

---

# **Data Modeling and Relational Database Design**

**Student Guide – Volume 1**

---

20000GC13  
Edition 1.3  
August 2002  
D37096

**ORACLE®**

## **Author**

Patrice Daux  
Jeff Gallus  
Jan Speelpenning

## **Technical Contributors and Reviewers**

Kate Heap  
Simmie Kastner  
Joni Lounsberry  
Satyajit Ranganathan  
Sunshine Salmon  
Stijn Vanbrabant  
Gabriella Varga

## **Publisher**

Christine Markusic

Copyright © Oracle Corporation, 2002. All rights reserved.

This documentation contains proprietary information of Oracle International Corporation. It is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited. If this documentation is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

### **Restricted Rights Legend**

Use, duplication or disclosure by the Government is subject to restrictions for commercial computer software and shall be deemed to be Restricted Rights software under Federal law, as set forth in subparagraph (c) (1) (ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

This material or any portion of it may not be copied in any form or by any means without the express prior written permission of Oracle International Corporation. Any other copying is a violation of copyright law and may result in civil and/or criminal penalties.

If this documentation is delivered to a U.S. Government Agency not within the Department of Defense, then it is delivered with 'Restricted Rights,' as defined in FAR 52.227-14, Rights in Data-General, including Alternate III (June 1987).

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them in writing to Education Products, Oracle Corporation, 500 Oracle Parkway, Box, Redwood Shores, CA 94065. Oracle International Corporation does not warrant that this document is error-free.

Oracle is a registered trademark and Oracle, SQL\*Plus, SQL\*Net, Oracle Developer, Oracle8i, Oracle9i, Oracle9i Designer and PL/SQL are trademarks or registered trademarks of Oracle Corporation.

All other products or company names are used for identification purposes only and may be trademarks of their respective owners.

# Contents

## 1 Introduction to Entities, Attributes, and Relationships

Overview	1-2
Why Create a Conceptual Model?	1-3
Between Dream and Reality...	1-5
Entity Relationship Modeling	1-7
Goals of Entity Relationship Modeling	1-8
Database Types	1-9
Entity	1-10
Entities and Instances	1-11
Entities and Sets	1-12
Attribute	1-13
Attribute Examples	1-14
Relationships	1-15
Relationship Examples	1-16
Employees have Jobs	1-17
Entity Representation in Diagram	1-18
Attributes in Diagrams	1-20
Relationship in Diagrams	1-21
Diagrams are to Communicate	1-22
Characteristics of the Relationship Line	1-23
Two Perspectives	1-24
One Way	1-25
The Other Way	1-26
Reading a Relationship End	1-27
Functions Drive Data	1-34
Weather Forecast	1-35
Weather Forecast, a Solution	1-38
Graphical Elements of ER Diagram	1-39
Summary	1-40
Practices	1-41
Practice: Instance or Entity?	1-42
Practice: Guest	1-43
Practice: Reading	1-44
Practice: Read and Comment	1-45
Practice: Hotel	1-46

## **2 Entities and Attributes in Detail**

- Overview 2-2
- Data Compared to Information 2-3
- Data 2-4
- Entities 2-6
  - 1: Mandatory m 3-9
- Some Background Information 2-9
- Some Desired Functionality 2-13
- Evolution of an Entity Definition 2-15
- Business Functions 2-17
- An Attribute... 2-19
- Nouns, Entities, Attributes 2-20
- EM Entities and Attributes 2-22
- Attribute and Entity 2-23
- Redundancy 2-24
- A Subtype ... 2-25
- Subtype: Example 2-26
- Subtype: Rules 2-27
- Subtypes: Three Levels 2-28
- More on Subtypes 2-29
- Summary 2-30
- Practices 2-31
- Summary 2-34
- Shop List 2-36
- Subtypes 2-38
- Practice: Address (1/2) 2-40
- Practice: Address (2/2) 2-41

## **3 Relationships in Detail**

- Overview 3-2
- Establishing a Relationship 3-3
- Relationship Names 3-5
- Naming the Relationship 3-6
- Optionality 3-7
- Optionality 3-8
- Mandatory

