Oracle® Fusion Middleware

Migrating From Oracle Warehouse Builder to Oracle Data Integrator

12*c* (12.2.1)

E57356-02

April 2016

This document describes migration from Oracle Warehouse Builder to Oracle Data Integrator 12c (12.2.1).

ORACLE

Oracle Fusion Middleware Migrating From Oracle Warehouse Builder to Oracle Data Integrator, 12*c* (12.2.1)

E57356-02

Contents

A Message Reference

B Reference to Migration Details

C Migration Details for Operators

D Special Migration Cases

E Known Issues and Solutions

List of Examples

List of Tables

Preface

This document describes migration from Oracle Warehouse Builder 11gR2 (11.2.0.4) to Oracle Data Integrator 12c (12.2.1).

Audience

This document is intended for developers and administrators who will perform the migration. Knowledge of data integration and Oracle Warehouse Builder is assumed.

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.

Related Documents

For more information, see the following documents in *Oracle Data Integrator Library*:

- *Release Notes for Oracle Data Integrator*
- *Understanding Oracle Data Integrator*
- *Developing Integration Projects with Oracle Data Integrator*
- *Installing and Configuring Oracle Data Integrator*
- *Upgrading Oracle Data Integrator*
- *Integrating Big Data with Oracle Data Integrator*
- *Application Adapters Guide for Oracle Data Integrator*
- *Developing Knowledge Modules with Oracle Data Integrator*
- *Connectivity and Knowledge Modules Guide for Oracle Data Integrator*
- *Migrating From Oracle Warehouse Builder to Oracle Data Integrator*

- *Oracle Data Integrator Tool Reference*
- *Data Services Java API Reference for Oracle Data Integrator*
- *Open Tools Java API Reference for Oracle Data Integrator*
- *Getting Started with SAP ABAP BW Adapter for Oracle Data Integrator*
- *Java API Reference for Oracle Data Integrator*
- *Getting Started with SAP ABAP ERP Adapter for Oracle Data Integrator*
- *Oracle Data Integrator 12c Online Help*, which is available in ODI Studio through the JDeveloper Help Center when you press **F1** or from the main menu by selecting **Help**, and then **Search** or **Table of Contents**.

Conventions

The following text conventions are used in this document:

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

1

Understanding the Migration Process

This chapter provides an overview of migration from Oracle Warehouse Builder (OWB) to Oracle Data Integrator (ODI).

The following topics are addressed here:

About Migration

ODI is Oracle's strategic product for heterogeneous data integration. Because many Oracle Database customers have significant investment in OWB, Oracle supports a phased migration from OWB 11gR2 (11.2.0.4) to ODI 12c. The following features are provided to make the transition to ODI easier:

- ODI 12c supports the execution and administration of OWB 11gR2 jobs directly within ODI Studio and ODI Console, providing a single orchestration and monitoring solution. This feature enables you to migrate OWB objects over a longer period of time and in a way that makes sense for your business. For more information about this feature, see "OdiStartOwbJob" in the tools reference section of the *Developer's Guide for Oracle Data Integrator*.
- ODI 12c supports an easier mapping between OWB 11gR2 concepts and objects and their ODI 12c counterparts. A migration utility is provided that automatically translates many OWB objects and mappings into their ODI equivalents. For more information about the migration utility, see About the Migration Utility.

About the Migration Utility

The migration utility is a command-line tool that enables you to migrate design-time metadata from OWB to ODI. Runtime data and physical objects are not migrated. The migration utility uses the settings in the migration utility configuration file to perform the migration.

The migration utility is an aid only to assist migration of design-time metadata from OWB to ODI. OWB and ODI are two different products with overlapping but not identical features, and hence behaviors of migrated artifacts will differ. Customers shall expect to do post-migration work to ensure the migrated artifacts in ODI are working as expected. Post-migration work shall include but is not limited to

examining the migrated artifacts, making modifications in order to obtain desired behaviors in ODI, and executing artifacts in ODI runtime environment.

For information about obtaining the patch, see Migration Requirements.

For information about planning the migration, see Roadmap for Migrating.

What Is and Is Not Migrated

The migration utility is an aid to migration only, and not all types and variants of OWB objects are migrated. Manual effort should be expected including further modifications of the migrated Mappings in ODI or extensive testing to verify the Mappings.

The following topics are addressed here:

- Objects That Are Migrated
- Objects That Are Not Migrated

Objects That Are Migrated

The following OWB objects are supported for migration when you run the migration utility:

- modules (source and target)
- locations
- data objects
  - table (columns, keys, indexes)
  - view (columns, keys)
  - materialized view (columns, keys, indexes)
  - external table (columns)
  - file (records, fields)
  - sequence
- mappings
  - classic mappings
  - Code Template mappings
  - pluggable mappings
- mapping operators
  - Aggregator
  - Constant
  - Deduplicator
  - Expression
  - External Table
  - Filter
  - Flat File
  - Joiner

  - Key Lookup
  - Mapping Input Parameter
  - Materialized View
  - Pivot
  - Pre/Post Mapping Process
  - Sequence
  - Set
  - Sorter
  - Splitter
  - Subquery
  - Table
  - Table Function
  - Transformation

    Note: Transformation objects are actually not migrated, but transformation operator in mapping is migrated as expression component in ODI, only if the transformation object is present in ODI repository.
  - Unpivot
  - View

Objects That Are Not Migrated

The following OWB objects are not supported for migration when you run the migration utility:

- data objects
  - table (partitions, attribute sets, data rules)
  - view (attribute sets, data rules)
  - materialized view (partitions, attribute sets, data rules)
  - external table (data rules, locations)
  - sequence (columns)
- dimensional modeling metadata
- Oracle Discoverer metadata and derived Oracle Business Intelligence Suite Enterprise Edition (OBI EE) metadata
- custom transformations, PL/SQL (procedure, package, and so on)
- queues, streams, CDC (Change Data Capture) configurations, user-defined types
- process flow
- mappings using dimension and cube, cursor-based maps, name and address, match-merge, data rules, data auditors, iterators, expand, construct, Anydata Cast, Data Generator
- data quality, data profiles, data auditors
- configuration details (security, user extensions, transportable modules, schedules/collections, user folders)

- OWB Experts
- OMB*Plus scripts
- Internal variable, which is used by the OWB runtime during code generation, for example, get_model_name

Roadmap for Migrating

The activities to migrate from OWB to ODI would require considerable amount of planning ahead and involve multiple teams and resources. An overall plan should be in place and discussed with all involved parties before the actual activities are carried out.

The overall plan should include the following suggested phases:

- **Pre-Migration phase**

  Helps to prepare the environment for migration.

- **Planning phase**

  Helps all parties involved to be familiar with the Migration Utility and learn about what it can do and its limitation, and hence identify potential gaps that would require alternate migration activities.

- **Using the Migration Utility phase**

  Actually does the migration using the utility but also identifies objects that cannot be migrated.

- **Manual migration phase**

  Handles alternate migration activities for objects that cannot be migrated by the Migration Utility

- **Post Migration Development phase**

  The migrated solution (using the Migration Utility or by manual) is reviewed, re-examined, and compared with the OWB solution to ensure the same end results are achieved. Note that additional changes or development are expected on the migrated solution to achieve the same result.

- **Post Migration Testing/QA phase**

  When the migrated solution has been examined to work the same as in OWB and enters into full testing and QA.

- **Rolling out the ODI Solution phase**

  The final phase when the ODI solution is rolled out. One should plan on gradually cutting over from the original OWB instance to the new migrated ODI instance until all the new jobs in ODI are working satisfactorily. That is, both systems would be kept up and running in production until the last OWB job are moved over to ODI and tested to work.

Table 1–1 provides a high-level summary of the steps required to migrate from OWB to ODI. The table also lists where to find more information for each step.

Table 1–1 OWB to ODI Migration Roadmap

Step	Description	Documentation
Phase: Pre-Migration Phase		
The goal of this phase is to prepare the environment for migration.		
Back up existing OWB repositories	Before running the migration utility, backup your existing OWB repositories.	See OWB documentation.
Verify your system environment	Before running the migration utility, verify that your system meets requirements and that you are not connected to the design repository.	See Migration Requirements
Phase: Planning		
The goal of this phase is to get familiar with the Migration Utility, learn about what it does and its limitations, identify potential gaps that the Migration Utility may not be able to assist with, plan the migration activities using the Migration Utility and alternate migration activities without using the Migration Utility.		
Review the entire Migration Utility document, especially the section on "Supported and Unsupported objects".	Make sure you understand what will and will not be migrated.	See What Is and Is Not Migrated
Edit the migration utility configuration file for a test migration.	Edit the migration utility configuration file and make sure the settings are correct for your environment. The configuration file contains connection information and other details required for migration. Set MIGRATION_MODE to FAST_CHECK or DRY_RUN to do a test run of the Migration Utility.	See Creating the Migration Utility Configuration File
Perform a test migration by running the migration utility in FAST_CHECK or DRY_RUN mode.	Run the migration utility to migrate OWB objects to ODI using the settings in the migration utility configuration file. Before running the migration utility, verify that you are not connected to the design repository.	See Using the Migration Utility to Migrate
Review the migration utility log file	After migration is complete, review the migration utility log file. The file contains details about objects that were migrated, and error messages if errors occurred.	See Reviewing the Migration Utility Log File
Review the migration utility exclusion report	After test migration is complete, review the migration utility exclusion report. The report provides a summary of objects that can be migrated, and lists whether migration succeeded or failed for each object. For objects excluded from migration, manual migration steps will be needed.	See Reviewing the Migration Utility Exclusion Report
Finalize migration plan	Based on the test migration run and the result, the objects that will be migrated by the migration utility and those that cannot be migrated by the migration utility will be known. For those objects that cannot be migrated, some manual effort will be needed to recreate these objects in ODI. Create a list of all these objects that will require a manual migration.	
Phase: Using the Migration Utility to do the migration		
The goal of this phase is to actually perform the migration of objects that can be migrated by the Migration Utility.		

Table 1–1 (Cont.) OWB to ODI Migration Roadmap

Step	Description	Documentation
Edit the migration utility configuration file	Edit the migration utility configuration file and make sure the settings are correct for your environment. The configuration file contains connection information and other details required for migration.	See Creating the Migration Utility Configuration File
Run the migration utility to perform the migration using MIGRATION_MODE=RUN	Run the migration utility to migrate OWB objects to ODI using the settings in the migration utility configuration file. Before running the migration utility, verify that you are not connected to the design repository.	See Using the Migration Utility to Migrate
Review the migration utility log file	After migration is complete, review the migration utility log file. The file contains details about objects that were migrated and error messages if errors occurred.	See Reviewing the Migration Utility Log File
Review the migration utility exclusion report	After migration is complete, review the migration utility exclusion report. The report provides a summary of objects that were migrated, and lists whether migration succeeded or failed for each object.	See Reviewing the Migration Utility Exclusion Report
Verify your migration	In ODI Studio, connect to your ODI environment and perform post-migration testing to verify your migration.	See Verifying Your Migration
Phase: Manual Configuration		
For the objects not migrated by the Migration Utility, manual migration will be needed.		
Create objects in ODI manually	For any objects not migrated by the Migration Utility, some manual effort will be needed to recreate these objects in ODI. The list of objects excluded from migration by Migration Utility is noted in the Planning phase above.	
Verify your migration	In ODI Studio, connect to your ODI environment and perform post-migration testing to verify your migration.	See Verifying Your Migration
Phase: Post-Migration Development		
After running the migration plans (either using the Migration Utility or manual steps), the migrated repository should be examined, reviewed and verified. This should be the most crucial and probably the biggest phase of any migration project. It shall involve reviewing each migrated artifact, executing all executable artifacts in ODI, as well as examining results. It is expected that the behavior of migrated artifacts will not be the same as in OWB and modifications may be needed to make the artifact behave as desired. Customers shall plan to invest significant amount of time in this phase.		
Verify your migration	In ODI Studio, connect to your ODI environment and perform post-migration testing to verify your migration.	See Verifying Your Migration

Table 1–1 (Cont.) OWB to ODI Migration Roadmap

Step	Description	Documentation
Review gaps or differences, re-create or re-implement existing logic	For artifacts that do not execute with the same results as in OWB, review the OWB artifacts and compare with the corresponding ODI artifacts. The ODI artifacts may need tweaking or re-design for the artifact to behave similar to the OWB artifact. One may need to re-create or re-implement the OWB logic in ODI.	See Special Migration Cases
Phase: Post Migration Testing / QA		
After all the mappings have been migrated (by the Migration Utility or manually created) and verified in the phases above, the migrated ODI solution should be handed over to testing team for full QA testing.		
Phase: Rolling out the ODI solution		
Before cutting over to ODI, the migrated ODI solution should run in concurrent with OWB until all the artifacts in ODI have been reviewed, verified and stabilized.		
When the ODI solution is running as expected or desired, the cut-over from OWB can be done.		

2

Preparing to Migrate

This chapter lists migration requirements and describes how to create the configuration file used for migration.

The following topics are addressed here:

- Migration Requirements
- Creating the Migration Utility Configuration File

Migration Requirements

Migration is supported on Linux and Windows 64-bit x86 systems only. Before migrating, ensure that the following requirements are met:

- OWB 11.2.0.4 installed (plus Migration Patch applied. Note: Please contact Oracle Support to get the latest Migration Patch to be applied to your environment.)
- ODI 12.2.1
- OWB workspace exists
- ODI repositories exist (When migration mode is FAST_CHECK, this pre-condition is optional)
- `ODI_HOME` and `JAVA_HOME` environment variables set. The `ODI_HOME` variable should be set to the ODI installation directory, such as `/home/oracle/Middleware`. The `JAVA_HOME` variable should be set to the JDK installation directory, such as `/java/jdk<version>/`.

  Note: The JDK version should be 1.8 or higher.

  If you have the standalone version of OWB, there is no need to set the `JAVA_HOME` environment.
- Migration utility configuration file created

Also ensure that you have the following information:

- ODI master repository password (When migration mode is FAST_CHECK, this pre-condition is optional)
- ODI user password (When migration mode is FAST_CHECK, this pre-condition is optional)
- OWB workspace owner password
- Full path to the migration utility configuration file and the file name

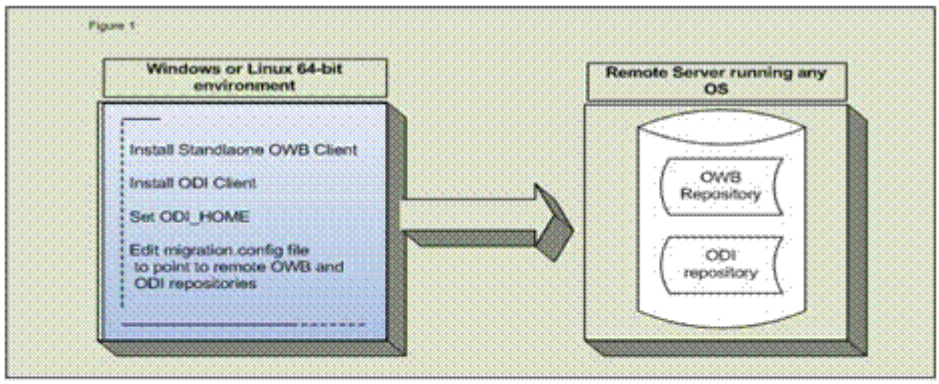
Note: Download the required patches from My Oracle Support (https://support.oracle.com). Apply the patches using the instructions in the patch readme files.

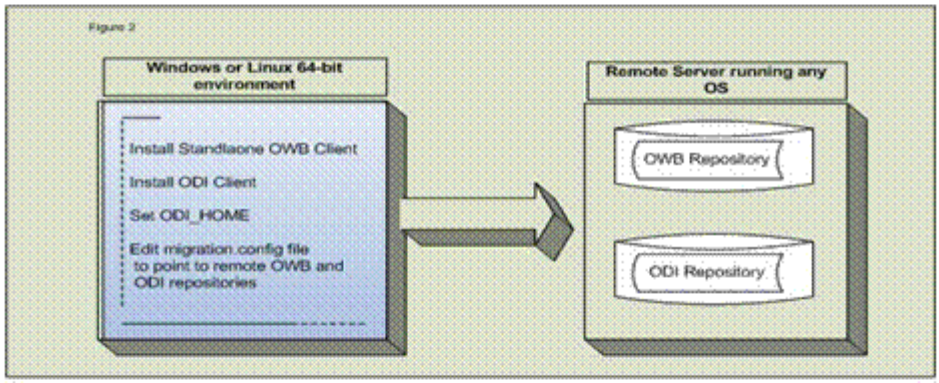
Migration Utility Run on a Non-64-bit Operating System

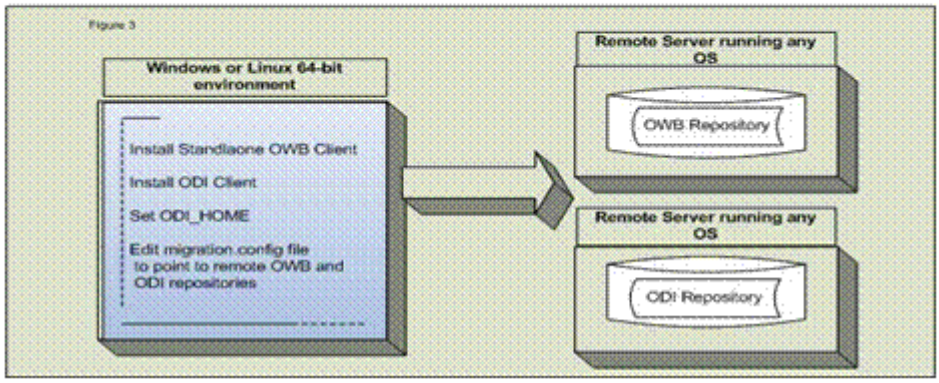
If your OWB repository resides on an environment other than Windows 64-bit or Linux 64-bit, you have to install both OWB and ODI clients on the same machine. Set the `ODI_HOME` environment variable to point to the ODI home. The `migration.config` file has to have all the parameters set correctly and pointing to the right location of the repositories.

Migration Utility Run for Remote Repositories

You can have three different scenarios when the repositories are remote as shown in the figures below:







In the 1st case the repositories can reside on the same server and the same database.

In the 2nd case the repositories can reside on same server but different databases.

In the 3rd case the repositories can reside on different servers.

Creating the Migration Utility Configuration File

Before migrating, you must first create the configuration file used to perform the migration. The configuration file is a text-based properties file that contains connection information and other details.

The following topics are addressed here:

- To Create the Migration Utility Configuration File

To Create the Migration Utility Configuration File

A template file is provided to make creation of the migration utility configuration file easier. Use this template as your starting point and edit the settings to fit your specific environment and needs.

The template file is named `migration.config` and is located in the *OWB_HOME*`/bin/admin` directory, where *OWB_HOME* is your OWB installation directory.

To create the migration utility configuration file:

1. Open the `migration.config` file in a text editor.
2. Edit the settings to fit your specific environment and needs. For more information about each parameter, see Configuration File Parameters.
3. Save the file. The file can be named whatever you like and saved to the location of your choice.

   Make note of the file name and its path, because you will need this information when you run the migration utility.

Configuration File Parameters

Table 2–1 lists the parameters in the migration utility configuration file.

Table 2–1 Migration Utility Configuration File Parameters

Parameter	Mandatory	Description
`ODI_MASTER_USER=<user_name>`	Yes	User name for the ODI master repository connection.
`ODI_MASTER_URL=<JDBC_URL>`	Yes	JDBC URL used to connect to the ODI master repository. This URL must be quoted if it contains one of the following characters: ■ semicolon (;) ■ backslash (\) ■ double quote (") ■ back quote (`) ■ dollar sign ($) ■ less than (<) ■ greater than (>) The default value is `jdbc:oracle:thin:@localhost:1521:mydb`
`ODI_MASTER_DRIVER=<JDBC_driver_name>`	Yes	JDBC driver used to connect to the ODI master repository. The default value is `oracle.jdbc.OracleDriver`
`ODI_USERNAME=<user_name>`	Yes	Supervisor user name for ODI. The default value is `SUPERVISOR`.
`ODI_WORK_REPOSITORY_NAME=<user_name>`	Yes	User name used to connect to the ODI work repository. The default value is `WORKREP1`.

Table 2–1 (Cont.) Migration Utility Configuration File Parameters

Parameter	Mandatory	Description
`OWB_WORKSPACE_OWNER=<workspace_owner>`	Yes	OWB workspace owner.
`OWB_URL=<URL>`	Yes	URL used to connect to the OWB workspace. The default value is `localhost:1521:mydb`
`OWB_WORKSPACE_NAME=<workspace_name>`	Yes	Name of the OWB workspace to connect to, specified in one of the following formats: ■ Workspace owner and workspace name, separated by a period. For example, `REP_1.WS1` or `rep_1.ws1`. ■ Workspace name only. For example, `WS1` or `ws1`. The migration utility can be used to migrate just one workspace at a time. Edit this parameter (and others as necessary) and run the migration utility for each workspace that you want to migrate. If the workspace owner owns just one workspace, you do not need to specify this parameter. If the workspace owner owns multiple workspaces and no value is specified for this parameter, an error is returned. If a workspace has the same name as the workspace owner, the workspace is migrated. If the specified workspace does not exist, the connection fails.
`MIGRATION_LOG_FILE=<path_to_log_file>`	No	Full path to the migration utility log file, which is generated when you run the migration utility. The migration utility exclusion report is also generated, and uses the same prefix as the log file, with a `.report` extension. This parameter is used to specify the name and location for both the log file and the report file. If no path is specified, the log and report files are generated in the same directory from which the migration utility was executed, for example, *OWB_HOME*`/owb/bin/unix`. By default, the file names are `migration.log` and `migration.report`. For more information about these files, see Reviewing Log and Report Files.
`MIGRATION_REPORT_INCLUDE=<PASSED|FAILED|ALL>`	No	Content to be included in the migration utility exclusion report. Options are: ■ `PASSED`: Include only objects that succeeded. ■ `FAILED`: Include only objects that failed. ■ `ALL`: Include all objects. The default value is `ALL`.

Table 2–1 (Cont.) Migration Utility Configuration File Parameters

Parameter	Mandatory	Description
MIGRATION_MODE=<FAST_CHECK|DRY_RUN|RUN>	No	Migration mode. Options are: ■ FAST_CHECK: The migration utility performs a quick check for selected objects and provides a report that lists objects that can and cannot be migrated to the target ODI repository. Use this mode to quickly determine which objects can and cannot be migrated. This mode can be used without installing and setting up the ODI environment. ■ DRY_RUN: The migration utility checks whether the specified objects can be created in the target ODI repository, and executes the migration without committing the objects to the repository. This mode can be used without installing and setting up the ODI environment. Meanwhile, when ODI related parameters invoke migration.sh/migration.bat(ODI_MASTER_USER,ODI_USERNAME,ODI_MASTER_PASSWORD,ODI_USER_PASSWORD), it might not be correct. ■ RUN: The migration utility executes the migration and commits migrated objects to the target ODI repository. Use this mode to perform the migration from OWB to ODI. The default value is RUN. For more information about using the FAST_CHECK and DRY_RUN modes to perform a test migration, see Performing a Test Migration.
MIGRATION_STRATEGY	No	Indicate whether migrating the object or not when there is an object with the same name already existed in ODI repository. This parameter has two options, CREATE and NODUP. The default value is CREATE. ■ CREATE always creates a new object in ODI. If there is an existing object with the same name in the repository, the new object is created with a name suffixed with _# where "#" is a number. ■ NODUP matches objects existing in ODI repository with the name, if exists, the object is not migrated and the existing one in ODI repository is used.

Table 2–1 (Cont.) Migration Utility Configuration File Parameters

Parameter	Mandatory	Description
MIGRATE_DEPENDENCIES=<TRUE|FALSE>	No	Controls whether dependent objects are migrated with the objects selected for migration. The default value is FALSE (dependent objects are not migrated). Recursive dependency is supported when MIGRATE_DEPENDENCIES is set to TRUE. For example: Mapping MAP_1 has a map operator bound to table T_1, and table T_1 has an FK (foreign key) relationship with table T_2. Both T_1 and T_2 are considered as dependencies and are migrated along with mapping MAP_1.
STOP_ON_ERROR=<TRUE|FALSE>	No	Indicates whether to continue the migration process or stop when an error occurs. When set to TRUE, the migration process stops and no objects are migrated. When set to FALSE, the migration process continues even if an error occurs, and successful objects are migrated. The default value is FALSE.
SPLIT_JOIN_FOR_ANSI_SYNTAX=<TRUE|FALSE>	No	Indicates whether to split the join operator to binary join when the property Use ANSI Syntax of the OWB mapping is set to TRUE. The default value is TRUE (join operator is split).
MIGRATE_UNBOUND_OPERATOR=<TRUE|FALSE>	No	Determines whether mappings that contain unbound operators (excluding Code Template mappings) are migrated. Unbound operators include external table, table, view, materialized view, lookup, and pluggable mapping. When set to TRUE, mappings that contain unbound operators are migrated. For unbound entity operators (external table, table, view, materialized view, and lookup), an ODI datastore corresponding to the unbound operator is created in the ODI model that is migrated from the OWB module where the OWB mapping exists. The unbound operator is migrated to an ODI mapping component bound to the newly created ODI datastore. For an unbound pluggable mapping operator, an ODI reusable mapping is created in an ODI folder named STAND_ALONE. The unbound pluggable mapping operator is migrated to the ODI reusable mapping component bound to the newly created reusable mapping. The default value is FALSE, which means any mappings that contain unbound operators are not migrated.

Table 2–1 (Cont.) Migration Utility Configuration File Parameters

Parameter	Mandatory	Description
MIGRATION_OBJECTS=<objects>	No	Specifies the OWB objects to be migrated. The default value is the wild card asterisk (*), which means that all projects in the designated OWB workspace are migrated. For more information about migrating specific objects, see Migrating Specific Objects in an OWB Workspace.
FLUSH_BATCH_SIZE=<number_of_mappings>	No	Indicates the number of mappings to be processed or migrated at a time. Use this parameter to avoid out of memory issues if the OWB workspace has a very large number of mappings. The default value is 50. Reduce this value if out of memory issues occur.

Configuration File Example

Example 2–1 shows the values for a sample migration utility configuration file.

Example 2–1 Sample Migration Utility Configuration File

```
ODI_MASTER_USER=ODIREP
ODI_MASTER_URL=jdbc:oracle:thin:@localhost:1521:machine
ODI_MASTER_DRIVER=oracle.jdbc.OracleDriver
ODI_USERNAME=SUPERVISOR
ODI_WORK_REPOSITORY_NAME=WORK0
OWB_WORKSPACE_OWNER=rep_0
OWB_URL=localhost:1521:machine.oracle.com
OWB_WORKSPACE_NAME=REP_0_WS_0
MIGRATION_LOG_FILE=/tmp/migration.log
MIGRATION_REPORT_INCLUDE=ALL
MIGRATION_MODE=RUN
MIGRATION_STRATEGY=CREATE
MIGRATE_DEPENDENCIES=TRUE
STOP_ON_ERROR=FALSE
SPLIT_JOIN_FOR_ANSI_SYNTAX=TRUE
MIGRATE_UNBOUND_OPERATOR=TRUE
MIGRATION_OBJECTS=PROJECT.MY_PROJECT
FLUSH_BATCH_SIZE=50
```

3

Using the Migration Utility to Migrate

This chapter describes how to use the migration utility to migrate objects from OWB to ODI.

Migration is performed using the settings specified in the migration utility configuration file. For more information about this file, see Creating the Migration Utility Configuration File.

> **Note:** The OWB workspace and the ODI repository should not be in use when you run the migration utility and perform the migration.

The following topics are addressed here:

- Migration Utility Syntax and Parameters
- Migrating an OWB Workspace
- Migrating Specific Objects in an OWB Workspace
- Performing a Test Migration

Migration Utility Syntax and Parameters

The migration utility is started from the command line and takes several parameters as inputs to perform the migration.

On Linux, the migration utility file is named `migration.sh` and is executed from the *OWB_HOME*`/owb/bin/unix` directory, where *OWB_HOME* is your OWB installation directory.

On Windows, the migration utility file is named `migration.bat` and is executed from the *OWB_HOME*`/owb/bin/win` directory, where *OWB_HOME* is your OWB installation directory.

The syntax to run the migration utility and perform the migration is as follows:

```
./migration.sh <odi_master_password> <odi_user_password> <owb_workspace_owner_
password> <configuration_file>
```

For example:

```
./migration.sh odi_master supervisor migration /scratch/jsmith/Migration/owb_
migration.properties
```

The command parameters are as follows:

- `odi_master_password`: ODI master repository password (When migration mode is FAST_CHECK, this password might not be the real one)

- odi_user_password: ODI user password (When migration mode is FAST_CHECK, this password might not be the real one)
- owb_workspace_owner_password: OWB workspace owner password
- configuration_file: Full path to the migration utility configuration file and the file name

If you want migration.sh to refer to the ODI libraries you specified, you need to list all the necessary ODI public SDK jars in a file and use the following syntax to start the migration utility.

```
 ./migration.sh -Dodi.classpath= <odi_classpath_file> <odi_master_
password> <odi_user_password> <owb_workspace_owner_password>
<configuration_file>

odi_classpath_file: Full path to the odi classpath file and the file name.
```

In the odi classpath file, you need to list the full path of the ODI jars line by line.

Note that -Dodi.classpath should be placed just after the migration.sh and before the other parameters.

> **See Also:** For more information about the migration modes used to test and perform the migration, see MIGRATION_MODE in Configuration File Parameters.

After migration is complete, you are returned to the command prompt. You can then review the migration utility log file and exclusion report for details about the migration. For more information about these files, see Reviewing Log and Report Files.

Migrating an OWB Workspace

To migrate an entire OWB workspace, use the wild card asterisk (*) as the value for the MIGRATION_OBJECTS parameter in the migration utility configuration file. For example:

```
MIGRATION_OBJECTS=*
```

All projects and supported objects in the OWB workspace specified by the OWB_WORKSPACE_NAME parameter in the configuration file will be migrated.

> **Note:** You can migrate just one workspace at a time. Edit the configuration file and run the migration utility for each workspace that you want to migrate.

Migrating Specific Objects in an OWB Workspace

To migrate specific objects in an OWB workspace, configure the MIGRATION_OBJECTS parameter in the migration utility configuration file to migrate just those objects. You can specify a project, folder, or single non-folder object, or a set of objects that share the same type and the same folder.

Use a string that concatenates the qualifying names of all objects included in the object's path, from the top-level object to the leaf object. Qualifying names are specified as *OBJECT_TYPE*.*OBJECT_PHYSICAL_NAME*, with a period (.) separating the object type from its physical name. For example, to select table T_1 in Oracle module MOD_1 in project PRO_1, set the value of the MIGRATION_OBJECTS parameter to PROJECT.PRO_1.MODULE.MOD_1.TABLE.T_1.

