Oracle® Fusion Middleware

ETL Guide for Oracle Business Intelligence Applications

11*g* Release 1 (11.1.1.7)

E37987-01

April 2013

Explains how to set up, perform, and monitor ETL processes.

ORACLE®

Oracle Fusion Middleware ETL Guide for Oracle Business Intelligence Applications, 11*g* Release 1 (11.1.1.7)

E37987-01

Contents

3 Additional Administration Tasks

A Manage Load Plans User Interface Reference

B Knowledge Modules Reference

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 managers and implementors of Oracle BI Applications.

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 Documents

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 Guide

Oracle Fusion Middleware ETL Guide for Oracle Business Intelligence Applications is a new guide for Oracle Business Intelligence Applications 11g Release 1 (11.1.1.7).

1

ETL Overview

This chapter provides an overview of the fundamental concepts related to Oracle BI Applications ETL processes.

This chapter contains the following topics:

- Before You Begin Running ETL Processes
- About ETL Architecture
- About ETL Phases
- About Load Plans
- About the ODI Repository
- About Changed Data Capture
- About Knowledge Modules
- About Reverse Knowledge Modules
- About ETL Roles

Before You Begin Running ETL Processes

Before you begin running Oracle BI Applications ETL processes, you must have completed the installation and setup of Oracle BI Applications, as documented in *Oracle Fusion Middleware Installation Guide for Oracle Business Intelligence Applications*.

You must also have run the domains load plan, which loads source-specific data into Oracle BI Applications Configuration Manager tables. This enables Configuration Manager to display the appropriate source-specific values as choices in drop-down lists for setup objects. For instructions on running the domains load plan, see *Oracle Fusion Middleware Installation Guide for Oracle Business Intelligence Applications*.

About ETL Architecture

Figure 1–1 illustrates the ETL architecture. Typically, the extract-load-transform process has two main steps: The first step is the extract and stage load step, and the second step is the load transform step. The extract and stage load step is generated from a combination of the main interface and the nested temporary interface. The load transform step is generated as a result of the integration knowledge module (IKM). In this example, step 1 issues a SQL statement on the source that joins the GL_SET_OF_ BOOKS table with the HR_ORGANIZATION_INFORMATION table. The join is executed on the source database, and the resulting data is staged. Then, a second join occurs at the load transform stage between the W_DOMAIN_G table and the

temporary stage table, which results in the loading of the stage table W_INV_ORG_DS.

Note that Oracle Database is the only database type supported for the Oracle BI Applications repository schemas and the Business Analytics Warehouse.

Figure 1–1   ETL Architecture
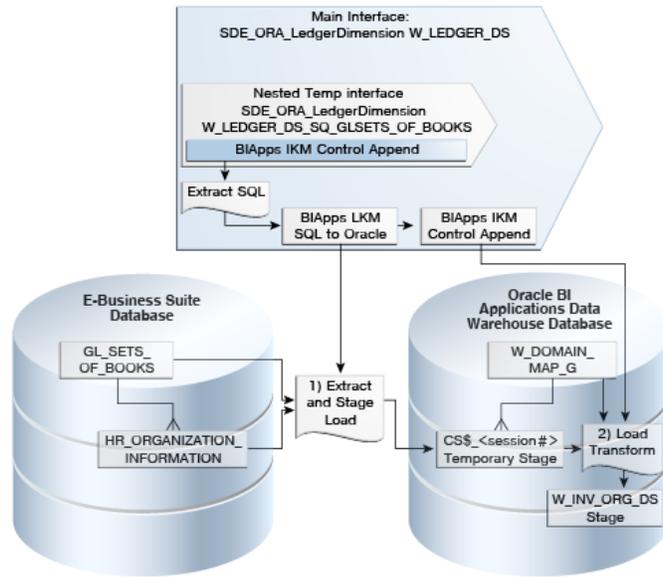
Figure 1–2 shows the general load plan pattern. There are four main stages: The SDE (source dependent extract) tasks extract data from the source dimension and fact tables and load the the data into universal dimension and fact staging tables. The SIL tasks are common and load data from the universal staging tables into the warehouse staging tables. Figure 1–2 depicts a dependency between the dimension table and the fact table. Therefore, the SIL DIM must be executed before the SIL FACT, to resolve the dimension key. The SIL DIM has a database sequence that generates the key, and then the SIL FACT looks up that key when loading the fact staging table.

Figure 1–2   Load Plan Pattern
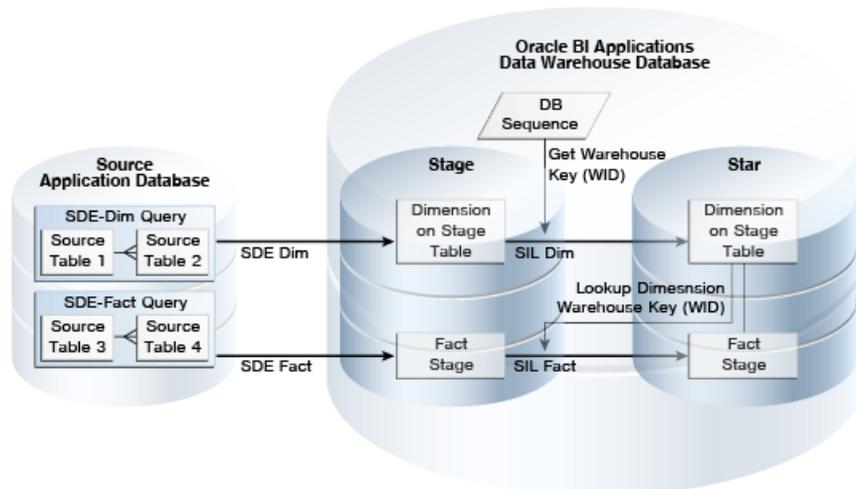
About ETL Phases

Oracle BI Applications ETL processes include the following phases:

- **SDE.** SDE stands for Source Dependent Extract. In the first phase, SDE tasks extract data from the source system and stage it in staging tables. SDE tasks are source specific.

- **SIL**. SIL stands for Source Independent Load. Load tasks transform and port the data from staging tables to base fact or dimension tables. SIL tasks are source independent.

- **PLP.** PLP stands Post Load Process. PLP tasks are only executed after the dimension and fact tables are populated. A typical usage of a PLP task is to transform data from a base fact table and load it into an aggregate table. PLP tasks are source independent.

About the ODI Repository

The ODI Repository for Oracle BI Applications comprises two repositories:

- **Master Repository**. Topology of resources, security, version management. A master repository is usually associated with multiple work repositories, but work repositories are always attached to a single Master repository

- **Work Repository**. Contains data models and projects. This is the development and execution repository.

The master and work repositories must reside in the same database schema, and the database type must be Oracle. Both the master and work repositories are set up during the Oracle BI Applications installation process.

Note that the default ODI repository ID is 500. This ID is part of the internal ID for every object that is created within the ODI repository. Having a repository ID greater than or equal to 500 is critical to ensure that the objects you create do not overlap with any current or future Oracle BI Applications objects. If you change the default ODI repository ID, make sure the new value is greater than 500.

About Load Plans

A load plan is an executable object that comprises and organizes the child objects (referred to as *steps*) that carry out the ETL process. A load plan is made up of a sequence of several types of steps. Each step can contain several child steps. Depending on the step type, the steps can be executed conditionally, in parallel or sequentially.

You define a load plan in Oracle BI Applications Configuration Manager by selecting a data source and one or more fact groups. This selection determines which steps need to be performed during the ETL process. Each fact group belongs to a specific functional area or areas that are associated with one or more offerings, which, in turn, are related to a data server. A transactional data source is associated with one or more data servers.

After you define the load plan, you then generate it to build it in the ODI repository. You then execute the load plan to perform the ETL process.

For more information about working with Oracle BI Applications load plans, see Chapter 2, "Managing Load Plans." For information about the topic of load plans in the context of Oracle Data Integrator, see *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

About Changed Data Capture

Oracle BI Applications has two ETL modes for loading data into the Oracle Business Analytics Warehouse: full and incremental.

During a full load, Oracle BI Applications extracts:

- All records from tables that are sources for dimension tables.

- Records created after an "Initial Extract Date" from tables that are sources for fact tables. The Initial Extract Date defines a cut-off so that not all records are loaded into the data warehouse. You set the Initial Extract Date value for each data source in the Oracle BI Applications Configuration Manager.

An ETL process can extract a record from a single table or from multiple tables. When a record is the result of joining multiple tables, one of these tables is identified as the *base* table, which defines the granularity of the record. When extracting fact records, Oracle BI Applications only compares the "Created Date" of the base table to the Initial Extract Date.

During an incremental load, Oracle BI Applications extracts records that have changed or were created after a "Last Extract Date." This is done by comparing the Last Extract Date value to a "Last Updated Date" (or LUD) type column in the source table. If the source table does not have such a column, Oracle BI Applications extracts all records from that table. The Last Extract Date is a value that is calculated based on the last time data was extracted from that table less a "Prune Days" value. The Prune Days parameter is used to extend the window of the ETL extract beyond the last time the ETL actually ran. This is to ensure records that may have somehow been missed in a prior ETL are picked up in the next ETL. Records can be missed in an ETL process when a record is being updated while the ETL process is running and was not committed until after the ETL completed.

You set the Prune Days parameter value in Oracle BI Applications Configuration Manager. Setting a small value means the ETL will extract fewer records, thus improving performance; however, this increases the chances that records are not detected. Setting a large number is useful if ETL runs are infrequent, but this increases the number of records that are extracted and updated in the data warehouse. Therefore, you should not set the Prune Days value to a very large number. A large Prune Days number can also be used to trigger re-extracting records that were previously processed but have not changed. The value for Prune Days should never be set to 0.

As stated above, an ETL process can extract a record from a single table but more commonly extracts records that are the result of joining multiple tables. When extracting from multiple tables, one table is identified as the base table, which defines the granularity of the record. When there is a change in the base table, an extract is triggered for the record. However, there can be cases where a change occurs in a non-base table but not in the base table itself. If a change occurs in a non-base table and this should trigger an extract of the record, these tables are referred to as "auxiliary" tables. Thus, when determining if a record should be extracted, Oracle BI Applications compares not only the LUD column in the base table but the LUD columns in all auxiliary tables. If the LUD column changed in any of these tables, the record is extracted. If changes can occur in a table that is not relevant to triggering an extract, this table's LUD column is not compared in the incremental filtering logic.

