Oracle® Business Intelligence Applications

Administering Business Intelligence Applications

12c (11.1.1.10.3 PS4)
F83298-03
December 2023

ORACLE

Oracle Business Intelligence Applications Administering Business Intelligence Applications, 12c (11.1.1.10.3 PS4)

F83298-03

Primary Author: Hemala Vivek

Contributors: Nick Fry, Christine Jacobs, Padma Rao

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

Contents

Preface

Functional Setup Manager - Statement of Direction v
Audience v
Documentation Accessibility v
Diversity and Inclusion vi
Related Documentation vi
Conventions vi

1 About Multi-Language Support

About Pseudo-Translations 1-2
About Oracle BI Applications Domains 1-2
About Dimension Translation Tables 1-4

2 Localizing Oracle Business Intelligence Deployments

Maintaining Translation Tables Workflow for Oracle Analytics 2-1
Adding String Localizations for Oracle BI Repository Metadata 2-1
About Translating Presentation Services Strings 2-3
Changing the Default Currency in Oracle BI Applications 2-4

3 Oracle Business Analytics Warehouse Naming Conventions

Naming Conventions for Oracle Business Analytics WarehouseTables 3-1
Table Types for Oracle Business Analytics Warehouse 3-2
Aggregate Tables in Oracle Business Analytics Warehouse 3-4
Dimension Class Tables in Oracle Business Analytics Warehouse 3-4
Dimension Tables in Oracle Business Analytics Warehouse 3-4
Dimension Tables With Business Role-Based Flags 3-4
Fact Tables in Oracle Business Analytics Warehouse 3-5
Helper Tables in Oracle Business Analytics Warehouse 3-5
Hierarchy Tables in Oracle Business Analytics Warehouse 3-5
Mini-Dimension Tables in Oracle Business Analytics Warehouse 3-5

Staging Tables in Oracle Business Analytics Warehouse 3-6
Translation Tables in Oracle Business Analytics Warehouse 3-6
Internal Tables in Oracle Business Analytics Warehouse 3-7
Standard Column Prefixes in Oracle Business Analytics Warehouse 3-7
Standard Column Suffixes in Oracle Business Analytics Warehouse 3-8
System Columns in Oracle Business Analytics WarehouseTables 3-8
Multi-Currency Support for System Columns 3-10
Oracle Business Analytics Warehouse Primary Data Values 3-10
About Multi-Language Support in Oracle Business Analytics Warehouse 3-11
Oracle Business Analytics Warehouse Currency Preferences 3-11

4 Administering Oracle GoldenGate and Source Dependent Schemas

Source Dependent Schema Architecture 4-1
Tasks for Setting Up Oracle GoldenGate and the Source Dependent Schema 4-2
Setup Step: Configure Source and Target Database 4-2
Setup Step: Install Oracle GoldenGate on Source and Target Systems 4-5
Setup Step: Configure Configuration Manager and ODI to Support the Source Dependent Schema 4-8
Adding SDS Physical Schemas in ODI 4-8
Setup Step: Generate, Deploy, and Populate the Source Dependent Schema Tables on Target Database 4-8
Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Systems 4-13
Setup Step: Start Oracle GoldenGate on Source and Target Systems 4-19
Replicate Views from Source 4-19
ETL Customization 4-21
Patching 4-22
Troubleshooting Oracle GoldenGate and SDS 4-22
Create the SDS Tables 4-22
Using the DML Option to Perform an Initial Load 4-23
Create SDS Indexes and Analyze the SDS Schema 4-24
Setting up Ledger Correlation using Oracle GoldenGate 4-25

Preface

Oracle Business Intelligence Applications (Oracle BI 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 Analytics Server, 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.

Functional Setup Manager - Statement of Direction

Functional Setup Manager (FSM) has been desupported in the current release. Ignore the references to FSM that you might see in the online Help.

Audience

This information is intended for system administrators and ETL team members who are responsible for managing Oracle BI Applications.

It contains information about ETL customization, domains and localization, Oracle Business Analytics Warehouse naming conventions, and system administration tasks, including setting up and using Oracle GoldenGate and Source-Dependent Schemas to support ETL performance.

Documentation Accessibility

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

Access to Oracle Support

Oracle customers that have purchased support 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.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documentation

See the Oracle BI Applications documentation library for the complete set of Oracle BI Applications documents.

Conventions

This document uses these text conventions.

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

1

About Multi-Language Support

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

- About Pseudo-Translations
- About Oracle BI Applications Domains
- About Dimension Translation Tables

Configuring Base and Installed Data Warehouse Languages

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

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

To configure data warehouse languages, see Manage Warehouse Languages, *Oracle Business Intelligence Applications Functional Configuration Reference*.

Translation Tables

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

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

About Pseudo-Translations

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

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

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

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

About Oracle BI Applications Domains

A domain refers to the possible, unique values of a table column in a relational database. In transactional systems, domains are often referred to as list of values (LOVs), which present attribute selections in the user's session language.

The storage of the transaction is independent of the user's language; and, therefore, the field is stored using a language independent identifier. This identifier is typically a character code but can also be a numeric ID. The LOV or domain is then based on an ID-value pair, referred to as a member, and the LOV presents the values in the user's session language. At run time, the IDs are resolved to the value for the user's session language.

In the Oracle Business Analytics Warehouse, the number of unique values in any particular domain is relatively small and can have a low cardinality relative to the dimension it is associated with. For example, the Person dimension may have the domain 'Gender' associated with. The dimension may have millions of records, but the domain will generally have two or three members (M, F and possibly U). In the Oracle Business Analytics Warehouse, the Gender Code is stored in the Person dimension

which acts as a foreign key to the Domains Translation table which stores the translated values. When a query is run, the user-friendly text associated with the code value is returned in the user's session language.

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

Data Model

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

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

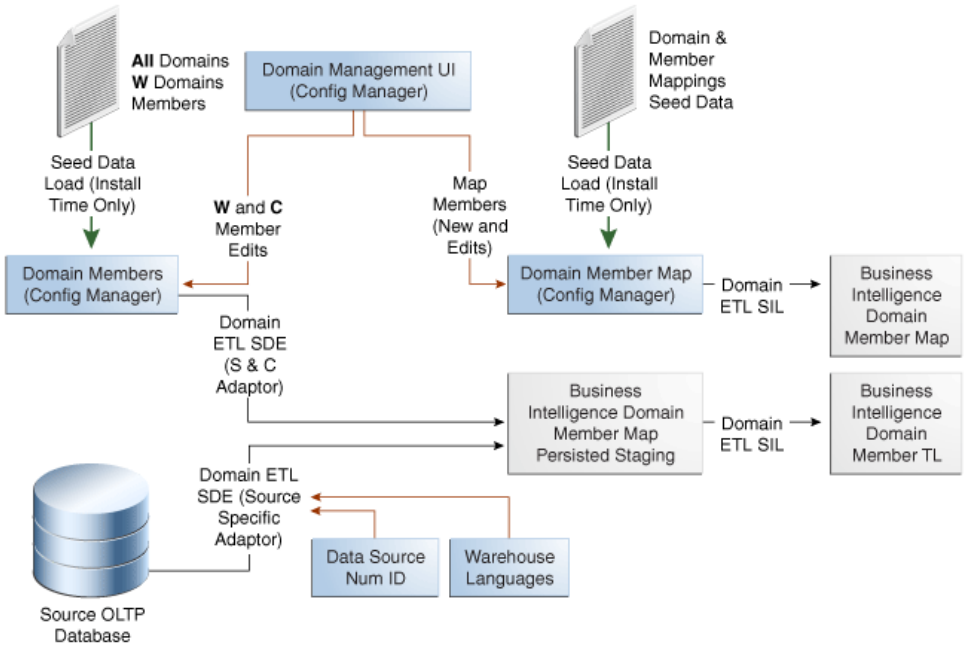
ETL Process

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

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

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

The figure shows an overview of the Oracle BI Applications domain ETL process.



