

Oracle Data Modeling and Relational Database Design

Volume I • Student Guide

D56497GC10

Edition 1.0

September 2010

D67007

ORACLE®

Author

Marcie Young

Technical Contributors and Reviewers

Sue Harper
Philip Stoyanov
Nancy Greenberg
Rick Green
Brian Pottle
Anjula Subbiahpillai
Gerry Jurrens
Nick Donatone
David Lapoint
Tom Provenzano
Mike Ritz
Tim Trauernicht
Zhicheng Xu
Ron Berry
David Lyons
Kim Bell
Maria Billings
Steve Friedberg
Bryan Roberts
Priyanka Sharma
Matthew Gregory
Angelika Krupp

Editors

Daniel Milne
Vijayalakshmi Narasimhan

Graphic Designer

Rajiv Chandrabhanu

Publishers

Shaik Basha
Jayanthi Keshavamurthy

Copyright © 2010, 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

I Setting the Stage

- Overview I-2

1 Introduction to Modeling

- Objectives 1-2
- Why Model? 1-3
- Why Model: A Practical Example 1-4
- Database and Application Development Life Cycle 1-5
- Process Modeling 1-6
- Logical Data Modeling 1-8
- Database Design 1-10
- Database Generation 1-11
- Data Type Model 1-12
- Multidimensional Model 1-13
- Quiz 1-15
- Approaches to Modeling 1-17
- Top-Down Modeling 1-18
- Bottom-Up Modeling 1-19
- Targeted Modeling 1-20
- Quiz 1-21
- Summary 1-23
- Practice 1-1 Overview: Identify the Modeling Approach 1-24

2 Documenting the Business Background

- Objectives 2-2
- Documenting the Business Direction 2-3

Components of a Business Direction Statement	2-4
Business Objectives	2-5
Assumptions	2-6
Critical Success Factors	2-7
Key Performance Indicators	2-8
Problems	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

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

Overview II-2

3 Building a Process Model (Data Flow Diagram)

Objectives 3-2

What Is a Process Model? 3-3

Why Create a DFD? 3-4

Components of a Data Flow Diagram 3-5

Process 3-6

External Agents 3-7

Information Stores 3-8

Information Flows 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

Oracle SQL Developer Data Modeler 4-3

Oracle SQL Developer Data Modeler Main Window 4-5

Specifying General Options: General 4-6

Specifying General Options: Model 4-7

Specifying General Options: Diagram 4-8

Specifying General Options: Naming Standard 4-9

Building a Data Flow Diagram 4-10

Editing the Diagram Layout 4-16

Adding and Reusing Process Events	4-20
Opening and Saving Your Model	4-21
Summary	4-22
Practice 4-1 Overview: Build a Data Flow Diagram in Oracle SQL Developer Data Modeler	4-23
 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
 III Developing a Logical Data Model	
Overview	III-2
 6 Identifying Entities and Attributes	
Objectives	6-2
What Is a Logical Data Model?	6-3
Why Create an ERD?	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
Relationships 7-3
Components of a Relationship 7-4
Relationships: Additional Examples 7-5
Quiz 7-6
Class Practice: Define Business Rules 7-7
Relationship Types 7-8
Many-to-One Relationships 7-9
Many-to-Many Relationships 7-10
One-to-One Relationships 7-11
Recursive Relationships 7-12
Quiz 7-13
Using a Relationship Matrix 7-14
Determining a Relationship's Existence 7-16
Naming the Relationship 7-17
Determining the Relationship's Cardinality 7-18
Validating the Relationship 7-20
Quiz 7-21
Class Practice: Build a Relationship Matrix 7-22
Summary 7-23
Practice 7-1 Overview: Analyze and Model Relationships 7-24
Practice 7-2 Overview: Analyze and Model Relationships 7-25

8 Assigning Unique Identifiers

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

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

Diagram

Objectives 9-2

Building an Entity Relationship Diagram 9-3

Specifying Logical Model General Option 9-9

Specifying Logical Model Diagram Defaults 9-10

Modifying Model Properties 9-11

Notation Types 9-12

Editing a Diagram Layout: Moving an Object 9-13

Editing a Diagram Layout: Redrawing Lines 9-14

Editing a Diagram Layout: Moving a Relationship Line 9-15

Editing a Diagram Layout: Adding an Elbow 9-17

Editing a Diagram Layout: Showing Levels of Detail 9-18

Editing a Diagram Layout: Resizing Multiple Objects 9-19

Editing a Diagram Layout: Aligning Objects 9-21

What Is a Subview? 9-22

Creating a Subview 9-23

What Is a Display? 9-24

Creating a Display 9-25

Opening and Saving a Model 9-26

Exporting a Model 9-27

Importing a Model 9-28

Quiz 9-29

Summary 9-31

Practice 9-1 Overview: Build an ERD in Oracle SQL Developer Data Modeler 9-32

10 Validating Your Entity Relationship Diagram

Objectives 10-2

ERD Checklist 10-3

Attribute Rules 10-5

Distinguishing Attributes and Entities 10-6

Attribute Optionality 10-8

Naming Standards 10-9

Defining Naming Standards 10-11

Using a Glossary 10-13

Creating a Glossary 10-14

Applying the Glossary to the Naming Standards 10-15

Defining Abbreviations 10-16

Applying Design Rules 10-17

Adding Additional Information to the ERD 10-18

Quiz 10-19
Summary 10-21
Practice 10-1 Overview: Develop and Validate Your ERD 10-22

IV Utilizing Advanced Data Modeling Techniques

Overview IV-2

11 Normalizing Your Data Model

Objectives 11-2
What Is Normalization? 11-3
First Normal Form (1NF) 11-4
Second Normal Form (2NF) 11-5
Third Normal Form (3NF) 11-6
Quiz 11-7
Normalization Example: Unnormalized Data 11-8
Normalization Example: Transforming to First Normal Form 11-9
Normalization Example: Transforming to Second Normal Form 11-11
Normalization Example: Transforming to Third Normal Form 11-12
Summary 11-13
Practice 11-1 Overview: Normalize an ERD 11-14
Practice 11-2 Overview: Validate ERD for Normalization 11-15

12 Validating Relationships

Objectives 12-2
Resolving M:M Relationships 12-3
Quiz 12-6
Modeling Hierarchical Data 12-7
Examining Recursive Relationships 12-8
Resolving a M:M Recursive Relationships 12-11
Quiz 12-12
Modeling Exclusive Relationships 12-13
Creating an Exclusive Relationship in Oracle SQL Developer Data Modeler 12-14
Quiz 12-16
Entity Type Hierarchies 12-17
Modeling Subtypes in Oracle SQL Developer Data Modeler 12-19
Representing Entity Type Hierarchies 12-20
Changing Preference for Box-in-Box Presentation 12-21
Quiz 12-22
Model Data Over Time 12-23
Quiz 12-28
Summary 12-29

Practice 12-1 Overview: Resolve M:M Relationships	12-30
Practice 12-2 Overview: Model Hierarchical Data	12-31
Practice 12-3 Overview: Model Hierarchical Data and Recursive Relationships	12-32
Practice 12-4 Overview: Examine Exclusive Relationships	12-33
Practice 12-5 Overview: Examine Exclusive Relationships	12-34

13 Adding and Using Data Types

Objectives	13-2
Attribute Data Types	13-3
Logical Type	13-4
Types Administration	13-5
Domain	13-6
Adding a Check Constraint to a Domain	13-7
Adding Ranges or Value Lists to a Domain	13-8
Preferred Logical Types and Domains	13-9
Creating Domains from Logical Types	13-10
Data Type Model	13-11
Distinct Type	13-12
Structured Type	13-13
Using Distinct Types Within a Structured Type	13-14
Collection Type	13-15
Building a Data Type Model	13-16
Assigning Data Types to an Attribute	13-17
Quiz	13-18
Summary	13-20
Practice 13-1 Overview: Create and Assign Data Types	13-21

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

V Transforming Your Logical Model to a Relational Design

15 Mapping Your Entity Relationship Diagram to a Relational Database Design

Objectives	15-2
Why Create a Relational Model?	15-3
REVIEW: Database Design	15-4
Relational Database Overview	15-5
Terminology Mapping	15-6

Naming Conventions	15-7
Naming Restrictions with Oracle	15-11
Ensuring That Your Logical Data Model Is Complete	15-12
Mapping Simple Entities	15-13
Naming Entities	15-14
Engineering Entities	15-15
Mapping Attributes to Columns	15-16
Mapping Attributes to Columns: Column Names	15-17
Engineering Attributes	15-18
Reviewing the Glossary	15-19
Adding the Glossary as the Naming Standard	15-20
Mapping Attributes to Columns with the Glossary	15-21
Applying Name Abbreviations	15-22
Mapping Unique Identifiers to Primary Keys	15-23
Engineering Unique Identifiers	15-24
Mapping Relationships to Foreign Keys	15-25
Defining Naming Templates	15-27
Applying Templates to One Table	15-29
Applying Templates to the Relational Model	15-30
Managing Prefixes	15-31
Quiz	15-32
Practice 15-1 Overview: Create an Initial Relational Model	15-34
Mapping Exclusive Relationships to Foreign Keys	15-35
Engineering Exclusive Relationships	15-36
Mapping Subtypes to Tables	15-37
Engineering Subtypes	15-38
Mapping Subtypes to a Single Table	15-39
Changing the FWD Engineering Strategy	15-40
Engineering Subtypes to Table per Child	15-41
Mapping Subtypes for a Table per Child	15-42
Changing the FWD Engineering Strategy	15-43
Mapping Subtypes for a Table for Each Entity	15-44
Quiz	15-45
Applying General Options	15-46
Setting Compare/Copy Options	15-47
Viewing the Mapping Comparison	15-48
Synchronizing Deleted Objects	15-49
Identifying Overlapping and Folding Keys	15-50
Summary	15-52
Practice 15-2 Overview: Forward Engineer a Model	15-53

VI Evaluating Your Design for Database Creation

Overview VI-2

16 Analyzing Your Relational Model

- Objectives 16-2
- General Options: Relational Diagram 16-3
- Reviewing Table Properties 16-4
- Previewing the DDL for a Table 16-5
- General Options: Classification Types 16-6
- Assigning a Classification Type to One Table 16-7
- Changing the Color for Classified Tables 16-8
- Changing the Prefix for Classified Tables 16-9
- Assigning Classification Types to Multiple Tables 16-10
- Reviewing Column Properties 16-11
- Defining a Unique Constraint 16-12
- Defining Indexes 16-13
- Defining a Table-Level Constraint 16-15
- Specifying Volume Properties 16-16
- Defining Spatial Properties 16-17
- Defining Column Groups 16-21
- Analyzing Your View 16-22
- Quiz 16-24
- Summary 16-26
- Practice 16-1 Overview: Analyze Your Relational Model 16-27

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 18-9
- Adding Segment Templates (Storage) 18-10
- Associating Physical Objects with Your Table 18-11
- Propagating Properties to Other Physical Objects 18-12
- Partitioning a Table 18-13
- Creating a Materialized View 18-15
- Cloning a Database 18-16
- Quiz 18-18
- Summary 18-19
- Practice 18-1 Overview: Create a Physical Model 18-20

19 Generating Your Database

- Objectives 19-2
- Database Generation 19-3
- Generating DDL: Selecting a Database 19-4
- Generating DDL: ‘Create’ Selection 19-5
- Generating DDL: DDL Script 19-6
- Generating DDL: Assigned to Users 19-7
- Generating DDL: “Drop” Selection 19-8
- Generating DDL: Name Substitution 19-9
- Generating DDL: Including Table Scripts 19-10
- Generating DDL: Masking Oracle Errors 19-11
- Generating DDL: Using Find 19-13
- DDL General Options 19-14
- DDL/Migration General Options 19-17
- Summary 19-18
- Practice 19-1 Overview: Generate DDL 19-19

VII Other Needs for Modeling

- Overview VII-2

20 Altering an Existing Design

- Objectives 20-2
- Approaches to Modeling 20-3

Using Import	20-4
Importing an Existing Database	20-6
Importing Domains	20-11
Quiz	20-12
Creating a Logical Data Model from Your Relational Model	20-13
Reviewing and Making Changes to Your Logical Model	20-14
Checking Design Rules	20-15
Forward Engineering to a New Relational Model	20-16
Comparing Your Relational Model Changes with What Is in the Database	20-18
Mapping to an Existing Column	20-21
Compare Mapping	20-22
Previewing the DDL	20-23
Comparing and Merging Two Models	20-24
Exporting Your Model	20-28
Exporting to a Data Modeling Design	20-29
Producing Data Modeling Metadata Reports	20-30
Steps to Produce Data Modeler Reports	20-31
Creating a SYSTEM Connection	20-32
Creating a New User for Reporting	20-33
Creating a Connection for the New Reporting User	20-34
Exporting Your Model to the Reporting Schema	20-35
Running Data Modeler Reports	20-37
Quiz	20-41
Summary	20-42
Practice 20-1 Overview: Re-Engineer the HR Schema	20-43

21 Creating a Multidimensional Model

Objectives	21-2
What Is a Multidimensional Model?	21-3
Measures	21-4
Measure Types	21-5
Dimensions	21-6
Sharing Dimensions	21-7
Hierarchy	21-8
Hierarchy: Example	21-10
Level	21-11
Types of Hierarchy	21-12
Attributes	21-13
Dimensional Model Summarized	21-14
Quiz	21-15

Steps to Build a Multidimensional Model in Oracle SQL Developer Data Modeler	21-17
Importing a Database with Dimensions	21-18
Reverse Engineering Your Model	21-21
Creating Your Multidimensional Model	21-23
Reviewing Your Multidimensional Model	21-24
Reviewing Multidimensional Object Properties	21-25
Modifying Properties for the Time Dimension	21-26
Reviewing Properties of Multidimensional Object Components	21-27
Reviewing Detailed Properties of Object Components	21-28
Creating New Multidimensional Objects	21-29
Impact Analysis	21-30
Creating an Oracle AW	21-31
Exporting the Multidimensional Model	21-32
Upgrading Your Oracle AW by Using AWM 11g	21-33
Summary	21-34
Practice 21-1 Overview: Build a Multidimensional Model	21-35

VIII Additional Information

Course Overview

ORACLE

Copyright © 2010, 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Course Objectives

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

- 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Course Objectives (continued)

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: Setting the Stage

- 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Agenda: Day 1

Day 1 provides an overview of what types of models there are. 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 examine 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Agenda: Day 2

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 with 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Agenda: Day 3

The beginning of day 3 is devoted to completing the model by adding data types to your attributes. You will 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 has been 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: Creating a Multidimensional Model



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

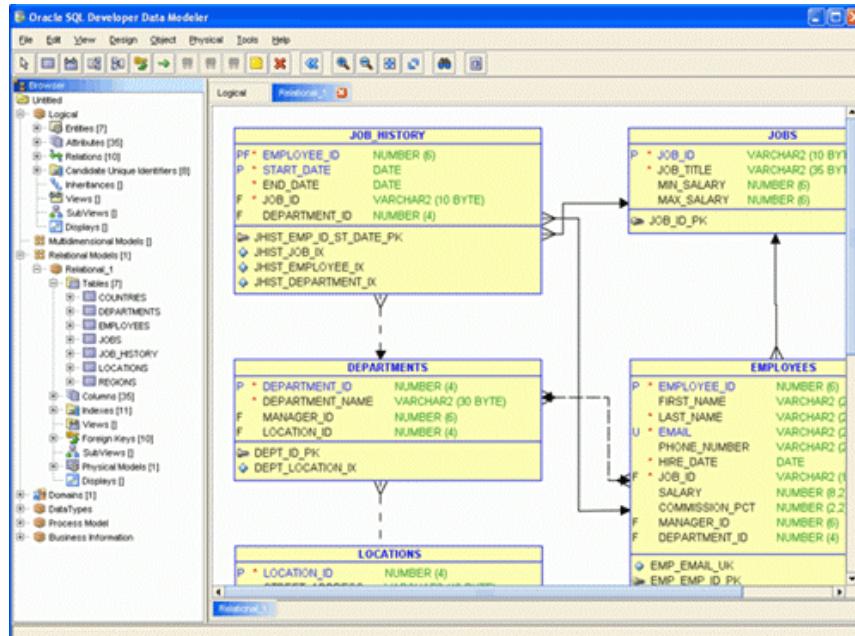
Agenda: Day 4

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 topics that you discuss and perform are altering an existing design and creating a multidimensional model 7

Oracle SQL Developer Data Modeler

The tool used throughout this course to document your models



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Oracle SQL Developer Data Modeler

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.

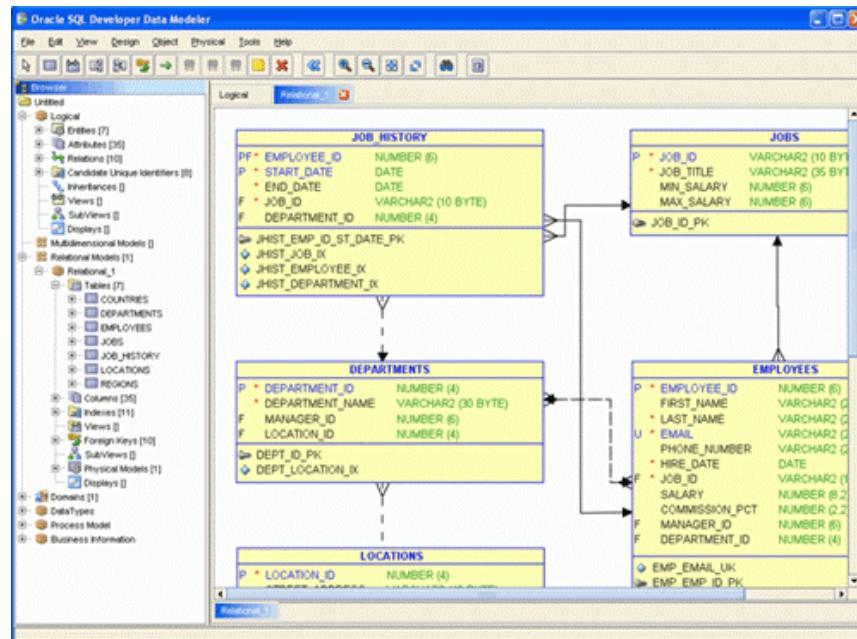
SQL Developer Data Modeler follows the Zachman Framework for defining data structures and is designed for data modelers, from business architects to DBAs and from database to application developers. The role of SQL Developer Data Modeler is to simplify data modeling development tasks and serves as a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



ORACLE

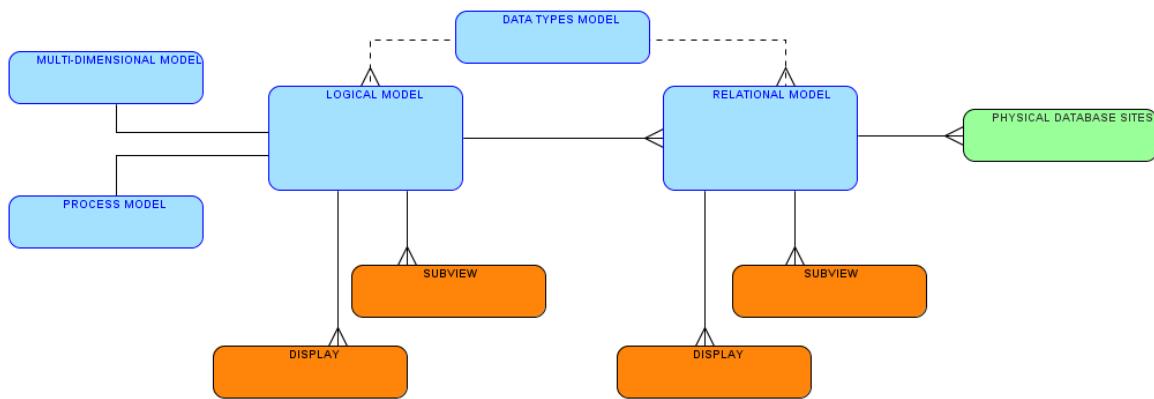
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Oracle SQL Developer Data Modeler Viewer

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

Oracle SQL Developer Data Modeler

Modeling Structure



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Oracle SQL Developer Data Modeler

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 to 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 Oracle SQL Developer Data Model to document the data flow diagram. In addition, you can build a multidimensional model to document the dimensions, measures, hierarchies, and so on.

I

Setting the Stage

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Overview

In this unit, you examine the following areas:

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Overview

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

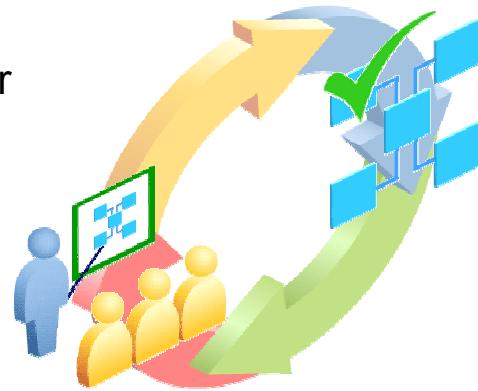
Objectives

In this lesson, you examine the reasons for modeling and 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Why Model?

Remember the saying “A picture is worth a thousand words.” Pictures help humans 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. Models are easy to change because they are 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 up front, 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.

Why Model: A Practical Example

Building a House

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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

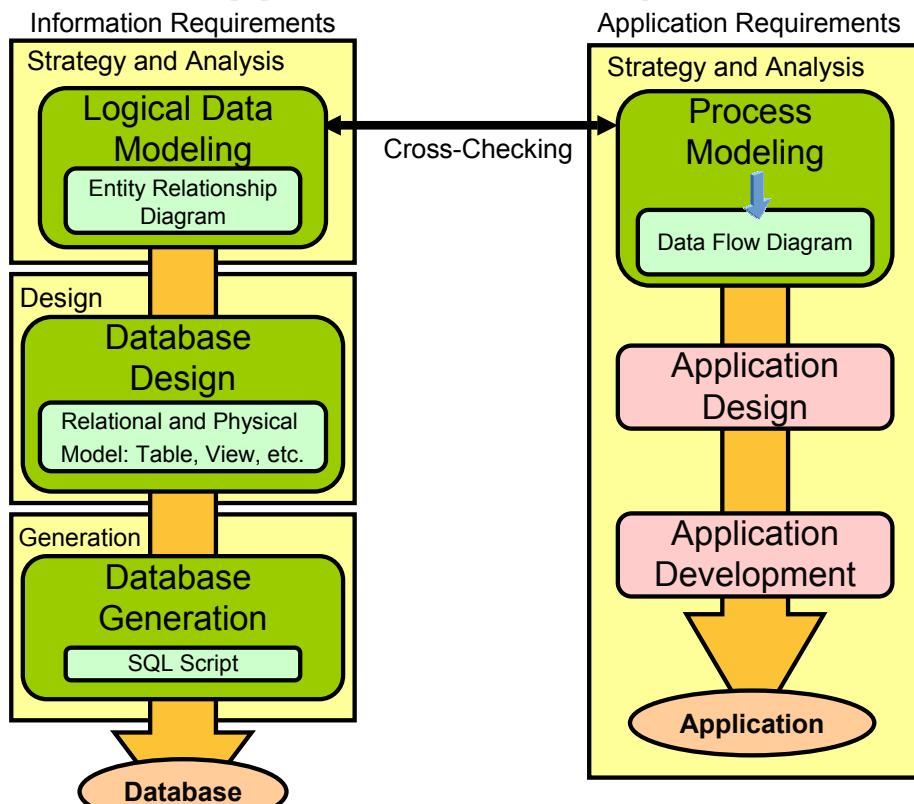
Why Model: A Practical Example

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 home owners as ideas, or as pieces of various dreams. Sometimes the future home owners 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 set of blueprints 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 on the blueprint. In order for the blueprint to be completed, the architect works with the future home owner to take the ideas and desires and build a blueprint 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 that allows for discussion and analysis, giving advice, describing sensible options, documenting it, and confirming it with the future home owners. This diagram provides the future home owners with a plan of the home they want.

Database and Application Development Life Cycle



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Database and Application Development Life Cycle

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

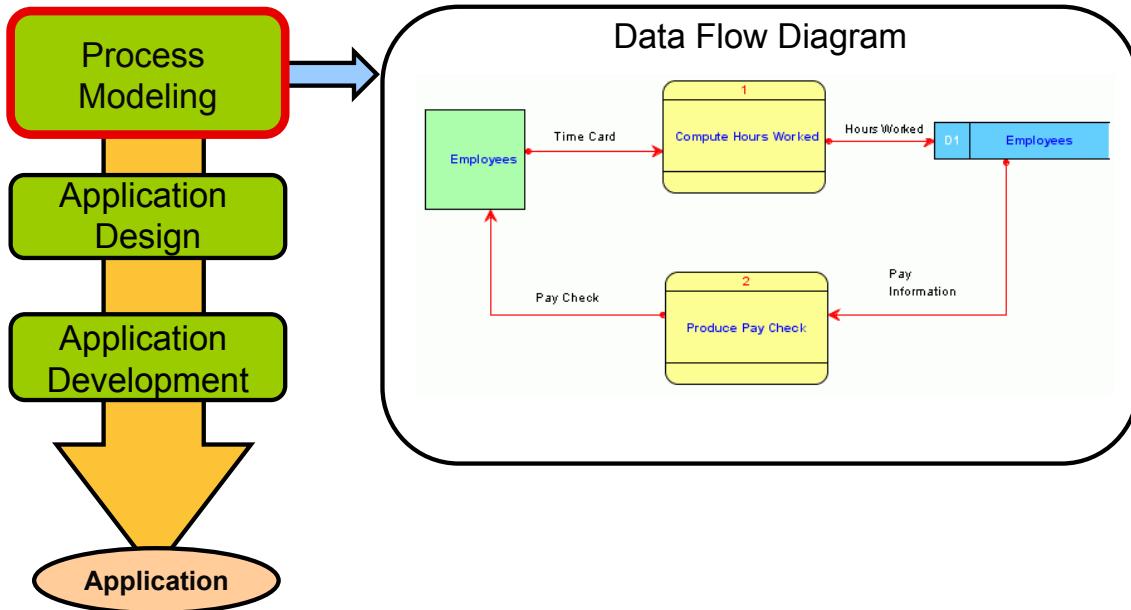
In this course, you examine how to build a logical data model in the Strategy and Analysis phase of the life cycle using the entity relationship diagram (ERD). In order to build the logical data model, it is important to understand what processes use and produce the data. This is done through the process 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 will 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Process Modeling

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 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:

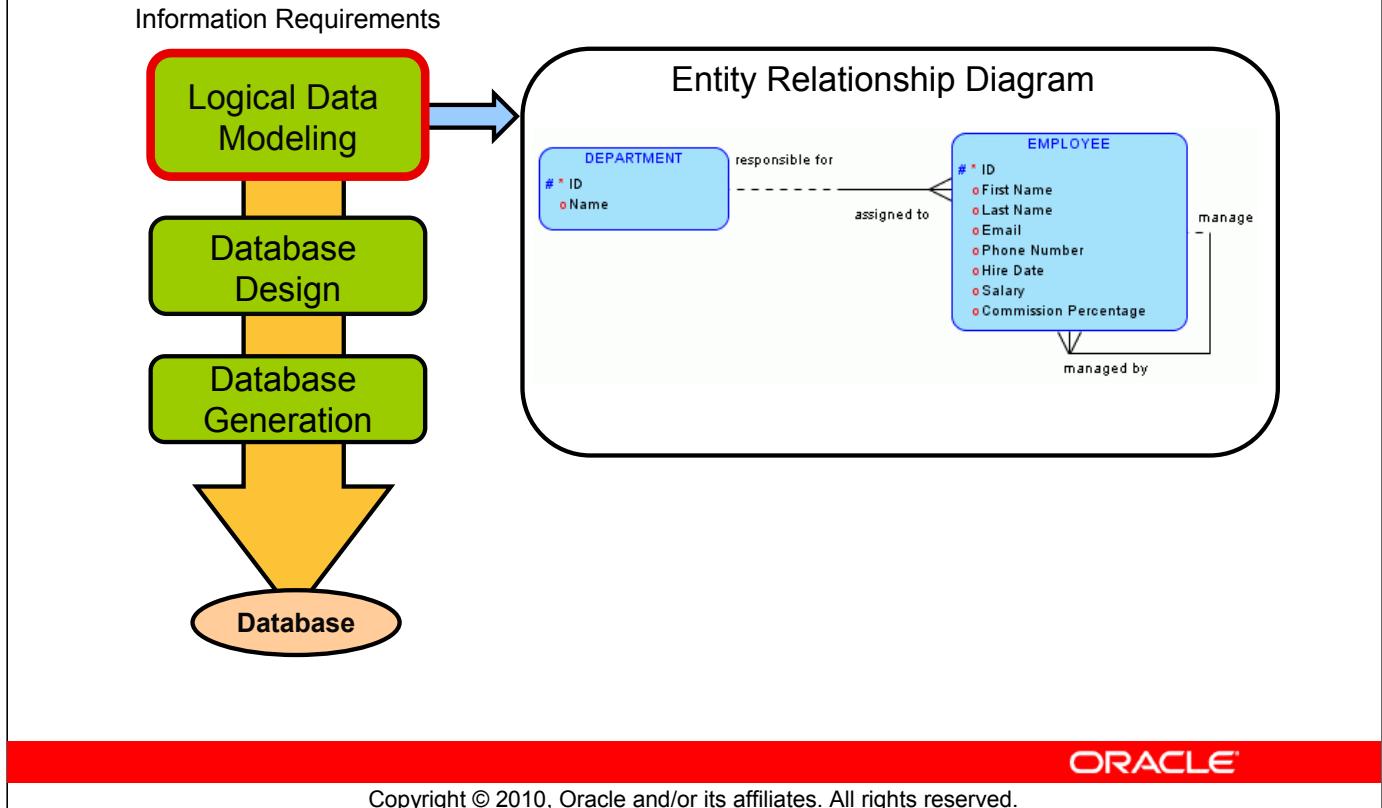
- **Process:** represents a discrete piece of work or function
- **External agents:** represents a class of things or people that represent a source or destination of transactions
- **Data flow:** represents the flow of information between objects
- **Information store:** represents data stored between 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 done 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).

Process Modeling (continued)

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 aids in identifying the data that must be stored in the logical data model.

Logical Data Modeling



Logical Data Modeling

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:

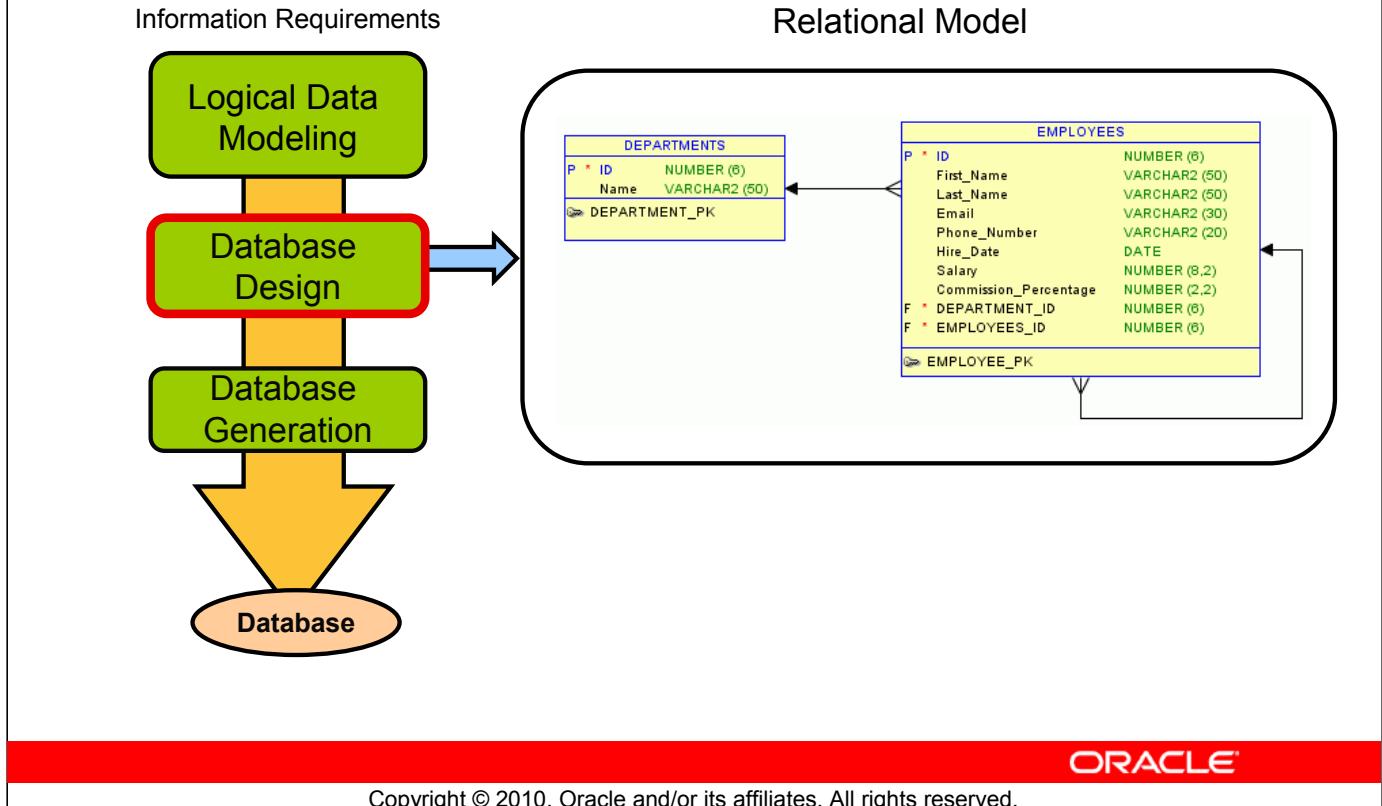
- **Entities:** things of significance about which information must be held
- **Relationships:** how the things of significance are related
- **Attributes:** specific information that must be held

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 employees). The names next to each relationship designate the business rules that you will examine in more detail later in the course.

Logical Data Modeling (continued)

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Database Design

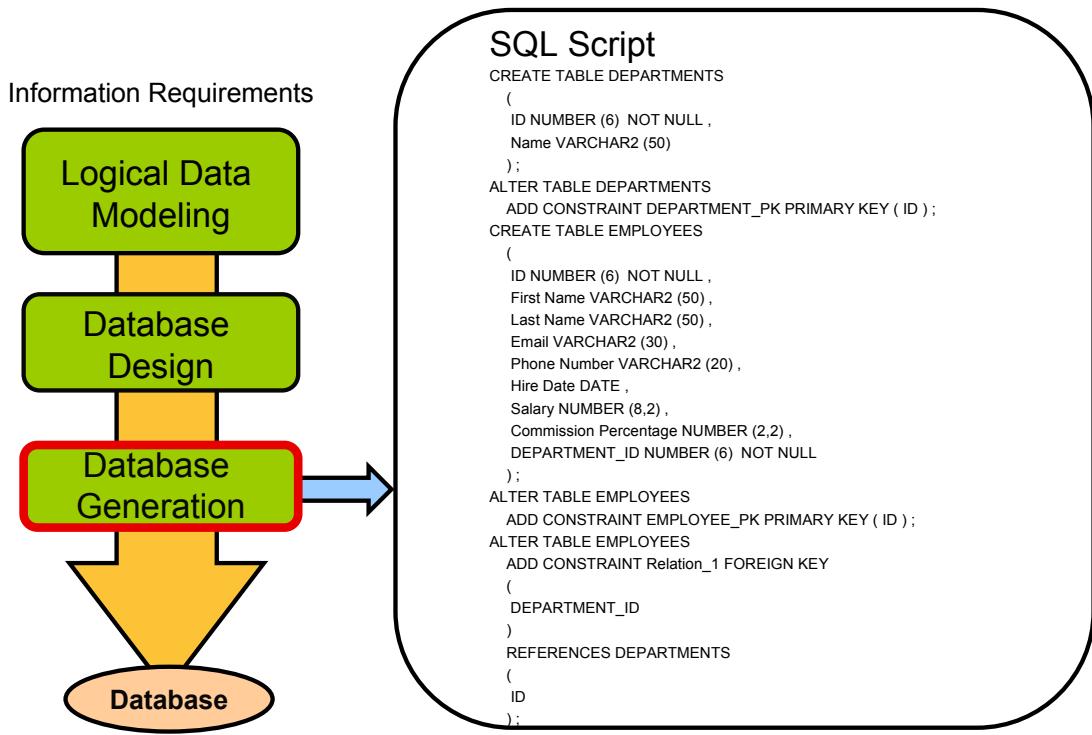
The database design, often called the relational model, is the model that is built during the Design phase of the database development life cycle. The purpose of this model is to describe the database objects that must be created when the database is generated. This model is typically built by a database 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 (and which are discussed in more depth in a later lesson):

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

Database Generation



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

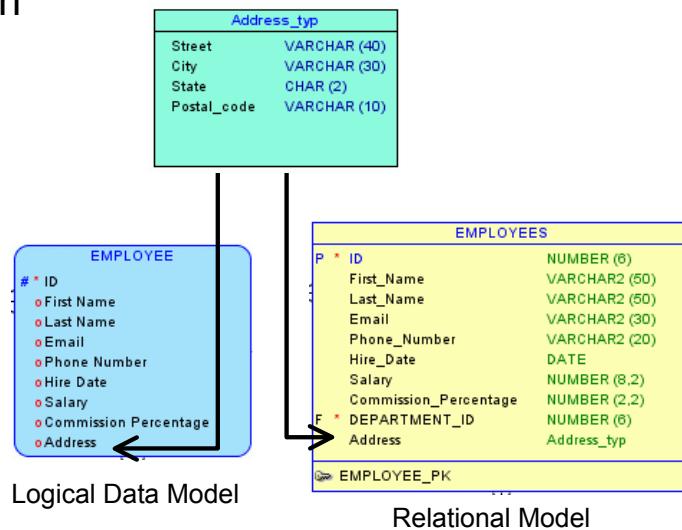
Database Generation

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 the Oracle Database 11g.

Throughout this course you will 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

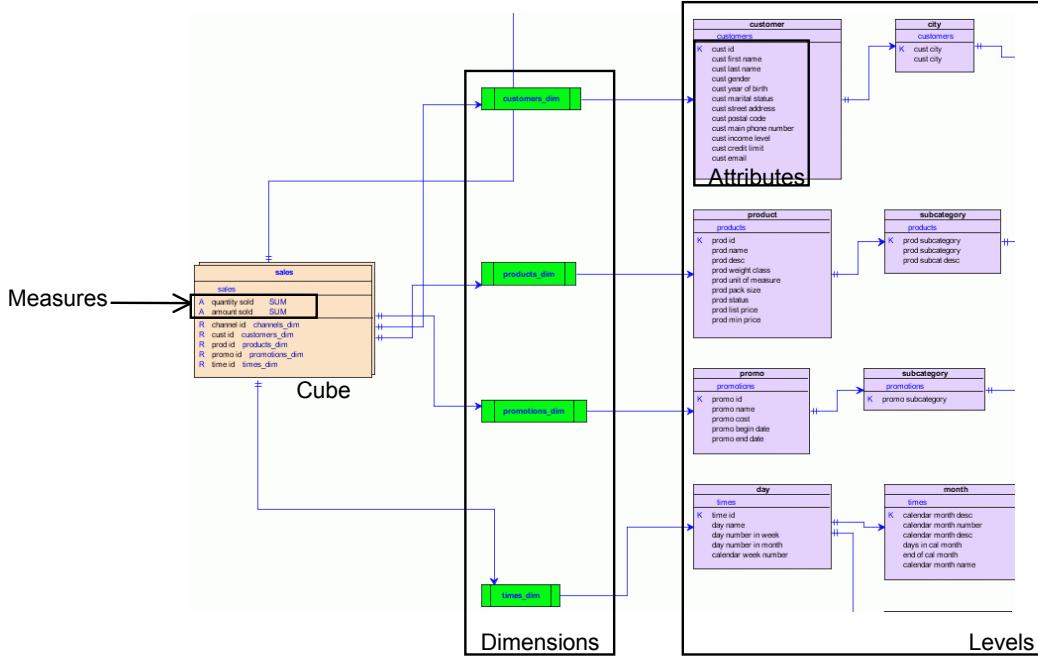
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Data Type Model

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 represents 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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Multidimensional Model

A multidimensional model is a model of business activities in terms of facts and dimensions. The components of a Multidimensional Model include the following:

- **Cubes:** Corresponds to a single fact table or view
- **Measures:** Populates 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) only has meaning 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 affect these trends have on a larger sector of the business.