Degree 3-10  
Nontransferability 3-12  
Relationship Types 1:m 3-13  
Relationship Types m:1 3-15  
Relationship Types m:m 3-16  
Relationship Types 1:1 3-18  
1:1 Relationships Roles 3-19  
1:1 Relationships Process 3-20  
Redundant Relationships 3-21  
Relationships and Attributes 3-22  
Attribute Compared to Relationship 3-23  
Attribute or Entity 3-24  
Attribute Compared to Relationship 3-25  
Relationship Compared to Attribute 3-26  
m:m Relationships May Hide Something 3-27  
Quantity Is Attribute of ... 3-28  
Attribute of Relationship? 3-29  
New Entity ORDER 3-30  
Multiple PRODUCTS for an ORDER 3-31  
Another New Entity: ORDER ITEM 3-32  
Tables 3-33  
Resolving m:m Relationship 3-34  
Resolving m:1 Relationship 3-37  
Normalization Rules 3-39  
First Normal Form in Data Modeling 3-40  
Second Normal Form in Data Modeling 3-41  
Third Normal Form in Data Modeling 3-42  
Summary 3-43  
Practices 3-44  
Practice: Read the Relationship 3-45  
Find a Context (1) 3-46  
Find a Context (2) 3-47  
Find a Context (3) 3-48  
Find a Context (4) 3-49  
Practice: Name the Intersection Entity 3-50  
Practice: Receipt 3-51  
Practice: Moonlight P&O 3-52  
Practice: Price List 3-54  
Practice: E-Mail 3-55  
Practice: Holiday 3-56  
Practice: Normalize an ER Model 3-57

## **4 Constraints**

- Overview 4-2
- Rembrandt 4-3
- Identification and Representation 4-5
- Unique Identifier Examples 4-6
- Unique Identifier 4-7
- Unique Identifiers 4-8
- Multiple Relationship UID 4-10
- Well-defined Unique Identifiers 4-12
- Incorrect Unique Identifiers 4-13
- Information-Bearing Codes 4-14
- Arcs 4-15
- Exclusive Arc 4-16
- Possible Arc Constructs 4-17
- Some Incorrect Arc Constructs 4-18
- Arc or Subtype 4-20
- Arc and Subtypes 4-21
- Subtypes Hide Relationships in Arc 4-22
- Value sets 4-23
- Other Constraints: Range Check 4-24
- Other Constraints: State Value Transition 4-25
- Conditional Relationship 4-26
- Boundaries 4-28
- Summary 4-29
- Practices 4-30
- Practice: Identification Please 4-31
- Practice: Identification 1 4-32
- Practice: Identification 2 4-33
- Practice: Identification 3 4-34
- Practice: Identification 4 4-35
- Practice: Identification 5 4-36
- Practice: Identification 6 4-37
- Practice: Moonlight UID 4-38
- Practice: Table 14-39
- Practice: Table 2 4-41
- Practice: Constraints 1 4-42
- Practice: Constraints 2 4-43
- Practice: Constraints 3 4-44

## **5 Modeling Change**

- Overview 5-2
- Change and Time 5-3
- Entity DAY 5-4
- Modeling Change 5-6
- Even a Country Has a Life Cycle 5-8
- Products and Prices 5-10
- What Price to Pay? 5-11
- Price List Search 5-12
- Order for Priced Products 5-13
- Negotiated Prices 5-14
- Current Prices 5-16
- Journaling 5-17
- Summary 5-19
- Practices 5-20
- Practice: Shift 5-21
- Practice: Strawberry Wafer 5-22
- Practice: Bundles(1) 5-24
- Practice: Bundles(2) 5-25
- Practice: Product Structure 5-26

## **6 Advanced Modeling Topics**

- Overview 6-2
- Patterns 6-3
- Patterns: Master–Detail 6-5
- Pattern: Basket 6-6
- Patterns: Classification 6-7
- Patterns: Hierarchy 6-8
- Patterns: Chain 6-10
- Patterns: Network 6-11
- Bill of Material 6-13
- Bill of Material-Example 6-14
- Symmetric Relationship 6-15
- Patterns: Roles 6-16
- Roles 6-17
- Fan Trap 6-18
- Fan Trap Resolved 6-19
- Patterns: Data Warehouse 6-20
- Drawing Conventions 6-21
- Use Conventions Sensibly 6-22
- Model Readability 6-23
- Generic Modeling 6-24
- Generic Model 6-26

