

**Oracle Data Integrator 12c:
Integration and Administration**
Student Guide – Volume I

D82167GC10
Edition 1.0
May 2014
D86564

ORACLE®

Author

Steve Friedberg

**Technical Contributors
and Reviewers**

Phil Scott
Gerry Jurrens
Brent Dayley
Surendra Babu
Rick Green
Viktor Tchemodanov
Julien Testut
Alex Kotopoulos
Alessandro Leite

Editor

Daniel Milne

Graphic Designer

Seema Bopaiah

Publishers

Michael Sebastian
Veena Narasimhan

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Disclaimer

This document contains proprietary information and is protected by copyright and other intellectual property laws. You may copy and print this document solely for your own use in an Oracle training course. The document may not be modified or altered in any way. Except where your use constitutes "fair use" under copyright law, you may not use, share, download, upload, copy, print, display, perform, reproduce, publish, license, post, transmit, or distribute this document in whole or in part without the express authorization of Oracle.

The information contained in this document is subject to change without notice. If you find any problems in the document, please report them in writing to: Oracle University, 500 Oracle Parkway, Redwood Shores, California 94065 USA. This document is not warranted to be error-free.

Restricted Rights Notice

If this documentation is delivered to the United States Government or anyone using the documentation on behalf of the United States Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS

The U.S. Government's rights to use, modify, reproduce, release, perform, display, or disclose these training materials are restricted by the terms of the applicable Oracle license agreement and/or the applicable U.S. Government contract.

Trademark Notice

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Contents

1 Introduction to Integration and Administration

Course Objectives	1-2
Lesson Objectives	1-3
Agenda of Lessons	1-4
Agenda	1-7
Why Oracle Data Integrator?	1-8
Conventional Integration Process: ETL	1-10
Extract Load Transform (E-LT)	1-11
ODI Architecture and Components	1-13
ODI Architecture	1-14
ODI Components: Overview	1-15
Using ODI Studio	1-16
Designer Navigator (Work Repository)	1-17
Operator Navigator (Work Repository)	1-18
Topology Navigator (Master Repository)	1-19
Security Navigator (Master Repository)	1-20
What Is an Agent?	1-21
ODI Agents	1-22
Three Types of Agents: Java EE, Standalone, Collocated Standalone	1-23
Using the Three Types of Agents	1-24
Standalone Agent: Example	1-25
ODI Console	1-26
Enterprise Manager FMW Console	1-27
Management Pack for ODI for Enterprise Manager Cloud Control	1-28
Management Pack for ODI for EM CC ODI Home Page	1-29
Agenda	1-30
ODI Repositories	1-31
Master and Work Repositories	1-32
Repository Setup: Example	1-34
Repository Setup: Multiple Master Repositories	1-35
Components: Global View	1-36
Possible ODI Methodology	1-37
Checklist of Practice Activities	1-38
Starting Oracle Data Integrator	1-39
Using Online Help	1-40

Quiz 1-41
Summary 1-43
Practice 1-1: Logging In and Help Overview 1-44

2 Administering ODI Repositories

Objectives 2-2
Agenda 2-3
Initial Repository Administration Tasks 2-4
Steps to Set Up ODI Repositories 2-5
1. Run Repository Creation Utility 2-6
1a. Create Schemas 2-7
1b. Create Passwords and Tablespaces 2-8
2. Connect to the Master/Work Repository 3. Create a Wallet 2-9
Connecting to the Master/Work Repository 2-10
Exporting the Master Repository 2-11
Importing the Master Repository 2-12
Creating a Work Repository 2-13
Changing the Work Repository Password 2-15
Quiz 2-16
Summary 2-17
Checklist of Practice Activities 2-18
Practice 2-1: Creating and Connecting to ODI Master and Work Repositories 2-19

3 ODI Topology Concepts

Objectives 3-2
Agenda 3-3
What Is Topology? 3-4
What Is in the Topology? 3-5
Agenda 3-6
What Is a Data Server? 3-7
Data Servers: Examples 3-8
Important Guideline 1 3-9
What Is a Physical Schema? 3-10
Physical Schemas: Properties 3-11
Technology Terminology Among Vendors 3-12
Important Guideline 2 3-13
Agenda 3-14
Infrastructure for Two Production Sites: Example 3-15
ODI Design: Physical Architecture of the Two Production Sites 3-16
Logical Schemas and Contexts 3-17
What Is a Logical Schema? 3-18

Important Guideline 3	3-19
Logical Versus Physical Architecture	3-20
Design Time Versus Run Time	3-21
What Is a Context?	3-22
A Context Maps a Logical to a Physical Schema	3-23
Defining Contexts	3-24
Mapping Logical and Physical Resources	3-25
Agenda	3-27
ODI Physical Agents	3-28
Creating a Physical Agent	3-29
ODI Agent Parameters	3-30
Launching a Stand-Alone Agent: Examples	3-32
Stopping the ODI Agent	3-33
Deploying and Configuring a Java EE Agent	3-34
Load Balancing: Example	3-37
Important Guideline 5	3-39
Infrastructure with Agents: Example	3-40
Defining Agents: Example	3-41
Special Case: Fragmentation Problem	3-42
Special Case: Important Guideline 6	3-44
Special Case: Defining the Physical Architecture	3-45
Special Case: The Infrastructure	3-46
Special Case: Physical Architecture in ODI	3-47
Agenda	3-48
Planning the Topology	3-49
Matrix of Logical and Physical Mappings	3-50
Quiz	3-51
Summary	3-54
Checklist of Practice Activities	3-55
Practice 3-1: Configuring a Standalone Agent by Using the Common Administration Model	3-56

4 Describing the Physical and Logical Architecture

Objectives	4-2
Agenda	4-3
What Topology Navigator Contains	4-4
Topology Navigator: Overview	4-5
Review: Context Connects Logical to Physical	4-7
Objects You Create in the Practice	4-8
Defining a Context	4-9
Agenda	4-10

Physical Architecture View	4-11
Prerequisites for Connecting to a Server	4-12
Important Note	4-13
Creating a Data Server	4-14
Creating a Data Server: JDBC	4-15
JDBC Driver	4-16
JDBC URL	4-17
Creating a Data Server: JNDI	4-18
Testing a Data Server Connection	4-19
Creating a Physical Schema	4-20
Agenda	4-21
Logical Architecture and Context Views	4-22
Creating a Logical Schema	4-23
Creating a Logical Agent	4-24
Editing a Context to Link Logical and Physical Agents	4-25
Quiz	4-26
Summary	4-28
Checklist of Practice Activities	4-29
Practice 4-1: Working with Topology	4-30

5 Setting Up a New ODI Project

Objectives	5-2
Agenda	5-3
What Is a Project?	5-4
Oracle Data Integrator Projects: Overview	5-5
How to Use ODI Projects in Your Work	5-6
Creating a New Project	5-7
Agenda	5-8
What Is a Folder?	5-9
Creating a New Folder	5-10
Organizing Projects and Folders	5-11
Agenda	5-12
What Is a Knowledge Module?	5-13
Types of Knowledge Modules	5-14
Which Knowledge Modules Are Needed?	5-15
Knowledge Modules: Examples	5-16
Importing Knowledge Modules	5-17
Replacing Existing KMs	5-18
Knowledge Module Editor	5-20
Editing a Knowledge Module	5-21
Agenda	5-22

Exporting and Importing	5-23
Exporting an Object	5-24
Importing an Object	5-25
ID Numbers: Overview	5-26
Import Types	5-27
Choosing the Import Mode	5-28
Import Report	5-29
Agenda	5-30
What Is a Marker?	5-31
Tagging Objects with Markers	5-32
Removing Markers	5-33
Marker Groups	5-34
Project and Global Markers	5-35
Creating a Marker Group	5-36
Quiz	5-37
Summary	5-39
Checklist of Practice Activities	5-40
Practice 5-1: Setting Up a New ODI Project	5-41

6 Oracle Data Integrator Model Concepts

Objectives	6-2
What Is a Model?	6-3
Agenda	6-4
Relational Model	6-5
Relational Model: Tables and Columns	6-6
Relational Model: Keys	6-7
Relational Model: Foreign Keys	6-8
Relational Model: Constraints	6-9
Relational Model: Indexes	6-11
Relational Model Support in ODI	6-12
Additional Metadata in ODI	6-13
FlexFields	6-15
Agenda	6-16
What Is Reverse-Engineering?	6-17
Methods for DBMS Reverse-Engineering	6-18
Other Methods for Reverse-Engineering	6-19
Standard Versus Customized Reverse-Engineering	6-20
Reverse-Engineering Life Cycle	6-21
Agenda	6-22
Creating a Model by Reverse-Engineering	6-23
Step 1: Creating and Naming a New Model	6-24

Note: Creating and Naming a New Model	6-25
Step 2: Defining a Reverse-Engineering Strategy	6-26
Step 3: Starting the Reverse-Engineering Process	6-28
Using RKM for Customized Reverse-Engineering	6-29
Selective Reverse-Engineering	6-31
Step 4: Fleshing Out Models	6-32
Shortcuts	6-33
Smart Export and Import	6-34
Quiz	6-35
Summary	6-37
Checklist of Practice Activities	6-38
Practice 6-1 Overview: Creating Models by Reverse-Engineering	6-39

7 Organizing ODI Models and Creating ODI Datastores

Objectives	7-2
Agenda	7-3
What Is a Model Folder?	7-4
Creating a Model Folder	7-5
What Is a Submodel?	7-6
Creating a Submodel	7-7
Organizing Datastores into Submodels	7-8
Setting Up Automatic Distribution	7-9
Agenda	7-10
Creating Datastores	7-11
Creating a Datastore in a Model	7-12
Adding Columns to a Datastore	7-13
Agenda	7-14
What Is a Constraint in ODI?	7-15
Constraints in ODI	7-16
Creating a Mandatory Column	7-17
Agenda	7-18
Creating a Key	7-19
Checking a Key	7-20
Creating a Reference	7-21
Creating a Simple Reference	7-22
Creating a Complex Reference	7-23
Checking a Reference	7-24
Agenda	7-25
Creating a Condition	7-26
Checking a Condition	7-27
Agenda	7-28

Audit/Explore: When and Why	7-29
Audit/Explore Process: Overview	7-30
Agenda	7-31
Displaying the Contents of a Datastore	7-32
Viewing the Distribution of Values	7-33
Analyzing the Contents of a Datastore	7-34
Agenda	7-35
Defining Business Rules in ODI	7-36
From Business Rules to Constraints	7-37
Deducing Constraints from Data Analysis	7-38
Testing a Constraint	7-39
Auditing a Model or Datastore	7-40
Reviewing Erroneous Records	7-41
Quiz	7-42
Summary	7-44
Checklist of Practice Activities	7-45
Practice 7-1: Checking Data Quality in the Model	7-46

8 ODI Mapping Concepts

Objectives	8-2
Agenda	8-3
What Is a Mapping?	8-4
Business Rules for Mappings	8-5
Where Are the Rules Defined?	8-6
Agenda	8-7
What Is an Expression?	8-8
What Is a Join?	8-9
What Is a Filter?	8-10
What Is a Lookup?	8-11
What Is a Set?	8-12
What Are Some of the Others?	8-13
New with Patch: Pivot and Unpivot	8-14
Agenda	8-15
How Does ODI Implement Business Rules?	8-16
Business Problem	8-17
Implementing the Rules	8-18
Integration Process	8-19
Process Details	8-20
Process Implementation: Example 1	8-21
Process Implementation: Example 2	8-22
Process Implementation: Example 3	8-23

Agenda	8-24
What Is the Staging Area?	8-25
Execution Location	8-26
Agenda	8-27
From Business Rules to Processes	8-28
Knowledge Modules	8-29
What Is a Knowledge Module?	8-30
Code Generation	8-31
KM Types Used in Mappings	8-32
Agenda	8-33
Purpose of a Mapping	8-34
What Is an Expression?	8-35
Creating a One-to-One Mapping	8-36
Creating and Naming a Mapping	8-37
Defining the Target Datastore	8-38
Multiple Targets	8-39
Defining the Source Datastore	8-40
Connecting the Ports to Make the Map	8-41
Defining the Expressions	8-42
Valid Expression Types	8-43
Saving the Mapping	8-44
Running the Mapping	8-45
Quiz	8-46
Summary	8-48
Checklist of Practice Activities	8-49
Practice 8-1: Mapping: Simple Transformations	8-50

9 Designing Mappings

Objectives	9-2
Agenda	9-3
Multiple-Source Datastores	9-4
Creating a Join Manually	9-5
Advanced Joins	9-6
Types of Joins	9-7
Setting Up a Join	9-8
Creating Lookups	9-10
Using Lookups	9-11
Agenda	9-13
Filters in ODI	9-14
Defining a Filter Manually	9-15
Setting Up a Filter	9-16

Agenda	9-17
Physical Mapping Diagram	9-18
Flow in the Physical Diagram	9-20
What Defines the Flow?	9-21
Scenario	9-22
Basic Process	9-23
Agenda	9-24
Purpose of a Staging Area	9-25
Placing the Staging Area	9-26
Important Note	9-27
Specifying the Staging Area	9-28
Agenda	9-29
Options for Expressions	9-30
Setting Options for Expressions	9-31
Disabling an Expression	9-32
Enabling a Mapping for Inserts or Updates	9-33
Agenda	9-34
Execution Location and Syntax	9-35
Why Change the Execution Location?	9-36
Changing the Execution Location	9-37
ODI Mapping Execution Simulation	9-38
Agenda	9-39
Which KMs for Which Flow?	9-40
Knowledge Modules: Additional Information	9-42
Identifying IKMs and LKMs	9-43
IKMs and LKMs: Strategies and Methods	9-44
Specifying an LKM	9-45
Specifying an IKM	9-46
Common KM Options	9-47
Flow: Example 1	9-48
Flow: Example 2	9-49
Flow: Example 3	9-50
Quiz	9-51
Summary	9-52
Checklist of Practice Activities	9-53
Practice 9-1: Mapping: Complex Transformations	9-54
Practice 9-2: Mapping: Implementing Lookup	9-55

10 Mappings: Monitoring and Troubleshooting

Objectives 10-2

Agenda 10-3

Operator Navigator: Viewing the Log	10-4
Using Operator Navigator	10-5
Hierarchy: Sessions, Steps, Tasks	10-6
Viewing Details of Sessions, Steps, and Tasks	10-7
Monitoring Execution of an Mapping	10-8
Troubleshooting a Session	10-9
1. Identifying the Error	10-10
2. Reviewing the Code	10-11
3. Fixing the Code and Restarting the Session	10-12
4. Fixing the Mapping	10-13
Keys to Reviewing the Generated Code	10-14
Agenda	10-15
Common Errors and Symptoms	10-16
Important Note	10-18
Tips for Preventing Errors	10-19
Using Attribute Panel for Quick Edits	10-20
Quiz	10-21
Summary	10-23
Checklist of Practice Activities	10-24
Practice 10-1: Creating ODI Mapping: Exporting a Flat File to a Relational Table	10-25

11 Designing Mappings: Advanced Topics 1

Objectives	11-2
Agenda	11-3
Business Rules in Mappings	11-4
Business Rule Elements	11-5
More Elements	11-6
Expression Editor	11-7
Agenda	11-9
Using a Variable in Code	11-10
Binding Versus Substitution	11-12
Case Sensitivity	11-13
Agenda	11-14
Defining a Dataset	11-15
Using Set-Based Operators	11-16
Example of SET: UNION	11-17
Agenda	11-18
Types of Sequences	11-19
Support for Native Sequences	11-20
Creating a Native Sequence	11-21

Referring to Sequences	11-22
Note: Sequences Updated by Agent	11-23
Using Standard Sequences in Mappings Correctly	11-24
Using ODI Standard Sequences in Mappings	11-25
Populating Native Identity Attributes	11-26
Sequences: Best Practices	11-27
Automatic Temporary Index Management	11-28
Tracking Variables and Sequences	11-29
How Variable and Sequence Tracking Works	11-30
Variable Actions	11-31
Definition Tab of Session Step or Session Task	11-32
Quiz	11-33
Summary	11-34
Checklist of Practice Activities	11-35
Practice 11-1: Using Native Sequences with ODI Mapping	11-36
Practice 11-2: Using Temporary Indexes	11-37
Practice 11-3: Using Sets with ODI Mapping	11-38

12 Designing Mappings: Advanced Topics 2

Objectives	12-2
Agenda	12-3
Partitioning	12-4
Definition in Datastore After Reverse-Engineering	12-5
Using Partitioning in a Mapping	12-6
Agenda	12-7
Reusable Mappings	12-8
Using Reusable Mappings: Example	12-9
Derived Select (Subselect) for Reusable Mappings	12-10
Agenda	12-11
What Is a User Function?	12-12
Why Use User Functions?	12-13
Properties of User Functions	12-15
Using User Functions	12-16
Creating a User Function	12-17
Defining an Implementation	12-18
Syntax and Implementations	12-19
User Functions at Design Time	12-20
User Functions at Run Time	12-21
Note: Functions in Execution Log	12-22
Agenda	12-23
Using Substitution Methods	12-24

Substitution Methods: Examples	12-26
Agenda	12-27
Description of KM Steps	12-28
Details of the Steps	12-29
Setting KM Options	12-30
Developing Your Own KM: Guidelines	12-31
Complex File Technology	12-33
Quiz	12-34
Summary	12-35
Checklist of Practice Activities	12-36
Practice 12-1: Creating and Using Reusable Mappings	12-37
Practice 12-2: Developing a New Knowledge Module	12-38

13 Using ODI Procedures

Objectives	13-2
Agenda	13-3
What Is a Procedure?	13-4
Procedure: Examples	13-5
Creating Procedures: Overview	13-7
Agenda	13-8
Creating a New Procedure	13-9
Agenda	13-10
Creating a Command	13-11
Arranging Tasks in Order	13-13
Which Parameters Should Be Set?	13-14
Valid Types of Commands	13-15
More Elements	13-16
Why Use a Source Command?	13-17
Agenda	13-18
Types of Options	13-19
Creating a New Option	13-20
Making a Command Optional	13-21
Using an Option Value in a Command	13-22
Agenda	13-23
Procedure Execution	13-24
Using the Operator Navigator to View Results	13-25
Quiz	13-26
Summary	13-28
Checklist of Practice Activities	13-29
Practice 13-1: Creating an ODI Procedure	13-30

14 Using ODI Packages

- Objectives 14-2
- Agenda 14-3
- What Is a Package? 14-4
- Creating a Package 14-5
- Agenda 14-6
- Creating and Naming a Package 14-7
- Package Diagram 14-8
- Package Diagram Toolbar 14-9
- Agenda 14-11
- Package Steps 14-12
- Creating a Package Step 14-13
- What Is an ODI Tool? 14-14
- Creating an ODI Tool Step 14-15
- Tool Steps: Best Practices 14-16
- Agenda 14-17
- Sequencing Steps 14-18
- A Simple Package 14-19
- Sequencing Package Steps 14-20
- Agenda 14-21
- Executing a Package 14-22
- Agenda 14-23
- Basic Step Types 14-24
- Advanced Step Types 14-25
- Agenda 14-26
- Creating Model, Submodel, and Datastore Steps 14-27
- Models, Submodels, and Datastore Steps 14-28
- Agenda 14-30
- Creating a Variable Step 14-31
- Variable Steps 14-32
- Agenda 14-34
- Controlling Execution 14-35
- Error Handling 14-36
- Creating a Loop 14-37
- The Advanced Tab 14-38
- Quiz 14-39
- Summary 14-41
- Checklist of Practice Activities 14-42
- Practice 14-1: Creating an ODI Package 14-43
- Practice 14-2: Using ODI Packages with Variables and User Functions 14-44

15 Step-by-Step Debugger

Objectives 15-2
Agenda 15-3
Overview 15-4
Agenda 15-5
Process Overview 15-6
Starting a Session in Debug mode 15-7
Specifying Debug Properties 15-8
Control Execution Flow 15-9
Screen Step Numbering 15-10
Agenda 15-11
New Functionalities 15-12
Benefits for End Users 15-15
Agenda 15-16
Debug Toolbar 15-17
Toolbar: Current Cursor 15-18
Toolbar: Get Data 15-19
Toolbar: Step Into 15-20
Toolbar: Run to Task End 15-21
Toolbar: Run to Next Task 15-22
Toolbar: Run to Step End 15-23
Toolbar: Run to Next Step 15-24
Toolbar: Pause 15-25
Toolbar: Resume 15-26
Summary 15-27
Checklist of Practice Activities 15-28
Practice 15-1: Debugging Mappings 15-29

16 Managing ODI Scenarios

Objectives 16-2
Agenda 16-3
What Is a Scenario? 16-4
Properties of Scenarios 16-5
Agenda 16-6
Scenario-Related Tasks 16-7
Generating a Scenario 16-8
Regenerating a Scenario 16-9
Generation Versus Regeneration 16-10
Executing a Scenario from the GUI 16-11
Executing a Scenario from a Command Line 16-12
Executing a Scenario from a Package 16-13

Exporting a Scenario	16-14
Agenda	16-15
Preparing Scenarios for Deployment	16-16
Automating Scenario Management	16-17
Scheduling the ODI Scenario	16-18
Scheduling ODI Scenario with External Scheduler	16-21
Managing Schedules	16-22
Quiz	16-23
Summary	16-24
Checklist of Practice Activities	16-25
Practice 16-1: Creating and Scheduling Scenarios	16-26

17 Using Load Plans

Objectives	17-2
Should You Organize Executions with Load Plans?	17-3
What Are Load Plans?	17-4
Load Plan Editor	17-5
Load Plan Steps	17-6
Defining the Restart Behavior	17-7
Are Load Plans Substitutes for Packages or Scenarios?	17-9
Benefits of Using Load Plans	17-10
Handling Failed Load Plans	17-11
Quiz	17-12
Summary	17-13
Checklist of Practice Activities	17-14
Practice 17-1: Using Load Plans	17-15

18 Enforcing Data Quality with ODI

Objectives	18-2
Agenda	18-3
Why Data Quality?	18-4
When to Enforce Data Quality	18-5
Data Quality in Source Applications	18-6
Data Quality Control in the Integration Process	18-7
Data Quality in the Target Applications	18-8
Agenda	18-9
Data Quality Business Rules	18-10
From Business Rules to Constraints	18-11
Agenda	18-12
Data Quality System: Overview	18-13
Static and Flow Controls: Differences	18-14

Data Quality Control: Properties	18-15
Synchronous Control	18-16
What Is a Constraint?	18-17
What Can Be Checked?	18-18
Enforcing Data Quality in a Mapping	18-19
Agenda	18-20
Setting Up Static or Flow Control	18-21
Enabling Static or Flow Control	18-22
Agenda	18-23
Setting the Physical Options	18-24
Setting the Logical Options	18-25
Agenda	18-26
Selecting Which Constraints to Enforce	18-27
Selecting Which Constraints to Check	18-28
Differences Between Control Types	18-29
Agenda	18-30
Reviewing Erroneous Records	18-31
EnterpriseDataQuality Tool	18-32
Using the EDQ Tool	18-33
Quiz	18-34
Summary	18-36
Checklist of Practice Activities	18-37
Practice 18-1: Enforcing Data Quality with ODI Mappings	18-38

19 Working with Changed Data Capture

Objectives	19-2
Why Changed Data Capture?	19-3
CDC Techniques	19-4
Changed Data Capture in ODI	19-5
Journalizing Components	19-6
CDC Infrastructure in ODI	19-7
Simple Versus Consistent Set Journalizing	19-8
Limitations of Simple CDC Journalizing: Example	19-9
Consistent CDC Journalizing	19-10
Consistent CDC: Infrastructure	19-11
Setting Up Journalizing	19-12
Setting CDC Parameters: Example	19-13
Adding a Subscriber: Example	19-14
Starting Journal: Example	19-15
Journalizing Status	19-16
Viewing Data/Changed Data: Example	19-17

Using Changed Data	19-18
Oracle GoldenGate Integration	19-20
Oracle GoldenGate Integration in ODI 12c	19-21
Quiz	19-22
Summary	19-24
Checklist of Practice Activities	19-25
Practice 19-1: Implementing Changed Data Capture	19-26

20 Advanced ODI Administration

Objectives	20-2
Agenda	20-3
Introduction to ODI Security Navigator	20-4
Security Concepts: Overview	20-6
Defining Security Policies	20-8
Creating Profiles	20-9
Using Generic and Nongeneric Profiles	20-10
Built-in Profiles	20-11
Creating Users	20-12
Assigning a Profile to a User	20-13
Assigning an Authorization by Profile or User	20-14
Defining Password Policies	20-15
Setting User Preferences	20-17
ODI Security Integration: Overview	20-18
Implementing External Authentication (OPSS)	20-19
Implementing External Authentication (OPSS): Switching the Authentication Mode	20-21
Implementing External Password Storage	20-22
Agenda	20-24
Types of ODI Reports	20-25
Generating Topology Reports	20-26
Generated Topology Report: Example	20-27
Version Comparison Report: Example	20-28
Generating Object Reports	20-29
Agenda	20-30
Integration of ODI with Enterprise Manager	20-31
Java EE Agent and Enterprise Manager Configuration with WebLogic Domain: Overview	20-32
Using ODI Console: Example	20-33
Quiz	20-34
Summary	20-35
Checklist of Practice Activities	20-36

Practice 20-1: Setting Up ODI Security 20-37

Practice 20-2: Integration with Enterprise Manager and Using ODI Console 20-38

1

Introduction to Integration and Administration

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Course Objectives

After completing this course, you should be able to:

- Install Oracle Data Integrator (ODI) 12c and configure using Repository Creation Utility (RCU)
- Describe ODI architecture and apply ODI topology concepts for data integration
- Describe Oracle Data Integrator model concepts
- Design ODI mappings, procedures, and packages to perform data transformations
- Explore, audit data, and enforce data quality with ODI
- Administer ODI resources and set up security with ODI
- Implement Changed Data Capture with ODI

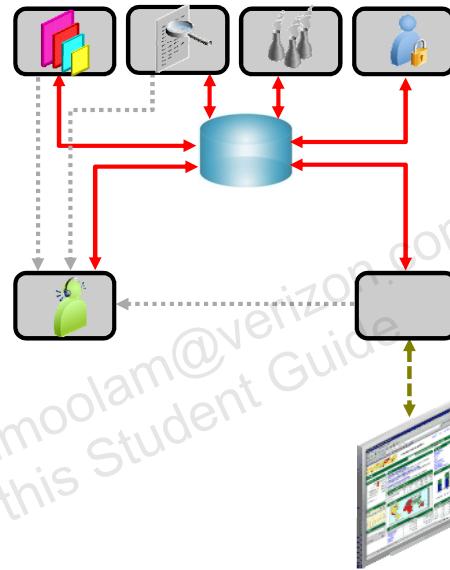


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Lesson Objectives

After completing this lesson, you should be able to:

- Describe the course objectives and agenda of the lessons
- Describe the benefits of using Oracle Data Integrator (ODI)
- Describe the ODI 12c architecture and components
- Describe how to use ODI Studio to create, administer, and monitor ODI objects
- Start ODI Studio
- Access online Help and Tutorials



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This lesson provides a general overview of the ODI architecture. You learn the roles of the different ODI components. The lesson also covers repositories—the most important component of ODI—in detail.

Agenda of Lessons

- Day 1:
 - Lesson 1: Introduction to Integration and Administration
 - Lesson 2: Administering ODI Repositories
 - Lesson 3: ODI Topology Concepts
 - Lesson 4: Describing the Physical and Logical Architecture
- Day 2:
 - Lesson 5: Setting Up a New ODI Project
 - Lesson 6: Oracle Data Integrator Model Concepts
 - Lesson 7: Organizing ODI Models and Creating ODI Datastores
 - Lesson 8: ODI Mapping Concepts



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The schedule is subject to change depending on the instructor and the class.

Agenda of Lessons

- Day 3:
 - Lesson 9: Designing Mappings
 - Lesson 10: Mappings: Monitoring and Troubleshooting
 - Lesson 11: Designing Mappings: Advanced Topics 1
 - Lesson 12: Designing Mappings: Advanced Topics 2
- Day 4:
 - Lesson 13: Using ODI Procedures
 - Lesson 14: Using ODI Packages
 - Lesson 15: Step-by-Step Debugger
 - Lesson 16: Managing ODI Scenarios



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The schedule is subject to change depending on the instructor and the class.

Agenda of Lessons

- Day 5:
 - Lesson 17: Using Load Plans
 - Lesson 18: Enforcing Data Quality with ODI
 - Lesson 19: Working with Changed Data Capture
 - Lesson 20: Advanced ODI Administration



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The schedule is subject to change depending on the instructor and the class.

Agenda

- Oracle Data Integrator: Introduction
 - Architecture
 - Components
- Oracle Data Integrator Repositories



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This section covers Oracle Data Integrator's approach to data integration, including its architecture and modular components.

Why Oracle Data Integrator?

- E-LT architecture provides high performance.
 - E-LT faster than ETL
- Active integration enables real-time data warehousing and operational data hubs.
 - Changed Data Capture technology for real-time data warehousing
 - Data services provided to the Oracle SOA Suite
- Declarative design improves developer productivity.
 - Business users specify what they want; ODI generates the flows and code.
- Knowledge Modules provide flexibility and extensibility.
 - Predefined, reusable code templates with built-in connectivity to all major databases



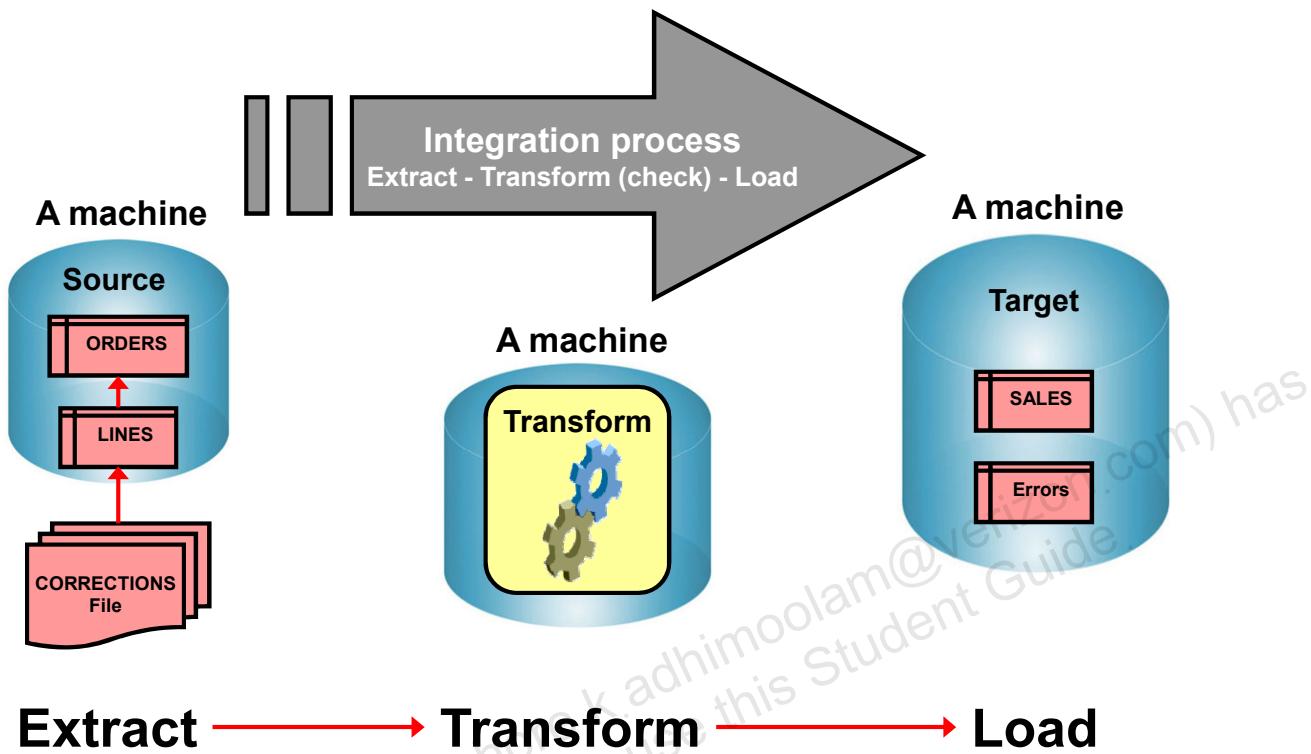
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- ODI's E-LT architecture leverages disparate RDBMS engines to process and transform data. This approach optimizes performance and scalability, and lowers overall solution costs.
- ODI turns the promise of active integration into reality by providing all the key components that are required to enable real-time data warehousing and operational data hubs. ODI combines three styles of data integration: data-based, event-based, and service-based. ODI unifies silos of integration by transforming large volumes of data in batch mode, by processing events in real time through its advanced Changed Data Capture, and by providing data services to the Oracle SOA Suite.
- Oracle Data Integrator shortens implementation times with its declarative design approach. Designers specify what they want to accomplish with their data, and then the tool generates the details of how to perform the task. With ODI, the business user or the developer specifies the rules to apply to the integration processes. The tool automatically generates data flows and administers correct instructions for the various source and target systems. With declarative design, the number and complexity of steps is greatly reduced, which in turn shortens implementation times. Automatic code generation reduces the learning curve for integration developers.

- Knowledge Modules are at the core of the ODI architecture. They make all ODI processes modular, flexible, and extensible. Knowledge Modules implement the actual data flows and define the templates for generating code across the multiple systems involved in each process. ODI provides a comprehensive library of Knowledge Modules, which can be tailored to implement existing best practices (for example, for highest performance, for adhering to corporate standards, or for specific vertical knowhow). By helping companies capture and reuse technical expertise and best practices, ODI's Knowledge Module framework reduces the cost of ownership. It also enables metadata-driven extensibility of product functionality to meet the most demanding data integration challenges.

ODI streamlines the high-performance movement and transformation of data between heterogeneous systems in batch, real-time, synchronous, and asynchronous modes. It dramatically enhances user productivity with an innovative, modularized design approach and built-in connectivity to all major databases, data warehouse appliances, analytic applications, and SOA suites.

Conventional Integration Process: ETL



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

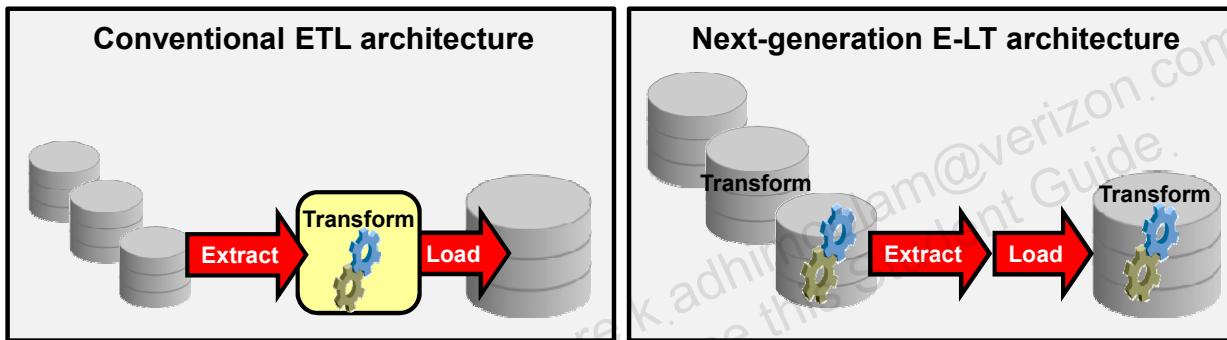
This integration process is also known as an extract, transform, and load (ETL) process.

- The first part of an ETL process involves extracting data from the source systems. Most data warehousing projects consolidate data from different source systems.
- The transform stage applies a series of rules or functions to the data extracted from the source to derive the data for loading into the target. Some data sources require very little or even no manipulation of data. In other cases, transformations (such as filtering, joining, sorting, and so on) may be required to meet the business and technical needs of the target database.
- The load phase loads the data into the target, usually the data warehouse.

Note: You can add to this process the checks that ensure the quality of data flow, as shown in the slide.

Extract Load Transform (E-LT)

1. **Extract:** Extracting data from various sources
2. **Load:** Loading the data into the destination target
3. **Transform:** Transforming the data according to a set of business rules



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Data is one of the most important assets of any company, and data integration constitutes the backbone of any enterprise's IT systems. Choosing the technology for data integration is critical for productivity and responsiveness of business divisions within an enterprise.

E-LT stands for extract, load, and transform. It includes the processes that enable companies to move data from multiple sources, reformat and cleanse the data, and load it into another database, or a data warehouse for analysis, to support a business process.

ODI provides a strong and reliable integration platform for IT infrastructure. Built on the next-generation architecture of extract, load, and transform (E-LT), ODI delivers superior performance and scalability connecting heterogeneous systems at a lower cost than traditional, proprietary ETL products. Unlike the conventional extract, transform, and load (ETL) design, with ODI, E-LT architecture extracts data from sources, loads it into a target, and transforms it by using the database power according to business rules. The tool automatically generates data flows, manages their complexity, and administers correct instructions for the various source and target systems.

Extract

The first step in the E-LT process is extracting data from various sources. Each of the source systems may store its data in a completely different format. The sources are usually flat files or RDBMS, but almost any data storage can be used as a source for an E-LT process.

Load

This step involves loading the data into the destination target, which might be a database or data warehouse.

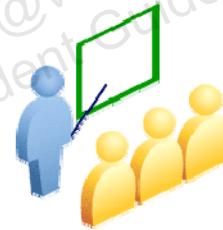
Transform

After the data has been extracted and loaded, the next step is to transform the data according to a set of business rules. The data transformation may involve various operations including, but not limited to, filtering data, sorting data, aggregating data, joining data, cleaning data, generating calculated data based on existing values, and validating data.

ODI Architecture and Components

For more information, see the *Installation Guide* and *User's Guide*.

To find ODI documentation, go to otn.oracle.com/goto/odi



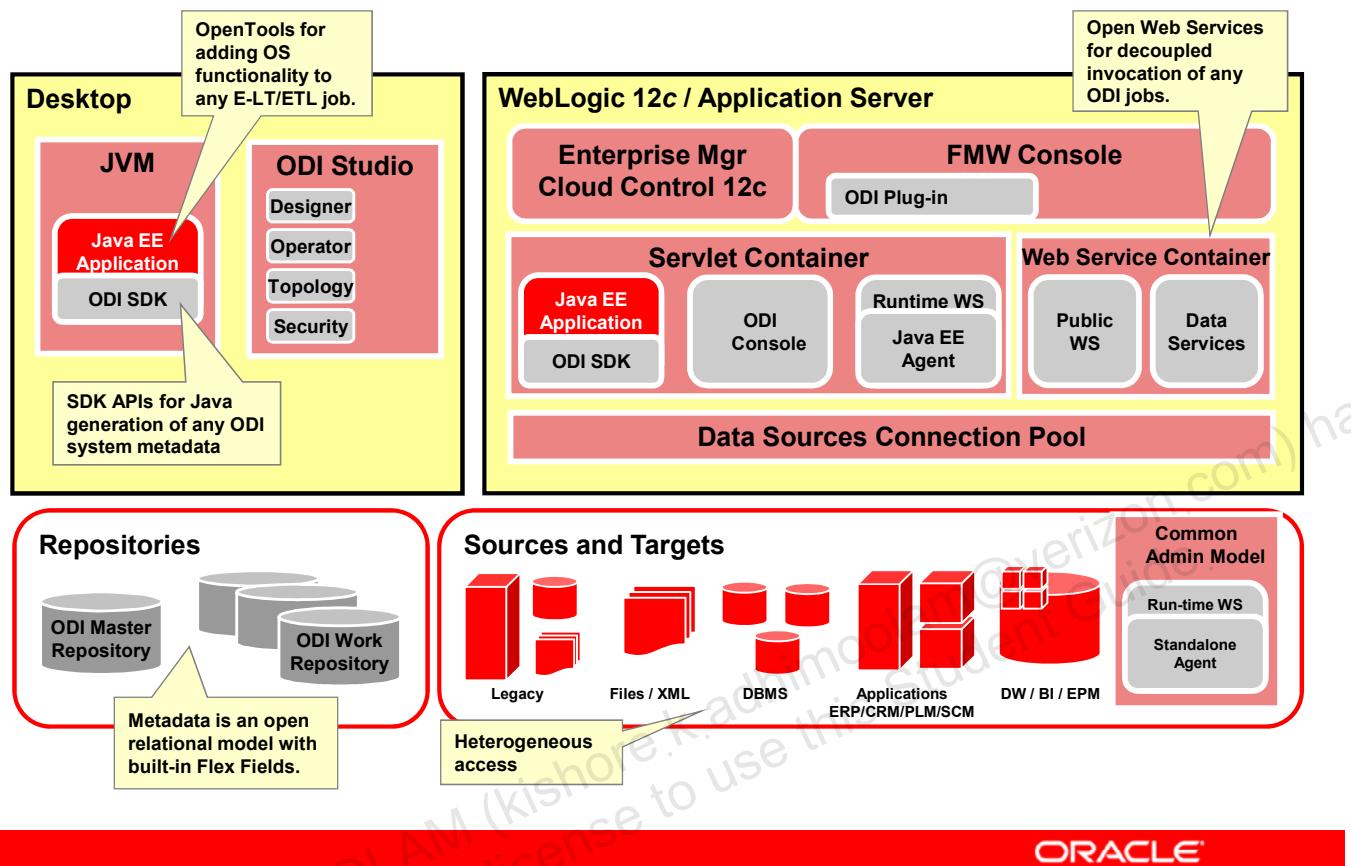
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Documentation is available in many formats: HTML, PDF, and ePub (Kindle).

To fully understand the ODI architecture, you must look at each of its components.

ODI Architecture



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The repository forms the central component of the ODI architecture. It stores configuration information about the IT infrastructure—the metadata for all applications, projects, scenarios, and execution logs. Repositories can be installed in an online transaction processing (OLTP) relational database. The repository also contains information about the ODI infrastructure, defined by the administrators. The two types of ODI repositories are Master and Work Repositories.

At design time, developers work in a repository to define metadata and business rules. The resulting processing jobs are executed by the agent, which orchestrates the execution by leveraging existing systems. The agent connects to available servers and asks them to execute the code. It then stores all return codes and messages in the repository. The agent also stores statistics, such as the number of records processed, and the elapsed time. Several repositories can co-exist in an IT infrastructure. The graphic in this slide shows two repositories: one for the development environment and the other for the production environment. Developers release their projects in the form of scenarios that are sent to production.

In production, these scenarios are scheduled and executed on a Scheduler Agent that also stores all its information in the repository. Operators have access to this information and can monitor the integration processes in real time.

Business users as well as developers, administrators, and operators, can gain web-based read access to the repository by using the ODI Console.

ODI Components: Overview

- ODI Studio components:



- ODI agents
- ODI Console
- ODI repositories

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

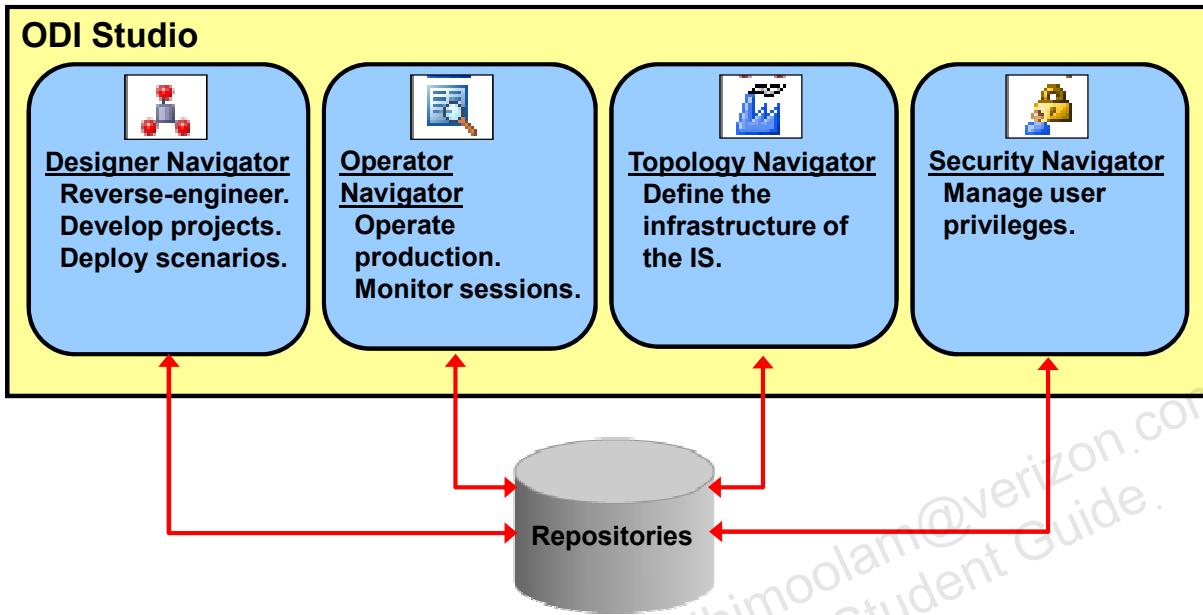
ODI Studio provides four navigators for managing the different aspects and steps of an ODI integration project:

- Designer Navigator
- Topology Navigator
- Operator Navigator
- Security Navigator

The navigators are discussed in detail in subsequent slides. In addition to the navigators are several other components:

- ODI agents are runtime processes that orchestrate executions.
- ODI Console provides users with web access to ODI metadata.
- ODI repositories store all of your ODI objects as databases in a relational database management system.

Using ODI Studio



The Fusion Client Platform (FCP)–based UI provides an efficient and flexible way to manage navigators, panels, and editors.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

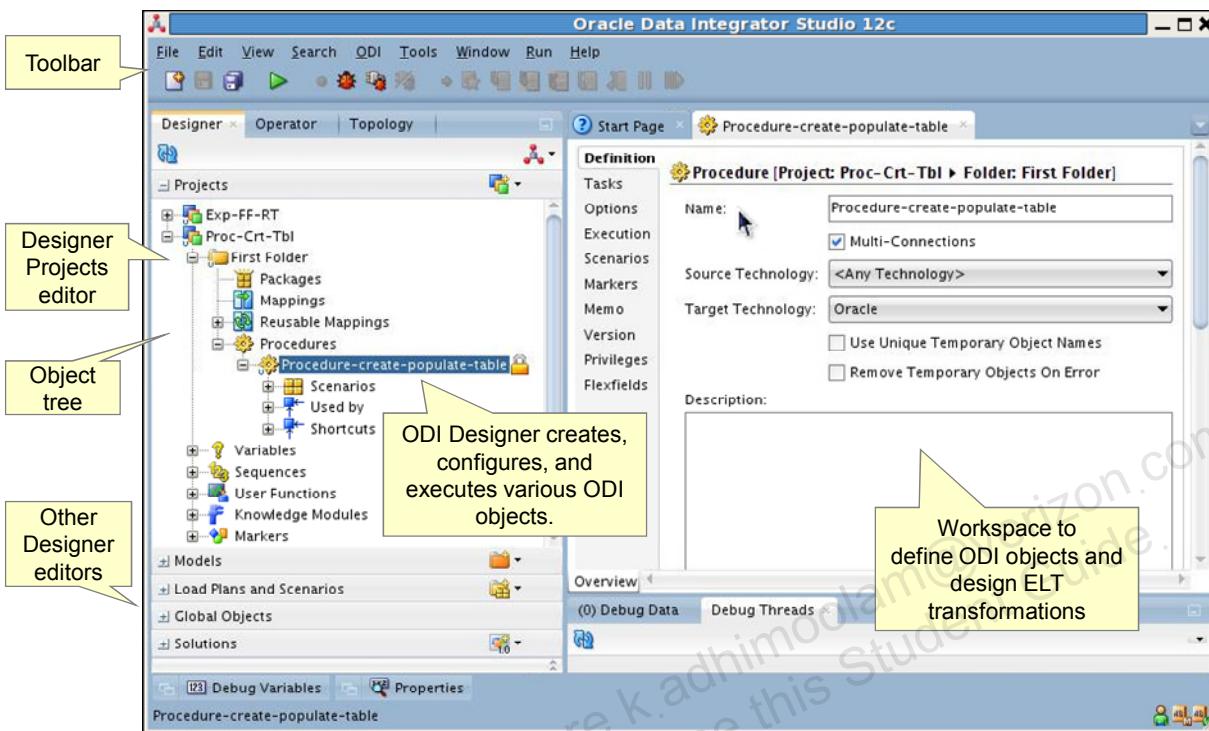
Graphical Navigators

Administrators, developers, and operators use Oracle Data Integrator Studio to access the repositories. This Fusion Client Platform (FCP)–based UI is used for administering the Information System (IS) infrastructure (security and topology), reverse-engineering the metadata, developing projects, and scheduling, operating, and monitoring executions. FCP provides an efficient and flexible way to manage navigators, panels, and editors.

Business users (as well as developers, administrators, and operators) can have read access to the repository. They can also perform topology configuration and production operations through a web-based UI called the Oracle Data Integrator Console. This web application can be deployed in a Java EE application server such as Oracle WebLogic Server (WLS).

The four ODI graphical navigators are based on the Java programming language and can be installed on any platform that supports Java Virtual Machine 1.7 or higher, including Windows, Linux, HP-UX, Solaris, and pSeries. All ODI navigators store their information in the centralized ODI repository.

Designer Navigator (Work Repository)



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

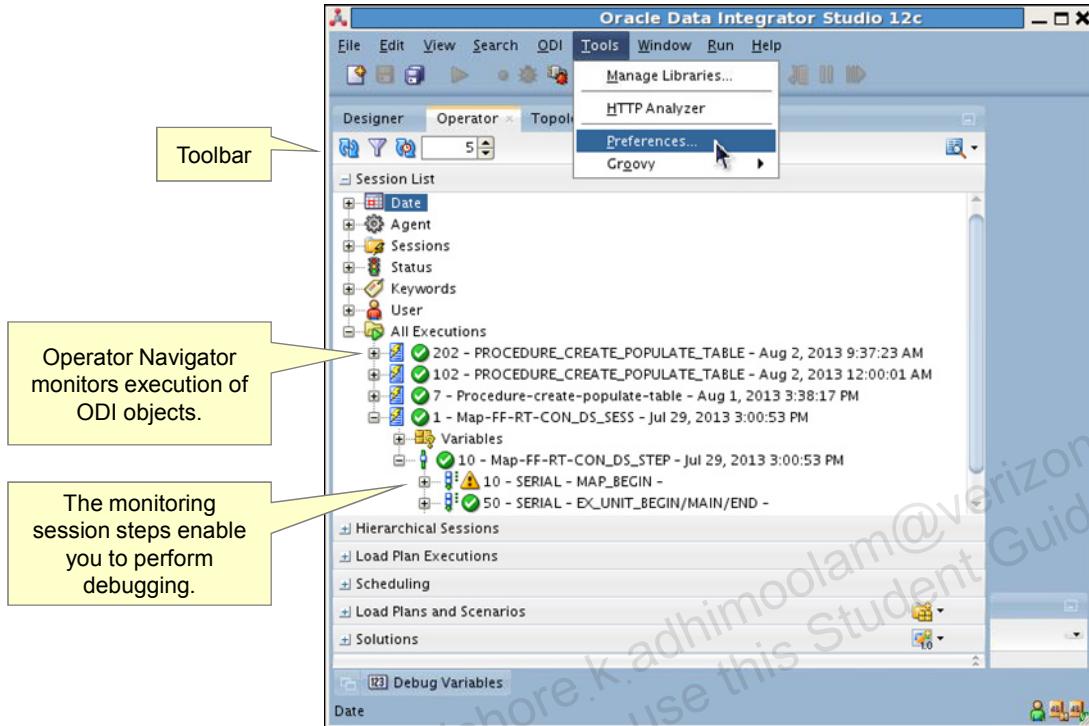
Designer Navigator is the GUI for defining metadata and rules for transformation and data quality. ODI uses this information to generate scenarios for production, and is where all the project development takes place. Designer Navigator is the core module for developers and metadata administrators.

Through the Designer Navigator, you can handle the following:

- **Models:** Descriptions of data and application structures
- **Projects:** Development of various ODI objects

Note: Designer Navigator stores this information in a Work Repository, while using the topology and the security information defined in the Master Repository.

Operator Navigator (Work Repository)



ORACLE

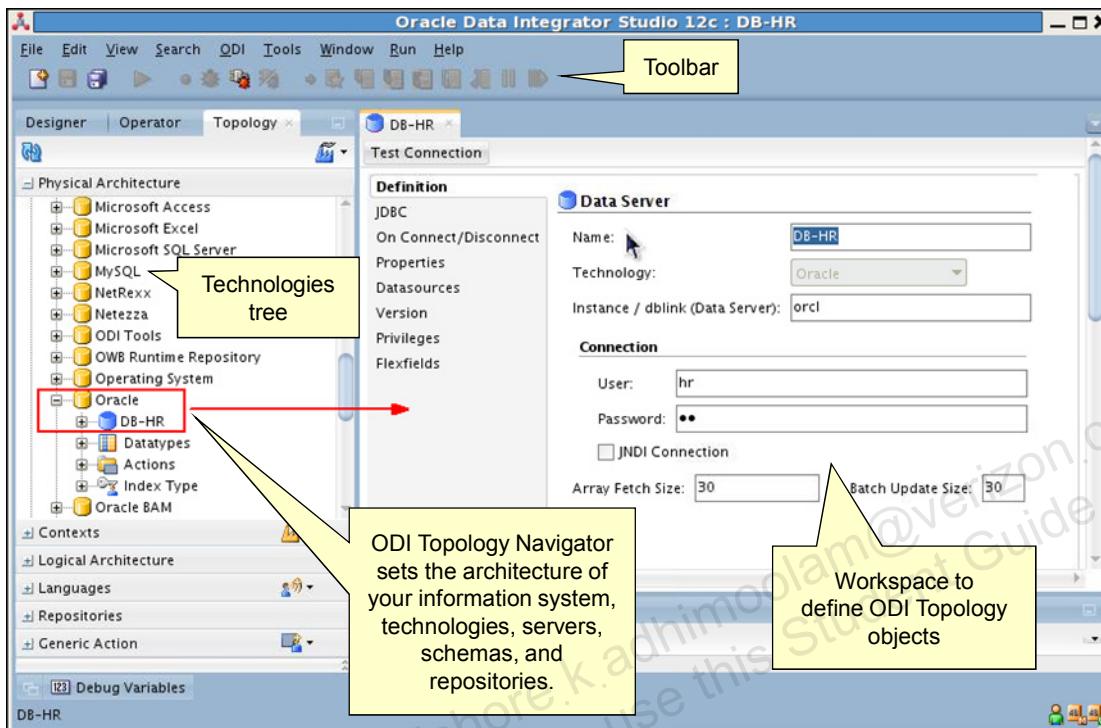
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Operator Navigator is used to manage and monitor ODI in production. It is designed for production operators, and displays the execution logs with error counts, the number of rows processed, execution statistics, and so on. At design time, developers use the Operator Navigator for debugging purposes.

Through the Operator Navigator, you can manage your mapping executions in the sessions, as well as the scenarios in production.

The Operator Navigator stores this information in a Work Repository, while using the topology defined in the Master Repository.

Topology Navigator (Master Repository)



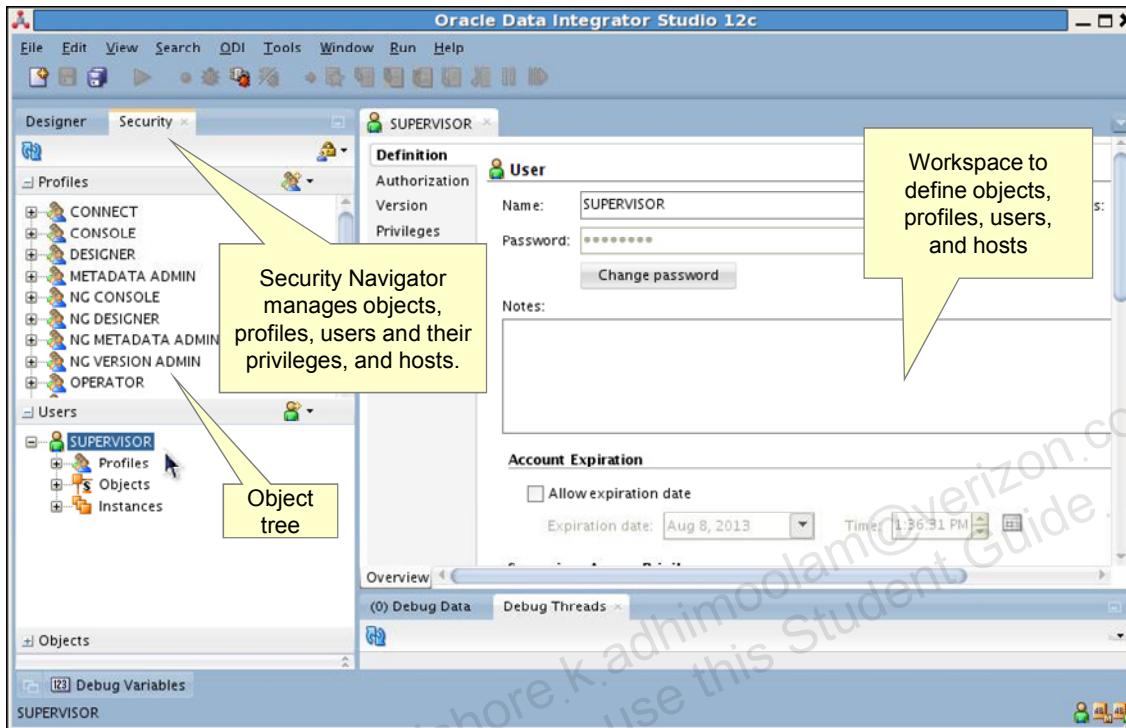
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Topology Navigator manages the physical and logical architecture of the infrastructure. Servers, schemas, and agents are registered in the ODI Master Repository—a major ODI component that contains information about the topology of the company's IT resources, security, and ODI resources that will be discussed later in this course.

Using the Topology Navigator, you can define the topology of your information system to ODI so that it can be accessed by other ODI modules. In addition, the Topology Navigator enables you to manage the repositories. The Topology Navigator stores this information in a Master Repository. This information can be used by all the other modules.

Security Navigator (Master Repository)



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Security Navigator manages users and their privileges in ODI. It is used to create profiles and provide rights to users to access ODI objects and features. This navigator is usually used by security administrators.

It is used to assign user rights for methods (edit and delete, for example) on generic objects (data server and data types, for example), and to fine-tune these rights on the object instances (Server 1 and Server 2, for example).

The Security Navigator stores this information in a Master Repository. This information can be used by all the other modules.

What Is an Agent?

- An agent is a runtime component of ODI that orchestrates the integration process.
- It is a lightweight Java program that retrieves code from the repository at run time.
- At design time, developers generate scenarios from the business rules that they have designed. The code of these scenarios is then retrieved from the repository by the agent at run time.
- This agent then connects to the data servers, and orchestrates the code execution on these servers.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

An agent is a special ODI component that runs in the background.

At design time, developers generate scenarios from the business rules that they have designed.

The code of these scenarios is then retrieved from the repository by the agent at run time. This agent then connects to the data servers, and orchestrates the code execution on these servers.

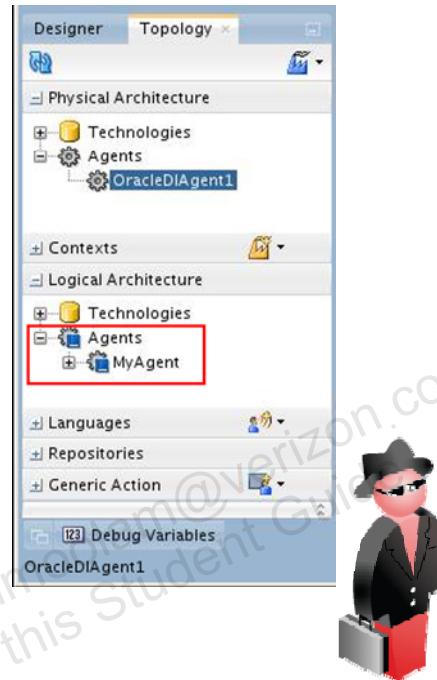
The agent orchestrates the integration process by sending commands to data servers, the operating system, or other technologies. It retrieves the return codes and messages for the execution, as well as additional logging information, such as the number of rows processed, execution time, and so on, in the repository.

The agent does not require ODI Studio to run, but does require ODI Studio to be configured.

ODI Agents

Agents:

- Are lightweight Java processes that orchestrate the execution of objects at run time
- Can do one of the following:
 - Execute objects on demand
 - Execute according to predefined schedules



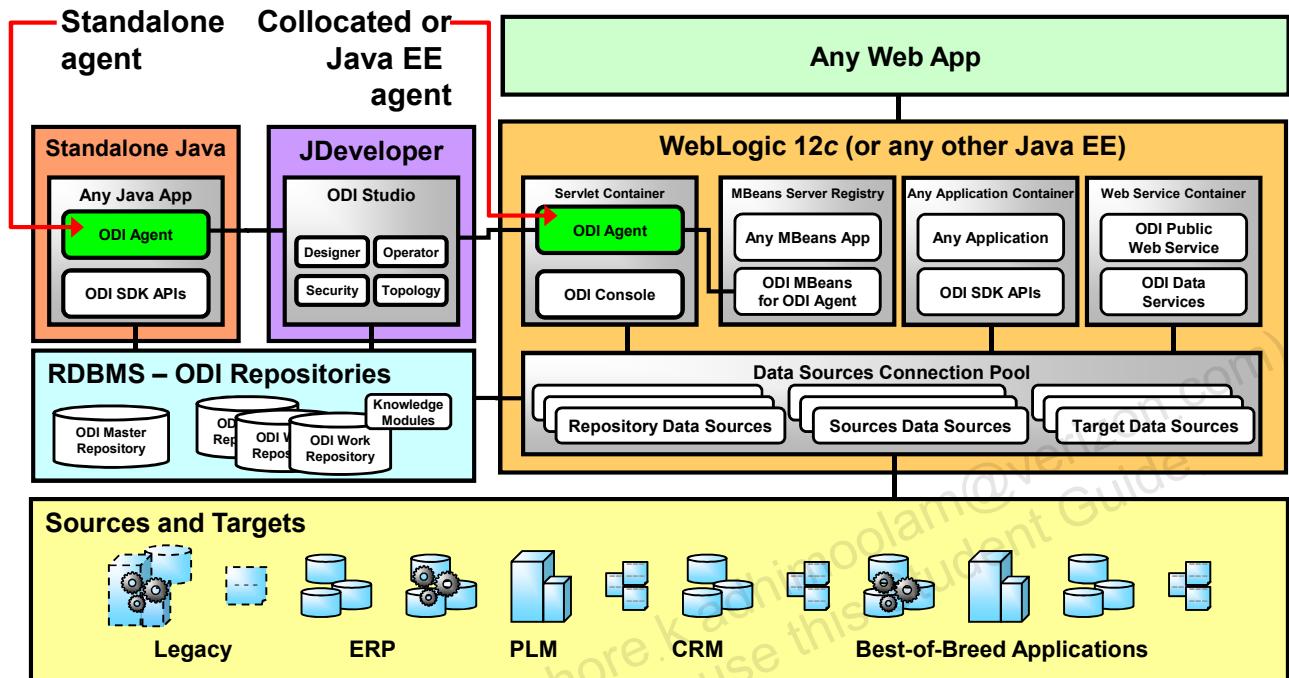
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can install an agent on any machine with network access. This is worth bearing in mind when planning your deployment strategy.

Note: Before ODI 11g, ODI had two types of agents: listener agents and scheduler agents. With ODI 12c, agents are always connected to a Master Repository, and are started with the built-in scheduler service activated. This scheduler service takes its schedules from all the Work Repositories attached to the connected Master.

Three Types of Agents: Java EE, Standalone, Collocated Standalone



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Agents come in three types:

- The **Java Enterprise Edition** (Java EE; formerly J2EE) agent can be deployed as a web application and benefit from the features of an application server, for example, WLS.
- The **standalone** agent runs in a simple Java Machine and can be deployed where needed to perform the integration flows.
- The **Collocated standalone** agents can be installed on the source or target systems as well. They can be managed by using Oracle Enterprise Manager (EM) and must be configured with an Oracle WebLogic domain. Collocated standalone agents can run on a separate machine from the Oracle WebLogic Administration Server. Even though it uses WLS code, it is managed through ODI or EM, not WLS.

All the agents are multithreaded Java programs that support load balancing (discussed in the lesson titled “Administering ODI Repositories”) and can be distributed across the information system. An agent can hold its own execution schedule, which can be defined in ODI, and can also be called from an external scheduler. It can also be invoked from a Java API or a web service interface.

Using the Three Types of Agents

- Deploying a Java EE agent in a Java EE Application Server (Oracle WebLogic Server):
 1. In ODI, define the Java EE agent in the Topology Navigator.
 2. In ODI, create the WLS template for the Java EE agent.
 3. Deploy the template directly by using the WLS Configuration Wizard.
- Using a standalone agent:
 1. Launch an agent.
 2. Display scheduling information.
 3. Stop the agent.
- Advantages of Java EE agents over standalone agents:
 - High availability (WLS NodeManager + Clusters)
 - Multiple agents, using Coherence
 - Load balancing
 - Connection pooling back to repositories



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Deploying an Oracle Data Integrator Agent in Oracle WebLogic Server (WLS)

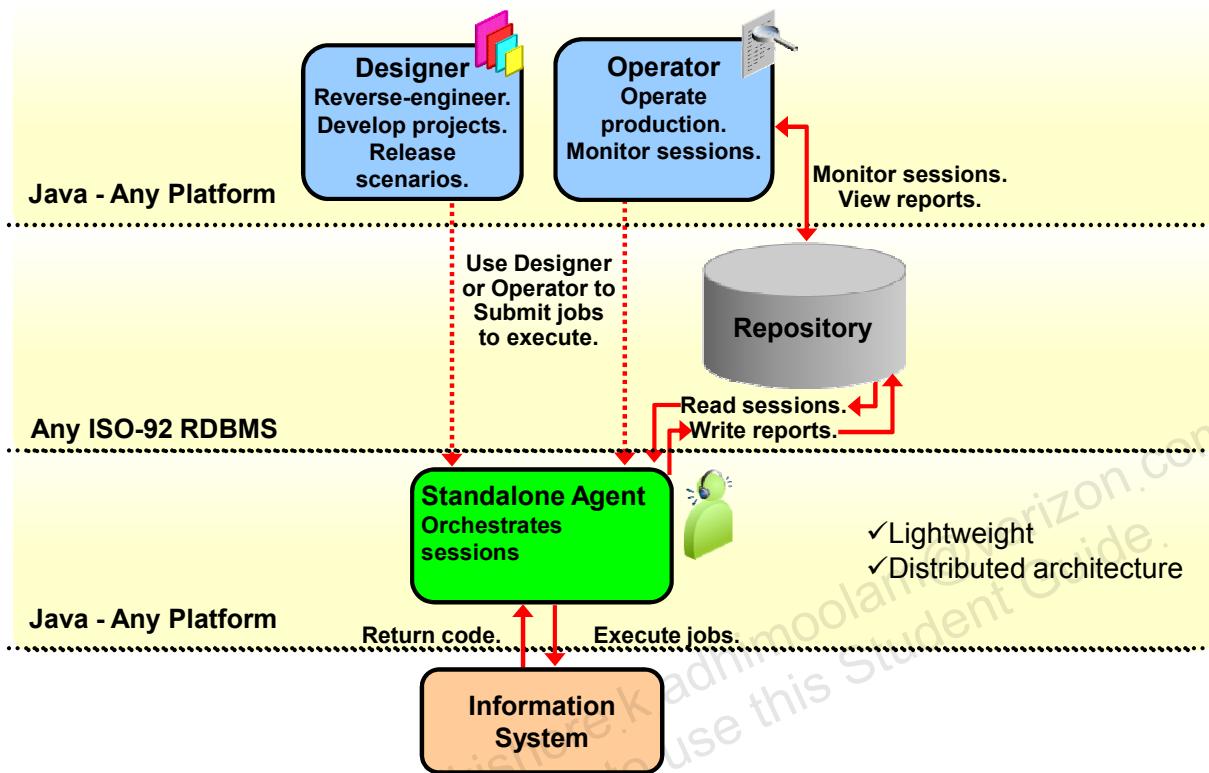
The easiest way to deploy an Oracle Data Integrator agent in Oracle WebLogic Server (WLS) is to generate a WLS template with Oracle Data Integrator. This template can directly be deployed by using the WLS Configuration Wizard.

Deploying an agent in a Java EE Application Server (Oracle WebLogic Server) involves the following tasks:

- **Task 1:** Define the Java EE agent in the Topology. Defining a Java EE agent consists of two tasks.
 - First, you need to create the physical agent corresponding to your Java EE agent.
 - Then create a logical agent.
- **Task 2:** Create the WLS template for the Java EE agent. Oracle Data Integrator provides a WLS Template Generation Wizard to help you create a WLS template for a runtime agent.
- **Task 3:** Deploy the template directly by using the WLS Configuration Wizard.

The next slide discusses using a standalone agent.

Standalone Agent: Example



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

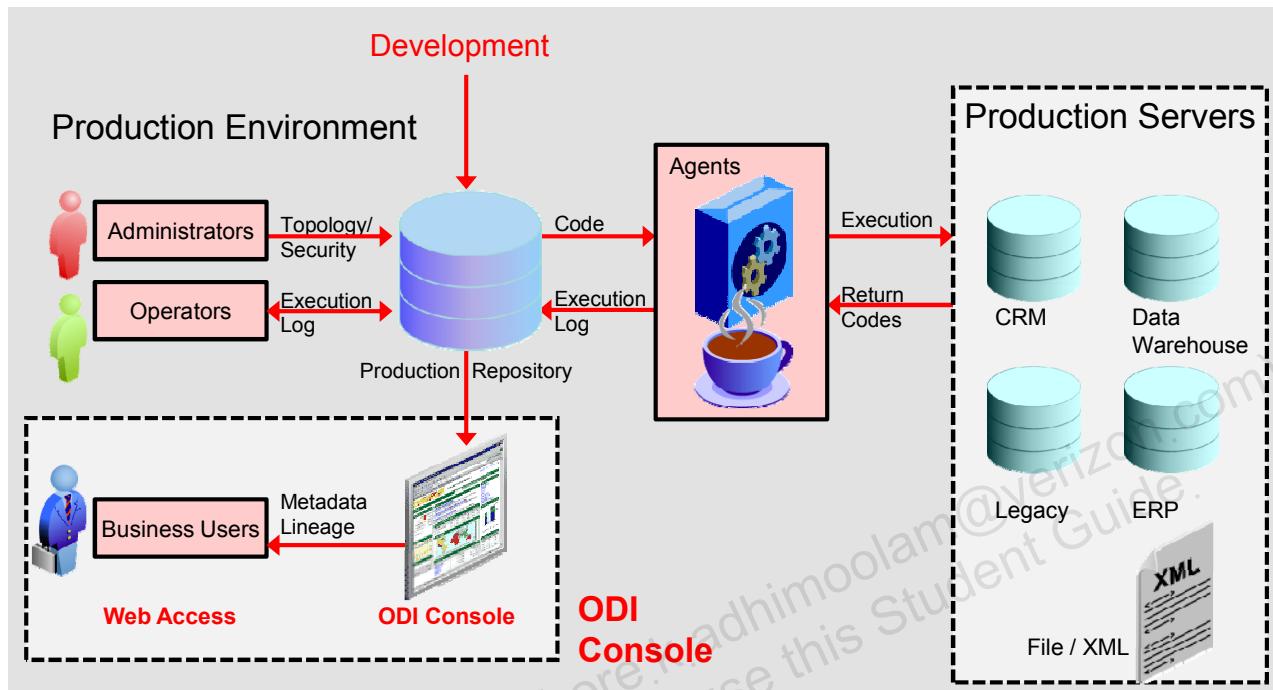
At run time, a standalone agent orchestrates the execution of the developed scenarios. It can be installed on any platform provided that it supports a JVM 1.7 at minimum (Windows, Linux, HP-UX, Solaris, pSeries, iSeries, zSeries, and so on).

Execution may be from one of the graphical modules or by using the built-in scheduler.

Due to the ELT architecture of ODI, the standalone agent rarely performs transformation itself. It usually retrieves code from the execution repository and requests database servers, operating systems, or scripting engines to execute it. When the execution is completed, the standalone agent updates the logs in the repository, reporting error messages and execution statistics. The execution log can be viewed from the Operator Navigator.

Note: Although it can act as a transformation engine, the agent is rarely used for this purpose. Agents are installed at tactical locations in the information system to orchestrate integration processes and leverage existing systems. Agents are lightweight components in this distributed integration architecture.

ODI Console



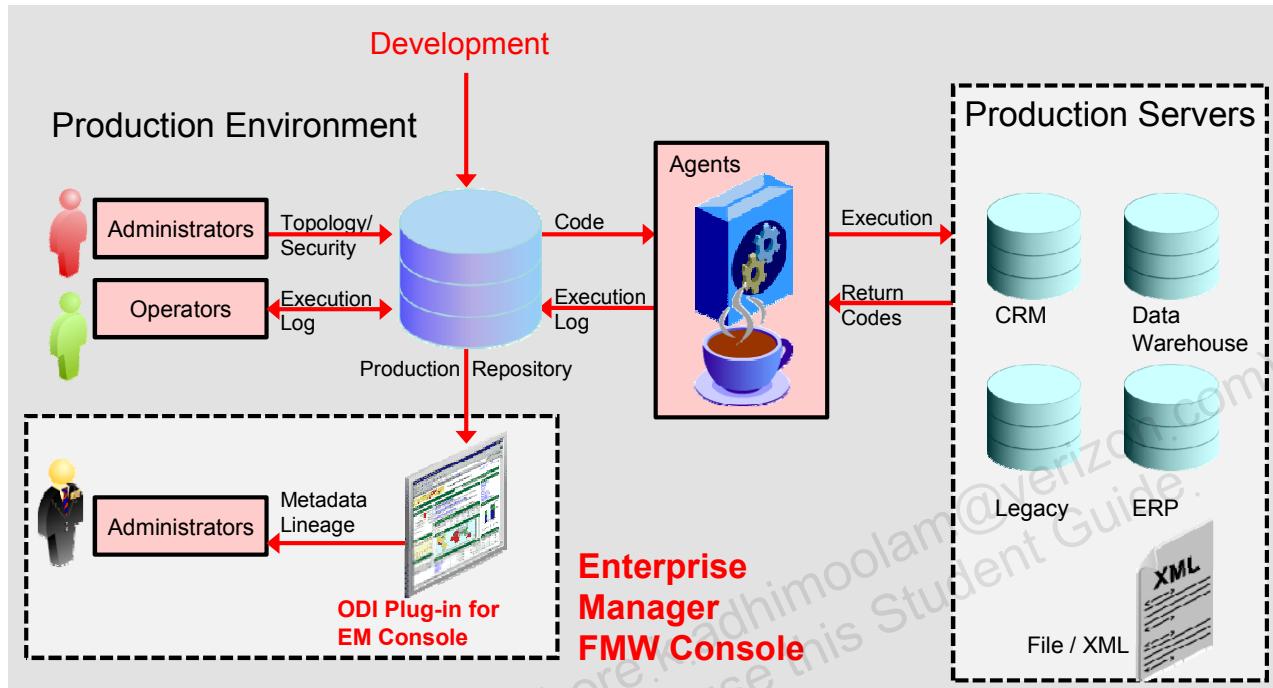
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The ODI Console provides web access to ODI repositories. It enables users to navigate projects, models, logs, and so on. Business users, developers, operators, and administrators use their web browsers to access the ODI Console. The ODI Console replaces the Metadata Navigator of ODI releases before ODI 11g.

Note that with the ODI Console, you also can perform executions. The ODI Console will be installed in the practices for the last lesson, “Advanced ODI Administration.”

Enterprise Manager FMW Console



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Data Integrator provides an extension that is integrated into the Enterprise Manager Fusion Middleware Control Console. The Oracle Data Integrator components can be monitored as a “domain server” (a WebLogic Server term) through this Console, and administrators can have a global view of these components, along with other Fusion Middleware components from a single administration console.

The Enterprise Manager FMW Console will be installed in the practices for the last lesson, “Advanced ODI Administration.”

Management Pack for ODI for Enterprise Manager Cloud Control

- Discover, manage, and monitor the performance of an entire ODI infrastructure.
 - Agents
 - Repositories
 - Source and target DBs
- Gain end-to-end execution monitoring and drill-down capabilities into Oracle database activity.
- Leverage advanced service-level management capabilities.
- Track and compare configuration changes with out-of-the-box configuration management features.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Benefits

- Provides visibility into complex ODI deployments across the enterprise
- Reduces the costs associated with monitoring the health of the overall ODI infrastructure
- Minimizes troubleshooting and performance tuning effort
- Enables efficient monitoring of service-level agreement (SLA) compliance using powerful alerting capabilities

Management Pack for ODI for EM CC ODI Home Page

The screenshot shows the Oracle Enterprise Manager Cloud Control 12c Home Page for Oracle Data Integrator. The page is divided into several sections:

- Master Repositories Health:** Shows status for Master Repositories (WORKREP3, WORKREP) and Work Repositories.
- ODI Agents Health:** Shows status for Agents (Status, Undiscovered, with incidents).
- Work Repositories Health:** Shows status for Work Repositories (Status, Undiscovered, with incidents).
- Data Servers Health:** Shows status for Data Servers (Status, Undiscovered, with incidents).
- Load Plan Executions/Sessions Quick Links:** Provides quick access to sessions with errors, showing counts for Sessions With Errors, Sessions With Error Records, LPEs With Errors, and LPEs With Error Records.

Yellow callout boxes highlight specific monitoring areas:

- Master Repositories monitoring (overlays the Master Repositories Health section)
- Work Repositories monitoring (overlays the Work Repositories Health section)
- ODI agents monitoring (overlays the ODI Agents Health section)
- Sources and targets monitoring (overlays the Data Servers Health section)
- Quick access to sessions with errors (overlays the Load Plan Executions/Sessions Quick Links section)

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The functionality shown here may require other plug-in packs as well. More information about prerequisites can be found at <http://www.oracle.com/us/products/middleware/data-integration/management-pack-for-odi/overview/index.html>

Agenda

- Oracle Data Integrator: Introduction
- **Oracle Data Integrator Repositories**



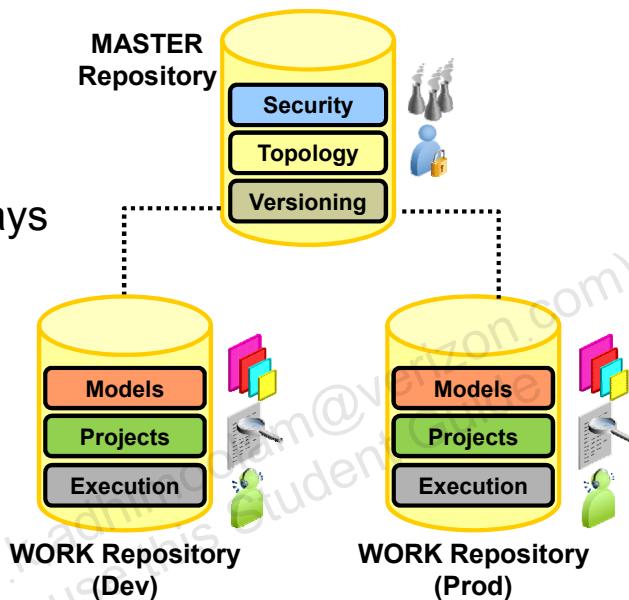
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A repository is the central component in ODI. The following slides cover which repository stores what information.