About Knowledge Modules

Knowledge Modules (KMs) implement different tasks within the Oracle Business Analytics Warehouse system. The different types of KMs are:

- **Reverse-engineering (RKM)**. Used for reading the table and other object metadata from source databases and to import tables, columns, and indexes into a model. For more information about RKMs, see "About Reverse Knowledge Modules".

- **Loading (LKM)**. Used for efficient extraction of data from source databases for loading into a staging area (database-specific bulk unload utilities can be used where available).

- **Integration (IKM).** Used to load data into a target with different strategies, for example, slowly changing dimensions and insert/update strategies.

- **Check (CKM)**. Used to validate and cleanse data.

- **Journalizing (JKM)**. Used to record the new and changed data within either a single table or view or a consistent set of tables or views.

- **Service (SKM)**. Exposes data in the form of Web services.

For a detailed description of the KMs available with Oracle BI Applications, see Appendix B, "Knowledge Modules Reference."

About Reverse Knowledge Modules

Oracle BI Applications uses the ODI reverse engineering process to populate the repository with metadata from the source system. RKMs retrieve metadata from data storage and load it into the repository. For example, RKMs detects the description of tables, columns, data types, constraints, and comments from a database to load the repository. RKMs support various technologies, such as databases, XML files, and various types of flat files. You can also use RKMs to retrieve non-standard metadata from databases or applications, such as Oracle E-Business Suite, Siebel CRM, PeopleSoft, and so on.

The RKM role is to perform customized reverse engineering for a model. The RKM handles connecting to the application or metadata provider then transforming and writing the resulting metadata into the ODI repository. The metadata is written into temporary tables, and then the RKM calls the ODI API to read from these tables and write to the ODI metadata tables of the Work repository in incremental update mode.

Note that the Oracle BI Applications ODI repository contains the relevant source data models. Therefore, you would need to run an RKM only if you have customized tables in the source system and want to import these changes to the ODI repository. For more information about customizing tables and tasks, see *Oracle Fusion Middleware Administrator's Guide for Oracle Business Intelligence Applications*.

About Multi-Source Environments

Oracle BI Applications supports the loading of data into the Oracle Business Analytics Warehouse from multiple source systems as long as the source systems are different. Oracle BI Applications does not support multiple instances of the same source system. For example, your environment could have a PeopleSoft 9.0 source and a Oracle EBS 11.5.0 source but not two instances of the Oracle EBS 11.5.10 source.

Keep in mind that multi-source ETL processes are different from single-source ETL processes in that you could potentially have both sources being used to load the same fact groups and dimensions.

Multi-Source Example

Consider a scenario in which you have a PeopleSoft 9.0 source and an Oracle EBS 11.5.10 source, both loading the target fact table ASTXACT_FG. Loading this target table includes three serial steps:

1. Initialize Target Table (ASTXACT_FG).

2. Load Target Table (ASTXACT_FG).

3. Finalize Target Table (ASTXACT_FG).

Figure 1–3 shows these load plan steps in ODI Studio.

Figure 1–3   Multi-Source Load Plan

Data from both sources, PeopleSoft 9.0 and Oracle EBS 11.5.10, is being loaded into the target table (ASTXACT_FG) for the first time, which means a full load will occur; and, generally, when a full load occurs, the target table is truncated. In this case, the load plans can run in parallel because the data sources are not dependent on each other; however, the load from the second source should not truncate the target table if it already contains data from the first source. This issue is resolved as follows: The Initialize Target Table step truncates the target table. The Load Target Table step, in turn, has two child parallel steps. Each of these loads the target from each source system. Finally, the Finalize Target Table step creates any indexes and analyzes the table. Thus, the generated load plan ensures that the table is truncated only at the required time (in this case, only once before any source system loads it).

Note that you can have separate load plans for each source, but load plans should not run in parallel.

> **Caution:** Oracle BI Applications Configuration Manager and ODI will not stop load plans from running in parallel; however, it is recommended that you do not do so because of the following reasons:
>
> - All load plans truncate target tables upon the initial load. If you run load plans in parallel, one load plan can truncate the data loaded by the preceding load plan.
>
> - The mappings from SILOS onwards are common and not based on the source system. If load plans are run in parallel, you can have a situation in which only partial data is loaded from the second source due to the data that was loaded from the first source. To resolve this issue, you need to make sure that the first load plan completes successfully before the second load plan runs.
>
>   In other words, when loading a fact table, the fact table could be connecting with multiple dimension and lookup tables to get the final data. When load plans are running parallel, some of the dimension, lookup, and staging tables could also have data from a second source. This could lead to some lookups and dimensions not returning appropriate value for the second source, since they have not yet been completely loaded

About ETL Roles

Oracle BI Applications has two duty roles for ETL operations:

- Load Plan Operator Duty
- Load Plan Administrator Duty

Oracle BI Applications has the following additional duty roles:

- BI Applications Administrator Duty
- BI Applications Functional Developer Duty
- BI Applications Implementation Manager Duty

Access to Configuration Manager and Functional Setup Manager is controlled through these duty roles.

The security administrator must grant the appropriate duty roles to a user based on the user's job responsibilities. For information on the Configuration Manager and Functional Setup Manager screens that each duty role has access to, see *Oracle Fusion Middleware Security Guide for Oracle Business Intelligence Applications*.

The BI Applications administrator, load plan operator, and load plan administrator users will require appropriate access to ODI. In addition to these users being created in the LDAP system, these users must also be created in the ODI Repository, and they must be granted the Supervisor profile or an appropriate ODI profile. The BI Applications administrator must be granted the Supervisor role in ODI. Work with your security administrator to obtain the appropriate duty roles.

For more information about managing security in ODI, see *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator* security guide.

2

Managing Load Plans

This chapter provides information about managing load plans for Oracle BI Applications using Oracle BI Applications Configuration Manager.

The tasks for managing load plans are usually performed by either ETL developers or ETL operators. The tasks in this section are grouped by the role that generally performs the task, as listed below.

ETL Developer Tasks

- Defining Load Plans
- Copying Load Plans
- Editing Load Plans
- Generating Load Plans
- Scheduling Load Plans
- Executing Load Plans

ETL Operator Tasks

- Monitoring Load Plan Runs
- Restarting Load Plans
- Stopping Load Plans

Overview of Load Plan Life Cycle

A load plan life cycle comprises the following phases:

- **Phase 1: Define load plan**

  In this phase, you define load plan properties in the Oracle BI Applications Configuration Manager, including selecting a data source and one or more fact groups. This selection determines which steps need to be performed during the ETL process.

- **Phase 2: Generate load plan**

  In this phase, you launch a generation process from Oracle BI Applications Configuration Manager that propagates the load plan properties to the ODI Repository, where the load plan is built.

- **Phase 3: Execute load plan**

  In this phase, you start a load plan run from Oracle BI Applications Configuration Manager, which executes the steps of the load plan. Executing a load plan creates a

load plan instance and a first load plan run. If a run is restarted a new load plan run is created under this load plan instance. Each execution attempt of the load plan instance is preserved as a different load plan run in the log.

■ **Phase 4: Monitor load plan**

In this phase, you monitor the load plan run in the Load Plan Details page of the Oracle BI Applications Configuration Manager. The Load Plan Details page provides a view of the ODI Repository through Oracle Data Integrator Console.

Defining Load Plans

Follow this procedure to define a load plan.

To define a load plan:

1. In the Tasks pane of Oracle BI Applications Configuration Manager, select **Manage Load Plans**, which appears under the Load Plans Administration heading.

   The Manage Load Plans page is displayed.
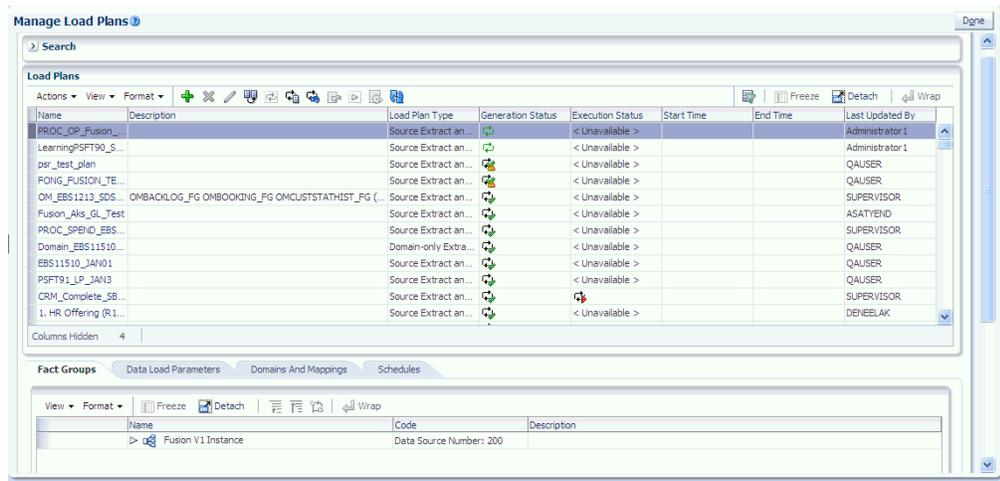
2. In the Load Plans toolbar, click the **Add** icon.

   The Create Load Plan page is displayed.

3. On the first page of the Create Load Plan series, specify the following information:

Field	Description
Name	Enter a unique name for the load plan.
Description	(Optional) Enter additional information about the load plan.

Field	Description
Load Plan Type	Select a load plan type. Possible values are the following:
	■ **Source Extract (SDE)** - Includes only those tasks that extract from the source and loads data into staging tables.
	■ **Source Extract and Load (SDE, SIL and PLP)** - Includes all tasks to extract from the source and load the data warehouse tables.
	■ **Warehouse Load (SIL and PLP)** - Includes only those tasks that extract from the staging tables and load the data warehouse tables.
	Note that it may be useful to generate separate source-specific and data warehouse-specific load plans. By decoupling the load plans, this allows scheduling a source-specific load plan to run during hours when most users are not using the source system and scheduling a separate load of the data warehouse when most users are not using the data warehouse.
	■ **Domain-only Extract and Load (SDE and SIL)** - Includes all tasks required to extract domain-related records from the source and load the data into the domain-related tables in the Oracle Business Analytics Warehouse. Note that domains are used extensively in Oracle BI Applications Configuration Manager and several properties must be configured before executing a regular load plan. These properties depend on the domain values found in the transactional database.
Source Instances	Select the data sources from which the fact groups will be selected.

4. Click **Next.**

   The second page of the Create Load Plan series is displayed.

5. In the Available Fact Groups tab, select the fact groups you want to include in the load plan definition.

   Note that fact groups may belong to a hierarchy of fact groups. You can select only the top level parent fact group and not a child fact group.