The following values can be used for *OBJECT_TYPE*:

- EXTERNAL_TABLE
- FLAT_FILE_MODULE
- FLAT_FILE
- GENERIC_FOLDER
- GENERIC_MODULE
- LOCATION
- MODULE
- MAPPING
- MATERIALIZED_VIEW
- PLUGGABLE_MAPPING
- PLUGGABLE_MAPPING_FOLDER
- PROJECT
- SAP_MODULE
- SEQUENCE
- TABLE
- VIEW

Use a semicolon (;) to separate multiple items, for example:

```
MIGRATION_OBJECTS=PROJECT.PRO_1.MODULE.MOD_1.TABLE.T_1;PROJECT.PRO_2.MODULE.MOD_2;
```

Use a backslash (\) at the end of a line to improve readability of the configuration file if there are multiple items, for example:

```
MIGRATION _OBJECTS=
PROJECT.OWB_MIGRATION.SAP_MODULE.MY_SAP_MOD;\
PROJECT.MY_PROJECT.MODULE.ORA_MOD;\
PROJECT.MY_PROJECT.MODULE.DB2_MOD
```

Use the wild card asterisk (*) at the end of a string instead of an object name to select all objects of a specific type in a folder. The following example selects all tables in module MOD_1:

```
MIGRATION_OBJECTS=PROJECT.PRO_1.MODULE.MOD_1.TABLE.*
```

Use the wild card asterisk (*) at the end of an object name to select all objects with that name. The following example selects all tables in module MOD_1 with a name that starts with MYTEST:

```
MIGRATION_OBJECTS=PROJECT.PRO_1.MODULE.MOD_1.TABLE.MYTEST*
```

The following uses are not supported:

```
MIGRATION_OBJECTS=PROJECT.*.MODULE.MOD_1.TABLE.T_1;
MIGRATION_OBJECTS=PROJECT.PRO_1.MODULE.*.TABLE.T_1;
MIGRATION_OBJECTS=PROJECT.*.MODULE.*.TABLE.*;
MIGRATION_OBJECTS=PROJECT.PRO_1.MODULE.*.TABLE.MYT*;
MIGRATION_OBJECTS=PROJECT.PRO_1.MODULE.MYMOD*.TABLE.MYT_1;
```

When an invalid object is specified, an error is returned:

```
[ERROR][Migration][MU-1005] The selected object {0} does not exist or the
selection is invalid {invalid object name}. It will be skipped.
```

For more information about error messages that you might encounter when you run the migration utility, see Message Reference.

Example 3–1 Migrating Specific Objects

This section provides additional examples of migrating specific objects.

The following example migrates Oracle Database module `ORACLE_EBS_D` in project `SSAD`:

```
MIGRATION_OBJECTS=PROJECT.SSAD.MODULE.ORACLE_EBS_D;
```

The following example migrates pluggable mapping `DEBS_EDW_MAP1` in pluggable mapping folder `DWPR_SUB`:

```
MIGRATION_OBJECTS=PROJECT.PROJ_DW.PLUGGABLE_MAPPING_FOLDER.DWPR_SUB.PLUGGABLE_
MAPPING.DEBS_EDW_MAP1;
```

The following example migrates standalone pluggable mapping `PLUGGABLE_MAPPING_1` in project `SSA`:

```
MIGRATION_OBJECTS=PROJECT.SSA.PLUGGABLE_MAPPING.PLUGGABLE_MAPPING_1
```

Performing a Test Migration

To test your migration before executing it, set the `MIGRATION_MODE` parameter in the migration utility configuration file to `FAST_CHECK` or `DRY_RUN`.

The `FAST_CHECK` option checks which objects can and cannot be migrated. The `DRY_RUN` option checks whether the specified objects can be created in the target ODI repository, and executes the migration without committing the objects to the repository. For more information about these options, see `MIGRATION_MODE` in Configuration File Parameters.

After performing a test migration, review the migration utility log file and exclusion report for details. You can use these files to identify objects that can and cannot be migrated and to address any issues before performing the actual migration. For more information about these files, see Reviewing Log and Report Files.

4

Reviewing Your Migration

This chapter describes how to review and verify your migration.

The following topics are addressed here:

- Reviewing Log and Report Files
- Verifying Your Migration

Reviewing Log and Report Files

Two files are created after migration is complete or after you perform a test migration. By default, the files are named `migration.log` and `migration.report`. Use these files to review, refine, and troubleshoot your migration.

By default, the files are saved in the same location as the migration utility configuration file. You can specify a different file name and location using the `MIGRATION_LOG_FILE` parameter in the configuration file. For more information about this parameter, see `MIGRATION_LOG_FILE` in Configuration File Parameters.

The following topics are addressed here:

- Reviewing the Migration Utility Log File
- Reviewing the Migration Utility Exclusion Report

Reviewing the Migration Utility Log File

The migration utility log file contains details about objects that were migrated, rejected, or skipped, and error messages if any errors occurred.

The log file is organized in the following sections:

- Log file header with migration mode, log file creation time, OWB and ODI details, full path to the log file, and configuration options.
- Migration start time.
- Detailed information about the migration status of each object (whether migration succeeded, was rejected, or skipped) and error messages if errors occurred. For more information about informational, warning, and error messages that you might encounter when you run the migration utility, see Message Reference.
- Summary information organized by object type, including the path to each object.
- Log file footer with total execution time and migration end time.

Example 4–1 Sample Migration Utility Log File

This example shows a sample migration utility log file, with `MIGRATION_MODE` set to `RUN`.

```
*******************************************************************************
* Oracle Warehouse Builder - Migration Utility - Log
* Created: 4/3/14 11:26 PM
* Migration Report Style - RUN
*
* OWB Release:11.2.0.4.0 - OWB Repository:
OWB_REPO_MIG/machine.oracle.com:1521:orcl11204 - OWB Workspace:
OWB_REPO_MIG.OWB_REPO_WKSP1
*
* ODI Release:12.1.3.0.0 - ODI Master
mig12c/jdbc:oracle:thin:@machine:1521:orcl11203 - ODI User/Work Repository:
SUPERVISOR/WORKREP1
*
* Log File: /tmp/migration.log
*
* Configuration Options
*
 ----------------------
* MIGRATION_REPORT_INCLUDE=ALL
* MIGRATION_MODE=RUN
* MIGRATE_DEPENDENCIES=true
* STOP_ON_ERROR=false
* SPLIT_JOIN_FOR_ANSI_SYNTAX=true
* MIGRATE_UNBOUND_OPERATOR=false
* FLUSH_BATCH_SIZE=50
* MIGRATION_STRATEGY=NODUP
* MIGRATION_OBJECTS=PROJECT.PRO_MIGRATION
*******************************************************************************

Migration started at 4/3/14 11:26 PM Pacific Standard Time

*******************************************************************************
----START MIGRATE LOCATION MIGRATION_SRC_MOD_LOC.
----SUCCESSFULLY MIGRATED MIGRATION_SRC_MOD_LOC.
----START MIGRATE LOCATION OPERATOR_MIGRATION_LOC.
--------[INFO][Migration][MU-1010]OPERATOR_MIGRATION_LOC is skipped because it
already exists.
----END MIGRATE OPERATOR_MIGRATION_LOC. START MIGRATE PROJECT PRO_MIGRATION. FLUSH
OdiDataServer[1]  COST(MS):47
----START MIGRATE MODULE_FOR_LOGICALSCHEMA TEST_MOD.
----SUCCESSFULLY MIGRATED TEST_MOD.
----START MIGRATE MODULE TEST_MOD. FLUSH OdiLogicalSchema[1]  COST(MS):27
--------START MIGRATE TABLE DEPT.
--------SUCCESSFULLY MIGRATED DEPT.
--------START MIGRATE TABLE EMP.
--------SUCCESSFULLY MIGRATED EMP.
--------START MIGRATE TABLE TGT_EMP.
--------SUCCESSFULLY MIGRATED TGT_EMP.
--------START MIGRATE TABLE TGT_EMP_CONSTRAINTS.
--------SUCCESSFULLY MIGRATED TGT_EMP_CONSTRAINTS.
--------START MIGRATE VIEW TGT_V_EMP.
--------SUCCESSFULLY MIGRATED TGT_V_EMP.
----SUCCESSFULLY MIGRATED TEST_MOD.
----START MIGRATE MODULE_FOR_LOGICALSCHEMA TEST_SRC_MOD. FLUSH OdiDataStore[5]
COST(MS):373
----SUCCESSFULLY MIGRATED TEST_SRC_MOD.
```

```
----START MIGRATE MODULE TEST_SRC_MOD.
--------START MIGRATE TABLE EMPLOYEES. FLUSH OdiModel[1]  COST(MS):78
--------SUCCESSFULLY MIGRATED EMPLOYEES.
----SUCCESSFULLY MIGRATED TEST_SRC_MOD.
----START SECOND PASS FOR TABLE.
--------FOREIGN KEY CREATED: TGT_EMP_CONSTRAINTS.DEPTNO --> DEPT.DEPTNO
--------FOREIGN KEY CREATED: TGT_V_EMP.DEPTNO --> DEPT.DEPTNO
----END SECOND PASS.
----START MIGRATE MAPPING_MODULE TEST_MOD.
--------START MIGRATE MAPPING BASIC_JOIN. FLUSH MAPPING, MIGRATED 0 COST(MS):57
--------SUCCESSFULLY MIGRATED BASIC_JOIN.
--------START MIGRATE MAPPING TLO_MAP.
--------SUCCESSFULLY MIGRATED TLO_MAP.
--------START MIGRATE MAPPING UNSUPPORT_MAP.
------------[INFO][Migration][MU-5001]Migration of mapping with mapping operator
MATCHMERGE:MATCHMERGE is not supported.
--------FAILED MIGRATE UNSUPPORT_MAP.
----SUCCESSFULLY MIGRATED TEST_MOD.
----START MIGRATE MAPPING_MODULE TEST_SRC_MOD. FLUSH Mapping[2]  COST(MS):880
----SUCCESSFULLY MIGRATED TEST_SRC_MOD. SUCCESSFULLY MIGRATED PRO_MIGRATION.
********************************************************************************
TABLE[TOTAL:5 MIGRATED:5 REJECTED:0 SKIPPED:0].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_MOD].TABLE[DEPT].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_MOD].TABLE[EMP].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_MOD].TABLE[TGT_EMP].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_MOD].TABLE[TGT_EMP_CONSTRAINTS].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_SRC_MOD].TABLE[EMPLOYEES].
LOCATION[TOTAL:2 MIGRATED:1 REJECTED:0 SKIPPED:1].
----PASSED: PROJECT[PUBLIC_PROJECT].LOCATION[MIGRATION_SRC_MOD_LOC].
----SKIPPED: PROJECT[PUBLIC_PROJECT].LOCATION[OPERATOR_MIGRATION_LOC]. MAPPING_
MODULE[TOTAL:2 MIGRATED:2 REJECTED:0 SKIPPED:0].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_MOD].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_SRC_MOD].
VIEW[TOTAL:1 MIGRATED:1 REJECTED:0 SKIPPED:0].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_MOD].VIEW[TGT_V_EMP].
MODULE[TOTAL:2 MIGRATED:2 REJECTED:0 SKIPPED:0].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_MOD].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_SRC_MOD].
PROJECT[TOTAL:1 MIGRATED:1 REJECTED:0 SKIPPED:0].
----PASSED: PROJECT[PRO_MIGRATION].
MAPPING[TOTAL:3 MIGRATED:2 REJECTED:1 SKIPPED:0].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_MOD].MAPPING[BASIC_JOIN].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_MOD].MAPPING[TLO_MAP].
----FAILED: PROJECT[PRO_MIGRATION].MODULE[TEST_MOD].MAPPING[UNSUPPORT_MAP].
MODULE_FOR_LOGICALSCHEMA[TOTAL:2 MIGRATED:2 REJECTED:0 SKIPPED:0].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_MOD].
----PASSED: PROJECT[PRO_MIGRATION].MODULE[TEST_SRC_MOD].
********************************************************************************

Migration ended at 4/3/14 11:27 PM Pacific Standard Time

Total migration time (hh:mm:ss): 00:00:56
```

Reviewing the Migration Utility Exclusion Report

The migration utility exclusion report contains a summary of the objects migrated, and lists whether migration succeeded, was rejected, or skipped for each object.

The exclusion report is organized in the following sections:

- Exclusion report header with migration mode, report creation time, OWB and ODI details, full path to the report file, and configuration options.
- Migration start time.
- Migration statistics including how many projects were migrated, and total number of objects migrated for each project.
- Detailed migration status for each selected object (whether migration succeeded, was rejected, or skipped).
- Exclusion report footer with total execution time and migration end time.

Example 4–2 Sample Migration Utility Exclusion Report

This example shows a sample migration utility exclusion report, with `MIGRATION_MODE` set to `RUN`.

```
********************************************************************************
*Oracle Warehouse Builder - Migration Utility - Summary Report
*Created: 4/3/14 11:26 PM
*Migration Report Style - RUN
*
*OWB Release:11.2.0.4.0 - OWB Repository:
OWB_REPO_MIG/machine.oracle.com:1521:orcl11204 - OWB Workspace:
OWB_REPO_MIG.OWB_REPO_WKSP1
*
*ODI Release:12.1.3.0.0 - ODI Master
mig12c/jdbc:oracle:thin:@machine:1521:orcl11203 - ODI User/Work Repository:
SUPERVISOR/WORKREP1
*
*Report File: /tmp/migration.report
*
Configuration Options
---------------------

*MIGRATION_REPORT_INCLUDE=ALL
*MIGRATION_MODE=RUN
*MIGRATE_DEPENDENCIES=true
*STOP_ON_ERROR=false
*SPLIT_JOIN_FOR_ANSI_SYNTAX=true
*MIGRATE_UNBOUND_OPERATOR=false
*FLUSH_BATCH_SIZE=50
*MIGRATION_STRATEGY=NODUP
*MIGRATION_OBJECTS=PROJECT.PRO_MIGRATION
********************************************************************************

Migration started at 4/3/14 11:26 PM Pacific Standard Time

Statistics
------------

Total Projects Migrated: 2

********************************************************************************
PROJECT: PUBLIC_PROJECT
Object Types                  Migrated     Rejected     Skipped
---------------------------   ---------    ---------    --------
LOCATION:                            1            0           1

********************************************************************************
PROJECT: PRO_MIGRATION
```

```
Object Types                  Migrated     Rejected     Skipped
--------------------------    ---------    ---------    --------
                    TABLE:            5            0           0
          MAPPING_MODULE:             2            0           0
                     VIEW:            1            0           0
                   MODULE:            2            0           0
                  MAPPING:            2            1           0
MODULE_FOR_LOGICALSCHEMA:             2            0           0

Details
--------

********************************************************************************
PROJECT: PUBLIC_PROJECT
Object Types                 Status
--------------------------   -------
LOCATION
  MIGRATION_SRC_MOD_LOC      SUCCESS
  OPERATOR_MIGRATION_LOC     [INFO][Migration][MU-1010]OPERATOR_MIGRATION_LOC is
skipped because it already exists.

********************************************************************************
PROJECT: PRO_MIGRATION
Object Types                 Status
--------------------------   -------
  TABLE DEPT                 SUCCESS
  EMP                        SUCCESS
  TGT_EMP                    SUCCESS
  TGT_EMP_CONSTRAINTS        SUCCESS
  EMPLOYEES                  SUCCESS
MAPPING_MODULE
  TEST_MOD                   SUCCESS
  TEST_SRC_MOD               SUCCESS
VIEW
  TGT_V_EMP                  SUCCESS
MODULE
  TEST_MOD                   SUCCESS
  TEST_SRC_MOD               SUCCESS
MAPPING
  BASIC_JOIN                 SUCCESS
  TLO_MAP                    SUCCESS
  UNSUPPORT_MAP              [INFO][Migration][MU-5001]Migration of mapping with
mapping operator MATCHMERGE:MATCHMERGE is not supported.
MODULE_FOR_LOGICALSCHEMA
  TEST_MOD                   SUCCESS
  TEST_SRC_MOD               SUCCESS

Migration ended at 4/3/14 11:27 PM Pacific Standard Time

Total migration time (hh:mm:ss): 00:00:56
```

Verifying Your Migration

When migration is complete, perform the following steps in ODI to verify the mappings that were migrated from OWB:

- Use ODI Studio to connect to the ODI environment. See "Connecting to a Work Repository" in the *Developer's Guide for Oracle Data Integrator*.

- Navigate to Topology Navigator and review the data server settings. You may need to edit some of the information such as user names, passwords, or JDBC URLs depending on your environment. Test each connection to make sure that each migrated data server is correctly configured. See "Setting Up the Topology" in the *Developer's Guide for Oracle Data Integrator*.
- Navigate to Designer Navigator and review the migrated models and datastores in the **Models** panel. See "Creating and Using Data Models and Datastores" in the *Developer's Guide for Oracle Data Integrator*.
- Navigate to Designer Navigator and verify the migrated mappings in the **Projects** panel by running the mappings. See "Creating and Using Mappings" in the *Developer's Guide for Oracle Data Integrator*.

A

Message Reference

This appendix describes messages you might encounter when you run the migration utility.

If objects cannot be migrated, informational messages appear.

If objects are migrated with warnings, warning messages appear.

If the objects cannot be migrated due unexpected errors, error messages appear.

The informational, warning, and error messages are written to the migration utility log in the following formats:

- `[ERROR|WARN|INFO][Migration][MU-XXXX]`: Indicates the message is coming from the migration utility (XXXX is the message ID).
- `[ERROR|WARN][Migration][ODI]`: Indicates the message is coming from ODI.
- `[ERROR|WARN][Migration][OWB]`: Indicates the message is coming from OWB.

For more information about the migration utility log file, see Reviewing Log and Report Files.

Table A–1 provides example OWB and ODI error and warning messages. The message text is as it appears in the message.

Table A–1 Example OWB and ODI Error and Warning Messages

Message	Cause	Action
[ERROR][Migration][OWB] Unable to connect to OWB workspace! Details: {0}	The connection to the OWB workspace cannot be established. The credential information used to connect to the OWB workspace may be invalid.	Verify the following parameters in the migration utility configuration file when running the migration utility (`migration.sh`): ■ `OWB_WORKSPACE_OWNER` ■ `OWB_URL` ■ `OWB_WORKSPACE_NAME` For more information about these parameters, see Configuration File Parameters. Also verify the password for the OWB workspace owner.
[ERROR][Migration][ODI] Unable to connect to ODI repository! Details: {0}	The connection to the ODI repository cannot be established. The credential information used to connect to the ODI repository may be invalid.	Verify the following parameters in the migration utility configuration file when running the migration utility (`migration.sh`): ■ `ODI_MASTER_USER` ■ `ODI_MASTER_URL` ■ `ODI_MASTER_DRIVER` ■ `ODI_USERNAME` ■ `ODI_WORK_REPOSITORY_NAME` For more information about these parameters, see Configuration File Parameters. Also verify the passwords for the ODI master repository and the ODI user.

Table A–2 lists migration utility error and warning messages. Messages are listed in numeric order by message ID. The message text is as it appears in the message.

Table A–2 Migration Utility Informational, Warning, and Error Messages

Message	Cause	Action
[MU-1001] Invalid number of parameters. You have to provide 4 parameters: password for ODI master repository, password for ODI user, password for OWB, full path for settings file.	Required parameters were not supplied when running the migration utility (`migration.sh`).	Provide the required parameters when running the migration utility (`migration.sh`). For more information about the correct syntax, see Migration Utility Syntax and Parameters.
[ERROR] [Migration] [OWB] Unable to connect to OWB workspace! Details: {0}	The connection to OWB workspace cannot be established. The credential information used to connect to OWB workspace may be invalid.	Verify the following parameters in the migration utility configuration file when running the migration utility (`migration.sh`): ■ `OWB_WORKSPACE_OWNER` ■ `OWB_URL` ■ `OWB_WORKSPACE_NAME`
[ERROR] [Migration] [ODI] Unable to connect to ODI repository! Details: {0}	The connection to ODI repository cannot be established. The credential information used to connect to ODI repository may be invalid.	Verify the following parameters in the migration utility configuration file when running the migration utility (`migration.sh`): ■ `ODI_MASTER_USER` ■ `ODI_MASTER_URL` ■ `ODI_MASTER_DRIVER` ■ `ODI_USERNAME` ■ `ODI_WORK_REPOSITORY_NAME`
[MU-1004] Unable to load configuration file {0}. Details:{1}	The migration utility configuration file does not exist or is not readable or accessible.	Make sure the migration utility configuration file exists and is readable and accessible. Specify the full path to the configuration file and the file name. For more information about the configuration file, see Creating the Migration Utility Configuration File.
[MU-1005] The selected object {0} does not exist or the selection is invalid. It will be skipped.	An invalid or nonexistent object is specified for the `MIGRATION_OBJECTS` parameter in the migration utility configuration file.	Verify the value specified for the `MIGRATION_OBJECTS` parameter in the migration utility configuration file. For more information about this parameter, see `MIGRATION_OBJECTS` in Configuration File Parameters. Also see Migrating Specific Objects in an OWB Workspace.
[MU-1006] Invalid object name {0} in selection {1}, the selection will be skipped.	An invalid object name is specified for the `MIGRATION_OBJECTS` parameter in the migration utility configuration file.	Verify the value specified for the `MIGRATION_OBJECTS` parameter in the migration utility configuration file. For more information about this parameter, see `MIGRATION_OBJECTS` in Configuration File Parameters. Also see Migrating Specific Objects in an OWB Workspace.
[MU-1007] Migration failed. Details: {0}	As described in the message.	Review the message to determine the cause of the problem and take appropriate action.

Table A–2 (Cont.) Migration Utility Informational, Warning, and Error Messages

Message	Cause	Action
[MU-1008] Unable to write to log or report file {0}. Details:{1}	The log or report file is not accessible to the migration utility.	Verify the path specified for the `MIGRATION_LOG_FILE` parameter in the migration utility configuration file. Make sure the specified location permits new files to be created and that enough disk space exists to write the files. For more information about this parameter, see `MIGRATION_LOG_FILE` in Configuration File Parameters.
[MU-1009] Invalid configuration option {0}. It will be ignored.	An invalid parameter is specified in the migration utility configuration file.	Verify the parameters in the migration utility configuration file, make sure they are correct.
[MU-1010] {0} is skipped because it already exists.	The parameter `MIGRATION_STRATEGY` in the migration utility configuration file is specifies to `NODUP`. When `MIGRATION_STRATEGY` is set to `NODUP`, migration utility will match with objects existing in ODI repository with the name, if exists, the object will not be migrated and the existing one in ODI repository is used.	No action.
[MU-2001] Migration of location {0} in platform {1} is not supported.	The location for this platform is not supported for migration.	No action.
[MU-2002] Migration of location {0} with no associated platform is not supported.	The location is not associated with a platform.	No action.
[MU-3001] Unable to load file {0}. Details: {1}.	The file `PlatformMappingsForMigration.xml` does not exist in the *OWB_HOME*`/owb/bin/admin` directory or the directory is not accessible to the migration utility.	Verify that the file `PlatformMappingsForMigration.xml` exists in the *OWB_HOME*`/owb/bin/admin` directory and that the directory is accessible to the migration utility (*OWB_HOME* is your OWB installation directory). This file contains the mappings between OWB platforms and ODI technologies. For more information about this file, see OWB Platform to ODI Technology.
[MU-3002] Unable to find ODI technology corresponding to the OWB platform: {0}.	The mapping of the specified OWB platform to any ODI technology is missing in the file `PlatformMappingsForMigration.xml`.	Add the mapping of the specified OWB platform to one ODI technology in the file `PlatformMappingsForMigration.xml`. This file contains the mappings between OWB platforms and ODI technologies. For more information about this file, see OWB Platform to ODI Technology.

Table A–2 (Cont.) Migration Utility Informational, Warning, and Error Messages

Message	Cause	Action
[MU-3003] Unable to find technology: {0} in ODI.	The specified technology is not defined in the ODI repository.	Define the specified technology in ODI, or modify the file `PlatformMappingsForMigration.xml` to refer to a correct ODI technology. This file contains the mappings between OWB platforms and ODI technologies. For more information about this file and these mappings, see OWB Platform to ODI Technology.
[MU-4001] Migration of {0}:{1} is not supported because unsupported data type {3} is used in column {2}.	The data type used by the specified column is not supported for migration.	Change the data type in OWB if possible. For more information about data types supported for migration, see Reference to Migration Details.
[MU-4002] {0}:{1} has multiple primary keys. Only one primary key is allowed in ODI, the redundant primary keys will be migrated as alternate keys.	An OWB table can be defined with several primary keys, but an ODI data store can have just one primary key. Only one of the primary keys in OWB will be migrated as the primary key in ODI. The rest will be migrated as alternate keys.	No action.
[MU-4003] {0}:{1} is not migrated because it has multiple columns with the same name {2}.	An OWB table may have duplicate columns due to previous OWB issues.	Check the OWB table, and rename the columns. Make sure the name of the column is unique in the table.
[MU-5001] Migration of mapping with mapping operator {0}:{1} is not supported.	The specified mapping operator is not supported for migration.	No action.
[MU-5002] Migration of mapping with mapping operator {0}:{1} which contains multiple return attributes is not supported.	Function operators with multiple return attributes are not migrated.	No action.
[MU-5003] Migration of mapping with mapping operator {0}:{1} which contains OUT parameter {2} is not supported.	Function operators with `OUT` parameters are not migrated.	No action.
[MU-5004]Migration of mapping with mapping operator {0}:{1} which contains INOUT parameter {2} is not supported.	Function operators with `INOUT` parameters are not migrated.	No action.
[MU-5005] Migration of mapping with complex data type {2} used in attribute {3} in mapping operator {0}:{1} is not supported}.	Mapping operators with complex data types used in mapping attributes are not migrated.	No action.
[MU-5006] Migration of mapping with mapping operator {0}:{1} that does not define return attribute is not supported.	Function operators with no return attribute are not migrated.	No action.
[MU-5007] Mapping is not migrated because the function name of the mapping operator {0}:{1} cannot be determined.	The property `FUNCTION_NAME` on the function operator is not defined.	Set the value for the property `FUNCTION_NAME` on the function operator.

Table A–2 (Cont.) Migration Utility Informational, Warning, and Error Messages

Message	Cause	Action
[MU-5008] Unable to set Extract Knowledge Module on physical node {0} in ODI. Details: {1}	As described in the message.	Review the message to determine the cause of the problem and take appropriate action.
[MU-5009] Mapping is not migrated because the bound object of the mapping operator {0}:{1} is not being migrated.	The bound object of a mapping operator is not migrated.	Check the migration utility log to determine why the bound object was not migrated.
[MU-5010]Mapping is not migrated because the mapping operator{0}:{1} has no output attribute group.	The Lookup operator has no output attribute group.	Modify the Lookup operator in OWB, and add the output attribute group for it.
[MU-5011] mapping is not migrated because the output attribute group {1} in Lookup {0} is unbound. Use the configuration option of migration utility "MIGRATE_UNBOUND_OPERATOR" or fix the mapping with unbound output attribute groups.	The output attribute group of the Lookup operator is unbound.	Bind the output attribute group of the Lookup operator or set the `MIGRATE_UNBOUND_OPERATOR` parameter in the migration utility configuration file to `TRUE`. For more information about this parameter, see `MIGRATE_UNBOUND_OPERATOR` in Configuration File Parameters.
[MU-5012] Mapping is not migrated because the bound object of the mapping operator {0}:{1} for output attribute group {2} is not being migrated.	The bound object of the output attribute group of the Lookup operator is not migrated.	Check the migration utility log to determine why the bound object was not migrated.
[MU-5013] Mapping is not migrated because the input attribute group is not defined for output attribute group {1} in Lookup {0}.	The output attribute group of the Lookup operator has no corresponding input attribute group.	Modify the Lookup operator, and add the input attribute group for each output attribute group.
[MU-5018] Mapping is not migrated because unsupported data type {3} is used in attribute {2} in mapping operator {0}:{1}.	Data type {3} set on the mapping attribute is not supported for migration.	Change the data type of the mapping attribute to a supported data type if possible. For more information about data types supported for migration, see Reference to Migration Details.
[MU-5019] Unable to set expression [{1}] on attribute {0}. Details: {2}.	As described in the message.	Review the message to determine the cause of the problem and take appropriate action.
[MU-5020] Unable to split mapping joiner operator {0} into binary joins due to {1}.	The join condition of the join operator cannot be parsed successfully.	Check the join condition and modify it if possible.
[MU-5021] The mapping joiner operator {0} will be split into binary joins after migration because some input group(s) have role set to "Outer", even though the mapping property "ANSI SQL Syntax" is set to false or the configuration option for migration utility "SPLIT_JOIN_FOR_ANSI_SYNTAX" is set to false.	The role is set to `Outer` for some input groups of the joiner operator. The joiner operator will be split to binary joins. The value for the `SPLIT_JOIN_FOR_ANSI_SYNTAX` parameter in the migration utility configuration file will be ignored. For more information about this parameter, see `SPLIT_JOIN_FOR_ANSI_SYNTAX` in Configuration File Parameters.	No action.

Table A–2 (Cont.) Migration Utility Informational, Warning, and Error Messages

Message	Cause	Action
[MU-5022] Unable to find corresponding integration type in ODI according to the loading type {0} in OWB for operator {1}:{2}. Default integration type {3} is used.	ODI does not support integration types such as delete.	No action.
[MU-5023] Mapping is not migrated because the mapping operator {0}:{1}. Use the configuration option of migration utility "MIGRATE_UNBOUND_OPERATOR" or fix the mapping with unbound operators.	A mapping operator is unbound.	Configure the `MIGRATE_UNBOUND_OPERATOR` parameter in the migration utility configuration file or fix the mapping with unbound operators. For more information about this parameter, see `MIGRATE_UNBOUND_OPERATOR` in Configuration File Parameters.
[MU-5024] Migration of mapping operator {0}:{1} with data rules is not supported.	A mapping operator with data rules set is not supported for migration.	No action.
[MU-5025] The bound object of mapping operator {0}:{1} is not selected.	The bound object of the mapping operator is not selected for migration.	Check whether the bound object is explicitly selected using the `MIGRATION_OBJECTS` parameter in the migration utility configuration file, or whether the `MIGRATE_DEPENDENCIES` parameter is set to `TRUE`. For more information about these parameters, see Configuration File Parameters.
[MU-5026] Unable to generate ODI ExternalTable access parameter option for operator {0}:{1}. Details: {2}.	As described in the message.	Review the message to determine the cause of the problem and take appropriate action.
[MU-5027] Unable to migrate mapping with operator {0} because no {1} DataStore component hold the generated {2} for it.	The given mapping has no source data store component to hold the generated `BEGIN_MAPPING_SQL` or has no target data store component to hold the generated `END_MAPPING_SQL`. The Pre/Post mapping operator is migrated to `BEGIN/END_MAPPING_SQL` in ODI, but these two options rely on the source/target data store component. An exception is raised if the source/target data store component is not found.	No action.
[MU-5028]Unable to migrate mapping with operator {0} when store generated {1} into {2} DataStore component raised error: {3}.	Storing the generated `BEGIN/END_MAPPING_SQL` into a given ODI data store's KM option raised an unknown problem (for example, an illegal expression).	No action.

