

Oracle Data Modeling and Relational Database Design

Student Guide – Volume I

D56497GC20
Edition 2.0
March 2012
D76443



Authors

Lauran K. Serhal
Swarnapriya Shridhar
Anupama Mandy

**Technical Contributors
and Reviewers**

Marcie Young
Philip Stoyanov
Kris Rice
Sharma Priyanka
Angelika Krupp
Bryan Roberts
Gregory Matthew
Ganesh Pitchaiah
Gerry Jurrens
Manish Pawar
Nicholas Donatone
David LaPoint
Thomas Provenzano
Michael Ritz
Timothy Trauernicht
Zhicheng Xu
Maria Billings
Charles Murray
Yanti Chang

Editors

Richard Wallis
Malavika Jinka

Graphic Designer

Seema Bopaiyah

Publishers

Nita Brozowski
Michael Sebastian Almeida

Copyright © 2012, 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

O Course Overview

Course Objectives	O-2
Agenda: Day 1	O-4
Agenda: Day 2	O-5
Agenda: Day 3	O-6
Agenda: Day 4	O-7
Oracle SQL Developer Data Modeler	O-8
Oracle SQL Developer Data Modeler Viewer	O-9
Oracle SQL Developer Data Modeler	O-10

1 Introduction to Modeling

Objectives	1-2
Why Model?	1-3
Modeling: A Practical Example	1-4
Database and Application Development Life Cycle	1-5
Process Modeling	1-6
Logical Data Modeling	1-7
Database Design	1-8
Data Type Model	1-10
Multidimensional Model	1-11
Quiz	1-13
Approaches to Modeling	1-15
Top-Down Modeling	1-16
Bottom-Up Modeling	1-17
Targeted Modeling	1-18
Quiz	1-19
Summary	1-21
Practice 1-1 Overview: Identify the Modeling Approach	1-22

2 Documenting the Business Background

Objectives	2-2
Documenting the Business Direction	2-3
Components of a Business Direction Statement	2-4
Business Objective	2-5
Assumption	2-6

Critical Success Factor 2-7
Key Performance Indicator 2-8
Problem 2-9
Devising Business Direction Objectives and Actions 2-10
Quiz 2-11
Summary 2-13
Practice 2-1 Overview: Identify Types of Business Direction Information 2-14

3 Building a Process Model (Data Flow Diagram)

Objectives 3-2
What Is a Process Model? 3-3
Benefits of Data Flow Diagrams 3-4
Components of a Data Flow Diagram 3-5
Process 3-6
External Agent 3-7
Information Store 3-8
Information Flow 3-9
Quiz 3-10
Events 3-14
Analyzing Event Responses 3-15
Quiz 3-16
Class Practice: Create a Data Flow Diagram 3-18
Summary 3-19
Practice 3-1 Overview: Create a Data Flow Diagram 3-20

4 Using Oracle SQL Developer Data Modeler to Create Your Data Flow Diagram

Objectives 4-2
Running Oracle SQL Developer Data Modeler on Linux 4-3
Running Oracle SQL Developer Data Modeler on Windows 4-4
Main Window 4-5
Setting User Preferences 4-6
Setting User Preferences: Data Modeler 4-8
Building a Data Flow Diagram 4-11
Editing the Diagram Layout 4-17
Adding and Reusing Process Events 4-21
Saving Your Model 4-22
Opening a Saved Model 4-24
Summary 4-25
Practice 4-1 Overview: Build a Data Flow Diagram in Oracle SQL Developer
Data Modeler 4-26

5 Validating Your Data Flow Diagram

Objectives 5-2
DFD Rules: Process 5-3
DFD Rules: External Agents 5-4
DFD Rules: Information Store 5-5
DFD Rules: Information Flow 5-6
Design Rules in Oracle SQL Developer Data Modeler 5-7
Quiz 5-8
Types of Processes 5-10
Primitive Process 5-11
Composite Process 5-12
Transformation Task Process 5-14
Process Decomposition 5-17
Decomposition Guidelines 5-18
Quiz 5-19
Summary 5-20
Practice 5-1 Overview: Decompose a Process in Your Data Flow Diagram 5-21

6 Identifying Entities and Attributes

Objectives 6-2
What Is a Logical Data Model? 6-3
Benefits of Creating an Entity Relationship Diagram 6-4
Components of an Entity Relationship Diagram 6-5
Entity 6-6
Entity Types 6-7
Entities and Instances 6-8
Entities Represent Sets 6-9
Quiz 6-10
Attributes 6-12
Attribute Characteristics 6-13
Class Practice: Identify Entities and Attributes 6-14
Summary 6-15
Practice 6-1 Overview: Identify Entities and Attributes 6-16
Practice 6-2 Overview: Identify Entities and Attributes 6-17

7 Identifying Relationships

Objectives 7-2
Lesson Agenda 7-3
Relationships 7-4
Components of a Relationship 7-5

Relationships: Additional Examples	7-6
Quiz	7-7
Class Practice: Define Business Rules	7-8
Relationship Types	7-9
Many-to-One and One-to-Many Relationships	7-10
Many-to-Many Relationships	7-11
One-to-One Relationships	7-12
Recursive Relationships	7-13
Quiz	7-14
Lesson Agenda	7-15
Using a Relationship Matrix	7-16
Determining a Relationship's Existence	7-18
Naming the Relationship	7-19
Determining the Relationships Cardinality	7-20
Validating the Relationship	7-22
Quiz	7-23
Class Practice: Build a Relationship Matrix	7-24
Summary	7-25
Practice 7-1 Overview: Analyze and Model Relationships	7-26
Practice 7-2 Overview: Analyze and Model Relationships	7-27

8 Assigning Unique Identifiers

Objectives	8-2
Lesson Agenda	8-3
Unique Identifier	8-4
Unique Identifier: Examples	8-5
Identifying Relationships	8-6
Identifying Relationships with Multiple Entities	8-7
Non-Identifying Relationship	8-8
Lesson Agenda	8-9
Primary and Secondary Unique Identifiers	8-10
Searching for Unique Identifiers	8-11
Quiz	8-12
Class Practice: Specify Unique Identifiers	8-13
Summary	8-14
Practice 8-1 Overview: Identify Unique Identifiers	8-15
Practice 8-2 Overview: Identify Unique Identifiers	8-16

9 Using Oracle SQL Developer DataModeler to Create an Entity Relationship

Diagram

Objectives	9-2
Lesson Agenda	9-3
Building an Entity Relationship Diagram	9-4
Specifying Logical Model General Option	9-10
Specifying Logical Model Diagram Defaults	9-11
Modifying Model Properties	9-12
Notation Types	9-13
Lesson Agenda	9-15
Editing a Diagram Layout: Moving an Object	9-16
Editing a Diagram Layout: Redrawing Lines	9-17
Editing a Diagram Layout: Moving a Relationship Line	9-18
Editing a Diagram Layout: Adding an Elbow	9-20
Editing a Diagram Layout: Showing Levels of Detail	9-21
Editing a Diagram Layout: Resizing Multiple Objects	9-22
Editing a Diagram Layout: Aligning Objects	9-24
Lesson Agenda	9-25
What Is a Subview?	9-26
Creating a Subview	9-27
Lesson Agenda	9-28
What Is a Display?	9-29
Creating a Display	9-30
Opening and Saving a Model	9-31
Exporting a Model	9-32
Importing a Model	9-33
Quiz	9-34
Summary	9-36
Practice 9-1 Overview: Build anERD in Oracle SQL Developer DataModeler	9-37

10 Validating Your Entity Relationship Diagram

Objectives	10-2
Lesson Agenda	10-3
ERD Checklist	10-4
Attribute Rules	10-6
Distinguishing Attributes and Entities	10-7
Attribute Optionality	10-9
Adding Additional Information to the ERD	10-10
Lesson Agenda	10-11
Data Modeler Reports	10-12
Generating DataModeler Reportsas RTF, HTML, or PDF Formats	10-13

Producing Data Modeling Metadata Reports	10-15
Steps to Produce Data Modeler Reports	10-16
Creating a SYSTEM Database Connectionin Oracle SQL Developer	10-17
Creating a New Userfor Reporting in Oracle SQL Developer	10-18
Creating a Connection for the New Reporting Userin Oracle SQL Developer	10-19
Exporting Your Model to the Reporting Schema in Oracle SQL Developer Data Modeler	10-20
Running Data Modeler Reports in Oracle SQL Developer	10-22
Quiz	10-26
Summary	10-28
Practice 10-1 Overview: Developand Validate Your ERD	10-29

11 Normalizing YourData Model

Objectives	11-2
Lesson Agenda	11-3
What Is Normalization?	11-4
First Normal Form (1NF)	11-5
Second Normal Form (2NF)	11-6
Third Normal Form (3NF)	11-7
Quiz	11-8
Lesson Agenda	11-9
Normalization Example: Unnormalized Data	11-10
Normalization Example: Transforming toFirst Normal Form	11-11
Normalization Example: Transformingto Second Normal Form	11-13
Normalization Example: Transformingto Third Normal Form	11-14
Summary	11-15
Practice 11-1 Overview: Normalizean ERD	11-16
Practice 11-2 Overview: Validate ERD for Normalization	11-17

12 Validating Relationships

Objectives	12-2
Lesson Agenda	12-3
Resolving M:M Relationships	12-4
Quiz	12-7
Modeling Hierarchical Data	12-8
Examining Recursive Relationships	12-9
Resolving a M:M Recursive Relationships	12-12
Quiz	12-13
Lesson Agenda	12-14
Modeling Exclusive Relationships	12-15
Creating an Exclusive Relationship inOracle SQL Developer Data Modeler	12-16

Quiz	12-18
Entity Type Hierarchies	12-19
Modeling Subtypes in Oracle SQL Developer DataModeler	12-21
Representing Entity Type Hierarchies	12-22
Changing Preferences for Box-in-Box Presentation	12-23
Quiz	12-24
Model Data over Time	12-25
Quiz	12-30
Summary	12-31
Practice 12-1 Overview: Resolve M:M Relationships	12-32
Practice 12-2 Overview: Model Hierarchical Data	12-33
Practice 12-3 Overview: Model Hierarchical Data and Recursive Relationships	12-34
Practice 12-4 Overview: Examine Exclusive Relationships	12-35
Practice 12-5 Overview: Examine Exclusive Relationships	12-36

13 Adding and Using Data Types

Objectives	13-2
Lesson Agenda	13-3
Attribute Data Types	13-4
Logical Type	13-5
Types Administration	13-6
Domain	13-7
Adding a Check Constraint to a Domain	13-8
Adding a Range or Value List to a Domain	13-9
Preferred Logical Types and Domains	13-10
Creating Domains from Logical Types	13-11
Lesson Agenda	13-12
Data Type Model	13-13
Distinct Type	13-14
Structured Type	13-15
Using Distinct Types in a Structured Type	13-16
Collection Type	13-17
Building a Data Type Model	13-18
Assigning Data Types to an Attribute	13-19
Quiz	13-20
Summary	13-22
Practice 13-1 Overview: Create and Assign Data Types	13-23

14 Putting It All Together

- Objectives 14-2
- Practice 14-1 Overview: Develop and Validate Your ERD 14-3
- Practice 14-2 Overview: Develop and Validate Your ERD (Optional) 14-4
- Summary 14-5

15 Mapping Your EntityRelationship Diagram to aRelational Database Design

- Objectives 15-2
- Lesson Agenda 15-3
- Benefits of Creating a Relational Model 15-4
- Review: Database Design 15-5
- Relational Database Overview 15-6
- Terminology Mapping 15-7
- Naming Conventions 15-8
- Naming Restrictions with Oracle Database 15-12
- Ensuring That Your Logical Data Model Is Complete 15-13
- Lesson Agenda 15-14
- Mapping Simple Entities 15-15
- Naming Entities 15-16
- Engineering Entities 15-17
- Mapping Attributes to Columns 15-18
- Mapping Attributes to Columns: Column Names 15-19
- Engineering Attributes 15-20
- Reviewing the Glossary 15-21
- Adding the Glossary as the Naming Standard 15-22
- Mapping Attributes to Columns with the Glossary 15-23
- Applying Name Abbreviations 15-24
- Mapping Unique Identifiers to Primary Keys 15-25
- Engineering Unique Identifiers 15-26
- Mapping Relationships to Foreign Keys 15-27
- Defining Naming Templates 15-29
- Applying Templates to One Table 15-31
- Applying Templates to the Relational Model 15-32
- Managing Prefixes 15-33
- Quiz 15-34
- Practice 15-1 Overview: Create an Initial Relational Model 15-36
- Lesson Agenda 15-37
- Mapping Exclusive Relationships to Foreign Keys 15-38
- Engineering Exclusive Relationships 15-39
- Mapping Subtypes to Tables 15-40
- Engineering Subtypes 15-41

Mapping Subtypes to a Single Table	15-42
Changing the FWD Engineering Strategy	15-43
Engineering Subtypes to Table per Child	15-44
Mapping Subtypes for a Table per Child	15-45
Changing the FWD Engineering Strategy	15-46
Mapping Subtypes for a Table for Each Entity	15-47
Quiz	15-48
Applying General Options	15-49
Setting Compare/Copy Options	15-50
Viewing the Mapping Comparison	15-51
Synchronizing Deleted Objects	15-52
Identifying Overlapping and Folding Keys	15-53
Summary	15-55
Practice 15-2 Overview: Forward-Engineera Model	15-56

16 Analyzing Your Relational Model

Objectives	16-2
Lesson Agenda	16-3
Relational Model and Relational Model Diagram Preferences	16-4
Reviewing Table Properties	16-5
Previewing the DDL for a Table	16-6
Preferences: Classification Types	16-7
Assigning a Classification Type to One Table	16-8
Changing the Color for Classified Tables	16-9
Changing the Prefix for Classified Tables	16-10
Assigning Classification Types to Multiple Tables	16-11
Reviewing Column Properties	16-12
Column Auto Increment	16-13
Discovering Foreign Keys	16-15
Defining a Unique Constraint	16-16
Defining Indexes	16-17
Defining a Table-Level Constraint	16-19
Table To View Wizard	16-20
View To Table Wizard	16-21
Specifying Volume Properties	16-22
Lesson Agenda	16-23
Defining Spatial Properties	16-24
Defining Column Groups	16-28
Creating Views	16-29
Quiz	16-33
Summary	16-35
Practice 16-1 Overview: Analyze Your Relational Model	16-36

17 Denormalizing Your Design to Increase Performance

Objectives 17-2
What Is Denormalization? 17-3
Storing Derivable Values 17-4
Pre-Joining Tables 17-5
Hard-Coded Values 17-6
Keeping Details with the Master Table 17-8
Repeating Current Detail with the Master Table 17-9
END_DATE Columns 17-10

CURRENT_INDICATOR Column 17-11
Hierarchy Level Indicator 17-12
Short Circuit Keys 17-13
Quiz 17-14
Summary 17-16
Practice 17-1 Overview: Denormalize Your Relational Model 17-17

18 Defining Your Physical Model

Objectives 18-2
What Is a Physical Model? 18-3
Creating a Physical Model 18-4
RDBMS Administration 18-5
RDBMS Administration: Changing the Default RDBMS Sites 18-6
Creating Physical Model Objects 18-7

Adding a User 18-10
Adding Segment Templates (Storage) 18-11
Associating Physical Objects with Your Table 18-12
Propagating Properties to Other Physical Objects 18-13
Partitioning a Table 18-14
Creating a Materialized View 18-16
Cloning a Database 18-17
Quiz 18-19
Summary 18-20
Practice 18-1 Overview: Create a Physical Model 18-21

19 Generating Your Database

Objectives 19-2
Lesson Agenda 19-3

Database Generation 19-4
Generating DDL: Selecting a Database 19-5
Generating DDL: 'Create' Selection 19-6
Generating DDL: DDL Script 19-7

Generating DDL: Assigned to Users 19-8
Generating DDL: ‘Drop’ Selection 19-9
Generating DDL: Name Substitution 19-10
Generating DDL: Including Table Scripts 19-11
Generating DDL: Masking Oracle Errors 19-12
Generating DDL: Using Find 19-14
DDL Preferences 19-15
DDL/Migration Preferences 19-18
Lesson Agenda 19-19
Design Rules 19-20
Working with Rule Sets 19-22
Working with Custom Rules 19-23
Working with Libraries 19-24
Working with Transformations 19-25
Summary 19-26
Practice 19-1 Overview: Generate DDL 19-27

20 Altering an Existing Design

Objectives 20-2
Lesson Agenda 20-3
Approaches to Modeling 20-4
Using Import to Create a Model 20-5
Importing an Existing Database 20-7
Importing Domains 20-12
Quiz 20-13
Lesson Agenda 20-14
Creating a Logical Data Model from Your Relational Model 20-15
Reviewing and Making Changes to Your Logical Model 20-16
Checking the Design Rules 20-17
Forward Engineering to a New Relational Model 20-18
Comparing Your Relational Model Changes with What Is in the Database 20-20
Compare Mapping 20-23
Previewing the DDL 20-24
Comparing and Merging Two Models 20-25
Exporting Your Model 20-29
Exporting to a Data Modeling Design 20-30
Exporting and Importing Preferences, Connections, and Recent Designs 20-31
Quiz 20-33
Lesson Agenda 20-34
Synchronizing the Data Dictionary with Changes in a Model 20-35

Synchronizing the Data Dictionary with Changes in a Model: Example	
Overview	20-36
Modifying the SALARY Column and Synchronizing Data Dictionary with the Model	
20-37	
Reviewing the Change and Setting Some Additional Options	20-38
Reviewing the Generated DDL Script	20-39
Backup Strategy Options	20-40
Running the Generated DDL Script in Oracle SQL Developer and Confirming	
the Change	20-41
Summary	20-42
Practice 20-1 Overview: Re-Engineer the HR Schema	20-43

21 Working in a Collaborative Environment

Objectives	21-2
Lesson Agenda	21-3
Collaborative Environment: An Example	21-4
What Is Subversion?	21-5
Benefits of Working in a Collaborative Environment	21-6
Creating a Subversion (SVN) Connection	21-7
Creating a Remote Directory	21-8
Adding the Design Model to the Repository	21-9
Adding the Design Model into the Repository	21-10
Making Changes to the Versioned Model	21-11
Reviewing Pending Changes	21-12
Synchronizing Changes	21-14
Resolving Conflicts Between Users	21-15
Reverting Changes	21-17
Reviewing and Comparing Versions in the Repository	21-18
Quiz	21-19
Lesson Agenda	21-21
Branching	21-22
Creating and Working with a Branch	21-23
Creating a Remote Connection and Directory in a Subversion Repository	21-24
Saving a Model in the Subversion Repository	21-25
Creating a Branch Based on the Model	21-26
Committing Changes to the Repository (Trunk)	21-27
Merging Changes into the Branch	21-28
Summary	21-29
Practice 21 Overview: Working in a Collaborative Environment	21-30

A Using SQL Developer

Objectives	A-2
What Is Oracle SQL Developer?	A-3
Specifications of SQL Developer	A-4
SQL Developer 3.1 Interface	A-5
Creating a Database Connection	A-7
Browsing Database Objects	A-10
Displaying the Table Structure	A-11
Browsing Files	A-12
Creating a Schema Object	A-13
Creating a New Table: Example	A-14
Using the SQL Worksheet	A-15
Executing SQL Statements	A-19
Saving SQL Scripts	A-20
Executing Saved Script Files: Method 1	A-21
Executing Saved Script Files: Method 2	A-22
Formatting the SQL Code	A-23
Using Snippets	A-24
Using Snippets: Example	A-25
Debugging Procedures and Functions	A-26
Database Reporting	A-27
Creating a User-Defined Report	A-28
Search Engines and External Tools	A-29
Setting Preferences	A-30
Resetting the SQL Developer Layout	A-31
Data Modeler in SQL Developer	A-32
Summary	A-33

B Creating a Multidimensional Model

Objectives	B-2
What Is a Multidimensional Model?	B-3
Measures	B-4
Measure Types	B-5
Dimensions	B-6
Sharing Dimensions	B-7
Hierarchy	B-8
Hierarchy: Example	B-10
Level	B-11
Types of Hierarchy	B-12
Attributes	B-13
Dimensional Model Summarized	B-14

Quiz	B-15
Steps to Build a Multidimensional Model in Oracle SQL Developer	
Data Modeler	B-17
Importing a Database with Dimensions	B-18
Reverse Engineering Your Model	B-21
Creating Your Multidimensional Model	B-23
Reviewing Your Multidimensional Model	B-24
Reviewing Multidimensional Object Properties	B-25
Modifying Properties for the Time Dimension	B-26
Reviewing Properties of Multidimensional Object Components	B-27
Reviewing Detailed Properties of Object Components	B-28
Creating New Multidimensional Objects	B-29
Impact Analysis	B-30
Creating an Oracle AW	B-31
Exporting the Multidimensional Model	B-32
Upgrading Your Oracle AW by Using AWM 11g	B-33
Summary	B-34

Course Overview

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Course Objectives

After completing this course, you should be able to do the following:

- Create a data flow diagram by identifying processes, external agents, information stores, and information flows that show how the information flows and how it is being transformed
- Create an entity relationship diagram by identifying entities, attributes, relationships, and constraints from a set of requirements
- Normalize the entity relationship diagram to third normal form
- Enhance the entity relationship diagram to utilize many data modeling techniques



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This course teaches the concepts required to create models that are used to generate a database. In this course, you create a process model by creating a data flow diagram and develop a data model by creating an entity relationship diagram. Then you normalize your entity relationship diagram and verify your model by utilizing many advanced techniques.

Course Objectives

(continued)

- Engineer the entity relationship diagram into an initial relational database design
- Optimize the relational database design
- Complete the physical model and generate the DDL
- Re-engineer an existing database
- Generate a multidimensional model
- Use Oracle SQL Developer Data Modeler to document all the concepts learned throughout the course



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

After your entity relationship diagram is complete, you engineer the model to create a relational database design and optimize the design for performance. In addition, you add additional components to the physical model and then generate the DDL. You also learn how to re-engineer an existing database and generate a multidimensional model.

Throughout this course, you build the models using Oracle SQL Developer Data Modeler, which automates many tasks that a data modeler needs to perform throughout the process.

Agenda: Day 1

Unit I: Getting Started

- Lesson 1: Introduction to Modeling
- Lesson 2: Documenting the Business Background

Unit II: Representing the Flow of Data by Using a Process

Model (Data Flow Diagram)

- Lesson 3: Building a Process Model (Data Flow Diagram)
- Lesson 4: Using Oracle SQL Developer Data Modeler to Create Your Data Flow Diagram
- Lesson 5: Validating Your Data Flow Diagram

Unit III: Developing a Logical Data Model

- Lesson 6: Identifying Entities and Attributes

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Day 1 provides an overview of the types of models. In addition, you start building a data flow diagram to help you understand what data is needed or transformed through various processes. By the end of the day, you will have learned how to identify entities and attributes in your entity relationship diagram.

Agenda: Day 2

- Lesson 7: Identifying Relationships
 - Lesson 8: Assigning Unique Identifiers
 - Lesson 9: Using Oracle SQL Developer Data Modeler to Create an Entity Relationship Diagram
 - Lesson 10: Validating Your Entity Relationship Diagram
- Unit IV: Utilizing Advanced Data Modeling Techniques
- Lesson 11: Normalizing Your Data Model
 - Lesson 12: Validating Relationships

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Day 2 is dedicated to building the entity relationship diagram and validating it. By the end of this day, you should have a very good understanding of what is involved in creating a data model.

Agenda: Day 3

- Lesson 13: Adding and Using Data Types
- Lesson 14: Putting It All Together

Unit V: Transforming Your Logical Model to a Relational Design

- Lesson 15: Mapping Your Entity Relationship Diagram to a

Relational Database Design

Unit VI: Evaluating Your Design for Database Creation

- Lesson 16: Analyzing Your Relational Model



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The beginning of day 3 is devoted to completing the model by adding data types to your attributes. You then build an entity relationship diagram from a new case study and use Oracle SQL Developer Data Modeler to document the model. After the model is validated, you define your strategy when the relational database design is created and perform the task in Oracle SQL Developer Data Modeler. In addition, you begin to evaluate the design.

Agenda: Day 4

- Lesson 17: Denormalizing Your Design to Increase Performance
- Lesson 18: Defining Your Physical Model
- Lesson 19: Generating Your Database

Unit VII: Other Needs for Modeling

- Lesson 20: Altering an Existing Design
- Lesson 21: Working in a Collaborative Environment

ORACLE®

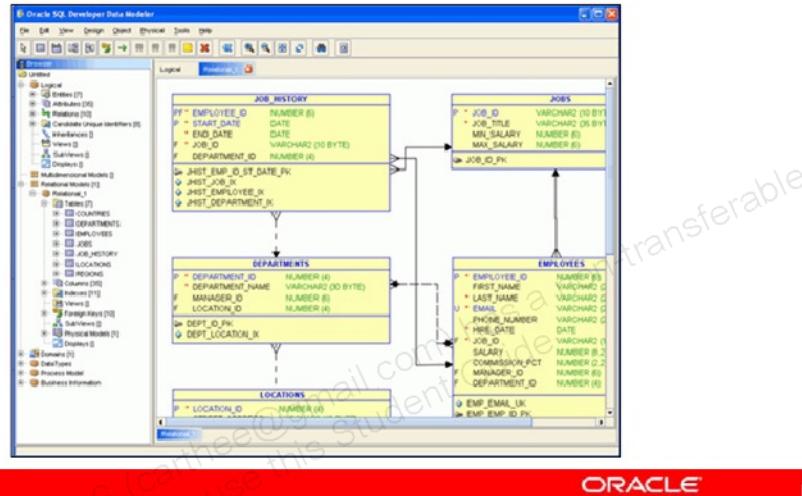
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

On day 4, you denormalize your design for performance purposes and define your physical model. At this point, you generate the DDL for your database.

Other tasks that you discuss and perform include altering an existing design and creating a multidimensional model 7

Oracle SQL Developer Data Modeler

This tool is used throughout the course to document your models.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Oracle SQL Developer Data Modeler is a graphical tool that enhances productivity and simplifies database modeling tasks. Using SQL Developer Data Modeler, users can browse, edit, and create logical, multidimensional, data types, relational, and physical data models.

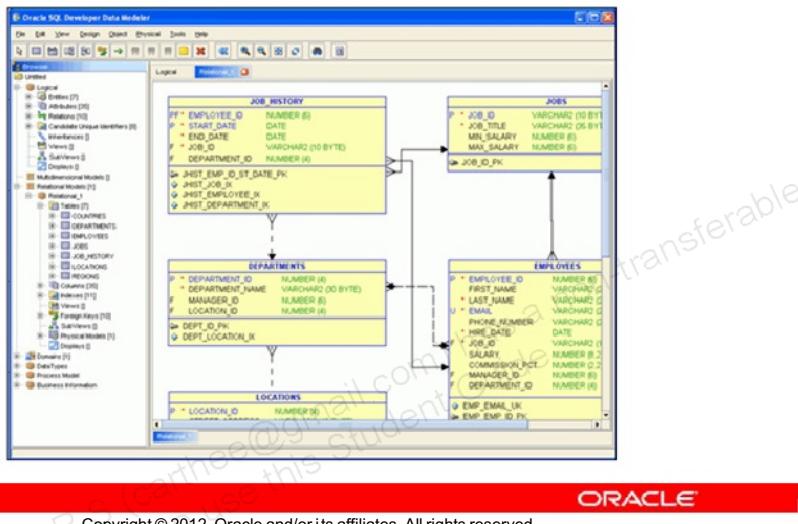
SQL Developer Data Modeler follows the Zachman Framework for defining data structures and is designed for data modelers, from business architects to DBAs and from database to application developers. The role of SQL Developer Data Modeler is to simplify data modeling development tasks and serves as a powerful communication tool between developers and business users.

Developed in Java, SQL Developer Data Modeler runs on Windows, Linux, and Mac OS X. This is a great advantage to the increasing numbers of developers who use multiple platforms. To install SQL Developer Data Modeler, simply unzip the downloaded file.

With SQL Developer Data Modeler, users can connect to all supported Oracle databases. There is also support for non-Oracle databases (IBM mainframe DB2 and IBM/UDB), Microsoft SQL Server, or a standard ODBC/JDBC driver, for selective import of database objects and data browsing and migration.

Oracle SQL Developer Data Modeler Viewer

Used by end users to view and review models



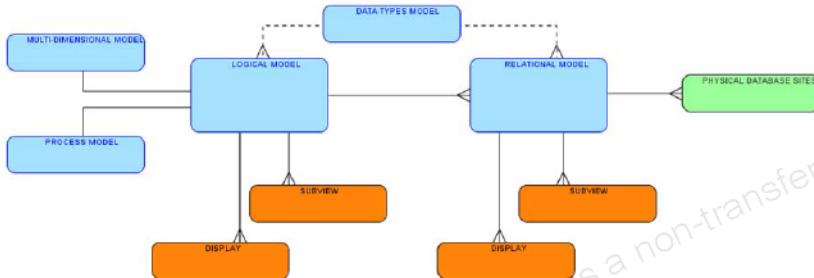
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

After the model is built, you want end users to view and review the model by using Oracle SQL Developer Data Modeler Viewer. This tool does not allow you to change any of the models—you can only view them.

Oracle SQL Developer Data Modeler

Modeling structure



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

At the core of SQL Developer Data Modeler is the logical model, which provides a true implementation-independent view of enterprise information and acts as the mediator that maps definitions in the dimensional models to different physical implementations. A logical model, or a part of it (subject area or subview), can be transformed into one or more relational models. Each relational model can have an unlimited number of physical implementations in the form of physical models (referred to as *RDBMS sites* within SQL Developer Data Modeler), with each physical model based on a supported type of database.

The process model is also available in SQL Developer Data Modeler to document the data flow diagram. In addition, you can build a multidimensional model to document the dimensions, measures, hierarchies, and so on.

I

Getting Started

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Overview

This unit includes the following lessons:

1. Introduction to Modeling
2. Documenting the Business Background



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This first introductory unit comprises two lessons. These lessons set the stage for the rest of the course. In this unit, you are introduced to terminology and a number of case studies that you use throughout the course.

1

Introduction to Modeling

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- List the reasons that modeling is important
- Describe the phases of the database and application development life cycle
- Identify the appropriate modeling approach to use for a given situation



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you examine the purpose of modeling and the reasons why it is beneficial to the overall development process.

Why Model?

Top five reasons to model:

- Easy to change
- Communication method to gather requirements
- Business rules validation
- Target user involvement
- Documentation



What other reasons can you think of?

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Remember the saying "A picture is worth a thousand words"? Pictures help people understand concepts and ideas. They help us visualize what things look like before they are completed.

Models facilitate communication between systems people and end users so that both parties can validate and confirm what the requirements are. A model is easy to change because it is just a picture rather than a fully developed system that has taken a long time to develop. Mistakes are costly, and if communication discrepancies and ideas can be fully defined in advance, the possibility of potential error later in the project can be minimized.

Models empower and provide end users with a sense of ownership because a picture is something tangible and can be used as documentation throughout the project. Often, end users do not know what the requirements are until they can see it on paper. Getting your users involved early in the project and engaged in the building and validation of the model will increase the quality and adoption of the system after it is built.

Modeling: A Practical Example

Building a house

- The architect develops the plan.
- The future homeowners approve the plan and hire a builder.
- The builder determines the timing, supplies, and people needed to build the house.



ORACLE®

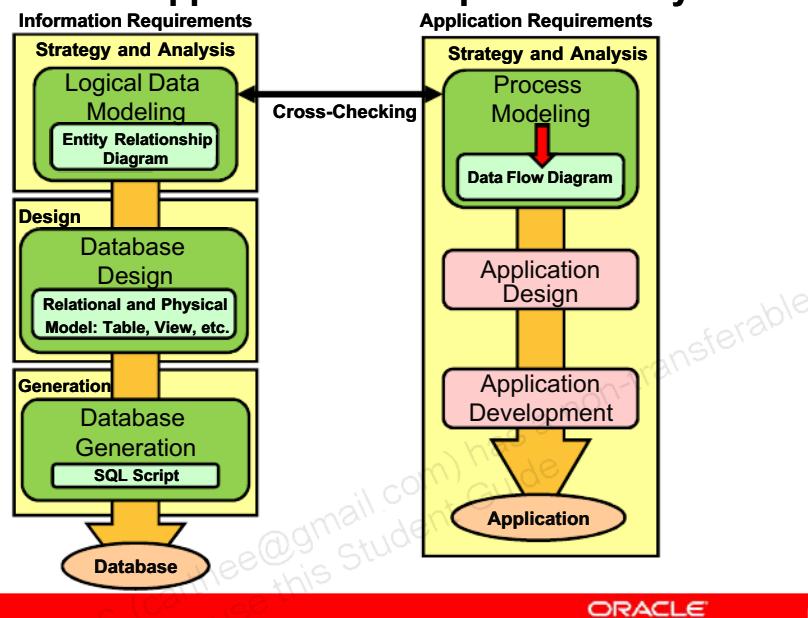
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

It would be unthinkable to build a house without a plan or blueprint. Initially, the house may only exist in the mind of the future homeowners as ideas, or as pieces of various dreams. Sometimes the future homeowners may not even know what they want, or know whether what they want is even possible. The ideas may be full of internal contradictions and impossibilities. This is not a problem in a dream world, but in the physical realm any inconsistencies and obstacles must be resolved before someone can start to construct a house.

A building contractor needs a solid plan: a blueprint of the house with an exact description of the materials to be used, the size of the roof beams, the capacity of the plumbing, and many other specifications. The builder follows the plan and has the knowledge to construct what is in the blueprint. For the blueprint to be completed, the architect works with the future homeowner to take the ideas and desires and build a model that is feasible for the building contractor to create.

The architect is trained in the skills of translating ideas into models. The architect listens to the description of the ideas and asks many questions that are then put into a diagram (the blueprint) that allows for discussion and analysis, giving advice, describing sensible options, documenting it, and confirming it with the future homeowners. This diagram provides the future homeowners with a plan of the home they want.

Database and Application Development Life Cycle



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

The database development process is tightly coupled with the application development process. Data and function cannot be treated separately.

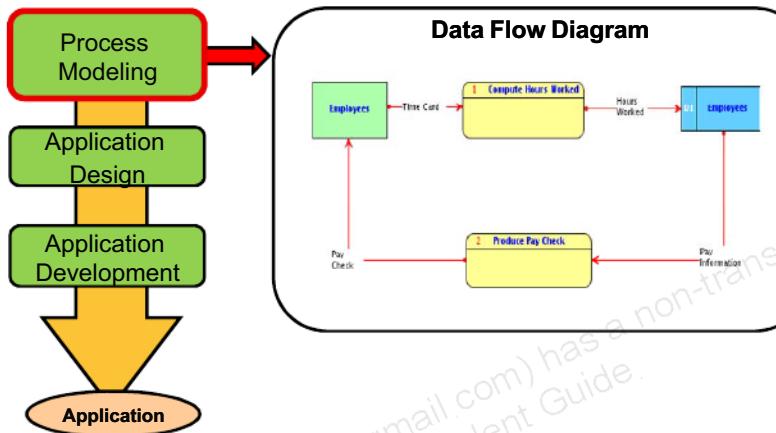
In this course, you examine how to build a logical data model in the "Strategy and Analysis" phase of the life cycle using the entity relationship diagram (ERD). To build the logical data model, it is important to know which processes use and produce the data. This is done through the process model by using the data flow diagram.

When the logical data model is complete, in the Design phase of the life cycle, you define how the model will be implemented through the relational and physical models. This is where you add the objects that will be used to create the database itself.

After the database design is complete, you generate the database SQL script that can be used to create the database.

The Application Design and Application Development phases of the life cycle are beyond the scope of this course.

Process Modeling



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Process Model is the model that is built during the "Strategy and Analysis" phase of the application development life cycle. The purpose of this model is to clarify and satisfy the needs of the business users and validate the application requirements. The model is typically built by system architects or analysts. The diagram that is used to build the Process Model is called the **Data Flow Diagram** (DFD). The basic components of a DFD include:

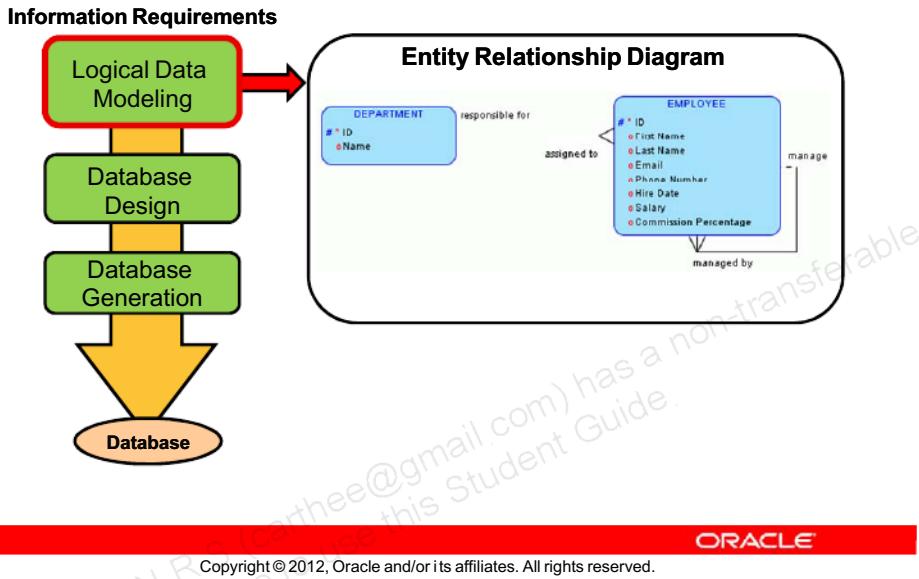
- **Process:** A discrete piece of work or function
- **External Agents:** A class of things or people that represent a source or destination of transactions
- **Data Flow:** The flow of information between objects
- **Information Store:** Data stored between processes

In the DFD in the slide, the Compute Hours Worked process begins when the time card from an employee (external agent) is received. After the process finishes, the information is stored in the information store called Employees. The Produce Pay Check process is performed

twice a month when pay checks are due. The pay information from the Employees information store is used by the process to produce the pay check that is then sent to the employee (external agent).

The logical data model and the process model are often built at the same time during the life cycle, because they both document similar business requirements. The process model helps identify the data that must be stored in the logical data model.

Logical Data Modeling



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

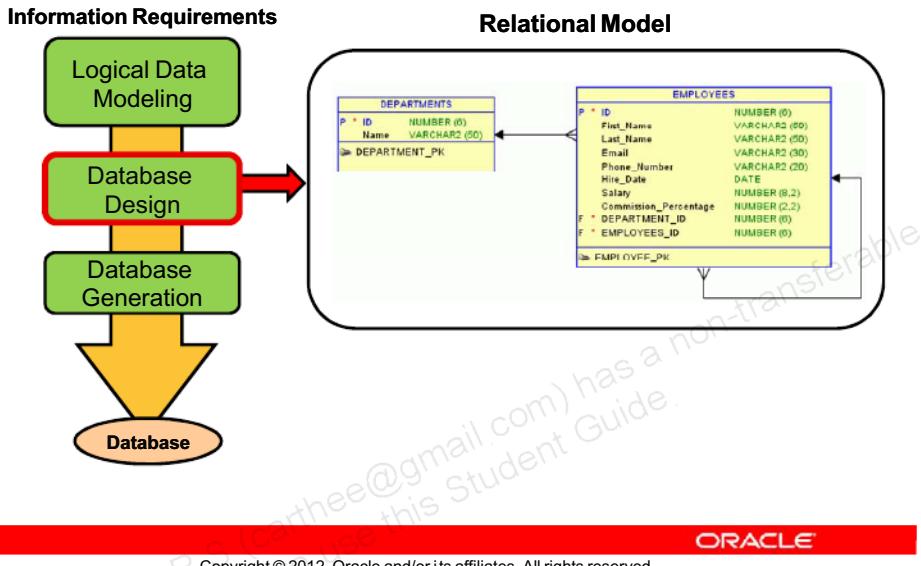
The Logical Data Model is the model that is built during the "Strategy and Analysis" phase of the database development life cycle. The purpose of this model is to clarify and satisfy the needs of the business users and validate the information requirements. The logical data model is independent of the hardware or software to be used for implementation and is typically developed by data architects or analysts. The diagram that is used to build the logical data model is called the **Entity Relationship Diagram (ERD)**. The basic components of an ERD include:

- **Entities:** Things of significance about which information must be held
- **Relationships:** How the things of significance are related
- **Attributes:** Specific information that must be held

The ERD in the slide shows two entities: DEPARTMENT and EMPLOYEE. The ERD also shows two relationships: a relationship between DEPARTMENT and EMPLOYEE and between EMPLOYEE and itself (to represent which employees manage which other employees). The names next to each relationship designate the business rules that you will examine in more detail later in the course.

Note: A logical data model is often called a *conceptual data model*. In addition, there can also be many levels of abstraction at this phase of the life cycle. For example, the ERD could show Human Resources, Operations, and Manufacturing as entities with relationships between them. Each entity would then have a separate lower-level ERD that describes the entities and attributes for that particular level in the hierarchy.

Database Design



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Database Design, often called the *Relational Model*, is the model that is built during the Design phase of the database development life cycle. The purpose of this model is to describe the database objects that must be created when the database is generated. This model is typically built by a database administrator. The basic components of a database design include objects such as relational tables, columns, and primary and foreign keys. The database design maps to the objects in the logical data model.