   A load plan must contain at least one fact group, and multiple fact groups may be selected from one or more data sources.

6. Click **Save**. A submenu is displayed with the following options

   ■ Click **Save** to save the load plan. After a load plan is saved, it is displayed in the Load Plans master list.

   ■ Click **Save and Generate Load Plan** to save the load plan and immediately generate it.

Copying Load Plans

Follow this procedure to make a copy of an existing load plan.

To make a copy of an existing load plan:

1. In the Tasks pane of Oracle BI Applications Configuration Manager, select **Manage Load Plans**, which appears under the Load Plans Administration heading.

   The Manage Load Plans page is displayed.

2. In the Load Plans master list, select the load plan you want to copy.

3. In the Load Plans toolbar, click the **Copy** icon.

   The Copy Load Plan page is displayed.

4. On the first page of the Copy Load Plan series, specify the following information:

Field	Description
Name	Enter a unique name for the load plan.
Description	(Optional) Enter additional information about the load plan.
Load Plan Type	(Optional) Select a load plan type. Possible values are the following:
	■ Source Extract (SDE)
	■ Source Extract and Load (SDE, SIL and PLP)
	■ Warehouse Load (SIL and PLP)
	■ Domain-only Extract and Load (SDE and SIL)
Source Instances	(Optional) Select the data sources from which the fact groups will be selected.

5. Click **Next**.

   The second page of the Copy Load Plan series is displayed.

6. In the Available Fact Groups tab, select the fact groups you want to include in the load plan definition.

   Note that fact groups may belong to a hierarchy of fact groups. You can select only the top level parent fact group and not a child fact group.

   A load plan must contain at least one fact group, and multiple fact groups may be selected from one or more data sources.

7. Click **Save**. A submenu is displayed with the following options

   ■ Click **Save** to save the load plan. After a load plan is saved, it is displayed in the Load Plans master list.

   ■ Click **Save and Generate Load Plan** to save the load plan and immediately generate it.

Editing Load Plans

Follow this procedure to edit an existing load plan.

To edit an existing load plan:

1. In the Tasks pane of Oracle BI Applications Configuration Manager, select **Manage Load Plans**, which appears under the Load Plans Administration heading.

   The Manage Load Plans page is displayed.

2. In the Load Plans master list, select the load plan you want to edit.

3. In the Load Plans toolbar, click the **Edit** icon.

   The Edit Load Plan page is displayed.

4. You can edit the following properties:

Field	Description
Name	Enter a unique name for the load plan.
Description	(Optional) Enter additional information about the load plan.
Load Plan Type	(Optional) Select a load plan type. Possible values are the following: ■ Source Extract (SDE) ■ Source Extract and Load (SDE, SIL and PLP) ■ Warehouse Load (SIL and PLP) ■ Domain-only Extract and Load (SDE and SIL)
Source Instances	(Optional) Select the data sources from which the fact groups will be selected.

5. Click **Next**.

    The second page of the Edit Load Plan series is displayed.

6. In the Available Fact Groups tab, select the fact groups you want to include in the load plan definition.

    Note that fact groups may belong to a hierarchy of fact groups. You can select only the top level parent fact group and not a child fact group.

    A load plan must contain at least one fact group, and multiple fact groups may be selected from one or more data sources.

7. Click **Save**. A submenu is displayed with the following options

    ■   Click **Save** to save the load plan. After a load plan is saved, it is displayed in the Load Plans master list.

    ■   Click **Save and Generate Load Plan** to save the load plan and immediately generate it.

Generating Load Plans

When you generate a load plan, the load plan is built in the ODI Repository. A load plan must be generated successfully before it can be executed.

Note: Load plans must be generated serially or the process will fail. Do not launch a second load plan generation if one is already underway. You must wait until the first generation process completes before you launch the next generation process.

To generate a load plan:

1. In the Tasks pane of Oracle BI Applications Configuration Manager, select **Manage Load Plans**, which appears under the Load Plans Administration heading.

    The Manage Load Plans page is displayed.

2. In the Load Plans master list, select the load plan you want to generate.

3. In the Load Plans toolbar, click the **Generate** icon.

    The following icons show the possible generation statuses that appear in the Generation Status column in the Load Plan master list. Click the **Refresh** icon to refresh the display.

Generation Status Icon	Description
	Starting
	In Progress
	Succeeded
	Failed

You can execute a load plan or schedule it for execution after it has been successfully generated.

Scheduling Load Plans

Follow this procedure to schedule a load plan for execution.

To schedule a load plan for execution:

1. In the Tasks pane of Oracle BI Applications Configuration Manager, select **Manage Load Plans**, which appears under the Load Plans Administration heading.

   The Manage Load Plans page is displayed.

2. In the Load Plans list, select the load plan you want to schedule.

3. Select the **Schedules** tab.

4. Click **Add** in the Schedules tab toolbar.

   The Schedule Load Plan dialog is displayed.

5. Specify the following information:

Field	Description
Context	The ODI context to be used when the load plan is run. Note that **Global** is the only supported context.
Local Agent	The ODI local agent to be used when the load plan is run.
Log Level	The level of logging information to retain. The Oracle BI Applications Configuration Manager uses Oracle Diagnostic Logging. For information about managing log files and diagnostic data, see *Oracle Fusion Middleware Administrator's Guide*.
Status	Status of the schedule. Possible values are the following ■ Active ■ Inactive ■ Active for the period

Field	Description
Recurrence	Frequency of occurrence. Possible values are the following: ■ On Agent Startup ■ Simple ■ Hourly ■ Daily ■ Weekly ■ Monthly (day of the month) ■ Monthly (week day) ■ Yearly Depending on the Recurrence option you select, options for selecting the date and time are dynamically displayed.

6. Click **Schedule**.

Executing Load Plans

Follow this procedure to execute a load plan. Note the following points:

■ You can have separate load plans for each source, but load plans should not run in parallel

■ You can only execute a load plan if it was successfully generated. See "Generating Load Plans" for instructions.

To execute a load plan:

1. In the Tasks pane of Oracle BI Applications Configuration Manager, select **Manage Load Plans**, which appears under the Load Plans Administration heading.

   The Manage Load Plans page is displayed.

2. In the Load Plans list, select the load plan you want to execute.

3. In the Load Plans toolbar, click the **Execute** icon.

   The Execute Load Plan Dialog is displayed.

4. Specify the following information:

Field	Description
Context	The ODI context to be used when the load plan is run. Note that **Global** is the only supported context.
Local Agent	The ODI local agent to be used when the load plan is run.
Oracle Data Integrator Work Repository	The name of the ODI Work repository.

5. Click **OK**.

   The following icons show the possible execution statuses that appear in the Execution Status column in the Load Plan master list. Click the **Refresh** icon to refresh the display.

Generation Status Icon	Description
	Starting
	In Progress
	Succeeded
	Failed
	Not Executed

6. (Optional) Click the **Show Execution Status Details** icon to display detailed information about the status of the execution.

Monitoring Load Plan Runs

You can monitor a load plan run by viewing the execution status information on the Load Plan Execution Details page of Oracle BI Applications Configuration Manager.

To view load plan execution details:

1. In the Tasks pane of Oracle BI Applications Configuration Manager, select **Manage Load Plans**, which appears under the Load Plans Administration heading.

   The Manage Load Plans page is displayed.

2. In the Load Plans master list, select the load plan whose run you want to view.

3. In the Load Plans toolbar, click the **Show Execution Status Details** icon.

   The Oracle Data Integrator Console is displayed.

4. Log into the Oracle Data Integrator Console by entering an appropriate **User ID** and **Password**.

   The Load Plan Details page is displayed.

   For a description of the information displayed on this page, see "Load Plan Details Page".

Restarting Load Plans

This section provides information about restarting load plans after a failure. It includes the following sections:

- Overview of Load Plan Restartability
- About Restartability Grain
- Restarting Load Plans

- Troubleshooting Load Plans

- Alternate Options for Restarting Load Plans

- Related Features and Considerations

Overview of Load Plan Restartability

When you run ETL to load data from a source system into the Oracle Business Analytics Warehouse (OBAW), it is possible that you may need to restart the ETL load after a failure. This section details all such cases, explains the options available for restart, and describes the implications of using each of those options.

Examples of circumstances and reasons for load plan failure include:

- Problem with access either to source or target database due to network failure or expired or otherwise incorrect user names and passwords.

- Failure of ODI agent.

- Problem with space or storage. Able to connect to source or target database but the query fails to run due to lack of temp space, disk space, and so on. For files it could due to inadequate space where the file needs to be placed.

- Problem with data, for example incorrect data with lengths larger than the target column can hold, or null values in Not Null columns.

After such a failure during ETL, to avoid restarting the entire load plan after a failure, which would require inefficient re-runs of all ETL tasks, you must restart the load from the same point in its execution once the cause of failure has been diagnosed and resolved.

About Restartability Grain

When you restart a load plan after a failure, you may not restart again from the exact point of failure, depending on where it occurred and on dependencies between load plan steps. The point of restartability is that the end result of the load plan execution is the same regardless of any load plan failure.

The following example describes one such dependency-driven requirement for re-running a step which has already completed: In a load plan with two steps, the first step truncates the table and the second inserts records into the table, intermittently committing the records. The load plan is run and fails at the second step due to a space issue. After the issue is resolved, restarting the load plan from the second step would be incorrect because the target has some inserted rows. Restart should instead begin with the first step so that the target table is truncated again and newly inserted data does not cause any duplicates.

To maintain data integrity in the case of restart, the grain would vary depending on the location in the step hierarchy of the failed step and on the Restart setting for the step in the Load Plan Manager.

Within the Steps Hierarchy in Load Plan Manager, you can view and edit the Restart setting of a step in the Restart column. The default settings for different steps in the hierarchy support data integrity in restarts:

- Root steps are set to 'Restart From Failure' if Serial and 'Restart from failed Children' if Parallel.

- Sub steps are set to 'Restart From Failure' if Serial and 'Restart from failed Children' if Parallel.

■ Scenario steps are set to 'Restart from Failed Step'

The below examples highlight the implications for each type of load plan step.

Serial Load Plan Step

Serial steps are represented by a horizontal icon in the Steps hierarchy in Load Plan Manager and by default have a Restart setting of Restart from Failure. In a case where the load plan fails when running such a step to load a Dimension Group with multiple serial sub-steps loading individual dimensions, the load plan on restart would start from the individual sub-step that failed. Any successfully completed serial sub-steps would not be run again.