Multidimensional Model (continued)

- **Level:** Represents a position in the hierarchy. Each level above the base (or 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, thus 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 dimension table or view. Examples of attributes are colors, flavors, or sizes. These attributes can be used for data selection and 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



Copyright © 2010, 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

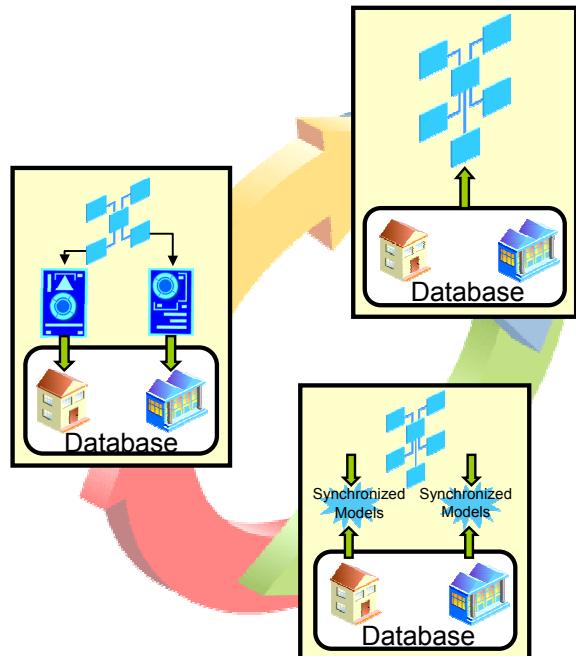


Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: d

Approaches to Modeling

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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

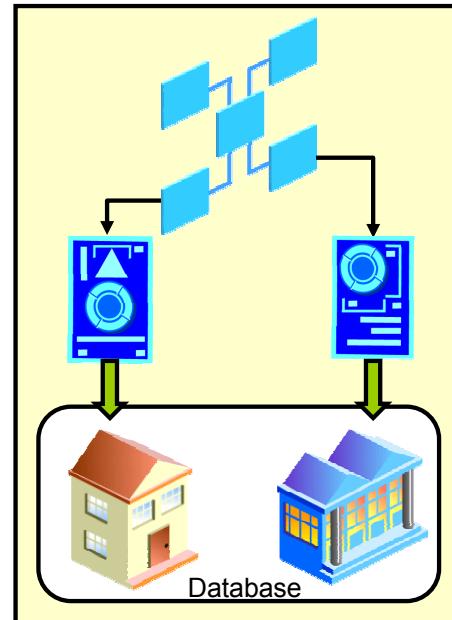
Approaches to Modeling

There are three approaches to modeling: top-down, bottom-up, and targeted. All three approaches meet 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 will examine bottom-up and targeted modeling.

Top-Down 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 © 2010, Oracle and/or its affiliates. All rights reserved.

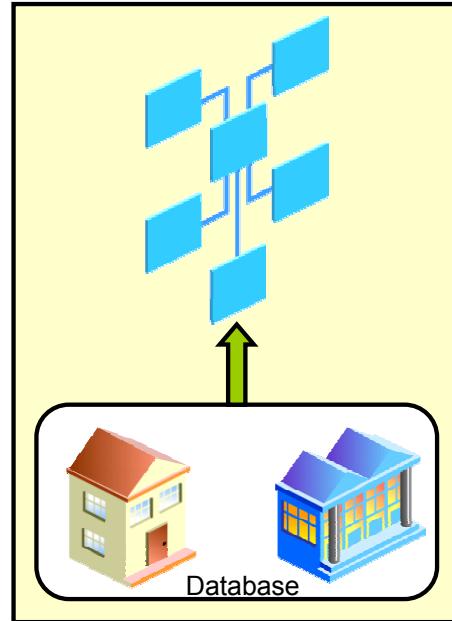
Top-Down Modeling

Top-down modeling is used for designing a new database. Top-down modeling gathers information about business requirements, and proceeds to define processes, a logical model of the data, one or more relational models, and 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 needs.

Bottom-Up Modeling

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 © 2010, Oracle and/or its affiliates. All rights reserved.

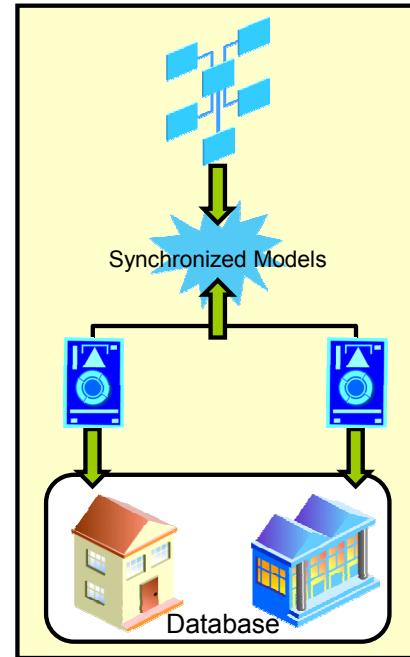
Bottom-Up Modeling

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 needs.

Targeted Modeling

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Targeted Modeling

The targeted modeling approach maintains the existing models by adapting to new requirements. Depending on the requirement, you must determine which model to modify. If the new requirement 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 from XYZ Corporation, has been contacted because there is a new mandate to maintain information about employees' dependents. Which type of modeling approach should Mary use?

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: b, c

It depends. 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 Incorporated would like to review the business rules and possibly make some changes. Which modeling approach should they use?

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: c

Summary

In this lesson, you should have learned how to:

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

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 there are numerous approaches to modeling and when each approach is used.

Practice 1-1 Overview: Identify the Modeling Approach

This practice covers the following topics:

- Identifying which models must be developed
- Identifying which modeling approach to use



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 1-1 Overview: Identify the Modeling Approach

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

Documenting the Business Background

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

In this lesson, you examine the various components to a business direction statement and how to establish business direction objectives.

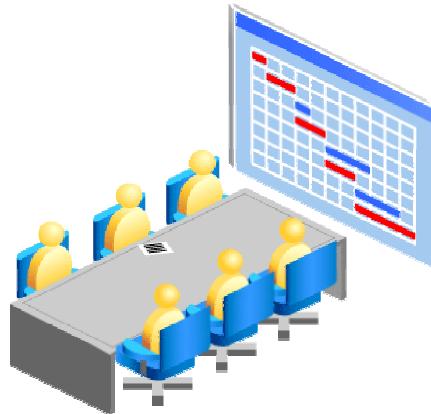
Documenting the Business Direction

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 © 2010, Oracle and/or its affiliates. All rights reserved.

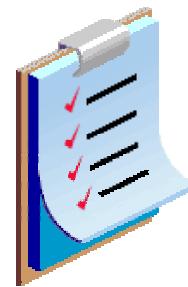
Documenting the Business Direction

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 department functions. 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Components of a Business Direction Statement

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 may be made about a system, specifically those that may 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 towards achieving a business objective
- **Problem:** a business event or state that may inhibit the progress of the enterprise towards its objectives

Business Objectives

- 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Business Objectives

When defining a business objective, you must be able to quantify the objective 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.

Assumptions

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



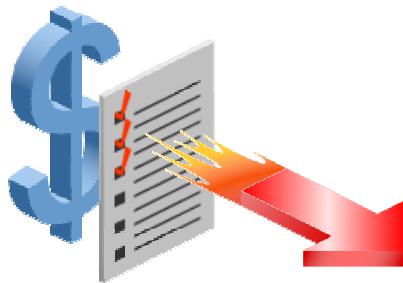
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Assumptions

Assumptions are statements that define the scope of the system. Assumptions may 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 Factors

- 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

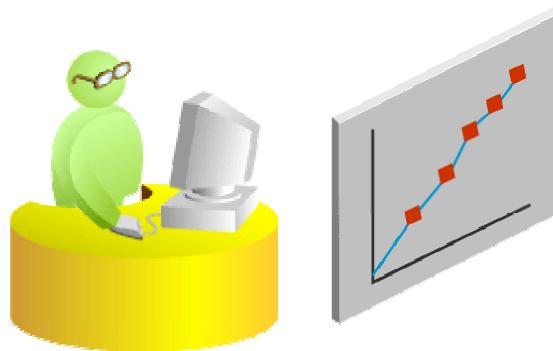
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Critical Success Factors

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 as compared to competitors, the business objective may not be met.

Key Performance Indicators

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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Key Performance Indicators

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 15% per month to achieve the business objective of increasing membership fee revenue by 30%.

Problems

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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Problems

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 what actions must take place to achieve the desired result.

Business Direction Objective	Actions 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 upgrading one's membership.
Reduce overdue DVDs by 3%.	Notify the customer immediately when a DVD is overdue.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Devising Business Direction Objectives and Actions

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 will specify these objectives as documents for each object in your logical data models (ERDs) and process models (DFDs) within Oracle SQL Developer Data Modeler.

Quiz

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

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: b

Quiz

Employee retention levels should be > 75% annually. Which of the following business direction components does this represent?

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



Copyright © 2010, 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

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

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

This practice covers the following topics:

- Reviewing a business direction
- Identifying the type of information it is



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

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

In this practice, you review a series of business direction statements and decide what type it is: business objective, assumption, key performance indicator, critical success factor, or problem.

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

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Overview

In this unit, you examine the following areas:

- 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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Overview

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)

ORACLE

Copyright © 2010, 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



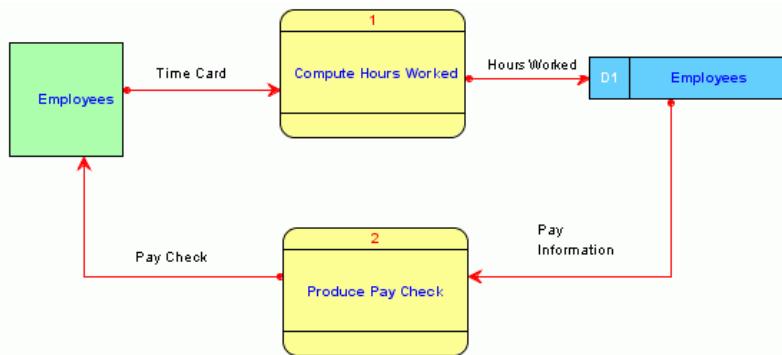
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

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 © 2010, Oracle and/or its affiliates. All rights reserved.

What Is a Process Model?

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 or DFD.

Why Create a DFD?

Data flow diagrams help to:

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



ORACLE

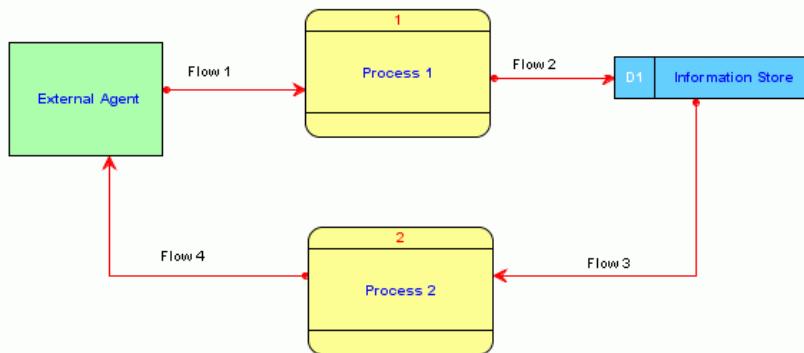
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Why Create a DFD?

Often data flow diagrams are overlooked, but they are useful in many ways. DFDs help to document the boundaries of a project so that you know exactly who must be involved and for what purpose. DFDs help everyone involved understand the current area of interest independent of the implementation. DFDs often uncover inefficiencies in the current way of doing things, allowing improvements to be made. DFDs allow end users to communicate complex ideas and processes in a way that everyone can understand. DFDs can also 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 © 2010, Oracle and/or its affiliates. All rights reserved.

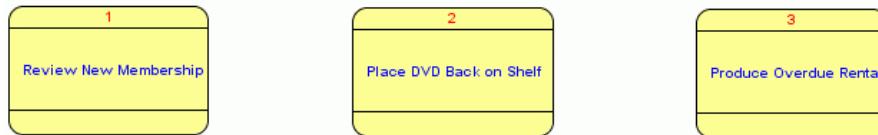
Components of a Data Flow Diagram

A data flow diagram has four main components, as follows:

Component	Purpose
Process	Is a discrete piece of work or function
External agents	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 © 2010, Oracle and/or its affiliates. All rights reserved.

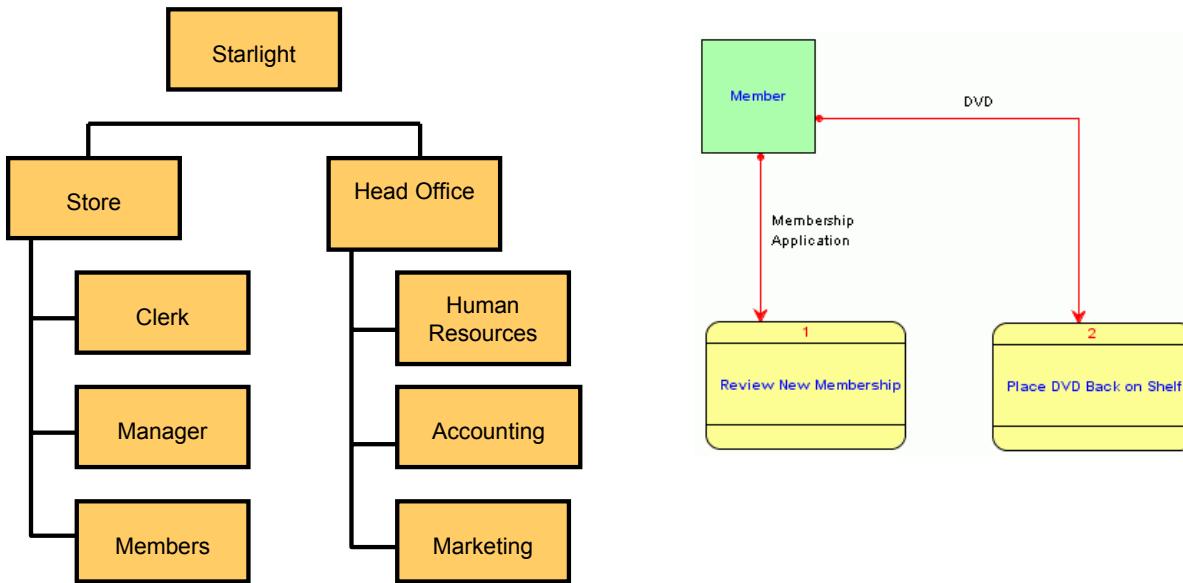
Process

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

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

External Agents

An external agent represents an organizational unit, department, job, role, or actor.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

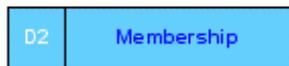
External Agents

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 roles within the 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 Stores

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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Information Stores

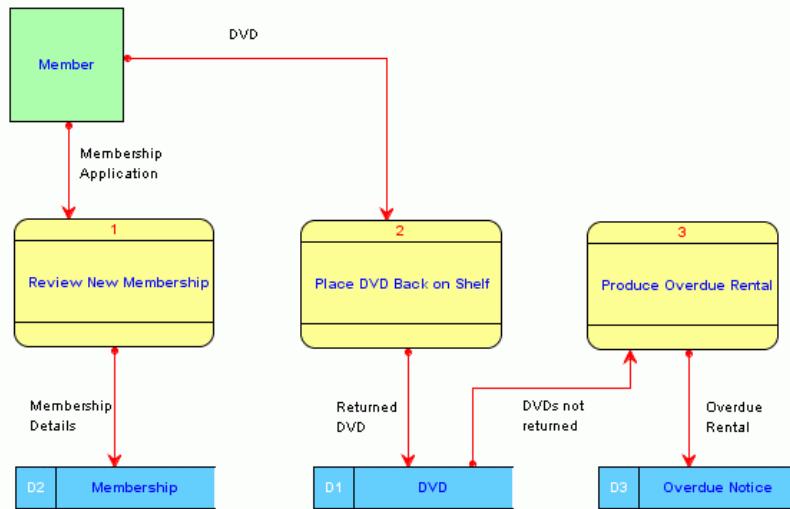
An information store represents “data at rest” or a collection of information or materials. For example, an information store can represent a warehouse containing goods, a filing cabinet containing copies of customers’ proof of identity, or a database containing records of rental transactions.

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, only add information stores to a DFD when they aid in the understanding of the process flow.

Note that external agents cannot supply information stores with information directly. A process is needed to extract information from an information store or an external agent and then insert or update the same or different information store or external agent.

Information Flows

- Information flows show the flow of information into and out of a process.
- Examples:



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Information Flows

Information flows show how the information flows into and out of a process. An information flow has to have one of its ends connected to a process. An information flow cannot be used between two information stores, 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



Copyright © 2010, 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



Copyright © 2010, 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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: b, c

The answer could be a or c.

Quiz

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

- a. External agent
- b. Information store
- c. Information flow
- d. Process



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: a

Events

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



External (Other)



Internal (System)



Temporal (Time)

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Events

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 requesting 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 help to understand day-to-day workings of a particular business area.

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Analyzing Event Responses

Understanding the day-to-day workings of a particular business area requires identifying responses to events. Users are encouraged to identify events that occur on a daily basis and how they respond to the event when it is presented. Providing examples as shown in the slide encourages ideas. This analysis is used to examine the current system and define the new functional requirements.

Events are represented in the Processes dialog box in the Events property.

Quiz

Which of the following is an event?

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



Copyright © 2010, 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.



Copyright © 2010, 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 an instructor to teach the course based on their schedule and the courses they teach. 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 that teach each course."



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Class Practice: Create a Data Flow Diagram

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 3-1 Overview: Create a Data Flow Diagram

Read the scenario and build a data flow diagram.

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



ORACLE®

Copyright © 2010, 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 © 2010, Oracle and/or its affiliates. All rights reserved.

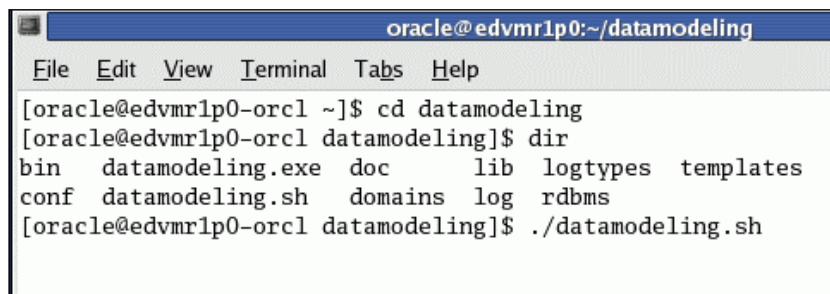
Objectives

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

Oracle SQL Developer Data Modeler

- Download from OTN and unzip to a local directory.
- On Linux, open a terminal window and execute the following commands:

```
cd datamodeling  
./datamodeling.sh
```



A screenshot of a terminal window titled 'oracle@edvmr1p0:~/datamodeling'. The window shows the following command sequence:

```
oracle@edvmr1p0:~/datamodeling  
File Edit View Terminal Tabs Help  
[oracle@edvmr1p0-orcl ~]$ cd datamodeling  
[oracle@edvmr1p0-orcl datamodeling]$ dir  
bin datamodeling.exe doc lib logtypes templates  
conf datamodeling.sh domains log rdbms  
[oracle@edvmr1p0-orcl datamodeling]$ ./datamodeling.sh
```

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Oracle SQL Developer Data Modeler

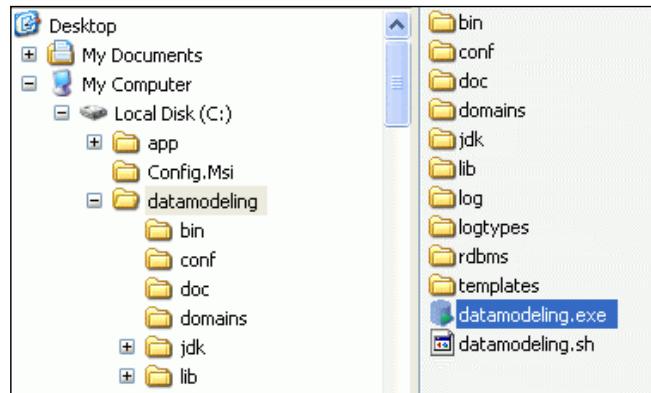
Oracle SQL Developer Data Modeler is very easy to install and execute.

To install, go to <http://otn.oracle.com/sqldeveloper> 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.

Oracle SQL Developer Data Modeler

On Windows, in Windows Explorer, navigate to the datamodeling directory and double-click datamodeling.exe.



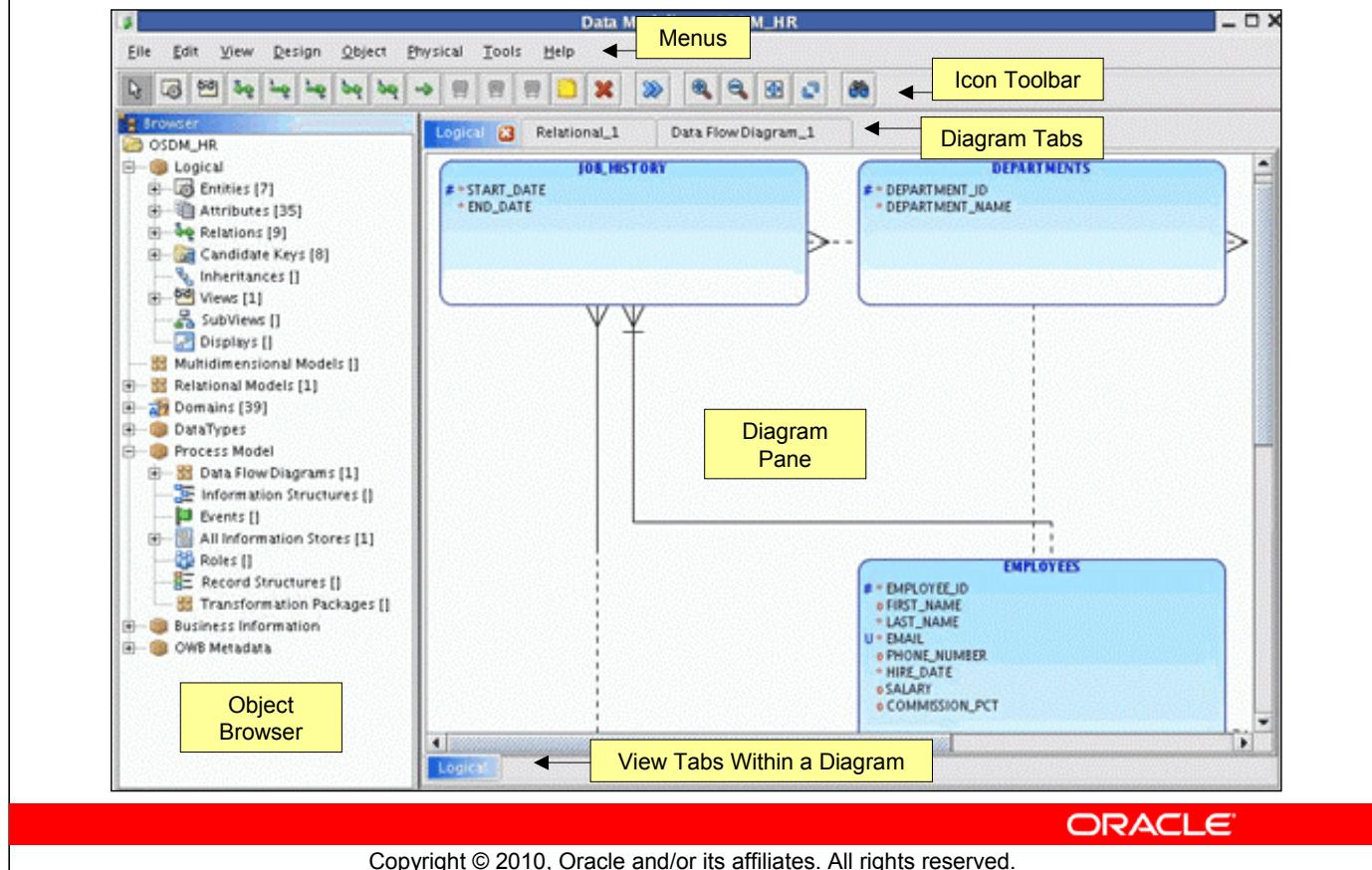
ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Oracle SQL Developer Data Modeler (continued)

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

Oracle SQL Developer Data Modeler Main Window



Oracle SQL Developer Data Modeler Main Window

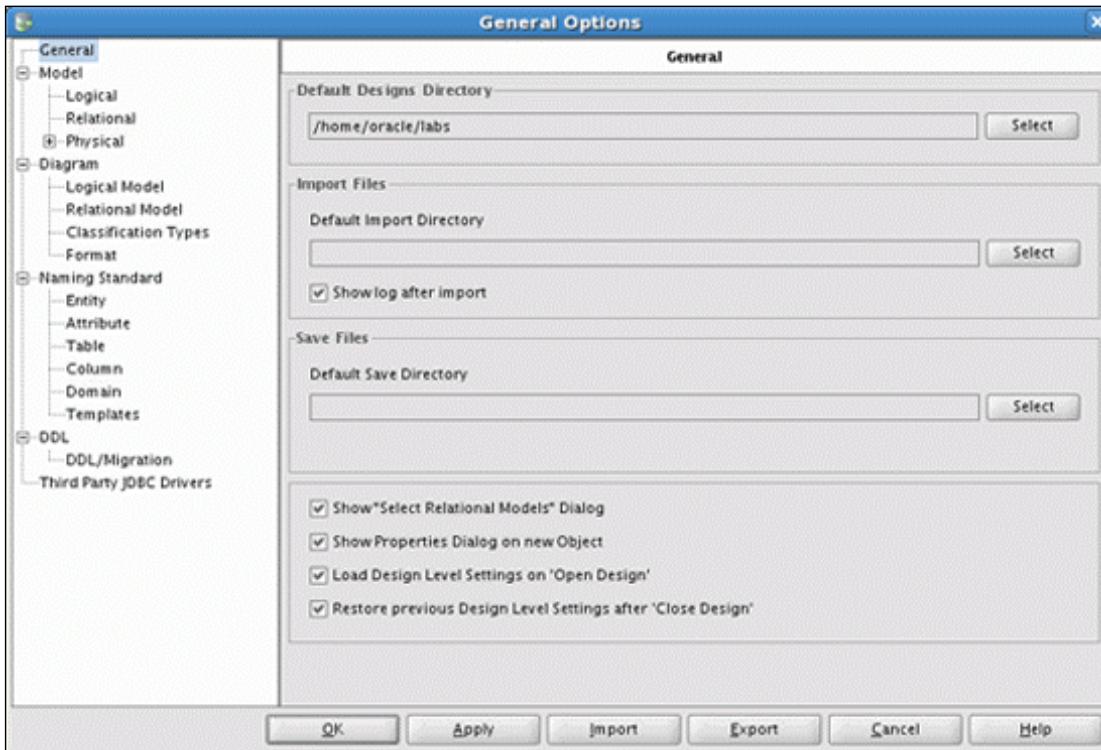
Generally, the left side of Oracle SQL Developer Data Modeler main window is used for navigation to find and select objects, and the right side to view information about selected objects.

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 followed by Alt+C opens Help and then Contents.

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, and Generate DDL. 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).

Specifying General Options: General



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

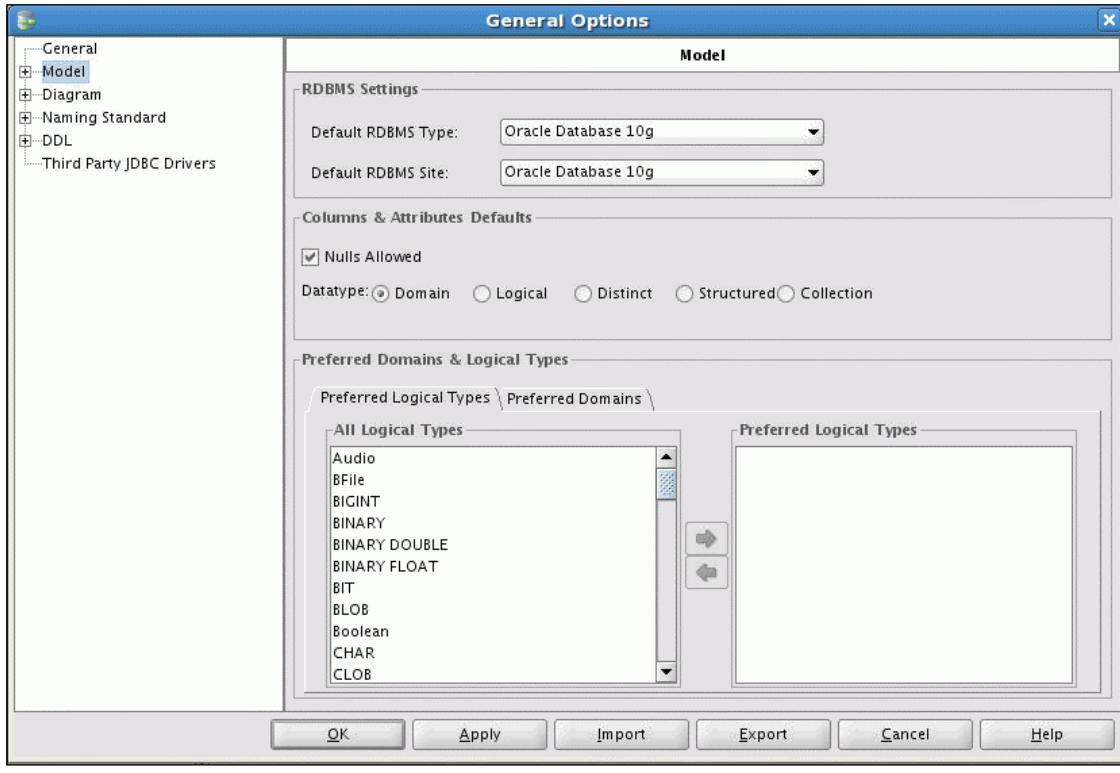
Specifying General Options: General

Within Oracle SQL Developer Data Modeler, you can set various properties. To open the General Options window, select Tools > General Options.

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

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

Specifying General Options: Model



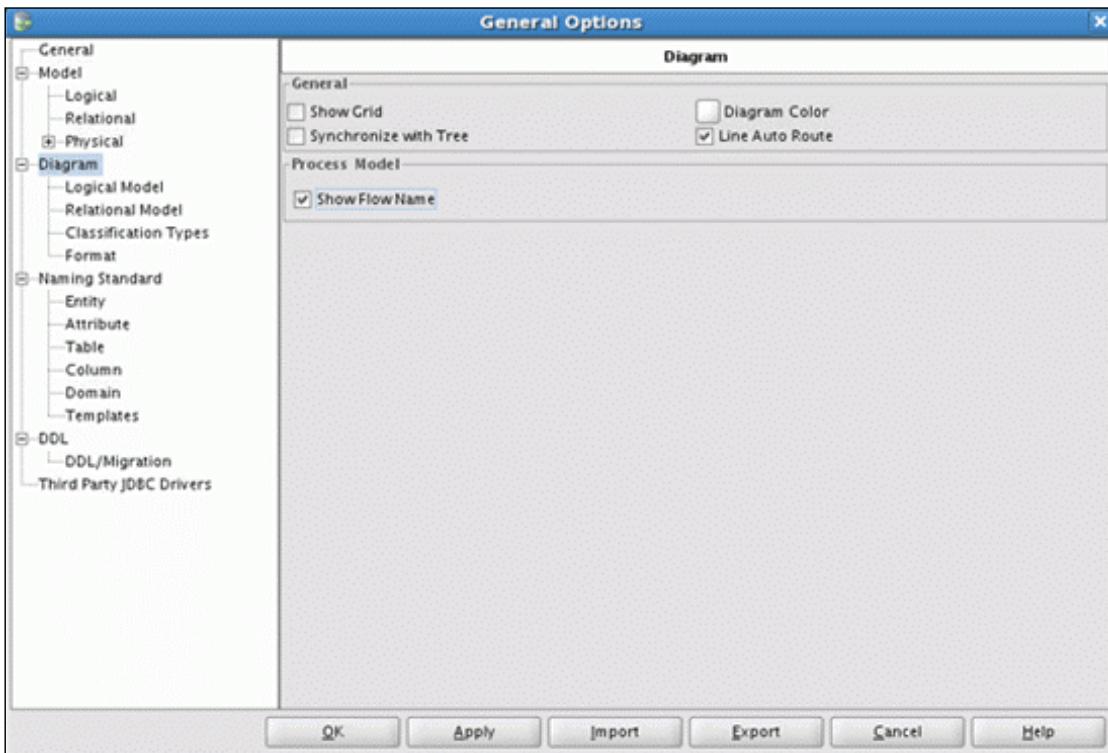
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Specifying General Options: Model

The Model pane contains options that apply to several types of models.

- **Default RDBMS Type:** Default database type
- **Default RDBMS Site:** Default site within the default database type
- **Columns and Attributes Defaults: Nulls Allowed:** Controls whether new columns and attributes are allowed to have null values. If this option is disabled, new columns and attributes are by default mandatory (a value is required). You can also specify the default data type when creating model objects.

Specifying General Options: Diagram



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Specifying General Options: Diagram

The Diagram pane contains general options that affect the appearance of model diagrams.

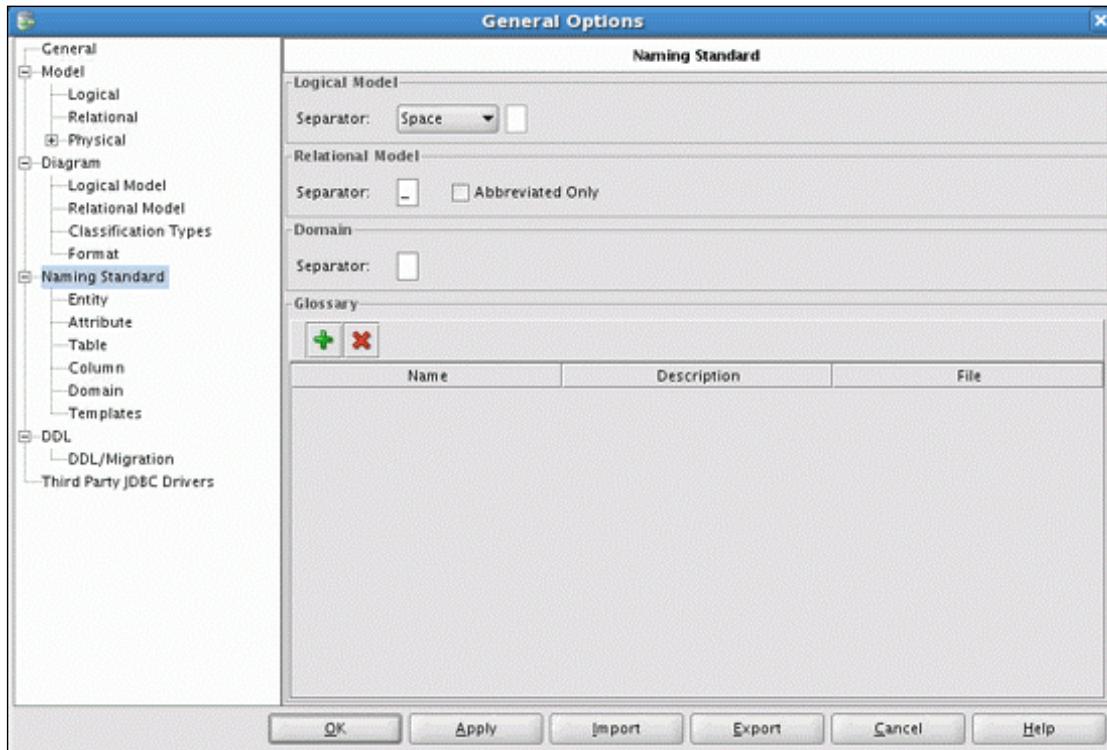
General:

- Show Grid:** Controls whether a grid is displayed in the background in diagrams. Seeing the grid can help you to align objects vertically and horizontally in the diagram.
- Synchronize with Tree:** Controls whether the focus in an active diagram is automatically moved to reflect the selection of objects under that model in the object browser.
- Diagram Color:** Displays a dialog box for selecting the color scheme for the background in diagrams
- Line Auto Route:** Controls whether lines representing relations, foreign key relations, inheritances, flows, and other relationships are automatically drawn in diagrams. If you deselect this option, you determine how these lines are drawn. For example, you may want to add or move break points manually, in order to enhance the clarity of your models.

Process Model:

- Show Flow Name:** Controls whether the flow name is displayed in a box on each flow line in data flow diagrams

Specifying General Options: Naming Standard



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

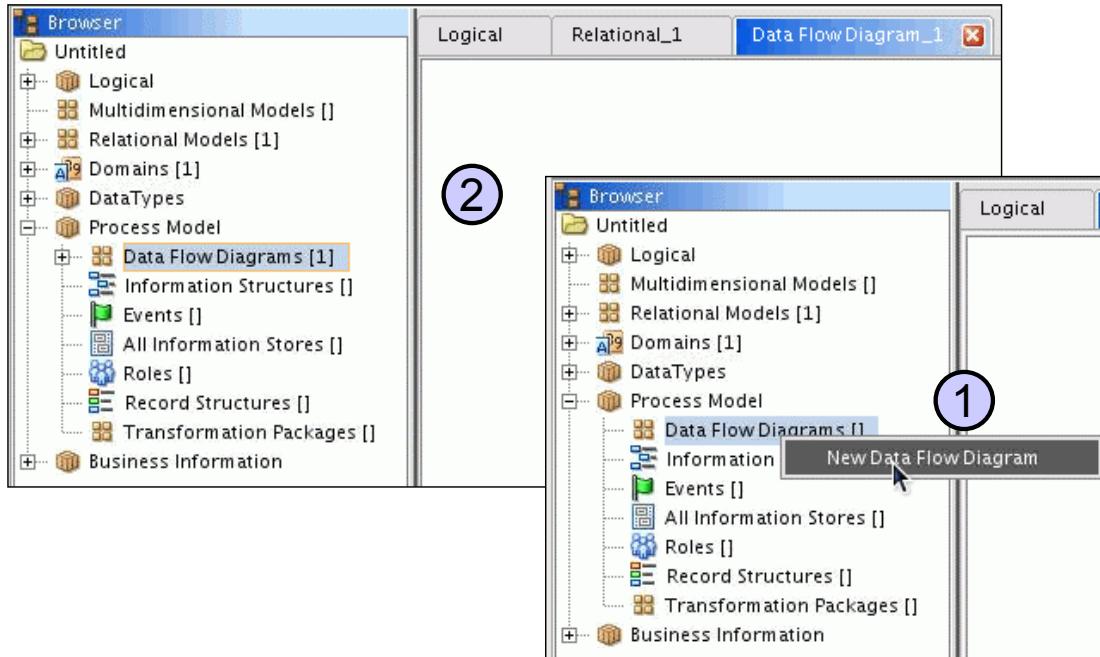
Specifying General Options: Naming Standard

The Naming Standard pane allows you to implement naming standardization. Here, you can view, add, and modify naming standards for logical and relational model objects and for domains. These standards will be checked when you apply Design Rules (covered later in the course), and any violations of the standards will be reported as errors or warnings.