ODI Repositories

- Two types of repositories are included in ODI:
 - Master Repository
 - Work Repository
 - Development repository
 - Execution repository
- Work Repositories are always attached to a single Master Repository.



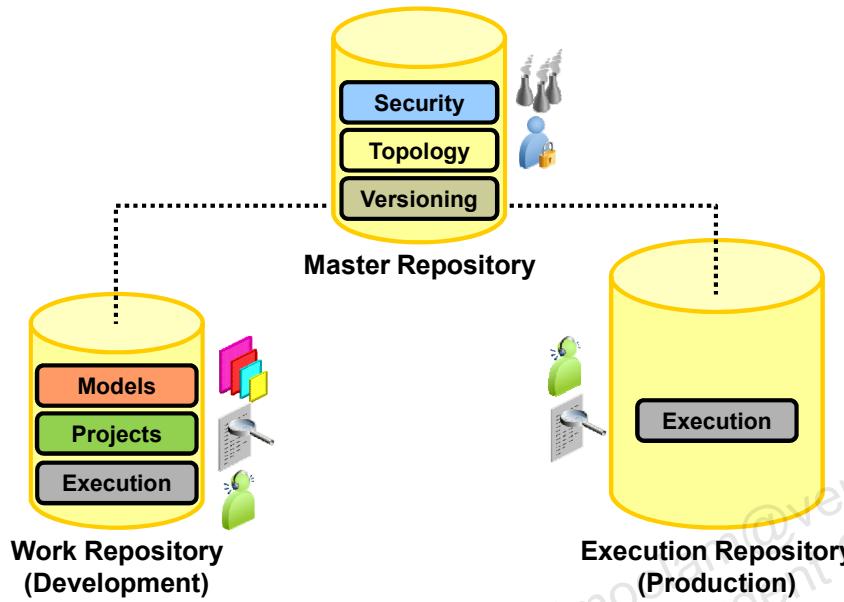
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ODI repositories are databases stored in relational database management systems. There are two types of ODI repositories: Master Repository and Work Repository. All the objects configured, developed, or used by the ODI modules are stored in one of these two types of repositories. The repositories are accessed in client/server mode by various components of the ODI architecture.

Whereas a Master Repository is usually associated with multiple Work Repositories, each Work Repository can belong to only one Master Repository. This restriction supports ODI version management.

Master and Work Repositories



You can create a Work Repository that stores only execution information, usually for PRODUCTION purposes. This type of Work Repository can be called an execution repository.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Two types of repositories are included in ODI:

- **Master Repository:** It is the data structure containing information about the topology of the company's IT resources, security, and version management of projects and data models. This repository is stored on a relational database accessible in client/server mode from the different ODI modules. In general, you need only one Master Repository.
- **Work Repository:** It is the data structure containing information about data models, projects, and their use. This repository is stored on a relational database accessible in client/server mode from the different ODI modules.

Several Work Repositories can be designated with several Master Repositories, if necessary. However, a Work Repository can be linked with only one Master Repository for version management purposes. The ODI Repository comprises a Master Repository and several Work Repositories. There is usually only one Master Repository, which contains the following information:

- Security information, including users, profiles, and access privileges for the ODI platform
- Topology information, including technologies, definitions of servers and schemas, contexts, and languages
- Old versions of objects

The information contained in the Master Repository is maintained with the Topology Navigator and the Security Navigator. All modules access the Master Repository because they all need the topology and security information stored there.

The Work Repository is where projects are developed. Several Work Repositories may co-exist in the same ODI installation. This is useful, for example, in maintaining separate environments or in reflecting a particular versioning life cycle.

A Work Repository stores information for the following:

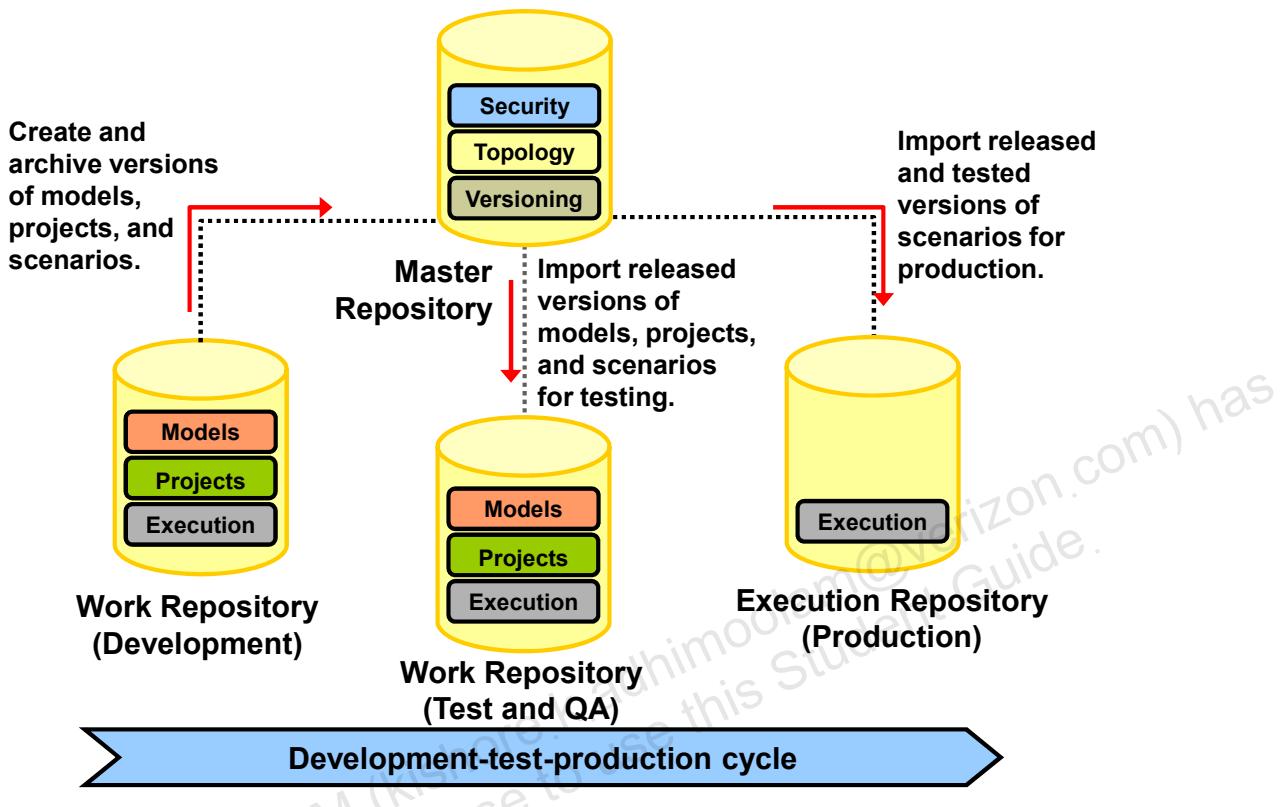
- Data models, which include the descriptions of schemas, datastore structures and metadata, fields and columns, data quality constraints, cross-references, data lineage, and so on
- Projects, which include business rules, packages, procedures, folders, Knowledge Modules, variables, and so on
- Execution, which means scenarios, scheduling information, and logs

The contents of a Work Repository are managed by using Designer and Operator. They are also accessed by the agent at run time.

When a Work Repository is used only to store execution information (typically for production purposes), it is called an execution repository. Execution repositories are accessed at run time by using Operator and also by using agents.

Note: Work Repositories are always attached to one Master Repository.

Repository Setup: Example

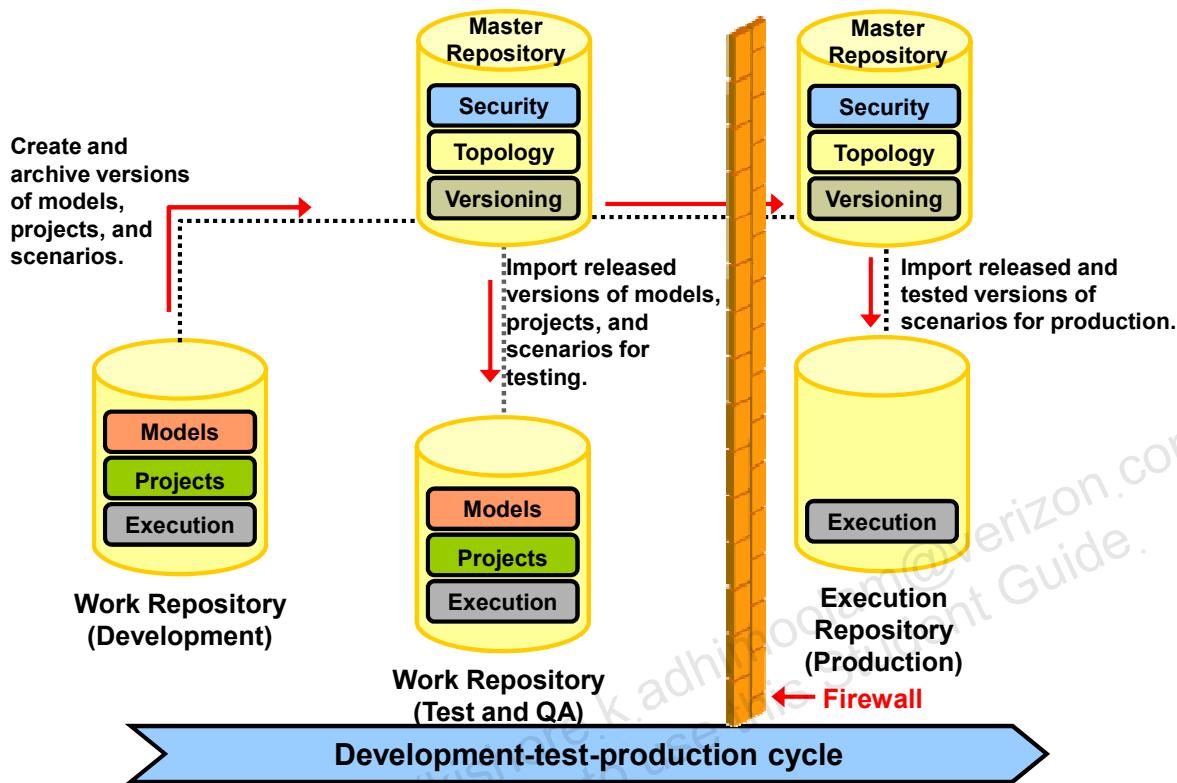


ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The graphic in the slide gives an overview of a typical repository architecture where development, testing, and production are carried out in separate Work Repositories. When the development team finishes work on certain projects, it releases them into the Master Repository. The testing team imports these released versions for testing in a separate Work Repository, thereby allowing the development team to continue working on the subsequent versions. When the test team successfully validates the developed items, the production team exports the executable versions (called scenarios) into the final production Work Repository. This repository structure corresponds to a simple development-test-production cycle.

Repository Setup: Multiple Master Repositories

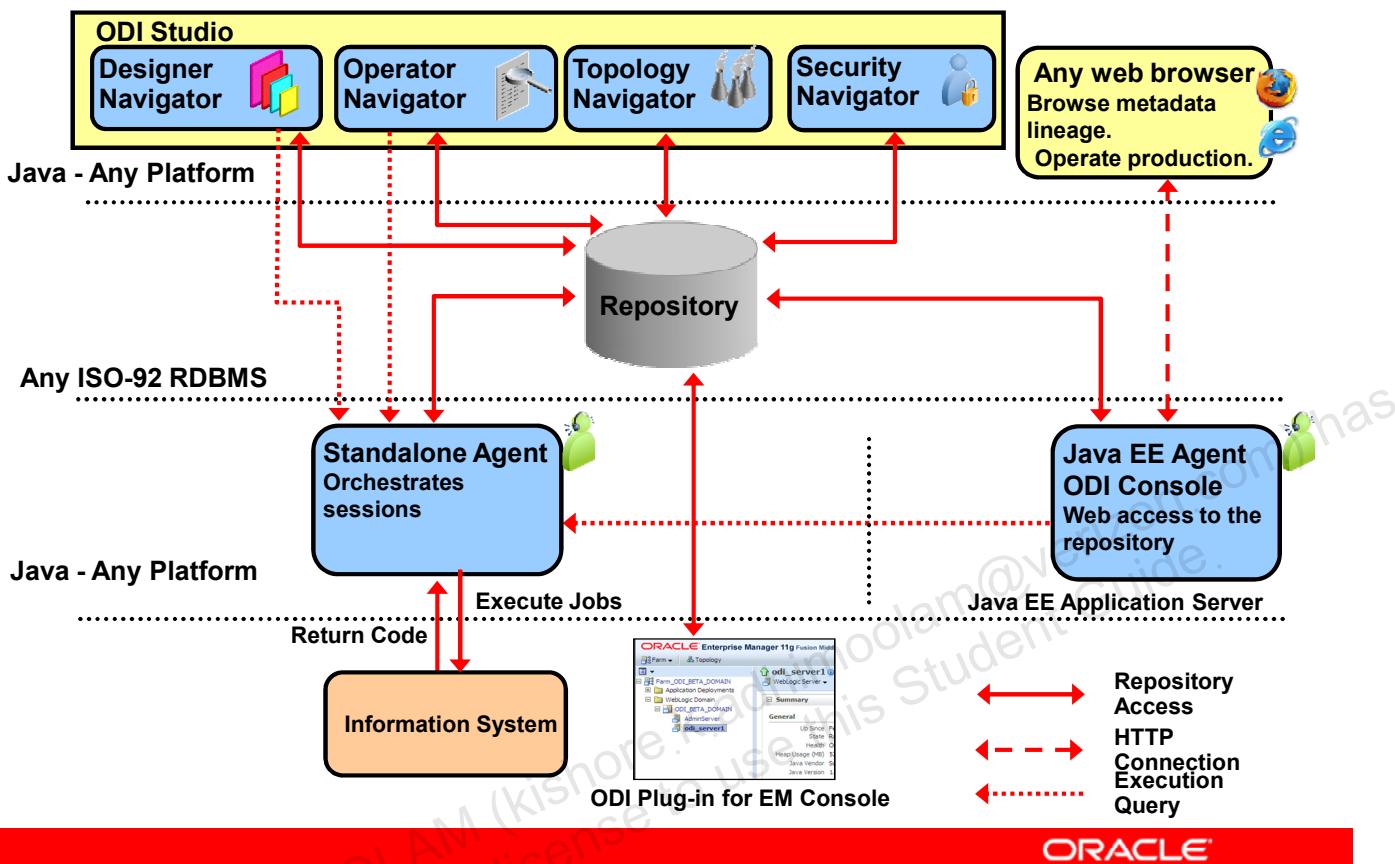


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

Imagine that a firewall protects Production from Development/Test/QA. This diagram shows that a second Master Repository can be used to deal with the firewall.

Components: Global View



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

You now have a global view of the components that make up ODI:

- Graphical components
- Repositories
- Agents
 - Standalone
 - Java EE agent
- ODI Console / ODI Studio

Possible ODI Methodology

Install	1. Install the GUI. 2. Create the Master and Work Repositories. 3. Define users and profiles.
Security	4. Define the IS architecture. a. Identify the sources and targets. b. Physical and logical schemas, contexts to associate them c. Source and target data server connections, agents
Topology	5. Metadata into data models Table, views, constraints
Designer	6. Define elementary transformations. a. Define transformation rules and control rules. b. Define the transfer rules.
• Model Definition • Project Mapping	7. Unit tests of mappings a. Understand the outcome. b. Debug.
Operator	8. Optimize strategies. Knowledge Modules
Designer	9. Define the sequencing. a. Order the mappings and procedures in packages. b. Integration tests.
• Project/KM • Project/Package • Project/Scenario • Solution/ Versioning	10. Dev to QA a. Generate scenario. b. Create a solution for the project. c. Version the solution. d. Restore solution in QA repository.
OEM or Operator	11. QA to Prod a. Export scenario from QA. b. Import in Production.
	12. Operations a. Define execution schedules. b. Follow up executions.

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This slide suggests, at a very high level, one possible methodology for using ODI in a project. Due to class time constraints, the practices of this course do not follow this generic checklist. The next slide provides a checklist of the practices that you perform in this course.

Checklist of Practice Activities

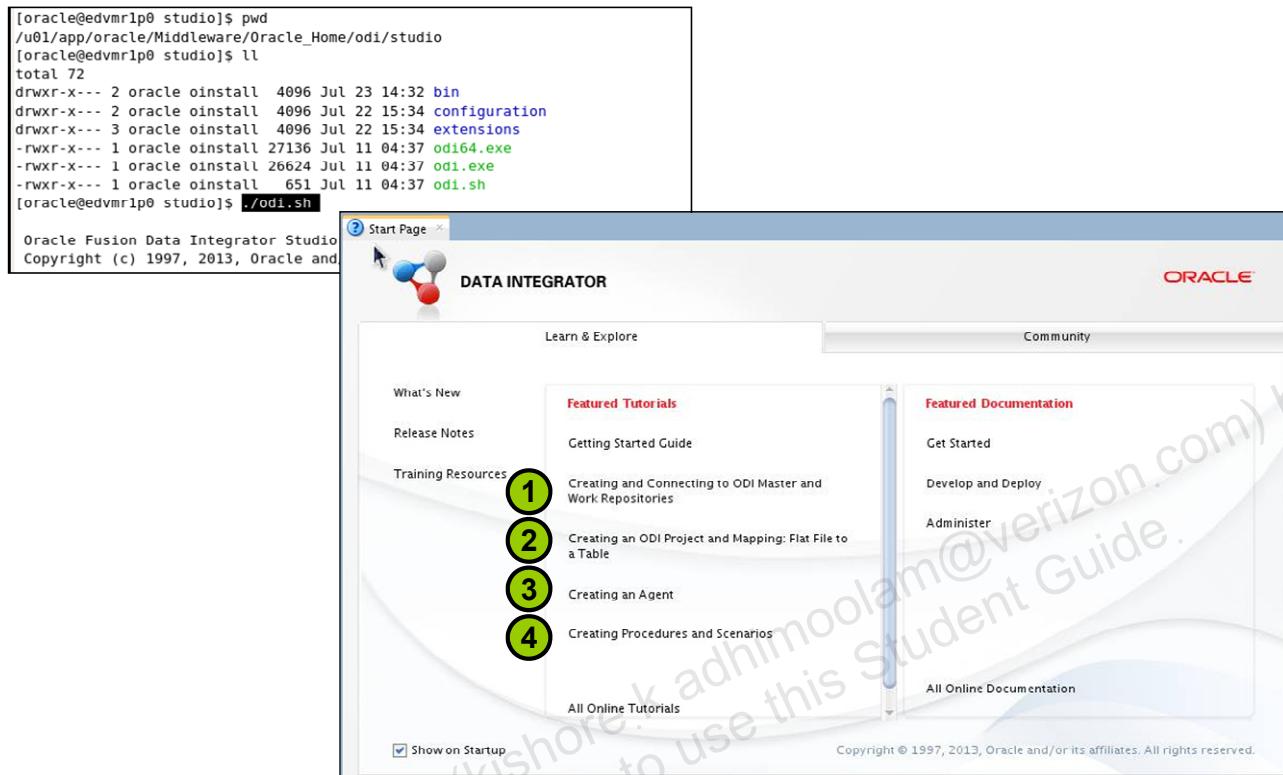
- 1-1: **Logging In and Help Overview**
- 2-1: Creating and Connecting to ODI Master and Work Repositories
 - 3-1: Configuring Standalone Agents Using the Common Admin Model
 - 4-1: Working with Topology
 - 5-1: Setting Up a New ODI Project
 - 6-1: Creating Models by Reverse-Engineering
 - 7-1: Checking Data Quality in the Model
 - 8-1: Creating ODI Mapping: Simple Transformations
 - 9-1: Creating ODI Mapping: Complex Transformations
 - 9-2: Creating ODI Mapping: Implementing Lookup
 - 10-1: Creating ODI Mapping: Exporting a Flat File to a Relational Table
 - 11-1: Using Native Sequences with ODI Mapping
 - 11-2: Using Temporary Indexes
 - 11-3: Using Sets with ODI Mapping
 - 12-1: Creating and Using Reusable Mappings
 - 12-2: Developing a New Knowledge Module
 - 13-1: Creating an ODI Procedure
 - 14-1: Creating an ODI Package
 - 14-2: Using ODI Packages with Variables and User Functions
 - 15-1: Debugging Mappings
 - 16-1: Creating and Scheduling Scenarios
 - 17-1: Using Load Plans
 - 18-1: Enforcing Data Quality with ODI Mappings
 - 19-1: Implementing Changed Data Capture
 - 20-1: Setting Up ODI Security
 - 20-2: Integration with Enterprise Manager and Using ODI Console



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This checklist provides a preview of the practices that you perform in this course. These hands-on practices begin in the next lesson.

Starting Oracle Data Integrator



ORACLE

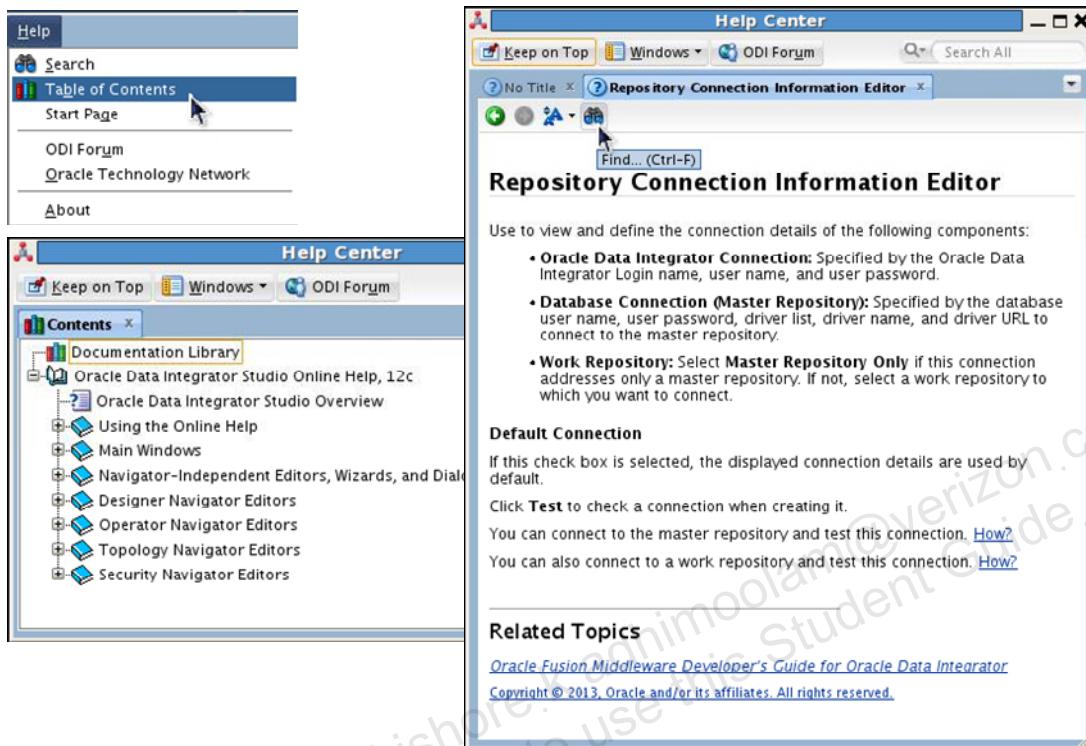
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can find the ODI Studio client in the `$ODI_HOME/studio` subfolder. In this class, a desktop icon is provided to start the ODI Studio client.

When you log in to the ODI client for the first time, you are presented with the Start Page, which has links to Oracle By Example (OBE) tutorials in the Oracle Learning Library (OLL). The top level of tutorial links is organized by a typical sequence of ODI tasks:

1. Create and Connect to Master and Work Repositories
2. Create Project and Mapping for a Flat File to a DB Relational Table
3. Creating and Starting Agents
4. Creating Procedures, Compiling Scenarios, and Scheduling Scenarios to Run Later

Using Online Help



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The online Help is available immediately, even before creating or connecting to repositories, and has been entirely rewritten to support the new ODI 12c user interface.

The Help Center provides quick access to help and common tasks, as well as links to useful Oracle resources.

Quiz

Which of the following does not belong to the ODI runtime components?

- a. ODI Operator
- b. ODI standalone agent
- c. ODI Topology Navigator
- d. ODI Security Navigator
- e. ODI Repository



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: c, d

Explanation: The ODI Topology Navigator and the ODI Security Navigator are not runtime components. The Topology Navigator is used for creating physical and logical architecture and is not involved in runtime processing. Similarly, the Security Navigator is not a runtime component because it is used for managing users and maintaining security policies.

Quiz

If necessary, several Master Repositories can be designated with several Work Repositories.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Explanation: A Work Repository can be linked with only one Master Repository, for version management purposes. A Master Repository might have exactly one Work Repository. However, several Master Repositories can be designated with several Work Repositories, if necessary. Examples:

- M1 can own W1 and W2.
- M2 can own W3.
- M3 can own W4, W5, and W6.
- M4 cannot own W3.

Summary

In this lesson, you should have learned how to describe:

- Course objectives and agenda of lessons
- Benefits of using ODI
- ODI 12c architecture and components
- How to use ODI Studio to create, administer, and monitor ODI objects
- Starting Oracle Data Integrator (ODI) Studio client
- Accessing online Help



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 1-1: Logging In and Help Overview

1. Start the ODI Studio client and log in.
2. Use the Start Page and Help system.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Database, ODI, and WLS binaries are pre-installed. In this practice, you learn how to start the ODI Studio client and log in. You also practice using the Start Page and Help system.

Administering ODI Repositories



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

KISHORE ADHIMOOlam (kishore.k.adhimoolum@verizon.com) has
a non-transferable licence to use this Student Guide.

Objectives

After completing this lesson, you should be able to:

- Create repositories using Repository Creation Utility (RCU)
- Connect to the Master Repository
- Export and import the Master Repository
- Create, connect, and set a password to the Work Repository



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the practices, all passwords that you create will be **welcome1**. Note the final character is the numeral one. Of course, in real life you would not do that.

Agenda

- Administering the ODI Repositories
 - Master
 - Work



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the next few slides, you will learn how to create and administer ODI Master and Work Repositories.

Initial Repository Administration Tasks

- ODI Master and Work Repositories require creating repository storage spaces (database schema/user).
 - Name
 - Password
 - Privileges
- You need to set up connections to ODI Master and Work Repositories:
 - Setting ODI connection information (login name, user, and password)
 - Entering database connection information (driver and URL)
- You may need to import or export an existing repository in ODI.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The initial tasks involved in setting up an ODI environment include creation of Master and Work Repository database schemas/users, and defining connections to the Master and Work Repositories.

Steps to Set Up ODI Repositories

1. Run Repository Creation Utility to create (if necessary):
 - A. Schemas
 - B. DB tablespaces, users, passwords
 - C. Objects
2. Connect to the Master Repository.
Connect to the Work Repository.
3. Create Password Wallet.

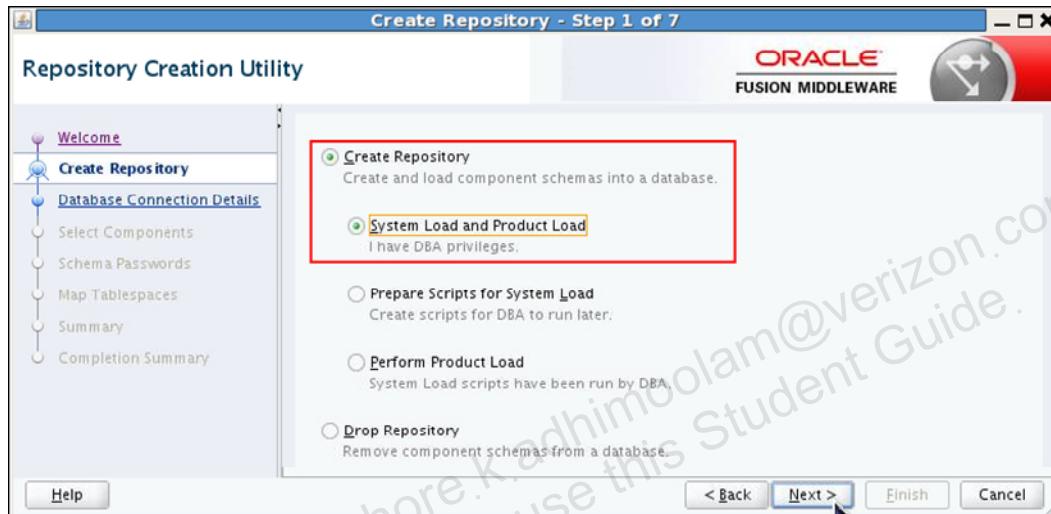


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

It is possible to use existing users and tablespaces, or RCU can create them as needed. These three steps for setting up ODI repositories are covered in detail in the following slides.

1. Run Repository Creation Utility

```
[myuser@myhost ~] $ cd $FMW_HOME/oracle_common/bin  
[myuser@myhost bin] $ pwd  
/u01/app/oracle/Middleware/Oracle_Home/oracle_common/bin  
[myuser@myhost bin] $ ./rcu
```



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You may find other scripts to start RCU in other directories such as WLS or SOA directories, but the one you want for ODI is installed by installing ODI.

Start the RCU GUI and, after the Welcome splash screen, accept the defaults to create a new set of ODI repositories. After repositories are created, if they are no longer used, you have the option to come back and drop them.

Even though it says, "Step 1 of 7" on the title bar, after you get into the configuration, the total number of steps may change (usually increase.)

Note: Having Oracle Database Vault configured in DB 12c may interfere with RCU. You need to have privileges to run Vault, or need to have Vault disabled, even if you are running as sysdba.

1a. Create Schemas

RCU: Database Connection Details

The screenshot shows the Oracle RCU interface. On the left, the 'Database Connection Details' panel is displayed, containing fields for Host Name (localhost), Port (1521), Service Name (orcl.us.oracle.com), Username (sys), and Password (*****). A green circle labeled '1' is drawn around the Host Name, Port, and Service Name fields. On the right, the 'Select Components' panel lists various Oracle components and their schema owners. A green circle labeled '2' is drawn around the 'Oracle Data Integrator' component, which is highlighted with a yellow background. A red arrow points from the '2' circle to the 'Oracle Data Integrator' row.

Component	Schema Owner
<input type="checkbox"/> Oracle AS Repository Components	
<input checked="" type="checkbox"/> AS Common Schemas	
<input type="checkbox"/> Metadata Services	MDS
<input checked="" type="checkbox"/> Audit Services	DEV_IAU
<input checked="" type="checkbox"/> Audit Services Append	DEV_IAU_APPEND
<input checked="" type="checkbox"/> Audit Services Viewer	DEV_IAU_VIEWER
<input checked="" type="checkbox"/> Oracle Platform Service Services	DEV_OPSS
<input type="checkbox"/> User Messaging Service	UMS
<input type="checkbox"/> WebLogic Services	WLS
<input type="checkbox"/> ServiceTable	DEV_STB
<input type="checkbox"/> SOA Suite	
<input checked="" type="checkbox"/> Oracle Data Integrator	DEV_ODI_REPO
<input type="checkbox"/> Master and Work Repository	

RCU: Select Components

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

Creating the Master and Work Repositories consists of creating the tables and the automatic importing of definitions for the different technologies.

To create the Master Repository, perform the following:

1. For the **Database Connection Details**, configure:
 - a. **Database Type**, such as Oracle Database, MySQL, IBM DB2, and so on
 - b. **Host Name**, either IP address, DNS name, or `localhost`
 - c. DB Listener **Port**, by default 1521
 - d. DB **Service Name** (not the instance name), usually it is the instance and the host name together
 - e. **Username** with DBA privileges, such as `SYS`
 - f. **Password** of the DBA username, which will not be displayed
 - g. **Role**, which changes as you type the username
2. For **Select Components**, select only **Oracle Data Integrator**. The other required lines will be selected for you automatically. Note the many schemas created, some with the prefix that you specify, for example `DEV` or `DEV2`, or `PROD99`.

1b. Create Passwords and Tablespaces

RCU: Schema Passwords

Define passwords for main and auxiliary schema users.

Use same passwords for all schemas

Password: *********

Alpha numeric only. Cannot start with a number. No special characters.

Confirm Password: *********

Use main schema passwords for auxiliary schemas

Specify different passwords for all schemas

RCU: Custom Variables

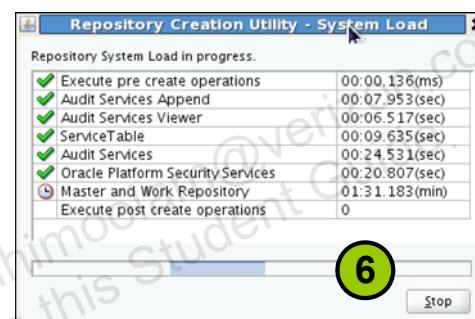
Component	Custom Variable	Value
Master and Work Repository	Supervisor Password	*****
	Confirm Supervisor Password	*****
	Work Repository Type: (D) Developme...	D
	Work Repository Name (WORKREP)	workrep
	Work Repository Password	*****
	Confirm Work Repository Password	*****

Default and temporary tablespaces for the selected components appear in the table below. To create new tablespaces or modify existing tablespaces, use the 'Manage Tablespaces' button.

Manage Tablespaces

Component	Schema Owner	Default Tablespace	Temp Tablespace
Audit Services	DEV_IAS	*DEV_IAS_IAU	*DEV_IAS_TEMP
Audit Services Append	DEV_IAS_APPEND	*DEV_IAS_IAU	*DEV_IAS_TEMP
Audit Services Viewer	DEV_IAS_VIEWER	*DEV_IAS_IAU	*DEV_IAS_TEMP
Oracle Platform Secu...	DEV_OPSS	*DEV_IAS_OPSS	*DEV_IAS_TEMP
Master and Work Re...	DEV_ODI_REPO	*DEV_ODI_USER	*DEV_ODI_TEMP
ServiceTable	DEV_STB	*DEV_STB	*DEV_IAS_TEMP

RCU: Map Tablespaces



RCU: Progress

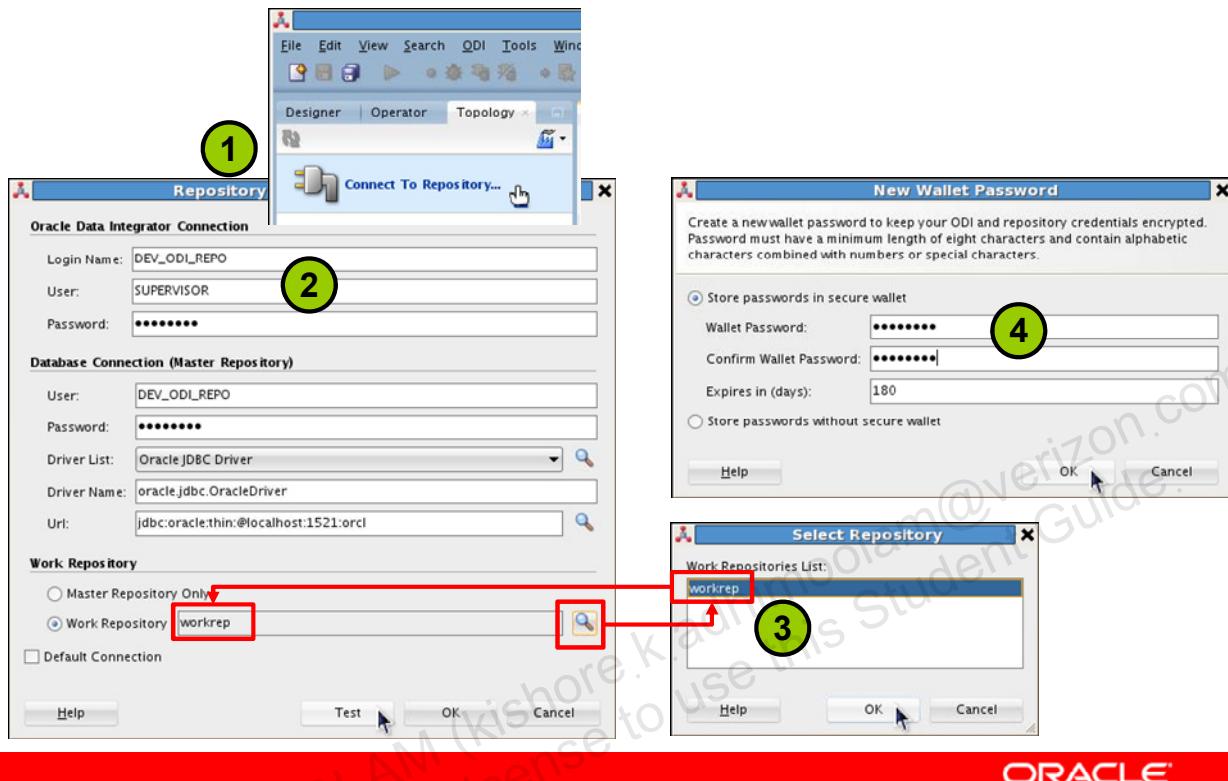
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

3. Select the schema passwords you will use. In a test or development environment, making all the passwords the same is easier, but in a production environment you may not have that luxury. (The practice uses **Welcome1**.)
4. Select the passwords that ODI will use for the repositories and supervisory users. Enter:
 - a. **Supervisor password** twice; it will not be displayed (the practice uses **Welcome1**)
 - b. **D** for Development (as opposed to E for Execution)
 - c. Work Repository Name, for example **workrep**, **workrep2**, and so on
 - d. **Work Repository Password** twice. Again, in a test environment, you can make all the passwords the same, but for the added security needed for a production environment, you might not want to do that in real life.
5. You can make dedicated **tablespaces** for ODI repositories (recommended), or you can use existing tablespaces.
6. The progress bar indicates the elapsed time used to create the various components.

2. Connect to the Master/Work Repository

3. Create a Wallet



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To connect to the new Master Repository, perform these steps in ODI Studio:

1. On the left, on either the **Designer** or **Topology** tab, click **Connect to Repository...**

2. Complete the fields that define a connection in ODI and a connection to the database:

Oracle Data Integrator Connection:

- **Login Name:** A generic alias (for example, **Master Repository**)
- **User:** SUPERVISOR (use uppercase)
- **Password** that you indicated on the previous panel in step 4a.

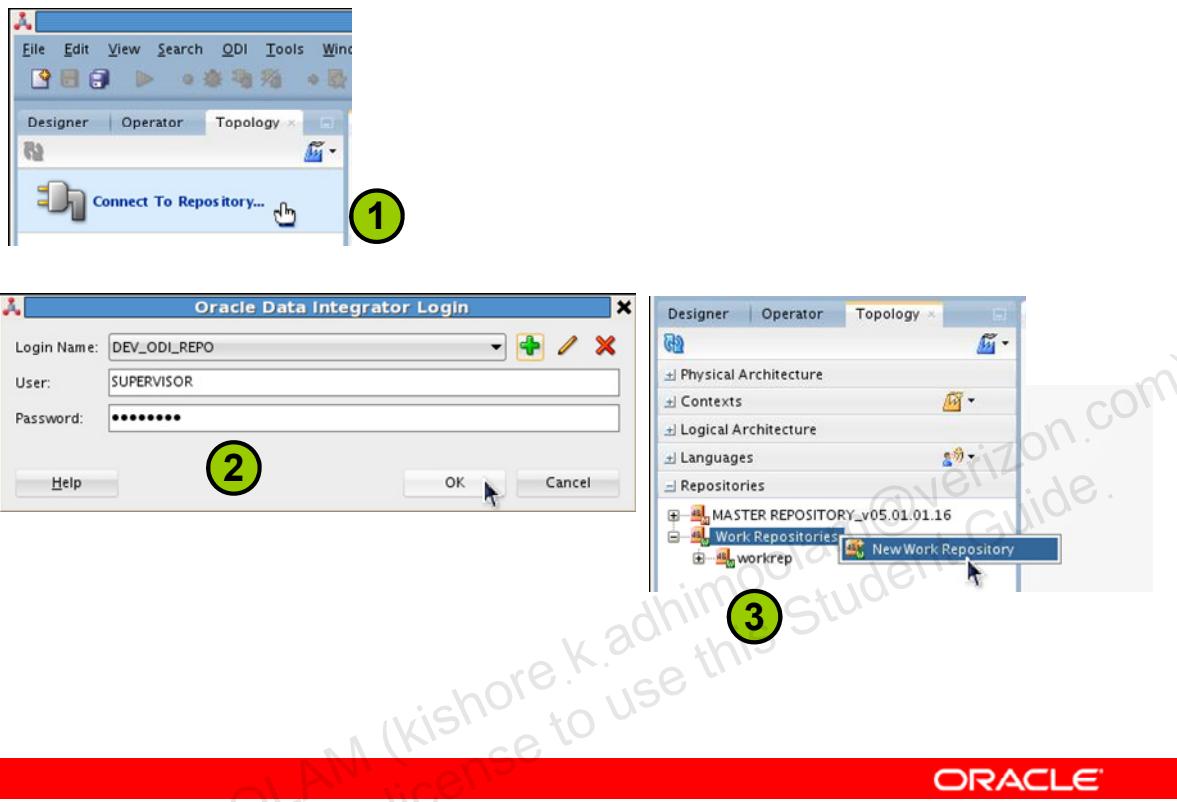
Database Connection:

- **User:** User ID or login of the owner of tables you created for the Master Repository
- **Password:** The user's password
- **Driver List and Driver Name:** Choose the driver required to connect to the DBMS supporting the Master Repository that you have just created.
- **Url:** Enter the complete path of the data server hosting the repository.

3. Click the magnifying glass to select the valid work repository. Click **Test** to check whether the connection is working. Validate by clicking **OK** twice.

4. Optionally store all of these passwords in a Wallet in your home directory.

Connecting to the Master/Work Repository



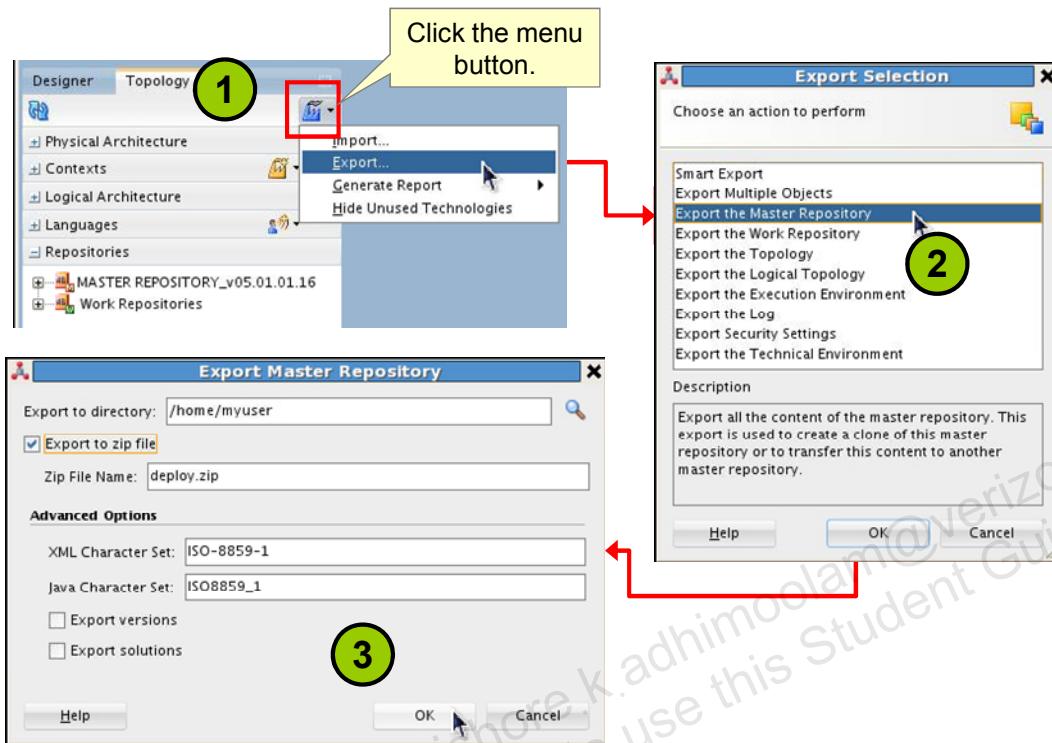
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

Hereafter, each time the information to connect is stored in the wallet.

1. To connect after the first time, click **Connect To Repository** as you did before.
2. The login is a Master Repository plus a Work Repository, and an ODI username, all stored together.
3. After it builds the lists, you can see which repository you are connected to by navigating to **Topology > Repositories**. A Master Repository can have several Work Repositories, but a Work Repository may belong to only one Master.

Exporting the Master Repository



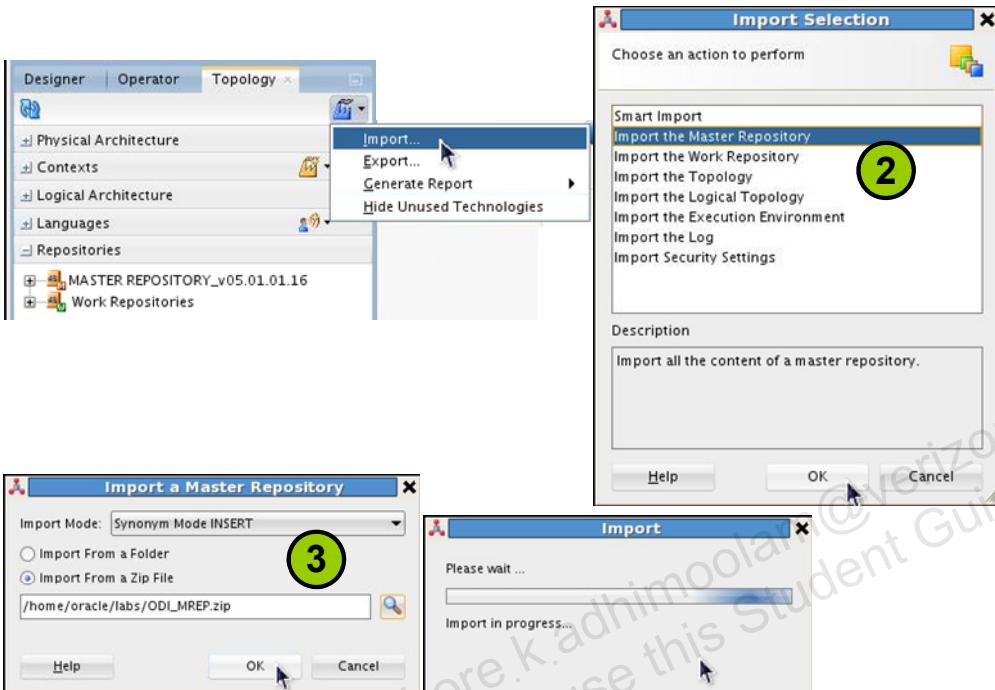
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Master Repository Import/Export procedure enables you to optionally transfer the whole repository (Topology and Security domains included) from one repository to another.

1. In the Topology Navigator, click the menu button in the upper-right corner of the Topology tab, and select **Export > Export the Master Repository**.
2. Fill in the export parameters:
 - **Export to directory:** Directory in which the export file(s) are created
 - **Export to zip file:** When this option is selected, a unique compressed file containing all export files is created. Otherwise, a set of export files is created.
 - **Zip File name:** Name of the compressed file if the "Export to zip" option is selected
 - **XML Character Set:** Encoding specified in the export file. Parameter encoding in the Extensible Markup Language (XML) file header
 - **Java Character Set:** Java character set used to generate the file
 - **Export versions:** Exports all stored versions of objects that are stored in the repository. You may want to deselect this option to reduce the size of the exported repository, and to avoid transferring irrelevant project work.
 - **Export solutions:** Exports all stored solutions that are stored in the repository
3. Click **OK**. The export file(s) are created in the Export Directory specified.

Importing the Master Repository



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Optionally, instead of creating a new empty Master Repository, you can create a new Master Repository and populate it by importing an existing one.

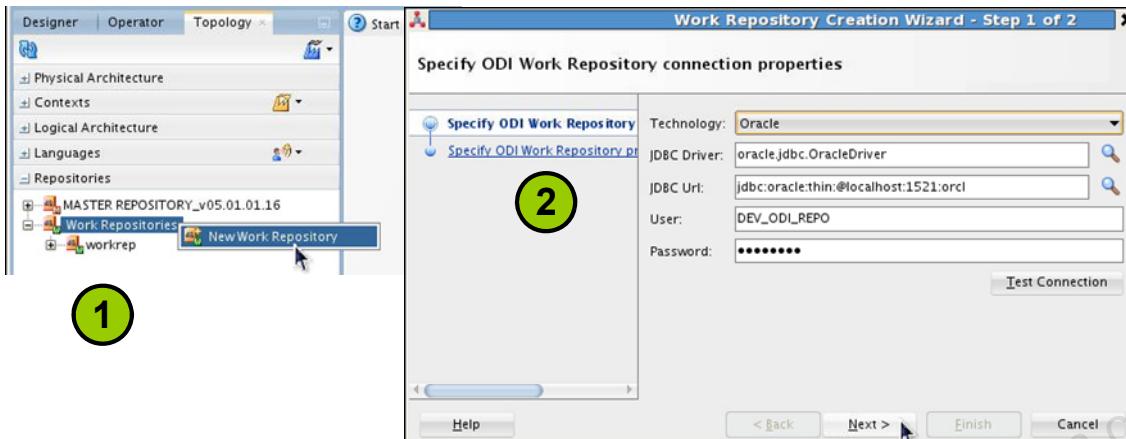
The Master Repository Import/Export procedure enables you to optionally transfer the whole repository (Topology and Security domains included) from one repository to another. It can be performed in Topology, to import the exported objects in an existing repository, or while creating a new Master Repository.

To import a Master Repository into an existing Master Repository, perform the following:

1. On the **Topology** tab, click the menu button and select **Import**.
2. Select **Import Master Repository**.
3. Navigate to the folder of the zip file to locate the **Master** Repository that you are importing and click **OK**.

The specified file or files are imported into the current Master Repository.

Creating a Work Repository



ORACLE

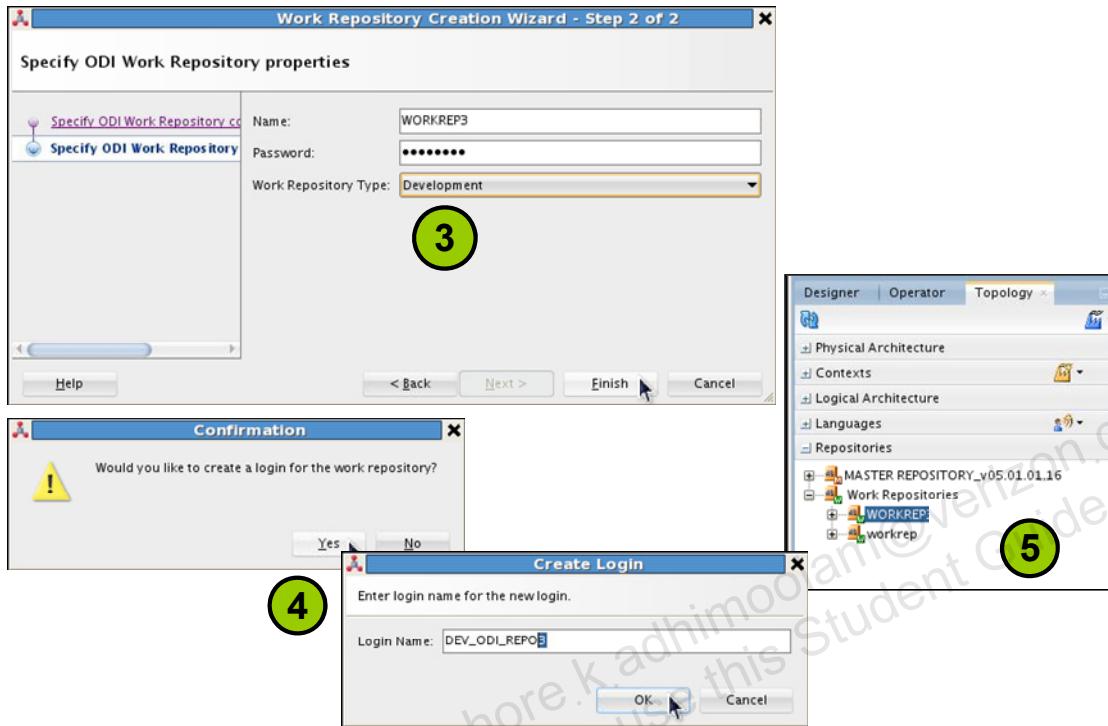
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You use ODI Work Repositories when designing new transformations with ODI Designer. Several Work Repositories can be designated with several Master Repositories, if necessary. However, a Work Repository can be linked with only one Master Repository for version management purposes.

To launch a Work Repository creation, perform the following steps:

1. Connect to your Master Repository through the Topology Navigator. In the Topology Navigator, click the **Repositories** tab. Right-click **Work Repositories** and select **New Work Repository**. A Create Work Repository Wizard appears, asking you to complete the connection parameters for your new Work Repository.
2. In the wizard's first window, complete the following parameters:
 - **Technology:** Select the technology of the server to host your Work Repository.
 - **JDBC Driver:** Set the JDBC Driver (the driver required for the connection to the DBMS to host the Work Repository).
 - **JDBC URL:** Set the JDBC URL (the path of the data server to host the Work Repository).
 - **User:** Enter the user ID of the owner of the tables that you will create and host in the Work Repository. You might want a new user such as **DEV_ODI_REPO_2**.
 - **Password:** Enter the user's password.

Creating a Work Repository



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

3. In the wizard's second window, complete the following parameters:

Name: Give a unique name to your Work Repository (for example, WORKREP3).

Password: your choice.

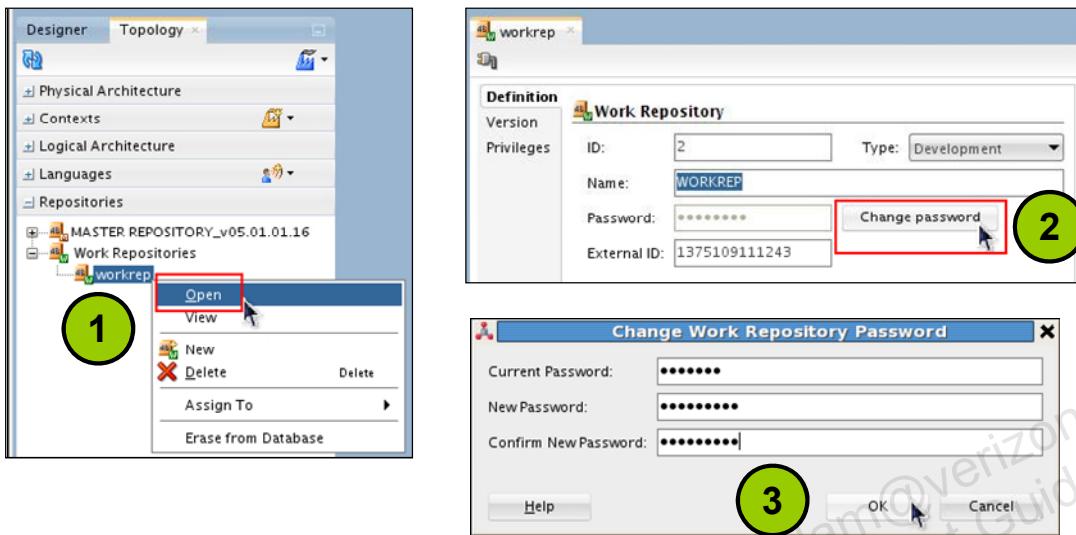
Note: You are urged not to set a password for the Work Repository (it can be more painful than useful). See the lesson titled “Advanced ODI Administration” to learn about using the Security Navigator to handle passwords at the security level.

Type: Select Development from the list.

Note: The Development type of repository enables management of design-time objects such as data models and projects (including interfaces, procedures, and so on). A development repository also includes the runtime objects (scenarios and sessions). This type of repository is suitable for development environments. Click **Finish**.

4. In the Create Work Repository login window, click **Yes**. Enter the Login name: DEV_ODI_REPO3 as shown in the screenshot. Click **OK**.
5. Verify that the newly created Work Repository is now in the Work Repositories tree view. When the Work Repository has been created, the Work Repository window closes. You can now access this repository through the Designer and Operator modules.

Changing the Work Repository Password



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For security purposes, you might have to change the Work Repository password. To change the password, you perform the following steps:

1. On the Repositories tab of the Topology Navigator, double-click the **Work** Repository. A repository window opens.
2. Click the **Change password** button.
3. Enter the old password and the new one. Click the **OK** button.

Quiz

The following steps are performed to set up an ODI repository environment:

1. Create repository storage spaces.
2. Create repository schemas.
3. Create the Master Repository.
4. Create one or more Work Repositories.
 - a. True
 - b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Explanation: Storage spaces are created in a database by using SQL*Plus or SQL Developer or RCU. One Master Repository is defined in ODI. One or more Work Repositories are defined in ODI.

Summary

In this lesson, you should have learned how to:

- Create and connect to the Master Repository
- Export and import the Master Repository
- Create, connect, and set a password to the Work Repository



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

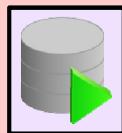
Checklist of Practice Activities

- ✓ 1-1: Logging In and Help Overview
- 2-1: **Creating and Connecting to ODI Master and Work Repositories**
- 3-1: Configuring Standalone Agents Using the Common Admin Model
- 4-1: Working with Topology
- 5-1: Setting Up a New ODI Project
- 6-1: Creating Models by Reverse-Engineering
- 7-1: Checking Data Quality in the Model
- 8-1: Creating ODI Mapping: Simple Transformations
- 9-1: Creating ODI Mapping: Complex Transformations
- 9-2: Creating ODI Mapping: Implementing Lookup
- 10-1: Creating ODI Mapping: Exporting a Flat File to a Relational Table
- 11-1: Using Native Sequences with ODI Mapping
- 11-2: Using Temporary Indexes
- 11-3: Using Sets with ODI Mapping
- 12-1: Creating and Using Reusable Mappings
- 12-2: Developing a New Knowledge Module
- 13-1: Creating an ODI Procedure
- 14-1: Creating an ODI Package
- 14-2: Using ODI Packages with Variables and User Functions
- 15-1: Debugging Mappings
- 16-1: Creating and Scheduling Scenarios
- 17-1: Using Load Plans
- 18-1: Enforcing Data Quality with ODI Mappings
- 19-1: Implementing Changed Data Capture
- 20-1: Setting Up ODI Security
- 20-2: Integration with Enterprise Manager and Using ODI Console

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 2-1: Creating and Connecting to ODI Master and Work Repositories



RCU

- Create **DEV1_ODI_REPO** database user and tablespaces for ODI *Master Repository* and for *Work Repository*.
- Create Master/Work Repositories.



Oracle Data Integrator

- Create Login **ODI_REPO1**
 - Create *Master Repository* connection “**DEV1_ODI_REPO**” with SUPERVISOR as the user.
 - Create **WORKREP1** *Work Repository* connection with SUPERVISOR as the user.

Note: These are practice repositories. In later lessons, you use other Master and Work Repositories that are partially predefined with source and target objects.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In this practice, you use two tools. You use RCU to create the database user and tablespaces for the Master and Work Repositories. You use ODI Studio to create the connection to the Master and Work Repositories.

1. Create RDBMS users and tablespaces for the Master and Work Repositories by using Repository Creation Utility.
2. Create the JDBC connection to the ODI Master and Work Repository by using the ODI Studio New Login panel.

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

KISHORE ADHIMOOLAM (kishore.k.adhimoolam@verizon.com) has
a non-transferable license to use this Student Guide.

ODI Topology Concepts

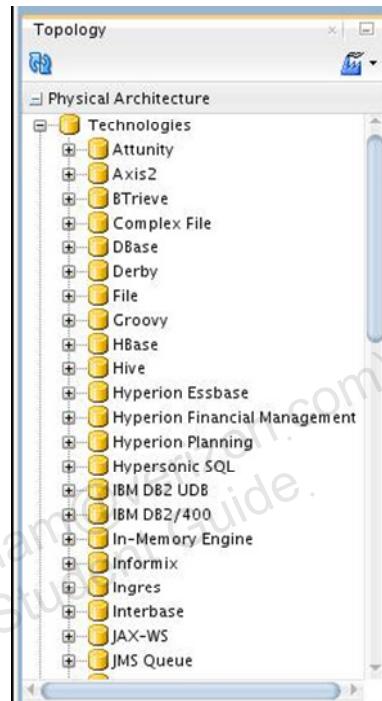
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the basic concepts of the ODI topology
- Describe the logical and physical architecture
- Plan a topology
- Use best practices to set up a topology
- Launch ODI agents and set agent parameters



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This lesson is a detailed introduction to the Oracle Data Integrator (ODI) topology and gives some basic recommendations about how to set it up. In this lesson, you should learn:

- The basic concepts of the ODI topology
- The definition of a physical architecture in ODI and its relationship to a logical architecture
- To plan a topology by using a simple set of guidelines
- Some of the current best practices in setting up a topology in ODI

Agenda

- **ODI Topology: Overview**
 - Logical and Physical Architectures
 - Contexts
- Data Servers and Physical Schemas
- Defining Topology
- Agents in Topology
- Planning a Topology



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the following slides, you learn what the ODI topology contains, how it defines the logical and physical architectures of your information system, and how the contexts link the two together.

What Is Topology?

- Topology is the representation of the information system in ODI.
- ODI uses the topology to connect to the resources in the information system for integration processes.



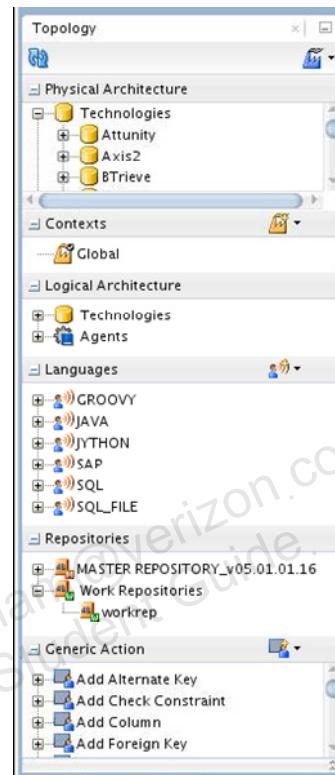
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In ODI, the Topology Navigator is used to define a complete representation of your information system.

It includes everything—from data servers and schemas through reserved keywords in languages used by different technologies. ODI uses this topology to access the resources available in the information system to carry out integration tasks.

What Is in the Topology?

- Physical architecture
 - Technologies: Oracle, DB2, File, and so on
 - Data types for the given technology
 - Data Servers: Definition of your servers, databases, and so on
 - Schemas: Subdivisions of data servers
 - Agents: ODI runtime modules
- Contexts: Mapping logical to physical
- Logical architecture
 - Technologies and Agents
- Languages
- Repositories
- Generic Actions



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

All available technologies are defined in the topology. For each technology, the available data types are defined. You rarely, if ever, need to modify these. However, you define all the data servers that use the technology in your information system. For each data server, you then define the subdivisions that are known as schemas in ODI.

Next, you must define the agents that carry out the integration tasks at run time. You also set up the contexts that enable you to define an integration process at an abstract logical level, and then link it to the physical data servers where it will be performed.

Languages and actions are also found in the topology. Languages specify the keywords that exist for each technology, and actions are used to generate the data definition language (DDL) scripts. You would need to modify these parts of the topology only if you were adding a new technology to ODI.

Agenda

- ODI Topology: Overview
- **Data Servers and Physical Schemas**
- Defining Topology
- Agents in Topology
- Planning a Topology



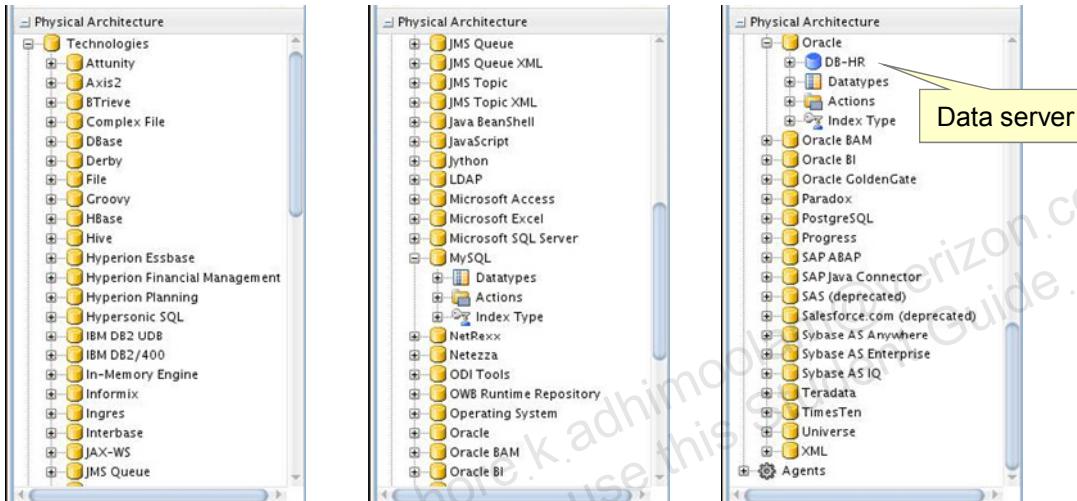
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The most essential things to define in your topology are your data servers and physical schemas. These enable you to connect to your data through ODI.

What Is a Data Server?

- A data server is any system that is capable of storing data and making it available in the form of tables.
- A data server is always attached to a specific technology.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The definition of a data server in ODI is fairly broad. A data server may not always be a traditional DBMS. Instead, any system that is capable of storing data and making that data available in the form of tables is a potential data server in ODI. In ODI topology, a data server is always attached to a specific technology, such as Oracle, Oracle Essbase, Sybase, Extensible Markup Language (XML), or Microsoft Access.

Data Servers: Examples

- ODI connects to a data server by using JDBC or JNDI.
- Examples of data servers:
 - Oracle instance
 - Microsoft SQL Server instance
 - IBM DB2
 - Oracle's MySQL
 - Oracle E-Business instance
 - File server



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

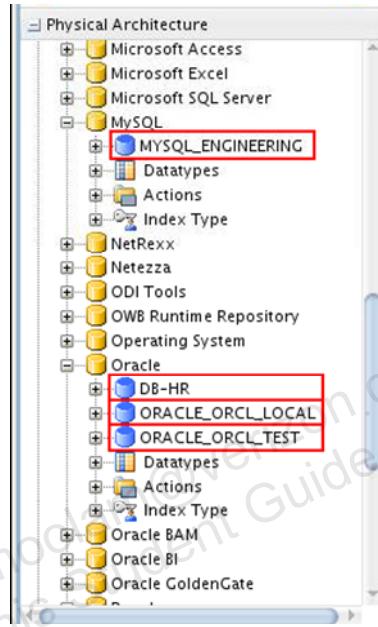
There are multiple ways of connecting a data server to ODI. Most often, Java Database Connectivity (JDBC) is used. JDBC drivers exist for a wide range of technologies. Java Naming and Directory Interface (JNDI) is another way to connect to your data server through a directory service. Additionally new technologies can be created that enable connectivity to application-specific data sources such as Oracle's MySQL.

Some concrete examples of data servers may be helpful. Each instance of a traditional database engine, such as Oracle or Microsoft SQL Server, is a data server. Thus, if you have two instances that are running on the same machine, ODI considers them to be separate data servers.

Important Guideline 1



Guideline 1: Each physical data server must be defined once in the topology.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

There are a number of simple guidelines to remember when setting up your topology in ODI. Guideline 1 is that every data server in your information system should appear once in your topology. For example, if you have a machine running two instances of Oracle and one Paradox data source, you have three physical data servers.

What Is a Physical Schema?

- A physical schema is a technology-dependent subdivision of a data server.
- Datastores on a data server are located in a physical schema.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A physical schema is a subdivision of a data server whose precise definition depends on the technology involved. Physical schemas indicate the physical location of the datastores, such as tables, files, topics, and queues in a data server. The names of physical schemas are used to prefix object names when generating code to produce their qualified names.

The physical schemas that need to be accessed must be defined in their corresponding data servers. You will now see some examples of how this works on different technologies.

Physical Schemas: Properties

- An ODI physical schema always consists of two data server schemas:
 - The data schema, which contains the datastores
 - The work schema, which stores temporary objects
- A data server schema is technology-dependent.
 - Catalog name and/or schema name
 - Example: Database and owner, schema
- A data server has:
 - One or more physical schemas
 - One default physical schema for server-level temporary objects



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

An ODI physical schema comprises two separate data server schemas: A data schema where the datastores for the physical schema are located and a work schema that is used by ODI to store temporary objects during integration processing. Several physical schemas can share the same work schema.

A data server schema in a data server is uniquely identified, depending on the technology, by a catalog name or a schema name, or both. The terms for the catalog name and the schema name differ depending on the technology.

For example, in Microsoft SQL Server, the catalog is called “database,” and the schema is called the “owner.”

In Oracle, there is no catalog, and a schema is called a “schema” or “user.” You will see some examples later in this lesson.

Though a data server can have several physical schemas, it has one default physical schema that is used to select a work schema to store server-level temporary objects during certain integration processes.

Technology Terminology Among Vendors

Technology	Data Server	Schema
Oracle	Instance	Schema
Microsoft SQL Server	Server	Database/Owner
Sybase ASE	Server	Database/Owner
DB2/400	Server	Library
Teradata	Server	Schema
Microsoft Access	Database	(N/A)
JMS Topic	Router	Topic
File	File Server	Directory



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The terms “data server” and “schema” are general terms in ODI that represent different concepts in different database technologies. The table in the slide shows the relationship between the ODI terms “data server” and “schema” with their equivalents in some common technologies.

For example, an Oracle server “instance” is equivalent to an ODI “data server.” A “schema” on this Oracle instance is represented as a “schema” in ODI.

However, for SQL Server, an ODI schema represents a combination of “database” and “owner.” For example, a table might be called “salesdb.bill.mytable.” In ODI, the schema is “salesdb.bill.”

Not every technology supports schemas. For example, an Access “database” corresponds to an ODI data server. However, Access databases are not subdivided into schemas. Thus, when you define an Access data server, you do not need to specify a schema.

ODI also supports technologies that are not true databases. For example, you can set up a directory of files as a database. Here, the file server becomes the data server and a specific directory is an ODI schema. Individual files in that directory are then treated as datastores.

Note: The recommended practice in ODI is to create a separate area on each data server specifically for the temporary objects. You should use this dedicated area as the work schema for your physical schema. Thus, there is no risk of temporary objects created by ODI polluting your production data.

Important Guideline 2



- **Guideline 2:** Under each data server, define a physical schema for each subdivision of the server used in ODI.
- Recommendations:
 - All schemas in the same instance of the database must be defined under the same data server.
 - The best possible login to the data server is the owner of the staging schema (also called “work schema”).
 - Ensure that all data servers have a DEFAULT schema (by default, the first one that is created). This is required for most Knowledge Modules.

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

For each data server in your topology, you must define a physical schema to represent each subdivision of the server that will be used. So, if a server has four technology-specific subdivisions, but you want to use only two in ODI, you must define two physical schemas for that server.

Agenda

- ODI Topology: Overview
- Data Servers and Physical Schemas
- **Defining Topology**
 - Example
- Agents in Topology
- Planning a Topology

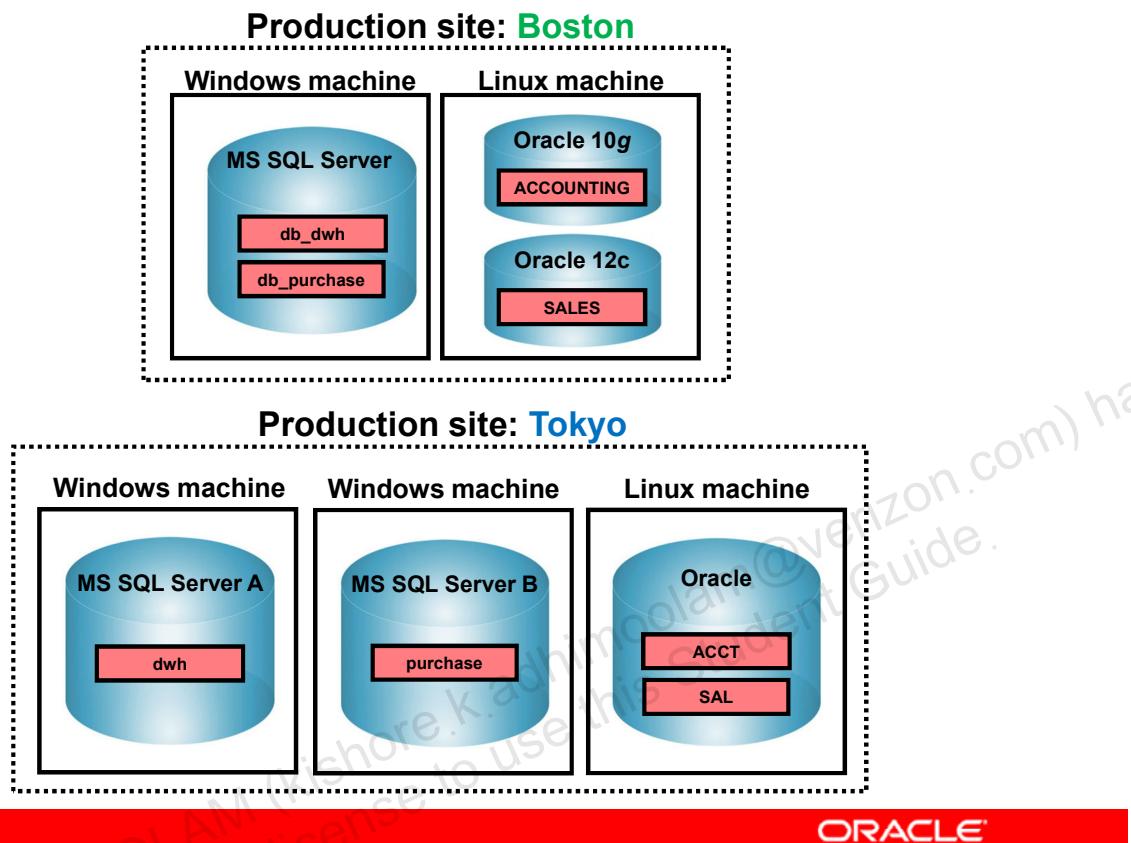


ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The most essential things to define in your topology are your data servers and physical schemas. These enable you to connect to your data through ODI.

Infrastructure for Two Production Sites: Example



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Here is an example that you will refer to throughout this lesson. It is a simple setup with accounting and sales information that is used to populate a data warehouse and a purchasing database.

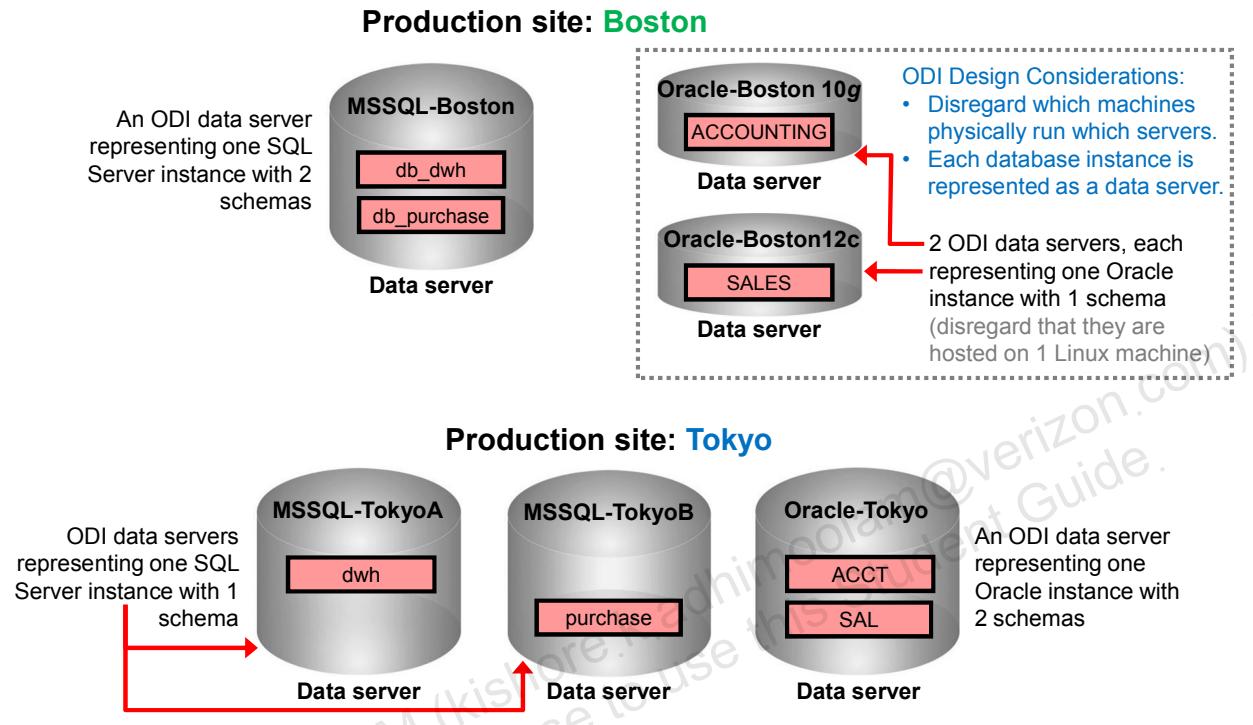
There are two separate production sites, with the first based in Boston. The Boston production site has a Windows machine that runs SQL Server. The data warehouse for Boston is hosted on this machine. The SQL Server database is named `db_dwh` at this site and the Boston purchasing database, also hosted on this machine, is named `db_purchase`.

The example also shows a Linux machine, which, for historical reasons, has two different versions of Oracle running on the same system. One version has a schema for accounting, and the other version has a schema for sales.

A second production site is in Tokyo. Here, the data warehouse and purchasing databases are split onto different Windows machines that run SQL Server. The data servers are labeled "MS SQL Server A" and "MS SQL Server B." However, the accounting and sales databases run on a single Oracle server. Note that the physical schemas at this site have different names from those at the Boston site.

The next slide shows how to model this physical architecture in ODI.

ODI Design: Physical Architecture of the Two Production Sites



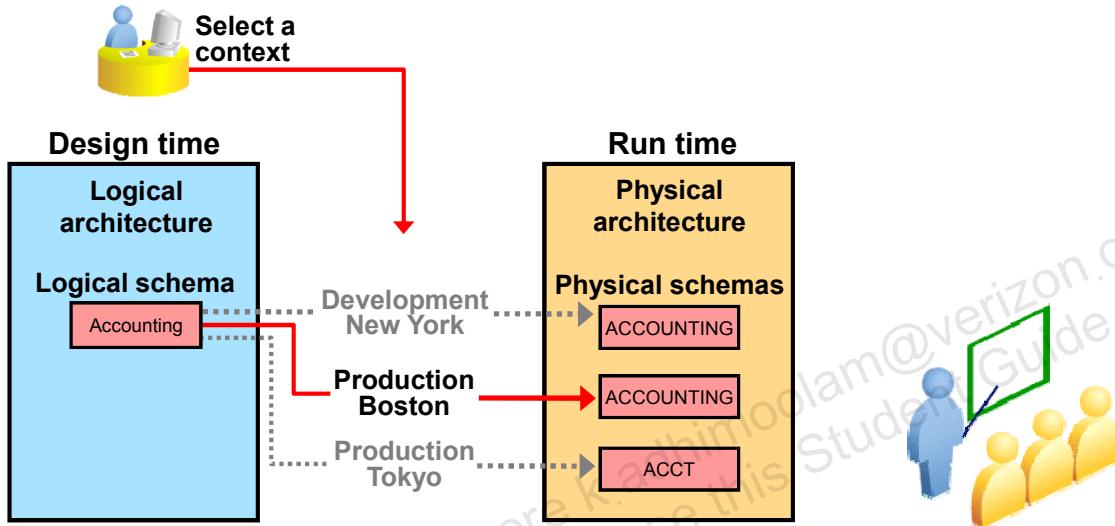
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

Applying the previous slide's architecture in ODI design, guidelines 1 and 2 enable you to design this physical architecture in ODI. You disregard which machines physically run which servers.

