A, W_RTL_SLS_IT_WK_A, W_RTL_SLS_SC_DY_CUR_A, W_RTL_SLS_SC_WK_CUR_A, W_RTL_SLS_LC_DY_A, W_RTL_SLS_LC_WK_A	W_RTL_SLS_IT_LC_DY_SN_A, W_RTL_SLS_IT_LC_WK_SN_A, W_RTL_SLS_IT_DY_SN_A, W_RTL_SLS_IT_WK_SN_A
Sales Forecast	W_RTL_SLSFC_IT_LC_DY_F, W_RTL_SLSFC_IT_LC_WK_F	W_RTL_SLSFC_SC_LC_DY_A, W_RTL_SLSFC_SC_LC_WK_A	W_RTL_SLSFC_SC_LC_DY_CUR_A, W_RTL_SLSFC_SC_LC_WK_CUR_A	W_RTL_SLSFC_IT_DY_A, W_RTL_SLSFC_IT_WK_A, W_RTL_SLSFC_SC_DY_A, W_RTL_SLSFC_SC_WK_A	W_RTL_SLSFC_IT_DY_A, W_RTL_SLSFC_IT_WK_A, W_RTL_SLSFC_SC_DY_CUR_A, W_RTL_SLSFC_SC_WK_CUR_A	W_RTL_SLSFC_IT_LC_DY_SN_A, W_RTL_SLSFC_IT_LC_WK_SN_A, W_RTL_SLSFC_IT_DY_SN_A, W_RTL_SLSFC_IT_WK_SN_A
Sales Pack	W_RTL_SLSPK_IT_LC_DY_F	W_RTL_SLSPK_IT_LC_WK_A		W_RTL_SLSPK_IT_DY_A, W_RTL_SLSPK_IT_WK_A	W_RTL_SLSPK_IT_DY_A, W_RTL_SLSPK_IT_WK_A	W_RTL_SLSPK_IT_LC_DY_SN_A, W_RTL_SLSPK_IT_LC_WK_SN_A, W_RTL_SLSPK_IT_DY_SN_A, W_RTL_SLSPK_IT_WK_SN_A
Sales Promotion	W_RTL_SLSPR_IT_LC_DY_F					
Stock Ledger	W_RTL_STCK_LDGR_SC_LC_WK_F, W_RTL_STCK_LDGR_SC_LC_MH_F					

Table 6–1 Retail Analytics Aggregates

Fact	Base Fact Table	Aggregate Tables As-Was	Aggregate Tables As-Is	Corporate Aggregate Tables (As-Was)	Corporate Aggregate Tables (As-Is)	Season Aggregates (As-Is/As-Was)
Supplier Compliance	W_RTL_SUPPCM_IT_LC_DY_F, W_RTL_SUPPCMUF_LC_DY_F	W_RTL_SUPPCM_IT_LC_WK_A, W_RTL_SUPPCMUF_LC_WK_A	W_RTL_SUPPCM_IT_LC_WK_A, W_RTL_SUPPCMUF_LC_WK_A	W_RTL_SUPPCM_LC_WK_A	W_RTL_SUPPCM_LC_WK_A	
Supplier Invoice	W_RTL_SUPP_IVC_PO_IT_F					
Unit Cost	W_RTL_BCOST_IT_LC_DY_F			W_RTL_BCOST_IT_DY_A	W_RTL_BCOST_IT_DY_A	
Wholesale Franchise	W_RTL_SLSWF_IT_LC_DY_F	W_RTL_SLSWF_IT_LC_WK_A, W_RTL_SLSWF_SC_LC_DY_A, W_RTL_SLSWF_SC_LC_WK_A	W_RTL_SLSWF_IT_LC_WK_A, W_RTL_SLSWF_SC_LC_DY_CUR_A, W_RTL_SLSWF_SC_LC_WK_CUR_A	W_RTL_SLSWF_IT_DY_A, W_RTL_SLSWF_IT_WK_A	W_RTL_SLSWF_IT_DY_A, W_RTL_SLSWF_IT_WK_A	

7

Frequently Asked Questions

The following issues may be encountered while implementing Retail Analytics. The accompanying solutions will help you work through the issues.

Issue:
Why am I getting the Login Denied error with the following message when I try to run a report using Oracle BI Presentation Services?

ORACLE ERROR CODE: 1017, MESSAGE: ORA-01017: INVALID USERNAME/PASSWORD; LOGON DENIED
Solution:
Ensure that the repository connection pool has the right login credentials in the Oracle BI Administration Tool and check the tnsnames.ora file.

Issue:
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
Solution:
Ensure the repository connection pool has the right login credentials in Oracle BI Administration tool or check the tnsnames.ora entry.

Issue:
Why can't I see query activity at the individual user level in the NQQUERY.LOG file?

Solution:
Check the logging level field in the user dialog box in the User tab. If the logging level is set to zero, the administrator may have disabled query logging. Contact the Oracle BI administrator to enable query logging.

Issue:
Why is the data not loaded to the fact table, even though I have valid data in the staging table?

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

Issue:
Why is the data not loaded to the dimension table, even though I have valid data in staging table?

Solution:
Parent data may be missing in the corresponding dimension table. This applies to dimensions with hierarchy.

Issue:
Why is the data not loaded to the fact table, even though I have valid data in the fact staging tables and all the corresponding keys in the dimensions?

Solution:
Check the effective start and end date values in the dimension tables. If any of these dimension's effective from date values is greater than the fact date value, those will not be loaded to the fact tables as those are the future dimension records.

Issue:
Description of a subclass is changed from a to b in the source system but I cannot see both the records in Retail Analytics after the loading process?

Solution:
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; as it is tracked as scd type 1 change. For more information, refer to the *Oracle Retail Analytics Operations Guide.*

Issue:
Why the extract or the load program performance is not improving even after using ODI multi-threading?

Solution:
This can occur because of several reasons. Check the following settings:

- The number of threads must be appropriate for the hardware and data volume.
- The number of partitions on the intermediate temp table must be equal to or higher than the number of threads.