Do not confuse naming standardization with using the Name Abbreviations dialog box, which makes immediate name changes to enforce consistency in spellings and abbreviations, and which is limited to relational model name strings.

Building a Data Flow Diagram

1. Create a data flow diagram.

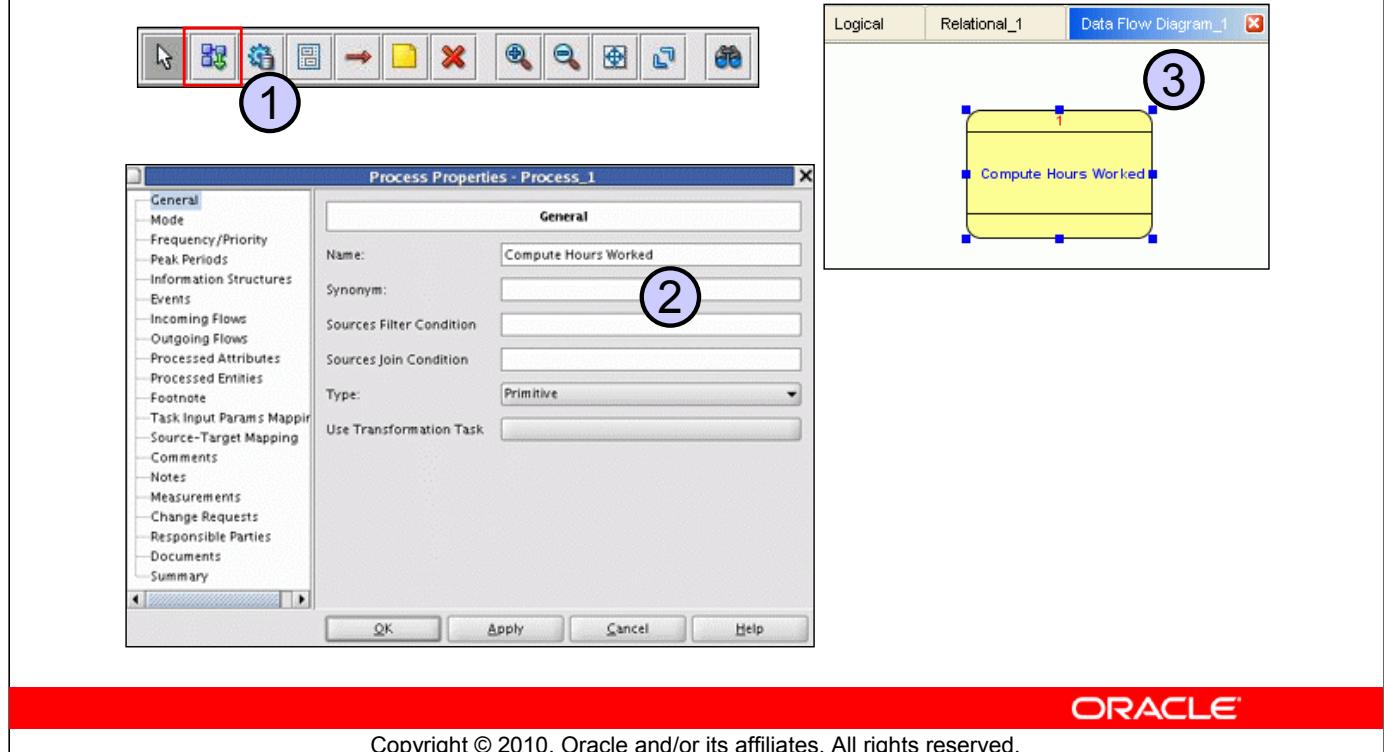


Building a Data Flow Diagram

The first step to 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.



Building a Data Flow Diagram (continued)

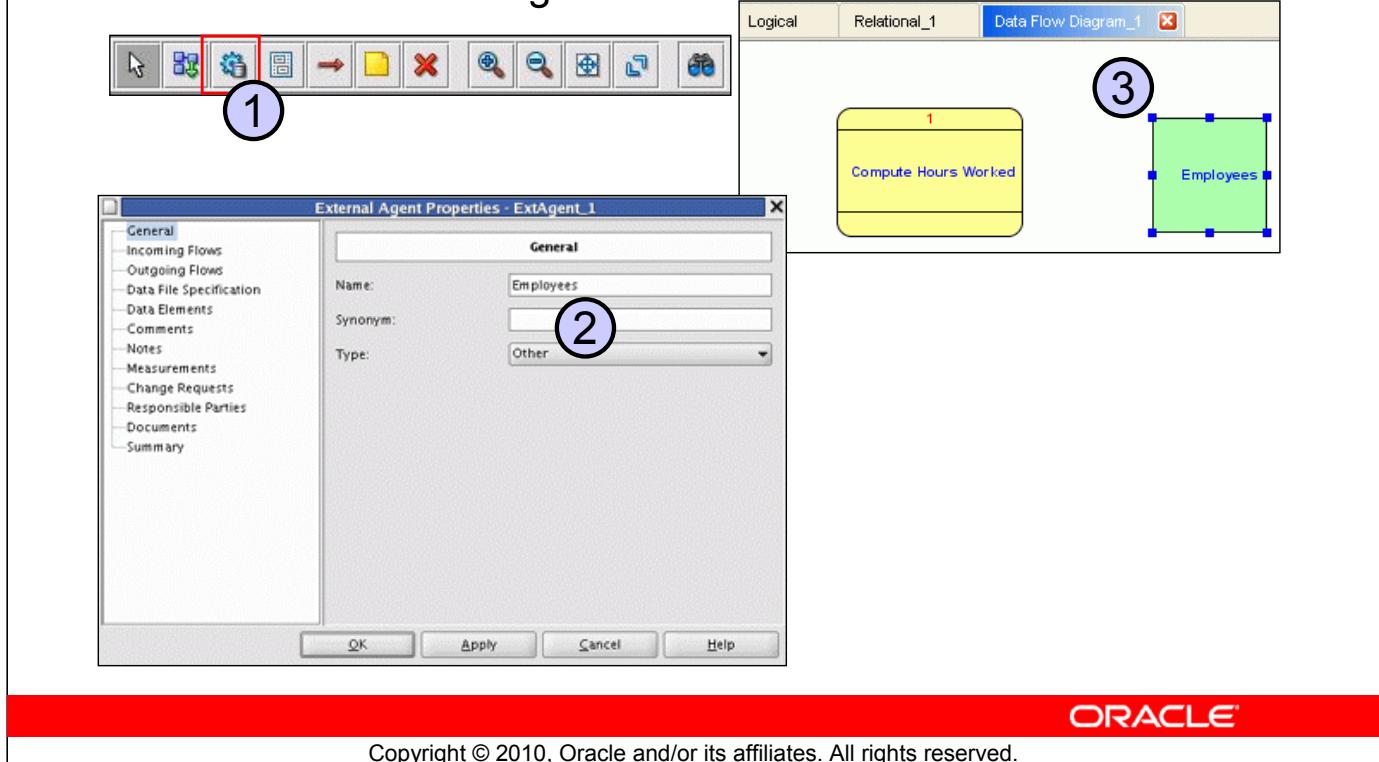
The next step to building a Data Flow Diagram is to create a process.

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 re-open the Process Properties window.

Building a Data Flow Diagram

3. Create an external agent.



Building a Data Flow Diagram (continued)

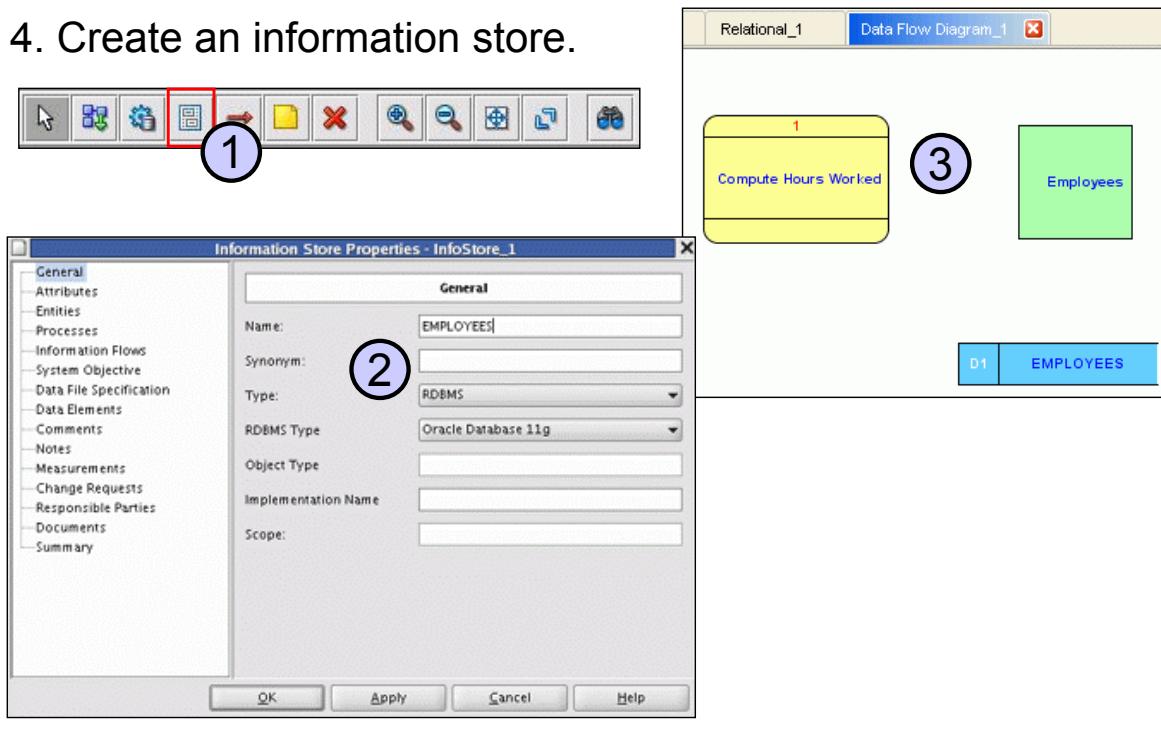
The next step to building a data flow diagram is to create an external agent.

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 re-open the External Agent Properties window.

Building a Data Flow Diagram

4. Create an information store.



Building a Data Flow Diagram (continued)

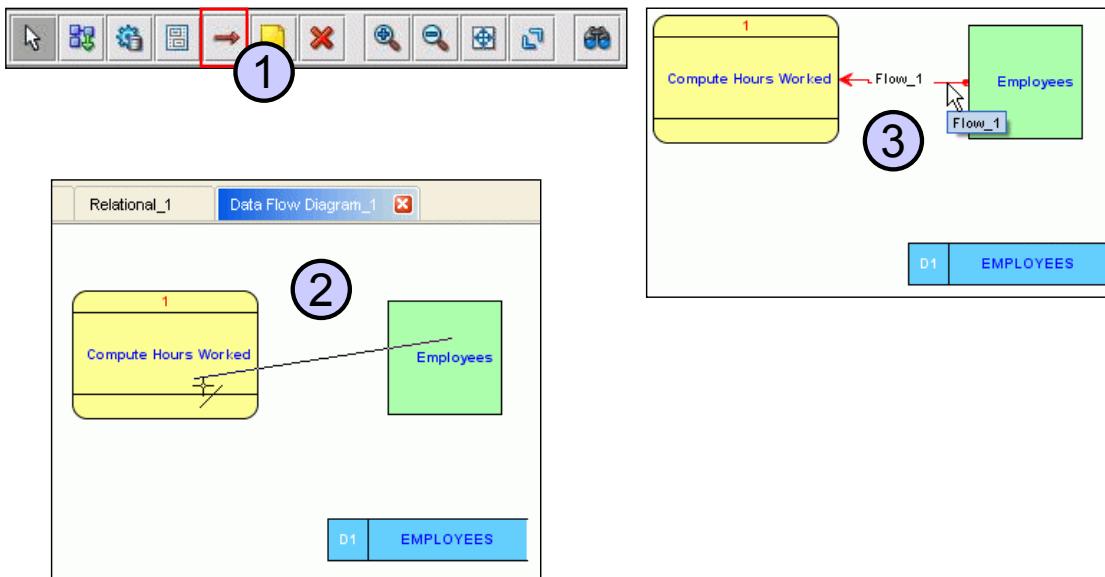
The next step to building a data flow diagram is to create an information store.

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 re-open the Information Store Properties window.

Building a Data Flow Diagram

5. Create an information flow.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

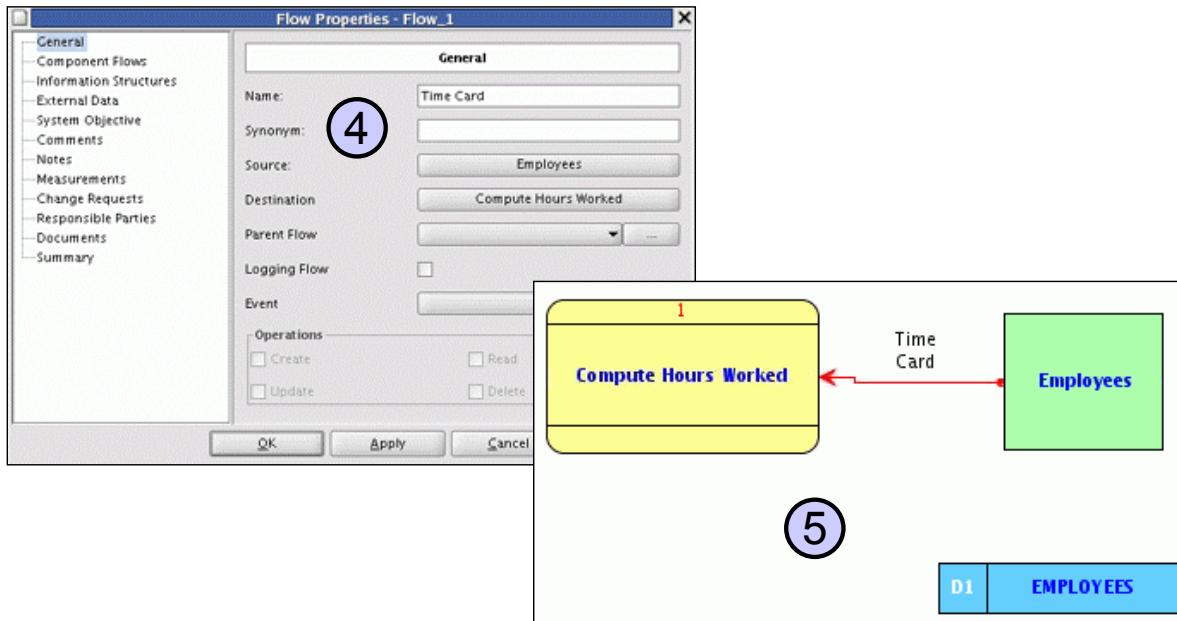
Building a Data Flow Diagram (continued)

The next step to building a data flow diagram 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, double-click the flow line to open the Properties dialog box.

Building a Data Flow Diagram

5. Create an information flow.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

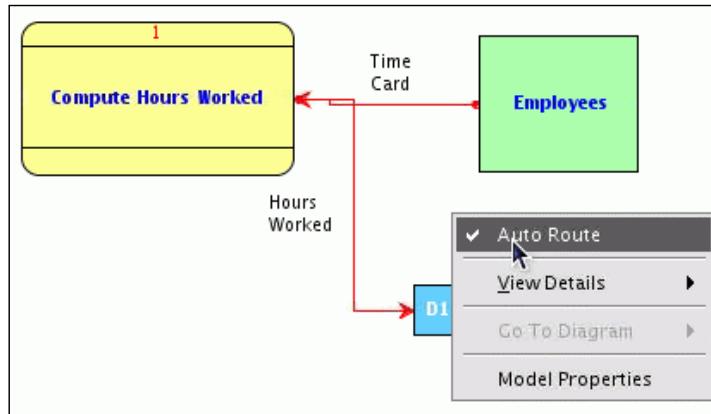
Building a Data Flow Diagram (continued)

Creating a flow between two objects (continued):

4. In the Properties dialog box, change the name of the flow and then click OK.
5. The name of the flow has changed. You can expand the size of the label and move the box.

Editing the Diagram Layout

Turn off Auto Route.



ORACLE

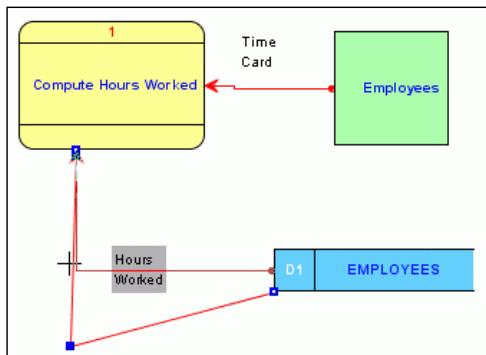
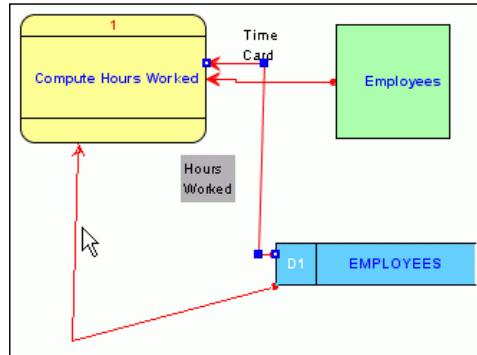
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Editing the Diagram Layout

In order 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 and select Auto Route.

Editing the Diagram Layout

Ctrl-drag the line to a different location.



Drag a blue corner box to a different location.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Editing the Diagram Layout (continued)

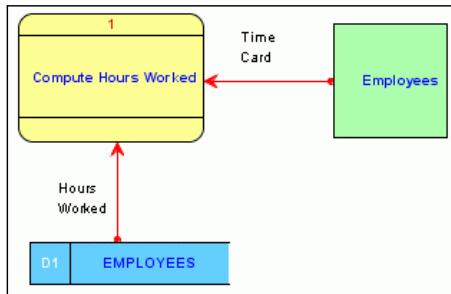
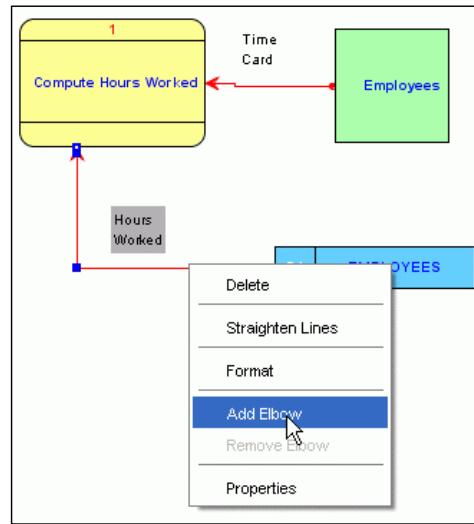
There are many ways to redraw lines:

- Ctrl-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.



Move an object.



ORACLE

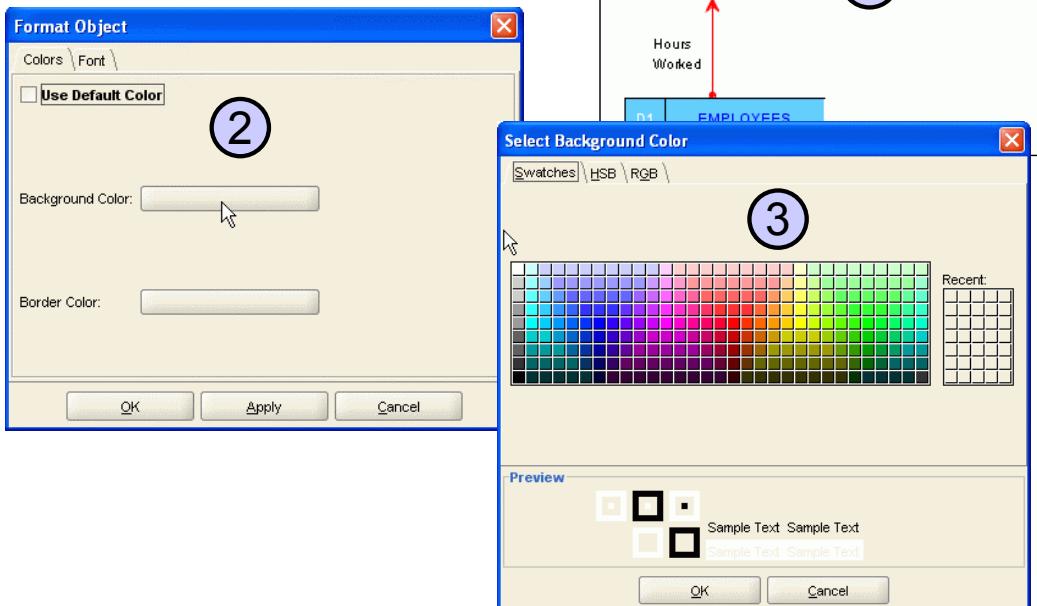
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Editing the Diagram Layout (continued)

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

Editing the Diagram Layout

Change the format of the flow label.



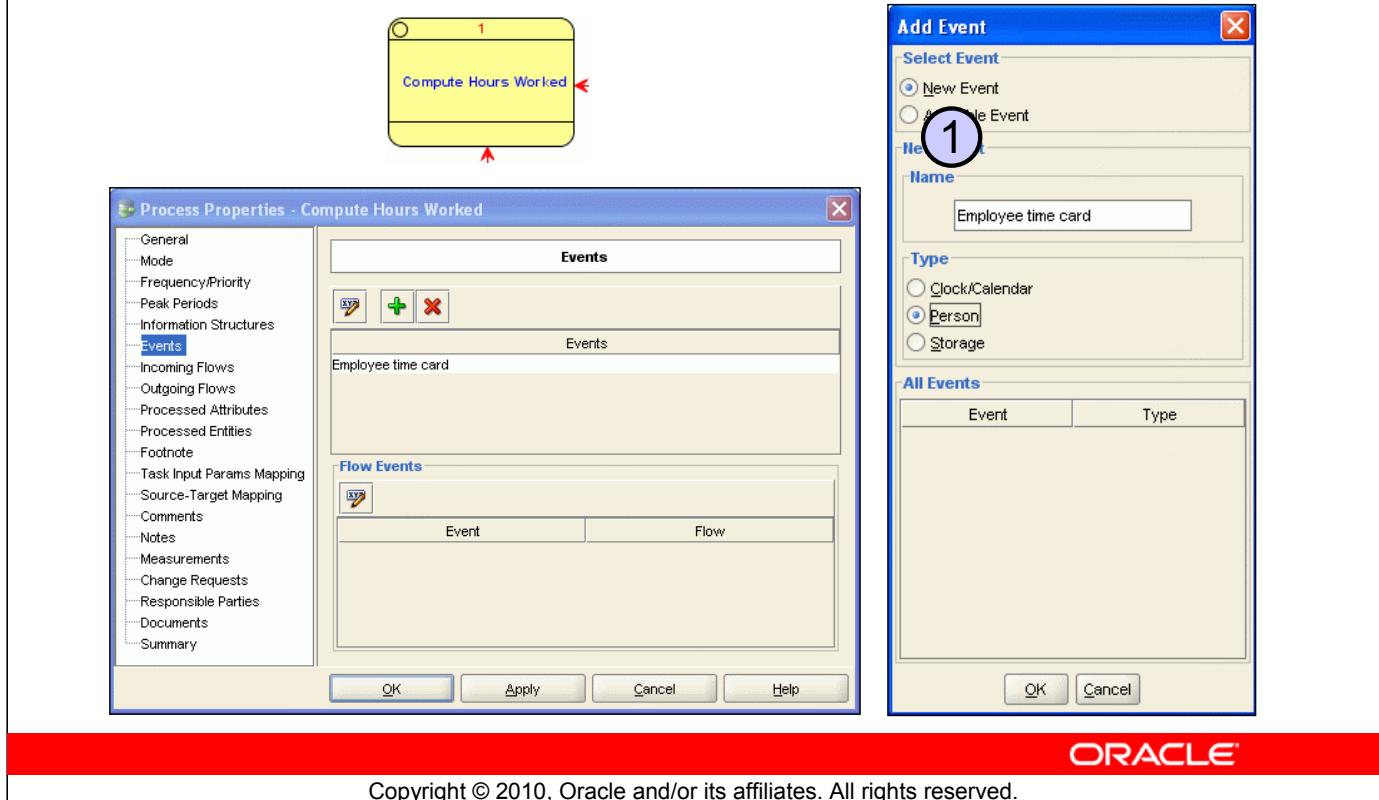
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Editing the Diagram Layout (continued)

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 Color and select the box next to Background Color or Border Color.
3. Select the desired color and click OK.
4. Click OK again.

Adding and Reusing Process Events



Oracle University and Bridge Human Skills Developments, GCC use only.

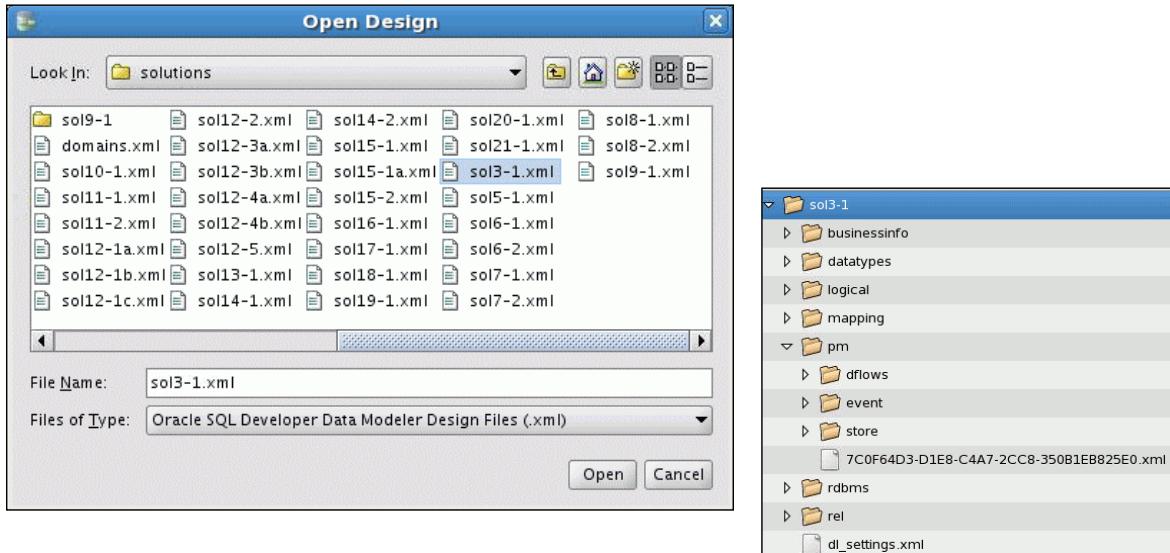
Adding and Reusing Process Events

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

1. Double-click the process that you want to define the event for.
2. Select the Event category in the left navigator.
3. Click the Add '+' icon.
4. Specify whether this is a new or available event.
 - If a new event, enter the name and select the event type and click OK.
 - If an available event, select it from the list of events and click OK.
5. Click OK again.

Opening and Saving Your Model

Saving a model creates an .xml file and a directory with process model objects.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Opening and Saving Your Model

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

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

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to build, load, and set the default options for building a DFD in Oracle SQL Developer Data Modeler and how to edit its layout.

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

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

In this practice, you use Oracle SQL Developer Data Modeler to build the data flow diagram for the Starlight DVD case study.

5

Validating Your Data Flow Diagram

ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

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



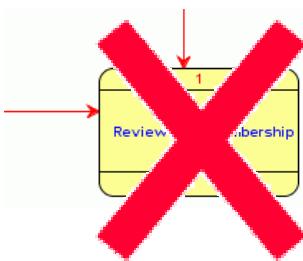
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

In this lesson, you review some of the rules associated with building data flow diagrams (DFDs), identify different types of processes, and then decompose the processes into the lowest-level primitive process.

DFD Rules: Process

- Every process has to have at least one input and one output.
- Use verb phrase labels.
- Each process has a unique name.



ORACLE

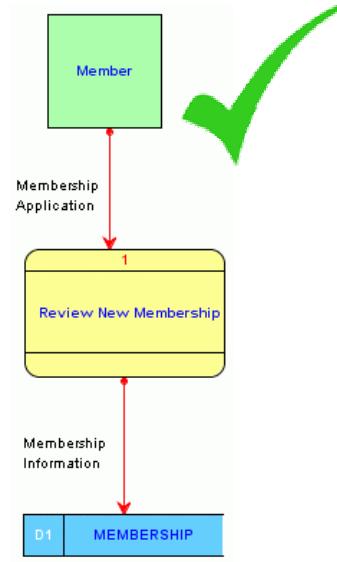
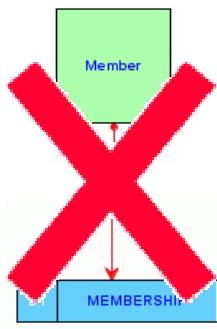
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

DFD Rules: Process

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 side of the slide only has input processes and therefore is invalid. The process on the right side of the slide has an input and an output, so it is considered valid.

DFD Rules: External Agents

- External agents move directly to and from a process.
- Use a noun phrase label.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

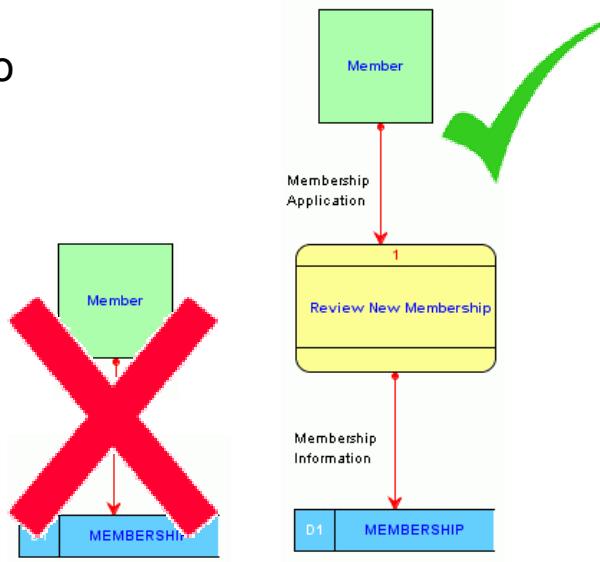
ORACLE

DFD Rules: External Agents

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.

DFD Rules: Information Store

- Data cannot move directly from one information store to another information store.
- An information store cannot move directly from or to an external agent.
- One end of an information flow must be a process.
- Use a noun phrase label.



ORACLE

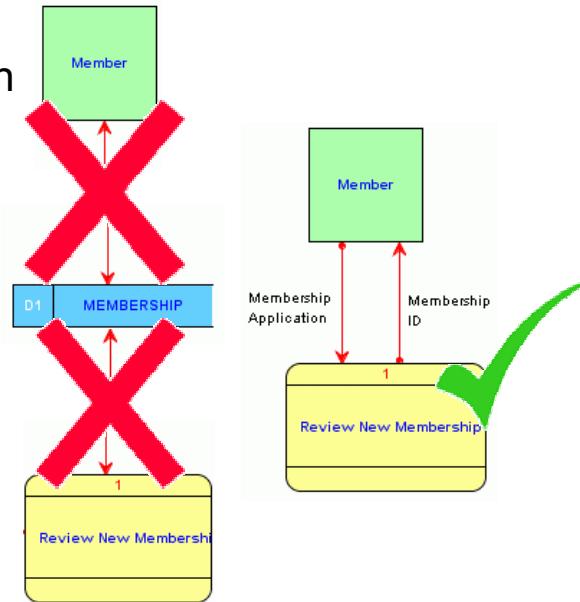
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

DFD Rules: Information Store

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.

DFD Rules: Information Flow

- An information flow has only one direction of flow between objects (not a double arrow line).
- An information flow to an information store creates, updates, or deletes data.
- An information flow from an information store retrieves data.
- Use a noun phrase label.



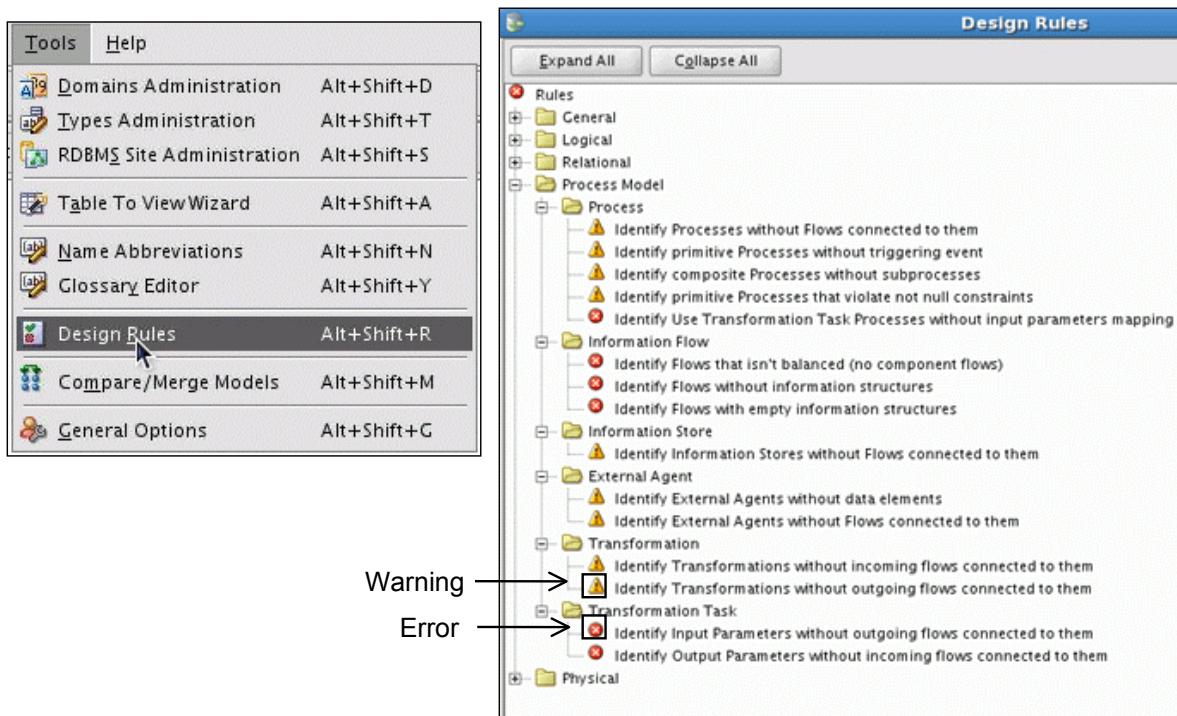
ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

DFD Rules: Information Flow

An information flow does not allow bidirectional flows where both sides of the line have an arrow. In order 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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Design Rules in Oracle SQL Developer Data Modeler

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 process to process
- b. A flow from process to information store
- c. A flow from information store to external agent
- d. A flow from external agent to process



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: c

Quiz

Which of the following is valid?

- 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



Copyright © 2010, 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Types of Processes

You will 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.

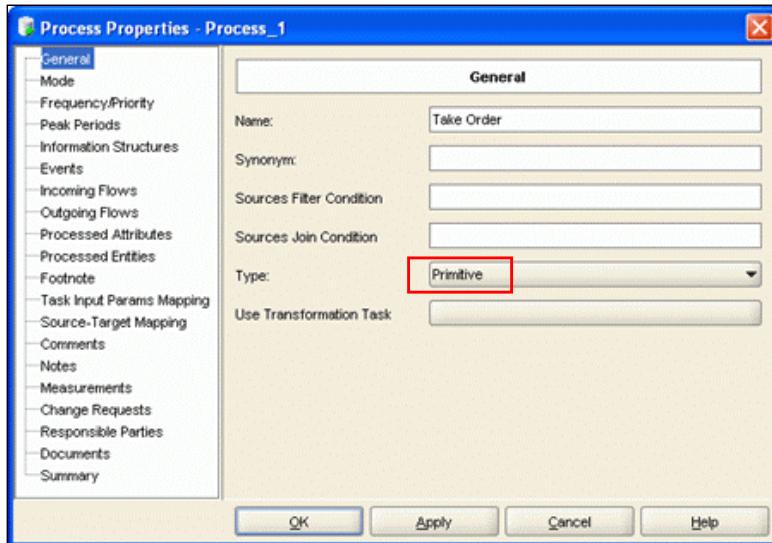
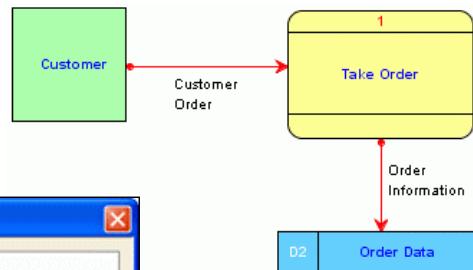
Primitive: The lowest-level stand-alone process. A primitive process usually has one input and one output, and the process cannot be divided any further. For example, the Take Order process is one specific task that cannot be subdivided into lower-level processes.

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.



ORACLE

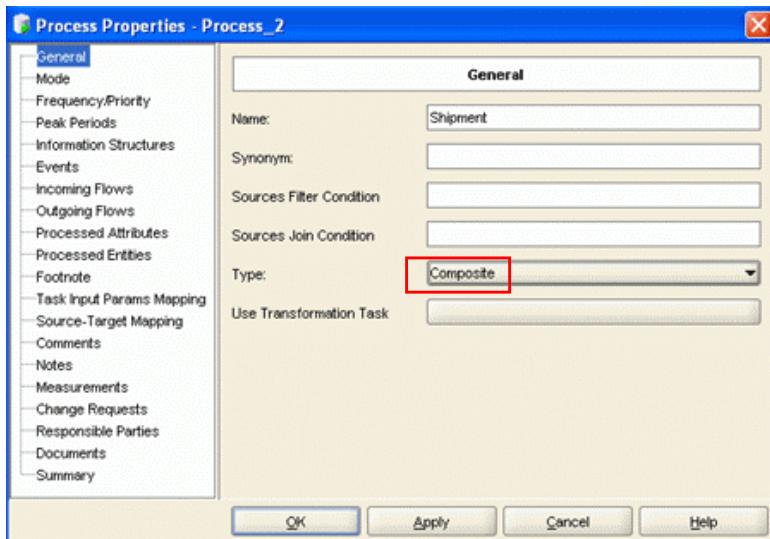
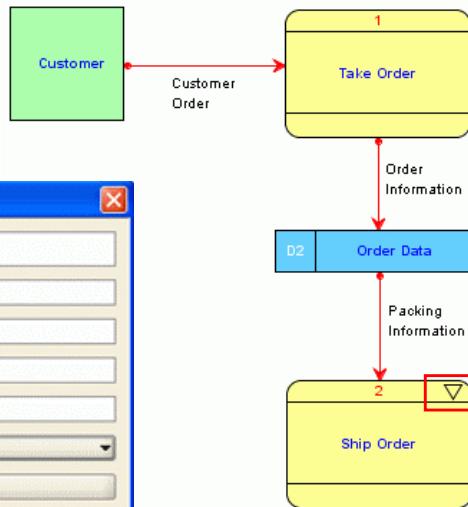
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Primitive Process

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.