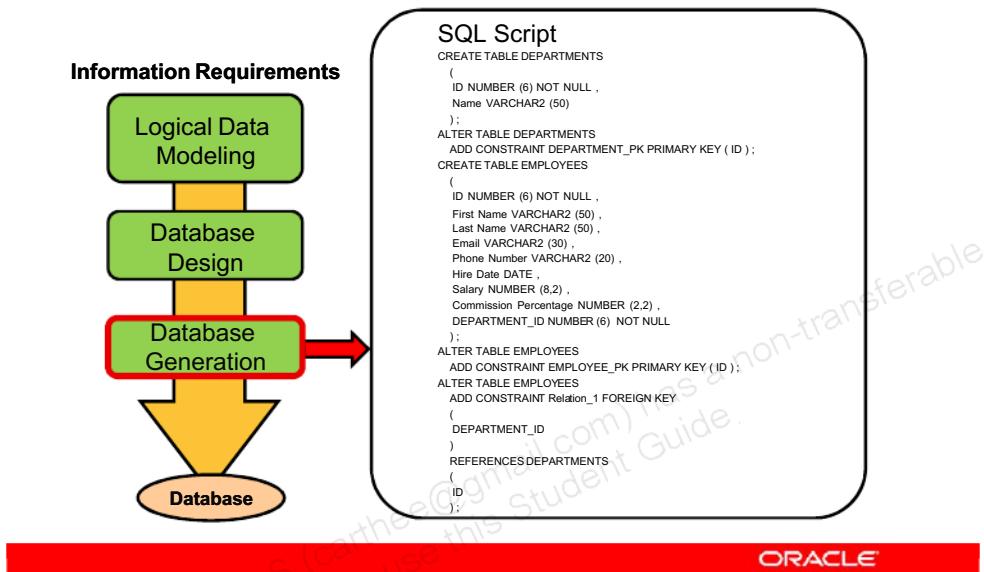
In the relational model in the slide, there are two tables: DEPARTMENTS and EMPLOYEES. Each table has a primary key. The EMPLOYEES table has two foreign keys, one for MANAGER_ID and one for DEPARTMENT_ID.

Some characteristics to keep in mind:

- The values of a column are atomic.
- Each row in a table is unique.
- Each column value has the same data type.
- The sequence of rows and columns is insignificant.
- Each column has a unique name.

Note: These characteristics are discussed in more detail in a later lesson.

Database Generation



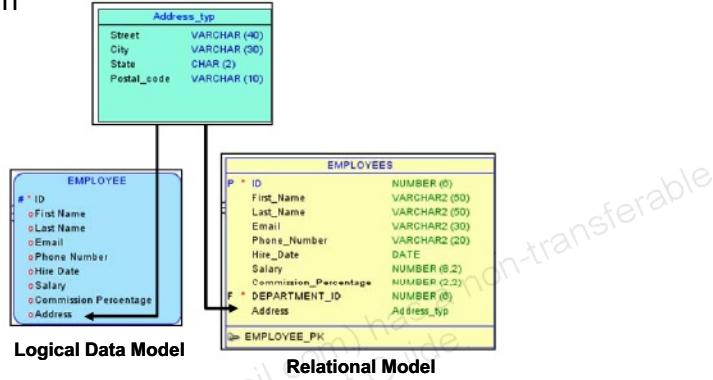
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Database Generation process involves completing the physical model and generating the SQL script that will contain the SQL statements necessary to create the database. The purpose of the physical model is to describe a database in terms of Oracle Database objects (tablespaces, tables, views, triggers, and so on) that are based on a relational model. Each relational model can have one or more physical models. Each physical model is based on an RDBMS site object. An RDBMS site is a name associated with a type of database supported by data modeling, such as Oracle Database 11g.

Throughout this course, you use Oracle SQL Developer Data Modeler to generate the SQL script that you can use to generate your database. In the slide is a portion of the SQL script that was generated for the DEPARTMENTS and EMPLOYEES tables.

Data Type Model

User data types that can be used in a logical data model or relational design



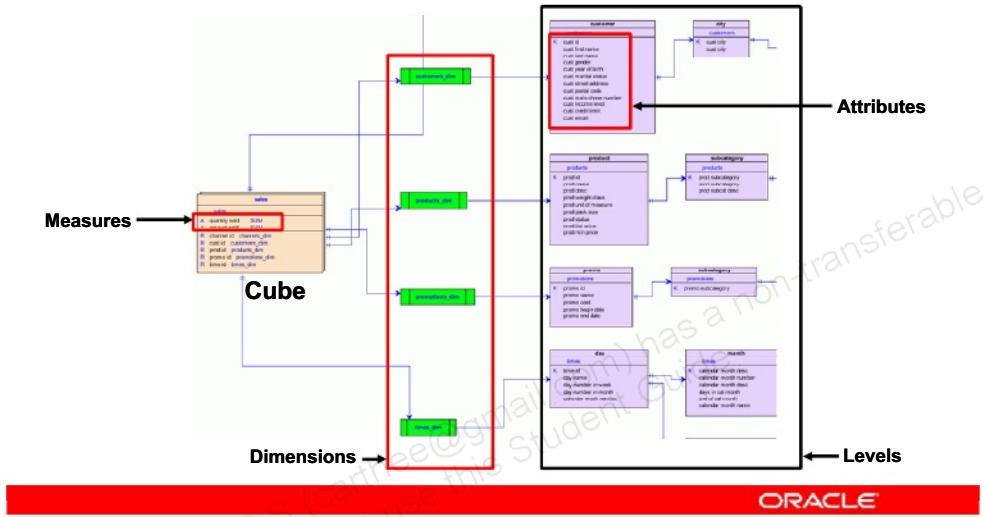
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The **Data Type Model** allows modeling of SQL99 structured types, which can be used in the **Logical Model** and in **Relational Models** as data types. The slide shows how you define a structured type in the data type model called **Address_typ** with a set of attributes. You then add an Address attribute to an entity in your logical data model and select the **Address_typ** structured data type for the type. You can also add a column to a table in your relational model and add the **Address_typ** structured data type to it.

Multidimensional Model

A model of business activities in terms of facts and dimensions



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

A Multidimensional Model is a model of business activities in terms of facts and dimensions.

The components of a multidimensional model include the following:

- **Cube:** Corresponds to a single fact table or view
- **Measures:** Populate cells of a cube with the facts collected about business operations. Measures are organized by dimensions, which typically include a Time dimension.
- **Dimensions:** Contain a set of unique values that identify and categorize data. They form the edges of a cube, and thus of the measures within the cube. Because measures are typically multidimensional, a single value in a measure must be qualified by a member of each dimension to be meaningful. For example, the Sales measure has four dimensions: Time, Customer, Product, and Channel. A particular Sales value (43,613.50) has meaning only when it is qualified by a specific time period (Feb-06), a customer (Warren Systems), a product (Portable PCs), and a channel (Catalog).
- **Hierarchy:** Organizes data at different levels of aggregation. In viewing data, analysts use dimension hierarchies to recognize trends at one level, drill down to lower levels to identify reasons for these trends, and roll up to higher levels to see what effect these trends have on a larger sector of the business.

- **Level:** Represents a position in the hierarchy. Each level above the base (or the most-detailed) level contains aggregate values for the levels below it. The members at different levels have a one-to-many relationship. For example, Q1-10 and Q2-10 are the children of 2010, so 2010 is the parent of Q1-10 and Q2-10.
- **Attribute:** Provides additional information about the data. Each attribute typically corresponds to a column in a dimension table or view. Examples of attributes are colors, flavors, and sizes. These attributes can be used for data selection and for answering questions such as:
 - Which colors were the most popular in women's dresses in the summer of 2010?
 - How does this compare with the previous summer?

Time attributes can provide information about the Time dimension that may be useful in some types of analysis, such as identifying the last day or the number of days in each time period.

The graphic in the slide depicts a multidimensional model with a cube, dimensions, and levels.

Quiz

Which model is used to define the business information requirements?

- a. Process model
- b. Data Type model
- c. Logical Data model
- d. Relational model

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Quiz

Which model is used to define the database objects?

- a. Process model
- b. Data type model
- c. Logical data model
- d. Relational model

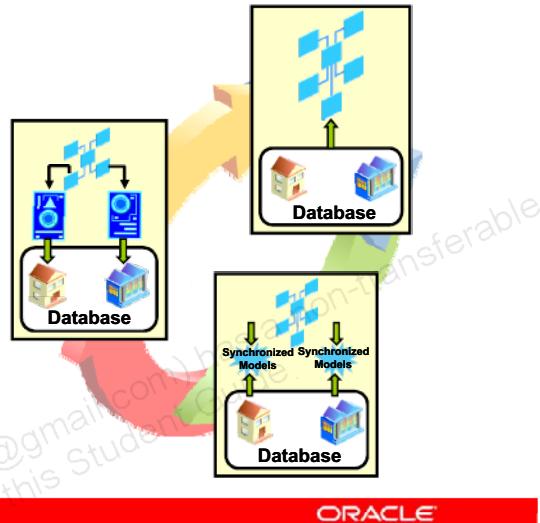
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: d

Approaches to Modeling

- Top-down modeling
- Bottom-up modeling
- Targeted modeling



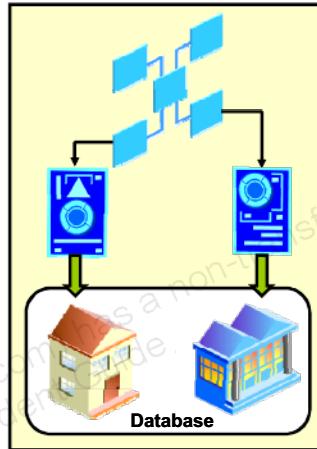
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The slide lists the three approaches to modeling. Each approach meets a different need, so you may be involved in projects that use different approaches. In this course, you focus most on top-down modeling, which involves building a model from scratch and following it through to completion. Later in this course, you examine bottom-up and targeted modeling.

Top-Down Modeling

Designing a new database:

1. Business information
2. One process model (DFD)
3. One logical data model (ERD)
4. One multidimensional model
5. One or more relational models
6. One or more physical models for each relational model



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Top-down modeling is used for designing a new database. Top-down modeling gathers information about business requirements, and then proceeds to define the following:

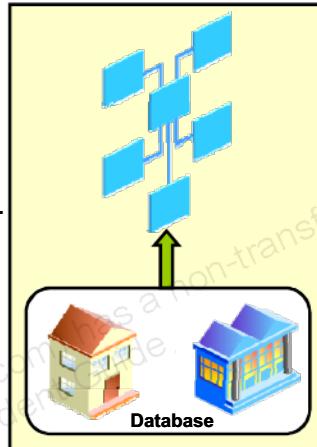
- Processes
- A logical model of the data
- One or more relational models
- One or more physical models for each relational model

The steps and information requirements can range from simple to elaborate, depending on your needs. Top-down modeling can involve the steps shown in the slide, but you can abbreviate or skip steps as appropriate for your requirements.

Bottom-Up Modeling

Modifying an existing database:

1. Produce a Relational model.
2. Modify the Relational model and create additional Relational models.
3. Reverse-engineer the Logical model from the Relational model.
4. Modify and check the design rules for the Logical model.
5. Generate the modified DDL code.



ORACLE®

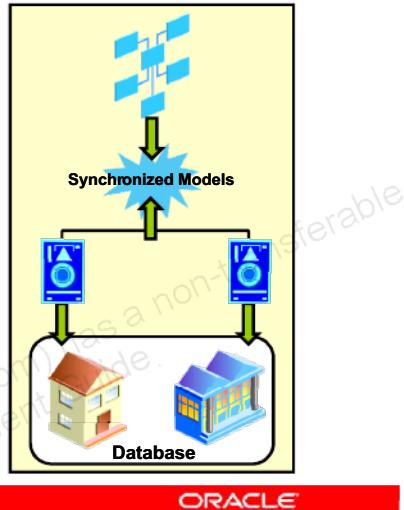
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Bottom-up modeling approach modifies an existing database definition. Bottom-up modeling builds a database design based on either metadata extracted from an existing database or a file with DDL code that implements an existing database. The resulting database is represented as a relational model and a physical model, and you reverse-engineer the logical model from the relational model. Bottom-up modeling can involve the steps listed in the slide, but you can abbreviate or skip some steps as appropriate for your requirements.

Targeted Modeling

Maintaining existing models by adapting to new requirements:

1. Change the Logical Data model.
 2. Engineer to modify the Relational model.
- ↑ ↓
2. Engineer to modify the Logical Data model.
 1. Change the Relational Data model.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Targeted Modeling approach maintains the existing models by adapting to new requirements. Depending on the requirements, you must determine which model to modify.

If the new request is to add or modify a new business requirement, you typically modify the Logical Data model and then forward-engineer the change to the Relational model to synchronize the models.

If the new requirement is to add or modify an existing database definition, you typically modify the Relational model and reverse-engineer to synchronize with the Logical Data model.

Quiz

Mary, an analyst in the HR department at XYZ Corporation, has been contacted because there is a new mandate to maintain information about employees' dependents. Which modeling approaches could Mary use?

(Choose all that apply.)

- a. Top-down approach
- b. Bottom-up approach
- c. Targeted approach
- d. None of the above

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b, c

The answer depends on the model.

If the model already exists, the answer is (c). If there is no existing model, the answer is (b).

Quiz

Tom, the Finance Manager at ABC Incorporated, implemented Oracle E-Business Suite Financials in the last six months.

ABC would like to review its business rules and possibly make some changes. Which modeling approach should Tom use?

- a. Top-down approach
- b. Bottom-up approach
- c. Targeted approach
- d. None of the above

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary

In this lesson, you should have learned how to:

- List the reasons that modeling is important
- Describe the phases of the database and application development life cycle
- Identify the modeling approach to use for a given situation



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you learned why modeling is important and which models to create at each phase of the database and application development life cycle. You also learned about the three approaches to modeling and the business situations in which each approach is most appropriate.

Practice 1-1 Overview: Identify the Modeling Approach

This practice covers identifying:

- The models that must be developed
- The modeling approach to use

In this practice, you read the Starlight DVD case study and then discuss the models that must be developed and the appropriate modeling approach to use.

Documenting the Business Background



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Define and identify business objectives, assumptions, critical success factors, key performance indicators, and problems
- Establish business direction objectives



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you learn about the various components of a business direction statement and about establishing business direction objectives.

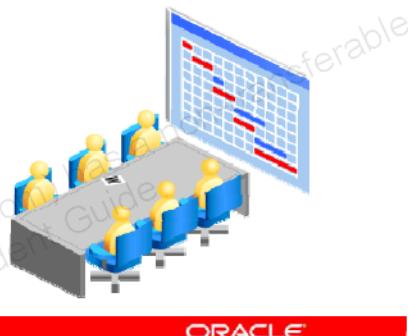
Documenting the Business Direction

Goals

- Establish an understanding of business goals.
- Set priorities for information systems development.
- Identify high-level requirements definition.

Audience

- Senior management
- Analysts and business users



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The project team must have a very clear understanding of the business background before the detailed analysis phase of a project can begin. The business direction statement is part of the high-level definition of requirements that is used to set priorities for information systems development that will ultimately achieve the business goals.

The people involved with setting the business direction are senior management and/or analysts and business users who understand the details of how their departments function. There are many ways to gather the information to construct and document the business direction. Some useful ways include: workshops, facilitation sessions, and one-on-one interviews.

Components of a Business Direction Statement

- Business objectives
- Assumptions
- Critical success factors
- Key performance indicators
- Problems



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The components that make up a business direction statement include:

- **Business objective:** A statement of business intent that can be measured quantitatively
- **Assumption:** Any assumption that can be made about a system, specifically those that can impact the design and implementation of tables
- **Critical success factor:** Any business event, dependency, deliverable, or other factor whose non-attainment would seriously impair the likelihood of achieving a business objective
- **Key performance indicator (KPI):** An indicator that quantifies or monitors the progress that is made toward achieving a business objective
- **Problem:** A business event or state that can inhibit the progress of the enterprise toward its objectives

Business Objective

- A statement of business intent that can be measured quantitatively
- **Example:** Increase membership fee revenue by 30% by offering corporate memberships to local companies.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When defining a business objective, you must be able to quantify it so that you know it has been achieved when complete. In the example in the slide, the business objective is met when the membership fee revenue has increased by 30%. In addition, you can specify how to achieve the objective by offering corporate memberships to local companies.

Assumption

- A statement made about a system that can affect the design and implementation of the result
- **Example:** Oracle Database 11g release 1 will be used on Linux.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Assumptions are statements that define the scope of the system. Assumptions can impact the design and implementation of the system. For example, if your organization has been mandated to run on Oracle Database only in a Linux environment, this would be an assumption you must make.

Critical Success Factor

- A factor whose non-attainment would seriously impair the likelihood of achieving a business objective
- **Example:** Our pricing strategy must be aggressive against our competitors.



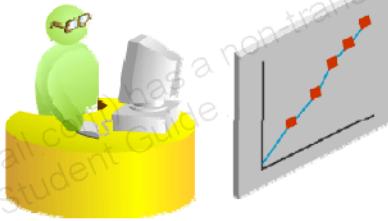
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Critical success factors are business events, dependencies, deliverables, or other factors whose non-attainment would seriously impair the likelihood of achieving a business objective. The example in the slide indicates that if the pricing strategy is not aggressive when compared to competitors, the business objective may not be met.

Key Performance Indicator

- An indicator that quantifies or monitors the progress toward a business objective
- **Example:** Rate of upgrade to silver and gold memberships must be at least 15% per month.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Key performance indicators monitor the progress of a business objective so that you know it is being achieved. In the example in the slide, silver and gold memberships must increase by at least 15% per month to achieve the business objective of increasing membership fee revenue by 30%.

Problem

- A business event or state that can inhibit the progress of the enterprise toward its objectives
- **Example:** Store clerks cannot easily identify DVDs that are seriously overdue.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Problems are identified so that they can be addressed and resolved during the life cycle of the project.

Devising Business Direction Objectives and Actions

Find associations to create business direction objectives and the actions that must take place to achieve the desired results.

Business Direction Objective	Action to Achieve Results
Increase membership levels by 15% monthly.	Stock a wide range of DVDs and sufficient copies.
Increase membership fee revenue by 30% through corporate offerings.	Promote new corporate memberships at a discount.
Increase membership fee revenue by 20% through upgrade offerings.	Promote the advantages of membership upgrades.
Reduce overdue DVDs by 3%.	Notify the customer immediately when a DVD is overdue.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Reviewing all the business direction components allows you to find associations between them to devise business direction objectives. These objectives can then be analyzed to find actions that can be taken to achieve the objective. In the example table in the slide, you see that the business direction objective to increase membership levels by 15% monthly can be accomplished by stocking a wide range of DVDs and sufficient copies from which a customer can choose, because the likelihood that a customer will rent a DVD increases if a good selection is available.

Later in this course, you specify these objectives as documents for each object in your logical data models (ERDs) and process models (DFDs) in Oracle SQL Developer Data Modeler.

Quiz

XYZ Corporation must make sure that its salaries are competitive in the industry to retain its employees. Which of the following business direction components does this represent?

- a. Business objective
- b. Critical success factor
- c. Key performance indicator
- d. Problem

ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Quiz

Annual employee retention levels should be greater than 75%. Which of the following business direction components does this represent?

- a. Business objective
- b. Critical success factor
- c. Key performance indicator
- d. Problem

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary

In this lesson, you should have learned how to:

- Define and identify business objectives, assumptions, critical success factors, key performance indicators, and problems
- Establish business direction objectives



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you learned about the different components of a business direction statement and how to differentiate among them.

Practice 2-1 Overview: Identify Types of Business Direction Information

This practice covers the following topics:

- Reviewing a business direction statement
- Identifying the type of component represented by the statement



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this practice, you review a series of business direction statements and decide which components they represent: business objective, assumption, key performance indicator, critical success factor, or problem.

III Representing the Flow of Data by Using a Process Model (Data Flow Diagram)

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Overview

This unit includes the following lessons:

3. Building a Process Model (Data Flow Diagram)
4. Using Oracle SQL Developer Data Modeler to Create Your Data Flow Diagram
5. Validating Your Data Flow Diagram



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this unit, you learn how to build a data flow diagram from a case study, and then use Oracle SQL Developer Data Modeler to document and validate the model.

Building a Process Model (Data Flow Diagram)

3

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- List the reasons why Process Modeling is useful
- Describe the components of a data flow diagram (DFD)
- Build a data flow diagram from a case study

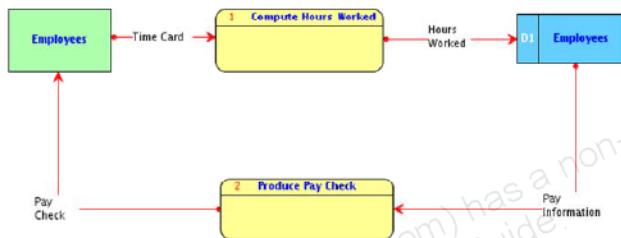


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you examine why process modeling is useful, learn what the components of a DFD are, and build a DFD from a case study.

What Is a Process Model?

- A Process Model documents the processes that the business performs and shows how the data flows through processes between External sources and Information Stores.



- The diagram for a Process Model is called a Data Flow Diagram (DFD).

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A **Process Model** is used to document the business processes that are performed within a particular area of the business. The Process Model shows how the data flows in and out of a Process and between External Agents and Information Stores. The diagram that is used to document the Process Model is called a *Data Flow Diagram* (DFD).

Benefits of Data Flow Diagrams

Data Flow Diagrams help to:

- Document project boundaries
- Understand current area of interest
- Find inefficiencies in the current process
- Communicate complex ideas and processes to users
- Determine data flow identification for Data Modeling



ORACLE®

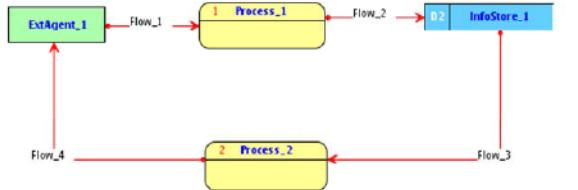
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Data Flow Diagrams are often overlooked, but they are useful in many ways. They do the following:

- Help to document the boundaries of a project so that you know exactly who must be involved and for what purpose
- Help everyone involved understand the current area of interest independent of the implementation
- Often uncover inefficiencies in the current way of doing things, allowing improvements to be made
- Enable end users to communicate complex ideas and processes in a way that everyone can understand
- Can help determine the flow of information that will be modeled in the logical data model (covered later in this course)

Components of a Data Flow Diagram

Data Flow Diagrams contain four different objects:



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A Data Flow Diagram has four main components:

Component	Purpose
Process	Is a discrete piece of work or function that is performed for a specific reason
External Agent	Includes a class of things or people that represent a source or destination of transactions
Information Flow	Represents the flow of information between objects
Information Store	Represents data stored between processes

Process

- A Process is a discrete piece of work.
- Examples:



ORACLE®

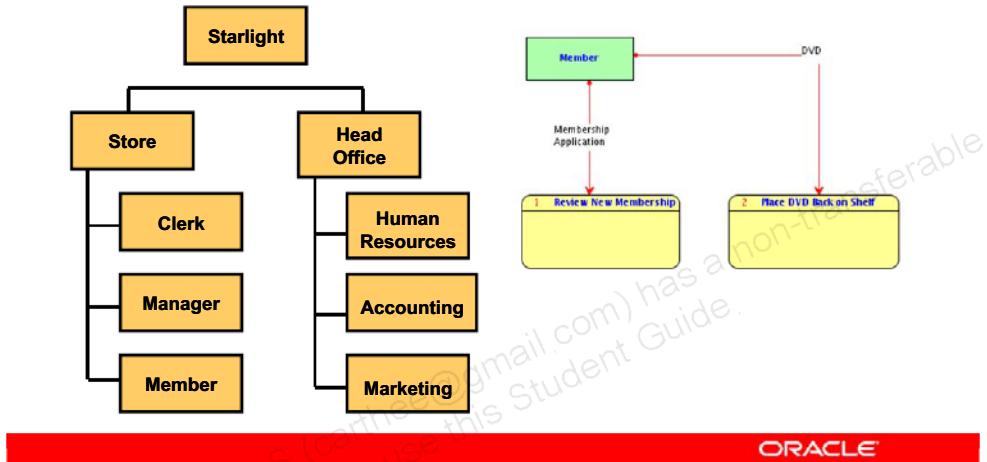
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A **Process** is a discrete piece of work. It is an action that has a specific begin and end point.
The slide shows three example processes:

- **Review New Membership:** Is performed when the new membership information is received and ends after the membership information has been recorded
- **Place DVD Back on Shelf:** Begins when a clerk receives a DVD from a customer, and ends when a clerk places it back on the shelf
- **Produce Overdue Rental Notices:** Begins on the last day of each month and ends when the notices are created

External Agent

An External Agent represents an organizational unit, department, job, role, or actor.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

An **External Agent** can be an organizational unit, system, or role that represents a source or destination for a process. For example, employees provide the dates that they will be on vacation. This information is then processed to calculate the vacation time that the employee has taken and is eligible for.

The left side of the slide shows a hierarchy of organizational units and the roles within each organizational unit. On the right, the Member role is represented as an External Agent in the DFD because it supplies information to the "Review New Membership" and "Place DVD Back on Shelf" processes.

Information Store

- An Information Store is a collection of information or materials.
- Examples:



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

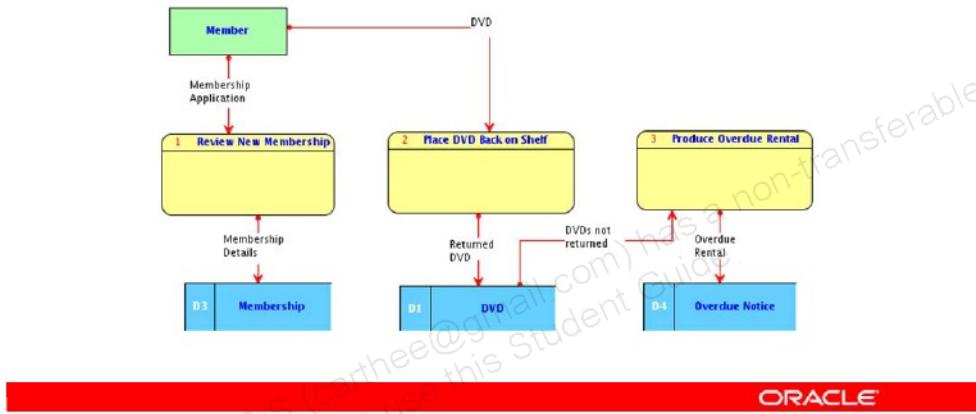
An **Information Store** represents “data at rest” or a collection of information or materials. For example, an information store can represent a warehouse containing goods, a filing cabinet containing copies of customers’ proof of identity, or a database containing records of rental transactions.

The reason for including information stores in a process model is to show any significant data or materials that are of central importance within the business area. It is neither necessary nor appropriate to show every information store defined for the system. As a general rule, you should add information stores to a DFD only when they aid in the understanding of the process flow.

Note that External Agents cannot directly supply information to Information Stores. A Process is needed to extract information from an Information Store (or External Agent) and then insert or update the same or different Information Store (or External Agent).

Information Flow

- An Information Flow shows the flow of information into and out of a Process.
- Examples:



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Information Flows show how the Information Flows into and out of a process. An Information Flow must have one of its ends connected to a process. An Information Flow cannot be used between two Information Stores or two external agents, or between an external agent and an Information Store. Remember that a process model shows the flow of information through a series of processes. Therefore each Information Flow must be an input or output to a process.

There are three types of Information Flows:

Flow Type	Description
Data	The flow of information between two elements
Material	The flow of tangible objects between two elements
Temporal	A time-based dependency that indicates that the process step at the destination of the flow cannot begin until the process step at the source has completed

Quiz

“Verify Membership” is considered to be which type of DFD component?

- a. External Agent
- b. Information Store
- c. Information Flow
- d. Process

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: d

Quiz

“Membership Information” is considered to be which type of DFD component?

- a. External Agent
- b. Information Store
- c. Information Flow
- d. Process

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b, c

The answer could be b or c.

Quiz

“Membership” is considered to be which type of DFD component?

- a. External Agent
- b. Information Store
- c. Information Flow
- d. Process

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b, c

The answer could be b or c.

Quiz

“Member” is considered to be which type of DFD component?

- a. External Agent
- b. Information Store
- c. Information Flow
- d. Process

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a

Events

- Events are occurrences that cause a response to execute.
- Types of events:



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Identifying and analyzing events is a useful technique to aid in the development of a DFD. Events trigger processes to begin. Events also result from a process completing or from an outcome.

The following three types of events can occur:

- **External (Other) Event:** Initiated outside the business area (for example, a customer requests membership)
- **Internal (System) Event:** Initiated inside the business area (for example, the inventory is updated with details of a new DVD)
- **Temporal (Time) Event:** Occurs when real (actual) time reaches a predetermined date or time (for example, a membership is due for renewal)

Analyzing Event Responses

Identifying responses to events helps in understanding the day-to-day workings of a particular business area.

Event	Response
Member returns DVD.	LogDVDreturn.
DVD shipment is received.	Pay supplier invoice.
Position becomes available.	Hire employee.
Member presents DVD for rental.	Verify membership.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Quiz

Which of the following is an event?

- a. Produce employee paycheck.
- b. Mail check to employee.
- c. Deposit check directly into employee account.
- d. End of the month

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: d

a, b, and c are all processes. d is an event that triggers the other options in this quiz.

Quiz

Which of the following is a response?

- a. Customer places an order.
- b. End of the year
- c. Issue refund to customer.
- d. Customer returns item.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Class Practice: Create a Data Flow Diagram

“I’m the manager of a training company that provides instructor-led courses to customers. We teach many courses, such as *Introduction to Linux* and *Introduction to SQL*, which are two of the most popular courses. Courses vary in length from one day to four days. We have a pool of instructors who teach one or more of our classes. Another group within our company develops the courses and then gives us the material we teach. My group schedules the courses by looking at the rooms available. We then assign instructors to teach the courses based on their schedules and subject expertise. The schedule is then sent to the instructors to let them know what classes they are assigned to teach. We keep information about the various training locations, the courses we teach, and the instructors who teach each course.”

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

As a class, discuss what processes, external agents, and information stores are necessary for the scenario in the slide. Create a data flow diagram to document the flow of information.

Summary

In this lesson, you should have learned how to:

- List the reasons why process modeling is useful
- Describe the components of a data flow diagram
- Build a data flow diagram from a case study



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you examined why process modeling is useful, learned what the components of a DFD are, and built a DFD from the Starlight Rental case study.

Practice 3-1 Overview: Create a Data Flow Diagram

This practice covers the following topics:

- Identifying data flow diagram objects
- Building a data flow diagram for Starlight Rental

Read the scenario and build a data flow diagram.

Using Oracle SQL Developer Data Modeler to Create Your Data Flow Diagram



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Load and set the default options for Oracle SQL Developer Data Modeler
- Build a data flow diagram by using Oracle SQL Developer Data Modeler
- Edit the layout of your Data Flow Diagram



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you learn how to use **Oracle SQL Developer Data Modeler** to document the DFD that you created in the previous lesson.

Running Oracle SQL Developer Data Modeler on Linux

1. Download from OTN and unzip to a local directory.
2. Open a terminal window and execute the following commands:

```
cd datamodeler  
./datamodeler.sh
```

The terminal window shows the following command sequence:

```
oracle@EDRSR20P1:/opt/datamodeler  
File Edit View Terminal Tabs Help  
[oracle@EDRSR20P1 opt]$ cd datamodeler  
[oracle@EDRSR20P1 datamodeler]$ dir  
datamodeler datamodeler.sh jdev rdbms timingframework  
datamodeler64.exe ide jlib sleepycat  
datamodeler.exe jdbc modules sqldeveloper  
[oracle@EDRSR20P1 datamodeler]$ ./datamodeler.sh  
  
Oracle SQL Developer Data Modeler  
Copyright (c) 1997, 2011, Oracle and/or its affiliates. All rights reserved.
```



ORACLE

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Oracle SQL Developer Data Modeler is very easy to install and execute.

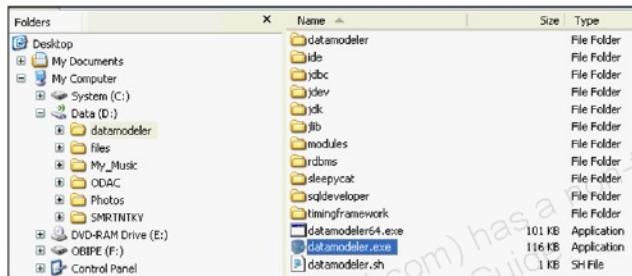
To install, go to <http://www.oracle.com/technetwork/developer-tools/datamodeler/overview/index.html> and click the download link for Oracle SQL Developer Data Modeler. You must have a valid license to use this product.

To run Oracle SQL Developer Data Modeler on Linux, open a terminal window and execute the commands shown in the slide.

Note: The first time that you run Oracle SQL Developer Data Modeler, you may be asked to enter your JDK directory location.

Running Oracle SQL Developer Data Modeler on Windows

1. Download from OTN and unzip to a local directory.
2. In Windows Explorer, navigate to the datamodeler directory and double-click datamodeler.exe.

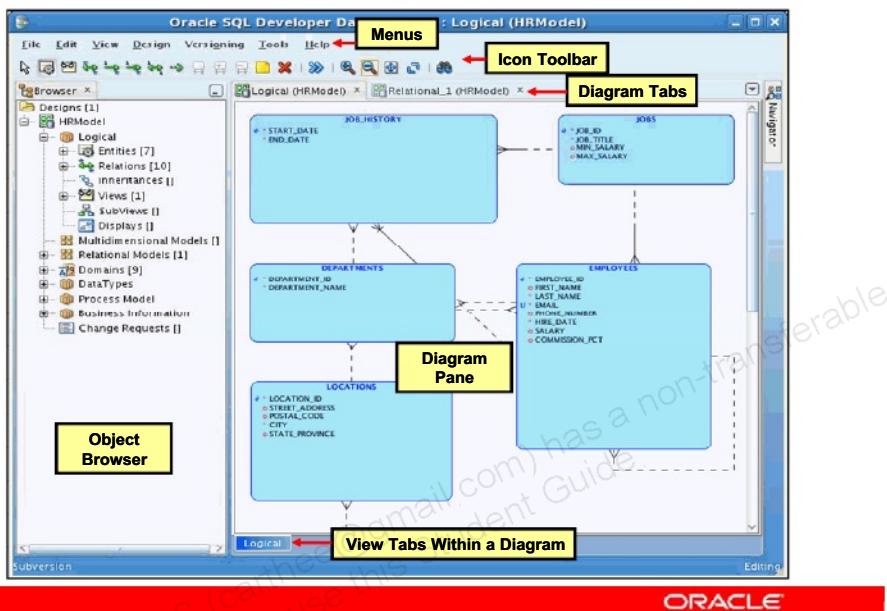


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

To run Oracle SQL Developer Data Modeler on Windows, open Windows Explorer and execute datamodeler.exe from the datamodeler directory.

Main Window



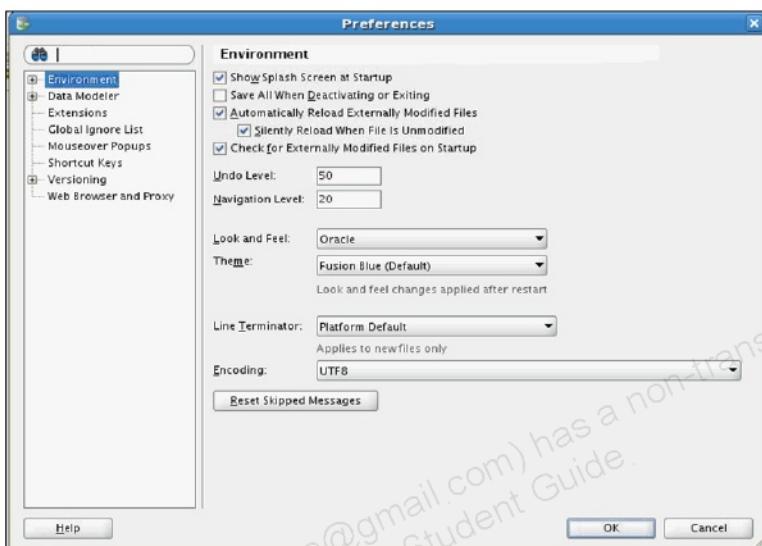
Generally, the left side of **Oracle SQL Developer Data Modeler** main window is used for navigation to find and select objects, and the right side to view information about selected objects.

The menus at the top of the window contain some standard entries, plus entries for features specific to **SQL Developer Data Modeler**. You can use shortcut keys to access menus and menu items. Examples: Alt+F or F10 opens the File menu. Alt+E opens the Edit menu. Alt+H opens Help.

Icons under the **Menus** perform actions relevant to what is currently selected for display on the right side of the window, such as the logical model, a relational model, or a data flow diagram. For example, for a relational model the icons include New Table, New View, Split Table, Merge Tables, New FK Relation, New Type Substitution, New Note. To see the name of any icon, hover the pointer over the icon. The actions for the icons are also available from the Object menu.

The left side of the **Data Modeling** window has an object browser with a hierarchical tree display for data modeling objects. The right side of the **Data Modeling** window has tabs and panes for diagrams that you select or open. The tabs at the top of the diagram pane window allow you to navigate from one type of diagram to another. The tabs at the bottom of each diagram allow you to navigate from various subviews within a diagram type (subviews are discussed later in the course).

Setting User Preferences



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Within **Oracle SQL Developer Data Modeler**, you can customize various aspects of Data Modeler environment and interface by modifying user preferences. To modify user preferences, select **Tools > Preferences**. The **Preferences** are grouped in the following categories:

- Environment:** This pane contains options that affect the startup and overall behavior and appearance of Data Modeler where you can specify certain operations be performed automatically or not.
- Data Modeler:** This pane contains various options and preferences that affect the startup and overall behavior and appearance of Data Modeler.
- Extensions:** This pane determines which optional extensions Data Modeler uses when it starts.
- Global Ignore List:** This pane specifies filters that determine which files and file types will not be used in any processing.
- Mouseover Popups:** This pane specifies text to be displayed on hover-related mouse actions over relevant object names.
- Shortcut Keys:** This enables you to view and customize the shortcut key mappings for Data Modeler

- **Versioning:** Versioning preferences affect the behavior of the version control and management systems that you can use with Data Modeler.
- **Web Browser and Proxy:** The Web Browser and Proxy pane settings are relevant only if Data Modeler needs to access the World Wide Web (such as when you use the Check for Updates feature on the Help menu), and only if your system is behind a firewall.

Setting User Preferences: Data Modeler



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The Model pane contains options that affect the startup and overall behavior and appearance of Data Modeler.

- **Default Designs Directory:** The default directory or folder from which to open a design or in which to create a design.
- **Default Import Directory:** The default directory or folder from which to import designs.
- **Show Log After Import:** Controls whether a Log window is displayed after an import operation. The window contains informational messages and any warning or error messages.
- **Default Save Directory:** The default directory or folder in which to save files.
- **Default System Types Directory:** The default directory or folder for storing type definition files.
- **Default Reports Directory:** The default directory or folder in which to generate Data Modeler Reports.
- **Path to Saxon XSLT 2.0 Jar File:** This is used to set the path to Saxon XSLT 2.0 Jar File.

Show Select Relational Models Dialog: Controls whether the dialog box for selecting relational models to be included is displayed when you open a Data Modeler design. If this option is disabled, all relational models are included by default when you open a Data Modeler design.

Load Design Level Settings on Open Design: Controls whether to load any settings specific to that design when opening a design. If this option is disabled, any design-specific settings are ignored and the Data Modeler global settings are used.

Show Properties Dialog on New Object Controls whether the Properties dialog box for objects of that type is displayed when you create a new model object

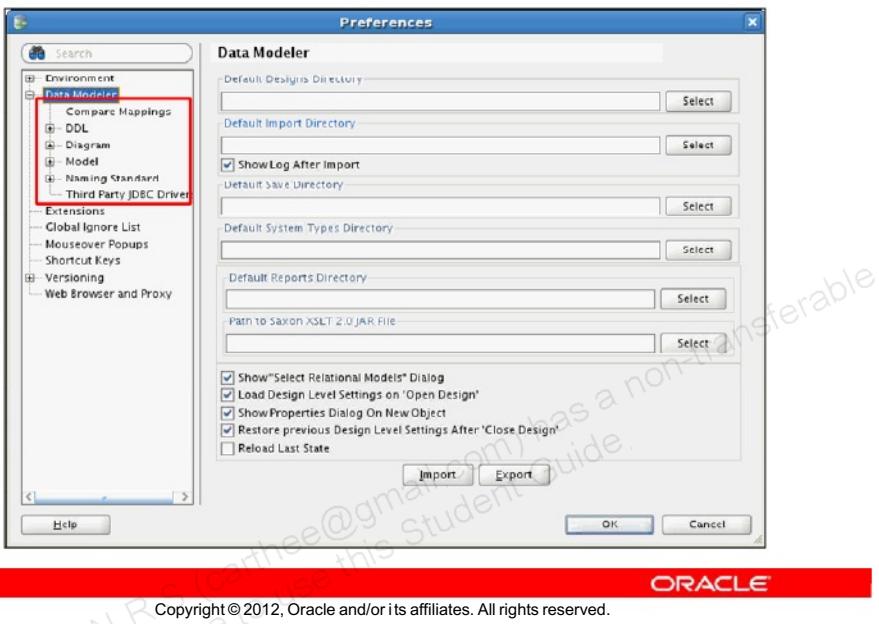
Restore Previous Design Level Settings After Close Design: Controls whether to restore any setting specific to the previous current design after closing the current design. If this option is disabled, the Data Modeler global settings are restored.