Each Oracle and SQL Server instance is represented as a data server in the ODI topology. Similarly, each Oracle schema or SQL Server database is represented as a physical schema.

Logical Schemas and Contexts



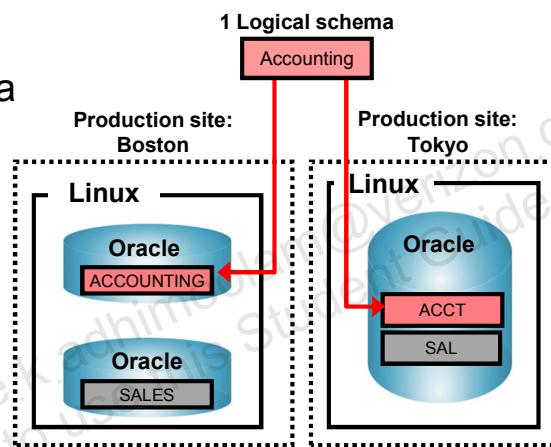
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You learned about the physical architecture, which describes how ODI physically accesses data. Now, you look at the logical architecture, which is a more general view that takes into account the similarities between the physical schemas in different environments.

What Is a Logical Schema?

- A logical schema is a single alias for different physical schemas that have similar data structures based on the same technology, but in different contexts.
- If two data schemas stored in data servers contain the same data structures, they are declared as:
 - Two physical schemas
 - One single logical schema



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The essential concept to understand in the logical architecture is the logical schema. A logical schema is a single alias for different physical schemas. These schemas should have similar or identical data structures, and must be based on the same technology. The logical schema thus brings together different physical schemas representing the same kind of data in different contexts.

If two similar data schemas are stored in two different data servers, but they have the same data structures, you declare the two data schemas as physical schemas in ODI, according to guideline 2. However, you create a single logical schema that represents the pair of schemas. The context determines the physical schema that will be used at any given time.

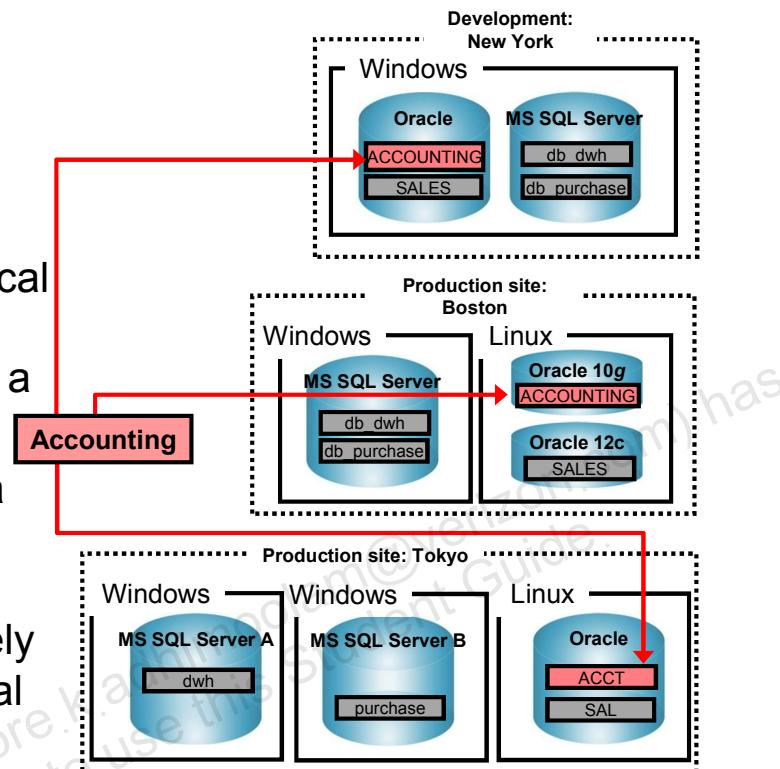
A good example of a logical schema is found in the information system example. In Boston, the accounting database is stored in a schema called ACCOUNTING. In Tokyo, the schema is called ACCT. Both of them have the same data structure; they contain all tables for an accounting application. In the ODI topology, you consider them as one logical schema: Accounting.

Important Guideline 3



Guideline 3: Define one logical schema for each group of physical schemas containing a similar data structure.

- Note that for similar data structures, only the *structure* counts. The *content* can be completely different on each physical schema.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You should define one logical schema each time you have a group of physical schemas containing similar data structures.

In this example, Accounting represents the ACCOUNTING Oracle schema on the Windows machine in New York, the ACCOUNTING schema on the Oracle 10g server running on Linux in Boston, and the ACCT schema running on Linux in Tokyo.

Note: A simple way to understand guideline 3 is to remember that one logical schema corresponds to the data structure for one application, implemented in several places called physical schemas.

Logical Versus Physical Architecture

- Physical architecture:
 - Defines physical resources
 - Describes the real locations of servers and schemas
 - If the accounting application data schema exists at several places, it is declared as different physical schemas.
- Logical architecture:
 - Abstract view of the resources
 - The accounting application schema is defined only once here. It is one logical schema implemented in different contexts.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

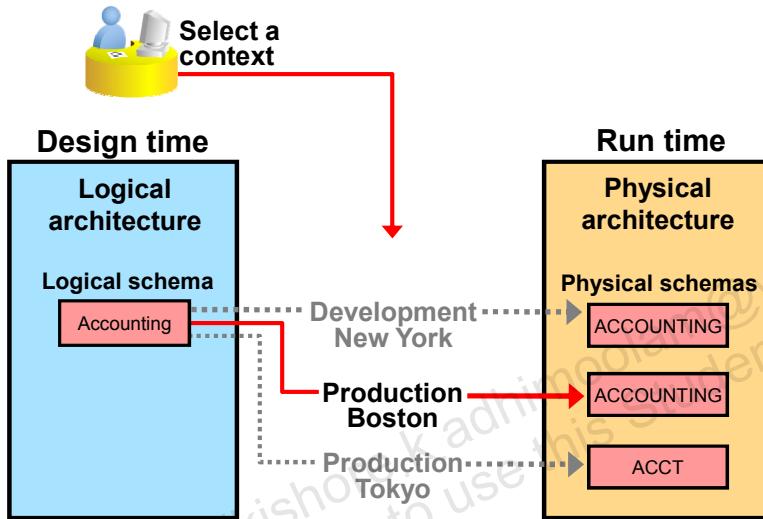
This distinction can be generalized into the logical architecture as distinct from the physical architecture.

The physical architecture tells ODI about physical resources, such as connections to data servers and agents, which will be covered later in the lesson. The physical architecture describes the locations of servers that exist in your information system and the schemas that you want to use on them. For example, you may logically have one data schema representing your accounting application. But, if this schema exists in different places, you should declare these as separate physical schemas.

The logical architecture, on the other hand, gives a more abstract view of these resources. In this example, your accounting application schema is defined only once in the logical architecture. It is a single schema implemented in several different contexts. The contexts are covered in the following slides.

Design Time Versus Run Time

- In ODI, you work at design time on logical resources.
- At run time, execution is started in a particular context, allowing access to the physical resources of that context.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

It is important to understand that in ODI, you always work at design time using objects defined in the logical architecture. For example, you would refer to the datastores defined in the logical schema Accounting without specifying whether you meant the server in Tokyo or Boston.

At run time, physical resources are required to carry out the integration process. You, therefore, specify a particular context to execute within. ODI is then able to provide access to the physical resources of that context.

What Is a Context?

- A context is an ODI object that you define that represents a “situation” where a similar group of resources appears.
- A context maps logical resources onto their implementations as physical resources (logical architecture onto a physical architecture).
- In a given context, one logical resource is mapped to one unique physical resource.
- Example of contexts:
 - Sites: Boston, Tokyo, and Development
 - Environments: Development, Test, and Production

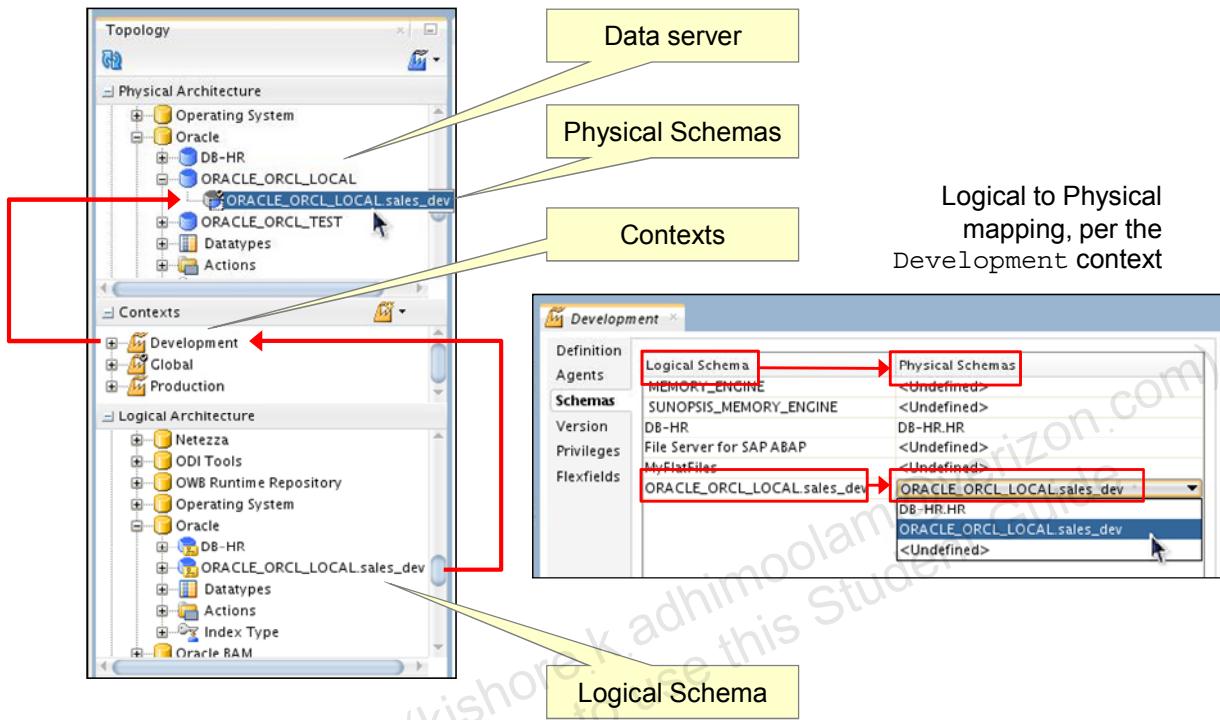
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A context represents a “situation” where you see the same group of resources. From a technical perspective, a context maps individual logical resources onto individual physical resources. So, given a context and a logical resource, ODI can determine the unique physical resource that is appropriate.

In terms of topology, the logical architecture is mapped onto the physical architecture through a context. In your example, Boston, Tokyo, and Development are contexts representing different geographical sites. However, it can also represent situations for different purposes, such as development, test, or production sites.

A Context Maps a Logical to a Physical Schema



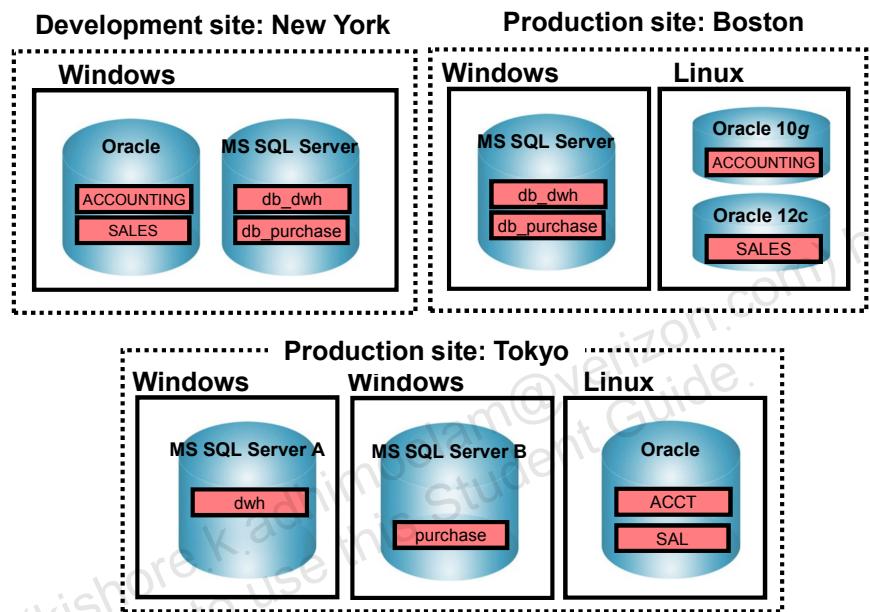
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A logical schema can have multiple physical schemas resolved per context. In this example, for the Development context, the ORACLE_ORCL_LOCAL.SALES_DEV logical schema maps to the ORACLE_ORCL_LOCAL.SALES_DEV physical schema. The logical and physical names do not have to be identical, nor do they have to include technology_instance prefixes. A logical schema could be present in several contexts.

Defining Contexts

- **Guideline 4:** Similar resources appear on the three different sites.
- Three contexts:
 - Development
 - Boston
 - Tokyo



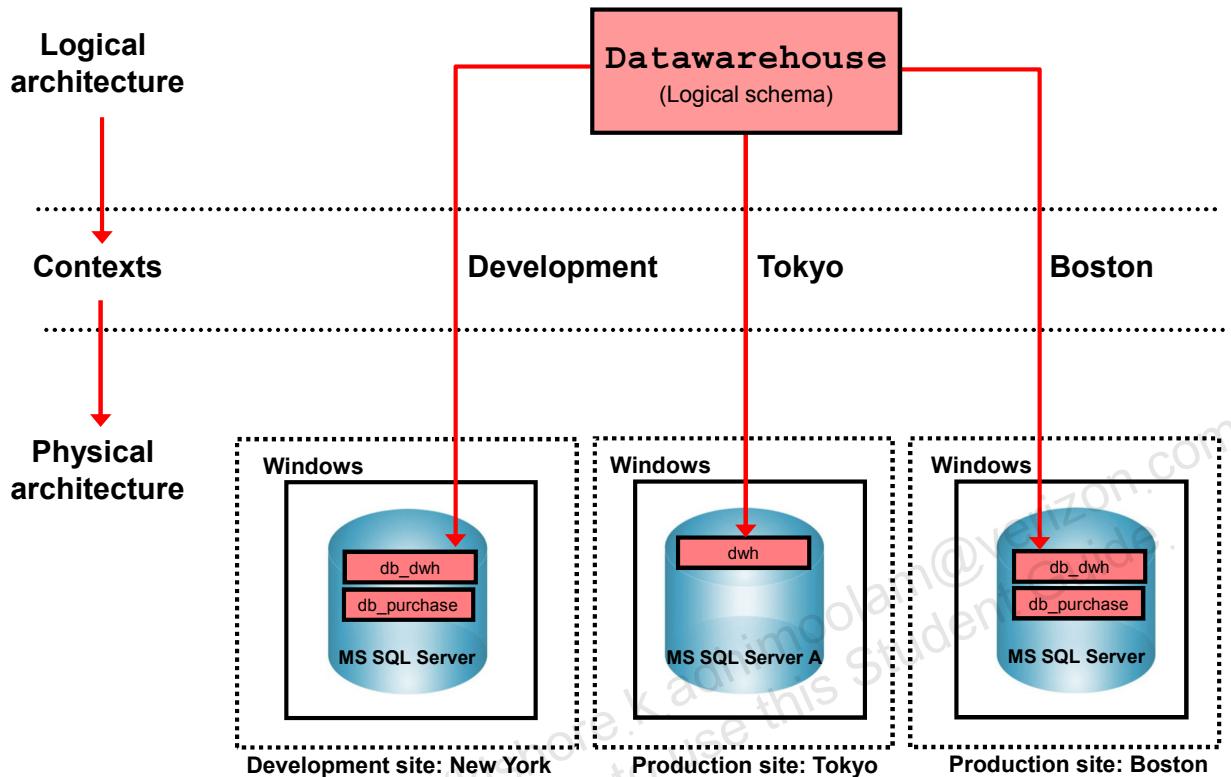
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Guideline 4: If you have similar groups of resources in different situations, you should define one context for each situation.

If you apply guideline 4 to the example, each site clearly represents the same collection of four different resources. Therefore, you have three contexts: Development, Boston, and Tokyo.

Mapping Logical and Physical Resources



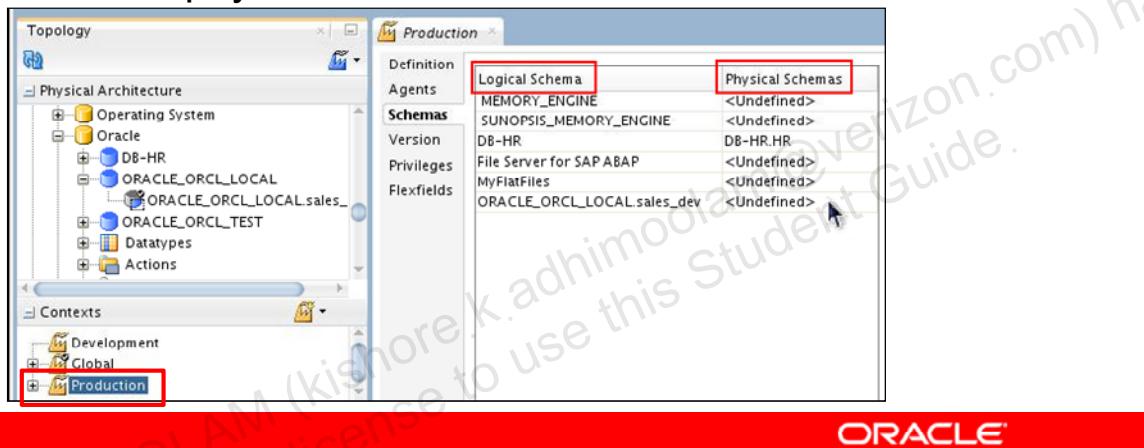
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You now illustrate the meaning of a context graphically by using the infrastructure example. You know about the data warehouses in New York, Tokyo, and Boston that have different names but share similar data structures. At the logical architecture level, you have a single logical schema called Datawarehouse. However, you have three contexts. In the “Development” context, this logical schema is mapped onto the physical schema db_dwh at the New York site. In the “Tokyo” context, it is mapped onto the physical schema dwh at the Tokyo site. In the Boston context, the logical schema is mapped onto the physical schema db_dwh at the Boston site.

Mapping Logical and Physical Resources

- Logical resources may remain unmapped to any physical resource in a given context. *However*, unmapped resources cannot be used in that context.
- A single physical resource may be mapped in several contexts.
- In a given context, a logical resource is mapped at the most to one physical resource.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

You may wonder what would happen if you had a large number of logical resources and a large number of contexts. Would you have to map every single logical resource in every context?

The answer is no. You can leave logical resources unmapped in any context. However, when executing an object in that context, unmapped logical resources cannot be reached.

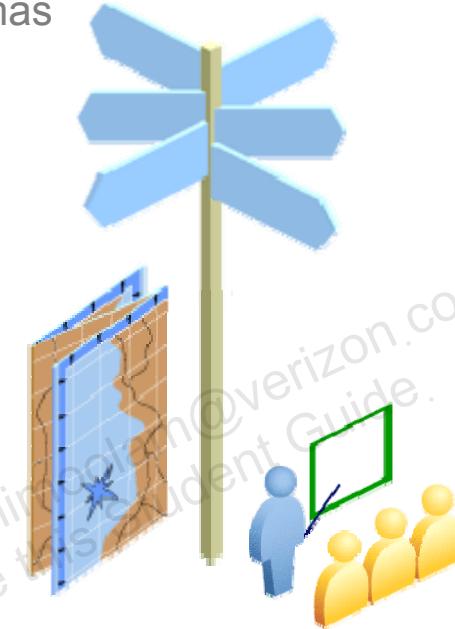
In addition, one single physical resource may be useful in several contexts. For example, you may want to retrieve information from the company web server in various contexts. This does not pose a problem. You simply map it to a logical resource in each context.

There is a restriction regarding contexts and mapping logical and physical resources. In a given context, a logical resource can be mapped to one physical resource at the most. That is, in a certain context, the data warehouse logical schema cannot be mapped to the data warehouses in Boston and Tokyo simultaneously.

This means that when you ask for one logical resource in one context, you will always have access to one physical resource if the logical resource is mapped in the context.

Agenda

- ODI Topology: Overview
- Data Servers and Physical Schemas
- Defining Topology
- **Agents in Topology**
- Planning a Topology



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The following slides cover defining agents in your topology. In any production environment, you have at least one agent running in ODI.

ODI Physical Agents



- Agents are lightweight runtime components.
 - Can start execution on demand or on schedule
 - Can be installed on any machine
- Agents orchestrate the integration process.
 - Send generated code to be executed by data servers
 - Update the execution log
- Agents must be declared in the topology.
 - Physical agents represent components running at run time.
 - Physical agents must also be abstracted as logical agents.
- Three types of agents:
 - The Java EE agent can be deployed as a web application and benefit from the features of an application server.
 - The stand-alone agent runs in a simple JVM and can be deployed where needed to perform integration flows.
 - Collocated stand-alone has WLS installed but ignores it.

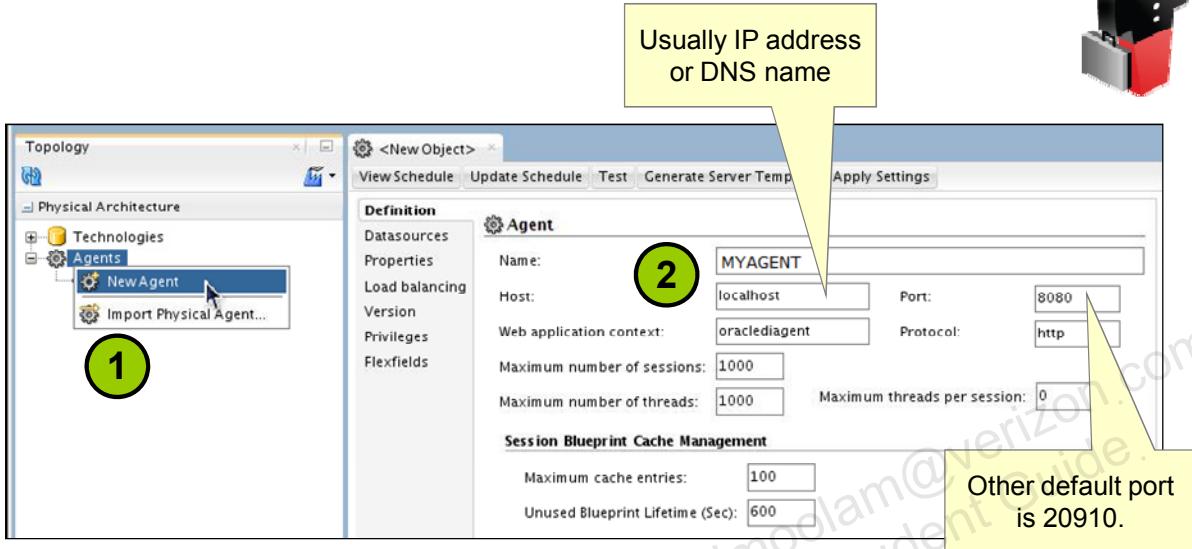
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The physical agent is a Java service, which can be placed as a listener on a TCP/IP port. An agent orchestrates the entire process of integration. It carries out data transformations by sending generated code to the relevant technologies. This could mean sending SQL statements to a data server, shell commands to the operating system, or even Simple Mail Transfer Protocol (SMTP) commands to an email server.

Just like data servers, agents are a part of your topology. Physical agents in the topology correspond to the agents that run at run time. However, you must also define logical agents that are mapped onto physical agents through contexts.

Creating a Physical Agent



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Though you do not need to have an agent running before you declare it in ODI, the best practice is to have it running. The physical agent should be created with ODI Topology Navigator. Connect to the Topology Navigator. Log in to the Master Repository.

1. Right-click the **Agents** node on the Physical Architecture tab and select **New Agent**.
2. Define the agent's parameters. These parameters essentially tell ODI how to find the agent on the network.

For agent name, you can specify anything. It is, however, recommended that you use the name of the machine, an underscore, and then the port that it is running on. For example: myhostA_8001. Then specify the host or IP address of the machine, and finally the port.

Note: If you want to set up load balancing, click the Load balancing tab and select a set of physical agents to which the current agent can delegate executions. When the agent has queued more than the specified number of sessions, it will reject further requests. You can also define other agents to which this agent can delegate sessions. Load balancing will be discussed in detail later in this lesson.

ODI Agent Parameters

- Agent parameters:

```
-port=<port>: Port on which the agent is listening (default port is 20910)
-help: Displays the options and parameters that can be used, without launching the
      agent
-name=<agent_name>: The name of the physical agent used
-v=<trace_level>: Enables the agent to generate traces (verbose mode)
```

- Work Repository connection parameters:

```
-SECU_DRIVER=<driver_name>: JDBC driver to access the Master Repository
-SECU_URL=<url>: JDBC URL to access for the Master Repository
-SECU_USER=<user>: User of the Master Repository connection
-SECU_PASS=<password>: Password of the user of the Master Repository.
      This password must be encrypted.
-WORK_REPOSITORY=<work_repository_name>: Name of the Work Repository
      containing scenarios to be executed
```

Connection parameters are specified in the `odiparams` file.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

These parameters are generally stored in the `odiparams` file for use by command-line invocation. This slide lists the ODI agent parameters that enable the agent to be configured. The parameters are preceded by the “-” dash character and the possible values are preceded by the “=” equals character. The ODI agent parameters include:

- -port=<port>: Is the port on which the agent is listening. If this parameter is not specified, the agent runs as a listener on the default port 20910.
- -help: Displays the options and parameters that can be used, without launching the agent
- -name=<agent_name>: Is the name of the physical agent used. ODI needs this parameter to identify the agent that executes a session in the following cases:
 - More than one agent is declared on the same machine (on different ports).
 - Your machine's IP configuration is insufficient to enable ODI to identify the agent (this problem occurs frequently on AS/400).
 - The agent's IP address does not enable the agent to be identified (127.0.0.1 or “loopback”).
- -v=<trace_level>: Enables the agent to generate traces (verbose mode). There are five trace levels:
 - Displays the start and end of each session

- Displays level 1, and the start and end of each step
- Displays level 2, and each task executed
- Displays the SQL queries executed
- Complete trace, generally reserved for support

Some of these traces can be voluminous and so it is recommended that you redirect them to a text file by using the following command:

- In Windows: agent.bat "-v=5" > trace.txt
- In UNIX: agent.sh -v=5 > trace.txt

- Work Repository connection parameters. These parameters, which are used to specify the connection to the Work Repository, are specified in the `odiparams` file.
 - `-SECU_DRIVER=<driver_name>`: JDBC driver to access the Master Repository, for example: `oracle.jdbc.driver.OracleDriver`
 - `-SECU_URL=<url>`: JDBC URL to access for the Master Repository, for example, `jdbc:oracle:thin@localhost:1521:XE`
 - `-SECU_USER=<user>`: User of the Master Repository connection (the database user)
 - `-SECU_PASS=<password>`: Password of the user of the Master Repository connection. This password must be encrypted by using the command `agent ENCODE <password>`.
 - `-WORK_REPOSITORY=<work_repository_name>`: Name of the Work Repository containing the scenarios to be executed

Launching a Stand-Alone Agent: Examples

- Windows:

```
C:\> Agent.bat "-port=20300" "-v=5"
```

Launches the stand-alone agent on port 20300 with a level 5 trace on the console.

- UNIX:

```
[bin] $ ./agent.sh "-port=20300" "-name=Agent001"
```

Launches the stand-alone agent and names it Agent001.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note: Each agent will have its own directory.

By default, the agent .sh files are in <odi_home>/oracledi/agent/bin.

Resiliency

ODI agents can optionally use WLS NodeManager to guarantee high availability, failover, and restart.

Stopping the ODI Agent



- Windows:

```
C:\> agentstop "-PORT=20300"
```

Stops the Listener agent on port 20300.

- UNIX:

```
[bin]$ ./agentstop.sh
```

Stops the Listener agent on the default port.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can gracefully stop a stand-alone ODI agent, which is listening on a TCP/IP port by using the `agentstop` command (or if you want to kill it, simply press `Ctrl + C`).

To stop an agent:

1. Open a UNIX shell, CMD, or QSH (for AS/400) session.
2. Launch the appropriate command file (`agentstop.bat` on Windows, `agentstop.sh` on UNIX or AS/400-QSH), with any required parameters. The listening agent is stopped.

Note: For security reasons, you can stop an agent only from a command line launched on the same machine from which the agent's process was started. You cannot stop a remote agent.

Deploying and Configuring a Java EE Agent

The deployment features:

- Oracle WebLogic Server Integration
- Java EE Agent Template Generation
- Oracle WebLogic Configuration Wizard
- Automatic Data Source Creation for WebLogic Server



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

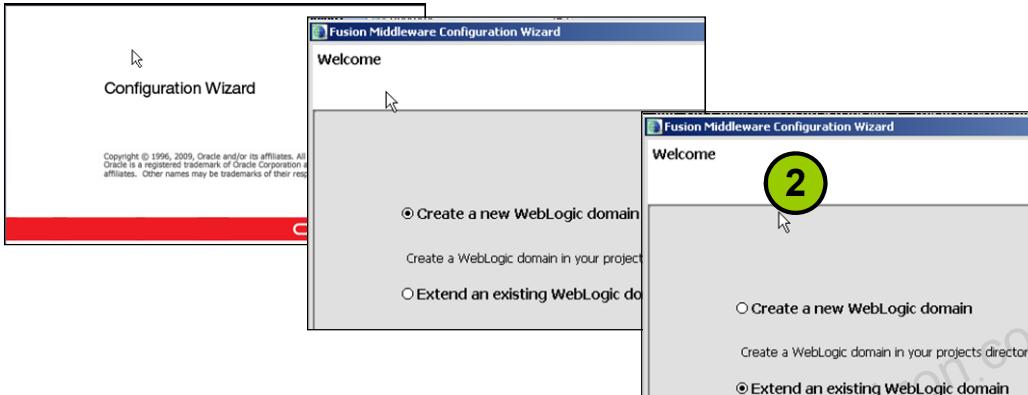
The runtime agent can be deployed as a Java EE component within an application server. It benefits in this configuration from the application server layer features such as clustering and connection pooling for large configurations. This Java EE agent exposes an MBeans interface, enabling lifecycle operations (start/stop) from the application server console and metrics that can be used by the application server console to monitor the agent activity and health.

- **Oracle WebLogic Server Integration:** Oracle Data Integrator components integrate seamlessly with the Java EE application server.
- **Java EE Agent Template Generation:** Oracle Data Integrator provides a wizard to automatically generate templates for deploying Java EE agents in Oracle WebLogic Server. Such a template includes the Java EE agent and its configuration, and can optionally include the JDBC data source definitions required for this agent, as well as the drivers and library files for these data sources to work.
- **Oracle WebLogic Configuration Wizard:** By using the Oracle WebLogic Configuration Wizard, domain administrators can extend their domains or create a new domain for the Oracle Data Integrator Java EE runtime agents.

Deploying and Configuring a Java EE Agent

```
[OS prompt]$ cd $FMW_HOME/odi/common/bin  
[OS prompt]$ ./config.sh
```

1



```
[OS prompt]$ cd $FMW_HOME/user_projects/domains/my_dom/bin  
[OS prompt]$ ./startWebLogic.sh  
(...many lines omitted for clarity...)
```

3

```
[OS prompt]$ ./startManagedWebLogic.sh odi_server  
(...many lines omitted for clarity...)
```

4

ORACLE

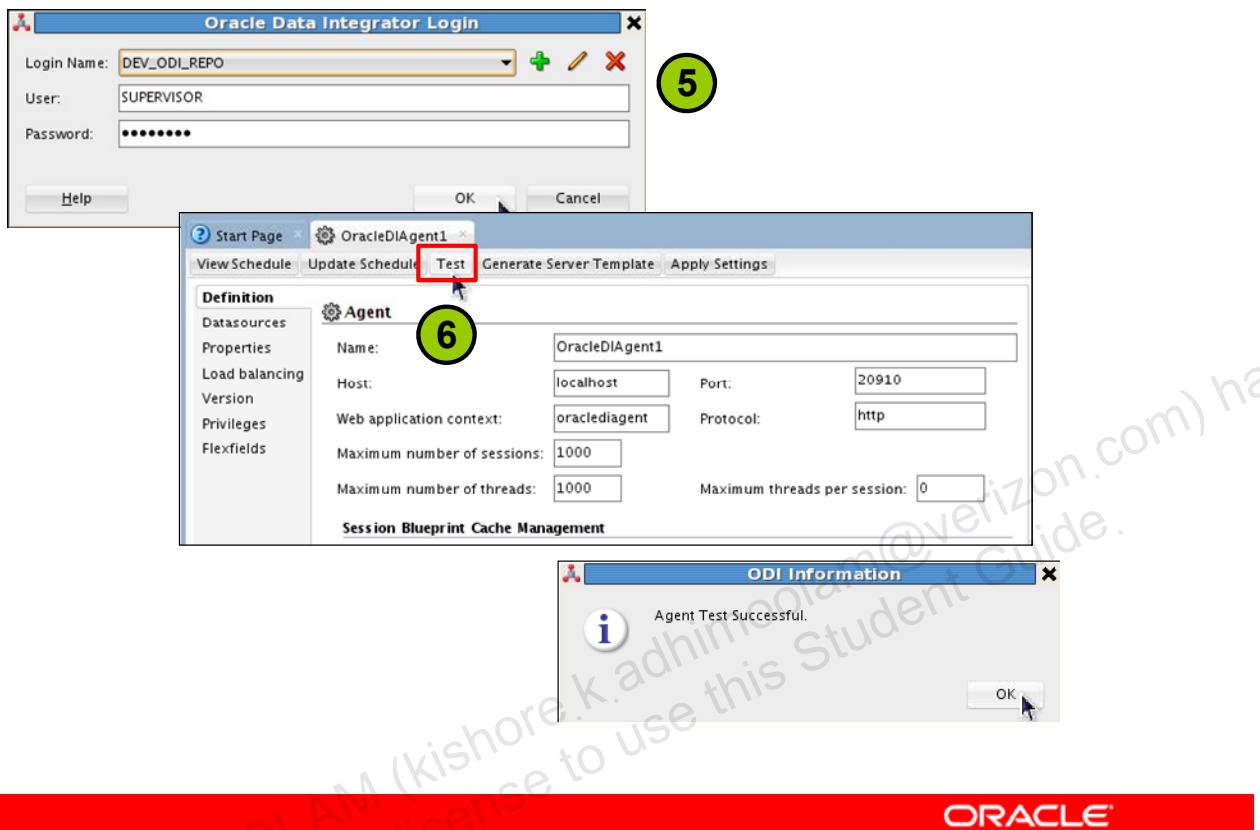
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Automatic Data Source Creation for WebLogic Server: Java EE Components use JDBC data sources to connect to the repositories as well as to the source and target data servers, and benefit, when deployed in an application server, from the connection pooling feature of their container.

To facilitate the creation of these data sources in the application server, Oracle Data Integrator Studio provides an option to deploy a data source into a remote Oracle WebLogic application server.

1. To deploy and configure domains with WLS, start the `config.sh` file, which is found in the ODI Home Install subfolder. On the Configuration Wizard Welcome screen, select “Create new WebLogic domain.”
2. Extend the domain with the `ODI_AGENT` application.
3. Start the Admin Server.
4. Start your WebLogic domain from the command shell.

Deploying and Configuring a Java EE Agent



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

5. Start ODI and connect to the Work Repository.
6. Test the connectivity of the ODI agent using **Topology Navigator > Physical Architecture > Agents**. (There are several places from which you can start the Test.)

Now you can start executing your ODI objects with the configured Java EE agent.

Load Balancing: Example

The figure consists of two side-by-side screenshots of the Oracle Data Integrator (ODI) interface, specifically the 'Topology' view. Both screenshots show the 'Physical Architecture' tree on the left with nodes for Technologies, Agents, and specific agents like OracleDIAgent1, agent86, and agent99. The right pane shows the configuration for each agent's 'Load balancing' tab.

Screenshot 1 (Top): Configuration for agent86

- The 'Linked Agent name' section shows three checked boxes: OracleDIAgent1, agent86, and agent99.
- A yellow callout box to the right states: "agent86 is linked to OracleDIAgent1 and to itself."

Screenshot 2 (Bottom): Configuration for agent99

- The 'Linked Agent name' section shows three checked boxes: OracleDIAgent1, agent86, and agent99.
- A yellow callout box to the right states: "agent99 is linked to agent86, OracleDIAgent1 and to itself."

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ODI implements load balancing between physical agents. Each physical agent is defined with the maximum number of sessions it can execute simultaneously and optionally with a number of linked physical agents to which it can delegate sessions' executions. An agent's load is determined at a given time by the ratio (number of running sessions/maximum number of sessions) for this agent.

The maximum number of sessions is a value that must be set depending on the capabilities of the machine running the agent. It can also be set depending on the amount of processing power you want to give to the Data Integrator agent.

Delegating Sessions

When a session is started on an agent with linked agents, ODI determines which one of the linked agents is less loaded, and the session is delegated to this linked agent. If the user parameter "Use new load balancing" is in use, sessions are also rebalanced each time a session finishes. This means that if an agent runs out of sessions, it will possibly be reallocated some sessions from another agent.

Note: An agent can be linked to itself. An agent not linked to itself can only delegate sessions to its linked agents, and will never execute a session. Delegation works on cascades of linked agents. Besides, it is possible to make loops in agents' links. This option is not recommended.

Agent Unavailable

When for a given agent the number of running sessions is equal to its maximum number of sessions, the agent will set incoming sessions in a “queued” status until the number of running sessions falls below the maximum number of sessions for this agent.

To set up load balancing:

- Define a set of physical agents and link them to a root agent (see “Create a Physical Agent”).
- Start the root and linked agents.
- Run the executions on the root agent. ODI will balance the load of the executions between its linked agents.

Note: You can see the execution agent for a session in the session window in Operator.

Important Guideline 5



Guideline 5: You need one physical agent defined in ODI for every agent you have started.

ORACLE®

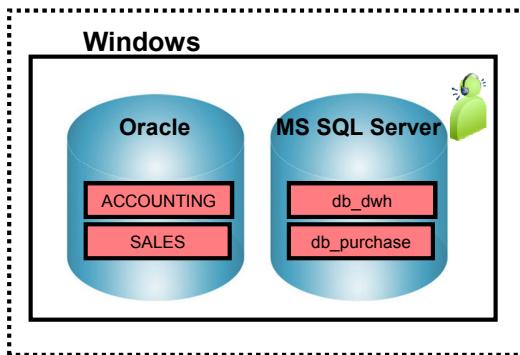
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Important Note 5

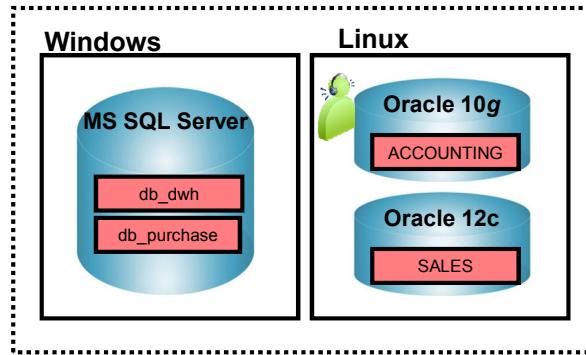
For every agent that you have started, you need to create one physical agent in the topology.

Infrastructure with Agents: Example

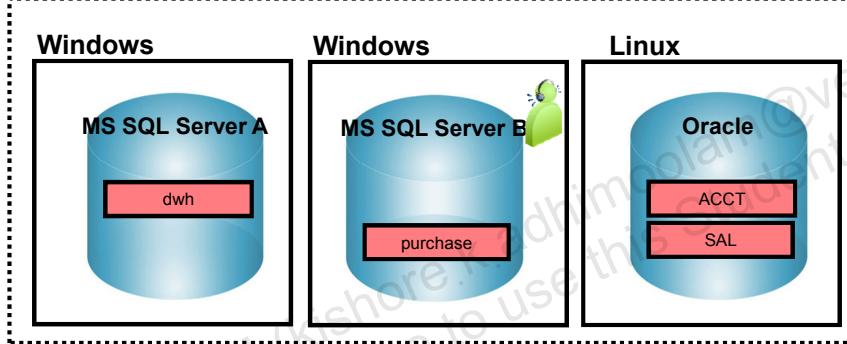
Development site: New York



Production site: Boston



Production site: Tokyo



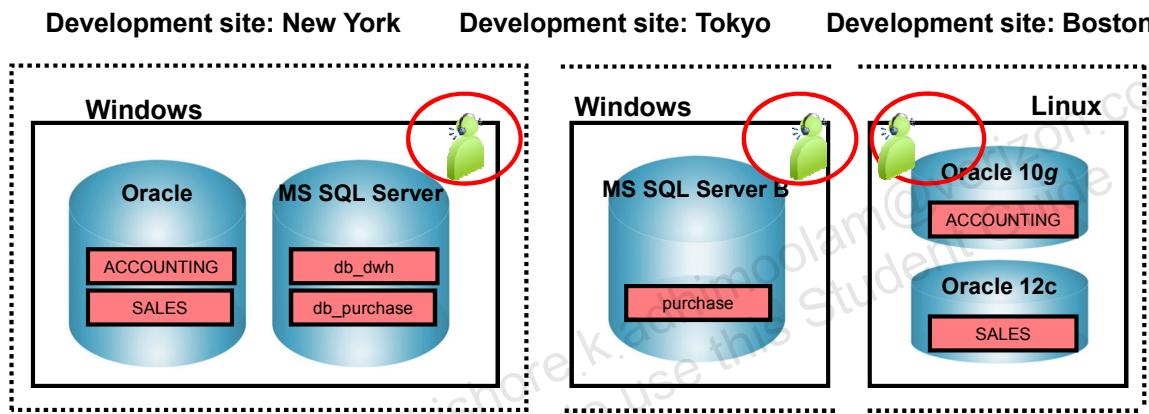
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Now you add the agents that you have installed to the graphic. Here, you have one agent in each context. In the New York and Tokyo sites, the agent runs on a Windows machine. At the Boston site, it runs on Linux.

Defining Agents: Example

- In ODI, you need to define each of these physical agents: one per agent launched.
- One logical agent (depending on the context) is mapped onto one of the physical agents.



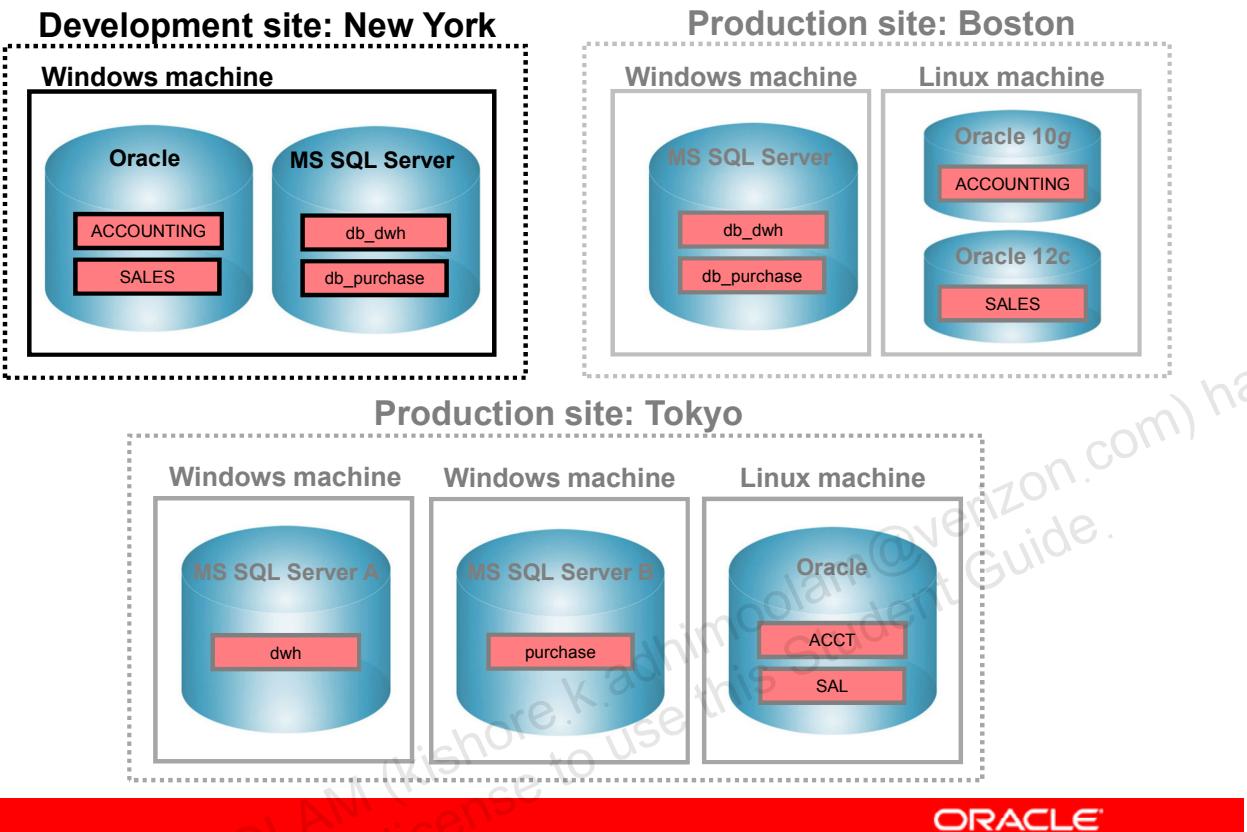
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In ODI, you must define each of these agents as a physical agent. Then you must define a logical agent. For example, for the “Development” context, you should map it onto the agent running at the New York development site.

Note: The logical agent is selected when processes are started. The execution context then defines which physical agent will orchestrate the process.

Special Case: Fragmentation Problem

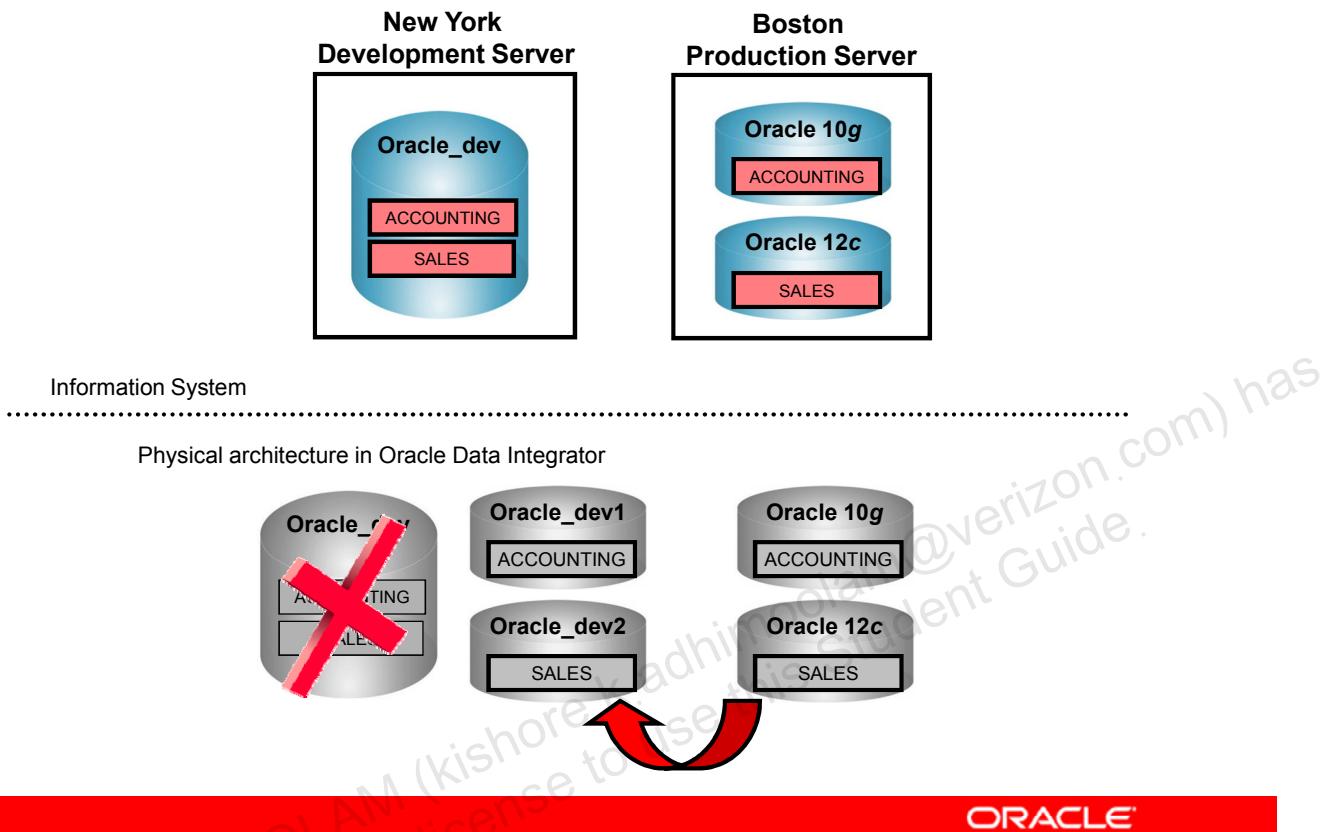


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Returning to the earlier example of the infrastructure for two production sites, the development site must now be added. The site where most of the ODI development occurs is based in New York. This site has access to the development versions of the databases. For cost reasons, all databases at this site are hosted on a single Windows machine. This machine runs one Oracle instance and one SQL Server instance.

Special Case: Fragmentation Problem



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You now see how to define the development databases in ODI.

You know that the production site in Boston has the accounting and sales schemas on two separate Oracle instances. A data server was created for each schema in the ODI topology. However, at the development site in New York, a single Oracle instance, **Oracle_dev**, contains the two schemas. Normally, you create a single data server in the ODI topology with two physical schemas.

When developing the integration processes, you can take advantage of the fact that the accounting and sales databases are on the same Oracle instance. However, when you try to put these into production on the Boston site, the processes will not work because the development environment does not reflect the production environment. For example, a join in the source in New York might work, though it fails when moved into production in Boston.

Therefore, you must develop the processes such that they reflect the production environment closely. That is, the development site must reflect the fragmentation of databases at the production site.

You need to define two data servers at the Boston site. In reality, both of them point to the same server. However, by separating them this way, you prevent the developers and ODI from making false assumptions.

Special Case: Important Guideline 6



Guideline 6: The topology of the development environment should always reflect the most fragmented production environment.

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This problem and solution is generalized as guideline number six. When you set up the topology for the development environment, it should mirror the most fragmented production environment. That is, if a production environment has four schemas on four different servers, the development environment should be defined in the same way. If several production environments are fragmented in different ways, the development environment should take all these fragmentations into account.

This is an exception to guideline one, which states that data servers should be defined only once.

Special Case: Defining the Physical Architecture

- Data servers
 - Guideline 1: One data server per Oracle instance or MS SQL Server
 - Tokyo: 3 + Boston: 3
 - Guideline 6: Reflect the fragmented production environment.
 - New York (development): 2 Oracle data servers + 2 MS SQL data servers
 - Total: 10 Data Servers
- Physical schemas
 - Guideline 2: One physical schema per Oracle schema or MS SQL database/owner
 - Tokyo: 4 + Boston: 4 + New York: 4
 - Total: 12 physical schemas



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Here is how you define this situation in ODI. First, how many data servers do you need?

Guideline 1 states that you must have one data server for every instance of a server. In this example, you need to have three servers in Tokyo and three in Boston.

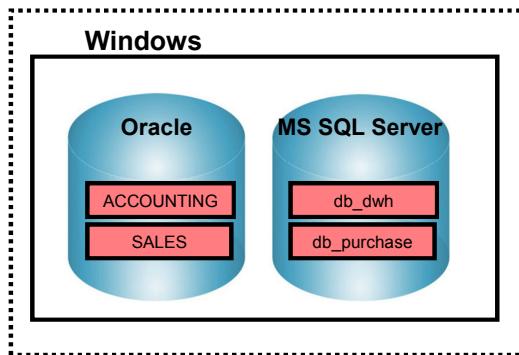
If you apply guideline 6, you get two Oracle servers to reflect the fragmentation of the two Oracle servers in Boston. Similarly, you have two SQL Servers because the two SQL Servers in Tokyo are split. That gives a total of 10 data servers.

How many physical schemas do you need?

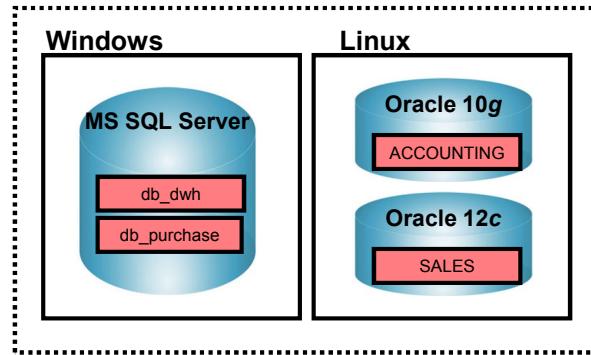
Guideline 2 states that each physical schema used by ODI must be defined. You have two Oracle schemas and two MS SQL database/owner combinations in New York, Boston, and Tokyo. That gives 12 physical schemas in all, each attached to the respective data server.

Special Case: The Infrastructure

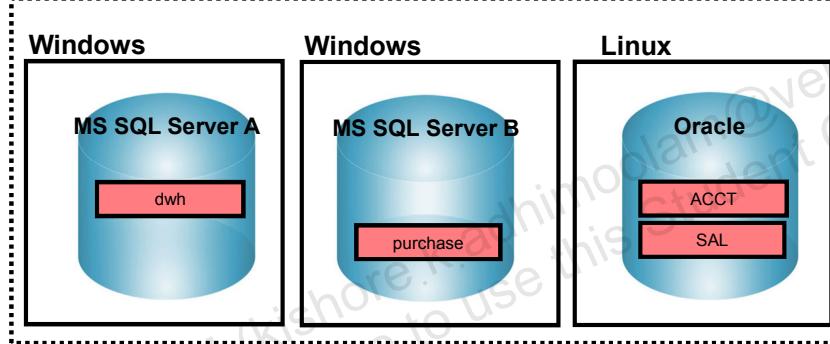
Development site: New York



Production site: Boston



Production site: Tokyo

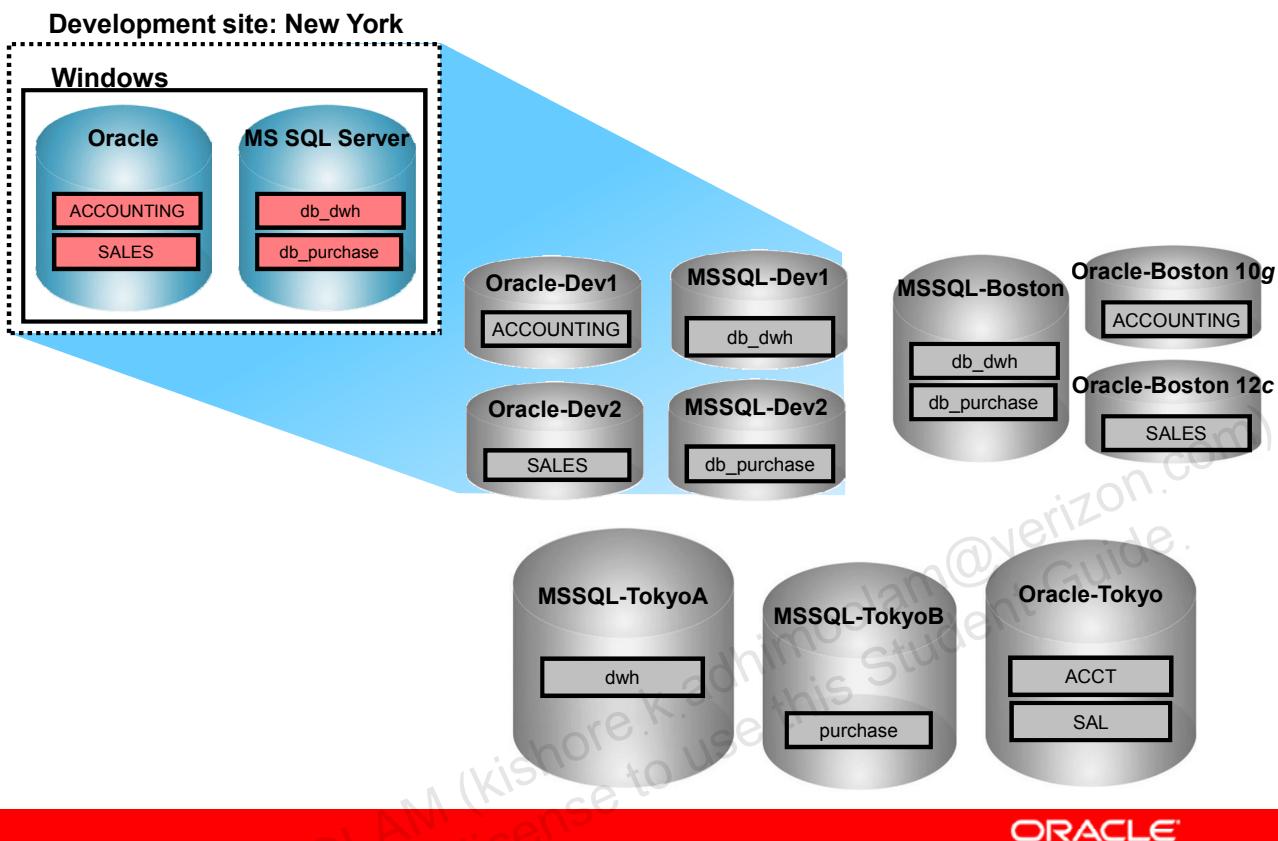


ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Eight servers are in the information infrastructure.

Special Case: Physical Architecture in ODI

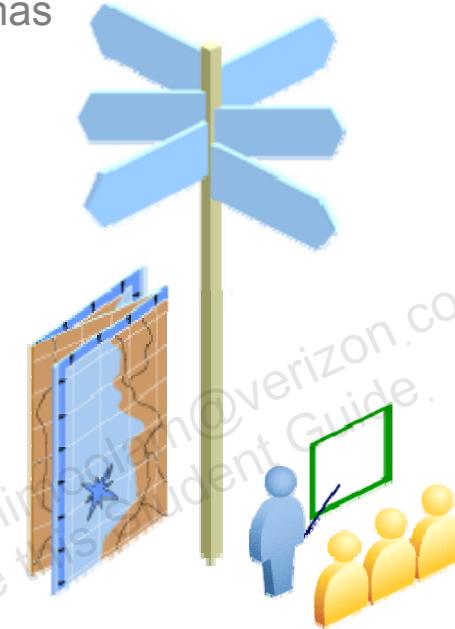


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The eight servers in the infrastructure are defined in ODI as 10 data servers containing 12 physical schemas.

Agenda

- ODI Topology: Overview
- Data Servers and Physical Schemas
- Defining Topology
- Agents in Topology
- **Planning a Topology**
 - **Best Practices**



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the following slides, you learn how to set up a topology to capture your information system in ODI.

Planning the Topology

1. Identify the physical architecture.
 - All data servers
 - All physical schemas
 - Required physical agents
2. Identify the contexts you have by looking for similar data schemas and agents in different situations.
3. Define the logical architecture by naming:
 - The logical schemas
 - The logical agents
4. On paper, write a matrix of the logical and physical mappings.
 - This matrix helps you plan your topology.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To plan your topology:

1. Identify the physical architecture that you have in place. All physical data servers and schemas need to be defined in ODI. Similarly, consider the agents you would need and the machines on which they would be located.
2. Identify the different contexts you have by looking for similar data schemas and agents in different situations.
3. Provide names for those similarities to create a logical architecture. Provide a name for each logical schema and logical agent.
4. Finally, write a matrix of the mappings from your logical architecture to your physical architecture. (Use pen and paper.) The next slide shows an example of a matrix.

Matrix of Logical and Physical Mappings

		Logical schemas	
Contexts	Accounting	Sales	
Development	ACCOUNTING in Oracle on Windows	SALES in Oracle on Windows	...
Tokyo	ACCT in Oracle on Linux
...

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This is a simple matrix of mappings from the logical architecture to the physical architecture.

1. In horizontal order, make a list of all the logical schemas (Accounting, Sales).
2. In vertical order, write down all the contexts that you have defined (Development, Tokyo).
3. Fill in the squares. For each combination of logical schema and context, write down the name of the physical schema, the data server, and the technology.

Quiz

You use a data server that has five technology-specific subdivisions. You want to use one of them in ODI. How many physical schemas should you define for that server?

- a. One
- b. Two
- c. Three
- d. Five



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Explanation: For each data server in your topology, you must define a physical schema to represent each subdivision of the server that will be used in ODI.

Quiz

In a given context, how many physical resources can a logical resource be mapped to at the most?

- a. None
- b. One
- c. Two
- d. Any number



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Explanation: A context maps individual logical resources onto individual physical resources. So, given a context and a logical resource, ODI can determine the unique physical resource that is appropriate.

Quiz

In how many contexts can a physical schema be used?

- a. None
- b. One
- c. Two
- d. Any number



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: d

Explanation: There is no limit (actually the limit is the number of contexts).

Summary

In this lesson, you should have learned how to:

- Describe the basic concepts of a topology
- Describe the logical and physical architecture
- Plan a topology
- Use best practices to set up a topology
- Launch ODI agents and set agent parameters



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Checklist of Practice Activities

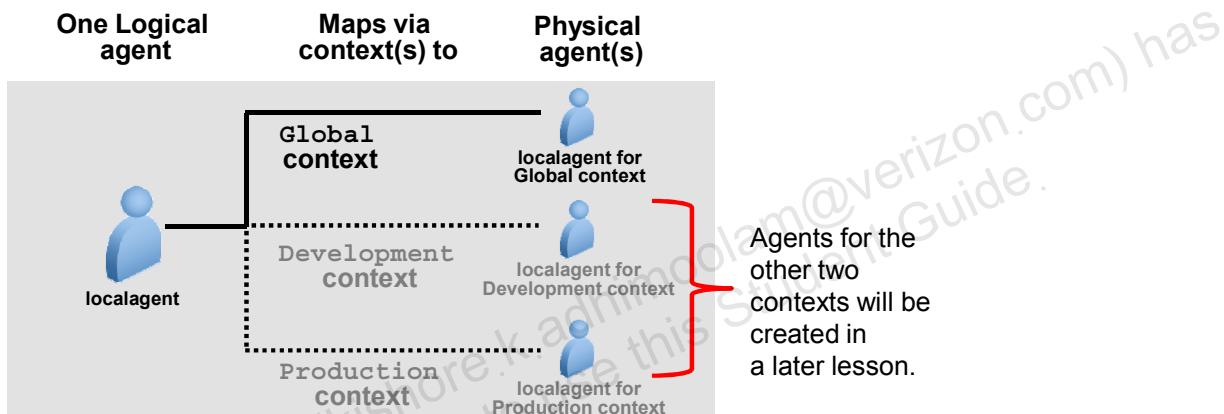
- ✓ 1-1: Logging In and Help Overview
- ✓ 2-1: Creating and Connecting to ODI Master and Work Repositories
- 3-1: **Configuring Standalone Agents Using the Common Admin Model (CAM)**
 - 4-1: Working with Topology
 - 5-1: Setting Up a New ODI Project
 - 6-1: Creating Models by Reverse-Engineering
 - 7-1: Checking Data Quality in the Model
 - 8-1: Creating ODI Mapping: Simple Transformations
 - 9-1: Creating ODI Mapping: Complex Transformations
 - 9-2: Creating ODI Mapping: Implementing Lookup
 - 10-1: Creating ODI Mapping: Exporting a Flat File to a Relational Table
 - 11-1: Using Native Sequences with ODI Mapping
 - 11-2: Using Temporary Indexes
 - 11-3: Using Sets with ODI Mapping
 - 12-1: Creating and Using Reusable Mappings
 - 12-2: Developing a New Knowledge Module
 - 13-1: Creating an ODI Procedure
 - 14-1: Creating an ODI Package
 - 14-2: Using ODI Packages with Variables and User Functions
 - 15-1: Debugging Mappings
 - 16-1: Creating and Scheduling Scenarios
 - 17-1: Using Load Plans
 - 18-1: Enforcing Data Quality with ODI Mappings
 - 19-1: Implementing Changed Data Capture
 - 20-1: Setting Up ODI Security
 - 20-2: Integration with Enterprise Manager and Using ODI Console

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 3-1: Configuring a Standalone Agent by Using the Common Administration Model

1. Run config Wizard.
2. In ODI, define a **physical** agent named *localagent*.
3. In ODI, define a **logical** agent named *localagent*.
4. Execute *agent.sh*, to create, install, and start an agent named *localagent*.
5. Verify connection to the newly created agent *localagent* in ODI.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A common task that is performed by using ODI is to set up and install ODI Agent as a service. After the ODI scenarios are created, they can be scheduled and orchestrated by using an ODI agent, which is a lightweight Java process that orchestrates the execution of ODI scenarios. In this practice, you learn how to set up and install an ODI agent, which will then be used in subsequent practices for orchestration of the execution of ODI objects.

Describing the Physical and Logical Architecture

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Use the Topology Navigator to create physical and logical architectures
- Create data servers and physical schemas
- Link the physical and logical architectures via contexts



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the previous lessons, you created Master and Work Repositories and installed an ODI agent as a background service. To complete setting up your ODI infrastructure, you need to create contexts, a data server, and physical and logical schemas.

This lesson introduces you to the Topology Navigator and takes you through the process of defining your physical architecture.

In this lesson, you gain a basic understanding of how to use the Topology Navigator to create your physical architecture in Oracle Data Integrator (ODI). You also learn to create data servers and define physical schemas on them, and to declare the physical agents so that ODI can use them to execute tasks.

Agenda

- **Topology Navigator**
 - Physical
 - Contexts
 - Logical
- Creating Physical Architecture
- Creating Logical Architecture



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You learn about the Topology Navigator and get an overview of what is represented in the ODI physical architecture in the next few slides.

What Topology Navigator Contains

- Physical architecture
 - Data servers, physical schemas, and agents
- Logical architecture
 - Logical schemas and agents, contexts
- Technology-related information
 - Technologies, languages, actions
- ODI architecture components
 - Repositories and agents



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Topology Navigator provides access to everything that is considered part of the ODI topology. The three principal elements of the physical architecture are the data servers, physical schemas, and physical agents.

The logical architecture provides the logical aliasing of physical objects—that is, logical schemas and logical agents, as well as the contexts that link logical objects with their physical counterparts.

Also, elements that define the technologies are available. Languages define the reserved keywords and functions available in the Expression Editor. Actions are used to generate data definition language (DDL) statements.

Topology Navigator also contains information specific to ODI architecture, including information about repositories and agents.

However, this lesson deals only with physical and logical architectures, including data servers, physical schemas and agents, logical schemas and agents, and contexts.

Topology Navigator: Overview

- Topology Navigator stores all its information in the Master Repository.
 - In contrast, Designer Navigator stores project-related information in Work Repositories.
- Making changes in Topology Navigator:
 - Affects all attached repositories, possibly other people's work
 - May affect the behavior of work in progress or running processes



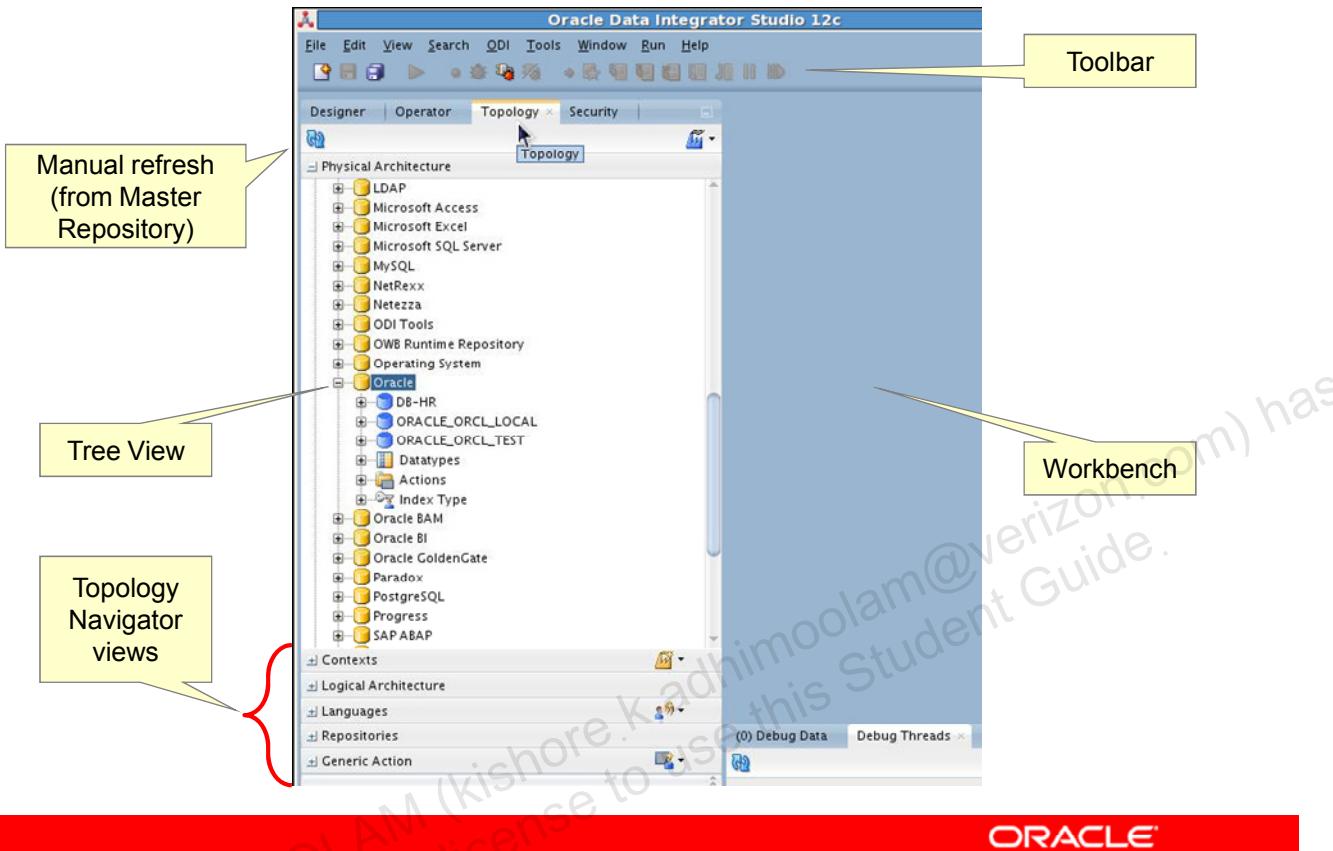
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

All that you can create or modify in Topology Navigator is stored in the Master Repository. This contrasts with Designer, which stores project-related information in Work Repositories.

However, be aware that any changes made in Topology Navigator immediately affect all the Work Repositories attached to it. Such changes may also affect other people's work.

For example, modifying the definition of a technology immediately affects any code that is subsequently generated for that technology. Similarly, modifying the connection parameters for a data server can prevent a scheduled scenario from connecting to that server.

Topology Navigator: Overview



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

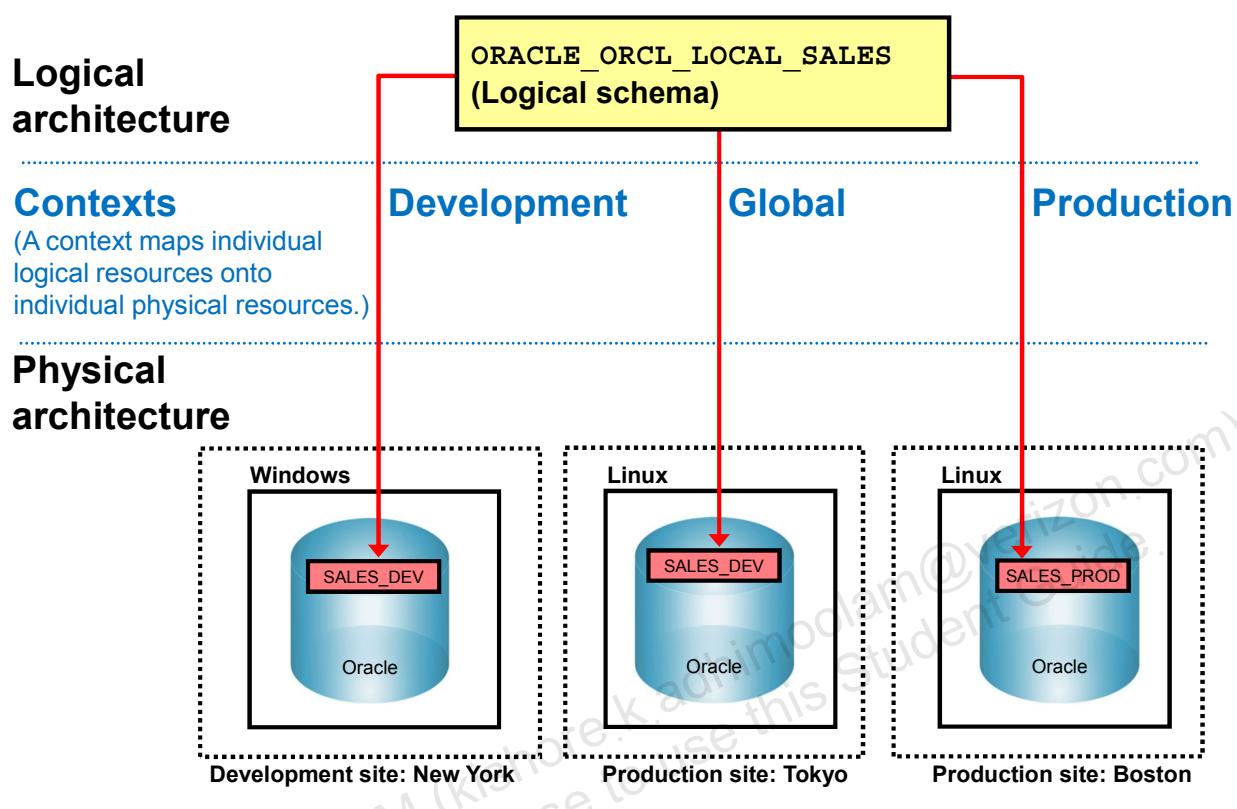
This is what the Topology Navigator looks like.

A number of tree views provide access to different kinds of information. You can expand, collapse, dock, undock, and rearrange these views to customize your work environment. For example, the slide shows the Physical Architecture view that lists data servers and physical schemas sorted by technology.

The background area on the right is referred to as the Workbench. When you double-click an element, its panel appears in this area.

At the top is the toolbar. Manual refresh refreshes the screen from the Master Repository.

Review: Context Connects Logical to Physical



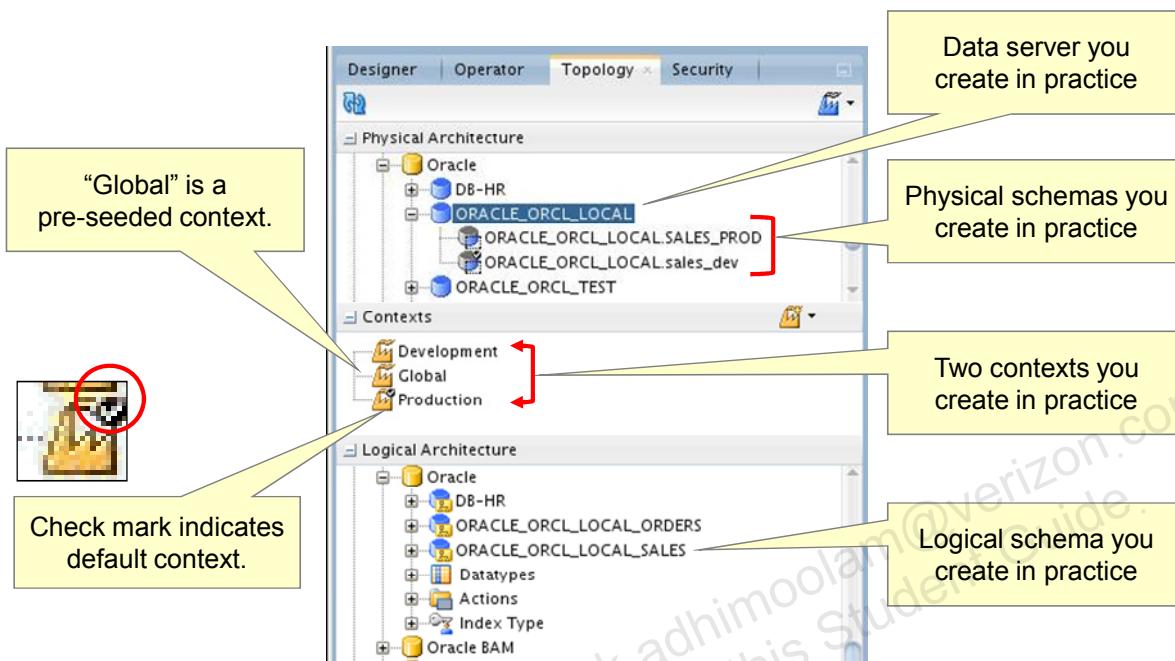
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A context is a set of resources that enables the operation or simulation of one or more data processing applications. Contexts enable the same jobs (Development, Test, Production, and so on) to be executed on different databases and/or schemas.

You will define contexts in the practice, and edit them to point logical schemas to physical schemas.

Objects You Create in the Practice



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In this lesson's practice, you first define two new contexts, Development and Production, in addition to the pre-seeded context, Global.

You then define a data server, ORACLE_ORCL_LOCAL. Next, you define two physical schemas for that data server, SALES_DEV and SALES_PROD.

Then you define a logical schema, ORACLE_ORCL_LOCAL_SALES.