Parallel Load Plan Step

Parallel steps are represented by a vertical icon in the Steps hierarchy in Load Plan Manager and by default have a Restart setting of Restart from Failed Children. In a typical run, a parallel step with five parallel sub-steps under it would have all five sub-steps executed in parallel, subject to free sessions being available. If two of those five steps completed and then the load plan were to fail, when the load plan was restarted all the steps that did not complete or failed would be started again.

Scenario Step

At the lowest order in any load plan are the scenario steps. While the parent steps, whether serial or parallel, are used to set the dependencies, the scenario steps are those which load the tables. A scenario step in turn could have one or more sub-steps, corresponding to the number of steps inside the package.

In the case of a scenario step failure during execution, consider that the scenario step may have multiple steps, all under the same session in Operator log but identified with different step numbers: 0, 1, 2, and so on. In the case of restart, the scenario would execute from the failed parent scenario step, re-running all sub-steps.

> **Note:** If you use the Load Plan Generator to generate any load plan it would automatically conform to the above standard. If you are manually altering a generated load plan or creating a new load plan without using Load Plan Generator, then you should ensure that you conform to the above standard.

Restarting Load Plans

Use ODI Studio or ODI Console to restart a load plan. This section describes how to restart load plans. It includes the following sections:

■ Restarting Using ODI Studio

■ Restarting Using ODI Console

Restarting Using ODI Studio

Follow this procedure to restart a load plan using ODI Studio. The restart option is enabled only on the last run for a load plan. A load plan can be restarted any number of times and each time it progresses from the last failure.

To restart a load plan using ODI Studio:

1. In ODI Operator, navigate to the Operator log and select the last failed run for a load plan.

2. Double-click the load plan run and select the Restart Option. You can also right-click the last run in the Operator log and select Restart.

Restarting Using ODI Console

Follow this procedure to restart a load plan using ODI Console. The restart option is enabled only on the last run for a load plan. A load plan can be restarted any number of times and each time it progresses from the last failure.

To restart a load plan using ODI Operator:

1. In ODI Console, navigate to Runtime > Sessions/Load Plan Executions and select the load plan execution that has failed.

2. Click the Restart button. The Restart button is displayed only when the selected load plan is the most recent run of the load plan.

Troubleshooting Load Plans

A load plan must be restarted when it has stopped with an error. An alternate case where restart may be required is when a load plan is not doing anything at all, for example when a load plan is executed and nothing has changed after 30 minutes. The following checklist can be used to assist in troubleshooting a non-responsive load plan.

1. Check the maximum number of sessions set to run against the agent. In ODI Operator, verify that the number of sessions running is equal to the maximum. If so, then the other sessions are waiting for the running sessions to complete. Proceed to the next step.

2. Clean out stale sessions. Stale sessions are sessions that are incorrectly left in a running state after an agent or repository crash. If an agent crashes or loses its connection the repository after it has started a session, it is not be able to update the status of the session in the repository, and such a session becomes stale. Until the stale session is cleaned it shows up as running in the repository but actually is not.

   Stale sessions are cleaned in multiple ways. Some examples are listed below:

   - You can manually request specific agents to clean stale sessions in Operator Navigator or Topology Navigator.

   - Stale sessions are cleaned when you restart an agent.

   - When an agent starts any new session, it checks for and resolves stale sessions. However, if the agent has lost connection to the repository, then it cannot clean stale sessions.

3. Check if the agent is alive. To test the agent to see if it is running and still has a connection to the repository, open it in the Topology Navigator in ODI Studio and select the Test tab. If the agent test fails, then restart the agent after fixing the issue.

4. Verify that the ODI Repository and the server hosting it are running and have not experienced a failure.

5. If your load plan is in error and you have verified all of the above, then restart the Load plan.

Alternate Options for Restarting Load Plans

This section describes alternate ways to approach restarting failed load plans. It includes the following sections:

- [Using Mark as Complete](#)

- [Running a Scenario Standalone](#)

Using Mark as Complete

In most cases the load plan restart method described earlier in this section is the recommended approach. This approach ensures data integrity and leaves no scope for manual error. However, at times you may want to run a load plan step manually. For example, if a step is inserting duplicate records which are causing failure, rerunning the step would still insert duplicates. In such a case, you may need to manually correct the data outside of the load plan and then skip that step when you restart the load plan. For this kind of situation, you can use the Mark as Complete option.

When you mark a load plan step as complete, it ensures that when the load plan is restarted, the marked step is not executed. It is then the responsibility of the person making this setting to ensure that the load for that step is carried out outside the load plan.

To mark a step as complete, right-click the step and select Mark As Complete. This can be done at the scenario step or at any step higher than that.

Marking a step complete at a higher level in the step hierarchy would mean that none of the child steps under that parent step would be executed upon load plan restart, even if they are otherwise eligible. For this reason, marking a step as complete should be treated as an advanced task and must be done only with a full understanding of its impact. There is no single recommendation that pertains in all cases, so the setting must be done carefully and only on a case-by-case basis.

Running a Scenario Standalone

When you are monitoring a load plan, you may not completely know how to fix a scenario step failure, but may wish to use the 'mark as complete' option for the failed scenario step instead of waiting for complete resolution. This prevents a step failure from precluding an entire load plan completing, while allowing you to inform the ETL team about the failed scenario step and work on a resolution. The ETL team might then fix the scenario and want to run it standalone outside the load plan to complete the load.

As in marking a step as complete, running a scenario standalone should be treated as an advanced task and the person running the scenario must be aware of the following:

- A scenario run outside of a load plan by itself invokes the Table Maintenance process. This could, depending on the setting, truncate the table before the load.

  To understand this, consider that when a scenario is run inside a load plan table maintenance tasks are carried out as explicit steps (the parent step name would be either Initialize or Finalize). The scenario by itself does not call the Table Maintenance process when run from within the load plan. Rather, this is controlled by the EXECUTION_ID variable, which is set to the load plan instance ID. If this variable has a value greater than 0 when a scenario is run, the Table Maintenance process is not invoked, as would be the case when a scenario is run from within a load plan with an instance ID. However, if this variable does not have a value greater than 0, then the scenario invokes the Table Maintenance process. This is the case when a scenario is run outside the load plan. If you set a value for the EXECUTION_ID when invoking the scenario from outside a load plan, the table maintenance steps would not be called.

- A scenario step could have many variable values set, either dynamically in the case of a refresh variable or explicitly by overriding its value at that scenario step

in the load plan. When running a scenario outside the load plan, all the scenario variables would have only their default values. For this reason, care should be taken to set the variables appropriately before calling a scenario from outside the load plan. You can check the variable values that are present in the load plan by looking at the Operator log, provided the log level was set to 6 when the load plan ran. The Oracle BI Applications Configuration Manager uses Oracle Diagnostic Logging. For information about managing log files and diagnostic data, see *Oracle Fusion Middleware Administrator's Guide*.

Related Features and Considerations

This section lists some of the Oracle BI Applications features that are related to restartability and describes some related considerations.

Using CKM to Filter Erroneous Data

If a scenario step is failing due to bad source data, it may sometimes be desirable to enable the CKM option to load the valid records and route the error records to a separate table. Examples of situations where this may be appropriate are the load of null values when they should have a value or data lengths longer than allowed target column lengths. Once the load completes, you could correct the erroneous data and have it automatically picked up in a subsequent load.

> **Note:** Use of CKM can slow the load considerably because every record and column could potentially be checked before loading. For this reason, this is not an option that you want to turn on across the entire load plan.

Regenerating a Scenario During a Load Plan Execution

Consider a case where a load plan is started and fails at a scenario step. You fix the issue and regenerate the scenario, then restart the load plan and may expect it to pick the new scenario, but this is not what happens. If a load plan has been started and a scenario regenerated, the regenerated scenario code is not picked up when the load plan is restarted. To force the regenerated scenario to be picked up, you have two options:

- Start a new load plan run, accepting the overhead associated with restarting the load from the beginning.

- Run the regenerated scenario as stand-alone outside the load plan, marking that scenario step as complete in the load plan before restarting the load plan. Refer to "Using Mark as Complete" and "Running a Scenario Standalone" for implications of using these options.

Restarting Long Running Jobs

Consider a case where you have a scenario that takes two hours to run. The scenario fails at the insert new rows step, after loading the C$ and I$ steps. On restart, the scenario attempts to reload the C$ again. You instead want it to restart from the insert new rows steps only.

This use is not supported. The restartability mechanism has been put in place in such a way that restarting a load plan is all you need to do. You do not need to clean up any data in between load plan executions, because the data is committed to the target table only after all the Knowledge Module steps are successful. If the load fails before complete success, no data is committed to that specific target table as part of the failed

session. (Note: C$ and I$ tables are created afresh on restart and hence data to these tables would be committed in between).

> **Note:** New C$ and I$ tables are created on restart and hence data to these tables would be committed in between load plan start and restart.

Stale C$ and I$ Tables

On restart, new C$ and I$ tables are created, but since the previous load did not complete, these tables from the previous session are not dropped. The load plan generated using Load Plan Generator has a step at the end of the load plan called Clean Stale Work Tables which takes a variable called ETL_DTOP_STG_OLDER_ THAN_DAYS whose default value is 30 days. When this step runs, it drops any C$ and I$ tables that are older than the specified variable value.

> **Note:** C$ and I$ tables are useful tables when you want to debug a failed scenario. It might not be advisable to set the ETL_DTOP_STG_ OLDER_THAN_DAYS value as too small—for example, one day—as you might lose valuable information for debugging.

Stopping Load Plans

You can stop a load plan run from by accessing the Load Plan Execution page in Configuration Manager (click Show Execution Status Details in the toolbar) or from ODI Studio.

To stop a load plan run from ODI Studio:

1. In Operator Navigator, select the running or waiting load plan run to stop from the Load Plan Executions accordion.

2. Right-click, and select **Stop Normal** or **Stop Immediate**.

   - **Stop Normal** - In normal stop mode, the agent in charge of stopping the load plan sends a Stop Normal signal to each agent running a session for this load plan. Each agent will wait for the completion of the current task of the session and then end the session in error. Exception steps will not be executed by the load plan and once all exceptions are finished the load plan is moved to an error state.

   - **Stop Immediate** - In immediate stop mode, the agent in charge of stopping the load plan sends a Stop Immediate signal to each agent running a session for this load plan. Each agent will immediately end the session in error and not wait for the completion of the current task of the session. Exception steps will not be executed by the load plan and once all exceptions are finished the load plan is moved to an error state.

3. In the Stop Load Plan dialog, select an agent to stop the load plan.

4. Click **OK**.

   For more information about stopping load plan runs from ODI Studio, see *Oracle Fusion Middleware Developer's Guide for Oracle Data Integrator*.