B

Reference to Migration Details

This appendix provides reference information about migrating OWB to ODI.

This appendix contains the following topics:

OWB Repositories

OWB Workspace to ODI Work Repository

When invoking the migration utility, the OWB Workspace Owner and its password are needed to connect to the OWB Repository. Each OWB Workspace Owner may have multiple workspaces. Only one workspace will be migrated with each migration. Therefore, one workspace name must be specified for each migration. Each OWB workspace will be migrated to ODI as one ODI Work repository.

If an OWB Workspace owner has multiple OWB Workspaces, each OWB Workspace should be migrated to an ODI Work repository of an ODI Master repository. The migration utility can only migrate at most one OWB Workspace at each time.

OWB Platform to ODI Technology

OWB Platforms and their associated data types are mapped to ODI Technologies and their associated data types. This platform and data type mapping is stored in a configuration file.

For the predefined platforms in OWB, the mappings to ODI can be found in the file `PlatformMappingsForMigration.xml` located in the *<ORACLE_HOME for OWB>*`/owb/bin/admin` directory.

If a user has defined new or custom Platforms in OWB, the mapping of this platform and its data types to ODI technology and its data types can be defined in the same configuration file. The physical name of the OWB Platform should be specified in the mapping, and the internal name of the ODI technology should be used.

The following table shows the predefined OWB Platform to ODI Technology mappings.

OWB Platform	ODI Technology
GENERIC	Generic SQL
ORACLE (including Oracle Workflow, Apps Concurrent manager)	Oracle
DB2UDB	IBM DB2 UDB
SQLSERVER	Microsoft SQL Server
SAP	SAP ABAP
FILE	File
OBIEE, OBISE, J2EE	These are not migrated.

Data type mapping differs for each OWB Platform mapping. The following tables show the data type mappings for each predefined OWB Platform.

If an OWB data type that has no mapping in ODI is used in Data Objects like Table, View, Materialized View, and External Table, the data object is reported as not migrated.

If an OWB data type that has no mapping in ODI is used in a Mapping Attribute, the data type of the mapping attribute is not set.

Data Type Mapping for OWB GENERIC Platform to ODI Generic SQL Technology

OWB Data Type (GENERIC)	ODI Data Type (Generic SQL)
BIGINT	BIGINT
BINARY	BINARY
BINARY_DOUBLE	BINARY_DOUBLE
BINARY_FLOAT	BINARY_FLOAT
BLOB	BLOB
BOOLEAN	CHAR
CHAR	CHAR
CLOB	CLOB
DATE	DATE
DATETIME	DATETIME
DECIMAL	DECIMAL
DOUBLE	DOUBLE
FLOAT	FLOAT
IMAGE	BLOB
INTEGER	INTEGER
INTERVAL DAY TO SECOND	INTERVAL DAY TO SECOND
INTERVAL YEAR TO MONTH	INTERVAL YEAR TO MONTH
LONG	CLOB
LONGVARBINARY	BLOB

OWB Data Type (GENERIC)	ODI Data Type (Generic SQL)
LONGVARCHAR	CLOB
MONEY	MONEY
NCHAR	NCHAR
NCLOB	NCLOB
NTEXT	NCLOB
NUMERIC	NUMERIC
NVARCHAR	NVARCHAR
NVARCHAR(MAX)	NCLOB
REAL	REAL
SMALLINT	SMALLINT
TEXT	CLOB
TIME	TIME
TIMESTAMP	TIMESTAMP
TIMESTAMP WITH TIME ZONE	TIMESTAMP WITH TIME ZONE
TINYINT	TINYINT
VARBINARY	VARBINARY
VARBINARY(MAX)	BLOB
VARCHAR	VARCHAR
VARCHAR(MAX)	CLOB
XMLTYPE	XMLTYPE

Data Type Mapping for OWB ORACLE Platform to ODI Oracle Technology

OWB Data Type (ORACLE)	ODI Data Type (Oracle)
BINARY_DOUBLE	BINARY_DOUBLE
BINARY_FLOAT	BINARY_FLOAT
BLOB	BLOB
CHAR	CHAR
CLOB	CLOB
DATE	DATE
FLOAT	FLOAT
INTEGER	NUMBER
INTERVAL DAY TO SECOND	INTERVAL DAY TO SECOND
INTERVAL YEAR TO MONTH	INTERVAL YEAR TO MONTH
LONG	LONG
LONG RAW	LONG RAW
MDSYS.SDOAGGRTYPE	
MDSYS.SDO_DIM_ARRAY	

OWB Data Type (ORACLE)	ODI Data Type (Oracle)
MDSYS.SDO_DIM_ELEMENT	
MDSYS.SDO_ELEM_INFO_ARRAY	
MDSYS.SDO_GEOMETRY	MDSYS.SDO_GEOMETRY
MDSYS.SDO_ORDINATE_ARRAY	
MDSYS.SDO_POINT_TYPE	
NCHAR	NCHAR
NCLOB	NCLOB
NUMBER	NUMBER
NVARCHAR2	NVARCHAR2
RAW	RAW
ROWID	ROWID
SYS.ANYDATA	
SYS.AQ$_JMS_BYTES_MESSAGE	
SYS.AQ$_JMS_MAP_MESSAGE	
SYS.AQ$_JMS_MESSAGE	
SYS.AQ$_JMS_STREAM_MESSAGE	
SYS.AQ$_JMS_TEXT_MESSAGE	
SYS.LCR$_ROW_RECORD	
TIMESTAMP	TIMESTAMP
TIMESTAMP WITH LOCAL TIME ZONE	TIMESTAMP WITH LOCAL TIME ZONE
TIMESTAMP WITH TIME ZONE	TIMESTAMP WITH TIME ZONE
UROWID	UROWID
VARCHAR	VARCHAR2
VARCHAR2	VARCHAR2
XMLFORMAT	XMLFORMAT
XMLTYPE	XMLTYPE

Data Type Mapping for OWB DB2UDB Platform to ODI IBM DB2 UDB Technology

OWB Data Type (DB2UDB)	ODI Data Type (IBM DB2 UDB)
BIGINT	BIGINT
BLOB	BLOB
CHARACTER	CHAR
CLOB	CLOB
DATE	DATE
DBCLOB	DBCLOB
DECIMAL	DECIMAL
DOUBLE	DOUBLE

OWB Data Type (DB2UDB)	ODI Data Type (IBM DB2 UDB)
FLOAT	FLOAT
GRAPHIC	GRAPHIC
INTEGER	INTEGER
LONG VARCHAR	LONG VARCHAR
LONG VARGRAPHIC	LONG VARGRAPHIC
NUMERIC	NUMERIC
REAL	REAL
SMALLINT	SMALLINT
TIME	TIME
TIMESTAMP	TIMESTAMP
VARCHAR	VARCHAR
VARGRAPHIC	VARGRAPHIC
XML	

Data Type Mapping for OWB SQLSERVER Platform to ODI Microsoft SQL Server Technology

OWB Data Type (SQLSERVER)	ODI Data Type (Microsoft SQL Server)
BIGINT	BIGINT
BINARY	BINARY
BIT	BIT
CHAR	CHAR
DATETIME	DATETIME
DECIMAL	DECIMAL
FLOAT	FLOAT
IMAGE	IMAGE
INT	INT
MONEY	MONEY
NCHAR	NCHAR
NTEXT	NTEXT
NUMERIC	NUMERIC
NVARCHAR	NVARCHAR
NVARCHAR(MAX)	NTEXT
REAL	REAL
SMALLDATETIME	SMALLDATETIME
SMALLINT	SMALLINT
SMALLMONEY	SMALLMONEY
SQL_VARIANT	SQL_VARIANT

OWB Data Type (SQLSERVER)	ODI Data Type (Microsoft SQL Server)
TEXT	TEXT
TIMESTAMP	TIMESTAMP
TINYINT	TINYINT
UNIQUEIDENTIFIER	UNIQUEIDENTIFIER
VARBINARY	VARBINARY
VARBINARY(MAX)	IMAGE
VARCHAR	VARCHAR
VARCHAR(MAX)	TEXT
XML	

Data Type Mapping for OWB FILE Platform to ODI File Technology

OWB Data Type (FILE)	ODI Data Type (File)
BYTEINT	BINARY_SIGNED_BIG_ENDIAN
CHAR	STRING
DECIMAL	EBCDIC_SIGNED_ZONED_DECIMAL
DATE	DATE
DECIMAL EXTERNAL	NUMERIC
DOUBLE	NUMERIC
FLOAT	NUMERIC
FLOAT EXTERNAL	NUMERIC
INTEGER	BINARY_SIGNED_BIG_ENDIAN
INTEGER UNSIGNED	BINARY_UNSIGNED_BIG_ENDIAN
INTEGER EXTERNAL	NUMERIC
INTERVAL DAY TO SECOND	DATE
INTERVAL YEAR TO MONTH	DATE
SMALLINT	BINARY_SIGNED_BIG_ENDIAN
SMALLINT UNSIGNED	BINARY_UNSIGNED_BIG_ENDIAN
TIMESTAMP	DATE
TIMESTAMP WITH TIME ZONE	DATE
TIMESTAMP WITH LOCAL TIME ZONE	DATE
VARRAWC	BINARY_SIGNED_BIG_ENDIAN
VARCHAR	STRING
VARCHARC	STRING
ZONED EXTERNAL	ASCII_SIGNED_ZONED_DECIMAL
ZONED	ASCII_SIGNED_ZONED_DECIMAL

Data Type Mapping for OWB SAP Platform to ODI SAP ABAP Technology

OWB Data Type (SAP)	ODI Data Type (SAP ABAP)
ACCP	ACCP
CHAR	CHAR
CLNT	CLNT
CUKY	CUKY
CURR	CURR
DATS	DATS
DEC	DEC
FLTP	FLTP
INT1	INT1
INT2	INT2
INT4	INT4
LANG	LANG
LCHR	LCHR
LRAW	LRAW
NUMC	NUMC
PREC	PREC
QUAN	QUAN
RAW	RAW
TIMS	TIMS
UNIT	UNIT

OWB Location to ODI Data Server

Each OWB Location is associated with an OWB Platform or equivalent ODI technology. Hence OWB location will be migrated to an ODI Data Server under the equivalent ODI technology.

Location Name to Data Server Name

Location Name will be migrated to ODI Data Server Name. Since OWB Location Name is unique within an OWB Workspace, while ODI Data Server Name is unique within the master repository, when there are several OWB workspaces for a Workspace Owner, each OWB Workspace should be migrated to a different ODI Master repository to avoid name conflicts.

Location Properties to Data Server Properties

The following table shows mapping of properties of OWB Location to properties of ODI Data Server:

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
		dataServerId (I_CONNECT)	NUMBER(10,0)	This number will be generated.

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
platform	NUMBER(9)	technology (I_TECHNO)	NUMBER(10,0)	
Name	VARCHAR2(1000)	name (CON_NAME)	VARCHAR2(35 CHAR)	
Driver Class	VARCHAR2(4000)	jdbcDriverName (JAVA_DRIVER)	VARCHAR2(400 CHAR)	
Url	VARCHAR2(4000)	jdbcUrl (JAVA_URL)	VARCHAR2(250 CHAR)	
User Name (CONNECT_AS_USER)	VARCHAR2(4000)	username (USER_NAME)	VARCHAR2(400 CHAR)	
Batch Update Size (UPDATE_SIZE)	NUMBER	batchUpdateSize (BATCH_UPDATE_SIZE)	NUMBER(10,0)	
Array Fetch Size (FETCH_SIZE)	NUMBER	fetchArraySize (FETCH_ARRAY_SERV)	NUMBER(10,0)	
Schema	VARCHAR2	schemaName (SCHEMA_NAME)	VARCHAR2(128 CHAR)	
Work Schema	VARCHAR2	workSchemaName (WSCHEMA_NAME)	VARCHAR2(128 CHAR)	
Catalog	VARCHAR2	catalogName (CATALOG_NAME)	VARCHAR2(128 CHAR)	
Work Catalog	VARCHAR2	workCatalogName (WCATALOG_NAME)	VARCHAR2(128 CHAR)	

Specific Location

For OWB Location using Database Link as the Connection Type, the location will be migrated to a new ODI Data Server, with the location name as the data server name. Other information for the location will not be migrated.

For File Location using FTP as the Connection Type, the location will be migrated to a new ODI Data Server with the location name as the data server name. Other information for the location will not be migrated.

OWB Modules to ODI Models

OWB Modules will be migrated to ODI Models.

Module Name to Model Name

To create a unique model name, the ODI Model name will be a concatenation of OWB Module Name and OWB Project name. If the resulting name is longer than the allowed length in ODI Model name, the resulting name will be trimmed.

Module Properties to Model Properties

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
		modelId (I_MOD)	NUMBER(10,0)	This number will be generated by the migration utility.

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Name	VARCHAR2(1000)	name (MOD_NAME)	VARCHAR2(400 CHAR)	
Platform	NUMBER(9)	technology (TECH_INT_NAME)	VARCHAR2(35 CHAR)	
		logicalSchema (LSCHEMA_NAME)	VARCHAR2(35 CHAR)	Will be created according to the OWB module name.
Name	VARCHAR(1000)	code (COD_MOD)	VARCHAR2(35 CHAR)	
Project	NUMBER(9)	parentModelFolder (I_MOD_FOLDER)	NUMBER(10,0)	
description	VARCHAR2(4000)	description (I_TXT_MOD)	NUMBER(10,0)	Description

Additional Migration of OWB Modules to ODI Folders

Some OWB Modules will also be migrated to ODI as ODI Folders, in addition to ODI Models. The following OWB modules will also be migrated as ODI Folders:

- Oracle Database Module
- Template Mapping Module

OWB Oracle Database Module will be migrated as ODI Model where the OWB Data Objects are migrated to, and also as ODI Folder where OWB mappings are migrated to.

OWB Template Mapping Module and Pluggable Mapping Folder will be migrated as ODI Folder.

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Name	VARCHAR2(1000)	Name (FOLDER_NAME)	VARCHAR2(400 CHAR)	

Physical Schema and Logical Schema

OWB supported a list of Data Locations for use with a module but only one location is selected to use at a time. This location is called the active location. During migration, only the active location will be migrated to ODI. The location is migrated as ODI Data Server. Corresponding to the location user name, a new ODI Physical Schema will be created in ODI if one does not exist already. The new ODI Physical Schema will be from the Location Schema of OWB Database Location, or the directory path for File Location.

Corresponding to the physical schema, a logical schema will be created in ODI if none with the same name as the Model name exists. The logical schema will set to "LS_" plus model name, and will be associated with the physical schema in the global context.

OWB Projects to ODI Projects

OWB Project will be migrated as ODI Project.

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
		projectId (I_PROJECT)	NUMBER(10,0)	This number will be generated.
Name	VARCHAR2(1000)	Name (PROJECT_NAME)	VARCHAR2(400 CHAR)	
Name	VARCHAR2(1000)	code (PROJECT_CODE)	VARCHAR2(35 CHAR)	

OWB Folders to ODI Folders

Two types of OWB Folders will be migrated to ODI:

- OWB Pluggable Mapping Folders

  OWB Pluggable Mapping Folders are migrated to ODI Folders; the name of the OWB Pluggable Mapping Folder will be the name of the ODI Folder.

- OWB Pluggable Mapping Standalone Folders

  Pluggable mappings in this OWB folder will be migrated to an ODI Folder named STAND_ALONE.

OWB Data Objects

OWB Table to ODI Datastore

OWB Table is migrated to ODI Datastore. The following related attributes of tables are migrated:

- Columns
- Keys
- Indexes

Attribute Sets and Data Rules are not migrated.

For Partitions, the partition name and the description are migrated, other properties are not migrated.

Attributes or properties of OWB Table are migrated to ODI Datastore as described in Table B–1, " OWB Data Object (Table, View, External Table, File, Materialized View) to ODI Datastore".

Attributes or properties of OWB Table Columns are migrated to ODI Datastore Columns as described in Table B–2, " OWB Table Column to ODI Datastore Column".

OWB Table supports these types of keys: Primary Key, Unique Key, Foreign Key, and Constraint.

- The attributes/properties of OWB Table Primary Keys and Unique Keys are migrated to ODI Keys as described in Table B–5, " OWB Key to ODI Key".
- The attributes/properties of OWB Table Constraints are migrated to ODI Condition as described in Table B–6, " OWB Check Constraint to ODI Condition".
- The attributes/properties of OWB Table Foreign Keys are migrated to ODI Reference as described in Table B–7, " OWB ForeignKey to ODI Reference".

The attributes/properties of Indexes are migrated to ODI Datastore Key as described in Table B–8, " OWB Index to ODI Key", which lists the mappings between the OWB Index and ODI Key.

OWB supports four types of indexes: unique, non-unique, bitmap, and function-based. A unique index will be mapped to OdiKey, and the key type will be set to ALTERNATE_KEY. A non-unique index will be mapped to OdiKey, and the key type will be set to INDEX. Bitmap and function-based keys are not migrated.

OWB View to ODI Datastore

OWB View is migrated to ODI Datastore. The following related attributes of OWB View are migrated:

- Columns
- Keys

Attribute Sets and Data Rules are not migrated.

Attributes or properties of OWB View are migrated to ODI Datastore as described in Table B–1, " OWB Data Object (Table, View, External Table, File, Materialized View) to ODI Datastore".

Attributes or properties of OWB View Columns are migrated to ODI Datastore Columns as described in Table B–2, " OWB Table Column to ODI Datastore Column".

OWB Table supports these types of keys: Primary Key, Unique Key, Foreign Key, and Constraint.

- The attributes/properties of OWB View Primary/Unique Keys are migrated to ODI Keys as described in Table B–5, " OWB Key to ODI Key".
- The attributes/properties of OWB View Constraints are migrated to ODI Condition as described in Table B–6, " OWB Check Constraint to ODI Condition".
- The attributes/properties of OWB View Foreign Keys are migrated to ODI Reference as described in Table B–7, " OWB ForeignKey to ODI Reference".

OWB Materialized View to ODI Datastore

OWB Materialized View is migrated to ODI Datastore. The following related attributes of Materialized views are migrated:

- Columns
- Keys
- Indexes

Attribute Sets and Data Rules are not migrated.

For Partitions, the partition name and the description are migrated, other properties are not migrated.

Attributes or properties of OWB Materialized View are migrated to ODI Datastore as described in Table B–1, " OWB Data Object (Table, View, External Table, File, Materialized View) to ODI Datastore".

Attributes or properties of OWB Materialized View Columns are migrated to ODI Datastore Columns as described in Table B–2, " OWB Table Column to ODI Datastore Column".

OWB Materialized View supports these types of keys: Primary Key, Unique Key, Foreign Key, and Constraint.

The attributes/properties of OWB Materialized View Primary Keys and Unique Keys are migrated to ODI Keys as described in Table B–5, " OWB Key to ODI Key".

The attributes/properties of OWB Materialized View Constraints are migrated to ODI Condition as described in Table B–6, " OWB Check Constraint to ODI Condition".

The attributes/properties of OWB Materialized View Foreign Keys are migrated to ODI Reference as described in Table B–7, " OWB ForeignKey to ODI Reference".

The attributes/properties of Indexes are migrated to ODI Datastore Key as described in Table B–8, " OWB Index to ODI Key", which lists the mappings between the OWB Index and ODI Key.

OWB External Table to ODI Datastore

OWB External Table is migrated to ODI Datastore. The following related attributes of External Table are migrated:

- Columns

Data Rules are not migrated. Associated locations will be migrated as ODI Data Server if the migration configuration option MIGRATE_DEPENDENCIES is set to true.

Attributes or properties of OWB External Table are migrated to ODI Datastore as described in Table B–1, " OWB Data Object (Table, View, External Table, File, Materialized View) to ODI Datastore".

Attributes or properties of OWB External Table Columns are migrated to ODI Datastore Columns as described in Table B–2, " OWB Table Column to ODI Datastore Column".

OWB External Table has association to OWB FLAT FILE and its access parameters. These associations will not be migrated to ODI.

OWB Flat File to ODI Datastore

OWB Flat File is migrated to ODI Datastore. The following related attributes of OWB Files are migrated:

- Records
- Fields

Attributes or properties of OWB Flat File are migrated to ODI Datastore as described in Table B–1, " OWB Data Object (Table, View, External Table, File, Materialized View) to ODI Datastore".

OWB Flat File may contain one or more Records. Each Record will be migrated as one ODI Datastore. The naming convention for the ODI Datastore name is `<FlatFileName>_<RecordName>`.

Attributes or properties of OWB File Record are migrated to ODI Datastore Columns as described in Table B–4, " OWB File Record to ODI Datastore Column".

Attributes or properties of OWB File Record Field are migrated to ODI Datastore Columns as described in section Table B–3, " OWB File Record Field to ODI Datastore Column".

OWB Sequence to ODI Sequence

OWB Sequence is migrated to ODI Sequence (Native sequence). OWB Sequence contains Columns, which are not migrated to ODI.

Attributes or properties of OWB Sequence are migrated to ODI Sequence as described in Table B–10, " OWB Sequence to ODI Sequence".

Property Migration Mapping Tables

Table B–1 OWB Data Object (Table, View, External Table, File, Materialized View) to ODI Datastore

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Name	VARCHAR2(1000)	defaultAlias (TABLE_ALIAS)	VARCHAR2(128 CHAR)	
ClassName	VARCHAR2(255)	dataStoreType (TABLE_TYPE)	VARCHAR2(2 CHAR)	OWB Table, File, and External Table are mapped to TABLE. OWB View and Materialized view are mapped to VIEW.
Description	VARCHAR2(4000)	Description (TABLE_DESC)	VARCHAR2(250 CHAR)	
SelectQuery	CLOB()			For view.
DefaultLocation	NUMBER(9)			For External Table.
RefersTo	NUMBER(9)			For External Table.
FixedWidth	CHAR(1)	Format (FILE_FORMAT)	VARCHAR2(1 CHAR)	For file/record.
FieldDelimiter	VARCHAR2(40)	fieldSeparator (FILE_SEP_FIELD)	VARCHAR2(24 CHAR)	For file/record.
leftEnclosure and rightEnclosure	VARCHAR2(40)	textDelimiter (FILE_ENC_FIELD)	VARCHAR2(2 CHAR)	For file/record.
RecordDelimiter	VARCHAR2(40)	rowSeparator (FILE_SEP_ROW)	VARCHAR2(24 CHAR)	For file/record.
SkipRecords	NUMBER(9)	skipHeadingLines (FILE_FIRST_ROW)	NUMBER(10,0)	For file/record.
Sourcefrom	VARCHAR2(255)	resourceName (RES_NAME)	VARCHAR2(400 CHAR)	For file/record.

Table B–2 OWB Table Column to ODI Datastore Column

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Name	VARCHAR2(1000)	Name (COL_NAME)	VARCHAR2(128 CHAR)	
		COL_DESC	VARCHAR2(250 CHAR)	Short description.
TypeDefinition	NUMBER(9) Stores an Id which points to a record in another table.	dataTypeCode (SOURCE_DT)	VARCHAR2(35 CHAR)	
Position	NUMBER(9)	position (POS)	NUMBER(10,0)	

Table B–2 (Cont.) OWB Table Column to ODI Datastore Column

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Length	NUMBER(9)	length (LONGC)	NUMBER(10,0)	
Precision	NUMBER(9)	Length (LONGC)	NUMBER(10,0)	
Scale	NUMBER(9)	scale (SCALEC)	NUMBER(10,0)	
Nullable	CHAR(1)	mandatory (COL_MANDATORY)	VARCHAR2(1 CHAR)	
dafaultValue	VARCHAR(4000)	defaultValue (DEF_VALUE)	VARCHAR2(100 CHAR)	
		scdType (SCD_COL_TYPE)	VARCHAR2(2 CHAR)	
description	Varchar(4000)	description (I_TXT_COL_DESC)	NUMBER(10,0) Stores an Id which points to a record in another table.	
fractionalsecondsprecision	NUMBER(9)	length (LONGC)	NUMBER(10)	

Table B–3 OWB File Record Field to ODI Datastore Column

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Name	VARCHAR2(1000)	Name (COL_NAME)	VARCHAR2(128 CHAR)	
TypeDefinition	NUMBER(9) Stores an Id which points to a record in another table.	dataTypeCode (SOURCE_DT)	VARCHAR2(35 CHAR)	
Position	NUMBER(9)	position (POS)	NUMBER(10,0)	
Sqlprecision	NUMBER(9)		NUMBER(10,0)	
Sqlscale	NUMBER(9)		NUMBER(10,0)	
Precision	NUMBER(9)	bytes (BYTES)	NUMBER(10,0)	
Scale	NUMBER(9)	scale (SCALEC)	NUMBER(10,0)	
StartPostion	NUMBER(9)	startPosition (FILE_POS)	NUMBER(10,0)	Only for file/record.
FieldLength	NUMBER(9)	bytes (BYTES)	NUMBER(10,0)	Only for file/record.
Nullable	CHAR(1)	mandatory (COL_MANDATORY)	VARCHAR2(1 CHAR)	
dafaultValue	VARCHAR(4000)	defaultValue (DEF_VALUE)	VARCHAR2(100 CHAR)	

Table B–3 (Cont.) OWB File Record Field to ODI Datastore Column

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
description	VARCHAR(4000)	description (I_TXT_COL_DESC)	NUMBER(10,0) Stores an Id which points to a record in another table.	
sqllength				
mask	VARCHAR2(255)	format (SNP_COL.COL_FORMAT)	VARCHAR2(35)	

Table B–4 OWB File Record to ODI Datastore Column

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Name		name (TABLE_NAME)		
Name	VARCHAR2(1000)	defaultAlias (TABLE_ALIAS)	VARCHAR2(128 CHAR)	
classname	VARCHAR2(255)	dataStoreType (TABLE_TYPE)	VARCHAR2(2 CHAR)	
Description	VARCHAR2(4000)	Description (TABLE_DESC)	VARCHAR2(250 CHAR)	
Prefix	VARCHAR2(40)			
Position	NUMBER(9)			
RecordClassifierValue	VARCHAR2(40)			
RecordSize	NUMBER(9)			

Table B–5 OWB Key to ODI Key

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Name	VARCHAR2(1000)	Name (KEY_NAME)	VARCHAR2(128 CHAR)	
Primarykey	BOOLEAN	keyType (CONS_TYPE)	VARCHAR2(2 CHAR)	keyType: PRIMARY_KEY(PK) ALTERNATE_KEY(AK)
Appslabel	VARCHAR2(255)			

Table B–6 OWB Check Constraint to ODI Condition

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Name	VARCHAR2(1000)	Name (KEY_NAME)	VARCHAR2(128 CHAR)	
Primarykey	BOOLEAN	keyType (CONS_TYPE)	VARCHAR2(2 CHAR)	keyType: PRIMARY_KEY(PK) ALTERNATE_KEY(AK)
Appslabel	VARCHAR2(255)			

Table B–7 OWB ForeignKey to ODI Reference

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
		referenceId (I_JOIN)	NUMBER(10,0)	This number will be generated.
Name		name (FK_NAME)	VARCHAR2(128 CHAR)	
Should map to DB_ REFERENCE		referenceType (FK_TYPE)	VARCHAR2(1 CHAR)	referenceType: DB_REFERENCE, ODI_REFERENCE, COMPLEX_ REFERENCE
		primaryDataStore (I_TABLE_PK)	NUMBER(10,0)	Find the table by Unique key.
module		primaryModel (PK_I_MOD)	NUMBER(10,0)	
		primaryDataStoreSchemaName (PK_SCHEMA)	VARCHAR2(128 CHAR)	Find the schema based on the model of the primary table.
		primaryDataStoreName (PK_TABLE_NAME)	VARCHAR2(128 CHAR)	Find primary table name by unique key.
		primaryDataStoreAlias (PK_TABLE_ALIAS)	VARCHAR2(128 CHAR)	Find the alias by primary data store.
Appslabel	VARCHAR2(255)			
Mandatory	CHAR(1)			
OnetoOne	CHAR(1)			

Table B–8 OWB Index to ODI Key

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Name	VARCHAR2(1000)	Name (KEY_NAME)	VARCHAR2(128 CHAR)	
Indextype		keyType (CONS_TYPE)	VARCHAR2(2 CHAR)	keyType: ALTERNATE_KEY(AK) INDEX(I)
Appslabel	VARCHAR2(255)			
Expression	CLOB()			
LocalIndex	CHAR(1)			
LocalPartitionType	VARCHAR2(40)			

Table B–9 OWB Partition to ODI Partition

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Name	VARCHAR2(1000)	name (PARTITION_NAME)	VARCHAR2(400 CHAR)	
Description	VARCHAR2(4000)	Description (PARTITION_DESC)	VARCHAR2(250 CHAR)	
classname	VARCHAR2(255)			

Table B–9 (Cont.) OWB Partition to ODI Partition

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
Attribute	VARCHAR2(40)			
Autosubpartitionordering	VARHCAR2(40)			
Hashsubpartitioncount	VARCHAR2(40)			
IsDefault	CHAR(1)			
IsSubPartition	CHAR(1)			
PartitionOrder	VARCHAR2(40)			

Table B–10 OWB Sequence to ODI Sequence

OWB Property Name	OWB Property Type	ODI Property Name	ODI Property Type	Note
		sequenceId (SEQ_ID)	NUMBER(10,0)	This number will be generated.
Project	NUMBER(9)	project (I_PROJECT)	NUMBER(10,0)	
Name		SEQ_NAME	VARCHAR2(400 CHAR)	
Increment By	VARCHAR2(2000)	incrementValue (INCR)	NUMBER(10,0)	Retrieve from active configuration.
		seqType (SEQ_TYPE)	VARCHAR2(1 CHAR)	OWB sequence is migrated as project sequence.
		type (IND_STD)	VARCHAR2(1 CHAR)	OWB sequence is migrated as native sequence.
		logicalSchemaName (LSCHEMA_NAME)	VARCHAR2(35 CHAR)	Via OWB module, the ODI Model's logical schema is used here.
Name		nativeSequenceName (DB_SEQ_NAME)	VARCHAR2(128 CHAR)	
Prefix	VARCHAR2(40)			
ExternalElementName	VARCHAR2(40)			
Proxy	CHAR(1)			
SynonymFor	VARCHAR2(255)			
ValidationResult	NUMBER(9)			

OWB Mappings

OWB Mapping is migrated to ODI Mapping. OWB Mappings are contained in Oracle module or Template Mapping Module while ODI Mappings are contained in Project Folder. OWB Project is migrated to ODI project, OWB Oracle Module or Template Mapping Module is migrated to ODI Project Folder.