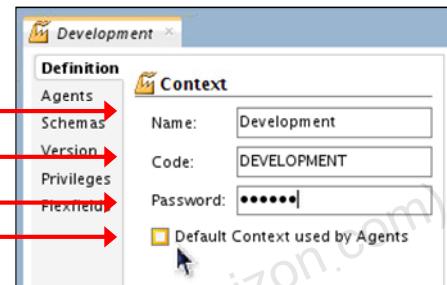
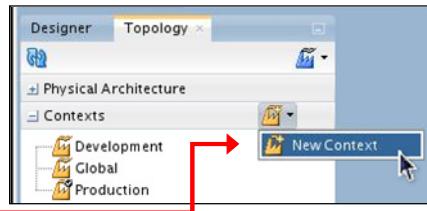
Next, you map your one logical schema to your two physical schemas, in terms of the three contexts.

Finally, you check the mappings of several other predefined logical schemas against predefined physical schemas, in terms of the three contexts.

Note: When executing any object, the Default context is the context selected by default in the Execution dialog box. If an invalid context name is specified, the default context in Designer is used.

Defining a Context

1. Go to the Topology > Contexts bar pull-down.
2. Click **New Context**.
3. Enter:
 - The name of the context
 - The code for the context
 - The password (optional)
4. If you want this context to be the default, select the **Default** check box.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To define a context, perform the following steps:

1. Go to the Contexts tab of the Topology Navigator.
2. Click the **New Context** button. This opens a new context window.
3. Give the context a name and a code. This is the name of the context as it appears in ODI. The code is used to make reference to the context.
4. You can also define a password that must be entered to launch a session in this context. To make the context the default context for all sessions launched, select the **Default** check box. This determines the execution context that is set by default each time Designer is loaded.

Note: Although you can specify a password for a context, it is not a best practice. The addition of context passwords for security would impose upon users the need to manage additional passwords in addition to their login password. Users should rely on standard ODI security features for password access control. Nevertheless, special cases might warrant the use of passwords to control context switching between, for example, Test and Development.

Agenda

- Topology Navigator
- **Creating Physical Architecture**
- Creating Logical Architecture

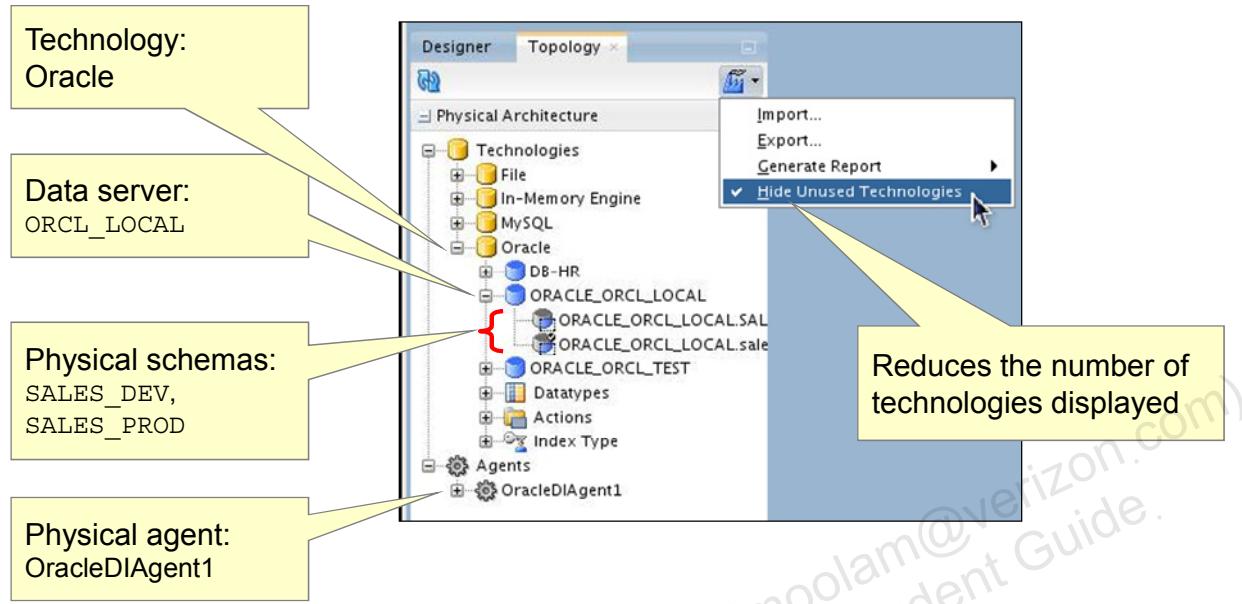


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the next few slides, you learn how to create physical architecture in ODI Topology Navigator.

Physical Architecture View



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Physical Architecture view is where you modify the definition of data servers, physical schemas, and physical agents.

This view is organized by technology. So, to see data servers and physical schemas based on Oracle, expand the **Oracle** node.

Under this node, all data servers are displayed. There are also data types, actions, and index types for the given technology.

Under a data server, you can access the physical schemas defined for the server. Here you have a single data server, ORACLE_ORCL_LOCAL, with three physical schemas named ORACLE_ORCL_LOCAL.ORDERS, ORACLE_ORCL_LOCAL.SALES_DEV, and ORACLE_ORCL_LOCAL.SALES_PROD.

All physical agents are displayed separately. These will be covered later in the lesson.

The number of technologies available in ODI can be somewhat overwhelming. Fortunately, you can make Topology Navigator display only technologies with at least one attached data server. To do this, select **Hide Unused Technologies** from the Topology Navigator's menu. Remember to redisplay the technologies when you want to add a data server to a new technology.

Prerequisites for Connecting to a Server

- Drivers (JDBC or JMS):
 - Drivers must be installed in the appropriate driver subdirectory for ODI Studio, the stand-alone agent, or the Java EE agent.
 - This installation must be performed on all the machines connecting to the data server.
 - Machines running an ODI GUI
 - Machines running an ODI Agent
- Connection settings (server-dependent):
 - Machine name (IP address), port
 - User/password
 - Instance/database name, if the server is a database server



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ODI connects to a data server by using standard Java connectivity methods. These methods usually require server-specific drivers.

For example, to connect to a database server supporting Java Database Connectivity (JDBC), ODI needs the database's JDBC driver. Similarly, to connect to a message-oriented middleware (MOM) router supporting the Java Message Service (JMS), it needs the MOM's JMS client.

These drivers must be installed in the driver's subdirectory. A driver is either a `.jar` or a `.zip` file. Driver installation should be performed for every machine that may connect to the data server with an ODI graphical user interface or an ODI agent.

You also need to specify appropriate settings to connect to the data server. The settings vary depending on the type of data server.

Typically, you need to specify an IP address or a machine name and a port to connect to the appropriate service on the machine. You also need a username and password with sufficient privileges on the server.

Lastly, you may need server-specific information, such as the database and/or instance name for a database server.

Important Note



- The username that you specify for a data server is used to access all underlying schemas, databases, or libraries in the data server.
- Ensure that this user account has sufficient privileges.
For database:
 - SELECT
 - INSERT/UPDATE
 - CREATE/DROP
 - READ

Additional Note: It is possible to define commands for a data server that will be automatically executed when connections to this data server are created or closed by ODI components or by a session.

ORACLE®

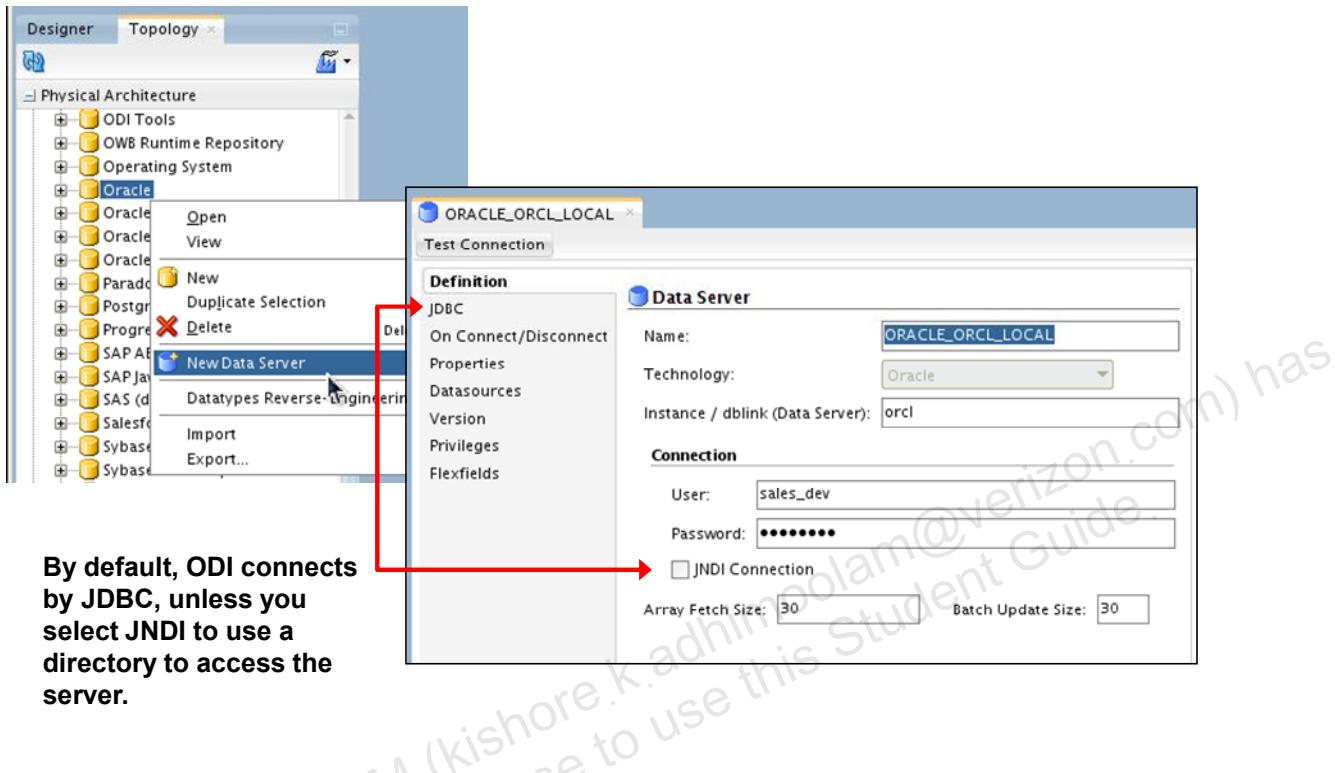
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Note that the username that you supply for a data server is used for many purposes by ODI. This includes accessing and altering data or data structure in schemas, databases, or libraries in the data server, depending on the technology. Therefore, you must ensure that the user account has sufficient privileges to be able to do this everywhere.

In the case of a database, the user must have `SELECT` privileges to all the schemas defined in the data server. The user must also have `INSERT/UPDATE` privileges to the tables into which ODI will integrate data. The user must have sufficient privileges to `CREATE/DROP` objects into temporary, or work, schemas, and should be able to `READ` the structure of the tables to reverse-engineer them.

Note: As a best practice, generally use the same owner as the schema that owns the Work schema.

Creating a Data Server



ORACLE

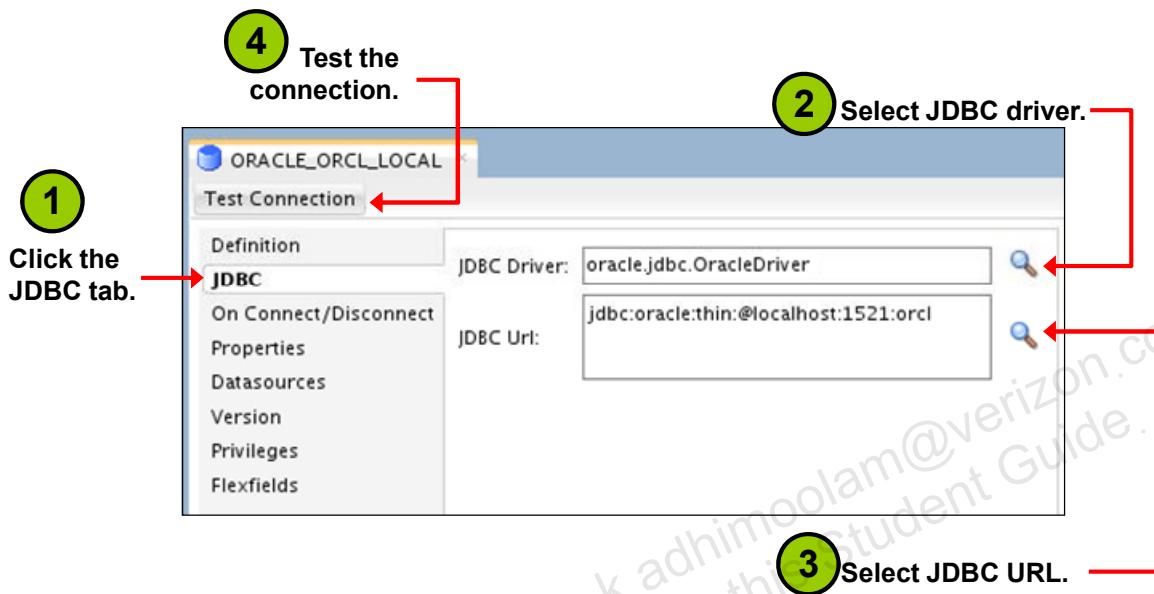
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a data server connection in ODI, perform the following steps:

1. In Topology Navigator, expand the Technologies node in the Physical Architecture panel.
2. Select the technology to which you want to attach your data server.
3. Right-click and select New Data Server.
4. Give a name to the data server. Prefix the name of the server with the name of the technology, for example, ORACLE_DW.
5. Enter the specific connection settings. Enter the username and password for a sufficiently privileged user. It is a good idea to create a user especially for ODI.
6. Optionally, you select the JNDI Connection check box to use a directory to gain access to the server. By default, ODI assumes that you want to connect directly to your data server by using JDBC. This happens if you do not select the JNDI Connection check box.

Note: The Data Server field is a technology-specific value that identifies the server. In many cases, this represents the server instance, as understood by the database engine.

Creating a Data Server: JDBC



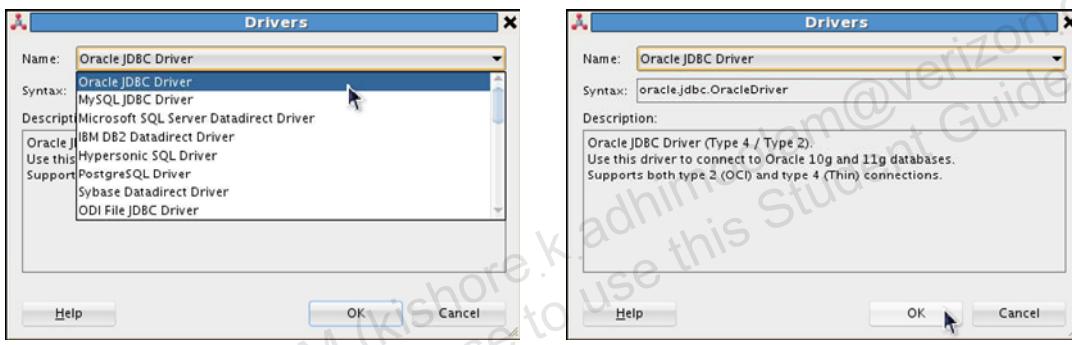
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. To complete the connection settings, click the JDBC tab.
2. You must specify the JDBC driver with which to access the defined technology. However, this must be correctly installed. You can use the browse button (magnifying glass) to select a driver from the list.
3. You then need to provide a URL that defines the location of the server. You can use the “select” button (magnifying glass) to choose an available URL format. Then, set the URL parameters with your server’s connection parameters.
4. Test the connection by clicking the Test Connection button. This operation will be covered in greater detail later in the lesson.

JDBC Driver

- A JDBC driver is a Java driver that provides access to a type of database.
 - Type 4: Provides direct access through TCP/IP (generally preferred type)
 - Type 3: Is specific to three-tier architectures
 - Type 2: Requires the database client layer
 - Type 1: Is the generic driver to connect ODBC data sources
- Drivers are identified by a Java class name.
- Drivers are distributed as .jar or .zip files.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

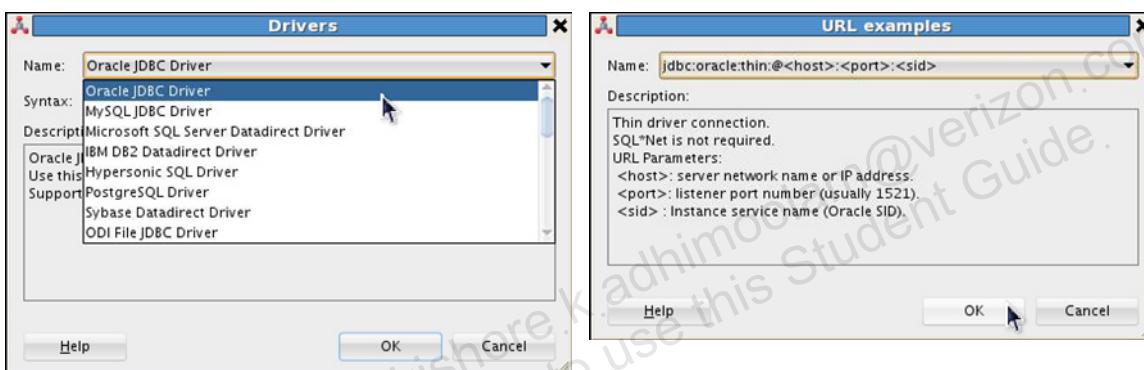
The JDBC driver is always a Java class. It provides access to a particular type of database. Four categories of databases require different levels of network infrastructure:

- **Type 4** drivers are generally preferred. They connect directly to the target machine over TCP/IP. They do not require any other component.
- **Type 3** drivers are specific to three-tier architectures. These drivers connect to an intermediate server that is used as a gateway to the target machine.
- **Type 2** drivers require the database-specific client layer to be present and correctly configured on the client machine. This can present difficulties when deploying scenarios onto distant servers, because each machine must be configured in exactly the same way.
- **Type 1** is a generic driver to connect to Open Database Connectivity (ODBC) data sources. It has the same problems as type 2 drivers, but can be even slower. This is why type 4 drivers are the best if they are available for the given technology.

When you specify the driver, you are actually specifying the name of a Java class. This class must be present in the Java classpath of ODI. The easiest way to ensure that it is on the classpath is to copy the driver's file—usually a .jar or .zip file—into the appropriate driver's subdirectory of the ODI installation directory.

JDBC URL

- The JDBC driver uses a URL to connect to a database system.
 - The URL describes how to connect to the database system.
 - The URL may also contain driver-specific parameters.
- Use the “select” button to choose the driver class name and the URL template.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

As mentioned earlier, the JDBC driver needs a special JDBC URL to connect to a particular database. This URL contains the information necessary to connect to this particular database system. For example, it often contains the IP address of the database server.

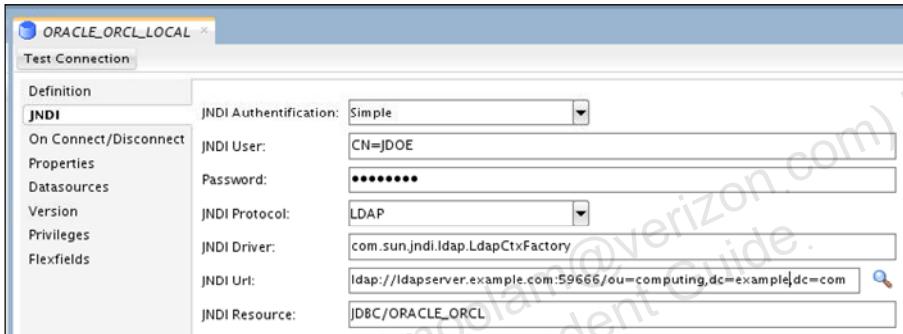
However, it can also contain other parameters defined specifically by the chosen JDBC driver. For example, the ODI file driver contains a parameter to specify the character encoding to use.

The “select” buttons (magnifying glass) next to the relevant fields on the JDBC tab enable you to fill in these fields easily. You could have selected the Sun JDBC-ODBC bridge, which is a standard driver built in Java. You can use it to connect to almost any ODBC data source. Note that when you select a URL template, a field is displayed for you to enter the name of the ODBC Data Source Name (DSN) alias.

Select the Oracle driver, and then click the button next to the JDBC URL field to select Oracle JDBC URL, as shown in the screenshot in the slide. Fill in the appropriate values for `<host>`, `<port>`, and `<sid>` to edit the URL. Be careful to preserve the separating colons.

Creating a Data Server: JNDI

1. Select JNDI on the Definition tab.
2. Click the JNDI tab.
3. Set the JNDI parameters:
 - Authentication
 - User
 - Password
 - Protocol
 - Driver
 - URL
 - Resource
4. Test the connection.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

If you selected the option to connect to your data server with JNDI by using a directory, the JDBC tab is replaced by the JNDI tab. Click this tab to begin defining the parameters. The meaning of these parameters depends on the type of directory you want to use and the JNDI driver. The parameters include the JNDI Authentication mode, a user and password, and the JNDI protocol.

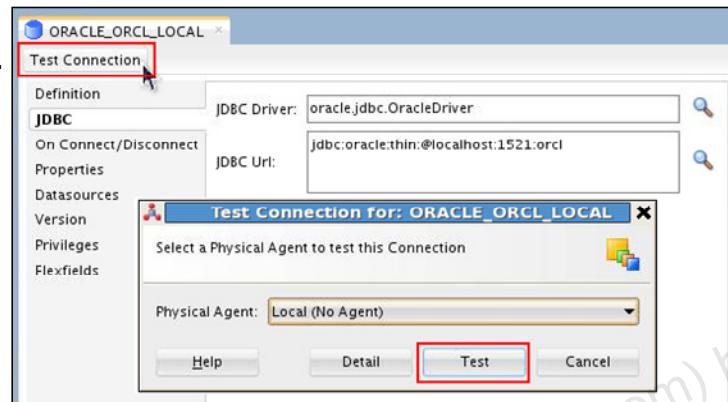
You must also specify the JNDI driver. The JNDI URL is specified in the same way as the JDBC URL, and depends on the type of directory connected. The JNDI Resource name is the directory resource corresponding to your data server. For database connections, it should correspond to a JDBC data source. For JDBC and JNDI connections, click **Test Connection**.

Click **OK** to save your settings.

Recall that when connecting a data server through a JNDI directory, you need both the JNDI directory driver and the data server driver (for example, a JDBC driver). Both these drivers should be installed in the driver's subdirectory.

Testing a Data Server Connection

1. Click **Test Connection**.
2. Select the agent to test this connection.
 - **Local (No Agent)** tests with the Topology Navigator.
3. Click **Test**.
4. This validates driver, URL, and network connectivity.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

After you have chosen a driver and set up a URL, you must test the connection. It is quicker to test the connection now than to find later that your connection parameters are wrong. To test, perform the following:

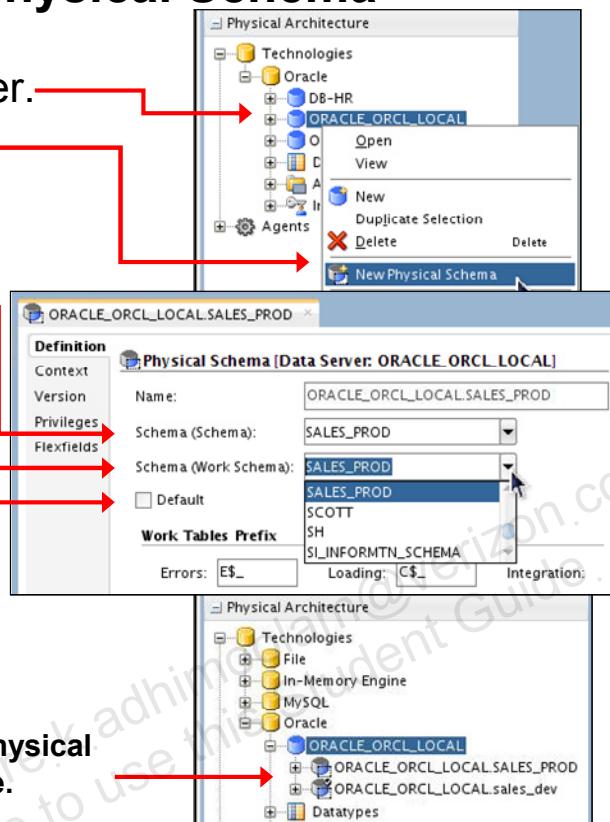
1. Click **Test Connection** in the data server window.
2. Be sure to test every data server connection for every agent you have defined. If you are connecting to a production data server, you should test to ensure that the agent that will be used in production can connect to it.
3. You can select **Local (No Agent)** to perform the test from your workstation. This setting uses the agent that is built into ODI, which is always available.
4. Click **Test** to launch the test. This validates the driver, the URL, and the network connectivity to the server. Generally, if the test is successful, a result is returned instantly. Note the second information window informing you to register at least one physical schema for your data server.

You can repeat the test with different agents.

Note: You must always test the connection to check that the data server is correctly configured.

Creating a Physical Schema

1. Right-click the data server.
2. Select New Physical Schema.
3. Pull down or fill in:
 - Data Schema
 - Work Schema
4. Select whether this is the Default schema.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Now that you have created a data server, the next step is to define the physical schemas in ODI. To do this:

1. Right-click the data server.
2. Select New Physical Schema. If you have just created a data server, this window opens automatically. The layout of the window depends on the technology. Certain technologies qualify each data server schema by using a catalog name and a schema name. Others use only one of these qualifiers, and some do not support multiple schemas at all.
3. In this window, you choose the data server schemas that you want to use as the data schema and work schema. To choose a data server schema, specify the catalog name, the schema name, both, or none. Remember that the data schema is where your data is stored and the work schema is a different schema where ODI stores temporary objects.
4. If you want to use the default schema for the data server, select the Default check box. This means that ODI will store temporary objects that belong to the entire data server in this work schema.

Note: The recommended practice in ODI is that you create a separate area on each data server specifically for ODI temporary objects—the dedicated area. You can then use this dedicated area as the work schema so that you have no risk of ODI temporary objects polluting your production data.

Agenda

- Topology Navigator
- Creating Physical Architecture
- **Creating Logical Architecture**



ORACLE

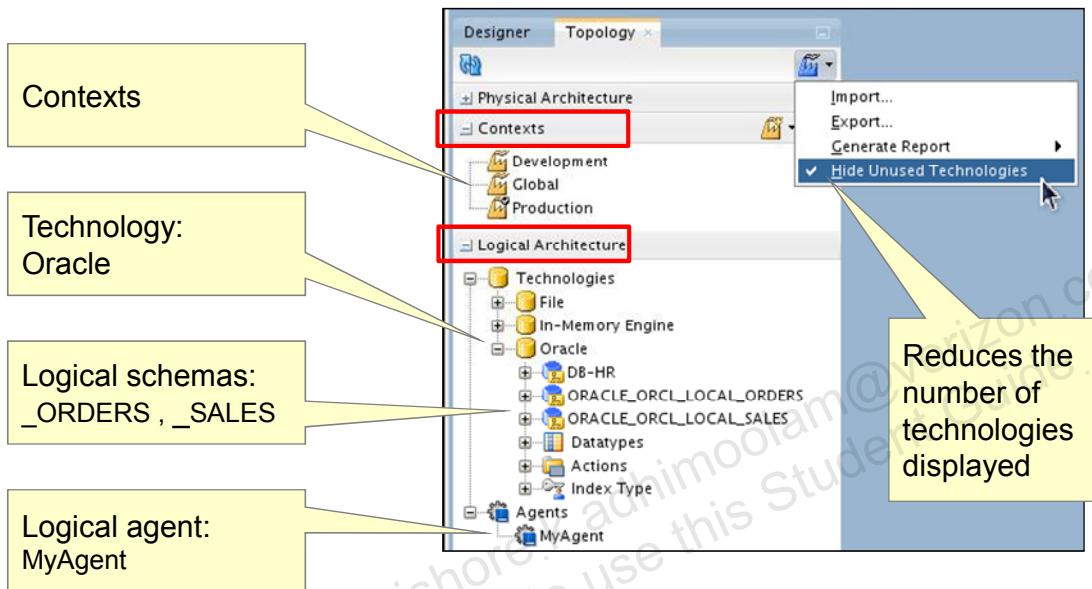
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You have defined a representation of your system's actual physical architecture.

Now, you define a higher-level logical representation of that architecture. You can then map the logical objects to specific physical versions of those objects, by way of the contexts that you defined.

Logical Architecture and Context Views

The same technologies are displayed in Physical and Logical Architecture views.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Logical Architecture view is where you modify the definition of the logical schemas and logical agents. The Logical Architecture is organized by technology. So, to see logical schemas based on Oracle, expand the Oracle node.

Under this node, all logical schemas are displayed. Here, you see two logical schemas, ORACLE_ORCL_LOCAL_SALES and ORACLE_ORCL_LOCAL_ORDERS, which are based on Oracle technology. You will define these two schemas in the practice.

The logical agent displayed will be covered at the end of this lesson.

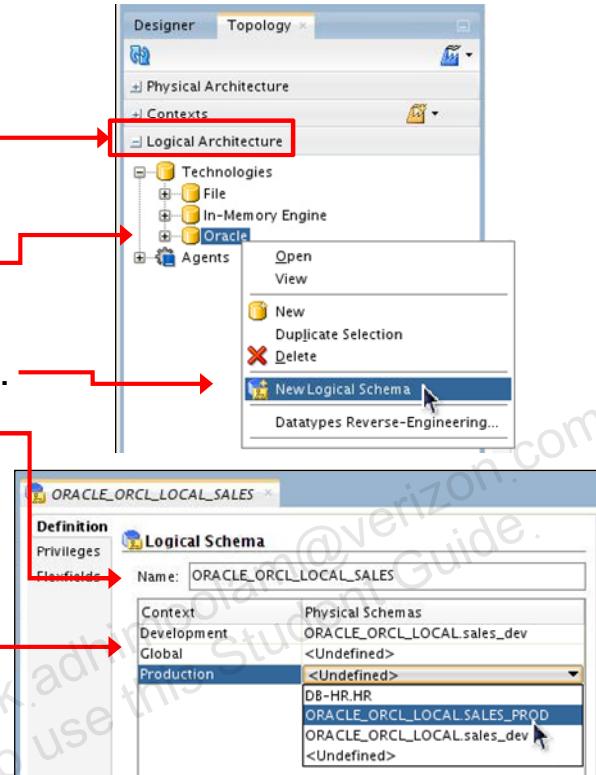
You can see the contexts that link the logical and physical architectures in the Contexts view.

In both the Logical Architecture and Physical Architecture views, the same list of technologies is presented. Thus, to modify the definition of a technology, you can use either view.

The number of technologies available in ODI can be somewhat overwhelming. Fortunately, you can make Topology Navigator display only those technologies on which at least one logical schema is defined. To do this, select Hide Unused Technologies from the Topology Navigator drop-down menu. Do not forget to redisplay the technologies when you want to define a logical schema on a new technology.

Creating a Logical Schema

1. Go to the Logical Architecture view.
2. Right-click the schema technology.
3. Select New Logical Schema.
4. Enter the schema name.
5. You can associate the schema with the physical schemas, in different contexts, here.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

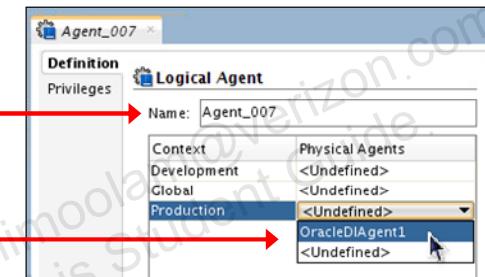
To create a logical schema, perform the following steps:

1. Go to the Logical Architecture view in Topology Navigator.
2. Find the technology appropriate for your logical schema and right-click its node.
3. From the context menu, click New Logical Schema.
4. In the window that opens, provide a name for the logical schema. The best practice is to include the name of the technology in the logical schema name.
5. To map this logical schema to physical schemas in different contexts, select the appropriate physical schema for each context.

Note: You can associate the logical schema with any physical schemas defined for this technology directly on the Definition tab. However, it does not have to be associated with a physical schema in every context.

Creating a Logical Agent

1. Go to the Logical Architecture view.
2. Right-click the Agents node.
3. Select New Logical Agent.
4. Enter the agent name.
5. You can associate the logical agent with the physical agents here.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

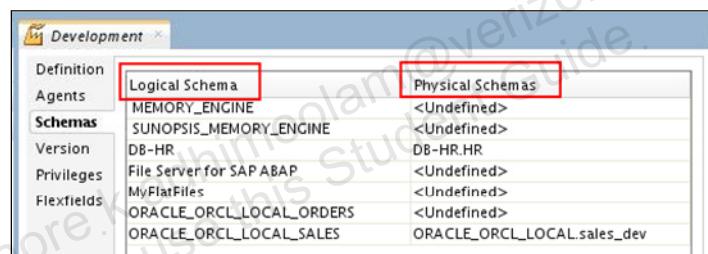
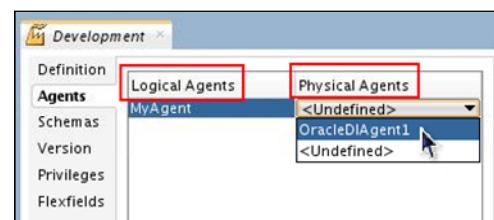
The procedure to create a logical agent is similar to creating a logical schema. Perform the following:

1. Go to the Logical Architecture view.
2. Right-click the Agents node.
3. Select New Logical Agent from the context menu.
4. Provide a name for the logical agent.
5. As with logical schemas, you can define the associated physical agents here.

Note: Physical agents should be defined before logical agents. You learned how to create physical agents in the lesson titled “Administering ODI Repositories.”

Editing a Context to Link Logical and Physical Agents

1. Double-click the context.
2. Click the Agents tab.
3. For each logical agent, select the corresponding physical agent in the context.
4. Click the Schemas tab.
5. For each logical schema, select a corresponding physical schema in the context.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

As you saw earlier, you can associate logical schemas and agents with physical objects in their respective windows. However, it is easier to set up an entire context by editing the context itself.

1. To do this, double-click the context in the Contexts view.
2. Click the Agents tab to assign agents. Now, the list of logical agents is displayed.
3. For each logical agent, you can assign a physical agent from the drop-down list. Or, you can set the physical agent to <Undefined> to make it unavailable in this context.
4. Similarly, click the Schemas tab.
5. Associate the logical schemas with the physical schemas.

Note: You do not have to associate every logical schema with a physical schema, or every logical agent with a physical agent, in every context. However, such an unresolved agent or schema cannot be used in the given context.

Quiz

Which of the following privileges must the user have to integrate data into the database tables and reverse-engineer them?

- a. SELECT, INSERT, **and** UPDATE
- b. READ, INSERT, **and** UPDATE
- c. SELECT, INSERT, UPDATE, CREATE, **and** DROP
- d. SELECT, INSERT, UPDATE, CREATE, DROP, **and** READ



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: d

Explanation: In the case of a database, the user should have SELECT privileges to all the schemas defined in the data server. The user must also have INSERT/UPDATE privileges to the tables into which ODI will integrate data. The user must have sufficient privileges to CREATE/DROP objects into temporary, or work, schemas, and should be able to READ the structure of the tables to reverse-engineer them.

Note: This is the reason why it is a best practice to log in with the owner of the work schema.

Quiz

On the Definition tab, you must associate the logical schema with the physical schemas defined for this technology in every context.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Explanation: You do not have to associate every logical schema with a physical schema, in every context. However, such unresolved schema cannot be used in the given context.

Summary

In this lesson, you should have learned how to:

- Use Topology Navigator to create physical and logical architectures
- Create data servers and physical schemas
- Link the physical and logical architectures via contexts



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Checklist of Practice Activities

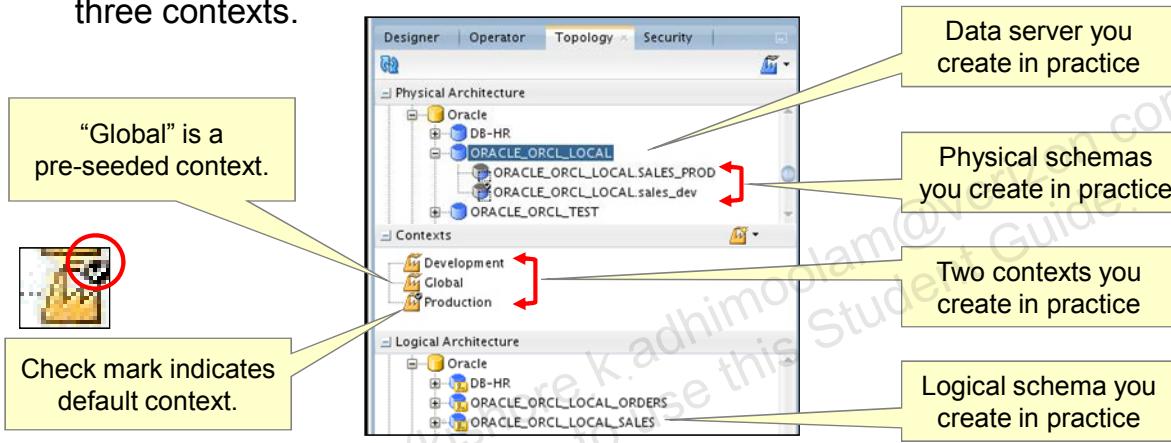
- ✓ 1-1: Logging In and Help Overview
- ✓ 2-1: Creating and Connecting to ODI Master and Work Repositories
- ✓ 3-1: Configuring Standalone Agents Using the Common Admin Model
- 4-1: **Working with Topology**
 - 5-1: Setting Up a New ODI Project
 - 6-1: Creating Models by Reverse-Engineering
 - 7-1: Checking Data Quality in the Model
 - 8-1: Creating ODI Mapping: Simple Transformations
 - 9-1: Creating ODI Mapping: Complex Transformations
 - 9-2: Creating ODI Mapping: Implementing Lookup
 - 10-1: Creating ODI Mapping: Exporting a Flat File to a Relational Table
 - 11-1: Using Native Sequences with ODI Mapping
 - 11-2: Using Temporary Indexes
 - 11-3: Using Sets with ODI Mapping
 - 12-1: Creating and Using Reusable Mappings
 - 12-2: Developing a New Knowledge Module
 - 13-1: Creating an ODI Procedure
 - 14-1: Creating an ODI Package
 - 14-2: Using ODI Packages with Variables and User Functions
 - 15-1: Debugging Mappings
 - 16-1: Creating and Scheduling Scenarios
 - 17-1: Using Load Plans
 - 18-1: Enforcing Data Quality with ODI Mappings
 - 19-1: Implementing Changed Data Capture
 - 20-1: Setting Up ODI Security
 - 20-2: Integration with Enterprise Manager and Using ODI Console

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 4-1: Working with Topology

1. Define Production context.
2. Define Development context (a third context, Global, is pre-seeded).
3. Define ORACLE_ORCL_LOCAL data server.
4. Define ODI physical schemas for data servers:
SALES_DEV, SALES_PROD.
5. Define ORACLE_ORCL_LOCAL_SALES ODI logical schema.
6. Map the logical schema to the two physical schemas, in terms of the three contexts.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In preceding practices, you created Master and Work Repositories and installed an ODI agent as a background service.

Before you begin working on your ODI projects, you need to describe your ODI infrastructure in the topology. You need to define contexts, a data server, physical and logical schemas, and mappings of your logical schema to the physical schemas in terms of contexts.

Setting Up a New ODI Project

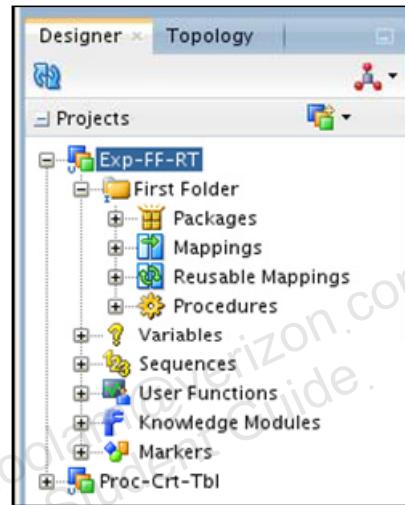
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Set up a new project
- Use folders to organize your work
- Import the right Knowledge Modules
- Import and export objects
- Use markers to manage the project



ORACLE

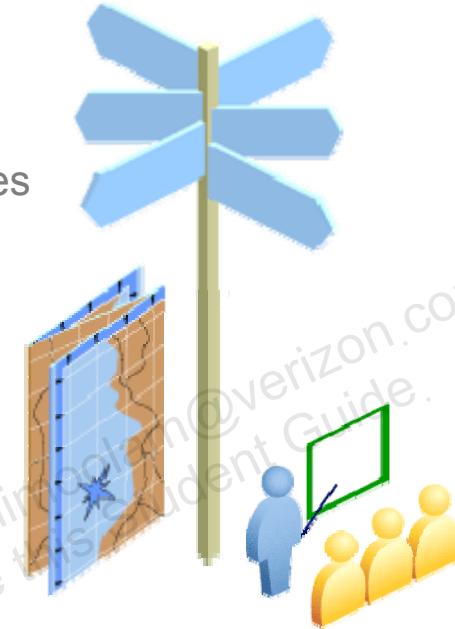
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This lesson explains the basic organization of projects by using markers and folders. In this lesson, you should learn how to:

- Set up a data integration project in Oracle Data Integrator (ODI)
- Use folders to organize your work into subprojects
- Import Knowledge Modules and know which ones to import
- Import and export objects to share your work
- Use markers to help organize your project by adding project-specific information to your objects

Agenda

- **ODI Projects**
 - Overview
 - Setup
- Using Folders
- Understanding Knowledge Modules
- Exporting and Importing Objects
- Using Markers



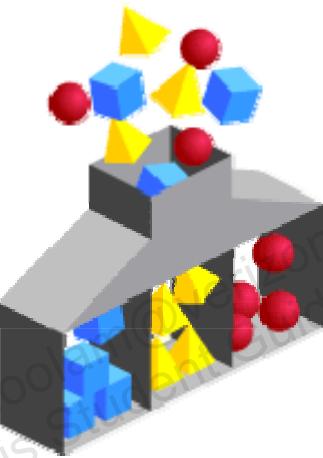
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This section provides an overview of what projects do and a description of how to set one up.

What Is a Project?

A project is a collection of ODI objects created by users for a particular functional domain.



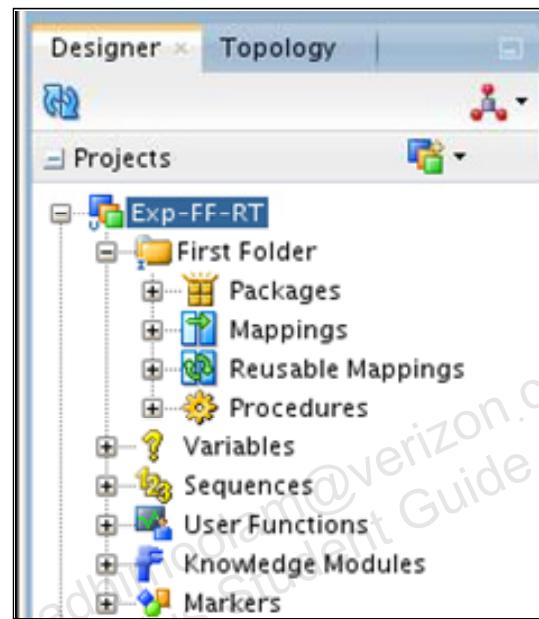
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Oracle Data Integrator Projects: Overview

What can a project contain?

- Folders:
 - Packages
 - Mappings (two kinds)
 - Procedures
- Variables
- Sequences
- User Functions
- Knowledge Modules
- Markers



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You know that projects are collections of ODI objects. However, only certain objects can belong to projects. Similarly, certain objects always belong to projects.

- Packages, procedures, and mappings always belong to folders, and folders always belong to projects.
- Variables, sequences, and user functions either belong to projects or can be created with global scope.
- Knowledge Modules (KMs) either belong to projects or can be imported with global scope. Also some KMs are considered built-in (neither project nor global).
- Markers always belong to a project.

In this lesson, you learn how to use folders and markers to organize projects. You also learn to import the correct Knowledge Modules, which is an important step in project administration.

How to Use ODI Projects in Your Work

- An ODI project should represent a functional domain or an integration project.
- Projects have their own objects (Knowledge Modules, variables, markers, and so on).
 - Objects in the projects can be shared by duplication.
 - You can also use global variables, sequences, Knowledge Modules, and so on.
- Examples of projects:
 - Chaining ACCOUNTING and INVOICING databases
 - Loading the data warehouse
 - Generating and sharing sales results



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can use projects in your work in different ways. But the best way is to use a project to represent a functional domain in your integration process or an integration project that you need to develop with ODI.

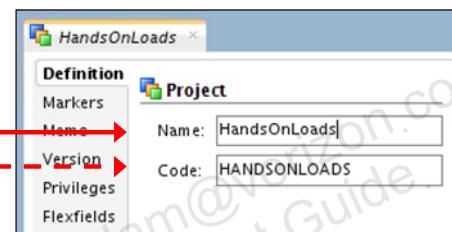
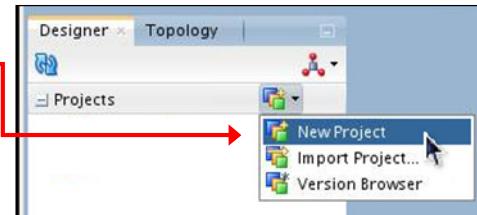
Each project contains its own Knowledge Modules, variables, markers, and other types of objects. These objects can be shared between different projects by duplicating them.

You can also use global objects such as variables, sequences, or Knowledge Modules to define parameters that are common to all projects.

Examples of suitable projects would be chaining the accounting and invoicing databases, populating the data warehouse, or generating and sharing sales results.

Creating a New Project

1. Select **New Project** from the drop-down menu.
2. Select a name.
3. Optionally, change the code.
 - Use the code to prefix project variables.
 - Keep it short.
4. Default markers are added automatically.
5. One folder is created to hold mappings, packages, and procedures.



To create a new project, perform the following:

1. In the Designer Navigator, select New Project from the drop-down menu at the top of Projects view.
2. Name the project. You should generally make it reflect the functional domain that it covers.
3. A code is generated in UPPERCASE but can be changed to anything you like. You may want to shorten the code to something more manageable. The code is used as a prefix when referring to variables created within this project.
4. Default marker sets are automatically added to your blank project.
5. Similarly, a folder is created automatically to hold the mappings, packages, and procedures that you develop.

Agenda

- ODI Projects
- **Using Folders**
- Understanding Knowledge Modules
- Exporting and Importing Objects
- Using Markers



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The following slides describe the folders that group packages, mappings, and procedures within projects.

What Is a Folder?

- A folder is a hierarchical grouping beneath a project and can contain other folders and objects.
- Every package, mapping, or procedure must belong to a folder.



ORACLE

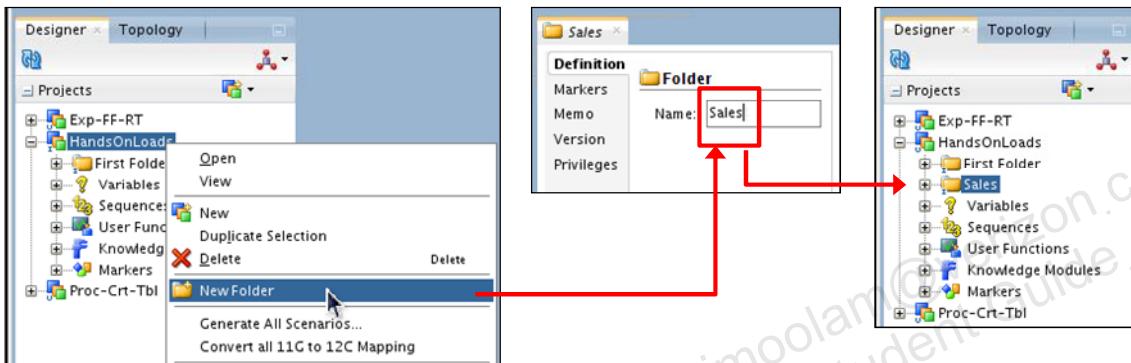
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When should you create folders? One guideline is to create a folder per “package” or scenario.

- This way, all mappings that are used in the same package are grouped together.
- The folder represents all that is necessary for a given execution unit.
- As a result, maintenance is typically simplified.

Creating a New Folder

1. Right-click a project or folder.
2. Select New Folder.
3. Name the folder.



You can drag folders to other folders.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can create folders within projects or within other folders as subfolders. To create a new folder, perform the following:

1. Right-click the project or folder where you want to create the new folder.
2. Select Insert Folder from the context menu.
3. Give a name to the folder.

After the folder is created, you can drag it to other folders, or onto the parent project, to reorganize the structure of your project.

Organizing Projects and Folders

- When do you need a new project?
 - When you start working on a new integration project or functional domain
 - When you want to keep objects separated
- When do you need a new folder?
 - For organizational purposes within an existing project
 - To define different user privileges within the project
- Project/folder boundaries:
 - Objects cannot be shared between projects.
 - Global variables, sequences, Knowledge Modules, and user functions can be used by any project.
 - Objects within a project can be used in all folders.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

It is not always easy to know when to create a project or when to create a subfolder within the same project. As a general rule, if you have a new functional domain or are starting an integration project, you should create a new project. Also, if you specifically want to keep objects separate from each other, you should create them in separate projects.

On the other hand, a folder is useful when you want to organize an existing project. When you have a large number of mappings, procedures, or packages in a project, you should consider grouping them into folders. You can also use folders to set up different security levels within the same project. Each folder can have its own unique privileges.

However, you should be aware that projects do impose strict boundaries on information sharing. This means that objects created in one project cannot be used by another project. If you want to enable the objects to be used by other projects, you should make the object global. Thus, you enable the global variables, sequences, Knowledge Modules, and user functions to be used by any project.

Folders, however, do not impose boundaries. Thus, objects in one folder can be used by any other object in the project.

Agenda

- ODI Projects
- Using Folders
- **Understanding Knowledge Modules**
- Exporting and Importing Objects
- Using Markers



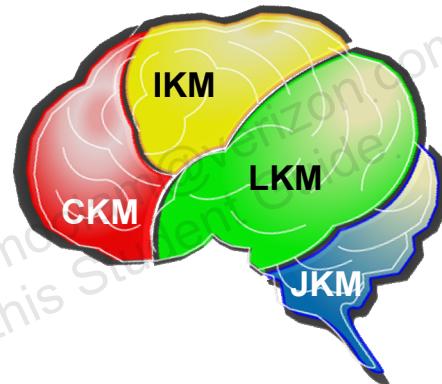
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Probably the most important thing when you create a project is importing the correct Knowledge Modules. These modules enable your integration mappings to work on the technologies used by your data servers. Without importing any Knowledge Modules, no mappings can be made or executed.

What Is a Knowledge Module?

- A Knowledge Module (KM) is a code template containing the sequence of commands necessary to implement a data integration task.
- There are different predefined KMs for loading, integration, checking, reverse-engineering, journalizing, and deploying data services.
- All KMs work by generating code to be executed at run time.
- KMs can be specified as “Global,” allowing them to be shared across multiple projects.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Knowledge Modules (KMs) are components of Oracle Data Integrator Open Connector technology. KMs contain the knowledge required by ODI to perform a specific set of tasks against a specific technology or set of technologies. Technically, this term describes a template containing the code necessary to implement a particular data integration task. These tasks include loading data, checking it for errors, or setting up triggers necessary to implement journalization. However, all Knowledge Modules basically work the same way: ODI uses them to generate code, which is then executed by a technology at run time.

ODI 11.1.1.6 introduced Global Knowledge Modules allowing specific KMs to be shared across multiple projects. In previous versions of ODI, Knowledge Modules were always specific to a project and could be used only within the project into which they were imported. Global KMs are listed in the Designer Navigator in the Global Objects accordion.

ODI 12.1.2 introduces basic preloaded KMs.

Note: Knowledge Modules are independent of the structure of the source and target datastores. The same KM can be used, no matter which source table you have, or how many source tables you have. Likewise, all target tables can use the same Knowledge Module.

Types of Knowledge Modules

There are six types of Knowledge Modules you may import.

Mappings	KM Type	Description	
Models	LKM	Loading	Assembles data from source datastores to the Staging Area
	IKM	Integration	Uses a given strategy to populate the target datastore from the Staging Area
	CKM	Check	Checks data in a datastore for errors statically or during an integration process
	RKM	Reverse-engineering	Retrieves the structure of a data model from a database. It is needed only for customized reverse-engineering.
	JKM	Journalizing	Sets up a system for Changed Data Capture to reduce the amount of data that needs to be processed
	SKM	Data Services	Deploys data services that provide access to data in datastores



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The first group of Knowledge Modules is critical for doing any work with mappings.

- **Loading** Knowledge Modules (LKMs) extract data from the source of mappings. So, if your data is stored in flat files, you need the “File to SQL” LKM.
- **Integration** Knowledge Modules (IKMs) implement a particular strategy for loading the target of a mapping. Thus, to load an Oracle table while taking into account the slowly changing dimensions properties, a particular IKM can be used.
- **Check** Knowledge Modules (CKMs) are selected in Mappings and constraints can be individually enforced by ODI at the mapping level.

The second group is used for setting up, checking, and configuring models.

- **Check** Knowledge Modules (CKMs) enforce constraints defined on the target datastore. CKMs are used by models to perform static checks outside of mappings.
- **Reverse-Engineering** Knowledge Modules (RKMs) are needed only to perform customized reverse-engineering. They are used to recover the structure of a data model and are used when standard reverse-engineering cannot be performed.
- **Journalizing** Knowledge Modules (JKMs) are used to set up Changed Data Capture. This makes mappings react only to changes in data and can vastly reduce the amount of data that needs to be transferred.
- **Service** Knowledge Module (SKM) is the code template for generating data services.

Which Knowledge Modules Are Needed?

- All Knowledge Modules to be used in a project must be imported into the project (or into Global objects).
- Which Knowledge Modules do you need?
 - Start with basic SQL Knowledge Modules, some built-in.
 - Add technology-specific Knowledge Modules as needed.
 - For your projects, you need LKMs, IKMs, and CKMs only.
- You need to become familiar with the library of available Knowledge Modules, and what they can do for you:
 - *Knowledge Module Developer's Guide for ODI*
<http://docs.oracle.com/middleware/1212/odi/ODIKD>
 - *Connectivity and Knowledge Modules Guide for ODI*
<http://docs.oracle.com/middleware/1212/odi/ODIKM>



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You now know what the various types of Knowledge Modules are. But, how do you know which ones need to be imported into your specific project?

The most important thing to remember is to import all Knowledge Modules that may be used in a project. Each time you create a project, you must import the Knowledge Modules that will be used by that project (or ensure that they are available globally). Other than taking up space, it does not hurt to have “too many” (excess, unused) KMs.

The basic strategy for importing Knowledge Modules is as follows:

- First, import the most basic SQL Knowledge Modules. These work on almost every database management system (DBMS) with acceptable performance. Some of the most basic KMs are preloaded.
- Then, consider importing more specific Knowledge Modules for the particular technologies involved in your project. If you are transferring data to an Oracle server, consider adding Oracle-based IKMs.
- Technology-specific Knowledge Modules can take advantage of certain characteristics or special tools provided by the technologies. However, it is often best to begin with the most generic Knowledge Modules to start your mapping, and then choose specific Knowledge Modules later, which may increase performance.

Knowledge Modules: Examples

- LKM File to Oracle (SQLLDR)
 - Uses Jython to run SQL*LOADER through the OS
 - Much faster than basic LKM File to SQL
- LKM Oracle to Oracle (DBLINK)
 - Loads from Oracle data server to Oracle data server
 - Uses Oracle DBLink (built-in KM)
- IKM Oracle Merge
 - Integrates data into target table in incremental update mode
 - The data is loaded directly into the target table with a single MERGE SQL statement (built-in KM).
- CKM Oracle
 - Enforces logical constraints during data load
 - Automatically captures error records



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To make the descriptions of Knowledge Modules more concrete, here are some examples of the Knowledge Modules that are shipped with ODI.

The first module is an LKM that is used to load an Oracle server from a flat file server. According to the naming convention, the SQLLDR in parentheses must correspond to the method by which data is loaded.

In this case, the Oracle bulk-loading tool, SQL*LOADER, is the method. This tool is capable of loading vast amounts of data directly into Oracle. Using this Knowledge Module is much faster than the generic File to SQL LKM.

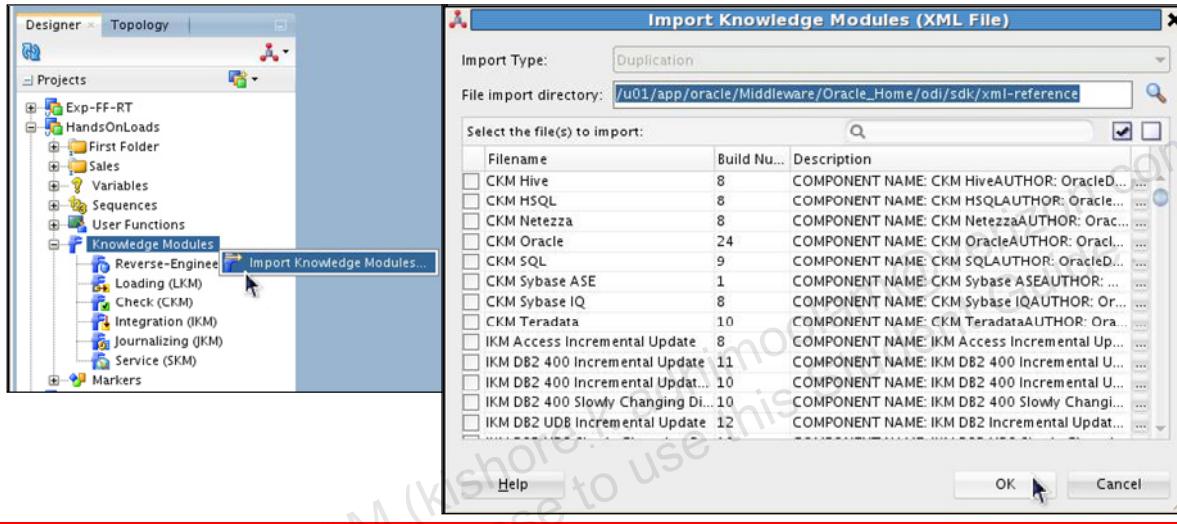
The next module is another LKM, this time loading from an Oracle data server to another Oracle data server by using Oracle DBLink. There are two flavors: a push and a pull. Pull is new with 12c.

A single merge is faster than an insert followed by an update. This KM is built-in, so you never have to import it.

CKM Oracle enforces logical constraints during data load and automatically captures error records.

Importing Knowledge Modules

1. Expand the project.
2. Select Knowledge Modules > Import Knowledge Modules.
3. Choose the import directory.
4. Select one or more Knowledge Modules.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The built-in KMs do not need to be imported. The built-in names have a suffix of .GLOBAL, for example:

IKM Oracle Insert.GLOBAL

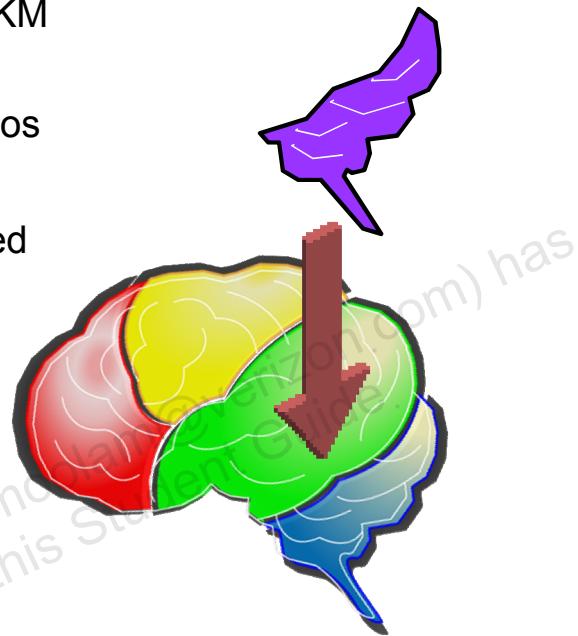
To import one or more other Knowledge Modules into your project, perform the following steps:

1. Expand the project in the Projects view.
2. Select Knowledge Modules > Import Knowledge Modules. (There are also other ways to get to the Import KMs option.)
3. Select the import directory. Each Knowledge Module is stored in a separate Extensible Markup Language (XML) file. The Knowledge Modules that come with ODI are stored in \$ODI_HOME/sdk/xml-reference.
4. The list of available Knowledge Modules is displayed. You can select multiple Knowledge Modules.

Note: The imported KMs can be edited. The built-in KMs cannot be edited or copied.

Replacing Existing KMs

- “Import replace” mode:
 - Updates all mappings that use the KM
 - Preserves existing options
 - Does not update generated scenarios
- Why replace an existing KM?
 - Because a newer version is released
 - To undo any undesirable changes made



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Normally, if you import a Knowledge Module with the same name as an existing Knowledge Module, you end up with two copies in your project. This can be useful if you want to make changes to the Knowledge Module without breaking your existing mappings.

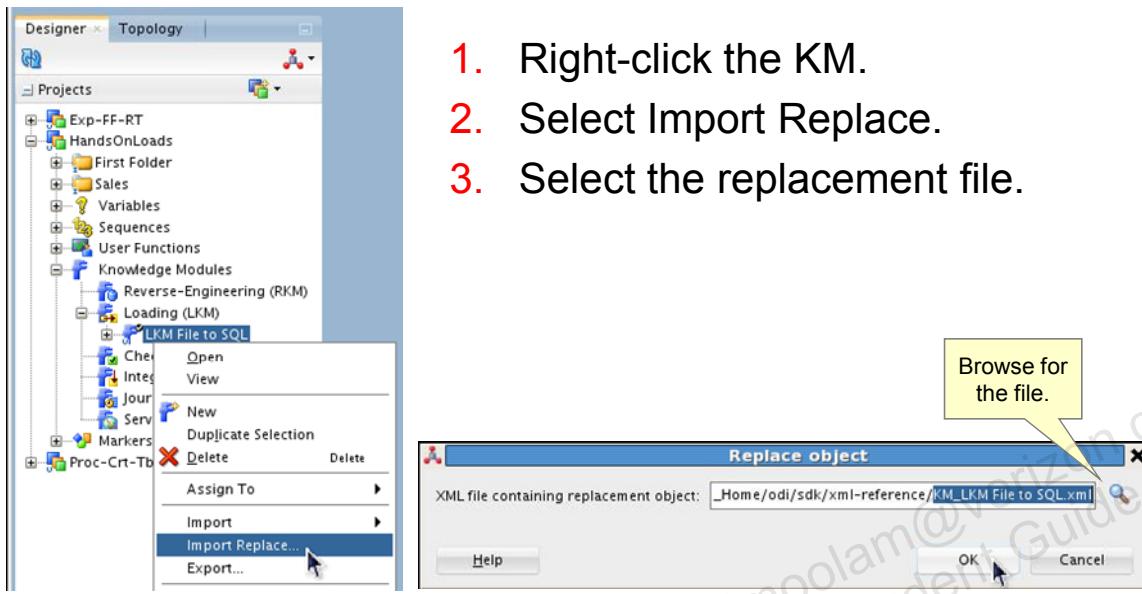
However, another mode of importing is available: the “import replace” mode. In this mode, the existing mapping is replaced with the version imported from the disk. All mappings that used the old Knowledge Module are automatically updated to use the new version. Also, values that are set for the Knowledge Module options in these mappings are transferred to the new version. However, any existing scenarios are not regenerated. If you want to incorporate changes to the Knowledge Module into your scenarios, you must regenerate them.

There are various reasons why you would want to replace a Knowledge Module. The most common reason is that a newer version of the Knowledge Module is released, perhaps by someone on your team. You import the Knowledge Module again and all your existing mappings will still work.

Similarly, you may have made some undesirable changes to your Knowledge Module. You can quickly undo these changes by re-importing from a saved version.

Note: Any mapping that uses the replaced KM also will be impacted.

Replacing Existing KMs



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

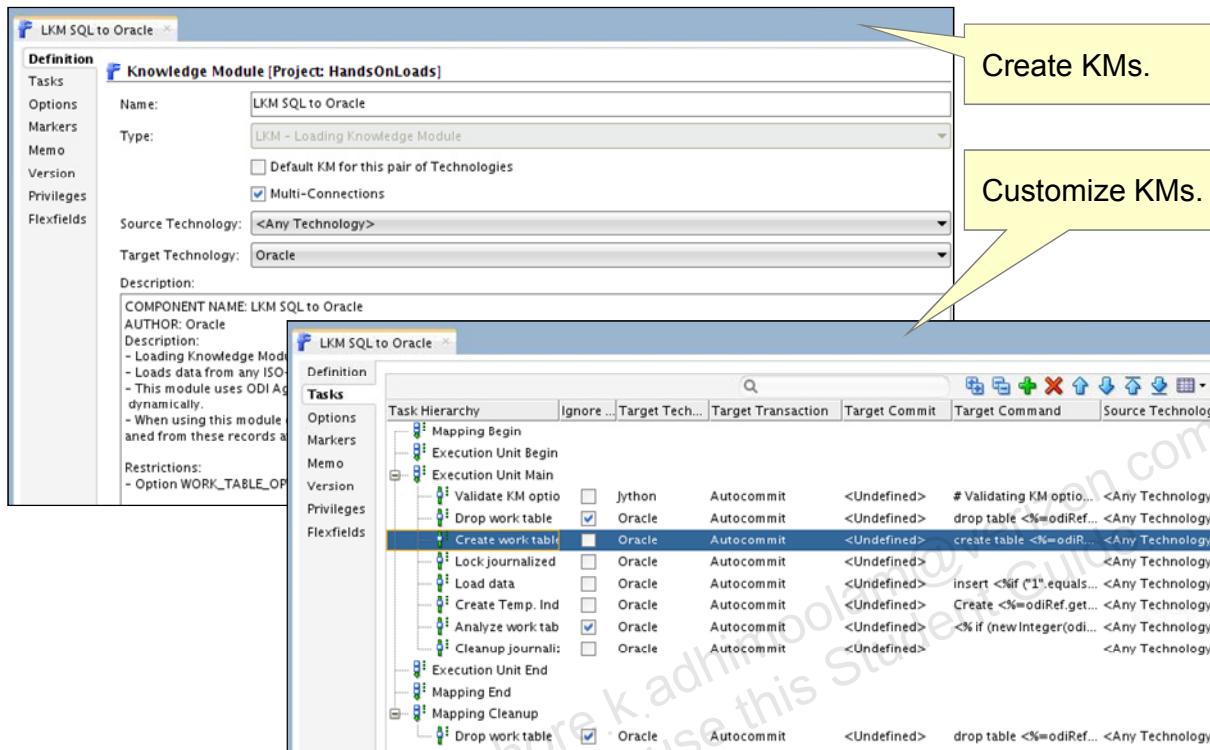
To import a Knowledge Module in replacement mode, perform the following:

1. Right-click the name of the Knowledge Module that is to be replaced.
2. Select Import Replace from the context menu.
3. Enter the name of the XML file that contains the new version of the Knowledge Module. You can use the browse button (magnifying glass) to select a file. Note that you can replace only one Knowledge Module at a time.

Consider regenerating any scenarios that use the Knowledge Module to take the changes into account. This includes both the mappings and packages. Regenerating scenarios is discussed later in the course.

Note: You should regenerate any scenarios that use the KM.

Knowledge Module Editor

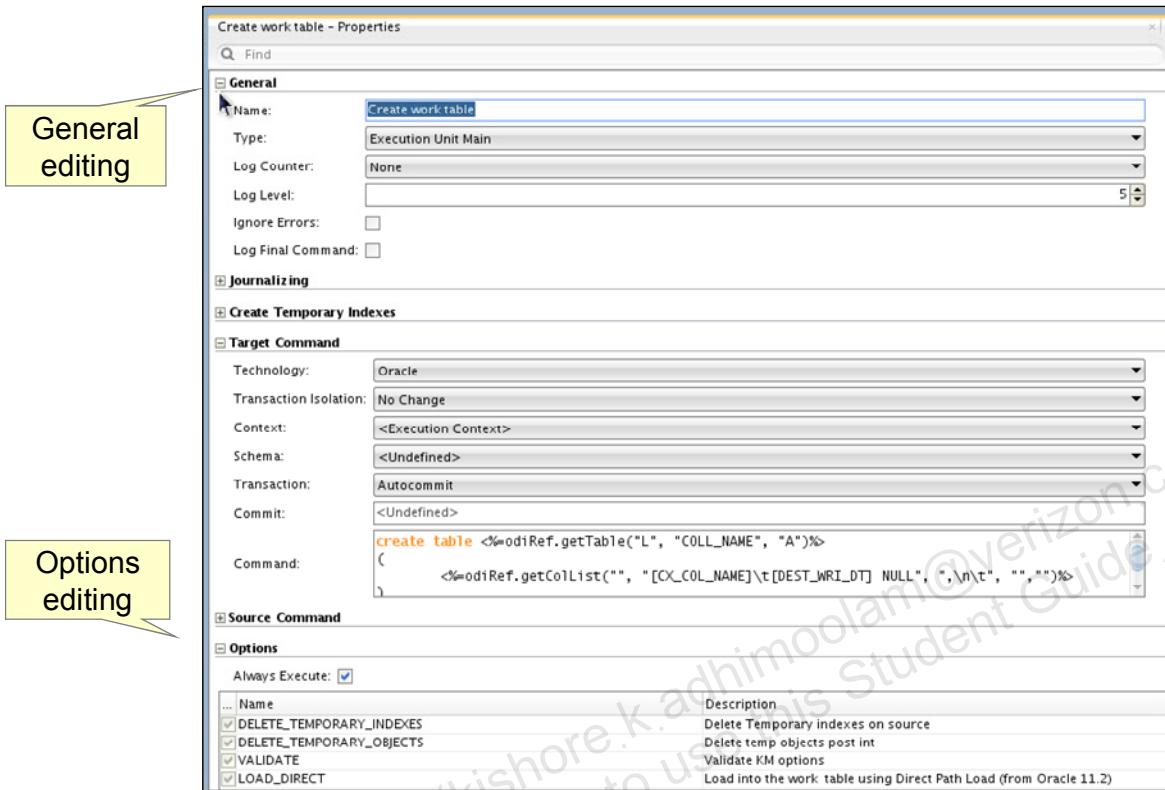


ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Use the Knowledge Module Editor to create and customize your Knowledge Modules. In this example, the third line is highlighted, “Create work table.” If you double-click this line, a detailed editor opens, as shown in the next slide.

Editing a Knowledge Module



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The KM editor has a section to edit general information about an item within the KM, such as this “Create work table” item in the “LKM SQL to Oracle” Knowledge Module.

The KM editor also has a section to edit the KM’s options.

Knowledge Modules are used later in this course. For additional detailed information about KMs, see the *ODI Knowledge Modules Reference Guide*.

Agenda

- ODI Projects
- Using Folders
- Understanding Knowledge Modules
- **Exporting and Importing Objects**
- Using Markers



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The last part of organizing a project is knowing how to import and export objects to share them with other projects.

Exporting and Importing

- Needs:
 - Moving objects between projects
 - Wanting to send a mapping to an off-site colleague
 - Sharing ODI objects between repositories
 - For example, migrating objects from a development repository to a maintenance repository
- Solution:
 - Export to an XML file.
 - Import the file into the other project.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The following slides discuss the solution to several problems that can arise.

You cannot drag and drop objects between projects. For example, you may want to copy a mapping from one project to another. Similarly, you may want to send an object you have developed to a colleague at a different site. Or, you may want to share an object between two repositories—for example, moving an object that was in development to a maintenance repository.

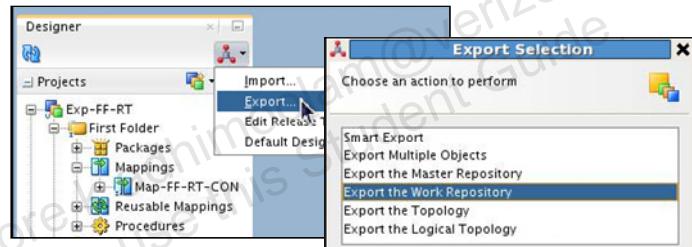
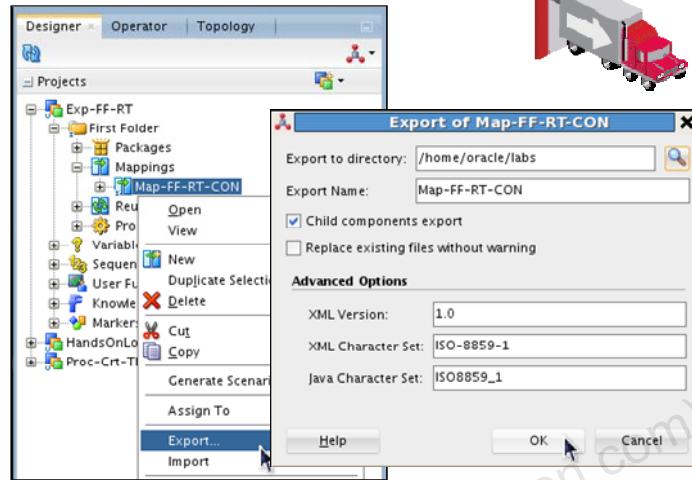
The solution to these three problems is the same: exporting, and then re-importing.

- First, you export the object to an XML file.
- Then, you re-import the object into the other project. This other project can be in a different repository, or even at a different site.

You will see how this is done in the next few slides.

Exporting an Object

1. Right-click the object.
2. Select Export.
3. Specify:
 - The export directory
 - The export file name (without .xml)
 - Whether to export child objects
4. To export a Work Repository, use the menu system.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To export an object, perform the following:

1. Right-click the object. In this example, you export a mapping. However, you can export any object.
2. Select Export from the context menu. The Export window appears.
3. Specify the directory where you want to export the object. You can use the browse button to select the directory by navigating to it.
4. You must also specify the file name to export to. The .xml extension will automatically be added to it.

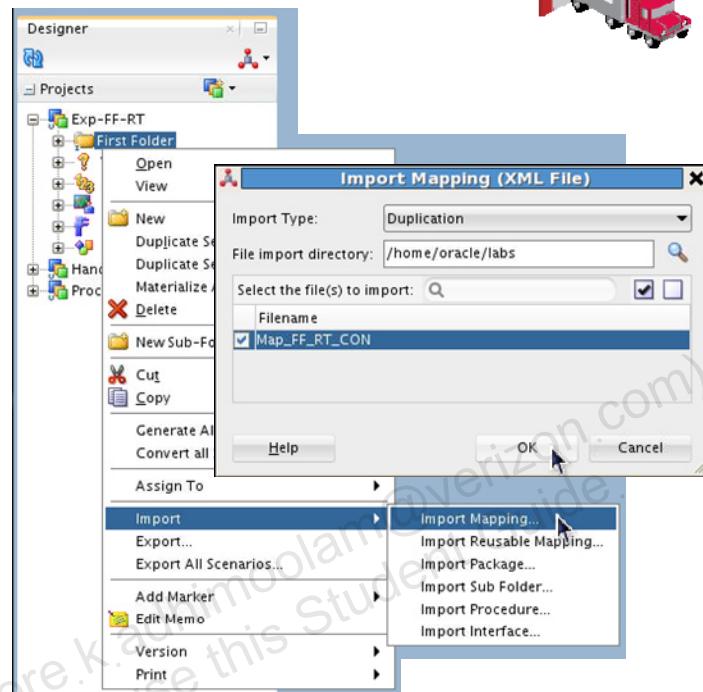
Some objects, such as folders, projects, and packages, can contain child objects. If you do not want these objects in the .xml file, deselect the "Child components export" check box.

Though almost every object in ODI can be exported by using this method, there are exceptions. To export a Work Repository, use the relevant option in the File menu in Designer Navigator.

Importing an Object



1. Right-click the project or folder.
2. Select Import > Import <object>.
3. Specify:
 - Import type
 - Import directory
 - File(s) to import
4. To import a repository or topology, use the File menu.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

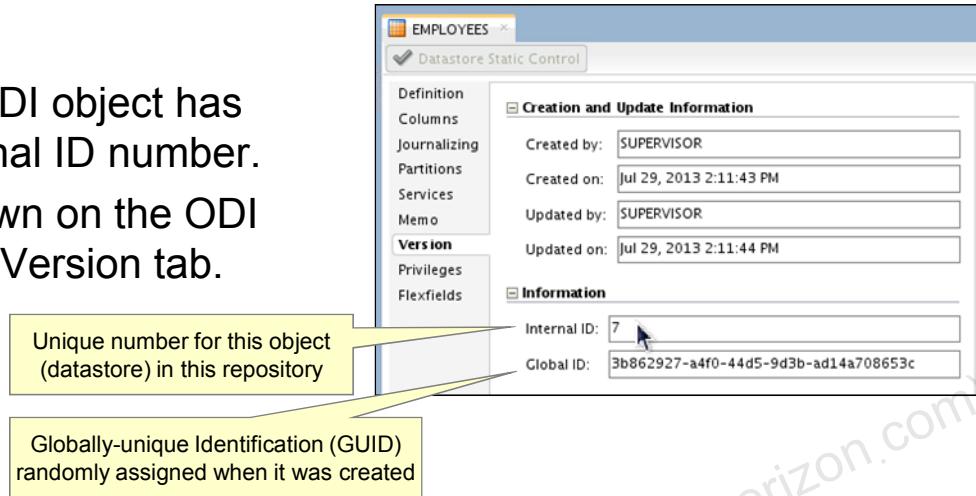
To import an object, perform the following:

1. Right-click the project or folder you want to import into. In this example, you import a mapping.
2. Select Import from the context menu, and then the type of object to import. The type of object depends on where you are importing. For folders, you can import packages, other folders, procedures, or mappings.
3. Specify the import type.
4. Specify the directory from where you want to import the object. You can use the browse button to select the directory by navigating to it.
5. ODI displays all importable objects in this directory as a list. Select one or more of these objects to import.
6. Click OK to begin the import.

To import a repository or the topology, use the relevant options in the File menu in Topology Navigator or Designer.

ID Numbers: Overview

- Every ODI object has an internal ID number.
- It is shown on the ODI object's Version tab.



- The Internal ID number is unique for each type of object.
- The GUID number is exported with the object.

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Before looking at what the import type does, you must understand the concept of ID numbers. Every ODI object has an ID number that can be seen on the Version tab of every object.

The ID number is unique for this type of object in this particular repository. In the example, 7 means that this is the seventh datastore created in this repository. There might also be a seventh package, and a seventh mapping, and so on.

History: In previous versions of ODI, 11g and before, the second part of the ID number was the identifier for the repository where the object was created. So object 7 in repository 001 was number 7001. Thus, the identifier was unique only for a given type of object within a given repository. In the example, another package or user function in the same repository can have ID number 4001. But only one datastore in this repository can have ID number 4001. When you exported an object, its ID number is recorded in the .xml file. That caused potential conflicts. This has all been solved with GUIDs in 12c.

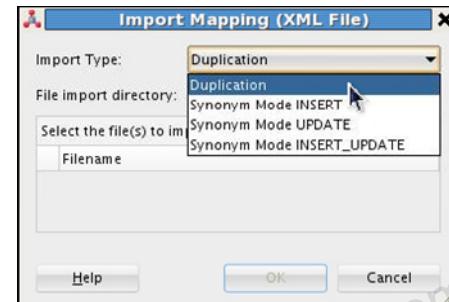
Import Types

There are two modes for importing ODI objects:

- Duplication:
 - A new ID number for the object is generated based on the new repository.
- Synonym:
 - Use the ID stored in the export file.