About Oracle BI Applications Domains and Oracle Analytics

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

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

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

About Dimension Translation Tables

Domains are dimensional attributes that have a relatively small number of distinct members, have a low cardinality relative to the number of records in the dimension, and are often used in aggregations. Dimensions have other attributes that require translation that may not fit one or more of these criteria. Dimensions may have

translatable attributes that have a high cardinality relative to the dimension or may have a large number of members, and, thus, are not likely candidates for aggregation.

If the domains ETL process was implemented in such cases, performance would be very poor. As a result, these particular attributes are implemented using dimension translation tables.

Data Model

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

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

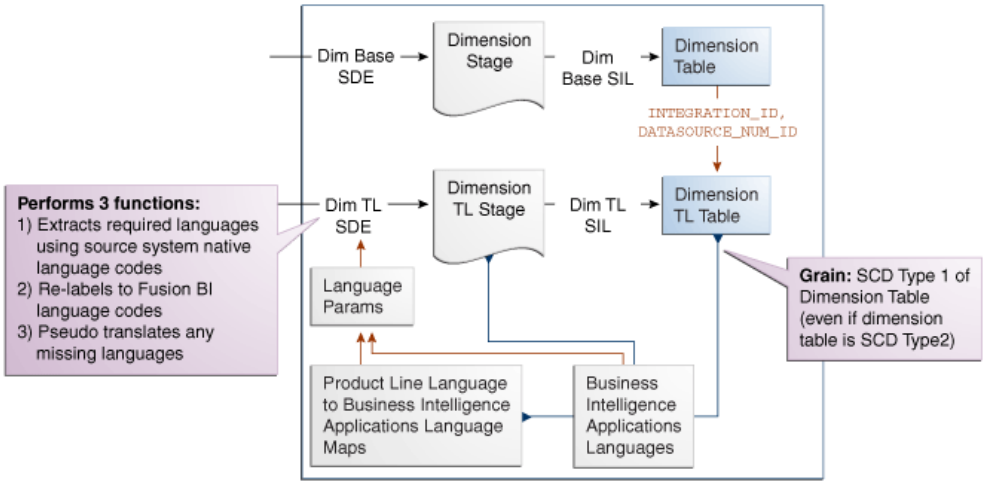
ETL Process

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

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

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

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



Oracle Analytics

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

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

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

Localizing New Domain Members and Oracle BI Repository Metadata

If you added new domain members that require localization or want to add string localizations in the Oracle BI Repository metadata, see About Setting Up Domain Member Mappings, in *Oracle Business Intelligence Applications Configuration Guide*.

2

Localizing Oracle Business Intelligence Deployments

Oracle Business Intelligence is designed to allow users to dynamically change their preferred language and locale preferences. You can configure Oracle Business Intelligence Applications (Oracle BI Applications) for deployment in one or more language environments other than English.

Topics:

- Maintaining Translation Tables Workflow for Oracle BI EE
- About Translating Presentation Services Strings
- Changing the Default Currency in Analytics Applications

Maintaining Translation Tables Workflow for Oracle Analytics

The Oracle Analytics Presentation layer supports multiple translations for any column name. When working with Oracle BI Answers or rendering a dashboard, users see their local language strings in their reports.

For example, English-speaking and French-speaking users would see their local language strings in their reports. There are two kinds of application strings requiring translation:

- Metadata

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

- Presentation Services

  Presentation Services objects are end-user created objects such as reports, dashboards, and pages. Translations for Presentation Services strings are stored in the XML caption files. .

Adding String Localizations for Oracle BI Repository Metadata

If you added a new domain member, you can add string localizations in the Oracle BI Repository metadata.

1. Stop the BI Services.

   Go to `[ORACLE_HOME]/user_projects/domains/bi/bitools/bin` and use the command: `./stop.sh`.

2. Open a database administration tool, and connect to the Oracle Business Analytics Warehouse schema.
3. Identify the strings for the following presentation objects:
   - Subject area

- Presentation table
- Presentation hierarchy
- Presentation level
- Presentation column

For example, for the subject area Payables Invoices - Prepayment Invoice Distributions Real Time, enter the following strings:

String	Presentation Object
Payables Invoices - Prepayment Invoice Distributions Real Time	Subject area
Time	Presentation table
Date - Year	Presentation hierarchy
Total	Presentation level
Year	Presentation level
Calendar Year	Presentation column

4. For each subject area, externalize the strings for localization and generate custom names for the presentation objects:

   a. In the Oracle BI Administration Tool, right-click the subject area and select **Externalize Display Names**, and then select **Generate Custom Names**.

   b. Save your work.

   See *Administering Oracle Analytics Server*.

5. Check the consistency of the repository, and remove any inconsistencies.

   See *Managing Metadata Repositories for Oracle Analytics Server*.

6. Enter the custom name of one of the presentation objects into the table C_RPD_MSGS:

```
INSERT INTO C_RPD_MSGS(MSG_ID, CREATED_BY, CREATION_DATE)
VALUES('<CUSTOM NAME OF PRESENTATION OBJECT>', 'CUSTOM',
SYSTIMESTAMP);
COMMIT;
```

   To view the values for custom names and logical columns in the Oracle BI Administration Tool, right-click the presentation object and select **Properties**. The data in the **Custom display name** field appears in the format VALUEOF(NQ_SESSION.VALUE, where VALUE is the custom name for a presentation object, or the logical value for a presentation column. This value is the value that you need to enter in the VALUES section of the SQL statement above.