Reload Last State

Import: Allows you to import Data Modeler preferences and other settings that had previously been exported

Export: Saves Data Modeler preferences and other settings to an XML file, so that you can later import the information

Setting User Preferences: Data Modeler



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

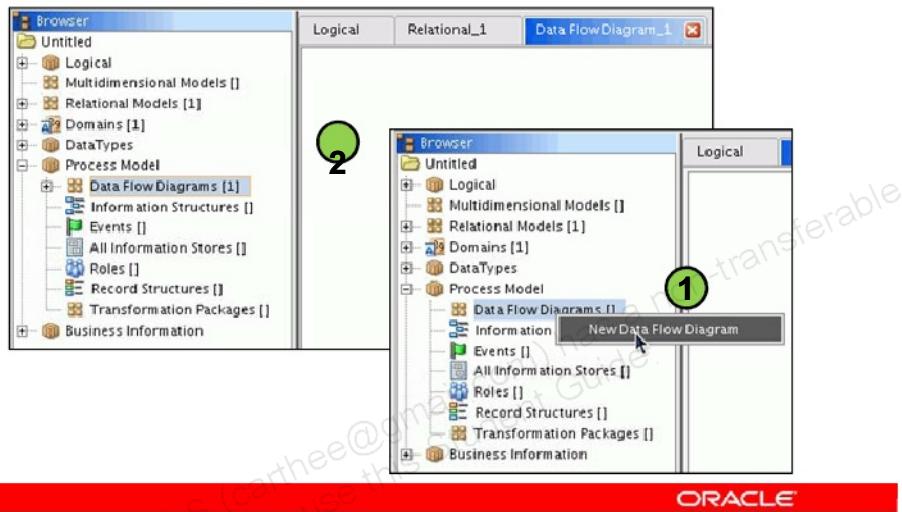
ORACLE®

Other **Data Modeler** preferences are grouped into the following categories:

- **Compare Mappings:** This pane contains options for comparing source and target objects.
- **DDL:** The DDL pane contains general options for Data Definition Language (DDL) statements in code to be generated.
- **Diagram:** This pane contains general options that affect the appearance of model diagrams.
- **Model:** The Model pane contains options that apply to several types of models.
- **Naming Standards:** The Naming Standard pane lets you implement naming standardization: you can view, add, and modify naming standards for logical and relational model objects and for domains.
- **Third Party JDBC Driver:** The Third Party JDBC Drivers pane specifies drivers to be used for connections to third-party (non-Oracle) databases.

Building a Data Flow Diagram

1. Create a Data Flow Diagram.

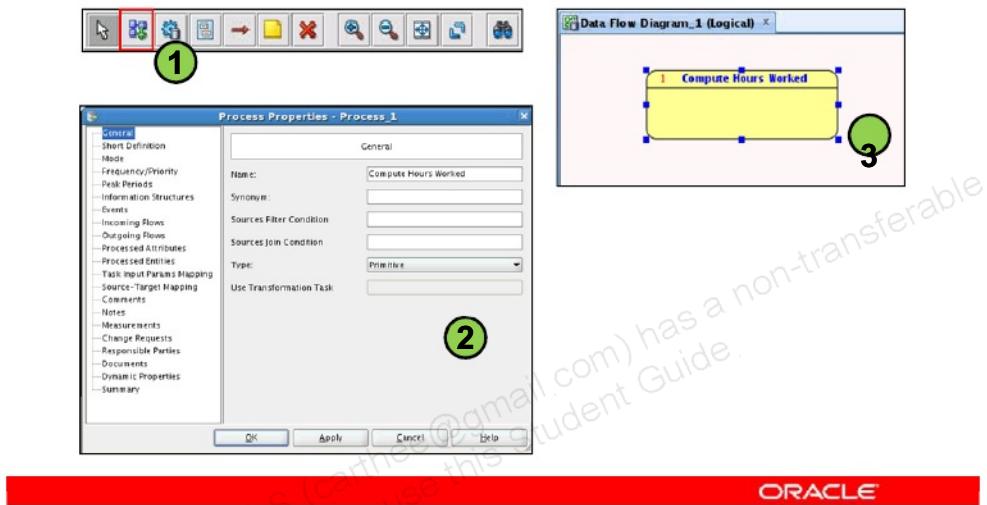


The first step in building a Data Flow Diagram is to create a New Data Flow Diagram, which will open a Diagram pane (with a tab).

In the object browser, expand `Process Model`, right-click `Data Flow Diagrams`, and select `New Data Flow Diagram`.

Building a Data Flow Diagram

2. Create a Process.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

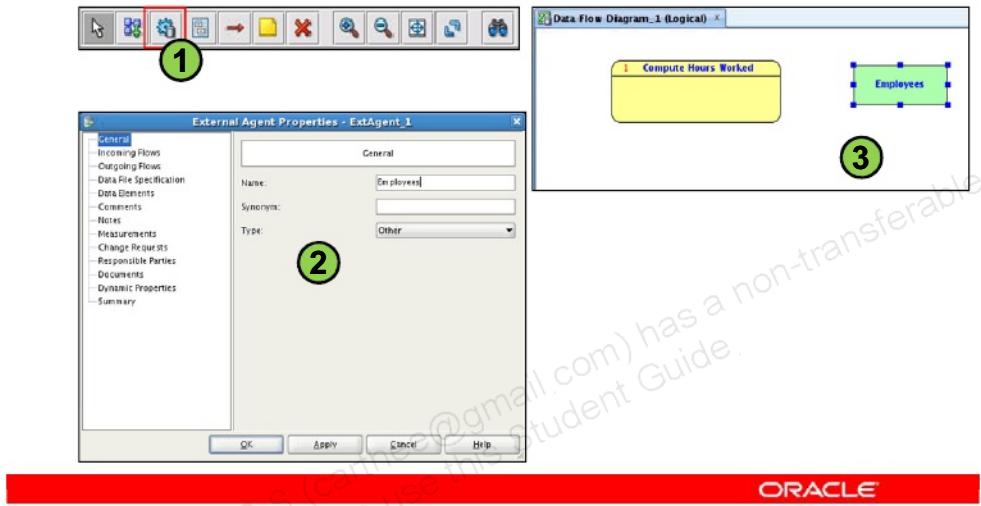
The next step is to create a **Process**.

1. On the toolbar, click the **Create a Process** icon and then click anywhere in the white space in the **Data Flow Diagram** pane. A **Process Properties** window appears.
2. In the **Process Properties** window, specify the name of the process and add other pertinent information about the process, and then click **OK**.
3. A process object appears in the diagram.

To make changes to the process definition, double-click the process to reopen the Process Properties window.

Building a Data Flow Diagram

3. Create an External Agent.



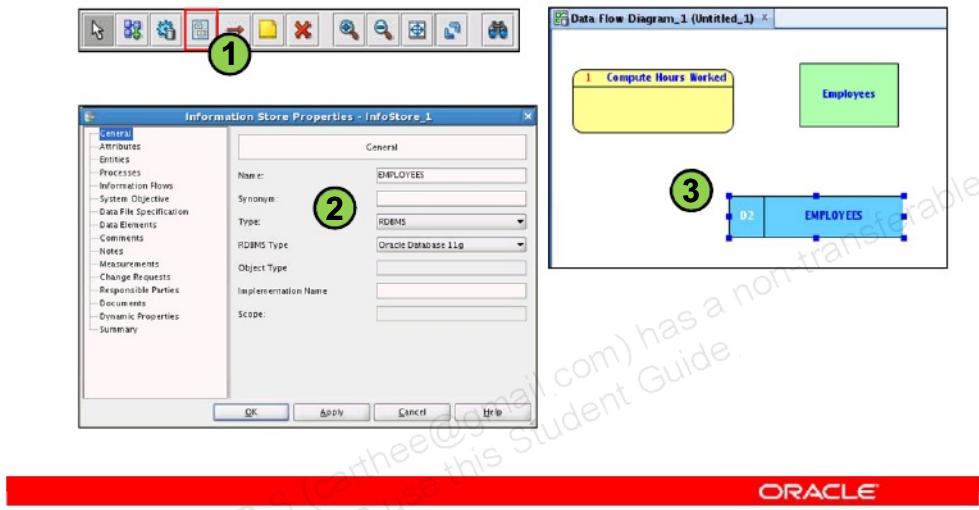
The next step is to create an **External Agent**.

1. On the toolbar, click the **Create an External Agent** icon and click anywhere in the white space in the **Data Flow Diagram** pane. An **External Agent Properties** window appears.
2. In the **External Agent Properties** window, specify the **Name** of the external agent and add other pertinent information, and then click **OK**.
3. The **External Agent** object appears in the diagram.

To make changes to the external agent definition, double-click the external agent to reopen the **External Agent Properties** window.

Building a Data Flow Diagram

4. Create an Information Store.



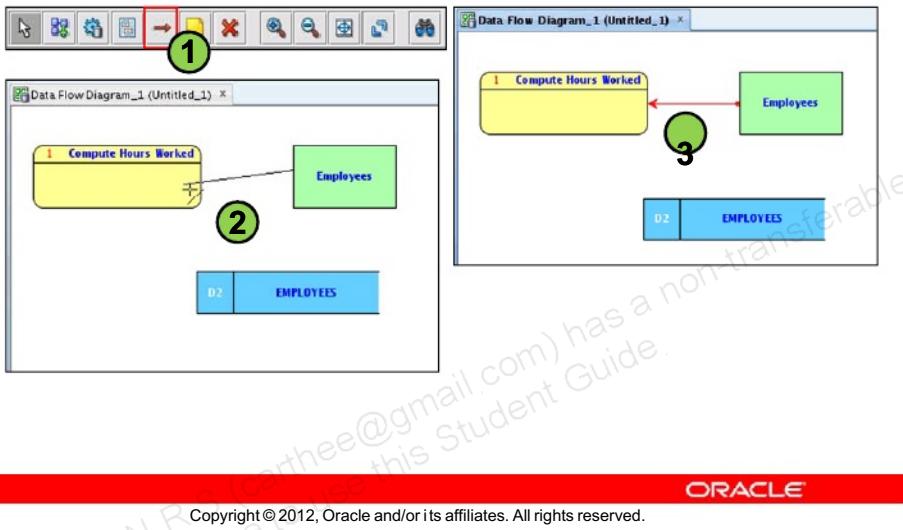
The next step is to create an **Information Store**.

1. On the toolbar, click the **Create an Information Store** icon and click anywhere in the white space in the **Data Flow Diagram** pane. An **Information Store Properties** window appears.
2. In the **Information Store Properties** window, specify the name of the information store and add other pertinent information, and then click **OK**.
3. The **Information Store** object appears in the diagram.

To make changes to the information store definition, double-click the **Information Store** to reopen the **Information Store Properties** window.

Building a Data Flow Diagram

5. Create an Information Flow.

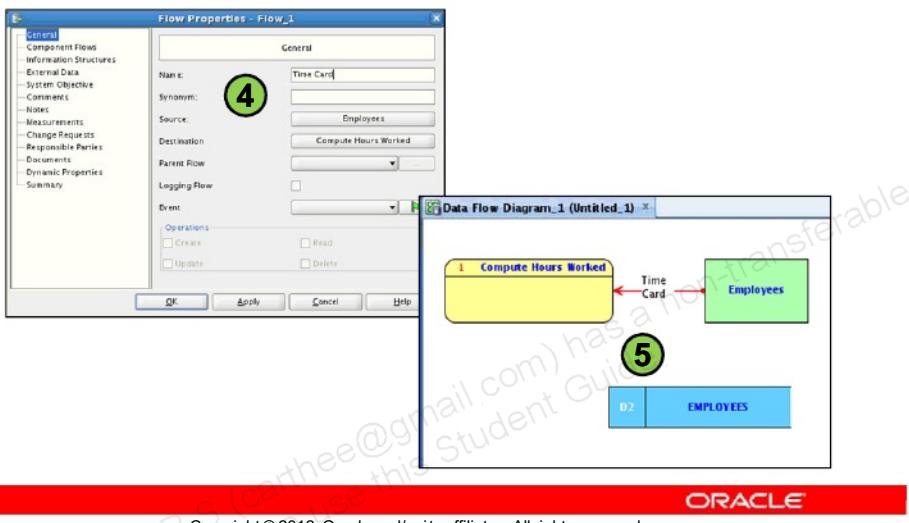


The next step is to create an **Information Flow** between the two objects.

1. On the toolbar, click the **Create Flow** icon.
2. Select the object that you want to create the Flow from, and then select the object you want to connect the Flow to. In the example in the slide, click the **Employees External Agent**, and then click the **Compute Hours Worked** process.
3. The **Flow** object appears in the diagram. To change the **Name** of the **Flow**, right-click the **Flow Line** to open the **Properties** dialog box.

Building a Data Flow Diagram

5. Create an information flow.

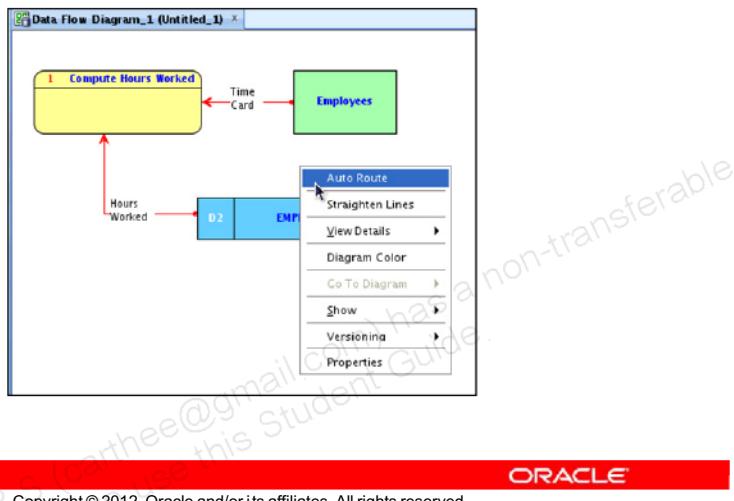


Creating a Flow between two objects:

4. In the **Properties** dialog box, change the **Name** of the **Flow** and then click **OK**.
5. To view the label of the **Flow Line**, right-click anywhere in the diagram and select **Show**. Select **Labels** to be able to view the label of the **Flow Line**.

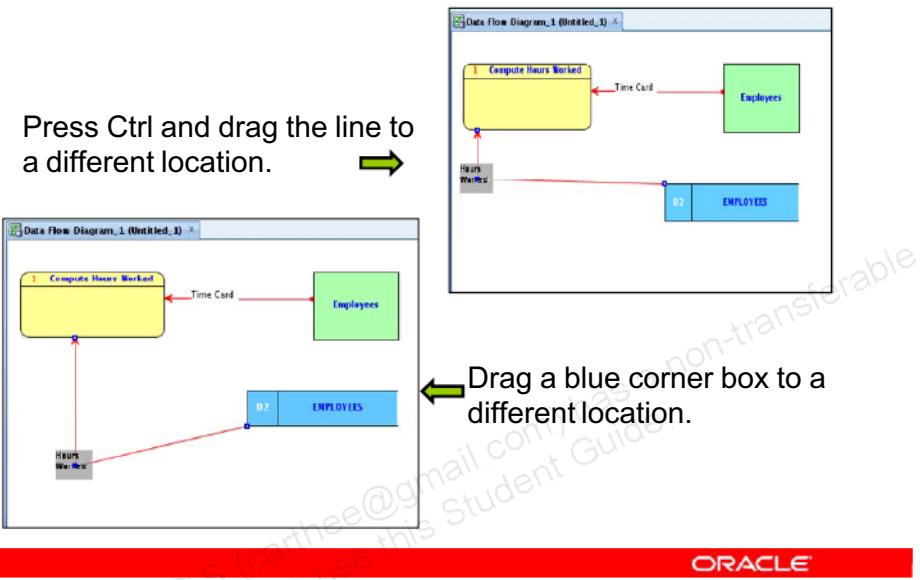
Editing the Diagram Layout

Turn off Auto Route.



To redraw lines (for example, to add an elbow or straighten a line), you must first turn **Auto Route** off. Right-click in the white space to deselect **Auto Route** (if it is selected).

Editing the Diagram Layout



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

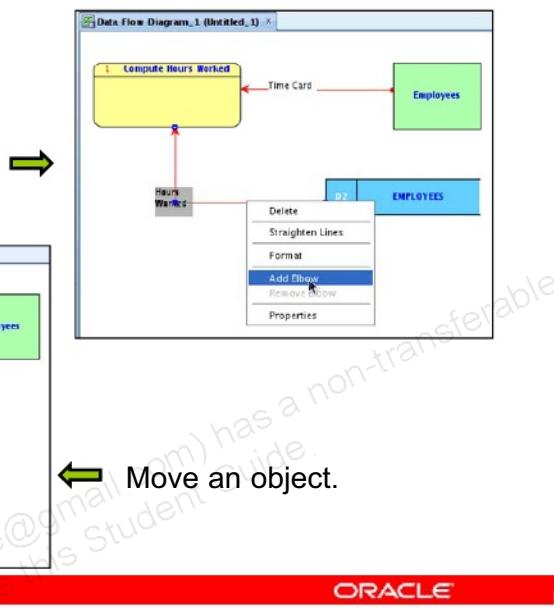
There are many ways to redraw lines:

- Press the **Ctrl** key and drag the line to a different location.
- Drag a blue corner to a different location.
- Create an elbow or straighten lines.
- Move an object.

The first two methods are shown in the slide. The last two methods are shown in the next slide.

Editing the Diagram Layout

Use the shortcut menu to add an elbow or to straighten lines.

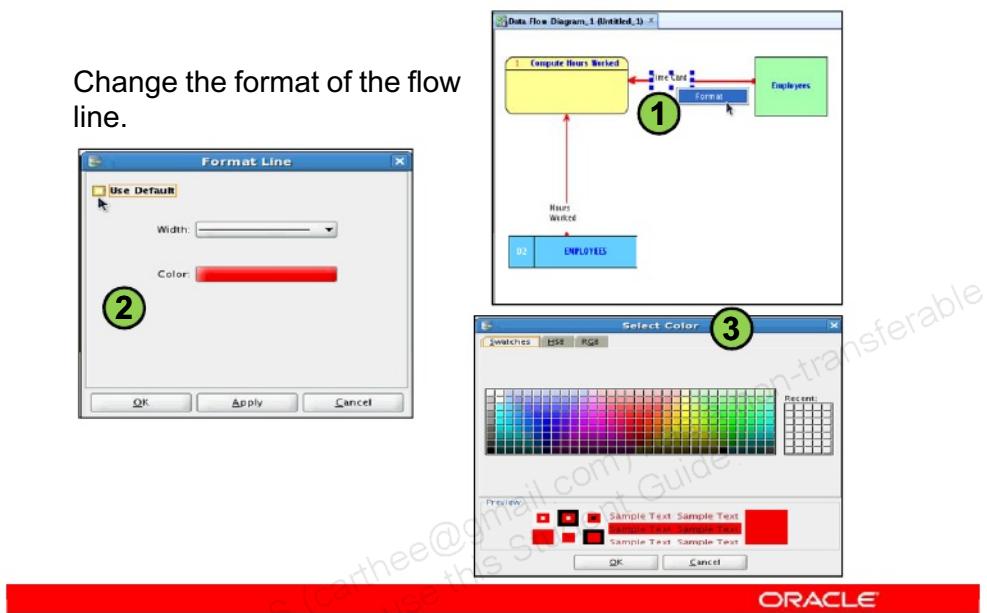


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The last two methods from the previous page are shown in this slide.

Editing the Diagram Layout

Change the format of the flow line.

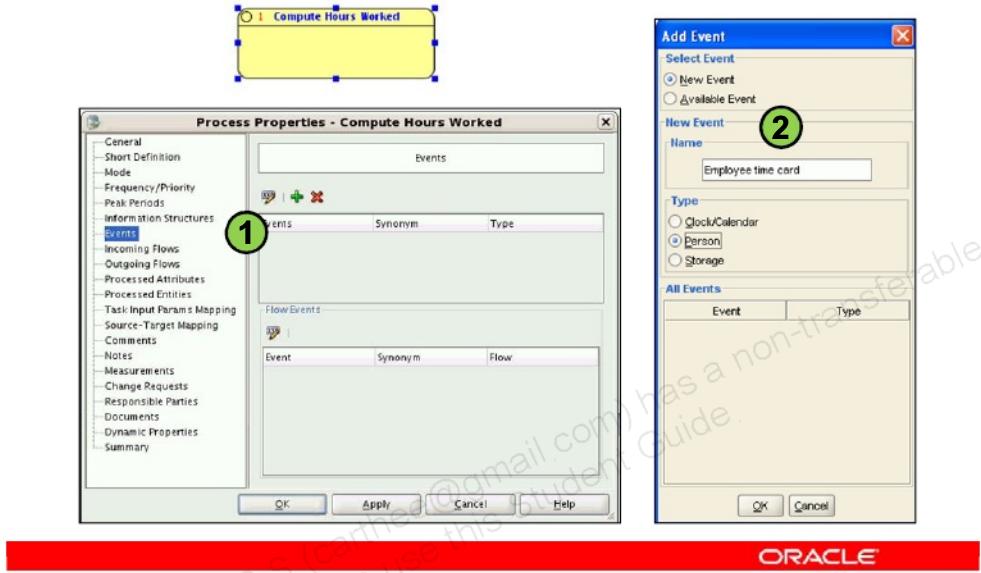


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To change the format of the flow label, perform the following steps:

1. Right-click the **Flow** label and select **Format**.
2. Deselect **Use Default** and click **Color** to find a color for the label. If you want to change the width of the **Flow Line**, select **Width** from the drop-down list.
3. Select the desired color and click **OK**.
4. Click **OK** again.

Adding and Reusing Process Events



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

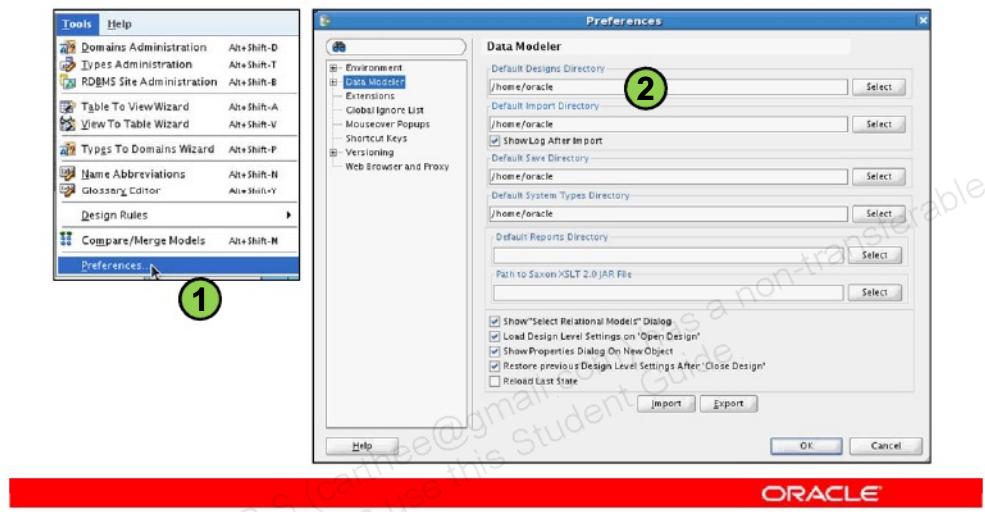
ORACLE®

To document what event triggers a process, perform the following steps:

1. Right-click the **Process** for which you want to define the event. Then select **Properties**.
2. Select the **Events** category in the left navigator.
3. Click the **Add '+'** icon.
4. Specify whether this is a **New** or **Available** Event.
 - **New event:** Enter the name and select the event type and click **OK**.
 - **Available event:** Select it from the list of events and click **OK**.
5. Click **OK** again.

Saving Your Model

Before saving your model, set the default directory.



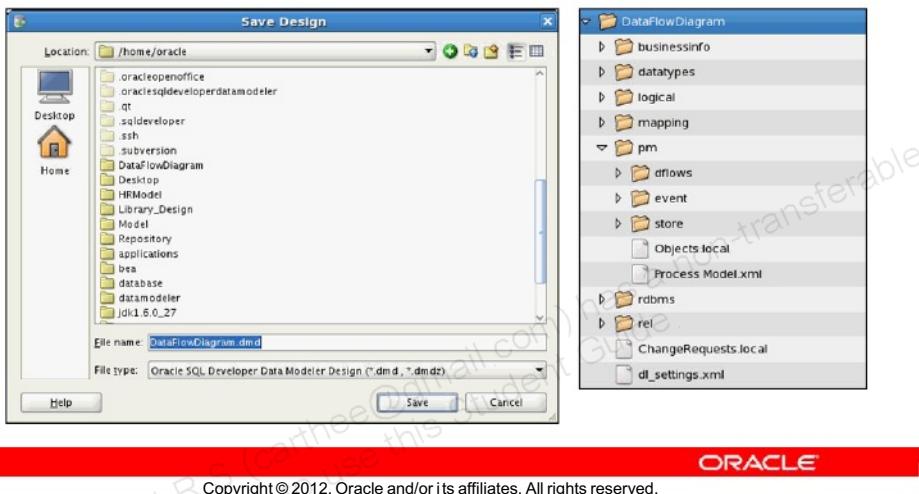
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To set the **Default Designs Directory**, **Default Import Directory**, **Default Save Directory**, and **Default System Types Directory**.

1. Select **Tools > Preferences**.
2. In the **Data Modeler Node**, set the **Default Directory** for Designs, Imports, Save, and System Types.

Saving Your Model

Saving a model creates a .dmd file and a directory with process model objects.

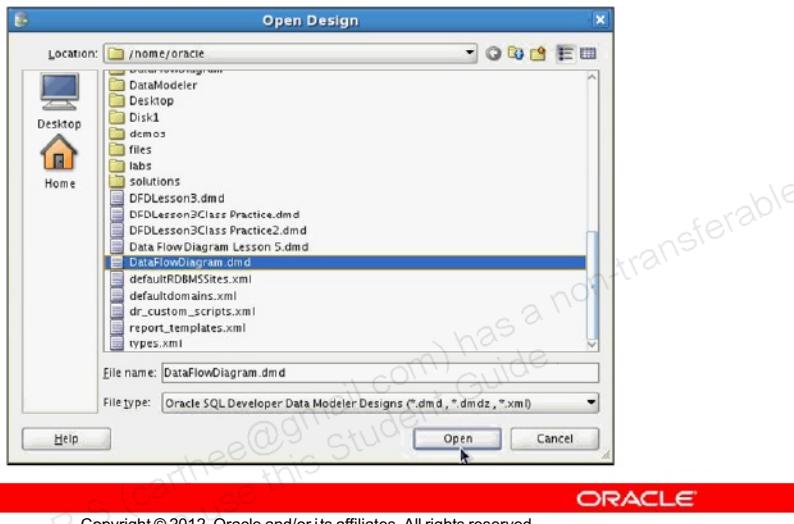


To save your model, select **Open**, **Save**, or **Save As** from the **File** menu.

The model is saved as a **DMD** file. For each DMD file, there is a directory with the same name that contains directories for each model type.

Opening a Saved Model

Open a model by selecting the `.dmd` file



To open a model, select **File > Open**. Then select the `.dmd` file of the model that you want to open.

Summary

In this lesson, you should have learned how to:

- Load and set the default options for Oracle SQL Developer Data Modeler
- Build a data flow diagram by using Oracle SQL Developer
- **Data Modeler** Edit the layout of your data flow diagram

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 4-1 Overview: Build a Data Flow Diagram in Oracle SQL Developer Data Modeler

This practice covers the following topics:

- Building a data flow diagram for the Starlight DVD case study
- Modifying the layout of the data flow diagram

In this practice, you use **Oracle SQL Developer Data Modeler** to build the **Data Flow Diagram** for the Starlight DVD case study.

Validating Your Data Flow Diagram

5

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Objectives

After completing this lesson, you should be able to:

- Validate a Data Flow Diagram (DFD) based on the set of DFD rules
- Identify different types of Processes
- Decompose processes into Primitive processes



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

DFD Rules: Process

- A Process must have at least one input and one output.
- Use verb-phrase labels.
- Each process has a unique name.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In a DFD, each **Process** must have at least one input and one output and have a unique name that is a verb phrase.

For example, the **Process** on the left in the slide has only input processes and therefore is invalid. The **Process** on the right has both an input and an output, so it is considered valid. In addition, its name is a verb phrase (starting with the verb *Review*).

DFD Rules: External Agents

- External Agents move directly to and from a Process.
- Use a noun-phrase label.



ORACLE®

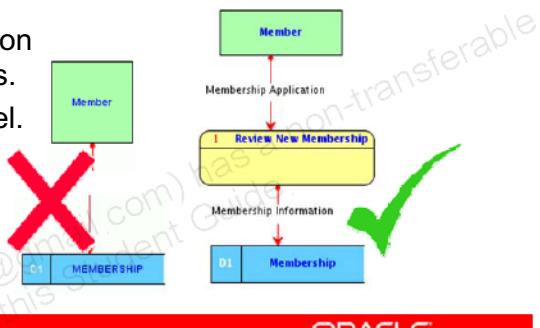
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

An **External Agent** must move to and from a process for it to be valid, and it must be labeled as a noun phrase.

On the left side of the slide, the **External Agent** has an arrow directly into an information store, which is invalid. On the right side of the slide, the **External Agent** has an information flow to a **Process**, which then has a **Flow** into an **Information Store**. This is considered valid. In addition, its name is a noun phrase (the noun *Member*).

DFD Rules: Information Store

- Data cannot move directly from one Information Store to another Information Store.
- An Information Store cannot move data directly from or to an External Agent.
- One end of an Information Flow must be a Process.
- Use a noun-phrase label.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

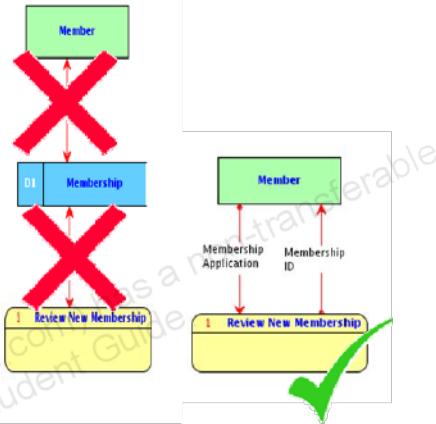
An Information Store cannot move directly to/from one Information Store to another or to/from an External Agent. One end of the Information Flow must be to a Process. The name of the Information Store must use a noun phrase.

In the example in the slide, the left graphic has an Information Flow from an External Agent directly to an Information Store, which is invalid. There must be a Process between an External Agent and an Information Store, which is the case with the graphic on the right side of the slide. In addition, the name of the Information Store is a noun phrase (the noun *Membership*).

DFD Rules: Information Flow

An Information Flow:

- Has only one direction of flow between objects (not a double arrow line)
- To an Information Store creates, updates, or deletes data
- From an Information Store retrieves data
- Uses a noun-phrase label

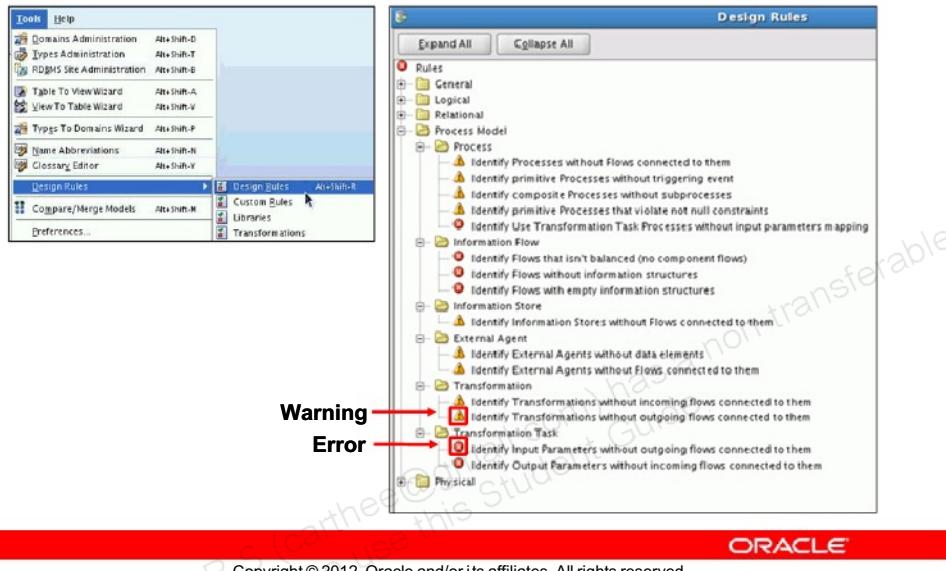


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

An **Information Flow** does not allow bidirectional **Flows** where both sides of the line have an arrow. To draw this on the DFD, you must create two**Information Flows**, each going in a different direction. An **Information Flow** that is input to an **Information Store** will create, update, or delete data in the information store. An **Information Flow** that is output to an **Information Store** will retrieve data from the information store. **Information Flows** should have a noun-phrase label.

Design Rules in Oracle SQL Developer Data Modeler



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In Oracle SQL Developer Data Modeler, you apply design rules to your model to check for various violations. For example, you can check whether any **Information Stores** lack input or output **Flows**.

To access the design rules, select **Tools > Design Rules**. Expand the **Process** model, expand each of the objects that you want to run design rules for, select the rule, and click **Apply**. Or you can use **Apply to All** to run the design rules against your entire model.

Quiz

Which of the following is not valid?

- a. A Flow from a Process to a Process
- b. A Flow from a Process to an Information Store
- c. A Flow from an Information Store to an External Agent
- d. A Flow from an External Agent to a Process

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Quiz

Which of the following are correct statements?

(Choose all that apply.)

- a. A Process label should be a verb phrase.
- b. An Information Store label should be a noun phrase.
- c. An External Agent label should be a noun phrase.
- d. All of the above

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: d

Types of Processes

- **Primitive:** A lowest-level stand-alone process
- **Composite:** A higher-level process that is broken down into a lower-level DFD
- **Transformation Task:** Includes input and output parameters, a sequence of simple activities (transformations) that can be executed as a single atomic unit



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You examine three types of processes in this course. The notation in the upper-right corner of the Process indicates which type of Process it is.

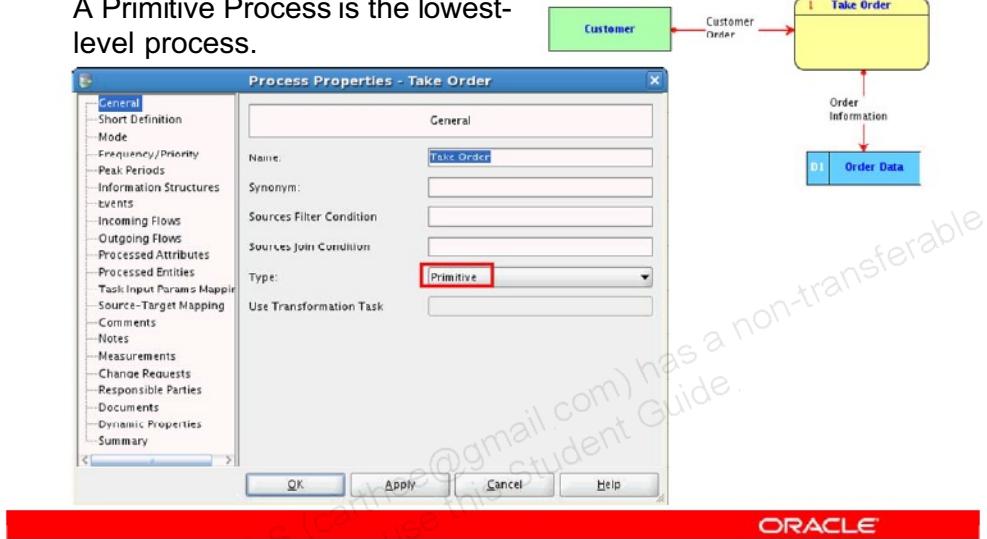
Primitive: The lowest-level stand-alone Process. A Primitive Process usually has one input and one output, and the Process cannot be divided any further. For example, the Take Order Process is one specific task that cannot be subdivided into lower-level Processes.

Composite: A higher-level Process that can be broken down into a lower-level DFD with multiple processes. For example, the Shipment Process can be broken down into multiple Processes: check inventory, pack shipment, notify customer with tracking information, and so on.

Transformation task: A Process that includes input and output parameters. It is a sequence of simple activities (transformations) that can be executed as a single atomic unit. For example, the ETL process executes a task that will process data from the orders system to the CRM system.

Primitive Process

A Primitive Process is the lowest-level process.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

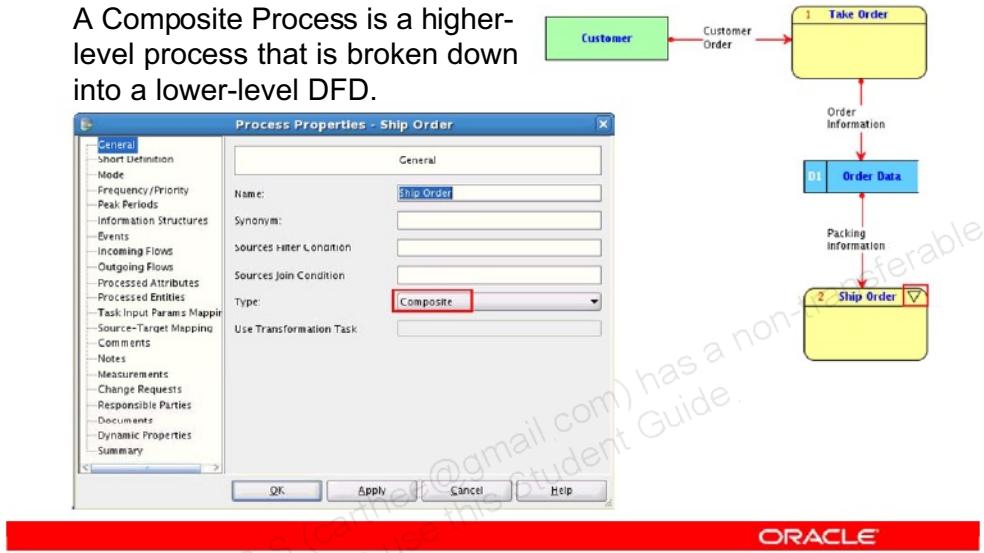
ORACLE®

A **Primitive Process** is the lowest-level process. It typically only has one input and one output. To create a **Primitive Process**, select **Primitive** for its type when you create the process.

In the example in the slide, the Take Order process accepts order information from the customer and then inserts the order information into the Order Data information store. The Take Order process only performs one task and cannot be subdivided.

Composite Process

A Composite Process is a higher-level process that is broken down into a lower-level DFD.

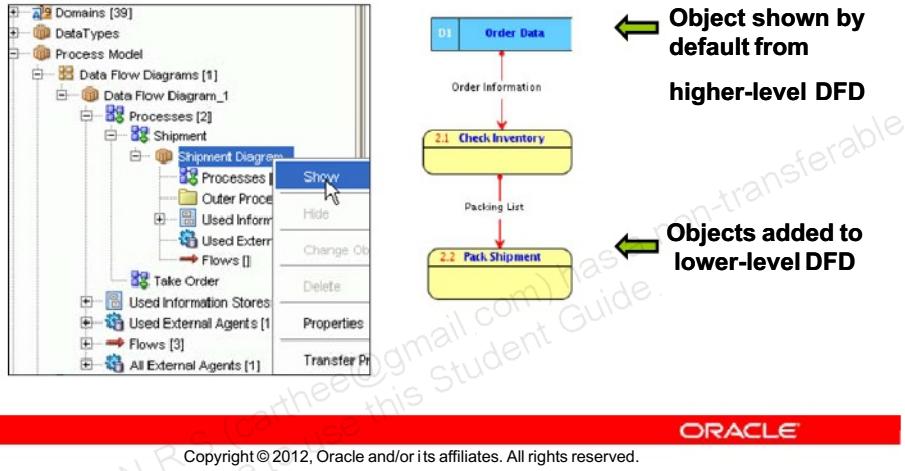


A **Composite Process** is a higher-level process that can be subdivided into a lower-level DFD. To create a **Composite Process**, select **Composite** for its type when you create the process.

In the example in the slide, the Shipment process has an upside-down triangle in the upper-right corner. The triangle indicates that the process is a composite process. Right-click the process and select **Go to Diagram** to see the lower-level DFD, which is shown in the next slide.

Composite Process

- Show DFD for composite process.
- Only objects that flow from/to process are displayed.



You can also navigate to the Shipment Diagram (lower-level DFD) from the object browser. As shown in the lower-level DFD, the Shipment process has two subprocesses. The Order Data information store is shown in the DFD because it is an input source to the Shipment process at the higher level.

Transformation Task Process

A Transformation Task is a sequence of simple activities (transformations) that can be executed as a single atomic unit.

Steps:

1. Create a Transformation Task package with a transformation task.
2. Create a DFD outlining the Flow of Information (for example, one Process with two Information Stores where the process will transform the data).
3. For each Information Flow, define source and target attributes.
4. Define source–target mappings in the process.
5. In the DFD, create a Process that uses the Transformation Task.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A Transformation Task process is a sequence of simple activities (transformations) that can be executed as single atomic unit.