OWB Mapping Properties

OWB Mapping Logical Properties

Logical properties of OWB Mapping are migrated to ODI Mapping Properties as follows:

OWB Property Name	Description	ODI Property Name	Note
Physical Name (NAME)		Name	
Business Name (LOGICAL_NAME)			
Execution Type (EXECUTION_TYPE)	BATCH, TRICKLE		TRICKLE mappings are not supported for migration.
Target Load Order (TARGET_LOAD_ORDER)		TARGET_LOAD_ORDER	
Created By			
Creation Time			
Description		Description	
Icon Object			
Last Update Time			
Update By			

OWB Mapping Physical Properties

Physical Properties of OWB Mappings are not migrated to ODI.

The following properties are not migrated:

OWB Property Name	Description	ODI Property Name	Note
Deployable (DEPLOYABLE)			Not migrated.
Generation Comments (GENERATION_COMMENTS)			Not migrated.
Language (GENERATION_LANGUAGE)	Choices = 'PLSQL, SQLLOADER, ABAP, UNDEFINED'		Not migrated.
Referred Calendar (REFERRED_CALENDAR)			Not migrated. Schedules are not supported for migration.

PLSQL Physical Properties

Chunking Options Properties for Chunking options are not migrated. Those properties are:

Chunk Method

Chunk table (NUMCOL_CHUNK_TABLE)
Chunk column (NUMCOL_CHUNK_COLUMN)
Chunk size (NUMCOL_CHUNK_SIZE)
Chunk table (ROWID_CHUNK_TABLE)
Chunk type (ROWID_CHUNK_TYPE)
Chunk size (ROWID_CHUNK_SIZE)
Chunk table (SQL_CHUNK_TABLE)
SQL statement (SQL_CHUNK_STATEMENT)
SQL statement chunk type (SQL_CHUNK_TYPE)

Runtime Parameters Properties for Runtime parameters are not migrated. Those properties are:

Analyze table sample percentage
Bulk size
Chunk execute resume task
Chunk force resume
Chunk number of times to retry
Chunk parallel level
Commit frequency
Default audit level
Default Operating Mode
Default purge group
Maximum number of errors

Code Generation Options Properties for code generation options are migrated as follows:

Property Name	Description	ODI Property Name	Note
Analyze table statements	Generate statistics collection statement if this is true.		Not migrated.
ANSI SQL Syntax (ANSI_SQL_SYNTAX)	A switch between ANSI and Oracle SQL syntax.	ODI has no such property defined on mapping, but ODI Join Component has similar property.	
AUTHID Option (AUTHID)	Generate the map with selected AUTHID option. Package will be executed with the permissions defined by the AUTHID clause rather than the package owner's permissions.		Not migrated.
Bulk Processing code	Generate bulk processing code if this is true.		Not migrated.
Commit Control (COMMIT_CONTROL)	Choices = 'AUTO_ COMMIT, AUTO_ CORR_COMMIT, MANUAL_COMMIT'		Not migrated.
Enable Parallel DML	Determine if Parallel DML is enabled at runtime.		Not migrated.

Property Name	Description	ODI Property Name	Note
Error trigger (ERROR_TRIGGER)	Error trigger procedure name		Not migrated.
Generation Mode	Choices = 'SET_ BASED, ROW_BASED, ROW_BASED_ TARGET_ONLY, SET_ BASED_FAIL_OVER_ TO_ROW_BASED, SET_BASED_FAIL_ OVER_TO_ROW_ BASED_TARGET_ ONLY, ALL_MODES'		Not migrated.
Optimized Code	Attempt to generate optimized code if this is true.		Not migrated.
PL/SQL Compilation Mode	Specifies the compilation mode for PL/SQL library unit. Choices = 'DEFAULT, INTERPRETED, NATIVE'		Not migrated.
Use Target Load Ordering (TARGET_LOAD_ORDERING)			Not migrated.

SQL*LOADER Physical Properties

SQL Loader Settings Properties for SQL Loader Settings are not migrated. Those properties are:

Bind Size
Byte Order Mark
Column Array Rows
Continue Load
Control File Location
Control File Name
Database File Name
Delimited File Record Termination
Direct Mode
Endian (Byte Order)
Errors Allowed
Load Last Field As Pieced
Log File Location
Log File Name
Multithreading
Nls Characterset
Operation Recoverable
Perform Parallel Load
Preserver Blanks
Read Buffers
Read Size
Records to Load
Records to Skip

Resumable
Resumable Name
Resumeable Timeout
Rows per Commit
Skip Index Maintenance
Skip Unusable Indexes
Stream size
Suppress discards
Suppress Errors
Suppress Feedback
Suppress Header
Suppress partitions

Runtime Parameters Properties for Runtime parameters are not migrated. Those properties are:

Audit
Default purge group

SQL Loader Data Files Properties for SQL Loader Data Files are not migrated. Those properties are:

Data File Name
Data File Location
Discard File Name
Discard File Location
Discard Max
Bad File Name
Bad File Location

ABAP Mapping Physical Properties

Runtime Parameters Properties for runtime parameters are not migrated, these properties are:

ABAP Report Name
Background Job
Control File Name
Data File Name
File Delimiter for Staging File
Include FTP
Install only
Log File Name
SAP Location
SAP System Version
Sql Join Collapsing
Staging File Directory
Timeout

SQL Loader Settings Properties for SQL Loader Setting are not migrated, those properties are:

NLS Characterset

SQLPLUS Mapping Physical Properties

SQL*Plus Settings The properties for SQL*Plus Settings are not migrated. Those properties are:

ARRAYSIZE
COPYCOMMIT
Log File Directory
Log File Name
LONG
SQL File Directory
SQL File Name

Runtime Parameters The properties for Runtime Parameters are not migrated. Those properties are:

Audit
Default purge group

Code Template Mappings Physical Properties

Chunking Options Properties for Chunking options are not migrated. (The same as PLSQL mappings.)

Code Generation Options Properties for code generation options are migrated as follows:

OWB Property Name	Description	ODI Property Name	Note
Analyze table statements	Generate statistics collection statement if this is true.		Not migrated.
ANSI SQL Syntax (ANSI_SQL_SYNTAX)	A switch between ANSI and Oracle SQL syntax.	ODI has no such property defined on the mapping, but ODI Join Component has a similar property, see migration on Join Operator.	
AUTHID Option (AUTHID)	Generate the map with selected AUTHID option. Package will be executed with the permissions defined by the AUTHID clause rather than the package owner's permissions.		Not migrated.
Bulk Processing code	Generate bulk processing code if this is true.		Not migrated.
Commit Control (COMMIT_CONTROL)	Choices='AUTO_ COMMIT, AUTO_ CORR_COMMIT, MANUAL_COMMIT'		Not migrated.

OWB Property Name	Description	ODI Property Name	Note
Enable Parallel DML	Determine if PDML is enabled at runtime.		Not migrated.
Error trigger (ERROR_TRIGGER)	Error trigger procedure name.		Not migrated.
Generation Mode	Choices='SET_BASED, ROW_BASED, ROW_BASED_TARGET_ONLY, SET_BASED_FAIL_OVER_TO_ROW_BASED, SET_BASED_FAIL_OVER_TO_ROW_BASED_TARGET_ONLY, ALL_MODES'		Not migrated.
Optimized Code	Attempt to generate optimized code if this is true.		Not migrated.
Use Enclosure Char			Not migrated.
Use Target Load Ordering (TARGET_LOAD_ORDERING)			Not migrated.

Runtime Parameters Properties for runtime parameters are not migrated. Those properties are:

Analyze table sample percentage
Bulk size
Commit frequency
Default audit level
Default Operating Mode
Default purge group
Maximum number of errors

SCD Updates Properties for SCD Updates are not migrated. Those properties are:

Strategy

Multiple Target Mapping Migration

For mappings with multiple targets, target load order and Multiple Target Insert (MTI) are considered for migration.

Target Load Order

The OWB Target Load Order property is migrated to the ODI Target Load Order property.

The OWB Use Target Load Ordering property is not migrated, because this property does not exist in ODI.

Multiple Target Insert (MTI)

When an OWB mapping has multiple targets to insert, the data is coming from the same sources, and the Optimized code option is set to true, during code generation, a

single insert statement for all targets may be generated instead of a multi-table insert SQL statement.

Because this property is a physical property and MTI occurs at code generation, MTI is not supported for migration.

Mapping Operator

Logical properties of the OWB Mapping Operator are migrated to ODI Mapping Component Properties as follows:

OWB Property Name	ODI Property Name	Note
Business Name (LOGICAL_NAME)	Business Name (BUSINESS_NAME)	
Create By		Not migrated.
Create Time		Not migrated.
Description (Description)	Description (DESCRIPTION)	
Icon Object		Not migrated.
Last Update Time		Not migrated.
Physical Name (NAME)	Name (NAME)	
Update By		Not migrated.

The above properties are common properties for the Mapping operator.

Mapping Attribute

General Properties

General properties of OWB Mapping Attribute are migrated to ODI Mapping Attribute properties as follows:

OWB Property Name	ODI Property Name	Note
Physical Name (NAME)	Name	
Business Name (LOGICAL_NAME)		Not migrated.
Created By		Not migrated.
Creation Time		Not migrated.
Description	Description	
Icon Object		Not migrated.
Last Update Time		Not migrated.
Update By		Not migrated.

Data Type Information

Data type information properties on attribute of OWB Table operator are as follows:

OWB Property Name	ODI Property Name	Note
Data Type (DATA_TYPE)	Data type	Convert the OWB data type to ODI data type according the data type mappings.
Fractional Seconds precision (FRACTIONAL_SECONDS_PRECISION)		Not migrated.
Length (Length)	Size	For data type which allows length.
Precision (Precision)	Size	For data type which allows precision.
Scale (Scale)	Scale	

Mapping Attributes of OWB Mapping Operator use OWB GENERIC platform data types. OWB GENERIC platform is mapped to ODI Generic SQL technology. See "Data Type Mapping for OWB GENERIC Platform to ODI Generic SQL Technology" on page B-2 for details.

OWB Pluggable Mappings

OWB Pluggable Mapping is migrated to ODI Reusable Mapping.

Also see Pluggable Mapping Operator.

Pluggable Mapping Folder

The OWB Pluggable Mapping Folder is migrated to an ODI Project Folder. Standalone pluggable mappings are migrated to a Project Folder named STAND_ALONE, which is created automatically during migration if it does not already exist.

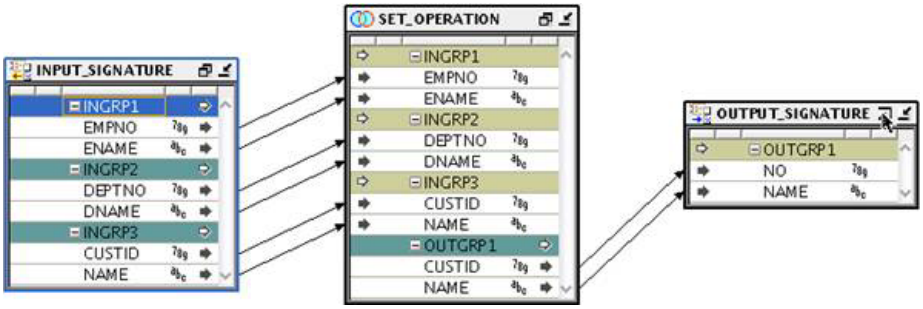
Properties of Pluggable Mapping

Only Physical name and Description are migrated. Physical name of OWB Pluggable Mapping is migrated to name of ODI Reusable Mapping. Description of OWB Pluggable Mapping is migrated to Description of ODI Reusable Mapping.

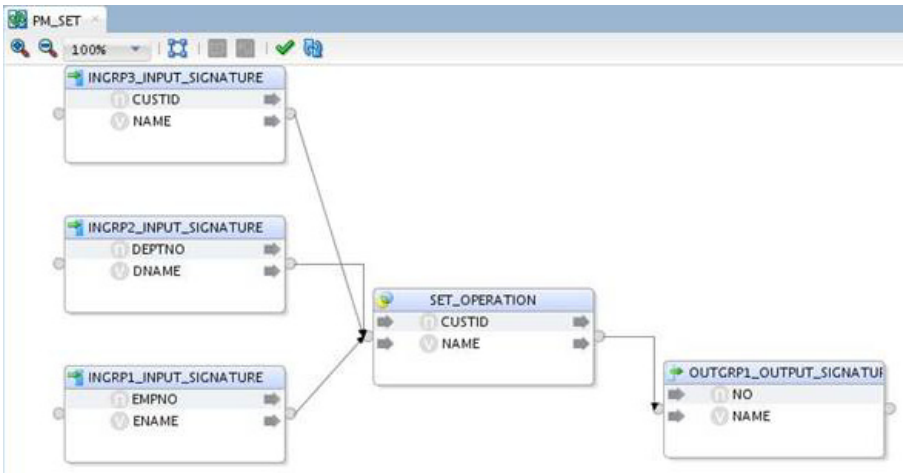
Input Signature and Output Signature

In OWB, Signature Operator can have unlimited attribute groups (for Input Signature Operator, the attribute groups are output groups; for Output Signature Operator, the attribute groups are input groups). In ODI, Signature Component can have only one connector point, so each attribute group of OWB Signature Operator is migrated to a Signature Component.

For example, the following figure shows a Pluggable Mapping for which the INPUT_SIGNATURE operator has three output groups (INGRP1, INGRP2, and INGPR3).



These OWB output groups are migrated to three Input Signature Components in ODI, as shown in the following figure.



The name of ODI Signature Component is composed of OWB attribute group name of Signature Operator, underscore (_), and Signature Operator name.

The attributes of Signature Operator in OWB are migrated to attributes of Signature Component in ODI. No special properties need to be migrated for signature attributes.

Join Operator in Pluggable Mapping

OWB Pluggable Mapping does not have the property ANSI SQL Syntax as does a regular OWB Mapping. Therefore, all Join Operators of a Pluggable Mapping are split into binary joins during migration to ODI unless the migration configuration option SPLIT_JOIN_FOR_ANSI_SYNTAX is set to false in the migration utility configuration file. For information about ordered join, see Join Operator.

C

Migration Details for Operators

This appendix provides reference information about migrating operators from OWB to ODI.

This appendix contains the following topics:

Common Properties

The following OWB properties are migrated to the same ODI properties across all the operators and attributes for which they are defined.

OWB Property Name	ODI Property Name
Physical Name	Name
Description	Description

Aggregate Operator

The OWB Aggregate operator is migrated to the ODI Aggregate component.

Logical Properties of the Aggregate Operator

OWB Property Name	Description	ODI Property Name	Note
Having Clause (HAVING_CLAUSE)	Having Clause	HAVING	
Group By Clause (GROUP_BY_CLAUSE)	Group By Clause	MANUAL GROUP BY CLAUSE	

Physical Properties of the Aggregate Operator

OWB Property Name	Description	ODI Property Name	Note
Inline view hint (INLINEVIEW_HINT)	Hint used when inline view is created for this operator		Not migrated.

Attribute Groups and Attributes of the Aggregate Operator

Output attributes of the Aggregate operator are migrated to output attributes of the Aggregate component in ODI. No specific properties of output attributes need to be migrated.

Deduplicator Operator

The OWB Deduplicator operator is migrated to the ODI Distinct component.

Properties of the Deduplicator Operator

No specific properties of the Deduplicator operator need to be migrated.

Attribute Groups and Attributes of the Deduplicator Operator

Input attributes of the Deduplicator operator are not migrated.

Output attributes of the Deduplicator operator are migrated. No specific properties of output attributes need to be migrated.

Expression Operator

The OWB Expression operator is migrated to the ODI Expression component.

Properties of the Expression Operator

No specific properties of the Expression operator need to be migrated.

Attribute Groups and Attributes of the Expression Operator

Input attributes of the Expression operator are not migrated.

Output attributes of the Expression operator are migrated.

For output attributes, the expression of the output attribute is migrated to the expression of the ODI attribute. The OWB properties Variable Initial Value and Variable Write condition are not migrated. No other specific properties of output attributes need to be migrated.

External Table Operator

OWB External Table operators inside OWB mappings are migrated to ODI Datastore components in the migrated ODI mappings. For detailed migration steps and behaviors, see "Migrating the External Table Operator" on page C-5.

Logical Properties of the External Table Operator

General Properties

OWB Property Name	Description	ODI Property Name	Note
Bound Name (BOUND_NAME)			If the OWB External Table operator is bound to an external table, the ODI Datastore component is bound to the corresponding data store.
Primary Source (PRIMARY_SOURCE)	A boolean value to indicate whether this is a primary source (only used in EDW). (YES/NO)		Not migrated.
Key (KEYS_READONLY)			Not migrated.

Chunking

As with the Table operator, properties for Chunking are not migrated.

Error Table

As with the Table operator, properties for Error Table are not migrated.

SCD Updates

As with the Table operator, properties for SCD Updates are not migrated.

Temp Stage Table

As with the Table operator, properties for Temp Stage Table are not migrated.

Physical Properties of the External Table Operator

General Properties

OWB Property Name	Description	ODI Property Name	Note
Schema (SCHEMA)			Not migrated.
Database link (DATABASE_LINK)	Database link used to access this entity during mapping.		Not migrated.
Location (DB_LOCATION)	Location, used to access referenced entity.		Not migrated.

Hints

OWB Property Name	Description	ODI Property Name	Note
Extraction hint (EXTRACTION_HINT)	Hint used when extracting from this table using SQL	SELECT_HINT	
Loading hint (LOADING_HINT)	Hint used when loading into this table using SQL	INSERT_HINT or UPDATE_HINT	
Automatic hints enabled (AUTOMATIC_HINTS_ENABLED)	Automatic hints enabled using SQL		Not migrated.

Partition Exchange Loading

As with the Table operator, properties for Partition Exchange Loading are not migrated.

Constraint Management

OWB Property Name	Description	ODI Property Name	Note
Enable Constraints (ENABLE_CONSTRAINTS)	Enable Constraints		Not migrated.
Exceptions Table Name (EXCEPTIONS_TABLE_NAME)	Exceptions Table Name		Not migrated.

Migrating the External Table Operator

OWB External Table operators inside OWB mappings are migrated to ODI Datastore components in the migrated ODI mappings.

The KM of the ODI Datastore's Physical Mapping is set to XKM Oracle External Table, and the following information is migrated from the OWB External Table Operator (or its bound external table) to KM options of the ODI Physical Node.

OWB Property Name	KM Option	Note
Default Location	SQL_DEFAULT_DIR	
Accessed Data Location	SQL_DIRECTORIES	The format is *DIR_NAME*:*path*,...; for example: MyDir:/tmp/mydir, MyDir2:/tmp/mydir2
Data Files	DIR_DATA_FILES	The format is *DIR_NAME*:*filename*,...; for example: MyDir:file1,MyDir:file2
Access Parameters	ACCESS_PARAMETERS	

Flat File Operator

OWB Flat File operators inside OWB mappings are migrated to ODI Datastore components in the migrated ODI mappings.

Logical Properties of the Flat File Operator

OWB Property Name	Description	ODI Property Name	Note
Loading type (LOADING_TYPE)	Choices = 'INSERT, UPDATE, NONE'	INTEGRATION_TYPE	Same as for the Table operator. See Notes About Loading Type.
SAMPLED_FILE_NAME	The default physical source file name.		Not migrated.
Source Data File Location (SOURCE_DATA_FILE_LOCATION)	The Locations of the File Module of this Flat File at the time of reconciliation. Stored as UOID.		Not migrated.
File Format (FILE_FORMAT)	File Format (Fixed or Delimited).		Not migrated.
Record Delimiter (RECORD_DELIMITER)	Character that indicates the end of the record.		Not migrated.
Continuation Character (CONTINUATION_CHARACTER)	Character that indicates the record is continued on the next line.		Not migrated.
Continuation Character on Next Line (CONTINUATION_CHARACTER_ON_NEXT_LINE)	If there is a continuation character, is it at the start of the line.		Not migrated.

OWB Property Name	Description	ODI Property Name	Note
Filed Termination Character (FIELD_TERMINATION_CHARACTER)	Character that separates the fields of a delimited file.		Not migrated.
Filed Enclosure Characters (FIELD_ENCLOSURE_CHARACTERS)	Characters that wrap fields. Example ' or ".		Not migrated.
Record Size (RECORD_SIZE)	Size of a fixed length record.		Not migrated.
Concatenate Records (CONCATENATE_RECORDS)	Number of Physical Records per Logical Record.		Not migrated.
Record Type Position (RECORD_TYPE_POSITION)	If this is a multi record file, this will indicate the position of the field that identifies the type of record.		Not migrated.
Record Type Length (RECORD_TYPE_LENGTH)	If this is a multi record file, this will indicate the length of the data that identifies the type of record. It is used with the Record Type Position.		Not migrated.
File contains a header row (FIELD_NAMES_IN_THE_FIRST_ROW)	Indicates whether file contains a header row		Not migrated.
Bound Name (BOUND_NAME)			If the OWB Flat File operator is bound to an OWB Flat File object, the corresponding ODI Datastore component is bound to the ODI Datastore.

Logical Properties of the Map Attribute Group of the Flat File Operator

OWB Property Name	Description	ODI Property Name	Note
Record Type Values (RECORD_TYPE_VALUES)			Not migrated.
Bound Name (BOUND_NAME)			Not migrated.

Logical Properties of the Map Attribute of the Flat File Operator

OWB Property Name	Description	ODI Property Name	Note
Field Data Type (FIELD_DATA_TYPE)	Choices = 'CHAR, DATE, INTEGER EXTERNAL, FLOAT EXTERNAL, DECIMAL, DECIMAL EXTERNAL, ZONED, ZONED EXTERNAL, RAW, TIMESTAMP, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIME ZONE, INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND, FLOAT, DOUBLE, BYTEINT, SMALLINT, SMALLINT UNSIGNED, INTEGER, INTEGER UNSIGNED, GRAPHIC, GRAPHICEXTERNAL, VARGRAPHIC, VARCHAR, VARCHARC, VARRAW, LONG VARRAW, VARRAWC'		Not migrated. Data type of ODI map attribute is determined by the data type of the column of the bound datastore.
Filed Length (FIELD_DATA_TYPE_LENGTH)	Length of the field in the file to which this operator is bound.		Not migrated. Length of ODI map attribute is determined by the length of the column of the bound datastore.
Field Precision (FIELD_DATA_TYPE_PRECISION)	Precision of the field in the file to which this operator is bound.		Not migrated. Precision of ODI map attribute are determined by the length of the column of the bound datastore.
Field Scale (FIELD_DATA_TYPE_SCALE)	Scale of the field in the file to which this operator is bound.		Not migrated.
Field starting position (FIELD_START_POSITION)			Not migrated.
Field ending position (FIELD_END_POSITION)			Not migrated.
Field Mask (FIELD_MASK)	Date mask of the field in the file to which this operator is bound.		Not migrated.

OWB Property Name	Description	ODI Property Name	Note
Field null if condition (FIELD_NULLIF_VALUE)	NULLIF value of the field in the file to which this operator is bound.		Not migrated.
Field default if condition (FIELD_DEFAULTIF_VALUE)			Not migrated.

Join Operator

The OWB Join operator is migrated to the ODI Join component. Attribute groups and attributes of the OWB Join operator are not migrated.

Properties of the Join Operator

For information about the general properties of the Join operator, see "Mapping Operator" on page B-24.

ANSI SQL syntax

ANSI SQL syntax is a property on the mapping level in OWB.

ODI does not have this property on the mapping level, but the ODI Join component has a property called Generate ANSI Syntax which has the same functionality.

The value of ANSI SQL syntax on the OWB mapping is migrated to the Generate ANSI Syntax property of the ODI Join component.

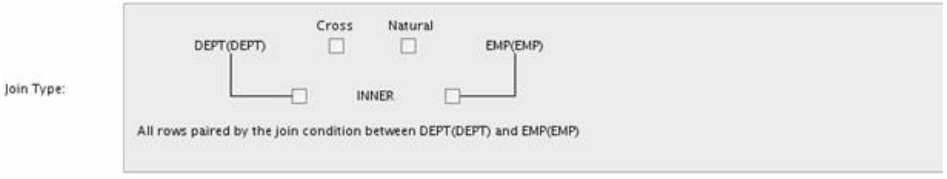
Join Condition

Join Condition on the OWB Join operator is migrated to Join Condition on ODI Join component. However, the OWB Join Condition references its own operator's input attributes, which is not supported in ODI; thus, the ODI Join Condition is configured to reference the attributes of the upstream sources to the OWB input attribute.

Join Input Role

Join Input Role is an attribute group level property of the OWB Join operator. It has three choices: STANDARD, OUTER and FULLOUTER.

The corresponding property on the ODI Join component is Join Type.



Join Input Role does not map directly to Join Type because Join Input Role supports multiple input groups, while Join Type supports only a binary join. During migration, complex joins are split into a series of the binary joins using the OWB code generation rules for the Join operator.

Migrating an ANSI Join Operator

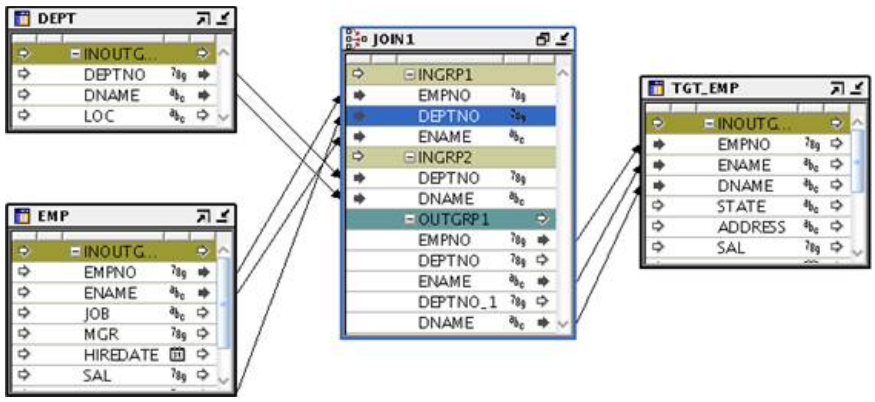
When ANSI SQL syntax of OWB mapping is set to true, the Join operator is by default split into binary joins during migration.

Setting the SPLIT_JOIN_FOR_ANSI_SYNTAX migration configuration option to false can override this default behavior and prohibit the Join operator from being split into binary joins. However, if a "Join Input Role" value is set on any of the Join operator's attribute groups, the value of the SPLIT_JOIN_FOR_ANSI_SYNTAX migration configuration option is ignored and the Join operator is split into binary joins during migration.

The following scenarios provide examples of migrating the Join operator when ANSI SQL Syntax is set to true for the mapping.

Scenario 1: Two Input Groups with Standard Join

OWB mapping description: two sources joining together, the join condition is standard join (not outer join). No "Join Input Role" is specified on input attribute groups of Join operator.

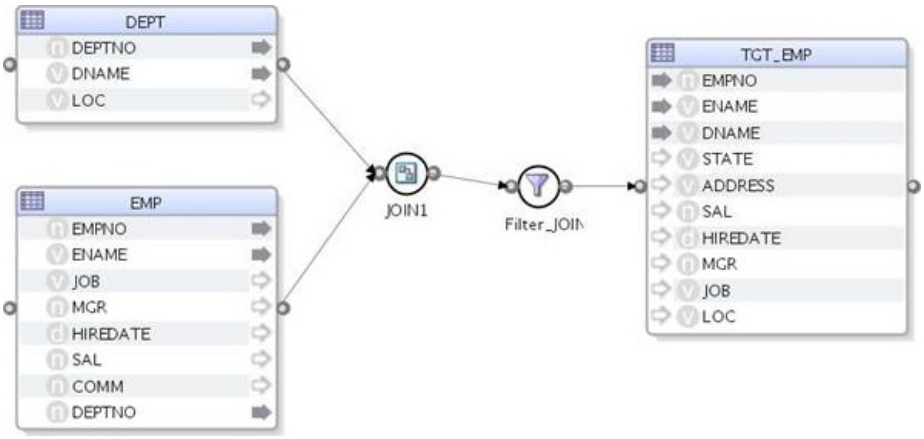


Join Condition is: `INGRP2.DEPTNO = INGRP1.DEPTNO and INGRP1.EMPNO > 1000`

The generated code (only displays the select clause) from OWB side is:

```
SELECT
  "EMP"."EMPNO" "EMPNO", "EMP"."ENAME" "ENAME", "DEPT"."DNAME" "DNAME"
FROM
  "DEPT" "DEPT" JOIN "EMP" "EMP"
ON ( ( "DEPT"."DEPTNO" = "EMP"."DEPTNO" ) )
WHERE ( "EMP"."EMPNO" > 1000 )
```

When this kind of mapping is migrated to ODI, the ODI mapping should look as follows:



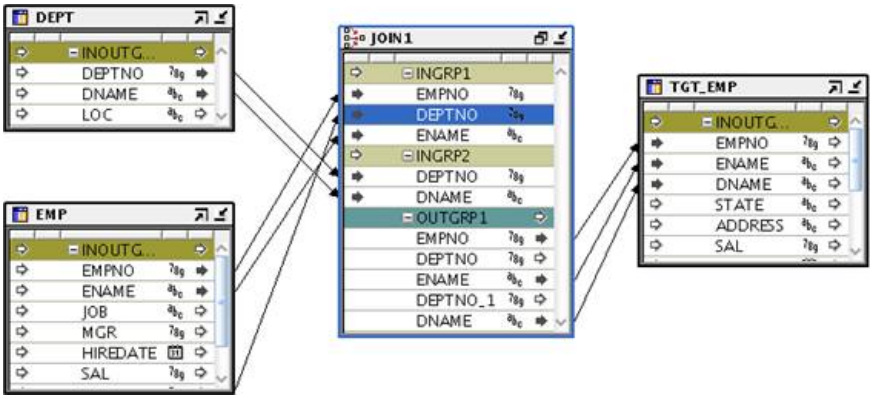
The join condition for JOIN1 is: `( DEPT.DEPTNO = EMP.DEPTNO )`

The filter condition for Filter_JOIN1 is: `( EMP.EMPNO > 1000 )`

The operator JOIN1 in OWB mapping is migrated to a Join component followed a Filter component in ODI.

Scenario 2: Two Input Groups with Outer Join Using (+) Style

The mapping is much similar with the mapping in scenario 1. The only difference is the join condition is not a standard join. It is an outer join using (+) style.