7. Enter the localized string for the presentation object in the previous step into the table C_RPD_MSGS_TL:

```
INSERT INTO C_RPD_MSGS_TL(MSG_ID, MSG_TEXT, LANGUAGE_CODE,
CREATED_BY, CREATION_DATE)
VALUES('<CUSTOM NAME OF PRESENTATION OBJECT>', '<LOCALIZATION OF
THE STRING'>, '<LANGUAGE CODE FOR TRANSLATED LANGUAGE>', 'CUSTOM',
```

```
SYSTIMESTAMP);
COMMIT;
```

To identify the language code for a particular language, use the following SQL:

```
SELECT LANGUAGE_CODE, NLS_LANGUAGE, NLS_TERRITORY
FROM FND_LANGUAGES_B
WHERE INSTALLED_FLAG IN ('B', 'I');
```

8. Enter additional details about the presentation object into the table C_RPD_MSGS_REL as indicated by the following SQL:

```
INSERT INTO C_RPD_MSGS_REL(MSG_ID, MSG_NUM, MESSAGE_TYPE, CREATED_BY,
CREATION_DATE)
VALUES('<CUSTOM NAME OF PRESENTATION OBJECT>', '<TRANSLATION OF THE
STRING'>, '<LANGUAGE CODE FOR TRANSLATED LANGUAGE>', 'METADATA','CUSTOM',
SYSTIMESTAMP);
COMMIT;
```

9. Repeat steps 6 through 8 for each presentation object requiring localization.
10. Validate that the physical connection of the session initialization block INIT_USER_LANGUAGE_CODE is operable:
    a. In the Oracle BI Administration Tool, select **Manage**, **Variables**, **Session Initialization Block**.
    b. Right-click **INIT_USER_LANGUAGE_CODE**.
    c. In the Properties dialog, click **Edit Data Source**.
    d. Click **Test**, and input the value for the language code. Then, click **OK**.

       For example, for Arabic enter 'AR'.

       The value `USER_LANGUAGE_CODE = '<language code>'` should be returned.

       If this value is not returned, the TNS entry for the data source is not properly configured.
11. Restart the BI services.

    Go to `[ORACLE_HOME]/user_projects/domains/bi/bitools/bin` and use the command: `./start.sh`.
12. Verify the localized strings in Oracle BI Answers. On the login page, specify the appropriate language.

About Translating Presentation Services Strings

The translations for such Presentation Services objects as report and page names are copied to this location during the Oracle BI Applications installation process: `SDD/service_instances/service1/metadata/content/msgdb/l_language_abbreviation/captions`, where SDD is the Singleton Data Directory for example, `DOMAIN_HOME/bidata`. In multiple language deployment mode, if you add any additional Presentation Services objects, such as reports and new dashboard pages, you also need to add the appropriate translations.

Add these translations using the Catalog Manager tool. See *Administering Oracle Analytics Server*.

Changing the Default Currency in Oracle BI Applications

In Oracle BI Applications, you might see a dollar sign used as the default symbol when amounts of money are displayed.

To change this behavior, you must edit the `currencies.xml` file. The `currencies.xml` file is located in the following directories:

- Windows: `ORACLE_HOME\bi\bifoundation\web\display\currencies.xml`
- UNIX: `ORACLE_HOME/bi/bifoundation/web/display/currencies.xml`

To change the default currency in Analytics Applications:

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

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

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

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

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

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

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

4. Save and close the `currencies.xml` file.

3

Oracle Business Analytics Warehouse Naming Conventions

Oracle Business Analytics Warehouse contains these types of tables and columns. Be sure to follow the specified naming conventions.

> **Note:**
>
> This information does not apply to objects in the Oracle Business Intelligence repository.

Topics:

- Naming Conventions for Tables
- Table Types
- Internal Tables
- Standard Column Prefixes
- Standard Column Suffixes
- System Columns in Tables
- Multi-Currency Support for System Columns
- Primary Data Values
- Primary Data Values
- About Multi-Language Support
- Currency Preferences

Naming Conventions for Oracle Business Analytics WarehouseTables

Oracle Business Analytics Warehouse tables use a three-part naming convention: PREFIX_NAME_SUFFIX.

Part	Meaning	Table Type
PREFIX	Shows Oracle Business Analytics Warehouse-specific data warehouse application tables.	W_ = Warehouse
NAME	Unique table name.	All tables.

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

Table Types for Oracle Business Analytics Warehouse

This table lists the types of tables used in the Oracle Business Analytics Warehouse.

Table Type	Description
Aggregate tables (_A)	Contain summed (aggregated) data.
Dimension tables (_D)	Star analysis dimensions.
Delete tables (_DEL)	Tables that store IDs of the entities that were physically deleted from the source system and should be flagged as deleted from the data warehouse. Note that there are two types of delete tables: _DEL and _PE. For more information about the _PE table type, see the row for Primary extract tables (_PE) in this table.
Dimension Hierarchy tables (_DH)	Tables that store the dimension's hierarchical structure.
Dimension Helper tables (_DHL)	Tables that store many-to-many relationships between two joining dimension tables.
Staging tables for Dimension Helper (_DHLS)	Staging tables for storing many-to-many relationships between two joining dimension tables.

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

Table Type	Description
Unbounded dimension (_UD)	Tables containing information that is not bounded in transactional database data but should be treated as bounded data in the Oracle Business Analytics Warehouse.
Staging tables for Usage Accelerator (_WS)	Tables containing the necessary columns for the ETL transformations.

Aggregate Tables in Oracle Business Analytics Warehouse

One of the main uses of a data warehouse is to sum up fact data with respect to a given dimension, for example, by date or by sales region. Performing this summation on-demand is resource-intensive, and slows down response time.

Oracle Business Analytics Warehouse precalculates some of these sums and stores the information in aggregate tables. In the Oracle Business Analytics Warehouse, the aggregate tables have been suffixed with _A.

Dimension Class Tables in Oracle Business Analytics Warehouse

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

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

Dimension Tables in Oracle Business Analytics Warehouse

The unique numeric key (ROW_WID) for each dimension table is generated during the load process. This key is used to join each dimension table with its corresponding fact table or tables. It is also used to join the dimension with any associated hierarchy table or extension table. The ROW_WID columns in the Oracle Business Analytics Warehouse tables are numeric.

In every dimension table, the ROW_WID value of zero is reserved for Unspecified. If one or more dimensions for a given record in a fact table is unspecified, the corresponding key fields in that record are set to zero.

Dimension Tables With Business Role-Based Flags

This design approach is used when the entity is logically the same but participates as different roles in the business process.

As an example, an employee could participate in a Human Resources business process as an employee, in the sales process as a sales representative, in the

receivables process as a collector, and in the purchase process as a buyer. However, the employee is still the same. For such logical entities, flags have been provided in the corresponding physical table (for example, W_EMPLOYEE_D) to describe the record's participation in business as different roles.

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

Fact Tables in Oracle Business Analytics Warehouse

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

Helper Tables in Oracle Business Analytics Warehouse

The helper tables are used to solve complex problems that cannot be resolved by simple dimensional schemas.

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

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

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

Hierarchy Tables in Oracle Business Analytics Warehouse

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

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

Mini-Dimension Tables in Oracle Business Analytics Warehouse

Mini-dimension tables include combinations of the most queried attributes of their parent dimensions. They improve query performance because the database does not need to join

the fact tables to the big parent dimensions but can join these small tables to the fact tables instead.

