Oracle® Fusion Middleware

Administrator's Guide for Oracle Business Intelligence Applications

11g Release 1 (11.1.1.7)

E37988-01

April 2013

Provides topics of interest to system administrators, including customization, multi-language support, localizing deployments, and using Oracle GoldenGate.

ORACLE

Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Applications, 11g Release 1 (11.1.1.7)

E37988-01

Primary Author: Dan Hilldale

Contributors: Oracle Business Intelligence development, product management, and quality assurance teams.

Contents

Preface vii

Audience vii
Documentation Accessibility vii
Related Documentation vii
Conventions viii

What's New in This Release ix

Notable Features in Oracle BI Applications Documented in This Guide ix

1 Customizing the Oracle Business Analytics Warehouse

1.1 Overview of Customization in Oracle Business Intelligence Applications 1-1
1.1.1 What is Customization in Oracle Business Intelligence Applications? 1-1
1.1.2 About the Customization Process 1-2
1.1.3 About the Impact of Patch Installation on Customizations 1-3
1.2 Category 1 Customizations: Adding Columns to Existing Fact or Dimension Tables 1-3
1.2.1 About Extending Mappings 1-4
1.2.2 Typical Steps to Extend Mappings in the Oracle Business Analytics Warehouse 1-4
1.2.3 Other Types of Customizations Requiring Special Handling 1-6
1.2.3.1 How to Modify Category 2 SCD Behavior 1-6
1.2.3.2 How to Add A Dimension to an Existing Fact 1-7
1.2.3.2.1 Create a Custom Dimension Datastore and Tasks 1-7
1.2.3.2.2 Customize Fact Datastores and Tasks 1-9
1.2.3.3 How to Add a DATE_WID column to a Fact 1-11
1.3 Category 2 Customizations: Adding Additional Tables 1-12
1.3.1 About Creating New Tables 1-12
1.3.1.1 About the Main Required Columns 1-12
1.3.2 About the DATASOURCE_NUM_ID Column 1-12
1.3.3 Additional Information About Customizing 1-13
1.3.3.1 About the Update Strategy 1-13
1.3.3.2 About Indices and Naming Conventions 1-13
1.3.4 Adding a New Fact Table to the Oracle Business Analytics Warehouse 1-13
1.4 Category 3 Customizations: Adding New Data as a Whole Row into a Standard Dimension Table 1-15
1.4.1 How to Add New Data as a Whole Row Into a Standard Dimension Table 1-15
1.4.2 Configuring Extracts 1-16

1.4.2.1 Extracting Additional Data ... 1-16
1.4.2.1.1 Extracting New Data Using an Existing Source Table ... 1-16
1.4.2.1.2 Extracting Data from a New Source Table ... 1-16
1.4.2.2 Setting Up the Delimiter for a Source File ... 1-17
1.4.3 Configuring Loads ... 1-18
1.4.3.1 About Primary Extract and Delete Mappings Process ... 1-18
1.4.3.2 About Working with Primary Extract and Delete Mappings ... 1-19
1.4.3.2.1 Deleting the Configuration for Source-Archived Records ... 1-19
1.5 Customizing Stored Lookups and Adding Indexes ... 1-19
1.5.1 About Stored Lookups ... 1-19
1.5.1.1 About Resolving Dimension Keys ... 1-20
1.5.1.1.1 Resolving the Dimension Key Using Lookup ... 1-20
1.5.2 How to add an index to an existing fact or dimension table ... 1-20

2 About Multi-Language Support

2.1 Introduction to Multi-Language Support ... 2-1
2.2 About Pseudo-Translations ... 2-2
2.3 About Oracle BI Applications Domains ... 2-2
2.4 About Dimension Translation Tables ... 2-4

3 Localizing Oracle Business Intelligence Deployments

3.1 Process of Maintaining Translation Tables for Oracle BI EE ... 3-1
3.1.1 Upgrading Oracle Business Intelligence Seed Data for Non-English Locales ... 3-2
3.1.1.1 Verify the Translation Table (W_LOCALIZED_STRING_G and Corresponding Indexes ... 3-2
3.1.1.2 Importing Locale Seed Data Into The Translation Table (W_LOCALIZED_STRING_G) ... 3-3
3.1.2 Externalizing Customer Metadata Strings ... 3-4
3.1.3 Adding Custom Translations to the W_LOCALIZED_STRING_G Table ... 3-4
3.1.3.1 Adding String Translations for Analytics Metadata ... 3-5
3.2 About Translating Presentation Services Strings ... 3-5
3.3 Changing the Default Currency in Analytics Applications ... 3-6

4 Oracle Business Analytics Warehouse Naming Conventions

4.1 Naming Conventions for Oracle Business Analytics Warehouse Tables ... 4-1
4.2 Table Types for Oracle Business Analytics Warehouse ... 4-2
4.2.1 Aggregate Tables in Oracle Business Analytics Warehouse ... 4-4
4.2.2 Dimension Class Tables in Oracle Business Analytics Warehouse ... 4-4
4.2.3 Dimension Tables in Oracle Business Analytics Warehouse ... 4-4
4.2.4 Dimension Tables With Business Role-Based Flags ... 4-4
4.2.5 Fact Tables in Oracle Business Analytics Warehouse ... 4-5
4.2.6 Helper Tables in Oracle Business Analytics Warehouse ... 4-5
4.2.7 Hierarchy Tables in Oracle Business Analytics Warehouse ... 4-5
4.2.8 Mini-Dimension Tables in Oracle Business Analytics Warehouse ... 4-5
4.2.9 Staging Tables in Oracle Business Analytics Warehouse ... 4-6
4.2.10 Translation Tables in Oracle Business Analytics Warehouse ... 4-6

4.3 Internal Tables in Oracle Business Analytics Warehouse 4-6
4.4 Standard Column Prefixes in Oracle Business Analytics Warehouse 4-7
4.5 Standard Column Suffixes in Oracle Business Analytics Warehouse 4-7
4.6 System Columns in Oracle Business Analytics Warehouse Tables 4-8
4.7 Multi-Currency Support for System Columns 4-9
4.8 Oracle Business Analytics Warehouse Primary Data Values 4-10
4.9 About Multi-Language Support in the Oracle Business Analytics Warehouse 4-10
4.10 Oracle Business Analytics Warehouse Currency Preferences 4-10

5 Administering Oracle GoldenGate and Source Dependent Schemas

5.1 Introduction 5-1
5.2 Source Dependent Schema Architecture 5-1
5.3 Setting Up Oracle GoldenGate and Source Dependent Schemas 5-2
5.4 Tasks for Setting Up Oracle GoldenGate and the Source Dependent Schema 5-2
5.4.1 Setup Step: Configure Source and Target Database 5-3
5.4.2 Setup Step: Install Oracle GoldenGate on Source and Target Systems 5-5
5.4.3 Setup Step: Configure BI Applications Configuration Manager and Oracle Data Integrator to Support the Source Dependent Schema 5-8
5.4.4 Setup Step: Generate, Deploy, and Populate the Source Dependent Schema Tables on Target Database 5-9
5.4.5 Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Machines 5-12
5.4.5.1 Generate Oracle GoldenGate Parameter Files 5-13
5.4.5.2 Configure the Source System 5-15
5.4.5.3 Configure the Target System 5-17
5.4.6 Setup Step: Start Oracle GoldenGate on Source and Target Machines 5-18

Index

Preface

Oracle Business Intelligence Applications is comprehensive suite of prebuilt solutions that deliver pervasive intelligence across an organization, empowering users at all levels - from front line operational users to senior management - with the key information they need to maximize effectiveness. Intuitive and role-based, these solutions transform and integrate data from a range of enterprise sources and corporate data warehouses into actionable insight that enables more effective actions, decisions, and processes.

Oracle BI Applications is built on Oracle Business Intelligence Suite Enterprise Edition (Oracle BI EE), a comprehensive set of enterprise business intelligence tools and infrastructure, including a scalable and efficient query and analysis server, an ad-hoc query and analysis tool, interactive dashboards, proactive intelligence and alerts, and an enterprise reporting engine.

Audience

This document is intended for system administrators and ETL team members who are responsible for managing Oracle BI Applications. It contains information about ETL customization, domains and localization, Oracle Business Analytics Warehouse naming conventions, and system administration tasks, including setting up and using Oracle GoldenGate and Source-Dependent Schemas to support ETL performance.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc`.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Related Documentation

See the Oracle Business Intelligence Applications documentation library for a list of related Oracle Business Intelligence Applications documents: `http://docs.oracle.com/cd/E38317_01/index.htm`

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.

What's New in This Release

Oracle BI Applications 11.1.1.7 is a new release. This chapter describes features in Oracle Business Intelligence Applications 11g Release 1 (11.1.1.7) documented in this guide, *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Applications*, that may be of note.

This chapter contains the following topics:

- Notable Features in Oracle BI Applications Documented in This Guide

Notable Features in Oracle BI Applications Documented in This Guide

New features in Oracle BI Applications 11g Release 1 (11.1.1.7) that are documented in *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Applications* include the following:

Oracle GoldenGate Support

Oracle BI Applications optionally provides source dependent schemas, supported by Oracle GoldenGate, which replicate OLTP schemas in the same database as the Oracle Business Analytics Warehouse schema, relieving contention and enhancing performance during Extract-Transform-Load processing. Chapter 5, "Administering Oracle GoldenGate and Source Dependent Schemas," explains how to set up and use Oracle GoldenGate and source-dependent schemas.

1

Customizing the Oracle Business Analytics Warehouse

This chapter describes concepts and techniques for customizing the ETL functionality in Oracle Business Intelligence Applications.

This chapter contains the following topics:

- Section 1.1, "Overview of Customization in Oracle Business Intelligence Applications"
- Section 1.2, "Category 1 Customizations: Adding Columns to Existing Fact or Dimension Tables"
- Section 1.3, "Category 2 Customizations: Adding Additional Tables"
- Section 1.4, "Category 3 Customizations: Adding New Data as a Whole Row into a Standard Dimension Table"
- Section 1.5, "Customizing Stored Lookups and Adding Indexes"

1.1 Overview of Customization in Oracle Business Intelligence Applications

This section provides an overview of customization in Oracle Business Intelligence Applications, and contains the following topics:

- Section 1.1.1, "What is Customization in Oracle Business Intelligence Applications?"
- Section 1.1.3, "About the Impact of Patch Installation on Customizations"

1.1.1 What is Customization in Oracle Business Intelligence Applications?

In Oracle Business Intelligence Applications, customization is defined as changing the preconfigured behavior to enable you to analyze new information in your business intelligence dashboards. For example, you might want to add a column to a dashboard by extracting data from the field HZ_CUST_ACCOUNTS.ATTRIBUTE1 and storing it in the Oracle Business Analytics Warehouse in the X_ACCOUNT_LOG field.

The type of data source that you have determines the type of customization that you can do. Data sources can be one of the following types:

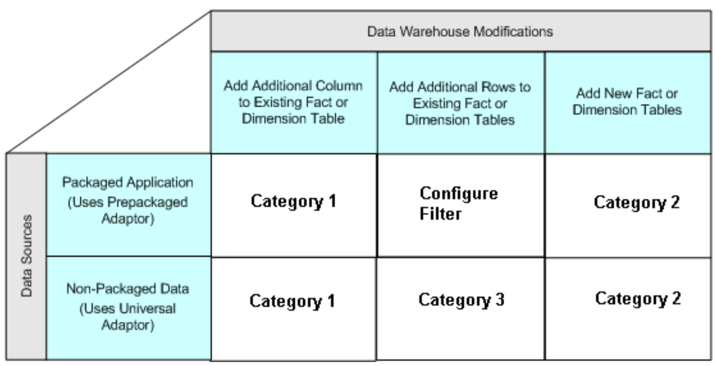
- Packaged applications (for example, Oracle EBS), which use prepackaged adapters.
- Non-packaged data sources, which use the Universal adapter.

Customizations are grouped into the following categories:

- **Category 1.** In a Category 1 customization, you add additional columns from source systems that have pre-packaged adapters and load the data into existing Oracle Business Analytics Warehouse tables. For more information about performing Category 1 customizations, see Section 1.2, "Category 1 Customizations: Adding Columns to Existing Fact or Dimension Tables".
- **Category 2.** In a Category 2 customization, you use pre-packaged adapters to add new fact or dimension tables to the Oracle Business Analytics Warehouse. Category 2 customizations normally require that you build new SDE and SIL mappings. For more information about performing Category 2 customizations, see Section 1.3, "Category 2 Customizations: Adding Additional Tables".
- **Category 3.** In a Category 3 customization, you use the Universal adapter to load data from sources that do not have pre-packaged adapters. For more information about performing Category 3 customizations, see Section 1.4, "Category 3 Customizations: Adding New Data as a Whole Row into a Standard Dimension Table".

The figure below summarizes the category of customization that you can perform for each type of data source and type of modification.

Figure 1–1 Supported customizations based on data source



For detailed information about tables and naming conventions, see *Oracle Business Analytics Warehouse Data Model Reference*.

When you customize ETL Packages and Interfaces, you usually work in the \Oracle BI Applications\Mappings folder in the Projects view in ODI Studio's Designer Navigator.

Note: The customization methodology is to make a copy of the ETL task and version both the original and copy while a datastore is simply versioned. These versions allow you to revert functionality if required as well as identify changes that have been introduced through customization, patches or upgrades.

1.1.2 About the Customization Process

This chapter explains how to customize your ETL functionality, after you have performed a Business Analysis and Technical Analysis. This chapter does not cover the other typical tasks that you need to perform, as follows:

- Business Analysis - before you start customization, you typically analyze your current BI dashboards to determine the changes you need to support your business or organization.

- Technical Analysis - when you have identified your business requirements, you need to determine the technical changes you need to make, by identifying source tables, staging tables, target tables, and ODI Packages and Interfaces that you need to modify.
- RPD Modification - having made the customizations in the ETL functionality, you need to modify your RPD to expose the new data in your dashboards. For more information about RPD modification, refer to the Oracle Business Intelligence Enterprise Edition documentation library.

1.1.3 About the Impact of Patch Installation on Customizations

This section explains what you must do to re-apply a customization to an object that has been patched. For example, if you install an Oracle Business Intelligence Applications patch that modifies the Supply Chain and Order Management application, you might need to manually re-apply customizations that you have made to the Supply Chain and Order Management application.

As part of customizing an ETL task (including interfaces and package under a specific task folder), you copy the task folder to be customized, version the original and version the copy. Any patches are applied to the current version of the original task. Leverage ODI's version compare utility to identify the changes introduced by the patch. The copy is also versioned so that any changes introduced can be isolated. Compare any changes with those introduced by the patch and verify there is no conflict, then manually apply the same changes introduced by the patch to the customized ETL tasks. For information about modifying and versioning ETL customizations, refer to Section 1.2.2, "Typical Steps to Extend Mappings in the Oracle Business Analytics Warehouse".

A patch only installs changed repository objects, not the whole Work Repository. Therefore, you only need to re-apply customizations to mappings that have been changed by the patch. For example, if a patch only modifies the Supply Chain and Order Management application, you only need to manually re-apply customizations that you have made to the Supply Chain and Order Management application. Customizations in other applications are not affected by the patch.

Note

All customization steps have you create a 'Custom' adaptor folder where customized ETL tasks are stored. This is not required but is considered a best practice to make identifying customized content easier.

1.2 Category 1 Customizations: Adding Columns to Existing Fact or Dimension Tables

Category 1 customizations add additional columns from source systems that have pre-packaged adapters and load the data into existing Oracle Business Analytics Warehouse tables.

This section contains the following topics:

- Section 1.2.1, "About Extending Mappings"
- Section 1.2.2, "Typical Steps to Extend Mappings in the Oracle Business Analytics Warehouse"
- Section 1.2.3, "Other Types of Customizations Requiring Special Handling"

1.2.1 About Extending Mappings

Category 1 customizations involve extracting additional columns from source systems for which pre-packaged adapters are included (for example, Oracle eBusiness Suite) and loading the data into existing Oracle Business Analytics Warehouse tables. For Category 1 customizations, data can also come from non-packaged sources, but this section assumes that the sources have already been mapped with a Universal adapter and only need to be extended to capture additional columns. (The initial mapping of a Universal adapter is considered a Category 3 customization. For information, see Section 1.4, "Category 3 Customizations: Adding New Data as a Whole Row into a Standard Dimension Table".)

In order to see additional columns in the Oracle Business Analytics Warehouse, the columns must first be passed through the ETL process. The existing mappings and tables are extensible. Oracle Business Intelligence Applications provides a methodology to extend preconfigured mappings to include these additional columns and load the data into existing tables.

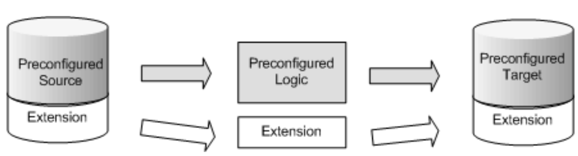
Oracle Business Intelligence Applications recognizes two types of customization: extension and modification. The supported extension logic allows you to add to existing objects. For example, you can extract additional columns from a source, pass them through existing mappings, and populate new columns added to an existing table. Generally, Oracle Business Intelligence Applications does not allow you to modify existing logic or columns. You should not change existing calculations to use different columns, and you should not remap existing columns to be loaded from different sources.

For example, if you want to calculate revenue differently from the existing logic, you should create a new column (for example, X_REVENUE) and populate it with a custom mapping expression. You can then remap the Oracle Business Intelligence repository to point to the new X_REVENUE column.

Most datastores have a single placeholder column named X_CUSTOM. Each ETL task has mapping expressions to populate this column. These serve as templates for customizing ODI datastores and interfaces. When creating new custom columns, follow the naming convention of including the X_ prefix to help distinguish custom columns.

In the figure below, the preconfigured logic is shaded in gray. You should not modify anything contained within these objects. You should add customizations to existing objects rather than creating new packages and interfaces, which allows them to run parallel to the existing logic.

Figure 1–2 Preconfigured logic and customizations



1.2.2 Typical Steps to Extend Mappings in the Oracle Business Analytics Warehouse

The most common reason for extending the Oracle Business Analytics Warehouse is to extract existing columns from a source system and map them to an existing Oracle Business Analytics Warehouse table (either fact or dimension). This type of change typically requires you to extend the interfaces within a SIL package. If the data is coming from a packaged source, then you will also need to extend the interfaces within an appropriate SDE adapter package. If the data is coming from a non-packaged source, then you must use a Universal adapter package. If an appropriate package does not already exist, you will need to create a Universal adapter package with interfaces.