ORACLE

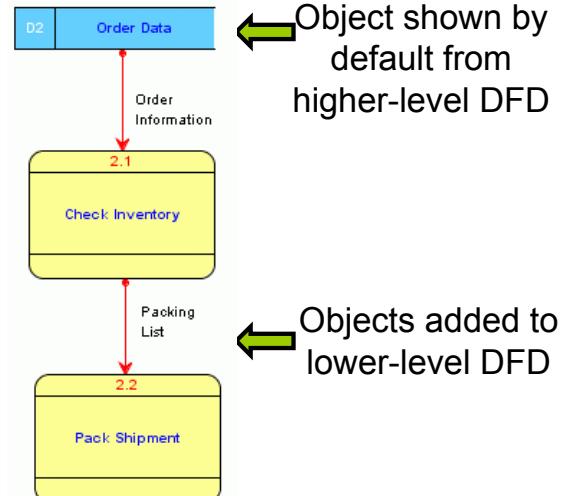
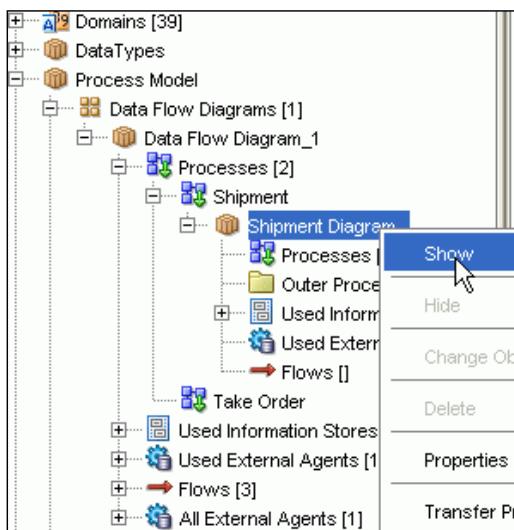
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Composite Process

A composite process is a higher-level process that can be subdivided into a lower-level DFD. To create a primitive 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 on the next page.

Composite Process

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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Composite Process (continued)

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Transformation Task Process

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

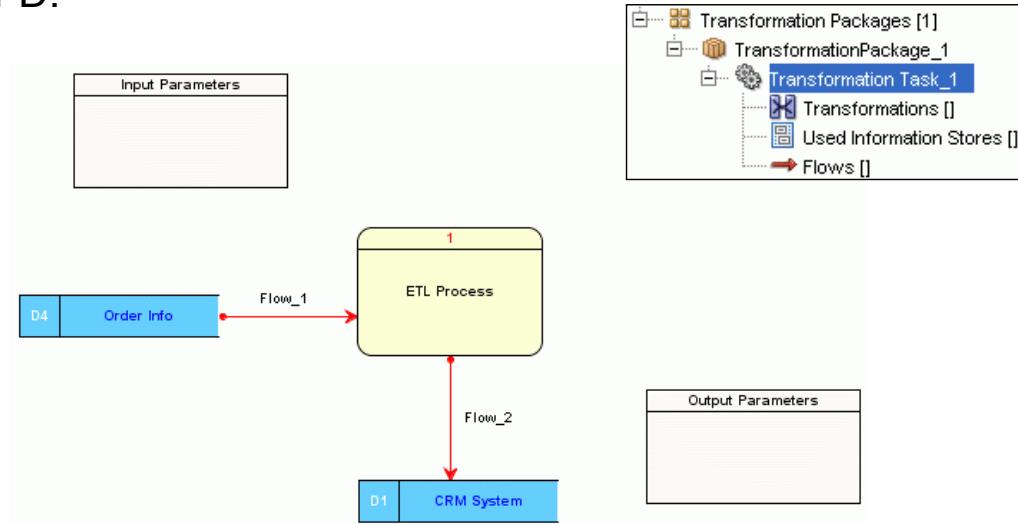
Transformation tasks have the following characteristics:

- 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 outlined in the slide.

Transformation Task Process

Create a transformation task package and a transformation task DFD.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

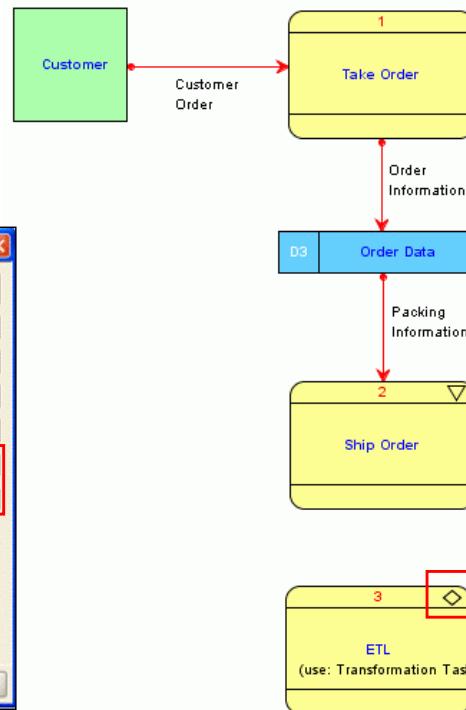
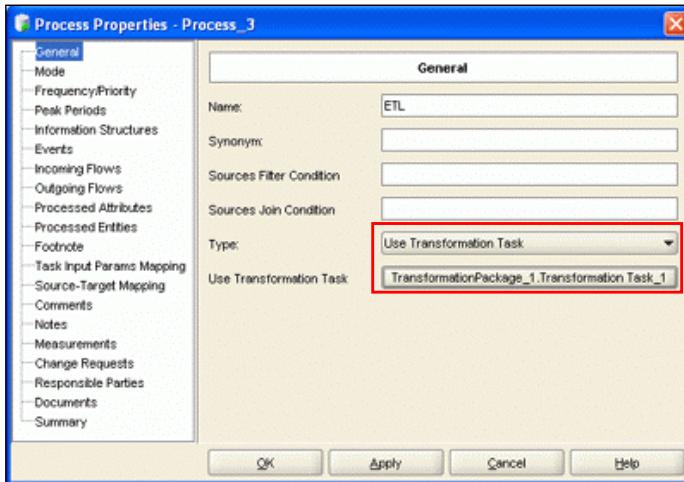
Transformation Task Process (continued)

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.



ORACLE

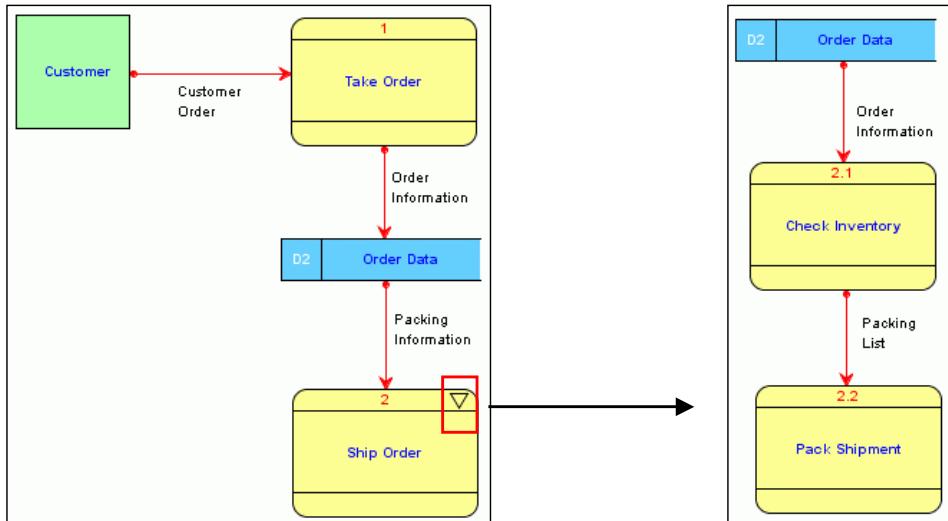
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Transformation Task Process (continued)

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Process Decomposition

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 may 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 may repeat information stores and external agents on a DFD.

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Decomposition Guidelines

There are a few guidelines to remember when building DFDs and decomposing 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 may add information flows to represent data that is transmitted under exception conditions.
- You may duplicate information stores and external agents in order 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



Copyright © 2010, 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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to validate a DFD based on a set of DFD Rules. In addition, you should have learned the different types of processes and how to decompose processes to the lowest primitive level.

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 © 2010, Oracle and/or its affiliates. All rights reserved.

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

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

Developing a Logical Data Model

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Overview

In this unit, you examine the following areas:

- Lesson 6: Identifying Entities and Attributes
- 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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Overview

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

Identifying Entities and Attributes



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Identify and diagram entities from a case study
- Identify and diagram attributes from a case study



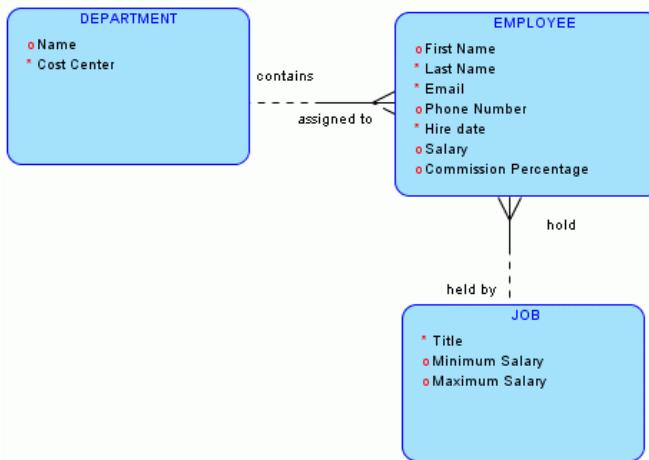
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

In this lesson, you learn how to identify entities and attributes from a case study.

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 © 2010, Oracle and/or its affiliates. All rights reserved.

What Is a Logical Data Model?

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 between the data. In the textual descriptions, you can understand the definitions behind the data and how the objects are related in more detail.

Why Create an ERD?

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

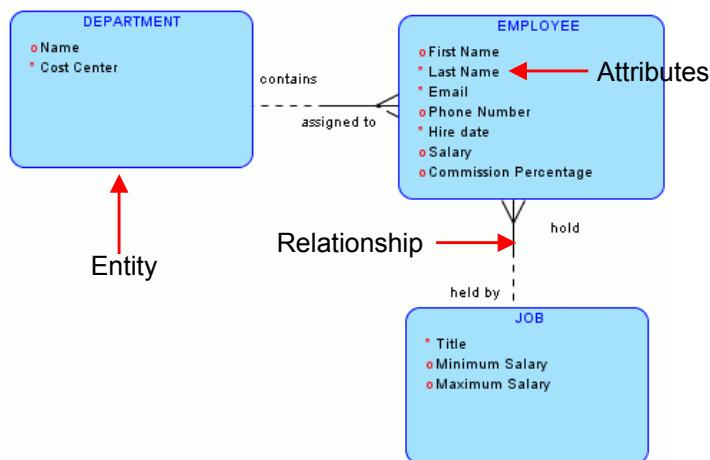
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Why Create an ERD?

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 the time up front validating the business rules before implementation decisions are made.

Components of an Entity Relationship Diagram

Entity relationship diagrams contain three different diagram objects.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Components of an Entity Relationship Diagram

An entity relationship diagram has three main diagram objects.

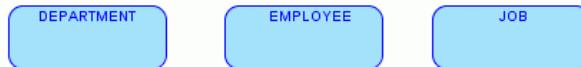
Component	Purpose
Entity	An object or concept about which you want to store information
Relationship	A natural association that exists between two or more entities
Attributes	Descriptions of entities and specific pieces of information that must be known

Entity

An entity is:

- Information that must be tracked
- A name for things that you can list
- Usually a noun

Examples:



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Entity

An entity is something of interest. Entities are categories of things that are important for a business, about which information must be kept. Entities contain facts 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:

- Rectangular box
- Has a unique name, usually in noun form
- Name is in uppercase
- 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Entity Types

An entity can be classified as a prime entity, a characteristic entity or an intersection entity.

A prime entity is independent and does not depend on the existence of any other entity. An example might be CUSTOMER or INSTRUCTOR.

A characteristic entity depends on the existence of another entity. An example might be an ORDER entity that is dependent on a CUSTOMER or on a CLASS OFFERING that is dependent on an INSTRUCTOR.

An intersection entity depends on the existence of two or more entities. An example might be an ORDER ITEM being dependent on ORDER and PRODUCT, or CLASS ENROLLMENT being dependent on CLASS OFFERING and STUDENT.

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



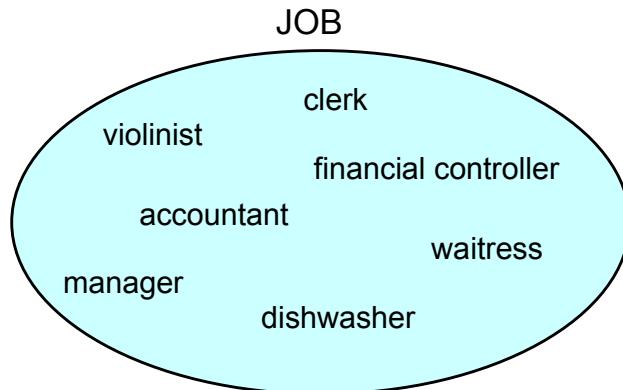
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Entities and Instances

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 Oistrach” 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Entities Represent Sets

You can regard entities as sets. The example in the slide 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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: c

Quiz

Which of the following is an instance of an entity?

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

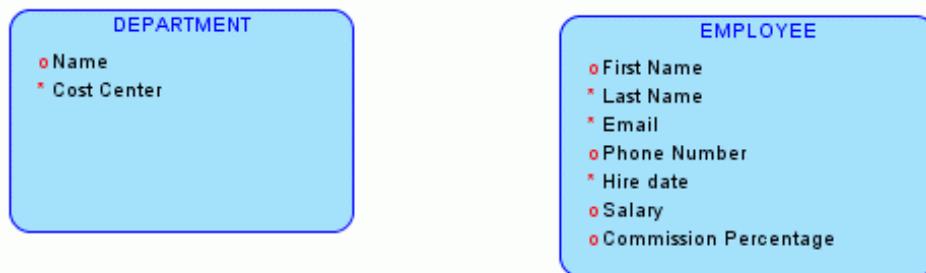
Answer: c

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

Attributes

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

Examples:



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Attributes

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 have the following characteristics:

- Attributes are shown within the entity box.
- Attribute names are singular and shown in mixed or lower case.
- 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, indicated by an o



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Attribute Characteristics

Attributes have the following characteristics:

- Attributes are displayed within the entity box on the ERD.
- Attribute names should be singular and in mixed or lower case.
- 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 “not null” (indicated by the asterisk * symbol next to the attribute) or “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 would like to model the entity and attributes for a set of requirements. The entity and attributes must store the following:

- Course information, including its name, 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Class Practice: Identify Entities

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 case study.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

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

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 6-1 Overview: Identify Entities and Attributes

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

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 6-2 Overview: Identify Entities and Attributes

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



Identifying Relationships

ORACLE

Copyright © 2010, 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 using a relationship matrix



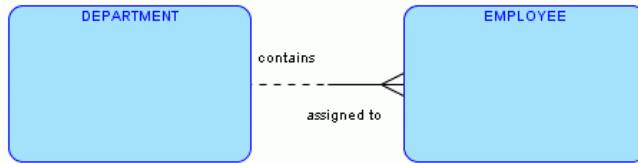
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

In this lesson, you identify and create relationships in an ERD.

Relationships

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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Relationships

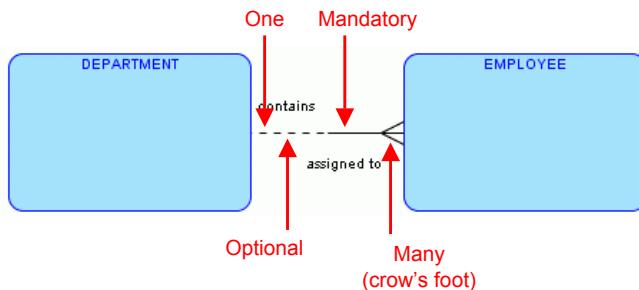
A relationship represents the business rules that link entities. There are always two business rules for each relationship. In the example in the slide, the business rules are:

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

Components of a Relationship

Each direction of a relationship has:

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



ORACLE®

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Components of a Relationship

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:

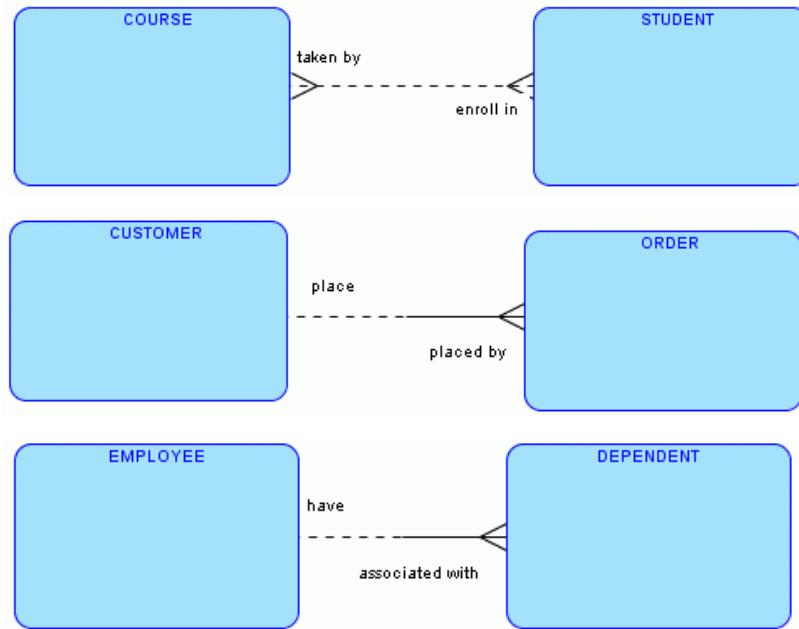
- **Optional:** Use “may be” or “may.”
 - **Mandatory:** Use “must be” or “must.”

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Relationships: Additional Examples

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:

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



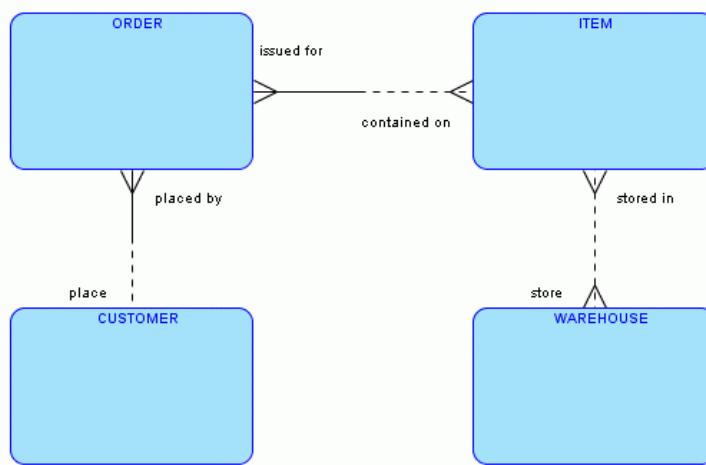
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

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 © 2010, Oracle and/or its affiliates. All rights reserved.

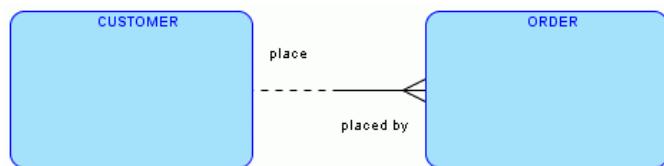
Class Practice: Define Business Rules

In this practice, you are presented with an ERD and you are tasked with creating the business rule sentences.

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Relationship Types

There are three types of relationships:

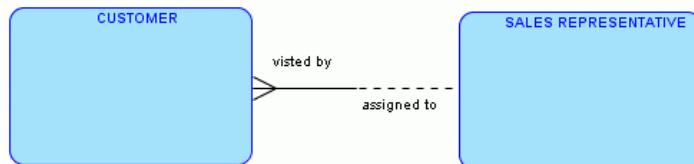
- **Many to one (M:1) or one to many (1:M):** There are crow's feet on one side of the relationship. The direction of the crow's feet determines whether the relationship is M:1 or 1:M. This type of relationship is the most common.
- **Many to many (M:M):** There are crow's feet on both sides of this 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, as follows:

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

Many-to-One Relationships

Many-to-one relationships (M:1 or 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

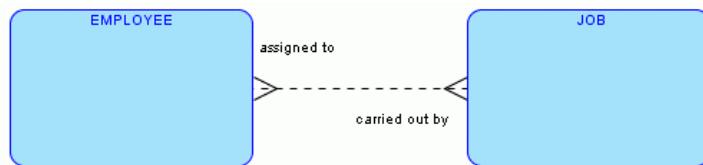
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Many-to-One Relationships

The example in the slide is a many-to-one relationship because the cardinality is one or more in one direction and one and only one in the other direction.

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

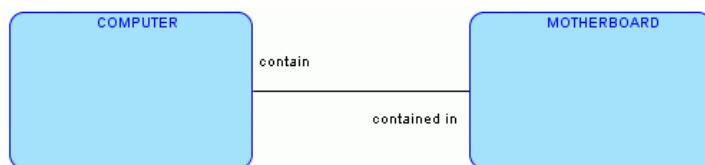
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Many-to-Many Relationships

The example in the slide is a many-to-many relationship because the cardinality is many to many in both directions.

One-to-One Relationships

One-to-one relationships (1:1) have cardinality of 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 © 2010, Oracle and/or its affiliates. All rights reserved.

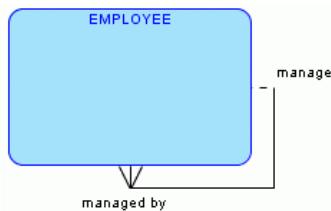
One-to-One Relationships

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Recursive Relationships

One additional relationship that must be mentioned in this lesson is a recursive relationship. Recursive relationships are relationships with an entity and itself. In the slide, there is a recursive relationship with the EMPLOYEE entity. There are still two business rules for this type of relationship.

Quiz

Which type of relationship would 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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: a

Using a Relationship Matrix

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

	CUSTOMER	ITEM	ORDER	WAREHOUSE
CUSTOMER			place	
ITEM			issued for	stored in
ORDER	placed by	contained on		
WAREHOUSE		store		

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

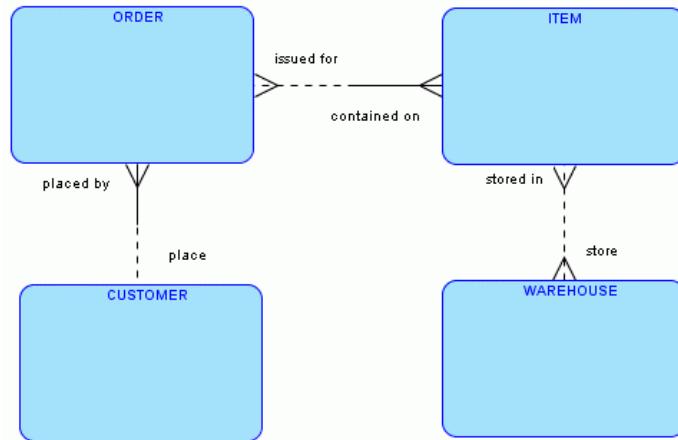
Using a Relationship Matrix

A relationship matrix has the following characteristics:

- A relationship matrix shows if and how each row entity on the left-hand 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-hand 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Using a Relationship Matrix (continued)

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Determining a Relationship's Existence

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.

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Naming the Relationship

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.

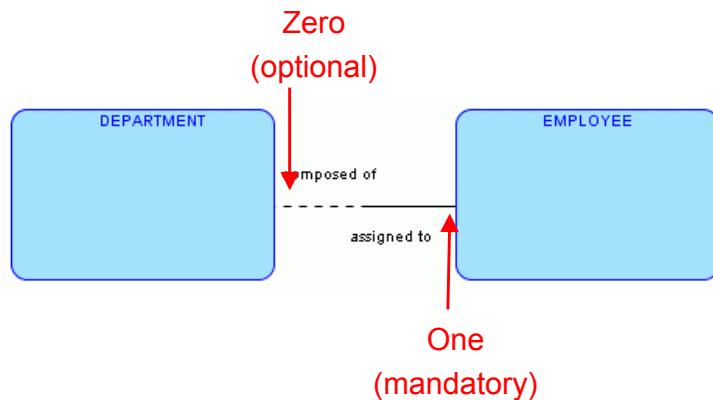
The following is a sample list of relationship name pairs to assist in naming relationships:

- based on the basis for
- bought from the supplier of
- description of for
- operated by the operator for
- represented by the representation of
- responsible for the responsibility of

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Determining the Relationship's Cardinality

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:

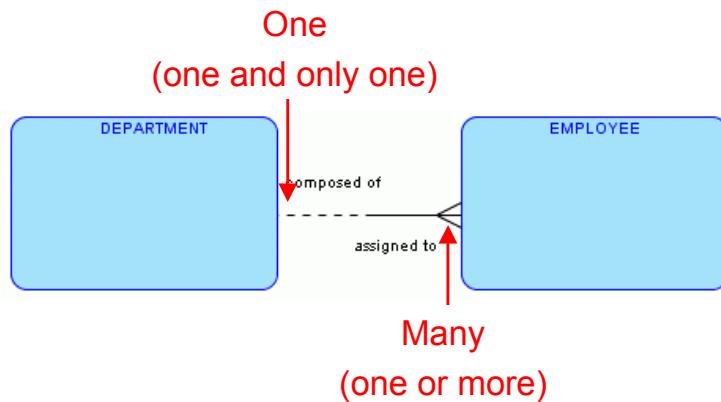
1. Must an EMPLOYEE be assigned to a DEPARTMENT? Always
2. Is there any situation in which an EMPLOYEE will not be assigned to a DEPARTMENT?
No, an EMPLOYEE must always be assigned to a DEPARTMENT. (Mandatory)
3. Must a DEPARTMENT be composed of an EMPLOYEE?
No, a DEPARTMENT does not have to be composed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

1. What is the maximum cardinality in each direction?



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Determining the Relationship's Cardinality (continued)

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

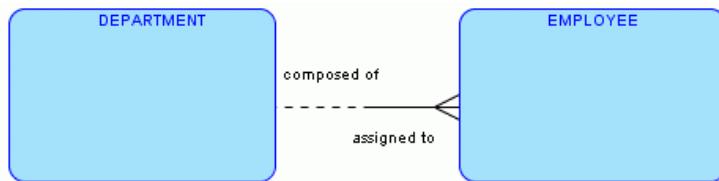
In the example in the slide, answer the following questions:

1. Must an EMPLOYEE be assigned to more than one DEPARTMENT?
No, an EMPLOYEE must always be assigned to one and only one DEPARTMENT. (One)
2. 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Validating the Relationship

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

Quiz

Which of the following entities have a valid relationship?

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



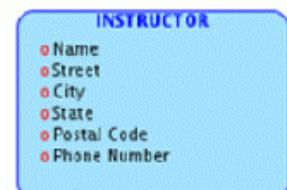
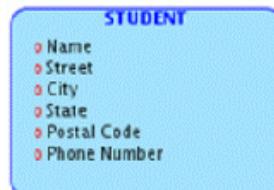
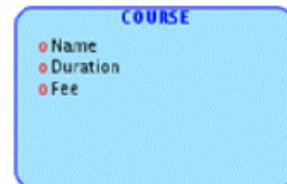
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

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

- Build a relationship matrix for some of the entities that you identified previously.
- Then establish the relationships in the diagram.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Class Practice: Build a Relationship Matrix

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 using a relationship matrix



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned the components of a relationship and how to create the relationships between two entities.

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 between entities



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 7-1 Overview: Analyze and Model Relationships

In this practice, you analyze and model relationships based on a set of requirements by using a relationship matrix.

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 between entities



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 7-2 Overview: Analyze and Model Relationships

In this practice, you analyze and model relationships based on a set of requirements by using a relationship matrix.

8

Assigning Unique Identifiers

ORACLE®

Copyright © 2010, 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.



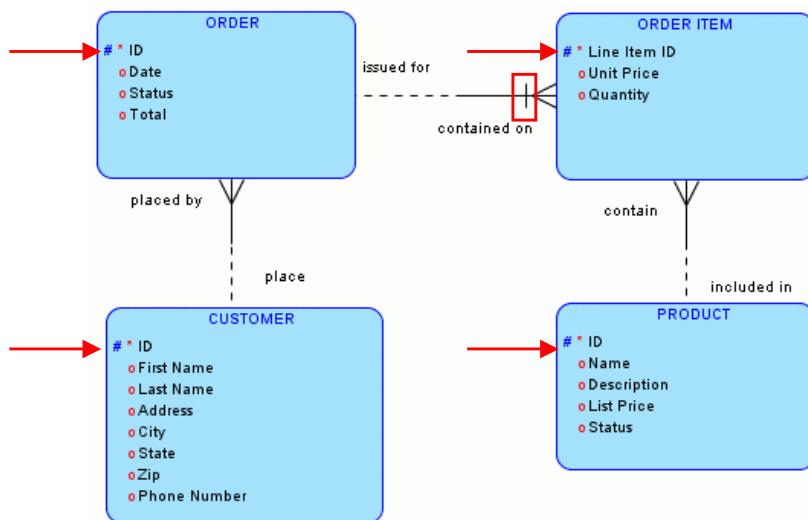
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

In this lesson, you identify the unique identifiers for your entities and relationships.

Unique Identifiers

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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

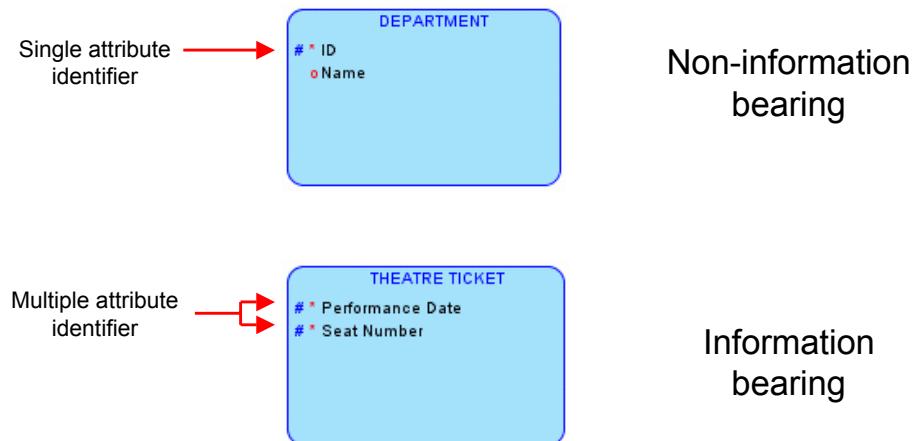
Unique Identifiers

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 will be discussed later in this chapter.

Unique Identifier Examples

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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Unique Identifier Examples

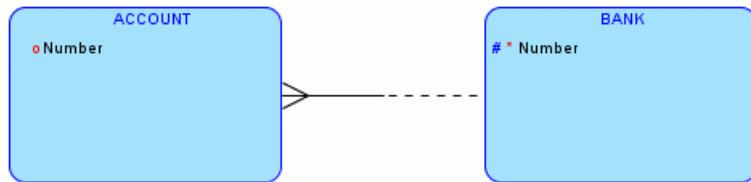
Each entity must have a unique identifier, or it is not an entity. The unique identifier can be information bearing, when the values of the unique identifier have some business meaning. Non-information bearing unique identifiers are sometimes referred to as a synthetic key. Information-bearing unique identifiers are 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 it is non-information bearing.

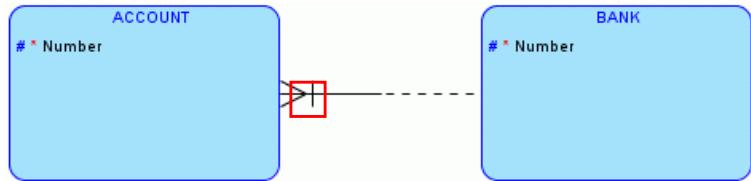
For the **THEATRE TICKET** entity, the unique identifier is the combination of the **Performance Date** and the **Seat Number** attributes. In this case, the unique identifier is more than one attribute and it is information bearing.

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 © 2010, Oracle and/or its affiliates. All rights reserved.

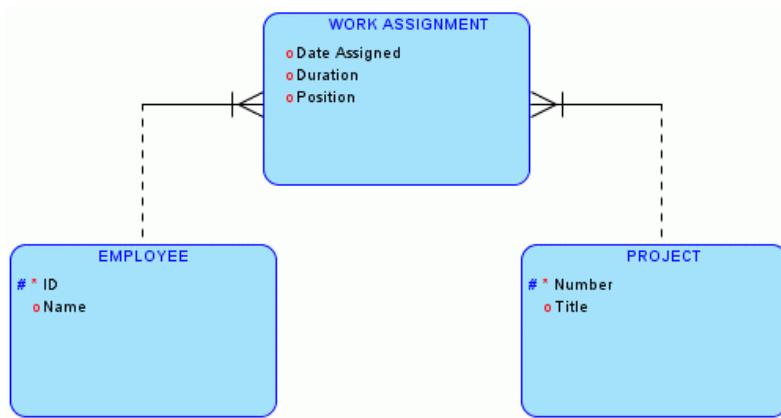
Identifying Relationships

An identifying relationship is created when the unique identifier for an entity includes the relationship with another entity in order 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 may be uniquely identified through multiple relationships.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

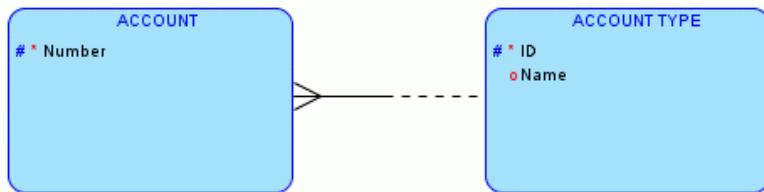
Identifying Relationships with Multiple Entities

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 which is the resolution between an M:M relationship. This topic will be discussed in a later lesson.

Non-Identifying Relationships

When the relationship between two entities is not required to be part of the unique identifier



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Non-Identifying Relationships

A non-identifying relationship is created when the unique identifier for an entity does not need the relationship with another entity in order for it to be unique. In the example in the slide, the ACCOUNT entity does not need the relationship with ACCOUNT TYPE to be unique. There can be instances of an ACCOUNT TYPE that do not have an ACCOUNT.

Primary and Secondary Unique Identifiers

An entity may 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Primary and Secondary Unique Identifiers

An entity may 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 identify 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?



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Searching for Unique Identifiers

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, perform 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. Always is 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

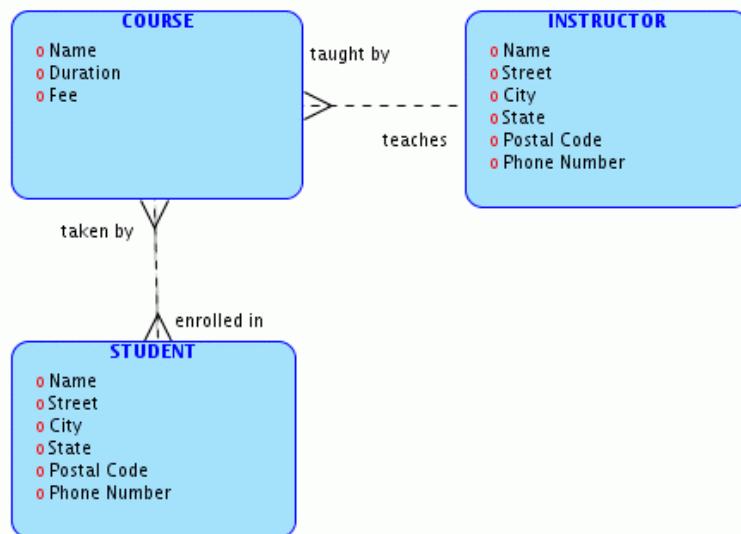


Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: c

Class Practice: Specify Unique Identifiers

Specify unique identifiers for each entity in the diagram.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Class Practice: Specify Unique Identifiers

In this practice, you specify the unique identifiers for the entities in the diagram in the slide.

Summary

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

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

Practice 8-1 Overview: Identify Unique Identifiers

This practice covers the following topics:

- Identifying unique identifiers for entities in your ERD



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 8-1 Overview: Identify Unique Identifiers

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 the following topics:

- Identifying unique identifiers for entities in your ERD



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 8-2 Overview: Identify Unique Identifiers

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

ORACLE

Copyright © 2010, 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



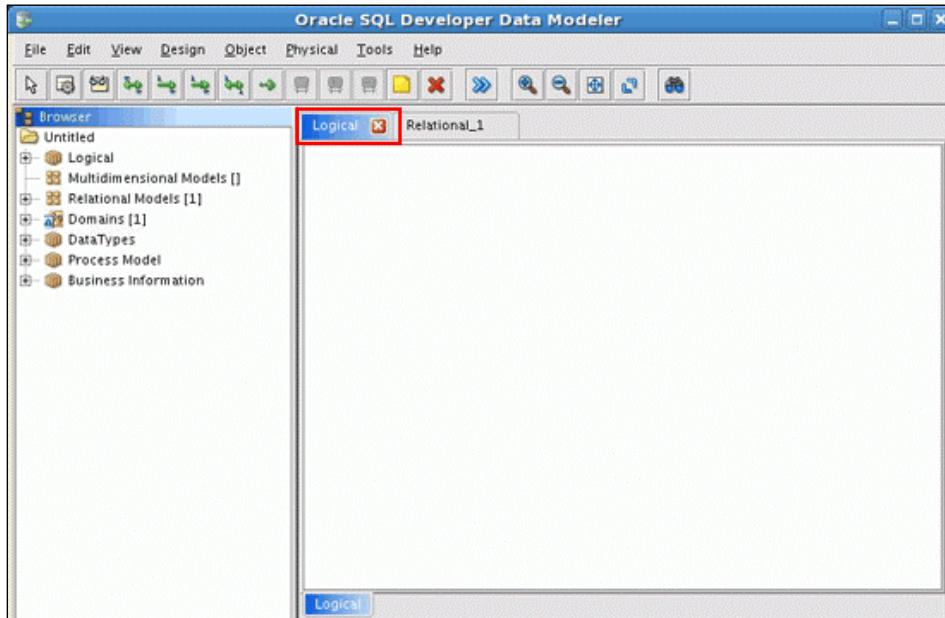
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

In this lesson, you learn how to build an Erd in Oracle SQL Developer Data Modeler.