The following table lists the mini-dimension tables in the Oracle Business Analytics Warehouse:

Table Name	Parent Dimension
W_RESPONSE_MD	Parent W_RESPONSE_D
W_AGREE_MD	Parent W_AGREE_D
W_ASSET_MD	Parent W_ASSET_D
W_OPTY_MD	Parent W_OPTY_D
W_ORDER_MD	Parent W_ORDER_D
W_QUOTE_MD	Parent W_QUOTE_D
W_SRVREQ_MD	Parent W_SRVREQ_D

Staging Tables in Oracle Business Analytics Warehouse

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

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

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

Translation Tables in Oracle Business Analytics Warehouse

Translation tables provide multi-language support by storing names and descriptions in each language that Oracle Business Analytics Warehouse supports.

There are two types of translation tables:

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

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

Internal Tables in Oracle Business Analytics Warehouse

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

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

Standard Column Prefixes in Oracle Business Analytics Warehouse

The Oracle Business Analytics Warehouseuses a standard prefix to indicate fields that must contain specific values.

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

Standard Column Suffixes in Oracle Business Analytics Warehouse

The Oracle Business Analytics Warehouse uses suffixes to indicate fields that must contain specific values.

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

System Columns in Oracle Business Analytics WarehouseTables

Oracle Business Analytics Warehouse tables contain system fields. These system fields are populated automatically and should not be modified by the user.

The following table lists the system columns used in data warehouse dimension tables:

System Column	Description
ROW_WID	Surrogate key to identify a record uniquely.
CREATED_BY_WID	Foreign key to the W_USER_D dimension that specifies the user who created the record in the source system.
CHANGED_BY_WID	Foreign key to the W_USER_D dimension that specifies the user who last modified the record in the source system.
CREATED_ON_DT	The date and time when the record was initially created in the source system.
CHANGED_ON_DT	The date and time when the record was last modified in the source system.

System Column	Description
AUX1_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX2_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX3_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
AUX4_CHANGED_ON_DT	System field. This column identifies the last modified date and time of the auxiliary table's record that acts as a source for the current table.
DELETE_FLG	This flag indicates the deletion status of the record in the source system. A value of Y indicates the record is deleted from the source system and logically deleted from the data warehouse. A value of N indicates that the record is active.
W_INSERT_DT	Stores the date on which the record was inserted in the data warehouse table.
W_UPDATE_DT	Stores the date on which the record was last updated in the data warehouse table.
DATASOURCE_NUM_ID	Unique identifier of the source system from which data was extracted. In order to be able to trace the data back to its source, it is recommended that you define separate unique source IDs for each of your different source instances.
ETL_PROC_WID	System field. This column is the unique identifier for the specific ETL process used to create or update this data.
INTEGRATION_ID	Unique identifier of a dimension or fact entity in its source system. In case of composite keys, the value in this column can consist of concatenated parts.
TENANT_ID	Unique identifier for a tenant in a multi-tenant environment. This column is typically be used in an Application Service Provider (ASP)/Software as a Service (SaaS) model.
X_CUSTOM	Column used as a generic field for customer extensions.
CURRENT_FLG	This is a flag for marking dimension records as "Y" in order to represent the current state of a dimension entity. This flag is typically critical for Type II slowly changing dimensions, as records in a Type II situation tend to be numerous.
EFFECTIVE_FROM_DT	This column stores the date from which the dimension record is effective. A value is either assigned by Oracle BI Applications or extracted from the source.
EFFECTIVE_TO_DT	This column stores the date up to which the dimension record is effective. A value is either assigned by Oracle BI Applications or extracted from the source.
SRC_EFF_FROM_DT	This column stores the date from which the source record (in the Source system) is effective. The value is extracted from the source (whenever available).
STC_EFF_TO_DT	This column stores the date up to which the source record (in the Source system) is effective. The value is extracted from the source (whenever available).

Multi-Currency Support for System Columns

The following table lists the currency codes and rates for related system columns:

System Column	Description
DOC_CURR_CODE	Code for the currency in which the document was created in the source system.
LOC_CURR_CODE	Usually the reporting currency code for the financial company in which the document was created.
GRP_CURR_CODE	The primary group reporting currency code for the group of companies or organizations in which the document was created.
LOC_EXCHANGE_RATE	Currency conversion rate from the document currency code to the local currency code.
GLOBAL1_EXCHANGE_RATE	Currency conversion rate from the document currency code to the Global1 currency code.
GLOBAL2_EXCHANGE_RATE	Currency conversion rate from the document currency code to the GLOBAL2 currency code.
GLOBAL3_EXCHANGE_RATE	Currency conversion rate from document currency code to the GLOBAL3 currency code.
PROJ_CURR_CODE	Code used in Project Analytics that corresponds to the project currency in the OLTP system.

Oracle Business Analytics Warehouse Primary Data Values

It is possible for various dimensions to have one-to-many and many-to-many relationships with each other. These kinds of relationships can introduce problems in analyses.

For example, an Opportunity can be associated with many Sales Representatives and a Sales Representative can be associated with many Opportunities. If your analysis includes both Opportunities and Sales Representatives, a count of Opportunities would not be accurate because the same Opportunity would be counted for each Sales Representative with which it is associated.

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

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

About Multi-Language Support in Oracle Business Analytics Warehouse

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

Oracle Business Analytics Warehouse Currency Preferences

Configure global currencies in Functional Setup Manager (FSM).

To set up currencies, see About Configuring Currencies.

The Oracle Business Analytics Warehouse supports the following currency preferences.

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

4

Administering Oracle GoldenGate and Source Dependent Schemas

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

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

- Source Dependent Schema Architecture
- Tasks for Setting Up Oracle GoldenGate and the Source Dependent Schema
- ETL Customization
- Patching
- Troubleshooting Oracle GoldenGate and SDS

In addition to the ETL use case, you can reconcile ledger information available in your Oracle E-Business Suite and Oracle Fusion Applications using Oracle GoldenGate.

- Setting up Ledger Correlation using GoldenGate

Source Dependent Schema Architecture

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

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

- Source Dependent Data Store (SDS): A separate schema on the Oracle Business Analytics Warehouse database that is a replication of the source OLTP systems tables. Also stores deletes and additional optimizations for incremental ETL.
- Oracle GoldenGate: This replication system is deployed on both source and Oracle Business Analytics Warehouse database systems. On the source database system, Oracle GoldenGate supports continuous asynchronous change data capture at a low level in the database, then compresses and ships the changed data across the network

to the target SDS schema on the analytical warehouse database instance. On the target Oracle Business Analytics Warehouse database instance, it receives the changed data from one or more source systems and loads them into the target database, specifically into the SDS schemas, one per ETL OLTP source.

- Oracle Data Integrator (ODI): ODI metadata stores definitions used to generate the SDS schemas and to support the Oracle GoldenGate replication processes.
- Oracle BI Applications SDS Components: Components used to support generation of the SDS schema and Oracle GoldenGate replication processes.

Tasks for Setting Up Oracle GoldenGate and the Source Dependent Schema

Perform these detailed tasks to set up Oracle GoldenGate and SDS.

> **Note:**
>
> You must perform the tasks in this section in the listed sequence.