To extend an ODI package in the Oracle Business Analytics Warehouse:

1. Create new SDE and SIL Adaptor folders (do not copy existing Adaptor folder as this will copy all subfolders). Rename folders to include 'Custom' or some other useful identifier in the name, and set Release Tag to match that of the existing Adaptor folder. Do this for both the SDE and SIL folders.

   a. Right-click the Mappings folder and select New Sub-Folder.

   b. Set Name as CUSTOM _<*Original Folder Name*>. For example, CUSTOM_SDE_ORA11510_Adaptor, CUSTOM_SILOS represent custom SDE and SIL folders.

   c. Click the Connect Navigator button in the Designer tab.

   d. Select Edit Release Tags.

   e. Select the release tag that corresponds to your source. For example, EBS_11_5_10.

   f. Select the custom SDE folder you created and add it to the release tag.

   g. Click Next.

   h. Click Finish.

   i. Repeat the above steps for the CUSTOM_SILOS folder, associating it with the BIA_11 Release Tag.

2. Enable versioning for the preconfigured Task Folder to be customized. The version comment should indicate this is the base version of the task. Subsequent patches applied to this task in the future would require increasing the version in the comment so that it can be compared to the original task to identify any changes.

   a. Right-click the Task folder and select Version > Create Version.

   b. Accept the default version number, 1.0.0.0.

   c. Add a description indicating that this is the original version of this task.

3. Duplicate the Task folder to be customized by copying it. Cut and paste the copied task folder to the Custom adaptor, and rename it to remove the 'Copy of…' prefix.

4. Using the same method as in step 2, enable versioning of copied Task folder. The version comment should indicate this is the original version. This versioning enables comparison of the customized task to a copy of the original version to determine all changes that have been introduced.

5. Create another version of the copied task. The version comment should indicate this is the customized version. Use the same steps as above.

6. Version the Model that the datastore to be customized exists in, for example, Oracle BI Applications. Submodels and datastores cannot be versioned. The version comment should indicate this is the base or original version.

7. Create a new version of the model, with a version comment indicating that this is where customizations are introduced. The models can now be compared to show differences. If the model ever needs to be patched, the model should be versioned again so that the patched version can be compared to the custom and original version.

8. Apply customizations to the datastore and task. Customizations should be additive as much as possible rather than overwriting existing content. For example, if you don't like the way a particular column is calculated, add a new custom column and map it in the way you prefer. In the RPD, have the logical column point to this new custom column rather than the original column.
9. Prior to generating scenarios, ensure the 'Scenario Naming Convention' User Parameter has a value of %FOLDER_NAME(2)%_%OBJECT_NAME%
10. Generate scenarios for any new Adaptors, using the option to generate the scenario as if all underlying objects are materialized. The scenario will be generated reflecting the custom adaptor name. In the future if you make changes to any of the interfaces or the package, you can either regenerate the existing scenario or generate a new scenario. Unless you need separate scenarios, it is recommended that you regenerate the existing scenario. To do this, right-click the scenario and select Regenerate.
11. Generate the Load Plan.
12. Update Load Plan steps in the generated Load Plan to reference the custom scenario.
13. Execute the Load Plan.

1.2.3 Other Types of Customizations Requiring Special Handling

This section contains the following topics:

- Section 1.2.3.1, "How to Modify Category 2 SCD Behavior"
- Section 1.2.3.2, "How to Add A Dimension to an Existing Fact"
- Section 1.2.3.3, "How to Add a DATE_WID column to a Fact"

1.2.3.1 How to Modify Category 2 SCD Behavior

The BI Applications ETL process supports Type I and Type II slowly changing dimension behavior. Some dimensions are enabled only for Type I behavior while other dimensions are enabled to also support Type II behavior. Of those dimensions that support Type II behavior, different dimension attributes have different Slowly Changing behavior including some attributes being treated as Type I.

To enable or disable Type II behavior associated with a dimension:

Note: Modifying the Type-II tracking logic is the only change that you should make to shipped logic.

To modify a Category 2 SCD Trigger:

1. In ODI Designer, modify the dimension datastore.
   a. In the Models view, expand the 'Oracle BI Applications' folder, Oracle BI Applications (Model), and Dimension (Submodel).
   b. Double-click the Dimension table.
   c. In the Definition tab, change the OLAP type value to either Dimension (only supports Type I changes) or Slowly Changing Dimension (supports Type II changes).
2. Modify the SIL Dimension Task.
   a. Navigate to the SIL task that populates this dimension.
   b. Double-click the 'Main' interface.

c. In the Flow subtab, select the 'Target (ORACLE_BI_APPLICATIONS) window.

d. If the Property Window is not visible, open it by clicking the Menu Options View – Property Inspector.

e. Change the IKM Selector value to 'IKM BIAPPS Oracle Slowly Changing Dimension' if enabling Type II behavior or 'IKM BIAPPS Oracle Incremental Update' if removing Type II behavior.

f. Regenerate the scenario.

The following describes how to modify which columns are treated as Type I or Type II in a dimension that is configured to support Type II behavior. If a dimension is configured to support only Type I behavior, the following changes will have no effect as all columns are treated as Type I.

To enable or disable Type II behavior associated with a dimension:

1. In ODI Designer, modify the dimension datastore. In the Models view, expand the 'Oracle BI Applications' folder, Oracle BI Applications (Model), Dimension (Submodel), and Columns.
2. Double-click the column whose SCD behavior you want to change.
3. In the Description subtab's 'Slowly Changing Dimensions Behavior' drop-down list, select the column behavior. To implement Type I behavior, select Overwrite on Change. To implement Type II behavior, select Add Row on Change.

If enabling Type II behavior for a custom dimension, be sure to set columns as follows:

- ROW_WID - Surrogate Key
- INTEGRATION_ID, DATASOURCE_NUM_ID - Natural Key
- CURRENT_FLG - Current Record Flag
- EFFECTIVE_FROM_DT - Starting Timestamp
- EFFECTIVE_TO_DT - Ending Timestamp

1.2.3.2 How to Add A Dimension to an Existing Fact

This section explains how to add a dimension to an existing fact, adding a dimension and dimension staging datastores as well as associated SDE and SIL processes, which also requires extending the fact and fact staging tables to reflect the association with the new dimension. This section includes the following topics:

- Section 1.2.3.2.1, "Create a Custom Dimension Datastore and Tasks"
- Section 1.2.3.2.2, "Customize Fact Datastores and Tasks"

1.2.3.2.1 Create a Custom Dimension Datastore and Tasks

Create the custom dimension datastores and tasks. Create a WC_<*dimension name*>_D datastore under the 'Oracle BI Applications – Dimension' model. Create a WC_<*dimension name*>_DS datastore under the 'Oracle BI Applications – Dimension Stage' model. Use the WC_SAMPLE_DS and WC_SAMPLE_D datastores as templates. These datastores include all required system columns. Custom tables should follow the WC_ naming convention to help distinguish from shipped tables.

Note: The specific submodel that a table belongs to drives the table maintenance behavior. For example, tables in the 'Dimension Stage' submodel will always be truncated at each ETL run while tables in the 'Dimension' submodel are truncated only during a Full ETL run. Do not create a 'Custom' submodel to place your datastores as table maintenance will not be implemented properly for tables in such a submodel.

As described below, a dimension can be defined either in ODI, generating DDL to create the table in the database, or by defining the table in the database and importing the definition into ODI using the BI Applications RKM. If you use the RKM, the imported table is automatically placed in the 'Other' submodel and needs to be moved into the 'Dimension Staging' and 'Dimension' submodels as appropriate. Also, the OLAP type will need to be set for the dimension to 'Dimension' or 'Slowly Changing Dimension' as appropriate.

Manually Create the Dimension Tables in ODI

To create the dimension and tasks manually using ODI:

1. In Designer, navigate to Models > Oracle BI Applications (Folder) > Oracle BI Applications (Model) > Dimension Stage (Submodel).
2. Right-click the WC_SAMPLE_DS datastore and select Duplicate Selection.
3. Double-click the new datastore and rename it. Name and Resource Name should match the actual table name. Alias can be the same or a more user friendly value.
4. In the Columns subtab, add all columns.
5. Repeat the same steps to create the Dimension Table by copying the WC_SAMPLE_D datastore under the Dimensions submodel.
6. For the dimension table, set the OLAP type to either Dimension if this is a Type I dimension or to Slowly Changing Dimension if this is a Type II dimension.

Import Custom Dimension Tables into ODI

To import custom dimension tables into ODI:

1. In Designer, navigate to Models > Oracle BI Applications (Folder) and double-click the Oracle BI Applications Model.
2. In the Reverse Engineer subtab, indicate the tables to be imported under the LIST_OF_TABLES option. To import multiple tables, provide a comma-separated list.
3. Click the Reverse Engineer button to start a session that imports the table(s) into ODI
4. The Reverse Engineer process places all tables in the Other submodel. Drag and drop W_%_DS tables into the Dimension Stage submodel and the W_%_D table into the Dimension submodel.
5. Double-click the new dimension datastore and set the OLAP type to either Dimension if this is a Type I dimension or to Slowly Changing Dimension if this is a Type II dimension.

Create an ODI Sequence for the Custom Dimension

Create an ODI sequence for the custom dimension. A database sequence is used to populate the ROW_WID column of the dimension. The Generate DDL procedure is used to generate the DDL required to create the database trigger in the database. Use WC_SAMPLE_D_SEQ as a template.

1. In Designer, navigate to Projects > BI Apps Project > Sequences.
2. Right-click the Sequence folder and select New Sequence.
3. Set name to <*Dimension Name*>_SEQ.
4. Select the Native sequence radio button.
5. Set the Schema to DW_BIAPPS11G.
6. Generally, the Native sequence name should match the ODI name unless this causes the name length to exceed 30 characters, in which case, you can shorten the name to meet this limit. This is the name of the database trigger created to populate the ROW_WID column.
7. Generate the DDL to create the table in the database. **Note:** If you manually created the dimension in ODI, this will generate the DDL to create both the table and sequence. If you imported the dimension into ODI, this will generate the DDL to create the sequence only.

Create SDE and SIL Tasks

Create SDE and SIL tasks in the Custom SDE and SIL adaptor folders. Use the SDE_<*Product Line Code*>_SampleDimension and SIL_SampleDimension tasks as a template. These tasks include the logic required to populate the system columns. Finally, generate scenarios for these tasks.

Add the Load Plan Step

Add Load Plan step to the '3 SDE Dims X_CUSTOM_DIM <*Product Line Version Code*>' Load Plan Component.

1. In Designer, navigate to Load Plans and Scenarios > BIAPPS Load Plan > Load Plan Dev Components > SDE - <*Product Line Version Code*> and double-click the '3 SDE Dims X_CUSTOM_DIM <*Product Line Version Code*>' Load Plan Component.
2. In the Steps subtab, select the 'X_CUSTOM_DIM' step.
3. Click the green '+' symbol near the top right and select Run Scenario Step.
4. Provide the Scenario Name, set the Version as -1, and set the Step Name to match the Task name . Set the Restart Type to 'Restart from failed step.'

1.2.3.2.2 Customize Fact Datastores and Tasks

The Fact related datastores and tasks must be extended to reflect the new dimension. Both the W_<*Fact Name*>_FS and W_<*Fact Name*>_F datastores must be extended.

To customize fact datastores and tasks:

1. Extend the Fact Staging datastore by adding an ID column that follows the naming convention X_<*name*>_ID and datatype VARCHAR2(80).
   a. The Oracle BI Applications Model should already be versioned.
   b. Navigate to Models > Oracle BI Applications (Folder) > Oracle BI Applications (Model) > Fact Stage (Submodel) and double-click the Fact Staging Table.
   c. In the Columns subtab, select the 'X_CUSTOM' column.
   d. Click the green '+' symbol to add a column below the X_CUSTOM column.

2. Extend the Fact datastore by adding a WID column that follows the naming convention X_<*name*>_WID and datatype NUMBER(10). Follow the same steps as above to add a column to the fact datastore.
3. Add a foreign key constraint to the fact table that refers to the custom dimension table created previously. The foreign key constraint ensures the Custom SIL task is included in the generated load plan. The Custom SDE task is included in the generated load plan because it populates the staging table that is used as a source for the custom SIL task.
   a. Drill into the Fact datastore.
   b. Right-click the Constraints subfolder below the Fact datastore and select New Reference.
   c. The naming convention is FK_<*Fact Table*>_<*Dimension Table*>. If there are multiple WID columns that need to reference the same dimension table, enumerate each with a numeric suffix, for example, FK_WC_CUSTOM_F_WC_CUSTOM_D1. Type must be 'User Reference'.
   d. Select the Custom Dimension from the Table drop-down list.
   e. In the Columns subtab, click the green '+' symbol to add a new column.
   f. For the Foreign Table column, select the custom WID column in the fact table. For the Primary Table column, select the ROW_WID column in the dimension table.
4. Add a non-unique bitmap index on the X_<*name*>_WID column.
   a. Drill into the Fact datastore.
   b. Right-click the Constraints subfolder below the Fact datastore and select New Key.
   c. The naming convention is <*Fact Table*>_F<*n*>. Enumerate each of these indexes with a numeric suffix, for example, WC_CUSTOM_F1.
   d. Select the Not Unique Index radio button.
   e. In the Columns subtab, add the WID column using the shuttle button.
   f. In the Control subtab, check the Defined in the Database and Active check boxes.
   g. In the Flexfields subtab, set the index type value to QUERY and the bitmap index value to Y.
5. Modify the Fact SDE task. Pass the value from the source table to the custom X_<*name*>_ID column in the staging table. In the mapping expression, include any necessary conversion functions to get the data into the VARCHAR2(80) format.
6. Modify the Fact SIL task. Add logic to retrieve the ROW_WID value from the custom dimension. This is usually done in one of the following ways. There is no significant difference between these two methods:
   a. Add the dimension as a source in the SQ temp interface. Join on the fact table's ID column and the dimension table's INTEGRATION_ID column and the fact and dimension DATASOURCE_NUM_ID columns. If the dimension is a Type II dimension, include a range join on the fact's canonical date between the dimension's effective dates. Configure the join as a Left Outer Join. Pass the ROW_WID column as an output.
   b. Add the dimension as a lookup in the main interface. The Lookup join is on the fact table's ID column and the dimension table's INTEGRATION_ID

column and the fact and dimension DATASOURCE_NUM_ID columns. If the dimension is a Type II dimension, include a range join on the fact's canonical date between the dimension's effective dates. Configure the Lookup Type as 'SQL left-outer join in the from clause'.

7. In the mapping expression to populate the custom WID column in the main interface, embed the ROW_WID column from the dimension table in a function that defaults NULL values to 0. For example, `COALESCE(SQ_W_AP_HOLDS_FS.PURCHASE_ORG_WID,0)`

1.2.3.3 How to Add a DATE_WID column to a Fact

This use case is similar to adding a regular Dimension to a fact but in this case, a Date dimension is used. There are several Date related dimension, each representing dates in a different manner (fiscal, enterprise, and so on) and different granularities (day, week, month, etc.).

Joins between a fact and Date dimension table are performed on a Date specific WID column. The Date WID column is a 'smart key' value that represents the date in YYYYMMDD format. There is no need to do a lookup to resolve the ROW_WID of the Date dimension, rather you pass the Date column through the ETL process and convert it to this format.

Each fact table has exactly one 'canonical' Date specific WID column. This is the primary date used to drive various date-related calculations. There is no particular metadata to identify this column but lookups to effective dated tables will use this column in the ETL and various date-related expressions in the RPD will also use this column. All packaged fact tables have a single canonical date already identified. When creating custom fact tables, one Date WID column should be nominated as the canonical date and consistently used.

Follow the same steps as adding a dimension to a fact with the following changes. There is no need to create a custom SDE as we use the existing Date dimension.

Customize Fact Datastores and Tasks

The Fact related datastores and tasks must be extended to reflect the new dimensionality. Both the W_<*Fact Name*>_FS and W_<*Fact Name*>_F datastores must be extended.

1. Extend the Fact Staging datastore by adding a DT column that follows the naming convention X_<*name*>_DT. This column should have the format DATE(7).
2. Extend the Fact datastore by adding both custom DT and DT_WID columns. These follow the naming convention X_<*name*>_DT and X_<*name*>_DT_WID.
3. Add a foreign key constraint to the Date dimension or dimensions. If there are multiple WID columns that need to reference the same date dimension table, enumerate each with a numeric suffix.
4. Modify the Fact SDE task. Pass the value from the source table to the custom X_<*name*>_DT column in the staging table. Apply any conversions required to get the data into DATE format.
5. Modify the Fact SIL task. Pass the X_<*name*>_DT value from the staging table to the corresponding column in the fact table. In the mapping expression to populate the custom X_<*name*>_DT_WID column in the main interface, embed the DT column in a function that calculates the DT_WID value, defaulting to 0 when the supplied DT value is NULL. For example, `CALCULATE_DT_WID_DFLT(SQ_W_AP_HOLDS_FS.HOLD_DATE,0)`

1.3 Category 2 Customizations: Adding Additional Tables

Category 2 customizations use pre-packaged adapters to add new fact or dimension tables to the Oracle Business Analytics Warehouse.

This section contains the following topics:

- Section 1.3.1, "About Creating New Tables"
- Section 1.3.2, "About the DATASOURCE_NUM_ID Column"
- Section 1.3.3, "Additional Information About Customizing"
- Section 1.3.4, "Adding a New Fact Table to the Oracle Business Analytics Warehouse"

1.3.1 About Creating New Tables

This section relates to building entirely new tables that will be loaded with data from a source table that is not already extracted from. For example, you might want to create a new Project dimension table. In this case, you create new dimension and staging tables as well as new extract and load ETL mappings.

When creating a new custom table, use the prefix WC_ to help distinguish custom tables from tables provided by Oracle as well as to avoid naming conflicts in case Oracle later releases a table with a similar name. For example, for your Project dimension you might create a WC_PROJECT_DS and a WC_PROJECT_D table.

When you create a new dimension or fact table, use the required system columns that are part of each of the Oracle Business Analytics Warehouse tables to maintain consistency and enable you to reference existing table structures. When you create a new table, you need to define the table and indices in ODI Designer Models area first. The destination model for the Oracle Business Analytics Warehouse is 'Oracle BI Applications'.

1.3.1.1 About the Main Required Columns

For custom staging tables, the following columns are required:

- **INTEGRATION_ID.** Stores the primary key or the unique identifier of a record as in the source table.
- **DATASOURCE_NUM_ID.** Stores the data source from which the data is extracted.

For dimension and fact tables, the required columns are the INTEGRATION_ID and DATASOURCE_NUM_ID columns as well as the following:

- **ROW_WID.** A sequence number generated during the ETL process, which is used as a unique identifier for the Oracle Business Analytics Warehouse.
- **ETL_PROC_WID.** Stores the ID of the ETL process information.