3

Additional Administration Tasks

This chapter provides information administration topics related to Oracle BI Applications ETL processes.

This chapter contains the following topics:

- Creating ETL Tables for Siebel Sources
- Generating DDL Warehouse Scripts
- Logging and Diagnostics

Creating ETL Tables for Siebel Sources

If your source system is Siebel and you are deploying one of the following offerings, you need to create S_ETL tables in the Siebel database:

- Oracle Marketing Analytics
- Oracle Price Analytics
- Oracle Sales Analytics
- Oracle Service Analytics

To create the S_ETL tables, you generate a DDL script, which then must be provided to the Siebel DBA, who will need to create the necessary tables in the Siebel database.

Note that the DB user used for the ETL connection will require all DML privileges (SELECT, INSERT, DELETE, UPDATES) and DDL (TRUNCATE) privileges to S_ETL tables.

To generate a DDL script to create S_ETL tables for Siebel sources:

1. Launch ODI Studio, and display the Designer navigator.

2. In the Projects editor, expand the following folders: BIApps Project, Components, Sources, Siebel, Oracle, Generate Siebel DDL, Packages, Generate Siebel DDL, Scenarios.

3. Right-click **GENERATE_SIEBEL_DDL Version 001**, and then click **Execute**.

4. Select the Context, ODI Agent, and Log Level, and then click **OK**.

5. Specify the following information:

Option	Description
CREATE_SCRIPT_FILE	Select **Latest Value** and Enter **Y** in the Value field.
REFRESH_MODE	Enter **FULL**.

Option	Description
CHAR_CLAUSE	Provided for Unicode support. If set to **Y**, the CHAR clause will be included in the DDL.
RUN_DDL	Enter **N**.
SCRIPT_LOCATION	Enter the path where you want the script to be created.
	The file name will be similar to BIA_Siebel_Schema_DDL_<run ID>.sql.
TABLE_MASK	The default value % will compare all tables.

Generating DDL Warehouse Scripts

The Business Analytics Warehouse tables are automatically deployed during the installation process when the Business Analytics Applications Suite Repository Creation Utility (RCU) executes a shipped DDL script. The RCU does not prompt for which tablespace to assign to the individual tables and related indexes nor does it provide a mechanism for you to alter the shipped DDL. To introduce changes to the Business Analytics Warehouse data model, you use ODI to generate a new DDL script.

For instructions on generating DDL scripts and assigning tablespaces to tables and indexes, see the topic "Generating DDL and Assigning Tablespaces to Tables and Indexes," in *Oracle Fusion Middleware Installation Guide for Oracle Business Intelligence Applications*.

Logging and Diagnostics

Oracle BI Applications Configuration Manager uses Oracle Diagnostic Logging. For information about managing log files and diagnostic data, see *Oracle Fusion Middleware Administrator's Guide*.

A

Manage Load Plans User Interface Reference

This section contains reference information about the Manage Load Plans user interface elements in the Oracle BI Applications Configuration Manager.

This section contains the following topics:

- Manage Load Plans Toolbar Options
- Manage Load Plans Main Page
- Create Load Plan Page
- Copy Load Plan Page
- Edit Load Plan Page
- Load Plan Details Page
- Schedule Load Plan Dialog

Manage Load Plans Toolbar Options

The Manage Load Plan window contains the following icons and menu options.

Icon or Menu Option	Description
Actions Menu: **Execute Reset Data Warehouse Scenario**	This command resets the data warehouse by truncating the W_ETL_LOAD_DATES table. This ensures that the subsequent load will truncate all target tables and do a fresh full load.
	Copy Enables you to define a new load plan with the same fact groups as the selected load plan definition but with a different name and identifier.
	Generate Starts the process to build the load plan in the ODI repository. A load plan must build successfully before it can be executed.
	Show Generation Status Details Shows the generation status for each step included in the load plan.

Icon or Menu Option	Description
	Reset Generation Status Resets the generation status to Not Generated.
	Show Load Plan Details Launches the Oracle Data Integrator Console, which enables you to view detailed information about the selected load plan.
	Execute Executes a load plan that has been successfully generated.
	Show Execution Status Details Displays the execution details of a load plan that has been executed.
	Generation Status: Not Generated Indicates the load plan has not been generated, or was modified in Configuration Manager without being regenerated.
	Generation Status: In Progress Indicates the load plan is in the process of being generated.
	Generation and Execution Status: Succeeded In the Generation Status column, indicates the load plan was successfully generated. In the Execution Status column, indicates the load plan completed its run successfully.
	Generation and Execution Status: Failed In the Generation Status column, indicates the load plan generation failed. In the Execution Status column, indicates the load plan run failed due to an error.
	Execution Status: Not Executed Indicates the load plan has not been executed.
	Execution Status: Running Indicates the load plan is currently running.

Manage Load Plans Main Page

Use the Manage Load Plans main page to view and edit existing load plans and to create new load plan definitions. You can search for an existing load plan by clicking **Search** above the master list panel.

The Load Plans list displays the load plans in the Configuration Manager repository. See "Manage Load Plans Toolbar Options" for a description of the toolbar options on this page.

Manage Load Plans: Fact Groups Tab

Use this tab to view the fact groups associated with a load plan selected in the Load Plans list. Fact groups displayed may belong to a hierarchy of fact groups. You can expand the fact group node to view the hierarchy. If a fact group is a child of another fact group in a hierarchy, it appears twice in the tree table, since it is associated with both the functional area and the parent fact group.

Manage Load Plans: Data Load Parameters Tab

Use this tab to view and edit the data load parameters associated with a load plan selected in the Load Plans list. The Data Load Parameters list includes both application-specific and global parameters. Application-specific parameters are associated with one or more fact groups included in the load plan definition. Global parameters apply to all applications and can also be associated with specific fact groups.

Key points about the Data Load Parameters tab:

- If a listed parameter requires a value but a value has not been assigned, the respective row in the table is tagged with an error icon. Parameters that do not require a value (value can be null) are not tagged even if no value has been assigned.

- You can filter the list of parameters to display only the data load parameters that have no value by using the Show drop-down list in the toolbar.

- You can export and save content displayed in the table to a Microsoft Excel formatted file by clicking the Export icon in toolbar.

- You can change a parameter value by selecting the parameter in the list, and then clicking the Edit icon in the toolbar. The Edit Parameter Value dialog is displayed. To change a parameter value, the user must have been assigned a role that has the appropriate privilege.

Manage Load Plans: Domains and Mappings Tab

Use the Domains and Mappings tab to view and edit domains and mappings related to a load plan selected in the Load Plan list. The domains and mappings are associated with the fact group included in the load plan definition.

Key points about the Domains and Mappings tab.

- If a source domain in the list contains members that have not been mapped to an appropriate warehouse domain member, the row in the table is tagged with an error icon. Some source domain members are not applicable, and, therefore, are not tagged even if they are unmapped.

- You can filter the list of mappings to display only the domains that have unmapped source members using the Show drop-down list in the toolbar.

- You can export and save content displayed in the table to a Microsoft Excel formatted file by clicking the Export icon in toolbar.

- You can change a domain mapping by selecting the mapping in the list, and then clicking the Edit icon in the toolbar. The Edit Domain Member Mappings dialog is displayed. To change a domain member mapping, the user must have been assigned a role that has the appropriate privilege.

Manage Load Plans: Schedules Tab

Use the Schedule tab to view, create, edit and delete schedules for the execution of a load plan. A load plan schedule includes the following required properties:

- **Context** - The ODI context to be used when the load plan is run. Note that **Global** is the only supported context.

- **Logical Agent** - The ODI Agent to be used when the load plan is run.

- **Recurrence** - Frequency of occurrence. Possible values are the following:

  - On Agent Startup

  - Simple

  - Hourly

  - Daily

  - Weekly

  - Monthly (day of the month)

  - Monthly (week day)

  - Yearly

- **Status** - Status of the schedule. Possible values are the following:

  - Active

  - Inactive

  - Active for the period

- **Scheduled Time** - Date and time the load plan is to be executed.

Schedules Tab Toolbar Icons

The toolbar icons provide the following functionality:

- **Add** - This command invokes the Schedule Load Plan dialog, which enables you to define a new schedule for the selected load plan.

- **Edit** - Applies to a selected schedule. This command invokes the Schedule Load Plan dialog, which enables you to make changes to the selected schedule.

- **Remove** - This command removes the schedule from the repository.

- **Update Schedules** - This command adds the schedule to the ODI Agent scheduler if the agent is already started in ODI. Note that if the ODI Agent is already running at the time you define the schedule in the Oracle BI Applications Configuration Manager, you must click **Update Schedules** to add the schedule to the ODI Agent scheduler.

Create Load Plan Page

Use the Create Load Plan pages to create a new load plan definition. A load plan comprises one or more fact groups as well as additional properties.

Element	Description

Create Load Plan first page	The first page of the Create Load Plan series requests the following information:
	
	■ **Name** - (Required) The unique name of the load plan.
	
	■ **Description** - (Optional) Additional information about the load plan.
	
	■ **Load Plan Type** - (Optional) Possible values are the following:
	    ■ Source Extract (SDE)
	    ■ Source Extract and Load (SDE, SIL and PLP)
	    ■ Warehouse Load (SIL and PLP)
	    ■ Domain-only Extract and Load (SDE and SIL)
	
	■ **Source Instance** - (Optional) The data sources from which fact groups will be selected.

Create Load Plan second page	The second page of the Create Load Plan series contains the following tabs: ■ **Available Fact Groups** - Enables you to select the fact groups you want to include in the load plan definition. ■ **Selected Fact Groups** - Displays the fact groups you selected in the Available Fact Groups tab. The Remove icon enables you to remove a selected fact group.

Copy Load Plan Page

Use the Copy Load Plan page to make a copy of an existing load plan. The Copy Load Plan page displays the load plan information for the load plan you want to copy. You must provide a unique name for the copy of the load plan. You can optionally change the following information:

■ First page of the Copy Load Plan series

  – Description

  – Load Plan Type

  – Source Instance

■ Second page of the Copy Load Plan series

  – Fact Group

Edit Load Plan Page

Use the Edit Load Plan page to edit an existing load plan. You can edit the following information:

■ First page of the Edit Load Plan series

  – Name

- – Description
- – Source Instance
- ■ Second page of the Edit Load Plan series
  - – Fact Group

Load Plan Details Page

Use the Load Plan Details page to view detailed information about the definition and execution status of the load plan selected in the Load Plan list.