- Setup Step: Configure Source and Target Database
- Setup Step: Install Oracle GoldenGate on Source and Target Systems
- Setup Step: Configure BI Applications Configuration Manager and Oracle Data Integrator to Support the Source Dependent Schema
- Setup Step: Generate, Deploy, and Populate the Source Dependent Schema Tables on Target Database
- Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Systems
- Setup Step: Start Oracle GoldenGate on Source and Target Systems
- Replicate Views from Source

Setup Step: Configure Source and Target Database

In this step, you create Oracle GoldenGate database users on source and target databases. Unlike other database schemas used by Oracle BI Applications, the SDS and OGG schemas are not automatically created during installation.

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

1. Create the OLTP database user for Oracle GoldenGate.

Each OGG process requires a dedicated database user. On the source system, the OGG user needs to be able to query various metadata.

Secure database practice is to avoid granting privileges to tables not in use, so `SELECT ANY TABLE` is not granted to the OGG database user. Instead, as part of the SDS DDL, `SELECT` privileges are granted only to those tables in the OLTP schema being replicated.

The user creation scripts use the following parameters:

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

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

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

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


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

2. Prepare the OLTP database and redo logs.

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

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

3. Create the target Oracle GoldenGate database user.

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

   The user creation scripts use the following parameters:

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

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

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

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

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

4. Create the SDS database user.

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

Source Instance Name	Model Code	Data Source Number	SDS
Oracle EBS 12.2	EBS_12_2	310	BIAPPS_SDS_ EBS_12_2_310
Siebel CRM 8.2.2	SEBL_8_2_2	625	BIAPPS_SDS_ SEBL_8_2_2_625

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

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

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

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

```
&BIAPPS_SDS_DATA_TS;

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

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

Setup Step: Install Oracle GoldenGate on Source and Target Systems

Download and install Oracle GoldenGate software first on the source and then on the target machines. The software is available from Oracle Technology Network.

See the Oracle GoldenGate Installation and Setup guides for your platform and database:

- *Oracle GoldenGate for Oracle Installation and Setup Guide*
- *Oracle GoldenGate for DB2 LUW Installation and Setup Guide*
- *Oracle GoldenGate for c-tree Installation and Setup Guide*

Installation Recommendations

When installing and configuring the Oracle GoldenGate software, consider the following recommendations:

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

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

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

Example Steps to configure the Oracle GoldenGate processes

These example steps illustrate how to configure the OGG processes. Modify these steps as appropriate for your environment.

For the source system, configure Extract and Data Pump processes. The initial steps in the example below effectively remove an existing instance of both processes. If none already exist, start with the `START MGR` command.

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

STOP MGR

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

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

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

--Start Manager on primary database
START MGR

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

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

Example:

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

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

STOP DP_310
DELETE RMTTRAIL ./dirdat/*
DELETE DP_310

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

--Start Manager on primary database
START MGR

--Primary database capture configuration
```

```
ADD EXTRACT EXT_310, TRANLOG, BEGIN NOW
ADD EXTTRAIL ./dirdat/tr, EXTRACT EXT_310, MEGABYTES 50

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

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

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

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

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

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

--Start Manager on target database
START MGR

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

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

Example:

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

--Stop GG processes
STOP REP_310
DELETE REP_310

--Delete CHECKPOINTTABLE
DELETE CHECKPOINTTABLE

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

--Start Manager on target database
START MGR

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

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

Setup Step: Configure Configuration Manager and ODI to Support the Source Dependent Schema

Configure Configuration Manager and ODI to support the Source Dependent Schema.

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

1. In Configuration Manager, select the **Source Instance**.
2. Click **Manage Data Load Parameters**.
3. Locate the IS_SDS_DEPLOYED parameter in the list.
4. In the Global Parameter Value, replace <Edit Value> with **Yes**.

   A warning is displayed indicating that the parameter is being updated globally for all Fact and Dimension Groups.
5. Click **Yes** to confirm or, if you prefer, set the global parameter to **No**, and then edit the parameter value at the Fact Group or Dimension Group level.

Adding SDS Physical Schemas in ODI

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

1. In ODI Studio's Topology Navigator, expand the Oracle technology in the Physical Architecture.
2. Right-click **BIAPPS_DW** and select **New Physical Schema**.
3. In the Definition, set Schema (Schema) and Schema (Work Schema) both to the SDS schema owner.
4. Select **Flexfields**.
5. For the DATASOURCE_NUM_ID flex field, uncheck the **Default** checkbox and assign the DSN value associated with that source as defined in Configuration Manager.
6. Save the physical schema definition.

   Ignore the message about the physical server not being assigned a context.

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

Generate and run the Data Definition Language (DDL) to create the SDS tables on the SDS schema in the target database.

1. Generate the SDS DDL.

Procedures are provided to generate the required objects to enable the SDS. To generate the required DDL, in ODI Designer execute the 'Generate DDL - SDS' scenario found under **BI Apps Project**, **Components**, **SDS**, **Oracle**, then **Generate SDS DDL**. Provide an appropriate context when prompted. As the procedure can only accept a single source type, this process needs to be repeated for each different type of Source system to be enabled.

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

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

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

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

2. Grant privileges to OLTP tables.

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

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

3. Create the SDS tables.

   The SDS DDL generator procedure creates a .SQL script that follows the naming convention BIA_SDS_Schema_DDL_<*unique ID*>.sql which contains the `CREATE`

or `ALTER` DDL statements to create or alter the tables in the SDS schema. Execute the SQL in this file against the SDS schema.

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

4. Perform initial Load of the SDS tables: create database link to OLTP database.

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

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

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

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

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

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

5. Perform initial load of the SDS tables: execute procedure to generate DML script.

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

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

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

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

6. Perform initial load of the SDS tables: execute DML script on SDS database.

   The resulting DML script can be executed using the SDS schema owner or the BIAPPS DW schema owner. If executed by the BIAPPS DW schema owner, this

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

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

7. Create SDS indexes and analyze the SDS schema.

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

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

Parameter files are used to control how Oracle GoldenGate operates. These files are deployed to the source system, where the Extract and Data Pump processes are executed, and the target system, where the Replicat process is executed.

An ODI procedure generates these parameter files based on the metadata defined in ODI. A scenario that executes this procedure is provided to generate the Oracle GoldenGate parameter files to populate the SDS.

Generate Oracle GoldenGateParameter Files

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

Parameter	Description
PARAM_FILE_LOCATION	Location on machine where ODI client is running where parameter files will be created. Example: `C:\temp\`
DATASOURCE_NUM_ID	Datasource Num ID value associated with the particular source for which parameter files are to be generated. Example: 310
DATAPUMP_NAME	Name of the Datapump Process specified when installing Oracle GoldenGate on the source machine. Limit is eight characters. Suggested naming convention is DP_*Datasource Num Id*, for example DP_310.
EXTRACT_NAME	Name of the Primary Extract Process specified when installing Oracle GoldenGate on the source machine. Limit is eight characters. Suggested naming convention is EX_*Datasource Num Id*, for example EXT_310.

Parameter	Description
EXTRACT_TRAIL	Path and name of trail file on source system. Can be a relative or fully qualified path, though actual file name must be two characters. In the example below, 'tr' is the name of the trail file. Example: `./dirdat/tr`
DEFSFILE	The relative or fully qualified path on the source system where the DEFGEN definition file should be created and file name. This value is included in the DEFGEN.prm parameter file that is generated by this procedure. The DEFGEN utility is executed on the source database, so the path provided must be a path available on the system the source database runs on. Suggested naming convention is DEF_*Datasource Num Id*.def. Example: `./dirdef/DEF_310.def`
SOURCE_GG_USER_ID	Database user dedicated to the Oracle GoldenGate processes on the source database. Example: GG_USER
SOURCE_GG_PASSWORD	Password for the database user dedicated to the Oracle GoldenGate processes on the source database. By default, the password is stored as clear text in the generated parameter file. If an encrypted value is desired, use the ENCRYPT PASSWORD utility and edit the generated parameter files accordingly. Example: GG_PASSWORD
SOURCE_PORT	Port used by the Oracle GoldenGate Manager Process on the source system. The default value when Oracle GoldenGate is installed is 7809.
REPLICAT_NAME	Name of the Replicat Process specified when installing Oracle GoldenGate on the target machine. Limit is eight characters. Suggested naming convention is REP_*Datasource Num Id*, for example REP_310
SOURCE_DEF	This is the Source Definitions file created by executing the DEFGEN utility on the source database and copied over to the target machine. This can be either a relative or fully qualified path to this definition file on the target system. Include the /dirdef subfolder as part of the path. Suggested naming convention is DEF_*Datasource Num Id*.def, for example ./dirdef/DEF_310.def Note that the file name is usually the same as the one defined for DEFSFILE but the paths are usually different as DEFSFILE includes the path where Oracle GoldenGate is stored on the source system, while SOURCE_DEFS includes the path where Oracle GoldenGate is installed on the target system.
REMOTE_HOST	IP address or Host Name of the target machine where the Replicat process runs.
REMOTE_TRAIL	Path and name of the trail file on target system. Can be a relative or fully qualified path though the actual file name must be two characters. In the example below, 'tr' is the name of the trail file. Example: `./dirdat/tr`
BIA_GG_USER_ID	Database user dedicated to the Oracle GoldenGate processes on the target database, for example GG_USER

Parameter	Description
BIA_GG_PASSWORD	Password for the database user dedicated to the Oracle GoldenGate processes on the target database. By default, the password is stored as clear text in the generated parameter file. If an encrypted value is desired, use the ENCRYPT PASSWORD utility and edit the generated parameter files accordingly. Example: GG_PASSWORD
BIA_GG_PORT	Port used by the Oracle GoldenGate Manager Process on the target system. The default value when Oracle GoldenGate is installed is 7809.

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

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

About JD Edwards Support

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

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

Component	Extract Name	Data Pump Name	Extract Trail	Defs File	Replicat Name	Replicat Trail	Source Defs
Control	EX_410A	DP_410A	./dirdat/ta	./dirdef/ DEF_310A.def	REP_410 A	./dirdat/ta	./dirdef/ DEF_310A.def
Data	EX_410B	DP_410B	./dirdat/tb	./dirdef/ DEF_310B.def	REP_410 B	./dirdat/tb	./dirdef/ DEF_310B.def
Data Dictionary	EX_410C	DP_410C	./dirdat/tc	./dirdef/ DEF_310C.def	REP_410 C	./dirdat/tc	./dirdef/ DEF_310C.def

Component	Extract Name	Data Pump Name	Extract Trail	Defs File	Replicat Name	Replicat Trail	Source Defs
System	EX_410D	DP_410D	./dirdat/td	./dirdef/ DEF_310D.def	REP_410 D	./dirdat/td	./dirdef/ DEF_310D.def

About PeopleSoft Learning Management Support

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

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

Component	Extract Name	Data Pump Name	Extract Trail	Defs File	Replicat Name	Replicat Trail	Source Defs
HCM Pillar	EX_518A	DP_518A	./dirdat/ta	./dirdef/ DEF_518A.def	REP_518 A	./dirdat/ta	./dirdef/ DEF_518A.def
LM Pillar	EX_518B	DP_518B	./dirdat/tb	./dirdef/ DEF_518B.def	REP_518 B	./dirdat/tb	./dirdef/ DEF_518B.def

Configure the Source System

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

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

- ADD_TRANDATA.txt

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

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

Edit the Extract parameter file

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

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

To support such features, edit the generated parameter files using the GGSCI EDIT PARAMS <parameter file> command. Also edit the generated param files to implement various tuning options that are specific to the environment.

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

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

Save and close the file.

Enable Table Level Logging

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

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

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

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

> **Note:**
>
> If a table does not have a primary key or any unique indexes defined, you may see a warning message like the following. This is a warning that a 'pseudo' unique key is being constructed and used by Oracle GoldenGate to identify a record. Performance is better if a primary key or unique index is available to identify a record but as we generally cannot add such constraints to an OLTP table when they do not already exists, Oracle GoldenGate creates this pseudo unique key.
>
> WARNING OGG-00869 No unique key is defined for table 'FA_ASSET_HISTORY'. All viable columns will be used to represent the key, but may not guarantee uniqueness. KEYCOLS may be used to define the key.

Generate Data Definition File on the Source System

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

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