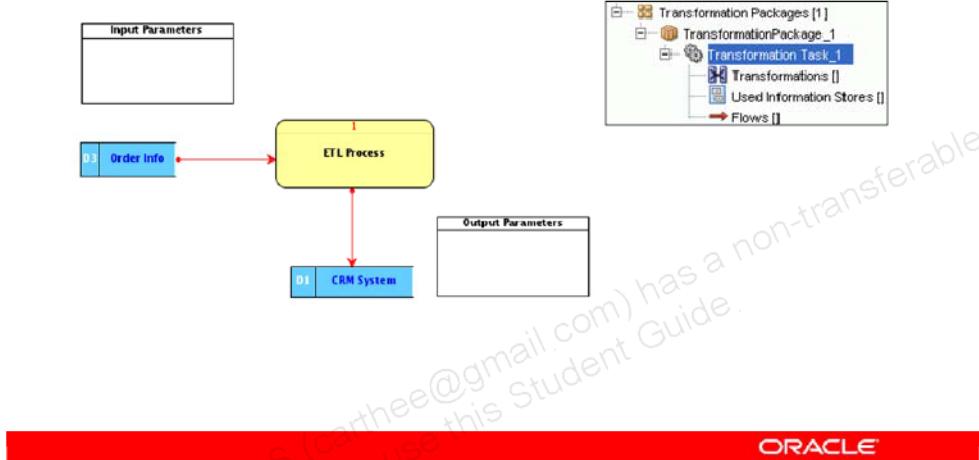
Transformation Tasks have the following characteristics:

- They are reusable.
- They have unlimited levels of decomposition.
- They are basically an interface with input and output parameters.
- They transform inputs into outputs.
- They do not interact with any other process in the diagram.
- They can map source attributes to a task's input parameters and a task's output parameters to target attributes.

The steps for creating a transformation task process are listed in the slide.

Transformation Task Process

Create a Transformation Task Package and a Transformation Task DFD.



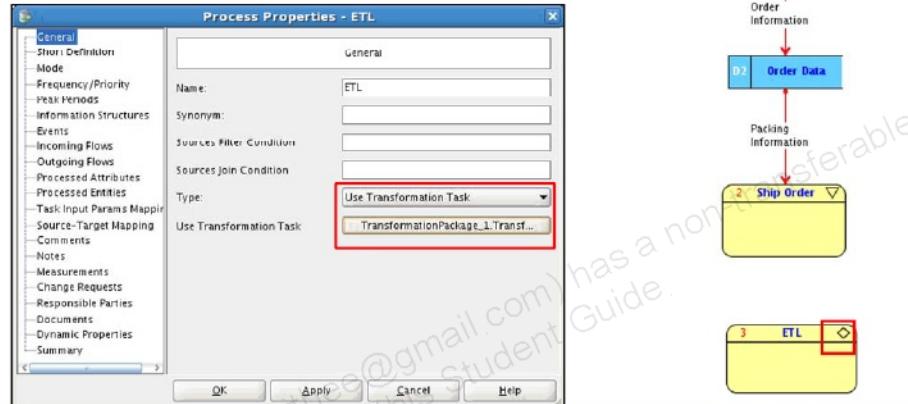
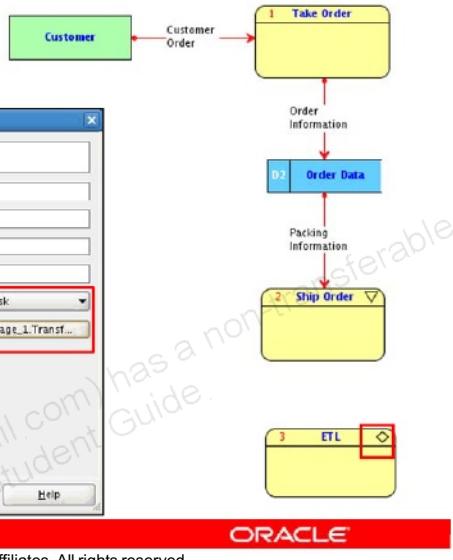
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To create a **Transformation Task** package, perform the following steps:

1. Under **Process Model** in the object browser, right-click **Transformation Task Package** and select **New Package**.
2. Right-click the **New Transformation Task Package** and select **New Transformation Task**.
3. The diagram editor opens. You can now create the DFD objects for your Transformation Task. In the example in the slide, the ETL process will retrieve information from the Order Info information store and then insert the data into the CRM System information store.
4. Define the source and target attributes that will be transformed in each of the Information Flows.
5. Map the source attributes to the target attributes in the process definition.

Transformation Task Process

Create a Transformation Task Process, and select Use Transformation Task.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

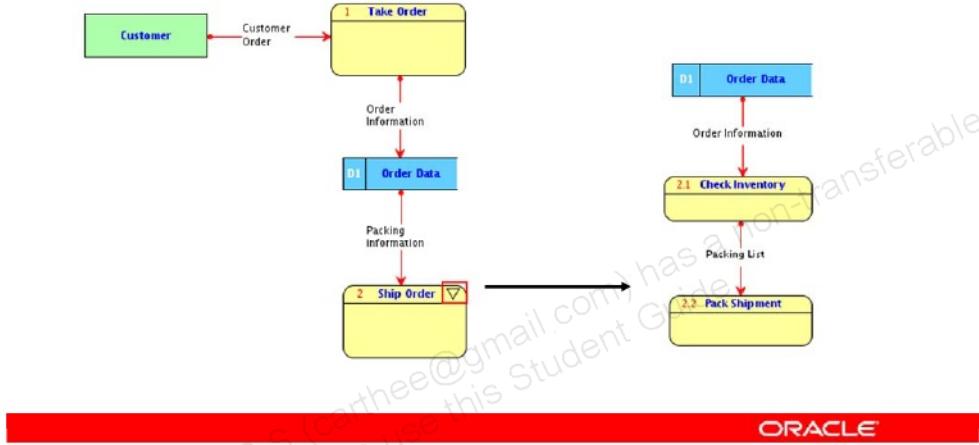
ORACLE®

After the **Transformation Task** has been created, you can then create a Transformation Task process in your DFD and use the Transformation Task.

In the slide, you see the ETL process that has the type User Transformation Task. To identify a Transformation Task process, look for a diamond in the upper-right corner of the process.

Process Decomposition

A Composite Process is decomposed into a lower-level DFD.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A **Composite Process** is decomposed into one or more lower-level DFDs until each process is at the **Primitive Process** level.

In the example in the slide, you see that the **Shipment** composite process is decomposed into a separate DFD with multiple processes.

Decomposition Guidelines

- Each level of the DFD should be on a separate page.
- No DFD level should have more than seven processes.
- Balance the flows at each level. The inputs to a process must be sufficient to produce the outputs.
- At the lowest level of DFDs, new Information Flows can be added to represent data that is transmitted under exceptional conditions (for example, error messages).
- To avoid having Information Flow lines cross each other, you can repeat Information Stores and External Agents on a DFD.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Follow these guidelines when you build DFDs and decompose processes to their primitive process level:

- Each level of the DFD should be in its own diagram.
- No DFD level should have more than seven processes. If you need more than seven, combine the processes into a higher-level Composite process, and then break that process down into another DFD level.
- Make sure that all the flows at one level balance at the next level. This means that if you have two inputs and one output for a Composite process, the next level DFD should have the same number of inputs and outputs.
- At the primitive level, you can add Information Flows to represent data that is transmitted under exception conditions.
- You can duplicate information stores and external agents to minimize crossing Information Flow lines. In Oracle SQL Developer Data Modeler, the names must be unique. If you enter a non-unique name, Oracle SQL Developer Data Modeler will append *V1* to the name.

Quiz

Which of the following characteristics does a composite process contain?

- a. A lower-level DFD
- b. The same number of inputs and outputs as the higher level
- c. A transformation task package
- d. The lowest-level process

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a

Summary

In this lesson, you should have learned how to:

- Validate a DFD based on the set of DFD rules
- Identify different types of processes
- Decompose processes into primitive processes

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 5-1 Overview: Decompose a Process in Your Data Flow Diagram

This practice covers the following topics:

- Decomposing the Gather Membership Information process
- Creating a Transformation Process to load into the CRM application



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this practice, you review a list of requirements, decompose the Gather Membership Information process, and create a Transformation process.

Unauthorized reproduction or distribution prohibited. Copyright© 2013, Oracle and/or its affiliates.

KARTHIKEYAN R S (carthee@gmail.com) has a non-transferable
license to use this Student Guide.

III **Developing a Logical Data Model**

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Overview

This unit includes the following lessons:

6. Identifying Entities and Attributes
7. Identifying Relationships
8. Assigning Unique Identifiers
9. Using Oracle SQL Developer Data Modeler to Create an Entity Relationship Diagram
10. Validating Your Entity Relationship Diagram

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this unit, you learn how to build an entity relationship diagram from a case study. You then use Oracle SQL Developer Data Modeler to document and validate the model.

Identifying Entities and Attributes



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to identify and diagram the following:

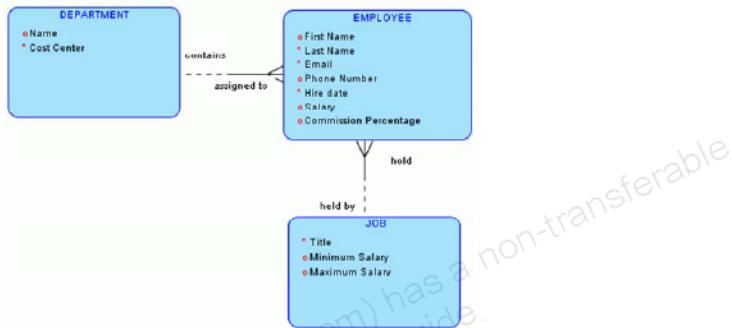
- Entities from a case study
- Attributes from a case study

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

What Is a Logical Data Model?

A logical data model documents the information requirements of the business.



The diagram for a logical data model is called an *entity relationship diagram (ERD)*.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The goal of a logical data model is to develop an entity relationship diagram that represents the information requirements of the business. Logical data modeling is independent of the hardware or software to be used for the implementation. The example in the slide is an entity relationship diagram that represents the information requirements of the Human Resources department.

A logical data model includes both graphic and textual components. On the surface of the diagram, you can view the relationships among the data. In the textual descriptions, you can understand the definitions behind the data and the details of the relationships among the objects.

Benefits of Creating an Entity Relationship Diagram

Entity relationship diagrams provide:

- A clear and precise format to an organization's information requirements
- A picture that users can easily understand
- A way to easily develop and refine the diagram
- A clear picture of the scope of the project
- A framework for integrating multiple applications, development projects, and purchased application packages



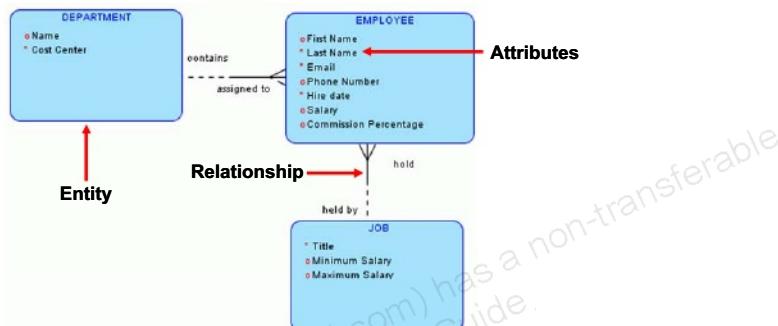
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Using an entity relationship diagram to convey the information requirements is a useful communication mechanism. Users can easily understand and change the model as it is developed. It is better to spend time in advance to validate the business rules before making implementation decisions.

Components of an Entity Relationship Diagram

Entity relationship diagrams contain three different objects.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

An entity relationship diagram has three main components:

Component	Purpose
Entity	An object or concept about which you want to store information
Relationship	A natural association that exists between two or more entities
Attributes	Descriptions of entities and specific pieces of information that must be known

Entity

An entity:

- Is information that must be tracked
- Is a name for things that you can list
- Usually has a noun as its name

Examples:



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

An entity is something of interest. Entities are categories of things that are important for a business and about which information must be kept. Entities contain facts and information that the business must know and remember. Some examples of entities might include the following:

- **PERSON:** Agent, insured, employee, customer
- **PLACE:** State, country, municipality
- **THING:** Inventory item, vehicle, product
- **CONCEPT:** Policy, risk, coverage, job
- **ORGANIZATION:** Agency, department
- **Event:** Service request, claim, election

Entity characteristics include the following:

- Is represented by a rectangular box
- Has a unique name, usually in noun form
- Has name in uppercase, with no hyphens or underscores

Entity Types

An entity can be classified into one of the following types:

Name	Description	Example
Prime	Exists independently	Customer, Instructor
Characteristic	Exists because of another (prime) entity	Order, Class Offering
Intersection	Exists because of two or more entities	Order Item, Class Enrollment



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Entities and Instances

Entities contain instances.

Entity	Instance
PERSON	John Smith
PRODUCT	2.5 x 35 mm copper nail
PRODUCT TYPE	Nail
JOB	Violinist
SKILL LEVEL	Fluent
DVD TYPE	DVD-R

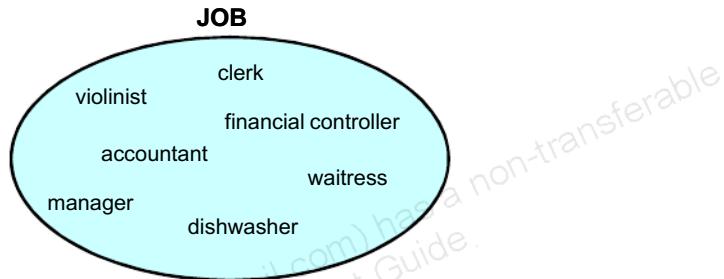


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Entities contain instances of the entity. Some entities have many instances; some have only a few. An instance of one entity may be an entity in its own right. The instance “violinist” of the entity JOB could be the name of another entity with instances like “David Oistrakh” or “Kyung-wha Chung” (who are famous violinists).

Entities Represent Sets

Entities represent a set of instances that are of interest to a particular business.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can regard entities as sets. The slide example shows a set `JOB`, and the set shows some of its instances.

At the end of the entity modeling process, entities are typically transformed into tables, with each row of a table representing an individual instance.

Quiz

Which of the following is not a candidate for an entity?

- a. PERSON
- b. ORDER
- c. FEMALE
- d. SKILL

Answer: c

Quiz

Which of the following is an instance of an entity?

- a. Name
- b. ORDER
- c. John Smith
- d. EMPLOYEE

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

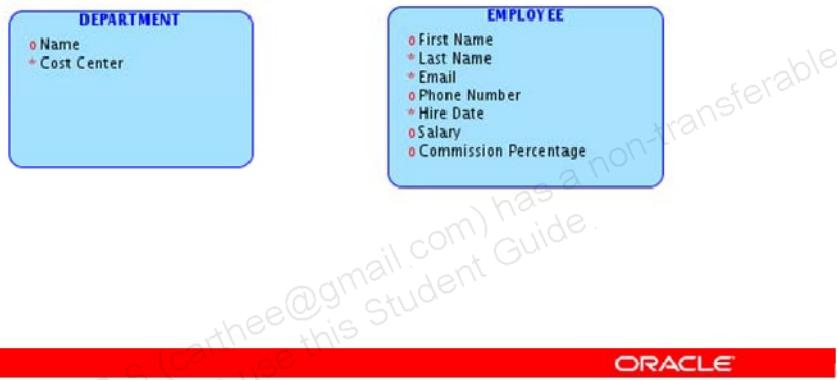
Answer: c

Name is an attribute because it describes an entity. ORDER and EMPLOYEE are considered to be entities. John Smith is an instance of an entity.

Attributes

Attributes describe entities and are the specific information that must be known.

Examples:



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Attributes are information about an entity that must be known or held. Attributes describe an entity by qualifying, identifying, classifying, quantifying, or expressing the state of the entity. Attributes represent a type of description or detail, not an instance.

Attribute names are singular and are represented within the entity box. The size of the entity boxes can be expanded to show all the attribute names.

Attribute Characteristics

- Attributes are shown within the entity box.
- Attribute names are singular and shown in mixed case or lowercase.
- The name of the attribute should not include the name of the entity, because attributes are qualified with the entity name.
- Attributes are either:
 - Not null (nulls are not allowed), indicated by *
 - Optional (nulls are allowed), indicated by an o



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Attributes have the following characteristics:

- Attributes are displayed within the entity box on the ERD.
- Attribute names should be singular and in mixed case or lowercase.
- Attributes are qualified with name of the entity. Therefore, the attribute name should not include the name of the entity. For example, rather than `employee_phone_number`, use `phone_number`.
- Attributes are classified as one of the following:
 - **Not null:** Indicated by the asterisk * symbol next to the attribute
 - **Optional** (nulls allowed): Indicated by the o (optional) symbol next to the attribute

Class Practice: Identify Entities and Attributes

The Training Manager introduced in Unit II wants to model the entity and attributes for a set of requirements. The entity and attributes must store the following:

- Course information (including its title, duration, and fee)
- Information about an instructor (such as name, phone number, and address)
- Information about each student (such as name, phone number, and address)
- Information about what courses a student took
- Information about what courses an instructor taught
- Training location information (such as name, location, and phone number)



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this practice, you identify and model the entities from the set of information requirements in the slide.

Summary

In this lesson, you should have learned how to identify entities and attributes from a set of information requirements in a case study.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 6-1 Overview: Identify Entities and Attributes

This practice covers the following topics:

- Identifying and modeling the entities and attributes from a set of information requirements
- Describing the entities

Practice 6-2 Overview: Identify Entities and Attributes

This practice covers the following topics:

- Identifying and modeling entities
- Describing an entity
- Identifying and modeling attributes

Unauthorized reproduction or distribution prohibited. Copyright© 2013, Oracle and/or its affiliates.

KARTHIKEYAN R S (carthee@gmail.com) has a non-transferable
license to use this Student Guide.

Identifying Relationships

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Create a relationship between two entities
- Model relationships by using a relationship matrix



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you identify and create relationships in an entity relationship diagram (ERD).

Lesson Agenda

- Create a relationship between two entities
- Model relationships by using a relationship matrix

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Relationships

A relationship is a bidirectional, significant association between two entities, or between an entity and itself.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A relationship represents the business rules that link entities. There are always two business rules for each relationship. In the example in the slide, the business rules are:

- A DEPARTMENT may contain one or many EMPLOYEES.
- An EMPLOYEE must be assigned to one and only one DEPARTMENT.

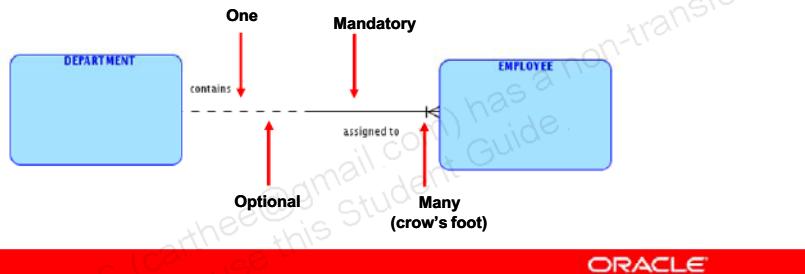
Each direction of a relationship has:

- A name (for example, "contain one" or "assigned to")
- An optionality (for example, either "must be" or "may be")
- A degree for example (for example, either "one and only one" or "one or more")

Components of a Relationship

Each direction of a relationship has:

- A name
 - Cardinality
 - Minimum value: Optional or Mandatory
 - Maximum value: One or Many



The components of the relationship include the following:

- **Name:** The label that appears close to the entity it is assigned to. Make sure that all relationship names are in lower case (examples: “assigned to” or “responsible for”).
 - **Cardinality:** The minimum and maximum number of values in the relationship
 - Minimum values can be either optional (zero) or mandatory (at least one).
 - Maximum values can be either one or many.

When reading the business rule sentence, use the following words for the minimum values:

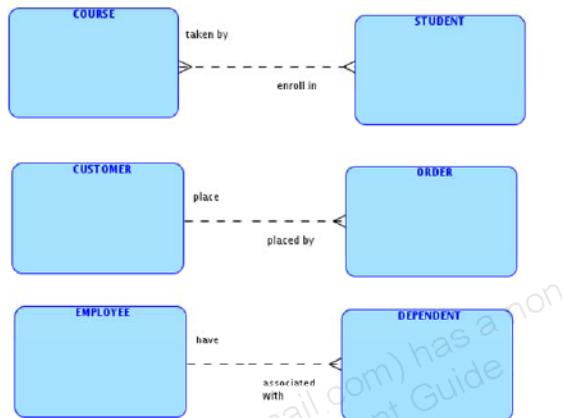
- **Optional:** Use “may be” or “may.”
 - **Mandatory:** Use “must be” or “must”

When reading the business rule sentence, use the following words for the maximum values:

- **Line:** Use “one and only one.”
 - **Crow’s feet:** Use “one or more.”

Business rule syntax is as follows:

Relationships: Additional Examples



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In order to establish the business rules for each relationship, first read a relationship in one direction and then read the relationship in the other direction.

In the examples in the slide, note the following:

Between STUDENT and COURSE, the business rules are:

- A COURSE may be taken by one or more STUDENTS.
- A STUDENT may be enrolled in one or more COURSES .

Between CUSTOMER and ORDER, the business rules are:

- A CUSTOMER may place one or more ORDERS.
- An ORDER must be placed by one and only one CUSTOMER.

Between EMPLOYEE and DEPENDENT, the business rules are:

- An EMPLOYEE may have one to many DEPENDENTS .
- A DEPENDENT may be associated with one and only one EMPLOYEE .

Quiz

Which of the following is a valid business rule between two entities?

- a. A PERSON may be assigned a LICENSE.
- b. An ITEM must be stored in one and only one WAREHOUSE.
- c. A ORDER may originate from a CUSTOMER.
- d. An EMPLOYEE has one or many SKILLS.

ORACLE®

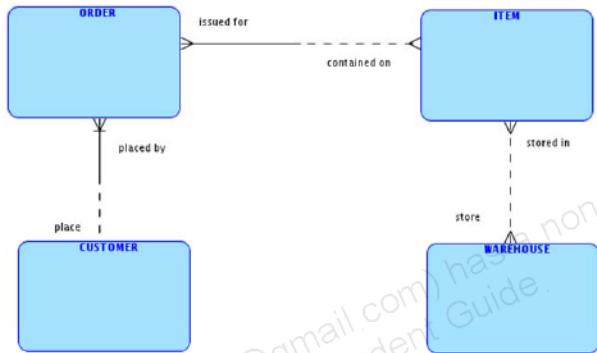
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

How would you correct the incorrect business rules to make them valid?

Class Practice: Define Business Rules

In this practice, you write the business rule sentences for the following ERD:



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Relationship Types

All relationships represent the information requirements and the rules of the business.

- Many-to-one (M:1) or one-to-many (1:M)
- Many-to-many (M:M)
- One-to-one (1:1)



Example of a 1:M relationship

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

There are three types of relationships:

- **Many to one (M:1) or one to many (1:M):** There are crow's feet on one side of the relationship. The direction of the crow's feet determines whether the relationship is M:1 or 1:M. This type of relationship is the most common.
- **Many to many (M:M):** There are crow's feet on both sides of this relationship. It is common to see M:M relationships in a high level ERD at the beginning of a project.
- **One to one (1:1):** This type of relationship is a line without crow's feet on either end. These types of relationships are rare.

Note: In Oracle SQL Developer Data Modeler, the notation is slightly different:

- One to many is 1:N.
- Many to many is M:N.

Many-to-One and One-to-Many Relationships

Many-to-one and one-to-many relationships (M:1 and 1:M) have cardinality of *one or more* in one direction and *one and only one* in the other direction.



Business rules:

- Each CUSTOMER must be visited by *one and only one* SALES REPRESENTATIVE.
- Each SALES REPRESENTATIVE may be assigned to *one or more* CUSTOMERS.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Many-to-Many Relationships

Many-to-many relationships (M:M) have cardinality of *one or more* in both directions.



Business rules:

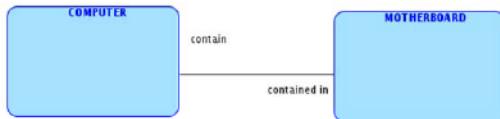
- Each EMPLOYEE may be assigned to *one or more* JOBS.
- Each JOB may be carried out by *one or more* EMPLOYEES.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

One-to-One Relationships

One-to-one relationships (1:1) have cardinality of *one and only one* in both directions.



Business rules:

- Each COMPUTER must contain *one and only one* MOTHERBOARD.
- Each MOTHERBOARD must be contained in *one and only one* COMPUTER.

ORACLE®

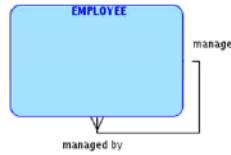
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The example in the slide is a one-to-one relationship because the cardinality is a line with no crow's feet in either direction.

These types of relations are the least common because they may instead be one entity with attributes contained in that entity.

Recursive Relationships

A relationship with an entity and itself



Business rules:

- Each EMPLOYEE may manage *one or more* EMPLOYEES.
- Each EMPLOYEE must be managed by *one and only one* EMPLOYEE.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Quiz

Which type of relationship do the following business rules reflect?

A PERSON may make one or more TICKET RESERVATIONS.

A TICKET RESERVATION must be made by one and only one

PERSON

- a. 1:M
- b. M:M
- c. Recursive
- d. 1:1

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a

Lesson Agenda

- Create a relationship between two entities
- Model relationships by using a relationship matrix

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Using a Relationship Matrix

A relationship matrix can be used to collect initial information about the relationships among a set of entities.

	CUSTOMER	ITEM	ORDER	WAREHOUSE
CUSTOMER			place	
ITEM			issued order	stored
ORDER	placedby	containedon		
WAREHOUSE		store		



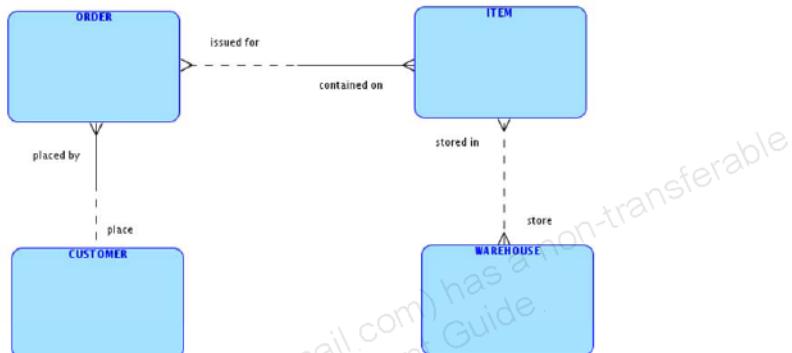
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A relationship matrix has the following characteristics:

- A relationship matrix shows if and how each row entity on the left side of the matrix is related to each column entity shown across the top of the matrix.
- All the entities are listed along both the left side of the matrix and the top of the matrix.
- If a row entity is related to a column entity, the name of that relationship is shown in the intersection box
- If a row entity is not related to a column entity, the intersection box is empty.
- Each relationship above the diagonal line is the inverse or mirror image of a relationship below the line.
- Recursive relationships are represented by the boxes on the diagonal.

Using a Relationship Matrix

Map the contents of the relationship matrix to an ERD.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To map the matrix to an ERD, you draw a box for each entity and then draw the relationship and state the business rule. Doing so helps to determine the type of each relationship and its cardinality.

To help in the transformation between the matrix and the ERD, follow these steps:

1. Determine a relationship's existence.
2. Name the relationship.
3. Determine the relationships cardinality.

Determining a Relationship's Existence

Examine each pair of entities to determine whether a relationship exists.

	ACTIVITY	DEPARTMENT	EMPLOYEE
ACTIVITY			✓
DEPARTMENT			✓
EMPLOYEE	✓	✓	

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The first step to creating a relationship is to determine its existence. Ask “Does a significant relationship exist between ENTITY A and ENTITY B?

In the example in the slide, consider the following questions:

- Is there a significant relationship between DEPARTMENT and EMPLOYEE? Yes.
- Is there a significant relationship between DEPARTMENT and ACTIVITY? No.
- Is there a significant relationship between ACTIVITY and EMPLOYEE? Yes.

Log the relationships among ACTIVITY, DEPARTMENT, and EMPLOYEE on a relationship matrix. The check marks indicate that a relationship exists.

A relationship matrix is used to systematically examine each pair of entities.

Naming the Relationship

Name each direction of a relationship.

	ACTIVITY	DEPARTMENT	EMPLOYEE
ACTIVITY			assigned to
DEPARTMENT			composed of
EMPLOYEE	participate in	assigned to	

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Name each direction of a relationship. Relationship names represent a role and tend to be passive verbs, noun role names, or prepositions. Try not to use “related to” or “associated with” as relationship names because they lack specific meaning and are weak verbs.

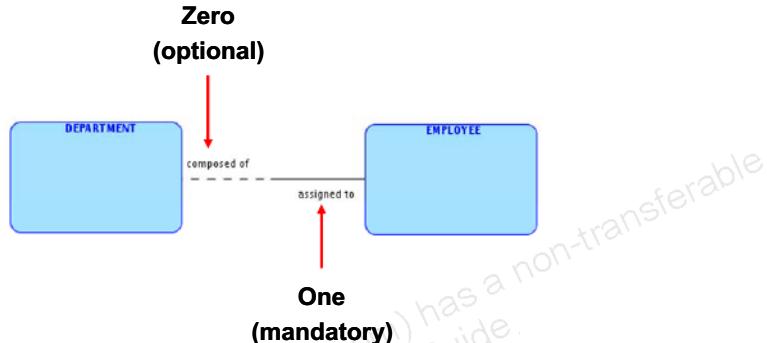
The following is a sample list of relationship name pairs to assist in naming relationships:

- basedon thebasisfor
- boughtfrom thesupplierof
- descriptionof for
- operatedby theoperatorfor
- representedby therepresentationof
- responsiblefor theresponsibilityof

Log the relationship names in the relationship matrix as shown in the slide.

Determining the Relationship's Cardinality

1. What is the minimum cardinality in each direction?



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The first question to answer is what is the minimum cardinality for each direction of the relationship?

In the example in the slide, answer the following questions:

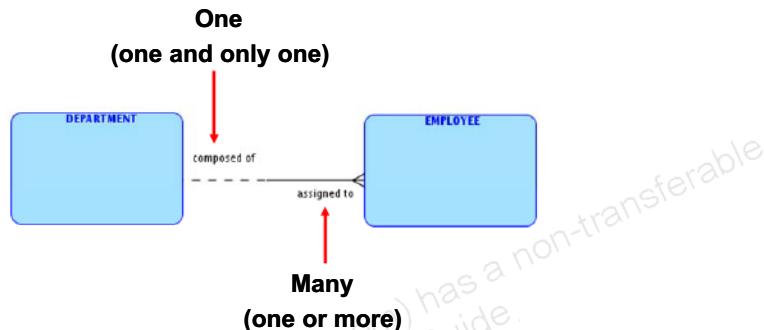
1. Must an **EMPLOYEE** be assigned to a **DEPARTMENT**? Always
2. Is there any situation in which an **EMPLOYEE** will not be assigned to a **DEPARTMENT**?
No, an **EMPLOYEE** must always be assigned to a **DEPARTMENT**. (Mandatory)
3. Must a **DEPARTMENT** be composed of an **EMPLOYEE**?
No, a **DEPARTMENT** does not have to be composed of an **EMPLOYEE** (Optional)

When the minimum cardinality is optional, the value could be zero. When the minimum cardinality is mandatory, the value must be at least one.

Note that the relationship line in the slide was intentionally drawn without the maximum cardinality.

Determining the Relationship's Cardinality

2. What is the maximum cardinality in each direction?



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The second question to answer is what is the maximum cardinality for each direction of the relationship?

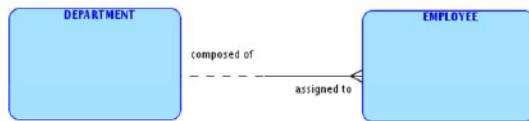
In the example in the slide, answer the following questions:

- Must an **EMPLOYEE** be assigned to more than one **DEPARTMENT**?
No, an **EMPLOYEE** must always be assigned to one and only one **DEPARTMENT**. (One)
- May a **DEPARTMENT** be composed of more than one **EMPLOYEE**?
Yes, a **DEPARTMENT** may be composed of one or more **EMPLOYEEs** (Many)

When the maximum cardinality is one, the value can only be one. When the maximum cardinality is many, the value can be one or more.

Validating the Relationship

Re-examine the ERD and validate the relationship.



- Each EMPLOYEE must be assigned to one and only one DEPARTMENT.
- Each DEPARTMENT may be composed of one or more EMPLOYEES.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Read the relationship aloud to make sure that it is readable and makes business sense.

Quiz

Which pair of entities has a valid relationship?

- a. ORDER and ORDER_ITEM
- b. PHONE_NUMBER and EMPLOYEE
- c. ORDER_ITEM and EMPLOYEE
- d. BOOK and MUSIC

ORACLE®

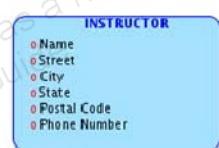
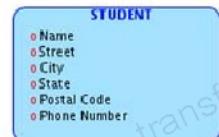
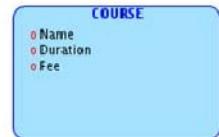
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a

Answer (b) could also be correct because there may be an unknown number of PHONE_NUMBERS (home, cell, work, and so on) for each EMPLOYEE.

Class Practice: Build a Relationship Matrix

1. Build a relationship matrix for some of the entities that you identified previously.
2. Establish the relationships in the diagram.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this practice, you build a relationship matrix for the entities that you identified in the lesson titled “Identifying Entities and Attributes” for the training scenario. Then you create the relationships in the diagram from the relationship matrix.

Summary

In this lesson, you should have learned how to:

- Create a relationship between two entities
- Model relationships by using a relationship matrix



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 7-1 Overview: Analyze and Model Relationships

This practice covers the following topics:

- Analyzing and modeling relationships based on a set of information requirements
- Using a relationship matrix to track the existence of relationships among the following entities
 - MEMBERSHIP
 - ORGANIZATION
 - CUSTOMER

Practice 7-2 Overview: Analyze and Model Relationships

This practice covers the following topics:

- Analyzing and modeling relationships based on a set of information requirements
- Using a relationship matrix to track the existence of relationships among the following entities:
 - RESERVATION
 - HOTEL
 - ROOM



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2013, Oracle and/or its affiliates.

KARTHIKEYAN R S (carthee@gmail.com) has a non-transferable
license to use this Student Guide.

Assigning Unique Identifiers

8

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to identify unique identifiers for entities and relationships.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

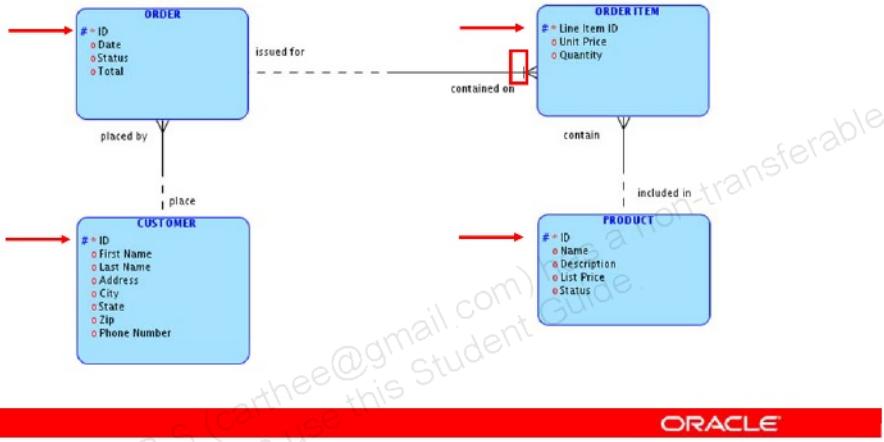
- Introduction to Unique Identifiers
- Primary and Secondary Unique Identifiers

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Unique Identifier

A special attribute (or group of attributes) that uniquely identifies a particular occurrence of a data entity



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

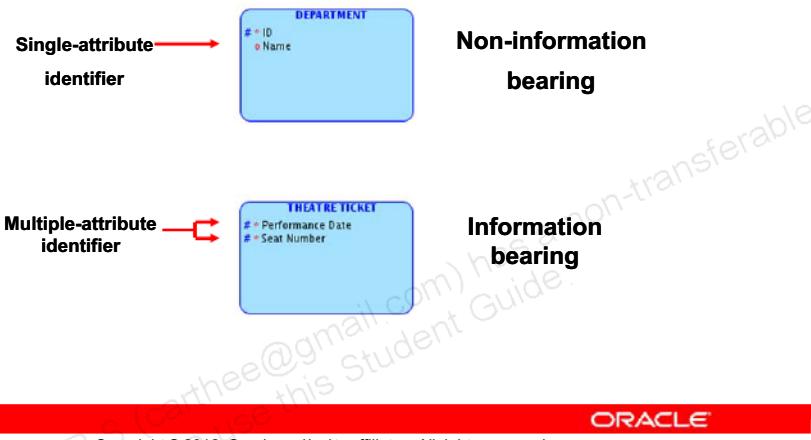
ORACLE®

A unique identifier (UID) is a special attribute (or group of attributes) that uniquely identifies a particular occurrence of a data entity. A unique identifier attribute is designated with a # symbol. Each component of a unique identifier must be mandatory.

In the example in the slide, the unique identifier for ORDER is Order ID, for CUSTOMER it is CUSTOMER ID, and for PRODUCT it is PRODUCT ID. The unique identifier for ORDER ITEM is a composite between Line Item ID and the relationship with the ORDER entity. The vertical line on the relationship indicates that it is part of the unique identifier in the ORDER ITEM entity (also called an *identifying relationship*). This concept is discussed later in this lesson.

Unique Identifier: Examples

Each entity must have a unique identifier. Otherwise, it is not an entity.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Each entity must have a unique identifier; otherwise, it is not an entity. The unique identifier can be information bearing, when the values of the unique identifier have some business meaning. A non-information-bearing unique identifier is sometimes referred to as a *synthetic key*. An information-bearing unique identifier is sometimes referred to as a *natural key*.

In the example in the slide, the unique identifier for the DEPARTMENT entity is the ID attribute. Notice that the DEPARTMENT entity unique identifier contains one attribute and the attribute has no business meaning, which means that it is not information bearing.

For the THEATRE TICKET entity, the unique identifier is the combination of the Performance Date and the Seat Number attributes. In this case, the unique identifier is more than one attribute, and it is information bearing.

Identifying Relationships

What is the unique identifier of the ACCOUNT entity?



The unique identifier requires both the ACCOUNT Number and the relationship between BANK and ACCOUNT.



ORACLE®

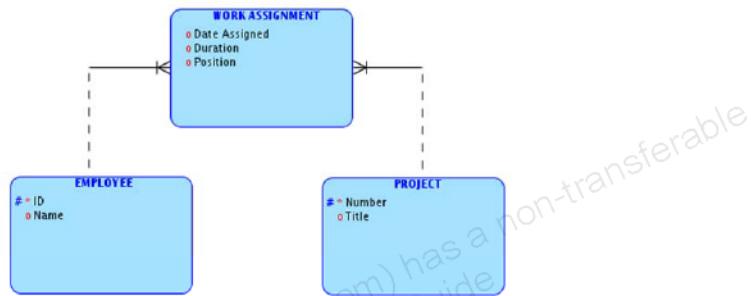
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

An identifying relationship is created when the unique identifier for an entity includes the relationship with another entity for it to be unique. In the example in the slide, the unique identifier for the ACCOUNT entity is the ACCOUNT Number as well as the relationship between BANK and ACCOUNT. The identifying relationship is depicted with a vertical bar on the relationship line.

Note that a relationship included in a unique identifier must be mandatory and one-and-only-one in the direction that participates in the unique identifier.

Identifying Relationships with Multiple Entities

An entity can be uniquely identified through multiple relationships.



ORACLE®

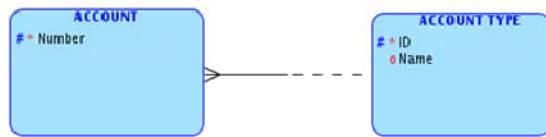
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

An entity may be uniquely identified through multiple relationships. In the example in the slide, an EMPLOYEE and PROJECT are needed to make WORK ASSIGNMENT unique, so both relationships are included in the unique identifier for WORK ASSIGNMENT.

Note that WORK ASSIGNMENT is an intersecting entity that is the resolution between an M:M relationship. This topic is discussed in a later lesson.

Non-Identifying Relationship

In a non-identifying relationship between two entities, the relationship is not required to be part of the unique identifier.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A non-identifying relationship is created when the unique identifier for an entity does not need the relationship with another entity for it to be considered unique.

In the example in the slide, the ACCOUNT entity does not need the relationship with ACCOUNT TYPE for it to be considered unique. There can be instances of an ACCOUNT TYPE that do not have an ACCOUNT.

Lesson Agenda

- Introduction to Unique Identifiers
- Primary and Secondary Unique Identifiers

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Primary and Secondary Unique Identifiers

An entity can have more than one unique identifier.

Candidate unique identifiers:

- Badge number
- Payroll number



Select one candidate unique identifier to be the primary unique identifier, and the others to be secondary unique identifiers.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

An entity can have more than one unique identifier. In the example in the slide, there are two candidate unique identifiers for the `EMPLOYEE` entity: badge number and payroll number.

When this situation occurs, select one candidate unique identifier to be the primary unique identifier, and the others to be secondary unique identifiers.

Note that there is no standard diagramming convention for tagging secondary unique identifiers.

Searching for Unique Identifiers

Evaluate your attributes:

- What mandatory attributes identify the entity? Seek out additional attributes that help identify the entity. Consider creating artificial attributes for identification.
- Does an attribute uniquely identify the entity?
- What combination of attributes uniquely identifies the entity?

Consider your relationships:

- Which of the relationships help identify the entity?
- Are there missing relationships that help identify the entity?
- Does the relationship help uniquely identify the entity?
- Is the relationship mandatory and one-and-only-one in the direction from the entity?

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can search for unique identifiers by evaluating your attributes and relationships using the questions supplied in the slide.