- Generic 6-27
- More Generic 6-28
- More Generic Plus 6-29
- Most Generic? 6-30
- Best of Two Worlds 6-31
- Summary 6-32
- Practices 6-33
- Practice: Patterns 6-34
- Practice: Data Warehouse 6-35
- Practice: Argos and Erats 6-38
- Practice: Synonym 6-39

## **7 Mapping the Entity Model**

- Overview 7-2
- Why Create a Data Design Model? 7-3
- Presenting Tables 7-4
- Transformation Process 7-6
- Terminology Mapping 7-7
- General Naming Topics 7-8
- Naming Restrictions with Oracle 7-11
- Basic Mapping for Entities 7-12
- Basic Mapping for Attributes 7-13
- Basic Mapping 7-14
- Rules for Relationships 7-16
- Mapping 1:m Relationships 7-17
- Mapping Barred and Nontransferable Relationships 7-19
- Mapping Cascade Barred Relationships 7-20
- Mapping m:m Relationships 7-21
- Mapping 1:1 Relationships 7-22
- Mapping Arcs 7-23
- Mapping Subtypes 7-25
- Supertype Implementation 7-27
- Subtype Implementation 7-29
- Supertype and Subtype (Arc) Implementation 7-31
- Storage Implication 7-33
- Storage Implication Supertype Implementation 7-34
- Storage Implication Subtype Implementation 7-35
- Storage Implication Supertype and Subtype (Arc) Implementation 7-36



Summary 7-37  
Practice 7-38  
Practice: Mapping basic Entities, Attributes and Relationships 7-39  
Practice: Mapping Supertype 7-40  
Partial ER model Moonlight 7-41  
Practice: Quality Check Subtype Implementation 7-42  
Practice: Quality Check Arc Implementation 7-43  
Practice: Mapping Primary Keys and Columns 7-44

## **8 Denormalized Data**

Overview 8-2  
Denormalization Overview 8-3  
Denormalization Techniques 8-4  
Storing Derivable Values 8-5  
EMail Example of Storing Derivable Values 8-6  
Pre-Joining Tables 8-7  
EMail Example of Pre-Joining Tables 8-8  
Hard-Coded Values 8-9  
Email Example of Hard-Coded Values 8-10  
Keeping Details with Master 8-11  
EMail Example Keeping Detail with Master 8-12  
Repeating Current Detail with Master 8-13  
EMail Example of Repeating Single Detail with Master 8-14  
Short-Circuit Keys 8-15  
EMail Example of Short-Circuit Keys 8-16  
End Date Column 8-17  
Example of End Date Column 8-18  
Current Indicator Column 8-19  
Example of Current Indicator Column 8-20  
Hierarchy Level Indicator 8-21  
Example of Hierarchy Level Indicator 8-22  
Denormalization Summary 8-23  
Practices 8-24  
Practice: Name that Denormalization (1/3) 8-25  
Practice: Name that Denormalization (2/3) 8-26  
Practice: Name that Denormalization (3/3) 8-27  
Practice: Triggers (1/6) 8-28  
Practice: Triggers (2/6) 8-29  
Practice: Triggers (3/6) 8-30  
Practice: Triggers (4/6) 8-31  
Practice: Triggers (5/6) 8-32  
Practice: Triggers (6/6) 8-33  
Practice: Denormalize Price Lists 8-34  
Practice: Global Naming 8-35

## **9 Database Design Considerations**

Overview 9-2

Why Adapt Data Design? 9-3

Oracle Data Types 9-4

Suggested Column Sequence 9-6

Primary Keys 9-7

Artificial Keys 9-11

Sequences 9-13

Foreign Key Behavior 9-14

Indexes 9-16

Choosing Indexes 9-17

Which Columns to Index? 9-19

When Can Indexes be Used? 9-21

Partitioning Tables and Indexes 9-22

Views 9-23

Reasons for Views 9-24

Old Fashioned Design 9-25

Generic Arc Implementation 9-26

Distributed Database 9-27

Benefits of Distributed Databases 9-28

Database Structure 9-29

Summary 9-31

Practices 9-32

Data Types (1) 9-34

Data Types (2) 9-35

## **Appendix A**

## **Appendix B**

# 1

## **Introduction to Entities, Attributes, and Relationships**

ORACLE®

Copyright © Oracle Corporation, 2002. All rights reserved.