1.3.2 About the DATASOURCE_NUM_ID Column

The tables in the Oracle Business Analytics Warehouse schema have DATASOURCE_NUM_ID as part of their unique user key. While the transactional application normally ensures that a primary key is unique, it is possible that a primary key is duplicated between transactional systems. To avoid problems when loading this data into the data warehouse, uniqueness is ensured by including the DATASOURCE_NUM_ID as part of the user key. This means that the rows can be loaded in the same data warehouse tables from different sources if this column is given a different value for each data source.

1.3.3 Additional Information About Customizing

This section contains additional miscellaneous information about customization in Oracle Business Intelligence Applications

1.3.3.1 About the Update Strategy

For loading new fact and dimension tables, design a custom process on the source side to detect the new and modified records. The SDE process should be designed to load only the changed data (new and modified). If the data is loaded without the incremental process, the data that was previously loaded will be erroneously updated again. For example, the logic in the preconfigured SIL mappings looks up the destination tables based on the INTEGRATION_ID and DATASOURCE_NUM_ID and returns the ROW_WID if the combination exists, in which case it updates the record. If the lookup returns null, it inserts the record instead. In some cases, last update date(s) stored in target tables are also compared in addition to the columns specified above to determine insert or update. Look at the similar mappings in the preconfigured folder for more details.

1.3.3.2 About Indices and Naming Conventions

Staging tables typically do not require any indices. Use care to determine if indices are required on staging tables. Create indices on all the columns that the ETL will use for dimensions and facts (for example, ROW_WIDs of Dimensions and Facts, INTEGRATION_ID and DATASOURCE_NUM_ID and flags). Carefully consider which columns or combination of columns filter conditions should exist, and define indices to improve query performance. Inspect the preconfigured objects for guidance. Name all the newly created tables as WC_. This helps visually isolate the new tables from the preconfigured tables. Keep good documentation of the customizations done; this helps when upgrading your data warehouse. Once the indices are decided upon, they should be registered in the ODI Model (for more information, see Section 1.5.2, "How to add an index to an existing fact or dimension table").

1.3.4 Adding a New Fact Table to the Oracle Business Analytics Warehouse

Custom tables should follow the WC_ naming convention to help distinguish from preconfigured tables. Follow this procedure to add a new fact table to the Oracle Business Analytics Warehouse.

To add a new fact table:

1. Create the custom fact datastores and tasks. Create a WC_<*fact name*>_F datastore under the 'Oracle BI Applications – Fact' model. Create a WC_<*fact name*>_FS datastore under the 'Oracle BI Applications – Fact Stage' model. Use the WC_ SAMPLE_FS and WC_SAMPLE_F datastores as templates. These datastores include all required system columns.

   Note that the specific submodel that a table belongs to drives the table maintenance behavior. For example, tables in the 'Fact Stage' submodel will always be truncated during each ETL run while tables in the 'Fact' submodel are only truncated during a Full ETL run.

   A fact can be defined in ODI either manually, by generating the DDL to create the table in the database or by defining the table in the database and importing the definition into ODI using the BI Apps RKM. If using the RKM, the imported table will automatically be placed in the 'Other' submodel and will need to be moved into the 'Fact Staging' and 'Fact' submodels as appropriate. The OLAP type also needs to be set for the fact table to 'Fact Table'.

To manually create a Fact Table:

a. In Designer, navigate to Models > Oracle BI Applications (Folder) > Oracle BI Applications (Model) > Fact Stage (Submodel), right-click the WC_SAMPLE_FS datastore and select Duplicate Selection.

b. Double-click the new datastore and rename it. Name and Resource Name should match the actual table name. Alias can be the same or a more user friendly value.

c. In the Columns subtab, add all columns.

d. Repeat the same steps to create the Fact Table by copying the WC_SAMPLE_F datastore under the 'Facts' submodel.

e. For the fact table, set the OLAP type to 'Fact Table'

f. Generate the DDL to create the table in the database.

To import Fact Tables into ODI:

a. In Designer, navigate to Models > Oracle BI Applications (Folder) and double-click the Oracle BI Applications model.

b. In the Reverse Engineer subtab, indicate the tables to be imported under the 'LIST_OF_TABLES' option. To import multiple tables, provide a comma separated list.

c. Click Reverse Engineer. A session is started that imports the table or tables into ODI.

d. The Reverse Engineer process places all tables in the 'Other' submodel. Drag and drop W_%_FS tables into the Fact Stage submodel and the W_%_F table into the Fact submodel.

e. Double-click the new fact datastore and set the OLAP type to 'Fact Table'.

f. Generate the DDL to create the table in the database.

2. Add a foreign key constraint to all dimension tables associated with this fact. The foreign key constraint ensures the Dimension SIL task is included in the generated load plan. The Dimension SDE task will be included in the generated load plan because it populates the staging table that is used as a source for the Dimension SIL task.

   a. Drill into the Fact datastore.

   b. Right-click the 'Constraints' subfolder below the Fact datastore and select New Reference. The naming convention is FK_<*Fact Table*>_<*Dimension Table*>. If there are multiple WID columns that need to reference the same dimension table, enumerate each with a numeric suffix. For example, FK_WC_CUSTOM_F_WC_CUSTOM_D1.

   c. Set the Type to 'User Reference', select the dimension from the Table drop-down list and, in the Columns subtab, click the green '+' button on the top right to add a new column.

   d. For the Foreign Table column, select the custom WID column in the fact table. For the Primary Table column, select the ROW_WID column in the dimension table.

3. Create an SDE and SIL task in the Custom SDE and SIL adaptor folders. Use the SDE_<*Product Line Code*>_SampleFact and SIL_SampleFact tasks as a template. These tasks include the logic required to populate the system columns.
4. Add Load Plan step to the '3 SDE Facts X_CUSTOM_FG <*Product Line Version Code*>' Load Plan Component.
   a. In Designer, navigate to Load Plans and Scenarios > BIAPPS Load Plan > Load Plan Dev Components.
   b. Navigate to SDE - <*Product Line Version Code*> and double-click the '3 SDE Facts X_CUSTOM_FG <*Product Line Version Code*>' Load Plan Component.
   c. Select the 'X_CUSTOM_FG' step.
   d. Click the green '+' symbol near the top right and select the 'Run Scenario Step' option.
   e. Provide the Scenario Name, Version should be -1, Step Name should match the Task name. Set the Restart Type to 'Restart from failed step.'
5. Add a Load Plan step to '3 SIL Facts X_CUSTOM_FG' Load Plan Component.
   a. In Designer, navigate to Load Plans and Scenarios > BIAPPS Load Plan > Load Plan Dev Components.
   b. Navigate to SIL and double-click the '3 SIL Facts X_CUSTOM_FG' Load Plan Component.
   c. Select the 'X_CUSTOM_FG' step.
   d. Click the green '+' symbol near the top right and select the 'Run Scenario Step' option.
   e. Provide the Scenario Name, Version should be -1, Step Name should match the Task name. Set the Restart Type to 'Restart from failed step.'

1.4 Category 3 Customizations: Adding New Data as a Whole Row into a Standard Dimension Table

Category 3 customizations use the Universal adapter to load data from sources that do not have pre-packaged adapters.

This section contains the following topics:

- Section 1.4.1, "How to Add New Data as a Whole Row Into a Standard Dimension Table"
- Section 1.4.2, "Configuring Extracts"
- Section 1.4.3, "Configuring Loads"

1.4.1 How to Add New Data as a Whole Row Into a Standard Dimension Table

Follow this procedure to add new data as a whole row into a standard dimension table in the Oracle Business Analytics Warehouse.

To add new data as a whole row into the standard dimension table:

1. Identify and understand the existing structure of staging tables. Refer to *Oracle Business Analytics Warehouse Data Model Reference* for the table structures. Non-system columns can include the null value.

2. Create a custom SDE interface to load the data into the staging table in the custom folder for this purpose. The staging table needs to be populated with incremental data (rows that have been added or changed since the last Refresh ETL process), for performance reasons.
3. Populate the INTEGRATION_ID column with the unique identifier for the record.

   The combination of INTEGRATION_ID and DATASOURCE_NUM_ID is unique. Populate the INTEGRATION_ID column with the unique identifier for the record. The combination of INTEGRATION_ID and DATASOURCE_NUM_ID is unique.
4. After the data is populated in the staging table, use the standard SIL interfaces to populate the dimension target tables.

1.4.2 Configuring Extracts

Each application has prepackaged logic to extract particular data from a particular source. This section discusses how to capture all data relevant to your reports and ad hoc queries by addressing what type of records you want and do not want to load into the Oracle Business Analytics Warehouse, and contains the following topics:

- Section 1.4.2.1, "Extracting Additional Data"
- Section 1.4.2.2, "Setting Up the Delimiter for a Source File"

1.4.2.1 Extracting Additional Data

You can configure extract mappings and Interfaces in the Oracle Business Analytics Warehouse to accommodate additional source data. For example, if your business divides customer information into separate tables based on region, then you would have to set up the extract interface to include data from these tables.

1.4.2.1.1 Extracting New Data Using an Existing Source Table Extract interfaces generally consist of source tables, expressions used in the target columns, and a staging table. If you want to extract new data using the existing interface, you have to modify the extract interface to include the new data by performing the following tasks:

To modify an existing interface to include new data:

1. Modify the existing interface to extract information from the source, and add it to an appropriate extension column.
2. Modify the Expressions in the target table to perform any necessary transformations.
3. Save the changes.
4. Regenerate the scenario.

You have to determine which type of extension column to map the data to in the staging table. After you modified the extract interface, you would also have to modify the corresponding load interfaces (SDE and SIL) to make sure that the extension columns that you added are connected all the way from the staging table to the target data warehouse table.

1.4.2.1.2 Extracting Data from a New Source Table Extract interfaces (which have the SQ_* naming convention) reside in source-specific folders within the repository. Extract interfaces are used to extract data from the source system. You can configure these extract interfaces to perform the following:

- Extract data from a new source table.

- Set incremental extraction logic.

1.4.2.2 Setting Up the Delimiter for a Source File

When you load data from a Comma Separated Values (CSV) formatted source file, if the data contains a comma character (,), you must enclose the source data with a suitable enclosing character known as a delimiter that does not exist in the source data.

Note: Alternatively, you could configure your data extraction program to enclose the data with a suitable enclosing character automatically.

For example, you might have a CSV source data file with the following data:

```
Months, Status
January, February, March, Active
April, May, June, Active
```

If you loaded this data without modification, ODI would load 'January' as the Months value, and 'February' as the Status value. The remaining data for the first record (that is, March, Active) would not be loaded.

To enable ODI to load this data correctly, you might enclose the data in the Months field within the double-quotation mark enclosing character (" ") as follows:

```
Months, Status
"January, February, March", Active
"April, May, June", Active
```

After modification, ODI would load the data correctly. In this example, for the first record ODI would load 'January, February, March' as the Months value, and 'Active' as the Status value.

To set up the delimiter for a source file:

1. Open the CSV file containing the source data.
2. Enclose the data fields with the enclosing character that you have chosen (for example, (").

   You must choose an enclosing character that is not present in the source data. Common enclosing characters include single quotation marks (') and double quotation marks (").
3. Save and close the CSV file.
4. In ODI Designer, display the Models view, and expand the Oracle BI Applications folder.

   Identify the data stores that are associated with the modified CSV files. The CSV file that you modified might be associated with one or more data stores.
5. In ODI Designer, change the properties for each of these data stores to use the enclosing character, as follows:

   a. Double-click the data source, to display the DataStore: <*Name*> dialog.

   b. Display the Files tab.

   c. Use the **Text Delimiter** field to specify the enclosing character that you used in step 2 to enclose the data.

   d. Click OK to save the changes.

   You can now load data from the modified CSV file.

1.4.3 Configuring Loads

This section explains how to customize the way that Oracle Business Intelligence Applications loads data into the Oracle Business Analytics Warehouse.

1.4.3.1 About Primary Extract and Delete Mappings Process

Before you decide to enable primary extract and delete sessions, it is important to understand their function within the Oracle Business Analytics Warehouse. Primary extract and delete mappings allow your analytics system to determine which records are removed from the source system by comparing primary extract staging tables with the most current Oracle Business Analytics Warehouse table.

The primary extract mappings perform a full extract of the primary keys from the source system. Although many rows are generated from this extract, the data only extracts the Key ID and Source ID information from the source table. The primary extract mappings load these two columns into staging tables that are marked with a *_PE suffix.

The figure below provides an example of the beginning of the extract process. It shows the sequence of events over a two day period during which the information in the source table has changed. On day one, the data is extracted from a source table and loaded into the Oracle Business Analytics Warehouse table. On day two, Sales Order number three is deleted and a new sales order is received, creating a disparity between the Sales Order information in the two tables.

Figure 1–3 Extract and load mappings

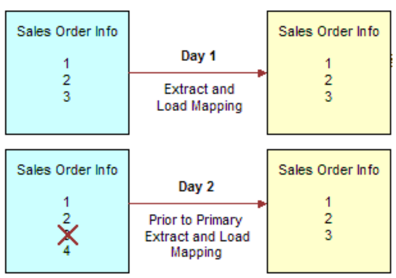
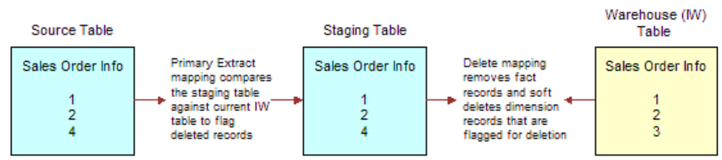


Figure 1–4 shows the primary extract and delete process that occurs when day two's information is extracted and loaded into the Oracle Business Analytics Warehouse from the source. The initial extract brings record four into the Oracle Business Analytics Warehouse. Then, using a primary extract mapping, the system extracts the Key IDs and the Source IDs from the source table and loads them into a primary extract staging table.

The extract mapping compares the keys in the primary extract staging table with the keys in the most current the Oracle Business Analytics Warehouse table. It looks for records that exist in the Oracle Business Analytics Warehouse but do not exist in the staging table (in the preceding example, record three), and sets the delete flag to Y in the Source Adapter, causing the corresponding record to be marked as deleted.

The extract mapping also looks for any new records that have been added to the source, and which do not already exist in the Oracle Business Analytics Warehouse; in this case, record four. Based on the information in the staging table, Sales Order number three is physically deleted from Oracle Business Analytics Warehouse, as shown in Figure 1–4. When the extract and load mappings run, the new sales order is added to the warehouse.

Figure 1–4 Primary Extract and Delete Mappings



1.4.3.2 About Working with Primary Extract and Delete Mappings

The primary extract (*_Primary) and delete mappings (*_IdentifyDelete and *_Softdelete) serve a critical role in identifying which records have been physically deleted from the source system. However, there are some instances when you can disable or remove the primary extract and delete mappings, such as when you want to retain records in the Oracle Business Analytics Warehouse that were removed from the source systems' database and archived in a separate database.

Because delete mappings use Source IDs and Key IDs to identify purged data, if you are using multiple source systems, you must modify the SQL Query statement to verify that the proper Source ID is used in the delete mapping. In addition to the primary extract and delete mappings, the configuration of the delete flag in the load mapping also determines how record deletion is handled.

You can manage the extraction and deletion of data in the following ways:

- Deleting the configuration for source-archived records
- Deleting records from a particular source
- Enabling delete and primary-extract sessions
- Configuring the Record Deletion flag
- Configuring the Record Reject flag

1.4.3.2.1 Deleting the Configuration for Source-Archived Records Some sources archive records in separate databases and retain only the current information in the main database. If you have enabled the delete mappings, you must reconfigure the delete mappings in the Oracle Business Analytics Warehouse to retain the archived data.

To retain source-archived records in the Oracle Business Analytics Warehouse, make sure the LAST_ARCHIVE_DATE parameter value is set properly to reflect your archive date. The delete mappings will not mark the archived records as 'deleted'. For more information about extract and delete mappings, see Section 1.4.3.2, "About Working with Primary Extract and Delete Mappings".

1.5 Customizing Stored Lookups and Adding Indexes

This section contains miscellaneous information that applies to all three categories of customization in Oracle Business Intelligence Applications, and contains the following topics:

- Section 1.5.1, "About Stored Lookups"
- Section 1.5.2, "How to add an index to an existing fact or dimension table"

1.5.1 About Stored Lookups

This section explains codes lookup and dimension keys.

1.5.1.1 About Resolving Dimension Keys

By default, dimension key resolution is performed by the Oracle Business Analytics Warehouse in the load mapping. The load interface uses prepackaged, reusable lookup transformations to provide pre-packaged dimension key resolution. This section describes how dimension keys are looked up and resolved.

There are two commonly used methods for resolving dimension keys. The first method, which is the primary method used, is to perform a lookup for the dimension key. The second method is to supply the dimension key directly into the fact load mapping.

1.5.1.1.1 Resolving the Dimension Key Using Lookup If the dimension key is not provided to the Load Interface through database joins, the load mapping performs the lookup in the dimension table. The load mapping does this using prepackaged Lookup Interfaces. To look up a dimension key, the Load Interface uses the INTEGRATION_ID, the DATASOURCE_NUM_ID, and the Lookup date, which are described in the table below.

Table 1–1 Columns Used in the load mapping Dimension Key Lookup

Port	Description
INTEGRATION ID	Uniquely identifies the dimension entity within its source system. Formed from the transaction in the Source Adapter of the fact table.
DATASOURCE_NUM_ID	Unique identifier of the source system instance.
Lookup Date	The primary date of the transaction; for example, receipt date, sales date, and so on.

If Type II slowly changing dimensions are enabled, the load mapping uses the unique effective dates for each update of the dimension records. When a dimension key is looked up, it uses the fact's primary or 'canonical' date to resolve the appropriate dimension key. The effective date range gives the effective period for the dimension record. The same entity can have multiple records in the dimension table with different effective periods due to Type II slowly changing dimensions. This effective date range is used to exactly identify a record in its dimension, representing the information in a historically accurate manner.

There are four columns needed for the load interface lookup: INTEGRATION ID, DATASOURCE_NUM_ID, and Lookup Date (EFFECTIVE_FROM_DT and EFFECTIVE_TO_DATE). The lookup outputs the ROW_WID (the dimension's primary key) to the corresponding fact table's WID column (the fact tables foreign key).

1.5.2 How to add an index to an existing fact or dimension table

Dimension and Fact Tables in the Oracle Business Analytics Warehouse use the following two types of index:

- ETL Index

  ETL Indexes are used for Unique/Binary Tree index.

- Query Index

  Query Indexes are used for Non-Unique/Bit Map Index.