To validate the unique identifier for each entity, consider the following:

- Examine with sample data. Does the selected combination of attributes and relationships uniquely identify each instance of an entity?
- Are all the attributes and relationships that are included in the unique identifier mandatory?
- Do you have control over the unique identifier? For example, if you choose U.S. Social Security Numbers (SSN) as the unique identifier, can these numbers be reassigned? If so, this is not the best choice for your unique identifier.

Quiz

A unique identifier:

- a. Is always created through a relationship
- b. Must be one attribute in an entity
- c. Must uniquely identify an occurrence of an entity
- d. Must be a non-identifying relationship

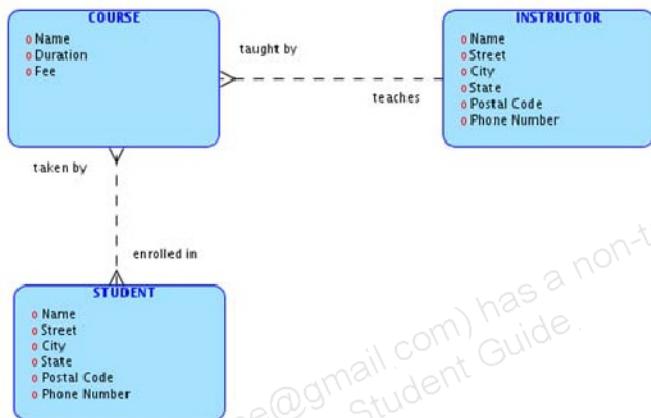
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Class Practice: Specify Unique Identifiers

Specify unique identifiers for each entity in the diagram.



Summary

In this lesson, you should have learned how to identify unique identifiers for entities and relationships.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 8-1 Overview: Identify Unique Identifiers

This practice covers identifying unique identifiers for the following entities in your entity relationship diagram:

- MEMBERSHIP
- ORGANIZATION
- CUSTOMER

In this practice, you identify unique identifiers for the ERD that you built in Practice 7-1.

Practice 8-2 Overview: Identify Unique Identifiers

This practice covers identifying unique identifiers for the following entities in your entity relationship diagram:

- GUEST
- RESERVATION
- HOTEL
- ROOM



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this practice, you identify unique identifiers for the ERD that you built in Practice 7-2.

Using Oracle SQL Developer Data Modeler to Create an Entity Relationship Diagram

9

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Examine the General options for logical data modeling
- Build an Erd in Oracle SQL Developer Data Modeler
- Edit the layout of an Erd
- Create a subview
- Create a display



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

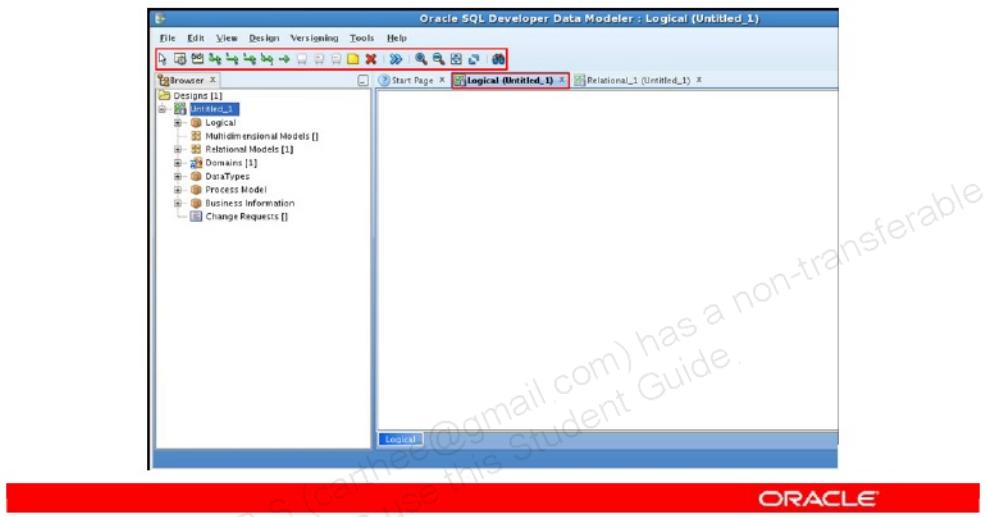
- Examine the General options for logical data modeling
- Build an ERD in Oracle SQL Developer Data Modeler
- Edit the layout of an ERD
- Create a subview
- Create a display

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Building an Entity Relationship Diagram

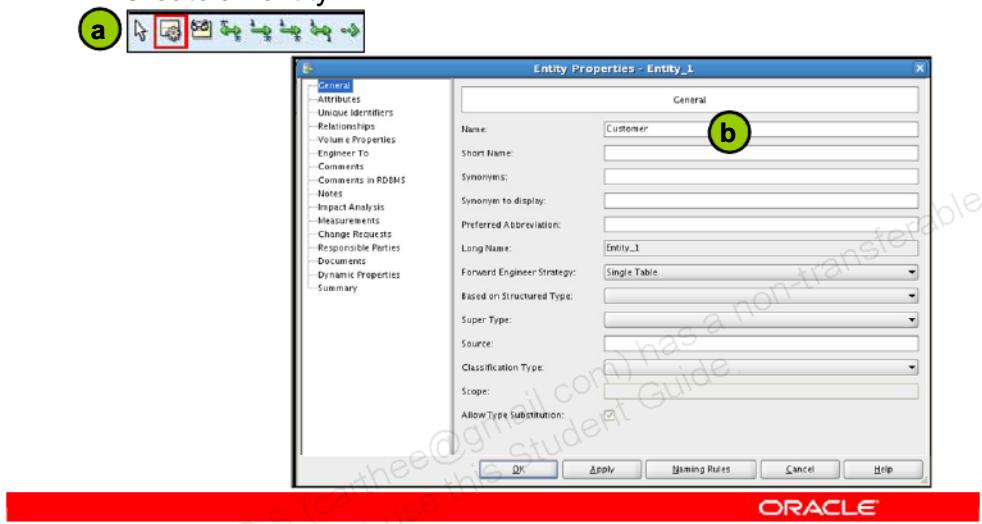
1. Click the Logical tab.



The first step to building an entity relationship diagram in Oracle SQL Developer Data Modeler is to click the Logical tab. Notice that the toolbar changes to display tools specifically for working with entity relationship diagrams.

Building an Entity Relationship Diagram

2. Create an entity.

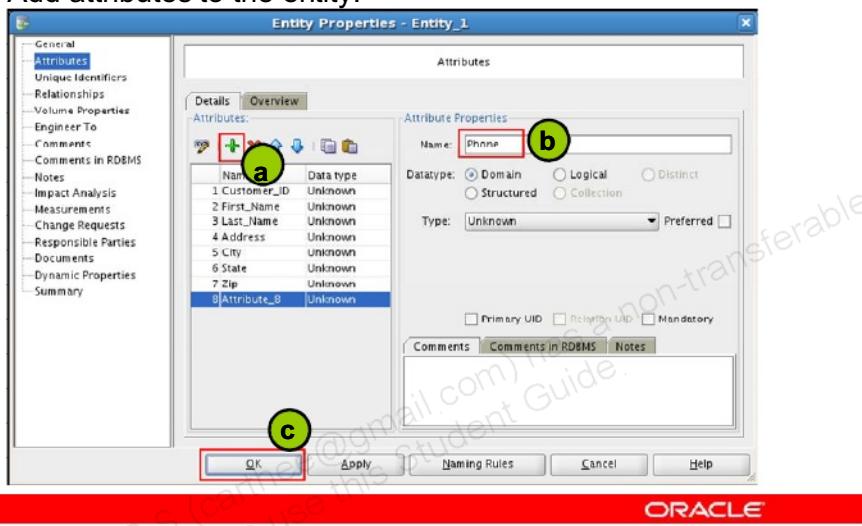


To create an entity, perform the following steps:

- On the toolbar, click the **New Entity** tool, and click anywhere in the white space of the **Logical** pane. The **Entity Properties** window appears.
- In the **Entity Properties** window, enter the name of the entity and other pertinent information.

Building an Entity Relationship Diagram

3. Add attributes to the entity.



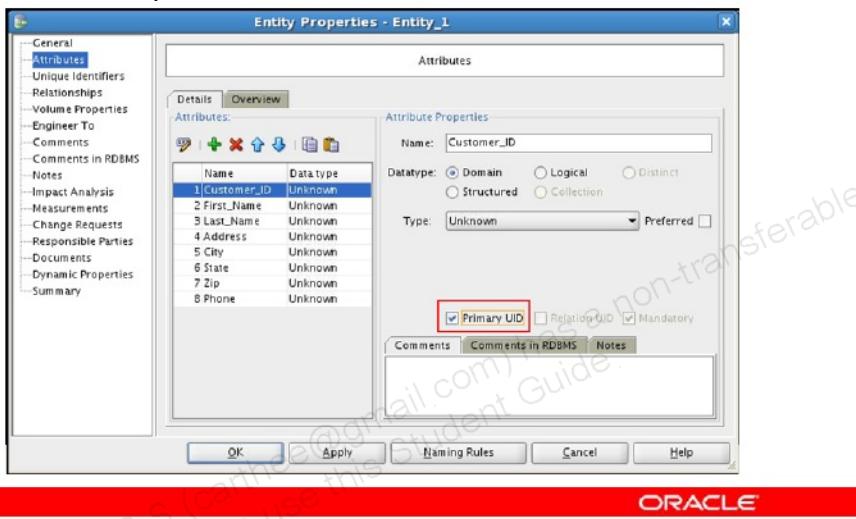
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To add attributes to the entity, perform the following steps:

- With **Attributes** selected in the left navigator of the **Entity Properties** window, click the **Add an Attribute** icon.
- In the **Name** field, enter a name for the attribute.
Repeat steps **a** and **b** until you have created all the attributes for this entity. Use the up and down arrows to reorder the columns.
- When all the attributes have been created, click **OK** to create the entity and attributes.

Building an Entity Relationship Diagram

4. Set the unique identifier.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE

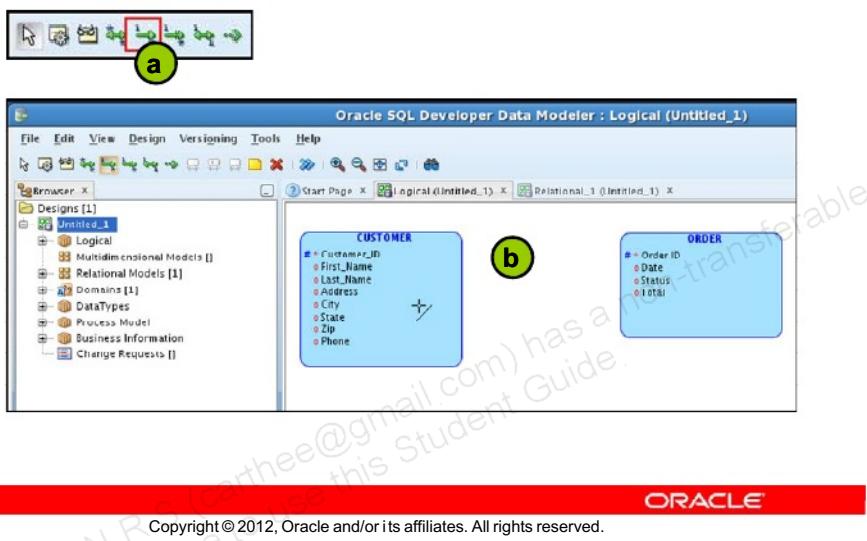
To set the unique identifier for the entity, perform the following steps:

1. Click the entity for which you want to define the unique identifier.
2. With **Attributes** selected in the left navigator of the **Entity Properties** window, select the attribute that you want to assign as the unique identifier.
3. Select the **Primary UID** check box.

Note: Primary UID means "primary key." The attribute that you assign as primary UID is automatically set to a mandatory attribute and will be engineered to a primary key in the relational model.

Building an Entity Relationship Diagram

5. Define the relationships between entities.

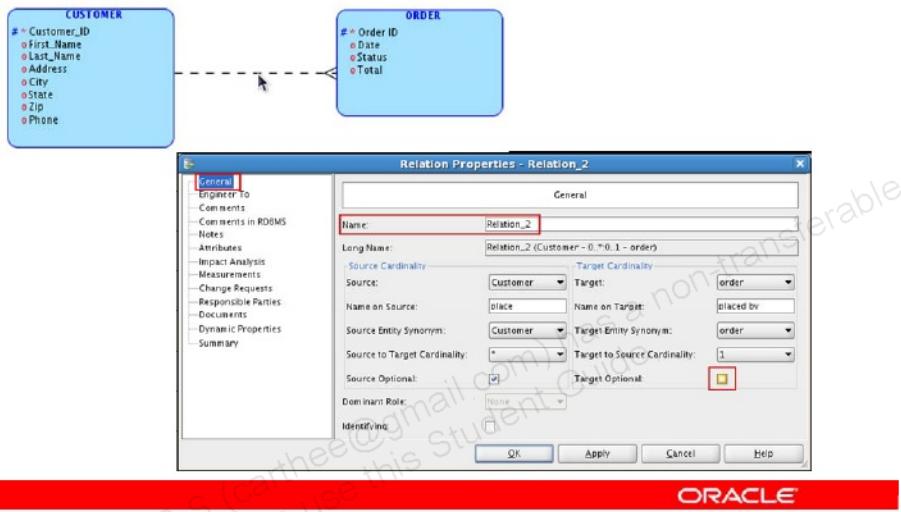


To define the relationships between entities, perform the following steps:

- Click the desired relationship type on the toolbar.
- Click the source entity and then the target entity. A relationship is created.

Building an Entity Relationship Diagram

6. Set the source and target values for the relationship.



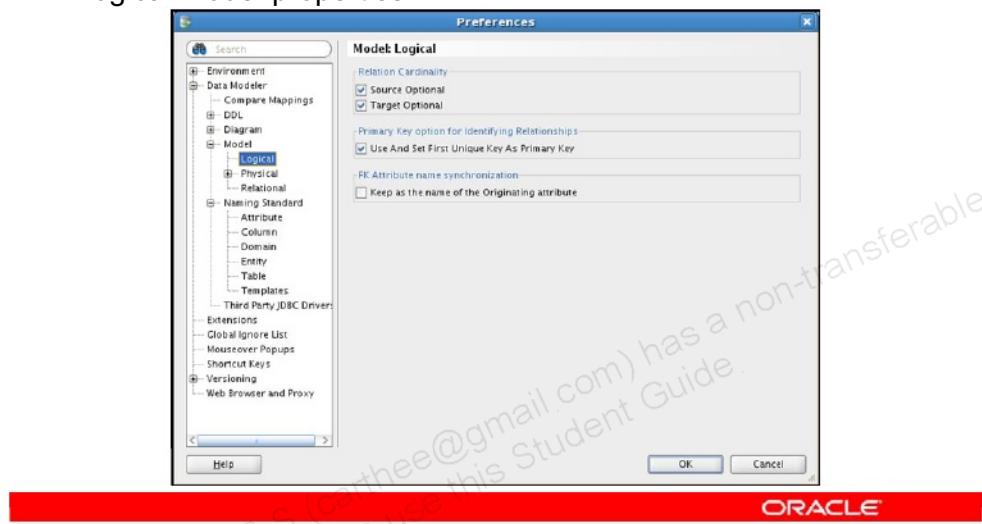
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To set the source and target values for the relationship, perform the following steps:

- Double-click the relationship in the diagram.
- Select the **General** property in the left navigator.
- Specify the name of the relationship.
- Specify the source and target names for the relationship. Note that these names will be specified in the diagram and will be used to validate the business rules for the relationship.
- Specify the minimum and maximum cardinality for the relationship. The **Source Optional** option controls whether the source entity in the relation must contain one or more instances. The **Target Optional** option controls whether the target entity in the relationship must contain one or more instances. In the example in the slide, the check box for **Target Optional** was deselected because there must be a **CUSTOMER** on each **ORDER**.
- Specify whether this is an identifying relationship by selecting the Identifying option.

Specifying Logical Model General Option

Logical Model properties



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Within Oracle SQL Developer Data Modeler, you can set various properties for the Logical Model. To open the Preferences window:

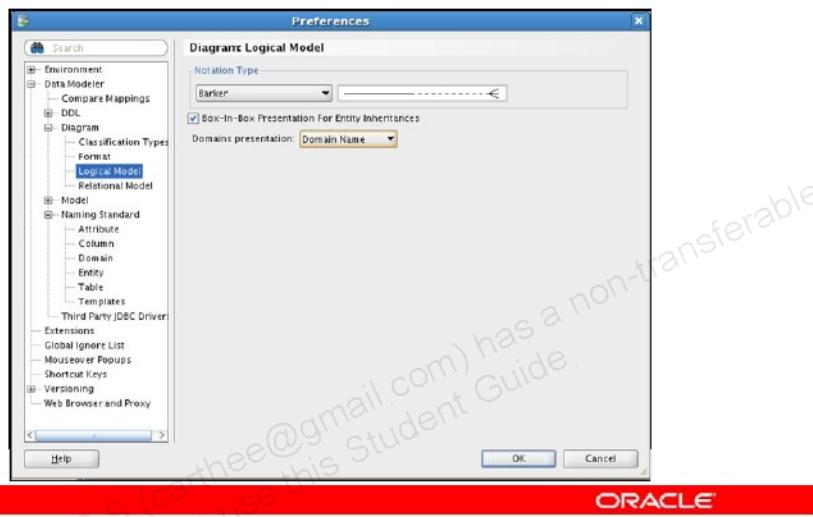
1. Select **Tools > Preferences**.
2. Expand **Data Modeler > Model**.
3. Select **Logical**.

The Logical Model pane contains options that affect the startup and overall behavior and appearance of SQL Developer Data Modeler. You can set the following:

- **Relation Cardinality: Source Optional:** Controls whether the source entity in a relationship must, by default, contain one or more instances. If this option is enabled, source instances are not required for all relationship types; if this option is disabled, one or more source instances are required for all relationship types.
- **Relation Cardinality: Target Optional:** Controls whether the target entity in a relationship must, by default, contain one or more instances. If this option is enabled, target instances are not required for all relationship types; if this option is disabled, one or more target instances are required for all relationship types.
- **Use And Set first Unique Key As Primary key:** Controls whether, by default, the first unique key attribute is set as the primary key when you create an entity.
- **FK Attribute name synchronization:** Keep as the name of the Originating attribute: Controls whether the supertype or referenced attribute must be used in foreign key naming. To be able to specify some other name, deselect this check box.

Specifying Logical Model Diagram Defaults

Logical Model Diagram properties



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To open the Logical Model Diagram pane:

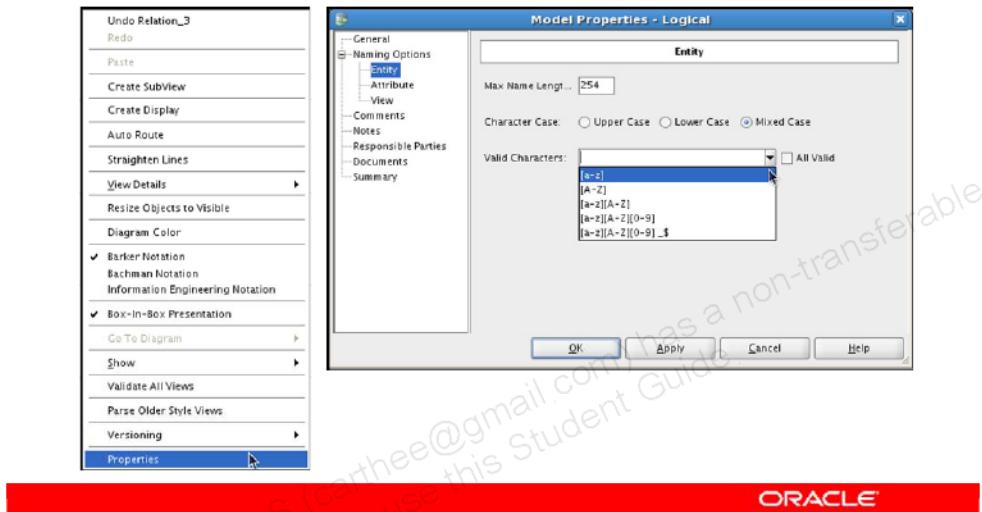
1. Select **Tools > Preferences**.
2. Expand **Data Modeler > Diagram**.
3. Select **Logical Model**.

The **Logical Model Diagram** pane contains the following options that apply to the ERD:

- **Notation Type:** Barker (sometimes called "crow's foot") or Bachman notation
- **Show Source/Target Name:** Controls whether the **Name on Source** and **Name on Target** values are displayed. If they are displayed, you can format the text and move the boxes. Note that, in the screenshot in the slide, this option was selected because you want the names to be displayed in the diagram.
- **Box-in-Box Presentation For Entity inheritances:** Displays subtypes in a box inside the supertype's box
- **Domains presentation:** Specifies what is displayed as the data type for an attribute based on a domain:
 - **Domain Name** causes the domain name to be displayed
 - **Used Logical Type** causes the logical type used in the domain definition to be displayed.

Modifying Model Properties

To change the properties for a model:



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To change the properties for the model, right-click the background of the diagram and select **Properties**.

When **General** is selected in the left navigator, you can specify the following:

- **Name:** The name of the logical model
- **Visible:** Controls whether the diagram for the logical model is displayed in the **Data Modeling** window. You can also control the visibility by selecting **Show** or **Hide** from the context menu after you right-click the logical model name in the object browser.

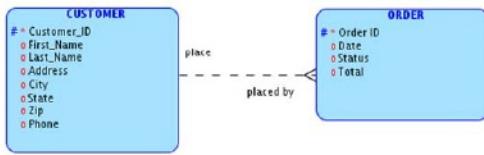
When **Naming Options** is selected in the left navigator, you can specify the following naming rules for entities, attributes, and views:

- **Max Name Length:** Specifies the maximum number of characters in the name
- **Character Case:** Controls whether you can use only uppercase or lowercase characters, or both uppercase and lowercase (that is, mixed case)
- **Valid Characters:** Specify either **All Valid** (no restrictions), or disable **All Valid** and then select the valid set of characters.

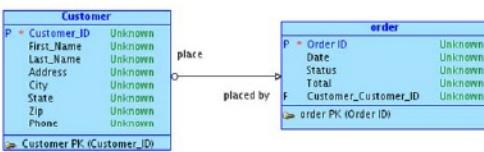
All other items in the left navigator are common for all property windows.

Notation Types

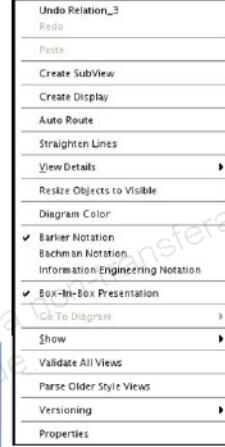
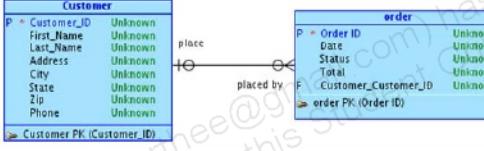
Barker
Notation



Bachman
Notation



Information
Engineering
Notation



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

There are three notation types available in Oracle SQL Developer Data Modeler. They are all correct—they just represent different data modeling methodologies.

Barker notation is used through the examples in this course.

Bachman notation has the following differences:

- **Entity:** Shows sharper cornered box instead of rounded corners
- **Attributes:** Shows * for not null and no notation for null attributes; shows P for unique identifier and F for the attribute created through the relationship (will represent the foreign key after engineering in the relational design)
- **Relationship line:** Shows an arrow instead of crow's feet for the maximum cardinality; shows an open circle or filled in circle instead of a dotted and solid line for the minimum cardinality
- **Datatype:** Shows the data type of each attribute. The default is null. This topic is discussed in the lesson titled “Adding and Using Data Types.”

Information Engineering notation has the following differences:

- **Entity:** Represented as named rectangles
- **Attributes:** Displayed in a compartment below the entity type name
- **Relationships:** Restricted to binary associations and depicted as named lines connecting to the entity types
- **Optional and cardinality settings:** Indicated by annotating the line ends:
 - To indicate that a role is optional, a circle is placed at the other end of the line.
 - To indicate that a role is mandatory, a stroke is placed at the other end of the line.
 - To depict a cardinality of "many," crow's feet are used.

Lesson Agenda

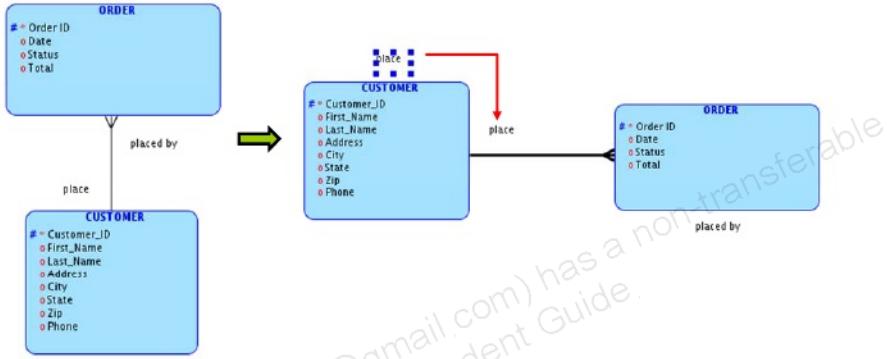
- Examine the General options for logical data modeling
- Build an ERD in Oracle SQL Developer Data Modeler
- **Edit the layout of an ERD**
- Create a subview
- Create a display

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Editing a Diagram Layout: Moving an Object

Drag the entity to another location and then release it. You may also have to drag the source and target labels.



ORACLE®

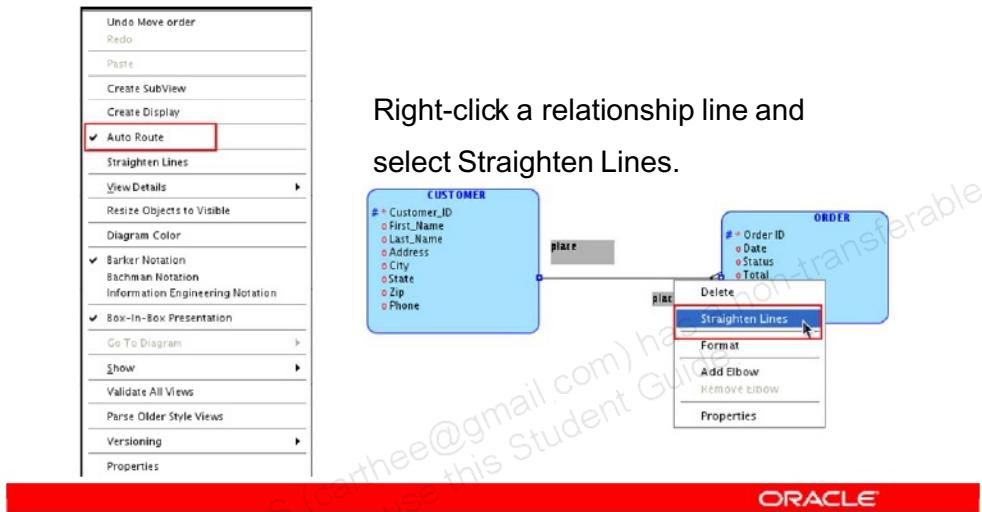
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, the CUSTOMER entity was dragged to a different location.

Note: You may have to drag the source and target names of the relationship to appropriate locations after you move the entity.

Editing a Diagram Layout: Redrawing Lines

Turn off Auto Route.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

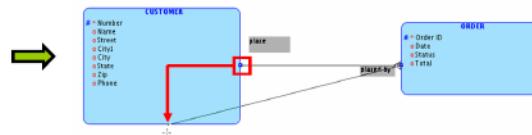
In order to redraw lines, you must first turn **Auto Route** off. Right-click in the white space and select **Auto Route** from the context menu.

After **Auto Route** is turned off, you can straighten lines by right-clicking the relationship line and selecting **Straighten Lines**.

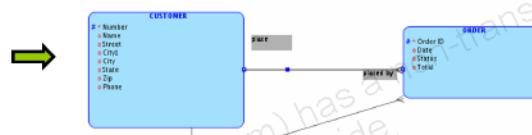
Editing a Diagram Layout: Moving a Relationship Line

To move a relationship line manually to another location:

1. Drag an anchor of the relationship to a new location.



2. Press Ctrl and drag points on the relationship line.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

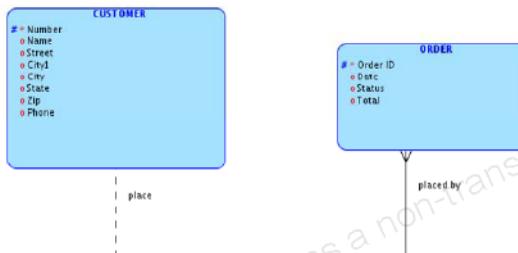
To move a relationship line manually to another location, perform the following steps:

1. Drag each anchor of the relationship to a new location. In the example in the slide, the anchor was dragged to the bottom of the **CUSTOMER** entity.
2. Press **Ctrl** and drag a point on the relationship line to a new location. An elbow will be created on the relationship line depending on where it was dragged.

Editing a Diagram Layout: Moving a Relationship Line

(continued)

3. Move the source and target names to the appropriate locations.



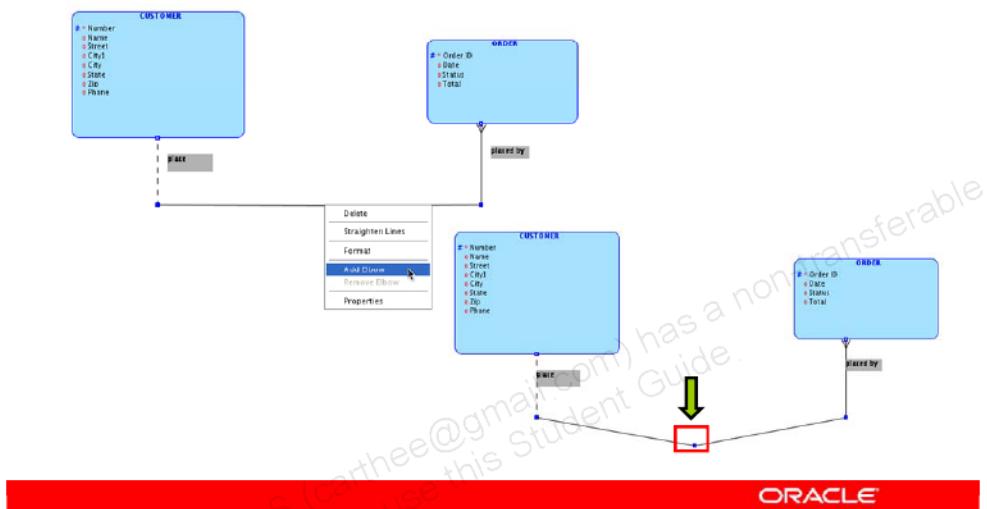
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

3. Move the source and target names for the relationship to their appropriate locations to complete the manual redraw of the relationship line.

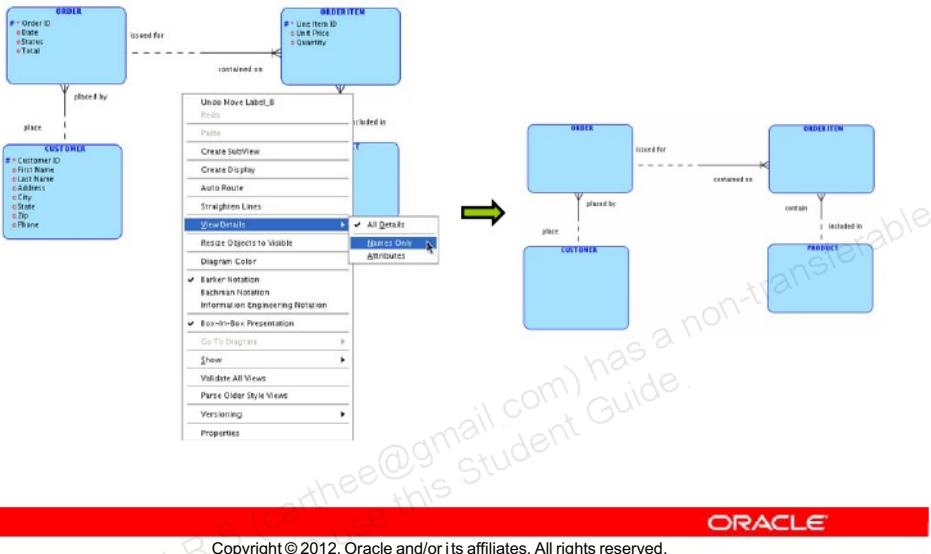
Editing a Diagram Layout: Adding an Elbow

To add an elbow to the relationship line:



To add an elbow to the relationship line, right-click the relationship line where you want the elbow to be created and select **Add Elbow**.

Editing a Diagram Layout: Showing Levels of Detail



ORACLE®

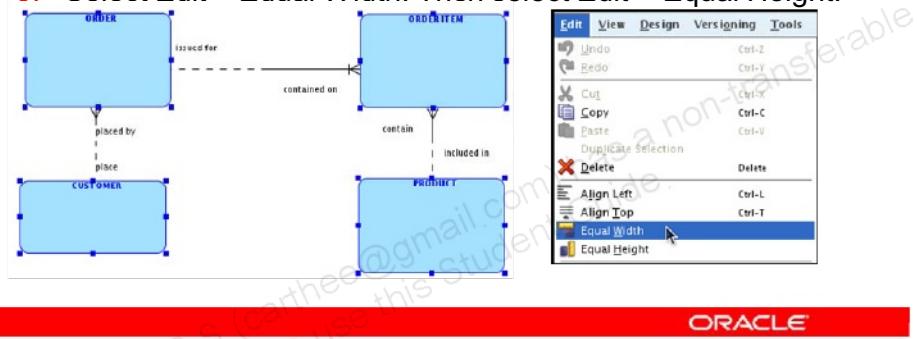
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To change the level of detail displayed, right-click the background of the diagram, select **View Details**, and then select the level that you want. In the example in the slide, **Names Only** was selected so that only the names of the entities appear.

Editing a Diagram Layout: Resizing Multiple Objects

To resize multiple objects at the same time:

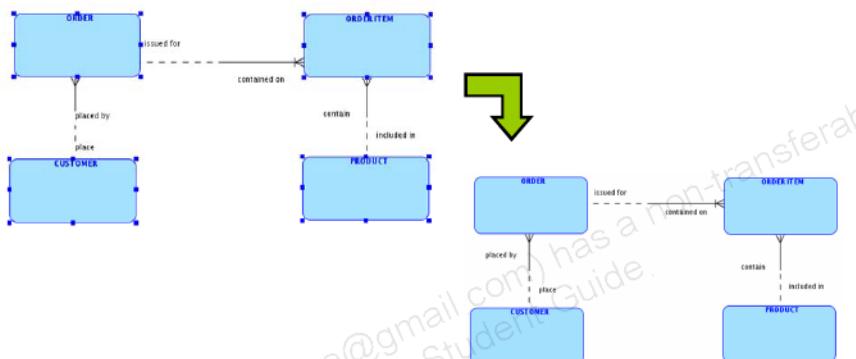
1. Resize one of the objects to the size that you want for all the objects.
2. Press Ctrl and select all the objects that you want to make the same size
3. Select Edit > Equal Width. Then select Edit > Equal Height.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Editing a Diagram Layout: Resizing Multiple Objects

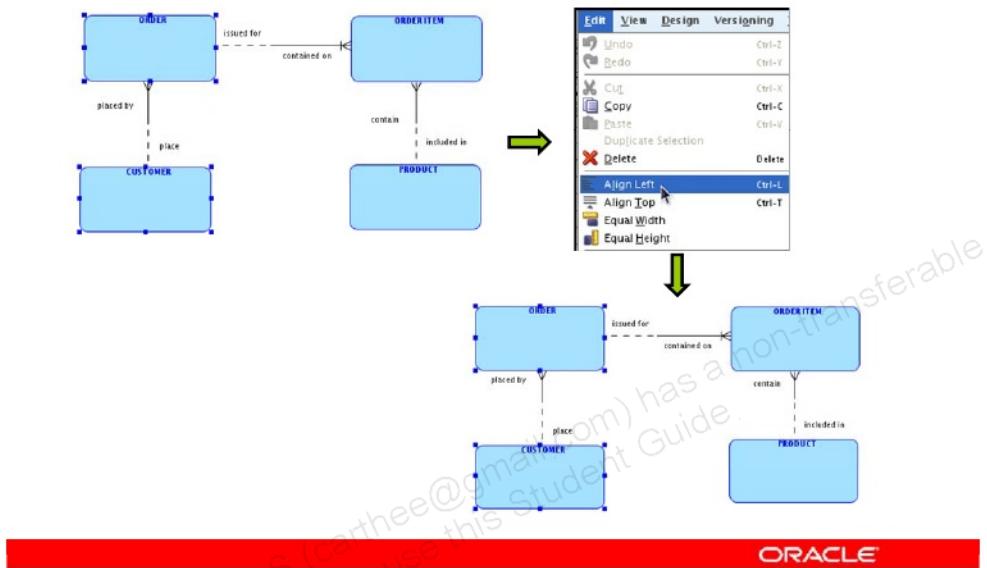
4. Move the objects to the location you want.
5. Restraighten relationship lines and source and target names.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Editing a Diagram Layout: Aligning Objects



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

To align objects at the left or top, perform the following steps:

- Press **Ctrl** and select the objects that you want to align.
- Select **Edit > Align Left** or **Align Top**.

The objects will now align.

Lesson Agenda

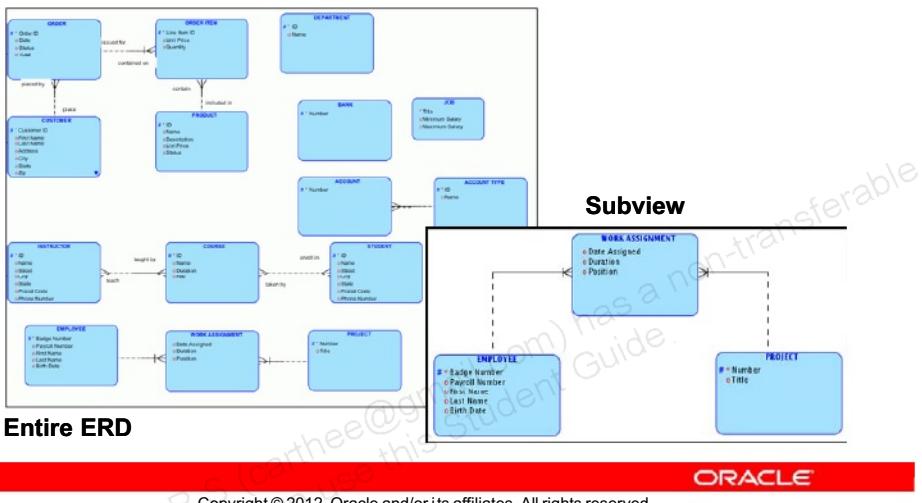
- Examine the General options for logical data modeling
- Build an ERD in Oracle SQL Developer Data Modeler
- Edit the layout of an ERD
- **Create a subview**
- Create a display

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

What Is a Subview?

A subview is a section of the logical model.

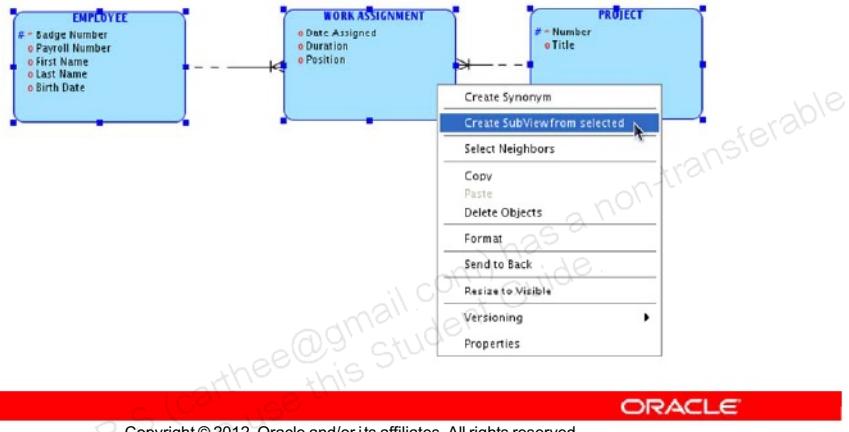


The logical model diagram contains graphical representations of entities, views, and links (relations and inheritances) between them. When you are working with a complex logical model, you may want to create subviews, each describing only a section of that model.

You can define several logical subviews for a single logical model, and you can assign entities and views to more than one subview. Links (relations) between two entities are displayed on the complete logical model and on logical subviews to which both referenced entities have been assigned. There is no difference between performing changes in one of the subviews or in the complete logical model. Any changes made are immediately reflected in the complete logical model and any relevant subviews. However, you can remove entities and views from a subview without deleting them from the complete logical model.

Creating a Subview

Select the objects that you want to include in the subview, right-click one of the objects, and select “Create SubView from selected.”



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To create a subview containing specific entities, perform the actions described in the slide.