# Overview

## Why conceptual modeling?

### Introduction of the Key role players:

- **Entities**
- **Attributes**
- **Relationships**

ORACLE®

1-2

Copyright © Oracle Corporation, 2002. All rights reserved.

## Introduction

This lesson explains the reasons for conceptual modeling and introduces the key role players: entities, attributes, and relationships.

### Objectives

At the end of this lesson, you should be able to do the following:

- Explain why conceptual modeling is important
- Describe what an entity is and give examples
- Describe what an attribute is and give examples
- Describe what a relationship is and give examples
- Draw a simple diagram
- Read a simple diagram

## Why Create a Conceptual Model?

- **It describes exactly the information needs of the business**
- **It facilitates discussion**
- **It helps to prevent mistakes, misunderstanding**
- **It forms important “ideal system” documentation**
- **It forms a sound basis for physical database design**
- **It is a very good practice with many practitioners**

ORACLE

1-3

Copyright © Oracle Corporation, 2002. All rights reserved.

### Why Conceptual Modeling?

This is a course on conceptual data modeling and physical data modeling. Why do you need to learn this? Why invest time in creating entity models when you need tables? Why bother about business functionality and interviews and feedback sessions when you need programs? In this course you learn why. You learn why it is a wise decision to spend time in modeling and why it is a good investment. You will learn even more, including how to create, read, and understand models and how to check them, as well as how to derive table and key definitions from them.

This list shows the reasons for creating a conceptual model. The most important reason is that a conceptual model facilitates the discussion on the shape of the future system. It helps communication between you and your sponsor as well as you and your colleagues. A model also forms a basis for the default design of the physical database. Last but not least, it is relatively cheap to make and very cheap to change.

## **What You Learn in This Course**

In this course you learn how to analyze the requirements of a business, how to represent your findings in an entity relationship diagram and how to define and refine the tables and various other database objects from that model.

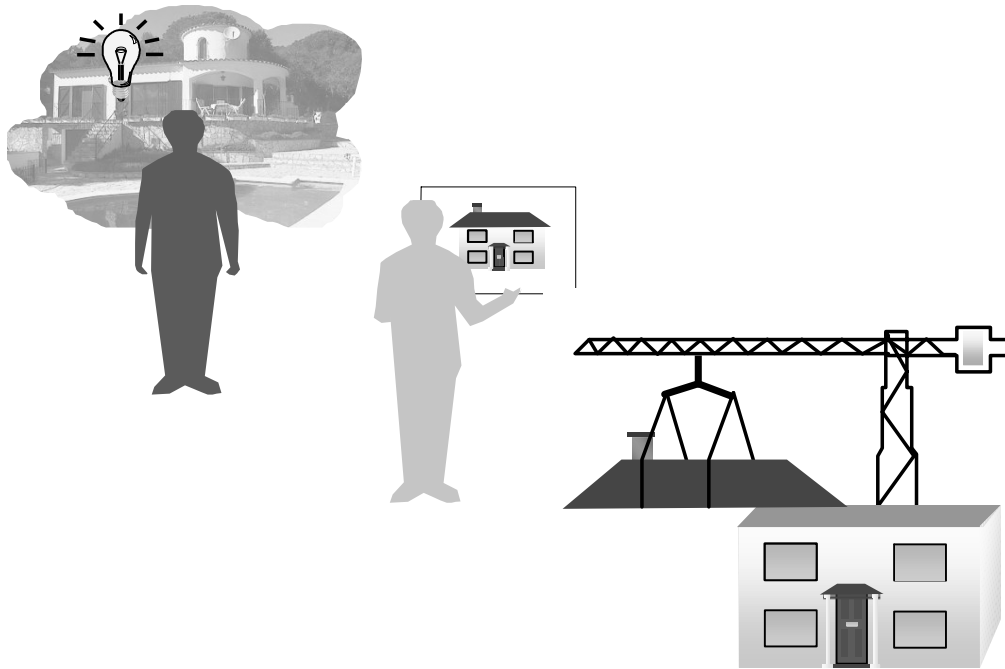
In summary, as a result of what you learn in this course you will know:

- How to model the information needs of a business and the rules that apply.
- Which tables you need in your database, and why.
- Which columns you need in your tables, and why.
- Which constraints and other database objects you require.