```
GGSCI>EDIT PARAMS DEFGEN
```

Save and close the file and exit GGSCI, then run the DEFGEN utility. The following is an example of executing this command on UNIX:

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

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

Configure the Target System

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

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

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

Edit the Replicat Parameter File

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

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

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

Save and close the file, and exit GGSCI.

Create a Checkpoint Table (Optional)

The procedure does not account for a checkpoint table in the target system. A checkpoint table is not required but is recommended; in which case, create a checkpoint table and edit the GLOBALS param file to reference this table.

Start the GGSCI command utility:

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

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

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

Setup Step: Start Oracle GoldenGate on Source and Target Systems

Start Oracle GoldenGate on source and target systems.

1. Start Oracle GoldenGate on the source system.

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

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

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

   Example:

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

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

2. Start Oracle GoldenGate on the target system.

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

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

   Example:

   ```
   START MGR

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

Replicate Views from Source

The ready-to-use "Generate SDS DDL" schema (warehouse) for Oracle GoldenGate creates source "Views" as "Tables". This is the expected behavior because Oracle GoldenGate can't replicate views from source.

When Oracle GoldenGate is the replication technology, the mappings that use a view as source are set to run in the non-SDS mode. Such mappings try to directly connect to the source OLTP. If you have any security restrictions on connecting directly to the source OLTP, then such mappings fail.

1. Create tables or materialized views on top of views in the OLTP schema and trigger a process to refresh these objects in OLTP before the ETL process starts.
2. Replicate these materialized views to the target tables in the SDS using Oracle GoldenGate.

Ensure that the SDS schema has these views as tables and that the first time replication is a full load of views to the target SDS tables.