There are three submodes:

 - INSERT: Performs inserts into the repository
 - UPDATE: Performs updates in the repository
 - INSERT_UPDATE: Inserts new records and updates existing records in the repository
 - No ODI objects are deleted.
 - Internal ID references are maintained.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You know now that when you export an object, the object ID is stored in the file and that each ID must be unique for each type of object. So, what happens if you try to import an object back into the repository?

The answer is that there are two ways of using the stored ID, as specified by the Import Type option.

- In Duplication mode, the ID in the file is ignored. Instead, a new ID is generated based on the ID of the new repository. The object is then completely duplicated.
- In Synonym mode, the ID in the file is retained. This can raise an issue if the original object is present in this repository. To deal with this problem, three options are available:

INSERT mode performs blind inserts into the repository. If an object has the same ID, the import fails.

UPDATE mode performs only updates. If no corresponding object exists, the import does not fail, but it does not create the missing object.

INSERT_UPDATE mode updates the corresponding objects and inserts them if they do not exist. This is the most useful mode in many situations.

In every mode, one rule is constant: No ODI objects are ever deleted.

Choosing the Import Mode

- Select Synonym mode:
 - To exchange objects between two repositories having different identifiers
 - To update (add components to) an object with an export file
 - To ensure that you import objects in the right order:
 - Import objects on which other objects depend first.
 - For example, before importing a mapping, you should import the models that it uses.
 - There is an optional module for dependency management.
- Select Duplication mode:
 - To create a duplicate of the original object from the export file
 - To reuse an object in a different project, in the same repository



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A couple of recommendations can help you select the mode for use.

Synonym mode effectively creates a link between two objects in two different repositories. It is useful when you want to exchange objects between two repositories. The repositories must have different identifiers.

If you use Synonym mode, you can update the same object later. Because the ID of the object does not change in a synonym import, you can replace it without breaking any links.

Note: You must be careful to import objects in the right order. It means that you must import objects before other objects that depend on them. For example, you must import models before mappings because mappings depend on models. There is also an optional module in ODI that can help with dependency management.

Duplication mode, on the other hand, is simpler. You use it when you want to create a duplicate of the original object—for example, when you want to have two packages that do the same thing, perhaps, with a slight change.

It is also useful for crossing project boundaries—for example, to reuse a marker group defined in one project in a different project, in the same repository. You cannot use the Synonym mode here because there would be a conflict of ID.

Import Report

Object Type	Object Name	Imported As	Original ID	New ID After Import	Global ID	Parent Type After Import	Parent Name After Import
Mapping	Map-FF-RT-CON	Copy of Map-FF-RT-CON	1	3	5ee6b0e9-52bd-4574-9bf2-5fb8d15b46d5	Folder	First Folder

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The import report is displayed after every import operation. Read it carefully to uncover errors from the import process.

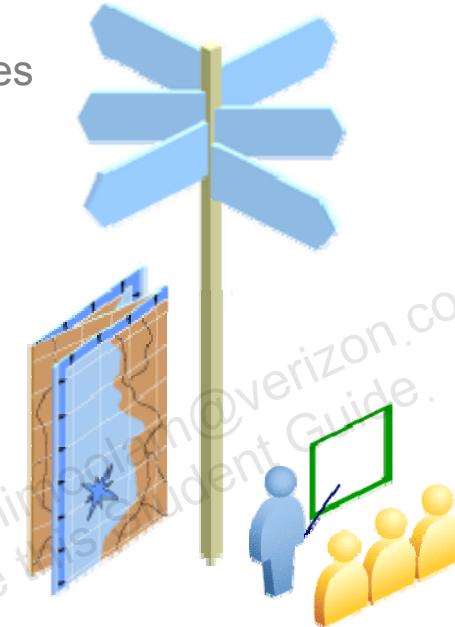
The import report gives you details on the following:

- **Import Mode:** This section appears at the top of the report without its own subheading, under the “Import Report” main heading. In this screen, the chosen import mode is indicated as “Duplication.”
- **Imported Objects:** For every imported object, this section indicates the object type, the original object name, the object name used for the import, the original ID, and the new, recalculated ID.
- **Deleted Objects:** For every deleted object, this section indicates the object type, the object name, and the original ID.
- **Created Missing References:** This section lists the missing references detected after the import.
- **Fixed Missing References:** This section lists the missing references fixed during the import. You can save the import report as an .xml or .html file.

Click Save to save the report, otherwise click Close.

Agenda

- ODI Projects
- Using Folders
- Understanding Knowledge Modules
- Exporting and Importing Objects
- **Using Markers**



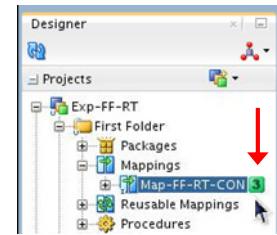
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Markers, which are simple organizational tags that can help in the planning and management of an integration project, are described in the following slides.

What Is a Marker?

- A marker is a tag that you can attach to any ODI object to help organize your project.
- Markers can be used to indicate progress, review status, or the life cycle of an object.
- Graphical markers attach an icon to the object, whereas nongraphical markers attach numbers, strings, or dates.
- Markers can be crucial for large teams, enabling communication among developers from within the tool.
 - Review priorities.
 - Review completion progress.
 - Add memos to provide details on what has been done or has to be done.
 - This can be helpful for geographically dispersed teams.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

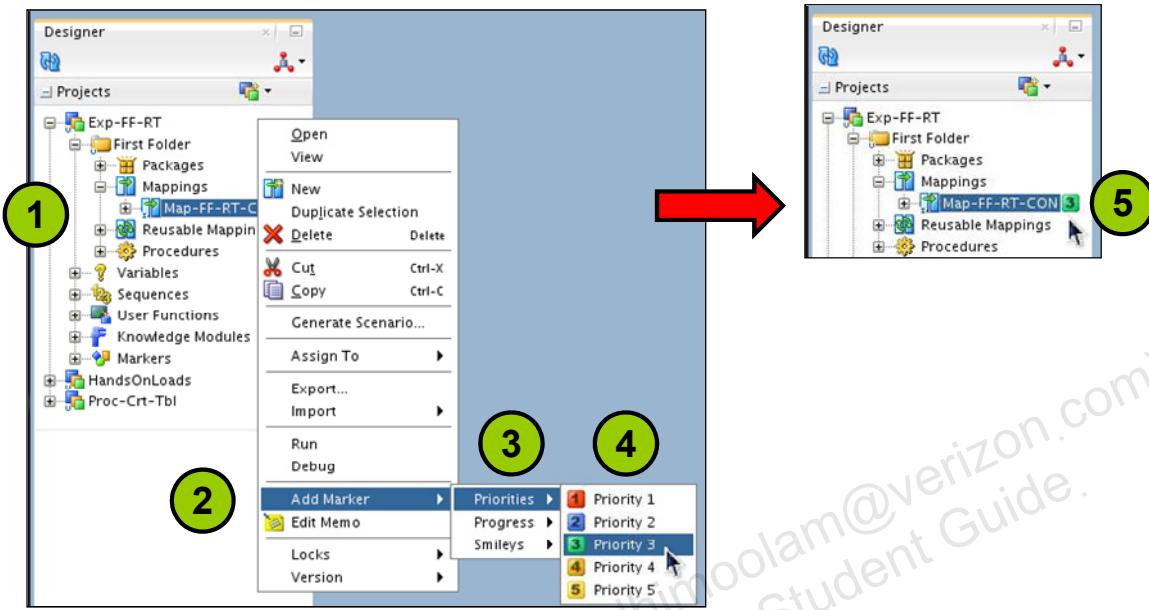
Markers are tags that you can use to attach arbitrary information to ODI objects. They can be attached to any objects: mappings, user functions, Knowledge Modules, columns of models, or even whole projects. Markers help you organize your project in whatever way makes sense to you. Typically, you use them to indicate progress in the development of an object or the current state in the life cycle of an object.

There are two types of markers:

- Graphical markers cause an icon to be displayed next to the object in Designer.
- Nongraphical markers attach other information, such as numbers or dates to the object.

Not only can you assign objects with markers, you can also *lock* those objects that are assigned to certain users. That way there is less confusion in a big project.

Tagging Objects with Markers



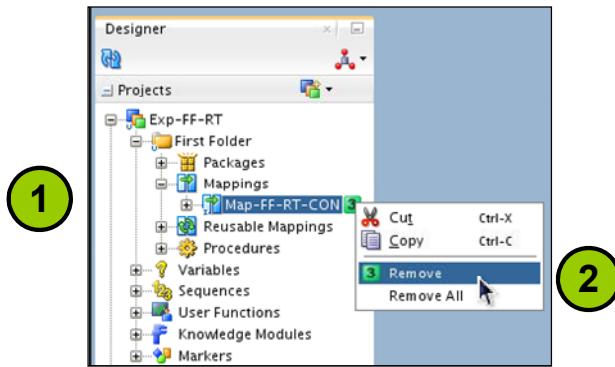
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Tagging an object with a marker is very simple. Perform the following:

1. Right-click the object that you want to attach it to. Almost any object can be marked in this way. In this example, you add a marker to a mapping.
2. Select Add Marker from the context menu.
3. Select the appropriate marker group. Marker groups are covered in greater detail later in this lesson.
4. Select the marker. You can have multiple markers per object.
5. The marker appears next to the package you selected. You can choose whether it appears to the left or the right of the object name.

Removing Markers



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To remove markers from an object:

1. Right-click the marker.
2. Select Remove or Remove All.

From the context menu, you can select either Remove to remove the one marker or Remove All to remove all the markers from this object. However, nongraphical markers are not removed.

Marker Groups

- A marker group is a collection of related markers.
 - Normal: One marker is selected at a time, to mark an object.
 - Multistate: Any combination of symbols can be selected to mark one object.
- One marker group can contain both graphical and nongraphical markers.

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A marker group is a collection of related markers. When you add a marker to an object, you first choose the group to select from, before choosing the individual marker.

Normally, only one marker can be selected for a given object at a given time. However, there are multistate markers. This means that several markers from the same group can simultaneously be attached to an object. For example, you can have a group called “Reviewed by” with markers Ann, Bill, and Chris. Then, you attach one marker for each person who has reviewed the selected object.

Note that a single marker group can contain both graphical and nongraphical markers.

Project and Global Markers

- Project markers:
 - Are created in the Markers folder under a project
 - Can be attached only to objects in the same project
- Global markers:
 - Are created in the Global Markers folder in the Others view
 - Can be attached to models and global objects



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

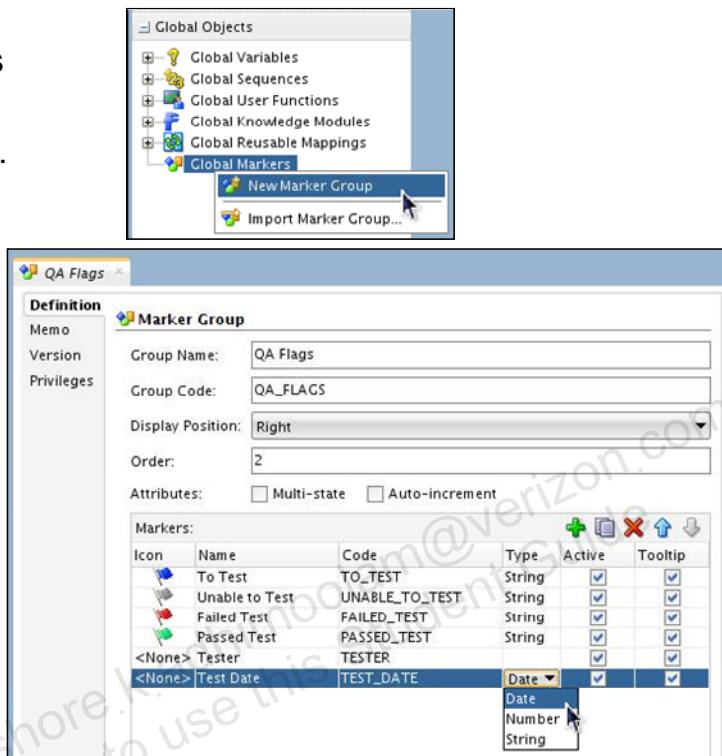
Marker groups are created in one of two places. Depending on where they are created, they can be attached to different kinds of objects.

Project markers are created in a project, under the Markers folder. They can be attached only to objects, such as mappings, packages, or user functions, in the same project.

Global markers, on the other hand, are created in the Global Markers folder in the Others view. They are primarily intended to be attached to models and components of models, such as datastores or columns. However, you can also attach them to other global objects, such as global variables or global user functions. They cannot be attached to project objects.

Creating a Marker Group

1. Right-click Global Markers or Markers.
2. Select New Marker Group.
3. Enter Group Name.
4. Enter Group options:
 - Display Position
 - Display Order
 - Auto-increment
 - Multi-state
5. Add markers.
6. For each marker, set:
 - Icon
 - Name
 - Type



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The basic process for creating a new marker group is as follows:

1. To create a global marker group, right-click Global Markers in the Others view. For a project marker group, right-click the Markers node in the Project view.
2. Select New Marker Group from the context menu. The Marker Group window appears.
3. Select a name for the marker group. This name is displayed wherever you can select a marker to attach to an object.
4. Set the options for the group as a whole.
 - a. Display Position specifies whether the icon for the marker appears to the left of the object, to the right, or not at all.
 - b. If several markers are attached to an object, markers with lower order numbers are displayed to the left of markers with higher numbers.
 - c. Auto-increment means that you can click the marker to automatically advance it to the next marker in the group. This is useful for progress-type markers, so you can quickly advance from 10% to 20%, and so on.
 - d. Multi-state enables several markers from this group to be attached to an object at once, as described earlier.
5. Add some markers to your group by clicking the Add Marker button.
6. Set the options for each marker. This includes the icon, name, and type of each marker.

Quiz

Integration Knowledge Modules (IKMs):

- a. Extract data from the source of mappings
- b. Implement a particular strategy for loading the target of a mapping
- c. Enforce constraints defined on the target datastorer
- d. Perform customized reverse-engineering



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Explanation: IKMs are used to integrate (load) data to the target tables.

Quiz

Which of the following is not true?

- a. INSERT mode performs blind inserts into the repository. If an object has the same ID, the import fails.
- b. UPDATE mode performs only updates. If no corresponding object exists, the import fails.
- c. INSERT_UPDATE mode updates the corresponding objects and inserts them if they do not exist.
- d. In Duplication mode, the ID in the file is ignored. A new ID is generated based on the ID of the new repository.
- e. In every mode, no ODI objects are ever deleted.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Explanation: UPDATE mode performs only updates. If no corresponding object exists, the import does not fail, but it does not create the missing object.

Summary

In this lesson, you should have learned how to:

- Set up a new project
- Use folders to organize your work
- Import the right Knowledge Modules
- Use markers to manage the project
- Import and export objects



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This lesson covered the different types of objects that projects can contain and the boundaries imposed by projects. You also learned about the following:

- Projects that can contain folders and using them to create subprojects
- The various types of Knowledge Modules and adding them to your project
- Creating and using markers to tag your ODI objects with information and graphical symbols
- Importing and exporting objects to share your work between projects

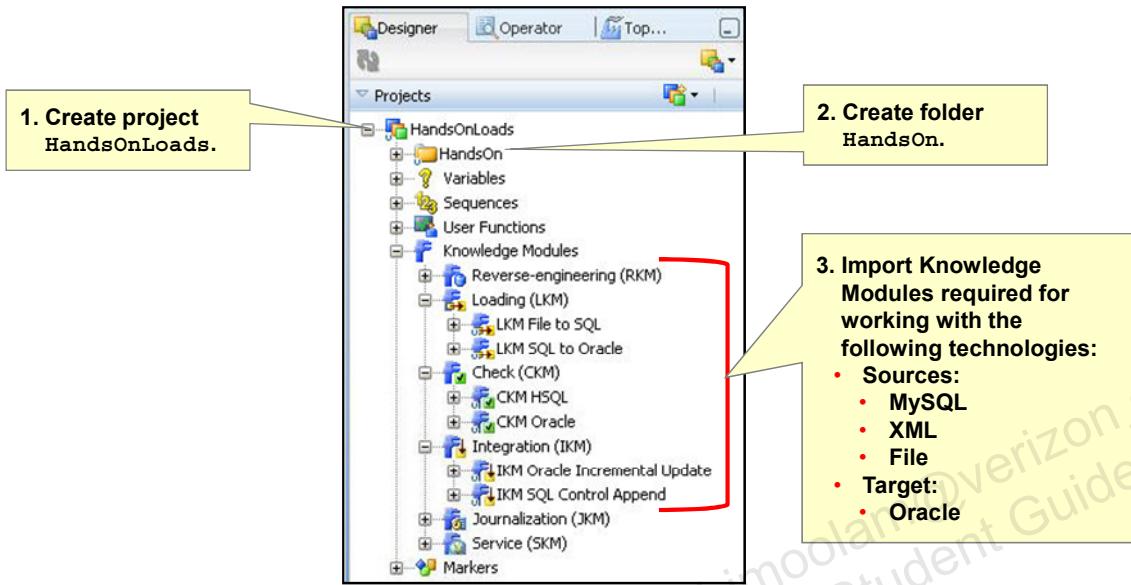
Checklist of Practice Activities

- ✓ 1-1: Logging In and Help Overview
- ✓ 2-1: Creating and Connecting to ODI Master and Work Repositories
- ✓ 3-1: Configuring Standalone Agents Using the Common Admin Model
- ✓ 4-1: Working with Topology
- **5-1: Setting Up a New ODI Project**
 - 6-1: Creating Models by Reverse-Engineering
 - 7-1: Checking Data Quality in the Model
 - 8-1: Creating ODI Mapping: Simple Transformations
 - 9-1: Creating ODI Mapping: Complex Transformations
 - 9-2: Creating ODI Mapping: Implementing Lookup
 - 10-1: Creating ODI Mapping: Exporting a Flat File to a Relational Table
 - 11-1: Using Native Sequences with ODI Mapping
 - 11-2: Using Temporary Indexes
 - 11-3: Using Sets with ODI Mapping
 - 12-1: Creating and Using Reusable Mappings
 - 12-2: Developing a New Knowledge Module
 - 13-1: Creating an ODI Procedure
 - 14-1: Creating an ODI Package
 - 14-2: Using ODI Packages with Variables and User Functions
 - 15-1: Debugging Mappings
 - 16-1: Creating and Scheduling Scenarios
 - 17-1: Using Load Plans
 - 18-1: Enforcing Data Quality with ODI Mappings
 - 19-1: Implementing Changed Data Capture
 - 20-1: Setting Up ODI Security
 - 20-2: Integration with Enterprise Manager and Using ODI Console



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 5-1: Setting Up a New ODI Project



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In this practice, you create a project named HandsOnLoads with a folder named HandsOn. You import Knowledge Modules for working with three sources (MySQL, XML, and File) and one target (Oracle).

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

KISHORE ADHIMOOLAM (kishore.k.adhimoolam@verizon.com) has
a non-transferable license to use this Student Guide.

Oracle Data Integrator Model Concepts



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Describe the purpose of Oracle Data Integrator (ODI) models and reverse-engineering in ODI
- Describe the methods of reverse-engineering
- Create ODI models by reverse-engineering
- Use shortcuts
- Use Smart Export and Import



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

What Is a Model?

- A model is an ODI description of a relational data model.
- It is a group of datastores stored in a given schema on a given technology.
- A model contains metadata reverse-engineered from the “real” data model, or created in ODI.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A model in ODI is a description of a relational data model. It is a representation of the structure of a number of interconnected datastores stored in a single schema on a particular technology. The model in ODI does not, however, hold data, such as banking figures or names of customers. Instead, it holds only metadata: the names of columns or constraints that exist between tables.

This metadata can be imported automatically into ODI by a process called reverse-engineering. However, you can also define the metadata manually, creating columns or constraints directly in ODI. These additions do not necessarily exist in any physical database table.

This lesson covers both methods of creating model metadata.

Agenda

- **Understanding the Relational Model**
- Understanding Reverse-Engineering
- Creating Models



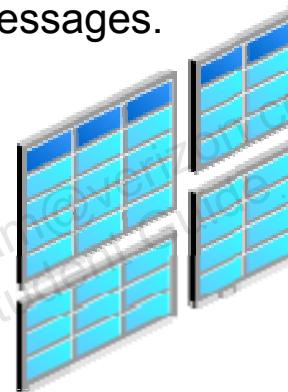
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

It is crucial that you have a solid understanding of the relational paradigm. In this lesson, you learn about the key concepts, and about how the relational paradigm is implemented in ODI.

Relational Model

- ODI is strongly based on the relational paradigm.
- In ODI, data is handled through tabular structures defined as datastores.
- Datastores are used for all types of “real” data structures, including flat files, Extensible Markup Language (XML) files, and Java Message Service (JMS) messages.



ORACLE®

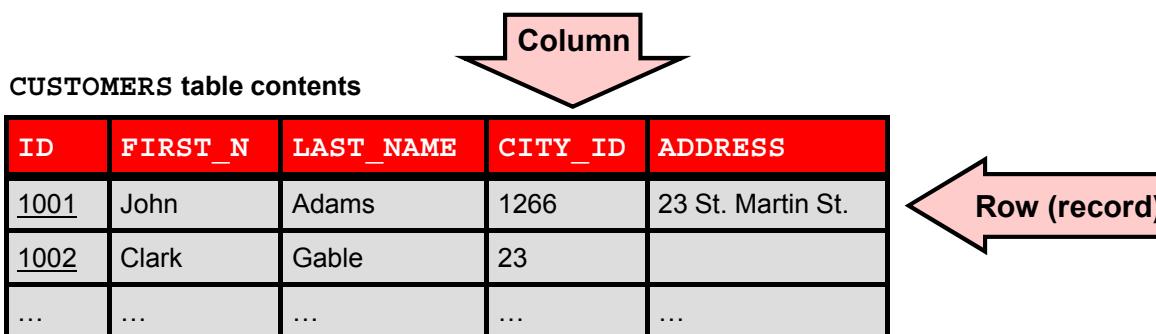
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ODI works with different databases and other sources of data. It is, therefore, not surprising that ODI is strongly based on the relational paradigm. This means that ODI uses the same terms and approach to data as is common to all database products. This commonality enables ODI to work seamlessly with all database engines.

According to this paradigm, the basic structure of data is the two-dimensional tabular structure, known as a table or datastore.

ODI uses the term “datastore” broadly to refer to the database tables, flat files, XML files, JMS messages, and so on. So, a database table is a datastore, but a datastore is not necessarily a database table.

Relational Model: Tables and Columns



- Tables store data in rows and columns.
- Each column has a data type.
 - Optionally, a length and precision
- Each column may or may not allow null values.
- Each column may have a default value.

CUSTOMERS table structure (example)

Column Name	Data Type
ID	NUMERIC(10), Not Null
FIRST_NAME	VARCHAR(255), Not Null
LAST_NAME	VARCHAR(255), Not Null
CITY_ID	NUMERIC(5), Not Null
ADDRESS	VARCHAR(500), Allows Nulls

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

First, you recall how a relational model works.

The table forms the basic structure and stores information relating to one subject, in this case, customers. Each row represents an individual customer, and contains different pieces of information about that one customer.

A column describes that piece of information. For example, the LAST_NAME column in this example contains the surname of each customer. You can find the surnames of all customers in the table by looking at the column.

Every column must have a data type that describes the kind of information stored. For example, the LAST_NAME field would contain a string of characters, not a number or date. At times, the length and precision are also stored. For example, you can limit the LAST_NAME field to 255 characters. Similarly, you may limit the ID number of a customer to 10 digits.

Each column can either allow or not allow null values. A null value means that no value is stored in this column for this row. The information is not known, not applicable, or not stored. By setting a column to “not null,” you prevent the gaps in data of this sort from occurring.

Columns may also have default values. These are values that are given by default when no other value is entered.

Relational Model: Keys

Candidate keys				
Primary key		Alternate Key		
ID	FIRST_N	LAST_NAME	CITY_ID	ADDRESS
1001	John	Adams	1266	23 St. Martin St.
1002	Clark	Gable	23	32 South Ave.
...

A *candidate key* can uniquely identify one row in the table.

- One of the keys is chosen as the *primary key* (PK).
- Other keys are called *alternate keys* (AK).



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

So far, you have not defined any rules on the data in the table or looked at what the ID field may be used for. Keys are important for defining relationships between tables—it is not useful to have information about customer orders if you are unable to look up the customer to find his or her address.

A candidate key is a set of columns that can uniquely distinguish one row in the table from any other row. In this example, each customer has a unique ID. So, you can find all information about a customer by just specifying the ID.

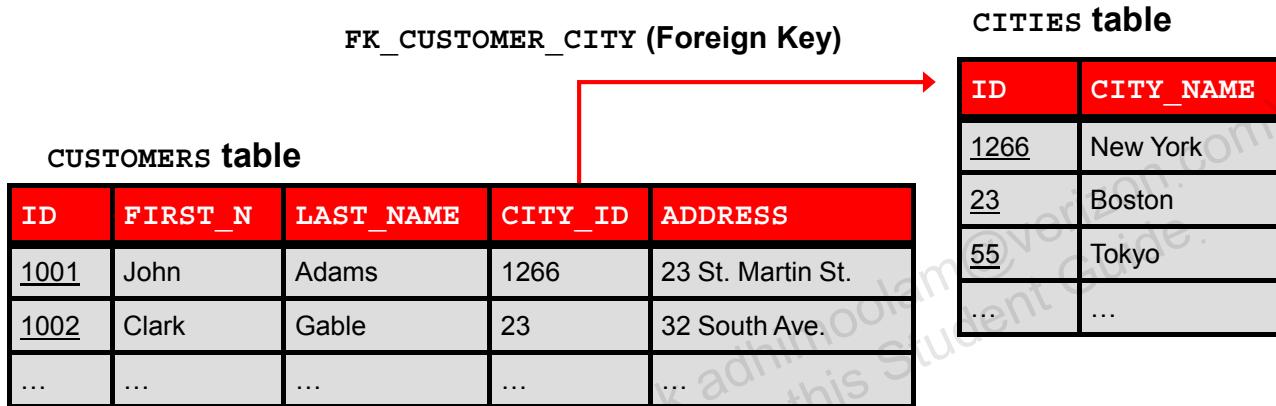
You can also find information about a customer by using the FIRST_NAME and LAST_NAME columns together. By specifying a customer's full name, you can locate him or her.

However, it is likely that you have two customers with the same name. Therefore, you select the ID column as the primary key. This means that this column must always contain unique values.

The other candidate keys are then referred to as the alternate keys. In this case, the alternate key is the combination of FIRST_NAME and LAST_NAME.

Relational Model: Foreign Keys

- Foreign keys (FK) define relationships between tables.
- A foreign key uses one or more columns from one table to reference a candidate key in another table.



ORACLE®

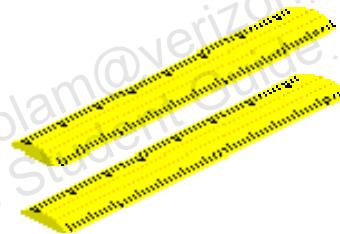
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Foreign keys are a way of linking several tables together. They are used to ensure the integrity of your information as a whole. More specifically, a foreign key defines a relationship between one or more columns in one table and a candidate key in another table.

In this example, the **CITY_ID** column in the **CUSTOMERS** table should always refer to an **ID** column in the **CITIES** table. Therefore, you define a foreign key from the **CITY_ID** column in the **CUSTOMERS** table to the **ID** column in the **CITIES** table. Note that foreign keys have a direction: Make sure that only the city ID of a customer exists. It is not important if there are no customers from a particular city.

Relational Model: Constraints

- Constraints enforce business rules on data.
- PKs, AKs, FKs, and Not Null are types of constraints.
 - A customer ID must be unique.
 - Two customers cannot have the same first name and last name.
 - A customer city must be referenced in the CITIES table.
 - A customer must have a last name.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Constraints are basically rules in the database that exist to enforce business rules on data. Primary keys, alternate keys, foreign keys, and not null columns are all types of constraints. The database prevents any data modification that does not respect these constraints. Here is a brief description of the different types again:

- The primary key on the `ID` column of the `CUSTOMERS` table ensures that each customer ID is unique.
- The alternate key comprising the `FIRST_NAME` and `LAST_NAME` columns in the `CUSTOMERS` table prevents any two customers having the same full name.
- The foreign key defined from the `CITY_ID` column in the `CUSTOMERS` table to the `ID` column in the `CITIES` table prevents invalid `CITY_ID` values from being added.
- Lastly, defining the `LAST_NAME` column as a “not null” column ensures that a value is always present in this field.

Relational Model: Constraints

- Check constraints are another type of constraint.
 - They map more complex business rules.
 - Example: “Every customer’s age must be between 16 and 120.”
- Constraints ensure quality of data.
 - A constraint key characterizes the identifier for an index.
 - (is unique, primary key, or not unique)
 - A constraint reference is a relationship between datastores; a constraint condition is the `where` clause.
 - Foreign keys are constraint references, not conditions.
- Trying to insert or update a value violating a constraint usually generates an error at the database level.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can define other, more complex types of constraints. For example, you can define check constraints on your tables. These constraints implement other kinds of business rules.

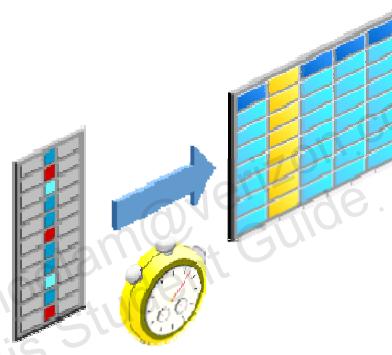
For example, you may know that all the customers of your organization are aged between 16 and 120. You can implement this rule as a check constraint. Thus, it will be impossible for invalid ages to be entered into the database.

The goal of constraints is to ensure quality of data. They do not define relationships between tables like foreign keys. Nor do they set a property for a whole table, like primary keys. Instead, they just define statements that must be true for each individual row of a table.

In the same way as other constraints, after check constraints are implemented and enabled, they prevent modifications that violate these rules. For example, attempting to create a customer with an age of zero usually triggers a database error. Thus, you are alerted immediately to the presence of “bad” data.

Relational Model: Indexes

- Indexes are defined on a group of columns in a table.
 - Indexes may speed up access to the table data when a query selects data through this group.
 - An index might not be used.
- Unique indexes are a specific type of index that prevent duplicate values.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Indexes are another property of database tables that are implemented in most database engines. They are not constraints, but a tool for improving database performance. Many factors go into determining whether an existing index is actually used in a given query. SQL tuning courses cover the specifics of index design and usage.

An index is defined on a group of one or more columns in a table. Thus, it is similar to a key. The difference is that an index is used specifically to speed up database access. A query that uses this group of columns to retrieve data can generally be performed faster if it has an index on it.

Unique indexes behave in the same way as unique keys. They are indexes with the additional property that duplicate values must not exist.

Relational Model Support in ODI

All elements of relational models can be described in ODI metadata:

Relational model	Description in ODI
Table; column	Datastore; attribute
Not null; default value	Not null/mandatory; default value
Primary keys; alternate keys	Primary keys; alternate keys
Indexes; unique indexes	Not unique indexes; alternate keys
Foreign key	Reference
Check constraint	Condition



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

As stated earlier, all key aspects of a relational model are supported in ODI.

Tables in a relational model are generally referred to as datastores in ODI. Columns (fields) in tables are referred to in ODI as attributes.

Not null columns can be specified directly in ODI or accessed from the database during reverse-engineering. They are often referred to as mandatory fields in ODI. Default values are similarly supported.

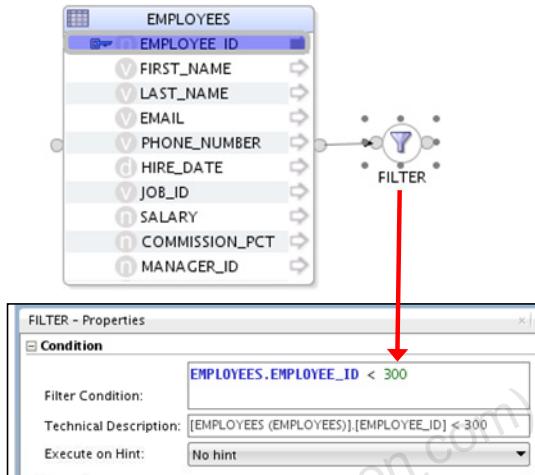
Primary keys and alternate keys have the same meaning in ODI. They can be enforced whether they exist in the database or not. ODI treats unique indexes as alternate keys and marks other types of indexes as not unique indexes.

Foreign keys in ODI are referred to as references. References in ODI can be defined between heterogeneous databases, which is not possible with traditional foreign keys.

Check constraints in a relational model are referred to as conditions in ODI for clarity. Thus, a condition is a type of constraint, along with references, primary keys, and so on.

Additional Metadata in ODI

- Filters:
 - Apply when data is loaded from a datastore
- Heterogeneous references:
 - Link datastores from different models or technologies
- Additional, optional, technical or functional metadata:
 - Read-only data types or columns
 - Online analytical processing (OLAP) type on datastores
 - Slowly changing dimension behavior on columns
 - User-defined metadata (FlexFields)



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ODI adds more capabilities to relational data models. For example, some types of metadata may not exist in most databases, but are useful to implement certain business rules.

Filters reduce the amount of data retrieved from source datastores by specifying conditions that must be matched, for example, processing only confirmed orders. This does not remove any data from the source datastores. It simply reduces the amount of data entering the data flow.

Heterogeneous references are foreign keys that work between heterogeneous systems. That is, you can specify that orders stored in an Oracle table reference a customer stored in a Sybase table. Although these references cannot exist in the database systems, ODI can implement them during integration.

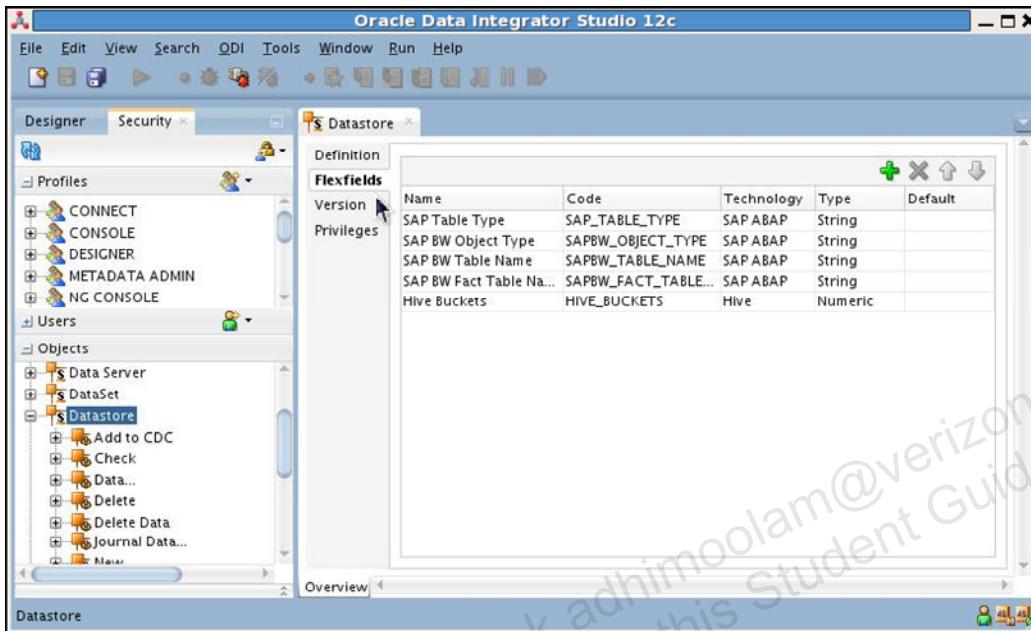
Other types of specialized metadata can optionally serve particular purposes. For example, datastores in ODI can have an OLAP type, such as dimension table or facts table.

Similarly, columns can have a “slowly changing dimension” behavior defined. For these columns, you use ODI Knowledge Modules that are designed specifically to manage the slowly changing dimension values of source datastores.

Read-only data types and columns serve similar purposes. They enable ODI to avoid inserting data into certain columns during the integration process.

ODI also enables you to define user-defined metadata in the form of FlexFields. For example, you can record that a datastore is a time or a regular dimension for OLAP purposes.

FlexFields



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

FlexFields are defined using the Security Navigator. They are defined for each object type (that can have FlexFields) by dragging the object (for example, “Datastore”) into the workspace to edit it.

Click the FlexFields tab to add them, after which you can edit them. Not all object types can have FlexFields defined for them.

Search “FlexFields” in Oracle Data Integrator Online Help for further information.

Agenda

- Understanding the Relational Model
- **Understanding Reverse-Engineering**
 - Retrieving Metadata
- Creating Models



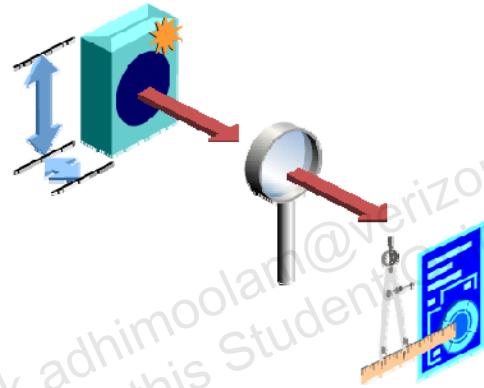
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Reverse-engineering in ODI is about automatically retrieving metadata from a database to populate an ODI model.

What Is Reverse-Engineering?

Reverse-engineering is an automated process to retrieve metadata to create or update a model in ODI.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Reverse-engineering is a completely automated process in ODI for retrieving metadata from a database. It can be used to create new models or to fill in gaps in the existing models.

You will use reverse-engineering in the practice to populate several models.

Methods for DBMS Reverse-Engineering

- Standard reverse-engineering:
 - Uses Java Database Connectivity (JDBC) features to retrieve metadata, which is then written to the ODI repository
 - Requires a suitable driver
- Customized reverse-engineering:
 - Reads metadata from the application/database system repository, and then writes this metadata in the ODI repository
 - Uses a technology-specific strategy, implemented in a Reverse-Engineering Knowledge Module (RKM)
 - For each technology, there is a specific RKM that tells ODI how to extract metadata for that specific technology.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ODI can perform reverse-engineering in two ways:

Standard reverse-engineering uses the standard capabilities of the JDBC API.

- ODI queries the JDBC driver for information about table structures, foreign keys, and so on. It then writes these to the ODI repository, where all metadata is stored.
- Although it is technically independent of the technology being used, standard reverse-engineering requires a sufficiently capable JDBC driver.

Customized reverse-engineering is quite different.

- In this case, ODI connects to the database by using the basic features of the JDBC driver and directly queries the system tables to retrieve the metadata.
- It then transforms and writes this metadata into the ODI repository.

Naturally, the method for doing this varies greatly depending on the technology that is being reverse-engineered. Thus, for each technology there is a specific RKM. This RKM tells ODI how to extract metadata for the given technology.

Other Methods for Reverse-Engineering

- Delimited format reverse-engineering
 - File parsing built in to ODI; based on file header
- Fixed-format reverse-engineering
 - Built in to ODI; uses COBOL copybook description files
- XML file reverse-engineering (standard)
 - Uses ODI JDBC driver for XML
- LDAP directory reverse-engineering (standard)
 - Uses ODI JDBC driver for LDAP



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In addition to standard and customized reverse-engineering for databases, you can also reverse-engineer files. There are two methods for this, as there are supported file formats.

Delimited files contain data separated by a delimiter, such as a comma or tab character. ODI can parse the file content directly and determine the number of columns and their name depending on the file header.

Fixed format files allocate a specific length to each field, rather than using a defined delimiter between fields. They are reverse-engineered with a different method.

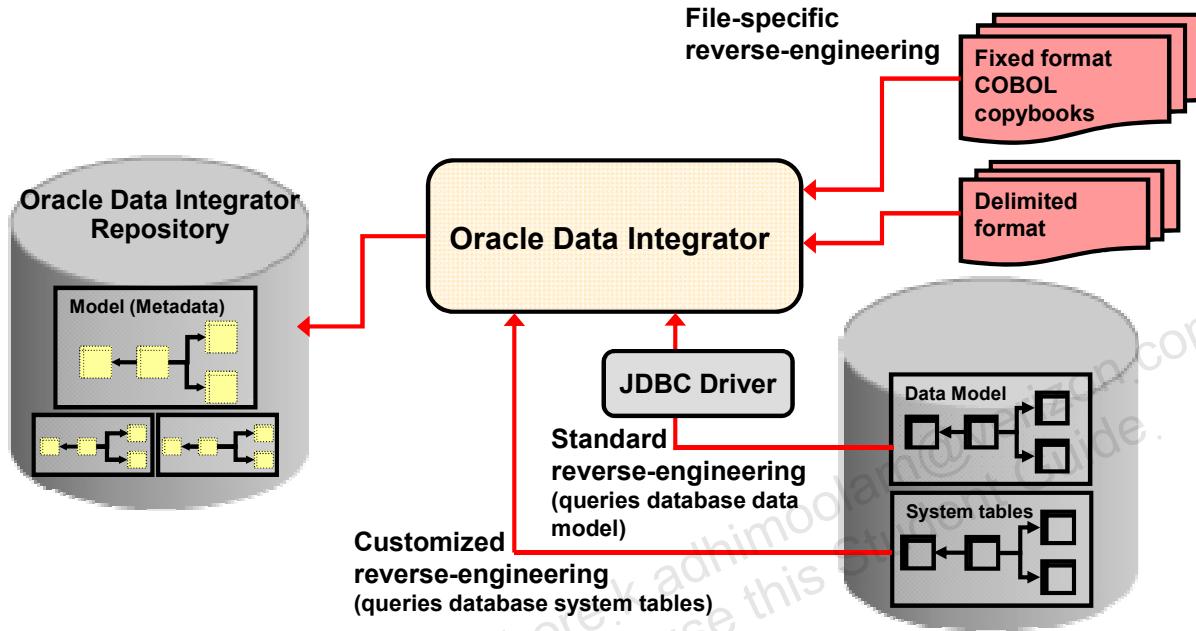
To collect metadata for these fields, you need a description of the fields in the COBOL copybook format. ODI can parse this type of file natively.

ODI can also reverse-engineer metadata in XML files through standard reverse-engineering. ODI uses the ODI JDBC driver for XML to access them like any other data source.

Similarly, lightweight directory access protocol (LDAP) directories can be reverse-engineered by using the standard method. Here, ODI uses the ODI JDBC driver for LDAP.

You should, therefore, consider XML files and LDAP directories as normal databases for the purposes of reverse-engineering.

Standard Versus Customized Reverse-Engineering



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A graphical representation of the differences between the types of reverse-engineering is shown in the slide. In the example shown, the goal is to get the models into ODI. The metadata of the model represents the structure of the data that is in a database, XML file, or LDAP directory.

Standard reverse-engineering queries the JDBC driver directly to retrieve the structure of these tables. However, you must remember that all databases must store their own metadata.

Often, this metadata is made available in the form of system tables. ODI can also query these system tables to extract the metadata directly. This is customized reverse-engineering.

As mentioned earlier, data can also be reverse-engineered from files. For example, ODI directly reads metadata from COBOL copybook files without passing through the JDBC driver.

Similarly, delimited format files are parsed directly.

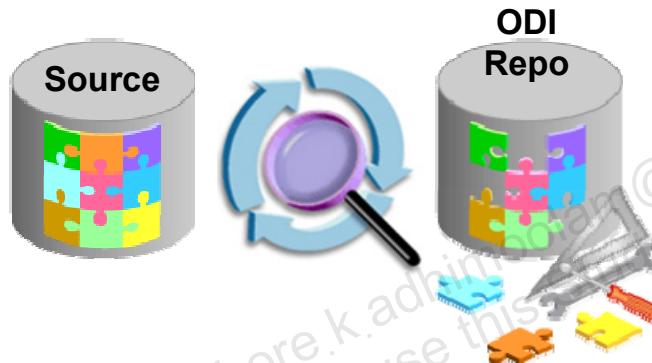
Having collected the metadata, ODI now stores it in the repository. This is where metadata for all models, datastores, projects, mappings, and so on, are stored.

In this example, the model that was originally retrieved is now stored in the repository. This includes a representation of the datastores, and the joins and foreign keys linking them.

It thus becomes one of a collection of many models that can be stored in the same repository.

Reverse-Engineering Life Cycle

- Reverse-engineering is iterative.
- New metadata is added, but old metadata is not removed.
 - If a table is removed from the database, you must manually remove the datastore from ODI as well.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

An important property of reverse-engineering in ODI is that it is an iterative, or incremental, procedure. ODI does not destroy the old metadata and then replace it with the new metadata. It just checks what is missing or changed from the current model and updates it with the new information from the database.

ODI never removes metadata from the repository during a reverse-engineering operation. For example, if a table is removed from the database, you must manually remove the datastore from ODI as well.

Agenda

- Understanding the Relational Model
- Understanding Reverse-Engineering
- **Creating Models**



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Next, take a look at the steps for creating ODI models in the tool's windows.

Creating a Model by Reverse-Engineering

1. Create an empty model.
2. Set up the reverse-engineering strategy.
 - Standard versus customized reverse-engineering
3. Run the reverse-engineering process.
 - Use the selective reverse option (if using standard reverse-engineering).
4. Develop the model.
 - Add datastores, columns, constraints, and so on.



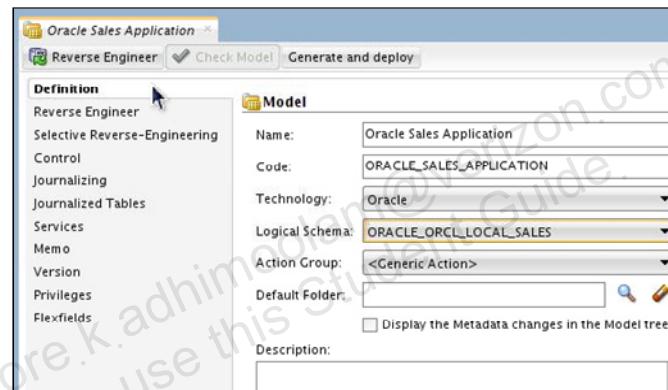
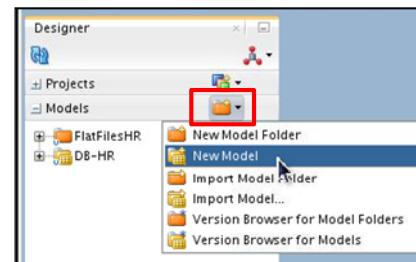
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The basic procedure to create a model in ODI is as follows:

1. Create and set up an empty model.
2. Define the reverse-engineering strategy that the ODI will follow. This includes selecting a Knowledge Module and setting up other parameters. In particular, you must choose between standard and customized reverse-engineering.
3. Tell ODI to run the reverse-engineering process using the strategy you defined. If you use standard reverse-engineering, you can choose to import only certain metadata. This is known as performing a selective reverse.
4. Flesh out the model that you have reverse-engineered. This means that you add information that was not retrieved during the reverse-engineering process. For example, you add datastores, columns, constraints, or ODI-specific metadata.

Step 1: Creating and Naming a New Model

1. Select the Models view.
2. Click the New Model icon.
3. Enter the name (and code).
4. Select the model technology.
5. Select the logical schema where the model is found.
6. Enter the description (optional).



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The basic steps in creating and naming a model are as follows:

1. Select the Models view. You can select a model folder to create the model in. Otherwise, you create a top-level model.
2. Click the New Model icon. The window for the new model appears.
3. Enter the name of your model. Like most objects in ODI, there is also a code to enter. By default, this is the same as the name, but uppercase with underscores instead of spaces.
4. A model must be linked to a specific technology. Specify the technology of the underlying database here.
5. A model always belongs to a specific logical schema. This specifies where in your logical architecture it is found. You later specify a context, which determines the physical schema that will be used as the source of the metadata. Select the logical schema from the drop-down list.
6. Enter a description for your model.

Note: You may notice the “Display the Metadata changes in the Model tree” check box. If you select this check box, ODI automatically compares its version of the model with the database, whenever the Model tree is refreshed. This can be useful, but slows down ODI.

Note: Creating and Naming a New Model

- A model is always defined in a given technology.
- If you change a model's technology, you must recheck every object related to that model.



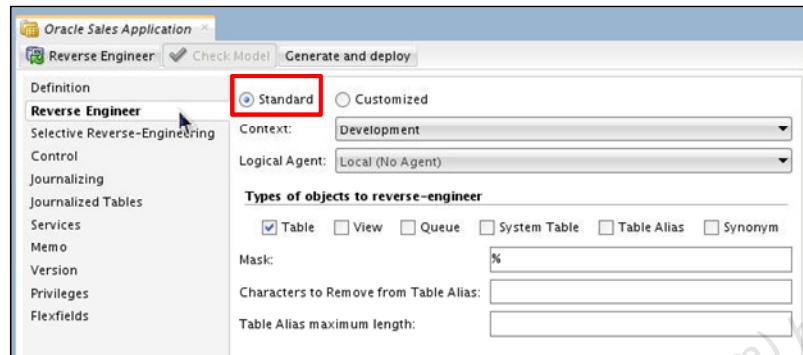
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

As mentioned earlier, a model is always linked to a specific technology. For example, your sales model may be defined as being an Oracle model. This defines the language and feature set that is used for all objects related to this model, including mappings, procedures, and packages.

It is, however, possible to change the technology of a model. You must then carefully check every related object to ensure that it is compatible with the new technology. You also need to change the Knowledge Modules, such as Check Knowledge Modules (CKMs) and Journalizing Knowledge Modules (JKMs), and possibly rewrite the mappings that use the model.

Step 2: Defining a Reverse-Engineering Strategy

1. Click the Reverse Engineer tab.
2. Select the reverse-engineering type.
3. Select the context for reverse-engineering.
4. Select the object type (optional).
5. Enter the object name mask and the characters to remove for the table alias (optional).
6. If customized:
 - Select the RKM
 - Select the Logical Agent



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

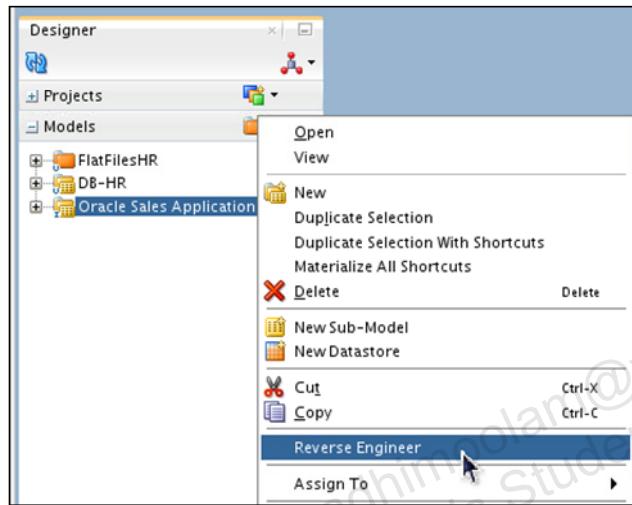
To define a reverse-engineering strategy, perform the following:

1. Click the Reverse Engineer tab in the model window.
2. Select the reverse-engineering type to be used. The standard mode requires a JDBC driver, which provides metadata for the specific technology. The customized mode requires a specific ODI Knowledge Module to directly access the system tables. This choice causes some differences in the reverse-engineering process, as will be discussed.
3. Select the reverse-engineering context. Remember that you only defined a logical schema, which describes where the model would fit in the logical architecture. However, this logical schema could correspond to several physical schemas in different physical contexts, such as development, testing, and production.
4. By selecting the context, you specify which one of those physical schemas will be used to provide metadata for the model. You should therefore select the context that best conforms to the structure of the model that you want to create.

5. By default, tables or datastores that closely resemble tables are reverse-engineered. However, you can choose to reverse-engineer different types of objects, such as views or table aliases.
6. By default, all objects of the correct type are reverse-engineered. However, there are situations where you want to reverse tables that are named according to a certain mask. For example, to load tables starting with “SALES,” set the mask to “SALES” followed by a percent sign (“SALES%”).
7. Similarly, you can specify a group of characters to remove to derive the alias for each table. This alias is used in mappings and is usually the first three characters of the table name. In the example, set the mask to “SALES.” Thus, a table with the name “SALES DATA” generates an alias of “DAT.”
8. You must set two additional options if you are using the customized mode: You must set the RKM, which tells ODI how to retrieve metadata for this technology. The RKM must have been imported into your project. You should also specify the logical agent to be used for executing the reverse-engineering tasks.

Step 3: Starting the Reverse-Engineering Process

Right-click the model and select Reverse Engineer.



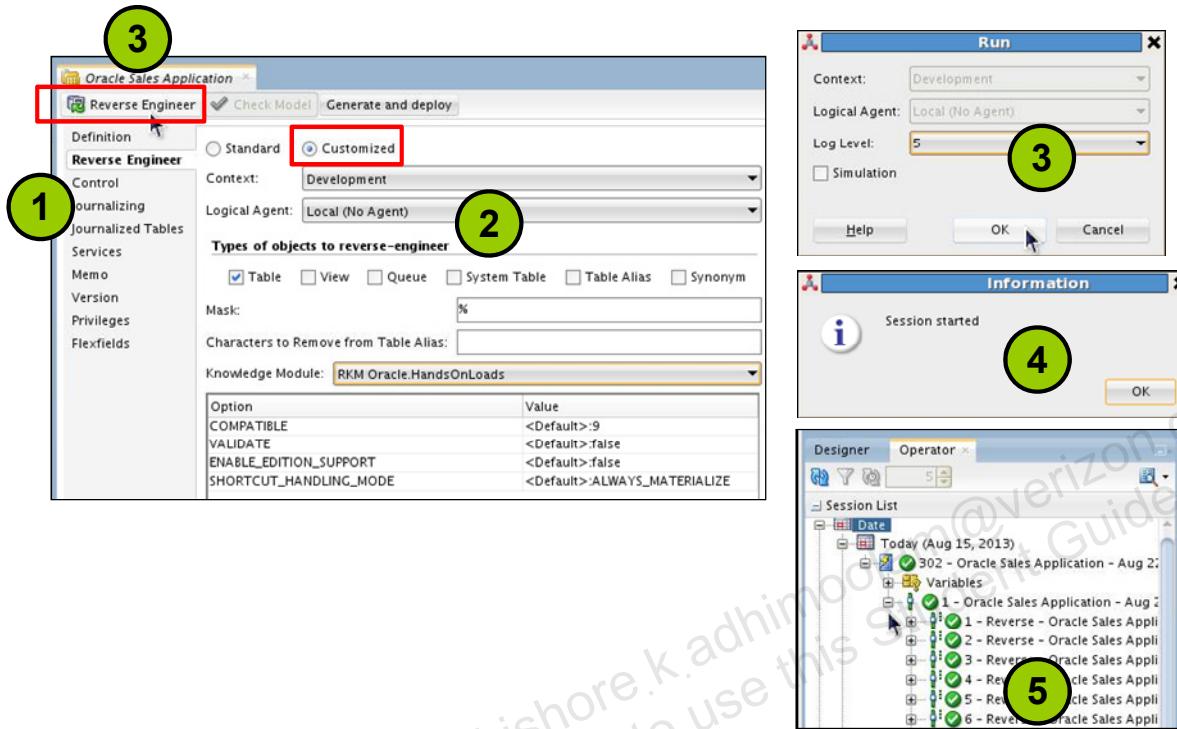
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To start the reverse-engineering process, right-click the model, and then select Reverse Engineer.

Note: You cannot use this method to perform Selective Reverse Engineering, which is discussed later.

Using RKM for Customized Reverse-Engineering



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Reverse-Engineering Knowledge Modules (RKMs) are used to perform customized reverse-engineering of data models for a specific technology. Customized reverse-engineering is more complete and entirely customizable, but also more complex to run. RKMs are provided only for certain technologies.

To perform customized reverse-engineering by using an RKM, perform the following steps:

1. Go to the Reverse Engineer tab of the model.
2. Click the Customized option button and fill in the following fields:

Context: Context used for the reverse-engineering

Object Type: Type of objects to reverse-engineer (tables, view, and so on)

Mask: %

Select the Knowledge Module: "<Project_Name>.<Project_Folder>" for example, "Knowledge Module: HandsOnLabs.HandsOnLoads"

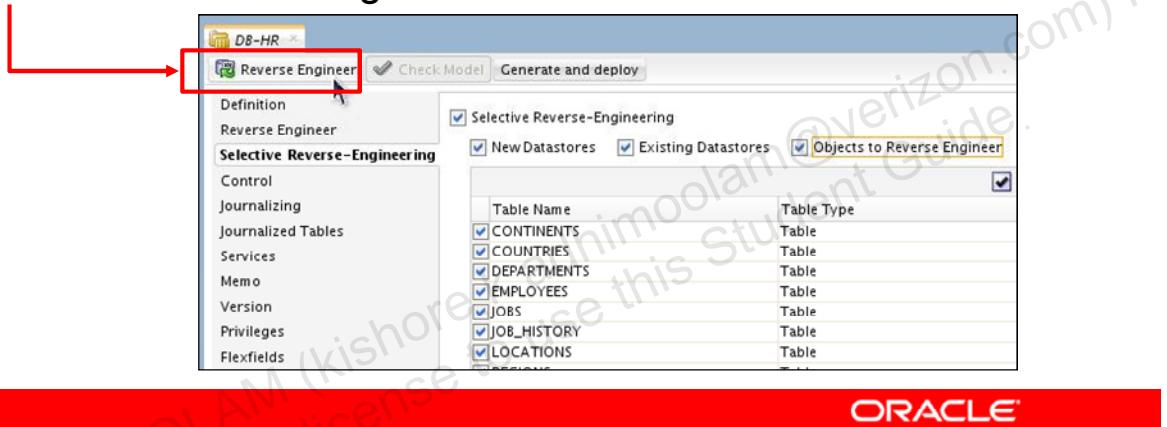
3. Click the Reverse button, and then click Yes to validate the changes. Click OK.
4. The "session started" window appears. Click OK.

5. Follow and validate your model's reverse-engineering operations in the Oracle Data Integrator Log. If it finished correctly, the reversed datastores appear under the reversed module in the tree.

Note: Using an RKM requires the import of this RKM into a project. Only imported RKMs are available for reverse operations. You can refine the reverse-engineering process by using the options of the Reverse and Selective Reverse tabs. See the RKM description in *Oracle Data Integrator Knowledge Modules Reference Guide*.

Selective Reverse-Engineering

1. Click the Selective Reverse-Engineering tab.
2. Select the Selective Reverse-Engineering check box.
3. Select the New Datastores and/or Existing Datastores check box.
4. Click the objects to reverse.
5. Select the datastores to reverse-engineer.
6. Click the Reverse Engineer button.



ORACLE

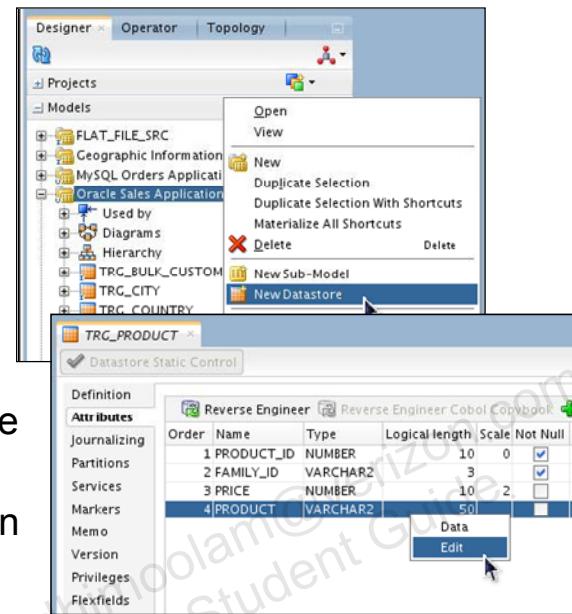
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

If you use standard reverse-engineering, you can individually select the tables to reverse-engineer. To do so, perform the following:

1. Click the **Selective Reverse-Engineering** tab.
2. Select the **Selective Reverse-Engineering** check box. This enables the next few options.
3. Now, you can select the **New Datastores** check box to reverse-engineer the datastores that do not exist in your model. Selecting the **Existing Datastores** check box selects the datastores that exist in your model. Selecting both options selects all datastores for reverse-engineering.
4. Select the **Objects to Reverse Engineer** check box to select the individual datastores that you want to include. The datastores that are displayed depend on the New Datastores and Existing Datastores options that you just selected.
5. Select (with a check mark) the datastore that you want to include in the reverse-engineering process.
6. Click the **Reverse** button to launch the process.

Step 4: Fleshting Out Models

- Add, remove, or edit model elements manually.
 - Use Designer.
 - Updates only ODI model, not database
- The model diagram is a GUI for editing models.
 - You can update the database with your changes.
 - Or, you can create a model in ODI, and then generate the code to implement this on a database server. (This is rarely done.)



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can manually add, remove, or edit any element of a model by using the ODI mapping. This includes creating datastores, columns, keys, and so on.

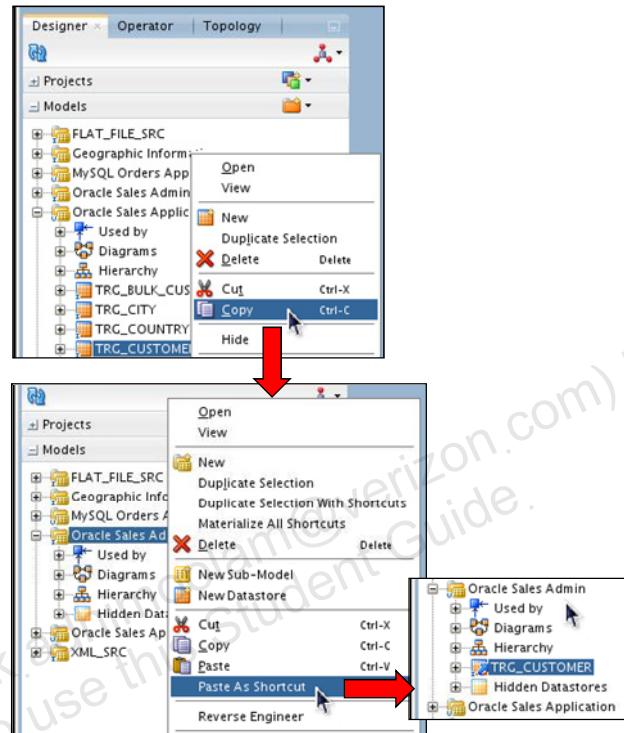
You can do this through the Designer Navigator. Note that these changes are applied only to the model in ODI. They do not update the underlying model in the database.

To edit models graphically, you can edit the model's diagram, a graphical depiction of the model and the relationship between its elements. You can drag and drop elements to define new relationships. This component also enables ODI to modify the model in the original database with your changes. Thus, you can create your model in ODI, and then generate the code to implement this on a database server.

Shortcuts

Shortcuts are links that designate common objects stored in separate locations.

1. Right-click a datastore.
2. Select Copy to copy the object.
3. Right-click a different model.
4. Select Paste As Shortcut to create a shortcut to this datastore in a separate model.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Shortcuts greatly improve productivity by allowing end users to express the large commonality that often exists between two different versions of the same source application, such as same tables and columns, constraints, and transformations.

Shortcuts are links to common Oracle Data Integrator objects that are stored in separate locations, and can be created for datastores, integration mappings, packages, and procedures. In addition, release tags have been introduced to manage the materialization of shortcuts based on specific tags. Release tags can be added to folders and model folders.

A referenced object is an object that is directly referenced by the shortcut. The referenced object of a shortcut may be a shortcut itself.

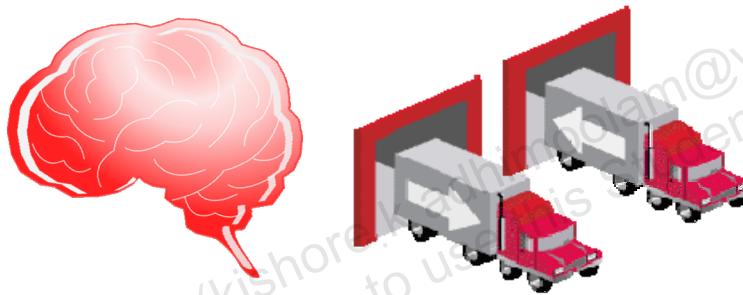
The base object is the original base object. It is the real object associated with the shortcut. Changes made to the base object are reflected in all the shortcuts created for this object.

When a shortcut is materialized, it is converted in the design repository to a real object with the same properties as the ultimate base object. The materialized shortcut retains its name, relationships, and object ID.

For more information, see Chapter 17 of the *Developer's Guide for ODI*.

Smart Export and Import

- Smart Export automatically exports an object with all its object dependencies.
- Smart Export and Import:
 - Use a lightweight and consistent export and import mechanism that provides several smart features
 - Help avoid most of the common issues that are encountered during an export or import, such as broken links or ID conflicts



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Exporting and importing Oracle Data Integrator objects between repositories is a common practice when working with multiple environments such as Development, Quality Assurance, and Production. The new Smart Export and Import feature guides you through this task, and provides advanced code management features.

Smart Export automatically exports an object with all its object dependencies. It is particularly useful when you want to move a consistent lightweight set of objects from one repository to another, including only a set of modified objects.

The Smart Export and Import feature is a lightweight and consistent export and import mechanism that provides several key features such as:

- Automatic and customizable object matching rules between the objects to import and the objects already present in the repository
- A set of actions that can be applied to the object to import when a matching object has been found in the repository
- Proactive issue detection and resolution that suggests a default working solution for every broken link or conflict detected during the Smart Import

For more information about Smart Export and Import, see Chapter 20, section 20.2.7, of the *Developer's Guide for ODI*.

Quiz

The goal of ODI constraint conditions is to ensure the quality of data. Which of the following statements is true?

- a. ODI constraint conditions define statements that must be true for each individual row of a table.
- b. ODI constraint conditions set properties for the whole table.
- c. ODI constraint conditions define the relationship between tables.
- d. All of the above
- e. None of the above



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Explanation: The goal of ODI user-defined constraints is to ensure quality of data. They do not define relationships between tables like foreign keys. Neither do they set a property for a whole table, like primary keys. Instead, they just define statements that must be true for each individual row of a table.

A constraint **key** characterizes the identifier for an index (is unique, primary key, or not unique). A constraint **reference** is a relationship between datastores. A constraint **condition** is the where clause. Foreign keys are constraint references, not conditions.

You are dealing here with ODI user-defined constraint conditions, not database constraints. ODI user-defined constraints just add a where clause to check the data. Answer c would also be correct if you were dealing with database constraints, but not for ODI constraints.

Quiz

You manually add and edit some elements of a model by using the ODI Designer. After the model is saved in ODI, these changes always update the underlying model in the database.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Explanation: You can manually add, remove, or edit any element of a model by using the ODI mapping. You can do this through the Designer component.

However, these changes are applied only to the model in *ODI*. They do not update the underlying model in the *database*. If you generate the DDL, the database model is not updated.

Summary

In this lesson, you should have learned how to:

- Describe the purpose of ODI models and reverse-engineering in ODI
- Describe methods of reverse-engineering
- Create ODI models by reverse-engineering
- Use shortcuts
- Use Smart Export and Import



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Checklist of Practice Activities

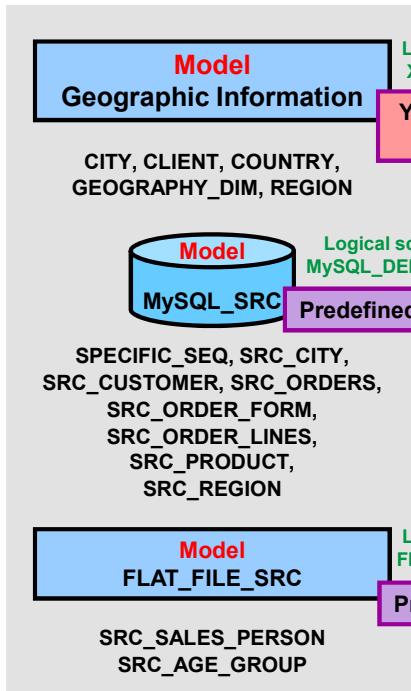
- ✓ 1-1: Logging In and Help Overview
- ✓ 2-1: Creating and Connecting to ODI Master and Work Repositories
- ✓ 3-1: Configuring Standalone Agents Using the Common Admin Model
- ✓ 4-1: Working with Topology
- ✓ 5-1: Setting Up a New ODI Project
- **6-1: Creating Models by Reverse-Engineering**
 - 7-1: Checking Data Quality in the Model
 - 8-1: Creating ODI Mapping: Simple Transformations
 - 9-1: Creating ODI Mapping: Complex Transformations
 - 9-2: Creating ODI Mapping: Implementing Lookup
 - 10-1: Creating ODI Mapping: Exporting a Flat File to a Relational Table
 - 11-1: Using Native Sequences with ODI Mapping
 - 11-2: Using Temporary Indexes
 - 11-3: Using Sets with ODI Mapping
 - 12-1: Creating and Using Reusable Mappings
 - 12-2: Developing a New Knowledge Module
 - 13-1: Creating an ODI Procedure
 - 14-1: Creating an ODI Package
 - 14-2: Using ODI Packages with Variables and User Functions
 - 15-1: Debugging Mappings
 - 16-1: Creating and Scheduling Scenarios
 - 17-1: Using Load Plans
 - 18-1: Enforcing Data Quality with ODI Mappings
 - 19-1: Implementing Changed Data Capture
 - 20-1: Setting Up ODI Security
 - 20-2: Integration with Enterprise Manager and Using ODI Console



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 6-1 Overview: Creating Models by Reverse-Engineering

Data sources



You will reverse-engineer.

Predefined

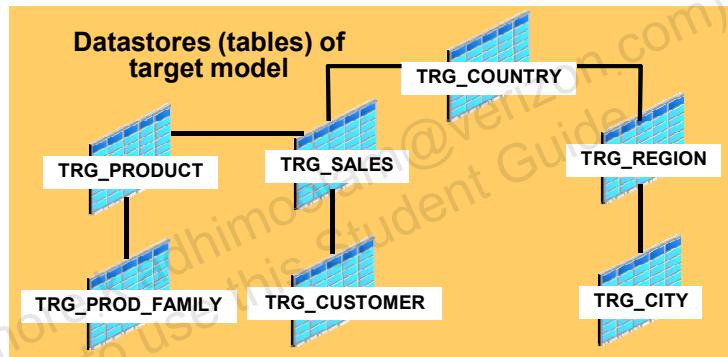
Predefined

Target



Logical schema: ORACLE_ORCL_LOCAL_SALES

You will reverse-engineer.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the previous practice, you configured the schemas containing the application datastore in the Oracle 12c database.

You now create the models corresponding to this data and reverse-engineer the schemas' data structures. You also reverse-engineer the structure of an XML file.

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

KISHORE ADHIMOOLAM (kishore.k.adhimoolam@verizon.com) has
a non-transferable license to use this Student Guide.

Organizing ODI Models and Creating ODI Datastores

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

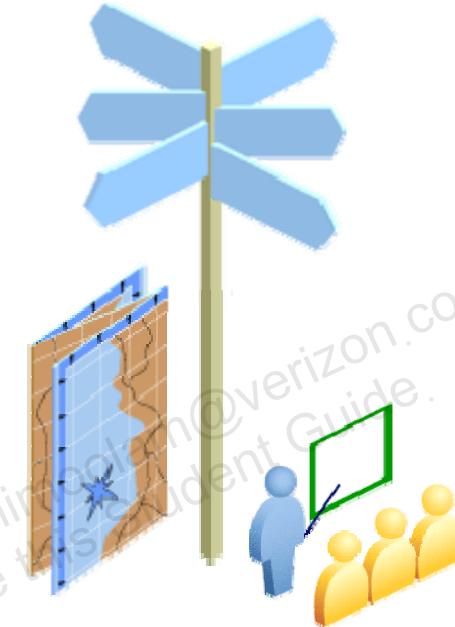
- Organize Oracle Data Integrator (ODI) models into folders
- Create datastores in ODI models
- Create constraints in ODI:
 - Mandatory columns
 - Keys and references in ODI models
 - Check conditions
- Explore data
- Audit:
 - ODI models
 - Datastores



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Agenda

- **Organizing Models**
 - Model Folders
 - Submodels
- Creating Datastores
- Constraints in ODI
- Creating Keys and References
- Creating Conditions
- Overview
- Exploring Your Data
- Constructing Business Rules



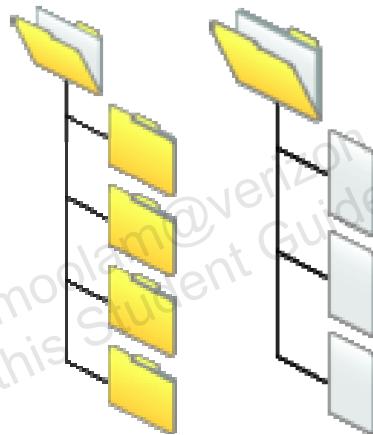
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

First, you learn about the two ways to organize data models: into model folders that contain data models, and into submodels that contain datastores.

What Is a Model Folder?

- A model folder is an optional means of grouping several models together.
- It can be used to organize models into a meaningful tree structure; for example, models relating to customers.



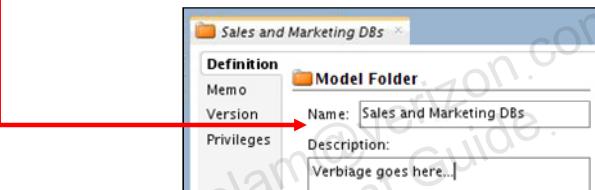
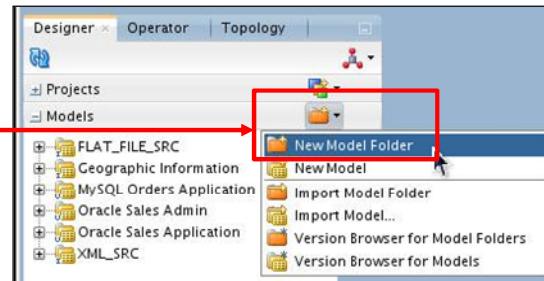
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A model folder is a device used to group several models together. You can use a model folder to create a meaningful collection, such as models relating to customers. Because model folders can also contain other model folders, you can build a tree structure this way.

Creating a Model Folder

1. In the Models view, click the New Model Folder icon, and select New Model Folder.
2. Enter the name.
3. Now you can drag models or other model folders into the folder.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a model folder, perform the following:

1. Select the Models view. Click the New Model Folder icon, and select New Model Folder from the menu.
2. The new Model Folder window appears. Enter the name to describe the model folder.
3. Click OK to save your changes.

Now you can drag models or other model folders into your new model folder.

What Is a Submodel?

- A submodel is a group of several datastores in the same model.
- Datastores can be arranged manually or automatically into submodels.

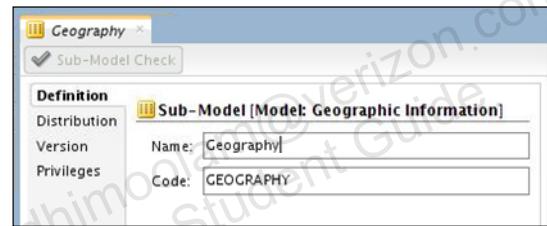
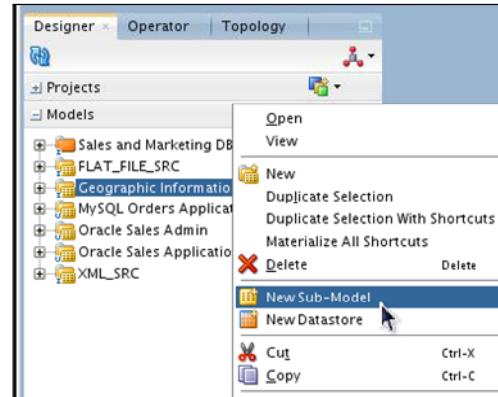


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A submodel is effectively the opposite of a model folder. It groups several datastores within a model. Thus, it serves to divide a model into subcomponents. In ODI, you can create submodels manually or automatically to distribute datastores into submodels.

Creating a Submodel

1. Select a model.
2. Right-click and select New Sub-Model.
3. Enter the name and code.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a submodel, perform the following:

1. Select the model that will contain the submodel.
2. Right-click and select New Sub-Model from the context menu.
3. Enter the name for the submodel. You must also specify a unique code.

Note: You cannot create a submodel directly in a model folder; you can only create a submodel within a model. Submodels can be created under models or submodels. You can have as many levels as you want.

Organizing Datastores into Submodels

The three possible ways to achieve this are:

- Manually: Drag datastores and submodels into submodels.
- Set up automatic distribution:
 - Datastores are distributed into submodels according to a name mask.
 - This moves all datastores or only those not already in a submodel.
 - Masks can be applied in a specific order after reverse-engineering.
 - It is possible only when using standard reverse-engineering.
- Use certain customized Reverse-Engineering Knowledge Modules (RKMs) to distribute datastores into submodels.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can organize your datastores into submodels in several ways.

If you want to organize your datastores manually, you just have to drag the datastores into submodels. You can also drag submodels into other submodels. However, this may not be practical when you have a large number of datastores. In this case, you can use automatic distribution.

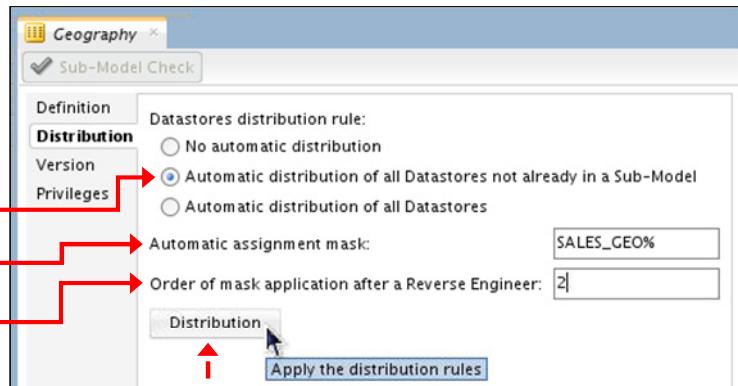
For automatic distribution, you define a mask that is compared with the datastores. Any datastore matching the mask is moved into that submodel.

You can choose to move only those datastores not currently in a submodel or all datastores. You can also specify the order in which to apply the masks, to determine what happens when a datastore matches several masks. This kind of automatic distribution is possible only when using standard reverse-engineering, without an RKM.

However, certain specialized RKMs can distribute datastores into submodels. The ODI Open Connectors to enterprise resource planning (ERP) packages use this feature to automatically organize the large number of tables reverse-engineered in the ERP models.

Setting Up Automatic Distribution

1. Click the Distribution tab.
2. Select a distribution rule.
3. Enter the mask.
4. Set the order in which the mask is applied.
 - Higher numbers take precedence.
5. Optionally, click the Distribution button to run this rule.