You will also know how to explain this to:

- Your sponsors.
- The developers.
- Your fellow designers.

## Between Dream and Reality...



ORACLE

1-5

Copyright © Oracle Corporation, 2002. All rights reserved.

### The House Building Metaphor

Imagine someone who wants to have a house built. Initially, the house only exists in the minds of the future home owners as ideas, or as pieces of various dreams. Sometimes the future inhabitants may not even know what they want, or know if what they want is even feasible. Dreams may be full of internal contradictions and impossibilities. This is not a problem in the dream world, but in the physical realm any inconsistencies and obstacles need to be resolved before someone can construct the house.

A building contractor needs a solid plan, a set of blueprints of the house with a description of the materials to be used, the size of the roof beams, the capacity of the plumbing and many, many other things. The contractor follows the plan, and has the knowledge to construct what is on the blueprint. But how do the ideas of the home owner become the blueprint for contractor? This is where the architect becomes involved.

#### The Architect

The architects are the intermediary between sponsor and constructor. They are trained in the skills of translating ideas into models. The architect listens to the description of the ideas and asks all kinds of questions. The architect's skills in extracting the ideas, putting it down in a format that allows discussion and analysis, giving advice, describing sensible options, documenting it, and confirming it with the home owners, are the cornerstones to providing the future home-owner with a plan of the home they want.

## **The House Building Metaphor**

### **Sketches**

The architect's understanding of the dreams is transformed into sketches of the new house—only sketches! These consist of floor plans and several artist's impressions, and show the functional requirements of the house, not the details of the construction. This is a conceptual model, the first version.

### **Easy Change**

If parts of the model are not satisfactory or are misunderstood, the model can easily be changed. Such a change would only need a little time and an eraser, or a fresh sheet of paper. Remember, it is only changing a model. The cost of change at this stage is very low. Certainly it is far less costly than making changes to the floor plan or roof dimensions after construction has started. The house model is then reviewed again, and further changes are made. The architect continues to explore and clarify the dreams and make alternative suggestions until all controversial issues are settled, and the model is stable and ready for the final approval by the sponsor.

### **Technical Design**

Then the architect converts the model into a technical design, a plan the contractor can use to build the house. Calculations are made to determine, for example, the number of doors, how thick the walls and floor beams must be, the dimensions of the plumbing, and the exact construction of the roof. These are technical issues that need not involve the customer.

### **What? as Opposed to How?**

While the conceptual model addresses the What? phase in the process, the design addresses the question of How? it is to be constructed.

Conceptual modeling is similar to the work of an architect—transforming things that only exist in people's minds into a design that is sufficiently substantial to be created physically.



# Entity Relationship Modeling

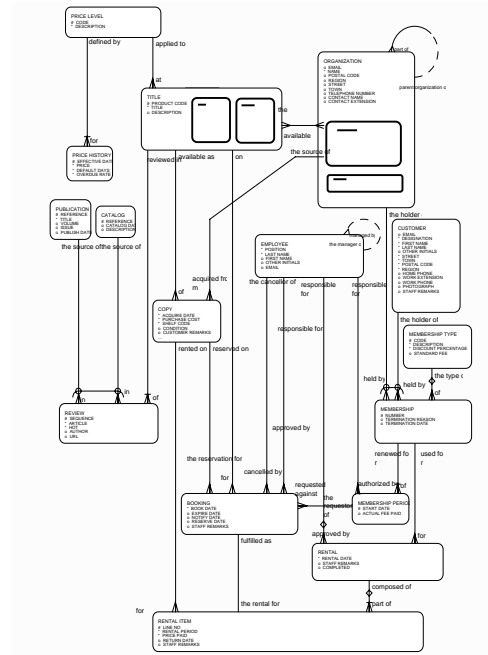
- Models business, not implementation
- Is a well-established technique
- Has a robust syntax
- Results in easy-to-read diagrams...