Lesson Agenda

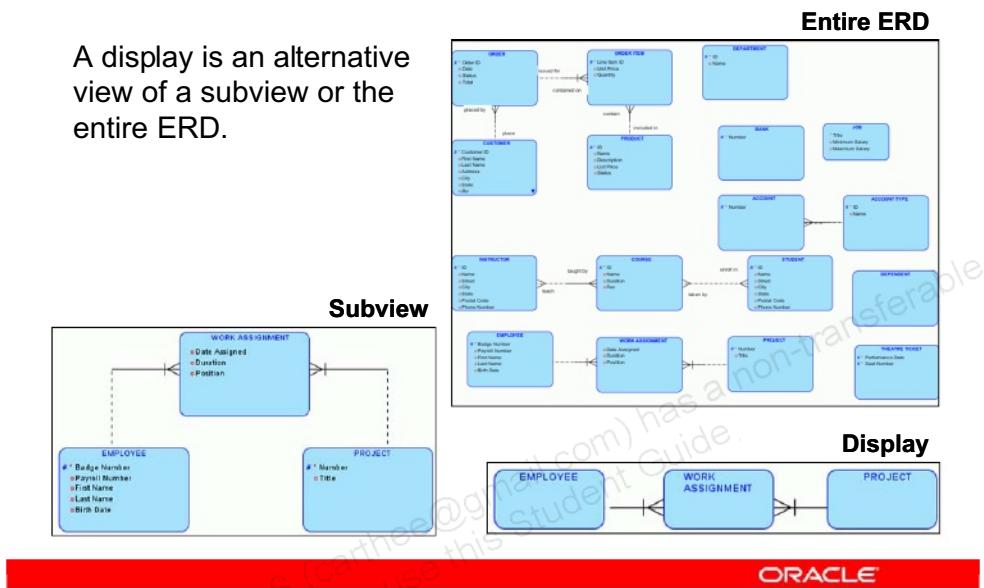
- Examine the General options for logical data modeling
- Build an ERD in Oracle SQL Developer Data Modeler
- Edit the layout of an ERD
- Create a subview
- Create a display

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

What Is a Display?

A display is an alternative view of a subview or the entire ERD.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

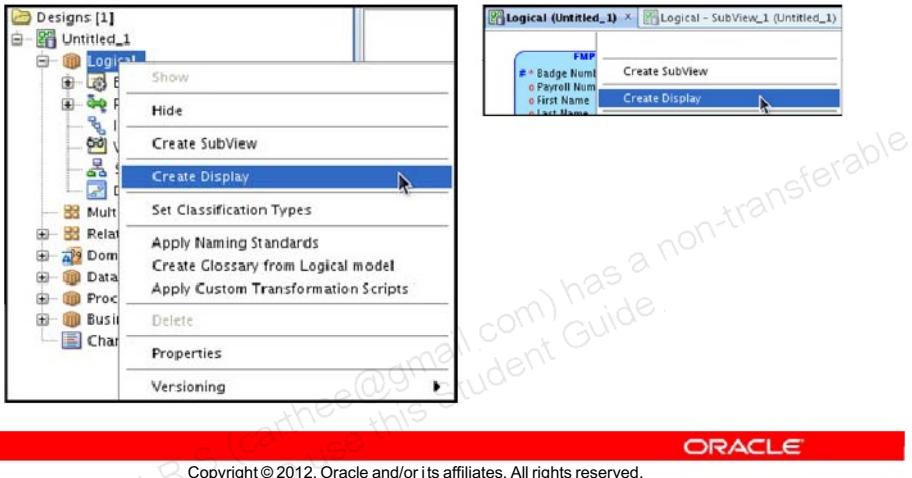
A display is an alternative view of the same objects in a subview or the entire ERD. For example, you might have one subview with three displays:

- Display 1 shows all the details (that is, table name, column name, and constraints).
- Display 2 consists of all the tables, but only displays table name.
- Display 3 displays only the keys.

The most common use of a display is to show the whole diagram, but with table names only. You can then move and resize the objects in the display, and then you can see how all the tables (or entities) relate to each other without all the other detail.

Creating a Display

Two places to access the context menu:



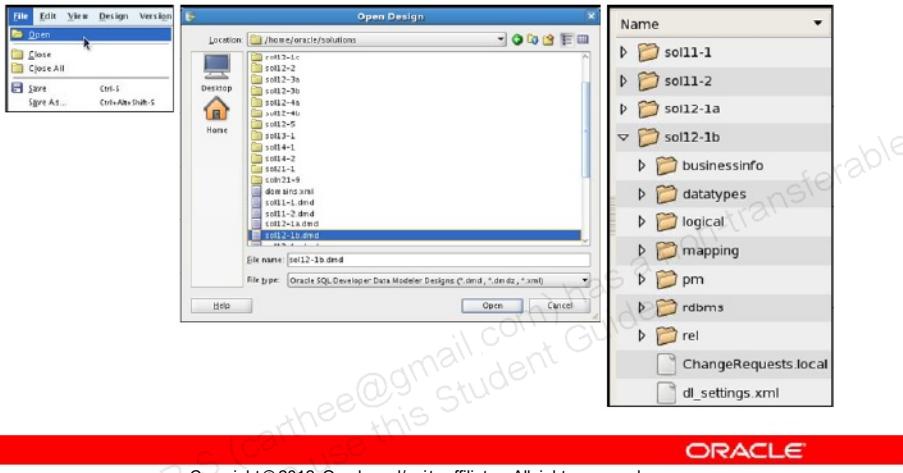
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

To create a display from a context menu, right-click the **Logical** or the **SubView** entry or right-click the tab, and then select **Create Display**.

Opening and Saving a Model

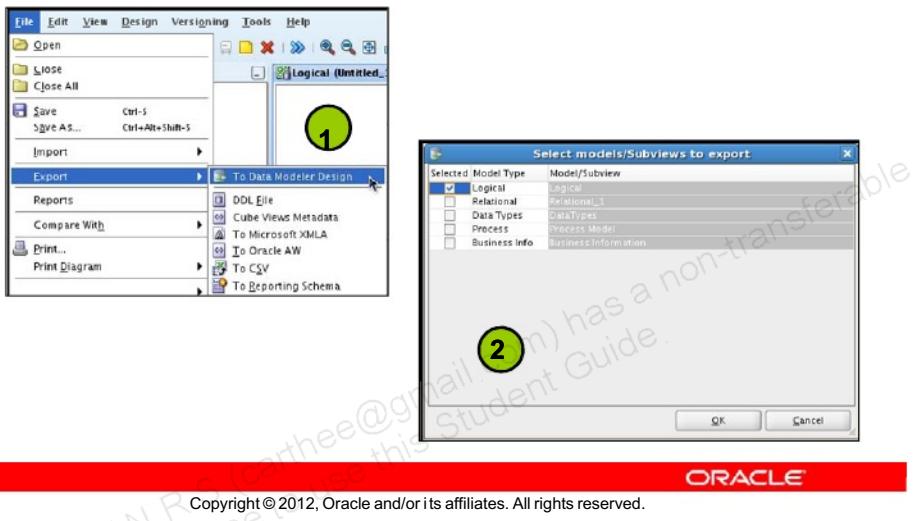
Opening or saving a model creates a .dmd file and a directory with data modeling objects.



To open or save the model, select **Open**, **Save**, or **Save As** from the **File** menu.
The model is saved as an DMD file. For each DMD file, there is a directory with the same name that contains directories for each model type.

Exporting a Model

Select the models to export.



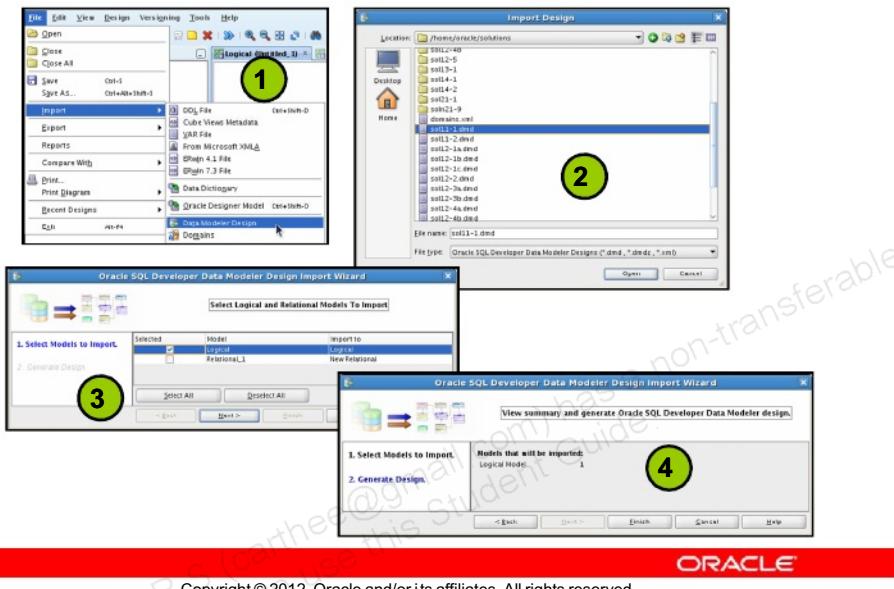
If you want to save certain models of the entire design, you can use the export function.

To export the model, perform the following steps:

1. Select **File > Export > To Data Modeler Design**. The **Select models/Subviews to export** dialog box appears.
2. Select the models that you want to export and click **OK**. In the example in the slide, the **Logical** model was exported. You can view the directory structure of the export and see that only the logical model was saved.

Note that you can open the exported version of the model and the **Logical** model will be available.

Importing a Model



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

If you want to import certain models of the entire design, you can use the Import function.

To import the model, perform the following steps:

1. Select **File > Import > Data Modeler Design**. The **Oracle SQL Developer Data Modeler Design Import Wizard** opens.
2. Select the **Design** you want to import, and click **Open**.
3. Select the models that you want to import, and click **Next**.
4. The wizard displays a list of the models to be imported. Click **Finish**. In the example in the slide, the **Logical** model was imported.

Note that you can set **General** options for imports. You can specify the default directory where the import files are located and also specify that a log be shown after the import takes place.

Quiz

A subview does which of the following? (Choose all that apply.)

- a. Examines a subset of the objects on the main diagram
- b. Enables you to move objects
- c. Examines the attributes for validity
- d. Permits the use of a DFD

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, b

Quiz

A display does which of the following? (Choose all that apply.)

- a. Has the same characteristics as a subview
- b. Enables you to show some attributes and not others in an entity
- c. Can show various levels of detail
- d. Allows only Bachman notation

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

Summary

In this lesson, you should have learned how to:

- Examine the General options for logical data modeling
- Build an Erd in Oracle SQL Developer Data Modeler
- Edit the layout of an Erd
- Create a subview
- Create a display

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 9-1 Overview: Build an ERD in Oracle SQL Developer Data Modeler

This practice covers the following topics:

- Building ERDs in Oracle SQL Developer Data Modeler for the following scenarios:
 - Class Practice from Lesson 8: Instructor/Student scenario
 - Practice 8-1: Hotel Scenario
 - Practice 8-2: DVD Membership Scenario
- Creating a subview and a display for each scenario

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2013, Oracle and/or its affiliates.

KARTHIKEYAN R S (carthee@gmail.com) has a non-transferable
license to use this Student Guide.

10

Validating Your Entity Relationship Diagram

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Apply diagram layout and attribute rules
- Distinguish entities from attributes
- Evaluate attribute optionality
- Supplement the entity relationship diagram with useful information
- Generate and view reports



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you validate your ERD to make sure that you apply some of the design rules necessary to make the model complete.

Lesson Agenda

- Apply diagram layout and attribute rules
- Distinguish entities from attributes
- Evaluate attribute optionality
- Supplement the ERD with useful information
- Generate and View Reports

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ERD Checklist

- Neatness
 - Align entity boxes.
 - Draw relationship lines straight.
 - Use white space to avoid congestion.
- Text
 - Avoid having too many parallel lines.
 - Make all text unambiguous.
 - Avoid abbreviations and jargon.
 - Align text horizontally.
 - Put relationship names at the ends of the line and the opposite sides of the line.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

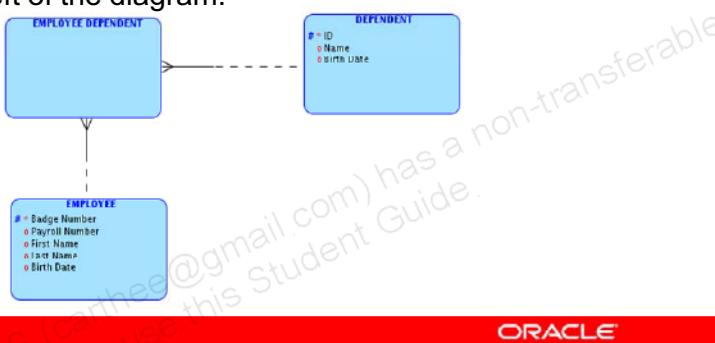
Keep the following rules in mind when verifying your ERD:

- Align your entities so they are lined up appropriately.
- Draw relationship lines straight (either horizontal or vertical). Relationship lines may have an elbow in the line; however, there should not be any diagonal lines.
- Space your entities out and use white space to avoid congestion and unreadability of relationship names.
- Avoid having too many parallel lines because they are difficult to follow.
- Ensure that all text is unambiguous.
- Avoid abbreviations and jargon.
- Align text horizontally.
- Put relationship names at the end of the line that it is related to. Put the opposite relationship name on the opposite side of the line.

ERD Checklist

Layout rules

- Draw crow's feet pointing up or to the left.
- The diagram should read left to right.
- Position higher-volume or more volatile entities toward the top and left of the diagram.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Keep the following rules in mind when verifying your ERD:

- Draw the crow's feet pointing up or to the left. Note that until an M:M relationship is resolved, at least one end of the relationship will point down or to the right.
- The diagram should read from left to right.
- Position higher volume, more volatile entities toward the top and left of the diagram.
- Position lower volume, less volatile entities toward the bottom and right of the diagram.

Attribute Rules

Attributes have the following rules:

- Each attribute should be assigned only to a single entity.
- Each attribute should be created at the lowest meaningful level.
- Aggregate attributes and embedded code fields are broken down into simple attributes.
- Each attribute has a single value for each entity instance.
- An attribute is not derived or calculated from existing values.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

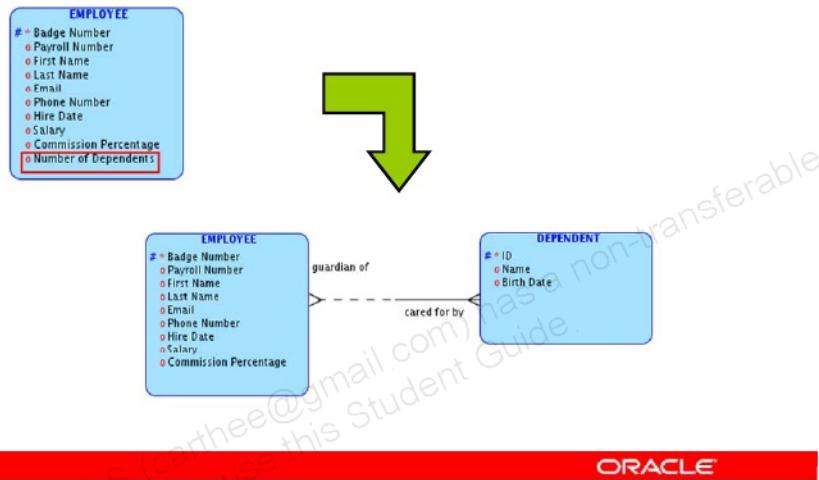
Keep the following rules in mind when identifying attributes:

- Each attribute should be assigned to only one entity. If you have an attribute such as address that has common data type characteristics, you can define an object type and then reference the object type as an attribute. This topic will be discussed later in the course.
- Each attribute should be created at its lowest meaningful level. For example, the name of a person can be broken down into first name and last name.
- Break down aggregate attributes and embedded code fields into simple attributes.
- Each attribute has a single value for each entity instance. A multivalued attribute or repeating group is not a valid attribute. This is examined during normalization, which is discussed in a later lesson.
- An attribute is not derived or calculated from the existing values of other attributes. Derived attributes are redundant, and redundancy can lead to inconsistent data values.

The derived data must be revised whenever the attributes upon which it is based are revised. You may address the option of storing derived data during database design (discussed in a later lesson).

Distinguishing Attributes and Entities

If an attribute has attributes of its own, it is an entity.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

In the example in the slide, "Number of Dependents" is an attribute in the EMPLOYEE entity, but if it is necessary to keep each dependent's name and age, DEPENDENT becomes an entity. The "Number of Dependents" attribute can now be derived.

Distinguishing Attributes and Entities

All entities are nouns, but not all nouns are entities.

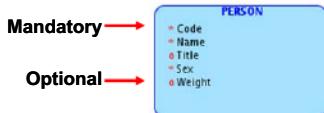
Entity	Attribute
Is anything about which information must be held	Qualifies an entity
Possesses one or more attributes	Does not possess attribute(s) of its own
If an entity has no attributes, it may be only an attribute.	If an attribute has an attribute, it is an entity or has no significance.
May have multiple occurrences associated with another entity via a relationship	Has a single value for each entity occurrence (no repeating groups)

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Attribute Optionality

Identify each attribute's optionality by using an attribute tag.



AttributeName	Code	Name	Title	Sex	Weight
Tags	*	*	o	*	o
SampleData	110	Jones	President	F	
	301	Smith	Treasurer	M	210
	134	Gonzales	-	F	110
	340	Johnson	Secretary	M	-
	589	Brown	-	M	195

ORACLE®

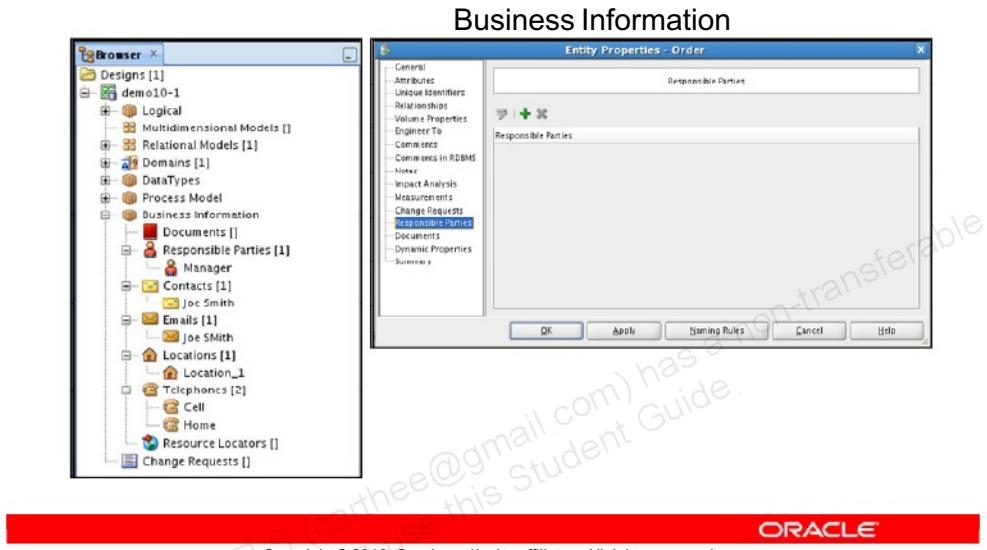
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Each attribute needs to be defined as a mandatory or optional attribute. If the attribute is mandatory, each entity occurrence must have a value. If the attribute is optional, each entity occurrence may or may not have a value.

In the example in the slide, the code, name and sex attributes in the `PERSON` entity are mandatory and must have a value for each occurrence. The title and weight are optional, meaning that they may or may not have a value for every occurrence.

You can use sample attribute instance data to validate attribute optionality. The Entity Instance Chart in the slide is useful for locating sample attribute data.

Adding Additional Information to the ERD



You can add supplemental information to any object in your ERD. The information that is useful to know is who the responsible party for a particular object is.

In the screenshot in the slide, Manager is responsible for the ORDER entity. You can drill down to see the list of contacts, locations, telephones, email, and so on. You can also see this information in the left navigator.

Lesson Agenda

- Apply diagram layout and attribute rules
- Distinguish entities from attributes
- Evaluate attribute optionality
- Supplement the ERD with useful information
- Generate and View Reports

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Data Modeler Reports

You can view reports on Data Modeler objects as follows:

- Generate reports as RTF, HTML, or PDF files on your local drives, and then open the files.
- Use SQL Developer to view reports based on information in the Data Modeler reporting repository.

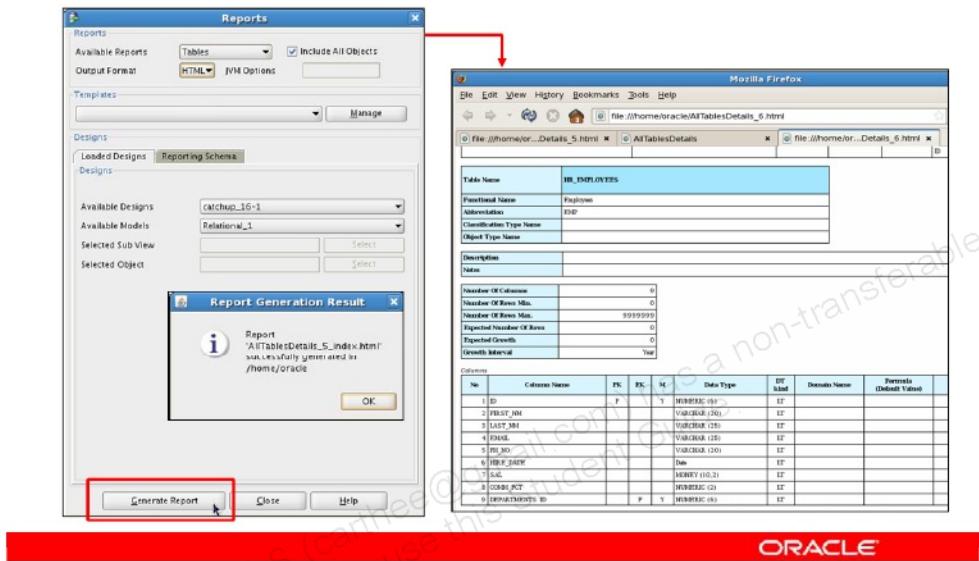


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Note

For additional information about creating reports, see the *Oracle SQL Developer Data Modeler User's Guide Release 3.0*.

Generating Data Modeler Reports as RTF, HTML, or PDF Formats



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

You can save individual reports as RTF, PDF, or HTML files, and then view each report. The reports are stored in the location specified or defaulted for Default Reports Directory under Data Modeler preferences. Data Modeler ensures unique names for each file. For example, if you generate a report on all tables and if `AllTablesDetails_5.html` already exists, `AllTablesDetails_6.html` is created.

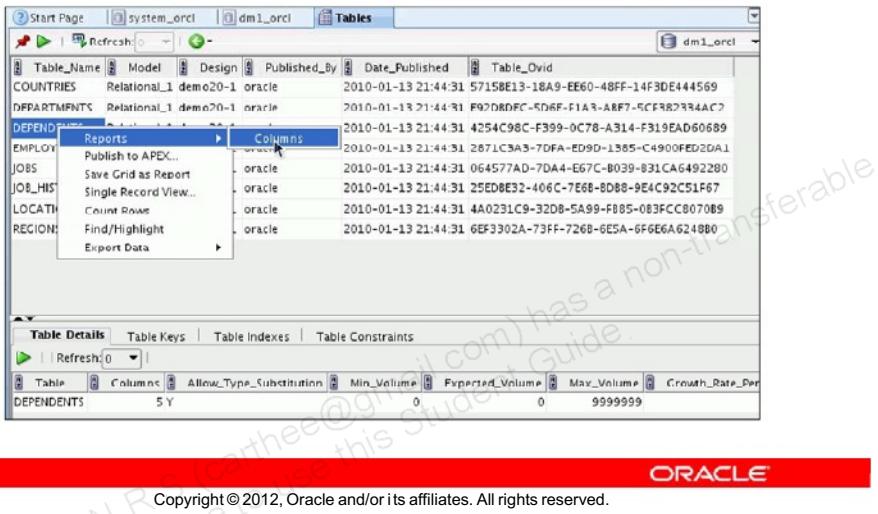
You can generate and view a report as follows:

1. Select **File > Reports**. The **Reports** dialog box is displayed.
2. For **Available Reports**, select one of the types of objects for which to report information: **Tables**, **Entities**, **Domains**, or **Glossaries**.
3. Select the **Output Format** as **RTF**, **HTML**, or **PDF**.
4. (Optional)Select **Include All Objects** to have the report contain information about all objects of the selected type. If you do not select **Include All Objects**, you will be able to click **Select** beside **Selected Object** on the **Loaded Des**igns tab to select an object by name.
5. Click one of the following tabs (if the desired tab is not already selected):
 - **LoadedDesigns**, to generate a report based on one or more currently loaded Data Modeler designs
 - **Reporting Schema**, to generate a report based on designs in the reporting repository in the reporting schema

6. For **Available Designs**, select the desired Data Modeler design.
7. For **Available Models**, select the desired model. (The list of models reflects the type of objects for the report.)

Producing Data Modeling Metadata Reports

Design content and rule reports in Oracle SQL Developer.



Predefined data modeler reports are included with the latest release of Oracle SQL Developer. These reports allow you to report on the objects in your model, and they also report design rule violations. To use the data modeler reports, you need to export your model to a reporting schema in Oracle SQL Developer Data Modeler and then run the reports within Oracle SQL Developer.

Steps to Produce Data Modeler Reports

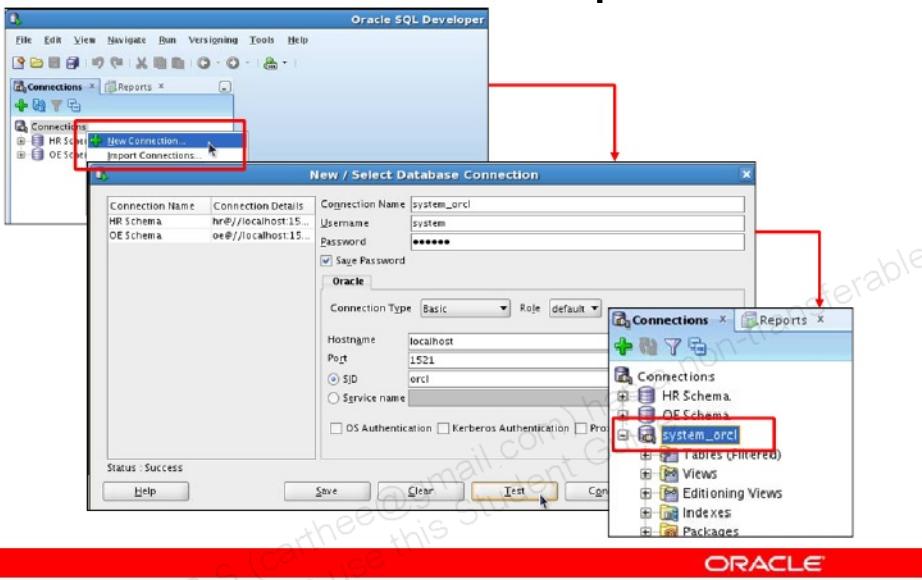
1. In Oracle SQL Developer:
 - a. Create a SYSTEM database connection.
 - b. Create a new reporting user.
 - c. Create a database connection for the new reporting user.
2. In Oracle SQL Developer Data Modeler:
 - a. Open the desired model.
 - b. Export the model to the reporting schema.
3. In Oracle SQL Developer:
 - a. Run the Data Modeler reports that are available in Oracle SQL Developer.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You must perform the tasks listed in the slide before you can do an export. The next few slides guide you through this process.

Creating a SYSTEM Database Connection in Oracle SQL Developer

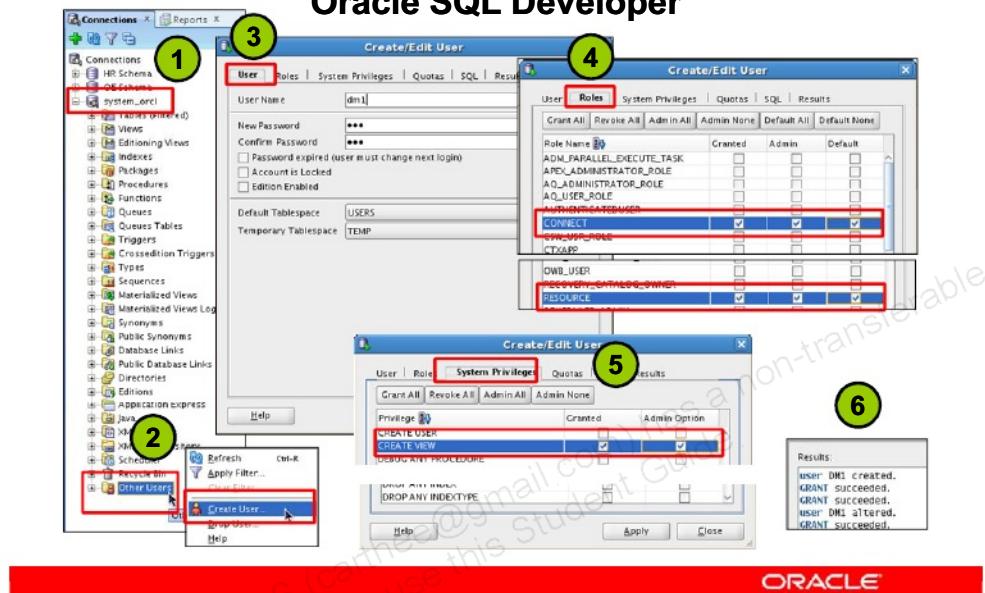


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This task is performed in Oracle SQL Developer. Create a connection to a user that has the privilege to create additional users. To create a SYSTEM connection, perform the following steps:

1. In Oracle SQL Developer, right-click the **Connections** node, and then select **New Connection** from the pop-up menu. The **New / Select Database Connection** dialog box is displayed.
2. Enter the following information, and then click **Connect**.
 - **Connection Name:** system_<sid>
 - **Username:** system
 - **Password:** <your system password>
 - **Hostname:** <your hostname>
 - **SID:** <your SID>

Creating a New User for Reporting in Oracle SQL Developer



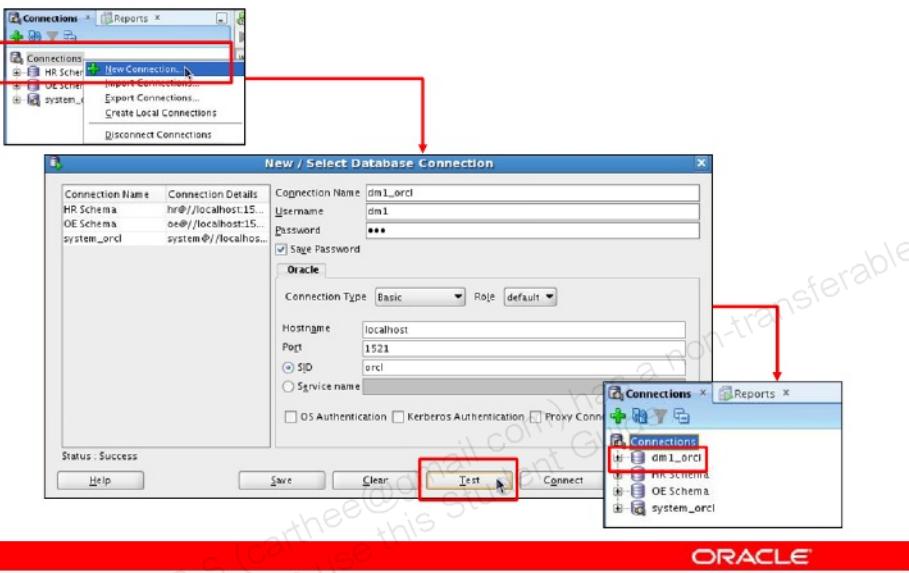
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Next create a new user that you will use for reporting purposes. You could use an existing user. However, there will be additional database objects created when you export to the reporting schema and this may not be desirable in an existing user schema.

Perform the following steps:

1. Expand your newly created system connection.
2. Right-click **Other Users**, and then select **Create User** from the pop-up menu. The **Create/Edit User** dialog box is displayed.
3. On the **User** tab, enter a **User Name** and **Password** (in this case `dm1`), specify a **Default Tablespace** such as `USERS`, and specify a **Temporary Tablespace** such as `TEMP`.
4. Click the **Roles** tab. Select the **CONNECT** and **RESOURCE** roles.
5. Click the **System Privileges** tab. Select the **CREATE VIEW** privilege, and then click **Apply**.
6. The **Results** tab is displayed. Click **Close**.

Creating a Connection for the New Reporting User in Oracle SQL Developer



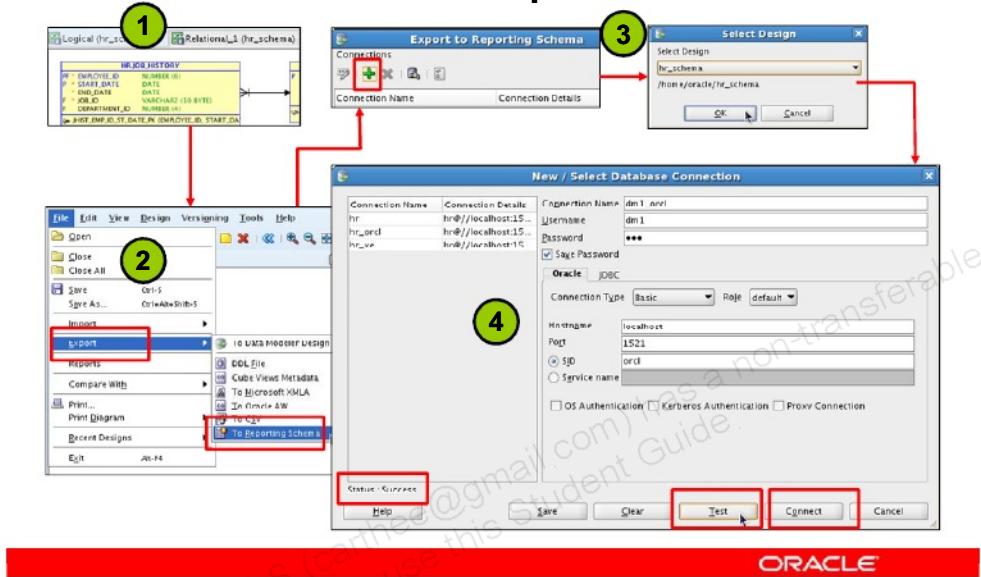
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Next you create a database connection for the new reporting user that you just created. This is the connection that you will use to run the reports after you export your model from Oracle SQL Developer Data Modeler.

Perform the following steps:

1. In **Oracle SQL Developer**, right-click **Connections**, and then select **New Connection**. The **New / Select Database Connection** dialog box is displayed.
2. Enter the following information, and then click **Connect**:
 - **Connection Name:** <dmuser>_<sid>
 - **Username:** <dmuser>
 - **Password:** <your dmuser password>
 - **Hostname:** <your hostname>
 - **SID:** <your SID>

Exporting Your Model to the Reporting Schema in Oracle SQL Developer Data Modeler



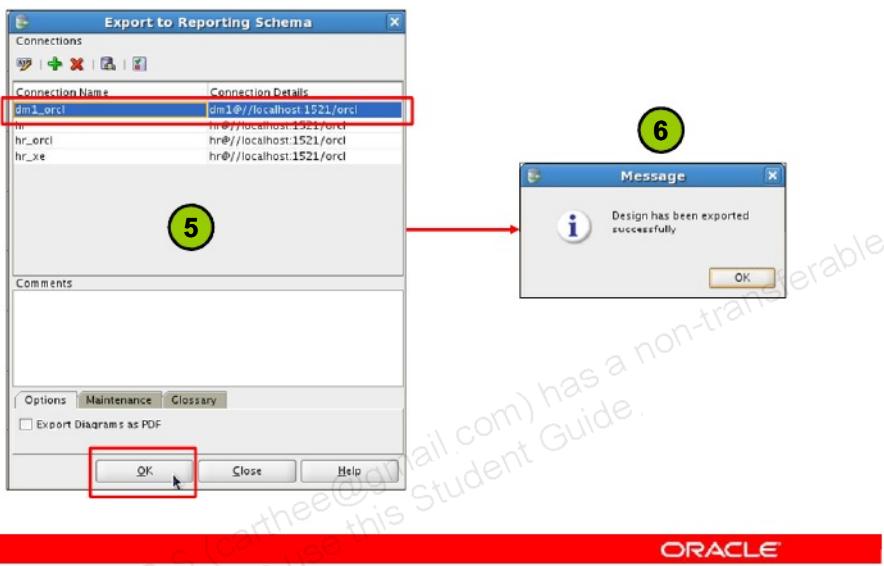
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Now you are ready to export your model to the reporting schema in Oracle SQL Developer Data Modeler.

Perform the following steps:

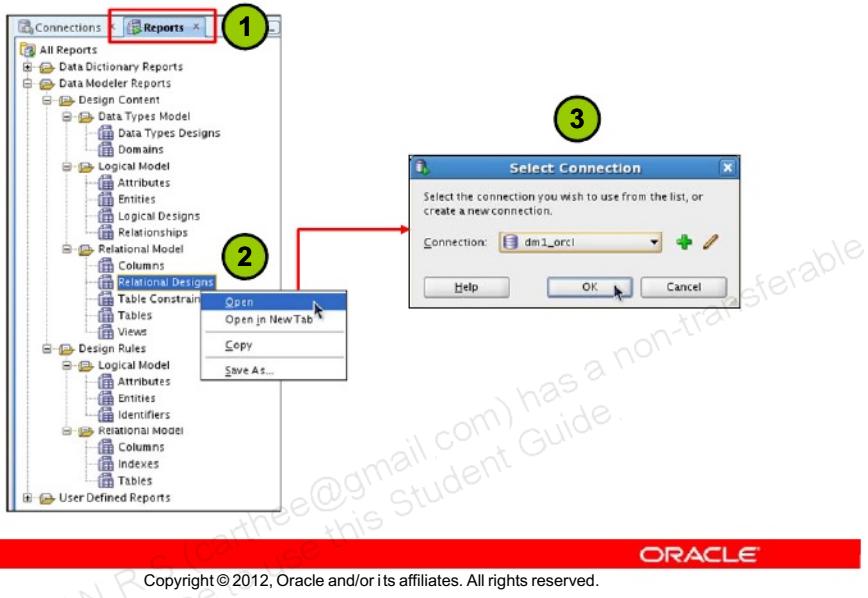
1. In **Oracle SQL Developer Data Modeler**, open the model that you want to export.
2. Select **File > Export > To Reporting Schema**.
3. The **Export to Reporting Schema** dialog box is displayed. Click the **Add '+' icon**. The **Select Design** dialog box is displayed. Select the design that you want to export, and then click **OK**.
4. The **New / Select Database Connection** window is displayed. Enter the following information, and then click **Connect**.
 - **Connection Name:** <dmuser>_<sid>
 - **Username:** <dmuser>
 - **Password:** <your dmuser password>
 - **Hostname:** <your hostname>
 - **SID:** <your SID>

Exporting Your Model to the Reporting Schema in Oracle SQL Developer Data Modeler



5. The **Export to Reporting Schema** window is re-displayed . Click the new connection that you created, dm1_orcl, and then click **OK**.
6. A **Message** dialog box displays the export status information.

Running Data Modeler Reports in Oracle SQL Developer

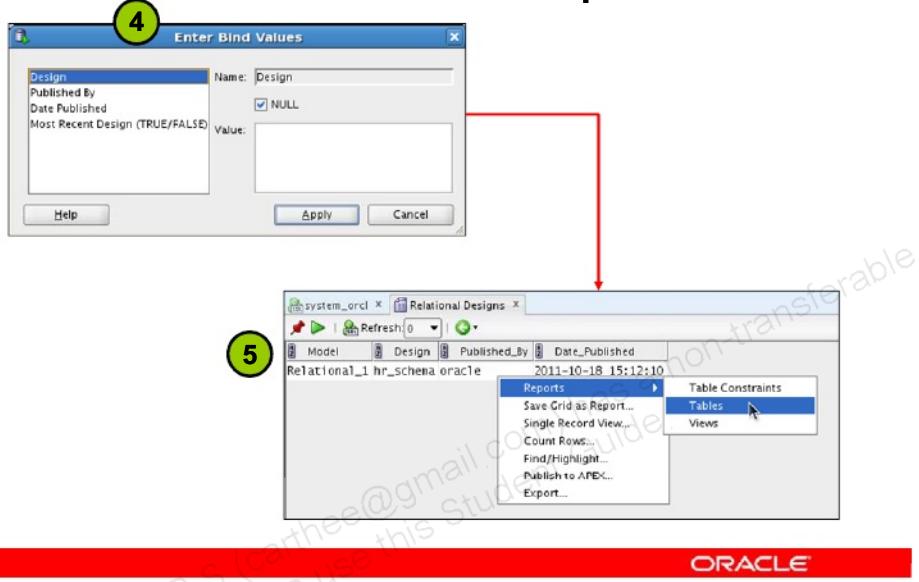


You are now ready to run the data modeler reports in Oracle SQL Developer. Under the **Design Content** node, you can run a series of reports that display information about the data types model, logical model, or relational model. You can also view Design Rules reports that show you objects that have violated a series of rules contained in the logical or relational model (shown in the screenshot in the slide).