Building an Entity Relationship Diagram

1. Click the Logical tab.



ORACLE

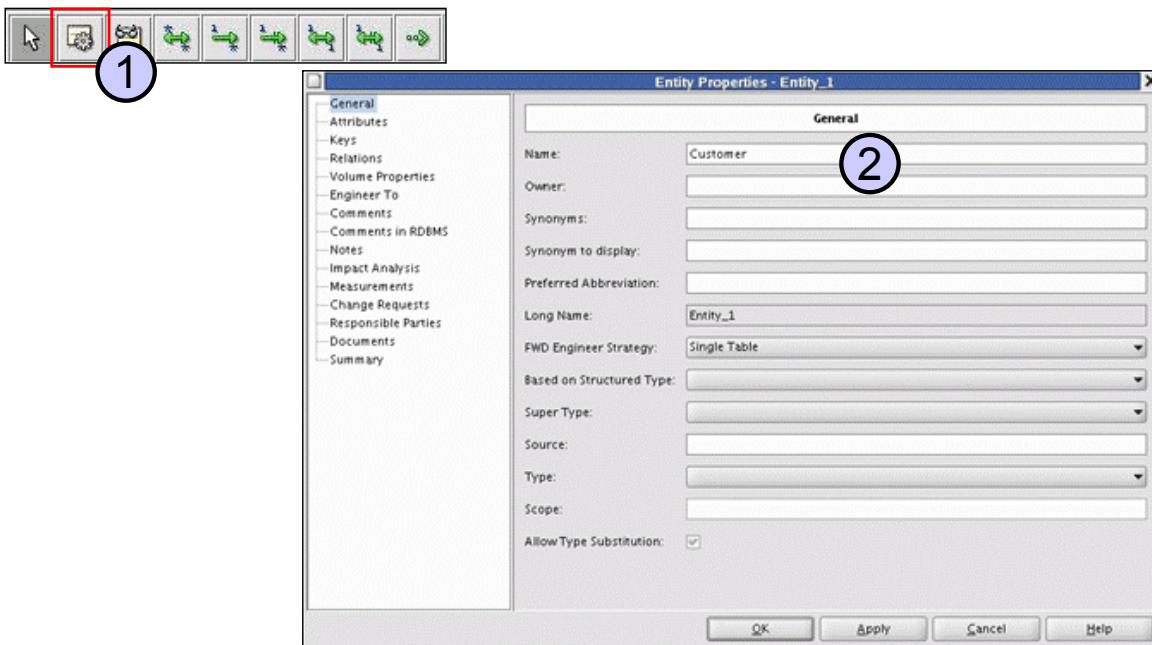
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Building an Entity Relationship Diagram

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.



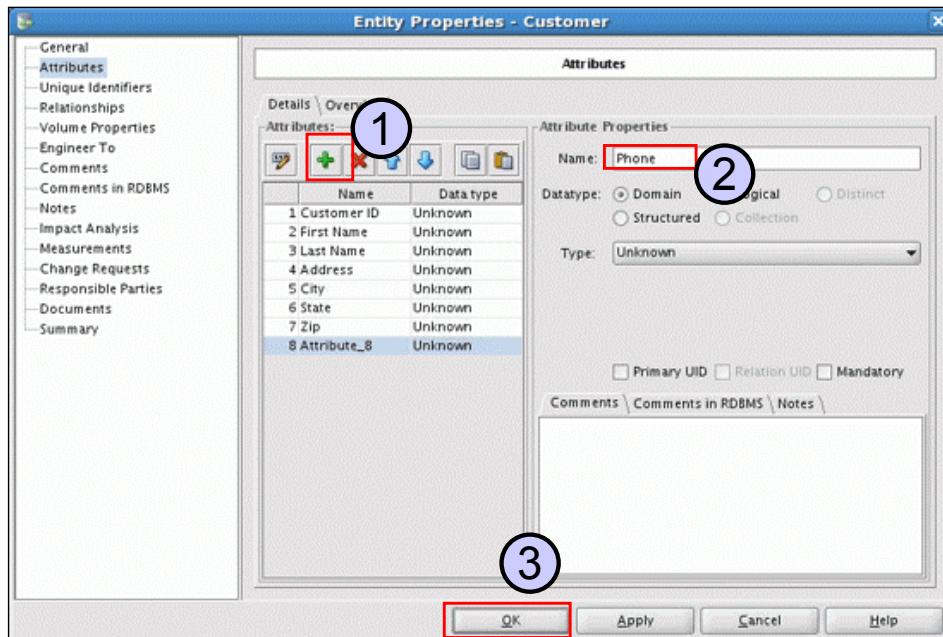
Building an Entity Relationship Diagram (continued)

To create an entity, perform the following:

1. On the toolbar, click the “Create an Entity” tool, and click anywhere in the white space of the Logical pane. The Entity Properties window appears.
2. 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 © 2010, Oracle and/or its affiliates. All rights reserved.

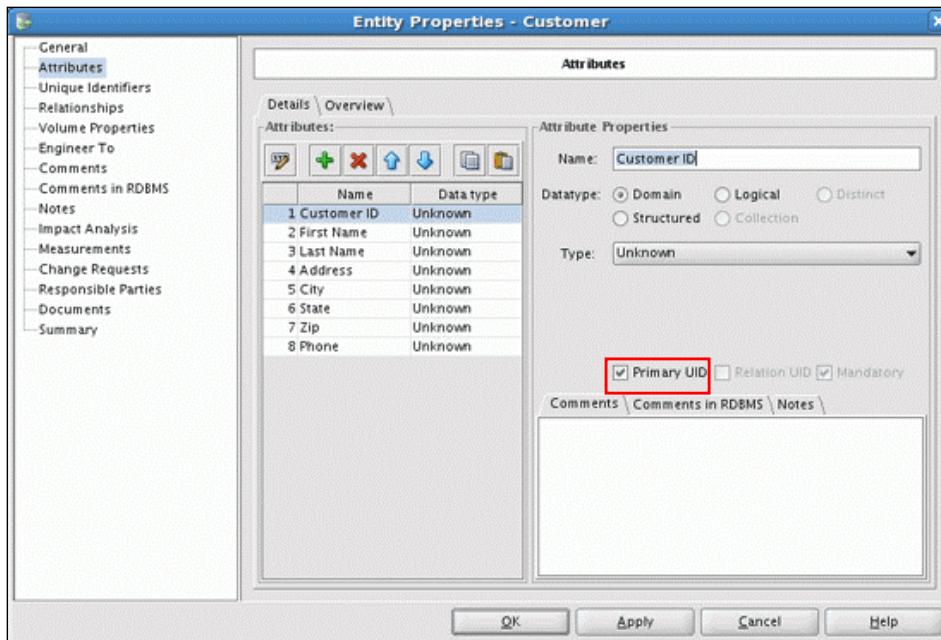
Building an Entity Relationship Diagram (continued)

To add attributes to the entity, perform the following:

1. With Attributes selected in the left navigator of the Entity Properties window, click the "Add an Attribute" icon.
2. In the Name field, enter a name for the attribute.
3. Repeat steps 1 and 2 until you have created all the attributes for this entity.
4. Use the up and down arrows to reorder the columns.
5. 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.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

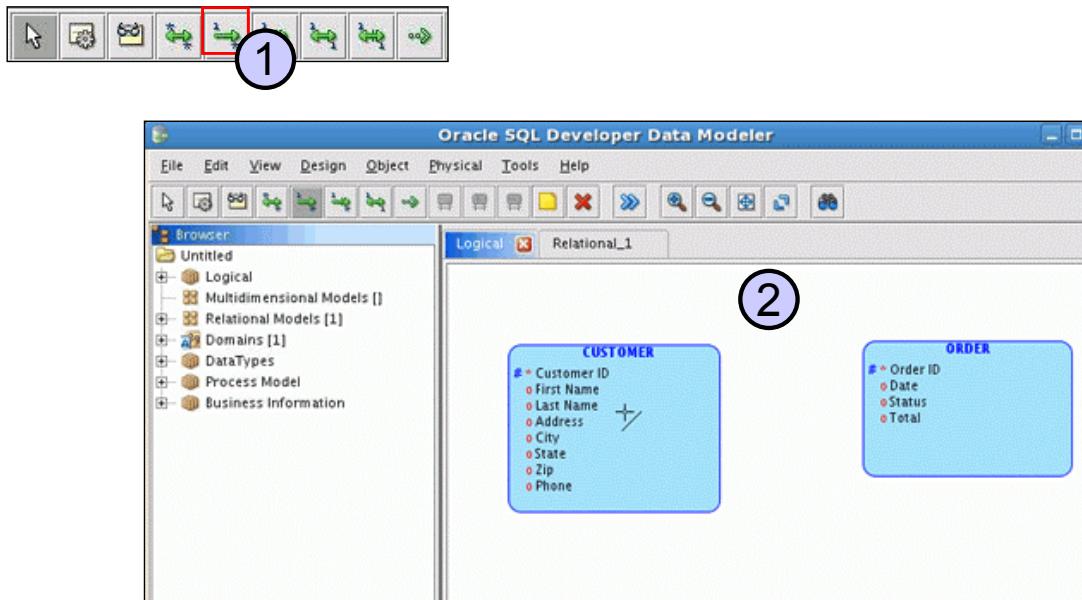
Building an Entity Relationship Diagram (continued)

To set the unique identifier for the entity, perform the following.

1. Click the entity that you want to define the unique identifier for.
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. Primary UID means “primary key.” Note that 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.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

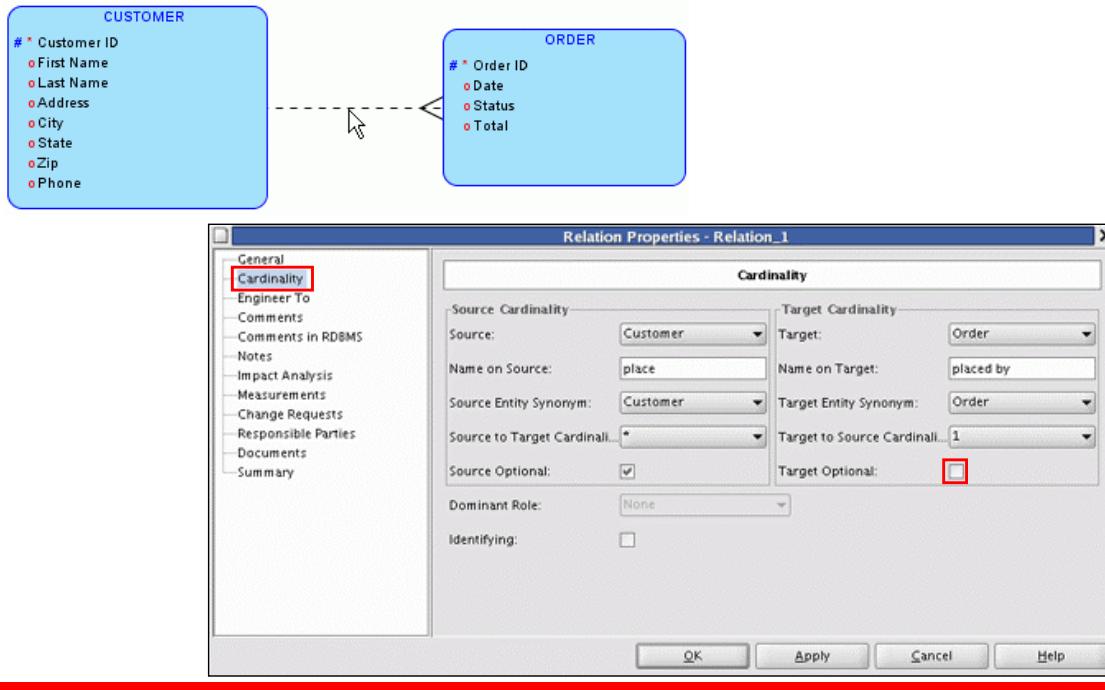
Building an Entity Relationship Diagram (continued)

To define the relationships between entities, perform the following:

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

Building an Entity Relationship Diagram

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

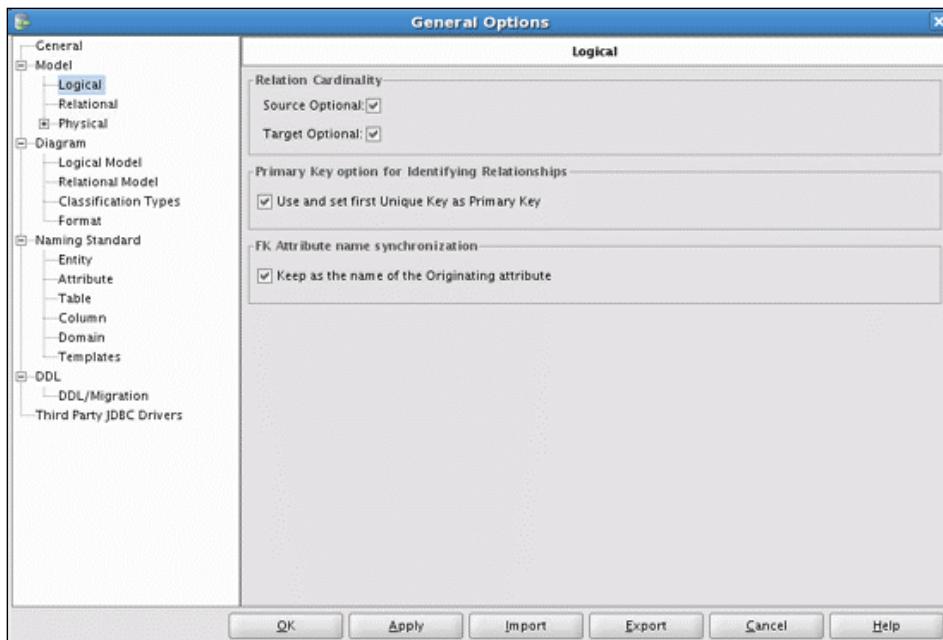
Building an Entity Relationship Diagram (continued)

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

1. Double-click the relationship in the diagram.
2. Select the Cardinality property in the left navigator.
3. Specify the source and target names for the relationship. Note that these names will be specified on the diagram and will be used to validate the business rules for the relationship.
4. You may also specify the minimum and maximum cardinality for the relationship. In the example in the slide, the check box for Target Optional was deselected because there must be a CUSTOMER on each ORDER.

Specifying Logical Model General Option

Logical Model Properties



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Specifying Logical Model General Option

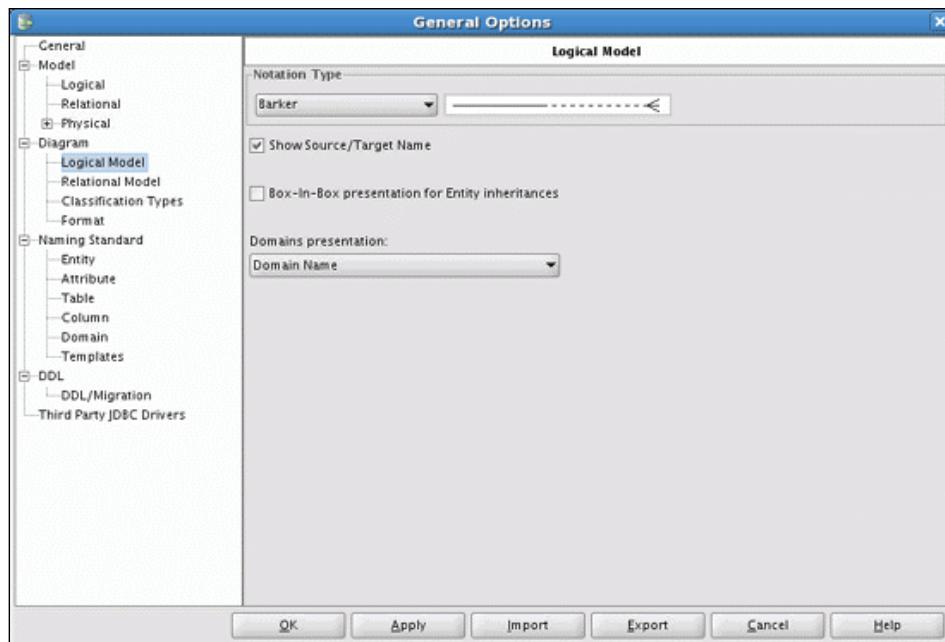
Within Oracle SQL Developer Data Modeler, you can set various properties for the Logical Model. To open the General Options window, select Tools > General Options. Expand Model and select Logical.

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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

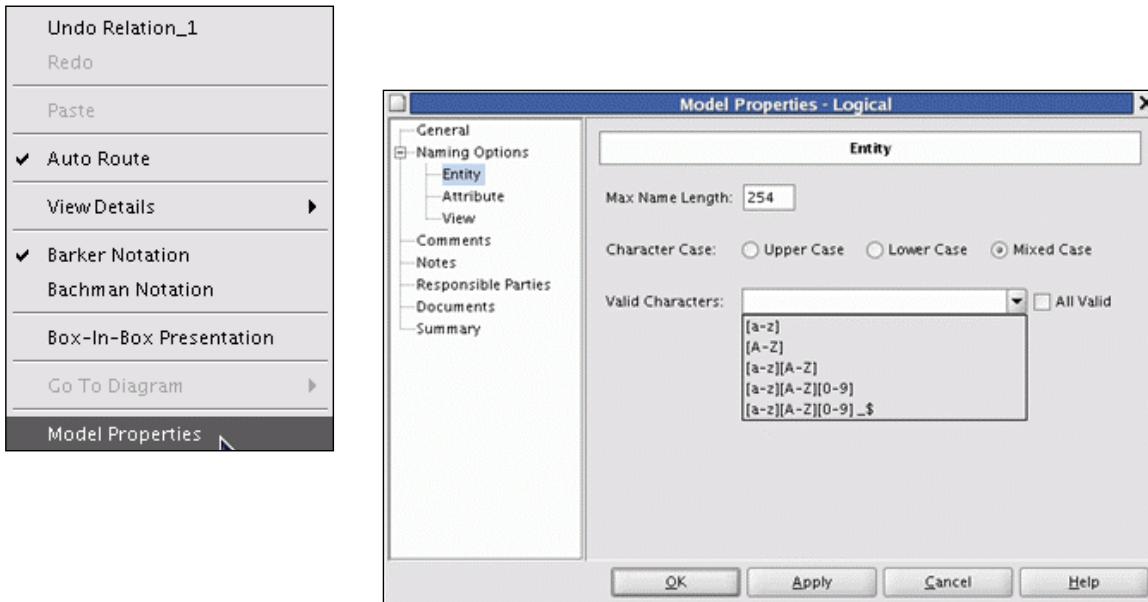
Specifying Logical Model Diagram Defaults

The Logical Model Diagram pane contains 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 in the screenshot in the slide, this option was selected because you want the names to display on 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 uses in the domain definition to be displayed.

Modifying Model Properties

To change the properties for a model:



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Modifying Model Properties

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

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

- **Name:** Is 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.

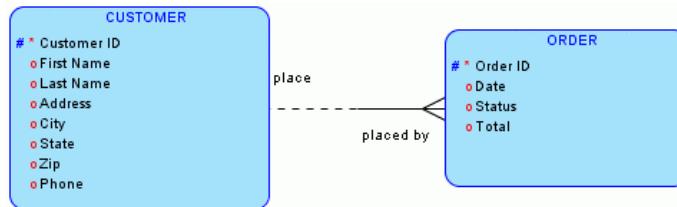
When Naming Options is selected in the left navigator, you can specify the following naming rules for entities, attributes, and views:

- **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.

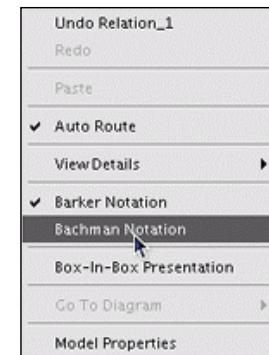
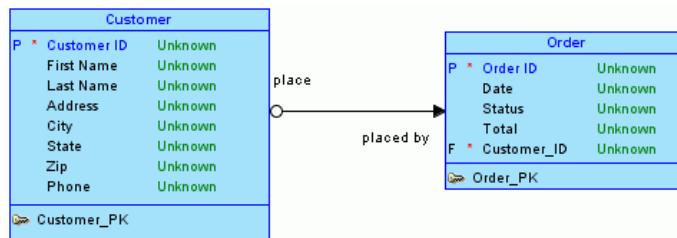
All other items on the left navigator are common for all property windows.

Notation Types

Barker
Notation



Bachman
Notation



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Notation Types

There are two notation types available in Oracle SQL Developer Data Modeler. Either notation is correct—they just represent different data modeling methodologies.

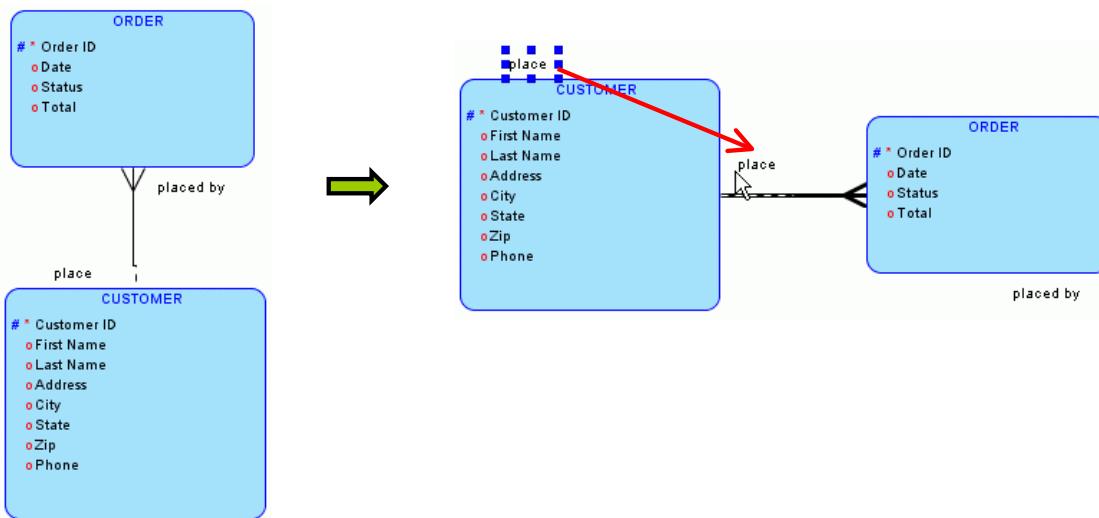
The Barker notation is used through the examples in this course.

The 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 will be discussed in Lesson 13.

Editing a Diagram Layout: Moving an Object

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



ORACLE

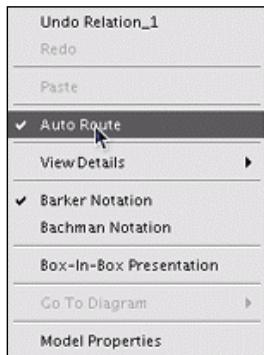
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Editing a Diagram Layout: Moving an Object

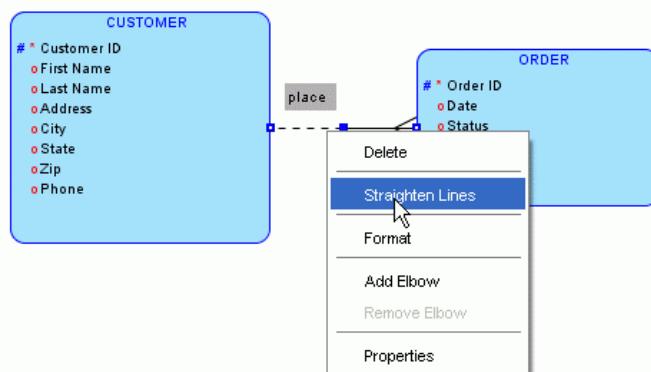
You can move an object by dragging the object to a new location and releasing the object. In the example in the slide, the CUSTOMER entity was dragged to a different location. Note: You may have to drag the source and target names of the relationship to appropriate locations after you move the entity.

Editing a Diagram Layout: Redrawing Lines

Turn off Auto Route.



Right-click a relationship line and select Straighten Lines.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Editing a Diagram Layout: Redrawing Lines

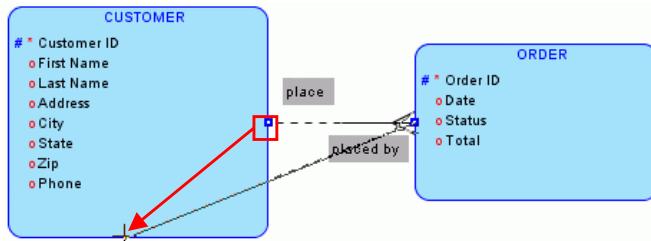
In order to redraw lines, you must first turn Auto Route off. Right-click in the white space and select Auto Route from the context menu.

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

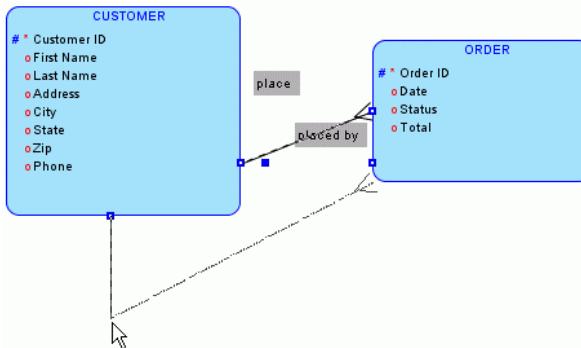
Editing a Diagram Layout: Moving a Relationship Line

To move a relationship line manually to another location:

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



2. Ctrl-drag points on the relationship line.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Editing a Diagram Layout: Moving a Relationship Line

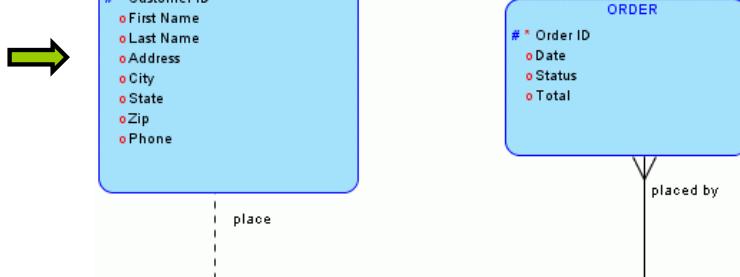
To move a relationship line manually to another location, perform the following:

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. Ctrl-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

To move a relationship line manually to another location:

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

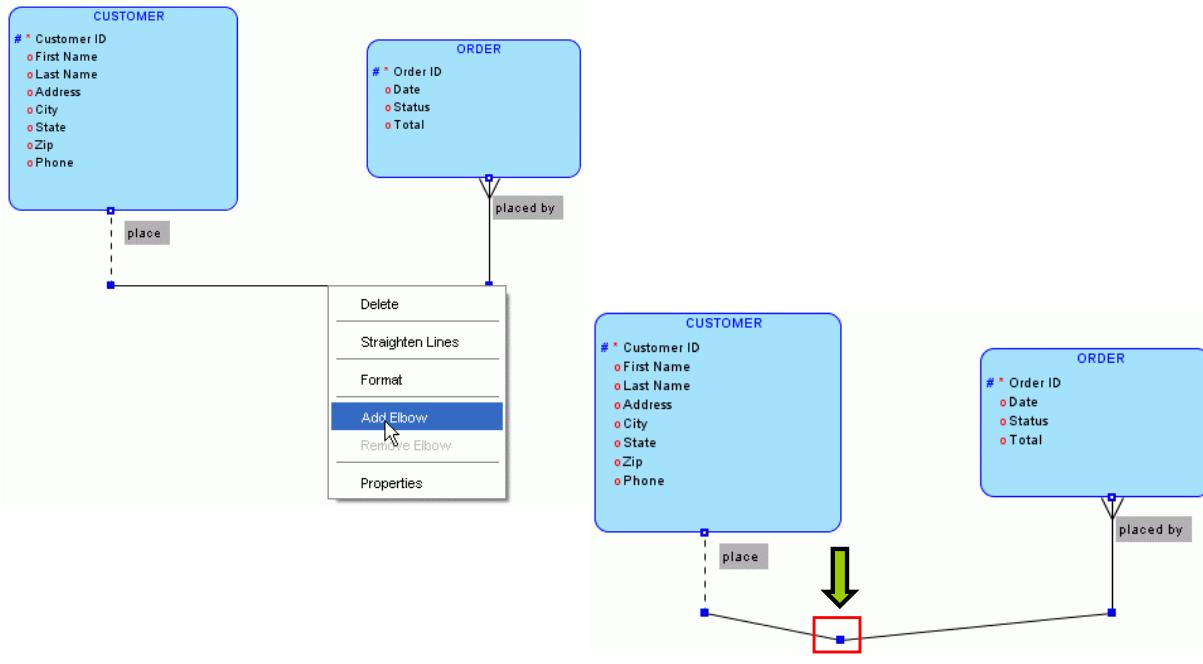
ORACLE

Editing a Diagram Layout: Moving a Relationship Line (continued)

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:



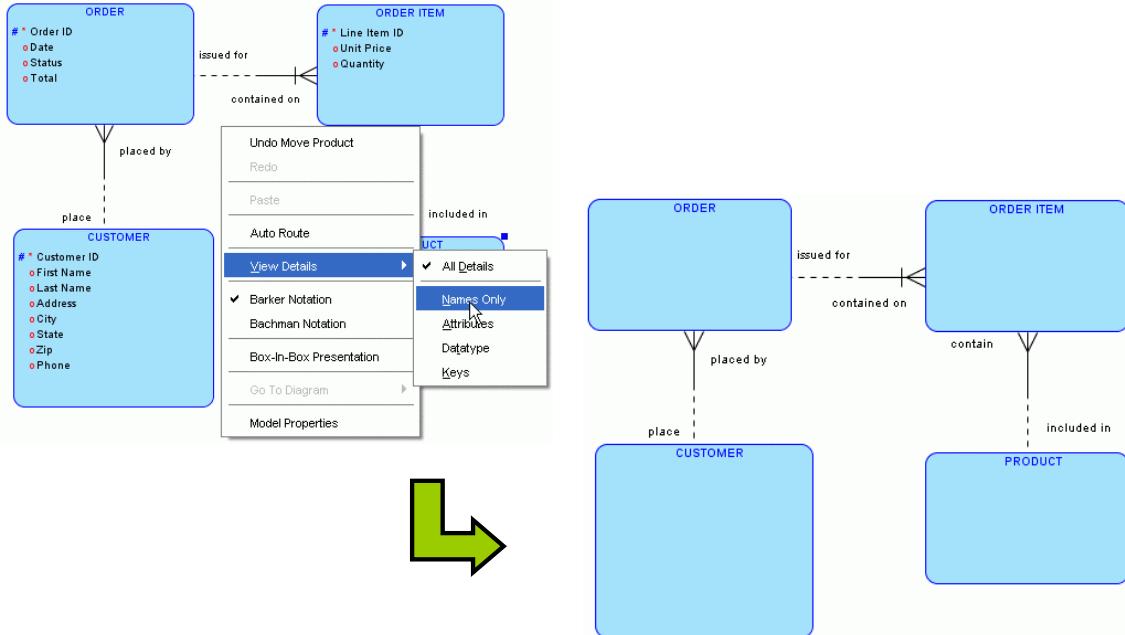
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

Editing a Diagram Layout: Adding an Elbow

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 © 2010, Oracle and/or its affiliates. All rights reserved.

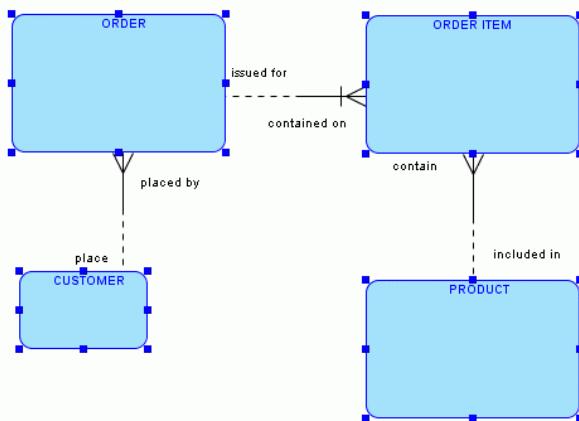
Editing a Diagram Layout: Showing Levels of Detail

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
2. Ctrl-select all the objects that you want to make the same size
3. Select Edit > Equal Width



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

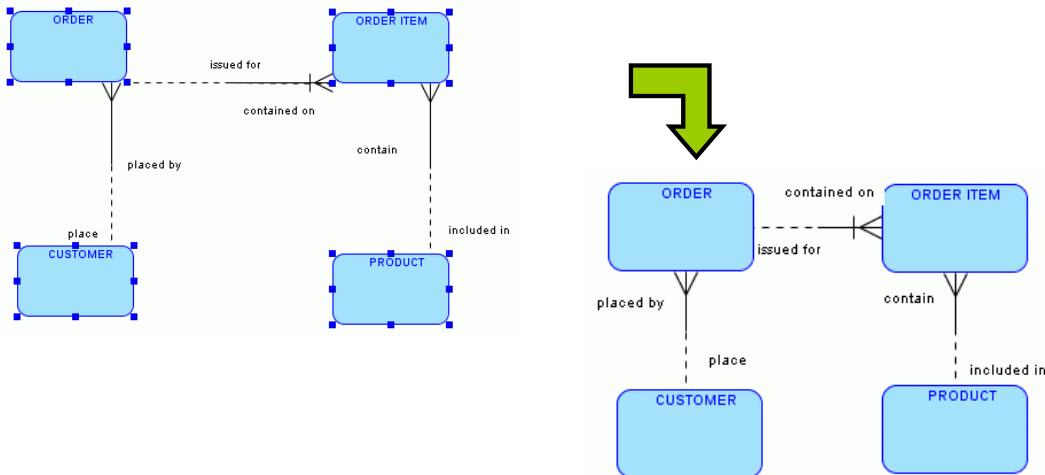
Editing a Diagram Layout: Resizing Multiple Objects

To resize multiple objects at the same time, perform the following:

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

Editing a Diagram Layout: Resizing Multiple Objects

4. Move the objects to their desired location.
5. Restraighten relationship lines and source and target names.



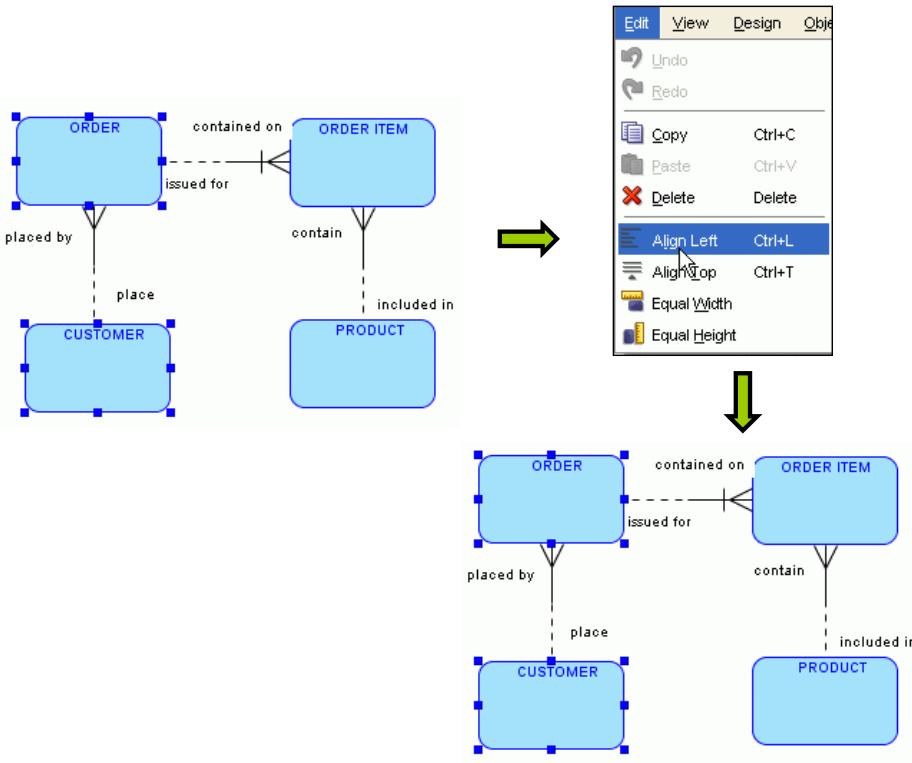
ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Editing a Diagram Layout: Resizing Multiple Objects (continued)

4. Move each object to its desired location to optimize space.
5. Restraighten each relationship line and source and target name.

Editing a Diagram Layout: Aligning Objects



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

Editing a Diagram Layout: Aligning Objects

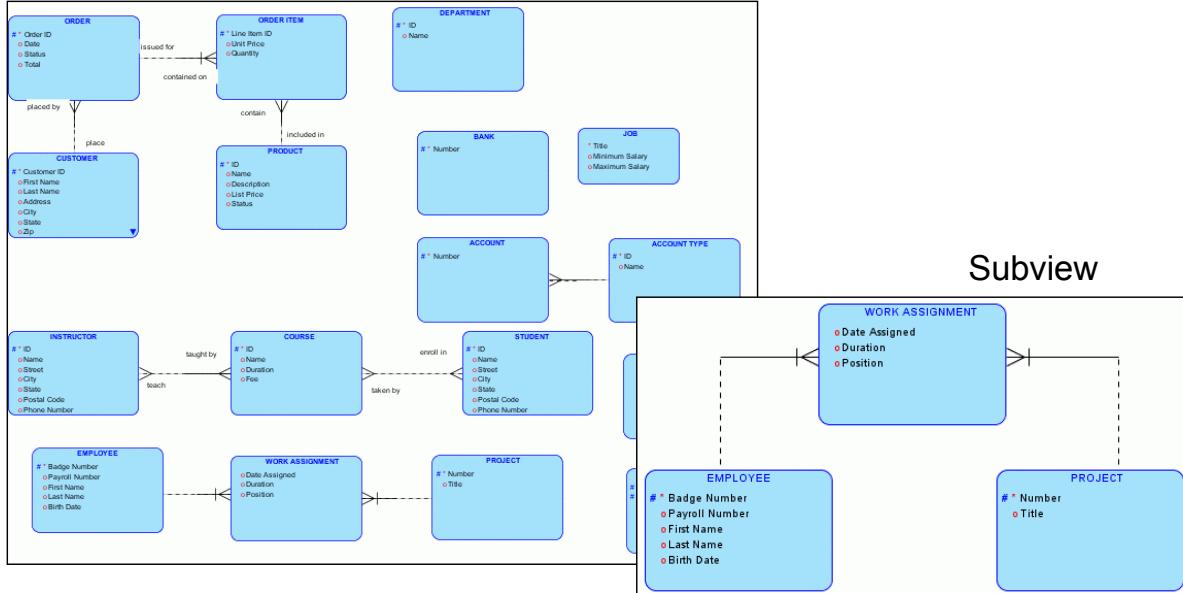
To align objects left or top, perform the following:

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

The objects will now align.

What Is a Subview?

A subview is a section of the logical model.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

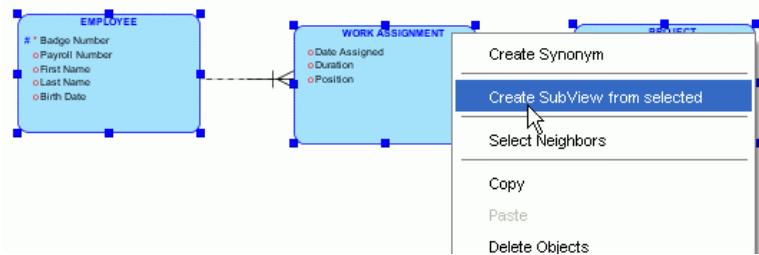
What Is a Subview?

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

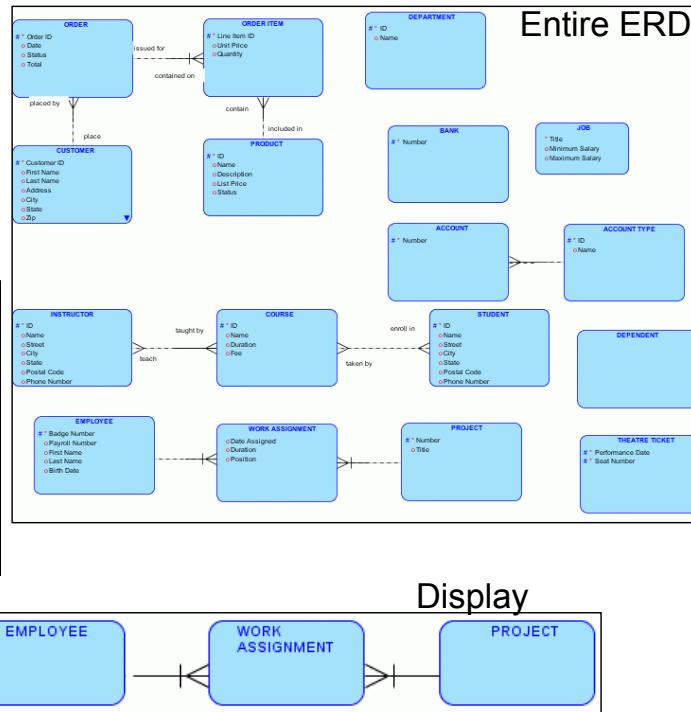
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Creating a Subview

To create a subview containing specific entities, you can select the desired entities in the logical model diagram, right-click, and select “Create Subview from selected.”