To add an index to an existing fact or dimension table:

1. In ODI Designer, display the Models view, and expand the 'Oracle BI Applications' folder.
2. Expand the Fact or Dimension node as appropriate.
3. Expand the Table in which you want to create the index.
4. Right-click on the Constraints node, and select Insert Key to display the Key: New dialog.
5. Display the Description tab.
6. Select the **Alternate Key** radio button, and update the name of the Index in the **Name** field.
7. Display the Column tab.
8. Select the column on which you want to create the index.
9. Display the FlexFields tab.
10. Use the settings to specify the index type, as follows:
    - For 'Query' type indexes (the default), define the index as an 'Alternate Key' for unique indexes and as 'Not Unique Index' for non-unique indexes.
    - For 'ETL' type indexes, clear the check box for the INDEX_TYPE parameter and set the value to 'ETL'. In addition, set the value of the IS_BITMAP parameter to 'N' and define the index as an 'Alternate Key' for unique indexes and as 'Not Unique Index' for non unique indexes.
11. Save the changes.

2

About Multi-Language Support

This chapter provides information about multi-language support in Oracle BI Applications.

This chapter contains the following topics:

- Section 2.1, "Introduction to Multi-Language Support"
- Section 2.2, "About Pseudo-Translations"
- Section 2.3, "About Oracle BI Applications Domains"
- Section 2.4, "About Dimension Translation Tables"

2.1 Introduction to Multi-Language Support

Oracle BI Applications provides multi-language support for metadata level objects exposed in Oracle BI Enterprise Edition dashboards and reports, as well as for data, which enables users to see records translated in their preferred language.

Configuring Base and Installed Data Warehouse Languages

After installing Oracle BI Applications, you use the Oracle BI Applications Configuration Manager (Configuration Manager) to configure which languages you want to support in the Oracle Business Analytics Warehouse. You must configure one "Base" language, and you can also configure any number of "Installed" languages. Typically, the Base language specified for the data warehouse should match the Base language of the source system. The Installed languages that you specify for the data warehouse do not have to match the languages that are installed in the source system. The data warehouse can have more, fewer, or completely different Installed languages compared to the source system. Note that for languages that match between the transactional system and the data warehouse, the corresponding record is extracted from the transactional system; languages that do not match will have a pseudo-translated record generated.

Note: You should only install the languages that you expect to use, because each installed language can significantly increase the number of records stored in the data warehouse and can affect overall database performance.

For information about how to configure data warehouse languages, see *Oracle Fusion Middleware Configuration Guide for Oracle Business Intelligence Applications*.

Translation Tables

There are two types of translation tables: the Domains translation table and Dimension translation tables. There is a single Domain translation table which holds a translated value in each supported language for a domain. Dimension translation tables are extension tables associated with a given dimension. Depending on certain characteristics of a translatable attribute, it will be found in either the domain or a dimension translation table.

The user's session language is captured in an Oracle BI Enterprise Edition session variable named USER_LANGUAGE_CODE. This is set when users log in from Answers, where they select their preferred language. If users decide to change their preferred language in the middle of a session by using the Administration option to change the current language, this session variable will detect this change. Records returned from a translation table are filtered to those records with a LANGUAGE_CODE value that matches this session variable.

2.2 About Pseudo-Translations

The ETL process extracts translation records from the source system that correspond to the languages installed in the data warehouse. If a record cannot be found in the source system that corresponds to a language that has been installed in the data warehouse, a pseudo-translated record will be generated. Without a pseudo-translated record, a user that logs in with the missing language as their preferred language will not see any records.

A pseudo-translated record is generated by copying the translation record that corresponds to the data warehouse Base language and flagging it with the missing record's language by populating the LANGUAGE_CODE column with the language value. SRC_LANGUAGE_CODE stores the language from which the pseudo-translated record was generated; this will always match the data warehouse Base language.

In the future, if a translation record is created in the source system, it will be extracted and the pseudo-translated record will be overwritten to reflect the actual translated value. Table 2–1 provides an example in which "US" is the data warehouse Base language, and "IT" and "SP" are the Installed languages. The source system only had translated records for "US" and "IT" but did not have a translated record for "SP". The "US" and "IT" records are extracted and loaded into the data warehouse. Because there is no translation record in the source system for the "SP" language, a pseudo-translated record is generated by copying the "US" record and flagging LANGUAGE_CODE as if it were an "SP" record. The pseudo-translated record can be identified because SRC_LANGUAGE_CODE is different from LANGUAGE_CODE, matching the Base Language.

Table 2–1 Example of Pseudo-Translated Record

INTEGRATION_ID	NAME	LANGUAGE_CODE	SRC_LANGUAGE_CODE
ABC	Executive	US	US
ABC	Executive	IT	IT
ABC	Executive	SP	US

2.3 About Oracle BI Applications Domains

A domain refers to the possible, unique values of a table column in a relational database. In transactional systems, domains are often referred to as list of values (LOVs), which present attribute selections in the user's session language. The storage of the transaction is independent of the user's language; and, therefore, the field is stored using a language independent identifier. This identifier is typically a character code but can also be a numeric ID. The LOV or domain is then based on an ID-value pair, referred to as a member, and the LOV presents the values in the user's session language. At run time, the IDs are resolved to the value for the user's session language.

In the Oracle Business Analytics Warehouse, the number of unique values in any particular domain is relatively small and can have a low cardinality relative to the dimension it is associated with. For example, the Person dimension may have the domain 'Gender' associated with. The dimension may have millions of records, but the domain will generally have two or three members (M, F and possibly U). In the Oracle Business Analytics Warehouse, the Gender Code is stored in the Person dimension which acts as a foreign key to the Domains Translation table which stores the translated values. When a query is run, the user-friendly text associated with the code value is returned in the user's session language.

Depending on certain properties associated with a domain, domains can be configured in the Configuration Manager. In addition to serving as a mechanism for supporting translations, domains can be used to conform disparate source data into a common set of data.

Data Model

Oracle BI Applications domains are associated with dimensions as fields in the dimension table that follow the %_CODE naming convention. For example, the Person dimension W_PARTY_PER_D would store the Gender domain in the GENDER_CODE column.

Oracle BI Applications domains are stored in the domain translation table W_DOMAIN_MEMBER_LKP_TL. This table stores the translated values for each domain member code. Translated values are usually either a Name or a Description value which are stored in the NAME and DESCR columns of this table. The DOMAIN_MEMBER_CODE column acts as a key column when joining with the %_CODE column in the dimension table. As domains come from various systems, a DATASOURCE_NUM_ID column is used to identify which system the translated value comes from and is used as part of the join key with the dimension table. A LANGUAGE_CODE column is used to identify the language the translated values are associated with. Note that the LANGUAGE_CODE column follows the %_CODE naming convention. Language is considered a domain with a given set of unique values.

ETL Process

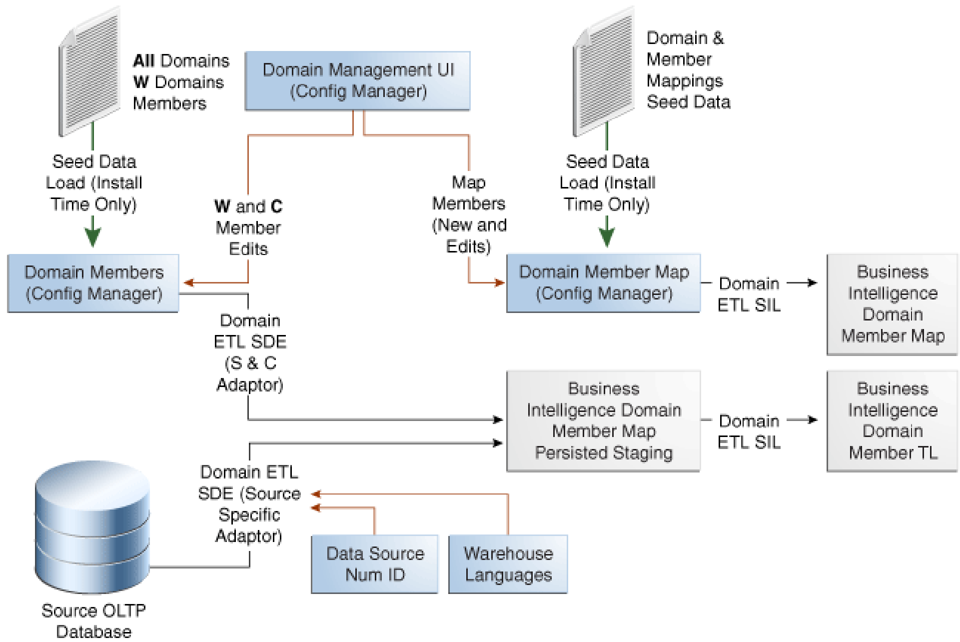
The W_DOMAIN_MEMBER_LKP_TL table stores both domains that are extracted from the source system as well as internally defined domains that are seeded in the Configuration Manager. For each of the %_CODE columns that have translated values available in the source system, an ETL process extracts the domain members from the transactional system and loads them into W_DOMAIN_MEMBER_LKP_TL. Internally defined domains—usually domains specific to the Oracle Business Analytics Warehouse and known as conformed domains but can also include source domains—are stored in the Configuration Manager schema and are similarly extracted and loaded into the W_DOMAIN_MEMBER_LKP_TL table through ETL processes.

Only those translation records that match one of the languages that have been installed in the data warehouse are extracted from the transactional system. If translated records are not found in the transactional system matching an installed language, the ETL will generate a 'pseudo-translated' record for that language.

Some source applications store translations that can be extracted and loaded into the translation table. Some source applications do not maintain translations for an entity that corresponds to a dimension table. In these cases, whatever record is available is extracted and used as the Base language record to generate pseudo-translations for all other installed languages.

Figure 2–1 shows an overview of the Oracle BI Applications domain ETL process.

Figure 2–1 Overview of BI Applications Domain ETL Process



About Oracle BI Applications Domains and Oracle BI Enterprise Edition

The exact mechanism used to retrieve the translated value in Oracle BI Enterprise Edition is the LOOKUP() function. When the LOOKUP() function is used, Oracle BI Enterprise Edition performs all aggregations before joining to the lookup table. The aggregated result set is then joined to the lookup table. Low-cardinality attributes tend to be involved in several aggregations, so it is useful to be joined after results are aggregated rather than before.

In a logical dimension, a Name or Description attribute will use the LOOKUP() function, passing the value in the %_CODE column associated with that Name or Description to the Domain Lookup Table. The LOOKUP() function includes the Domain Name to be used when looking up values. The results from the Domain Lookup table are filtered to match the user's session language and returned as part of the query results.

Domains can be either source or conformed (internally defined warehouse domains). Source domains can come from a variety of transactional systems and so must include a Datasource_Num_Id value to resolve. Conformed domains are defined as part of the Oracle BI Applications and do not require a Datasource_Num_ID to resolve. As a result, there are two lookup tables implemented in the Oracle BI Repository that are aliases of W_DOMAIN_MEMBER_LKP_TL. When resolving a source domain, the source domain lookup requires Datasource_Num_Id to be passed as part of the LOOKUP() function while the conformed domain lookup does not.

2.4 About Dimension Translation Tables

As mentioned in Section 2.3, "About Oracle BI Applications Domains,", domains are dimensional attributes that have a relatively small number of distinct members, have a low cardinality relative to the number of records in the dimension, and are often used in aggregations. Dimensions have other attributes that require translation that may not fit one or more of these criteria. Dimensions may have translatable attributes that have a high cardinality relative to the dimension or may have a large number of members, and, thus, are not likely candidates for aggregation. If the domains ETL process was implemented in such cases, performance would be very poor. As a result, these particular attributes are implemented using dimension translation tables.

Data Model

If a dimension has such high-cardinality attributes that cannot be treated as domains, the dimension will have an extension table that follows the _TL naming convention. If the _TL table has a one-to-one relationship with the dimension table (after filtering for languages), the _TL table name will match the dimension table name. For example, W_JOB_D_TL is the translation table associated with the W_JOB_D dimension table. If the _TL table does not have a one-to-one relationship with any dimension table, its name will reflect content.

The dimension and dimension translation table are joined on the translation table's INTEGRATION_ID + DATASOURCE_NUM_ID. If the translation and dimension tables have a one-to-one relationship (after filtering for language), the join to the dimension table is on its INTEGRATION_ID + DATASOURCE_NUM_ID. Otherwise, there will be a %_ID column in the dimension table that is used to join to the translation table.

ETL Process

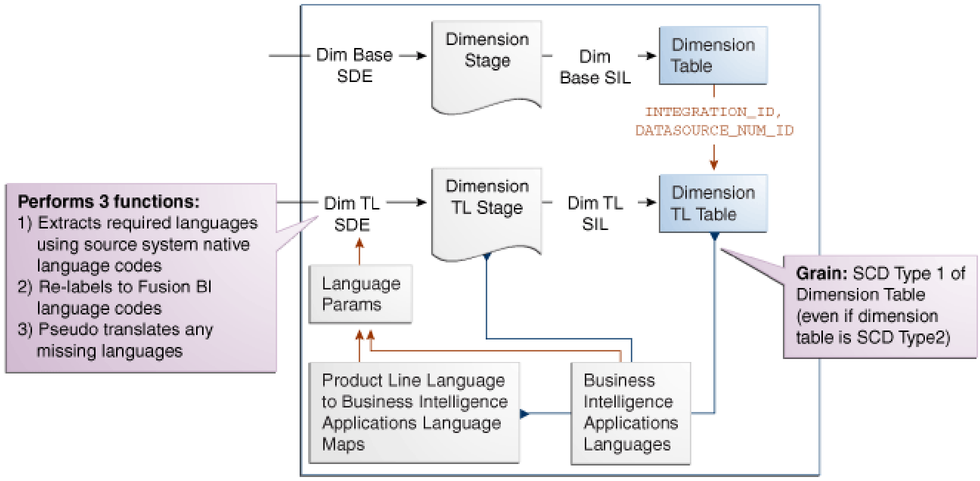
Similar to the Oracle BI Applications domain ETL process, when using dimension translation tables, ETL tasks extract the translated values from the transactional system. Rather than the domain staging table being loaded, the dimension's translation staging table is loaded. The ETL process then moves these records into the dimension translation table.

Only those translation records that match one of the languages that have been installed in the data warehouse are extracted from the transactional system. If translated records are not found in the transactional system matching a data warehouse Installed language, the ETL will generate a 'pseudo-translated' record for that language by copying the record that corresponds to the data warehouse Base language.

Some source applications store translations that can be extracted and loaded into the translation table. Some source applications do not maintain translations for an entity that corresponds to a dimension table. In these cases, whatever record is available is extracted and used as the Base language record, which is then used to generate pseudo-translations for all other Installed languages.

Oracle BI Applications does not support Type 2 SCD tracking of dimension translation attributes when the dimension and translation tables have a one-to-one relationship with each other. These tables are joined on INTEGRATION_ID + DATASOURCE_NUM_ID, and, therefore, can be joined to a single record in the translation table. Attributes in the dimension table can be Type 2-enabled, but the current and prior records will always have the same translated value. Figure 2–2 describes the ETL domain process.

Figure 2–2 Domain ETL Process



Oracle BI Enterprise Edition

In Oracle BI Enterprise Edition, joins are created between the dimension and translation tables as normal. The translation table is brought in as another supporting table in the logical table source. If a user selects an attribute from the translation table, it will be included as a joined table in the SQL that Oracle BI Enterprise Edition generates. If the user does not select a translation attribute, the translation table will not be included in the generated SQL.

To ensure this behavior, the physical join between the dimension and translation tables is configured as one-to-many with the dimension table on the many side.

An important consideration is filtering on a user's language. If the language filter is included in the logical table source as a content filter, the translation table will always be joined whether a user selects a translation attribute or not. To avoid this behavior, opaque views are created in the physical layer that include a WHERE clause on the user's session language. Filtering on the user's language is still possible, but as the filter criteria is not implemented as a logical table source content filter, it is ensured that the translation table is only joined when necessary.

Localizing New Domain Members and Oracle BI Repository Metadata

If you added new domain members that require localization, see the section titled "How to Localize a New Domain Member," in *Oracle Fusion Middleware Configuration Guide for Oracle Business Intelligence Applications*.

Also, to add string localizations in the Oracle BI Repository metadata, see "How to Add String Localizations for Oracle BI Repository Metadata," in *Oracle Fusion Middleware Configuration Guide for Oracle Business Intelligence Applications*.

3

Localizing Oracle Business Intelligence Deployments

This chapter describes concepts and techniques for localizing Oracle Business Intelligence Applications. Oracle Business Intelligence is designed to allow users to dynamically change their preferred language and locale preferences. This section explains how to configure Oracle BI Applications for deployment in one or more language environments other than English.

This chapter contains the following topics:

- Section 3.1, "Process of Maintaining Translation Tables for Oracle BI EE"
- Section 3.2, "About Translating Presentation Services Strings"
- Section 3.3, "Changing the Default Currency in Analytics Applications"

3.1 Process of Maintaining Translation Tables for Oracle BI EE

The Oracle Business Intelligence Presentation layer supports multiple translations for any column name. When working with Oracle BI Answers or rendering a dashboard, users see their local language strings in their reports. For example, English-speaking and French-speaking users would see their local language strings in their reports. There are two kinds of application strings requiring translation in Oracle Business Intelligence:

- Metadata

  Metadata strings are analytics-created objects in the Oracle Business Intelligence repository such as subject areas, metrics, and dimensions.

- Presentation Services

  Presentation Services objects are end-user created objects such as reports, dashboards, and pages. Translations for Presentation Services strings are stored in the XML caption files. For more information on accessing these strings and changing the translations, see *Oracle Business Intelligence Presentation Services Administration Guide*.

This process includes the following tasks:

- Section 3.1.1, "Upgrading Oracle Business Intelligence Seed Data for Non-English Locales"
- Section 3.1.2, "Externalizing Customer Metadata Strings"
- Section 3.1.3, "Adding Custom Translations to the W_LOCALIZED_STRING_G Table"

3.1.1 Upgrading Oracle Business Intelligence Seed Data for Non-English Locales

If Oracle Business Intelligence data in your deployment is to be viewed in a language other than English, you must do the following:

1. Verify creation of the Translation Table (W_LOCALIZED_STRING_G) and corresponding indexes, as described in Section 3.1.1.1, "Verify the Translation Table (W_LOCALIZED_STRING_G) and Corresponding Indexes".
2. Import Locale seed data into the W_LOCALIZED_STRING_G table, as described in Section 3.1.1.2, "Importing Locale Seed Data Into The Translation Table (W_LOCALIZED_STRING_G)".
3. Create an Initialization Block at the Session level to set the LOCALE variable.

   For example, you might do the following:

   a. In Oracle BI EE Administration Tool, choose Manage, then Variables, to open the Variable Manager dialog.
   b. From the Action menu, choose New, then Session, then Initialization Block.
   c. In the Session Variable Initialization Block dialog, type a name for the block. For example, LOCAL_INIT_BLOCK.
   d. Click the Edit data source button.
   e. In the Default initialization string box, type the SQL initialization string. For example:

      select 'VALUEOF(NQ_SESSION.WEBLANGUAGE)' from VALUEOF(OLAPTBO).DUAL
   f. Click Browse next to the Connection Pool field to select an appropriate connection pool. For example, "Oracle EBS OLTP"."Oracle EBS OLTP InitBlocks Connection Pool".
   g. In the Variable Manager dialog, navigate to Session > Variables > Non-System.
   h. Double click the LOCAL variable to open the Session Variable dialog.
   i. In the Session Variable dialog, use the Initialization Block list to select the new initialization block, for example, LOCAL_INIT_BLOCK.