Element	Description
Navigation tabs	In the Navigation tabs, you can browse for objects contained in the repository. When an object or node is selected, the Navigation Tab toolbar displays icons for the actions available for this object or node. If an action is not available for this object, the icon is greyed out. ■ **Browse** - The Browse tab enables you to browse for runtime, design time, and topology objects, ■ **Management** - The Management tab enables you to view repository connection information.
Load Plan tab	The Load Plan tab displays three panels, containing the following information: ■ **Definition** - This panel displays information about the load plan definition, such as name, ID, and parent folder as well as log details. ■ **Creation and Update Information** - This panel displays information about when the load plan was created and updated and which user performed the action. ■ **Relationships** - This panel displays the following information: **Steps** - Lists steps included in the load plan. **Exceptions** - Lists exceptions by step number. **Variables** - Lists variables associated with the load plan. **Executions** - Lists runs for the load plan, including execution status.
Load Plan Execution tab	The Load Plan execution tab displays three panels, containing the following information: ■ **Execution** - This panel displays information about the load plan run, including the status and any error messages. ■ **Definition** - This panel displays information about the load plan definition, such as name, ID, and parent folder as well as log details. ■ **Relationships** - This panel displays the following information. **Steps** - Lists steps included in the load plan. **Variables** - Lists variables associated with the load plan. **Sessions** - Lists information about the sessions included in the load plan, including start and end dates.

Schedule Load Plan Dialog

Use the Schedule Load Plan dialog to create and edit schedules for load plan execution. A load plan schedule includes the following properties:

- **Context -** The ODI context to be used when the load plan is run. Note that **Global** is the only supported context.

- **Logical Agent** - The ODI Agent to be used when the load plan is run.

- **Log Level -** The level of logging information to retain. The Oracle BI Applications Configuration Manager uses Oracle Diagnostic Logging. For information about managing log files and diagnostic data, see *Oracle Fusion Middleware Administrator's Guide*.

- **Status** - The status of the schedule. Possible values are the following:

  - Active

  - Inactive

  - Active for the period

- **Recurrence** - The frequency of occurrence. Possible values are the following:

  - On Agent Startup

  - Simple

  - Hourly

  - Daily

  - Weekly

  - Monthly (day of the month)

  - Monthly (week day)

  - Yearly

    Depending on the Recurrence option you select, options for selecting the date and time are dynamically displayed.

B

Knowledge Modules Reference

This section provides detailed information about the knowledge modules available with Oracle BI Applications.

This section contains the following topic:

- Oracle BI Applications Integration Knowledge Modules

Oracle BI Applications Integration Knowledge Modules

This section describes the Integration Knowledge Modules (IKMs) that are available in Oracle BI Applications.

- IKM BIApps Oracle Control Append
- IKM BIAPPS Oracle Event Queue Delete Append
- IKM BIAPPS Oracle Fact Incremental Update
- IKM BIAPPS Oracle Incremental Update
- IKM BIAPPS Oracle Period Delete Append
- IKM BIAPPS Oracle Slowly Changing Dimension
- IKM BIAPPS SQL Target Override
- Nested IKM BIAPPS Oracle Control Append
- Nested IKM BIAPPS Oracle Event Queue Delete Append

IKM BIApps Oracle Control Append

This IKM integrates data into an Oracle target table in append mode. All records are inserted without any key checks. Data can be controlled by isolating invalid data in the error table and recycling when fixed.

Prerequisites

Perquisites for using this IKM are the following:

- If the "Synchronize Deletions from Journal" process is executed, the deleted rows on the target are committed.
- To use FLOW_CONTROL and RECYCLE_ERRORS options, the update ley must be set on the interface.

Options for Functionality

- **Unspecified Record.** If the target table is a dimension, set this option to TRUE to automatically insert an "Unspecified" record. This is referenced by facts in case no other dimension record matches. The default column values are determined by model naming standards using the user-defined function GET_UNSPEC_VALUE.

Options for Performance Tuning

- **Hints** - This IKM allows the passing of hints into the generated SQL. For more information, see the article titled "Oracle Business Intelligence Applications Version 11.1.1.7.1 Performance Recommendations (Doc ID 1539322.1)" on My Oracle Support.

- **Alter Session List** - Applies a list of alter session commands to the session used by the KM. Commands should be separated by a semi-colon and without the "ALTER SESSION" prefix. Each command should be prefixed SRC or TGT depending on whether it should be executed on the source connection (relevant if using an LKM) or the target connection. For example:

```
SRC set TRACEFILE_IDENTIFIER='ODI_TRACE'; SRC set events '10046 TRACE NAME
CONTEXT FOREVER, LEVEL 8'; TGT set TRACEFILE_IDENTIFIER='ODI_TRACE_TGT';
```

IKM BIAPPS Oracle Event Queue Delete Append

This IKM integrates data into an Oracle target table using an event queue to process incremental changes. This IKM is used when working with data that is versioned over time (similar to a slowly changing dimension). In full load all records are inserted. The event queue tracks the natural keys that are changing, and the earliest date a change occurred. For each natural key, any existing target records on or after the earliest change are deleted and the new records inserted.

If there are effective from and to dates, these are maintained automatically in both full and incremental loads.

Data can be controlled by isolating invalid data in the error table, but recycling data is not supported.

Prerequisites

Prerequisites for using this IKM are the following:

- The event queue table must be defined using the Event Queue Table option.

    - It must have the column EARLIEST_CHANGE_DATE (DATE datatype)

    - It must follow the standard naming convention ending with _EQ_TMP

- Join between target table and event queue must be defined using the Event Queue Join option.

- Target table must have SCD behavior set for:

    - Natural key

    - Starting/ending timestamp

- Interface must only select the source data that is changing, as controlled by the event queue, which lists the natural keys that are changing and the earliest date of any change.

    - Either the data is selected from temporary or staging tables which only contain incremental data

        Or

- – Use the nested IKM BIAPPS Oracle Event Queue Delete Append to include a join to the event queue in incremental load

- If the "Synchronize Deletions from Journal" process is executed, the deleted rows on the target are committed.

Options for Functionality

- **Event Queue Table.** The name of the event queue table that holds the incremental changes. This option is mandatory.

  Prerequisites for this option are the following:

  - – Event queue table must contain EARLIEST_CHANGE_DATE column (DATE datatype)

  - – Oracle BI Applications naming standard for table ends with _EQ_TMP

- **Event Queue Join.** Assuming the alias T for Target Table and Q for Event Queue Table, define the equi-join between the Target Table and the Event Queue Table. This is used in the Event Queue Update and Event Queue Delete steps and in rare cases may be omitted if neither of those steps are required. The filter on EARLIEST_CHANGE_DATE should not be included in the join option.

- **Event Queue Delete.** Whether or not to delete records in the target that are being processed in incremental load. In most cases this should be enabled, but in rare cases where more than one interface loads the target table then it only needs to be enabled on the first one.

- **Event Queue Update.** Whether or not to correct effective dates on records in the target that are affected by the incremental load. In most cases this should be enabled, but in rare cases where more than one interface loads the target table then it only needs to be enabled on the last one.

- **High Data Value.** The default value to use for the maximum ending timestamp. In most cases the default value of #HI_DATE is fine, but for some persisted staging tables that reflect the OLTP might use a different value e.g. #ORA_HI_DATE.

- **Unspecified Record.** If the target table is a dimension, set this to TRUE to automatically insert an "Unspecified" record. This is referenced by facts in case no other dimension record matches. The default column values are determined by model naming standards using the user-defined function GET_UNSPEC_VALUE.

Options for Performance Tuning

- **Hints** - This IKM allows the passing of hints into the generated SQL. For more information, see the article titled "Oracle Business Intelligence Applications Version 11.1.1.7.1 Performance Recommendations (Doc ID 1539322.1)" on My Oracle Support.

- **Alter Session List** - Applies a list of alter session commands to the session used by the KM. Commands should be separated by a semi-colon and without the "ALTER SESSION" prefix. Each command should be prefixed SRC or TGT depending on whether it should be executed on the source connection (relevant if using an LKM) or the target connection. For example:

```
SRC set TRACEFILE_IDENTIFIER='ODI_TRACE'; SRC set events '10046 TRACE NAME
CONTEXT FOREVER, LEVEL 8'; TGT set TRACEFILE_IDENTIFIER='ODI_TRACE_TGT';
```

IKM BIAPPS Oracle Fact Incremental Update

This IKM integrates data into an Oracle target table in append mode. All records are inserted without any key checks. Data can be controlled by isolating invalid data in the error table and recycling when fixed.

Prerequisites

A prerequisite for using this IKM is the following:

- The scenario LOAD_PARTITION_METADATA must be run before any interface using this KM, passing in the target table name as a parameter. This will load W_ETL_PART_TABLES and W_ETL_PART_TABLE_PARTS with the required metadata.

Options for Functionality

For a description of the options available with this IKM, see the options described for "IKM BIAPPS Oracle Incremental Update".

IKM BIAPPS Oracle Incremental Update

This IKM integrates data into an Oracle target table in incremental update mode. New records are inserted and existing records are updated. Data can be controlled by isolating invalid data in the error table and recycling when fixed.

Prerequisites

Prerequisites for using this IKM are the following:

- The update key must be defined in the interface and the key columns should be indexed.

- If the "Synchronize Deletions from Journal" process is executed, the deleted rows on the target are committed.

Options for Functionality

- **Soft Delete.** There are several additional steps that you can perform if the soft delete option is enabled. The variables #SOFT_DELETE_FEATURE_ENABLED (global) and #SOFT_DELETE_PREPROCESS (can be set for each fact or dimension group) control exactly which steps are executed.

  If you are able to implement triggers to efficiently capture deletes on the source system, you can disable the expensive pre-process steps (which extract all source records and compare against all target records) and, instead, directly populate the delete table.

Step	Action	Control
Soft delete pre-process	Runs the "Identify Delete" step which compares the data in the primary extract table against the target table, and records any obsolete target rows in the delete table. ■ Uses #LAST_ARCHIVE_DATE to filter target by CREATED_ON_DT system column (set variable to NULL to disable this). ■ Only data sources that have implemented the primary extract are included. If a data source has no records in the primary extract table then no records will be added to the delete table for that data source.	#SOFT_DELETE_FEATURE_ENABLED #SOFT_DELETE_PREPROCESS
Soft delete on target	Runs the "Soft Delete" step which updates the DELETE_FLG column to 'Y' on the target table for records which have been identified for delete.	#SOFT_DELETE_FEATURE_ENABLED
Truncate delete table	Removes records from the delete table once they have been processed	#SOFT_DELETE_FEATURE_ENABLED

Note that all these steps are committed together, along with any other inserts and updates to the target table. This keeps the data warehouse consistent.