What Is a Display?

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

What Is a Display?

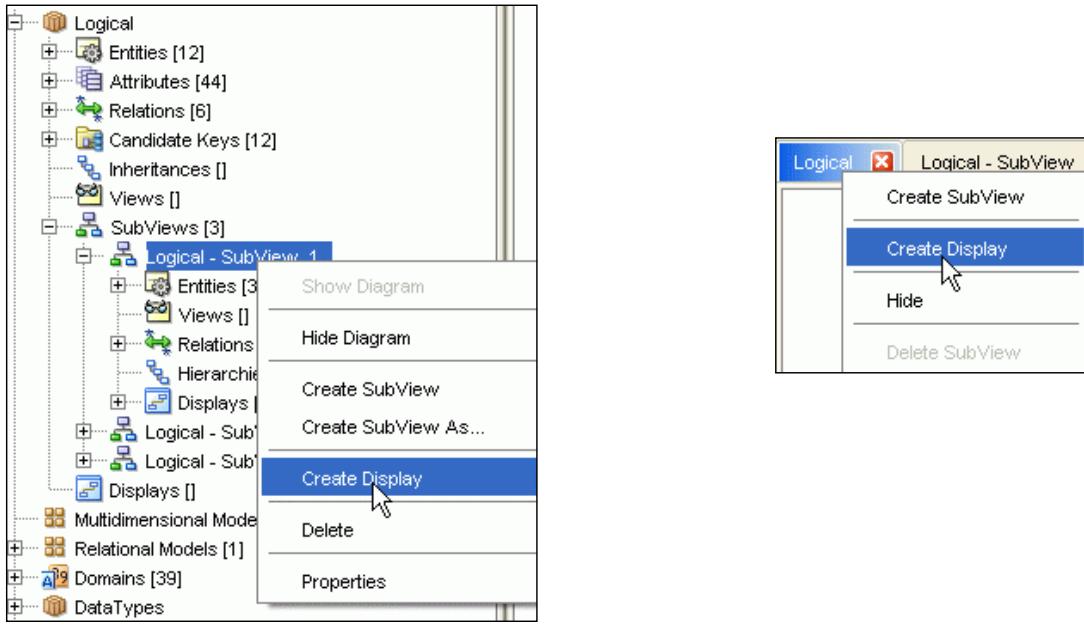
A display is an alternative view of the same objects in a subview or the entire ERD. For example, you might have one subview with three displays:

- 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 © 2010, Oracle and/or its affiliates. All rights reserved.

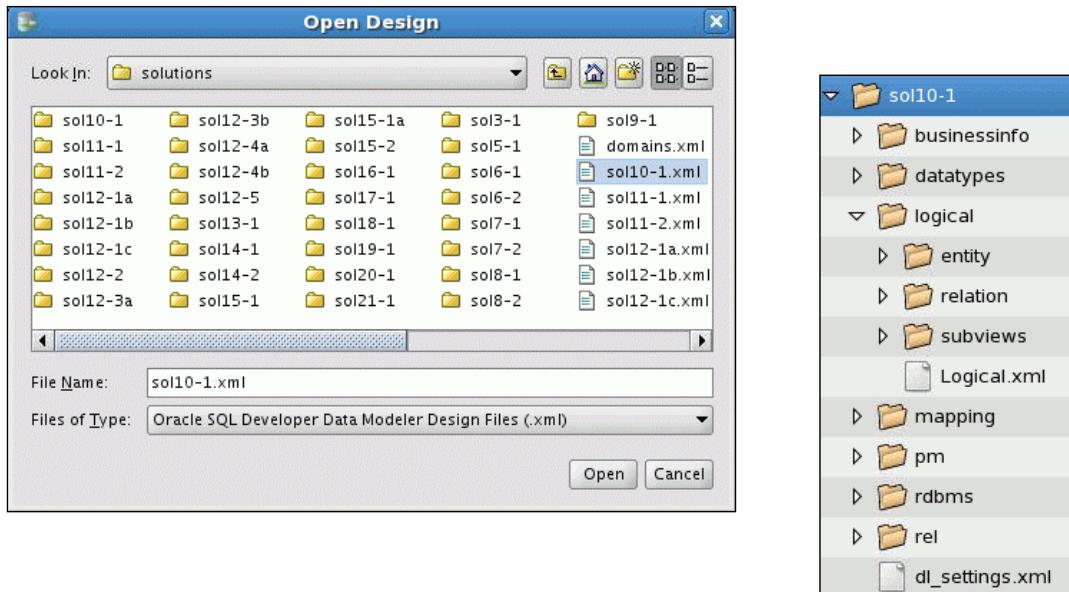
ORACLE

Creating a Display

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 an .xml file and a directory with data modeling objects.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

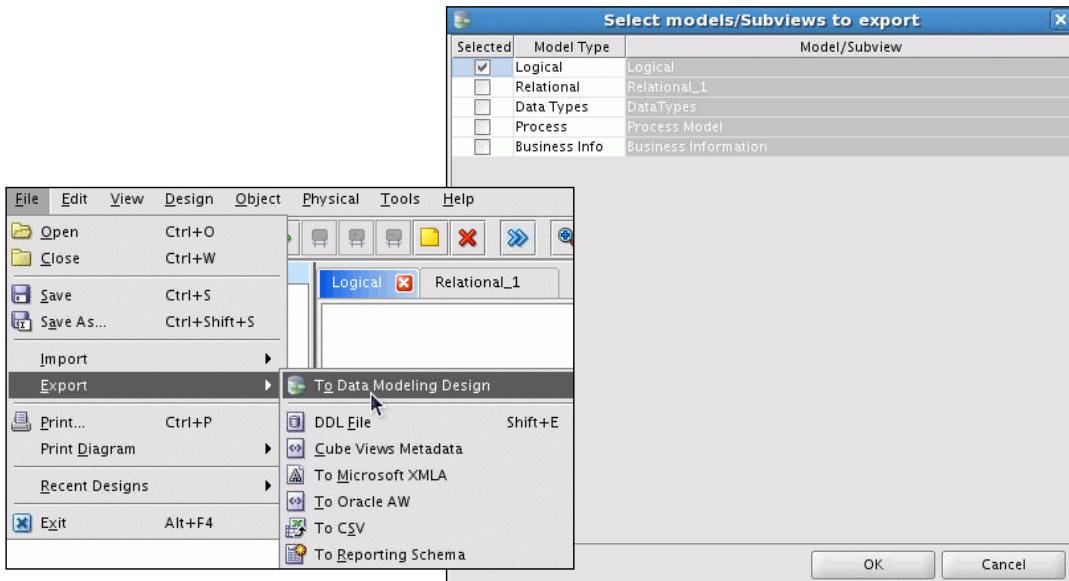
Opening and Saving a Model

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

The model is saved as an XML file and for each XML file, there is a directory with the same name that contains directories for each model type.

Exporting a Model

Select which models to export.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Exporting a Model

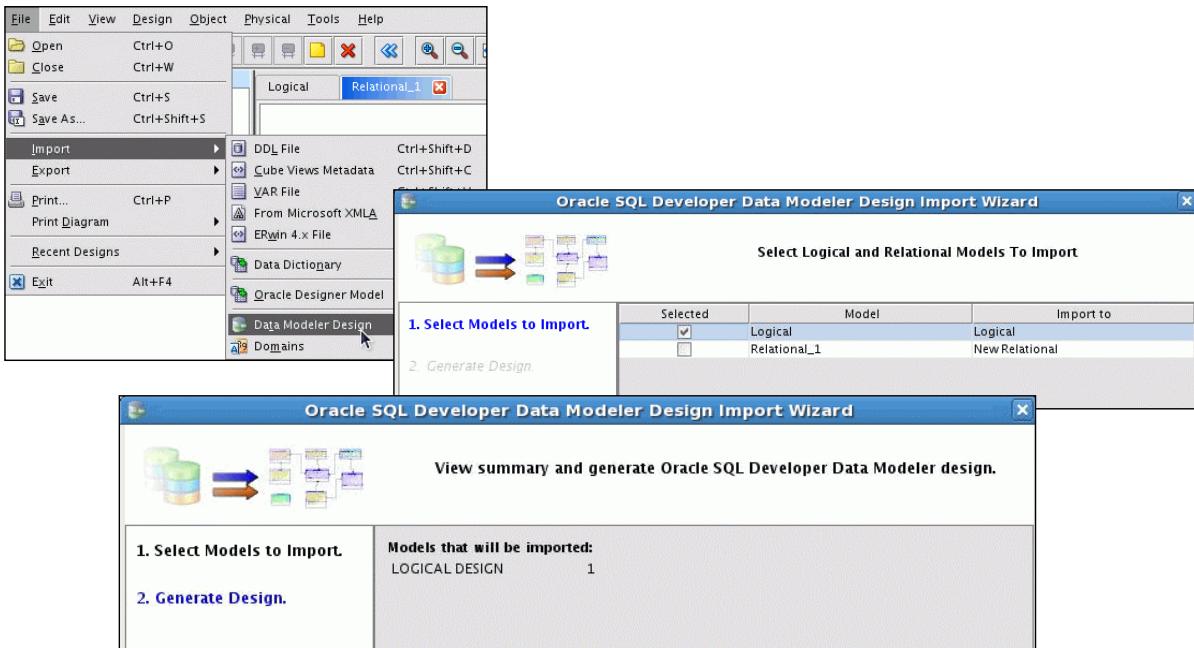
If you want to save certain models of the entire design, you can use the export function.

To export the model, perform the following:

1. Select File > Export > To Data Modeling 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



Importing a Model

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

To import the model, perform the following:

1. Select File > Import > Data Model Design. The Oracle SQL Developer Data Modeler Design Import Wizard opens.
2. In the first window of the wizard, select the models that you want to import, and click Next.
3. 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? (Select all that apply.)

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: a, b

Quiz

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

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



Copyright © 2010, 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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to build an ERD in Oracle SQL Developer Data Modeler.

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 Exercise from Lesson 8: Student/Instructor scenario
 - Practice 8-1: Hotel Scenario
 - Practice 8-2: DVD Membership Scenario
- Creating a subview and a display for each scenario



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

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

In this practice, you build ERDs for the scenarios from the previous lessons.

10

Validating Your Entity Relationship Diagram

ORACLE®

Copyright © 2010, 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
- Define naming standards, glossaries, and abbreviations
- Supplement the ERD with useful information



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

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

ERD Checklist

Neatness

- Align entity boxes.
- Draw relationship lines straight.
- Use white space to avoid congestion.
- Avoid many parallel lines.

Text

- Make all text unambiguous.
- Avoid abbreviations and jargon.
- Align text horizontally.
- Put relationship names at the ends of the line and opposite sides of the line.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ERD Checklist

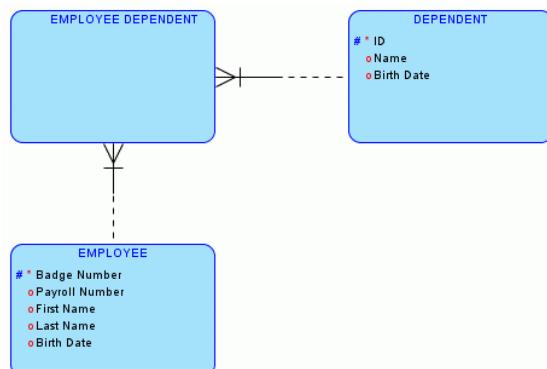
Keep the following rules in mind when verifying your ERD:

- 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 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 and 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.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ERD Checklist (continued)

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 © 2010, Oracle and/or its affiliates. All rights reserved.

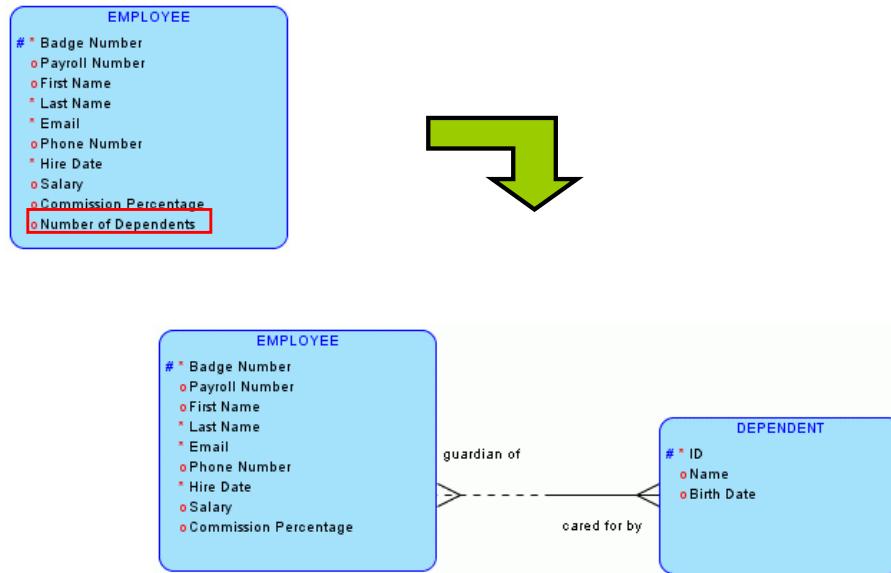
Attribute Rules

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 multi-valued 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.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Distinguishing Attributes and Entities

If an attribute has attributes of its own, it is entity. In the example in the slide, the "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)



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Distinguishing Attributes and Entities (continued)

The table in the slide shows a comparison between entities and attributes. You can validate the objects on your ERD by asking yourself the information supplied in the table. Remember that all entities are nouns, but not all nouns are entities. Do not disqualify a candidate entity too quickly. Attributes for that entity may appear later.

Attribute Optionality

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

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

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Attribute Optionality

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.

Naming Standards

Establishing a Naming Pattern or Template

Word Classification Types	Definition	Example
Prime word	Describes the object being defined; often used in searches	CUSTOMER
Class word	Specify the type of information maintained about the object	ADDRESS
Modifier	Gives additional information about class and prime words; often an adjective	CUSTOMER DELIVERY
Qualifier	Type of modifier, used with a class word to describe a characteristic	WEEKS, HOURS

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Naming Standards

When establishing a naming standard for objects, you should first establish a naming pattern or template. Typically you might define name patterns for entities, attributes, tables, columns, and domains. This pattern provides a structure for each name.

In Oracle SQL Developer Data Modeler, there are four word classification types:

- **Prime word:** Identifies the object or element being defined. Typically, these objects represent a person, place, thing, or event about which an organization wishes to maintain information. Prime words may act as primary search identifiers when querying a database system and provide a basic list of keywords for developing a general-to-specific classification scheme based on business usages. Examples: CUSTOMER, EMPLOYEE
- **Class word:** The most important noun in a data element name. Class words identify the use or purpose of a data element. Class words designate the type of information maintained about the object (prime word) of the data element name. Example: ADDRESS, EMAIL
- **Modifier:** Gives additional information about the class word or prime word. Modifiers may be adjectives or nouns. DELIVERY (as in Customer Delivery Address) is an example of a modifier. Examples: ANNUAL, QUARTERLY, MOST, LEAST
- **Qualifier:** A special kind of modifier that is used with a class word to further describe a characteristic of the class word within a domain of values, or to specify a type of information that can be attached to an object. Examples: FEET, METERS, SECONDS, WEEKS

Naming Standards

Entity Examples

- | | |
|-------------------|-------------------|
| • LOCATION | {Prime} |
| • JOB HISTORY | {Prime}{Class} |
| • ORDER | {Prime} |
| • EXEMPT EMPLOYEE | {Modifier}{Prime} |

Attribute Examples

- | | |
|----------------|----------------|
| • Email | {Class} |
| • Order Date | {Prime}{Class} |
| • Order Total | {Prime}{Class} |
| • Phone Number | {Prime}{Class} |

ORACLE

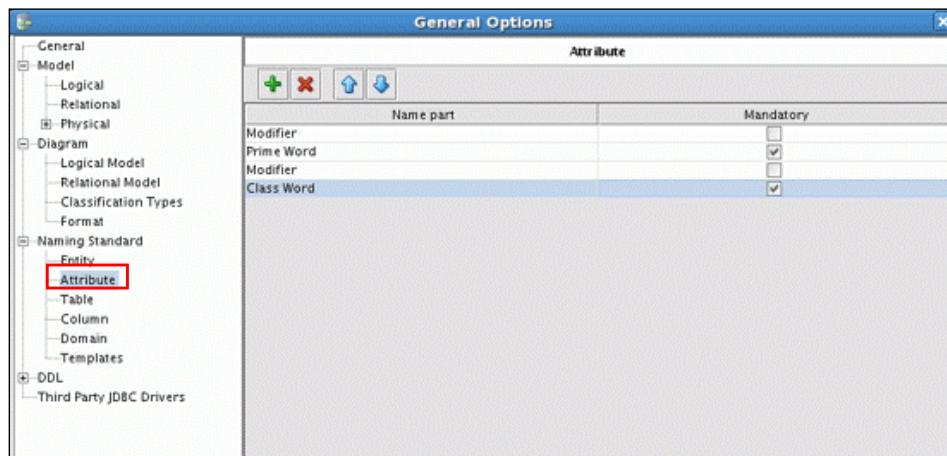
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Naming Standards (continued)

The examples in the slide show how entities and attributes equate to word classification types.

Defining Naming Standards

Tools > General Options – Naming Standards > Attribute



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Defining Naming Standards

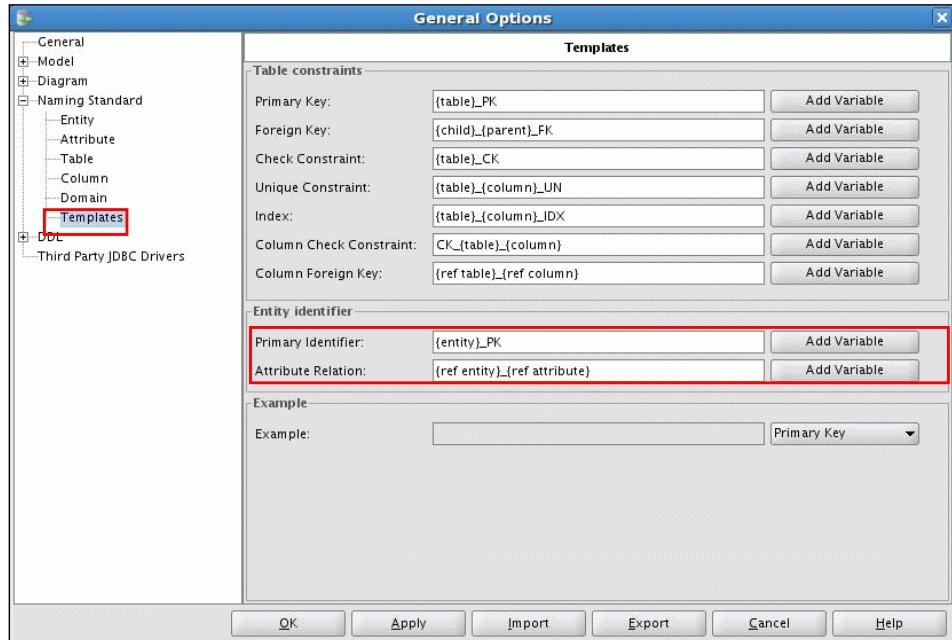
To define a naming standards pattern or template, perform the following steps:

1. Select Tools > General Options and then expand Naming Standards in the left navigator.
2. Select either Entity or Attribute, and then click the Add icon.
3. Select the word classification type and select whether the word type should be optional.

Note that the naming standard you set here is used only when running the design rules. It will not prevent you from creating an attribute that does not follow the naming standard.

Defining Naming Standards

Tools > General Options – Naming Standards > Templates



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

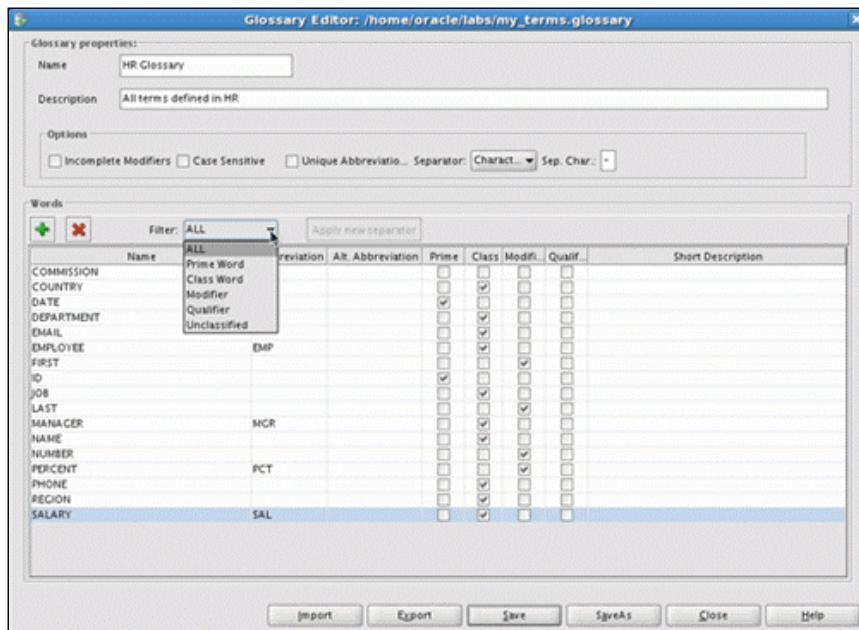
Defining Naming Standards (continued)

You can also define the template naming standard for particular objects. For the entity object, you can specify the naming template for Primary Identifier and Attribute Relation. In the example in the slide, the name of the primary identifier will always be generated as the name of the entity concatenated with "_PK". So if the entity is named, CUSTOMER, the name of the primary identifier will be CUSTOMER_PK.

Note that the other templates on this page will be discussed in later lessons on relational modeling.

Using a Glossary

Glossaries are used for name validation.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Using a Glossary

Oracle SQL Developer Data Modeler supports the use of a glossary where all terms (parts) used in names are defined. The glossary is used during the validation process. If no glossary is defined, no name translation occurs.

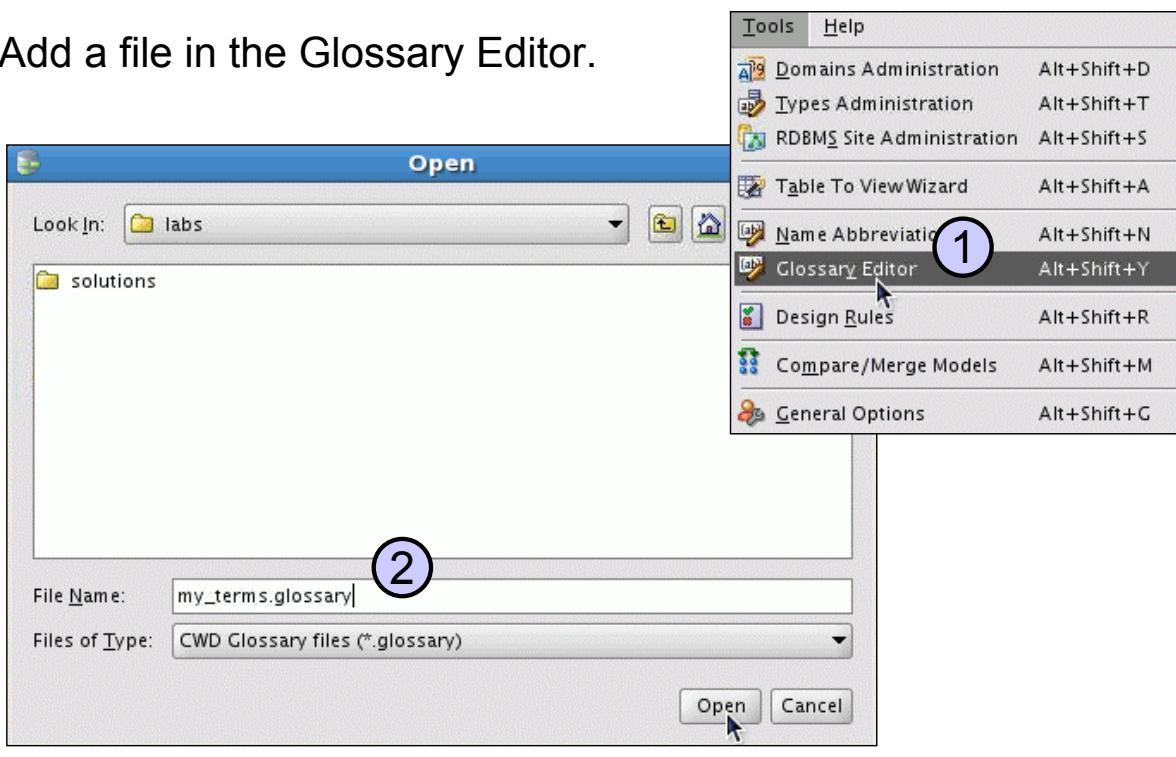
You can define one or more glossaries as validation glossaries. If there is more than one glossary, a name is considered to be valid if it can be validated using any one of the defined glossaries. You can use different glossaries to represent separate domains of interest. However, using many glossaries together can lead to unpredictable results, especially when abbreviations are used in the validation process. For example, AP could be "Accounts Payable" but also can match to "Actual Placement" defined in another glossary.

In addition to a name and description, you can also set the following options:

- **Incomplete Modifiers:** It is assumed that all terms used in names will be defined in the glossary; however, if you select this option, it is not mandatory that modifiers and qualifiers be defined.
- **Case Sensitive:** Defines whether validation is case-sensitive. For example, Code and CODE are different when this option is checked.
- **Unique Abbreviations:** Uniqueness of abbreviations is not forced, therefore an abbreviation can be used for all forms of a single word. For example, ADMIN equals Administrator, Administration, or Administrative. These names must be maintained carefully because the name validation and translation treat all terms with the same classification settings.

Creating a Glossary

Add a file in the Glossary Editor.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

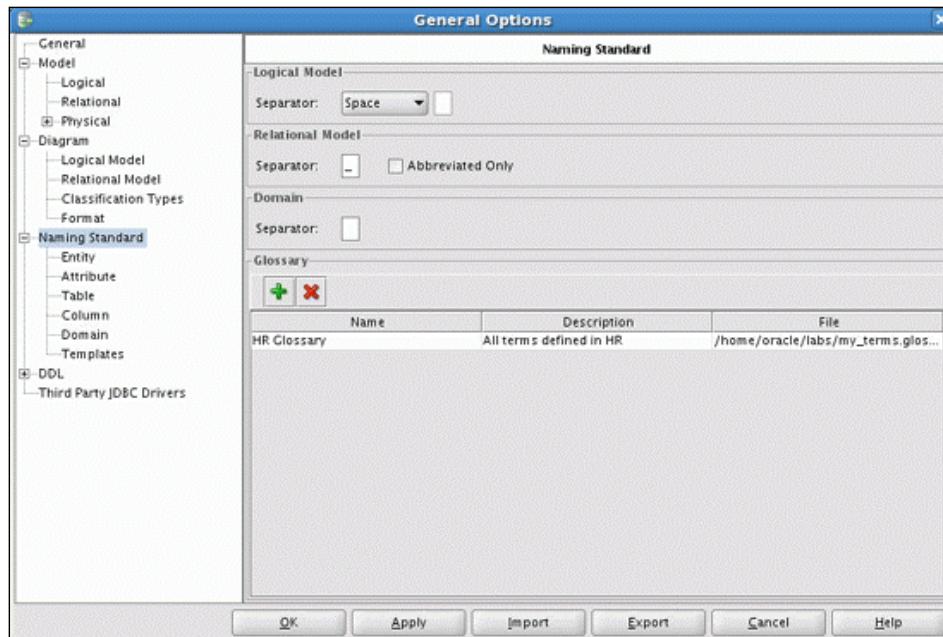
Creating a Glossary

To create a glossary, you must create a file that the glossary data will be stored in. Perform the following steps:

1. Select Tools > Glossary Editor.
2. Enter a name for your file and click Open. Note that the glossary will be created with a .glossary file extension so do not specify that in the name.
3. Enter a name and description for your glossary.
4. To add words to the glossary, click the Add icon and double-click in the Name field to enter the name.
5. Enter an abbreviation for each word, if desired.
6. Specify the word's word classification type. The word can be of more than one type.

Applying the Glossary to the Naming Standards

Tools > General Options > Naming Standards



ORACLE

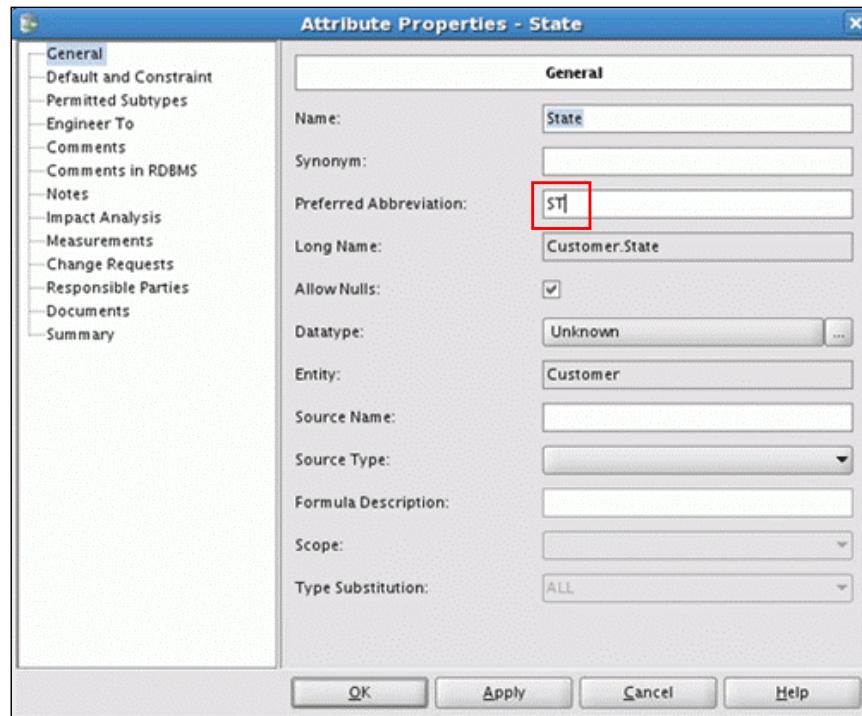
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Applying the Glossary to the Naming Standards

After your glossary has been created, you must apply it to the Naming Standards > Glossary page. Perform the following steps:

1. Select Tools > General Options and select Naming Standard.
2. Click the Add icon and select the glossary file that you created previously.

Defining Abbreviations



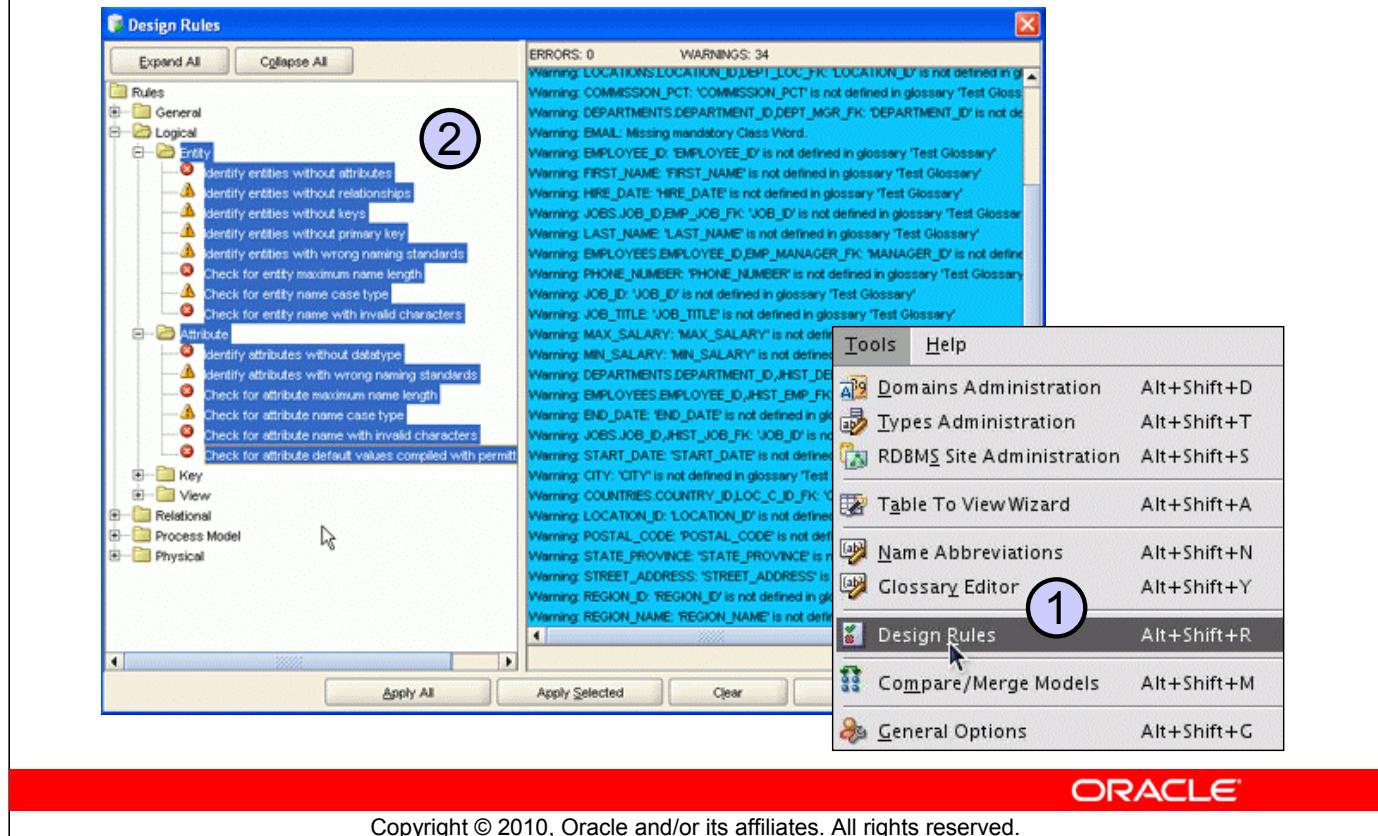
ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Defining Abbreviations

Oracle SQL Developer Data Modeler can use defined abbreviations when engineering between the logical and relational models. You can define abbreviations at the attribute and entity levels. These abbreviations can then be used when an attribute or entity name is transformed to a column or table name. In the screenshot in the slide, the abbreviation for the State attribute is ST.

Applying Design Rules



Applying Design Rules

Oracle SQL Developer Data Modeler provides a set of predefined design rules. You can validate your models by applying these design rules to catch possible errors. In the screenshot in the slide, you see a set of design rules for each object.

To apply a design rule, perform the following steps:

1. Select Tools > Design Rules.
2. Select a design rule and click Apply Selected (or click Apply All to apply all the rules). Notice that a list of violations for any of the selected rules is shown on the right.

Adding Additional Information to the ERD

The screenshot shows the Oracle Data Modeler interface. On the left is a tree view of the model structure under 'demo-10'. Under 'Business Information', there are several categories: Documents [0], Responsible Parties [1] (selected), Contacts [1], Emails [1], Locations [1], Telephones [2], and Resource Locators [0]. The right panel is titled 'Entity Properties - ORDER'. It has two main sections: 'Responsible Parties' and 'Manager'. The 'Responsible Parties' section contains a table with one row for 'Manager'. The 'Manager' row has a 'Details' button (pencil icon) and a 'Delete' button (red X). The 'Manager' row is highlighted. The left sidebar lists various properties for the ORDER entity, including General, Attributes, Unique Identifiers, Relationships, Volume Properties, Engineer To, Comments, Comments in RDBMS, Notes, Impact Analysis, Measurements, Change Requests, Responsible Parties (which is selected and highlighted in blue), Documents, and Summary.

Business Information

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

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.

Quiz

Which of the following should you examine 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.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: a, c

Quiz

Which of the following is a class word? (Select all that apply.)

- a. Order entity
- b. Status attribute
- c. Customer entity
- d. Description attribute



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: b, d

Order and Customer are prime words.

Summary

In this lesson, you should have learned how to:

- Apply diagram layout and attribute rules
- Distinguish entities from attributes
- Evaluate attribute optionality
- Define naming standards, glossaries, and abbreviations
- Supplement the ERD with useful information



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 10-1 Overview: Develop and Validate Your ERD

In this practice, you create an ERD for the Oracle User's Group and validate the model by using the rules discussed in this lesson.

Utilizing Advanced Data Modeling Techniques

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Overview

In this unit, you examine the following areas:

- Lesson 11: Normalizing your Data Model
- Lesson 12: Validating Relationships
- Lesson 13: Adding and Using Data Types
- Lesson 14: Putting It All Together



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

11

Normalizing Your Data Model

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Normalize your ERD to third normal form



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

In this lesson, you learn how to normalize your ERD to third normal form.

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 upon 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 © 2010, Oracle and/or its affiliates. All rights reserved.

What Is Normalization?

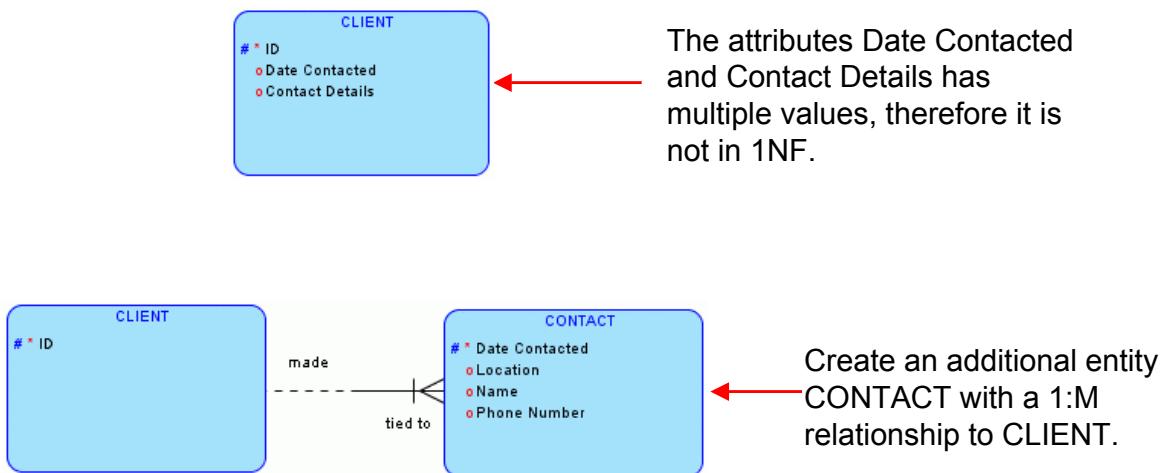
Normalization is a relational database concept. However, if you have created a validated ERD, the tables created during the design will conform to the rules of normalization. Each formal normalization rule from the relational database design has a corresponding data model interpretation. The interpretations 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.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

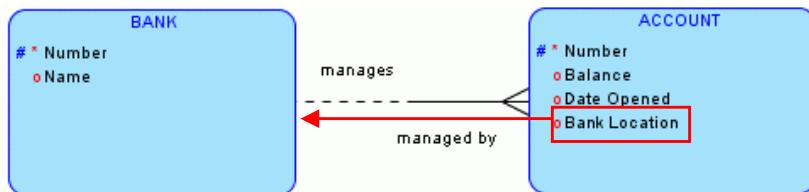
First Normal Form (1NF)

First normal form validates that each attribute has a single value for each occurrence of the entity. There should be no attribute that has a repeating value. In the example in the slide, the Date Contacted and Contact Details attributes could have more than one value for each CLIENT ID. It is not in 1NF. You must perform the following:

- Create another entity and move the attributes that are repeating to the new entity.
- Create an identifying 1:M relationship with the new entity.