3.1.1.1 Verify the Translation Table (W_LOCALIZED_STRING_G) and Corresponding Indexes

To verify the Translation Table (W_LOCALIZED_STRING_G) and corresponding indexes:

1. Verify that Oracle Business Analytics Warehouse contains the W_LOCALIZED_STRING_G table.
2. Lookup the definitions of the following indexes in DAC and create them manually in Oracle Business Analytics Warehouse:

   – W_LOCAL_STRING_G_U1
   – W_LOCAL_STRING_G_P1
   – W_LOCAL_STRING_G_M1
   – W_LOCAL_STRING_G_M2

Note: It is better to add these indexes to W_LOCALIZED_STRING_G prior to importing the locale seed data in the next section, in order to safeguard against inadvertently duplicating the data in the table.

3.1.1.2 Importing Locale Seed Data Into The Translation Table (W_LOCALIZED_STRING_G)

If the primary language being used is not English, you might have to import additional locale seed data (depending on the number of languages you use). This process must be performed once for each language in which users might use in their Web client.

Notes

- This task should be performed only by a BI Administrator.
- To perform task, you need the dataimp utility, which can only be used on 32-bit operating systems.
- During the Oracle Business Intelligence Applications installation, a directory named `ORACLE_HOME\biapps\seeddata` is created, which contains a sub directory for each language. Within each language sub directory is a .dat file (the data to be imported) and an .inp file (the WHERE clause governing the import).

To import Locale seed data into the Translation Table (W_LOCALIZED_STRING_G)

1. Open a command window and navigate to ORACLE_HOME\biapps\seeddata\bin directory.
2. Run the import command in step 3 after replacing these connection parameters with the values appropriate to your database environment:
   - UserName
   - Password
   - ODBCDataSource
   - DatabaseOwner
3. Run the import command:

```
ORACLE_HOME\biapps\seeddata\Bin\dataimp /u $UserName /p $Password /c
"$ODBCDataSource" /d $DatabaseOwner /f ORACLE_HOME\biapps\seeddata\l_
<XX>\analytics_seed_<XXX>.dat /w y /q 100 /h Log /x f /i ORACLE_
HOME\biapps\seeddata\l_<XX>\metadata_upgrade_<XXX>_<DBPlatform>.inp /l
metadata_upgrade_<XXX>.log
```

> **Note:** Replace the XX with the Oracle Business Intelligence two-letter language code (fr, it) and the XXX with the Siebel Systems three-letter code (FRA, ITA).

4. When you have finished importing the Locale seed data into the Translation Table (W_LOCALIZED_STRING_G), configure the initialization block in the Oracle BI Repository using the Oracle BI Administration Tool to connect to the database where this table resides.

> **Note:** Unicode connectivity can be used to access databases that do not support Unicode.

3.1.2 Externalizing Customer Metadata Strings

Metadata Strings are loaded by the Oracle BI Server from a database table. In the case of Oracle Business Intelligence applications, this table is W_LOCALIZED_STRING_G in the data warehouse. The initialization block 'Externalize Metadata Strings' loads the strings for the Server. It is recommended that you run a test to make sure that this initialization block runs successfully. An example of the translation table is shown in Table 3–1.

Table 3–1 Example of W_LOCALIZED_STRING_G Translation Table

MSG_NUM	MSG_TEXT	LANG_ID
CN_Customer_Satisfaction	Customer Satisfaction	ENU
CN_Customer_Satisfaction	Kundenzufriedenheit	DEU
CN_Customer_Satisfaction	Satisfação do cliente	PTB

By default, the Oracle Business Intelligence repository is configured to run in English only. To deploy in any other language, you must externalize the metadata strings, as described in the following procedure.

To externalize metadata strings in the Oracle Business Intelligence repository

1. Stop the Oracle BI Server.
2. Using the Oracle BI Administration Tool in offline mode, open OracleBIAnalyticsApps.rpd.
3. Select the entire Presentation layer and right-click the mouse to display the menu.
   - From the pop-up menu, select Externalize Display Names. (A check mark appears next to this option the next time you right-click on the Presentation layer.)
   - Unselect the Presentation layer.

   Note: When Externalize Display Names is checked, all metadata strings are read from the W_LOCALIZED_STRING_G table in the data warehouse.

4. In the Physical layer, select the Externalized Metadata Strings database icon. Expand the tree.
5. Double-click Internal System Connection Pool.

   In the Connection Pool dialog General tab, the field Data source name should point to the data warehouse.
6. Click OK and exit the Oracle BI Administration Tool.
7. Restart the Oracle BI Server.

3.1.3 Adding Custom Translations to the W_LOCALIZED_STRING_G Table

When you add custom objects to the metadata and choose to externalize these objects (by right-clicking the object and checking the Externalize Display Name option), the Oracle BI Server looks for the translations (including those for the native language) in the W_LOCALIZED_STRING_G table.

If you do not externalize the display names, you do not need to perform the following procedures.

> **Note:** The custom Presentation layer objects show up only in the native language of the metadata (the language in which you added these new objects).

3.1.3.1 Adding String Translations for Analytics Metadata

The following procedure describes how to add string translations for Oracle Business Intelligence metadata to the W_LOCALIZED_STRING_G table. This task occurs in any database administration tool, and in the Oracle BI Administration Tool.

To add string translations for Analytics metadata

1. Open a database administration tool and connect to your data warehouse database.
2. Query for the table named W_LOCALIZED_STRING_G and add a new record to the table, as defined below in steps 4 to 8.
3. Obtain the Message Key from the Oracle BI Administration Tool as follows:
   - In the Oracle BI Administration Tool, right-click on the new Presentation layer metadata object and select Properties from the menu.
   - The Message key is displayed in the dialog under Custom Display Name. The Message key is the part that starts with CN_.

     For example, double-click the Pipeline catalog directory in the Presentation layer. The Custom Display name is Valueof(NQ_SESSION.CN_Pipeline). CN_Pipeline is the Message Key.
4. Enter your deployment language in the new record.
5. Enter the Message Type required (for example, Metadata, FINS_Metadata).
6. Select the Message Level *AnalyticsNew*, then do the following:
   - In the Message Text column, add the translation of the object.
   - Check the flags (set to Yes) for the Translate and Active columns.
   - Set the Error Message # column to 0.
7. Enter the required Message Facility (for example, HMF, FIN).
8. Repeat Step 3 through Step 7 for each new metadata object string.
9. Exit the database administration tool, then restart the Oracle BI Server.

3.2 About Translating Presentation Services Strings

The translations for such Presentation Services objects as report and page names are stored in the xxxCaptions.xml files available in the `ORACLE_HOME\biapps\catalog\res\web\l_<Language Abbreviation>\Captions` directories. In multiple language deployment mode, if you add any additional Presentation Services objects, such as reports and new dashboard pages, you also need to add the appropriate translations. Add these translations using the Catalog Manager tool. For more information on using this utility, see *Oracle Business Intelligence Presentation Services Administration Guide*.

3.3 Changing the Default Currency in Analytics Applications

In Oracle Business Intelligence Applications, you might see a dollar sign used as the default symbol when amounts of money are displayed. In order to change this behavior, you must edit the currencies.xml file using the following procedure. The currencies.xml file is located in the following directories:

- Windows:

  ORACLE_HOME\bifoundation\web\display\currencies.xml

- UNIX:

  ORACLE_HOME/bifoundation/web/display/currencies.xml

To change the default currency in Analytics Applications

1. In a text editor, open the currencies.xml file.
2. Look for the currency tag for the warehouse default (tag="int:wrhs"):

```
<Currency tag="int:wrhs" type="international" symbol="$" format="$#" digits="2"
displayMessage="kmsgCurrencySiebelWarehouse">
   <negative tag="minus" format="-$#" />
</Currency>
```

3. Replace the symbol, format, digits and negative information in the warehouse default with the information from the currency tag you want to use as the default.

   For example, if you want the Japanese Yen to be the default, replace the contents of the warehouse default currency tag with the values from the Japanese currency tag (tag="loc:ja-JP"):

```
<Currency tag="loc:ja-JP" type="local" symbol="¥" locale="ja-JP" format="$#"
digits="0">
   <negative tag="minus" format="-$#" />
</Currency>
```

   When you are finished, the default warehouse currency tag for Japanese should look like the following example:

```
<Currency tag="int:wrhs" type="international" symbol="¥" format="$#" digits="0"
displayMessage="kmsgCurrencySiebelWarehouse">
   <negative tag="minus" format="-$#" />
</Currency>
```

4. Save and close the currencies.xml file.

4 Oracle Business Analytics Warehouse Naming Conventions

This chapter includes information on the types of tables and columns in the Oracle Business Analytics Warehouse, including the naming conventions used.

Note: This chapter contains naming conventions used for database tables and columns in the Oracle Business Analytics Warehouse. This information does not apply to objects in the Oracle Business Intelligence repository.

This chapter contains the following topics:

- Section 4.1, "Naming Conventions for Oracle Business Analytics Warehouse Tables"
- Section 4.2, "Table Types for Oracle Business Analytics Warehouse"
- Section 4.3, "Internal Tables in Oracle Business Analytics Warehouse"
- Section 4.4, "Standard Column Prefixes in Oracle Business Analytics Warehouse"
- Section 4.5, "Standard Column Suffixes in Oracle Business Analytics Warehouse"
- Section 4.6, "System Columns in Oracle Business Analytics Warehouse Tables"
- Section 4.7, "Multi-Currency Support for System Columns"
- Section 4.8, "Oracle Business Analytics Warehouse Primary Data Values"
- Section 4.8, "Oracle Business Analytics Warehouse Primary Data Values"
- Section 4.9, "About Multi-Language Support in the Oracle Business Analytics Warehouse"
- Section 4.10, "Oracle Business Analytics Warehouse Currency Preferences"

4.1 Naming Conventions for Oracle Business Analytics Warehouse Tables

Oracle Business Analytics Warehouse tables use a three-part naming convention: PREFIX_NAME_SUFFIX, as shown in Table 4–1.

Table 4–1 Naming Conventions for Oracle Business Analytics Data Warehouse Tables

Part	Meaning	Table Type
PREFIX	Shows Oracle Business Analytics-specific data warehouse application tables.	W_ = Warehouse
NAME	Unique table name.	All tables.
SUFFIX	Indicates the table type.	_A = Aggregate _D = Dimension _DEL = Delete _DH = Dimension Hierarchy _DHL = Dimension Helper _DHLS = Staging for Dimension Helper _DHS = Staging for Dimension Hierarchy _DS = Staging for Dimension _F = Fact _FS = Staging for Fact _G, _GS = Internal _H = Helper _HS = Staging for Helper _MD = Mini Dimension _PE = Primary Extract _PS = Persisted Staging _RH = Row Flattened Hierarchy _TL = Translation Staging (supports multi-language support) _TMP = Pre-staging or post-staging temporary table _UD = Unbounded Dimension _WS = Staging for Usage Accelerator

4.2 Table Types for Oracle Business Analytics Warehouse

Table 4–2 lists the types of tables used in the Oracle Business Analytics Warehouse.

Table 4–2 Table Types Used in the Oracle Business Analytics Warehouse

Table Type	Description
Aggregate tables (_A)	Contain summed (aggregated) data.
Dimension tables (_D)	Star analysis dimensions.
Delete tables (_DEL)	Tables that store IDs of the entities that were physically deleted from the source system and should be flagged as deleted from the data warehouse. Note that there are two types of delete tables: _DEL and _PE. For more information about the _PE table type, see the row for Primary extract tables (_PE) in this table.
Dimension Hierarchy tables (_DH)	Tables that store the dimension's hierarchical structure.
Dimension Helper tables (_DHL)	Tables that store many-to-many relationships between two joining dimension tables.
Staging tables for Dimension Helper (_DHLS)	Staging tables for storing many-to-many relationships between two joining dimension tables.
Dimension Hierarchy Staging table (_DHS)	Staging tables for storing the hierarchy structures of dimensions that have not been through the final extract-transform-load (ETL) transformations.

Table 4–2 (Cont.) Table Types Used in the Oracle Business Analytics Warehouse

Table Type	Description
Dimension Staging tables (_DS)	Tables used to hold information about dimensions that have not been through the final ETL transformations.
Fact tables (_F)	Contain the metrics being analyzed by dimensions.
Fact Staging tables (_FS)	Staging tables used to hold the metrics being analyzed by dimensions that have not been through the final ETL transformations.
Internal tables (_G, _GS)	General tables used to support ETL processing.
Helper tables (_H)	Inserted between the fact and dimension tables to support a many-to-many relationship between fact and dimension records.
Helper Staging tables (_HS)	Tables used to hold information about helper tables that have not been through the final ETL transformations.
Mini dimension tables (_MD)	Include combinations of the most queried attributes of their parent dimensions. The database joins these small tables to the fact tables.
Primary extract tables (_PE)	Tables used to support the soft delete feature. The table includes all the primary key columns (integration ID column) from the source system. When a delete event happens, the full extract from the source compares the data previously extracted in the primary extract table to determine if a physical deletion was done in the Siebel application. The soft delete feature is disabled by default. Therefore, the primary extract tables are not populated until you enable the soft delete feature. Note that there are two types of delete tables: _DEL and _PE. For more information about the _DEL table type, see the row for Delete table (_DEL) in this table.
Persisted Staging table (_PS)	Tables that source multiple data extracts from the same source table. These tables perform some common transformations required by multiple target objects. They also simplify the source object to a form that is consumable by the warehouse needed for multiple target objects. These tables are never truncated during the life of the data warehouse. These are truncated only during full load, and therefore, persist the data throughout.
Row Flattened Hierarchy Table (_RH)	Tables that record a node in the hierarchy by a set of ancestor-child relationships (parent-child for all parent levels).
Translation Staging tables (_TL)	Tables store names and descriptions in the languages supported by Oracle BI Applications.
Pre-staging or post-staging Temporary table (_TMP)	Source-specific tables used as part of the ETL processes to conform the data to fit the universal staging tables (table types_DS and _FS). These tables contain intermediate results that are created as part of the conforming process.
Unbounded dimension (_UD)	Tables containing information that is not bounded in transactional database data but should be treated as bounded data in the Oracle Business Analytics Warehouse.

Table 4–2 (Cont.) Table Types Used in the Oracle Business Analytics Warehouse

Table Type	Description
Staging tables for Usage Accelerator (_WS)	Tables containing the necessary columns for the ETL transformations.

4.2.1 Aggregate Tables in Oracle Business Analytics Warehouse

One of the main uses of a data warehouse is to sum up fact data with respect to a given dimension, for example, by date or by sales region. Performing this summation on-demand is resource-intensive, and slows down response time. The Oracle Business Analytics Warehouse precalculates some of these sums and stores the information in *aggregate tables*. In the Oracle Business Analytics Warehouse, the aggregate tables have been suffixed with _A.

4.2.2 Dimension Class Tables in Oracle Business Analytics Warehouse

A class table is a single physical table that can store multiple logical entities that have similar business attributes. Various logical dimensions are separated by a separator column, such as, type or category. W_XACT_TYPE_D is an example of a dimension class table. Different transaction types, such as, sales order types, sales invoice types, purchase order types, and so on, can be housed in the same physical table.

You can add additional transaction types to an existing physical table and so reduce the effort of designing and maintaining new physical tables. However, while doing so, you should consider that attributes specific to a particular logical dimension cannot be defined in this physical table. Also, if a particular logical dimension has a large number of records, it might be a good design practice to define a separate physical table for that particular logical entity.

4.2.3 Dimension Tables in Oracle Business Analytics Warehouse

The unique numeric key (ROW_WID) for each dimension table is generated during the load process. This key is used to join each dimension table with its corresponding fact table or tables. It is also used to join the dimension with any associated hierarchy table or extension table. The ROW_WID columns in the Oracle Business Analytics Warehouse tables are numeric. In every dimension table, the ROW_WID value of zero is reserved for Unspecified. If one or more dimensions for a given record in a fact table is unspecified, the corresponding key fields in that record are set to zero.

4.2.4 Dimension Tables With Business Role-Based Flags

This design approach is used when the entity is logically the same but participates as different roles in the business process. As an example, an employee could participate in a Human Resources business process as an employee, in the sales process as a sales representative, in the receivables process as a collector, and in the purchase process as a buyer. However, the employee is still the same. For such logical entities, flags have been provided in the corresponding physical table (for example, W_EMPLOYEE_D) to describe the record's participation in business as different roles.

While configuring the presentation layer, the same physical table can be used as a specific logical entity by flag-based filters. For example, if a particular star schema requires Buyer as a dimension, the Employee table can be used with a filter where the Buyer flag is set to Y.

4.2.5 Fact Tables in Oracle Business Analytics Warehouse

Each fact table contains one or more numeric foreign key columns to link it to various dimension tables.

4.2.6 Helper Tables in Oracle Business Analytics Warehouse

Helper tables are used by the Oracle Business Analytics Warehouse to solve complex problems that cannot be resolved by simple dimensional schemas.

In a typical dimensional schema, fact records join to dimension records with a many-to-one relationship. To support a many-to-many relationship between fact and dimension records, a helper table is inserted between the fact and dimension tables.

The helper table can have multiple records for each fact and dimension key combination. This allows queries to retrieve facts for any given dimension value. It should be noted that any aggregation of fact records over a set of dimension values might contain overlaps (due to a many-to-many relationship) and can result in double counting.

At times there is a requirement to query facts related to the children of a given parent in the dimension by only specifying the parent value (example: manager's sales fact that includes sales facts of the manager's subordinates). In this situation, one helper table containing multiple records for each parent-child dimension key combination is inserted between the fact and the dimension. This allows queries to be run for all subordinates by specifying only the parent in the dimension.

4.2.7 Hierarchy Tables in Oracle Business Analytics Warehouse

Some dimension tables have hierarchies into which each record rolls. This hierarchy information is stored in a separate table, with one record for each record in the corresponding dimension table. This information allows users to drill up and down through the hierarchy in reports.

There are two types of hierarchies in the Oracle Business Analytics Warehouse: a structured hierarchy in which there are fixed levels, and a hierarchy with parent-child relationships. Structured hierarchies are simple to model, since each child has a fixed number of parents and a child cannot be a parent. The second hierarchy, with unstructured parent-child relationships is difficult to model because each child record can potentially be a parent and the number of levels of parent-child relationships is not fixed. Hierarchy tables have a suffix of _DH.