ORACLE

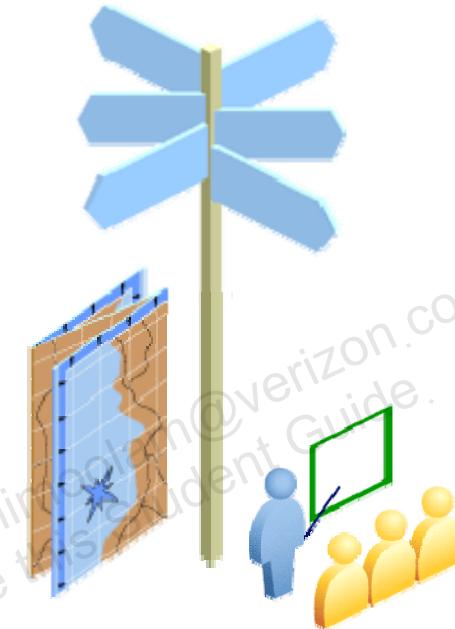
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Automatic distribution is set up on a submodel. It applies a mask to the names of datastores that have been reverse-engineered. If the mask matches, it moves the datastore to this submodel. To set up automatic distribution, perform the following:

1. Click the Distribution tab.
2. Select a distribution rule. If you have manually organized your datastores into submodels, select “Automatic distribution of all Datastores not already in a Sub-Model.” If you want to reorganize all datastores, select “Automatic distribution of all Datastores.”
3. Enter details in the mask field. This is specified in SQL format by using a percent sign to match any number of characters. For example, to move a datastore whose name begins with SALES_GEO, specify a mask of SALES_GEO%.
4. Now select the order in which this mask is applied. This is used to resolve conflicts when a datastore name matches several masks. For example, another submodel might have a distribution mask of %PARIS%, with a mask order of 5. Where should a datastore named SALES_GEO_PARIS go? The answer is that the mask with the higher order number, in this case, %PARIS%, takes precedence.
5. To have the distribution applied immediately, click the Distribution button. Otherwise, it is applied the next time reverse-engineering is performed on the model.

Agenda

- Organizing Models
- **Creating Datastores**
 - Reverse-Engineering
 - Manually
- Constraints in ODI
- Creating Keys and References
- Creating Conditions
- Overview
- Exploring Your Data
- Constructing Business Rules



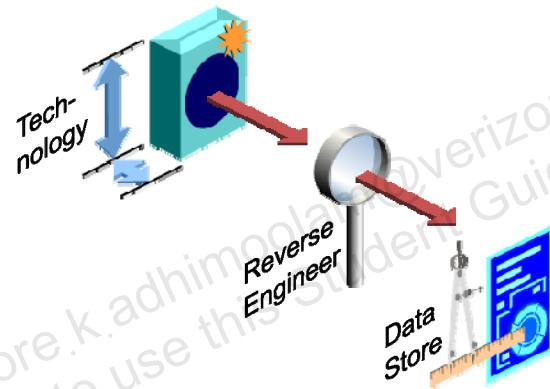
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Having just learned how models, model folders, and submodels are organized, you now learn how to create datastores manually.

Creating Datastores

- Datastores are usually created by reverse-engineering.
- However, you can create datastores manually:
 - In a model
 - Using Common Format Designer
 - Datastores do not need to exist in the data server.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

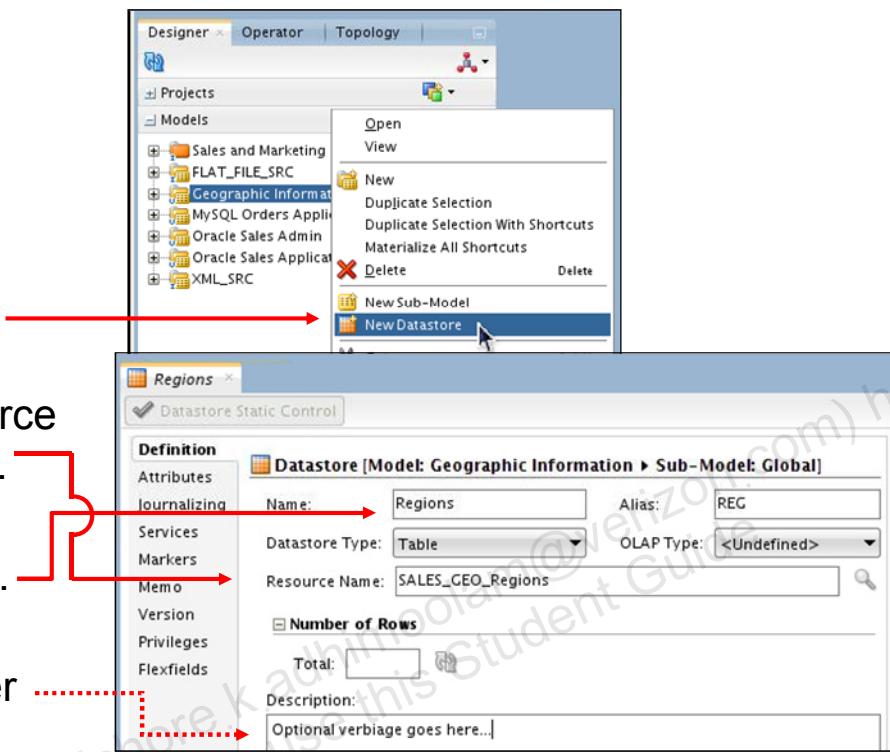
The easiest and most common way to create datastores in ODI is by reverse-engineering their structure from the technology (for example, Oracle or MySQL database). However, you can create datastores in two other ways:

- Directly in the model, as you see in the next few slides
- By using the Common Format Designer component (this infrequently discussed component is required for creating diagrams)

In these cases, the datastores do not necessarily have to exist in the data server. This can be useful for creating a new text file with a structure that you define.

Creating a Datastore in a Model

1. Select a model.
2. Right-click and select New Datastore.



ORACLE®

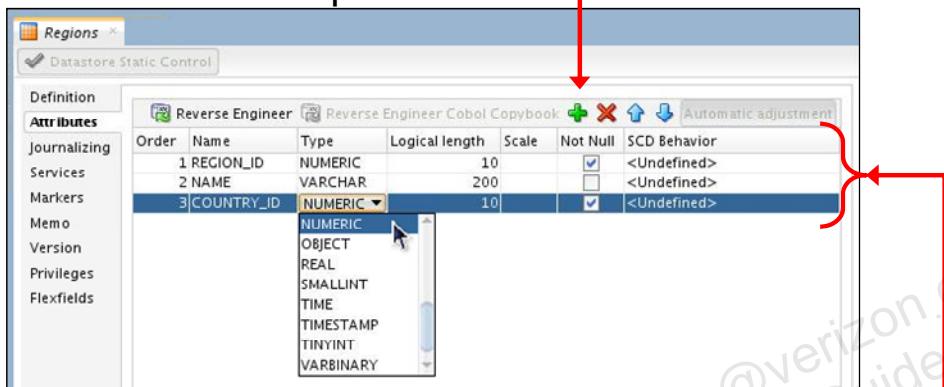
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a datastore directly in a model, perform the following:

1. Select a model.
2. Right-click and select New Datastore.
3. The Resource Name field refers to the physical name of the table in a database or the name of the file for a file-based datastore. If you use a mapping to create this datastore in a database, the resource name is the name of the table that is generated. The alias, on the other hand, is the name that you use to refer to the datastore in mappings, joins, and so on.
4. Set the name. This is the name that is displayed in ODI in the model tree view. Often, this is similar to the resource name.
5. Provide a description for the datastore (optional).

Adding Columns to a Datastore

1. Click the Attributes tab.
2. Click the Add Column “plus” icon.



3. Define the column's properties:
Name, Type, Length, Scale, Not Null, SCD Behavior
4. Repeat, and order the columns.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

After you set up the basic parameters, you may want to add columns to the definition. To do this, perform the following:

1. Click the Attributes tab.
2. Click the Add Column icon to add another column.
3. Define the properties for each column. The meanings of the Name, Type, Logical length, and Scale options depend on the technology of the model. Not Null means that the field is mandatory. This will be covered in greater detail later. You can set other properties in the columns by double-clicking their names in the Models view. “SCD” is Slowly Changing Dimension.
4. After you add all your columns, you can reorder them by using the up and down arrows.

Agenda

- Organizing Models
- Creating Datastores
- **Constraints in ODI**
- Creating Keys and References
- Creating Conditions
- Overview
- Exploring Your Data
- Constructing Business Rules



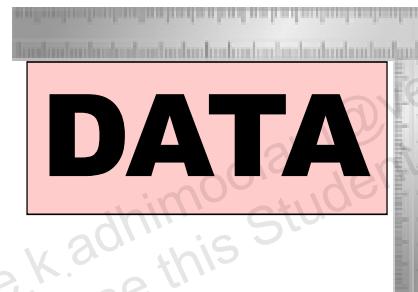
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The following slides describe adding constraints to datastores. This is a general term for the various types of rules that you want to impose on your data model.

What Is a Constraint in ODI?

- A constraint is a statement that defines the rules enforced on data in datastores.
- A constraint ensures the validity of the data in a given datastore and the integrity of the data of a model.
- Constraints on the target datastore are used in mappings to check the validity of the data before integration in the target.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A constraint defines the rules that are enforced on data in datastores. Constraints on source datastores ensure that invalid data is kept out of the data flow. Oracle Data Integrator manages constraints on data models including Keys, References, Conditions, and Mandatory Attributes. It includes a data integrity framework for ensuring the quality of a data model based on these constraints. You can also define constraints on the target datastore. This serves to test that the mapping itself is functioning correctly.

Constraints in ODI

There are four basic types of constraints in ODI:

- Mandatory columns known as “not null”
- Keys, including:
 - Primary keys
 - Alternate keys
 - Indexes
- References (equivalent to foreign keys):
 - Simple: column A = column B
 - Complex: column A = function (column B, column C)
- Conditions

ORACLE

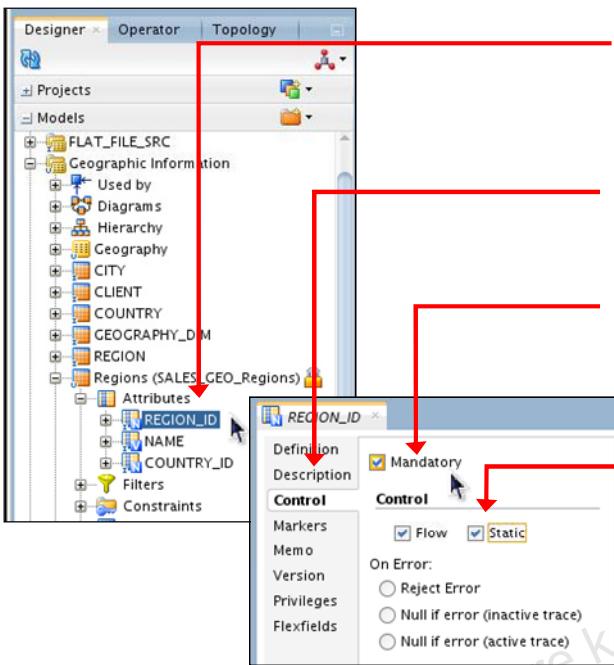
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

There are four basic types of constraints in ODI:

- Mandatory columns are often known as “not null” columns. You just specify that a column must have a non-null value for a given row to be valid.
- Keys include primary keys, alternate keys, and indexes.
- References are equivalent to foreign keys. In ODI, these can be either simple equalities or complex relationships between columns in different tables.
- Conditions are expressions that must evaluate to true for a row to be considered valid.

You will see how to add each of these types of constraints to a model in the next few slides.

Creating a Mandatory Column



1. Double-click an attribute in the Models view.
2. Click the Control tab.
3. Select the Mandatory check box.
4. Select when a constraint should be checked (Flow/Static).

ORACLE

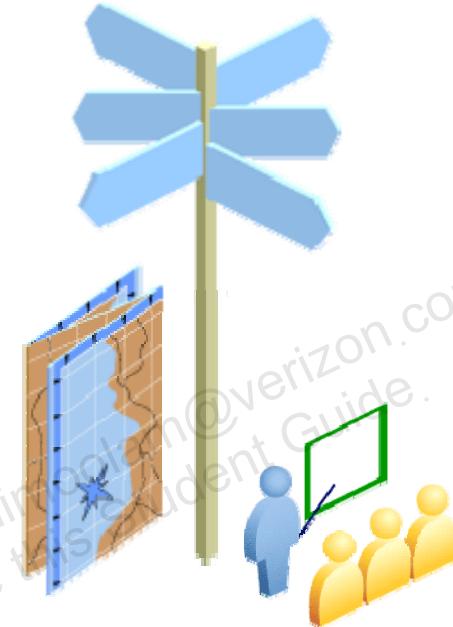
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a mandatory (not null) attribute, perform the following:

1. Double-click the attribute in the Models view.
2. Click the Control tab.
3. Select the Mandatory check box. This specifies that the column should always have a value, but does not actually enforce it.
4. To enforce the mandatory status of a column, select one or both of the Flow and/or Static check boxes. These act as default settings for future mappings. For example, if you select the Flow check box, any mappings that populate this column automatically enable the column for checking during flow control. Definitions:
 - **Flow Control** specifies whether data is checked for correctness before integration. This option enforces the constraints that you defined in ODI, such as uniqueness, reference, and conditional constraints. Only single-technology IKMs have this option.
 - **Static Control** specifies whether data is checked for correctness after being loaded. This option performs almost the same checks as the "flow control" option, but on the data that has been integrated.

Agenda

- Organizing Models
- Creating Datastores
- Constraints in ODI
- **Creating Keys and References**
- Creating Conditions
- Overview
- Exploring Your Data
- Constructing Business Rules



ORACLE

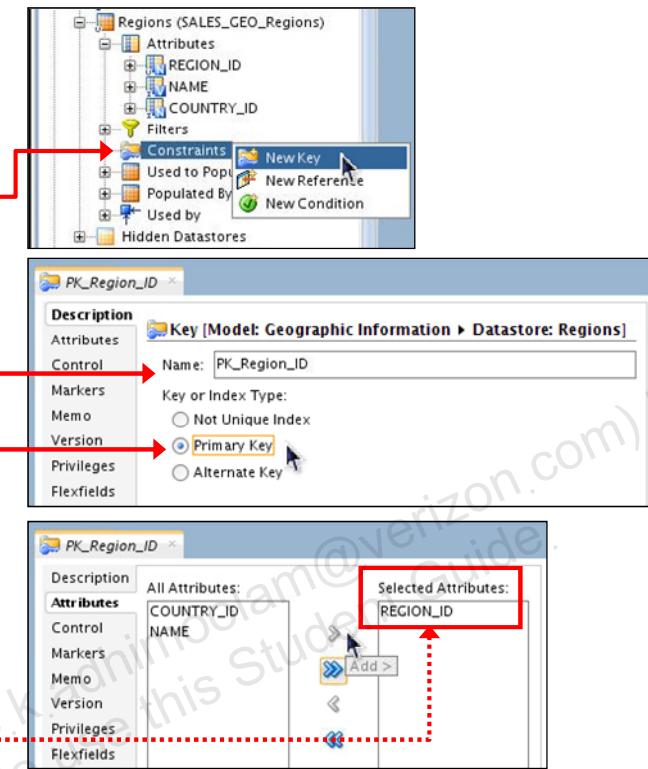
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You have examined manually creating datastores and their columns, and adding constraints to datastores.

Now you look at creating keys and references.

Creating a Key

1. Select the Constraints node under the datastore.
2. Right-click and select New Key.
3. Enter the name.
4. Select the key or index type option.
5. Click the Attributes tab.
6. Add or remove columns from the key.
 - Any number of columns can be selected.



ORACLE

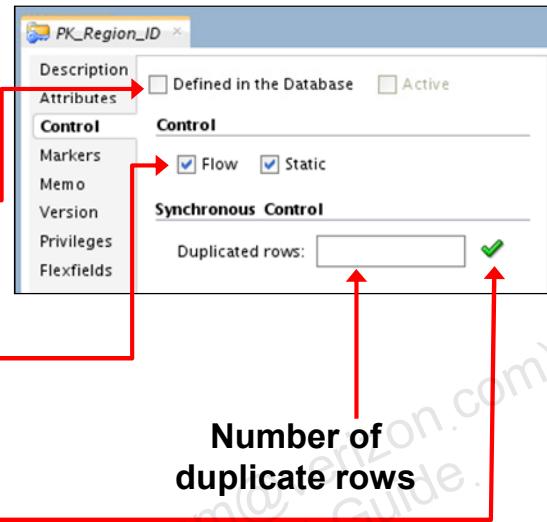
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a key, perform the following:

1. Select the Constraints node under the datastore.
2. Right-click and select New Key.
3. Name the key. This name is used only in ODI. The key will not be created in the database.
4. Then, select the type of key or index. "Not Unique Index" corresponds to the indexes. Both the primary key and alternate key can be used to enforce uniqueness.
5. Click the Attributes tab.
6. Add the columns that form part of this key by using the > and >> arrows. The combination of columns in the list on the right must contain unique values. You can include any number of columns in the key, not just one.

Checking a Key

1. Click the Control tab.
2. Select whether the key is defined in the database, and is active.
3. Select when the constraint must be checked (Flow/Static).
4. Click the Check button to perform a synchronous check of the key.



ORACLE

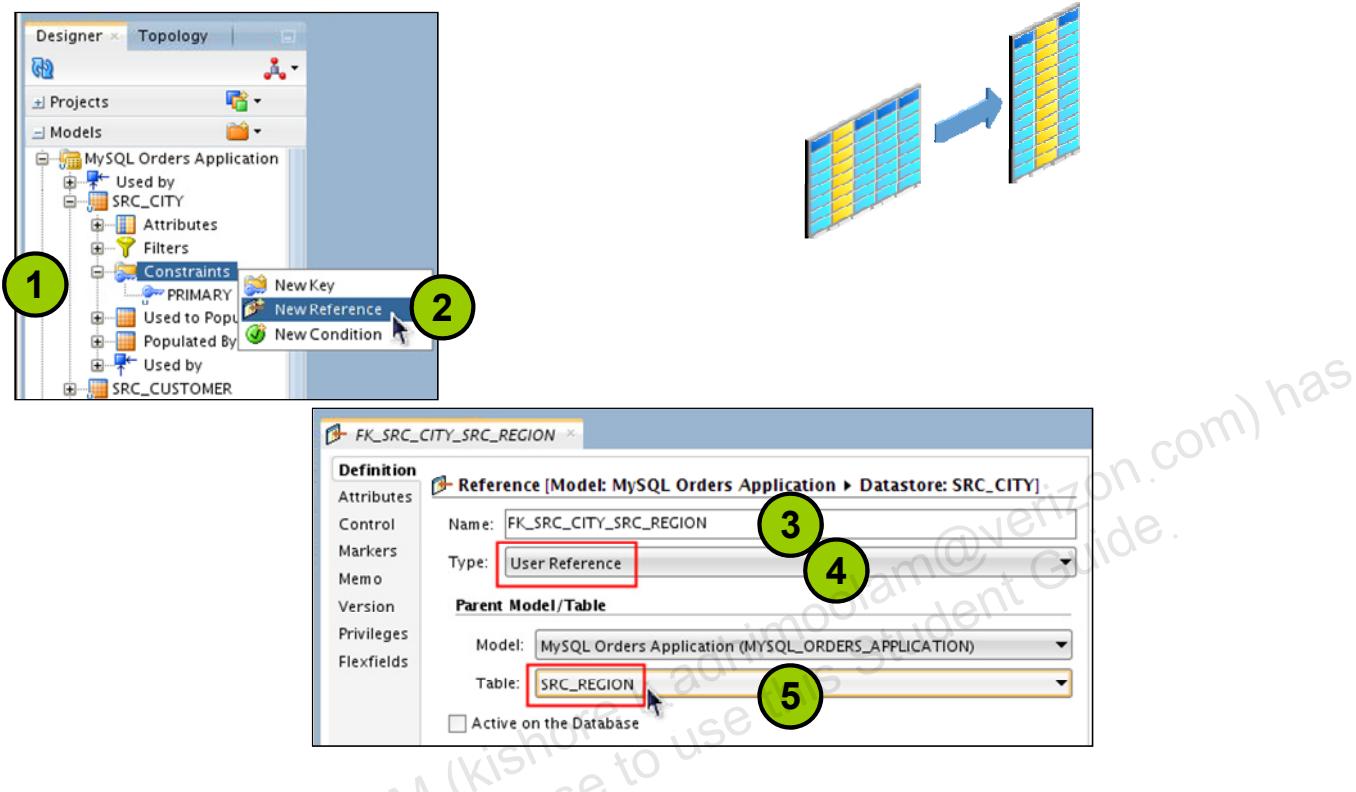
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

As with the mandatory columns, ODI has not yet been asked to check your constraint. To do this, perform the following:

1. Click the Control tab.
2. The “Defined in the Database” check box is purely informational. When a key is reverse-engineered from a database, it is set to true. Changing it has no meaning to ODI, but you may want to do so if a corresponding key in the database cannot be reverse-engineered for some reason. The Active flag is very similar: It indicates whether the key is enabled in the database.
3. To enforce the key, select the Flow or Static check boxes, or both. These behave the same as for mandatory columns; they are mere defaults for future mappings.
4. You can also click the Check button to perform a synchronous check on the column. This tells you the number of rows that violate the constraint. Use this button whenever possible to check whether your key is correct. For example, if the check says that half of your rows are duplicates, a mistake has probably occurred somewhere.

Note: Synchronous checks are available only on SQL-based systems. For instance, you cannot perform a synchronous check on a file-based datastore. Also, when compared to a normal static check, the result of a synchronous check is not saved anywhere.

Creating a Reference



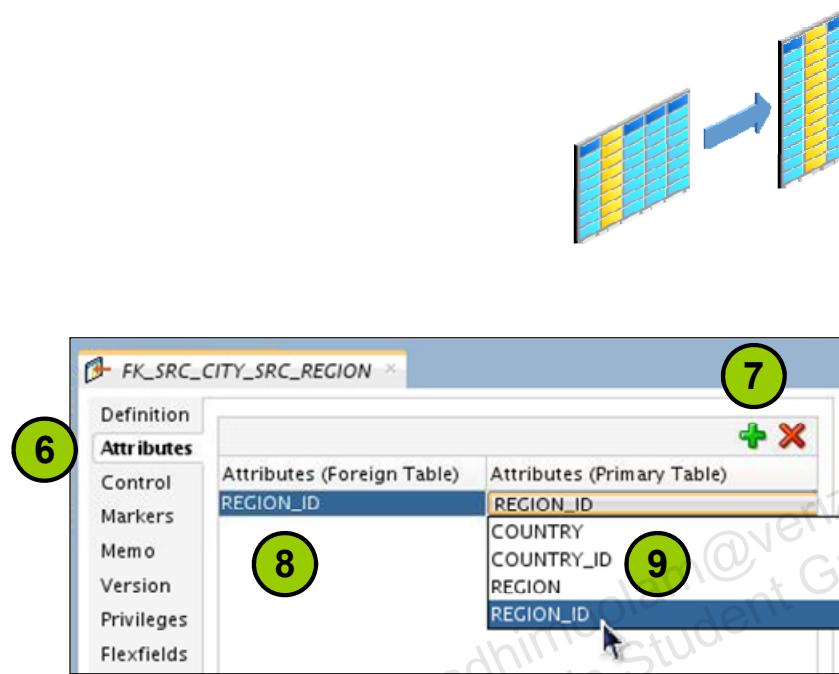
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Creating a reference is not similar to creating a key. Perform the following:

1. Select the Constraints node for the datastore you wish to add a constraint to.
2. Right-click and select New Reference.
3. Provide a name for the reference. The names can be auto-generated based on tables.
4. Select a reference type. The types are:
 - **Database Reference:** Similar to the “Defined in the Database” option for keys and is, therefore, not useful for this purpose
 - **User Reference:** An equality between a column in this datastore and a column in some other datastore, possibly in another model
 - **Complex User Reference:** An expression that relates two columns together.
5. Select the “parent” model and the datastore that this column relates to. For example, if this column is a city ID in an address table, the parent model and datastore would be a table of cities. The column in the cities table would be specified on the Columns tab. If you want to manually specify a specific physical table to relate to, set the model and table to <undefined>, and set the parameters enabling access to the physical table in the External Table fields group.

Creating a Simple Reference



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

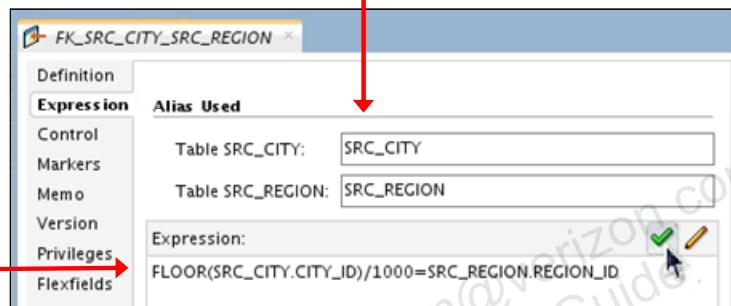
If you want to create a Simple User Reference, perform the following:

6. Click the Attributes tab to specify the columns that must be equal.
7. Click the Add icon (green plus) to add a new column pair.
8. The list of columns on the left contains the columns in this datastore. Select a column to include it in the reference.
9. Select a corresponding column from the Primary Key table on the right.

You can repeat these steps as necessary to build a reference that uses multiple attributes.

Creating a Complex Reference

1. Select Complex User Reference (in the Definitions tab).
2. Click the Expression tab.
3. Set or accept the aliases for the tables.
4. Code the expression:
 - Prefix with the tables' aliases.
 - Use Expression Editor (pencil).
 - Check syntax (green check mark).



ORACLE

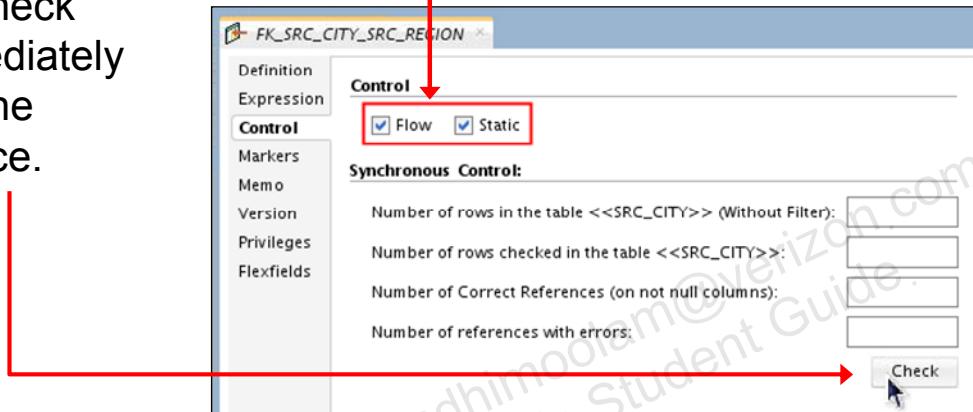
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

If you select Complex User Reference back in the Definition tab, the Attributes tab is replaced by the Expression tab.

You define an alias for the primary table that the reference will refer to. This enables you to write an unambiguous and concise expression. Then, you add a SQL expression that links the two tables. You should use the table aliases to prefix the names of all the columns. The best practice is to use the Expression Editor (pencil icon) to build the right expression. You can drag column names into the expression.

Checking a Reference

1. Click the Control tab.
2. Select when the constraint should be checked (Flow/Static).
3. Click Check to immediately check the reference.



ORACLE

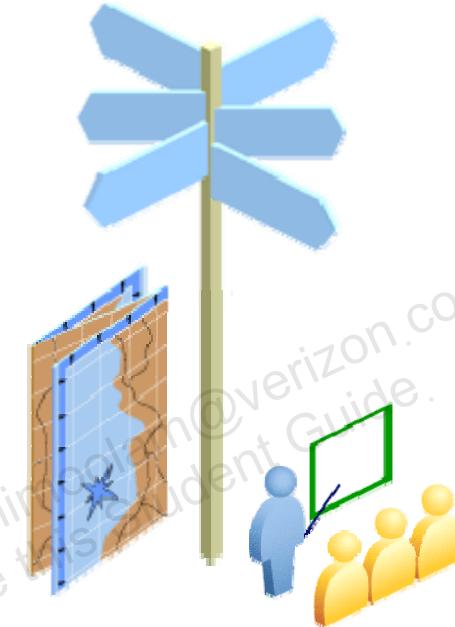
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Finally, as for the other types of constraints, click the Control tab to specify when to check this reference. Again, you can use Check to perform a synchronous check. This tells you the number of rows in the datastore, the number of rows for which the reference is satisfied, and the number of rows that fail the reference. However, this is not possible for a heterogeneous reference.

Note: Immediate check reference is not possible for heterogeneous references.

Agenda

- Organizing Models
- Creating Datastores
- Constraints in ODI
- Creating Keys and References
- **Creating Conditions**
- Overview
- Exploring Your Data
- Constructing Business Rules



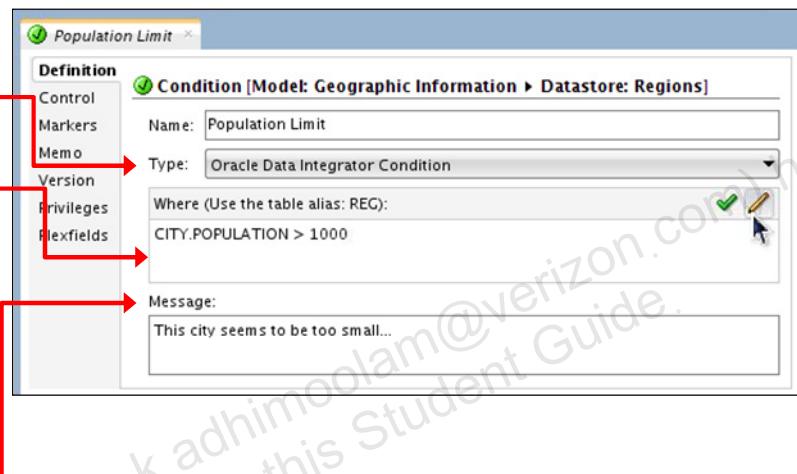
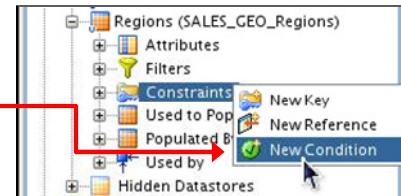
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The last type of constraints to create is conditions. These are simply SQL expressions that enforce certain rules based only on the columns within the datastore.

Creating a Condition

1. Right-click the Constraints node and select New Condition.
2. Enter the name.
3. Select the Oracle Data Integrator Condition type.
4. Edit the condition clause.
 - Prefix with the tables' aliases.
 - Use the Expression Editor.
5. Enter the error message for the condition.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a condition, perform the following:

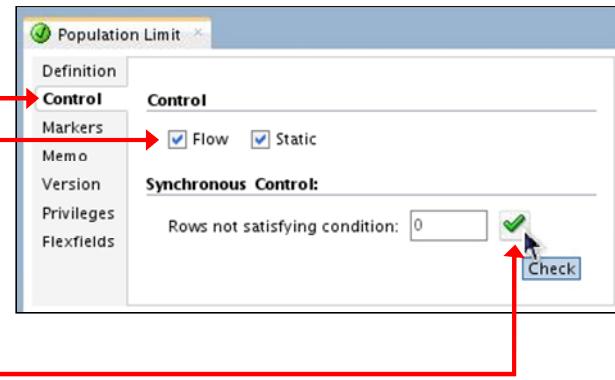
1. Right-click the Constraints node under a datastore and select New Condition.
2. Select a name that will appear only in ODI.
3. Select the Oracle Data Integrator Condition type from the Type drop-down list.
4. Add the expression that must evaluate to true. For example, here you specify that the population of the city must be greater than 1,000. It is recommended that you use the Expression Editor to do this.
5. Lastly, specify a message that is written to the execution log if the condition fails. A good practice is to assign a message number in the text and suggested action to remedy your messages, such as:

MY_MSG_100: This city seems to be too small.

See MY_DOCS page 123 for minimum sizes.

Checking a Condition

1. Click the Control tab.
2. Select when the constraint should be checked (Flow/Static).
3. Click the Check to perform a synchronous check of the condition.



To specify when to check the condition, you select the Flow and/or Static check boxes on the Control tab. Use the Check button to display the number of rows that do not satisfy the condition.

Agenda

- Organizing Models
- Creating Datastores
- Constraints in ODI
- Creating Keys and References
- Creating Conditions
- **Overview**
 - Exploring
 - Auditing
- Exploring Your Data
- Constructing Business Rules



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The following slides provide an overview of when and why you should use the data exploration and auditing process.

Audit/Explore: When and Why

- Why explore and audit your data?
 - To understand the data model
 - To evaluate the quality of the data
- When?
 - If you are using undocumented models
 - Unimplemented business rules
 - Totally unknown data



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

When and why should you begin exploring and auditing your data? You should explore and audit your data to:

- Improve your understanding of the data model
- Get a clear picture of the level of data quality after you have audited every table

You should explore and audit your data:

- In a situation where you face a set of undocumented models. This may come from old proprietary software or in-house systems.
- When your organization has business rules that have never been enforced. Before enforcing them, you should verify the extent to which the existing data respects these rules.
- In situations where you must attempt to work with data that is entirely unknown to you

Audit/Explore Process: Overview

1. Explore the data.
 - Contents of datastores
 - Distribution of values
2. Construct business rules based on your data exploration.
 - Add constraints.
 - Test the constraints.
3. Audit the data against your business rules.
 - Verify data quality in batch.

ORACLE®

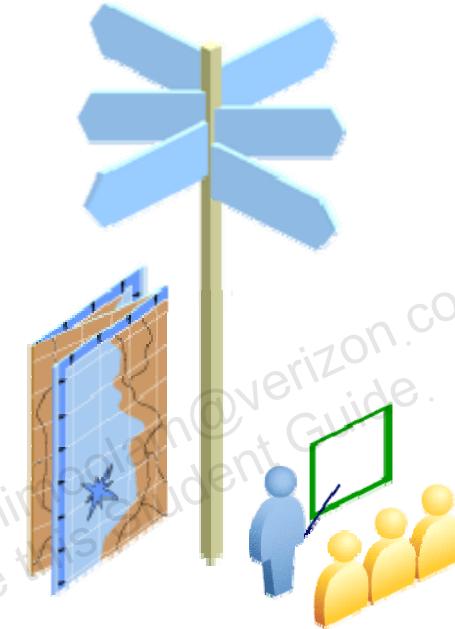
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The basic process is as follows:

1. You explore your data in ODI. You learn to view the contents of a datastore and analyze the data to see the distribution of values.
2. You learn to construct a coherent set of business rules appropriate for your data. For example, if columns contain only certain values, you may want to enforce that pattern on future data. After adding constraints, you learn to test them.
3. In the final phase, you audit the quality of your data against the new rules that you have set. You also learn to automate this process.

Agenda

- Organizing Models
- Creating Datastores
- Constraints in ODI
- Creating Keys and References
- Creating Conditions
- Overview
- **Exploring Your Data**
- Constructing Business Rules



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You begin by exploring and analyzing your data. Three techniques enable you to explore and analyze data. These techniques are covered in the following slides.

Displaying the Contents of a Datastore

1. Select the data store in the Models view.
2. Right-click, select Data.
3. In the Data window:
 - Navigate.
 - View/Edit.

The screenshot shows the Oracle Data Integrator interface. On the left, the 'Models' view is open, displaying a tree structure of databases and applications. A context menu is open over the 'SRC_CUSTOMER' node under the 'MySQL Orders Application'. The menu options include Open, View, New, Duplicate Selection, Delete (disabled), Cut, Copy, Hide, Data..., and View Data... (highlighted with a blue arrow). Below the models view, the 'Data: SRC_CUSTOMER' window is displayed. It contains a grid of customer data with columns: CUSTID, DEAR, LAST_NAME, FIRST_NAME, ADDRESS, CITY_ID, PHONE, AGE, and S. The grid has 7 rows of data. Row 2 is currently selected, indicated by a blue highlight. The Oracle logo is visible at the bottom right of the application window.

CUSTID	DEAR	LAST_NAME	FIRST_NAME	ADDRESS	CITY_ID	PHONE	AGE	S
1	101	O Brendt	Paul	10 Jasper Blvd.	107	(212) 555 2146	19	
2	102	O McCarthy	Robin	27 Pasadena Drive	11	(214) 555 3075	29	
3	103	O Travis	Peter	7835 Hartford Drive	12	(510) 555 4448	34	
4	104	O Larson	Joe	87 Carmel Blvd.	13	(213) 555 5095	45	
5	105	O Goldschmidt	Tony	91 Torre drive	14	(619) 555 6529	55	
6	106	O Baker	William	2890 Grant Avenue	15	(312) 555 7040	64	
7	107	O Swenson	Jack	64 Imagination Drive	19	(202) 555 8125	74	

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

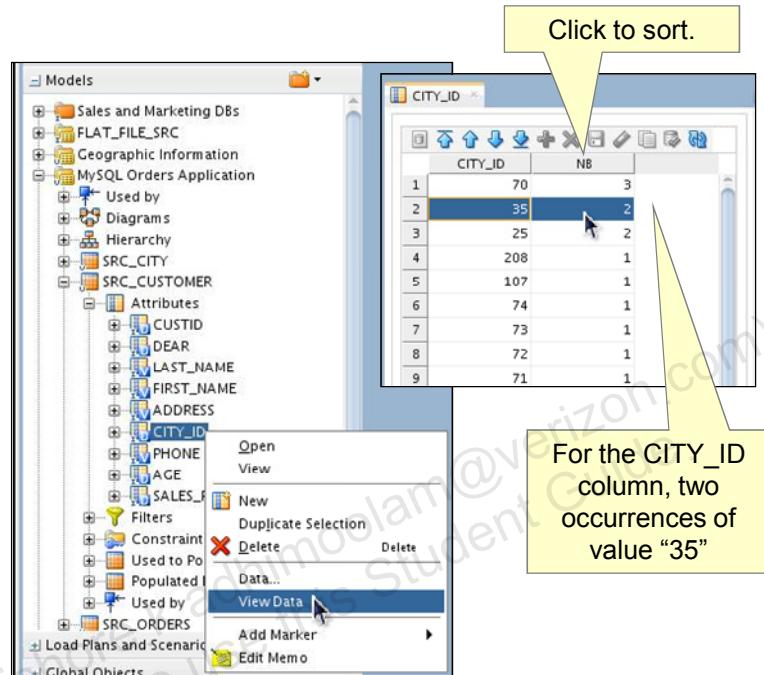
The simplest technique is to view all data in a datastore in raw format. To do this, perform the following:

1. Select the datastore in the Models view.
2. Right-click and select Data from the context menu.
3. The Data window appears.
4. You can use the arrow buttons to move around different rows.
5. To edit the value in a cell, click it.

Note: When you select Data from the context menu, the resulting grid is editable. When you select View Data, the grid is noneditable.

Viewing the Distribution of Values

1. Select the column in the Models view.
2. Right-click, select Data.
3. In the Data window:
 - Values
 - Number of occurrences (NB)



ORACLE®

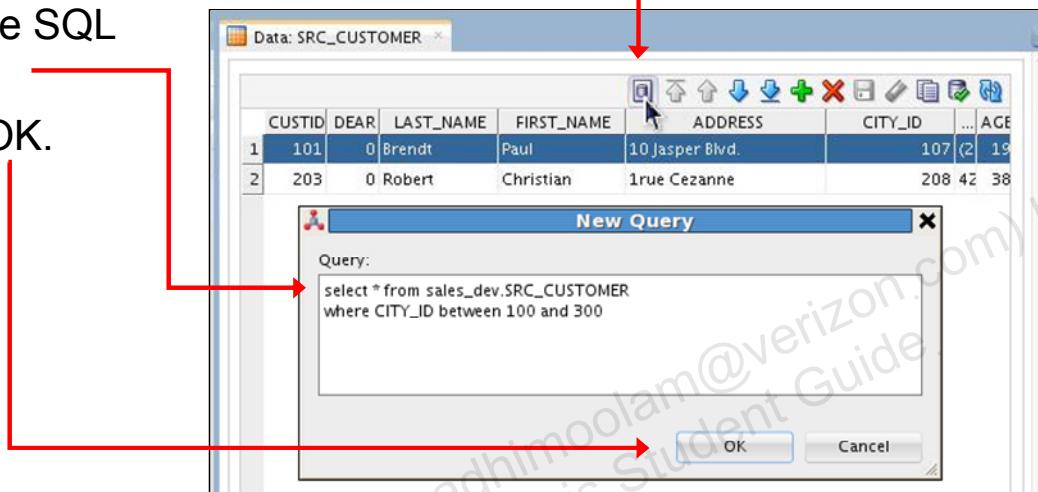
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In many cases, it is useful to see the types of values that appear in a column. To do this, perform the following:

1. Right-click the name of the column in the Models view.
2. Select Data (editable) or View Data (noneditable) from the context menu.
3. The Data window is slightly different this time. There are always two columns. The first column shows the different values and the second shows the number of times each value appears in this column. That is, for the CITY_ID column, if three rows have the value "70" in this column, there is a row here with the values "70" and "3." You can click the column headings to sort by the value or by the number of occurrences.

Analyzing the Contents of a Datastore

1. Open the data window for a datastore, as before.
2. Click the New Query icon.
3. Edit the SQL query.
4. Click OK.



Use any functions or features from the database engine.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Sometimes you might need to use more advanced queries to analyze your data. In this case, open the Data window for a datastore as before. Then, click the SQL icon. This enables you to enter an arbitrary SQL query directly. Click OK to launch it. You can use any database functions or features supported by the database engine that hosts the model.

Agenda

- Organizing Models
- Creating Datastores
- Constraints in ODI
- Creating Keys and References
- Creating Conditions
- Overview
- Exploring Your Data
- **Constructing Business Rules**



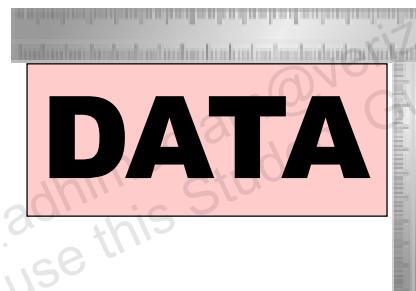
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You have analyzed your data. Now you construct a set of business rules to match the data in your model.

Defining Business Rules in ODI

- There are two ways to define business rules in ODI to maintain data quality:
 - Automatic retrieval with other metadata
 - Obtained by reverse-engineering schema
 - Addition by designers manually
 - User-defined rules
- Rules can be deduced by data analysis and profiling.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ODI enables application designers and business analysts to define business rules that target data quality maintenance. There are two ways to do this:

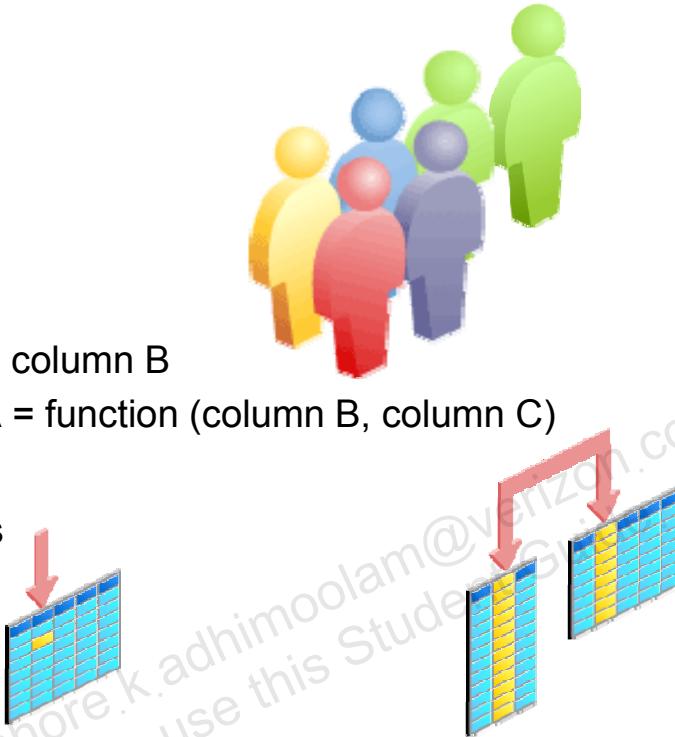
- First, by automatically reverse-engineering metadata from the source application data. ODI detects existing data quality rules defined at the database level. For example, it turns foreign keys into references.
- It also enables application designers to add additional, user-defined rules in the repository.

The rules can generally be reconstructed by data analysis and profiling. As you become familiar with the data, you can deduce the business rules that apply to them.

Refer to the lesson titled “ODI Mapping Concepts” for more details about creating business rules.

From Business Rules to Constraints

- Deduplication rules:
 - Primary keys
 - Alternate keys
 - Unique indexes
- Reference rules:
 - Simple: column A = column B
 - Complex: column A = function (column B, column C)
- Validation rules:
 - Mandatory columns
 - Conditions



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The business rules for data quality can include:

- **Deduplication rules (unique keys):**
 - These rules can be used to identify and isolate inconsistent data sets.
 - Examples: “Customers with the same email address” or “Products having the same item code and item family code”
 - These rules are defined in ODI as primary keys, alternate keys, or unique indexes.
- **Simple and complex reference rules:**
 - Examples: “Customers that are linked to nonexistent sales reps” or “Orders that are linked to customers that are marked as Invalid”
 - These rules are defined in ODI as references.
- **Validation rules:**
 - These rules enforce consistency at the record level.
 - Examples: “The list of customers with an empty zip code” or “The list of web prospect responses with an invalid email address”
 - These rules are defined in ODI as conditions or mandatory columns.

Deducing Constraints from Data Analysis

- If a column has no null values, consider a mandatory constraint.
- If a set of columns never contains a set of values more than once, consider an alternate key.
- If a column contains distinct, not-null, numeric values, consider a primary key with the column flagged as mandatory.
- If a set of columns contains few different values:
 - Consider an index
 - Consider a condition based on these values
- If a table has a group of columns with data matching a key of the same type from another table, consider a reference.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

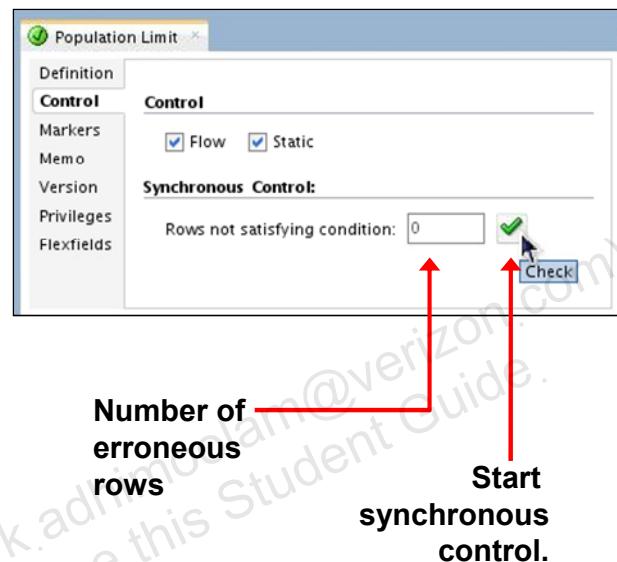
The following are some general principles for creating meaningful constraints from the results of data analysis:

- If a column has no null values, you may want to add a mandatory constraint.
- If every set of values in a set of columns is unique, you may want to add an alternate key.
- If every value in a column is unique and numeric, and there are no null values, you can add a primary key and flag the column as mandatory.
- When a set of columns contains the same values often, you may want to add an index for performance. You can also create a condition that prevents values outside this range from being entered.
- Lastly, consider adding a reference when data in a group of columns in one table matches data in another table. The two types should be the same, of course.

Testing a Constraint

Synchronous data check:

- “Quick” check
- Available on the Control tab of Constraint window for:
 - Keys
 - References
 - Conditions
- Useful to check whether a rule is correct
- Requires a SQL-enabled system



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

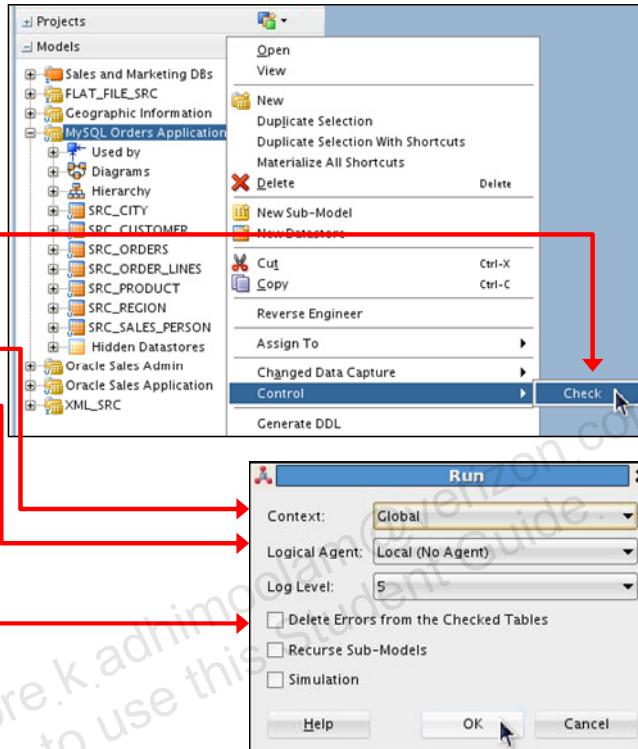
A synchronous option to perform a “quick” check of a constraint after it is created is available on the Control tab of the window for keys, references, and conditions.

By clicking Check, you can perform a synchronous check on the constraint, which tells you the number of rows that violate the constraint. Use this button whenever possible to check whether your rule is correct. For example, if this check returns half of your rows as duplicates, you probably have made a mistake somewhere.

You should remember that synchronous checks query the data server directly to obtain the number of invalid rows. This feature works only with SQL-based systems. Thus, you cannot perform a synchronous check on a file-based datastore. Also note that, unlike a normal static check, the result of a synchronous check is not saved anywhere.

Auditing a Model or Datastore

1. Select the datastore or model in Models view.
2. Right-click and select Control > Check.
3. Select:
 - Context
 - Execution Agent
 - Delete errorsThe session starts.
4. Use Operator to check the audit's progress.



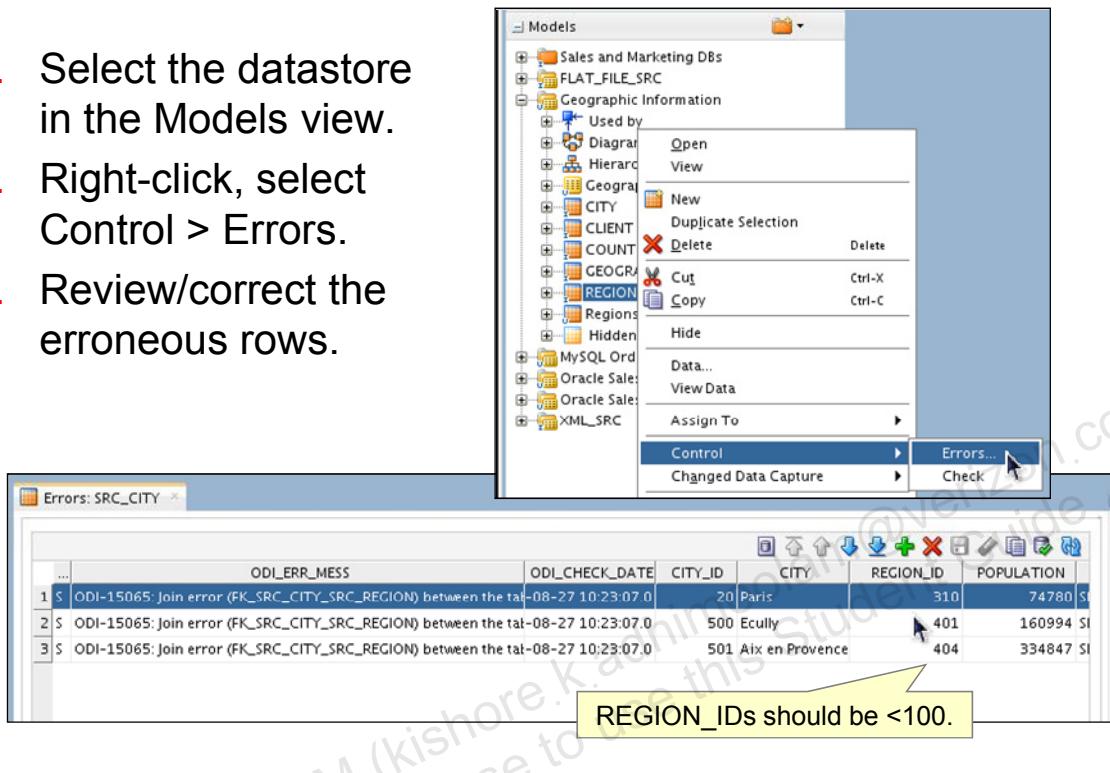
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To audit data in a model or a datastore, perform the following:

1. Select the datastore or model in the Models view.
2. Right-click and select Check from the Control submenu.
3. The audit must be performed by an ODI agent. You also need to specify the context that determines which instance of the data will be checked.
4. The agent now launches the session. You can follow its progress in Operator, if necessary. This will let you know when the audit has finished so that you can review the results. Note that to perform such a data check, a Check Knowledge Module must be defined in the data model that you are working with.

Reviewing Erroneous Records

1. Select the datastore in the Models view.
2. Right-click, select Control > Errors.
3. Review/correct the erroneous rows.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The process to view bad records that were picked up in the audit is similar to starting the audit. Perform the following:

1. Select the target datastore from the Models view.
2. Right-click and select Control > Errors. The erroneous rows are now displayed. For each error, the business rule that was violated is also shown.
3. You must check the results for each datastore individually.

Quiz

The only way to create datastores in ODI is by reverse-engineering their structure from the database.

- a. True
- b. False



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Explanation: The most common way to create datastores in ODI is by reverse-engineering their structure from the database. However, you can do so directly in the model.

Quiz

In ODI, references can be represented by (select all that apply):

- a. Simple equalities between columns in tables
- b. Complex relationships between columns in tables
- c. Expressions that must evaluate to true or false
- d. All of the above



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a, b

Explanation: References in ODI can be either simple equalities between columns in tables, or complex expressions that define a relationship between columns in tables.

Answers a and b compare across tables. Why not answer c? Answer c is more like a condition *within* a table.

Summary

In this lesson, you should have learned how to:

- Organize ODI models
- Create datastores in ODI models
- Create constraints in ODI:
 - Create mandatory columns
 - Create keys and references in ODI models
 - Create and check conditions
- Explore data
- Audit an ODI model or datastore



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

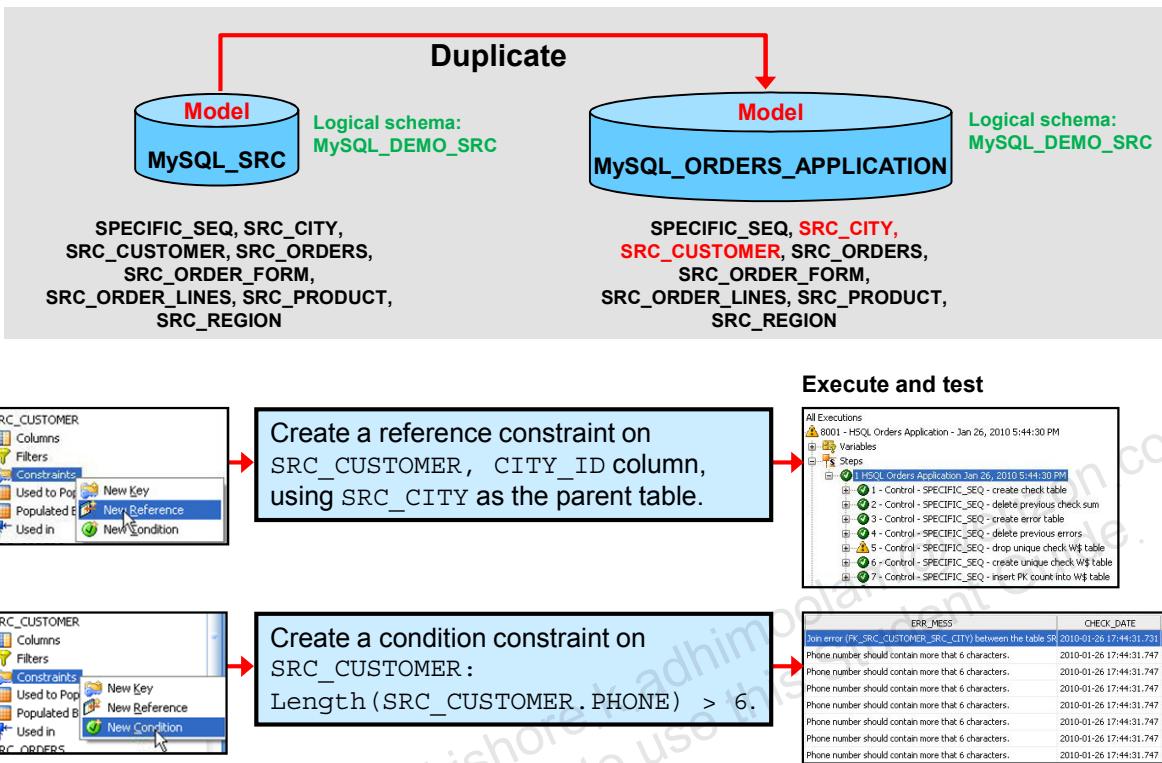
Checklist of Practice Activities

- ✓ 1-1: Logging In and Help Overview
- ✓ 2-1: Creating and Connecting to ODI Master and Work Repositories
- ✓ 3-1: Configuring Standalone Agents Using the Common Admin Model
- ✓ 4-1: Working with Topology
- ✓ 5-1: Setting Up a New ODI Project
- ✓ 6-1: Creating Models by Reverse-Engineering
- 7-1: **Checking Data Quality in the Model**
- 8-1: Creating ODI Mapping: Simple Transformations
- 9-1: Creating ODI Mapping: Complex Transformations
- 9-2: Creating ODI Mapping: Implementing Lookup
- 10-1: Creating ODI Mapping: Exporting a Flat File to a Relational Table
- 11-1: Using Native Sequences with ODI Mapping
- 11-2: Using Temporary Indexes
- 11-3: Using Sets with ODI Mapping
- 12-1: Creating and Using Reusable Mappings
- 12-2: Developing a New Knowledge Module
- 13-1: Creating an ODI Procedure
- 14-1: Creating an ODI Package
- 14-2: Using ODI Packages with Variables and User Functions
- 15-1: Debugging Mappings
- 16-1: Creating and Scheduling Scenarios
- 17-1: Using Load Plans
- 18-1: Enforcing Data Quality with ODI Mappings
- 19-1: Implementing Changed Data Capture
- 20-1: Setting Up ODI Security
- 20-2: Integration with Enterprise Manager and Using ODI Console

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 7-1: Checking Data Quality in the Model



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

After the models are defined, you need to check the quality of the data in these models. In this practice, you check the quality of data in the models and define constraints on models for the given sample application.

First you create a new model, MySQL_ORDERS_APPLICATION, as a duplicate of the model MySQL_SRC.

You then create a referential constraint on the SRC_CUSTOMER table's CITY_ID column, using SRC_CITY as the parent table.

Next, you create a condition constraint on the SRC_CUSTOMER table:

Length(SRC_CUSTOMER.PHONE) > 6

ODI Mapping Concepts

8

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

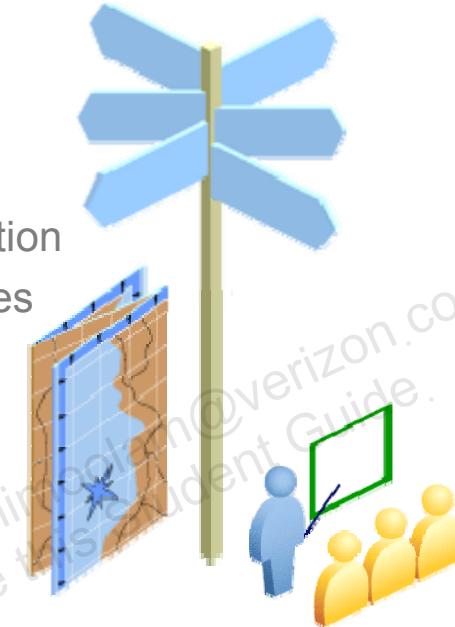
- Describe the concept of Oracle Data Integrator (ODI) Mapping
- Describe the concept of expressions, joins, and filters
- Describe the process of implementing business rules
- Describe the concepts of staging area and execution location
- Use Knowledge Modules with ODI Mapping
- Create and execute a basic ODI Mapping
- Use lookups



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Agenda

- **ODI Mappings**
 - Key Concepts
- Expressions, Join, Filter, Lookup, Sets, and Others
- Behind the Rules
- Staging Area and Execution Location
- Understanding Knowledge Modules
- Mappings: Overview



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This section provides a detailed description of some of the concepts involved in mappings and in using mappings to implement business rules.

What Is a Mapping?

- A mapping is an ODI object that loads one or more target datastores with data from one or more source datastores.
- A mapping defines the loading and integration strategies from sources to targets based on the selection of Knowledge Modules.
- A mapping implements business rules as expressions, joins, filters, and constraints.



ORACLE

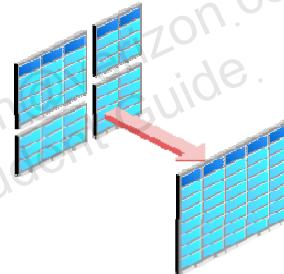
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A mapping is an object in ODI that serves primarily to populate one or more datastores, which is known as the target. The data comes from one or more source datastores, which can be located on different servers from each other, and from the target. The business rules that govern this data transfer are expressed directly in the mapping as expressions, joins, filters, and constraints. All these concepts are explored in detail in this lesson.

Business Rules for Mappings

Business rules are:

- Implemented in mappings as:
 - Expressions
 - Filters
 - Joins
 - Constraints
 - Lookups
 - Others
- Implemented within Designer in:
 - Data models
 - Mappings
- Stored in the Work Repository



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Business rules are visible in mappings in five ways:

- Expressions describe the transformation of data from source datastores to a target datastore. Most standard math/character formulas are acceptable.
- Filters specify the incoming rows that you want to treat.
- Joins link several source datastores.
- Constraints ensure that both incoming and outgoing data conform to business requirements.
- Lookups use a source as the driving table and a lookup datastore or mapping.
- Other includes Set (Union, Intersect, Minus, and so on), Aggregate (Min, Max, Avg, Sum, and so on), Pivot, and others on the Component Palette of ODI Studio.

Filters, joins, and constraints can be attached permanently to models in Designer. Expressions exist only within the context of a mapping.

Where Are the Rules Defined?

The business rules implemented in a mapping are located as follows:

Expressions	In the mapping diagram. Each attribute (column) of the target has at most one expression.
Filters	In the mapping diagram. Filters may be inherited from the datastore definition in the model or created manually.
Joins	In the mapping diagram. Joins may be inherited from the model definition or created manually.
Constraints	In target datastore definition in the model. Constraints to be enforced are selected on the mapping's Control tab.
Lookups	In the mapping editor. Create lookups by using a source as the driving table and a lookup datastore or mapping.
Others	In the Components Palette of the mapping editor.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

As mentioned previously, business rules are implemented as expressions, filters, joins, constraints, and lookups. You now see where these are found in the mapping window. Expressions, filters, and joins are found in the mapping diagram. Because mappings are tied to the target columns directly, you see them by clicking the names of columns in the target datastore.

Filters can be created for the purposes of a mapping or attached permanently to models.

Joins appear as lines in the mapping diagram. You can create them directly in the mapping diagram, or define references between models. These references are automatically converted into joins.

Constraints exist only on models. You can define constraints on source datastores through the Models view. These become static controls. You can also define constraints on target datastores in the same way. These constraints become flow controls and can be activated through the Controls tab.

Use the Properties panels and/or drag-and-drop attributes in the mapping editor to create lookups by using a source as the driving table and a lookup datastore or mapping.

Agenda

- ODI Mappings
- **Expressions, Join, Filter, Lookup, Sets, and Others**
- Behind the Rules
- Staging Area and Execution Location
- Understanding Knowledge Modules
- Mappings: Overview



ORACLE

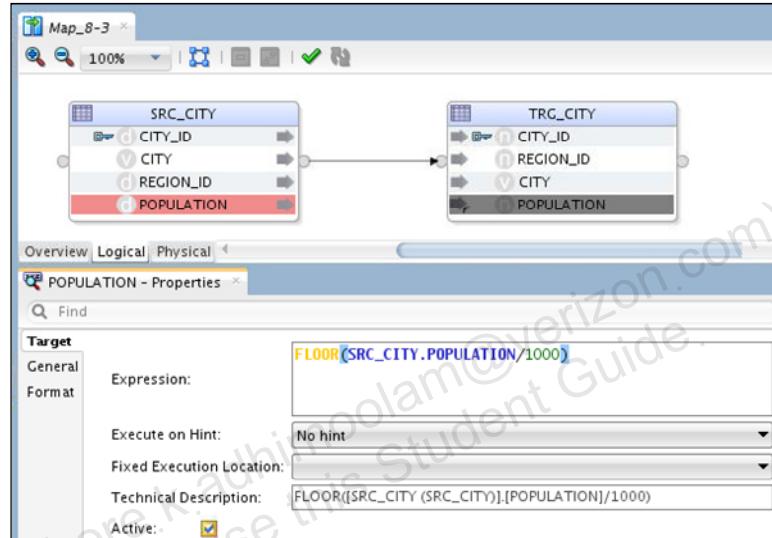
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The next few slides look at the types of business rule implementations.

What Is an Expression?

An expression is:

- A business rule implemented in ODI as a SQL clause
- A transformation rule that maps columns in source datastores onto one of the target datastore columns (attributes)
- Executed by a relational database server at run time



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

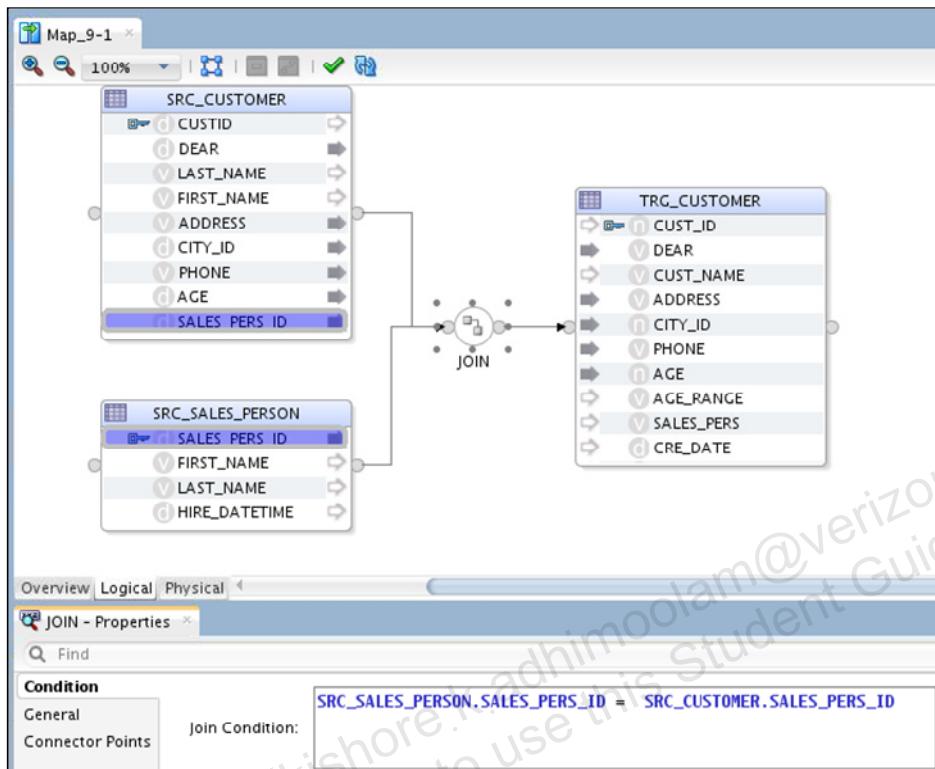
ORACLE

Mappings are the logical and physical organization of your data sources, targets, and the transformations through which the data flows from source to target. You create and manage mappings by using the Mapping Editor, a new feature of ODI 12c.

On the other hand, an expression is a business rule that maps one or more columns from one or more source datastores onto one of the target datastore columns, possibly transforming them in the process. It can be as simple as stating that the Age column in the target datastore represents the Age column in the source datastore, or as complex as calculating aggregate properties of multiple columns in various datastores located on different servers.

An expression can be defined manually by entering code in the expression field or by using the drag-and-drop functionality in the graphical user interface (GUI). ODI also includes an Expression Editor, which helps in building the mapping.

What Is a Join?



ORACLE

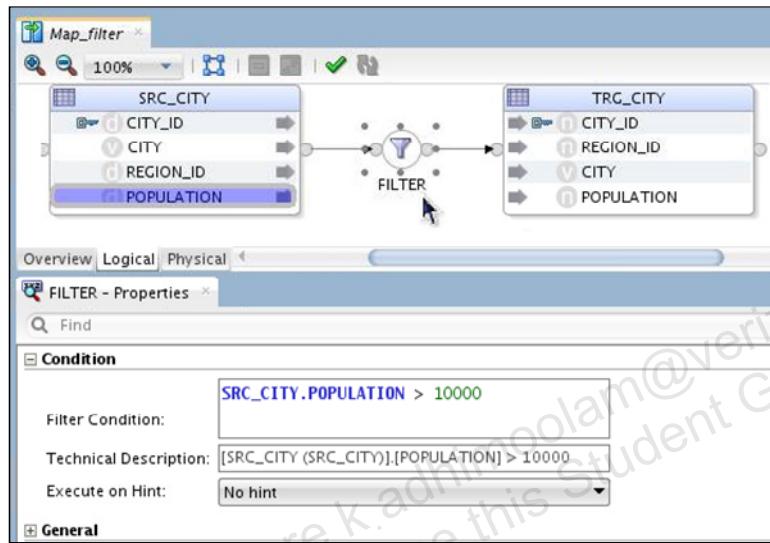
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In relational models, a join operation links individual records in several tables, from any technologies. A mapping uses the same concept of linking source datastores to act as a single source to populate the target. A join is implemented as a clause in SQL, and it defines a relation between the columns of two or more datastores.

For example, you use a join to link the records from the ORDER table to records from the CUSTOMER table. If a CUSTOMER_ID column is in both tables, you can use it to perform a join. Orders with a given CUSTOMER_ID value are linked to the customer having the same CUSTOMER_ID. Thus, you can populate a target table with not only the information about a given order, but also with information about the customer who is making the order (denormalizing the target table).

What Is a Filter?

A filter is a SQL clause applied to the columns of one source datastore. Only records matching this filter are processed by the mapping.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

More precisely, a filter is an expression written in SQL that is attached to one or more columns of one source datastore. The filter imposes a constraint that is applied to each row in the source datastore, and only rows that match continue to be processed by the rest of the mapping. A source datastore can have more than one filter or none at all.

What Is a Lookup?

- Drag a component to the Mapping Editor to create lookups by using a source as the driving table and a lookup datastore or mapping.
- Lookups appear as compact graphical objects in the mapping sources diagram.
- Choose how each lookup is generated:
 - Left Outer Join in the `FROM` clause
 - Expression in the `SELECT` clause
(in-memory lookup with nested loop)
- Benefits of using lookups:
 - Simplifies the design and readability of mappings that use lookups
 - Enables optimized code for execution



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

An expression in the `SELECT` clause is sometimes more efficient on small lookup tables.

New with ODI 12.1.2 patch number 17053768 “Oracle Warehouse Builder to Oracle Data Integrator Migration Utility Patch” are some additional features:

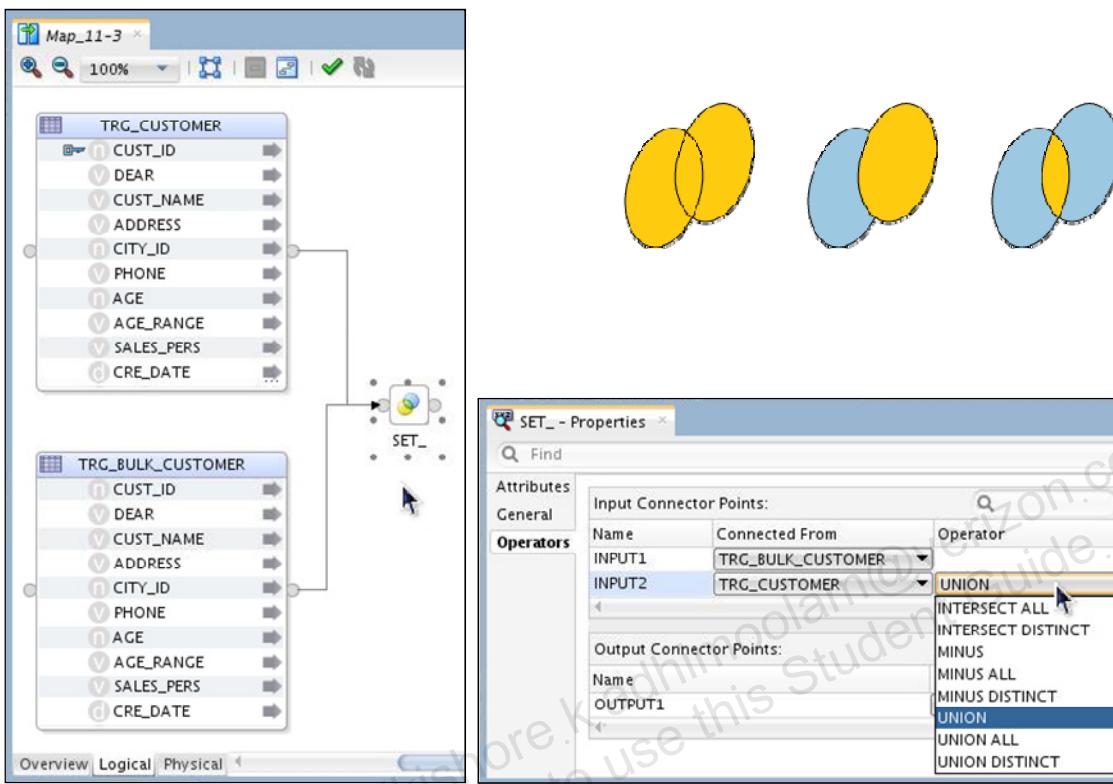
Multiple Match Rows

- The Lookup Type property has been replaced with Multiple Match Rows.
- The Multiple Match Rows property defines which row from the lookup result must be selected as the lookup result if the lookup returns multiple results. Multiple rows are returned when the lookup condition specified matches multiple records.
- You can select one of the following options to specify the action to perform when multiple rows are returned by the lookup operation:
 - Error: Multiple rows cause mapping to fail
 - All Rows (number of result rows may differ from the number of input rows)
 - Select any single row
 - Select first single row
 - Select nth single row

No-Match Rows Property

If there are no matching rows, then use the default values.

What Is a Set?



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A set (not to be confused with a data set) represents the data flow coming from a group of datastores. Several sets can be merged into the mapping target datastore by using set-based operators such as Union and Intersect. The support for sets, as well as the set-based operators supported, depends on the technology of the staging area.

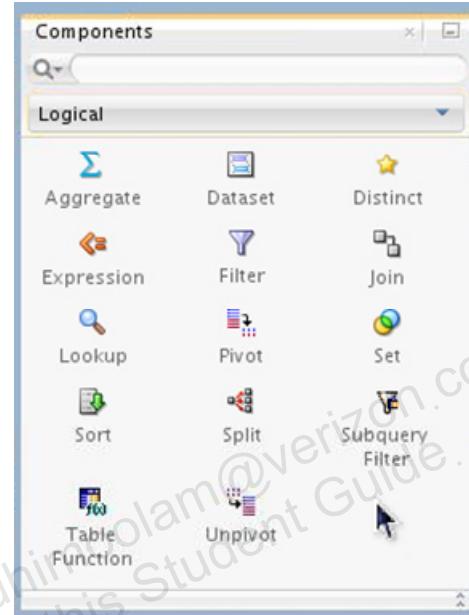
You can add, remove, and order the sets of a mapping and define the operators between them in the Set Properties > Operators panel. Note that the set-based operators are always executed on the staging area.

When designing the integration mapping, the mappings for each set must be consistent. This means that each set must have the same number of target columns mapped. One target is loaded with data coming from several sets. Set-based operators (Union, Union All, Minus, Intersect) are used to merge the different sets into the target datastore.

What Are Some of the Others?

Component Palette:

- Aggregate
- Distinct
- Filter
- Lookup
- Set
- Split
- Table Function
- Dataset
- Expression
- Join
- Pivot
- Sort
- Subquery Filter
- Unpivot



And you can add your own...

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Some of these components have been covered in preceding slides, such as Filter, Join, Lookup, and Dataset.

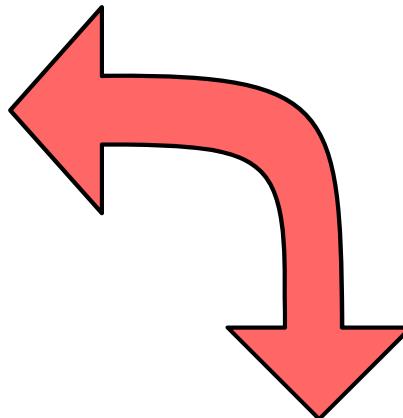
What may not be obvious is where you need an intermediate component in the flow of the mapping. Suppose that you have a source table of employees, you cannot simply map their salaries directly into a single target attribute by saying `SUM(SALARY)` in the expression of the target. ODI Studio *lets you do it*, but it will not work properly. What you need to do is to create a map from the source to a new component called Aggregate, and then map that to a target. Within the Aggregate there can be as many or as few different kinds of aggregations (sum, average, count, min, max, and so on) for as many or as few attributes as you need from that source.

A data set (not to be confused with data in a set) is primarily an 11g-style construct. It can be used to encapsulate one or more joins for simplicity.

Expression is a place to make reusable collections of complicated expressions, like a macro library.

New with Patch: Pivot and Unpivot

YEAR	QUARTER	SALES
2012	Q1	19.8
2012	Q2	28.7
2012	Q3	37.6
2012	Q4	46.5
2013	Q1	55.4
2013	Q2	64.3
2013	Q3	73.2
2013	Q4	82.1



YEAR	Q1_SALES	Q2_SALES	Q3_SALES	Q4_SALES
2013	55.4	64.3	73.2	82.1
2012	19.8	28.7	37.6	46.5

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Many technologies, such as Oracle Database 12c and Microsoft Excel, support a pivot function natively.

To create a mapping using the pivot function:

1. Create a new mapping.
2. Drag a source and target datastore onto the modeling area.
3. Drag a Pivot or Unpivot icon from the Component Palette and drop it in between the source and target datastores.
4. In the Pivot Properties panel, Row Locator tab, indicate the source Row Locator values to look for: 'Q1', 'Q2', 'Q3', and 'Q4'.
5. In the Attributes tab, indicate the target Attribute names: Q1_SALES, Q2_SALES, Q3_SALES and Q4_SALES.
6. Save and Run the mapping.

Agenda

- ODI Mappings
- Expressions, Join, Filter, Lookup, Sets, and Others
- **Behind the Rules**
 - **Transforming the Rules into Code**
- Staging Area and Execution Location
- Understanding Knowledge Modules
- Mappings: Overview



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The next set of slides shows how ODI implements business rules.

How Does ODI Implement Business Rules?

ODI:

- Is based on business rules
 - First, define the business rules.
 - Next, implement those rules in SQL.
- Does not require a proprietary engine
 - Exploits in-house RDBMS engines to perform the required processes
 - Generates native or standard SQL code
- Separates the rules from the processes
 - Business rules are defined on a mapping.
 - Processing instructions are implemented in Knowledge Modules (KMs).