Second Normal Form (2NF)

An attribute must be dependent upon its entity's entire unique identifier.



The Bank Location attribute is dependent on BANK rather than on ACCOUNT; therefore, it is not in 2NF.
Move the attribute to the BANK entity.

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

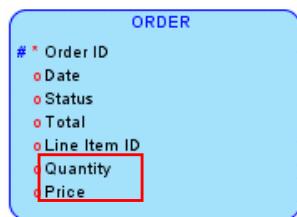
Second Normal Form (2NF)

Second normal form validates that each attribute is dependent upon its entity's unique identifier. Each specific instance of the UID must determine a single instance of each attribute. Each attribute is not dependent upon only part of its entity's UID.

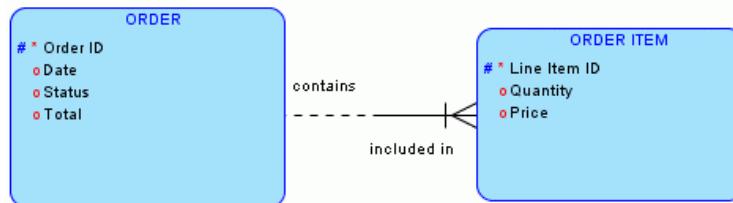
In the example in the slide, the Bank Location attribute is dependent on the BANK entity not the ACCOUNT entity; therefore, the attribute must be moved to the BANK entity.

Third Normal Form (3NF)

Each attribute depends only on the UID of its entity.



The Quantity and Price attributes are dependent on the Order ID (UID) and Line Item ID (non-UID), therefore it is not in 3NF.



Create a new ORDER ITEM entity, move the Line Item ID, Quantity, and Price attributes to the new entity, and create an identifying relationship.

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Third Normal Form (3NF)

Third normal form validates that each attribute depends only on the UID of its entity (and on nothing else).

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 its own 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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: b

Normalization Example: Unnormalized Data

Ordered by:

Customer ID :

Customer Name :

Address Line 1 :

Address Line 2 :

Address Line 3 :

City, State ZIP : ,

Ship to:

Ship Via :

Name :

Address Line 1 :

Address Line 2 :

Address Line 3 :

City, State ZIP : ,

Order ID

Order Date

<u>Item ID</u>	<u>Color</u>	<u>Size</u>	<u>Quantity</u>	<u>Description</u>	<u>Price</u>

Order Total:

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Normalization Example: Unnormalized Data

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, not 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



ORACLE

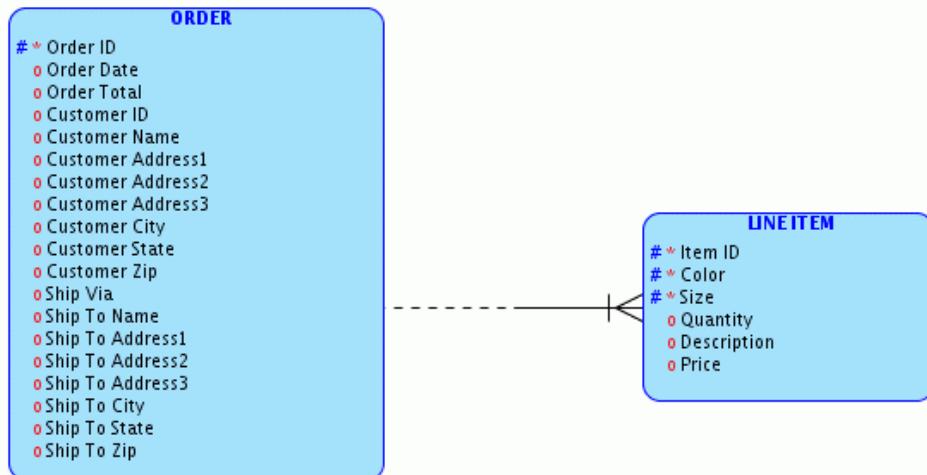
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

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 the 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

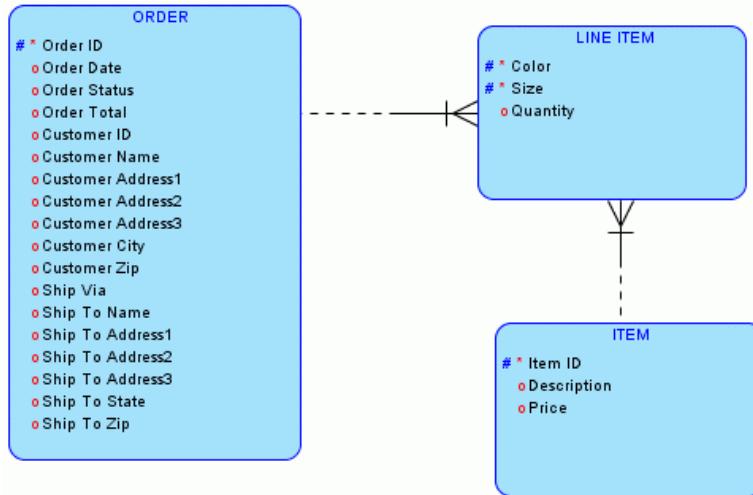
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Normalization Example: Transforming to First Normal Form (continued)

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

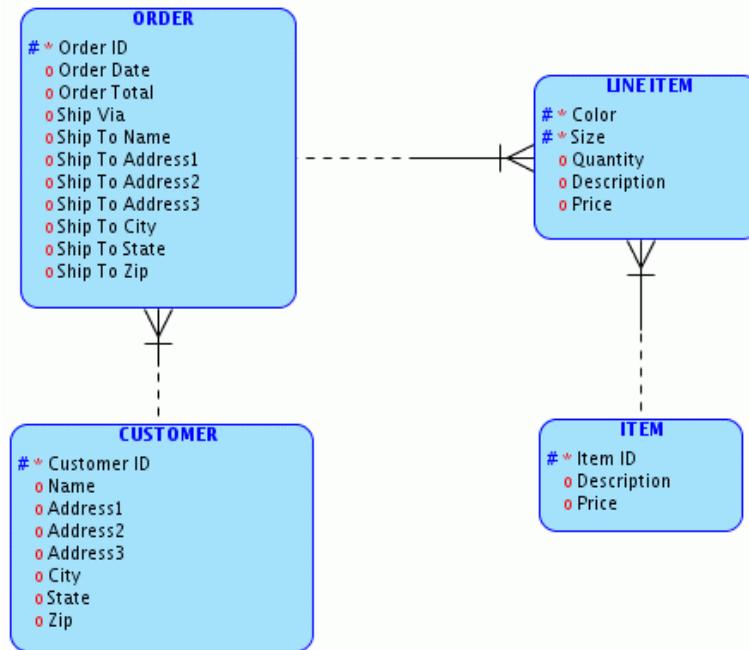
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Normalization Example: Transforming to Second Normal Form

To transform the model to second normal form (2NF), you must remove attributes that are dependent upon 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Normalization Example: Transforming to Third Normal Form

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 the non-directly-related 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

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 the following topics:

- Evaluating an ERD and normalizing it to third normal form



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 11-1 Overview: Normalize an ERD

In this practice, you normalize an ERD to third normal form.

Practice 11-2 Overview: Validate ERD for Normalization

This practice covers the following topics:

- Normalizing unnormalized data to third normal form



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 11-2 Overview: Validate ERD for Normalization

In this practice, you normalize unnormalized data to third normal form.

12

Validating Relationships

ORACLE

Copyright © 2010, 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 © 2010, Oracle and/or its affiliates. All rights reserved.

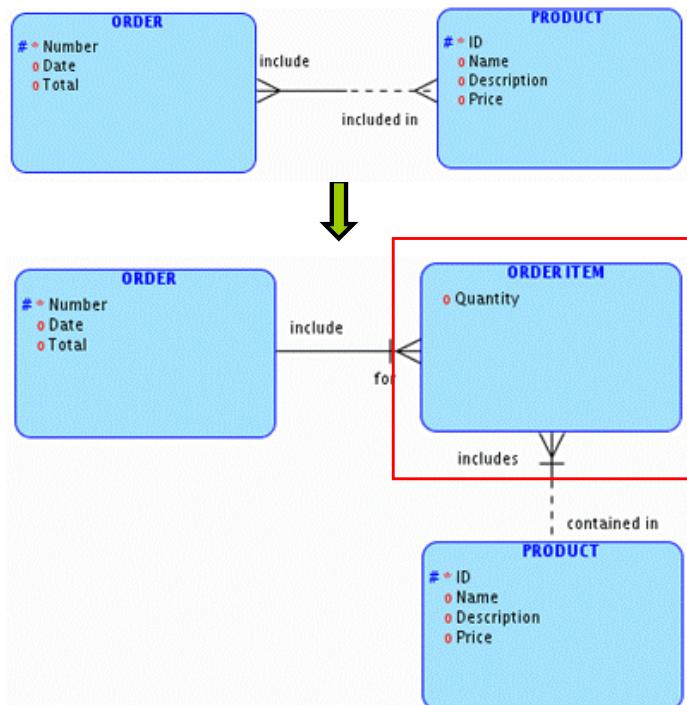
Objectives

In this lesson, you examine some of the more advanced data modeling techniques associated with relationships.

Resolving M:M Relationships

Attributes only describe entities. 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.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Resolving M:M Relationships

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 was created, ORDER ITEM, which stores additional attributes. The unique identifier will be the relationships between ORDER and PRODUCT.

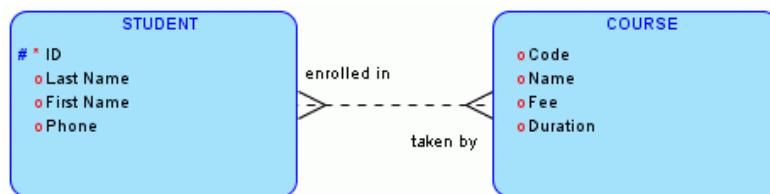
Intersection Entity Characteristics

- 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 originating relationships (identifying relationships).

Note that if you do not have any additional attributes in the intersection entity, you can leave it as an M:M and Oracle SQL Developer Data Modeler will create the intersection table in the relational model. This will be discussed in a later lesson.

Resolving M:M Relationships

How would you resolve the following?



ORACLE

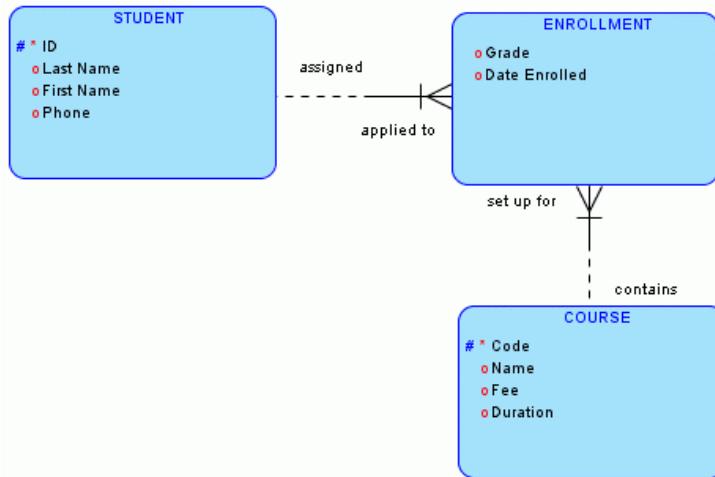
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Resolving M:M Relationships (continued)

How would you resolve an M:M relationship between STUDENT and COURSE?

Resolving M:M Relationships

Create an intersection entity with identifying relationships to the originating entities.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Resolving M:M Relationships (continued)

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 is true? (Select all that apply.)

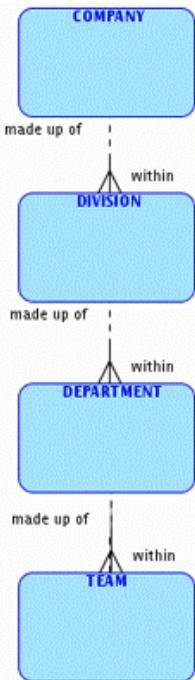
- a. Intersection entities always have additional attributes.
- b. Intersection entities only have identifying relationships as their UID.
- c. An intersection entity resolves an M:M relationship.
- d. Most M:M relationships should be resolved.



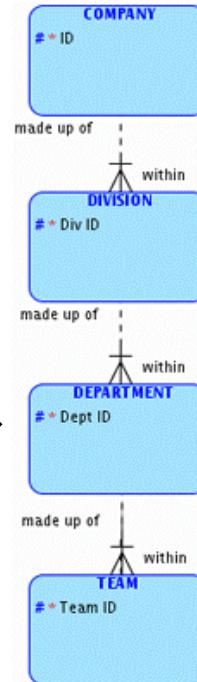
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: c, d

Modeling Hierarchical Data



← Represent hierarchical data as a set of 1:M relationships.



The UIDs for a set of hierarchical entities may be propagated through multiple relationships. →

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

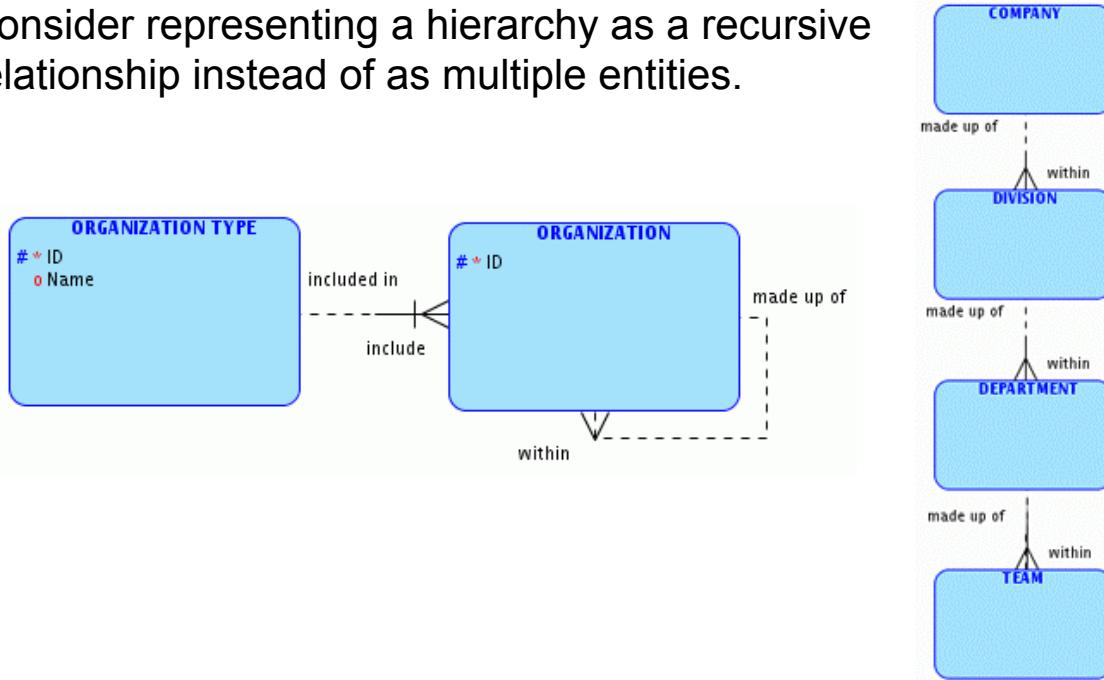
Modeling Hierarchical Data

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 and 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.



ORACLE

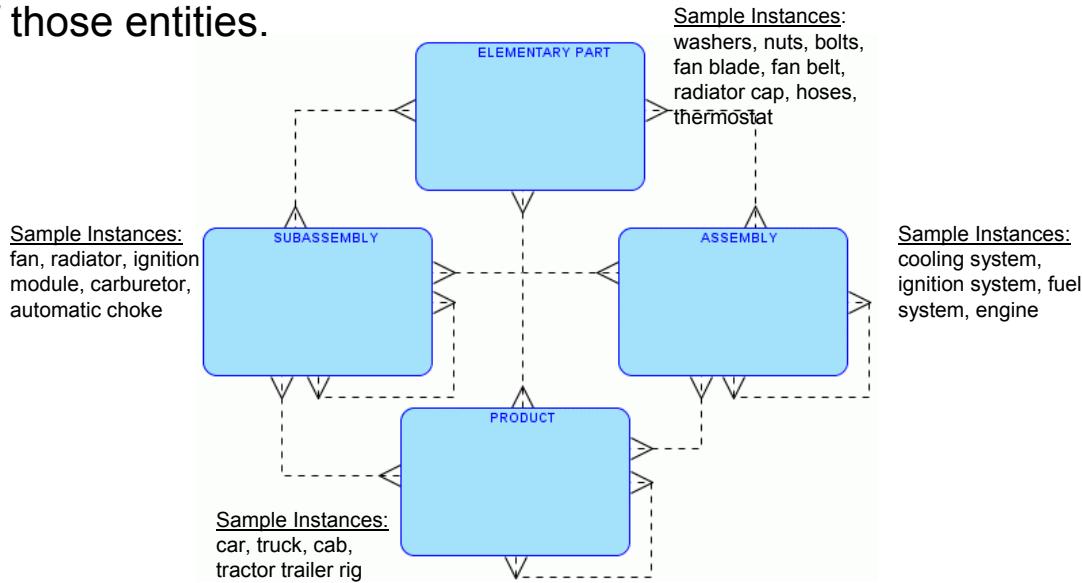
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Examining Recursive Relationships

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. Therefore, **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

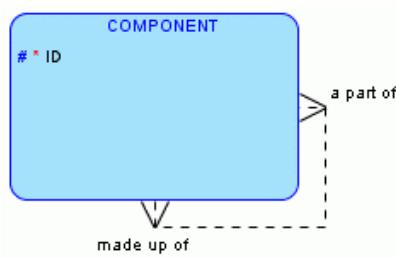
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Examining Recursive Relationships (continued)

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Examining Recursive Relationships (continued)

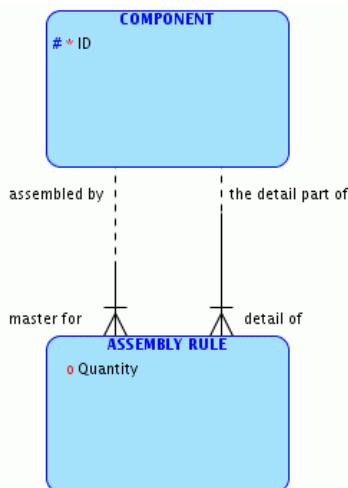
Consider all elementary parts, subassemblies, assemblies, and products as instances of an entity called COMPONENT. Then the complex ERD on the previous page can be remodeled as a simple recursive relationship.

If some of the instance data supplied on the previous page 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Resolving a M:M Recursive Relationship

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

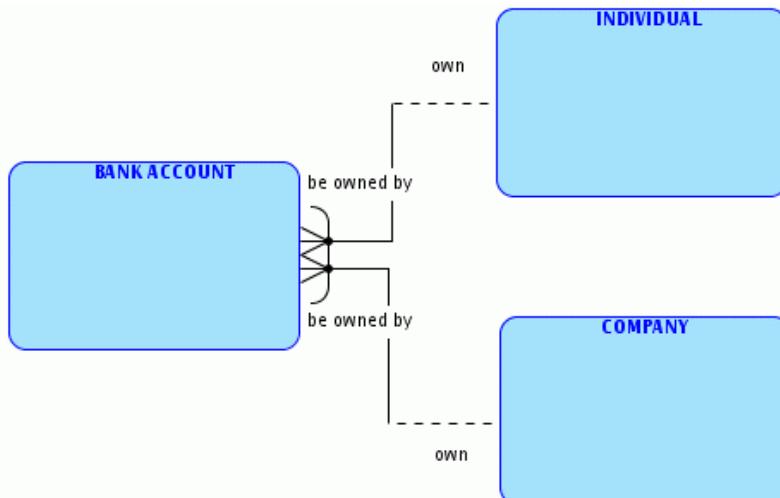


Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: a

Modeling Exclusive Relationships

Modeling two or more mutually exclusive relationships from the same entity using an arc.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

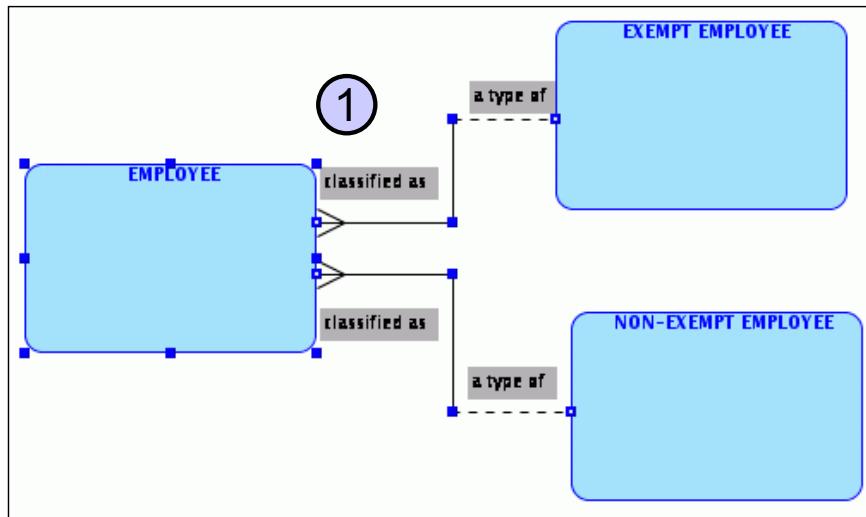
Modeling Exclusive Relationships

An exclusive relationship is 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: 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 only include relationships originating from that entity.
- An entity may have multiple arcs, but a specific relationship can only participate in a single arc.

Creating an Exclusive Relationship in Oracle SQL Developer Data Modeler



ORACLE

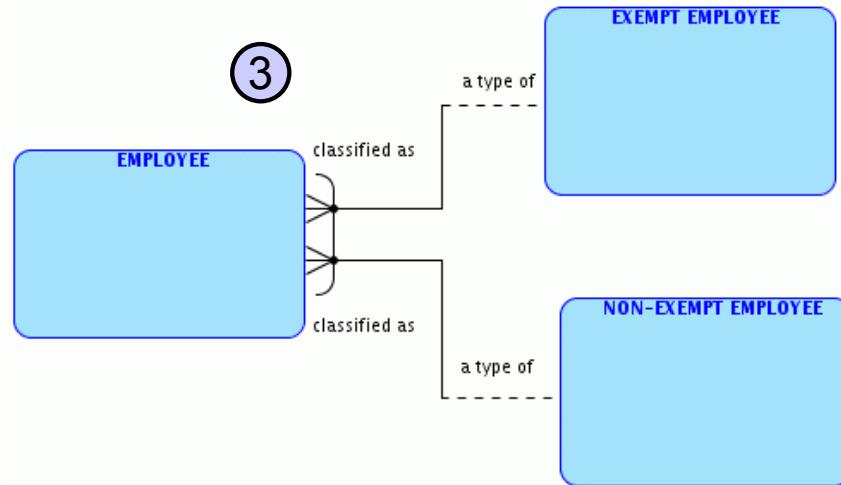
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

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 that you want to create the exclusive relationship on.
2. Select the New Arc icon in the toolbar.

Creating an Exclusive Relationship in Oracle SQL Developer Data Modeler



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Creating an Exclusive Relationship in Oracle SQL Developer Data Modeler (continued)

3. The exclusive relationship is created with the arc.

The business rule for this relationship is read as following: 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.

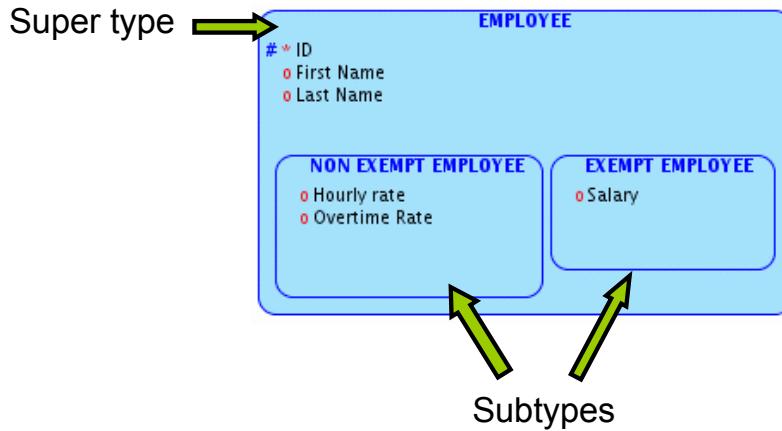


Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: b

Entity Type Hierarchies

Use entity type hierarchies to model exclusive entity types that have common attributes and common relationships.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Entity Type Hierarchies

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 that 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.

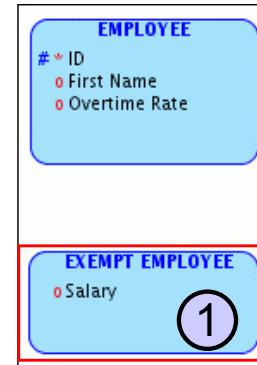
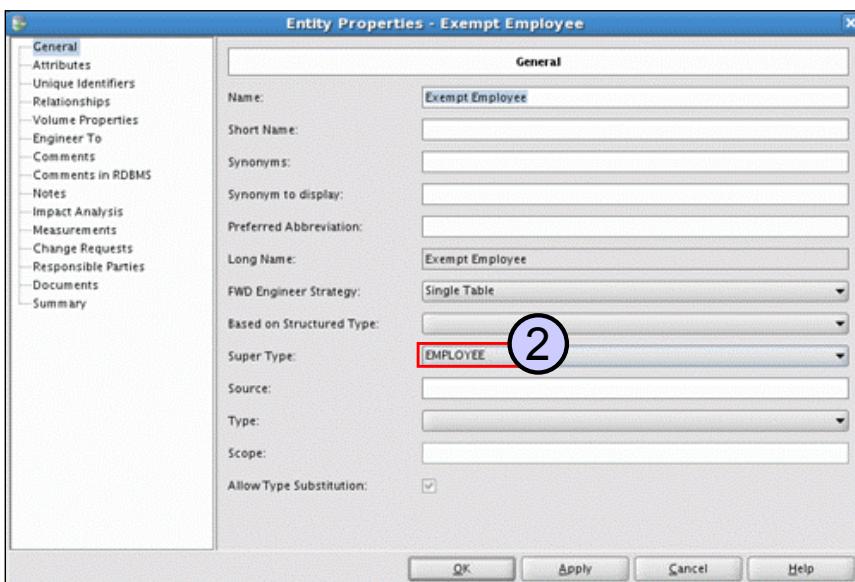
Entity Type Hierarchies (continued)

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 not subtypes in their own right.

Modeling Subtypes in Oracle SQL Developer Data Modeler

Select the super type entity for each subtype.



ORACLE

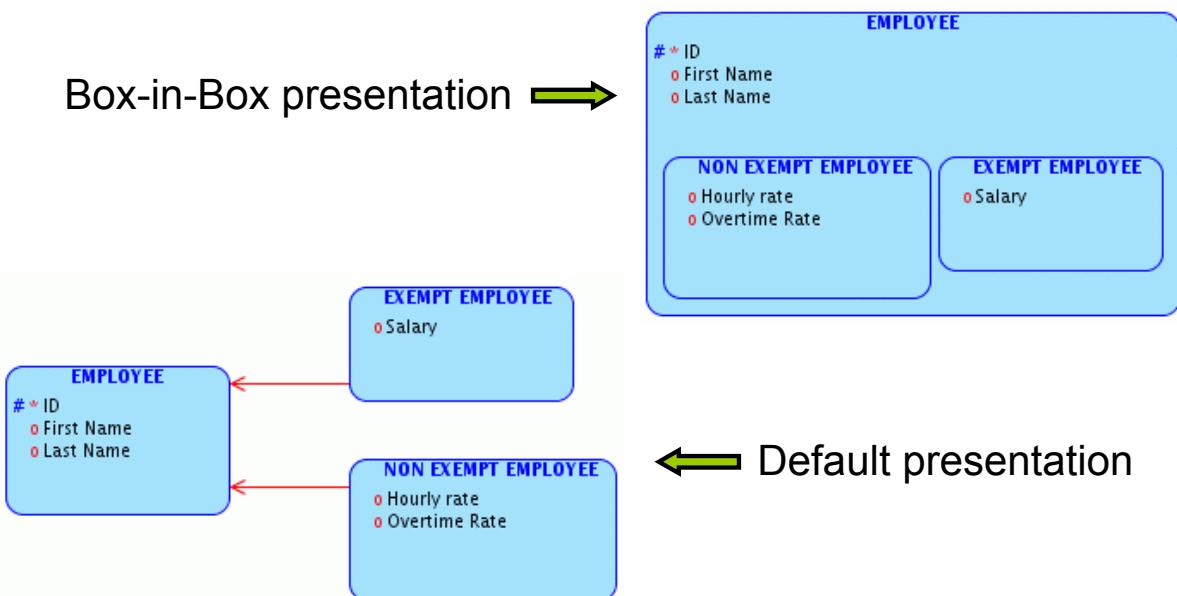
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Modeling Subtypes in Oracle SQL Developer Data Modeler

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 list of 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



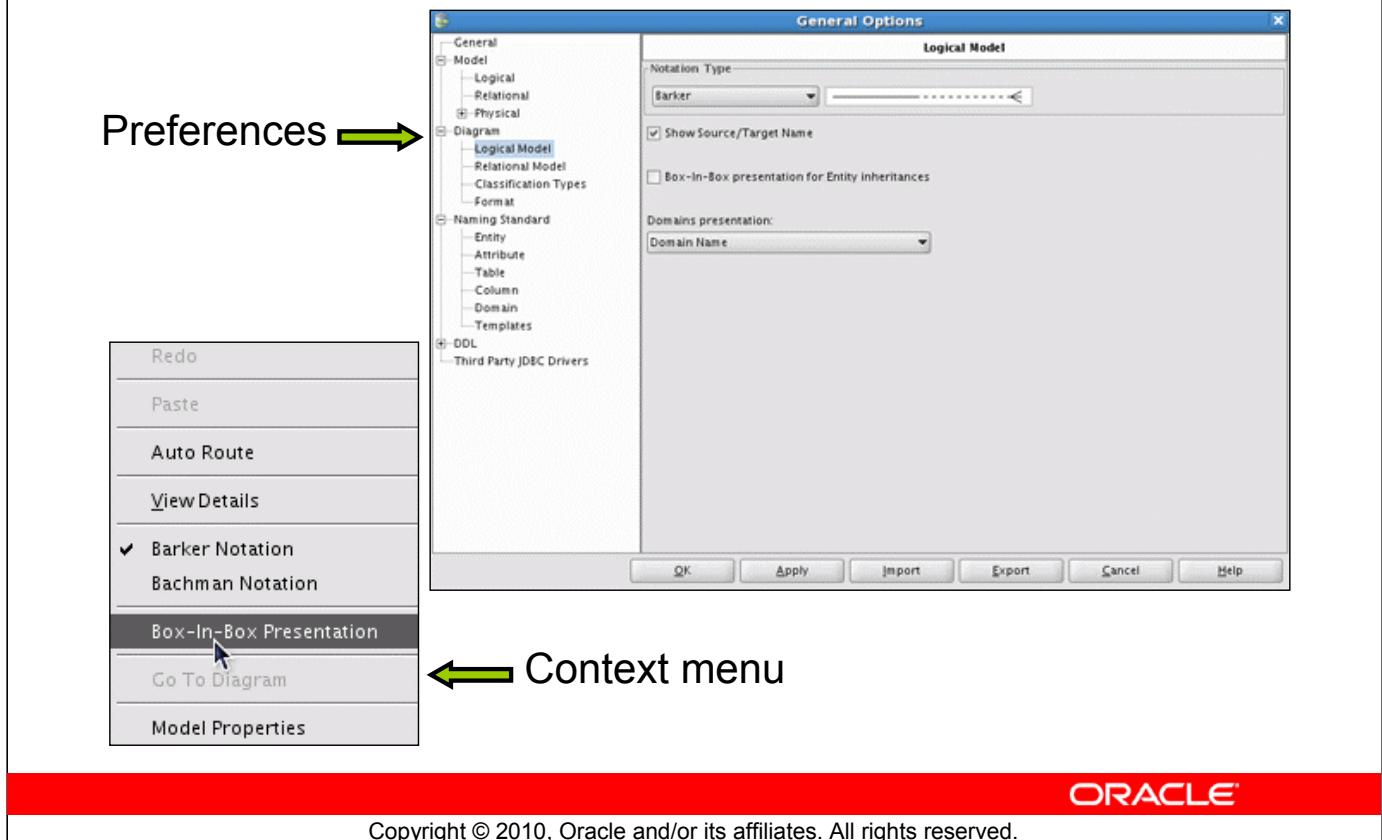
ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Representing Entity Type Hierarchies

There are two ways to represent entity type hierarchies: the default presentation and the Box-in-Box representation.

Changing Preference for Box-in-Box Presentation



Changing Preference for Box-in-Box Presentation

To change the default presentation to Box-in-Box Presentation, select Tools > General Options, expand 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 Box-In-Box Presentation.

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



Copyright © 2010, 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

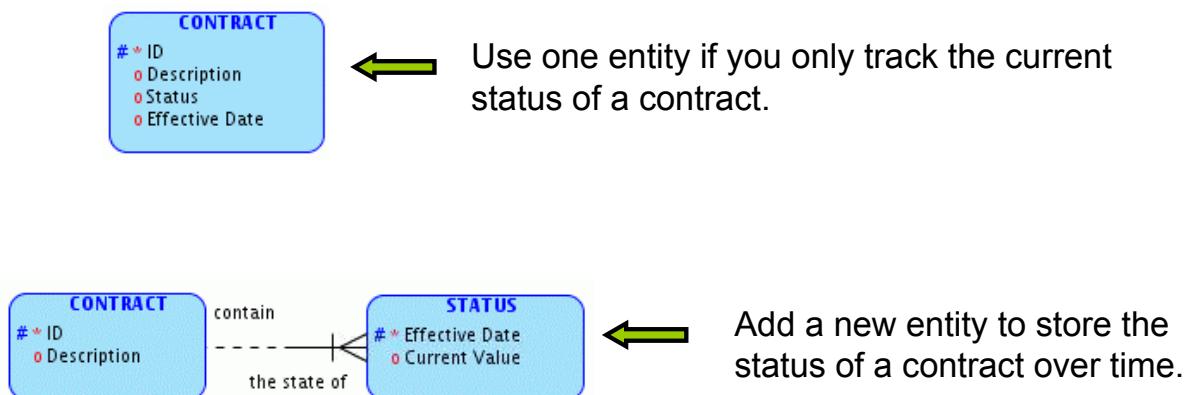
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Model Data Over Time

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.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

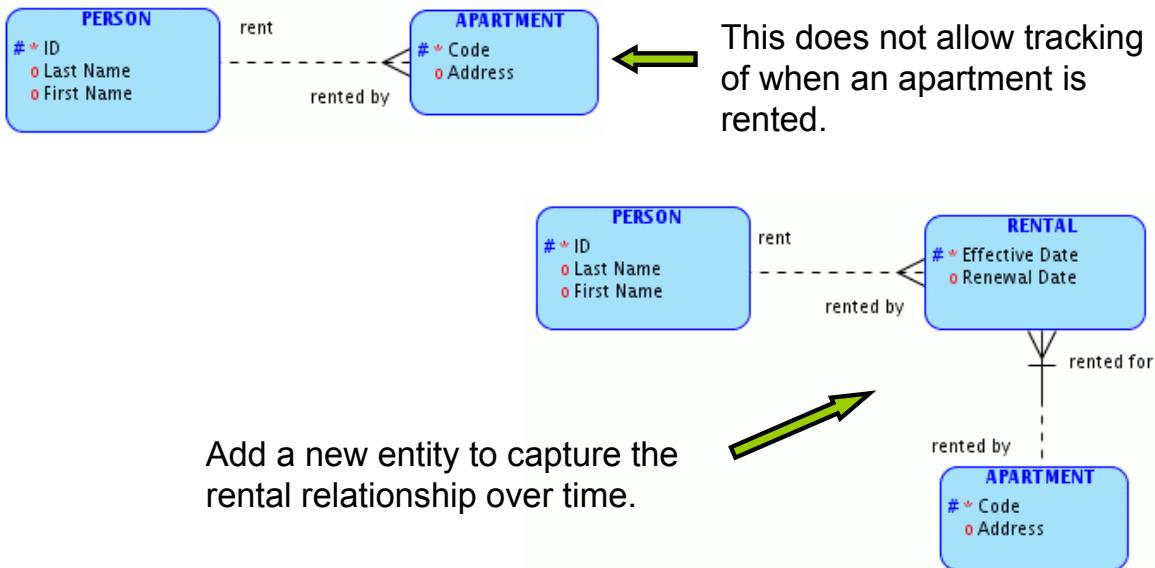
Model Data Over Time (continued)

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. 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, 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.



ORACLE

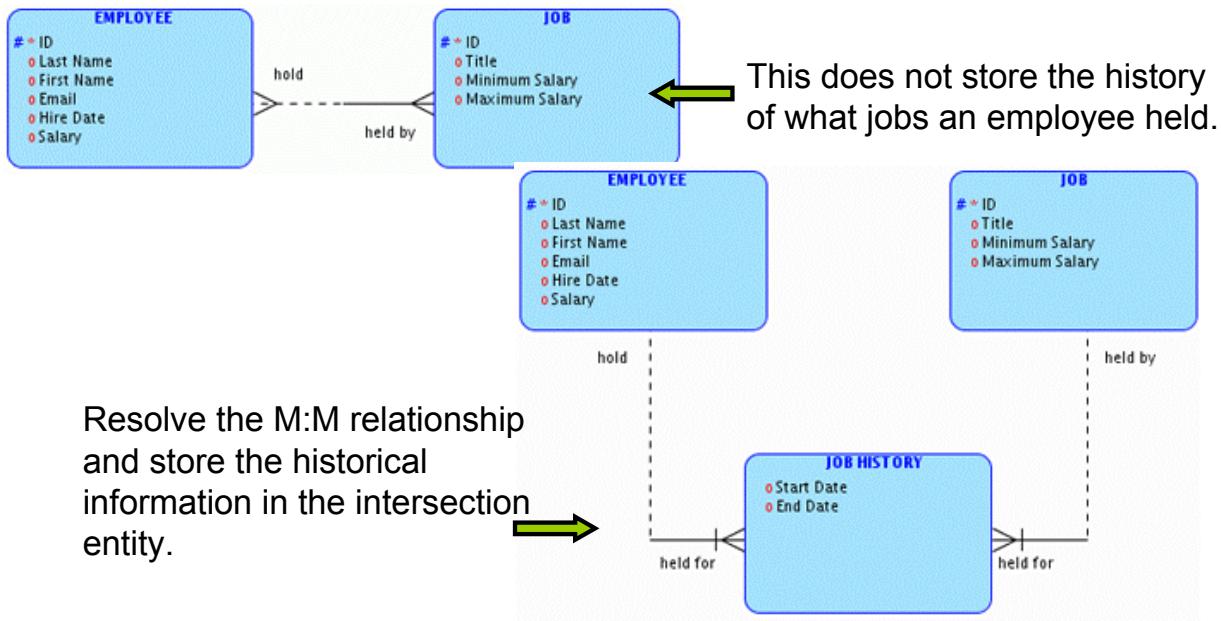
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Model Data Over Time (continued)

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.