3. Populate `CDC$_SRC_LAST_UPDATE_DATE` and `CDC$_DML_CODE` with thw current date and `'I'`.

   For incremental, you can create a materialized view on the source OLTP on top of the view using this SQL command:

   ```
   CREATE MATERIALIZED VIEW test_mv BUILD IMMEDIATE
   REFRESH COMPLETE ON DEMAND AS select * from test_v;
   ```

   In this SQL command, you can set REFRESH to FORCE or COMPLETE. For any change in the underlying view definition, you must recreate the materialized view before you run the load plans. Use the Oracle Database procedure, DBMS_MVIEW.REFRESH, to recreate the materialized view.

4. From the generated Replicat parameter file, remove the complete entry for these views and add a normal entry like:

   ```
   Test_v
   MAP ggtest.test_mv; TARGET ankur_test.test_v,COLMAP (USEDEFAULTS,
   CDC$_SRC_LAST_UPDATE_DATE = @GETENV
   ('GGHEADER','COMMITTIMESTAMP'),CDC$_DML_CODE =
       'I'),KEYCOLS(COL1,COL2,COL3);
   ```

> **Note:**
>
> An Oracle Data Integrator procedure generates the Oracle GoldenGate Replicat parameter files for Oracle BI Applications.

If no PK is defined on the target system, then you must select the appropriate key columns and define them in the Replicat processes of Oracle GoldenGate using the KEYCOLS keyword. In Oracle Data Integrator, alter the source OLTP connection to point to the SDS. If you encounter performance issues because the views contain a large amount of data, then you must explore an alternate approach to achieve this view replication.

List of such views in E-Business Suite:

- PO_VENDORS
- JTF_TASK_ASSIGNMENTS
- GL_SETS_OF_BOOKS
- ORG_ORGANIZATION_DEFINITIONS
- BOM_BILL_OF_MATERIALS
- MTL_ITEM_LOCATIONS_KFV
- CST_ORGANIZATION_DEFINITIONS
- CS_LOOKUPS
- PA_EGO_LIFECYCLES_PHASES_V
- GL_CODE_COMBINATIONS_KFV

- PON_AUCTION_HEADERS_ALL_V
- MTL_ITEM_CATALOG_GROUPS_B_KFV
- AP_INVOICES_V
- PER_WORKFORCE_CURRENT_X

List of such views in PeopleSoft:

- CM_ITEM_METH_VW
- DEPENDENT_BENEF
- EMPLOYMENT
- PERSON_ADDRESS

ETL Customization

Learn about SDS considerations for ETL customization.

Adding a Non-Custom Source Column to an Existing ETL Task

All columns in each source table are replicated. If you extend an existing ETL task with a column that is a standard part of the source system, no special steps are required.

Adding a Custom Source Column to an Existing ETL Task

If you add a custom source column to the ETL task, the Oracle GoldenGate process already extracts from this table and needs to be modified to include this column in the extract. In addition, the SDS table must be altered to include this column.

Run the RKM to add the column to the ODI model, then use the SDS DDL generator in incremental mode to add this column to the SDS table. If the SDS has already been populated with data, repopulate it by running the SDS Copy to SDS procedure, providing the customized table in the Table List parameter.

Adding a Non-Custom Source Table to an Existing ETL Tas

In cases where an ETL task is customized to use an additional table that is included as part of the standard OLTP application, if the table is already used by another ETL task then that table should already exist in the ODI model and is already replicated in the SDS. No special steps are required.

If the table is not already used by any other ETL task, run the RKM to add the table to the ODI model and use the SDS DDL generator in incremental mode to add this table to the SDS schema. Use one of the initial load options with this table in the Table List to repopulate the table. Regenerate the SDS parameter files to ensure the table is included as part of the replication process.

Creating a Custom ETL Task

If a custom ETL task sources a table that is already extracted from, no special steps are required. However, if the custom task extracts from a new table that is not already included as part of the standard Oracle BI Applications source-specific model, run the RKM to add the table to the ODI model and use the SDS DDL generator in incremental mode to add this table to the SDS schema. Use one of the initial load options with this table in the Table List to repopulate the table. Regenerate the SDS parameter files to ensure the table is included as part of the replication process.

Patching

After releasing Oracle BI Applications, Oracle may release patches. This section discusses patches that impact SDS related content and considerations when deploying those patches.

Patch Applied to ODI Datastores or Interfaces

ODI datastores and interfaces are the only Oracle BI Applications content that impacts SDS related content. Applied patches that impact OLTP-specific datastores are relevant to the SDS.

It is possible that an applied patch could change the definition of an OLTP-specific datastore, for example adding a column or changing a column's size or datatype. A patch could also introduce a new table. In these cases, run the SDS DDL generator in incremental mode, providing the list of datastores that are patched. Execute the generated DDL against the SDS schema. In case of a new column or table being introduced, run the initial load, specifying just the new or changed table in the table list in the provided procedure.

A patch could impact an interface by adding a new OLTP table from which data must be extracted. In the previous step, you would have generated the DDL and DML to create and populate this table. Run the Oracle GoldenGate parameter generator procedure to recreate the required parameter files and redeploy to the source and target systems. To create and recreate parameter files, see Setup Step: Generate and Deploy Oracle GoldenGate Parameter Files to Source and Target Machines.

Patch Applied to SDS-Related Procedure

In the case an SDS-related procedure is replaced by a patch, depending on the nature of the reason for the patch, it may be necessary to re-execute the procedure and re-deploy its output. If the patch is related to the SDS DDL or SDS Copy procedures, the procedure can be run in incremental mode to make some changes to the SDS or in full mode to completely replace the SDS. The patch notes will describe exactly what must be done to implement the patched procedure.

Troubleshooting Oracle GoldenGate and SDS

Review these troubleshooting tips and resolutions for common errors encountered during setup of Oracle GoldenGate and SDS.

Topics:

- Create the SDS Tables
- Using the DML Option to Perform an Initial Load
- Create SDS Indexes and Analyze the SDS schema

Create the SDS Tables

If you encounter any issues with the script generated by the GENERATE_SDS_DDL procedure, verify the following have been correctly set.

- The model you are specifying is associated with a logical schema.

- The logical schema's DATASOURCE_NUM_ID flexfield is assigned a numeric value. A value is automatically assigned when a datasource is registered in Configuration Manager.
- The logical schema is mapped to a physical schema in the context (for example, Global) you are executing the procedure with. The physical schema is automatically mapped to the Global context when the datasource is registered in Configuration Manager.
- The physical schema's DATASOURCE_NUM_ID flexfield is assigned the same numeric value as the logical schema. A value is automatically assigned when a datasource is registered in Configuration Manager.
- Under the same context, a physical schema is mapped to the DW_BIAPPS11G logical schema, for example BIAPPS_DW.OLAP.
- A new SDS physical schema has been added to the same physical server, for example BIAPPS_DW.SDS_EBS_12_2_310. This physical schema is manually added.
- The physical schema's DATASOURCE_NUM_ID flexfield is assigned the same numeric value as used previously. This value is manually assigned.

The following are some common error messages and issues you may encounter.

- `com.sunopsis.tools.core.exception.SnpsSimpleMessageException: com.sunopsis.tools.core.exception.SnpsSimpleMessageException: Exception getSchemaName("[logical schema name]", "D") : SnpPschemaCont.getObjectByIdent : SnpPschemaCont does not exist`

  Verify the logical schema is mapped in the context you are executing the procedure with.