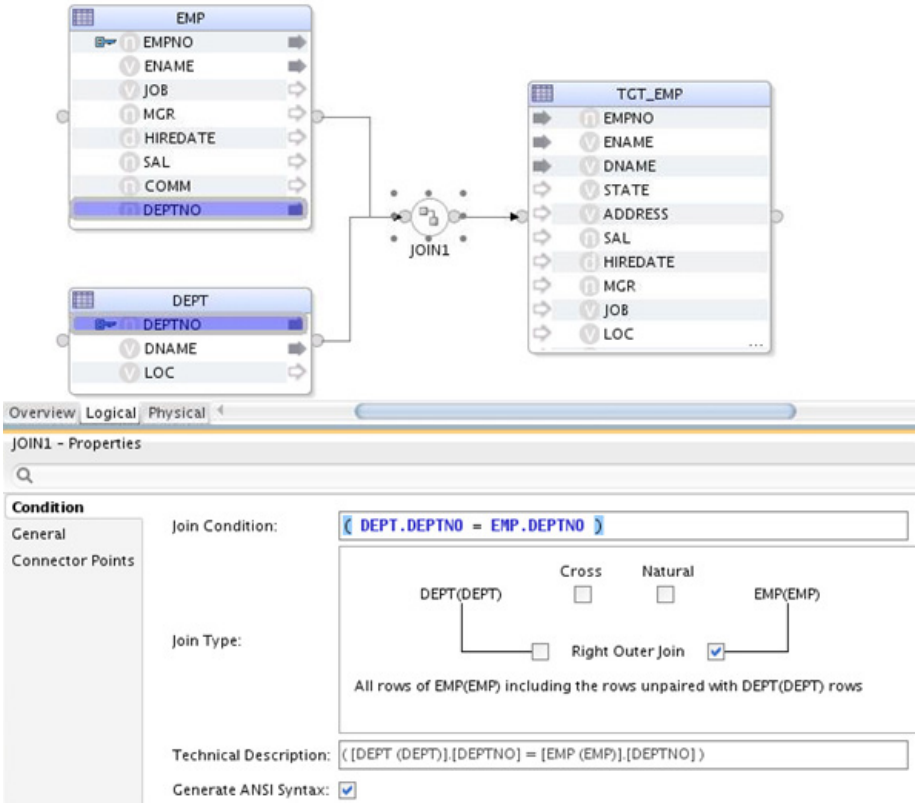


JOIN_CONDITION is: `INGRP2.DEPTNO(+) = INGRP1.DEPTNO`

The generated code (only displays the select clause) from OWB side is:

```
SELECT
  "EMP"."EMPNO" "EMPNO","EMP"."ENAME" "ENAME", "DEPT"."DNAME" "DNAME"
FROM
  "DEPT" "DEPT"
RIGHT OUTER JOIN "EMP" "EMP" ON ( ( "DEPT"."DEPTNO" = "EMP"."DEPTNO" ) )
```

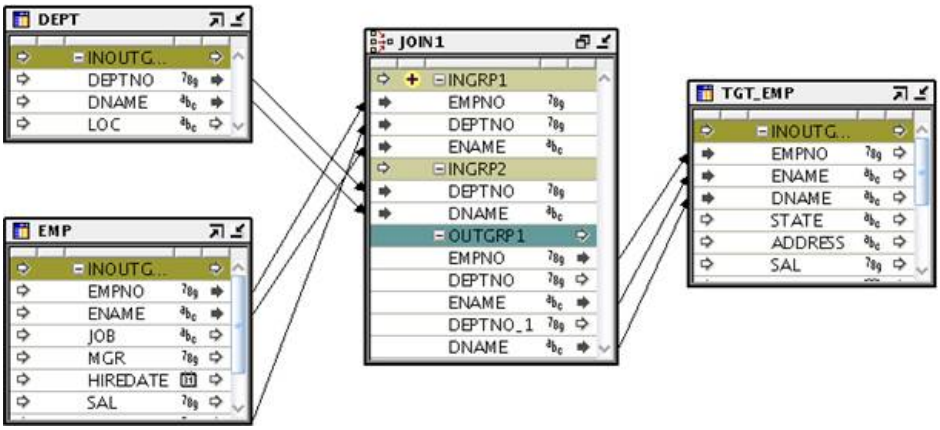
The migrated ODI mapping should look as follows:



The join condition is set to `DEPT.DEPTNO = EMP.DEPTNO`, and the join type is set to `DEPT RIGHT_OUTER join EMP`.

Scenario 3: Two Input Groups with Outer Join Using Join Input Role

Two sources joining together, the join condition is standard join, but "Join Input Role" is specified on some of the input attribute groups of Join operator. Take the following OWB mapping as an example:



The Join Input Role of INGRP1 is set to OUTER.

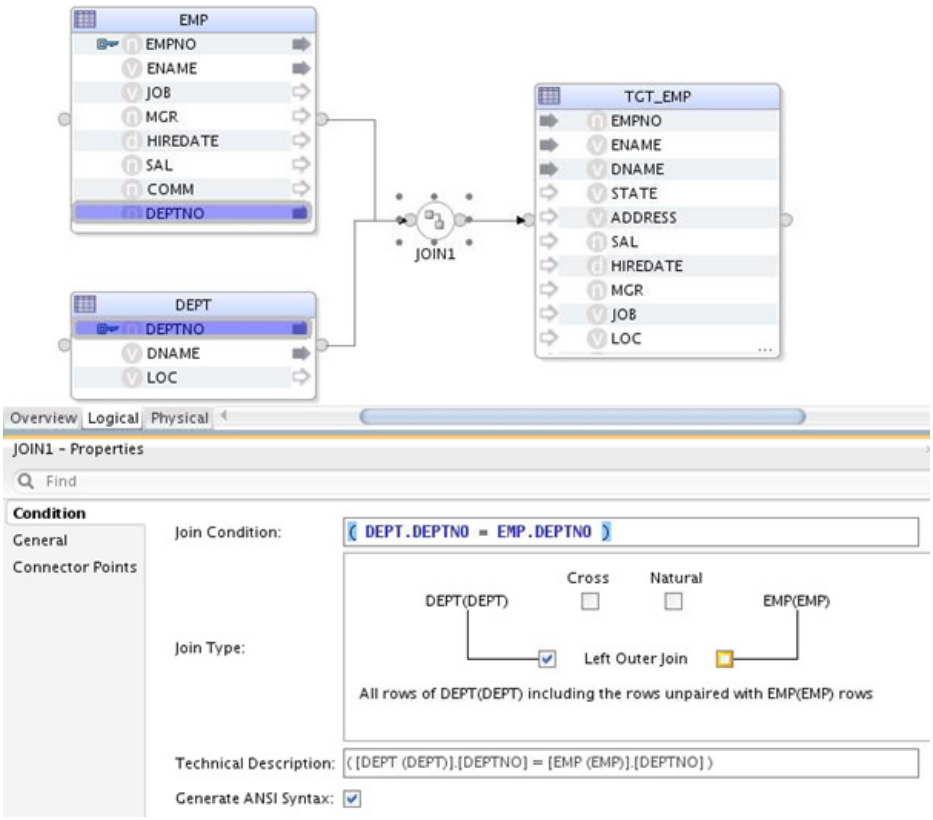
Join condition is: `INGRP2.DEPTNO = INGRP1.DEPTNO`

The generated code (only displays the select clause) from OWB side is:

```
SELECT
  "EMP"."EMPNO" "EMPNO",
```

```
  "EMP"."ENAME" "ENAME",
  "DEPT"."DNAME" "DNAME"
FROM
  "DEPT" "DEPT"
LEFT OUTER JOIN "EMP" "EMP" ON ( ( "DEPT"."DEPTNO" = "EMP"."DEPTNO" ) )
```

The migrated ODI mapping looks like the following:



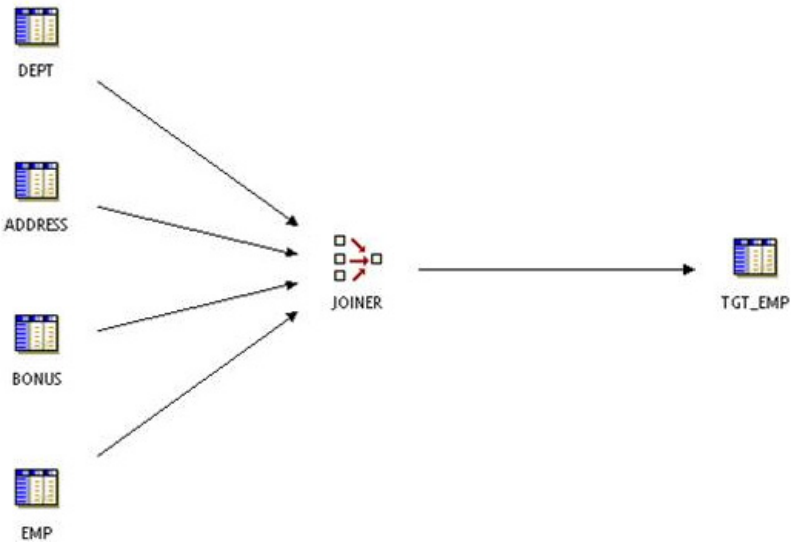
The join condition is set to `DEPT.DEPTNO = EMP.DEPTNO`, and the join type is set to `DEPT LEFT_OUTER join EMP`.

Scenario 4: Two Input Groups with both (+) Style and Join Input Role

In this case, OWB will use Join Input Role to generate code and ignore the (+) style. The migrated mapping will be the same as Scenario 3.

Scenario 5: Multiple Input Groups

Take the following mapping as an example:



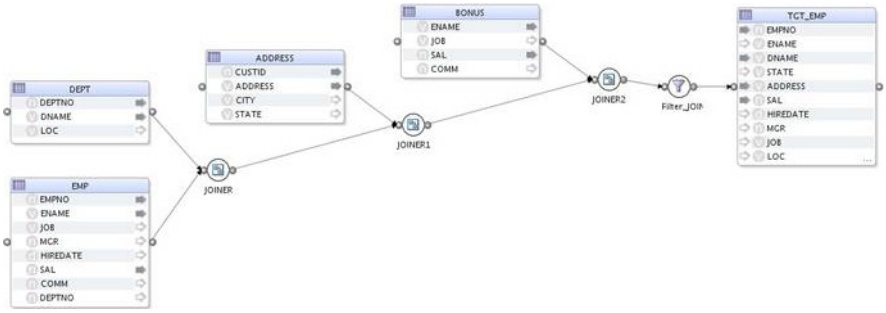
Join condition is:

```
INGRP1.SAL > 1000 and INGRP1.EMPNO(+) = INGRP2.DEPTNO
and INGRP3.ENAME = INGRP4.CUSTID and INGRP1.EMPNO = INGRP4.CUSTID
and SUBSTR(INGRP1.ENAME(+),0,2) = INGRP2.DNAME
```

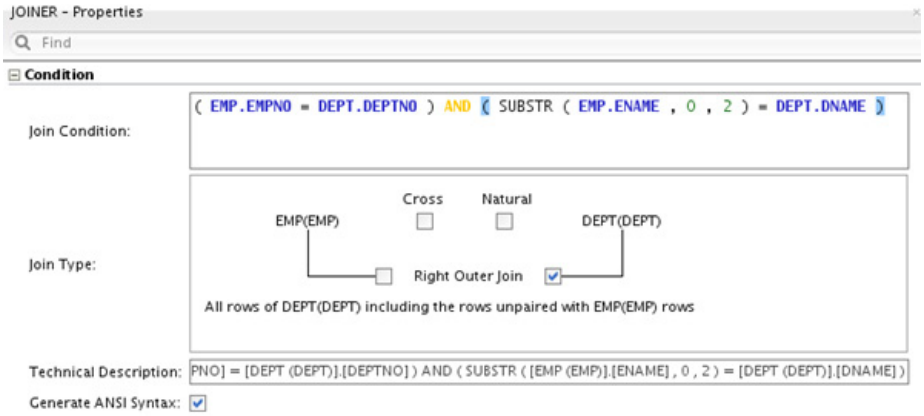
The generated code (only displays the select clause) from OWB side is:

```
SELECT
  /* EMP.INOUTGRP1 */
  "EMP"."EMPNO" "EMPNO",
  "EMP"."ENAME" "ENAME",
  "EMP"."JOB" "JOB",
  "EMP"."MGR" "MGR",
  "EMP"."HIREDATE" "HIREDATE",
  "EMP"."SAL" "SAL",
  "EMP"."COMM" "COMM",
  "EMP"."DEPTNO" "DEPTNO"
FROM
  "EMP" "EMP" ) "INGRP1"
RIGHT OUTER JOIN "DEPT" "DEPT" ON (
  (( "INGRP1"."EMPNO" = "DEPT"."DEPTNO" ))
  AND (( SUBSTR ( "INGRP1"."ENAME" , 0 , 2 ) = "DEPT"."DNAME" ))
)
JOIN "ADDRESS" "ADDRESS$1" ON ( ( "INGRP1"."EMPNO" = "ADDRESS$1"."CUSTID" ) )
JOIN "BONUS" "BONUS" ON ( ( "BONUS"."ENAME" = "ADDRESS$1"."CUSTID" ) )
WHERE
  ( "INGRP1"."SAL" > 1000 )
```

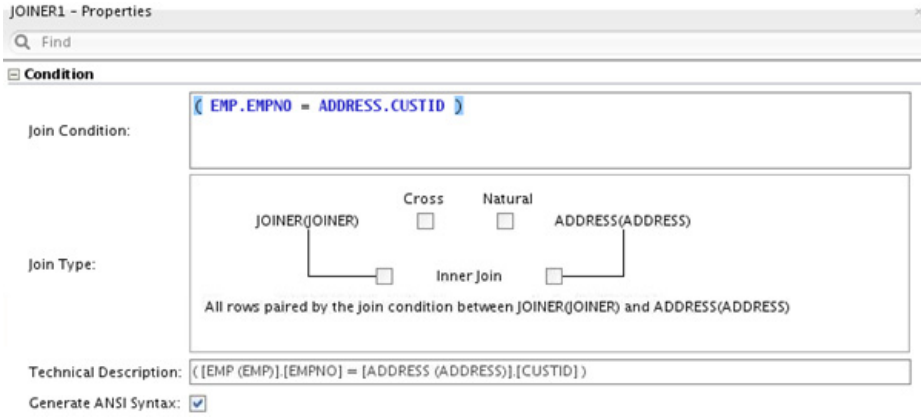
The migrated ODI mapping looks like this:



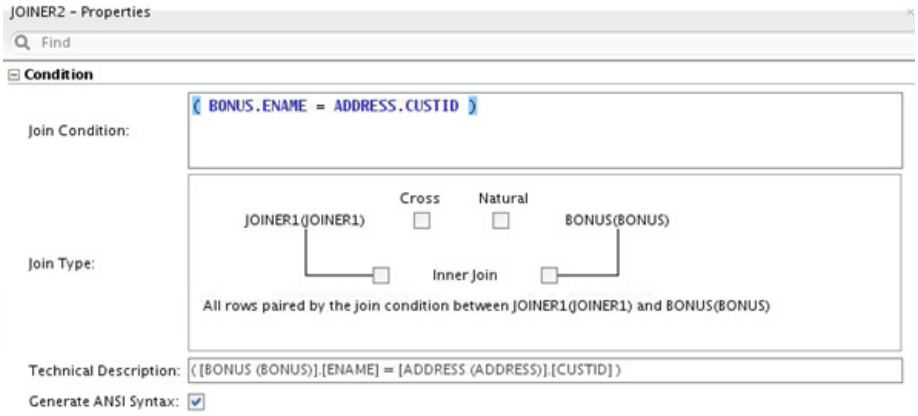
The properties of JOINER would be:



The properties of JOINER1 would be:



The properties of JOINER2 would be:



The properties of Filter_JOINER would be:

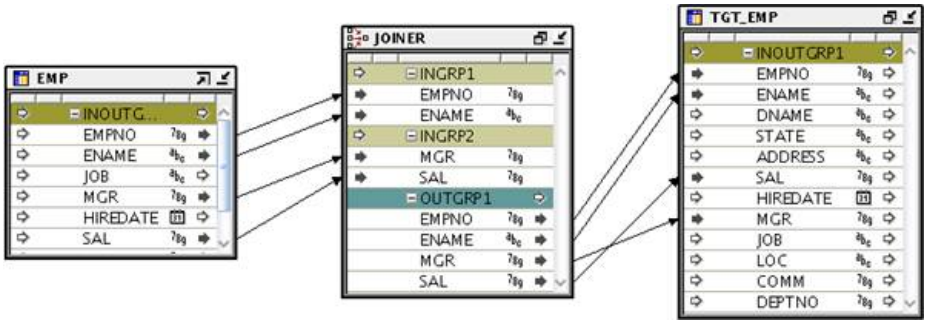
Migrating a Non-ANSI Join Operator

When the property ANSI SQL syntax of OWB mapping is set to false, the OWB Join operator will be migrated to one ODI Join component.

Exception: when "ANSI SQL syntax" is set to false, but "Join input Role" is set for some of the Join operator attribute groups. The OWB Join operator may be split into binary joins as described in "Join Input Role" on page C-8.

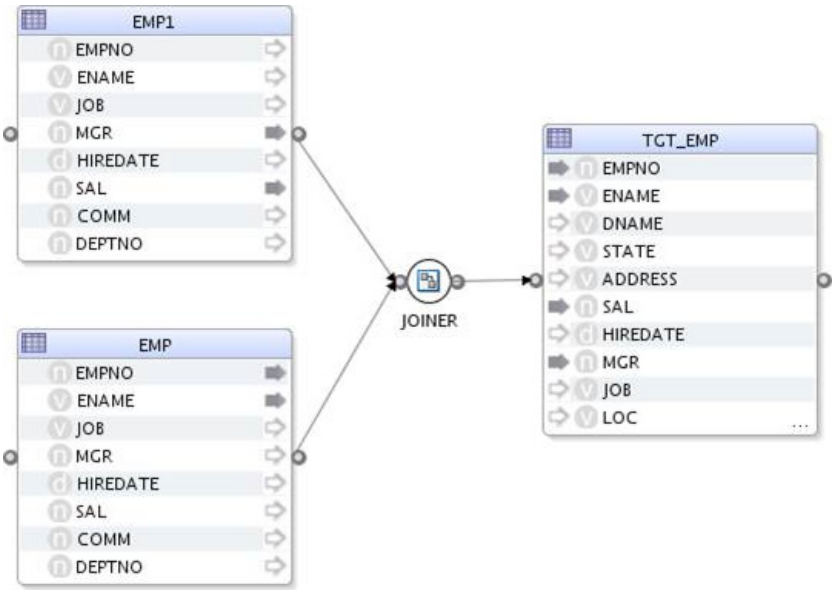
Migrating a Self Join

The following mapping is allowed in OWB, but it is not well supported in ODI 12.1.2.

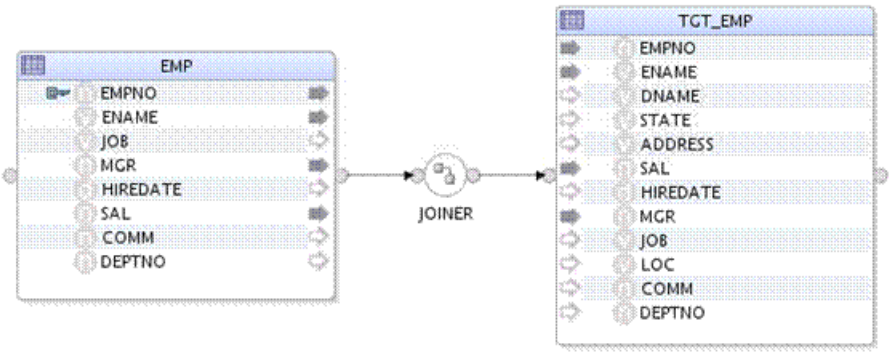


One source table operator is connected to two input groups of the Join operator.

To support this mapping in ODI 12.1.2, the source table operator is migrated twice, producing an ODI mapping like:



ODI 12.1.3 can support the self join just as the way OWB does, so there is no need to migrate the source table operator twice, and the mapping is migrated to ODI 12.1.3 as below:



EMP component is connected to JOINER component twice by 2 input connector points of JOINER component.

Lookup Operator

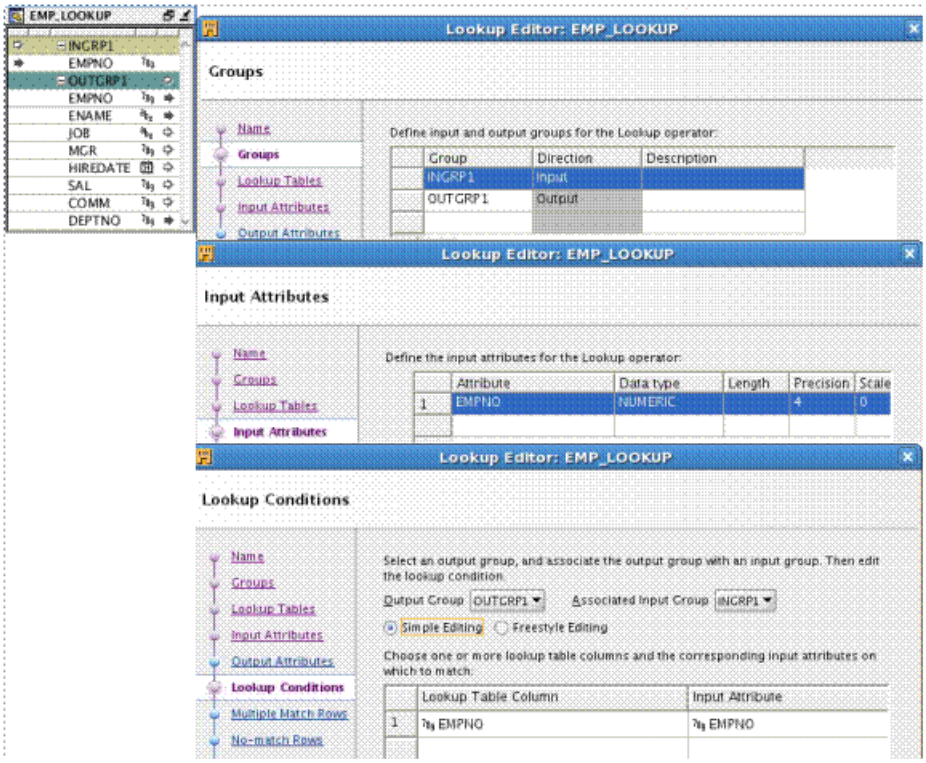
The OWB Lookup operator is not migrated to ODI directly. Instead, each of its input/output attribute group pairs is migrated to an ODI Lookup component.

If the OWB Lookup operator has multiple input/output attribute group pairs, the resulting ODI Lookup components are chained together as a binary tree.

Properties of the input/output attribute group pairs are migrated to properties of the ODI Lookup components.

The OWB in group and input attributes will be omitted after the lookup condition converted to ODI.

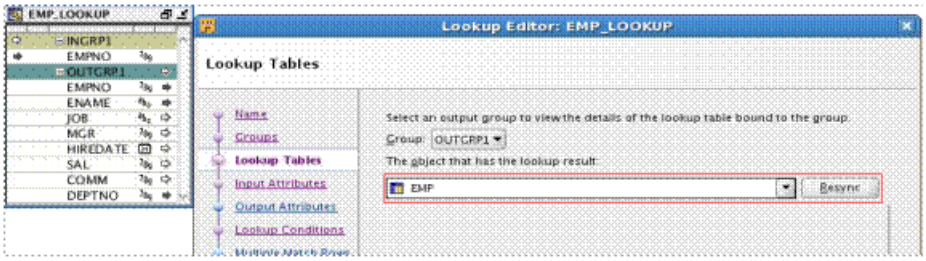
<OWB In Group, Attributes, and Lookup Conditions>



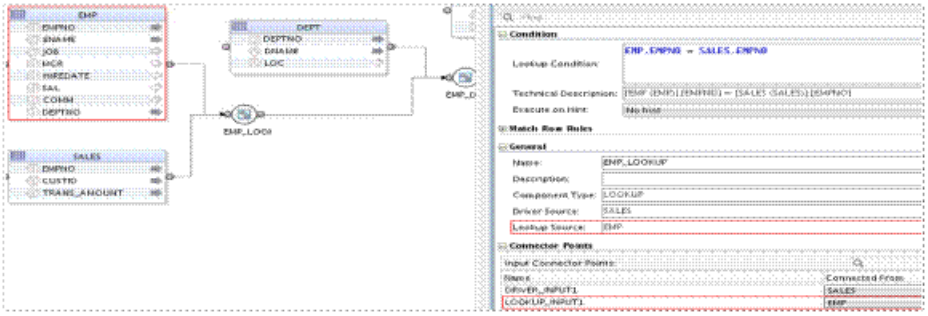
<ODI Lookup Condition>

The OWB Lookup Table is migrated as ODI Lookup Operator's <Lookup Source> and show up in the mapping.

<OWB Lookup Table>

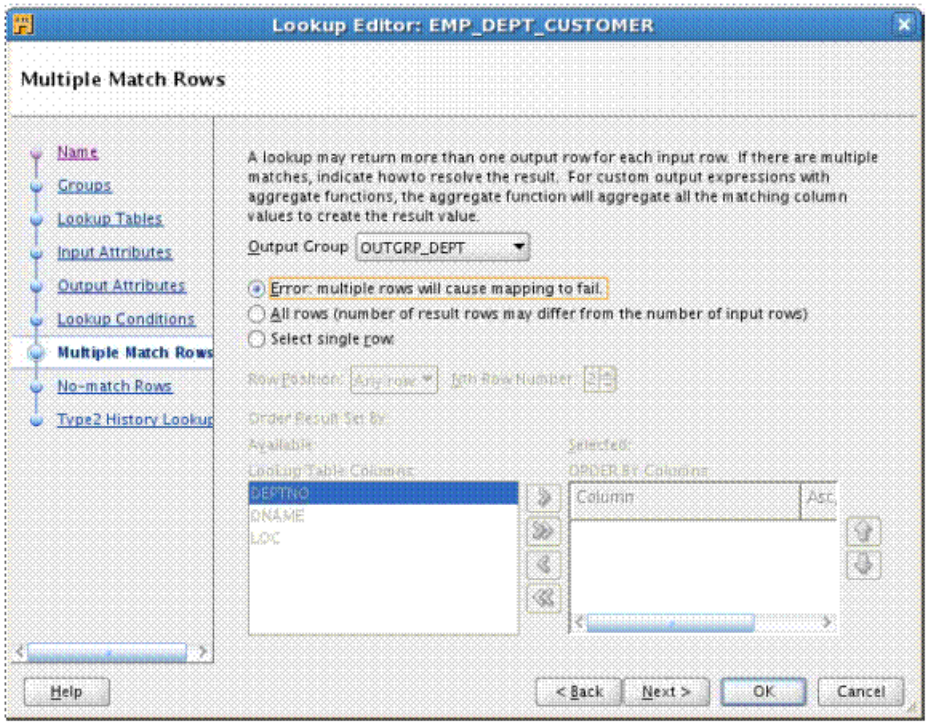


<ODI Lookup Source>

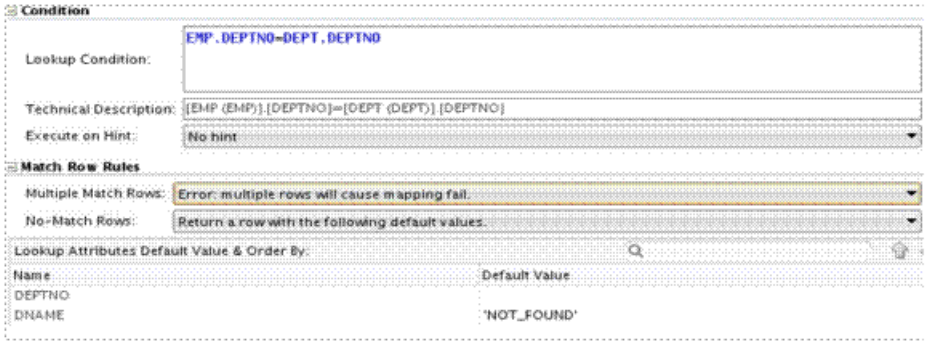


The OWB Multiple Match Rows Rules are migrated to ODI's <Multiple Match Rows>, <Nth Row Number> and <Lookup Attributes default value & order by> - Column <order by>