Perform the following steps:

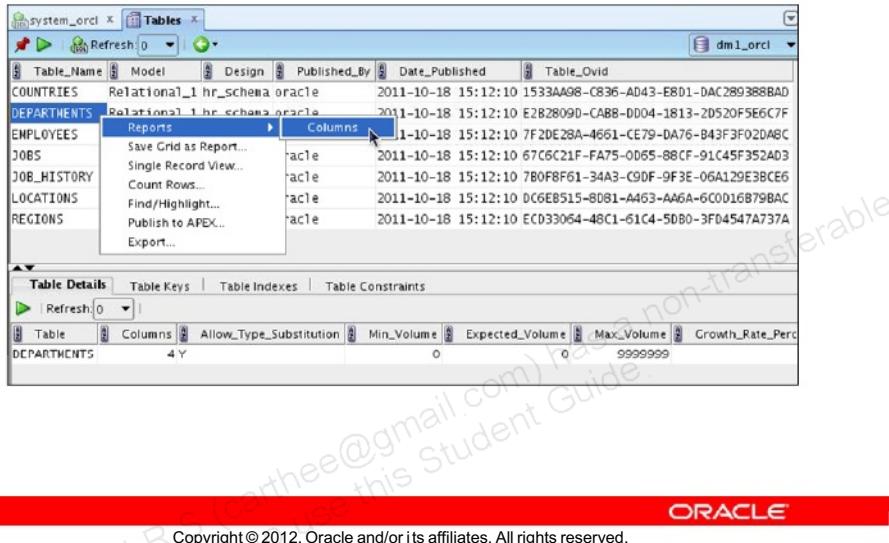
1. In **Oracle SQL Developer**, click the **Reports** tab.
2. Expand **Data Modeler Reports > Design Content**, and then select **Relational Designs**, or right-click **Relational Designs**, and then select **Open**.
3. Select the connection that you created for the reporting user `dm1_orcl`, and then click **OK**. The **Enter Bind Values** dialog box is displayed.

Running Data Modeler Reports in Oracle SQL Developer



4. In the **Enter Bind Values** dialog box, you can change the results by specifying bind value criteria. For example, if you have exported your model more than once, you can see all the designs that you exported by changing the bind value for **Most Recent Design** to **FALSE**. The default is **TRUE**. Click **Apply**.
5. The list of designs is displayed. You can drill down to display the detail in the design. Right-click the model name, and then select **Reports > Tables**.

Running Data Modeler Reports in Oracle SQL Developer



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

The list of tables is displayed. From the list of tables, you can drill down to see a list of columns. Notice that when you click a table, detailed information about that table (such as how many columns it has) is displayed in the bottom half of the window.

Running Data Modeler Reports in Oracle SQL Developer

The screenshot shows the Oracle SQL Developer interface with a report titled 'Columns'. The report displays the columns of the 'DEPARTMENTS' table from the 'Relational_1_hr_schema' model. The columns listed are DEPARTMENT_ID, DEPARTMENT_NAME, MANAGER_ID, and LOCATION_ID. Each row includes the column name, sequence, table name, model, design, published by, date published, and column Ovid.

Column_Name	Sequence	Table_Name	Model	Design	Published_By	Date_Published	Column_Ovid
DEPARTMENT_ID	1	DEPARTMENTS	Relational_1_hr_schema	oracle		2011-10-18 15:12:10	800ED57E-EB75-2
DEPARTMENT_NAME	2	DEPARTMENTS	Relational_1_hr_schema	oracle		2011-10-18 15:12:10	E6FB518C-5F0F-5
MANAGER_ID	3	DEPARTMENTS	Relational_1_hr_schema	oracle		2011-10-18 15:12:10	C2974FA3-5D6A-B8
LOCATION_ID	4	DEPARTMENTS	Relational_1_hr_schema	oracle		2011-10-18 15:12:10	7299ECC2-6BC2-0

The list of columns is displayed. If you click a column, column details are displayed in the bottom half of the window, such as the column's data type and length (size).

Quiz

Which of the following steps should you take to ensure the completeness of your ERD? (Select all that apply.)

- a. Make all text unambiguous.
- b. Make sure that you have defined all derived attributes.
- c. Align objects so that the diagram reads left to right.
- d. Add abbreviations when possible.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

Quiz

Which method can you use to see design issues in the relational model? (Choose all that apply.)

- a. Running the design rules in Oracle SQL Developer Data Modeler
- b. Exporting the DDL file
- c. Running the design rules report in Oracle SQL Developer
- d. Reviewing the relational model diagram

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: a, b, c, d

Summary

In this lesson, you should have learned how to:

- Apply diagram layout and attribute rules
- Distinguish entities from attributes
- Evaluate attribute optionality
- Supplement the entity relationship diagram with useful information
- Generate and view Reports

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you should have learned how to validate an ERD based on a set of rules. In addition, you should have learned how to define naming standards and a glossary in Oracle SQL Developer Data Modeler and supplement your model with additional information.

Practice 10-1 Overview: Develop and Validate Your ERD

This practice covers the following topics:

- Developing an ERD based on the Oracle User's Group case study in Oracle SQL Developer Data Modeler
- Validating the ERD by using the rules discussed in this lesson
- Working with the SQL Developer Data Modeler reporting repository

Unauthorized reproduction or distribution prohibited. Copyright© 2013, Oracle and/or its affiliates.

KARTHIKEYAN R S (carthee@gmail.com) has a non-transferable
license to use this Student Guide.

Utilizing Advanced Data Modeling Techniques

IV

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Overview

This unit includes the following lessons:

11. Normalizing Your Data Model
12. Validating Relationships
13. Adding and Using Data Types
14. Putting It All Together



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

11

Normalizing Your Data Model

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to normalize your entity relationship diagram to third normal form.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Lesson Agenda

- What is Normalization?
 - First Normal Form
 - Second Normal Form
 - Third Normal Form
- Example of Normalization

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

What Is Normalization?

Normalization is a relational database concept, but its principles apply to data modeling.

Rules:

Rule	Description
First normal form (1NF)	All attributes must be single-valued.
Second normal form (2NF)	An attribute must be dependent on its entity's unique identifier.
Third normal form (3NF)	No non-UID attribute can be dependent on another non-UID attribute.

Goal: Normalize to third normal form before transforming the model to your relational design



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

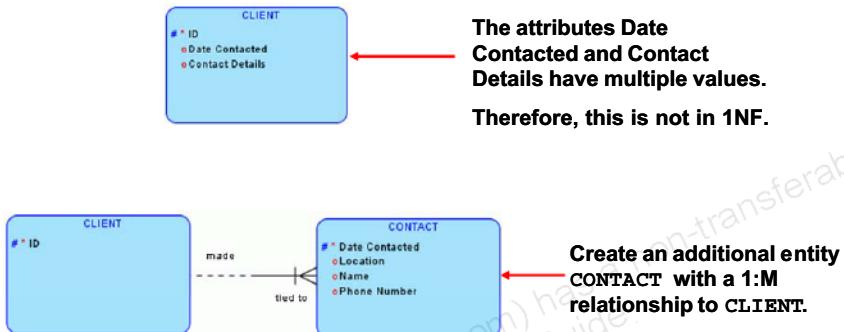
Normalization is a relational database concept. However, if you have created a validated ERD, the tables created during the design will conform to the rules of normalization. Each formal normalization rule from the relational database design has a corresponding data model interpretation. The interpretations that can be used to validate the placement of attributes in an ERD are shown in the table in the slide.

Benefits of Normalization

- Normalization ensures that each attribute appropriately belongs to the entity to which it has been assigned and not another entity.
- Normalization eliminates redundant storage of information; this simplifies application logic, because developers do not need to think about multiple copies of the same piece of information.
- Normalization ensures that you have one attribute in one place, with one name, with one value, at any one time.

First Normal Form (1NF)

All attributes must be single-valued.

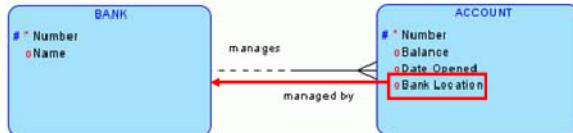


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Second Normal Form (2NF)

An attribute must be dependent on its entity's entire unique identifier.



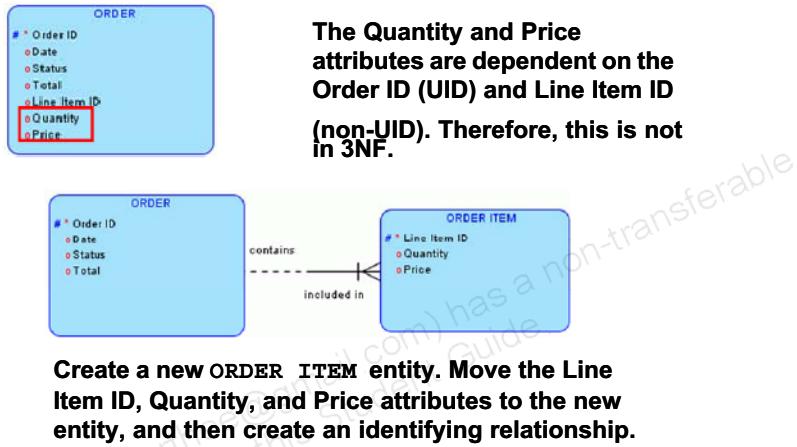
The Bank Location attribute is dependent on BANK rather than on ACCOUNT. Therefore, this is not in 2NF. Move the attribute to the BANK entity.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Third Normal Form (3NF)

Each attribute depends only on the UID of its entity.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Third normal form validates that each attribute depends only on the UID of its entity (and on nothing else). You need to move any non-UID attribute that is dependent on another non-UID attribute into a new entity.

In the example in the slide, the Quantity and Price attributes are dependent on both the Order ID (UID) and the Line Item ID (non-UID) attributes. Since these attributes are dependent in part on a non-UID attribute, the attributes, along with the non-UID attribute (Line Item ID) should be moved to a new entity and an identifying relationship should be created.

Quiz

If an entity contains four attributes, two attributes of which do not depend on the unique identifier, which normal form does this violate?

- a. First normal form
- b. Second normal form
- c. Third normal form
- d. Fourth normal form

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Lesson Agenda

- What is Normalization?
 - First Normal Form
 - Second Normal Form
 - Third Normal Form
- Example of Normalization

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Normalization Example: Unnormalized Data

Ordered by:		Ship to:																															
CustomerID	:	<input type="text"/>	<input type="text"/> Order ID																														
Customer Name	:	<input type="text"/>	<input type="text"/> Order Date																														
Address Line 1	:	<input type="text"/>	<input type="text"/> Ship/via																														
Address Line 2	:	<input type="text"/>	<input type="text"/> Name																														
Address Line 3	:	<input type="text"/>	<input type="text"/> Address Line 1																														
City, State ZIP	:	<input type="text"/> , <input type="text"/> , <input type="text"/>	<input type="text"/> Address Line 2																														
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> Address Line 3																														
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/> City, State ZIP																														
<table border="1"><thead><tr><th>Item ID</th><th>Color</th><th>Size</th><th>Quantity</th><th>Description</th><th>Price</th></tr></thead><tbody><tr><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td></tr><tr><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td></tr><tr><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td></tr><tr><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td><td><input type="text"/></td></tr></tbody></table>				Item ID	Color	Size	Quantity	Description	Price	<input type="text"/>																							
Item ID	Color	Size	Quantity	Description	Price																												
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>																												
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>																												
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>																												
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>																												
Order Total: <input type="text"/>																																	

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Data that has not been “normalized” is considered to be “unnormalized” data. This data is not to be confused with data that is denormalized. If an Entity Relationship Model was not created at the start of a database design project, you are likely to have unnormalized data rather than denormalized data. If you want to add redundancy, for faster performance or other reasons, you follow the rules defined during the process of denormalization after you forward-engineer the model to a database design (discussed in a later lesson).

Normalization Example: Transforming to First Normal Form

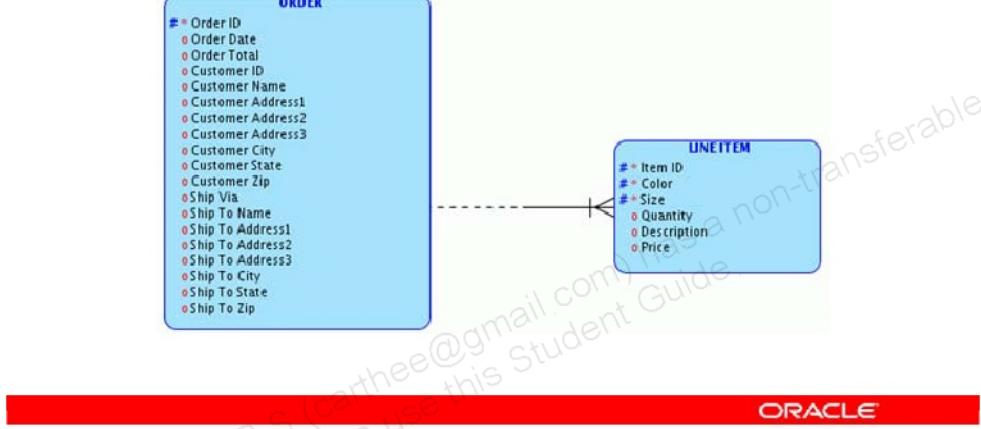


To transform the model to first normal form (1NF), the table must express a set of unordered, two-dimensional table structures. A table is considered to be in first normal form if it contains no repeating groups.

Perform the following steps:

1. Identify all the data attributes on the order form.
2. Group the data attributes into a single data entity.
3. Determine which attribute will serve as the primary key.

Normalization Example: Transforming to First Normal Form



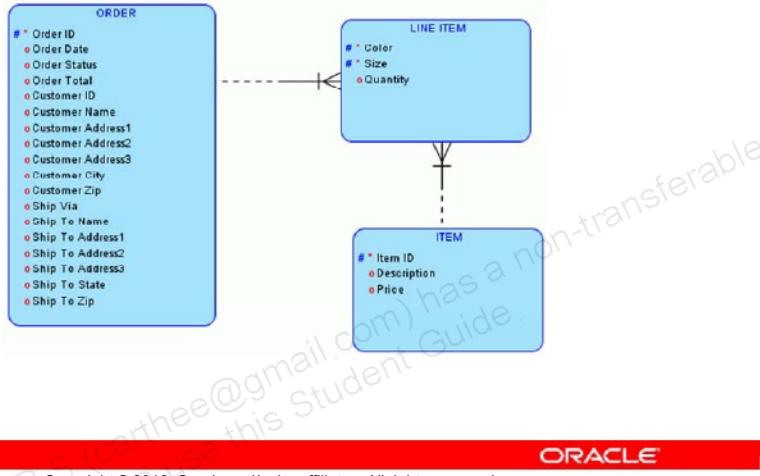
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can now remove the repeating groups by performing the following steps:

1. Create a new table with the primary key of the base table and the repeating columns.
2. To ensure that the primary key is unique, add another appropriate column.
3. Create a foreign key in the new table to link back to the original unnormalized table.

Normalization Example: Transforming to Second Normal Form



ORACLE®

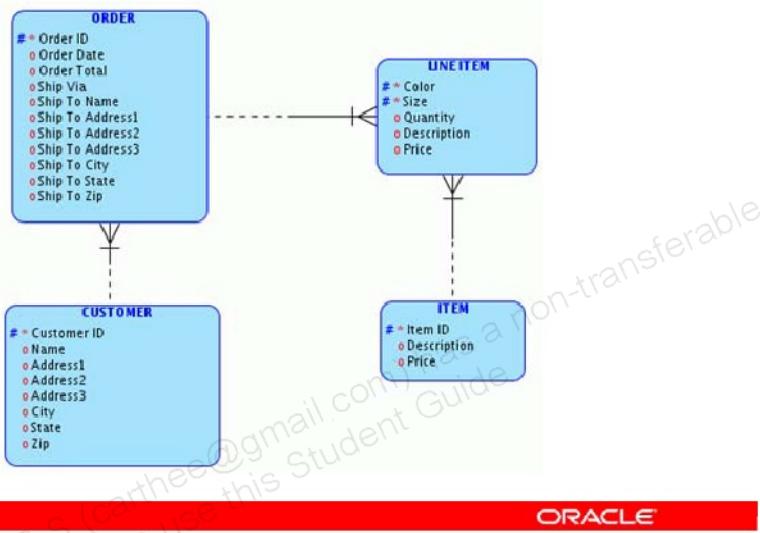
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To transform the model to second normal form (2NF), you must remove attributes that are dependent on only a piece of a data entity's multi-part key.

Perform the following steps:

1. Locate data entities that have multi-part keys.
2. Move attributes that relate to only a piece of the multi-part key to a new data entity.
3. The key for the new data entity consists of that part of the old multi-part key that uniquely identifies the attributes.

Normalization Example: Transforming to Third Normal Form



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To transform the model to third normal form (3NF), you must remove attributes that do not *directly* relate to the primary key. As a result, all attributes depend on non-primary key attributes within same data entity or in another data entity.

Perform the following steps:

1. Locate attributes that do not *directly* relate to the primary key.
2. Move those attributes to a new data entity.
3. The key for the new data entity is the attribute that uniquely defines the data entity.

Summary

In this lesson, you should have learned how to normalize your ERD to third normal form.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you should have learned what normalization is and what the normal forms are.

Practice 11-1 Overview: Normalize an ERD

This practice covers evaluating an ERD and normalizing it to third normal form.

Practice 11-2 Overview: Validate ERD for Normalization

This practice covers normalizing unnormalized data to third normal form.

Unauthorized reproduction or distribution prohibited. Copyright© 2013, Oracle and/or its affiliates.

KARTHIKEYAN R S (carthee@gmail.com) has a non-transferable
license to use this Student Guide.

12

Validating Relationships

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Resolve M:M relationships
- Model hierarchical data
- Examine recursive relationships
- Model exclusive relationships
- Model entity type hierarchies
- Model data over time



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you examine some of the more advanced data modeling techniques associated with relationships.

Lesson Agenda

- Resolve M:M relationships
- Model hierarchical data
- Examine recursive relationships
- Model exclusive relationships
- Model entity type hierarchies
- Model data over time

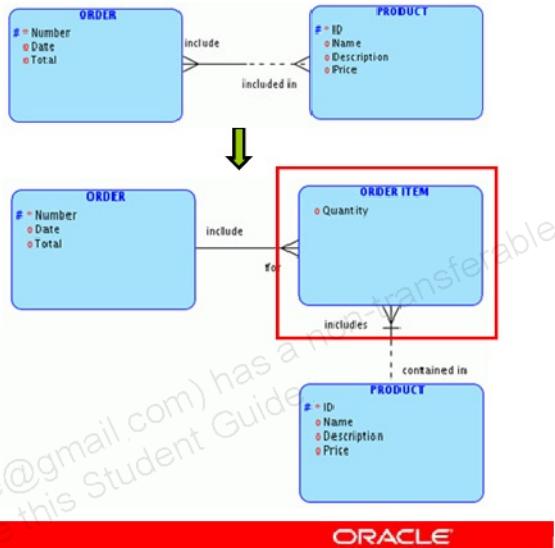
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Resolving M:M Relationships

Attributes describe entities only. If attributes describe a relationship, the relationship must be resolved.

Resolve a M:M relationship with a new intersection entity and two 1:M relationships.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Evaluate all M:M relationships to identify the attributes that may be included in an intersection entity. In the example in the slide, the M:M relationship between ORDER and PRODUCT needs to be resolved. An intersection entity (ORDER ITEM) was created that stores additional attributes. The unique identifier will be the relationships between ORDER and PRODUCT.

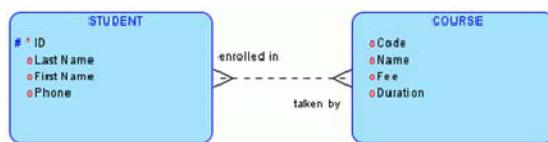
Intersection Entity Characteristics

- The relationships from the intersection entity are always mandatory.
- Intersection entities usually contain consumables like quantity used and dates. They tend to be high-volume and volatile entities.
- An intersection entity is identified by its two scinating relationships (identifying relationships).

Note that if you do not have any additional attributes in the intersection entity, you can leave it as a M:M relationship and Oracle SQL Developer Data Modeler will create the intersection table in the relational model. This is discussed in a later lesson.

Resolving M:M Relationships

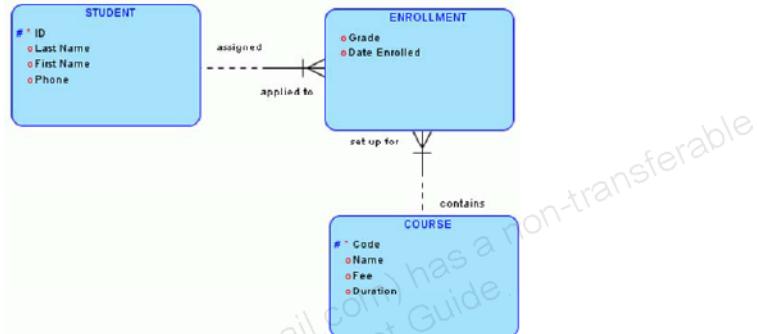
How would you resolve the following?



How would you resolve a M:M relationship between STUDENT and COURSE?

Resolving M:M Relationships

Create an intersection entity with identifying relationships to the participating entities.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To resolve the M:M relationship from the previous slide, perform the following steps:

1. Delete the M:M relationship.
2. Create a new entity called ENROLLMENT.
3. Create two identifying relationships: one from STUDENT and the other from COURSE.
4. Identify and create additional attributes in the ENROLLMENT intersection entity.
5. Evaluate whether the two identifying relationships constitute a unique identifier for the intersection entity, or whether other attributes are needed.

Note: Not all intersection entities will have additional attributes.

Quiz

Which of the following statements are true? (Choose all that apply.)

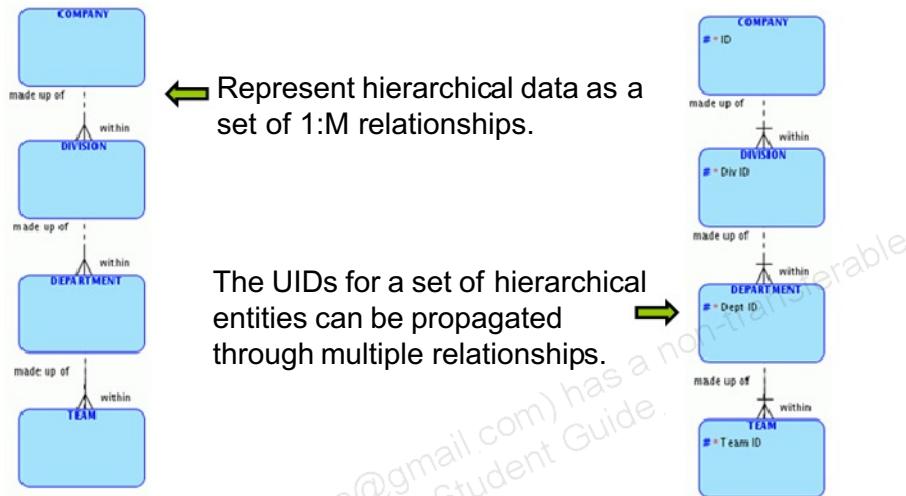
- a. Intersection entities always have additional attributes.
- b. Intersection entities only have identifying relationships as their **UID**.
- c. An intersection entity resolves a M:M relationship.
- d. Most M:M relationships should be resolved.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c, d

Modeling Hierarchical Data



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

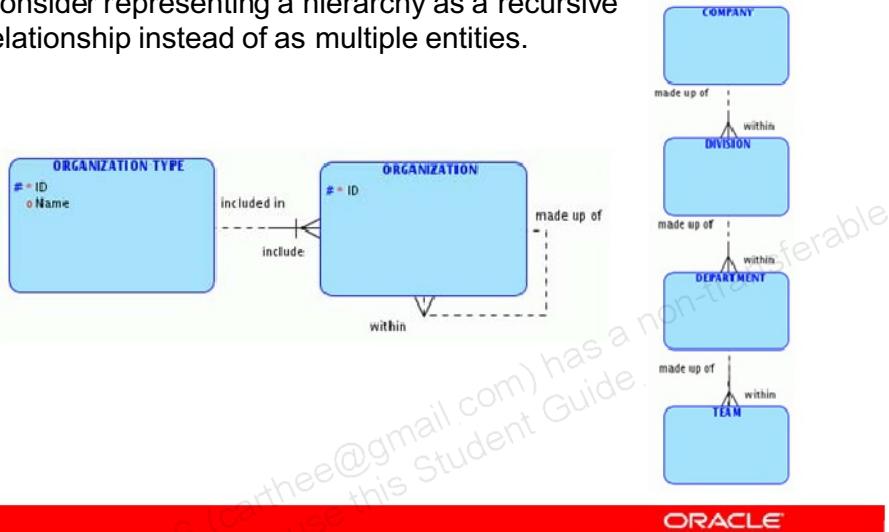
One technique for modeling relationships is to create a set of M:1 relationships hierarchically. In the example in the slide, an organization has a hierarchical structure:

- A COMPANY can consist of one or more DIVISIONS.
- A DIVISION can consist of one or more DEPARTMENTS.
- A DEPARTMENT can consist of one or more TEAMS.

The UIDs for a set of hierarchical entities may be propagated through multiple relationships. Consider creating artificial attributes to help identify entities in a hierarchical relationship.

Examining Recursive Relationships

Consider representing a hierarchy as a recursive relationship instead of as multiple entities.

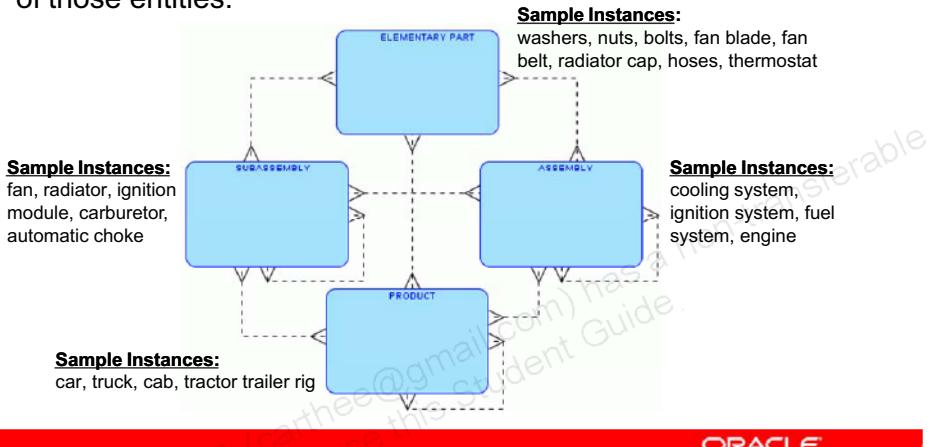


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The single recursive entity must include all of the attributes of each individual entity. Note that a recursive organization model cannot handle a mandatory relationship. If each ORGANIZATION must be within another ORGANIZATION, the organization hierarchy would have to be infinitely large. Therefore, the recursive relationship must be optional in both directions. In addition, there may be many organizations of the same type. As a result, ORGANIZATION TYPE should have its own entity to store the name and an identifying relationship with the ORGANIZATION entity.

Examining Recursive Relationships

“Bill of Materials” data can be modeled with multiple entities for each category of “part” and a set of relationships between each of those entities.



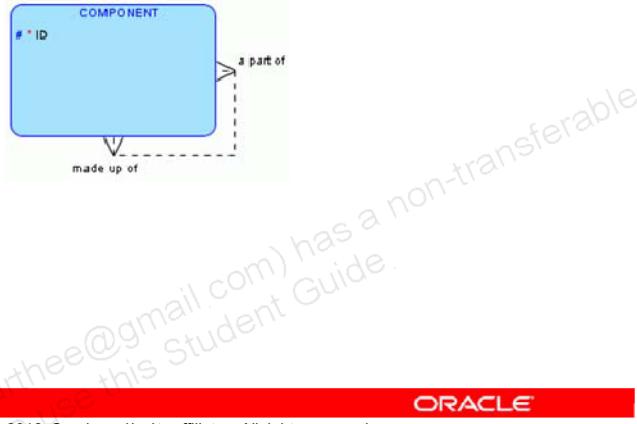
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

An automobile manufacturing organization needs to track elementary parts, subassemblies, assemblies, and products. The ERD in the slide depicts the data by considering each of these part categories as an entity.

Examining Recursive Relationships

Another way of modeling the same thing is to create one entity with a recursive relationship.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

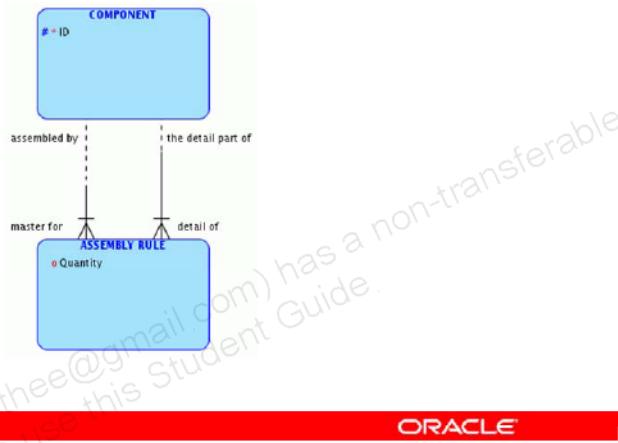
Consider all elementary parts, subassemblies, assemblies, and products as instances of an entity called **COMPONENT**. Then the complex ERD on the previous page can be remodeled as a simple recursive relationship.

Suppose that some of the instance data supplied in the previous slide is used to validate this model:

- Washers, nuts and bolts may be part of VARIOUS SUBASSEMBLIES, ASSEMBLIES, and PRODUCTS.
- A fan blade and a fan belt are part of a fan.
- A radiator cap, host, and thermostat are part of a radiator.
- A cooling system, ignition system, and engine block are part of a complete engine.

Resolving a M:M Recursive Relationships

Resolve the M:M recursive relationship with an intersection entity.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

The attribute **Quantity** is associated with the recursive relationship, and, therefore, it must be resolved. Resolve the M:M recursive relationship by adding the **ASSEMBLY RULE** intersection entity and two 1:M relationships back to the **COMPONENT** entity. **ASSEMBLY RULE** will have an attribute of **Quantity**.

Using the instance data again, the **ASSEMBLY RULE** instance for washers to fan will have a 1:M relationship to the **COMPONENT** instance for washer and a second 1:M relationship to the **COMPONENT** instance for fan. The **ASSEMBLY RULE** entity will record the quantity of washers that are a part of a single fan.

Quiz

An Employee is managed by a Manager. This is an example of a recursive relationship.

- a. True
- b. False

Answer: a

Lesson Agenda

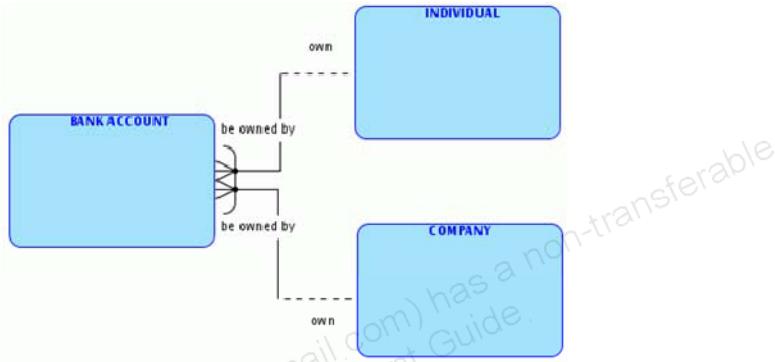
- Resolve M:M relationships
- Model hierarchical data
- Examine recursive relationships
- Model exclusive relationships
- Model entity type hierarchies
- Model data over time

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Modeling Exclusive Relationships

Modeling two or more mutually exclusive relationships from the same entity by using an arc.



ORACLE®

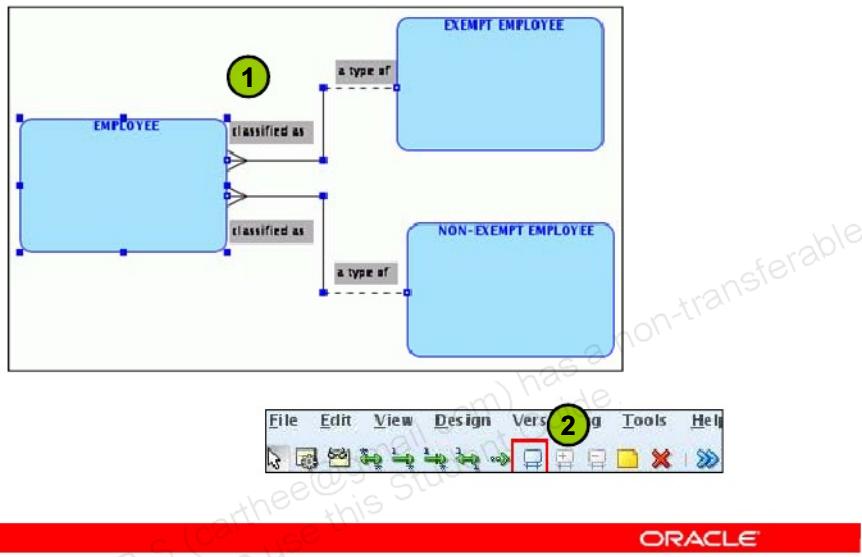
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

An exclusive relationship occurs when two or more mutually exclusive relationships from the same entity use an arc. The relationship implies an “or” condition. The business rule for the example in the slide is the following: Each BANK ACCOUNT either must be owned by one and only one INDIVIDUAL or must be owned by one and only one COMPANY.

The following characteristics exist with exclusive relationships:

- The relationships in an arc frequently have the same relationship name.
- The relationships in an arc must be either all mandatory or all optional.
- An arc belongs to a single entity and must include only relationships originating from that entity.
- An entity may have multiple arcs, but a specific relationship can participate in only a single arc.

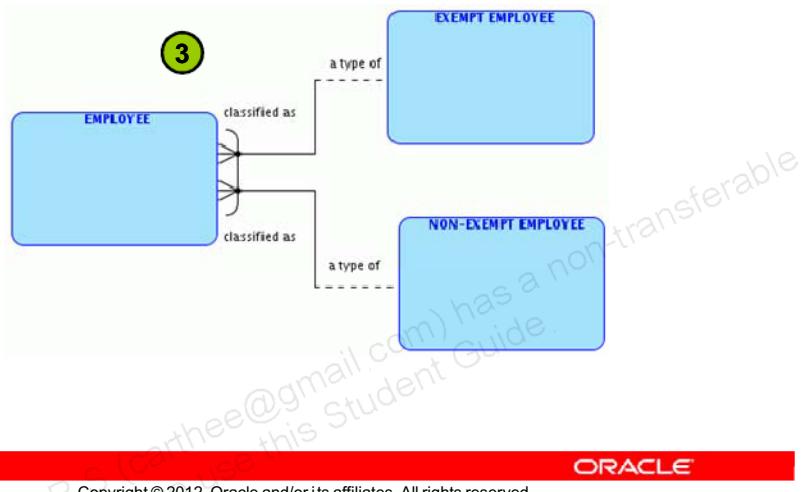
Creating an Exclusive Relationship in Oracle SQL Developer Data Modeler



To create an exclusive relationship in Oracle SQL Developer Data Modeler, you must perform the following steps:

1. Select the intersecting entity *and* both relationships on which you want to create the exclusive relationship.
2. Click the New Arc icon in the toolbar.

Creating an Exclusive Relationship in Oracle SQL Developer Data Modeler



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

3. The exclusive relationship is created with the arc.

The business rule for this relationship is read as follows:

"Each EMPLOYEE either must be classified as one and only one EXEMPT_EMPLOYEE or must be classified as one and only one NON_EXEMPT_EMPLOYEE."

Quiz

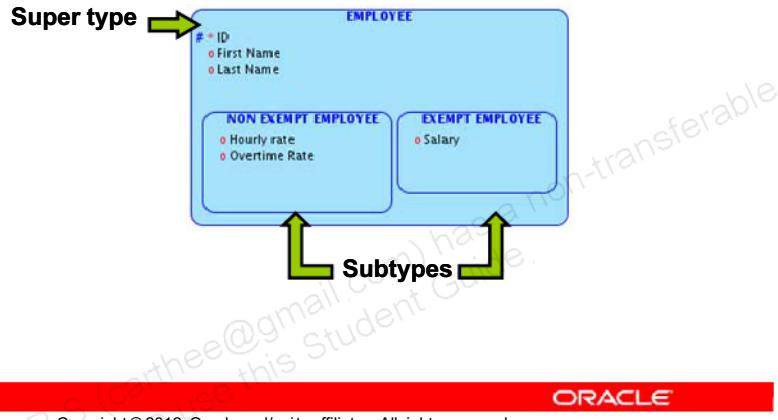
Which relationships are examples of an exclusive relationships?

- a. A DEPARTMENT can be housed in one location or another.
- b. An AIRCRAFT can be classified as either an AIRPLANE or HELICOPTER
- c. A DEPENDENT can be either Male or Female.
- d. ORDERS are either complete or not.

Answer: b

Entity Type Hierarchies

Use entity type hierarchies to model exclusive entity types that have common attributes and common relationships.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Entity type hierarchies are used to model exclusive entity types that have common attributes and common relationships. Often an exclusive relationship (discussed in the previous section) can be modeled as an entity type hierarchy. An entity type hierarchy contains the following components:

- **Super type:** An entity that contains two or more subtypes
- **Subtype:** An entity that is of a particular type or group of the super type and contains specific attributes that are not contained in other subtypes of the same super type

Entity type hierarchies have the following characteristics:

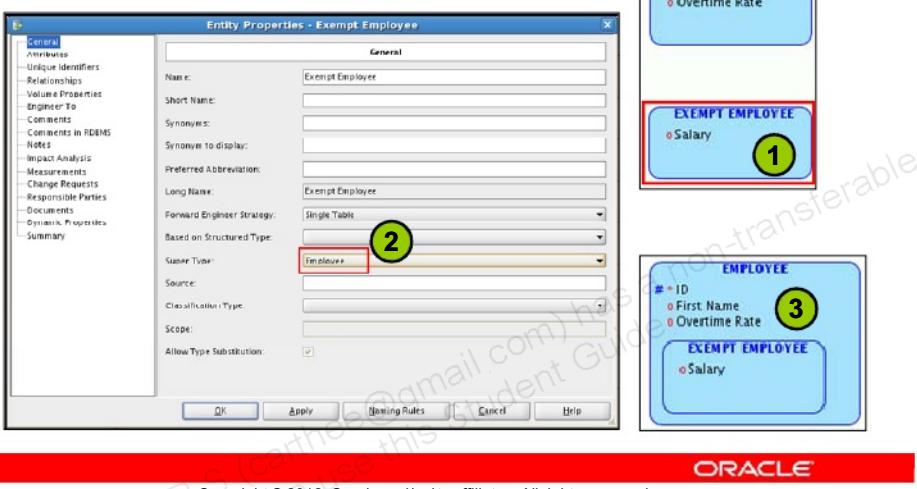
- Each super type must have at least two subtypes.
- The subtypes inherit all the attributes of the super type.
- Each subtype has its own attributes or relationships.
- A subtype may have its own primary key or inherit the primary key from the super type.
- Each instance of the super type entity must belong to one and only one subtype entity.
- Subtypes can be further subtyped. Try not to model more than two or three levels deep.

In the example in the slide, the EMPLOYEE example used in the previous section can alternatively be modeled as an entity type hierarchy because the attributes from the EMPLOYEE super type entity can be inherited by the subtypes EXEMPT EMPLOYEE and NON-EXEMPT EMPLOYEE. There are also specific attributes that are stored for the EXEMPT EMPLOYEE that are not needed for the NON-EXEMPT EMPLOYEE (and vice versa).

Note that some subtypes with no attributes or relationships of their own are synonyms of their super types and are not subtypes in their own right.

Modeling Subtypes in Oracle SQL Developer Data Modeler

Select the super type entity for each subtype.



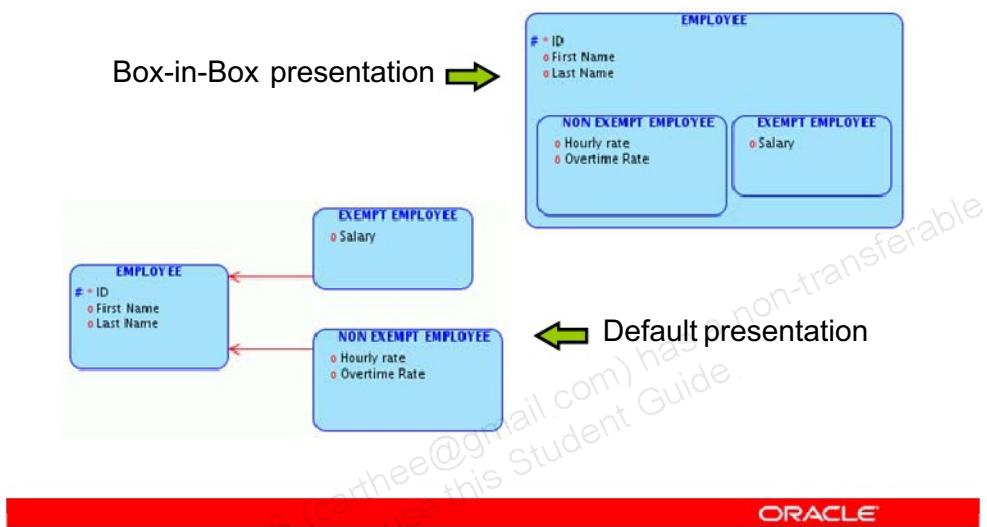
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

To define an entity as a subtype in Oracle SQL Developer Data Modeler, perform the following steps:

1. Double-click the entity that you want to make a subtype. In the example in the slide, you want to make EXEMPT EMPLOYEE a subtype of the EMPLOYEE super type. Double-click EXEMPT EMPLOYEE.
2. Select the EMPLOYEE super type entity from the Super Type list, and click OK.
3. The EXEMPT EMPLOYEE entity is now a subtype of the EMPLOYEE super type and will inherit all the attributes of the super type.