...although they may look rather complex at first sight

ORACLE

1-7 Copyright © Oracle Corporation, 2002. All rights reserved.

- ...although they may look rather complex at first sight**



## What is Involved in Modeling?

Entity Relationship modeling is about modeling a business. To be more precise: it is about modeling the data requirements for a business based on the current or desired functionality of the future system.

To model a business you have to understand to a fair degree of detail what the business is about.

Entity Relationship modeling is a technique used to describe the shared understanding of the information needs of a business. It is a well-established technique that leads to diagrams which are quite easy to read and therefore also easy to check.

## Goals of Entity Relationship Modeling

- **Capture *all* required information**
- **Information appears *only* once**
- **Model *no* information that is derivable from other information already modeled**
- **Information is in a predictable, logical place**

ORACLE

1-8

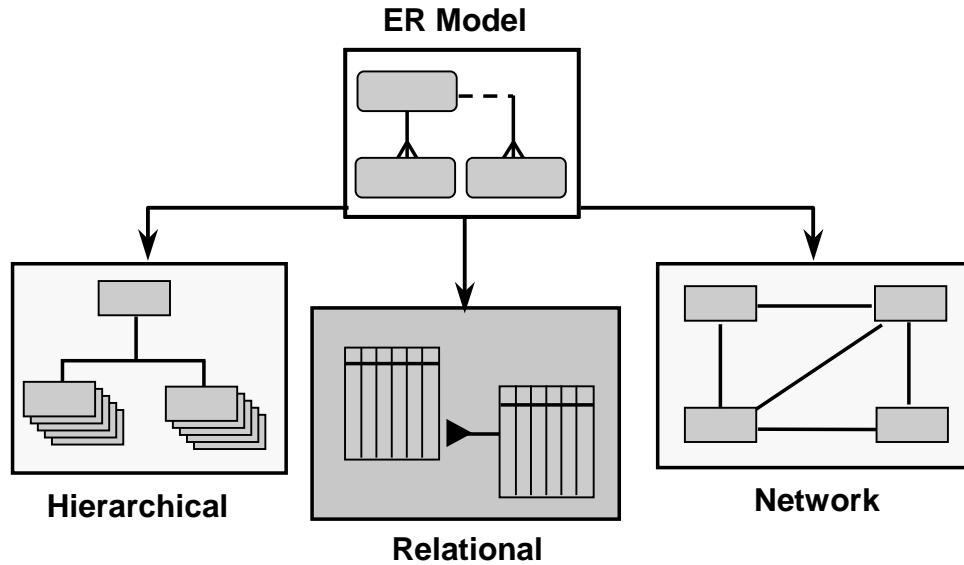
Copyright © Oracle Corporation, 2002. All rights reserved.

### Goals of Entity Relationship Modeling

The goals of conceptual data modeling are to ensure that:

- All pieces of information that are required to run a business properly are recognized.
- Models should be complete. Requirements should be known before you start implementing. Dependencies must be clear.
- Every single piece of required information appears only once in the model.
- This is an important goal. As soon as a system stores particular information twice, you run into the possibility that this information is not the same in both places. If you are a user of an information system and discover inconsistencies in the data, which information would you trust?
- This goal implies that an ideal system does not contain derivable information.
- In the future system, the information is made available in a predictable, logical place; related information is kept together.
- A proper Entity Relationship model leads to a set of logically coherent tables.

# Database Types



ORACLE

## Database Types

Entity Relationship modeling is independent of the hardware or software used for implementation. Although you can use an Entity Relationship model as a basis for hierarchical databases, network databases, and relational databases, it is strongly connected to the latter.

# Entity

- **An Entity is:**
  - “Something” of significance to the business about which data must be known
  - A name for the things that you can list
  - Usually a noun
- **Examples: objects, events**
- **Entities have instances**

ORACLE

1-10

Copyright © Oracle Corporation, 2002. All rights reserved.

## Definition of an Entity

There are many definitions and descriptions of an entity. Here are a few; some are quite informal, some are very precise.