4.2.8 Mini-Dimension Tables in Oracle Business Analytics Warehouse

Mini-dimension tables include combinations of the most queried attributes of their parent dimensions. They improve query performance because the database does not need to join the fact tables to the big parent dimensions but can join these small tables to the fact tables instead.

Table 4–3 lists the mini-dimension tables in the Oracle Business Analytics Warehouse.

Table 4–3 Mini-Dimension Tables in Oracle Business Analytics Warehouse

Table Name	Parent Dimension
W_RESPONSE_MD	Parent W_RESPONSE_D
W_AGREE_MD	Parent W_AGREE_D
W_ASSET_MD	Parent W_ASSET_D

Table 4–3 (Cont.) Mini-Dimension Tables in Oracle Business Analytics Warehouse

Table Name	Parent Dimension
W_OPTY_MD	Parent W_OPTY_D
W_ORDER_MD	Parent W_ORDER_D
W_QUOTE_MD	Parent W_QUOTE_D
W_SRVREQ_MD	Parent W_SRVREQ_D

4.2.9 Staging Tables in Oracle Business Analytics Warehouse

Staging tables are used primarily to stage incremental data from the transactional database. When the ETL process runs, staging tables are truncated before they are populated with change capture data. During the initial full ETL load, these staging tables hold the entire source data set for a defined period of history, but they hold only a much smaller volume during subsequent refresh ETL runs.

This staging data (list of values translations, computations, currency conversions) is transformed and loaded to the dimension and fact staging tables. These tables are typically tagged as <TableName>_DS or <TableName>_FS. The staging tables for the Usage Accelerator are tagged as WS_<TableName>.

The staging table structure is independent of source data structures and resembles the structure of data warehouse tables. This resemblance allows staging tables to also be used as interface tables between the transactional database sources and data warehouse target tables.

4.2.10 Translation Tables in Oracle Business Analytics Warehouse

Translation tables provide multi-language support by storing names and descriptions in each language that Oracle Business Analytics Warehouse supports. There are two types of translation tables:

- Domain tables that provide multi-language support associated with the values stored in the %_CODE columns.
- Tables that provide multi-language support for dimensions.

Domains and their associated translated values are stored in a single table named W_DOMAIN_MEMBER_LKP_TL. Each dimension requiring multi-language support that cannot be achieved with domains has an associated _TL table. These tables have a one-to-many relationship with the dimension table. For each record in the dimension table, you will see multiple records in the associated translation table (one record for each supported language).

4.3 Internal Tables in Oracle Business Analytics Warehouse

Internal tables are used primarily by ETL mappings for data transformation and for controlling ETL runs. These tables are not queried by end users. These tables are described in Table 4–4.

Table 4–4 Oracle Business Analytics Warehouse Internal Tables

Name	Purpose	Location
W_DUAL_G	Used to generate records for the Day dimension.	Data warehouse
W_COSTLST_G	Stores cost lists.	Data warehouse

Table 4–4 (Cont.) Oracle Business Analytics Warehouse Internal Tables

Name	Purpose	Location
W_DOMAIN_MEMBER_G	Staging table for populating incremental changes into W_DOMAIN_MEMBER_G and W_DOMAIN_MEMBER_G_TL.	Data warehouse
W_DOMAIN_MEMBER_G_TL	Stores translated values for each installed language corresponding to the domain member codes in W_DOMAIN_MEMBER_G_TL.	Data warehouse
W_DOMAIN_MEMBER_GS	Stores all the domain members and value for each installed language.	Data warehouse
W_DOMAIN_MEMBER_MAP_G	Used at ETL run time to resolve at target domain code base on the value of a source domain code.	Data warehouse
W_DOMAIN_MEMBER_MAP_NUM_G	Used at ETL run time to resolve a target domain code based on the comparison of a numeric value within the source numeric range.	Data warehouse
W_EXCH_RATE_G	Stores exchange rates.	Data warehouse
W_LANGUAGES_G	Stores the language translations supported in the data warehouse and is used during ETL to help generate missing translation records from the base language called pseudo-translation	Data warehouse
W_LOCALIZED_STRING_G		Data warehouse
W_LOV_EXCPT_G	Stores the list of values for the list of values types in which the ETL process finds exceptions.	Data warehouse
W_UOM_CONVERSION_G	Stores a list of From and To UOM codes and their conversion rates.	Data warehouse

4.4 Standard Column Prefixes in Oracle Business Analytics Warehouse

The Oracle Business Analytics Warehouse uses a standard prefix to indicate fields that must contain specific values, as shown in Table 4–5.

Table 4–5 Standard Column Prefix

Prefix	Description	In Table Types
W_	Used to store Oracle BI Applications standard or standardized values. For example, W_%_CODE (Warehouse Conformed Domain) and W_TYPE, W_INSERT_DT (Date records inserted into Warehouse).	_A _D _F

4.5 Standard Column Suffixes in Oracle Business Analytics Warehouse

The Oracle Business Analytics Warehouse uses suffixes to indicate fields that must contain specific values, as shown in Table 4–6.

Table 4–6 Standard Column Suffixes

Suffix	Description	In Table Types
_CODE	Code field. (Especially used for domain codes.)	_D, _DS, _FS, _G, _GS
_DT	Date field.	_D, _DS, _FS, _G, _DHL, _DHLS

Table 4–6 (Cont.) Standard Column Suffixes

Suffix	Description	In Table Types
_ID	Correspond to the _WID columns of the corresponding _F table.	_FS, _DS
_FLG	Indicator or Flag.	_D, _DHL, _DS, _FS, _F, _G, _DHLS
_WID	Identifier generated by Oracle Business Intelligence linking dimension and fact tables, except for ROW_WID.	_F, _A, _DHL
_NAME	A multi-language support column that holds the name associated with an attribute in all languages supported by the data warehouse.	_TL
_DESCR	A multi-language support column that holds the description associated with an attribute in all languages supported by the data warehouse	_TL

4.6 System Columns in Oracle Business Analytics Warehouse Tables

Oracle Business Analytics Warehouse tables contain system fields. These system fields are populated automatically and should not be modified by the user. Table 4–7 lists the system columns used in data warehouse dimension tables.

Table 4–7 System Columns Used in Data Warehouse Tables

System Column	Description
ROW_WID	Surrogate key to identify a record uniquely.
CREATED_BY_WID	Foreign key to the W_USER_D dimension that specifies the user who created the record in the source system.
CHANGED_BY_WID	Foreign key to the W_USER_D dimension that specifies the user who last modified the record in the source system.
CREATED_ON_DT	The date and time when the record was initially created in the source system.
CHANGED_ON_DT	The date and time when the record was last modified in the source system.
AUX1_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX2_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX3_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX4_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
DELETE_FLG	This flag indicates the deletion status of the record in the source system. A value of Y indicates the record is deleted from the source system and logically deleted from the data warehouse. A value of N indicates that the record is active.
W_INSERT_DT	Stores the date on which the record was inserted in the data warehouse table.

Table 4–7 (Cont.) System Columns Used in Data Warehouse Tables

System Column	Description
W_UPDATE_DT	Stores the date on which the record was last updated in the data warehouse table.
DATASOURCE_NUM_ID	Unique identifier of the source system from which data was extracted. In order to be able to trace the data back to its source, it is recommended that you define separate unique source IDs for each of your different source instances.
ETL_PROC_WID	System field. This column is the unique identifier for the specific ETL process used to create or update this data.
INTEGRATION_ID	Unique identifier of a dimension or fact entity in its source system. In case of composite keys, the value in this column can consist of concatenated parts.
TENANT_ID	Unique identifier for a tenant in a multi-tenant environment. This column is typically be used in an Application Service Provider (ASP)/Software as a Service (SaaS) model.
X_CUSTOM	Column used as a generic field for customer extensions.
CURRENT_FLG	This is a flag for marking dimension records as "Y" in order to represent the current state of a dimension entity. This flag is typically critical for Type II slowly changing dimensions, as records in a Type II situation tend to be numerous.
EFFECTIVE_FROM_DT	This column stores the date from which the dimension record is effective. A value is either assigned by Oracle BI Applications or extracted from the source.
EFFECTIVE_TO_DT	This column stores the date up to which the dimension record is effective. A value is either assigned by Oracle BI Applications or extracted from the source.
SRC_EFF_FROM_DT	This column stores the date from which the source record (in the Source system) is effective. The value is extracted from the source (whenever available).
STC_EFF_TO_DT	This column stores the date up to which the source record (in the Source system) is effective. The value is extracted from the source (whenever available).

4.7 Multi-Currency Support for System Columns

Table 4–8 lists the currency codes and rates for related system columns.

Table 4–8 Currency Codes and Rates for Related System Columns

System Column	Description
DOC_CURR_CODE	Code for the currency in which the document was created in the source system.
LOC_CURR_CODE	Usually the reporting currency code for the financial company in which the document was created.
GRP_CURR_CODE	The primary group reporting currency code for the group of companies or organizations in which the document was created.
LOC_EXCHANGE_RATE	Currency conversion rate from the document currency code to the local currency code.
GLOBAL1_EXCHANGE_RATE	Currency conversion rate from the document currency code to the Global1 currency code.

Table 4–8 (Cont.) Currency Codes and Rates for Related System Columns

System Column	Description
GLOBAL2_EXCHANGE_RATE	Currency conversion rate from the document currency code to the GLOBAL2 currency code.
GLOBAL3_EXCHANGE_RATE	Currency conversion rate from document currency code to the GLOBAL3 currency code.
PROJ_CURR_CODE	Code used in Project Analytics that corresponds to the project currency in the OLTP system.

4.8 Oracle Business Analytics Warehouse Primary Data Values

It is possible for various dimensions to have one-to-many and many-to-many relationships with each other. These kinds of relationships can introduce problems in analyses. For example, an Opportunity can be associated with many Sales Representatives and a Sales Representative can be associated with many Opportunities. If your analysis includes both Opportunities and Sales Representatives, a count of Opportunities would not be accurate because the same Opportunity would be counted for each Sales Representative with which it is associated.

To avoid these kinds of problems, the Oracle Business Analytics Warehouse reflects the primary member in the "many" part of the relationship. In the example where an Opportunity can be associated with many Sales Representatives, only the Primary Sales Representative is associated with that Opportunity. In an analysis that includes both Opportunity and Sales Representative, only a single Opportunity will display and a count of Opportunities returns the correct result.

There are a few important exceptions to this rule. The Person star schema supports a many-to-many relationship between Contacts and Accounts. Therefore, when querying the Person star schema on both Accounts and Contacts, every combination of Account and Contact is returned. The Opportunity-Competitor star schema supports a many-to-many relationship between Opportunities and Competitor Accounts, and the Campaign-Opportunity star schema supports a many-to-many relationship between Campaigns and Opportunities. In other star schemas, however, querying returns only the primary account for a given contact.

4.9 About Multi-Language Support in the Oracle Business Analytics Warehouse

Oracle BI Applications provides multi-language support for metadata level objects exposed in Oracle BI Enterprise Edition dashboards and reports, as well as data, which enables users to see records translated in their preferred language. For more information about multi-language support, see Chapter 2, "About Multi-Language Support."

4.10 Oracle Business Analytics Warehouse Currency Preferences

For information about setting up currencies, refer to the following task in Functional Setup Manager: **Common Areas and Dimensions Configurations\ Configure Global Currencies**.

The Oracle Business Analytics Warehouse supports the following currency preferences.

- **Contract currency.** The currency used to define the contract amount. This currency is used only in Project Analytics.

- **CRM currency.** The CRM corporate currency as defined in the Fusion CRM application. This currency is used only in CRM Analytics applications.
- **Document currency.** The currency in which the transaction was done and the related document created.
- **Global currency.** The Oracle Business Analytics Warehouse stores up to three group currencies. These need to be pre-configured so as to allow global reporting by the different currencies. The exchange rates are stored in the table W_EXCH_RATE_G.
- **Local currency.** The accounting currency of the legal entity in which the transaction occurred.
- **Project currency.** The currency in which the project is managed. This may be different from the functional currency. This applies only to Project Analytics.

5 Administering Oracle GoldenGate and Source Dependent Schemas

This chapter explains how to set up and use Oracle GoldenGate (OGG) and source dependent schemas (SDS).

This chapter contains the following sections:

- Section 5.1, "Introduction"
- Section 5.2, "Source Dependent Schema Architecture"
- Section 5.3, "Setting Up Oracle GoldenGate and Source Dependent Schemas"
- Section 5.4, "Tasks for Setting Up Oracle GoldenGate and the Source Dependent Schema"

5.1 Introduction

In a conventional ETL scenario, data is loaded from source online transaction processing (OLTP) schemas, which in many cases support full-time transactional systems with constant ongoing updates. Contention can arise during complex extracts from these sources, particularly in cases where significant OLTP data changes have occurred which must be processed and loaded by ETL processes.

To relieve this contention, you can set up source dependent schemas which replicate OLTP schemas in the same database as the Oracle Business Analytics Warehouse schema. In addition to segregating extract processing on the analytical system and eliminating contention on transactional systems, physical architecture and ETL performance benefits accrue from maintaining source data in the same physical location as the warehouse tables, consolidating multiple sources, regions and timezones, and eliminating network bottlenecks and incremental change capture during extraction and load.

5.2 Source Dependent Schema Architecture

The SDS is a separate schema usually stored on the same database as the Oracle Business Analytics Warehouse, which contains data extracted from an OLTP schema on a separate machine. The OLTP schema is treated as the source and the SDS schema as the target of the Oracle GoldenGate processes which maintain the replicated SDS.

The SDS Architecture is an optional addition to the existing BI Applications Architecture that solves many problems associated with data transport from the source OLTP system to the data warehouse and change data capture required for incremental ETL. The architecture consists of these main components:

- Source Dependent Data Store (SDS): A separate schema on the OBAW database that is a replication of the source OLTP systems tables. Also stores deletes and additional optimizations for incremental ETL.
- Oracle GoldenGate: This replication system is deployed on both source and OBAW database systems. On the source database system, Oracle GoldenGate supports continuous asynchronous change data capture at a low level in the database, then compresses and ships the changed data across the network to the target SDS schema on the analytical warehouse database instance. On the target OBAW database instance, it receives the changed data from one or more source systems and loads them into the target database, specifically into the SDS schemas, one per ETL OLTP source.
- ODI: ODI metadata stores definitions used to generate the SDS schemas and to support the Oracle GoldenGate replication processes.
- BI Application SDS Components: Components used to support generation of the SDS schema and Oracle GoldenGate replication processes.

5.3 Setting Up Oracle GoldenGate and Source Dependent Schemas

This section provides instructions for setting up and using Oracle GoldenGate and SDS.

List of Steps for Setting Up Oracle GoldenGate and Source Dependent Schemas

To install and set up Oracle GoldenGate and SDS, you must complete the following tasks, in order. Each high-level task breaks down into a list of detailed steps provided in the next section.

1. Configure the source and target databases, as described in Section 5.4.1, "Setup Step: Configure Source and Target Database."
2. Install Golden Gate software, as described in Section 5.4.2, "Setup Step: Install Oracle GoldenGate on Source and Target Systems."
3. Configure BI Applications Configuration Manager and ODI, as described in Section 5.4.3, "Setup Step: Configure BI Applications Configuration Manager and Oracle Data Integrator to Support the Source Dependent Schema."
4. Generate and run the Data Definition Language to create the SDS tables on the SDS schema in the target database, as described in Section 5.4.4, "Setup Step: Generate, Deploy, and Populate the Source Dependent Schema Tables on Target Database."
5. Generate and deploy OGG Parameter files, as described in Section 5.4.5, "Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Machines."
6. Start OGG, as described in Section 5.4.6, "Setup Step: Start Oracle GoldenGate on Source and Target Machines."

5.4 Tasks for Setting Up Oracle GoldenGate and the Source Dependent Schema

This section provides detailed tasks for setting up Oracle GoldenGate and SDS.

Note: You must perform the tasks in this section in the sequence described in Section 5.3, "Setting Up Oracle GoldenGate and Source Dependent Schemas".

5.4.1 Setup Step: Configure Source and Target Database

In this step, you create Oracle GoldenGate database users on source and target databases. Unlike other database schemas used by BI Applications, the SDS and OGG schemas are not automatically created during installation. Only the installation process can automatically create database users; because datasources are defined in Configuration Manager after installation is complete, the required Source Dependent Schemas associated with these datasources must be manually created. For this reason, an SDS schema must be manually defined on the target database. Additionally, the BI Apps installer is not able to create the OGG database user on the source OLTP system. This section describes how to create the OGG database user on the source database system and the OGG and SDS database users on the target database system.

Create OLTP Oracle GoldenGate Database User

Each OGG process requires a dedicated database user. On the source system, the OGG user needs to be able to query various metadata. Secure database practice is to avoid granting privileges to tables not in use, so `SELECT ANY TABLE` is not granted to the OGG database user. Instead, as part of the SDS DDL, `SELECT` privileges are granted only to those tables in the OLTP schema being replicated.

The user creation scripts use the following parameters:

Parameter	Description
&BIAPPS_OGG	Oracle GoldenGate Database User Name
&BIAPPS_OGG_PW	Oracle GoldenGate Database User Password

Run the following script on the source database to create the source database OGG user.

```
-- Create OGG User
CREATE USER &BIAPPS_OGG
IDENTIFIED BY &BIAPPS_OGG_PW
DEFAULT TABLESPACE USERS QUOTA UNLIMITED ON USERS;

GRANT CREATE SESSION TO &BIAPPS_OGG;
GRANT ALTER SESSION TO &BIAPPS_OGG;
GRANT SELECT ANY DICTIONARY TO &BIAPPS_OGG;
GRANT FLASHBACK ANY TABLE TO &BIAPPS_OGG;


-- OGG user requires ALTER ANY table to set up supplemental logging for individual
tables. Once accomplished, this privilege can be revoked:
GRANT ALTER ANY TABLE TO &BIAPPS_OGG;
```

Prepare OLTP Database and Redo Logs

Oracle GoldenGate requires that the database be configured for supplemental logging. Execute the following statement in the source database with a user with sufficient privileges.

```
ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;
```

Create Target Oracle GoldenGate Database User

Each OGG process requires a dedicated database user. On the target system, the OGG user needs to be able to execute various DML operations on the SDS tables as well as optionally create a checkpoint table. Secure database practice is to avoid granting privileges to tables not in use, so `SELECT ANY TABLE`, `INSERT ANY TABLE` and so on are not granted to the OGG database user. Instead, as part of the SDS DDL, required privileges are granted only to those tables in the SDS schema for the OGG database user.