Representing Entity Type Hierarchies

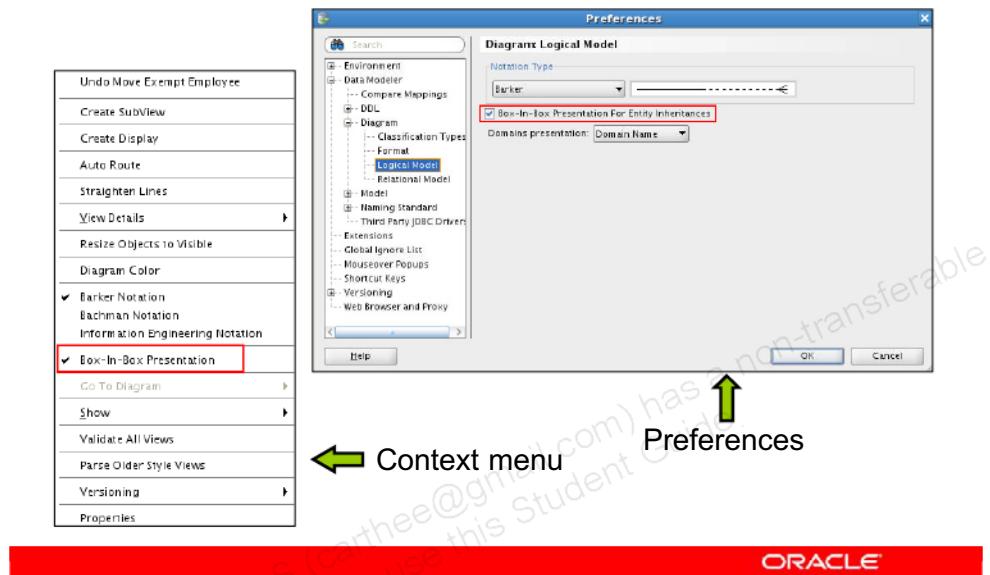


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

There are two ways to represent entity type hierarchies: the default presentation and the Box-in-Box representation.

Changing Preferences for Box-in-Box Presentation



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

To change the default presentation to Box-in-Box Presentation, select Tools > Preferences, expand Data Modeler > Diagram, and select Logical Model. Select “Box-In-Box Presentation For Entity Inheritances” and click OK.

To change the notation within the diagram, right-click the background of the logical model to get the context menu, and select or deselect the Box-In-Box Presentation check box.

Quiz

Which of the following is a characteristic of entity type hierarchies?

- a. The subtypes inherit the attributes from the super type.
- b. The subtypes can have relationships with other entities in the logical data model.
- c. There is an implied exclusive relationship between subtype entities.
- d. All of the above

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: d

Model Data over Time

Add additional entities and relationships to the model to accommodate historical data.

- Is an audit trail required?
- Can attribute values change over time?
- Can relationships change over time?
- Do you need to query older data?



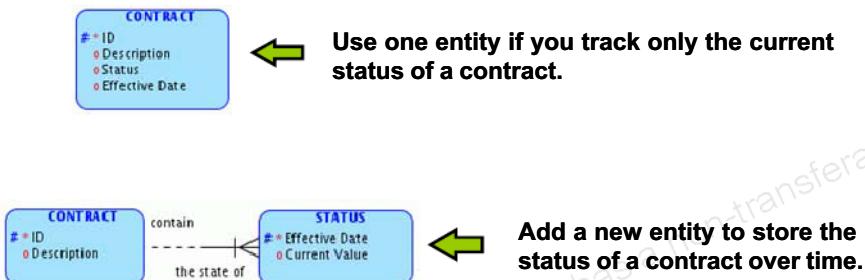
ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Every update of an attribute or transfer of a relationship means loss of information. Often information is no longer of use, but some systems must keep track of some or all of the historical values of an attribute. In addition, some systems are required to keep an audit trail of each transaction. Validate any requirements for storing historical data with the user, because storing unnecessary historical data can be costly.

Model Data over Time

Create additional entities to track attributes' values over time.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

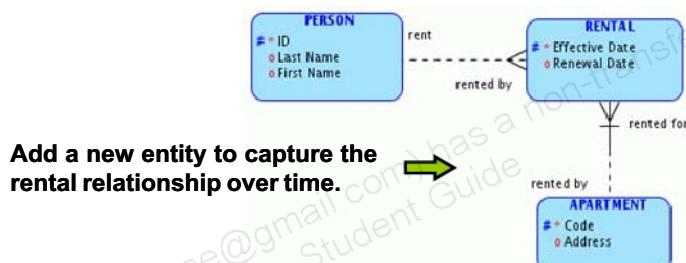
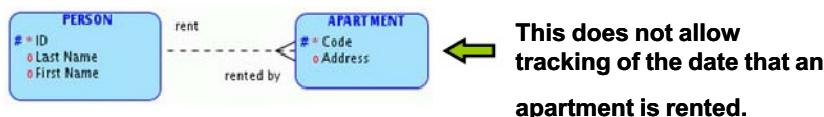
ORACLE®

Create an additional entity to track an attribute's value over time. As shown in the slide, the CONTRACT entity was initially created with four attributes. Because all the attributes are contained in one entity, the contract's status and effective date will change to the current status and effective date each time the contract changes. If you are required to track *when* the status of a particular contract changes, create an additional entity and move the status and effective date to the new entity. In this way, the information about when the status of a contract changed can be tracked.

Note that if the status changes more than once on the same day, you should include a current status value attribute in the unique identifier.

Model Data over Time

Add a new entity to accommodate a relationship that may change over time.



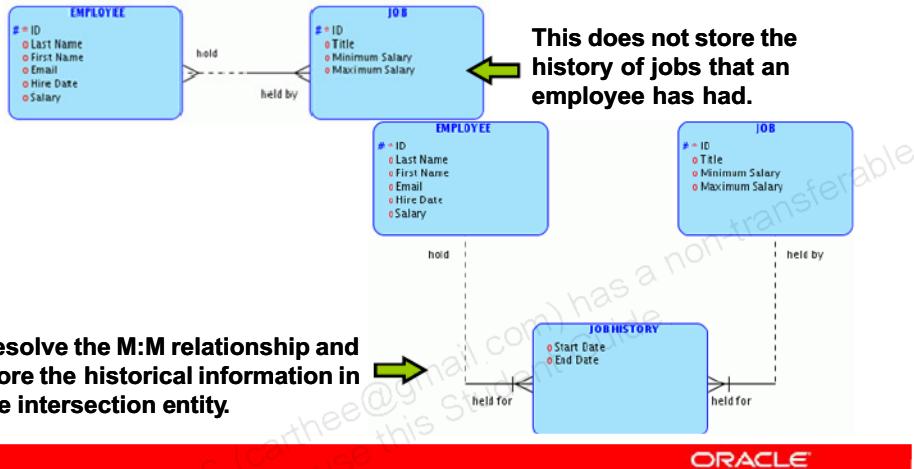
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

You may need to add an entity to keep track of how a relationship changes over time. In the example in the slide, a person can rent an apartment but an apartment can be vacant. If you want to track which apartments are rented at a particular point in time, add a new entity to store rental information about an apartment. In the RENTAL entity, the apartment code and rental effective date are the unique identifier. You would not include the person ID since you are storing the current and historical rentals. You would not have two rentals for the same apartment with the same effective date.

Model Data over Time

An intersection entity is frequently used to track information about a relationship that changes over time.



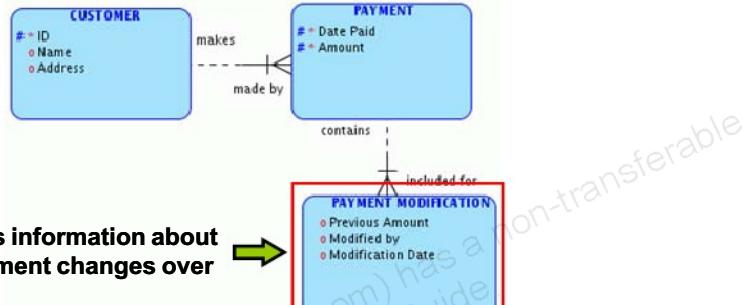
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

Resolving M:M relationships in an intersection entity often allows you to track information about a relationship over time. In the example in the slide, you have a M:M relationship between **EMPLOYEE** and **JOB**. There is no way to store information about what jobs a particular employee has had over time. The **JOB HISTORY** entity stores this information with two identifying relationships with the **EMPLOYEE** and **JOB** entities.

Model Data over Time

Additional entities and attributes store modification information.



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

When a system allows a user to modify or remove particular information, you may decide to keep the old values. This is called *logging* or *journaling* and is common in financial type models.

In the example in the slide, a customer may make multiple payments and you can make changes to a payment. If you want to track changes to the payments that are made, you can store the history in the PAYMENT MODIFICATION entity.

Quiz

Which of the following would involve creating a new entity to store additional information to track data changes over time?

- a. When an employee terminates employment
- b. When a customer's address changes
- c. When the price of an item changes
- d. When a person's lease is up

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary

In this lesson, you should have learned how to:

- Resolve M:M relationships
- Model hierarchical data
- Examine recursive relationships
- Model exclusive relationships
- Model entity type hierarchies
- Model data over time

Practice 12-1 Overview: Resolve M:M Relationships

This practice covers the following topics:

- Resolving M:M relationships
- Identifying attributes in the intersection entity

Practice 12-2 Overview: Model Hierarchical Data

This practice covers modeling entities, relationships, attributes, and unique identifiers that are hierarchical.



Practice 12-3 Overview: Model Hierarchical Data and Recursive Relationships

This practice covers modeling entities and relationships that are hierarchical and recursive.

Practice 12-4 Overview: Examine Exclusive Relationships

This practice covers determining how to model an exclusive relationship.

Practice 12-5 Overview: Examine Exclusive Relationships

This practice covers determining how to model an exclusive relationship from a case scenario.

In this practice, you build an ERD that contains exclusive relationships.

Adding and Using Data Types

13

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Create different data types
- Build a data type model
- Analyze various relationships between structured types in your data type model
- Assign data types to the attributes in your logical data model



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

This lesson describes the various data types as well as how to create them and use them in your model.

Lesson Agenda

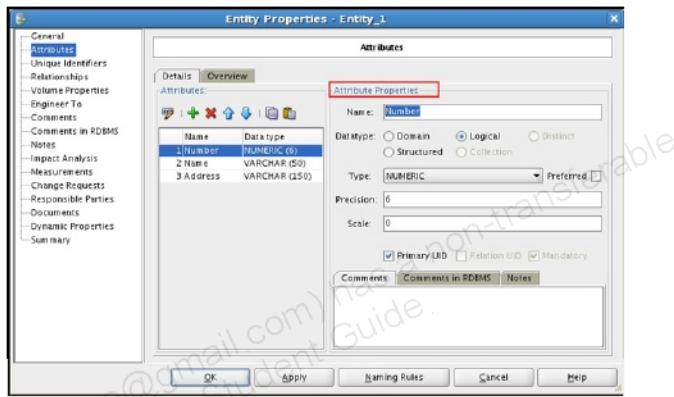
- Create different data types
- Build a data type model
- Analyze various relationships between structured types in your data type model
- Assign data types to the attributes in your logical data model

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Attribute Data Types

- A data type defines the format and length of an attribute.
- Data type classifications:
 - Logical
 - Domain
 - Distinct
 - Structured
 - Collection



ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

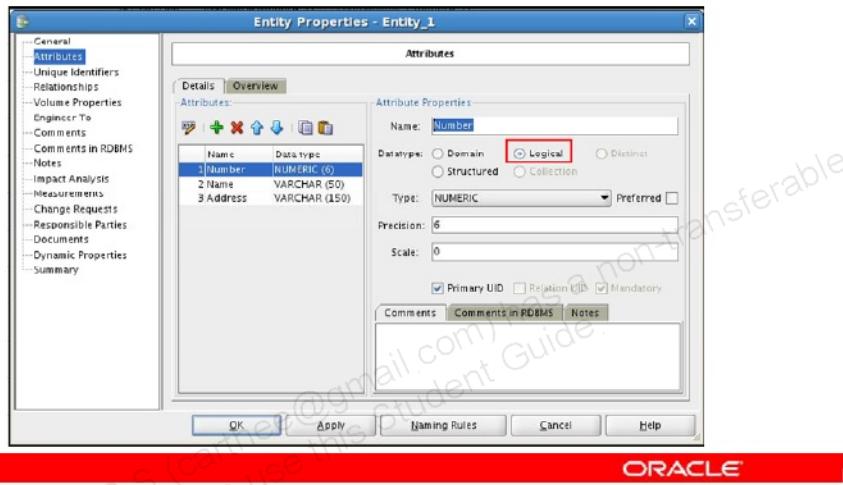
Each attribute in your logical data model must be assigned a data type that defines its format and length. The default data type is unknown.

Several data type classifications are discussed in this lesson:

- Logical
- Domain
- Distinct
- Structured
- Collection

Logical Type

A logical type is a basic element that is database independent.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Logical types are basic elements that are database independent. A set of predefined logical types has been supplied with Oracle SQL Developer Data Modeler. Examples of logical types include VARCHAR, CHAR, and NUMERIC. You can delete all the predefined logical types and create your own types; however, you must make two mappings for the following purposes:

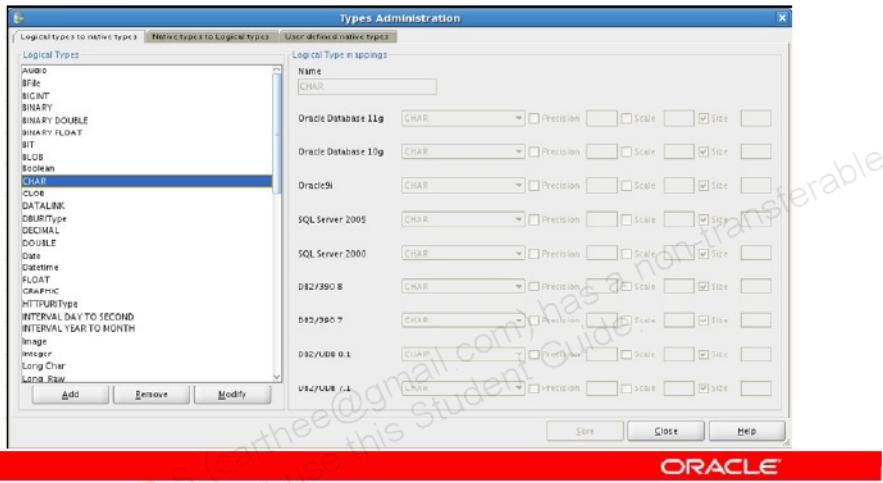
- To generate DDL, each logical type must be mapped to the native database type for each supported database type.
- To import from the data dictionary or from a DDL script, each native data type must be mapped to a logical type.

To associate a logical type with an attribute, select Logical as the Datatype and then select a logical data type from the Type drop-down list.

Note: Be careful not to delete predefined logical types unless the organization's procedures and standards require it.

Types Administration

In Types Administration, map logical data types and native data types for specific databases.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can access Types Administration from the Tools menu to manage the mappings between logical data types and native data types for specific supported database products. You can also add and remove logical types with Types Administration.

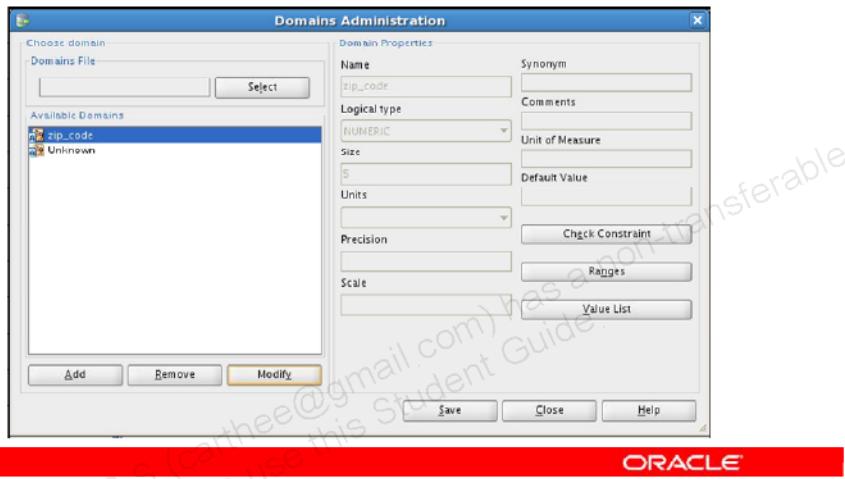
Logical types to native types: For each logical type, you can select the type in the list on the left to view its mappings to a type in each supported database product.

To add a logical type and specify its mappings, click Add. To delete a logical type, select the type and click Remove. To modify the mappings for a logical type (predefined or user-defined), select the type, click Modify, and specify the mapping information for any desired database products.

Native types to logical types: For each supported database product, you can view the mappings between its native types and Oracle SQL Developer Data Modeler logical types.

Domain

A domain adds meaning to a logical type and can define possible values.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

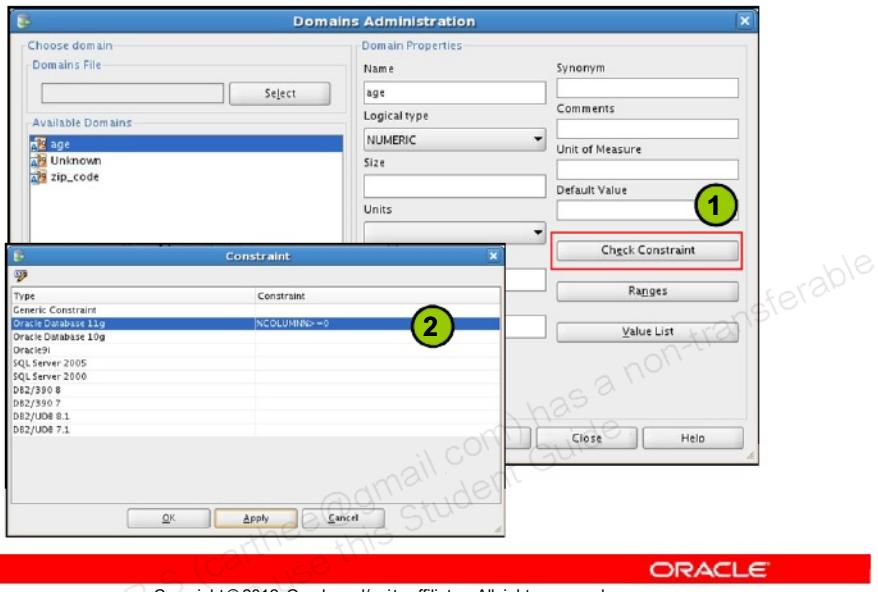
A domain adds meaning to a logical type such as zip_code, which is assigned the logical type NUMERIC with a size of five characters. In addition, you can define a possible set of values by using a check constraint, list of values, or list of ranges. Note that the list of values for a domain should be used only if the list of values is fixed (because it will become a constraint in the database). If the values change, create an entity to store the values (which will eventually become a lookup table in the database).

You can assign the same domain to multiple attributes. If the details of a domain change, all the attributes to which it is assigned will also have that change.

The Default Value property can be set or changed at the Column or Attribute level. A list of values defined at the Domain, Column, or Attribute level is used to pick up the Default Value.

To access Domains Administration, select Tools > Domains Administration. In Domains Administration, you can define your set of domains by adding, modifying, or removing domains and also by defining check constraints, ranges, and value lists.

Adding a Check Constraint to a Domain

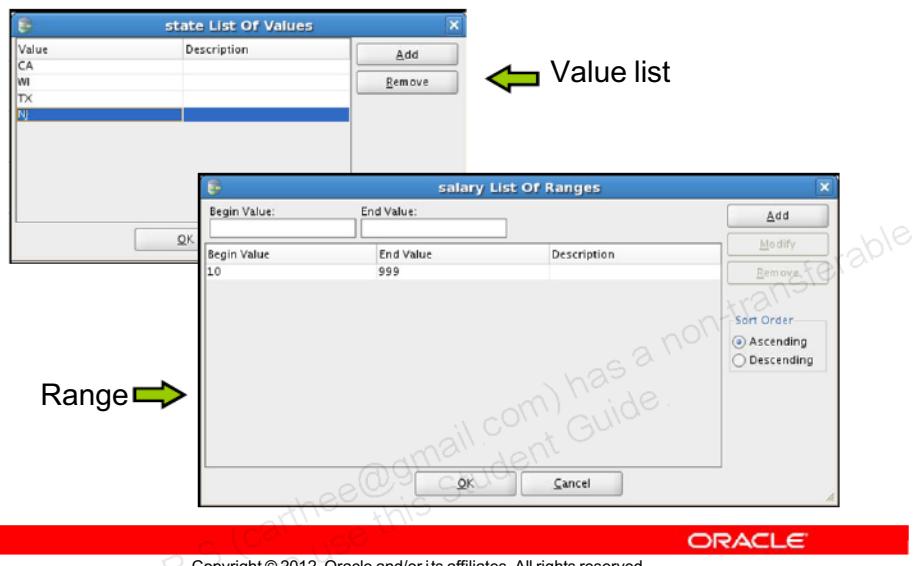


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To define a check constraint for a domain, perform the following steps:

1. Create your domain and click Check Constraint.
2. Enter the constraint for the database that you will be generating. In the example in the slide, you specify %COLUMN% \geq 0 for the constraint that is created when generating to Oracle Database 11g.

Adding a Range or Value List to a Domain



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

You can define a range where the attribute must be within the range defined. You can also define a list of values. For example in the slide, a list of states is provided as a valid set of values for the state domain.

To define a value list for a domain:

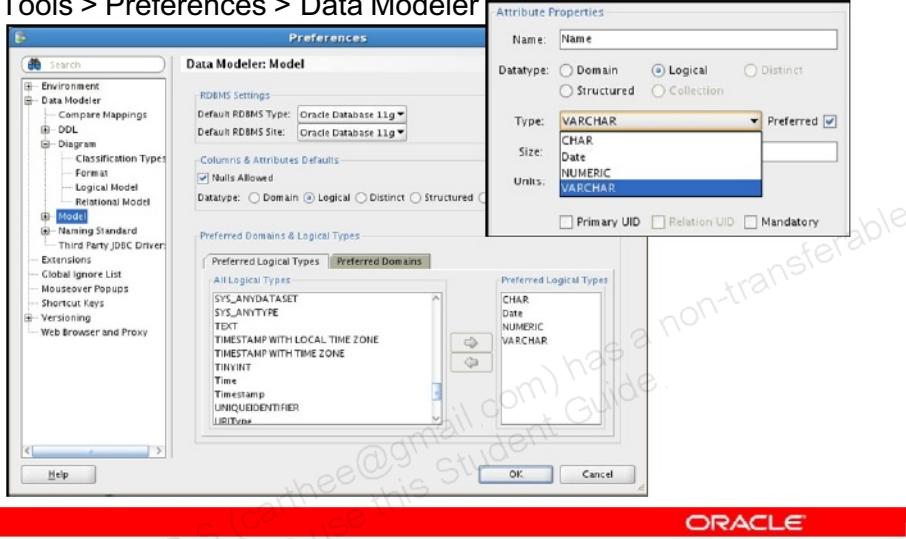
1. Create your domain.
2. Click Value List.

To define a range for a domain:

1. Create your domain.
2. Click Ranges.

Preferred Logical Types and Domains

Tools > Preferences > Data Modeler



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

To identify the default data type and to modify the preferred set of logical types and domains for your model, perform the following steps:

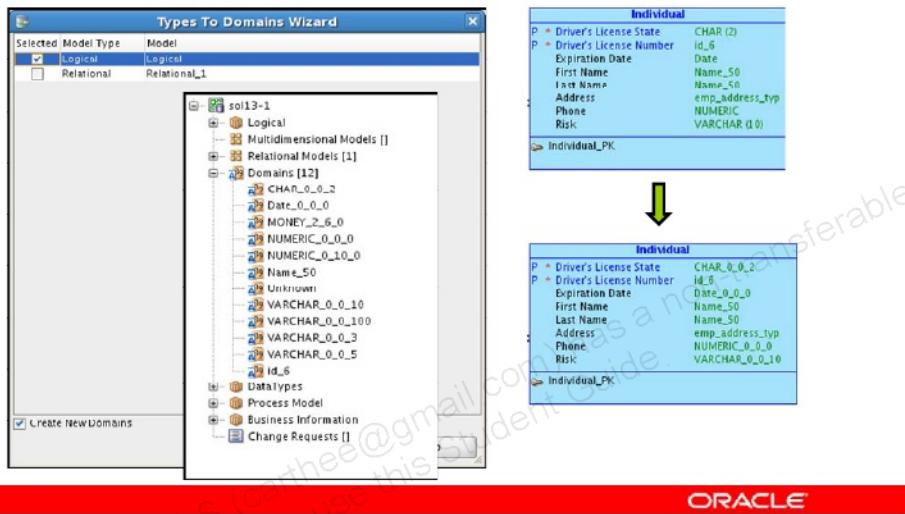
1. Select **Tools > Preferences > Data Modeler**, and select the **Model** node.
2. Select the default data type in the **Column & Attributes Defaults** section. In the example in the slide, **Logical** was selected as the default data type.
3. In the Preferred Domains and Logical Types section, select the logical types and click the right arrow to move them to the preferred area. Click the **Preferred Domains** tab and perform the same step.
4. Click **OK**.

To see the list of preferred logical types and domains, perform the following steps:

1. Double-click the entity (or table) in the diagram.
2. Click the Attribute (or Column) property in the left navigator.
3. Select or create your attribute (or column).
4. Select the Preferred check box and notice that the list of preferred logical types or domains is listed.

Creating Domains from Logical Types

Tools > Types To Domains Wizard



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

You can create domains from logical types already defined in your model. In the example in the slide, the INDIVIDUAL entity contains multiple logical types, such as Phone or Risk. If you want to create domains of these logical types, perform the following steps:

1. Select **Tools > Types** To Domains Wizard.
2. Select the model type that you want to evaluate and create domains for.
3. Make sure that the **Create New Domains** check box is selected, and click **OK**.
4. Expand the **Domains** node in the object browser to see the domains that were created.
You can also see in the screenshot in the slide that the domain was replaced as the data type for each attribute in the INDIVIDUAL entity.

Note that the domains created using this method are only saved for this model.

Lesson Agenda

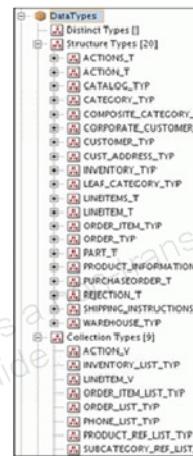
- Create different data types
- Build a data type model
- Analyze various relationships between structured types in your data type model
- Assign data types to the attributes in your logical data model

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Data Type Model

- A data type model is a set of defined object definitions that can be used in the logical model and in relational models as data types.
- Types of objects in a data type model:
 - Distinct type
 - Structured type
 - Collection type



ORACLE®

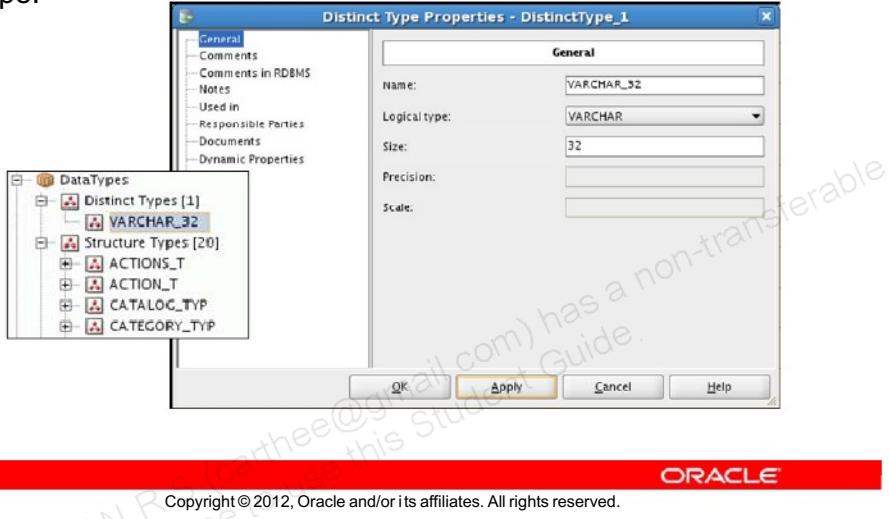
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

A data type model allows you to define object definitions of distinct, structured, and collection types that can be used in the logical model and in relational models as data types. These three types are SQL99 elements.

All data type model objects (except logical types and domains) are displayed in the object browser tree, but only structured type objects and their interrelations are represented graphically on data types diagrams.

Distinct Type

A distinct type is a data type derived from an existing logical type.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

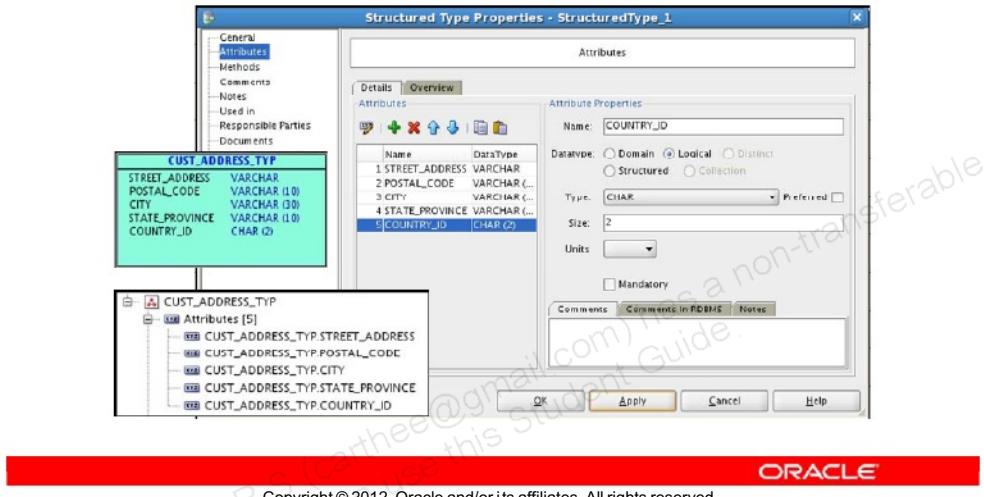
ORACLE®

Distinct types introduce meaning to a basic database type. DB2 and UDB have support for distinct types; however, they are not generated from the data type model.

Oracle SQL Developer Data Modeler can generate domains as a distinct type.

Structured Type

A structured type is a user-defined data type that has attributes and methods.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

In the example in the slide, a `cust_address_typ` structured type is defined and is made up of multiple attributes. This structured type, `cust_address_typ`, can be assigned to each address attribute in your logical data model. When the model is generated to create the database, the DDL to create the type is generated.

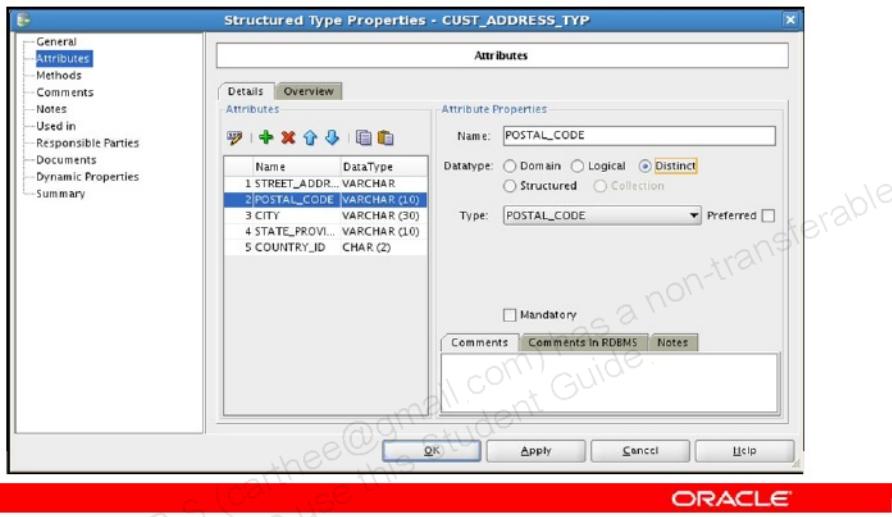
A structured type also includes a set of method specifications. Methods enable you to define behaviors for structured types. Like user-defined functions (UDFs), methods are routines that extend SQL. In the case of methods, however, the behavior is integrated only with a particular structured type.

The expanded structured types subfolder lists all structured type objects, with the hierarchy of attributes and methods for each.

The Oracle Spatial SDO_GEOOMETRY type is predefined as a structured type. In addition, you can create new structured types or edit the properties of existing structured types.

Using Distinct Types in a Structured Type

Assign attributes in a structured type to a distinct type.

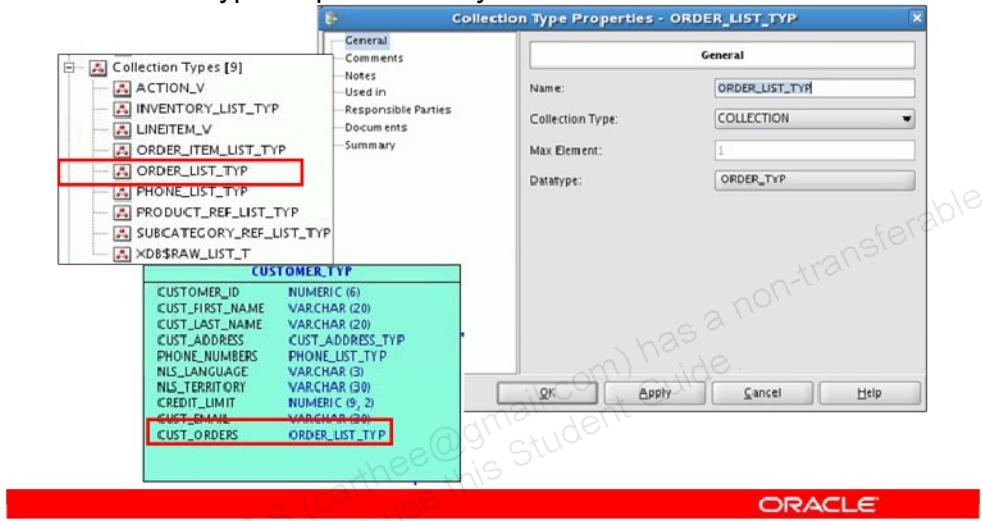


Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

You can assign an attribute in a structured type to a data type of a distinct type. In the example in the slide, the distinct type POSTAL_CODE was assigned to the POSTAL_CODE attribute in the structured type CUST_ADDRESS_TYP. Notice that the data type in the DataType column of the list of attributes shows the length and format of the distinct type instead of the distinct type name.

Collection Type

Collection types represent arrays or collections of elements.



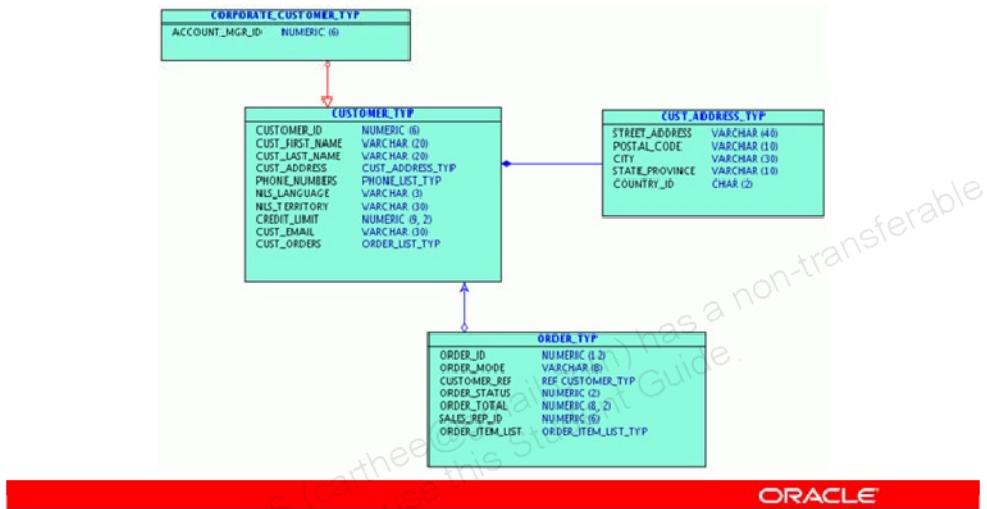
Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Collection types represent arrays or collections of elements (logical type, distinct type, structured type, or another collection) and are mapped to the Oracle VARRAY and nested table types in the database.

In the example in the slide, the `order_list_typ` collection type is assigned to the `cust_orders` attribute in the `customer_typ` structured type. The `order_list_typ` collection type's data type is the `order_typ` structured type.

Building a Data Type Model

Diagram of structured types



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

A data type model is a graphic display of structured types.

Structured types can also be part of a super type and subtype inheritance hierarchy within the data type model. In the example in the slide, `corporate_customer_typ` is a subtype of the `customer_typ` super type. A structured type can be defined based on a logical data type, a distinct type, another structured type, or a reference to a structured type, or it can be defined as a collection type.

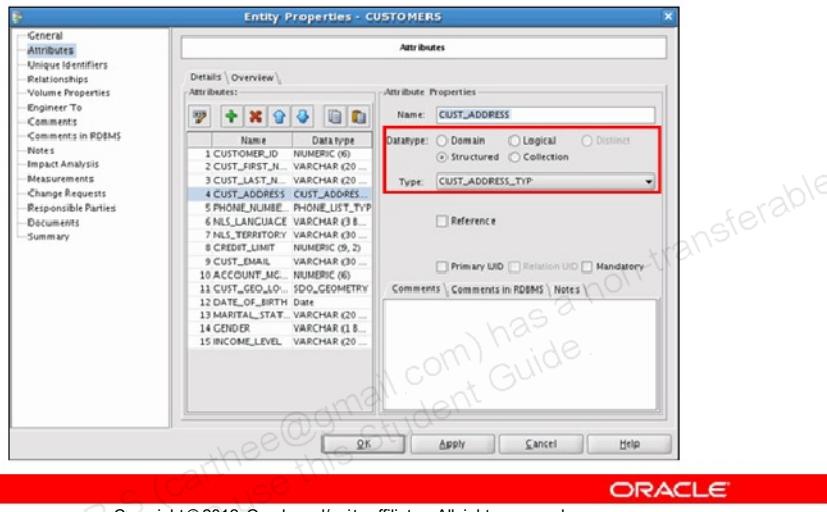
A structured type can be assigned to an attribute in another structured type. In the example in the slide, the `cust_address_typ` structured type is assigned to the `cust_address` attribute of the `customer_typ` structured type.

In addition, an attribute can reference another structured type. In the example in the slide, the `customer_ref` attribute references the `customer_typ` structured type.

Domains can be used in definitions of a data type model.

Assigning Data Types to an Attribute

Select a data type for an attribute, and then specify details.



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

ORACLE®

After you define your domains and data type model, you can assign each attribute its data type.

Perform the following steps:

1. Double-click the entity.
2. Select Attributes in the left navigator, and then select or create the attribute you want to define.
3. Select the data type category.
4. Depending on the category selected, select from the Type list or specify the length in the text field.

In the example in the slide, the data type for the `CUST_ADDRESS` attribute in the `CUSTOMER` entity is of the Structured category, and the `CUST_ADDRESS_TYP` is selected.

Quiz

Which data type is contained in the data type model diagram?

- a. Domain
- b. Structured type
- c. Logical type
- d. Collection type
- e. Distinct

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Quiz

You can specify domains in a data type model.

- a. True
- b. False

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Answer: b

Summary

In this lesson, you should have learned how to:

- Create different data types
- Build a data type model
- Analyze various relationships between structured types in your data type model
- Assign data types to the attributes in your logical data model

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Practice 13-1 Overview: Create and Assign Data Types

This practice covers the following topics:

- Creating a data type model
- Defining attribute data types

In this practice, you create a data type model and then assign data types to each attribute in your data model.

Unauthorized reproduction or distribution prohibited. Copyright© 2013, Oracle and/or its affiliates.

KARTHIKEYAN R S (carthee@gmail.com) has a non-transferable
license to use this Student Guide.

14

Putting It All Together

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Build an ERD in Oracle SQL Developer Data Modeler from a case study
- Revise and examine the ERD based on modeling techniques



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

In this lesson, you build an ERD from the beginning and examine the ERD based on the modeling techniques you have learned in previous lessons. The practices in this lesson bring together all the concepts and techniques learned up to this point in the course.

Practice 14-1 Overview: Develop and Validate Your ERD

- Develop an ERD in Oracle SQL Developer Data Modeling:
 - Identify the entities.
 - Identify the attributes and define the data types for each attribute.
 - Identify unique identifiers.
 - Identify relationships and label each cardinality.
- Validate the ERD by using techniques discussed in previous lessons:
 - Normalize to third normal form.
 - Examine for recursive or exclusive relationships.
 - Examine for entity type hierarchies.
 - Examine entities and relationships to make sure that the model accounts for data changing over time.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Examine the Cascade Cruises case study and complete each task in the slide.

Practice 14-2 Overview: Develop and Validate Your ERD (Optional)

- Develop an ERD in Oracle SQL Developer Data Modeling:
 - Identify the entities.
 - Identify the attributes and define the data types for each attribute.
 - Identify unique identifiers.
 - Identify relationships and label each cardinality.
- Validate the ERD using techniques discussed in previous lessons:
 - Normalize to third normal form.
 - Examine for recursive or exclusive relationships.
 - Examine for entity type hierarchies.
 - Examine entities and relationships to make sure that the model accounts for data changing over time.

ORACLE®

Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Examine the Law Firm case study and complete each task in the slide.

Summary

In this lesson, you should have learned how to:

- Build an ERD in Oracle SQL Developer Data Modeler from a case study
- Revise and examine the ERD based on modeling techniques



Copyright © 2012, Oracle and/or its affiliates. All rights reserved.

Unauthorized reproduction or distribution prohibited. Copyright© 2013, Oracle and/or its affiliates.

KARTHIKEYAN R S (carthee@gmail.com) has a non-transferable
license to use this Student Guide.