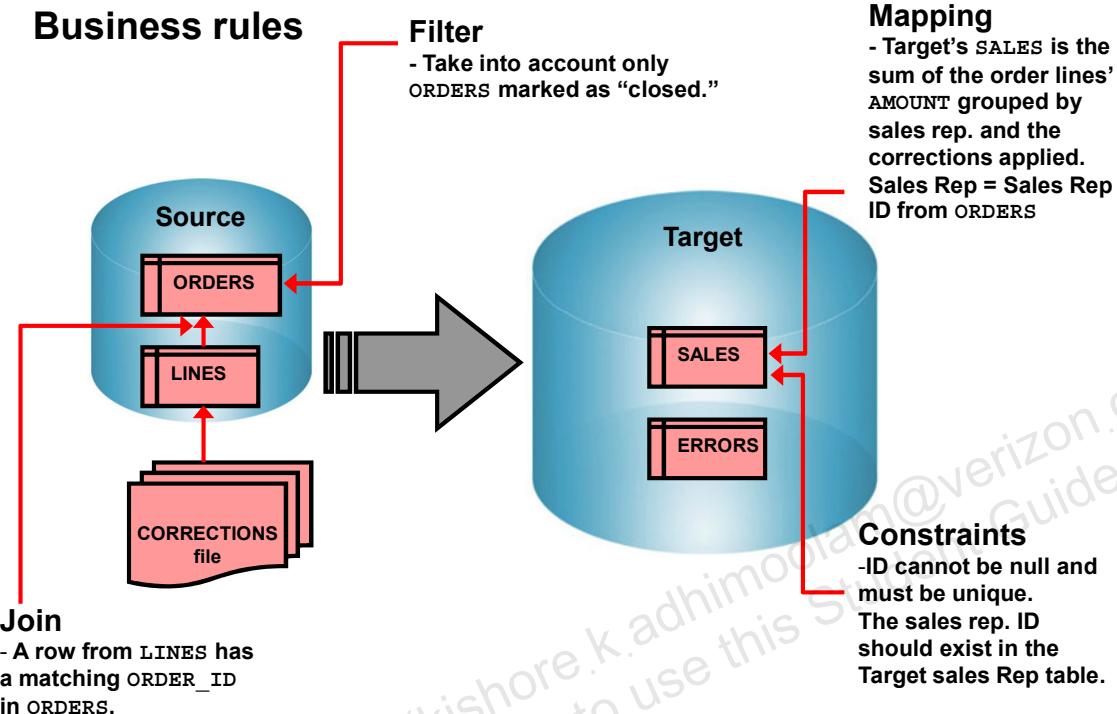


Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

So, what distinguishes ODI from similar products?

- Data integration in ODI is based entirely on business rules. ODI encourages you to first define your business rules in a way that makes sense to you. Then you express the rules in SQL and attach them directly to the relevant objects.
- You can use whatever database servers you have available to perform all the processing. ODI generates either specific code for your database engine or standard SQL code. Using native code or database-specific features may give you superior performance over a standard SQL. But the standard SQL module is more flexible and works on all servers.
- ODI makes a clear distinction between the transformation rules and the processing required to make them happen. You define the business rules on a mapping. Then you select a Knowledge Module, which knows how to implement these rules for a specific technology.

Business Problem



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You now see an example involving the collection of statistics on the sales force. First, you see how to express the scenario as natural language business rules. Then you see the same rules expressed in SQL.

Imagine that you have a source ORDERS database, in some RDBMS or XML datastore, which contains customer orders broken down into individual items. You also have a list of corrections stored in a flat file. You now want to update your SALES database, stored on another RDBMS server, for example, Oracle 12c. This database is used to determine who is the best sales representative for the month.

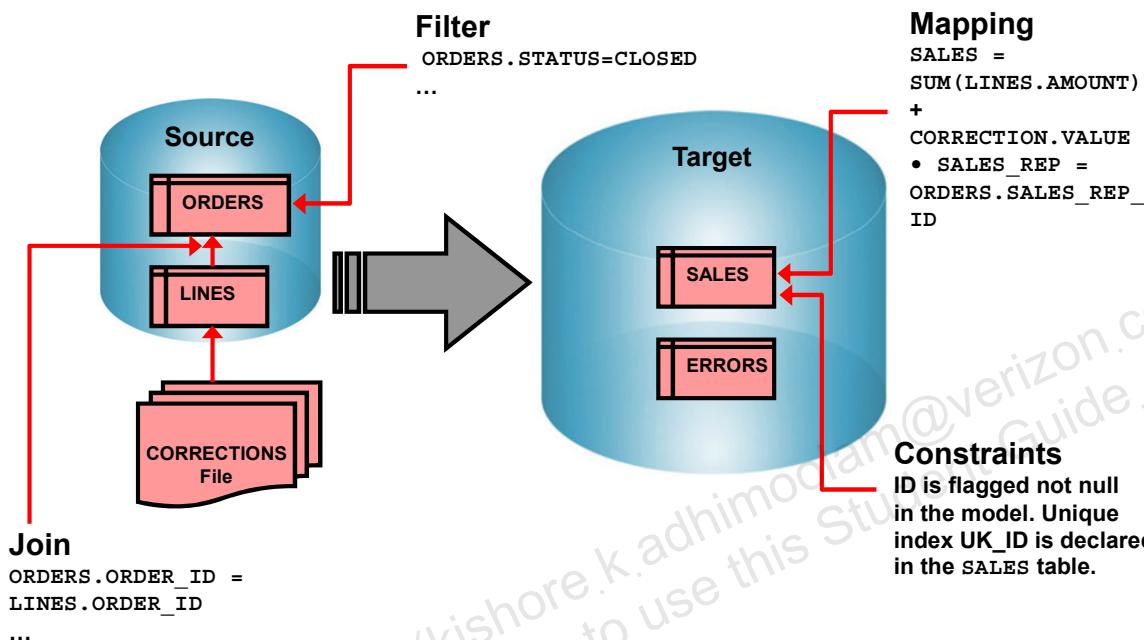
In particular, your business rule is that you want to store the sum of the sales of each sales representative. This is a mapping, because it links source data to target data. You must include the data from the corrections file as well.

Now, you want to use data only from orders marked as "closed," or finalized. This limitation on the source data is a filter in ODI.

Then, you need to link the individual items or "lines" of an individual order to the order. This is to enable you to calculate the total of each order. This linking of source tables is a join in ODI. Lastly, you want only one record per sales representative in the target database. You also need to ensure that each sales figure corresponds to an existing sales representative. Any figures that fail either of these criteria should be rejected and placed in an ERRORS table. This condition on a table is called a constraint.

Implementing the Rules

Business rules implemented in SQL



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Now you see the same scenario as expressed in SQL. The mapping of the orders information onto the sales figures is straightforward: In the target database, the sales figure is the sum of the amounts of individual `LINES` plus the correction value in the source database. The sales representative ID in the target is the same as the ID in the source.

Similarly, the filter is clear: You define that the “status” of the order is CLOSED. ODI filters out any rows that do not match these criteria.

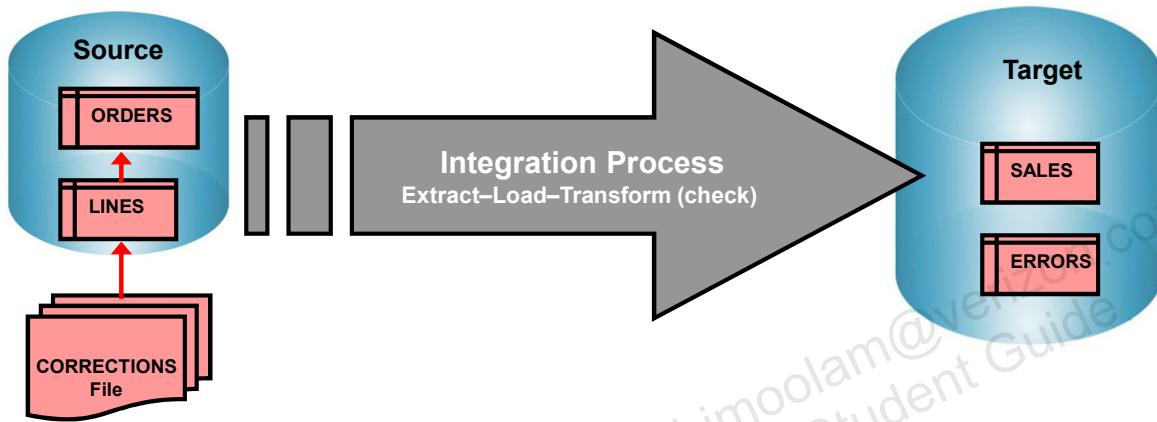
Your join consists of the equality `ORDERS.ORDER_ID = LINES.ORDER_ID`. You do not have to remember any SQL syntax other than creating an expression.

Constraints:

- First, you flag the ID as a “not null” column in the model. If the underlying table is already marked, ODI can detect this constraint.
- Similarly, you define a unique index in the `SALES` table. ODI automatically rejects any faulty rows and places them in an `ERRORS` table.

Integration Process

These business rules form the basis of the integration process.



ORACLE®

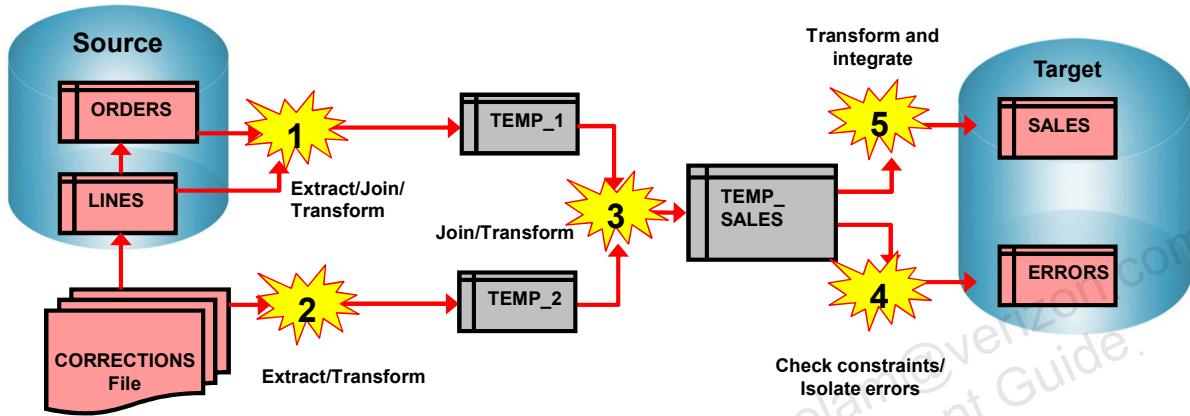
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ODI enables you to implement the integration process from the business rules that you have defined. Note that you add to this process the checks that ensure the quality of the data flow.

In the scenario described in the previous slide, discussions about how to move the data, which infrastructure connects the servers, or how to combine the corrections file into the other data, were deliberately excluded. This is because in ODI, you work at the conceptual ("business rules") level.

Process Details

Detailed sequence of operations for the process



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

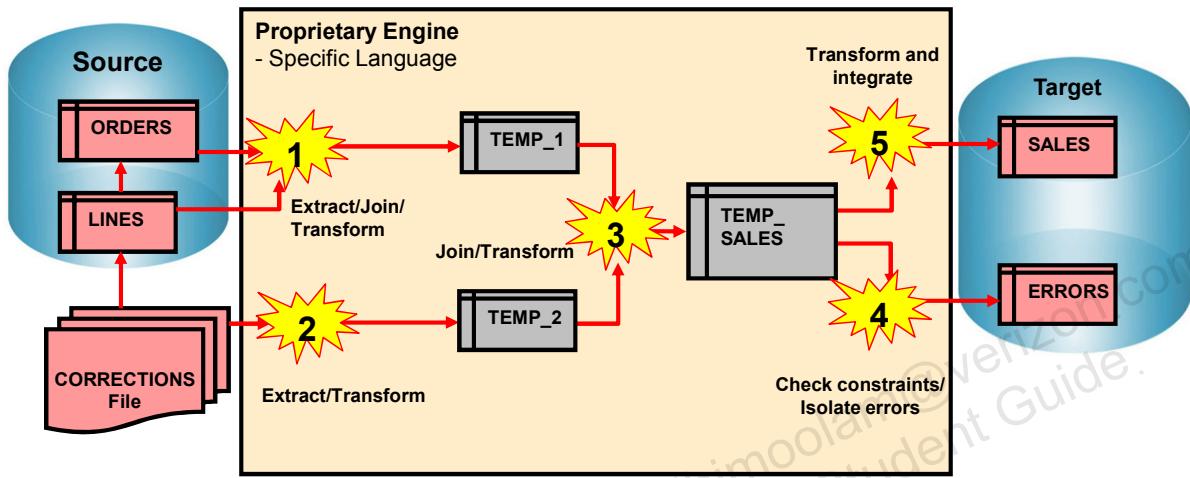
Now you look at how this process occurs. Imagine that you are attempting this process manually. What steps do you have to perform?

1. First, you extract the data from the `ORDERS` and `LINES` tables, then join it and place it into a temporary table.
2. Next, you extract the data from the `CORRECTIONS` file and place that data in another temporary table.
3. Then, you join these two tables to combine the corrections into the orders data and store it somewhere. You normally want this temporary table to closely resemble the target `SALES` table.
4. Before integrating the data into the destination table, you run some checks and isolate the errors in a separate table.
5. Finally, you integrate the remaining data in the `SALES` table with `INSERT`, `UPDATE`, or `MERGE` statements.

However, the implementing sequence of operations depends totally on the physical architecture and the tools that you have available.

Process Implementation: Example 1

Using a proprietary engine or application



ORACLE

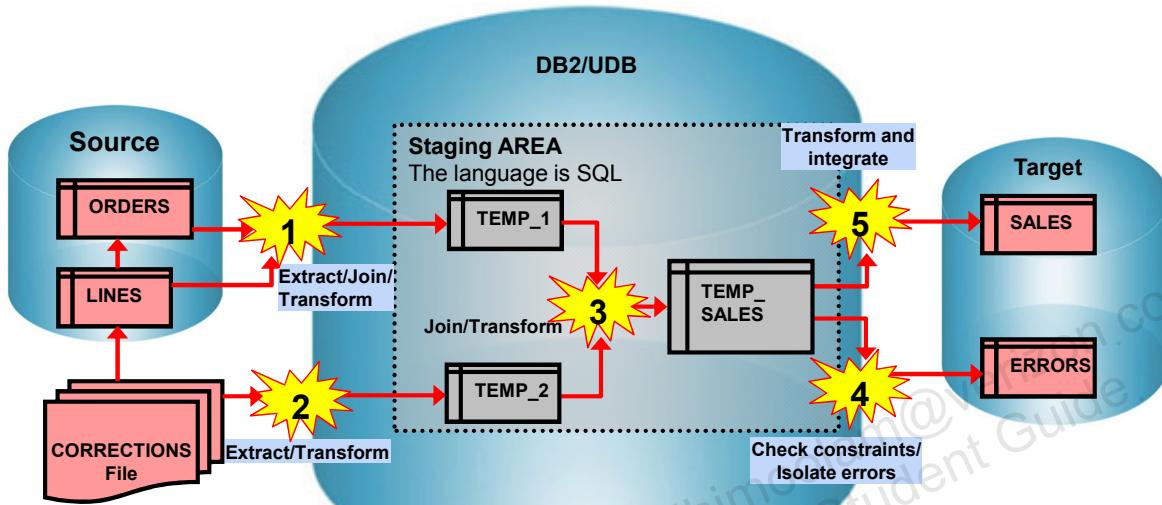
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

One solution is to use a custom application running on your desktop. You can, for example, code the whole operation in a Java applet. This applet connects to the Source server, perhaps by using a custom library, and retrieves the data. It can use a proprietary engine to perform the joins and transformation. It then connects to the Oracle server to write the results.

This can work, and with appropriate technologies, can result in an acceptable performance. However, the application has to be recoded every time any of the business rules changes. Similarly, if the results are to be saved to an Excel spreadsheet instead of the Oracle server, major changes are required.

Process Implementation: Example 2

Using an RDBMS data server: DB2/UDB



ORACLE

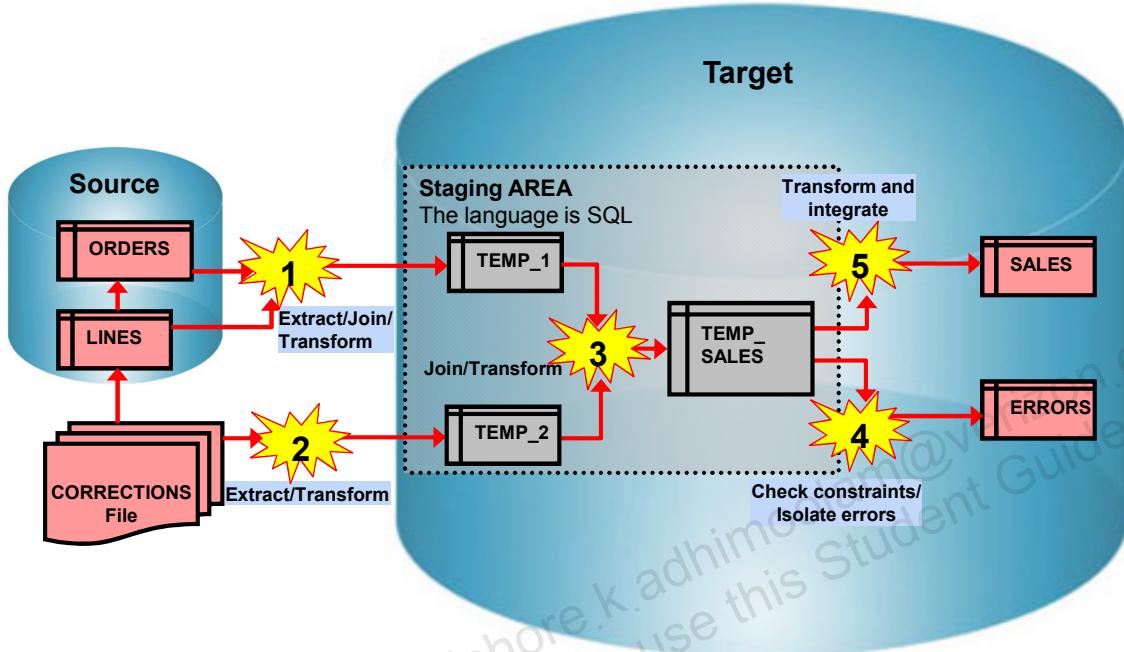
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A second, traditional solution is to use a dedicated Extract-Transform-Load (ETL) server. Here, data is first imported into the transformation server. Depending on the technology and the physical topology, it might be necessary to first dump the data from the source, copy the data across a network, and re-import it. Then the transformations would be hard-coded in SQL. Finally, a similar process is used to move the data into the Oracle server.

This is a very common solution to the problem. However, it means that the business rules end up being tightly coupled with the SQL code, which creates temporary tables and performs transformations. It also means that if either the source or target database servers change, some script files need to be modified. In short, the implementation is not flexible.

Process Implementation: Example 3

Using the Oracle target data server (the “ODI way”)



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ODI generally encourages the use of the E-LT method—extracting the data from the source, and then performing both transformation and loading on the target server. To do this, it uses the notion of a staging area.

In this example, you put the staging area on the target server. Now your problem consists only of extracting the data from the source server and getting it to the Oracle target. Then all the transformation and loading logic is performed directly on the Oracle server, in a separate schema. This keeps the temporary data separate from the usable data.

However, you still have not solved the problem of separating business rules from the implementation.

Agenda

- ODI Mappings
- Expressions, Join, Filter, Lookup, Sets, and Others
- Behind the Rules
- **Staging Area and Execution Location**
- Understanding Knowledge Modules
- Mappings: Overview



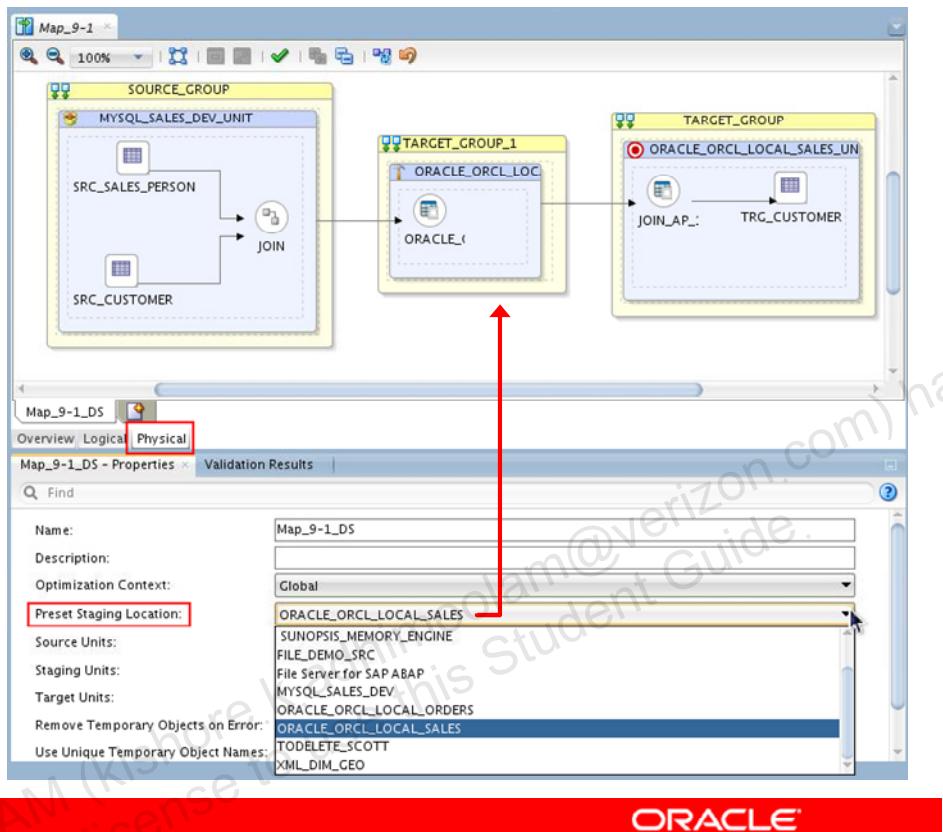
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The next two slides examine using staging areas and choosing the location to execute a mapping.

What Is the Staging Area?

- Logical
 - Hint
- Physical
 - Command



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You have just been introduced to the notion of a “staging area.” The goal is to create a place where a majority of the transformation and error checking are performed. Then all that is left to do is to copy or load the data into the target server.

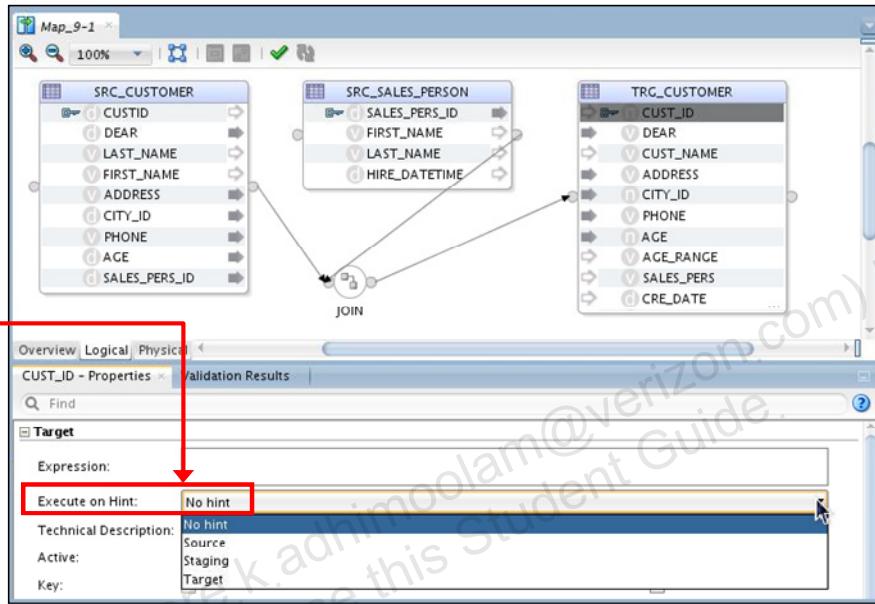
The staging area itself is a special area that you must create or specify somewhere in a database. ODI creates temporary tables and other objects in the staging area and uses these to perform data transformation.

You can place the staging area on your source server, target server, or another server altogether. However, the best place for the staging area is usually on the target server. The target server tends to give the best performance and reduces the complexity of data transfer processes.

The Staging Area location is merely a hint at the Logical level (which can be overridden), but is a command at the Physical level. When you choose a schema, that implies a location. A hint of “Undefined” means that ODI will pick what it perceives is the best location in that circumstance.

Execution Location

- In ODI Mapping, each rule may be executed on:
 - Source
 - Target
 - Staging
- Execution location is specified at design time.
- How a rule is implemented depends on the technology of its execution location.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Now you begin to see the flexibility of using ODI to perform your E-LT operations. Recall that when you described your mappings before, you specified only the transformation you wanted. You did not specify how or where this should happen. In fact, you can specify that a mapping be executed on the source or target servers, or in the staging area.

You specify the execution location at design time by selecting the appropriate option in the mapping window. The component with the suffix of _AP (Access Point) in the name is where the staging happens. The staging area could be the source, the target, or an intermediate location.

At run time, ODI generates the appropriate code to take this choice into account. However, the SQL expressions that you write must also be compatible with the technology.

Note: The staging area must be an RDBMS, with a schema dedicated to ODI temporary objects.

Agenda

- ODI Mappings
- Expressions, Join, Filter, Lookup, Sets, and Others
- Behind the Rules
- Staging Area and Execution Location
- **Understanding Knowledge Modules**
- Mappings: Overview



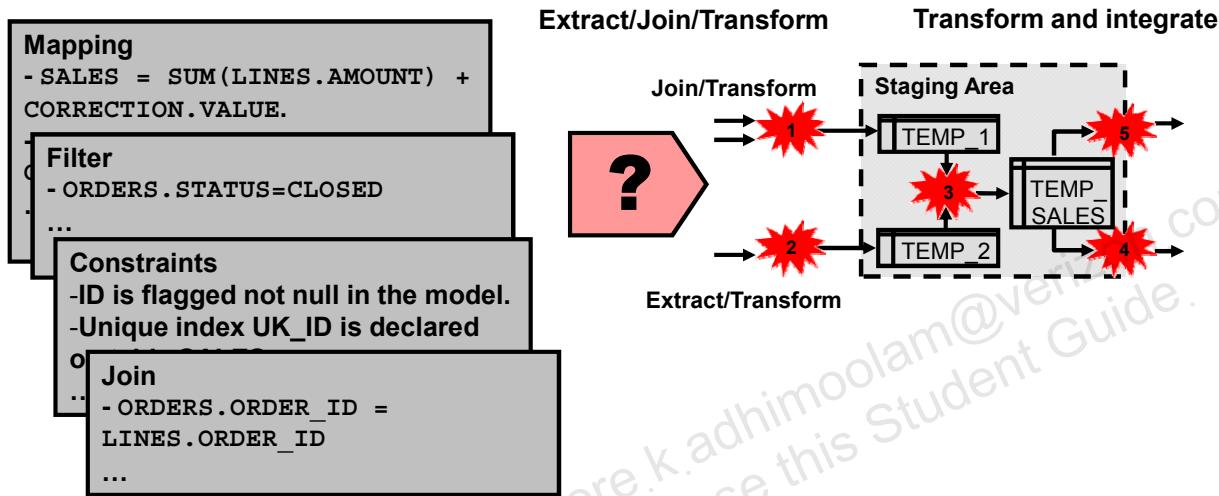
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The next few slides explore how to use Knowledge Module code templates to implement the abstract logic of your business rules on the data servers of a given technology.

From Business Rules to Processes

After the business rules are implemented, how does ODI generate the transformation processes?



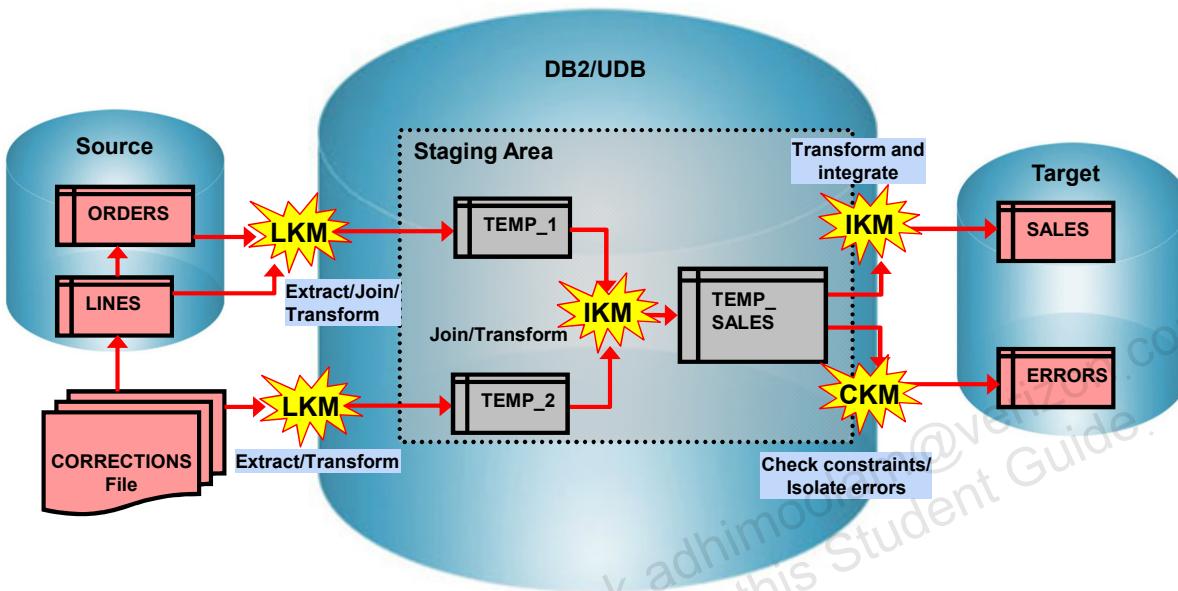
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

One of the key benefits of using ODI is separating business rules from the processes required to transform data. You have seen how you can implement these processes manually by using several different techniques. But how does ODI know how to do this and how does it remain flexible enough to handle any conceivable server topology, while keeping the business rules separate from the implementation?

Knowledge Modules

Knowledge Modules implement the required operations.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The answer is Knowledge Modules. Knowledge Modules define a way of implementing business rules on a given technology. They connect the abstract logic of your business rules to the concrete reality of your data servers.

If your data servers change, you just select a different Knowledge Module to match. Your business rules remain unchanged. If your business rules change, you do not have to modify any code—you just update the SQL expressions that define them in ODI.

In your example, you have Loading Knowledge Modules that describe how to move data from the source server to the DB2/UDB staging area. Integration Knowledge Modules perform the work of creating temporary tables and moving data onto the Oracle target server. Lastly, Check Knowledge Modules implement constraint checking and isolating errors in a separate table.

Note: Knowledge Modules are generic because they enable data flows to be generated regardless of the transformation rules. And they are highly specific because the code they generate and the integration strategy they implement are finely tuned for a given technology.

What Is a Knowledge Module?

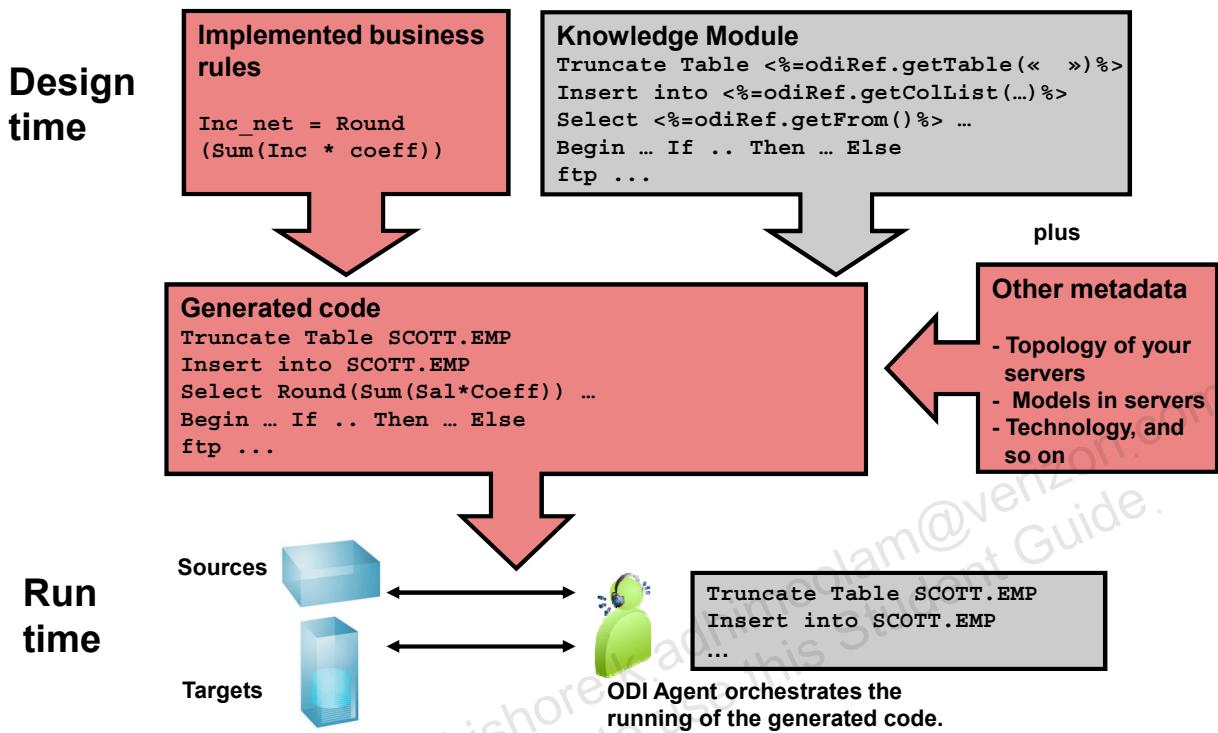
- It is a template designed for a specific processing task, such as loading, integrating, or checking in an integration process.
- It contains:
 - A sequence of commands for the technology it is designed for (in SQL, shell, native language, and so on)
 - Substitution tags, which connect the template to your business rules
- It is used to generate code from business rules and other metadata you defined in ODI.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Now that you have seen the usefulness of Knowledge Modules, how do they actually work? A Knowledge Module is a kind of template designed for a specific task, such as loading, integrating, or checking in part of an integration process.

Code Generation



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You have seen how business rules and their implementation on a server have been carefully kept separate. Now you will see how ODI combines the two at run time.

For example, you have a mapping that the net income of an employee is the sum of the income components multiplied by their coefficients. This is an expression in SQL, but there is no code to perform this integration.

You have a Knowledge Module that can wipe a destination table and fill it with source data. However, it knows nothing about your business rules.

Then you have all other metadata that you defined in ODI: the topology of your servers, the models that exist on them, the technologies used by each server, and so on.

When you put these three things together, ODI generates a code to carry out the integration. This code is specific to a technology, a layout, and a set of business rules. However, if any of these three things changes, all you need to do is regenerate the code.

At run time, the ODI agent orchestrates the running of the generated code. Based on execution locations, various parts of the generated code are executed either on the Source, Staging, or Target.

KM Types Used in Mappings

The following types of Knowledge Modules are used in mappings:

KM Type	Description
LKM	Loading
IKM	Integration
CKM	Check



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Mappings in ODI primarily rely on three types of Knowledge Modules. They are:

- **Loading Knowledge Modules (LKMs):** They define how to extract and then re-assemble data between specific technologies.
- **Integration Knowledge Modules (IKMs):** They determine the strategy for populating the target datastore(s). You should choose this strategy based on the type of data, how much data is being transferred, and whether you want to delete the target datastore(s) first.
- **Check Knowledge Modules (CKMs):** They enforce the constraints that you define on the target datastore(s). Generally, these are the least specific to a certain technology.

Note: Ensure that all useful Knowledge Modules are imported into the mapping project or into Global KMs.

Agenda

- ODI Mappings
- Expressions, Join, Filter, Lookup, Sets, and Others
- Behind the Rules
- Staging Area and Execution Location
- Understanding Knowledge Modules
- **Mappings: Overview**



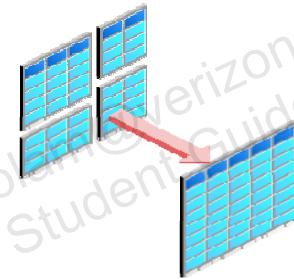
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You learn about the basic concept of mappings in the following slides.

Purpose of a Mapping

- A mapping is an ODI object that you define.
- It loads one or more target datastores with data from one or more source datastores, based on the business rules implemented as mappings.
- The attributes in the source datastores are linked to the attributes in the target datastores through business rules.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

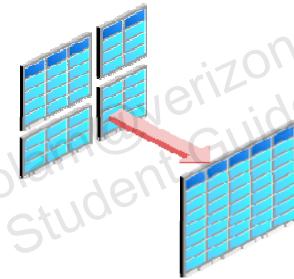
A mapping populates one datastore, called the “target,” with data from one or more other datastores, known as “sources.” The attributes in the source datastores are linked to the attributes in the target datastore through business rules. The business rules are implemented as “mappings.”

With ODI 11g you could only load one target. New with ODI 12c is the ability to load multiple targets.

What Is an Expression?

An expression is:

- A business rule implemented as a SQL clause
- A transformation rule that maps attributes in source datastores onto one of the target datastore attributes
- Executed by a database server at run time



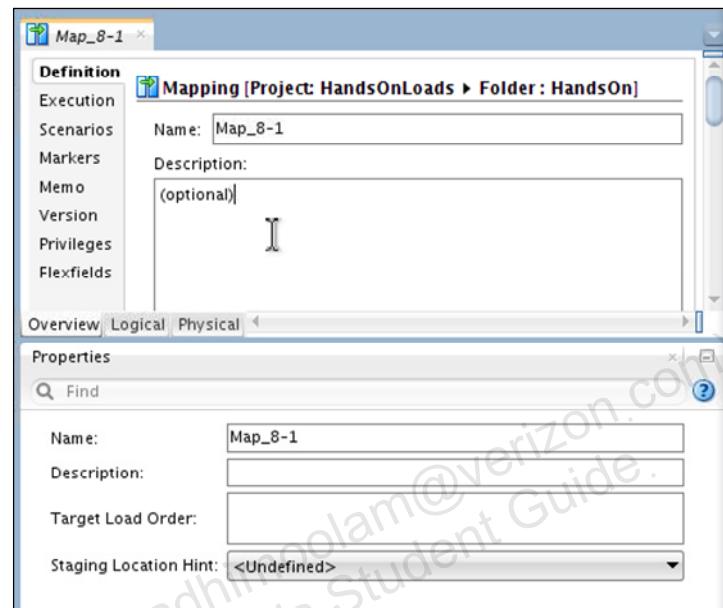
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

- It is a rule that maps one or more attributes (columns) from one or more source datastores onto one of the target datastore attributes (columns), possibly transforming them in the process.
- It can be as simple as stating that the AGE attribute in the target datastore corresponds to the AGE attribute in the source datastore, or as complex as calculating aggregates or operations on multiple attributes in various datastores located on different servers.
- An expression can be defined manually by entering code in the Expression field or by using the drag-and-drop functionality. You can now drag and drop directly over the target attribute. Multiple source attributes can be selected together for the drag and drop. You may need to “freeze” the panes while doing this.
- ODI also includes an Expression Editor that helps you build the mapping.

Creating a One-to-One Mapping

1. Create and name the mapping.
2. Define the target datastore.
3. Define the source datastore.
4. Define the mappings.
5. Save the mapping.



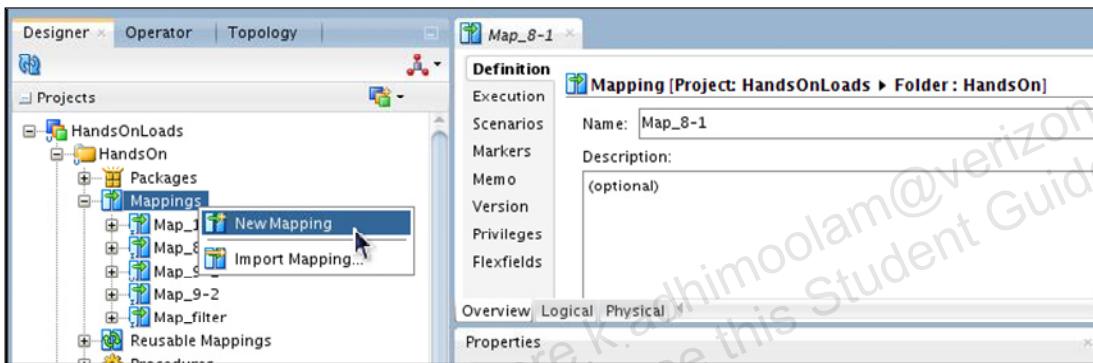
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A one-to-one mapping transforms data from one source datastore into one target datastore. The basic process for creating a simple one-to-one mapping is as shown. The following slides will cover each of these steps in detail.

Creating and Naming a Mapping

1. Navigate to the project and folder where you want to create your mapping.
2. Right-click the Mappings node and select New Mapping.
3. Enter the name.
4. Enter the description (optional).



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a blank mapping, perform the following:

1. In the Projects view, find the project and folder where you want to create the mapping.
2. Right-click the Mappings node and select New Mapping.
3. Select a name for the mapping. You should probably follow a naming convention defined for your project.
4. You can optionally enter the description for your mapping. This description generally explains the purpose of the mapping. It appears in the mapping documentation, and may be useful later when performing changes or maintenance on this mapping.

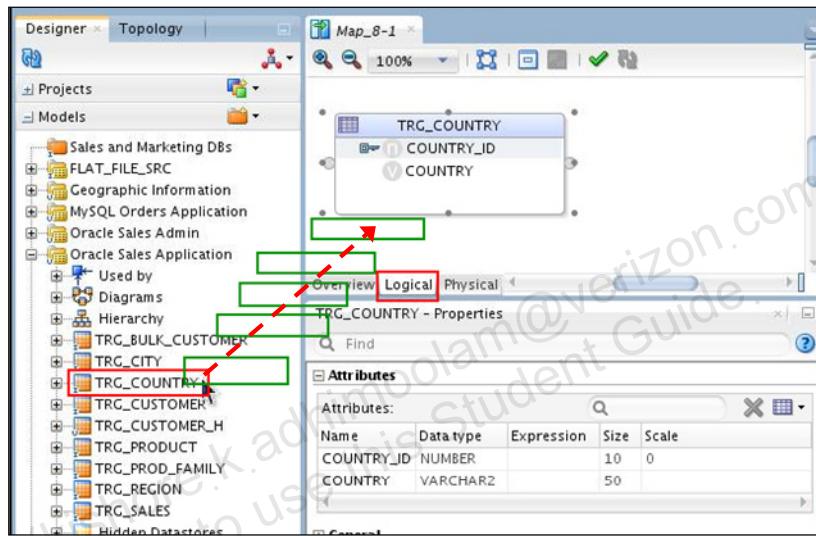
Optimization Context

Optimization Context is found on the Physical tab. Optimization Context sets the initial ***design*** context that determines which “version” of the sources and targets will be dragged onto the Mapping tab areas. It also determines the physical location where the data will come from when you right-click data on the object. It is **not** the ***execution*** context, which determines which physical schema the mapping will use at execution time.

Note: Ensure that the appropriate Knowledge Modules have been imported into the project.

Defining the Target Datastore

1. Select the Models view.
2. Expand the model containing the target datastore.
3. Click the mapping's Logical tab.
4. Drag the datastore from the Models view to the Logical map area.



ORACLE

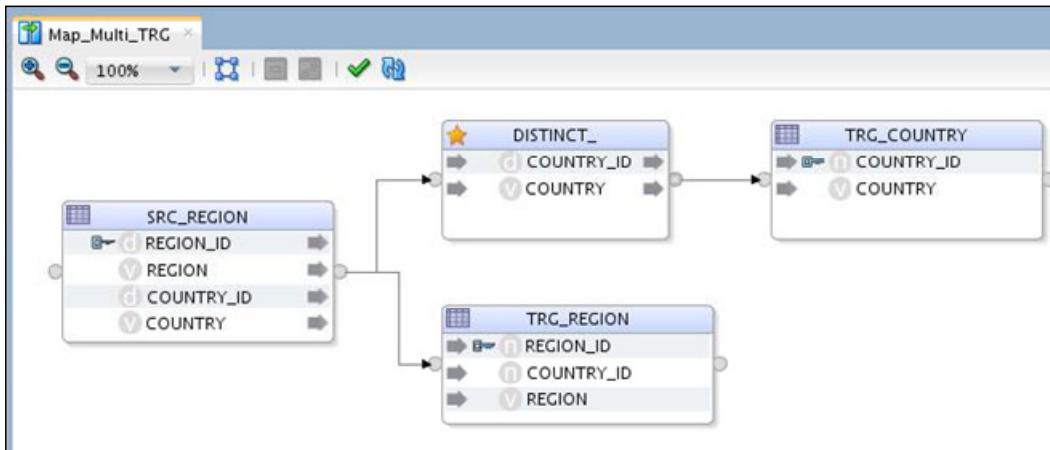
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The target datastore is populated by the mapping when it is executed.

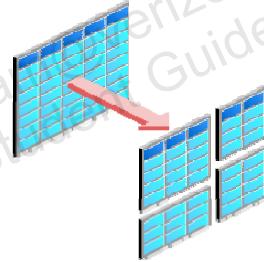
To define the target datastore:

1. You select the Models view usually found at the left of the Designer window.
2. Then, you locate and expand the node of the model containing the datastore that you want as the target of your mapping.
3. You now click the mapping's Logical tab. On this tab, you can define the source and target datastores, and the various mappings between them.
4. Finally, you drag the datastore from the Models view to the Logical map area.

Multiple Targets



Can map to several targets simultaneously



ORACLE

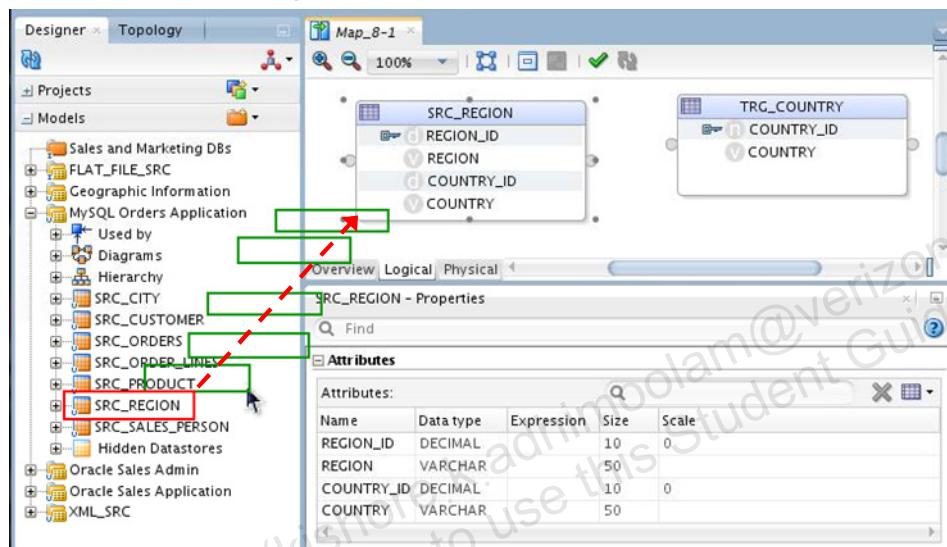
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

New in ODI 12c is the ability to map to several targets simultaneously.

- Can use mapping without a Split component
Example: SRC_REGION → TRG_REGION and TRG_COUNTRY
- Can use mapping with a Split component
Example:
 - HR.EMPLOYEES WHERE AGE < 65 Split to table1;
 - HR.EMPLOYEES WHERE AGE >= 65 Split to table2;

Defining the Source Datastore

1. Expand the model containing the source datastore.
2. Drag the datastore from the Models view to the Diagram zone in the mapping.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The source datastore supplies data that the mapping uses to populate the target datastore when it is executed.

To define the source datastore:

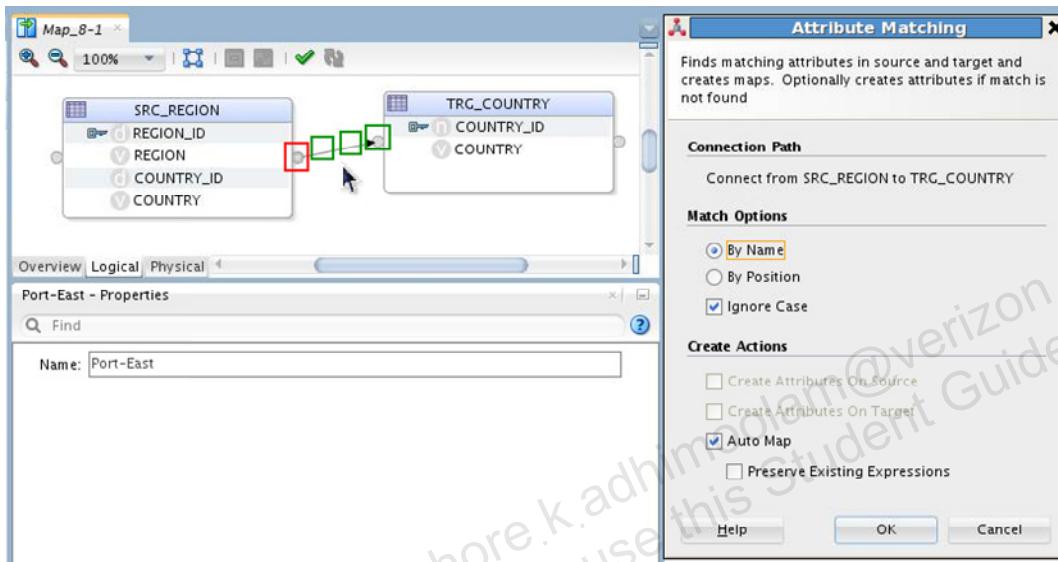
1. Find and expand the model that contains the datastore you want to use as the source of your mapping.
2. Drag the datastore from the Models view to somewhere on the mapping's Logical tab.

Note

- A mapping may have more than one source.
- For this lesson, you use only one source.
- Automatic Attribute Matching creates mappings by matching attribute names automatically. You may disable the information window.

Connecting the Ports to Make the Map

3. Drag the east port (source) to the west port (target).
4. Accept Attribute Matching.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

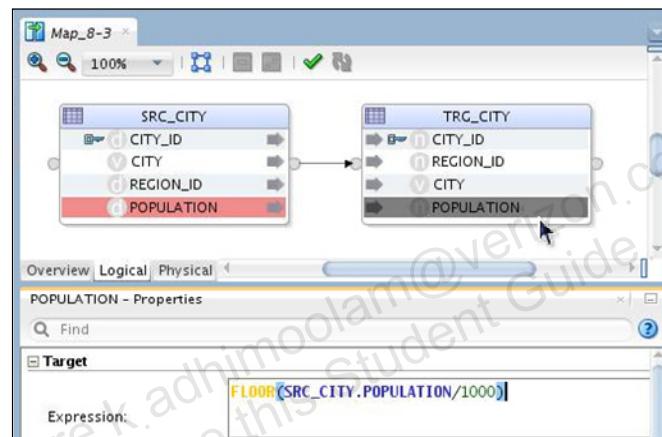
3. Drag the east port from the left table (source) to the west port of the right table (target). When the two ports are connected, an arrow appears between them with the arrowhead indicating the direction of flow (from source to target).
4. When the two ports are connecting, you have the option to automatically map identically-named (or identically-ordered) attributes one to the other. A pop-up window appears to notify you of this automatic mapping. You can disable this information window. Attribute Matching maps each source attribute onto the matching target attribute without any transformation. If you want to define transformations, you need to edit the mappings later.

Do not worry about placing the tables carefully. You have the option later to tidy up the map diagram and make all the icons aligned.

With ODI 12c you can now directly drag over the target attribute. Multiple source attributes can be selected together for the drag and drop.

Defining the Expressions

1. Select the target datastore attribute that you want to map.
2. Click the Target tab in the Properties pane, and then drag attributes from the source table directly into the expression area. You may use the Expression Editor.
3. If possible, validate the whole mapping (the green check mark).



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You now see how to manually define an expression. You can have one expression for every target attribute. It is also technically possible to have no expressions, which would have the effect of deleting the target. To create one expression, perform the following:

1. Select the attribute from the target datastore that you want to map. You usually have at least one source attribute in an expression, but you may have more than one.
2. Manually edit the expression by clicking in the expression field. You can drag attributes from the source table directly onto the expression code. You can then modify the expression by entering the code. You may also use the Expression Editor by clicking its icon at the right of the expression box.
3. Validate the mapping (including the expression) by clicking the green check mark. This sends a query to the DBMS engine to check that the expression syntax is correct. You must always try to check the code syntax. Often the color indicates valid/invalid real time while entering the formulas.

Valid Expression Types

The following types of clauses may be used in the expressions:

Value	String values should be enclosed within single quotation marks: 'SQL', but not for numeric values: 5, 10.3
Source Column	Drag the column or use the Expression Editor. It is prefixed by the datastore's alias. Example: SRC_SALES.PROD_ID
DBMS Function	Use the Expression Editor for the list of supported functions and operators, or enter them directly.
DBMS Aggregate	MAX(), MIN(), and so on in an Aggregate component. ODI automatically generates the GROUP BY clause.
Combination	Any combination of clauses is allowed: SRC_SALES_PERSON.FIRST_NAME ' ' UCASE(SRC_SALES_PERSON.LAST_NAME)



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

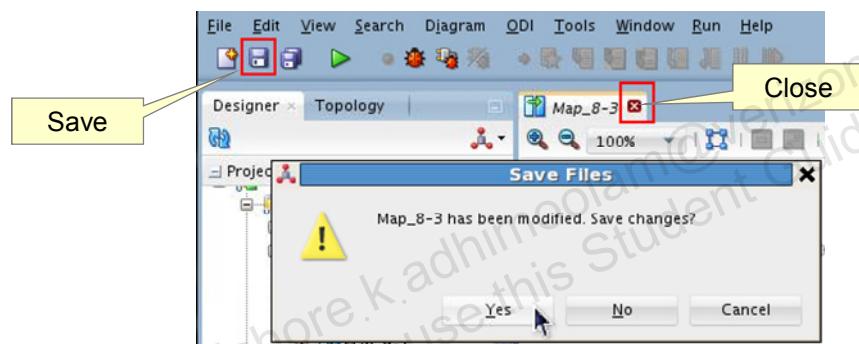
Expressions may include any type of clause that is allowed in SQL languages. This includes:

- **Constant values:** These represent fixed numeric or string values that do not depend on the source tables. For example, you may want to always populate a target field "Approved" with 0 to mean that sales always starts unapproved. Enter numeric values directly, but enclose strings within single quotation marks.
- **Source attributes:** Precede the attribute name with the name of the datastore. It is suggested that you drag attribute names from the diagram in the Expression Editor to add source attributes to an expression clause.
- **DBMS functions:** You can use any functions supported by the DBMS that will run the mapping's generated code.
- You can use aggregate functions such as MIN, MAX, and COUNT but they *MUST* be within an aggregate component. ODI automatically manages the generation of the specific GROUP BY clauses in these cases.
- You can combine constants, columns, DBMS, or aggregate functions in any legal SQL way in the code to build your mapping expressions.

This is just the start of what you can do with mappings. The rest is covered in the lessons about advanced mapping designs.

Saving the Mapping

- Click **Save** (diskette) to save the mapping.
- You can click **Yes** to save and close the mapping.
- **Cancel** closes the mapping without saving.
- Mappings are saved in the Work Repository.
- The Edit/Undo feature can Undo the last change, even if you have issued a Save!



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

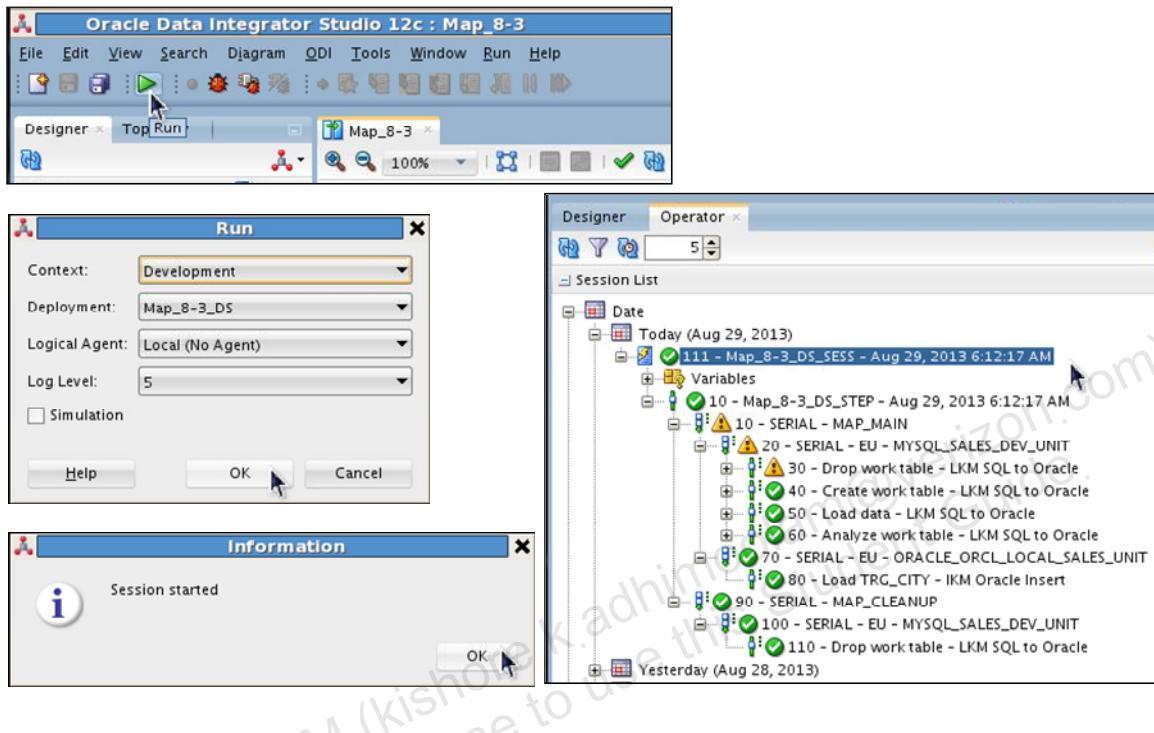
To save a mapping:

- Click **Save** in the menu toolbar. The italic name on the tab will change back to regular, and the diskette icon will be disabled. Clicking **Close** on the tab will cause the Save dialog box to display.
- You may also click **Yes** or **Cancel** to close the mapping after saving or discarding it, respectively. You will be prompted to confirm if you click **Cancel** to discard the changes that you have made.

Note

- Mappings are not saved locally on your machine, but in the centralized ODI Work Repository.
- Save your work regularly by clicking Save.

Running the Mapping



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

1. Click **Run** (green arrow) to execute the ODI Mapping.
2. Click **OK** in the Run window that appears, and then click **OK** when the Session Started message appears.

The execution results in ODI Operator are displayed as shown in the screenshot.

Quiz

Which of the following statements is *not* true?

- a. ODI is based on business rules.
- b. ODI requires a proprietary engine.
- c. ODI can generate native or standard SQL code.
- d. ODI business rules are defined on a mapping.
- e. In ODI, the processing instructions are implemented in Knowledge Modules.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b

Explanation: ODI exploits in-house RDBMS engines to perform the required processes. When ODI generates ANSI-standard SQL code, that code should run on any SQL database.

Quiz

How would you populate several targets simultaneously in ODI 12c from one source?

- a. You need to insert several datastores in the target area.
- b. You need to insert several datastores in the target area and you must use a Split component.
- c. It cannot be done in one shot, you have to make multiple mappings and put them together in a procedure.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: a

Explanation: In ODI 11g, an interface (currently called a mapping) could have only one target datastore. If you wanted to populate several targets in 11g, you needed to create several interfaces as in answer c. In 12c, you can have multiple targets, either with or without using a Split component.

Summary

In this lesson, you should have learned how to:

- Describe the concept of ODI Mapping
- Describe the concept of expressions, joins, and filters
- Describe the process of implementing business rules
- Describe the concepts of staging area and execution location
- Use Knowledge Modules with ODI Mapping
- Create and execute a basic ODI Mapping
- Use lookups



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

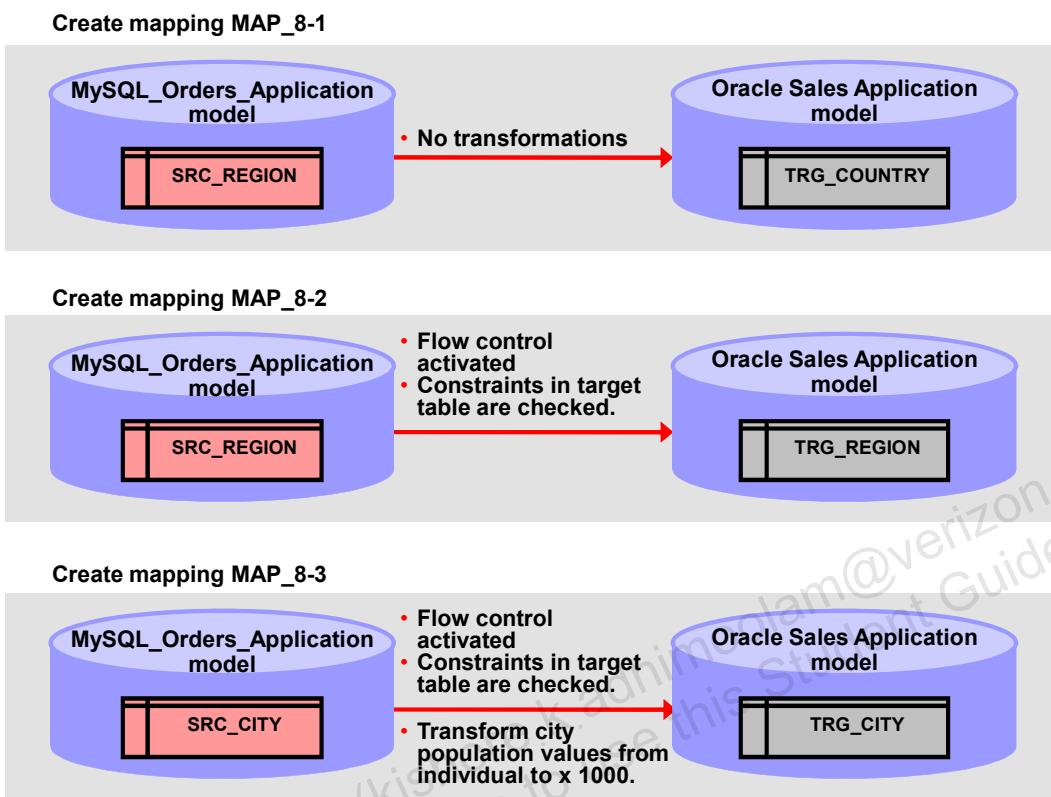
Checklist of Practice Activities

- ✓ 1-1: Logging In and Help Overview
- ✓ 2-1: Creating and Connecting to ODI Master and Work Repositories
- ✓ 3-1: Configuring Standalone Agents Using the Common Admin Model
- ✓ 4-1: Working with Topology
- ✓ 5-1: Setting Up a New ODI Project
- ✓ 6-1: Creating Models by Reverse-Engineering
- ✓ 7-1: Checking Data Quality in the Model
- 8-1: **Creating ODI Mapping: Simple Transformations**
- 9-1: Creating ODI Mapping: Complex Transformations
- 9-2: Creating ODI Mapping: Implementing Lookup
- 10-1: Creating ODI Mapping: Exporting a Flat File to a Relational Table
- 11-1: Using Native Sequences with ODI Mapping
- 11-2: Using Temporary Indexes
- 11-3: Using Sets with ODI Mapping
- 12-1: Creating and Using Reusable Mappings
- 12-2: Developing a New Knowledge Module
- 13-1: Creating an ODI Procedure
- 14-1: Creating an ODI Package
- 14-2: Using ODI Packages with Variables and User Functions
- 15-1: Debugging Mappings
- 16-1: Creating and Scheduling Scenarios
- 17-1: Using Load Plans
- 18-1: Enforcing Data Quality with ODI Mappings
- 19-1: Implementing Changed Data Capture
- 20-1: Setting Up ODI Security
- 20-2: Integration with Enterprise Manager and Using ODI Console

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 8-1: Mapping: Simple Transformations



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In this practice, you create three mappings. First, you create a mapping called MAP_8-1 by loading the TRG_COUNTRY datastore in the Oracle Sales Application model with the content of the SRC_REGION table from the MySQL Orders Application model. This simple mapping has no transformations.

The second mapping, MAP_8-2, starts out as a duplicate of MAP_8-1. The target is changed to TRG_REGION, flow control is activated, and constraints in the target table are checked.

The third mapping, MAP_8-3, loads the TRG_CITY datastore in the Oracle Sales Application model with the content of the SRC_CITY table from the MySQL Orders Application model. In this mapping, flow control is activated, constraints in the target table are checked, and city population values are transformed from individual to times 1000.

9

Designing Mappings

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Design Oracle Data Integrator (ODI) mappings with multiple-source datastores
- Create joins and lookups, and filter data
- Define the flow in ODI Mapping
- Specify an ODI Mapping Staging area and execution location
- Select Knowledge Modules



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Agenda

- **Multiple Sources and Joins**
- Filtering Data
- Overview of the Flow in ODI Mapping
- Selecting a Staging Area
- Configuring Expressions
- Execution Location
- Selecting a Knowledge Module



ORACLE

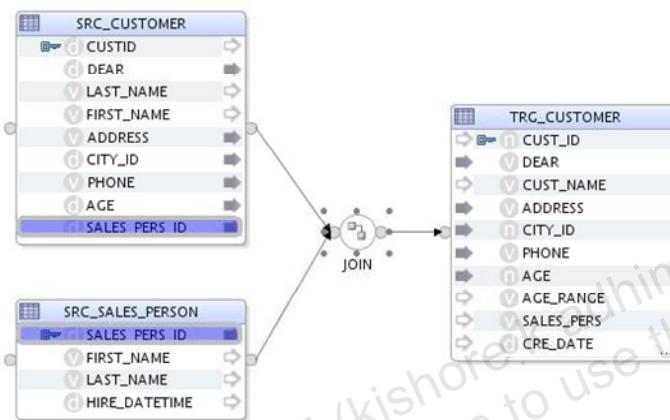
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You learned how to create a one-to-one mapping with a single source. Now you learn how to combine multiple sources in a single mapping.

You learn how to create mappings with multiple-source datastores, and how to create joins or relationships between these sources, in the following slides.

Multiple-Source Datastores

- You can add more than one source datastore to a mapping.
- Datastores must be linked by using joins.
- Joins must be manually defined in the diagram either by:
 - Drag-and-drop of attributes onto each other
 - Dragging a join component and typing in the Properties



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Populating a target datastore from multiple-source datastores has a wide range of uses. There should be a relationship between the tables, which is then expressed as a join. If any source datastores are unjoined when you try to execute a mapping, ODI displays an error.

Creating a Join Manually

1. Drag an attribute from one datastore onto an attribute in another datastore.
 - A join linking the two datastores appears in the diagram.
 - In the join code box, an expression joining the two attributes also appears.
2. Modify the join expression to create the required relation.
 - You can use the Expression Editor.
3. Check the expression's syntax.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

1. To manually create a join, either:
 - Drag an attribute from one datastore onto an attribute in another datastore, or
 - Drag a join component from the Component Palette and then edit the Properties.

ODI creates a simple join, linking rows where the values in these two attributes are identical. This join appears in the diagram as a bent line connecting the two attributes, and also in the join code box as an expression, such as
TES.ID=SRC_CUSTOMER.SALES_PERS_ID.
2. To express more complex relationships, you modify the join expression manually. Thus, you can incorporate database functions such as arithmetic or string-handling functions as more advanced ODI functionality. The best practice is to use the Expression Editor to avoid making errors in the code.
3. After creating the join expression, you should have its syntax checked by the DBMS, if possible. To do this, click the Check Expression button next to the join expression box.

Advanced Joins

- Heterogeneous joins:
 - Datastores from different models or schemas
 - Datastores from different technologies
- Joins between more than two datastores
 - Columns from any datastore in the diagram can be added to the join expression to create multiple datastore joins.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Simple joins, as seen earlier, feature two similar datastores from the same model, and thus share the same technology. However, you can create joins between datastores in different models (for example, two database tables in different schemas), and even across two servers that use different technologies. In these cases, ODI migrates the data to a Staging area and performs the join there.

Similarly, you can have joins between three or more tables. To do this, drag column names from other datastores in the diagram onto the join expression box.

Types of Joins

The following types of joins exist: cross, inner, outer, natural.

Cross-Join	Cartesian product; every combination of any customer with any order, without restriction
Inner Join	Only records where a customer and an order are linked
Left Outer Join	All customers combined with any linked orders, or blanks if none
Right Outer Join	All orders combined with any linked customer, or blanks if none
Full Outer Join	All customers and all orders

A **natural join** is a join statement that compares the common columns of two tables with each other. You should check whether common columns exist in both tables before doing a natural join.

Natural joins may cause problems if columns are added or renamed.

Also, no more than two tables can be joined by using this method.

So, it is best to avoid natural joins as much as possible.



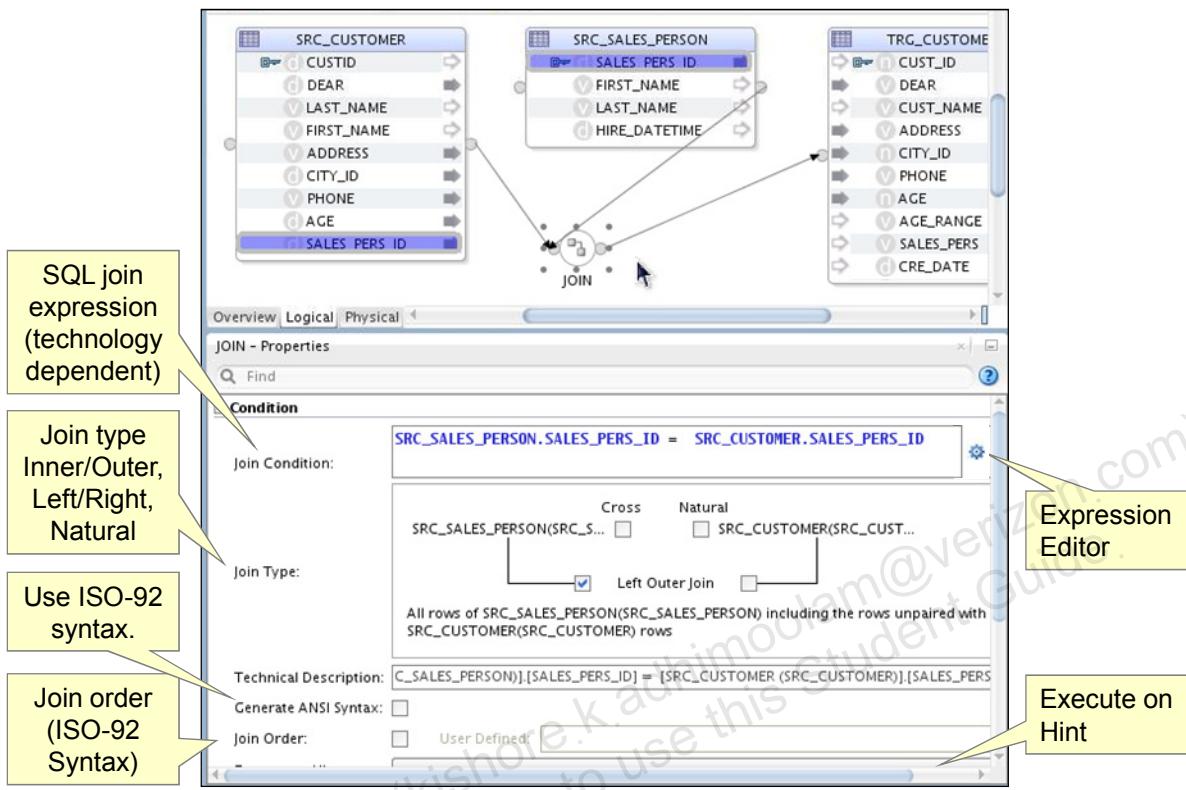
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The four types of joins are cross, inner, outer, and natural.

- A cross-join retrieves all possible row combinations between two tables, without any filtering.
- In an inner join, records from two tables are combined and added to a query's results only if the values of the joined fields meet the specified criteria.
- An outer join returns all the rows from the joined tables whether or not there is a matching row between them. In the three types of outer joins—left, right, and full—the type indicates the source of the main data.
 - When you use a left outer join to combine two tables, all the rows from the left table are included in the results. A right outer join is the same as a left outer join, but in the other way. Rows from the left table can be included more than once.
 - A full outer join retrieves all the rows from both joined tables. It returns all paired rows where the join condition is true, and the unpaired rows from each table concatenated with NULL rows from the other table. Effectively, it returns any rows that would be returned by either a left or right outer join.

Note: In some technologies, not all these operators are valid. In that case, ODI will not enable some types of join to be selected. Joins may be expressed either in a database-specific syntax or in a standard ISO-92 syntax, supported by most databases.

Setting Up a Join



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This is how the Join Property Inspector window appears when a join is selected.

The SQL join expression expresses the join, written in SQL. A valid SQL expression involving qualified column names from at least two different datastores is acceptable.

You can also launch the Expression Editor (gear icon), which helps you create more complex joins and avoid mistyping names.

The join type check boxes enable you to choose between a left, right, natural, or cross-join. Select a full outer join by selecting both left and right joins simultaneously. Note that ODI automatically rearranges a right outer join to be a left outer join, if possible, for readability. However, not every technology supports every type of join.

After you click the Validate Expression icon (green check mark: not shown; off the top of the slide), when available, the expression is sent to the DBMS for syntax checking. To avoid unpleasant surprises later, perform syntax checking when you finish a join.

After you click Save, the code is recorded in the repository. You should save frequently when working on complex joins. The expression does not have to be complete in order to be saved.

There are also some more advanced options. For example, the Execute on Hint (not shown; off the bottom of the slide) determines whether the joins are executed on the source, target, or the Staging area server. This choice can be important for performance or compatibility reasons.

You can select the Ordered Join (ISO) check box if it is important that the joins be performed in a certain order, using ISO-92 syntax, usually for performance reasons. In this case, you specify the Order Number to control the sequence of the joins. Low-numbered joins are performed before high-numbered joins.

The join test will not pass if the join is not entirely between two SQL sources on the same physical schema.

Note: Although ODI has always supported the ISO-92 Ordered Join syntax, from 11g Release 2 and higher versions of Oracle Database now support this syntax. This support was added to Oracle Server so that developers coming from other technologies can use a feature that is in other vendors' databases with which they are familiar.

Creating Lookups



Create and maintain lookups through the Lookup Component Properties.