The user creation scripts use the following parameters:

Parameter	Description
&BIAPPS_OGG	Oracle GoldenGate Database User Name
&BIAPPS_OGG_PW	Oracle GoldenGate Database User Password

Run the following script on the target table to create the target database OGG user.

```
-- Create OGG User
CREATE USER &BIAPPS_OGG
IDENTIFIED BY &BIAPPS_OGG_PW
DEFAULT TABLESPACE USERS QUOTA UNLIMITED ON USERS;

GRANT CREATE SESSION TO &BIAPPS_OGG;
GRANT ALTER SESSION TO &BIAPPS_OGG;
GRANT SELECT ANY DICTIONARY TO &BIAPPS_OGG;

-- Create Table privilege only required to create checkpoint table. Can be revoked
once table is created. Not required if not creating this table
GRANT CREATE TABLE TO &BIAPPS_OGG;
```

Create SDS Database User

A separate SDS database user must be configured in the target database for each OLTP system that will leverage the SDS. Each supported source instance requires a separate SDS schema. The recommended naming convention for the schema owner is <*BIAPPS*>SDS<*Model Code*>_<*DSN Number*> where <*BIAPPS*> is a user defined code representing BI Applications content, <*Model Code*> is the unique code assigned to each datasource type and <*DSN Number*> is the unique datasource ID assigned to a specific datasource instance. For example, if you have the following two datasources defined as supported source systems in the BI Applications Configuration Manager you would have the corresponding SDS schemas defined in the data warehouse database :

Source Instance Name	Model Code	Data Source Number	SDS
Oracle EBS 11.5.10	EBS_11_5_10	310	BIAPPS_SDS_ EBS_11_5_10_310
Siebel CRM 8.1.1	SEBL_8_1_1	625	BIAPPS_SDS_ SEBL_8_1_1_625

Use the following DDL as a template for creating each SDS database user. The following only represents a bare minimum of required DDL statements; adjust for your environment as necessary. Rerun for each supported source instance.

Parameter	Description
&BIAPPS_SDS_DATA_TS	Table space name

Parameter	Description
&ORADATA	Path where tablespace should be located
&BIAPPS_SDS	SDS User name
&BIAPPS_SDS_PW	SDS User password
&BIAPPS_OGG	Oracle GoldenGate Database User Name

```
-- Create tablespace. Following is only an example and may not reflect PSR
guidance:
CREATE TABLESPACE &BIAPPS_SDS_DATA_TS
DATAFILE '&ORADATA/&BIAPPS_SDS_DATA_TS..dbf' SIZE 100M AUTOEXTEND ON NEXT 10M
LOGGING
DEFAULT COMPRESS FOR OLTP;

-- Create SDS User
CREATE USER &BIAPPS_SDS
IDENTIFIED BY &BIAPPS_SDS_PW
DEFAULT TABLESPACE &BIAPPS_SDS_DATA_TS QUOTA UNLIMITED ON &BIAPPS_SDS_DATA_TS;

-- Required Grants
GRANT CREATE SESSION TO &BIAPPS_SDS;
GRANT CREATE TABLE TO &BIAPPS_SDS;

-- OGG user must be granted Quota to insert and update data
ALTER USER &BIAPPS_OGG QUOTA UNLIMITED ON &BIAPPS_SDS_DATA_TS;
```

5.4.2 Setup Step: Install Oracle GoldenGate on Source and Target Systems

Download and install Oracle GoldenGate software first on the source and then on the target machines. The software is available from Oracle Technology Network. For information about installation of Oracle Golden Gate, refer to the Oracle GoldenGate Installation and Setup Guide for your platform and database.

Installation Recommendations

When installing and configuring the OGG software, consider the following recommendations:

- For each OLTP instance supported, install a separate Replicate process on the target machine. As each OLTP instance has its own separate SDS schema on the target database, the Replicate process is populating different targets so a separate Replicate process is required for each.
- Install a Data Pump process on the source machine.
- The name of the Extract, Data Pump and Replicat processes are limited to eight characters. The suggested naming convention is as follows:

Process	Naming Convention	Example
Extract	EXT_<*Datasource Num Id*>	EXT_310
Data Pump	DP_<*Datasource Num Id*>	DP_310
Replicate	REP_<*Datasource Num Id*>	REP_310

- Follow the steps in the Oracle GoldenGate documentation to configure an instance of OGG on the source and target systems up to the point of starting the OGG processes.
- Note that as part of the installation and configuration, a procedure is run to generate BI Applications-specific parameter files, as discussed in the following section. For more information, see Section 5.4.5, "Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Machines." The install and configuration of the OGG processes are completed at this point.

Example Steps to configure the Oracle GoldenGate processes

The following is a set of example steps to configure the OGG processes, modify as appropriate for your environment. For the source system, configure Extract and Data Pump processes. The initial steps in the example below effectively remove an existing instance of both processes. If none already exist, start with the `START MGR` command.

```
--Stop Manager on primary database
dblogin USERID <GG User's DB ID, requirement depends on database>,
PASSWORD <GG User's DB password, requirement depends on database >

STOP MGR

--Stop GG processes
STOP <name of Extract process>
DELETE EXTTRAIL <relative or fully qualified path where Extract Trail files are
created on source system>/*
DELETE <name of Extract process>

STOP <name of Data Pump process>
DELETE RMTTRAIL <relative or fully qualified path where Replicat Trail files are
created on target system>/*
DELETE <name of Data Pump process>

--Delete Previous Trail Files
SHELL rm <relative or fully qualified path where Extract Trail files are created
on source system>/*

--Start Manager on primary database
START MGR

--Primary database capture configuration
ADD EXTRACT <name of Extract process>, TRANLOG, BEGIN NOW
ADD EXTTRAIL <relative or fully qualified path where Extract Trail files are to be
created on source system>, EXTRACT <name of Extract process>, MEGABYTES 50

--Primary database pump configuration:
ADD EXTRACT<name of Data Pump process>, EXTTRAILSOURCE <relative or fully
qualified path where Extract Trail files are to be created on source system>,
ADD RMTTRAIL <relative or fully qualified path where Replicat Trail files are to
be created on target system>, EXTRACT<name of Data Pump process>, MEGABYTES 50
```

Example:

```
--Stop Manager on primary database
dblogin userid gg, password gg
STOP MGR

--Stop GG processes
STOP EXT_310
DELETE EXTTRAIL ./dirdat/*
```

```
DELETE EXT_310

STOP DP_310
DELETE RMTTRAIL ./dirdat/*
DELETE DP_310

--Delete Previous Trail Files
SHELL rm ./dirdat/*

--Start Manager on primary database
START MGR

--Primary database capture configuration
ADD EXTRACT EXT_310, TRANLOG, BEGIN NOW
ADD EXTTRAIL ./dirdat/tr, EXTRACT EXT_310, MEGABYTES 50

--Primary database pump configuration:
ADD EXTRACT DP_310, EXTTRAILSOURCE ./dirdat/tr
ADD RMTTRAIL ./dirdat/tr, EXTRACT DP_310, MEGABYTES 50
```

Implement similar steps for the Replicate process in the target system. The initial steps effectively remove an existing instance of the Replicate process. If none already exist, start with the START MGR command.

```
--Stop Manager on target database
dblogin USERID <GG User's DB ID, requirement depends on database>,
PASSWORD <GG User's DB password, requirement depends on database >
STOP MGR

--Stop GG processes
STOP <name of Replicat process>
DELETE <name of Replicat process>

--Delete CHECKPOINTTABLE
DELETE CHECKPOINTTABLE <GG User's DB ID>.GGSCHKPT

--Delete Previous Trail Files
SHELL rm <relative or fully qualified path where Replicat Trail files are created
on target system>/*

--Start Manager on target database
START MGR

--Create CHECKPOINTTABLE in target database
dblogin USERID <GG User's DB ID>,
PASSWORD <GG User's DB password>
ADD CHECKPOINTTABLE <GG User's DB ID>.GGSCHKPT

--Target database delivery configuration
ADD REPLICAT <name of Replicat process>, exttrail <relative or fully qualified
path where Replicat Trail files are to be created on target system>
```

Example:

```
--Stop Manager on target database
dblogin userid gg, password gg
STOP MGR

--Stop GG processes
STOP REP_310
DELETE REP_310
```

```
--Delete CHECKPOINTTABLE
DELETE CHECKPOINTTABLE

--Delete Previous Trail Files
SHELL rm ./dirdat/*

--Start Manager on target database
START MGR

--Create CHECKPOINTTABLE in target database
dblogin userid gg, password gg
ADD CHECKPOINTTABLE

--Target database delivery configuration
ADD REPLICAT REP_310, exttrail ./dirdat/tr
```

5.4.3 Setup Step: Configure BI Applications Configuration Manager and Oracle Data Integrator to Support the Source Dependent Schema

Enable SDS in Configuration Manager

Enable the SDS option for each datasource defined in Configuration Manager. You can enable the SDS option for the entire datasource or for individual Fact Groups. The SDS option is enabled by setting the value for the IS_SDS_DEPLOYED parameter to 'Yes'.

1. In Configuration Manager, select the Source Instance.
2. Click Manage Data Load Parameters.
3. Locate the IS_SDS_DEPLOYED parameter in the list.
4. In the Global Parameter Value, replace <Edit Value> with 'Yes'.
5. A warning is displayed indicating that the parameter is being updated globally for all Fact and Dimension Groups. Click Yes to confirm or, if you prefer, set the global paramter to 'No', and then edit the parameter value at the Fact Group or Dimension Group level.

Add SDS Physical Schemas in ODI

For each source instance, you must manually add a corresponding physical schema under the 'BIAPPS_DW' physical server in ODI.

1. In ODI Studio's Topology Navigator, expand the Oracle technology in the Physical Architecture.
2. Right-click on Oracle_BI_Apps_DW and select New Physical Schema.
3. In the Definition, set Schema (Schema) and Schema (Work Schema) both to the SDS schema owner.
4. Select Flexfields.
5. For the DATASOURCE_NUM_ID flex field, uncheck the Default checkbox and assign the DSN value associated with that source as defined in Configuration Manager.
6. Save the physical schema definition. Ignore the message about the physical server not being assigned a context.

5.4.4 Setup Step: Generate, Deploy, and Populate the Source Dependent Schema Tables on Target Database

In this step, you generate and run the Data Definition Language to create the SDS tables on the SDS schema in the target database.

Generate the SDS DDL

Procedures are provided to generate the required objects to enable the SDS. To generate the required DDL, in ODI Designer execute the 'Generate DDL - SDS' scenario found under 'BI Apps Project > Components > SDS > Oracle > Generate SDS DDL. Provide an appropriate context when prompted.

As the procedure can only accept a single source type, this process needs to be repeated for each different type of Source system to be enabled.

To execute the scenario, right-click it and select Execute. When the scenario is executed, a prompt appears to provide values for the DDL execution options. Refer to the following table describing the options to provide appropriate values.

Option	Description
GG_USER_DW	Golden Gate database user on the BI Applications database
GG_USER_SOURCE	Golden Gate database user on the OLTP database.
SDS_MODEL	The OLTP model to be used to generate the SDS schema. Each model is associated with a logical schema. The logical schema is associated with a physical schema. The logical and physical schema DSN flexfields must match an SDS physical schema's DSN flexfield defined under the BI Apps DW physical server in order for this utility to determine the appropriate schema name to be used for the SDS. The SDS physical schema with the matching DSN flexfield value is used to identify changes and execute the DDL against if the RUN_DDL option is set to Y.
CREATE_SCRIPT_FILE	Y or N. Set to Y if you want to review the DDL or manually execute the script.
REFRESH_MODE	FULL or INCREMENTAL. Full drops and creates all tables. Incremental compares the repository with the SDS schema and applies changes using `ALTER` statements without dropping tables. Incremental process can take much longer than Full mode and should be used with filters to limit the number of tables compared.
CHAR_CLAUSE	Y or N. For Unicode support, will include the `CHAR` clause when defining string columns. For example `FULL_NAME VARCHAR2(10)` would be created as `FULL_NAME VARCHAR2(10 CHAR)`. This ensures that the right length is chosen when the underlying database is a unicode database.

Option	Description
RUN_DDL	Y or N. Determines whether to execute the DDL commands. The script will be executed against the physical SDS schema associated with the SDS_MODEL. Note that this script will be executed via the user defined in the BIAPPS_DW physical server which is usually the owner of the BIAPPS_DW schema and which may not have appropriate privileges to create or alter tables in another schema. To have the utility execute the DDL script, you may need to grant `CREATE ANY TABLE`, `CREATE ANY INDEX`, `ALTER ANY TABLE` and `ALTER ANY INDEX` to the BIAPPS_DW database user. It is recommended to set this option to N. If set to Y, both the tables and indexes will be created. Having the indexes on the tables will impact the performance of initially loading the tables. Rather, it is recommended that you set this option to N, manually execute the Table DDL, perform the initial load of the tables, then manually execute the Index DDL. Also, if an index or primary key constraint is not defined correctly in ODI, uniqueness or not null errors could be generated and a table could fail to be loaded. Indexes and primary keys are useful for Oracle GoldenGate but are not required. It is better to build the indexes and primary keys after the data is loaded and make any necessary corrections to the constraint's definition in ODI and attempt to build the index or primary key again.
SCRIPT_LOCATION	The location where the script should be created if the CREATE_SCRIPT_FILE option is true.
TABLE_MASK	To generate the DDL for all tables, use a wildcard (the default). To generate for only a subset of tables with names matching a particular pattern, use that pattern with a wildcard, such as PER_%.

If you set CREATE_SCRIPT_FILE to Y, four files are generated by the Generate SDS DDL procedure in the location specified by SCRIPT_LOCATION. One is a .SQL script to creates the tables. Another is a .SQL script to create the indexes and analyze the tables. This allows you to create the tables, perform an initial load of the tables without any indexes that could hurt performance, and then create the indexes and analyze the tables after they are loaded. Another .SQL script is generated which grants `SELECT` privileges to the OGG database user only for those tables that need to be selected from. The final file is a log file.

Grant privileges to OLTP Tables

The OGG user must be able to select from the tables in the OLTP database. Rather than grant the `SELECT ANY TABLE` privilege to the OGG user, `SELECT` privileges are granted only to those tables that actually need to be replicated to the target system.

The SDS DDL generator procedure creates a script to grant `SELECT` privileges to the OGG user. Refer to the script 'BIA_SDS_Schema_Source_Grants_DDL_<*unique ID*>.sql' and execute the contents in the OLTP database with a user with sufficient privileges to grant `SELECT` privileges on the OLTP tables.

Create the SDS Tables

The SDS DDL generator procedure creates a .SQL script that follows the naming convention BIA_SDS_Schema_DDL_<*unique ID*>.sql which contains the `CREATE` or `ALTER` DDL statements to create or alter the tables in the SDS schema. Execute the SQL in this file against the SDS schema.

The ETL process must be able to select from the SDS tables. Typically, the ETL process uses the BI Applications data warehouse schema owner. This must be granted `SELECT` privileges on the SDS tables. In addition, the Oracle GoldenGate user needs read and write access to these same tables. The SDS Generate DDL procedure grants `SELECT` privileges to the BI Applications data warehouse schema owner and `SELECT`, `INSERT`, `UPDATE` and `DELETE` privileges to the OGG user.

Performing Initial Load of the SDS Tables: Create Database Link to OLTP Database

A variety of methods can be used to initially load the data from the source database to the target database. A procedure is provided to generate a script to perform an initial load as described in the steps below. Note however, that you may opt for other methods. The procedure generates a script that executes DML statements that extract data over a database link.

> **Note:** LOB and LONG datatype columns are created in the SDS, but the provided utilities to initially copy data from the source to target system cannot support these datatypes, so columns with these datatypes are specifically excluded by these utilities. If data in these columns are required, an alternate method for performing an initial load of the SDS will need to be implemented.

> **Note:** In Siebel implementations, a small number of tables in the Siebel database are created when installing the BI Applications. These tables must be manually created and always have S_ETL as a prefix. Be sure these tables have already been created prior to executing these steps. If these tables do not already exist, a "table or view does not exist" error can occur when executing the following commands.

First, create a database link to the OLTP database on the target database. The procedure to generate the DML script requires that a database link already exist named "DW_TO_OLTP" prior to being executed. The procedure executes using the BIAPPS_DW physical server so the database link has to either be defined in the same schema as used in the BIAPPS_DW physical server or else defined as a public database link. This database link must be manually created, it is not automatically created.

The procedure only populates a single SDS schema at a time. If creating multiple SDS schemas to accommodate multiple sources, this database link will need to be updated prior to each run to point to a different OLTP instance.

> **Note:** The JDE application spreads data across four databases and is tracked under four different submodels under a single JDE specific model. The DML option will need to be executed for each separate submodel and the "DW_TO_OLTP" database link will need to be updated prior to executing the DML script.

Performing Initial Load of the SDS Tables: Execute Procedure to generate DML script

This DML script generation procedure requires that the ODI topology is set up correctly, ensuring the OLTP model logical schema DSN matches with the desired target warehouse SDS physical schema DSN. The DSNs are set in the logical or physical schema flexfields.

In ODI Designer, execute the COPY_OLTP_TO_SDS scenario found under BI Apps Project > Components > SDS > Oracle > Copy OLTP to SDS.

To execute the scenario, right-click it and select Execute. Provide an appropriate context when prompted. When the scenario is executed, a prompt appears to provide values for the DML execution options. Refer to the following table describing the options to provide appropriate values.

Option	Description
TABLE_LIST	A comma-separated list of tables. A wildcard match % may be used to match multiple tables. Do not include any line breaks. For example: PER_ALL_ASSIGNMENTS_ F,PER%ORG%INFO%,HR%UNIT,FND_LOOKUP_TYPES
CREATE_SCRIPT_FILE	Y or N. Set to Y if you want to review the DDL or manually execute the script.
RUN_DDL	Y or N. Whether to execute the DML commands. The script will be executed against the physical SDS schema associated with the SDS_MODEL. Note that this script will be executed via the user defined in the BIAPPS_DW physical server which is usually the owner of the BIAPPS_DW schema and which may not have appropriate privileges to insert data into tables in another schema. To have the utility execute the DDL script, you may need to grant `SELECT ANY TABLE` and `INSERT ANY TABLE` to the BIAPPS_DW database user. Alternatively, rather than have the procedure execute the script, create the script file and connect to the database as the SDS schema owner and execute the contents of the script file directly.
SDS_MODEL	The OLTP model to be used to generate the SDS schema.
SCRIPT_LOCATION	The location where the script should be created if the CREATE_ SCRIPT_FILE option is true.

Performing Initial Load of the SDS Tables: Execute DML Script on SDS Database

The resulting DML script can be executed using the SDS schema owner or the BIAPPS DW schema owner. If executed by the BIAPPS DW schema owner, this user must be granted the `SELECT ANY TABLE` and `INSERT ANY TABLE` privileges in order to populate data in another schema. If executed using the SDS schema owner, a private database link named "DW_TO_OLTP" must be created in the SDS schema (the SDS user must be granted the `CREATE DATABASE LINK` privilege to create this database link) or already created as a public database link.