Prerequisites for using this option are the following:

– The target table must have the ETL_PROC_WID and W_UPDATE_DT system columns.

– Tables <target>_PE and <target>_DEL must be created with the columns in the interface key.

– #LAST_ARCHIVE_DATE must be NULL if target table does not have the CREATED_ON_DT column.

– #SOFT_DELETE_PREPROCESS should be refreshed from Oracle BI Applications Configuration Manager by the load plan component.

■ **Date Track.** Automatically maintains effective from and to dates in the target table, similarly to a slowly changing dimension. It can handle dates or numbers (usually date keys, for example, YYYYMMDD). This can also set a current flag column, which will be 'Y' for the last record and 'N' for earlier records.

Prerequisites for using this option are the following:

– Set the slowly changing dimension behavior for the following table columns in the model:

- Natural key

- Starting/ending timestamp

- Current flag (optional)

- The natural key and starting timestamp columns should be indexed if they are not covered by the update key index.

- In the interface, map the effective to date column to a constant maximum value (usually #HI_DATE) set to execute on the target.

- If using current flag, map it to 'Y' again executing on the target.

- ETL_PROC_WID should be indexed.

- **Change Capture**. Captures target table changes to an image table to streamline the process of reflecting the changes in aggregates.

Step	Action	Control
Truncate change image table	Clears out image table ready to capture changes in the current process.	Always runs
Capture preload changes	Captures target table records that are about to be updated.	Always runs
Capture soft delete changes	Captures target table records that are about to be marked for soft delete.	Runs if soft delete feature is enabled
Capture post load changes	Captures target table records that have been inserted or updated.	Always runs

All these steps are committed together along with any other inserts and updates to the target table. This keeps the data warehouse consistent.

Prerequisites for using this option are the following:

- For W_FACT_F, the image table W_FACT_CMG must be created with all the columns of the target table as well as the following columns:

  CHANGED_IN_TASK

  PHASE_CODE

  PHASE_MULTIPLIER

- **Unspecified Record.** If the target table is a dimension, set this to TRUE to automatically insert an "Unspecified" record. This is referenced by facts in case no other dimension record matches. The default column values are determined by model naming standards using the user-defined function GET_UNSPEC_VALUE.

Options for Performance Tuning

- **Hints** - This IKM allows the passing of hints into the generated SQL. For more information, see the article titled "Oracle Business Intelligence Applications Version 11.1.1.7.1 Performance Recommendations (Doc ID 1539322.1)" on My Oracle Support.

- **Alter Session List** - Applies a list of alter session commands to the session used by the KM. Commands should be separated by a semi-colon and without the "ALTER SESSION" prefix. Each command should be prefixed SRC or TGT depending on whether it should be executed on the source connection (relevant if using an LKM) or the target connection. For example:

```
SRC set TRACEFILE_IDENTIFIER='ODI_TRACE'; SRC set events '10046 TRACE NAME
CONTEXT FOREVER, LEVEL 8'; TGT set TRACEFILE_IDENTIFIER='ODI_TRACE_TGT';
```

- **Analyze Target** - Statistics will be collected on the target table before it is loaded if the KM Option ANALYZE_TARGET is set to True. By default it is set to False.

- **Analyze Flow Table** - Statistics will be collected on the flow table (I$) after it is loaded if the KM Option ANALYZE_FLOW_TABLE is set to True. By default it is set to False. This option also affects how the effective dates are calculated in full load if the date track option is enabled. If the flow table is not analyzed then an UPDATE statement is used to set the effective to dates. Otherwise a MERGE statement is used.

- **Bulk Mode (variable #ETL_BULK_MODE)** - If enabled, bulk mode will use the direct path write to target (append hint) and bypass the I$ table (if no other KM options requiring it, for example, flow control, recycle errors, and date track). The bulk mode variable can be set to:

  Y - Enabled

  F - Enabled for full load only

  N - Disabled

IKM BIAPPS Oracle Period Delete Append

This IKM integrates data into an Oracle target table, aggregating data by time periods. New records are inserted, and changes to existing periods will be handled by deleting the old data for the period and then inserting the new data. Data for old periods may be purged. Data can be controlled by isolating invalid data in the error table and recycling when fixed.

Prerequisites

Prerequisites for using this IKM are the following:

- The Delete Period Type option must specify which period type the target table is based on.

- For Calendar Periods, the target table must contain a column for the period end date key:

  - Must be a date wid type (YYYYMMDD).

  - Must be named PERIOD_END_DT_WID or otherwise identified in the mapping using the UD1 check box.

- For MCAL calendar periods the target table must contain:

  - Period key, usually MCAL_PERIOD_WID or MCAL_DAY_WID, otherwise identified in the mapping using UD1 check box.

  - Calendar key, usually MCAL_CAL_WID, otherwise identified in the mapping using UD2 check box.

- Bitmap indexes on the above mentioned key columns.

Calendar Periods

Calendar periods of day, week, month, quarter and year are supported. For these period types, the incremental load assumes the interface is loading data for the current period only. The incremental update works by deleting any existing data for the current period. Then, the fresh set of data for the current period is inserted.

Optionally, data older than a specified age can be automatically purged with the Periods to Keep option. The option Periods To Delete does not apply for calendar periods.

MCAL Calendar Periods

MCAL calendar periods of MCAL Day and MCAL Period are supported. For these period types, the incremental load assumes the interface is loading data for the current period and a given number of previous periods. The incremental update works by deleting any existing data for these periods. Then, the fresh set of data is inserted.

The option Periods To Delete controls how many previous periods (as well as the current one) are being incrementally loaded by the interface. For example, a value of 1 would indicate reprocessing the current and previous period every load. The Periods to Keep option does not apply for MCAL calendar periods.

Options for Functionality

- **Delete Period Type** - The type of calendar periods the target table is based on. It is used for deleting current periods to reprocess or obsolete periods to purge. Valid values are the following:

  Calendar periods: CAL_DAY, CAL_WEEK, CAL_MONTH, CAL_QTR, CAL_YEAR.

  MCAL calendar periods: MCAL_DAY, MCAL_PERIOD.

  Prerequisites for using this option are the following:

  - For calendar periods, the target table must contain a column for the period end date key

    - Must be named PERIOD_END_DT_WID or otherwise identified in the mapping using the UD1 check box.

    - Must be a date wid type (YYYYMMDD).

  - For MCAL calendar periods the target table must contain:

    - Period key, usually MCAL_PERIOD_WID or MCAL_DAY_WID, otherwise identified in the mapping using UD1 check box.

    - Calendar key, usually MCAL_CAL_WID, otherwise identified in the mapping using UD2 check box.

- **Periods to Delete** - This option is for configuring additional functionality for the MCAL calendar periods only. It must be left at the default value (0 - delete current period only) for calendar periods. Setting this option to a value N > 0 will delete exactly N Julian periods of data based on the delete period type. For example, setting to 1 will mean the current and previous periods are deleted.

- **Periods to Keep** - This option is for configuring additional functionality for the calendar periods only. It must be left at the default value (0 - keep all periods) for calendar periods. Setting this option to a value N > 0 will delete data in periods which are exactly N julian periods older than the current julian period

Options for Performance Tuning

- **Hints** - This IKM allows the passing of hints into the generated SQL. For more information, see the article titled "Oracle Business Intelligence Applications Version 11.1.1.7.1 Performance Recommendations (Doc ID 1539322.1)" on My Oracle Support.

- **Alter Session List** - Applies a list of alter session commands to the session used by the KM. Commands should be separated by a semi-colon and without the "ALTER SESSION" prefix. Each command should be prefixed SRC or TGT depending on whether it should be executed on the source connection (relevant if using an LKM) or the target connection. For example:

```
SRC set TRACEFILE_IDENTIFIER='ODI_TRACE'; SRC set events '10046 TRACE NAME
CONTEXT FOREVER, LEVEL 8'; TGT set TRACEFILE_IDENTIFIER='ODI_TRACE_TGT';
```

IKM BIAPPS Oracle Slowly Changing Dimension

This option integrates data into an Oracle target table modeled as a Type 2 slowly changing dimension. New records are inserted, and changes to existing records can either trigger an insert or an update depending on the whether there is a change to any of the Type 2 columns. Data can be controlled by isolating invalid data in the error table and recycling when fixed. This option uses the variables #TYPE2_FLG and #UPDATE_ALL_HISTORY to control behavior.

Prerequisites

Prerequisites for using this IKM are the following:

- Update key must be defined in the interface and the key columns should be indexed (usually INTEGRATION_ID, DATASOURCE_NUM_ID, SRC_EFF_ FROM_DT).

- Slowly changing dimension behavior must be set for all target table columns (set in model) and must include the following:

    - Surrogate key (usually ROW_WID)

    - Natural key (usually INTEGRATION_ID, DATASOURCE_NUM_ID)

    - Start and end timestamps (usually EFFECTIVE_FROM_DT and EFFECTIVE_ TO_DT)

    - Current flag (usually CURRENT_FLG)

- End timestamp should be mapped to the maximum value (usually #HI_DATE).

- Current flag should be mapped to Y.

- Source from and to dates should be Not Null (default to #LOW_DATE or #HI_ DATE).

- ETL_PROC_WID should be indexed.

Column Classification

The following table describes how dimension columns should be categorized and the different behaviors for each classification. The categorization is done on individual table columns in the model.

Column	Description	SCD Behavior	Other Flexfields
Surrogate key	Warehouse generated primary key for dimension	Surrogate key	
Natural key	Business or source key, unique in combination with time	Natural key	
Start timestamp	Time record is effective from	Start timestamp	
End timestamp	Time record is effective to	End timestamp	

Column	Description	SCD Behavior	Other Flexfields
Current flag	Whether the record is the latest effective	Current flag	
Type 2 columns	Creates new version of record (insert) if there is a change to any of the Type 2 columns	Insert on change	
Update history columns	Always set to the value from the current record	Overwrite on change	Not a system column
Other system columns	Maintained as insert/update	Overwrite on change	System column
Change columns	Record is rejected if there is no change to any of these columns	Overwrite on change	Change column and system column
SCD1 key	Warehouse generated key corresponding to the natural key	Overwrite on change	SCD1 WID (column flexfield)

Slowly Changing Dimension Features

- **Type 2 Changes** - If the variable #TYPE2_FLG is turned off (set to 'N') then the dimension behaves as a Type 1 dimension. The natural key must be unique (no history allowed) because no maintenance of the start/end dates is performed. With #TYPE2_FLG on (set to 'Y') new records will be inserted; changes that update at least one Type 2 column will also trigger an insert, subject to some restrictions, and any other change will update the existing record.