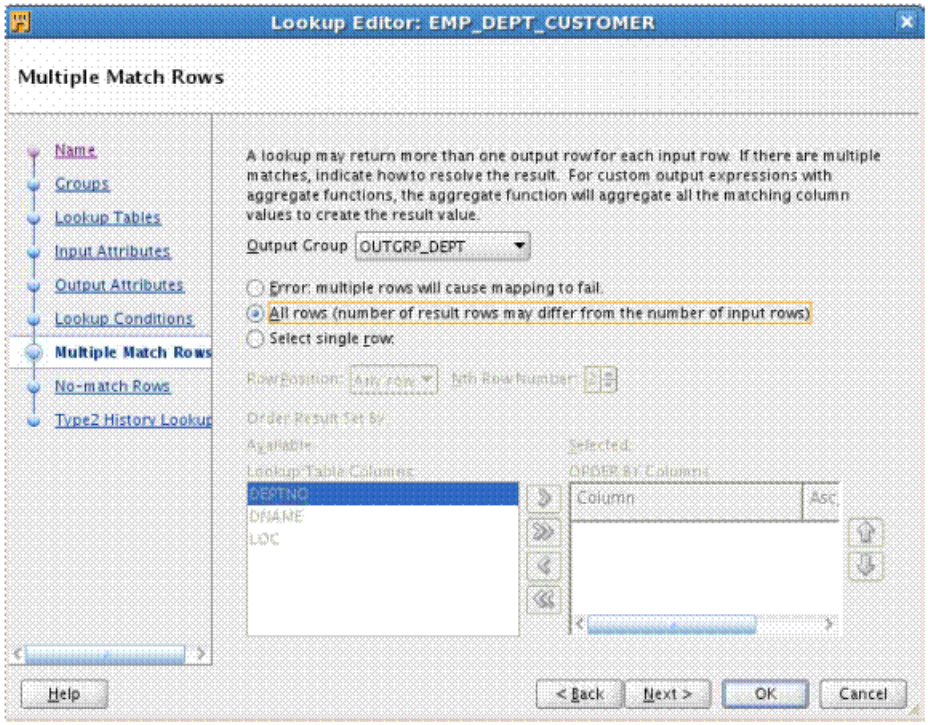
<OWB Multiple Match Rows – Error>



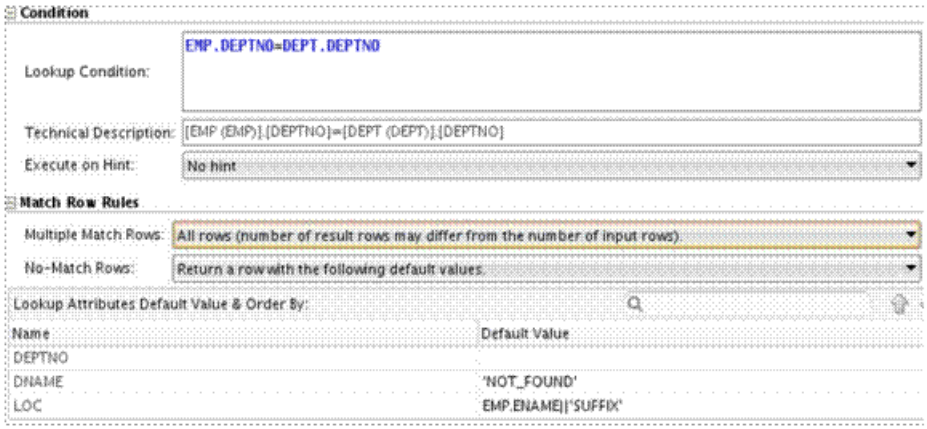
<ODI Multiple Match Rows – Error>



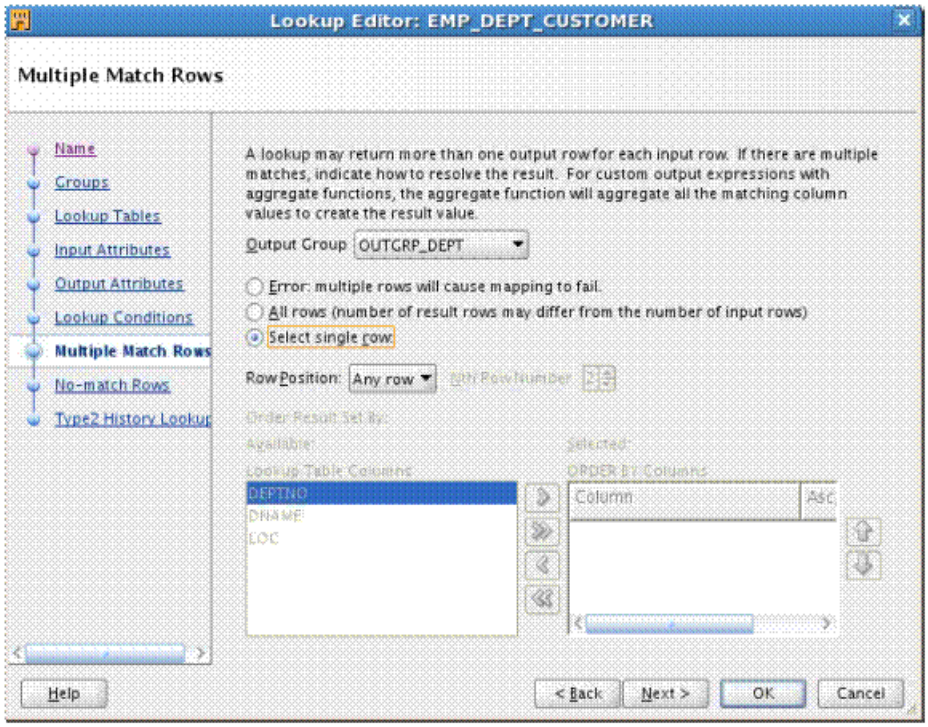
<OWB Multiple Match Rows - All Rows>



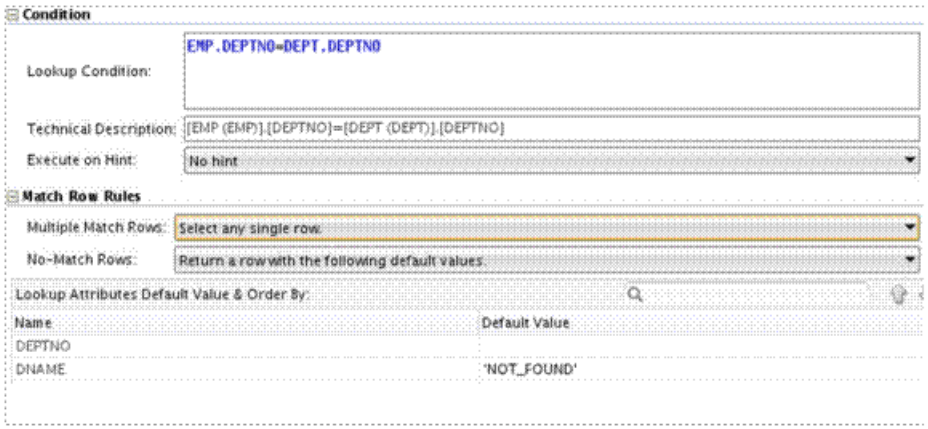
<ODI Multiple Match Rows - All Rows >



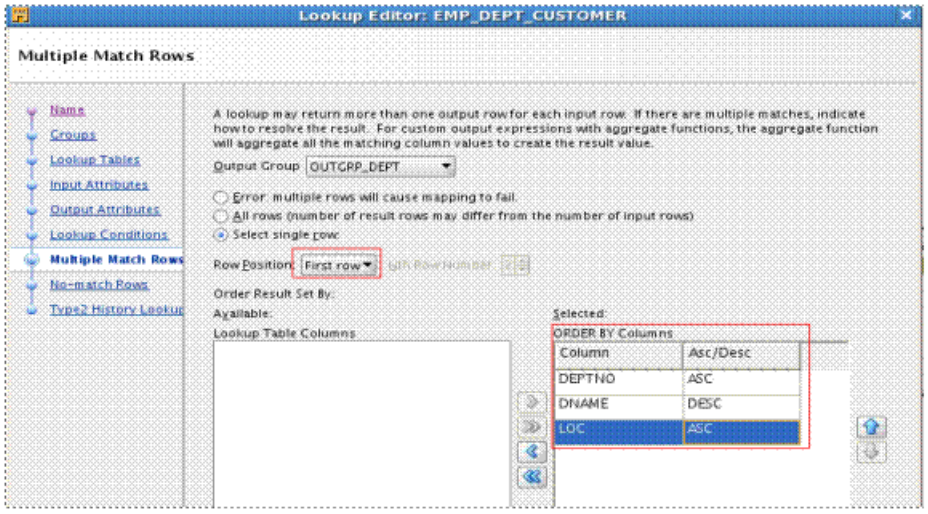
<OWB Multiple Match Rows - Single Row - Any Row>



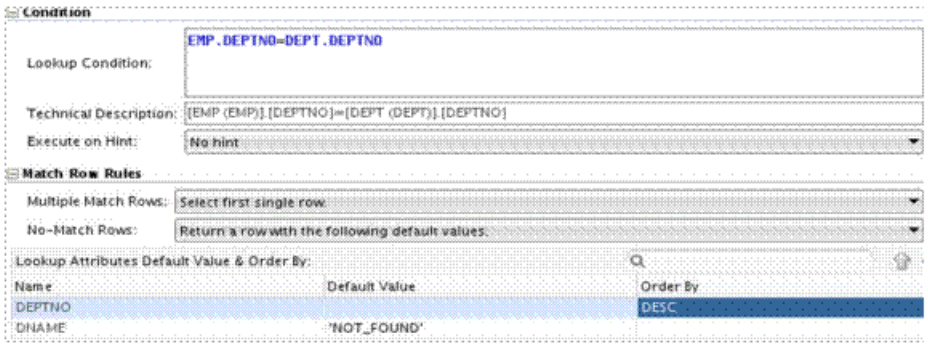
<ODI Multiple Match Rows - Select Any Single Row>



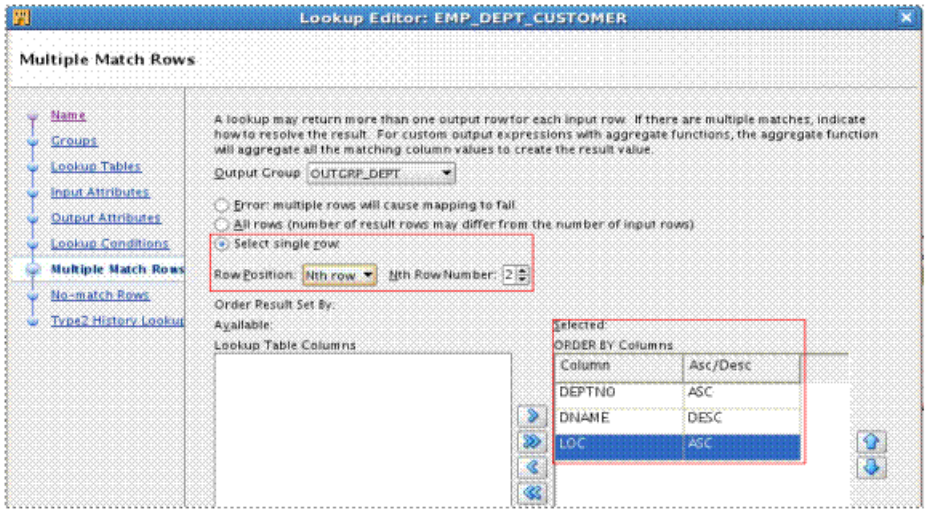
<OWB Multiple Match Rows - Single Row - First / Last Row>



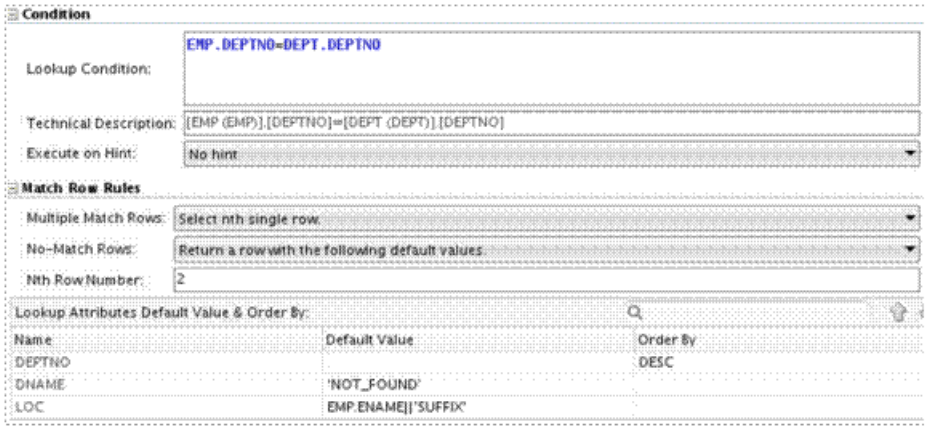
<ODI Multiple Match Rows -Select Single First / Last row>



<OWB Multiple Match Rows - Single Row - Nth Row>

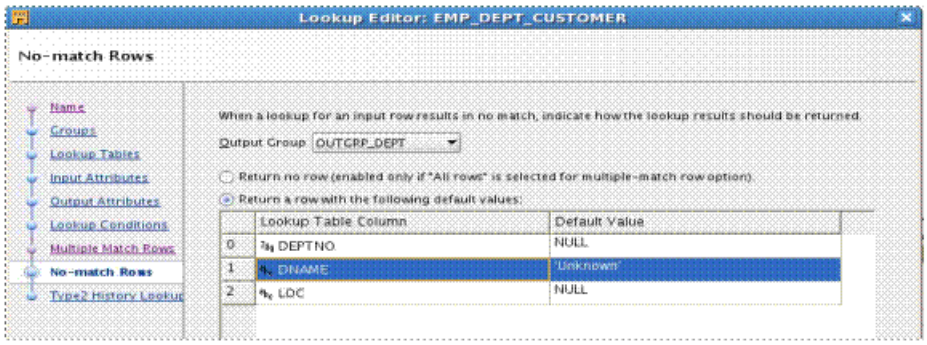


<ODI Multiple Match Rows - Select Single Nth Row>



OWB No Match Rows Rules are migrated to ODI No Match Rows Plus <Lookup Attributes Default Value & Order By> Default Value.

<OWB Match No Rows - Using Predefined Value>



<ODI Match No Rows - Using Default Value>

Lookup Attributes Default Value & Order By:

Name	Default Value	Order By
DEPTNO		DESC
DNAME	'NOT_FOUND'	
LOC	EMP.ENAME||'SUFFIX'	

Migration Path

To support OWB lookup migration, the concept Multiple Match Rows and No match Rows are introduced into ODI. The migration combines as following:

	OWB	ODI12.1.2	ODI12.1.3		
Multiple Match Rows	No Match Rows	Lookup Type	Multiple Match Rows	No Match Rows	Code Generated
ALL ROWS	DEFAULT VALUES	LEFT OUTER	ALL ROWS(LEFT OUTER)	DEFAULT VALUES	LEFT OUTER JOIN
ALL ROWS	NO ROW	N/A	ALL_ROWS (ALL_ROWS)	NO ROW	INNER JOIN
ERROR	DEFAULT VALUES	EXPRESSION IN SELECT	ERROR (ERROR_ WHEN_ MULTIPLE_ ROW)	DEFAULT VALUES	EXPRESSION IN SELECT
NTH ROW	DEFAULT VALUES	N/A	NTH ROW	DEFAULT VALUES	LEFT OUTER JOIN
ANY ROW	DEFAULT VALUES	N/A	ANY ROW	DEFAULT VALUES	EXPRESSION IN SELECT
FIRST ROW	DEFAULT VALUES	N/A	FIRST ROW	DEFAULT VALUES	EXPRESSION IN SELECT
LAST ROW	DEFAULT VALUES	N/A	LAST ROW	DEFAULT VALUES	EXPRESSION IN SELECT

Lookup Properties Migration

OWB Property Name	ODI Property Name
Name	Name
Input Group	Not Migrated
Input Attributes	Not Migrated
Multiple Match Rows	Multiple Match Rows
No-Match Row	No-Match Row
Nth Row Number	Nth Row Number
Default Value and Order By	Lookup Default Values & Order By
Lookup Condition	Lookup Condition
Each Group's Lookup Table	Lookup Operator's Lookup Table

Mapping Input Parameter Operator

Each attribute of an OWB Mapping Input Parameter operator is migrated as one ODI variable under the project tree panel.

The default value of an attribute in the OWB Mapping Input Parameter operator is migrated as the default value of the ODI variable. If the default value is not set, the expression of the attribute is used instead.

By default, the attribute name is migrated to the ODI variable name. If the name already exists, a number is automatically appended to create a unique name. If multiple attributes of the same name are migrated, increasing numbers are used to create unique names.

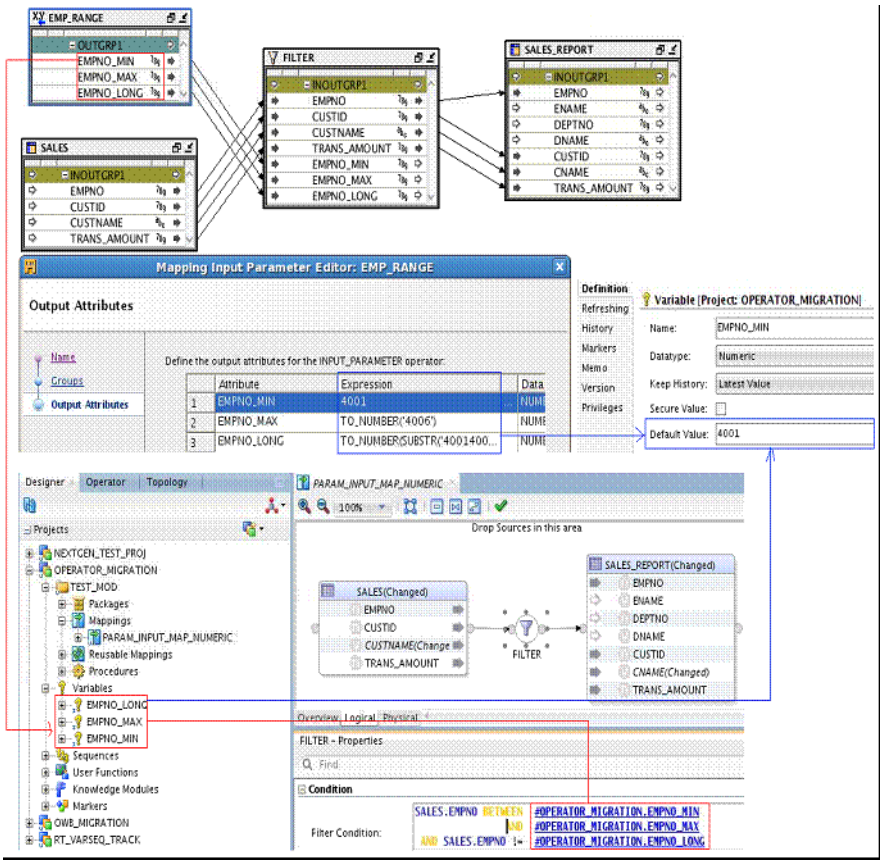
Properties of the Attributes of the Mapping Input Parameter Operator

OWB Property Name	ODI Property Name	Description
Physical Name	Name	If the name already exists, a number is automatically appended to create a unique name.
Default Value	Default Value	The default value of the attribute in the OWB Mapping Input Parameter will be migrated as the ODI Variable's default. If the default value of the attribute in the input parameter is not set, use the expression of the attribute instead.
Data Type; one of: ■ TIMESTAMP ■ TIMESTAMP_WITH_LOCAL_TIME_ZONE ■ TIMESTAMP_WITH_TIME_ZONE ■ DATE	Data Type: DATE	The attribute's default value (or expression if no default value is set) is converted to text and the ODI variable's data type is configured as SHORT_TEXT or LONG_TEXT: ■ If the length of the converted text exceeds 250, the ODI variable's data type is configured as LONG_TEXT. ■ Otherwise, the ODI variable's data type is configured as SHORT_TEXT.
Data Type; one of: ■ NUMBER ■ NUMERIC ■ FLOAT ■ BINARY_DOUBLE ■ BINARY_FLOAT ■ INTEGER	Data Type: NUMERIC	If the attribute's default value (or expression if no default value is set) cannot be parsed to numeric, it is converted to text and the ODI variable's data type is configured as SHORT_TEXT or LONG_TEXT: ■ If the length of the converted text exceeds 250, the ODI variable's data type is configured as LONG_TEXT. ■ Otherwise, the ODI variable's data type is configured as SHORT_TEXT.
Data Type; one of: ■ VARCHAR2 ■ VARCHAR ■ CHAR ■ NCHAR ■ NVARCHAR2	Data Type: SHORT_TEXT	If the length of attribute's default value (or expression if no default value is set) exceeds 250, the ODI variable's data type is configured as LONG_TEXT.
Data Type: Other	Not Supported	If the attribute's type in OWB is some other type, the operator will not be migrated.

Migration Logic

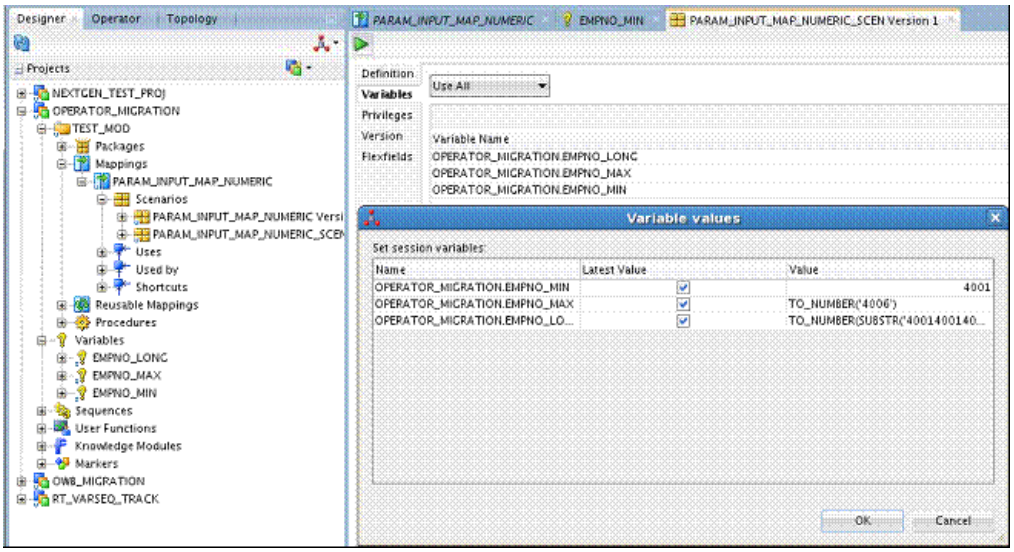
The following diagram provides an example of how the OWB Mapping Input Parameter is migrated to ODI. In this diagram, note the following:

1. Each attribute inside the Mapping Input Parameter EMP_RANGE is migrated to a standalone variable; for example, EMP_RANGE.EMPNO_MIN is migrated to the ODI project variable EMPNO_MIN.
2. The attribute's default value or expression is migrated to the ODI variable's default value; for example, the expression 4001 of EMP_RANGE.EMPNO_MIN in OWB is migrated to the ODI variable EMPNO_MIN's default value of 4001.
3. The downstream expressions of OWB Mapping Input Parameter attributes are parsed to use the variable; for example, the FILTER condition expression has been converted to #OPERATOR_MIGRATION.EMPNO_MIN.



How the Default Value Is Used

Once a mapping that contains an Input Parameter operator been migrated to ODI, it can be executed through the generated mapping scenario. During the execution, all the ODI variables migrated from OWB will be populated with the default value (OWB input parameter attribute's default value or expression). If necessary, you can change the value as needed, as shown in the following figure:



Materialized View Operator

The OWB Materialized View operator is migrated to the ODI Data store component.

Logical Properties of the Materialized View Operator

General Properties

OWB Property Name	Description	ODI Property Name	Note
Bound Name (BOUND_NAME)			If the OWB Materialized View operator is bound to a materialized view, the ODI Data store component will be bound to the corresponding data store.
Primary Source (PRIMARY_SOURCE)	A boolean value to indicate whether this is a primary source (only used in EDW). (YES/NO)		Not migrated.
Keys (KEYS_READONLY)			Not migrated.

OWB Property Name	Description	ODI Property Name	Note
Loading Type (LOADING_TYPE)	Choices = "INSERT, UPDATE, INSERT_ UPDATE, UPDATE_ INSERT, DELETE, NONE, TRUNCATE_ INSERT, DELETE_ INSERT, CHECK_ INSERT, DERIVE_ FROM_LCR"	INTEGRATION_ TYPE	Same as for the Table operator. See Notes About Loading Type.
Target Load Order (TARGET_LOAD_ORDER)	Map targets names in loading sequence.		Not migrated.

Chunking

As with the Table operator, properties for Chunking are not migrated.

Conditional Loading

Same as for the Table operator. See "Conditional Loading" on page C-37.

Data Rules

As with the Table operator, properties for Data Rules are not migrated.

Error Table

As with the Table operator, properties for Error Table are not migrated.

SCD Updates

As with the Table operator, properties for SCD Updates are not migrated.

Temp Stage Table

As with the Table operator, properties for Temp Stage Table are not migrated.

Physical Properties of the Materialized View Operator

Same as for the Table operator. See "Physical Properties of the Table Operator" on page C-39.

Logical Properties of the Attributes of the Materialized View Operator

Same as for the Table operator. See "Logical Properties of the Attributes of the Table Operator" on page C-39.

Migrating an Unbound Materialized View Operator

Same as for the Table operator. See "Migrating an Unbound Table Operator" on page C-41.

Pivot Operator

The OWB Pivot operator is migrated to the ODI Unpivot component.

Properties of the Pivot Operator

General Properties

OWB Property Name	Description	ODI Property Name	Note
Business Name (LOGICAL_NAME)		Business Name (BUSINESS_NAME)	
Physical Name (NAME)		Name (NAME)	If the OWB name includes the string "pivot", it is changed to "unpivot".

Row Locator

The output attribute that is set as the row locator of the OWB Pivot operator is migrated to the value of the Row Locator property of the ODI Unpivot component.

Pivot Transform

Pivot transform values of the OWB Pivot operator are migrated to unpivot transform values of the ODI Unpivot component.

Map Attribute Group and Map Attribute

Map attribute groups of the OWB Pivot operator are migrated to connector points of the ODI Unpivot component. No specific properties for attribute group of Pivot operator need to be migrated.

Input attributes of the OWB Pivot operator are not migrated.

Output attributes are migrated. Name, Data Type, Length, Precision, Scale, Second Precision and Description are general properties described in "Mapping Attribute" on page B-24. Migration of the Row Locator property is described in "Row Locator". The Expression property of the OWB Output attribute is migrated to the Expression property of the ODI Output attribute.

Pluggable Mapping Operator

The OWB Pluggable Mapping operator is migrated to the ODI Reusable Mapping component.

For general information about migrating pluggable mappings, see OWB Pluggable Mappings.

Properties of the Pluggable Mapping Operator

No specific properties of the Pluggable Mapping operator need to be migrated.

Attribute Groups and Attributes of the Pluggable Mapping Operator

Attribute groups and attributes in the Pluggable Mapping operator are not migrated.

In ODI, when a Reusable Mapping component is bound to a Reusable Mapping, the connector points and attributes of the Reusable Mapping component are created automatically according to the binding Reusable Mapping. Thus, if an OWB Pluggable Mapping operator is not consistent with its bound object in OWB, migration issues

might arise. To avoid any such issues, synchronize the Pluggable Mapping operator before migration.

Migrating an Unbound Pluggable Mapping Operator

A mapping containing an unbound Pluggable Mapping operator will not be migrated unless the MIGRATE_UNBOUND_OPERATOR migration configuration option is set to true.

During migration, a Reusable Mapping will be created in ODI based on the unbound Pluggable Mapping operator. The created Reusable Mapping is placed in the STAND_ALONE folder under the project where the mapping is placed. The unbound Pluggable Mapping operator is migrated to a Reusable Mapping component and bound to the newly created Reusable Mapping.

Post-Mapping Operator

For ODI 12.1.2 (plus the applied patch), the OWB Post-Mapping operator is converted to PL/SQL code and configured into the ODI container mapping's target node as the KM option END_MAPPING_SQL.

For ODI 12.1.3, the OWB Post-Mapping operator is migrated as SQL clause and saved into the "End Mapping Command" of the mapping. The operator's location information is migrated into Location for End Mapping Command, and the Technology for End Mapping Command would be populated as Oracle.

The downstream expressions which refer to the Output Attribute are resolved as NULL.

The data type of Attribute Process operator are limited to: `TIMESTAMP, TIMESTAMP_WITH_LOCAL_TIME_ZONE, TIMESTAMP_WITH_TIME_ZONE, DATE, NUMBER, NUMERIC, FLOAT, BINARY_DOUBLE, BINARY_FLOAT, INTEGER, VARCHAR2, VARCHAR, CHAR, NCHAR, NVARCHAR2`. Otherwise, the Attribute Process Operator is not migrated.

OWB Property Name	ODI Property Name	Note
Business Name		Not migrated.
Description		Not migrated.
Function Name	Function Name inside the End Mapping Command.	
Physical Name		Not migrated.
Post-Mapping Process Run Condition		Skipped after migrated to ODI Mapping.
Row based only		Not migrated.
Input Attribute Physical Name		Not migrated.
Output Attribute Physical Name		Not migrated.

Pre-Mapping Operator

For ODI 12.1.2 (plus the applied patch), the OWB Pre-Mapping operator is migrated to the KM option BEGIN_MAPPING_SQL of the source ODI Datastore component.

For ODI 12.1.3, the OWB Pre-Mapping operator is migrated as a SQL clause and saved into the "Begin Mapping Command" of the mapping. The operator's location information would be migrated into Location for Begin Mapping Command, and the Technology for Begin Mapping Command would be populated as Oracle.

The downstream expressions which refer to the Output Attribute are be resolved as NULL.

The data type of Attribute Process operator are limited to: `TIMESTAMP, TIMESTAMP_WITH_LOCAL_TIME_ZONE, TIMESTAMP_WITH_TIME_ZONE, DATE, NUMBER, NUMERIC, FLOAT, BINARY_DOUBLE, BINARY_FLOAT, INTEGER, VARCHAR2, VARCHAR, CHAR, NCHAR, NVARCHAR2`. Otherwise, the Attribute Process Operator is not migrated.

OWB Property Name	ODI Property Name	Note
Business Name		Not migrated.
Description		Not migrated.
Function Name	Function Name inside the Begin Mapping Command.	
Physical Name		Not migrated.
Post-Mapping Process Run Condition		Skipped after migrated to ODI Mapping.
Row based only		Not migrated.
Input Attribute Physical Name		Not migrated.
Output Attribute Physical Name		Not migrated.

Sequence Operator

OWB Sequences are migrated to ODI Sequences as described in "OWB Sequence to ODI Sequence" on page B-13. The OWB Sequence operator is not migrated; however, references to OWB Sequences in expressions are migrated to ODI as part of the migration of the expressions.

Set Operator

The OWB Set operator is migrated to the ODI Set component.

Properties of the Set Operator

Set Operation

Set operation is an operator level property in OWB. It has four choices: UNION, UNIONALL, INTERSECT, and MINUS.

ODI has a similar property, but the property is set on the input connector point. Hence, the operator-level OWB Set Operation property is migrated to each input connector point of the Set ODI component except the first input connector point which is left as empty.

The following table displays the migration from OWB Set Operation to ODI set operation type.

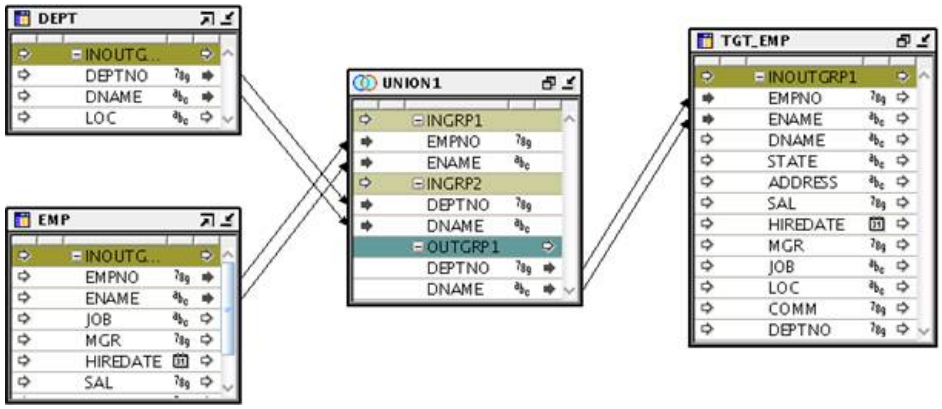
OWB Set Operation Type	ODI Set Operation Type
UNION	UNION
UNIONALL	UNION ALL
MINUS	MINUS
INTERSECT	INTERSECT

Attribute Groups and Attributes of the Set Operator

The operator attribute groups of the OWB Set operator are migrated to ODI component connector points. No specific properties need to be migrated for attribute groups of the Set operator.