You can generate lookups by using either of these methods:

- **Left outer join** (execution plan can be sort/merge, hash join)

```
select
    O.ORDER_ID,
    C.COUNTRY_NAME COUNTY_NAME_LOOKUP
  From ORDERS O left outer join COUNTRY C
    on (O.COUNTRY_ID = C. COUNTRY_ID)
```

- As an expression in the SELECT clause: **In-memory lookup with nested loops** (lookup table accessed in-memory)

```
select
    O.ORDER_ID,
    (select C.COUNTRY_NAME from COUNTRY C
      where O.COUNTRY_ID = C. COUNTRY_ID) COUNTY_NAME_LOOKUP
  From ORDERS O
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Properties panels in the mapping editor create lookups by using a source as the driving table and a lookup datastore or mapping. These lookups now appear as compact graphical objects in the mapping sources diagram.

The user can choose how each lookup is generated, either as a left outer join in the FROM clause or as an expression in the SELECT clause (in-memory lookup with nested loop). The second syntax is sometimes more efficient in small lookup tables.

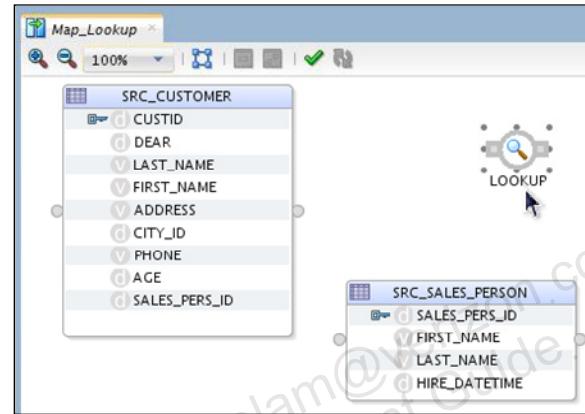
This feature simplifies the design and readability of mappings that use lookups, and enables optimized code for execution.

New with 12.1.2.0.1 patch is a set of options for what to do with Multiple-Match rows and No-Match rows.

Using Lookups



- Make sure that your chosen technology supports lookups.
Three possible levels of Lookup support:
 - None
 - Left outer join
 - Nested select
- Drag the Lookup icon from the Components palette.
- Connect the Driver and Lookup tables to the input ports.
- A Lookup acts like a special case of a join.



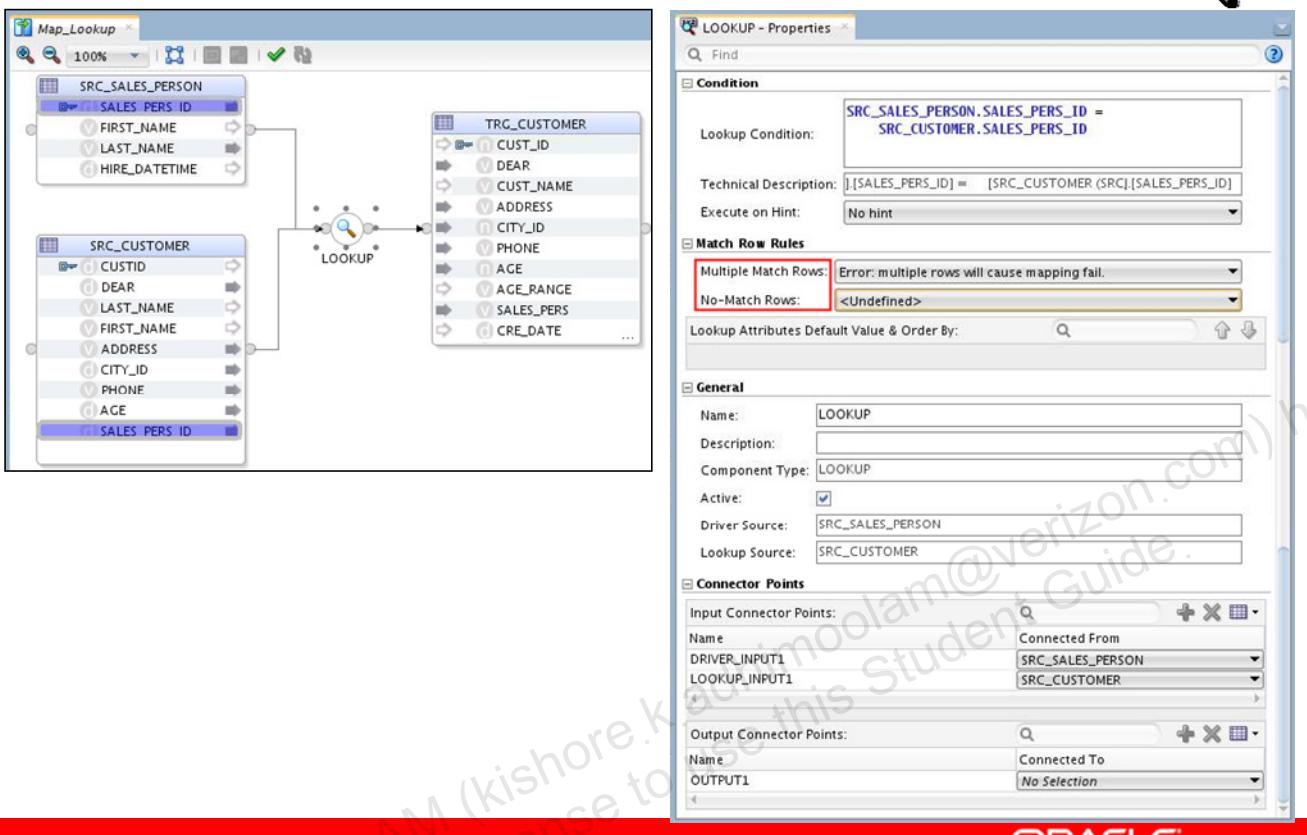
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Ensure that the technology you have chosen supports lookups. A lookup can be implemented in generated code either through a left outer join or a nested Select statement.

New with 12.1.2.0.1 are several additional lookup options, plus a reformatting of existing options.

Using Lookups



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

A lookup has two steps: First, you select your driving and lookup tables, possibly with drag-and-drop. Second, you define the lookup (join) condition, and can specify several options.

The Match Row Rules options shown, “multiple rows will cause mapping fail” and No-Match: <undefined>, is a “normal” lookup; that was the only behavior in 12.1.2.0.0. The other new options with the 12.1.2.0.1 patch are:

- Multiple Match Rows:
 - All rows (number of result rows may differ from the number of input rows)
 - Error: Multiple rows will cause mapping fail
 - Select any single row.
 - Select first single row.
 - Select last single row.
 - Select nth single row.
- No-Match Rows:
 - <Undefined>
 - Return a row with the following default values.
(Then each column is shown with room for a value.)

Agenda

- Multiple Sources and Joins
- **Filtering Data**
- Overview of the Flow in ODI Mapping
- Selecting a Staging Area
- Configuring Expressions
- Execution Location
- Selecting a Knowledge Module



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Data filtering aims at reducing the amount of source data to process in the mapping.

Filters in ODI

- Creating filters:
 - Filters defined on the datastores in their models are automatically copied into the diagram.
 - You can manually add additional filters in the diagram.
- Filters are used in mappings to reduce the amount of data retrieved from the source datastores.
- Filters are not constraints.
 - Nonmatching data is not “erroneous.”



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Filters in mappings are created in two ways, just as joins were.

- First, any filter defined in a datastore in the Models view is automatically copied into the mapping. This is useful when you want to filter data from a particular datastore always, for example, to never consider rows of the “temporary” type.
- Alternatively, you can manually add other filters to the datastore, which will apply only in this mapping. This is useful when different mappings work on different sets of data from the same datastore. For example, one mapping may treat only North American sales, whereas another works only on European sales. You will see how to do this in this lesson.

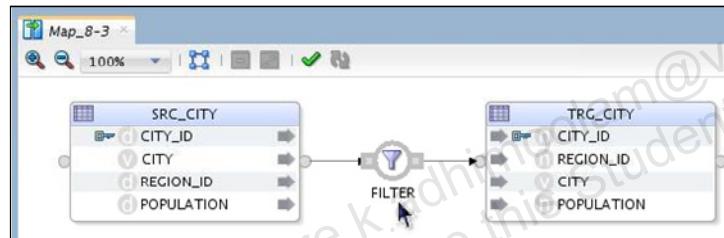
In either case, the goal of a filter in a mapping is the same: To reduce the amount of data being transferred to some specialized subset. Some examples might be “Sales worth more than \$1000,” “Orders from new customers,” or “Customers who have made at least one support request in the last six months.”

Ensure that you do not confuse filters with constraints. If the data that you want to remove is not erroneous, but irrelevant, you should use a filter rather than a constraint.

Defining a Filter Manually



1. Drag an attribute from one data source onto the diagram.
Or, drag a filter from the Components Palette.
 - A funnel representing the filter appears in the diagram.
 - In the filter expression field, the column name appears.
2. Modify the filter expression to implement your filtering rule.
 - You can use the Expression Editor.
3. Validate the mapping if possible.



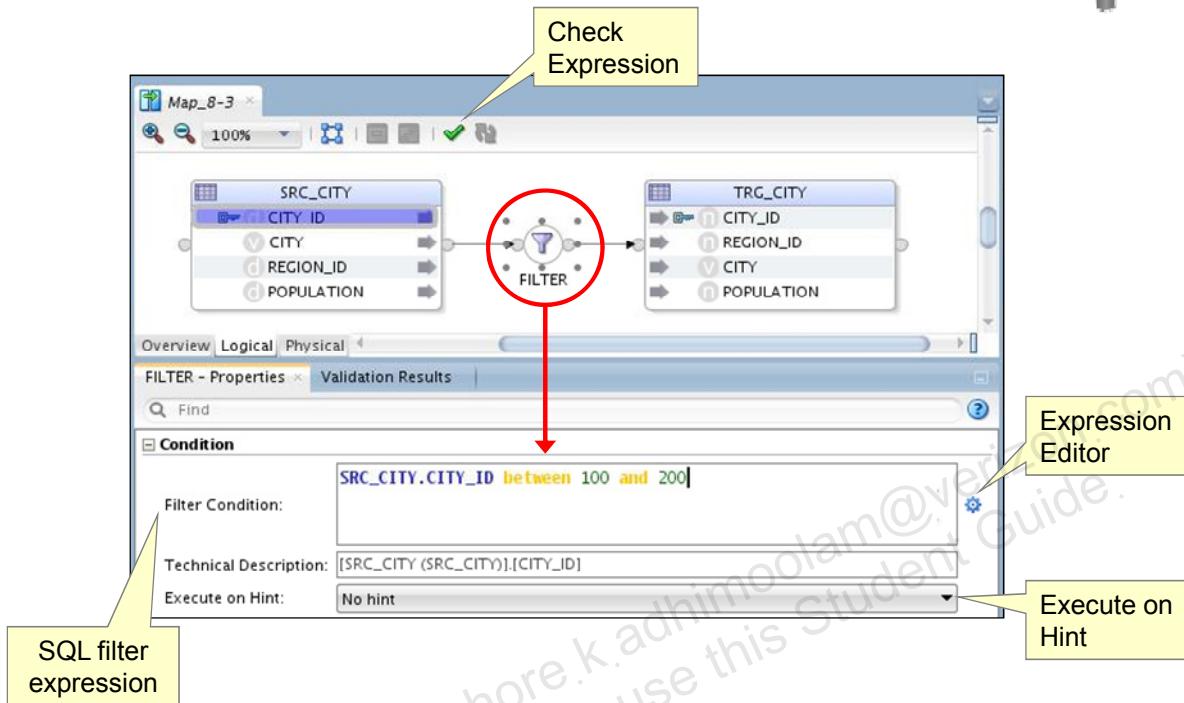
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To create a new filter manually, drag the name of an attribute onto an empty space in the diagram. Alternatively, you can drag a filter icon from the Components Palette (which accomplishes the same thing). You see a filter icon in the diagram connected to the column name by a line. Below the diagram, on the Implementation tab, you see the filter expression. For the moment, this consists of just the name of the column.

Modify this expression to implement the constraint that you want to impose. Make your modifications by entering details in the filter expression box, or by clicking the Expression Editor button and proceeding from there. Where possible, you should check the syntax of the expression by clicking the Validate Mapping (green check mark).

Setting Up a Filter



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In this slide, you see the entire Mapping screen as it appears when you create a filter. The SQL filter expression is where you type the condition that must be matched by the rows in the datastore.

Execute on Hint specifies the server on which the filtering is performed. This is examined more closely in later lessons. The Check Expression and Expression Editor buttons behave in the same way as for joins.

Agenda

- Multiple Sources and Joins
- Filtering Data
- **Overview of the Flow in ODI Mapping**
- Selecting a Staging Area
- Configuring Expressions
- Execution Location
- Selecting a Knowledge Module



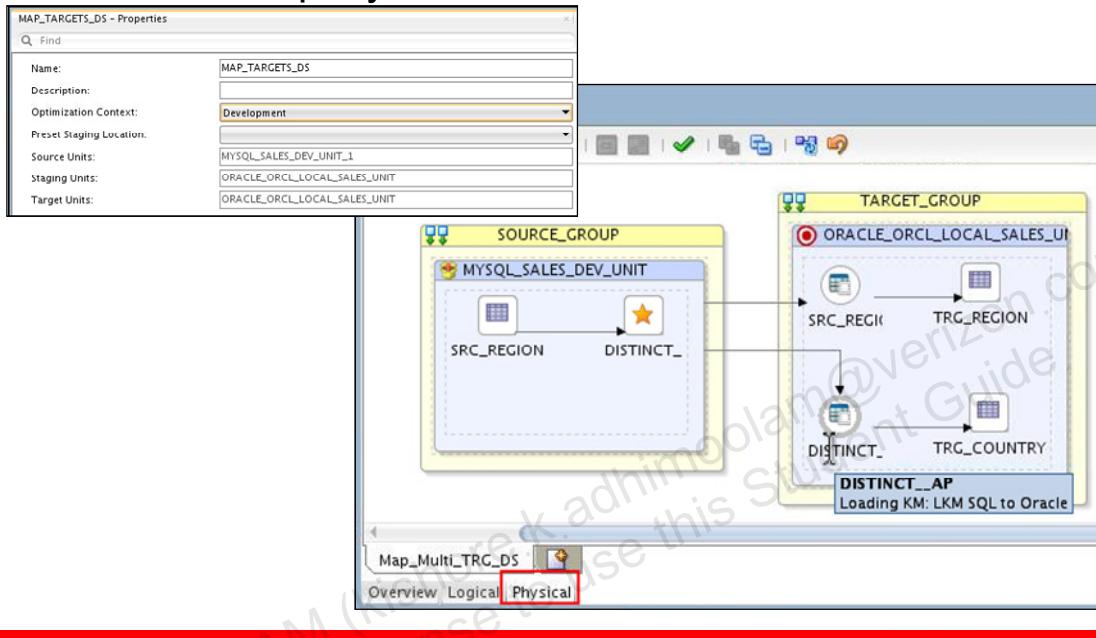
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The word “flow” has a special meaning in ODI. It refers to the passage of data between sources of data and integration targets. Understanding this flow is critical in configuring your integration path for maximum performance and flexibility.

Physical Mapping Diagram

- Deployment Specification
- Execution Groups: yellow boxes



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

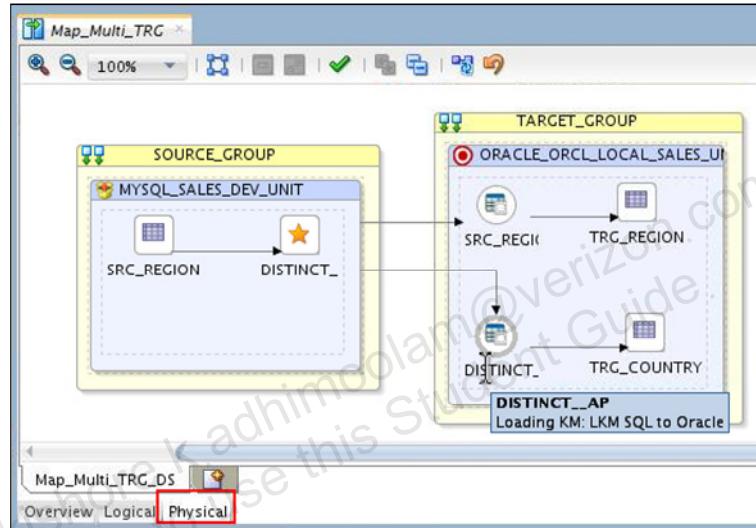
The following items appear in the physical diagram:

- **Deployment Specification:** The entire physical diagram represents one deployment specification. Click the background or select the white tab representing the deployment specification label to display the physical mapping properties. By default, the staging location is collocated on the target, but you can explicitly select a different staging location to cause ODI to automatically move staging to a different host.
- **Execution Groups:** Yellow boxes display groups of objects called execution units, which are executed in parallel among each other within the same execution group. These are usually Source Groups and Target Groups:
 - **Source Execution Group(s):** Source datastores that are within the same dataset or are located on the same physical data server are grouped in a single source execution group in the physical diagram. A source execution group represents a group of datastores that can be extracted at the same time.
 - **Target Execution Group(s):** Target datastores that are located on the same physical data server are grouped in a single target execution group in the physical diagram. A target execution group represents a group of datastores that can be written to at the same time.

- **Execution Units:** Within the yellow execution groups are blue boxes called execution units. Execution units within a single execution group are on the same physical data server, but may be different structures.
- **Access Points:** In the target execution group, whenever the flow of data goes from one execution unit to another, there is an access point (shown with a round icon). Loading Knowledge Modules (LKMs) control how data is transferred from one execution unit to another.
- An access point is created on the target side of a pair of execution units, when data moves from the source side to the target side (unless you use Execute on Hint in the logical diagram to suggest a different execution location). You cannot move an access point node to the source side. However, you can drag an access point node to the empty diagram area, and a new execution unit will be created between the original source and target execution units in the diagram.
- **Components:** Mapping components such as joins, filters, and so on are also shown in the physical diagram.

Flow in the Physical Diagram

- Physical flow shows the flow path taken by data from the source to the target in an ODI mapping.
- The physical flow determines *where* and *how* data is:
 - Extracted
 - Transformed
 - Integratedinto the target.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

The definition of physical “flow” in the slide is the one that you will use. Remember that a mapping takes data from source datastores to targets by mapping source columns onto target columns.

However, there are several possible routes to get that data from the source to the target. So, the flow of the mapping is the actual path that the data follows. It determines how the data is extracted from the source datastores, where it is transformed, and finally how it is integrated into the target datastore.

In the example in the slide, DISTINCT appears on both the source and the target, but the one with the _AP (Access Point) suffix (revealed by hovering with the mouse) is where the actual staging and execution takes place.

Note: Having a clear understanding of the physical flow of your data enables you to prevent many problems at run time. Mastering the physical flow of your data also enables you to maximize the performance of your data integration projects.

What Defines the Flow?

Three factors:

1. Where the Staging area is located
 - On the target, on a source, or on a third server
2. How expressions, filters, and joins are set up:
 - Execution location: Source, target, or Staging area
 - Whether transformations are “active”
3. Choice of Knowledge Modules:
 - Loading Knowledge Module (LKM)
 - Integration Knowledge Module (IKM)



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

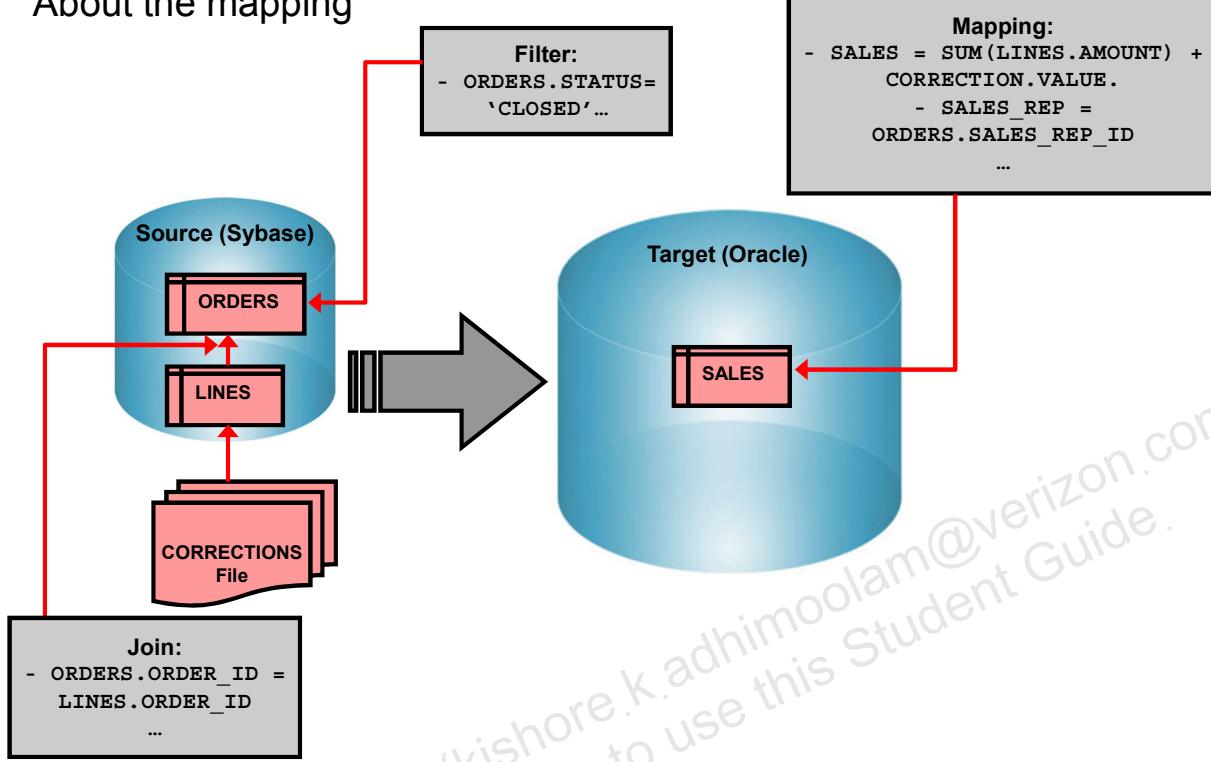
Each mapping defines its flow with three factors. These are effectively the choices that you make about how data integration occurs:

1. The most fundamental and important choice is where you put the Staging area. This temporary workspace can be placed on the target server, on a source server, or on a third server, which is neither a source nor the target.
2. Second, you can select the location of individual expressions, filters, and joins. Moving a join to the Staging area, for example, causes all source data to be transferred to the Staging area for the join. Moving it back to a source can mean that less data transfer is required. Similarly, you can mark all transformations as active or inactive. Making a transformation inactive also affects the flow of data.
3. You must select the Knowledge Modules that will generate the code to implement the integration. Different Knowledge Modules take advantage of different ways of moving data. Thus, Knowledge Modules define how data is loaded, transformed, and integrated.

Each of these three factors is examined in greater detail in the rest of the lesson.

Scenario

About the mapping



ORACLE

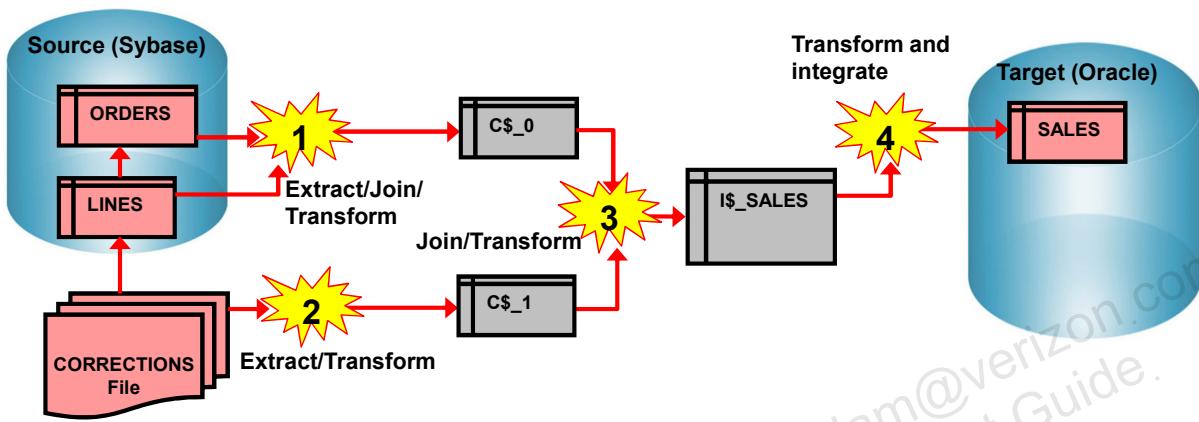
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This scenario may look familiar from previous lessons. Here, you integrate sales statistics from two tables on a Sybase source server to an Oracle target. You have one filter, two joins, and two mappings. One join links the ORDERS table to the LINES of each order. The other join updates the ORDERS table from a CORRECTIONS file. However, the actual data and how it is transformed is not very important for the purpose of this lesson.

Note: The labels “Sybase” and “Oracle” are used here for example purposes. Source and target technologies can be virtually anything, as long as there is a SQL-enabled Staging area between them.

Basic Process

Sequence of operations



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The following sequence of operations is more or less unavoidable to implement this integration. However, you have some latitude in controlling how and where they are performed.

- First, the two ORDERS tables must be extracted and joined. The constraint that orders must be closed is applied here.
- Second, data from the CORRECTIONS file must be extracted and transformed into the correct format. Now, data from these two sources must be joined and transformed into a temporary table, I\$_SALES. This temporary table looks identical to the target SALES table.
- Lastly, the data from this table must be copied to the Oracle server. You can also do a data check, or flow control, here.

Agenda

- Multiple Sources and Joins
- Filtering Data
- Overview of the Flow in ODI Mapping
- **Selecting a Staging Area**
- Configuring Expressions
- Execution Location
- Selecting a Knowledge Module



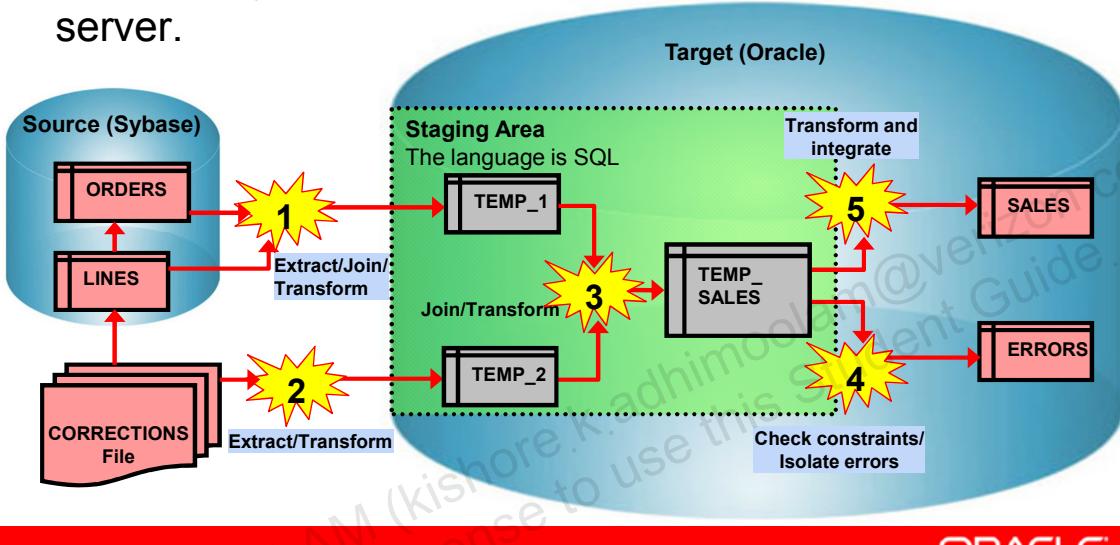
ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Now you look at the first of the three factors that structure the flow. The most important is choosing where to put the Staging area.

Purpose of a Staging Area

- A Staging area is a separate, dedicated area in an RDBMS where ODI creates its temporary objects and executes some of your transformation rules.
- You usually place the Staging area on the target data server.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Staging area is where most of the transformation and error checking is usually performed. After those tasks are performed, you just copy, or load, the data into the target server.

The Staging area is a special area that you create when you set up a database in ODI. ODI creates temporary tables in the Staging area, and uses them to perform data transformation.

You can place the Staging area on your source server, your target server, or another server altogether. However, the best place for the Staging area is usually on the target server. This gives you the greatest scope for data consistency checking, and minimizes network traffic. You now look at some of the consequences of different choices.

Placing the Staging Area

- The Staging area may be located on:
 - The Oracle target database (default)
 - A third RDBMS database server
 - The MySQL source database
- The Staging area cannot be placed on the file source data server.
 - This data server is not an RDBMS.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Important Note



- Only those schemas that are located on RDBMS technologies can act as the Staging area; files, MOM, LDAP, and OLAP databases cannot.
- When the target of the mapping is a non-RDBMS technology, the Staging area must be moved to another schema.

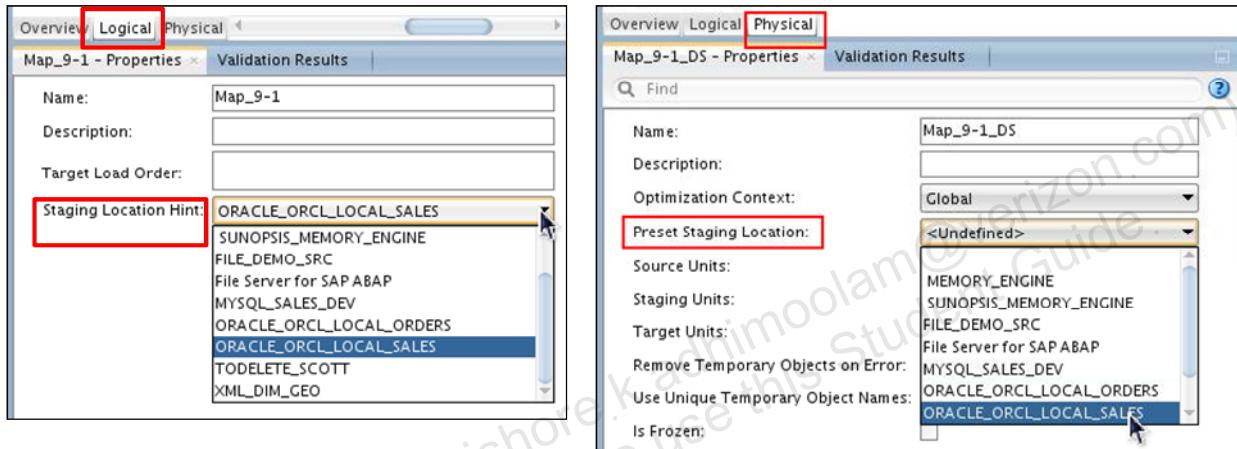
ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You must select an RDBMS technology for the Staging area. Other data servers, such as files, message-oriented middleware (MOM), Lightweight Directory Access Protocol (LDAP), or online analytical processing (OLAP) cannot be used this way. If the target of your mapping is a nonrelational technology, you must move the Staging area to another schema. Depending on your mapping, you may want to use either the source server or another server entirely.

Specifying the Staging Area

1. Click the mapping's Logical > Properties tab.
2. To place the Staging area on the a certain logical and/or physical schema, specify a hint and/or preset.
3. Click the Physical tab to see the new staging flow.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Before you select a logical schema in step 2, you must first define the logical schema in Topology Navigator.

In step 3, see how the new flow takes this change into account. Notice also how all transformations and joins are performed on the Staging area. If the Staging area is not the target, the integration to the target server becomes a simple copy.

Staging Area is now a hint in Properties, (no longer a menu choice on the map as it was in 11g). On the Logical side it is a *HINT*, on the Physical side it is a *COMMAND* (not overridable).

Agenda

- Multiple Sources and Joins
- Filtering Data
- Overview of the Flow in ODI Mapping
- Selecting a Staging Area
- **Configuring Expressions**
- Execution Location
- Selecting a Knowledge Module



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The second major factor in structuring your data flow is configuring expressions. These small changes can have a big effect on the performance of your mapping.

Options for Expressions

- Active :
 - When deselected, the expression is disabled for this mapping.
- Enable expression for update and/or insert.
 - It allows mappings to apply only to updates or inserts.
 - By default, both insert and update are enabled.
- Select the update key by selecting the Key check box.
- Change the execution location of the expression.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

There are several important options that apply to expressions.

Deselecting the Active check box disables an expression. This has the same effect as deleting it, except that you can restore it later.

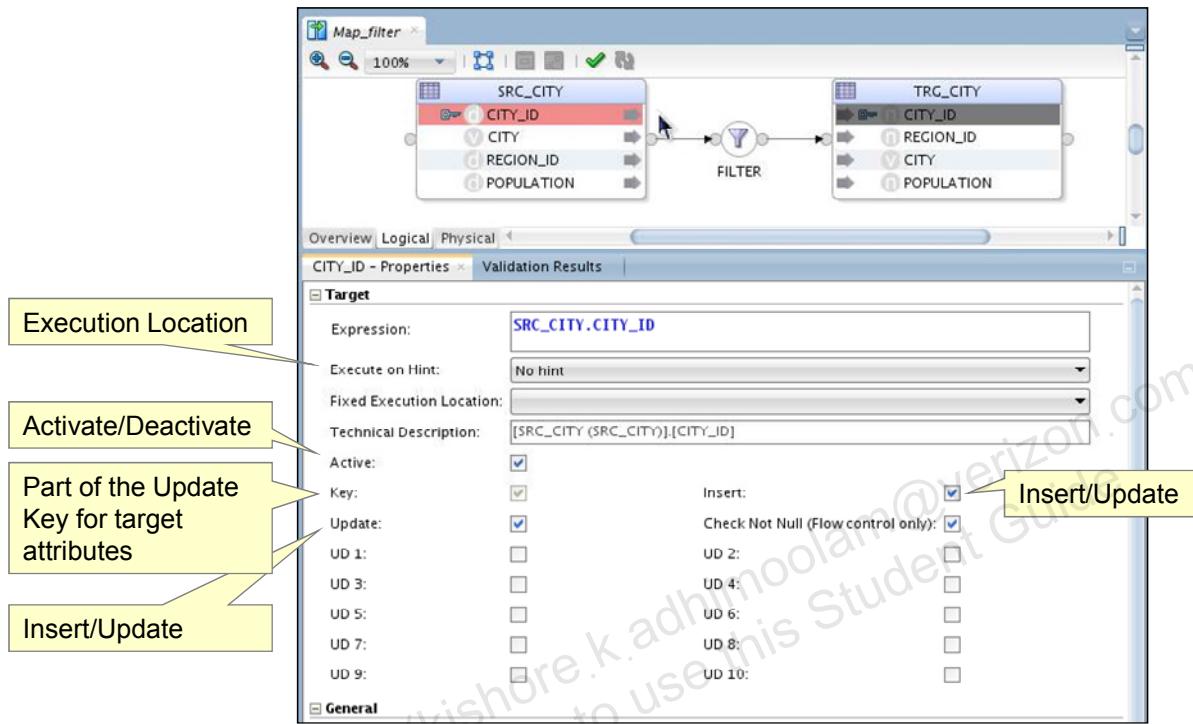
By default, the Insert and Update check boxes are enabled for expressions. This means that the target attribute receives a value when the datastore receives data by either the `INSERT` or `UPDATE` statements. However, by deselecting the Update option, you can leave the current value unmodified during `UPDATE` statements. This is useful for “creation date” fields.

Similarly, you can deselect the Insert option to always insert a `NONE` value. This may be useful for seeing attributes that have been set once, but never updated.

The update key that is used for `UPDATE` statements is a combination of a number of attributes on the target datastore. You can select the Key check box for each attribute that you want to use. This will be covered in detail in the following slides.

Lastly, you can tweak the flow of your mapping by hinting the execution location of the expression. For example, you can specify that a join be performed on the Staging area and not on the source. This is important when large amounts of data are involved. If a join returns very few rows, it is best to perform it near the source data. If a join produces many rows—for example, a cross-join or a Cartesian product—it is best to perform it later. For lookups, you can choose between executing at the source or a Staging area.

Setting Options for Expressions



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Notice where the options just described appear in the Property Inspector. These options appear on the Diagram Property tab when you select an expression.

The Active check box is below the box for the transformation's expression.

Each choice of execution location is displayed in a drop-down list in the "Execute on Hint" box.

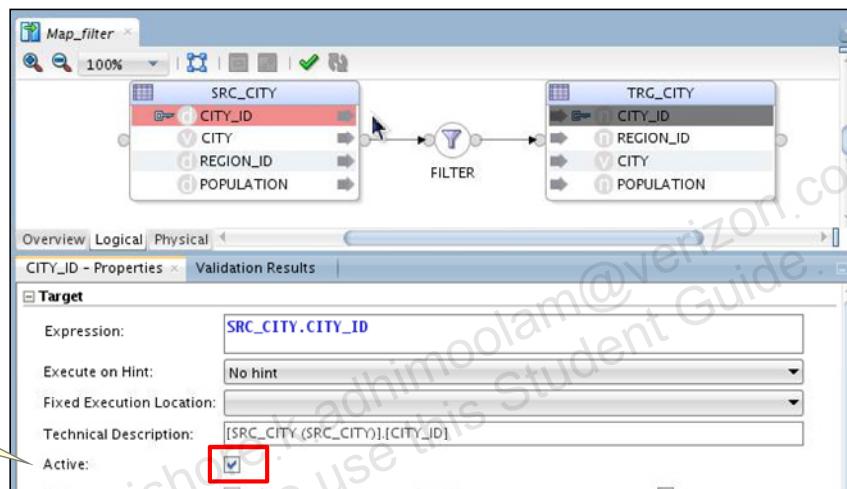
The Insert and Update options are displayed as check boxes. The other check boxes—UD1, UD2, and so on—behave in a similar way but are used by specialized Knowledge Modules.

Lastly, the check box that specifies whether a column is part of the update key is found below Active.

Options for lookups appear in the Lookup Properties panel.

Disabling an Expression

1. Open the mapping.
2. Select the attribute's expression that you want to edit.
3. Deselect Active.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To disable an expression:

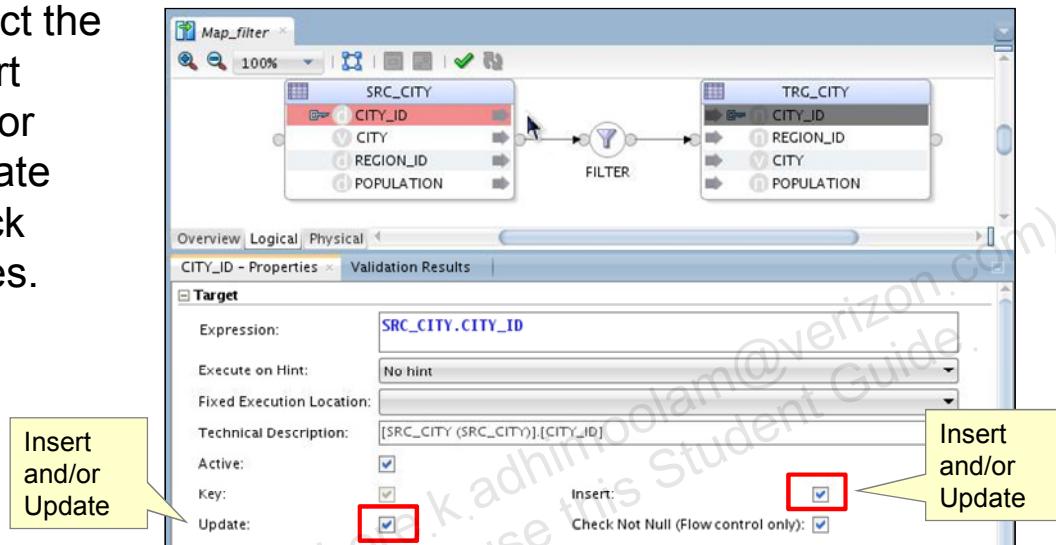
1. Open the appropriate mapping.
2. Select the expression.
3. In the Diagram Property panel, deselect the Active option that appears.

Note: Disabling an expression has the same effect as deleting it. However, ODI may not give you any warnings at design time. At run time, a number of problems can appear. For example, an inactive mapping may cause null data to be inserted into the target.

Creating a mapping with no active expressions does not work, because the Knowledge Modules assume at least one expression. You get runtime errors.

Enabling a Mapping for Inserts or Updates

1. Open the mapping.
2. Select the mapping that you want to edit.
3. Select the Insert and/or Update check boxes.



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

To enable a mapping for inserts or updates, perform the following:

1. Open the relevant mapping.
2. Select the mapping that you want to edit.
3. In the Update option group on the Diagram Property panel, select either Insert or Update. If neither option is selected, no data will be transferred. This is rarely useful.

In addition to populating a target directly from a source, you can use a variety of statements. For example, instead of using the source column `SRC_CUSTOMER.DEAR` directly, you can use the `CASE` statement to change the codes into the respective values while moving the data to the target. The value 0 in the source datastore can be transformed to 'Mr.' in the target:

```
CASE
WHEN SRC_CUSTOMER.DEAR = 0 THEN 'Mr.'
WHEN SRC_CUSTOMER.DEAR = 1 THEN 'Ms.'
WHEN SRC_CUSTOMER.DEAR = 2 THEN 'Mrs.'
ELSE NULL
END
```

Agenda

- Multiple Sources and Joins
- Filtering Data
- Overview of the Flow in ODI Mapping
- Selecting a Staging Area
- Configuring Expressions
- **Execution Location**
- Selecting a Knowledge Module



ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Now take a closer look at how to choose the execution location of your transformations.

Execution Location and Syntax

- The execution location determines the syntax and feature set that can be used in transformation rules.
- The technology of the given location is used.
 - Example: A filter executed on the source uses the syntax and feature set of the source technology.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Be aware of the consequences of selecting an execution location. In particular, the execution location determines the syntax used by transformation rules and the feature set available.

For example, for operations performed on a source schema, the syntax and functionality of that source technology are available.

This note should sound familiar because it was stated earlier: “The choice of Staging area determines the syntax used by all mappings, filters, and joins executed there.” Now you see the general principle: The technology of the location—source, Staging area, or target—of any transformation determines both the syntax to be used and the feature set available.

Note: Exercise care when changing the execution location of a transformation or moving the location of the Staging area. You must double-check that the syntax of the transformations you defined is still valid on the new configuration.

Take the example of performing a full outer join on the Staging area. When the Staging area is on Microsoft SQL Server, the full outer join works fine. However, the Staging area has now been moved onto a Hyperion SQL machine. Therefore, the full outer join is no longer valid and causes an error in ODI.

Why Change the Execution Location?

You may need to change the execution location if:

- The technology at the current location does not have the features required
 - Files, Java Message Service (JMS), and other nonrelational data sources do not support transformations.
 - A required function is not available.
- You want to achieve better performance
 - The default server is overloaded.
 - A different server has a better database engine.
- ODI does not allow this location
 - It is not possible to execute transformations on the target.



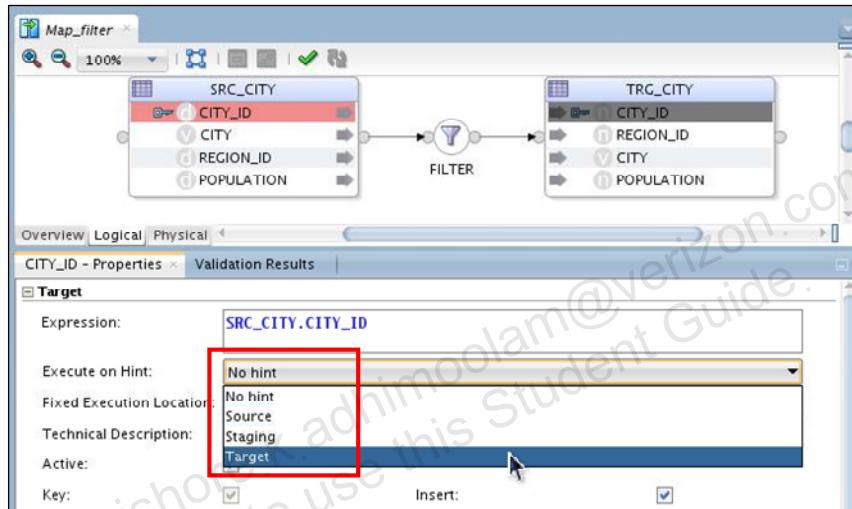
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You may want to change the execution location of a join, filter, or mapping because of performance reasons or ODI restrictions.

- If the default server is overloaded, you may want to shift as much processing onto other servers as possible.
- You can improve performance by transferring the lesser amount of data on the network (reduce the amount of data on the source, expand on the target). For instance, use substring on the source and concat on the target.
- Transformations must be performed on a source or in the Staging area. You can place the Staging area on the target server if required.

Changing the Execution Location

1. Open the mapping.
2. Select the filter, join, or mapping to edit.
3. Select an execution location from the Diagram Property panel.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

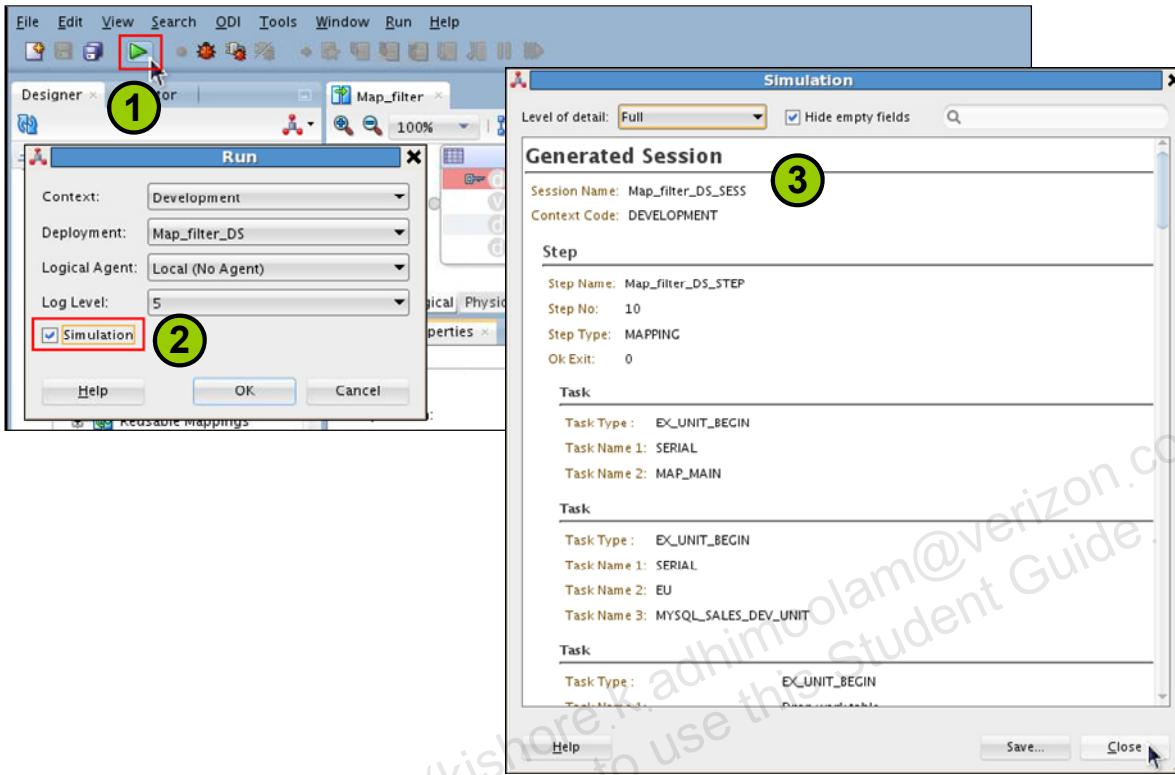
To change where a filter, join, or mapping is executed, perform the following:

1. Open the mapping.
2. Select the filter, join, or mapping that you want to modify.
3. In the Properties panel, select Source, Staging Area, or Target. Depending on the technologies, you may be unable to select one or more. If you cannot select any option, you probably need to set the transformation to Active.

All execution locations are not always possible. It is only a hint, not a command.

Make sure you also select Active Mapping.

ODI Mapping Execution Simulation



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

In Oracle Data Integrator you have the possibility at design time to simulate an execution. Simulating an execution generates and displays the code corresponding to the execution without running this code. Execution simulation provides reports suitable for code review.

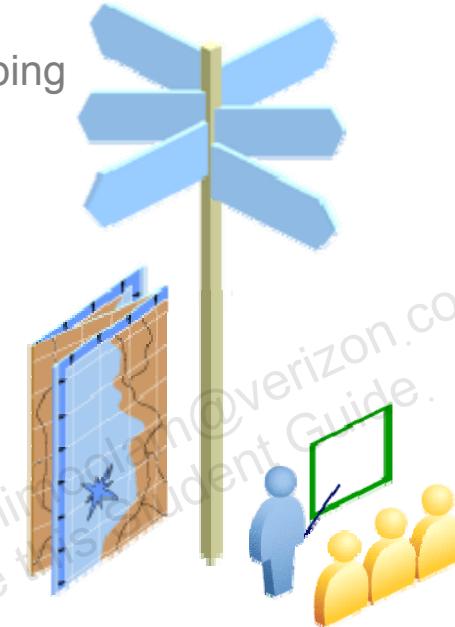
To simulate an execution:

1. In the Project view of the Designer Navigator, select the object that you want to execute, right-click, and select Execute. Alternatively, if your object is open, click the Execute button.
2. In the Execution dialog box, set the execution parameters and select Simulation.
3. Click OK. The Simulation report is displayed. You can click Save to save the report as an .xml or .html file.

Note: One limitation of Simulation is that if you have code that executes conditionally, it is likely that you will not see the results here.

Agenda

- Multiple Sources and Joins
- Filtering Data
- Overview of the Flow in ODI Mapping
- Selecting a Staging Area
- Configuring Expressions
- Execution Location
- **Selecting a Knowledge Module**



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You have learned how to set up the flow in ODI. However, to make the selected flow physically possible, you need to select the Knowledge Module (KM) appropriately. This is the final major factor in structuring your data flow.

Which KMs for Which Flow?

- If processing happens between two data servers, a data transfer KM is required.
 - Before integration (Source > Staging Area)
 - Requires a Loading Knowledge Module (LKM), which is always multitechnology
 - At integration (Staging Area > Target)
 - Requires a multitechnology Integration Knowledge Module (IKM)
- If processing happens within a data server, it is performed entirely by the server.
 - A single-technology IKM is required.
 - No data transfer is performed.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The basic questions to ask before selecting a Knowledge Module for a given step of your flow are as follows:

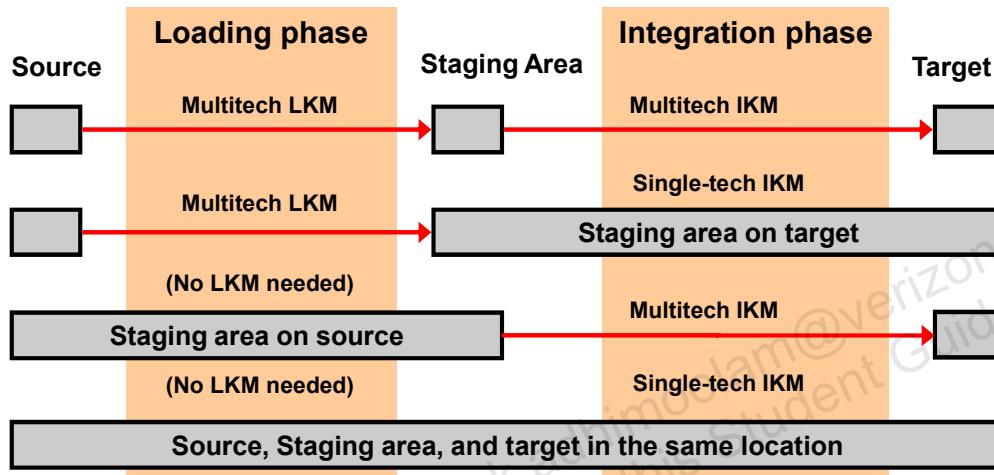
Is the processing taking place between two data servers or within one data server? In the first case, you need a Knowledge Module capable of performing data transfer. There are two types of these transfers: before integration or at integration.

- “Before integration” means processing between a source server and the Staging area. In this case, you must use an LKM specific to the two technologies, for example, an LKM that transfers data from an Oracle server to a SQL Server. Generic Knowledge Modules can do this, but a specific module is better if one exists.
- “At integration” means processing between the Staging area and the target, if these are different. You must use a multiple-technology IKM, similar to the previous situation.

If, on the other hand, the processing for a step happens within one server, the situation is simple. All the processing is performed by that data server. You just need a single-technology IKM to perform the integration.

Which KMs for Which Flow?

Four possible arrangements:



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You can now see the rules described in a visual form in the slide. Note the four possible groupings of source, Staging area, and target server. This graphic assumes that only one source server exists. Note also that the phase between the source and Staging area is called the loading phase and the phase between the Staging area and the target is called the integration phase. In the ETL model, the loading phase corresponds to “extract,” the Staging area is where “transformation” happens, and the integration phase corresponds to “load.”

- In the first arrangement, the three servers are separate. You describe the Staging area as being on a “third server.” You need a multiple-technology LKM appropriate to the technologies of the source and Staging area. You also need a multiple-technology IKM to be able to take data from the Staging area to the target.
- If your Staging area is on the target server, you need only a single-technology IKM.
- If the Staging area is on the source server, you do not need an LKM at all, but you need a multiple-technology IKM to transfer the data to the server. Note that if you have more than one source on different data servers, you need multiple-technology LKMs to collect data from the other sources.
- Lastly, if the Staging area, source, and target are all on the same data server, you do not need an LKM. A single-technology IKM is sufficient.

Knowledge Modules: Additional Information

- KMs can skip certain operations.
 - Unnecessary temporary tables are not created.
- Some KMs lack certain features.
 - Multiple-technology IKMs cannot perform flow control.
 - IKMs to File, JMS, and so on do not support static control.
- All KMs have configurable options.
- KMs can be specified as “Global,” allowing them to be shared across multiple projects.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You should know some features of working with Knowledge Modules. Knowledge Modules sometimes skip unnecessary operations, such as creating temporary tables. Also, some types of Knowledge Modules lack certain features. This may affect the way you implement your data quality control. For example, IKMs that perform data transfer cannot perform flow control. Similarly, IKMs that write to files or other nonrelational datastores cannot perform static control. Lastly, all KMs have options that can be configured to modify their behavior. For example, most LKMs have the `DELETE_TEMPORARY_OBJECTS` option. Disabling this option enables you to see the temporary tables even after a successful execution. This information is covered in detail later.

ODI 11.1.1.6 introduced Global Knowledge Modules (KMs) allowing specific KMs to be shared across multiple projects. In previous versions of ODI, Knowledge Modules were always specific to a project and could only be used within the project into which they were imported. Global KMs are listed in the Designer Navigator in the Global Objects accordion.

Identifying IKMs and LKMs

- Naming conventions:
 - LKM <source tech> to <target tech> [(<method>)]
 - IKM <tech1> [to <tech2>] <strategy> [(<method>)]
 - [to <tech2>] indicates a multiple-technology IKM.
- The technology name “SQL” stands for any SQL-enabled technology.
 - It can be used for most technologies that support the SQL-89 syntax.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Often, you have to select a Knowledge Module for a certain task. To select the Knowledge Module, you have to know how to find the right one in a list. Fortunately, the Knowledge Modules that come with ODI are named according to a system.

The name of a Loading Knowledge Module consists of the name of the source technology, the word “to,” and then the name of the target technology. If the loading method is important, it appears in parentheses afterward. For example, the module that bulk-loads files into MS SQL is called “File to MSSQL (BULK).”

For single-technology IKMs, the name is the name of the technology, followed by the integration strategy, optionally followed by the method.

For example, “SQL Incremental Update” populates a generic SQL datastore by using the incremental update method. For multiple-technology modules, both technologies are mentioned. An example is “SQL to SQL Append,” which transfers data between two different generic SQL datastores by using the append method.

When you see the word “SQL” in a Knowledge Module name, it can be used on any technology that supports the SQL-89 syntax. In many cases, a more specific Knowledge Module is preferred.

IKMs and LKMs: Strategies and Methods

- <method> is the physical means used to transfer data.
 - Database loader to perform transfers
 - Database-specific method to merge data
- <strategy> is the general technique used to integrate data into a target.
 - Append or control append
 - Incremental update
 - Slowly changing dimensions



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You saw the reference to two terms used in naming Knowledge Modules: method and strategy.

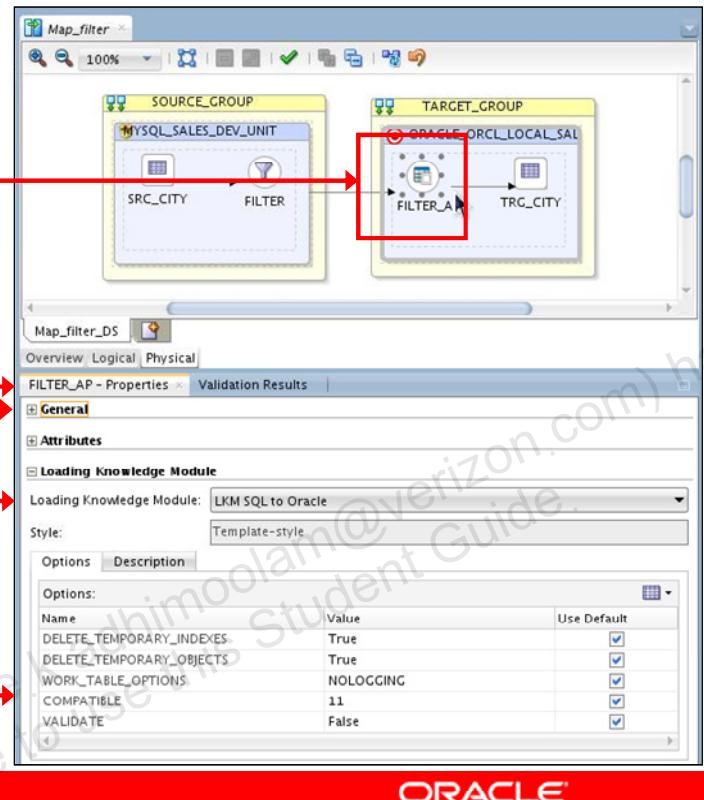
In this context, a method is a specific tool or means that ODI uses to transfer or manipulate data. It can be a bulk-loading tool supplied by a database vendor, such as SQL*LOADER for Oracle. It can also be a method contained in the database that allows special types of data transfers. For example, the LKM “MSSQL to MSSQL (LINKED SERVERS)” uses the “linked server” feature of the SQL Server to load data.

A strategy is not specific to a technology. It determines whether checks, updates, and inserts are performed.

- An “Append” is a simple strategy: Rows are inserted. A “Control Append” performs a data validity check or “flow control,” before inserting. With Control Append, you use CKM to check constraints.
- The “Incremental Update” strategy performs inserts if rows do not exist, or updates if they do. It typically also performs flow control.
- “Slowly Changing Dimensions” Knowledge Modules are more specialized. They let you define particular columns, which, if changed, trigger the addition of a new row in the datastore.

Specifying an LKM

1. Click the Properties tab.
2. Select the loading component.
The KM Property Inspector opens.
3. Change the name of the source set (optional).
4. Select an LKM.
5. Modify the LKM's options.



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

ORACLE

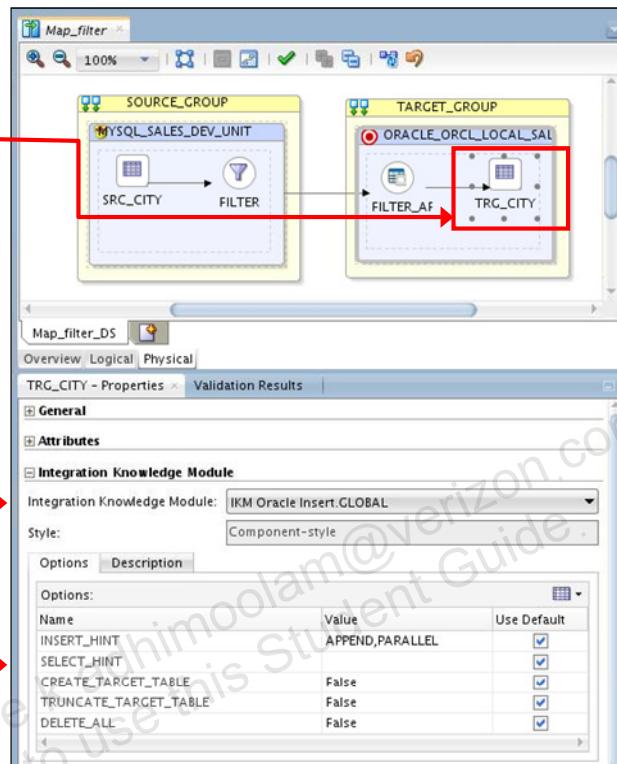
Having decided which Loading Knowledge Module to use, you need to set it in ODI. To do so, perform the following:

1. Click the Logical Properties tab in the mapping window.
2. Select the Access Point (also known as the Staging area, identifiable by the _AP suffix), for example, a source set or a filter component. A source set is a group of joined source datastores located in a server from which you want to extract the data. This appears as a box called, for example, SS_1. You now see the Knowledge Module Property panel.
3. Optionally, change the name of the source set on the Logical or Physical tab from, for example, "SrcSet0," to something more meaningful like "Source Region", for documentation purposes.
4. Now, select your LKM from the drop-down list.
5. Set the LKM's options.

Note: ODI can, in many cases, select a Default Knowledge Module. However, this may not be the best one for the job. For this reason, ODI highlights the relevant source set or datastore with an orange cross. If you have not chosen a KM, you will see a red cross (X).

Specifying an IKM

1. Click the Properties tab.
2. Select the Target set.
The KM property Inspector opens.
3. Use Distinct component, if needed.
4. Select an IKM.
5. Set the IKM's options.



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The process for specifying an IKM is similar.

1. Click the mapping's Physical tab.
2. Click the target this time. As with LKMs, the Knowledge Module's Property tab opens.
3. You may want to add the Distinct component. That means that only rows that are different (not duplicates) are manipulated. This effectively means a `SELECT DISTINCT` is used instead of `SELECT`. This does not prevent duplicates from occurring in your target data, if there are rows that exist in both source and target data.
4. Then, select an IKM from the drop-down list.
5. Configure the IKM's options as necessary. The list of KM options enables you to tweak the behavior of the KM. For example, you can recycle errors into the flow or avoid deleting the temporary tables. The next slide provides some examples.

In general, the default options behave the way you expect them to. The only ones you need to use frequently are `FLOW_CONTROL` and `STATIC_CONTROL`.

Note: The options available are entirely determined by the individual KM. For example, an IKM specific to a particular technology may have options that enable you to take advantage of the features of that technology.

Common KM Options

The following options appear in most KMs:

INSERT UPDATE	Should data be inserted/updated in the target?
COMMIT	Should the mapping commit the insert/updates? If “no,” a transaction can span several mappings.
FLOW CONTROL STATIC CONTROL	Should data in the flow be checked? Should data in the target be checked after the execution of the mapping?
TRUNCATE DELETE ALL	Should the target data be truncated or deleted before integration?
DELETE TEMPORARY OBJECTS	Should temporary tables and views be deleted or kept for debugging purposes?



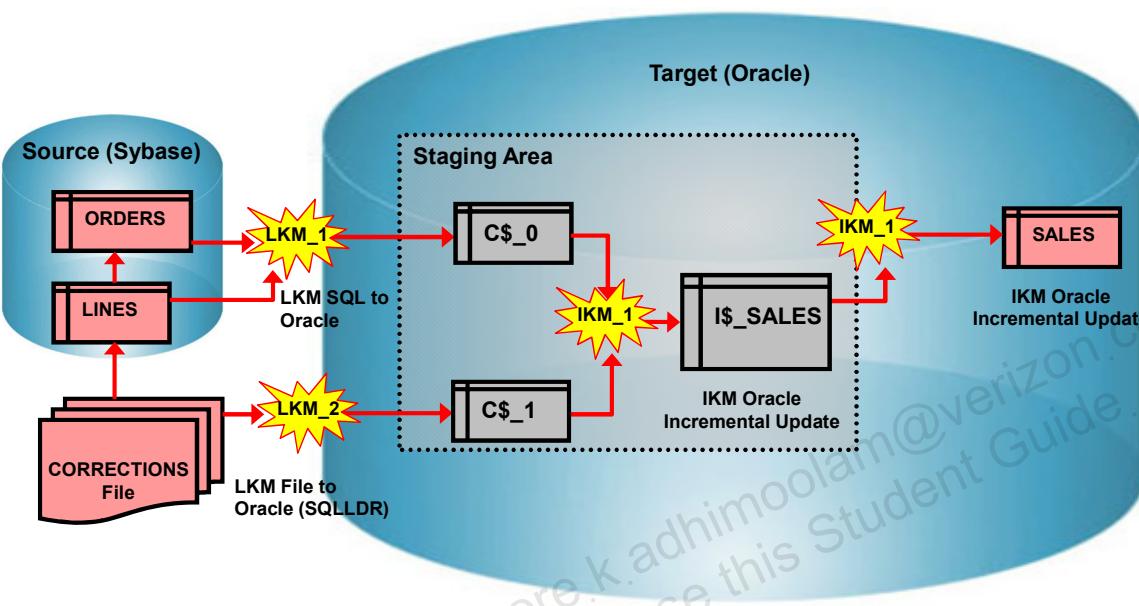
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Some of the typical options are as follows:

- **INSERT** and **UPDATE** are two options that apply to many IKMs. If you turn off **INSERT**, no rows will be added, but rows may still be updated, if applicable. Turning off **UPDATE** does the reverse. Typically, the “Incremental Update” IKMs have both options, whereas the “Append” IKMs have only the **INSERT** option.
- **COMMIT** causes ODI to generate **COMMIT** statements between inserts. Sometimes it is useful to turn this option off, to cause several steps to happen in the same transaction. One use of this is to delay the execution of triggers on Microsoft SQL Server.
- **FLOW CONTROL** specifies whether data is checked for correctness before integration. This option enforces the constraints that you defined in ODI, such as uniqueness, reference, and conditional constraints. Only single-technology IKMs have this option.
- **STATIC CONTROL** specifies whether data is checked for correctness after being loaded. This option performs almost the same checks as the “flow control” option, but on the data that has been integrated.
- **TRUNCATE DELETE ALL** and **DELETE TEMPORARY OBJECTS** enable you to specify your preferences for truncating or deleting target data and temporary objects.

Flow: Example 1

Using the target as the Staging area



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

You now return to the example scenarios to see which Knowledge Modules you should use. In this version, you have the Staging area on the target (Oracle) server.

LKM_1 transfers data from the Sybase source to the Staging area on the Oracle server. If there is a specific “Sybase to Oracle” LKM with special capabilities, you can use that. However, the standard “SQL to Oracle” LKM works fine.

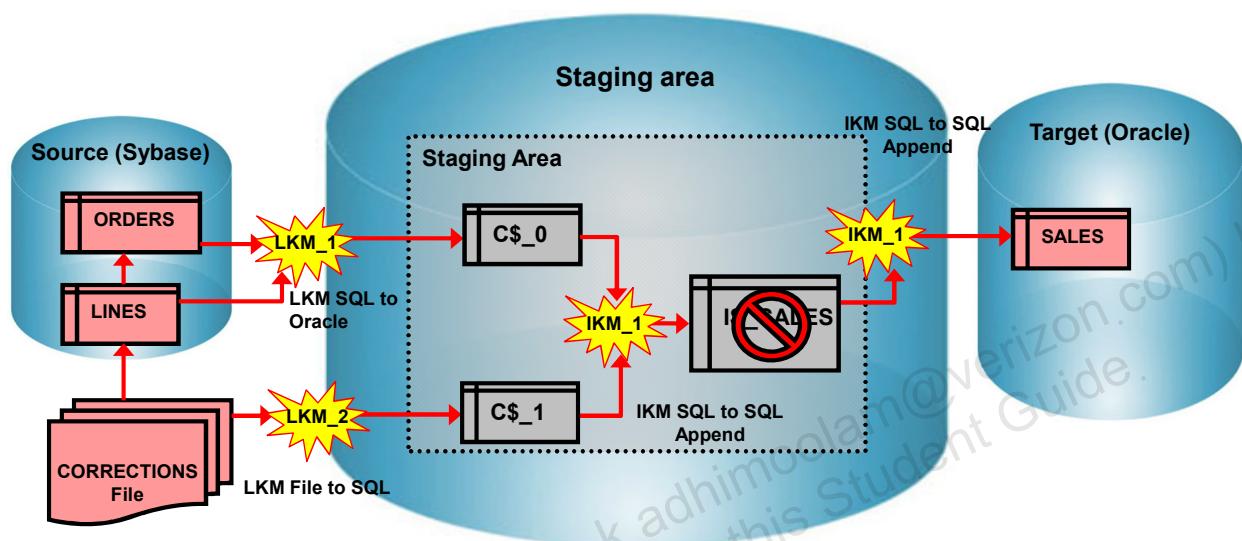
LKM_2 extracts data from the corrections file and loads it into the Staging area on the Oracle server. Here, you take advantage of the Oracle SQL*LOADER bulk-loading tool. Thus, you use the specific LKM called “File to Oracle (SQLLDR).” This is a classic example of a multiple-technology Knowledge Module—it is specific to two technologies: file data sources and Oracle.

Now that data is loaded on the Staging area, you need to integrate it. Here, because the integration happens within one server, you need a single-technology Knowledge Module.

In the scenario, you do not want to delete and repopulate the SALES table. Instead, you want to update it. Therefore, you choose the “Oracle Incremental Update” IKM, which is used twice. The first time, it joins the loaded data into a temporary table, I\$_SALES. This temporary table has a structure identical to the target SALES table. Then, it checks all constraints on the temporary table (flow control). Next, it performs the “incremental update” into the target table.

Flow: Example 2

Using a third server as the Staging area



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This is the second example flow with the same source and target. Here, you add a Staging area.

LKM_1 is no longer transferring data to the Oracle server, so you use the generic “SQL to SQL” LKM. This is still, technically, a “multiple-technology” Loading Knowledge Module: it transfers data between two technologies, which happen to be the same.

LKM_2 can no longer benefit from the Oracle bulk loader, because you are loading the file into a non-Oracle Staging area. Thus, you use the generic “File to SQL” LKM.

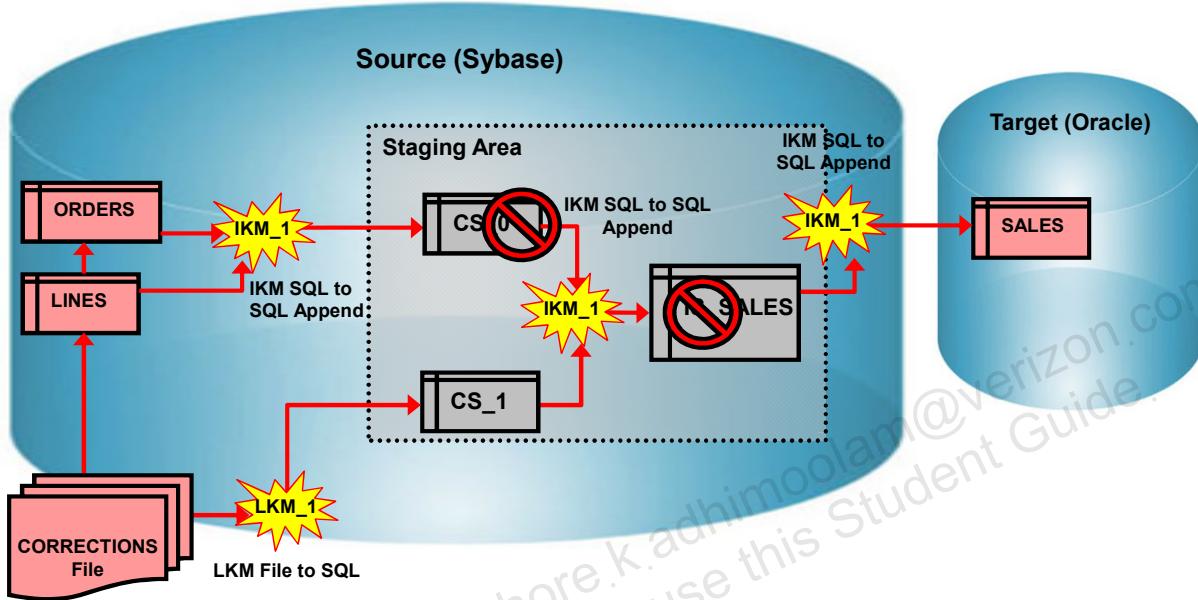
Now the integration takes place differently. It was mentioned earlier that flow control cannot be performed by multiple-technology IKMs. This is the same as saying that flow control can take place only on the target server. You can see why now: all conditions, primary keys, and foreign keys are defined on the Oracle server. In addition, the foreign keys refer to tables that exist only on the Oracle server. Similarly, you cannot test a primary key violation because all the rows that you need to test against are also on the Oracle server. This explains why it is preferable, whenever possible, to put the Staging area on the target.

Remember, flow control will not occur. Therefore, it is pointless to store data in the temporary table I\$. In this case, the temporary table will not be created.

IKM_1 now becomes a simple “SQL to SQL Append” without a check. You cannot use the incremental update that you used previously. This means that the target SALES table must be empty or contain no duplicates with the new data.

Flow: Example 3

Using the source as the Staging area



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the final configuration, the Staging area is on the Sybase source server. This location eliminates the need for a loading phase for the ORDERS and LINES tables.

Therefore, the first step now is to load the CORRECTIONS file onto the Sybase server. You use the “File to SQL” LKM to perform this step.

Then, you can immediately begin joining all the tables. You do not have to create a temporary table for the ORDERS and LINES table because they are on the same server. Therefore, the “SQL to SQL Append” IKM eliminates this step.

As in the previous case, no flow control can be performed. Because it is pointless to create a temporary integration table, this step is eliminated as well.

Thus, the actual flow becomes quite simple. The file data is loaded into a temporary table on the source server. Then, the CORRECTIONS data is joined with the ORDERS and LINES tables directly into the SALES table on the target server.

Quiz

Which of the following statements is *false*?

- a. Filters defined on the datastores in their models are automatically copied into the diagram.
- b. When you use a left outer join to combine two tables, all the rows from the left table are included in the results. ODI can generate native or standard SQL code.
- c. ODI automatically rearranges a right outer join to be a left outer join, if possible, for readability.
- d. It is possible to create a circular flow: datastore A connects to B, which connects to C, which connects back to A.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: d

Explanation:

Choice a was covered on page 9-14.

Choice b was covered on page 9-7.

Choice d is false: to create a valid flow, all source datastores must terminate in target datastores.

Summary

In this lesson, you should have learned how to:

- Design ODI mappings with multiple-source datastores
- Create joins and lookups, and filter data
- Define the flow in ODI Mapping
- Specify ODI Mapping Staging area and execution location
- Select Knowledge Modules



Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Checklist of Practice Activities

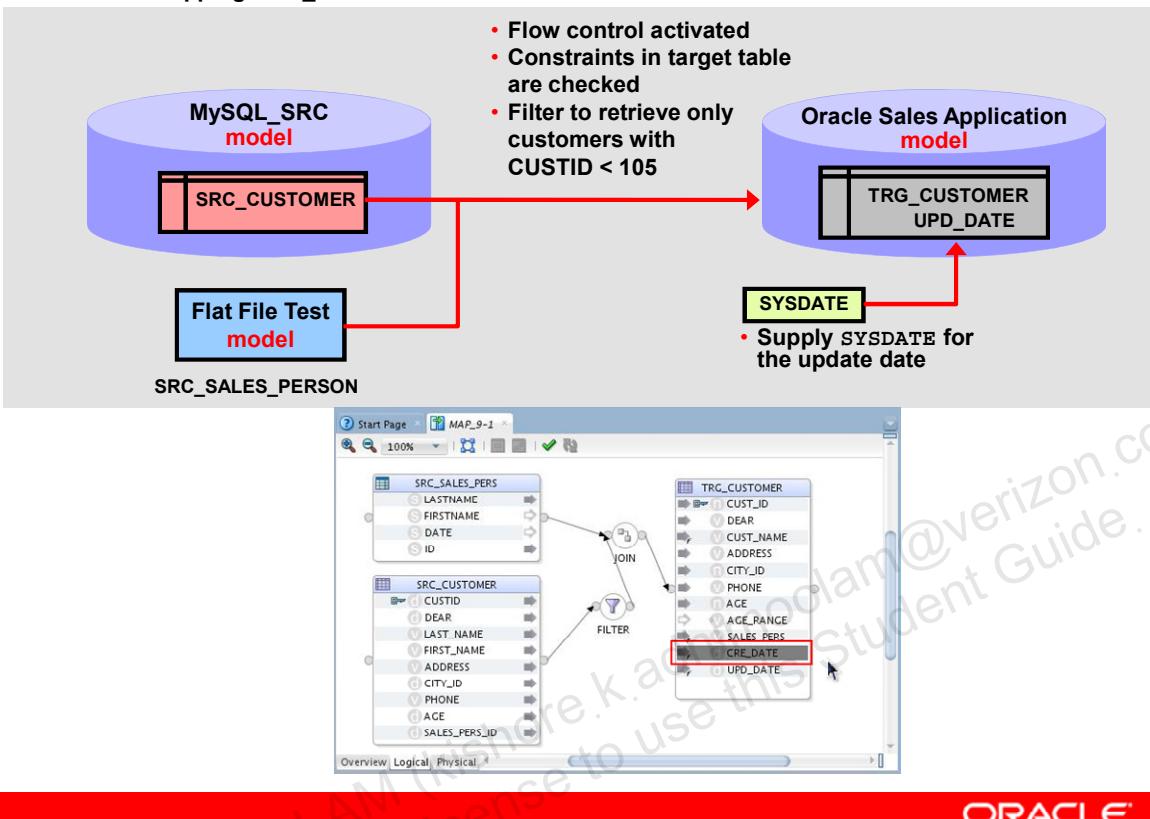
- ✓ 1-1: Logging In and Help Overview
- ✓ 2-1: Creating and Connecting to ODI Master and Work Repositories
- ✓ 3-1: Configuring Standalone Agents Using the Common Admin Model
- ✓ 4-1: Working with Topology
- ✓ 5-1: Setting Up a New ODI Project
- ✓ 6-1: Creating Models by Reverse-Engineering
- ✓ 7-1: Checking Data Quality in the Model
- ✓ 8-1: Creating ODI Mapping: Simple Transformations
- 9-1: **Creating ODI Mapping: Complex Transformations**
- 9-2: **Creating ODI Mapping: Implementing Lookup**
- 10-1: Creating ODI Mapping: Exporting a Flat File to a Relational Table
- 11-1: Using Native Sequences with ODI Mapping
- 11-2: Using Temporary Indexes
- 11-3: Using Sets with ODI Mapping
- 12-1: Creating and Using Reusable Mappings
- 12-2: Developing a New Knowledge Module
- 13-1: Creating an ODI Procedure
- 14-1: Creating an ODI Package
- 14-2: Using ODI Packages with Variables and User Functions
- 15-1: Debugging Mappings
- 16-1: Creating and Scheduling Scenarios
- 17-1: Using Load Plans
- 18-1: Enforcing Data Quality with ODI Mappings
- 19-1: Implementing Changed Data Capture
- 20-1: Setting Up ODI Security
- 20-2: Integration with Enterprise Manager and Using ODI Console

ORACLE®

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Practice 9-1: Mapping: Complex Transformations

Create mapping MAP_9-1



ORACLE

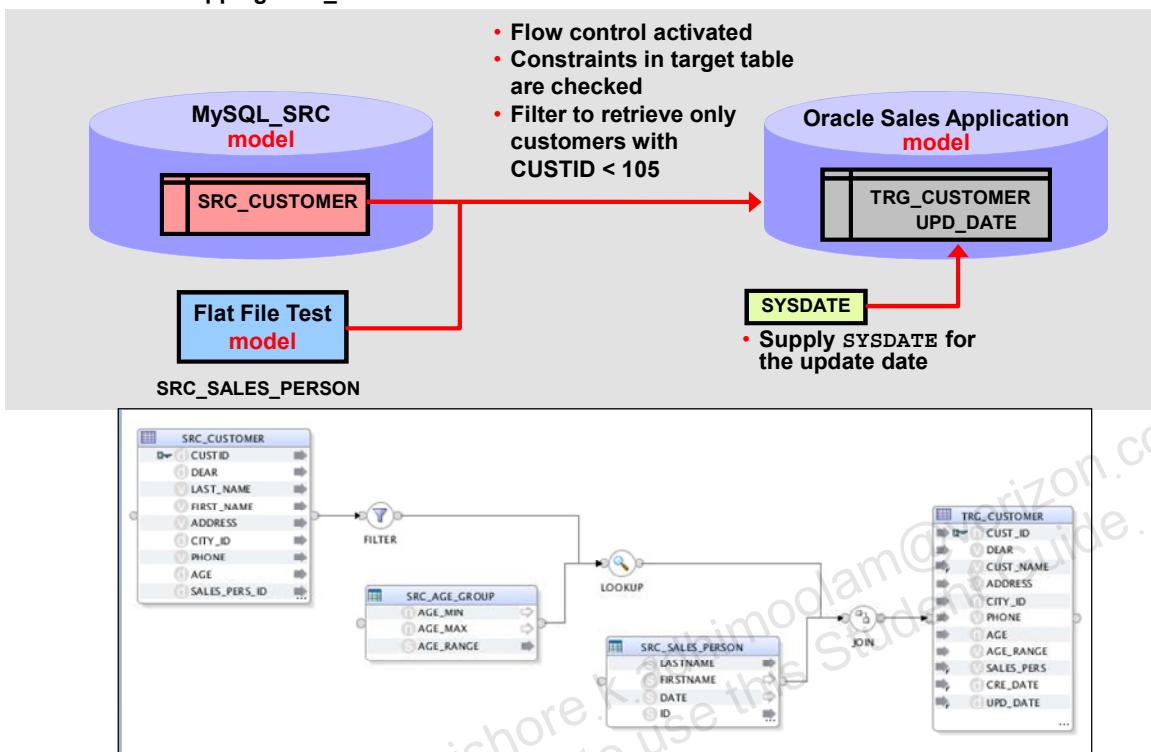
Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In previous practices, you learned how to create a simple ODI mapping.

In this practice, you create the more complex mapping MAP_9-1, loading the TRG_CUSTOMER datastore in the Oracle Sales Application model with the content of the SRC_CUSTOMER table from the MySQL_SRC model, and the content of the SRC_SALES_PERSON flat file in the Flat File Test model. In this mapping, flow control is activated, constraints in the target table are checked, and you apply filtering to retrieve only customers with CUST_ID < 105. In addition, you populate the update date (UPD_DATE) column with the System date in the mapping implementation field.

Practice 9-2: Mapping: Implementing Lookup

Create mapping MAP_9-2



ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the previous practices, you created a mapping with several sources to load the **TRG_CUSTOMER** datastore in the Oracle Sales Application model with the content of the **SRC_CUSTOMER** table and the **SRC_SALES_PERSON** files from different models. Now you implement the lookup to load data in the target according to the age range provided in the lookup table.

Unauthorized reproduction or distribution prohibited. Copyright© 2015, Oracle and/or its affiliates.

KISHORE ADHIMOOLAM (kishore.k.adhimoolam@verizon.com) has
a non-transferable license to use this Student Guide.