  Type 2 changes are triggered as follows:

  - The incoming record must have at least one type 2 column different when compared to the current dimension record.

  - The new Type 2 record start timestamp is calculated as follows: If there is a change column with a non-null date value, then use that (the Oracle BI Applications standard is to use CHANGED_ON_DT as the change column); otherwise, use the current timestamp (sysdate)

- **Update All History** - With #TYPE2_FLG and #UPDATE_ALL_HISTORY both on (set to 'Y'), then any update history columns will be updated with the value from the current version of the record (latest record with the same natural key).

Options for Functionality

- **Unspecified Record -** If the target table is a dimension, set this to TRUE to automatically insert an "Unspecified" record. This is referenced by facts in case no other dimension record matches. The default column values are determined by model naming standards using the user-defined function GET_UNSPEC_VALUE.

- **SCD1 Key** - Set this to TRUE to automatically maintain a surrogate natural key or Type 1 key. This is managed using a Type 1 table named according to a standard pattern. If the dimension table is W_DIMENSION_D, then the Type 1 table will be W_DIMENSION_T1_D. The sequence that generates the Type 1 key is also named according to a standard pattern. Continuing the example, it would be named W_DIMENSION_D_S1W. Additional columns that are at the same grain (also Type 1) can be automatically maintained on the Type 1 table if they are marked with the UD1 flag.

Prerequisites for W_DIMENSION_D are as follows:

- Type 1 key column must be identified by the SCD1 WID flexfield in the model.

- Type 1 key column should be mapped on source or staging to a constant value, for example, 0.

- Type 1 key column should be insert only.

- Type 1 table (W_DIMENSION_T1_D) must have at least the following columns:

  - Type 1 key column

  - Natural key columns

  - System columns W_INSERT_DT, W_UPDATE_DT and ETL_PROC_WID

  - Any columns marked as UD1

- Type 1 table (W_DIMENSION_T1_D) should have indexes on: Type 1 key columns and natural key columns.

- Type 1 Key sequence should be created (W_DIMENSION_D_S1W).

- Type 1 Key sequence should be created (W_DIMENSION_D_S1W) .

- **Fill Gaps** - Set this to TRUE to automatically extend the first records to cover any earlier date.

- **Soft Delete.** There are several additional steps that you can perform if the soft delete option is enabled. The variables #SOFT_DELETE_FEATURE_ENABLED (global) and #SOFT_DELETE_PREPROCESS (can be set for each fact or dimension group) control exactly which steps are executed.

  If you are able to implement triggers to efficiently capture deletes on the source system, you can disable the expensive pre-process steps (which extract all source records and compare against all target records) and, instead, directly populate the delete table.

Step	Action	Control
Soft delete pre-process	Runs the "Identify Delete" step which compares the data in the primary extract table against the target table, and records any obsolete target rows in the delete table.	#SOFT_DELETE_FEATURE_ ENABLED #SOFT_DELETE_PREPROCESS
	■ Uses #LAST_ARCHIVE_ DATE to filter target by CREATED_ON_DT system column (set variable to NULL to disable this).	
	■ Only data sources that have implemented the primary extract are included. If a data source has no records in the primary extract table then no records will be added to the delete table for that data source.	

Step	Action	Control
Soft delete on target	Runs the "Soft Delete" step which updates the DELETE_FLG column to 'Y' on the target table for records which have been identified for delete.	#SOFT_DELETE_FEATURE_ENABLED
Truncate delete table	Removes records from the delete table once they have been processed	#SOFT_DELETE_FEATURE_ENABLED

Note that all these steps are committed together, along with any other inserts and updates to the target table. This keeps the data warehouse consistent.

Prerequisites for using this option are the following:

– The target table must have the ETL_PROC_WID and W_UPDATE_DT system columns.

– Tables <target>_PE and <target>_DEL must be created with the columns in the interface key.

– #LAST_ARCHIVE_DATE must be NULL if target table does not have the CREATED_ON_DT column.

– #SOFT_DELETE_PREPROCESS should be refreshed from Oracle BI Applications Configuration Manager by the load plan component.

Options for Performance Tuning

■ **Detection Strategy** - To avoid updating the table if no changes have occurred, the incoming data is compared with the existing record on a set of change columns. These are either defined in the model column flexfield (OBI_CHANGE_COLUMN), or otherwise all columns are compared. To always process the changes, this option can be disabled.

■ **Hints** and **Full History** - This IKM allows the passing of hints into the generated SQL. For more information, see the article titled "Oracle Business Intelligence Applications Version 11.1.1.7.1 Performance Recommendations (Doc ID 1539322.1)" on My Oracle Support.

■ **Alter Session List** - Applies a list of alter session commands to the session used by the KM. Commands should be separated by a semi-colon and without the "ALTER SESSION" prefix. Each command should be prefixed SRC or TGT depending on whether it should be executed on the source connection (relevant if using an LKM) or the target connection. For example:

```
SRC set TRACEFILE_IDENTIFIER='ODI_TRACE'; SRC set events '10046 TRACE NAME
CONTEXT FOREVER, LEVEL 8'; TGT set TRACEFILE_IDENTIFIER='ODI_TRACE_TGT';
```

■ **Analyze Target** - Statistics will be collected on the target table before it is loaded if the KM Option ANALYZE_TARGET is set to True. By default it is set to False.

IKM BIAPPS SQL Target Override

This is a unique IKM in that it allows you to specify a custom SQL statement to execute against a target table. This SQL statement will execute once for each source record returned by the interface. For a given execution of this SQL statement, values from the source record may be referenced as parameters in the SQL.

This IKM has a variety of different use cases; however, because it does not implement an interface in the typical manner of ODI interfaces, it should be reserved only for implementing system defined logic that will never be customized.

If the custom SQL is not suitable for all target technologies, then platform specific-SQL statements can be given as alternatives. This IKM will run the SQL in the option corresponding to the target technology on execution if available. Otherwise, the SQL in the generic option will be used.

Prerequisites

Prerequisites for using this IKM are the following:

- Implements non-customizable system logic.

- If the generic SQL Override is not suitable for a particular platform, perhaps for performance or incompatibility reasons, then a platform specific SQL Override option must be provided.

- SQL Statements must begin with "INSERT", "UPDATE" or "DELETE" keywords.

- You must ensure hint placeholders are included in the SQL.

- You must ensure all required indexes are in place.

Referencing Source Data

To use data from the source data in the SQL override, do the following:

- Map the source data you want to reference to one of the target columns on the interface.

- Reference it in the SQL Override using the syntax :TARGET_COLUMN.

Options for Functionality

SQL Override Options - A generic SQL Override option is available. Also, platform-specific options are provided for cases where there are variations in syntax or performance for different target technologies.

Prerequisites for using this option are as follows:

- Begin SQL statements with "INSERT", "UPDATE" or "DELETE" keywords.

- If the generic SQL Override is not suitable for a particular platform, for example, due to performance or incompatibility reasons, then a platform-specific SQL Override option must be provided.

Options for Performance Tuning

- **Hints** - This IKM allows the passing of hints into the generated SQL. For more information, see the article titled "Oracle Business Intelligence Applications Version 11.1.1.7.1 Performance Recommendations (Doc ID 1539322.1)" on My Oracle Support.

- **Alter Session List** - Applies a list of alter session commands to the session used by the KM. Commands should be separated by a semi-colon and without the "ALTER SESSION" prefix. Each command should be prefixed SRC or TGT depending on whether it should be executed on the source connection (relevant if using an LKM) or the target connection. For example:

```
SRC set TRACEFILE_IDENTIFIER='ODI_TRACE'; SRC set events '10046 TRACE NAME
CONTEXT FOREVER, LEVEL 8'; TGT set TRACEFILE_IDENTIFIER='ODI_TRACE_TGT';
```

Nested IKM BIAPPS Oracle Control Append

For a nested or temporary interface, using this IKM gives additional functionality in the SQL generated for the nested SQL block. The main benefit is allowing hints to be included. In addition, Aggregate Lookup and Source SQL Override can also be useful for some applications.

Options for Functionality

- **Aggregate Lookup** - Aggregates the SQL block by using the user flexfields to define a grouping of records to restrict the output to one row per group.

  Prerequisites for using this option are the following:

  - The interface must set the following user flexfields:

    - UD1: Group by columns. These columns are usually the columns being joined to.

    - UD2: Aggregated columns. The fields being returned.

    - UD3: Column for active lookup (LKP_ACTIVE). This is set to 1.

- **Source SQL Override** - Allows the generated nested SQL block to be overridden. This is useful if the SQL needs to be more dynamic in nature, for example, using variables to reference dynamic tables.

  Prerequisites for using this option are the following:

  - For each target interface column there must be an identically aliased column in the SELECT clause of the SQL override.

  - Use of this option is discouraged. Make sure the use case is reviewed by the BI Apps Standards Group before implementing.

Options for Performance Tuning

- **Hints** - This IKM allows the passing of hints into the generated SQL. For more information, see the article titled "Oracle Business Intelligence Applications Version 11.1.1.7.1 Performance Recommendations (Doc ID 1539322.1)" on My Oracle Support.

Nested IKM BIAPPS Oracle Event Queue Delete Append

As part of a main interface using this IKM for Event Queue controlled updates, this IKM can also be used in a nested or temporary interface to give additional functionality in the SQL generated for the nested SQL block. The main benefit is the ability to use the event queue table as the driving table in incremental load, but exclude the event queue table altogether in full load. Also, the use of hints is supported.

Prerequisites for Nested KM Usage

- The main interface must use this KM.

- The event queue table must be specified in the Event Queue Table option.

- The interface must contain only the following:

  - Main source table

  - Event queue table

  - Join between these using ANSI syntax

Mapping Event Queue Columns

In some cases the event queue columns may need to be referenced in parent interfaces. They can be mapped directly in the nested interface if they are identified with the UD1 flexfield. This includes them in the generated SQL block only in incremental load when the event queue table is also included. For full load both the event queue table and any columns mapped from it will be excluded.

Options for Functionality

Event Queue Table - Explicitly name the event queue table that is controlling the incremental load. This should be one of the two source tables on the nested interface.

Options for Performance Tuning

- **Hints** - This IKM allows the passing of hints into the generated SQL. For more information, see the article titled "Oracle Business Intelligence Applications Version 11.1.1.7.1 Performance Recommendations (Doc ID 1539322.1)" on My Oracle Support.