Input attributes of the Set operator are not migrated.

Output attributes are migrated. The Output attribute of the ODI Set component can have multiple expressions. Each expression is associated with an input connector point. During migration, the expressions for the ODI attribute will be constructed according to the input attributes of the OWB Set operator. Take the following mapping as an example:



Union1 is a Set operator in OWB. It has two output attributes, and the two input attributes INGRP1.EMPNO and INGPR2.DEPTNO are mapped to OUTGRP1.DEPTNO.

Because INGRP1.EMPNO is connected from EMP.INOUTGRP.EMPNO and INGRP2.DEPTNO is connected from DEPT.INOUTGRP.DEPTNO, the expressions for the output attribute UNION1.DEPTNO in the ODI Set component are set to refer to EMP.EMPNO and DEPT.DEPTNO.

Sorter Operator

The OWB Sorter operator is migrated to the ODI Sorter component.

Logical Properties of the Sorter Operator

OWB Property Name	Description	ODI Property Name	Note
Order By Clause (ORDER_BY_CLAUSE)	The Order By Clause	ORDER_BY_CLAUSE	

Physical Properties of the Sorter Operator

OWB Property Name	Description	ODI Property Name	Note
Inline view hint (INLINEVIEW_HINT)	Hint used when inline view is created for this operator		Not migrated.

Splitter Operator

The OWB Splitter operator is migrated to the ODI Splitter component.

Properties of the Splitter Operator

Split Condition

Split Condition is an attribute group-level property in OWB. ODI has a similar property, which is set on the output connector point. The Split Condition property on the output attribute group in OWB is migrated to the split condition expression on the output connector point in ODI.

Attribute Groups and Attributes of the Splitter Operator

Output attribute groups of the Splitter operator in OWB are migrated to output connector points in ODI. The output attribute group with the name REMAINING_ROWS in OWB is migrated to the Remainder output connector point in ODI.

Attributes of the Splitter operator are not migrated.

Subquery Filter Operator

The OWB Subquery Filter operator is migrated to the ODI Subquery Filter component.

Properties of the Subquery Filter Operator

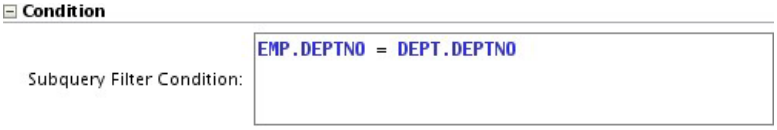
Name and Description

The physical name of the Subquery Filter operator is migrated to the Subquery Filter component name. The description is migrated to the component description.

Subquery Filter Condition

The OWB subquery filter condition is mapped to the ODI subquery filter condition.

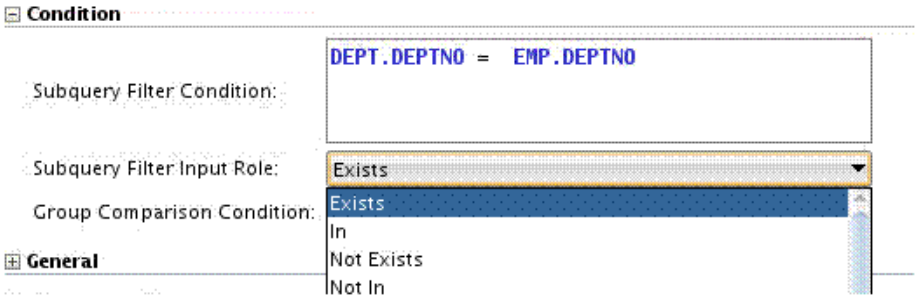
The subquery filter condition for the ODI Subquery Filter component is as follows:



Subquery Filter Input Role

The OWB subquery filter input role is migrated to the ODI subquery filter input role.

The subquery filter input role for the ODI Subquery Filter component is as follows:



Map Attribute Groups

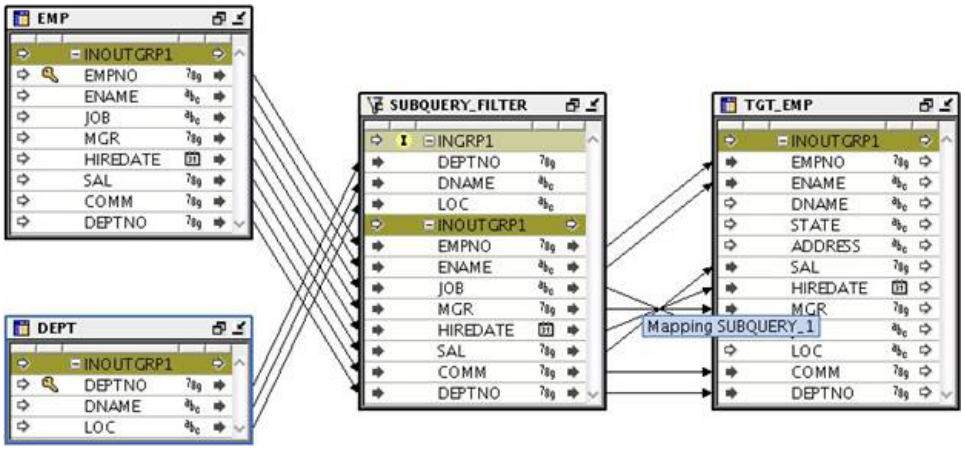
The OWB Subquery Filter operator has two attribute groups: input attribute group and inout attribute group. The input attribute group of the OWB Subquery Filter operator is migrated to the ODI SUBQUERY_FILTER_INPUT connector point of the ODI Subquery Filter component. The OWB inout attribute group of the Subquery Filter operator is migrated to the ODI DRIVER_INPUT connector point and the output connector point. The two connector points use the default name instead of the OWB inout attribute group name.

Attributes

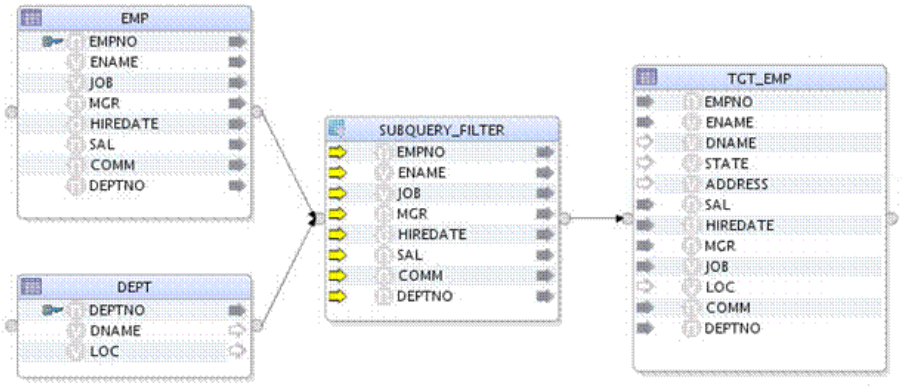
Attributes in the input attribute group are not migrated. Attributes in the inout group of the OWB Subquery Filter operator are migrated to output attributes of the ODI Subquery Filter component.

An output attribute of the Subquery Filter component has two expressions. The following example describes how these two expressions are set during migration.

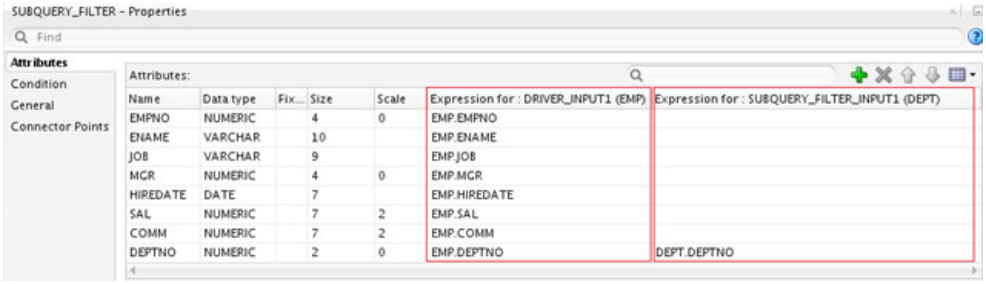
Using the following OWB mapping as an example:



This OWB mapping is migrated to the following ODI mapping:



The expressions for each migrated attribute are as follows:



Expression for DRIVER_INPUT Connector Point

For output attributes of the OWB Subquery Filter operator that are connected from an upstream attribute, the expression of these output attributes is set to the ODI DRIVER_INPUT connector point, and the expression references the upstream projector attribute.

In the previous OWB mapping, the attribute SUBQUERY_FILTER.INOUTGRP1.EMPNO is connected from EMP.EMPNO. After migration to ODI, the expression set on the DRIVER_INPUT connector point is EMP.EMPNO.

Expression for SUBQUERY_FILTER_INPUT Connector Point

For output attributes of the OWB Subquery Filter operator with an IN Matching Attribute property set, the expression of this property is set to the SUBQUERY_FILTER_INPUT connector point.

For example, if the IN Matching Attribute value is DEPTNO, when migrating to ODI, DEPT.DEPTNO is set as the expression for the SUBQUERY_FILTER_INPUT connector point in ODI.

Table Operator

The OWB Table operator is migrated to the ODI Datastore component.

Logical Properties of the Table Operator

General Properties

OWB Property Name	Description	ODI Property Name	Note
Bound Name (BOUND_NAME)			If the OWB Table operator is bound to a table, the ODI Datastore component will be bound with the corresponding data store.
Business Name (LOGICAL_NAME)		Business Name (BUSINESS_NAME)	
Create By			Not migrated.
Create Time			Not migrated.
Icon Object			Not migrated.
Keys (KEYS_READONLY)			Not migrated.
Last Update Time			Not migrated.
Primary Source (PRIMARY_SOURCE)	A boolean value to indicate whether this is a primary source (only used in EDW).		Not migrated.
Loading Type (LOADING_TYPE)	Choices = "INSERT, UPDATE, INSERT_UPDATE, UPDATE_INSERT, DELETE, NONE, TRUNCATE_INSERT, DELETE_INSERT, CHECK_INSERT, DERIVE_FROM_LCR"	INTEGRATION_TYPE	See Notes About Loading Type.
Target Load Order (TARGET_LOAD_ORDER)	Map targets names in loading sequence.		Not migrated. The TARGET_LOAD_ORDER property will be specified on the mapping level.
Update By			Not migrated.

Notes About Loading Type

The loading type of the OWB operator is migrated to the ODI integration type. The Loading Type property is migrated only when the operator is used as the target.

The following table displays the migration mappings from the OWB loading type to the ODI integration type.

OWB Loading Type	Description	ODI Integration Type	Note
INSERT		CONTROL_APPEND	A default IKM whose integration type is CONTROL_APPEND is assigned.
UPDATE		INCREMENTAL_UPDATE	A default IKM whose integration type is INCREMENTAL_UPDATE and subtype is UPDATE is assigned.
INSERT_UPDATE		INCREMENTAL_UPDATE	A default IKM whose integration type is INCREMENTAL_UPDATE and subtype is MERGE is assigned.
UPDATE_INSERT		INCREMENTAL_UPDATE	A default IKM whose integration type is INCREMENTAL_UPDATE and subtype is MERGE is assigned.
DELETE		Integration type is not set; a default integration type is used.	ODI does not support DELETE DML.
NONE		Integration type is not set; a default integration type is used.	
TRUNCATE_INSERT		CONTROL_APPEND	Similar to INSERT, and the KM option TRUNCATE_TARGET_TABLE (if it exists) is set to true.
DELETE_INSERT		CONTROL_APPEND	Similar to INSERT, and the KM option DELETE_ALL (if it exists) is set to true.
CHECK_INSERT		CONTROL_APPEND	Treated the same as INSERT. Note that there is no KM option to check whether the target table is empty prior to the insert action.
DERIVE_FROM_LCR		Integration type is not set; a default integration type is used.	ODI does not support DERIVE_FROM_LCR.

Change Data Capture

The following table displays the Change Data Capture (CDC) property mappings from OWB to ODI.

OWB Property Name	Description	ODI Property Name	Note
Enabled (IS_CDC)	Indicates if journaling is enabled for this entity.	Journalized Data Only (JOURNALIZING_ENABLED)	

OWB Property Name	Description	ODI Property Name	Note
Capture Consistency (CDC_METHOD)	Change Data Capture method for this entity. Choices: NONE, CONSISTENT, SIMPLE		Not migrated.
Change Data Capture Filter (CDC_FILTER_CONDITION)	The boolean filtering condition that identifies the data to be processed. Any row with a false condition is not migrated.	Journalized Data Filter (JOURNALIZED_DATA_FILTER)	
Trigger Based Capture (IS_TRIGGER_CDC)	Indicates if journaling triggers are generated for this entity.		Not migrated.

Chunking

Not migrated.

Conditional Loading

The following table displays the Conditional Loading property mappings from the OWB Table operator to the ODI Datastore component.

OWB Property Name	Description	ODI Property Name	Note
Target Filter for Update (TARGET_FILTER_FOR_UPDATE)	A condition on the rows in the target and if evaluated to true, that row participates in the update loading operation.		Not migrated.
Target Filter for Delete (TARGET_FILTER_FOR_DELETE)	A condition on the rows in the target and if evaluated to true, that row participates in the delete loading operation.		Not migrated.
Match by constraint (MATCH_BY_CONSTRAINT)	Indicates whether unique or primary key information on this target will override the matching criteria obtained from the "Match by constraint" property on the attributes of this target.	Update Key (UPDATE_KEY)	See Notes About Match By Constraint.

Notes About Match By Constraint

In OWB, the property "Match by constraint" can be set to ALL_CONSTRAINTS, NO_CONSTRAINT and a specific CONSTRAINT name (a PK or UK name of the entity).

ALL_CONSTRAINTS

If "Match by constraint" is set to ALL_CONSTRAINTS, no update key is set on the corresponding ODI Datastore component.

NO_CONSTRAINT

If "Match by constraint" is set to NO_CONSTRAINT, no update key is set on the corresponding ODI Datastore component.

Specific Constraint Name

If "Match by constraint" is set to a specific constraint name, the constraint name is used to find the corresponding key (PK or UK) in ODI that will be set as the update key.

Control CT

Migration details for Control CT (code template) mapping properties are as follows:

Primary Key, Foreign Key, Unique Key, Check Constraint

Based on the name of the Key of the OWB Table operator, if a constraint with the same name exists on the corresponding ODI Datastore component, the flow control value in OWB is migrated to the constraint value in ODI.

Not Null Attribute Property

The Not Null property is set on the attribute level. The flow control value of the OWB attribute is migrated to the Check Not Null property value on the ODI attribute.

Data Rules

Data Rules properties are not migrated.

Error Table

Error Table properties are not migrated.

SCD Updates

SCD Updates properties are not migrated.

Temp Stage Table

Temp Stage Table properties are not migrated.

Partition DML

The following table displays the Partition DML property mappings from the OWB Table operator to the ODI Datastore component.

OWB Property Name	Description	ODI Property Name	Note
DML Partition Type (DML_PARTITION_TYPE)	Choices: NONE, PARTITION, SUBPARTITION		Not migrated.
Is Partition Indexed by Name (IS_PARTITION_INDEXED_BY_NAME)	False if partition is indexed by partition key value; otherwise, it's indexed by partition name. (YES/NO)		Not migrated.
DML Partition Name (DML_PARTITION_NAME)		Uses OWB partition type and partition name to find the corresponding partition in ODI.	

OWB Property Name	Description	ODI Property Name	Note
Partition Key Value List (PARTITION_KEY_VALUE_LIST)	The partition key value list to search for the partition.		Not migrated.

Physical Properties of the Table Operator

Only those physical properties in the active configuration are considered for migration.

General Physical Properties

OWB Property Name	Description	ODI Property Name	Note
Conflict Resolution (CONFLICT_RESOLUTION)	Detect and resolve any conflicts that may arise during DML using the LCR APIs. (TRUE/FALSE)		Not migrated.
Optimize Merge (OPTIMIZE_MERGE)	(TRUE/FALSE)		Not migrated.
Schema (SCHEMA)			Not migrated.
Database link (DATABASE_LINK)	Database link used to access this entity during mapping.		Not migrated.
Location (DB_LOCATION)	Location, used to access the referenced entity.		Not migrated.

Hints

OWB Property Name	Description	ODI Property Name	Note
Extraction hint (EXTRACTION_HINT)	Hint used when extracting from this table using SQL.	SELECT_HINT	
Loading hint (LOADING_HINT)	Hint used when loading into this table using SQL.	INSERT_HINT or UPDATE_HINT	
Automatic hints enabled (AUTOMATIC_HINTS_ENABLED)	Automatic hints enabled using SQL.		Not migrated.

Partition Exchange Loading

Properties of Partition Exchange Loading for the Table operator are not migrated.

Logical Properties of the Attributes of the Table Operator

Loading Properties

OWB Property Name	Description	ODI Property Name	Note
Load Column when Inserting Row (LOAD_COLUMN_WHEN_INSERTING_ROW)	A boolean value to indicate whether this attribute will participate in the insert load operation. (YES/NO)	Insert Indicator	
Load when Updating Row Column (LOAD_COLUMN_WHEN_UPDATING_ROW)	A boolean value to indicate whether this attribute will participate in the update load operation. (YES/NO)	Update Indicator	
Match Column when Updating Row (MATCH_COLUMN_WHEN_UPDATING_ROW)	A boolean value to indicate whether this attribute will be used to construct the matching criteria between the incoming data and the existing data on the target during the update load operation. (YES/NO)	Key indicator	See Notes About Match Column When Updating Row.
Match Column when Deleting Row (MATCH_COLUMN_WHEN_DELETING_ROW)	A boolean value to indicate whether this attribute will be used to construct the matching criteria between the incoming data and the existing data on the target during the delete load operation. (YES/NO)		Not migrated.
Update Operation (UPDATE_OPERATION)	The computation to be performed on this attribute between the incoming data and the existing data on the target during the update load operation. Choices = '=, +=, -=, =-, *=, /=, =/, =||, ||='		Not migrated.

Notes About Match Column When Updating Row

Although the property of MATCH_COLUMN_WHEN_UPDATING_ROW in OWB is migrated to KEY_INDICATOR in ODI, several rules govern how the key indicator for the ODI map attribute is set.

When the property "Match by constraint" of the OWB Table operator is set to ALL_CONSTRAINTS, the value set on the property MATCH_COLUMN_WHEN_

UPDATING_ROW is not migrated, and the key indicator is set to true for the ODI attribute whose bound object is referenced by any PK/AK.

When the property "Match by constraint" of the OWB Table operator is set to NO_CONSTRAINT, the key indicator of the ODI attribute is set according to the property MATCH_COLUMN_WHEN_UPDATE_ROW of the OWB attribute. If MATCH_COLUMN_WHEN_UPDATE_ROW is set to YES, the key indicator of the ODI attribute should be set to true.

When the property "Match by Constraint" of the OWB Table operator is set to a specific constraint, an update key is set on the ODI Datastore component. The key indicator of the ODI attributes is set automatically when the update key is set.

Code Template Metadata Tags

OWB Property Name	Description	ODI Property Name	Note
UD1 (CODE_TEMPLATE_USER_DEFINED_1)	A boolean value indicating whether this attribute will be included in code template functions using the UD1 tag. (YES/NO)	UD_1	
UD2 (CODE_TEMPLATE_USER_DEFINED_2)	(YES/NO)	UD_2	
UD3 (CODE_TEMPLATE_USER_DEFINED_3)	(YES/NO)	UD_3	
UD4 (CODE_TEMPLATE_USER_DEFINED_4)	(YES/NO)	UD_4	
UD5 (CODE_TEMPLATE_USER_DEFINED_5)	(YES/NO)	UD_5	
UPD (CODE_TEMPLATE_UPDATE)	A boolean value indicating whether this attribute will be included in code template functions using the UPD tag. (YES/NO)		Not migrated.
SCD (CODE_TEMPLATE_SCD)	Choices = 'SCD_UND, SCD_SK, SCD_NK, SCD_INS, SCD_UPD, SCD_FLAG, SCD_START, SCD_END'		Not migrated.

Migrating an Unbound Table Operator

It is recommended to make all mapping operators in OWB to be bound to the corresponding object in the project tree.

Mappings that contain an unbound Table operator are not migrated, unless the migration configuration option MIGRATE_UNBOUND_OPERATOR in the migration utility configuration file is set to true.

If the migration configuration option MIGRATE_UNBOUND_OPERATOR is set to true, a data store is created in ODI based on the unbound Table operator. The bound name of the unbound Table operator is used as the ODI datastore name. The unbound OWB Table operator is migrated to the ODI Datastore component and is bound to the newly created ODI datastore. For each unbound Table operator in a mapping, a datastore is created, even the unbound Table operators have a same bound name.

No keys are created for the datastore in ODI after the migration. This may cause issues with mapping code generation. Users need to manually fix the datastore before running the mapping.

Table Function Operator

The OWB Table Function operator is migrated to the ODI Table Function component. OWB has a bound Table Function operator (the operator is bound to a table function) and an unbound Table Function operator, and these two kinds of operators are migrated to an unbound Table Function component in ODI. The OWB Table Function operator can have one input attribute group and one output attribute group. The attribute groups of the Table Function operator are migrated to ODI map connector points.

Logical Properties of the Table Function Operator

OWB Property Name	Description	ODI Property Name	Note
Table Function Name (TABLE_FUNCTION_NAME)	Name of the table function to be called.	FUNCTION_NAME	
Table Function is Target TABLE_FUNCTION_IS_TARGET	Indicates if this table function is being used as a target operator.		Not migrated. Even without this property, ODI still knows if this Table Function component is used as a target.
Bound Name (BOUND_NAME)	The name to be used by the code generator to identify this operator. By default, this is the same as the operator's physical name.		Not migrated.

Logical Properties of the Map Attribute Group of the Table Function Operator

OWB Property Name	Description	ODI Property Name	Note
Return Table of Scalar (RETURN_TABLE_OF_SCALAR)	Specifies whether the return of the table function is a TABLE of SCALAR.		Not migrated. If this property is set to true in OWB, then the expression of the output attribute in ODI is set to TABLE_FUNCTION_NAME.COLUMN_VALUE.

Logical Properties of the Map Attribute of the Table Function Operator

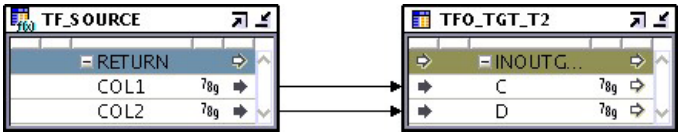
OWB Property Name	Description	ODI Property Name	Note
Bound Name (BOUND_NAME)	The name to be used by the code generator to identify this item. By default, this is the same physical name as the item.		Not migrated.
Type Attribute Name (TYPE_ATTRIBUTE_NAME)	The name of the field of the PLS Record or attribute of the Object Type or column of the ROWTYPE that corresponds to this attribute. This property is not applicable if the return type is TABLE of SCALAR.		Contributes to the expression of the output attribute in ODI.

Migrating the Table Function Operator

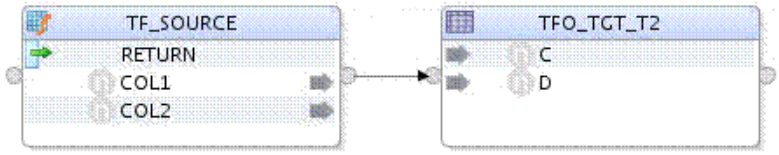
Scenarios for the Table Function operator in OWB mappings are as follows.

Scenario 1: Table Function operator acts as source, no input map attribute group, only return group (output attribute group).

OWB mapping:



Mapping in ODI after migration:

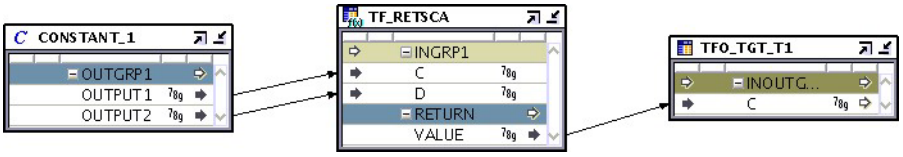


The OWB output attribute group RETURN is migrated to the output connector point RETURN in ODI.

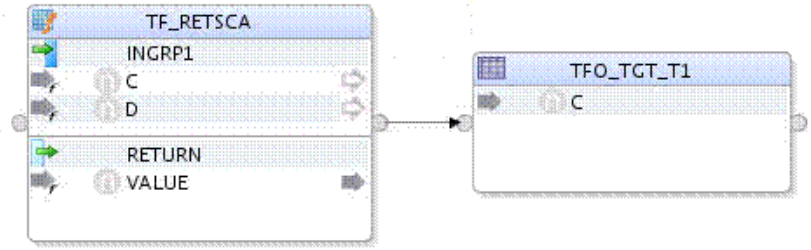
OWB output attributes in the group RETURN are migrated to output attributes in the connector point RETURN in ODI.

Scenario 2: Table Function Operator has one input attribute group and one output attribute group, data type of input attributes is scalar

OWB mapping:



Mapping in ODI after migration:



The operator CONSTANT_1 is not migrated. The expressions on its attributes are migrated to the ODI Table Function component attribute.

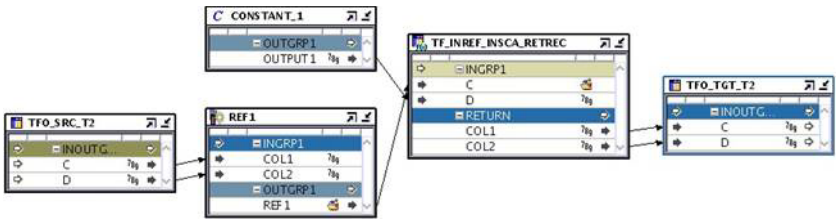
The OWB input attribute group INGRP1 of the Table Function operator is migrated to the input connector point INGRP1 in the ODI Table Function component. Attributes in the group INGRP1 are migrated to attributes in the connector point INGRP1. The property PARAMETER_TYPE of the input connector point INGRP1 is set to SCALAR.

The OWB output attribute group RETURN is migrated to the output connector point RETURN in ODI. Attributes in the group RETURN are migrated to attributes in the connector point RETURN. If the property RETURN_TABLE_OF_SCALAR of the

output attribute in OWB is set to true, the expression of the corresponding output attribute in ODI is set to TABLE_FUNCTION_NAME.COLUMN_VALUE.

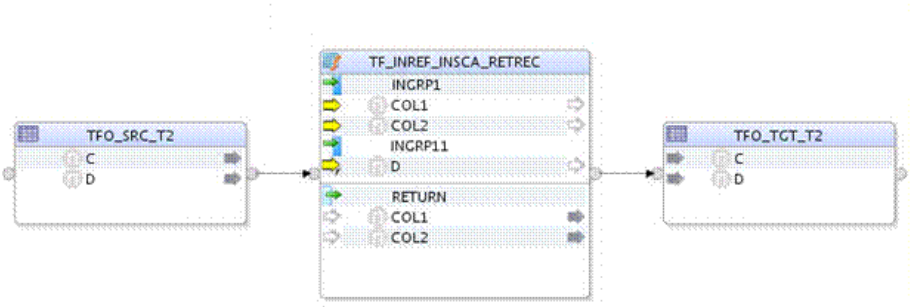
Scenario 3: Table Function operator has one input attribute group and one output attribute group, some data types of input attributes are REF_CURSOR

The following figure shows an OWB mapping for which the data type for attribute C in the operator TF_INREF_INSCA_REFREC is a PL/SQL Ref Cursor type, and the operator REF1 is responsible for constructing the Ref cursor.



If the input attribute group contains one or more REF_CURSOR type attributes in the Table Function operator in OWB, an input connector point is added for each REF_CURSOR type in ODI. If the REF_CURSOR type is constructed by a Constructed operator in OWB, the input attribute group of the Construct operator is used to define the REF_CURSOR input connector point for the Table Function component in ODI.

In this scenario, the OWB mapping in the preceding figure is migrated to the ODI mapping in the following figure:



Source TFO_SRC_T2 is connected to TF_INREF_INSCA_RETREC through the input connector point INGRP1. The property PARAMETER_TYPE of INGRP1 is set to REF_CURSOR. The property PARAMETER_TYPE of INGRP11 is set to SCALAR.

Transformation Function Operator

The OWB Transformation Function operator is migrated to the ODI Expression component.

Properties of the Transformation Function Operator

OWB Property Name	Description	ODI Property Name	Note
Scalar Type			

OWB Property Name	Description	ODI Property Name	Note
Return Type		Attribute under the output connector point.	1. The OWB output group RETURN is migrated as the ODI Expression's output connector point RETURN. 2. The OWB output parameter VALUE is migrated as the ODI attribute VALUE under the RETURN connector point. 3. The attribute's expression is migrated as it is in OWB (kept unchanged).
Input parameters (INPUT)	Accessed by the return attribute's expression field, for example: simpleFunc(INPUT.COL1,INPUT.COL2)	Migrated as the ODI Expression component's attributes under INPUTGROUP.	
Output parameters (OUTPUT)			Not migrated.
Input/Output parameters (INPUT_OUTPUT)			Not migrated.
Function Return Output parameters		Migrated as the ODI Expression component's attributes under OUTPUT GROUP.	If a given Transformation Function operator contains multiple Function Return attributes (at least two), the transformation operator is not migrated.

Note: Additional migration notes:

- If the OWB Transformation Function operator is configured as ROW BASED, the operator is not migrated.
- If the OWB Transformation Function operator has attributes of the BLOB, SYS_ANYDATA or XMLTYPE complex data types, the operator is not migrated.
- Multiple output attributes defined as Function Return are not migrated.

Logical Properties of the Transformation Function Operator

OWB Property Name	Description	ODI Property Name	Note
Function Name (FUNCTION_NAME)	Name of the transformation to be called.		Used to generate the expression on the ODI output attribute. Not migrated if Function Name is empty.
Row-based only (ROW-BASED_ONLY)	Indicates if this transformation must be used in row-based mode only. Some transformations can be used in SQL mode and row-based mode.		Not migrated.
Return type (RETURN_TYPE)	Return type for public transforms with UNSPECIFIED data type.		Not migrated.
Bound Name (BOUND_NAME)	Name to be used by the code generator to identify this operator. By default, this is the same as the operator's physical name.		Not migrated.
Function Expression Holder (FUNCTION_PLATFORM)	Function platform name.		Not migrated.

Physical Properties of the Transformation Function Operator

OWB Property Name	Description	ODI Property Name	Note
Schema (SCHEMA)			Not migrated.
Database Link (DATABASE_LINK)	Database link used to access this entity during mapping.		Not migrated.
Location (DB_LOCATION)	Location, used to access the referenced entity.		Not migrated.