The DML script that is generated includes all tables used by all ETL tasks. If you are not executing all ETL tasks, you may want to consider identifying the tasks you are not executing and removing the corresponding tables from this script so that they are not replicated, thus keeping the overall size of the SDS down. Refer to the parameter files to determine the tasks that use each table and edit this script to remove the tables you do not need to replicate.

Create SDS Indexes and Analyze the SDS schema

When the tables are populated, execute the 'BIA_SDS_Schema_Index_DDL_<*unique ID*>.sql' script to create indexes and analyze the SDS tables.

5.4.5 Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Machines

Parameter files are used to control how Oracle GoldenGate operates. These files are deployed to the source system, where the Extract and Data Pump processes are executed, and the target system, where the Replicat process is executed. An ODI procedure generates these parameter files based on the metadata defined in ODI. A scenario that executes this procedure is provided to generate the Oracle GoldenGate parameter files to populate the SDS.

5.4.5.1 Generate Oracle GoldenGate Parameter Files

To generate the required parameter files, execute the 'GENERATE_SDS_OGG_PARAM_FILES' scenario found under BI Apps Project > Components > SDS > Generate SDS OGG Param Files. When the scenario is executed, a prompt appears to provide values for the parameter file options. Refer to the following table describing the options to provide appropriate values to match your environment. As the procedure can only accept a single Source type, this process needs to be repeated for each different type of Source system to be enabled.

Parameter	Description
PARAM_FILE_LOCATION	Location on machine where ODI client is running where parameter files will be created. Example: C:\temp\
DATASOURCE_NUM_ID	Datasource Num Id value associated with the particular source for which parameter files are to be generated. Example: 310
DATAPUMP_NAME	Name of the Datapump Process specified when installing OGG on the source machine. Limit is eight characters. Suggested naming convention is DP_<*Datasource Num Id*>, for example DP_310.
EXTRACT_NAME	Name of the Primary Extract Process specified when installing OGG on the source machine. Limit is eight characters. Suggested naming convention is EX_<*Datasource Num Id*>, for example EXT_310.
EXTRACT_TRAIL	Path and name of trail file on source system. Can be a relative or fully qualified path, though actual file name must be two characters. In the example below, 'tr' is the name of the trail file. Example: ./dirdat/tr
DEFSFILE	The relative or fully qualified path on the source system where the DEFGEN definition file should be created and file name. This value is included in the DEFGEN.prm parameter file that is generated by this procedure. The DEFGEN utility is executed on the source database, so the path provided must be a path available on the system the source database runs on. Suggested naming convention is DEF_<*Datasource Num Id*>.def. Example: ./dirdef/DEF_310.def
SOURCE_GG_USER_ID	Database user dedicated to the Oracle GoldenGate processes on the source database. Example: GG_USER
SOURCE_GG_PASSWORD	Password for the database user dedicated to the Oracle GoldenGate processes on the source database. By default, the password is stored as clear text in the generated parameter file. If an encrypted value is desired, use the ENCRYPT PASSWORD utility and edit the generated parameter files accordingly. Example: GG_PASSWORD
SOURCE_PORT	Port used by the OGG Manager Process on the source system. The default value when OGG is installed is 7809.
REPLICAT_NAME	Name of the Replicat Process specified when installing OGG on the target machine. Limit is eight characters. Suggested naming convention is REP_<*Datasource Num Id*>, for example REP_310

Parameter	Description
SOURCE_DEF	This is the Source Definitions file created by executing the DEFGEN utility on the source database and copied over to the target machine. This can be either a relative or fully qualified path to this definition file on the target system. Include the /dirdef subfolder as part of the path. Suggested naming convention is DEF_<*Datasource Num Id*>.def, for example ./dirdef/DEF_310.def Note that the file name is usually the same as the one defined for DEFSFILE but the paths are usually different as DEFSFILE includes the path where OGG is stored on the source system, while SOURCE_DEFS includes the path where OGG is installed on the target system.
REMOTE_HOST	IP address or Host Name of the target machine where the Replicat process runs.
REMOTE_TRAIL	Path and name of the trail file on target system. Can be a relative or fully qualified path though the actual file name must be two characters. In the example below, 'tr' is the name of the trail file. Example: ./dirdat/tr
BIA_GG_USER_ID	Database user dedicated to the Oracle GoldenGate processes on the target database, for example GG_USER
BIA_GG_PASSWORD	Password for the database user dedicated to the Oracle GoldenGate processes on the target database. By default, the password is stored as clear text in the generated parameter file. If an encrypted value is desired, use the ENCRYPT PASSWORD utility and edit the generated parameter files accordingly. Example: GG_PASSWORD
BIA_GG_PORT	Port used by the OGG Manager Process on the target system. The default value when OGG is installed is 7809.

The procedure automatically creates subfolders under a folder you specify. The naming convention is DSN_<*DATASOURCE_NUM_ID*> where <*DATASOURCE_NUM_ID*> is the value you specify when executing this procedure. For example, if you specify 310 as the value for DATASOURCE_NUM_ID, there will be a folder named DSN_310. Under this folder are two more subfolders, 'source' and 'target'. The 'source' folder contains all of the files that need to be copied to the source system, while 'target' contains all of the files that need to be copied to the target system.

> **Tip:** The parameter files that are generated include all tables used by all ETL tasks. The task that uses the table is identified in the parameter file. If you are not executing all ETL tasks, you may want to consider identifying the tasks you are not executing and removing the corresponding tables from the parameter files so that they are not replicated. This keeps the overall size of the SDS down.

About JD Edwards Support

The JDE application spreads data across up to four databases. Each database instance must be assigned its own extract/datapump processes and a separate corresponding replicat process. If the JDE components are on a single database, generate a single set of parameter files. If the JDE components are spread across two, three or four databases, generate a corresponding number of parameter files.

For more details on this configuration, refer to the "Configuring Oracle GoldenGate for real-time data warehousing" chapter in the *Oracle GoldenGate Windows and UNIX Administrator's Guide*, which discusses multiple source databases replicating data to a single target database.

Keep the following in mind when generating the parameter files.Execute the procedure for each database instance. The name of each process and trail file should be unique. The following example assumes all four components are on different databases:

Component	Extract Name	Data Pump Name	Extract Trail	Defs File	Replicat Name	Replicat Trail	Source Defs
Control	EX_410A	DP_410A	./dirdat/ta	./dirdef/DEF_310A.def	REP_410A	./dirdat/ta	./dirdef/DEF_310A.def
Data	EX_410B	DP_410B	./dirdat/tb	./dirdef/DEF_310B.def	REP_410B	./dirdat/tb	./dirdef/DEF_310B.def
Data Dictionary	EX_410C	DP_410C	./dirdat/tc	./dirdef/DEF_310C.def	REP_410C	./dirdat/tc	./dirdef/DEF_310C.def
System	EX_410D	DP_410D	./dirdat/td	./dirdef/DEF_310D.def	REP_410D	./dirdat/td	./dirdef/DEF_310D.def

About PeopleSoft Learning Management Support

PeopleSoft has a Learning Management pillar which is tightly integrated with the Human Capital Management pillar. HCM can be deployed without LM but LM cannot be deployed without HCM. When both are deployed, BI Applications treats the HCM with LM pillars in a similar fashion as it treats JDE: the data is spread across two databases but is treated as a single application. As with the JDE application, in this configuration each database instance must be assigned its own extract/datapump processes and a separate corresponding replicat process.

For more details on this configuration, refer to the "Configuring Oracle GoldenGate for real-time data warehousing" chapter in the *Oracle GoldenGate Windows and UNIX Administrator's Guide*, which discusses multiple source databases replicating data to a single target database.

Keep the following in mind when generating the parameter files.Execute the procedure for each database instance. The name of each process and trail file should be unique.

Component	Extract Name	Data Pump Name	Extract Trail	Defs File	Replicat Name	Replicat Trail	Source Defs
HCM Pillar	EX_518A	DP_518A	./dirdat/ta	./dirdef/DEF_518A.def	REP_518A	./dirdat/ta	./dirdef/DEF_518A.def
LM Pillar	EX_518B	DP_518B	./dirdat/tb	./dirdef/DEF_518B.def	REP_518B	./dirdat/tb	./dirdef/DEF_518B.def

5.4.5.2 Configure the Source System

Copy all of the files from the 'source' directory on the ODI client to the corresponding directories in the source system:

Copy the following file to the <ORACLE OGG HOME> directory:

- ADD_TRANDATA.txt

Copy the following files to the <ORACLE OGG HOME>/dirprm directory:

- DEFGEN.prm
- <*EXTRACT_NAME*>.prm where <EXTRACT_NAME> is the value specified when generating the parameter files.
- <*DATAPUMP_NAME*>.prm where <DATAPUMP_NAME> is the value specified when generating the parameter files.

Edit the Extract parameter file

By default, the procedure creates a basic set of parameter files that do not include support for a variety of features. For example, the parameter files do not include support for Transparent Data Encryption (TDE) or unused columns. The procedure also does not include the options to encrypt data.

If your source tables have unused columns, edit the Extract parameter file to include DBOPTIONS ALLOWUNUSEDCOLUMN. If encrypting the data is desired, edit the parameter files to add the ENCRYPTTRAIL and DECRYPTTRAIL options.

To support such features, edit the generated parameter files using the GGSCI EDIT PARAMS <parameter file> command. Also edit the generated param files to implement various tuning options that are specific to the environment. Refer to the *Oracle GoldenGate Reference* guide for details on implementing these options.

Start the GGSCI command utility from the <ORACLE OGG HOME> directory. Execute the following command to edit the Extract parameter file - this should open the Extract parameter file you copied to <ORACLE OGG HOME>/dirprm:

```
GGSCI>EDIT PARAMS <EXTRACT_NAME>
```

Save and close the file.

Enable Table Level Logging

Oracle GoldenGate requires table-level supplemental logging. This level of logging is only enabled for those tables actually being replicated to the target system. The SDS Parameter file generator creates 'ADD_TRANDATA.txt' file to enable the table-level logging. This script is executed using the GGSCI command with the Oracle GoldenGate database user. This user must be granted the `ALTER ANY TABLE` privilege prior to executing this script. Once the script completes, this privilege can be removed. Alternatively, edit the script file to use a database user with this privilege. When the OGG database user is originally created, the `ALTER ANY TABLE` privilege is granted at that time. Once the script to enable table level supplemental logging completes, this privilege can be revoked from the OGG user.

Start the GGSCI command utility from the <ORACLE OGG HOME> directory and execute the following command:

```
GGSCI> obey ADD_TRANDATA.txt
```

Exit GGSCI, then connect to the database and revoke the `ALTER ANY TABLE` privilege.

Note: If a table does not have a primary key or any unique indexes defined, you may see a warning message like the following. This is a warning that a 'pseudo' unique key is being constructed and used by Oracle Golden Gate to identify a record. Performance is better if a primary key or unique index is available to identify a record but as we generally cannot add such constraints to an OLTP table when they do not already exists, Oracle Golden Gate creates this pseudo unique key.

WARNING OGG-00869 No unique key is defined for table 'FA_ ASSET_HISTORY'. All viable columns will be used to represent the key, but may not guarantee uniqueness. KEYCOLS may be used to define the key.

Generate Data Definition File on the Source System

As the source and target tables do not match exactly, configure the Replicat process to use a data definition file which contains definitions of the tables on the source system required to map and convert data. The procedure generates a basic DEFGEN.prm file used to create a data definition file. If required, edit this file to reflect your environment. Refer to the Oracle GoldenGate documentation for more details. For example, the DEFGEN.prm file does not leverage the encryption option, so if this or other options are desired, edit the parameter file to enable them.

To edit the DEFGEN.prm file, start the GGSCI command utility from the Oracle GoldenGate home directory. Execute the following command to open and edit the DEFGEN.prm file you copied to <ORACLE OGG HOME>/dirprm:

```
GGSCI>EDIT PARAMS DEFGEN
```

Save and close the file and exit GGSCI, then run the DEFGEN utility. Refer to Oracle GoldenGate documentation for more information about this utility and its execution. The following is an example of executing this command on UNIX:

```
defgen paramfile dirprm/defgen.prm
```

A data definition file is created in the *<ORACLE OGG HOME>*/ folder with the path and name specified using the DEFSFILE parameter. FTP the data definition file to the <ORACLE OGG HOME>/dirdef folder on the remote system using ASCII mode. Use BINARY mode to FTP the data definitions file to the remote system if the local and remote operating systems are different and the definitions file is created for the remote operating system character set.

5.4.5.3 Configure the Target System

Copy all of the files from the 'target' directory on the ODI client to the corresponding directories in the target system.

Copy the following file to the <ORACLE OGG HOME>/dirprm directory in the target system:

- *<REPLICAT_NAME>*.prm where <REPLICAT_NAME> is the value specified when generating the parameter files.

Edit the Replicat Parameter File

By default, the procedure creates a basic set of parameter files that do not include support for a variety of features. For example, the parameter files do not include support for Transparent Data Encryption (TDE) or unused columns. The procedure also does not include the options to encrypt data.If encrypting the data is desired, edit the generated parameter files to add the ENCRYPTTRAIL and DECRYPTTRAIL options. To support such features, edit the generated parameter files using the GGSCI EDIT PARAMS <*parameter file*> command. Also edit the generated param files to implement various tuning options that are specific to the environment. Refer to the *Oracle GoldenGate Reference* guide for details on implementing these options.

Start the GGSCI command utility from the <ORACLE OGG HOME> directory. Execute the following command to edit the Extract parameter file. This should open the Replicat parameter file - this should open the Replicat parameter file you copied to <ORACLE OGG HOME>/dirprm:

```
GGSCI>EDIT PARAMS <REPLICAT_NAME>
```

Save and close the file, and exit GGSCI.

Create a Checkpoint Table (Optional)

The procedure does not account for a checkpoint table in the target system. A checkpoint table is not required but is recommended. If a checkpoint table is desired, follow the steps detailed in Oracle GoldenGate documentation to create a checkpoint table and edit the GLOBALS param file to reference this table.

Start the GGSCI command utility

```
GGSCI> EDIT PARAMS ./GLOBALS
CHECKPOINTTABLE <OGG User>.<Table Name>
```

Save and close the file, then run the following commands:

```
GGSCI> DBLOGIN USERID <OGG User> PASSWORD <OGG Password>
GGSCI> ADD CHECKPOINTTABLE <OGG User>.<Table Name>
```

5.4.6 Setup Step: Start Oracle GoldenGate on Source and Target Machines

Start Oracle GoldenGate on the Source Machine

Use the following command to start the Extract and Data Pump processes on the source system.

```
START MGR
--Start capture on primary database
START <name of Extract process>

--Start pump on primary database
START <name of Data Pump process>
```

Example:

```
START MGR
--Start capture on primary database
START EXT_310

--Start pump on primary database
START DP_310
```

Start Oracle GoldenGate on the Target Machine

Use the following command to start the Replicat process in the target system.

```
START MGR
--Start delivery on target database
START <name of Replicat process>
```

Example:

```
START MGR

--Start capture on primary database
START REP_310
```

Index

A

aggregate tables
  data warehouse tables, 4-4

C

charts
  local seed data, importing, 3-3
  translation tables, process of maintaining, 3-1
  W_LOCALIZED_STRING_G table, adding custom translations, 3-4
  Web catalog strings, about translating, 3-5
cross-module configuration, performing
  dimension keys, resolving, 1-20
  loads, configuring, 1-18
CSV file
  delimiter, 1-17
currency conversion, 4-10
customization, 1-3
  adding an index, 1-20
  and patch installation, 1-3
  category 1, 1-3
  category 2, 1-12
  category 3, 1-15
  overview, 1-1
  Stored Lookups, 1-19

D

data warehouse languages
  configuring base and installed languages, 2-1
data warehouse tables
  aggregate tables, 4-4
  dimension class tables, 4-4
  dimension tables, 4-4
  fact tables, 4-5
  helper tables, 4-5
  hierarchy tables, 4-5
  internal tables, 4-6
  mini-dimension tables, 4-5
  naming conventions, 4-1
  primary data values, 4-10
  staging tables, 4-6
  standard column prefixes, 4-7
  supported list of values, 4-10
  system columns, 4-8
  table types, 4-2
  translation tables, 4-6
delimiter
  for CSV file, 1-17
dimension class tables
  data warehouse tables, 4-4
dimension tables
  business role-based flags, 4-4
  data warehouse tables, 4-4
dimensions
  key resolution, using a lookup, 1-20
  keys, resolving, 1-20

E

extracts, configuring
  extracting additional data, about, 1-16
  extracting data from new source table, 1-16
  extracting new data from existing source table, 1-16

F

fact tables
  data warehouse tables, 4-5

H

helper tables
  data warehouse tables, 4-5
hierarchy tables
  data warehouse tables, 4-5

I

internal tables, 4-6

L

loads, configuring
  about, 1-18

M

metadata
  externalizing strings, 3-4

mini-dimension tables
data warehouse tables, 4-5
multi-language support
about BI Applications domains, 2-2
about dimension translation tables, 2-4
about pseudo-translations, 2-2
introduction, 2-1
translation tables, 2-1

N

naming conventions
data warehouse tables, 4-1

O

Oracle BI Charts
metadata strings, externalizing, 3-4
Oracle BI seed data, updating for non-English locales, 3-2
Oracle BI Presentation layer
process of maintaining translation tables, 3-1
Oracle BI repository
externalizing metadata strings, 3-4
Oracle BI seed data
updating for non-English locales, 3-2

P

patch installation
impact on customizations, 1-3
primary extract and delete mapping
source-archived records, deletion configuration, 1-19
working with, 1-19
primary extract and delete mappings
understanding, 1-18
working with, 1-19

R

records
filtering and deleting, 1-18, 1-19
source-archived records, deletion configuration, 1-19
roadmap
customization, 1-1

S

seed data
importing, 3-3
source-archived records
deletion configuration, 1-19
staging tables
data warehouse tables, 4-6
standard column prefixes, 4-7
standard column suffixes, 4-7
string translations
adding for metadata, 3-5
system columns, 4-8
multi-currency codes for, 4-9

T

table types
data warehouse tables, 4-2
Transactional database
importing locale seed data, 3-3
translation tables
data warehouse tables, 4-6
process of maintaining, 3-1

U

Unicode
local seed data, importing, 3-3
metadata strings, externalizing, 3-4
Oracle BI seed data, updating for non-English locales, 3-2
string translations, adding for metadata, 3-5
translation tables, process of maintaining, 3-1
W_LOCALIZED_STRING_G table, adding custom translations, 3-4
Web catalog strings, about translating, 3-5

W

W_LOCALIZED_STRING_G table, adding custom translations, 3-4
Web Catalog strings
about translating, 3-5
Work Repository
customization and patch application, 1-3