ORACLE

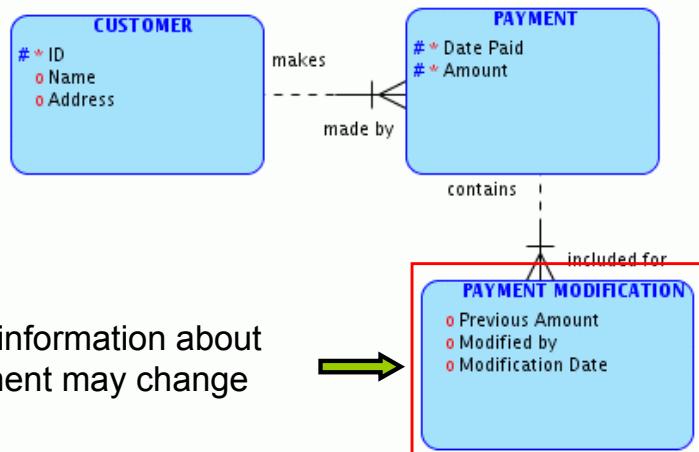
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Model Data Over Time (continued)

Often resolving M:M relationships in an intersection entity allows you to track information about a relationship over time. In the example in the slide, you have an 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.



This stores information about how a payment may change over time.

ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Model Data Over Time (continued)

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. This 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



Copyright © 2010, 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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you examined some of the more advanced data modeling techniques associated with relationships.

Practice 12-1 Overview: Resolve M:M Relationships

This practice covers the following topics:

- Resolving M:M relationships
- Identifying attributes in the intersection entity



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 12-2 Overview: Model Hierarchical Data

This practice covers the following topics:

- Modeling entities, relationships, attributes, and unique identifiers that are hierarchical



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 12-1 Overview: Model Hierarchical Data

In this practice, you model the entities, relationships, attributes, and unique identifiers that are hierarchical by nature.

Practice 12-3 Overview: Model Hierarchical Data and Recursive Relationships

This practice covers the following topics:

- Modeling entities and relationships that are hierarchical and recursive



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 12-4 Overview: Examine Exclusive Relationships

This practice covers the following topics:

- Determining how to model the exclusive relationship



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 12-4 Overview: Examine Exclusive Relationships

In this practice, you determine whether to use an exclusive relationship or an entity type hierarchy to model an exclusive relationship.

Practice 12-5 Overview: Examine Exclusive Relationships

This practice covers the following topics:

- Determining how to model the exclusive relationship from a case scenario



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 12-5 Overview: Examine Exclusive Relationships

In this practice, you build an ERD that contains exclusive relationships.

13

Adding and Using Data Types

ORACLE®

Copyright © 2010, 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

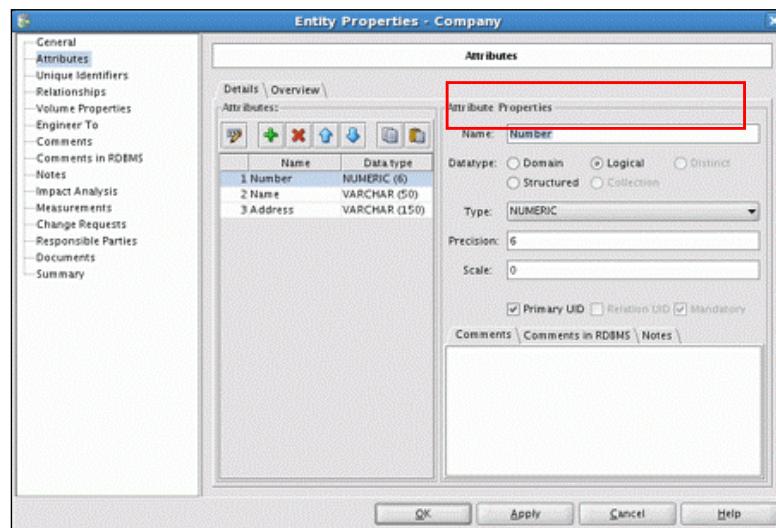
This lesson discusses the various data types and how to create them and use them in your model.

Attribute Data Types

A data type defines the format and length of an attribute.

Data Types Classifications

- Logical
- Domain
- Distinct
- Structured
- Collection



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Attribute Data Types

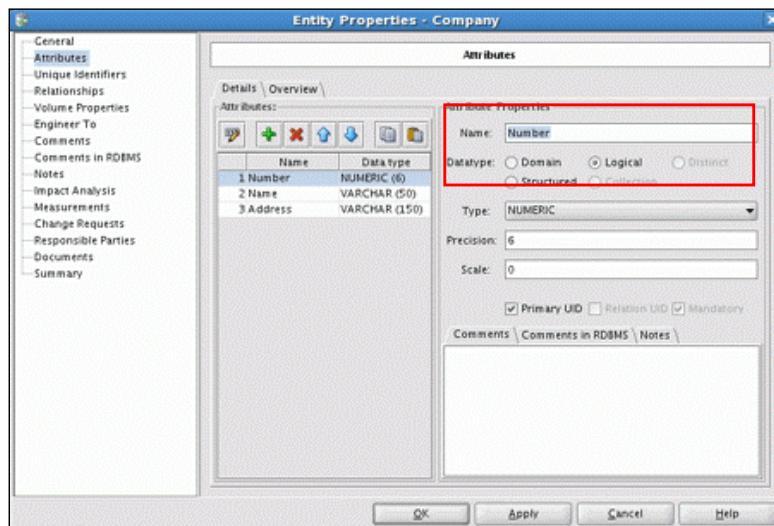
Each attribute in your logical data model must be assigned a data type that defines its format and length. The default data type is unknown.

Many data type classifications will be discussed throughout this lesson. They include:

- Logical
- Domain
- Distinct
- Structured
- Collection

Logical Type

Logical types are basic element types that are database independent.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Logical Types

Logical types are basic elements that are database independent. A set of predefined logical types have 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:

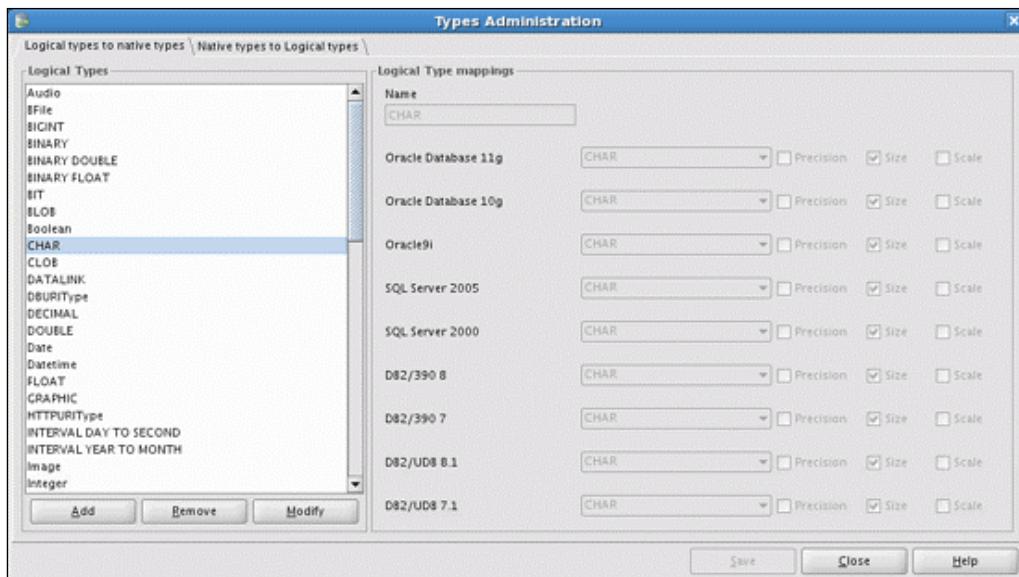
- To generate DDL, each logical type must be mapped to the native database type for each supported database type.
- To import from the data dictionary or from a DDL script, each native data type must be mapped to a logical type.

To associate a logical type with an attribute, select Logical for 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.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Types Administration

You can manage the mappings between logical data types and native data types for specific supported database products, and add and remove logical types, by using Types Administration, which can be accessed from the Tools menu.

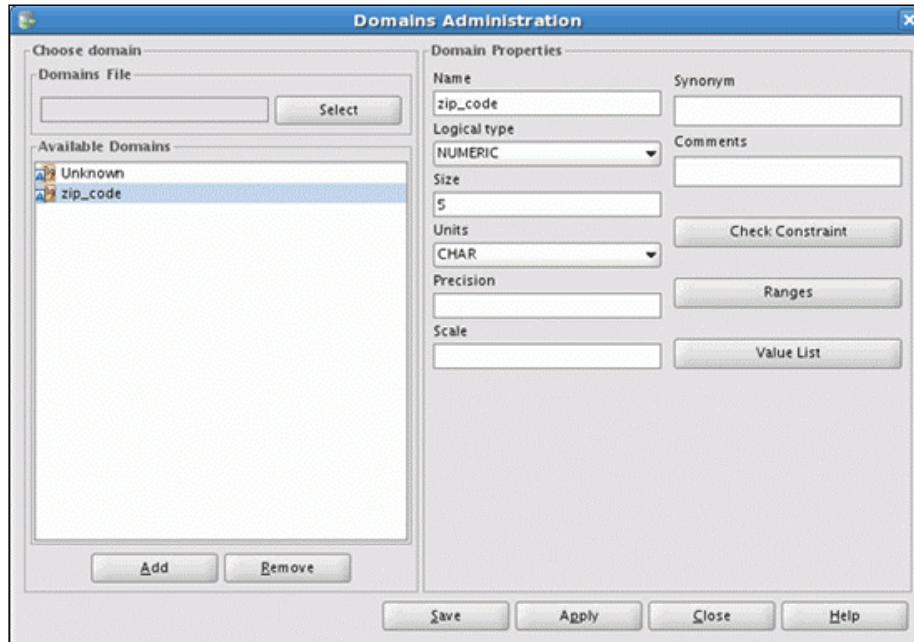
Logical Types to Native Types: For each logical type, you can select the type in the list on the left to view its mappings to a type in each supported database product.

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

Domains add meaning to logical types and can define possible values.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

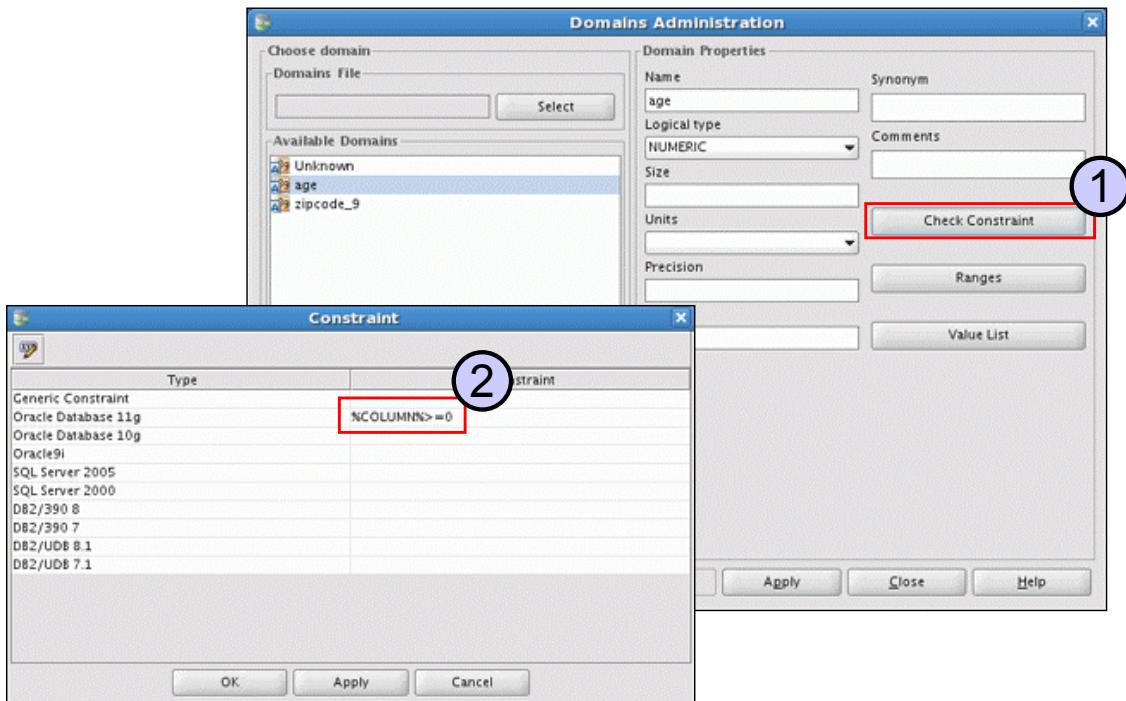
Domain

A domain adds meaning to a logical type such as zip_code, which is assigned the logical type NUMERIC with a size of 5 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 only be used 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 changes, all the attributes it is assigned to will have that change.

To access Domains Administration, select Tools > Domains Administration. In Domains Administration, you can define your set of domains by adding or removing domains and also by defining check constraints, ranges, and value lists.

Adding a Check Constraint to a Domain



ORACLE

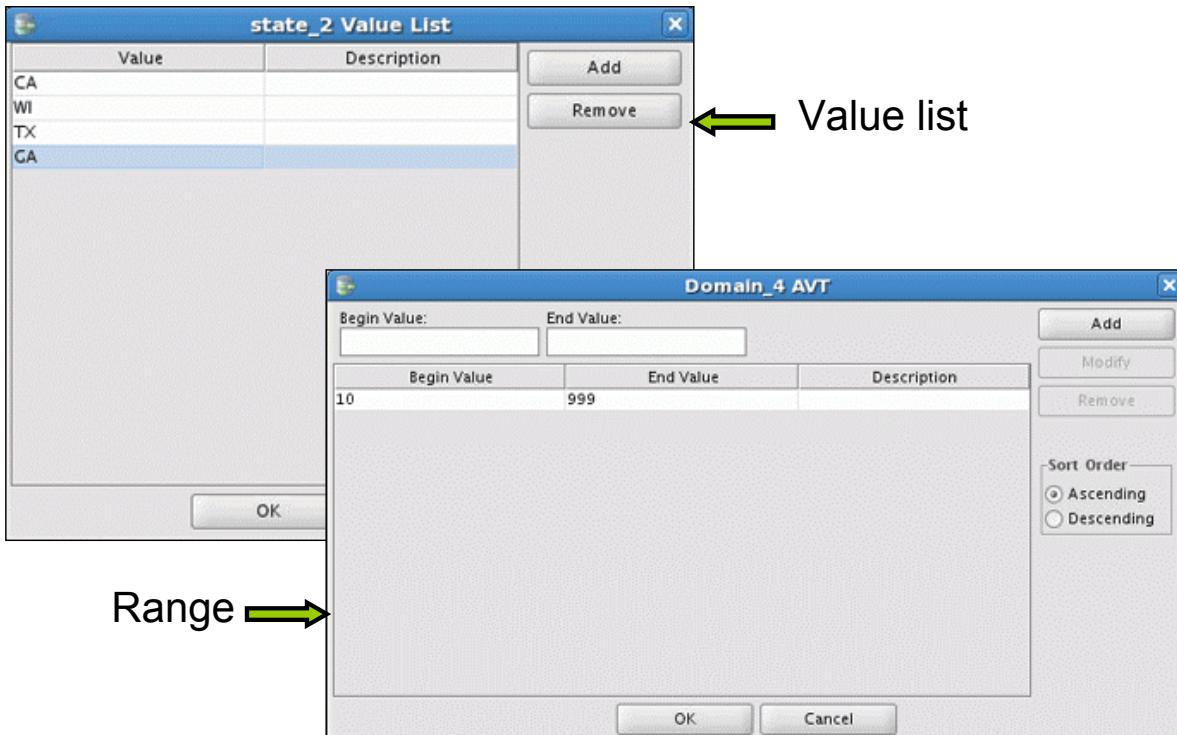
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Adding a Check Constraint to a Domain

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%>=0 for the constraint that is created when generating to Oracle Database 11g.

Adding Ranges or Value Lists to a Domain



ORACLE

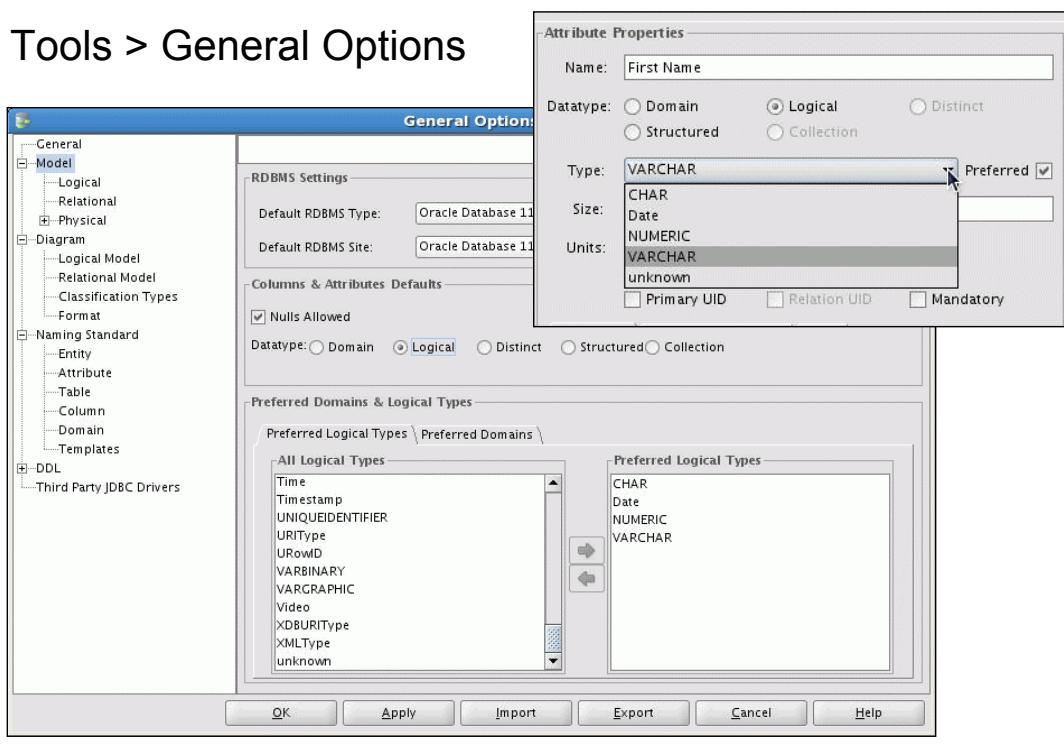
Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Adding Ranges or Value Lists to a Domain

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.

Preferred Logical Types and Domains

Tools > General Options



Preferred Logical Types and Domains

To identify the default data type and to modify the preferred set of logical types and domains for your model, perform the following steps:

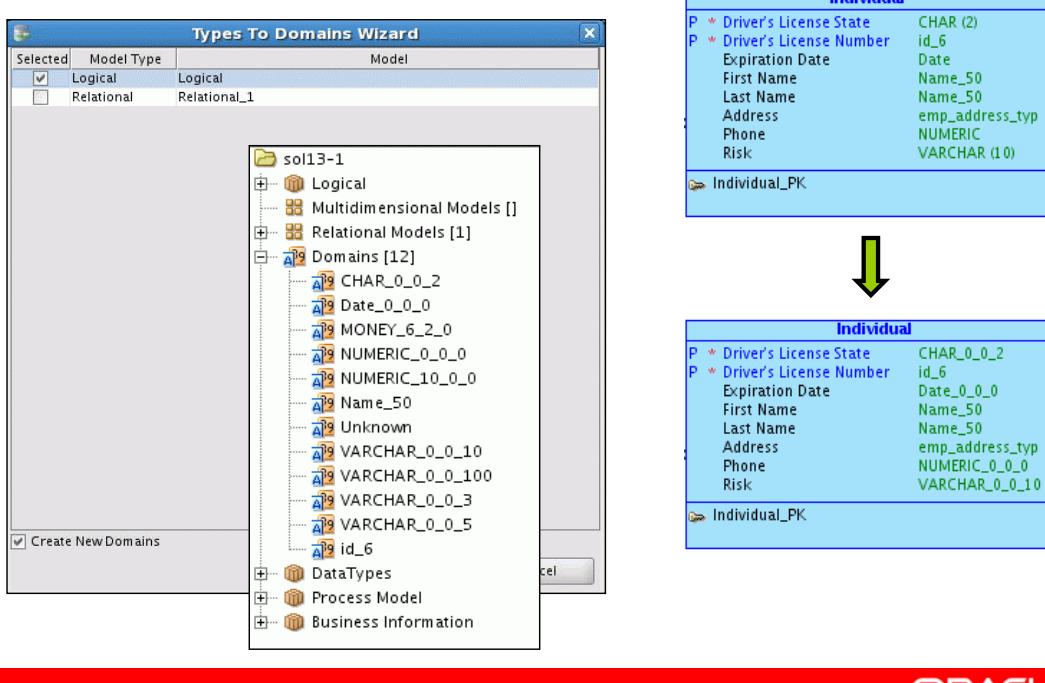
1. Select Tools > General Options, and select the Model node.
2. Select the default data type in the Column & Attributes Defaults section. In the example in the slide, Logical was selected as the default data type.
3. In the Preferred Domains and Logical Types section, select the logical types and click the right arrow to move them to the preferred area. Select 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. Click 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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Creating Domains from Logical Types

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:

1. Select Tools > Types to Domains Wizard.
2. Select the model type 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.

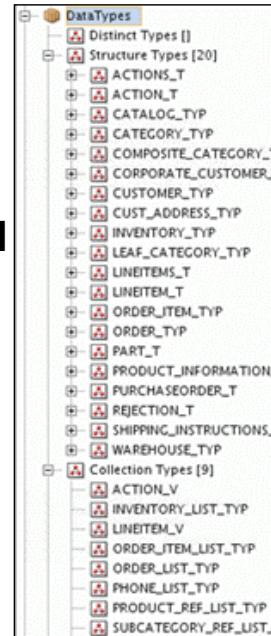
Note that the domains created using this method are only saved for this model.

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 © 2010, Oracle and/or its affiliates. All rights reserved.

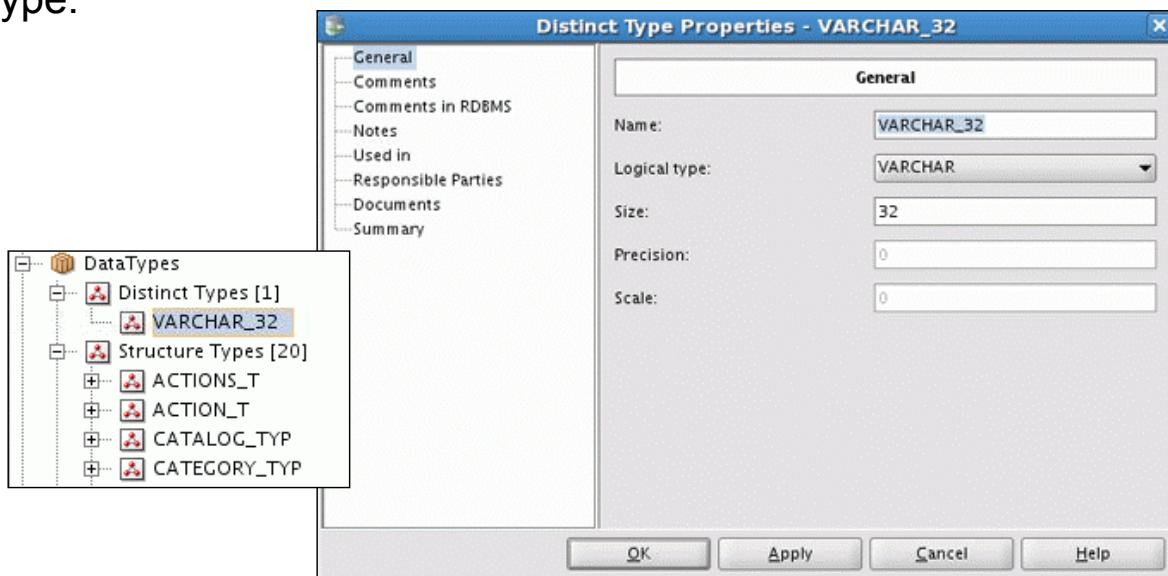
Data Type Model

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.



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

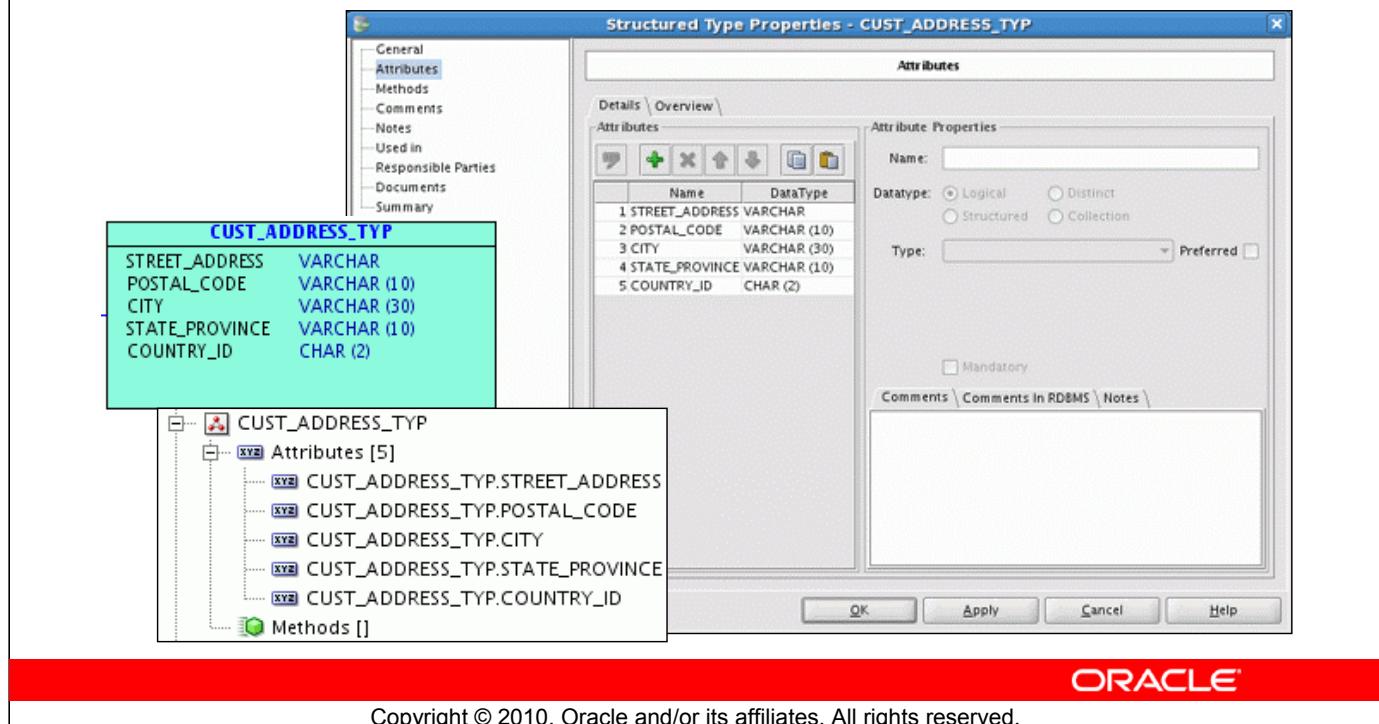
Distinct Type

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

Structured types are user-defined data types that have attributes and methods.



Structured Type

Structured types are user-defined data types that have attributes and methods. 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.

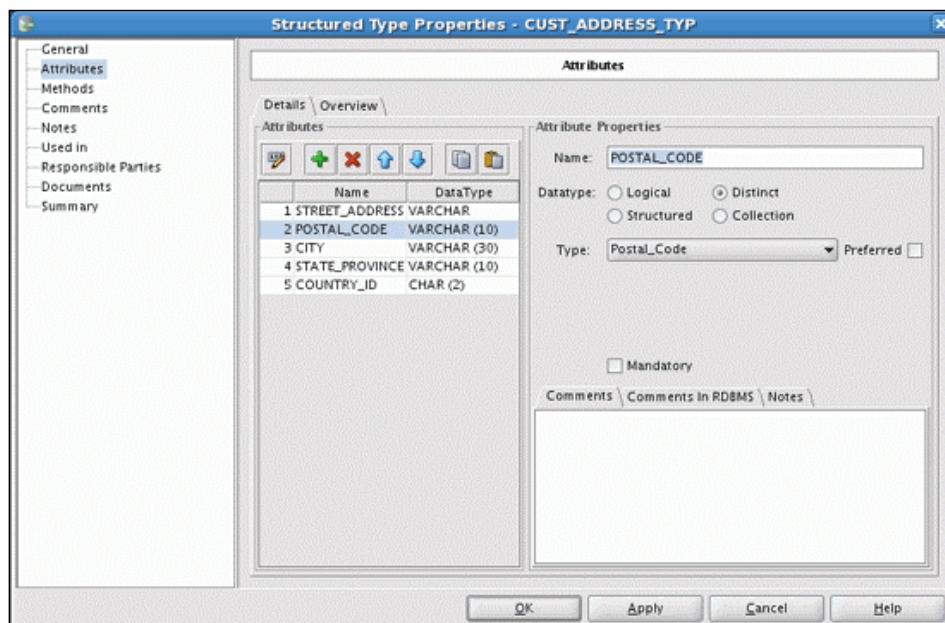
A structured type also includes a set of method specifications. Methods enable you to define behaviors for structured types. Like user-defined functions (UDFs), methods are routines that extend SQL. In the case of methods, however, the behavior is integrated solely with a particular structured type.

The expanded structured types subfolder lists all structured type objects, with the hierarchy of attributes and methods for each.

The Oracle Spatial SDO_GEOmetry 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 Within a Structured Type

Assign attributes in a structured type to a distinct type.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

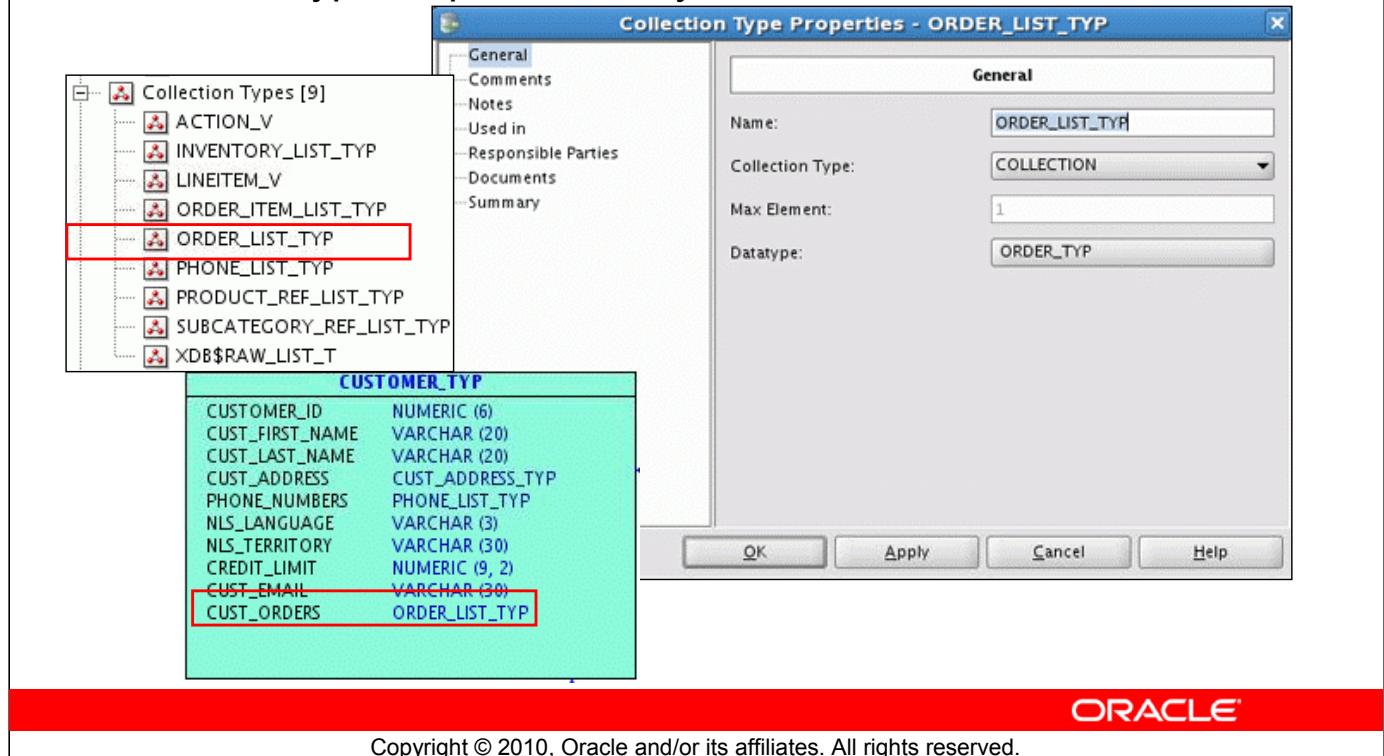
ORACLE

Using Distinct Types Within a Structured Type

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.



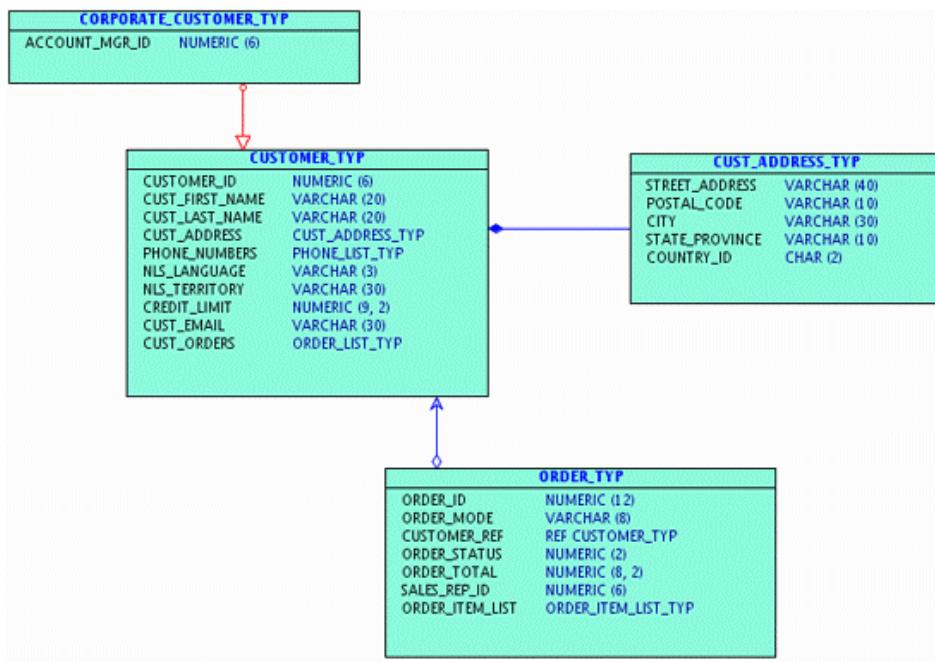
Collection Type

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



ORACLE

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Building a Data Type Model

A data type model is a graphic display of structured types.

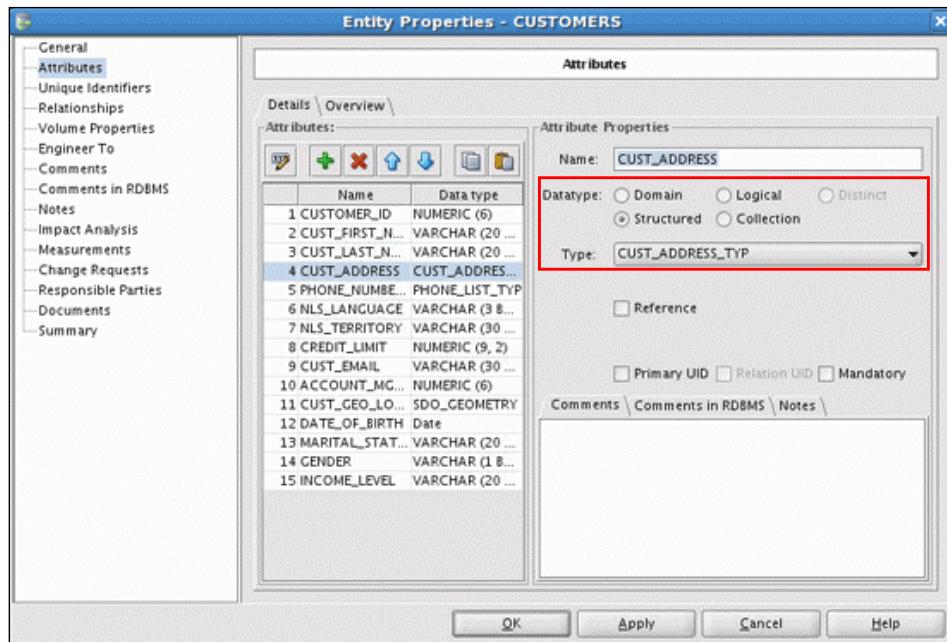
Structured types can also be part of a super type and subtype inheritance hierarchy within the data type model. In the example in the slide, `corporate_customer_typ` is a subtype of the `customer_typ` super type. A structured type can be defined based on a logical data type, a distinct type, another 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.

Assigning Data Types to an Attribute

Select a data type for an attribute, and then specify details.



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

ORACLE

Assigning Data Types to an Attribute

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 select or create the attribute you want to define.
3. Select the data type category.
4. Depending on the category selected, select from the list or specify in the text field the length.

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 within the data type model diagram?

- a. Domain
- b. Structured type
- c. Logical type
- d. Collection type
- e. Distinct



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Answer: b

Quiz

You can specify domains in a data type model.

- a. True
- b. False



Copyright © 2010, 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



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you should have learned how to build various data types and use them in your logical data model.

Practice 13-1 Overview: Create and Assign Data Types

This practice covers the following topics:

- Creating a data type model
- Defining attribute data types



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 13-1 Overview: Create and Assign Data Types

In this practice, you create a data type model and then assign data types to each attribute in your data model.

14

Putting It All Together

ORACLE®

Copyright © 2010, 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 © 2010, Oracle and/or its affiliates. All rights reserved.

Objectives

In this lesson, you build an ERD from the beginning and examine the ERD based on modeling techniques.

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 using techniques discussed in previous lessons
 - Normalize to 3rd NF
 - Examine for recursive or exclusive relationships
 - Examine for entity type hierarchies
 - Examine entities and relationships to make sure the model accounts for data changing over time



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 14-1 Overview: Develop and Validate your ERD

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 3rd NF
 - Examine for recursive or exclusive relationships
 - Examine for entity type hierarchies
 - Examine entities and relationships to make sure the model accounts for data changing over time



Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Practice 14-2 Overview: Develop and Validate your ERD (Optional)

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 © 2010, Oracle and/or its affiliates. All rights reserved.

Summary

In this lesson, you built an ERD from the beginning and examined the ERD based on modeling techniques. The practices in this lesson bring all the concepts and techniques learned to this point in the course together.