Properties of the Map Attribute Group of the Transformation Function Operator

OWB Property Name	Description	ODI Property Name	Note
Expression Inout (EXPRESSION_INOUT)	Condition that defines when to perform the attribute maps for the attributes in this group.		Not migrated.
Expression Out (EXPRESSION_OUT)	Condition that defines when to perform the attribute maps for the attributes in this group.		Not migrated.

Properties of the Map Attribute of the Transformation Function Operator

OWB Property Name	Description	ODI Property Name	Note
Is Optional (IS_OPTIONAL)	If true, the input is not required to be connected.		Not migrated.
Default Value (DEFAULT_VALUE)	Default Value for the function input parameter.		Not migrated.
Function Return	Specifies whether this output is the return value of this function.	If this property is set to true, the owning attribute is migrated to the ODI output attribute of the Expression component.	

Unpivot Operator

The OWB Unpivot operator is migrated to the ODI Pivot component.

Note that the operation carried out by the OWB Unpivot operator is the same as the ODI Pivot component, and the operation carried out by the OWB Pivot operator is the same as the ODI Unpivot component.

Properties of the Unpivot Operator

General Properties

OWB Property Name	Description	ODI Property Name	Note
Business Name (LOGICAL_NAME)		Business Name (BUSINESS_NAME)	
Physical Name (NAME)		Name (NAME)	If the OWB name includes the string "unpivot", it is changed to "pivot".

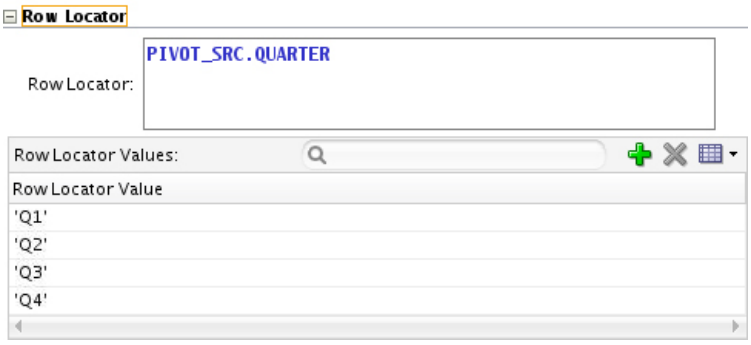
Row Locator

The Row Locator of the OWB Unpivot operator is migrated to the value of the Row Locator property of the ODI Pivot component.

The expression of the Row Locator in OWB must be redirected so that it references the attribute of the upstream source during migration.

Row Locator values in OWB are migrated to Row Locator values in ODI.

Row Locator and Row Locator values in ODI are as follows:



Map Attribute Group and Map Attribute

Map attribute groups of the OWB Unpivot operator are migrated to connector points of the ODI Pivot component. No specific properties for the attribute group of the Pivot operator need to be migrated.

Input attributes of the OWB Pivot operator are not migrated.

Output attributes are migrated. Name, Data Type, Length, Precision, Scale, Second Precision, and Description are general properties described in "Mapping Attribute" on page B-24.

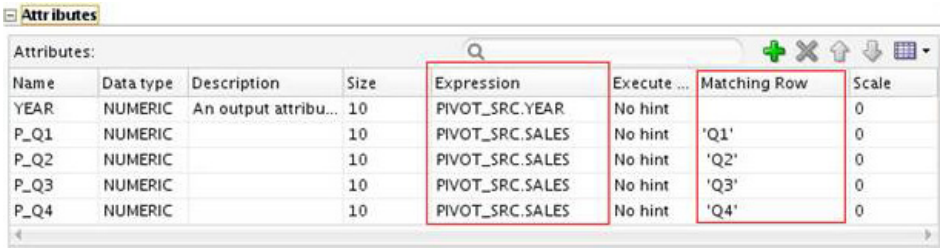
Properties in the Unpivot transform are as follows:

Define the unpivot expression for each of the output attributes:

Attribute	Matching row	Expression
YEAR		INGRP1.YEAR
P_Q1	'Q1'	INGRP1.SALES
P_Q2	'Q2'	INGRP1.SALES
P_Q3	'Q3'	INGRP1.SALES
P_Q4	'Q4'	INGRP1.SALES

The matching row of the output attribute in OWB is migrated to the matching row of the output attribute in ODI. The expression of the output attribute in OWB is migrated to the expression of the output attribute in ODI. The expression must be redirected to reference the attribute of the upstream source.

The following figure shows these properties in ODI:



View Operator

The OWB View operator is migrated to the ODI Datastore component.

Logical Properties of the View Operator

General Properties

OWB Property Name	Description	ODI Property Name	Note
Bound Name (BOUND_NAME)			If the OWB View operator is bound to a view, the ODI Datastore component is bound with the corresponding data store.
Primary Source (PRIMARY_SOURCE)	A boolean value to indicate whether this is a primary source (only used in EDW). (YES/NO)		Not migrated.
Keys (KEYS_READONLY)			Not migrated.
Inlined (INLINED)	If true, the view source in the generated code is inlined from the stored view query.		See the View Query property in this table.
View Query (VIEW_QUERY)	The view query for the View operator, used if the INLINED property is set to true.	If INLINED is set to true, View Query is migrated to the CUSTOM_TEMPLATE option of the KM.	
Loading Type (LOADING_TYPE)	Choices = "INSERT, UPDATE, INSERT_UPDATE, UPDATE_INSERT, DELETE, NONE, TRUNCATE_INSERT, DELETE_INSERT, CHECK_INSERT, DERIVE_FROM_LCR"	INTEGRATION_TYPE	Same as for the Table operator. See Notes About Loading Type.
Target Load Order (TARGET_LOAD_ORDER)	Map targets names in loading sequence.		Not migrated.

Change Data Capture

Same as for the Table operator. See "Change Data Capture" on page C-36.

Chunking

As with the Table operator, properties for Chunking are not migrated.

Conditional Loading

Same as for the Table operator. See "Conditional Loading" on page C-37.

Data Rules

As with the Table operator, properties for Data Rules are not migrated.

Error Table

As with the Table operator, properties for Error Table are not migrated.

SCD Updates

As with the Table operator, properties for SCD Updates are not migrated.

Temp Stage Table

As with the Table operator, properties for Temp Stage Table are not migrated.

Physical Properties of the View Operator

Same as for the Table operator. See "Physical Properties of the Table Operator" on page C-39.

Logical Properties of the Attributes of the View Operator

Same as for the Table operator. See "Logical Properties of the Attributes of the Table Operator" on page C-39.

Migrating an Unbound View Operator

Same as for the Table operator. See "Migrating an Unbound Table Operator" on page C-41.

D

Special Migration Cases

This appendix provides examples of special migration cases.

The following topics are addressed here:

- Tables with Multiple Primary Keys
- Special Cases for Mappings

Tables with Multiple Primary Keys

OWB tables are migrated to ODI data stores. In OWB, tables can have multiple primary keys. In ODI, data stores can have only one primary key. In the case of multiple primary keys, the first primary key is migrated as the primary key in ODI, and the others are migrated as alternate keys.

When this situation occurs, the following warning message is written to the migration utility log file:

```
{0}:{1} has multiple primary keys. Only one primary key is allowed in ODI, the
redundant primary keys will be migrated as alternate keys.
```

Special Cases for Mappings

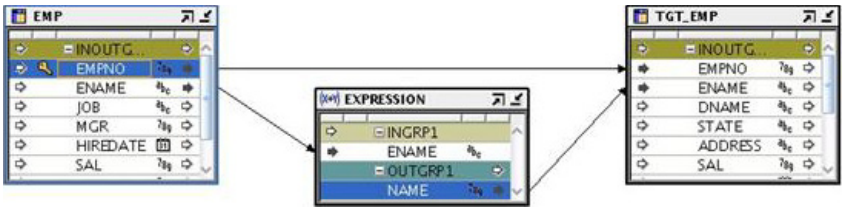
Some OWB mappings have different graph structures after they are migrated to ODI. The migration utility attempts to migrate OWB mappings to ODI as closely as possible, but in some cases the resulting ODI mappings may not correspond to the original OWB mapping structure.

The following special cases for mappings are addressed here:

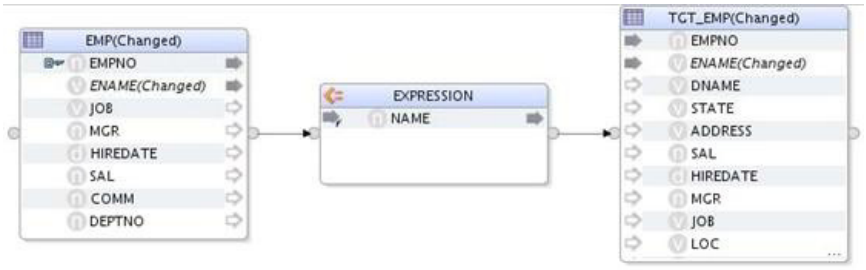
- Two Operators Connected to Same Downstream Operator
- Multiple Operators Connected From and To Same Operator
- Lookup Operator Has a Constant as Input
- Lookup Operators Have No Driver Table (Mapping Is Invalid)
- Multiple Operators Connected to Same Operator, Some with No Upstream Source
- Multiple Operators Connected to Same Operator, All with Different Upstream Operator
- Pluggable Mapping Operator with only Constant as Input

Two Operators Connected to Same Downstream Operator

The following figure shows an OWB mapping for which operators EMP and EXPRESSION are both connected to operator TGT_EMP through the same map attribute group INOUTGRP1. This is not allowed in ODI, because each input connector point in ODI can only be connected once.

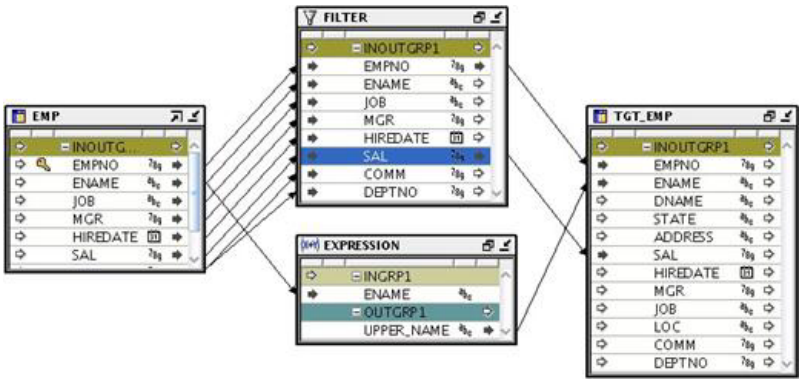


The OWB mapping in the preceding figure is migrated to the ODI mapping in the following figure.



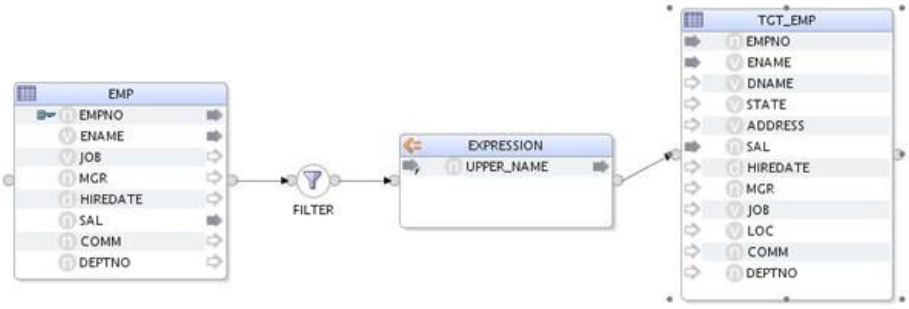
Multiple Operators Connected From and To Same Operator

The following figure shows an OWB mapping for which operators FILTER and EXPRESSION are both connected to operator TGT_EMP through the same map attribute group INOUTGRP1. This is not allowed in ODI.



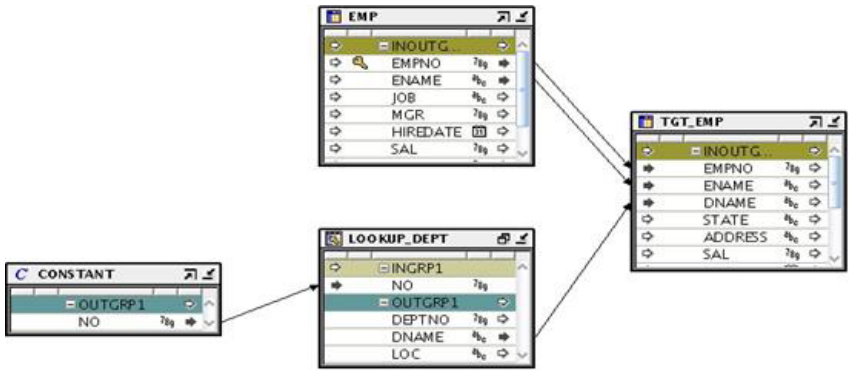
During migration, the FILTER and EXPRESSION operators are chained together to ensure that only one is connected to TGT_EMP. As a result, the ODI mapping may be EMP > FILTER > EXPRESSION > TGT_EMP or EMP > EXPRESSION > FILTER > TGT_EMP.

The OWB mapping in the preceding figure is migrated to the ODI mapping in the following figure.

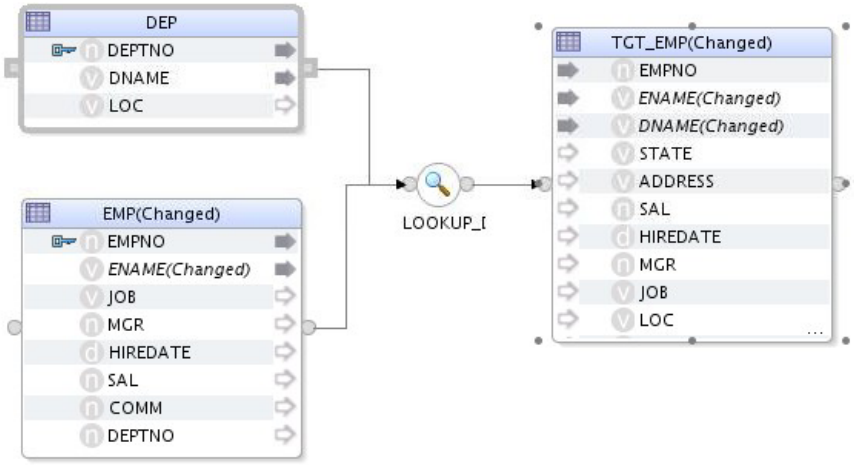


Lookup Operator Has a Constant as Input

The following figure shows an OWB mapping for which the Lookup operator has no upstream source operator, and is only connected from a constant.



The OWB mapping in the preceding figure is migrated to the ODI mapping in the following figure (DEP is the lookup table of the Lookup operator).

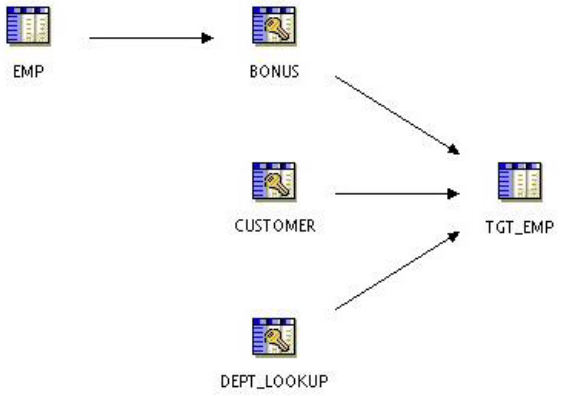


The constant operator CONSTANT in the OWB mapping is not migrated to any map component in ODI. Instead, the expression of the constant attribute is migrated, and that expression is set on the Lookup component.

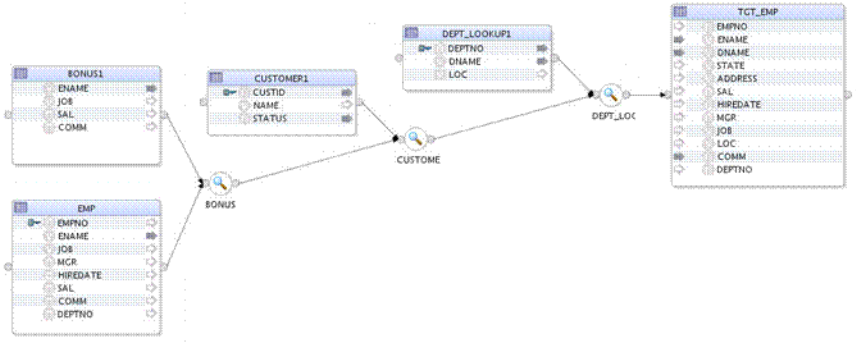
For example, in OWB, if the expression of the attribute CONSTANT.OUTGRP1.NO is set to 5, and the lookup condition of LOOKUP_DEPT is OUTGRP1.DEPTNO = INGRP1.NO, then after migration the lookup condition of LOOKUP_DEPT in ODI is DEP.DEPTNO = 5.

Lookup Operators Have No Driver Table (Mapping Is Invalid)

The following figure shows an OWB mapping for which several Lookup operators are connected to operator TGT_EMP, but some of the Lookup operators have no upstream operators as driver tables. This mapping is invalid, but will also be migrated. All Lookup operators are chained together to ensure that only one is connected to TGT_EMP.

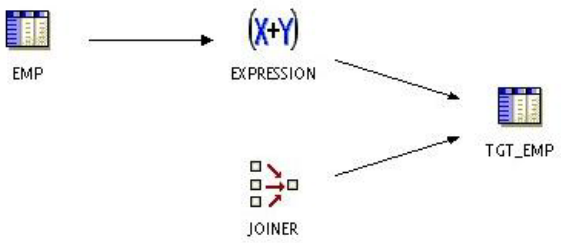


The OWB mapping in the preceding figure is migrated to the ODI mapping in the following figure.

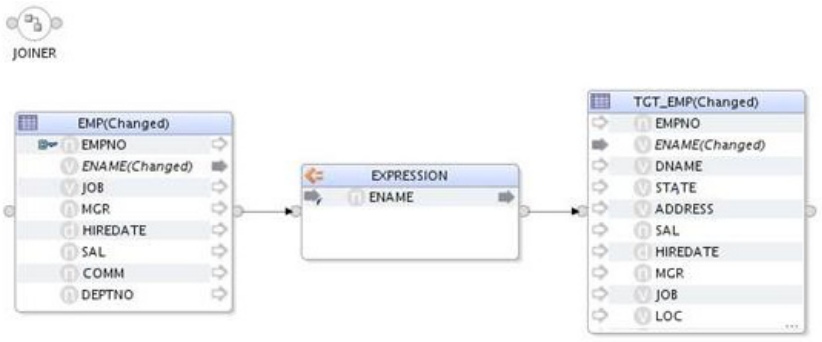


Multiple Operators Connected to Same Operator, Some with No Upstream Source

The following figure shows an OWB mapping for which two operators are connected to the same operator TGT_EMP. The EXPRESSION operator has an upstream source operator, while the JOINER operator does not. Only one map component can be connected to TGT_EMP in ODI. As a result, the operator with no upstream source operator will lose the connection to TGT_EMP.

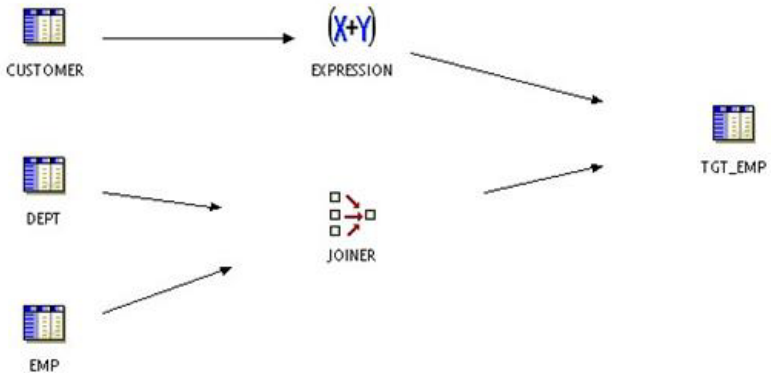


The OWB mapping in the preceding figure is migrated to the ODI mapping in the following figure.

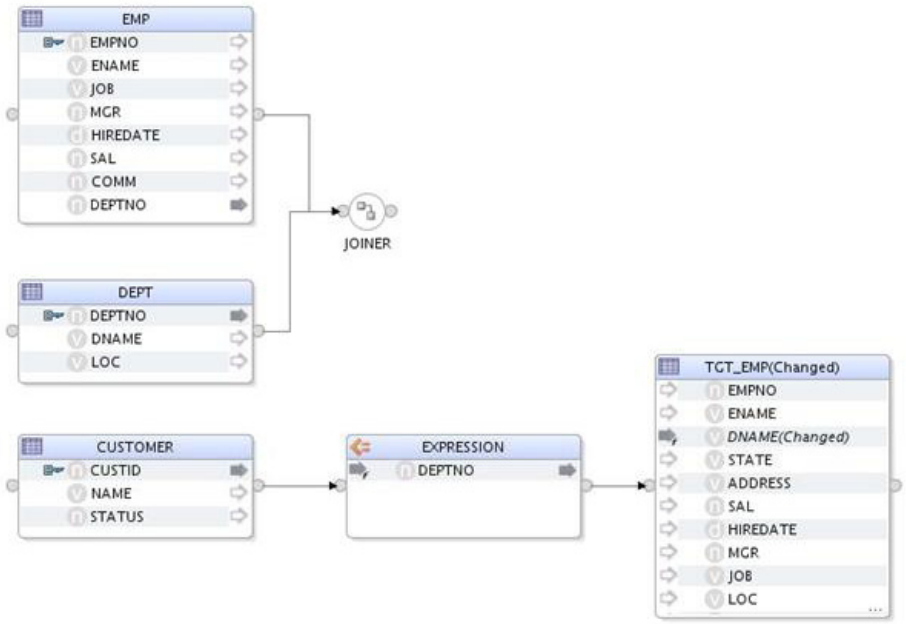


Multiple Operators Connected to Same Operator, All with Different Upstream Operator

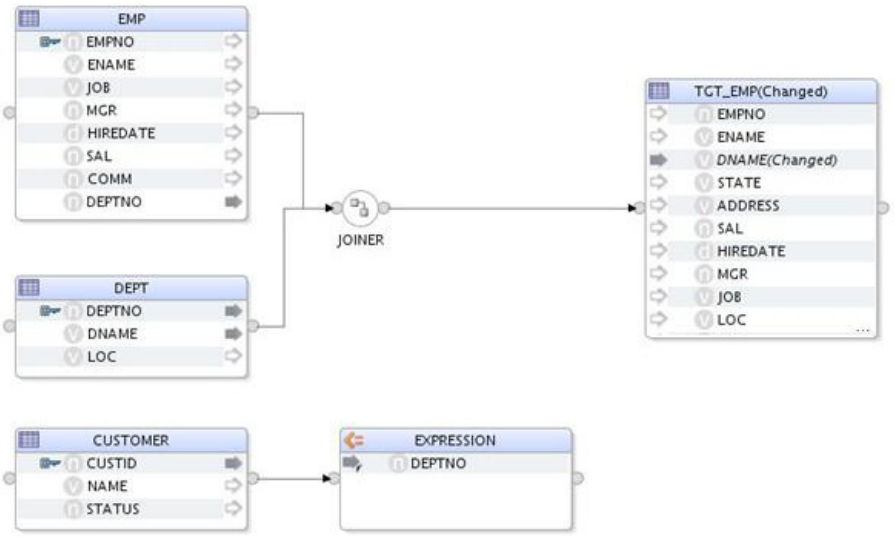
The following figure shows an OWB mapping for which two operators are connected to the same operator TGT_EMP. Both operators have an upstream operator. Only one map component can be connected to TGT_EMP in ODI. As a result, one operator will lose the connection to TGT_EMP.



The OWB mapping in the preceding figure is migrated to one of the ODI mappings in the following figures.



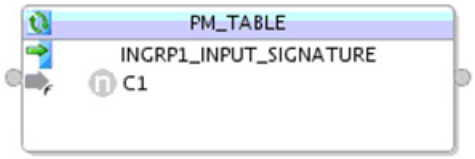
-or-



Pluggable Mapping Operator with only Constant as Input

The following figure shows an OWB mapping for which the pluggable mapping operator has no upstream source operator, and is only connected from a constant. Such kind of mapping may work in OWB.

The OWB mapping in the preceding figure is migrated to the ODI mapping in the following figure.



The constant operator CONSTANT in the OWB mapping is not migrated to any map component in ODI. Instead, the expression of the constant attribute is migrated, and that expression is set on the attribute of the reusable mapping component. Such ODI mapping has execution issue and it needs to be manually fixed after migration.

E

Known Issues and Solutions

This appendix lists the known issues and their solutions.

This appendix contains the following topics:

Known Issues and Solutions

The following are the issues are known at the time of the release. This section also provides the solutions to work around these issues.

1. **Symptom:** The OWB to ODI Migration Utility does not migrate the comments from OWB Joiner Condition to ODI Joiner Condition.

   Solution: Comments are migrated when the OWB Mapping is not ANSI. Otherwise, the comments are filtered out and not migrated. If you need Joiner comments to be migrated over for OWB mapping that is ANSI, you can use the migration configuration property SPLIT_JOIN_FOR_ANSI_SYNTAX=false and migrate the mapping again.

2. **Symptom:** A misspelled configuration parameter has been added to the configuration file. The Migration Utility execution does not report/list the misspelled configuration parameter in the generated log files.

   Solution: Check carefully the properties name in the migration utility configuration file.A sample configuration file is provided in the following location.

   ```
   <owb_home>/owb/bin/admin/migration.config
   ```

3. **Symptom:** After having migrated OWB mappings to ODI using the OWB-ODI Migration Utility, ODI generates database links for mappings that have source and target table in the same Oracle database.

   Cause: This is not a bug. In OWB, when having e.g. a mapping with one source and one target in the same database but in different schemas, one location is required for every schema. When these two locations are migrated using the migration utility, the utility will create two data servers in topology. However, when ODI generates code for a mapping having source and target from different data servers, ODI will generate a database link to the source. The disparity in the code generated in ODI when compared with OWB is coming from the conceptual differences about how connections and schemas are described in OWB and ODI. The semantics and assumption in ODI is that if two physical schemas are under different data servers, these two physical schemas are always considered to be in different database instances. As a result, ODI will create two execution units in the ODI mapping and thus a database link will be used to access the source table. This is the expected behavior in ODI. The OWB-ODI migration utility is also behaving

as designed. Since it is possible in OWB to have multiple locations with the same host/post/service name information but different user or schema, this will result in having multiple data servers with the same JDBC URL in ODI after migration.

Solution: Do the following:

- Create a new physical schema (pointing to the source) under the dataserver of the target.
- Change the context for the logical schema for the original source to make it point to the new physical schema.

4. **Symptom:** The Migration Utility fails to migrate OWB mappings containing the Data Generator operator. The Data Generator operator is used to introduce constants or sequences into a SQL*Loader mapping.

   migration.log shows:

   --------START MIGRATE MAPPING MAP1_DATA_GENERATOR.

   ----------[ERROR][Migration][MU-5001]Unable to migrate mapping with mapping operator DATA_GENERATOR:DATA_GENERATOR.

   --------FAILED MIGRATE MAP1_DATA_GENERATOR.

   The Migration Utility does not support upgrading OWB mappings that contain the Data Generator operator.

   Solution:

   - Before migrating, remove the Data Generator operator from the OWB mapping.
   - Migrate the mapping from OWB to ODI.
   - After migration add a constant or sequence (depending on the way the Data Generator was used) to the ODI mapping.

5. **Symptom:** When attempting to migrate an OWB Project with only a few selected objects into ODI 12c the result in unexpected.Instead of migrating in a 1 to 1 manner (for example to MY_PROJECT), the operation creates a new Project (ex MY_PROJECT_0) in ODI.

   Solution: A new value for parameter MIGRATION_STRATEGY has been introduced in migration utility for ODI 12.1.3 and above. Change the MIGRATION_STRATEGY parameter from CREATE to NODUP.

6. **Symptom:** The OdiStartOwbJob utility is used to execute Oracle Warehouse Builder objects (e.g. mappings, process flows) from within Oracle Data Integrator and to retrieve the execution audit data into Oracle Data Integrator. Trying to configure OdiStartOwbJob, the Location listbox shows the location PlatformSchema. However all other locations are missing. Also a situation might occur where location PlatformSchema is listed together with only a subset of locations that can be seen in the OWB Design Client.

   Cause: The location listbox of the OdiStartOwbJob utility, shows only locations that are registered in the OWB Control Center. Locations that are only registered in the Design Center are not listed by OdiStartOwbJob.

   Solution:

   - Start OWB 11.2.0.4 Design Client and login to the OWB Repository.
   - Open the Control Center and register locations (target locations, process flow locations) that are missing in the listbox of the OdiStartOwbJob utility.

- Exit from the OWB Design Client.
- Start ODI 12c Studio and login to the ODI Repository.
- Open the Package where OdiStartOwbJob is being used and verify that the location listbox of OdiStartOwbJob shows the registered locations.

7. **Symptom:** The OdiStartOwbJob utility is used to execute Oracle Warehouse Builder objects (e.g. mappings, process flows) from within Oracle Data Integrator and to retrieve the execution audit data into Oracle Data Integrator.

   Starting an ODI package containing OdiStartOwbJob fails with:

   ODI-13702: Unexpected error when connecting to OWB workspace OWB_WORKSPACE_OWNER.OWB_WORKSPACE_NAME.

   StatementCallback; bad SQL grammar [SELECT LOCATION_NAME, LOCATION_TYPE, LOCATION_TYPE_VERSION FROM OWBSYS.OWB_ODI_LOCATIONS]; nested exception is java.sql.SQLSyntaxErrorException: ORA-00942: table or view does not exist.

   Solution:

   1. Check if required privileges are missing.

      Using the query below, check if required privileges are missing:

      ```
      connect owbsys/<password>
      set lines 130
      SELECT grantee,
          table_name,
          privilege
       FROM user_tab_privs
      WHERE table_name IN ( 'OWB_ODI_LOCATIONS', 'OWB_ODI_TASKS',
                            'OWB_ODI_TASK_PARAMETERS', 'OWB_SNP_SESSIONS',
                            'OWB_SNP_SESS_STEPS', 'OWB_SNP_SESS_TASKS',
                            'WB_RT_ODIAUDIT' );
      ```

   2. Grant required privileges.

      When required privileges are missing, grant the correct privileges using following grants:

      ```
      connect owbsys/<password>
      grant execute on wb_rt_odiaudit to OWB_USER;
      grant select on owb_odi_locations to OWB_USER;
      grant select on owb_odi_tasks to OWB_USER;
      grant select on owb_odi_task_parameters to OWB_USER;
      grant select on owb_snp_sessions to OWB_USER;
      grant select on owb_snp_sess_steps to OWB_USER;
      grant select on owb_snp_sess_tasks to OWB_USER;
      ```