- `java.lang.Exception: The application script threw an exception: java.lang.Exception: Model with code '[logical schema]' does not exist`

  Verify the logical schema associated with your model has been assigned a value for the DATASOURCE_NUM_ID flexfield.

- `java.lang.Exception: The application script threw an exception: java.lang.Exception: Can't find physical schema for connection for DW_BIAPPS11G with DSN 310 in context Global`

  Verify a physical schema is created under the Data Warehouse physical server and assigned the same DATASOURCE_NUM_ID value as assigned to the OLTP.

Using the DML Option to Perform an Initial Load

If you encounter any issues with the script generated by the `COPY_OLTP_TO_SDS` procedure, verify the following have been correctly set.

A database link with the name DW_TO_OLTP is created in the database user schema used by the data warehouse Data Server (BIAPPS_DW) that points to the OLTP database. The procedure is executed by this user so Oracle looks for this database link in the user's schema, not the SDS schema. You still need a database link with this name in the SDS schema for other reasons, so you have a total of two database links to the same source database.

The following are some common error messages and issues you may encounter.

- `ODI-1228: Task Copy SDS Data (Procedure) fails on the target ORACLE connection BI_APPLICATIONS_DEFAULT`

PL/SQL: ORA-00942: table or view does not existPLS-00364: loop index variable 'COL_REC' use is invalid"

Verify a database link named DW_TO_OLTP exists in the schema owned by the database user associated with the DW physical server.

- `Insert statement only populates the CDC$ columns`

  The script has statements such as the following where only the CDC$ columns are populated:

```
truncate table SDS_EBS122_FULL.HR_LOCATIONS_ALL;
INSERT /*+ APPEND */ INTO SDS_EBS122_FULL.HR_LOCATIONS_ALL
(CDC$_SRC_LAST_UPDATE_DATE, CDC$_RPL_LAST_UPDATE_DATE, CDC$_DML_CODE) SELECT
SYSDATE, SYSDATE, 'I'
FROM HR_LOCATIONS_ALL@DW_TO_OLTP;
```

  Verify the database link DW_TO_OLTP is pointing to the correct OLTP database. The procedure gets a column list from the data dictionary on the OLTP database for the tables that correspond to the SDS tables. If the database link points to the wrong database, a column list will not be retrieved.

Create SDS Indexes and Analyze the SDS Schema

When executing the script to create indexes and primary key constraints on the SDS tables, you may see some of the following error or warning messages.

- **"such column list is already indexed"**

  You may see this message when executing the script that creates the indexes. This message can be ignored.

  Oracle GoldenGate works best when a primary key is defined on the target table in order to determine which record to update. If a primary key is not found, the replicat process searches for a unique index to determine which record to update. The definition of the tables in the SDS is based on the definition in the source system (leveraging both the application dictionary and data dictionary). If the table does not have a primary key in the source system but does have multiple unique indexes, a primary key may be added in the ODI definition to ensure Oracle GoldenGate can correctly identify the record to be updated. This primary key may be defined on the same column that a unique index is already defined on. The DDL script creates the primary key as an index and a constraint before creating the unique index. When creating the unique index, the database will report that the column is already indexed.

- **"column contains NULL values; cannot alter to NOT NULL"**

  This error can occur when a primary key constraint is being created. Primary key constraints are introduced in ODI when a primary key is defined in the OLTP system. A primary key constraint may also be introduced in ODI when there is no primary key in the OLTP system for the table but the table has multiple unique indexes; in this case, a primary key constraint is introduced to ensure Oracle GoldenGate does not use a unique index that may not correctly identify a record for updates. This error can occur for two reasons:

  – **OLTP table has Primary Key Constraint**

    Due to differences in patches and subversions, it is possible the OLTP instance used to originally import the datastores from had a primary key constraint that differs from the OLTP release you are using. If the OLTP table

has a primary key constraint, ensure the definition in ODI matches this primary key. If there is a difference, you can modify the Index DDL script to use the proper definition to create the primary key constraint in the target database. You should also update the constraint in ODI to match this definition.

If the OLTP and ODI definitions of the primary key constraint match, it is possible the initial load process did not populate one or more of the columns that make up the primary key. If the primary key includes a LOB or LONG datatype, data is not replicated in these columns, which would leave the column empty. In this case, no unique index or primary key can be created, and without data in this column the record cannot be uniquely identified. Any ETL task that extracts from this table needs to be modified to extract directly from the OLTP system. This is done by modifying the load plan step for this task, overwriting the IS_SDS_DEPLOYED parameter for that load plan step to a setting of 'N'.

If the OLTP and ODI definitions of the primary key constraint match and the key does not include a column that has either the LOB or LONG datatype, review the initial load files and verify whether the column is populated or not. See Using the DML Option to Perform an Initial Load.

- **OLTP table does not have Primary Key Constraint**

  Primary key constraints in the ODI model are introduced when a primary key may not exist in the original table. This primary key generally matches an existing unique index. Due to differences in patch and subversions for a given OLTP release, it is possible that the instance used when importing a unique index had a column that is not nullable but in another OLTP release, that column may be nullable. Unique indexes allow null values but primary keys do not. In this case, a unique index is created for the SDS table but the primary key constraint fails to be created. Oracle GoldenGate uses the first unique index it finds (based on the index name) to identify a record for update; if the index that the primary key constraint is based on is not the first index, rename this index in ODI to ensure it will be the first. Generally, the first unique index is the correct index to use to identify a record, in which case this error can be ignored.

- **"cannot CREATE UNIQUE INDEX; duplicate keys found"**

  Due to differences in patch and subversions for a given OLTP release, it is possible that the instance used when importing a unique index uses a different combination of columns than are used in your particular release of the same OLTP. For example, the OLTP subversion used to import an index uses 3 columns to define the unique index but a later subversion uses 4 columns, and you are using this later subversion. Check the definition of the unique index in your OLTP instance and modify the index script and corresponding constraint definition in ODI to match.

Setting up Ledger Correlation using Oracle GoldenGate

Reconcile ledger information available in your Oracle E-Business Suite and Oracle Fusion Applications using Oracle GoldenGate.

If you are analyzing data sourced from Oracle E-Business Suite and Oracle Fusion Applications and are using the Fusion GL Accounting feature, then you can drill in analyses from Fusion GL balances to related detailed EBS ledger information. This ledger correlation is supported by Oracle GoldenGate replication, and requires Oracle GoldenGate configuration on your source systems.

After setting up the required Oracle GoldenGate configurations on the sources, you need to expose the following Presentation columns in the applicable reports to support drilling:

- **Target GL Account ID** for **GL Account** of subject area **Financials – GL Balance Sheet**.
- **Target Ledger ID** for Ledger
- **Target Fiscal Period ID** for Time
- **Dim – Ledger.Source ID** (Data Source Num ID)

For information about working with Presentation columns, see *Managing Metadata Repositories for Oracle Analytics Server*.