- An entity is something of interest.
- An entity is a category of things that are important for a business, about which information must be kept.
- An entity is something you can make a list of, and which is important for the business.
- An entity is a class or type of things.
- An entity is a named thing, usually a noun.

Two important aspects of an entity are that it has instances and that the instances of the entity somehow are of interest to the business.

Note the difference between an entity and an instance of an entity.

## Entities and Instances

<b>PERSON</b>	<b>Mahatma Gandhi</b>
<b>PRODUCT</b>	<b>2.5 x 35 mm copper nail</b>
<b>PRODUCT TYPE</b>	<b>nail</b>
<b>EMPLOYMENT CONTRACT</b>	<b>my previous contract</b>
<b>JOB</b>	<b>violinist</b>
<b>SKILL LEVEL</b>	<b>fluent</b>
<b>TICKET RESERVATION</b>	<b>tonight: Hamlet in the Royal</b>
<b>PURCHASE</b>	<b>the CD I bought yesterday</b>
<b>ELECTION</b>	<b>for parliament next fall</b>
<b>PRINTER PREFERENCE</b>	<b>...</b>
<b>DOCUMENT VERSION</b>	<b>...</b>

ORACLE

1-11

Copyright © Oracle Corporation, 2002. All rights reserved.

### More on Entities

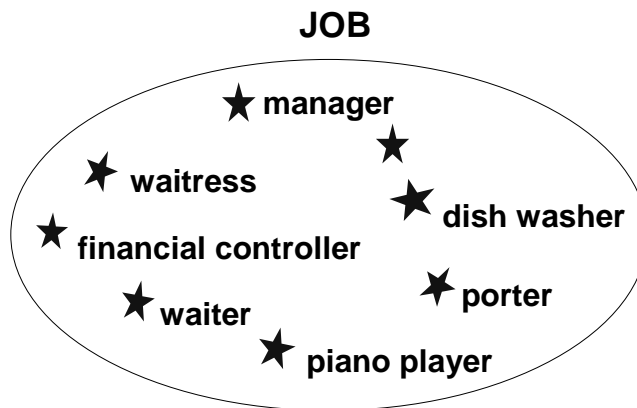
The illustration shows examples of entities and examples of instances of those entities.

#### Note:

- There are many entities.
- Some entities have many instances, some have only a few.
- Entities can be:
  - Tangible, like PERSON or PRODUCT.
  - Non-tangible, like REQUIRED SKILL LEVEL.
  - An event, like ELECTION.
- An instance of one entity may be an entity in its own right: the instance “violinist” of entity JOB could be the name of another entity with instances like “David Oistrach”, “Kyung-Wha Chung.”

# Entities and Sets

**An entity represents a set of instances that are of interest to a particular business.**



ORACLE

1-12

Copyright © Oracle Corporation, 2002. All rights reserved.

## Entities and Sets

You can regard entities as sets. The illustration shows a set **JOB** and the set shows some of its instances. At the end of the entity modeling process entities are transformed into tables; the rows of those tables represent an individual instance.

During entity modeling you look for properties and rules that are true for the whole set. Often you can decide on the rules by thinking about example instances. The following lessons contain many examples of this.

### Set Theory

Entity relationship modeling and the theory of relational databases are both based on a sound mathematical theory, that is, set theory.

# Attribute

- **Also represents something of significance to the business**
- **Is a *single valued* property detail of an entity**
- **Is a specific piece of information that:**
  - **Describes**
  - **Quantifies**
  - **Qualifies**
  - **Classifies**
  - **Specifies an entity**

ORACLE

1-13

Copyright © Oracle Corporation, 2002. All rights reserved.

## What is an Attribute?

An attribute is a piece of information that in some way describes an entity. An attribute is a property of the entity, a small detail about the entity.

### Entities Have Attributes

For now, assume that all entities have at least one attribute. Later, you discover exceptions to this assumption. The attribute describes, quantifies, qualifies, classifies, and specifies an entity. Usually, there are many attributes for an entity, but again, we are only interested in those attributes that are of importance to the business.

### Values and Data Types

Attributes have values. An attribute value can be a number, a character string, a date, an image, a sound, and even more. These are called data types or formats. Usually the values for a particular attribute of the instances of an entity all have the same data type. Every attribute has a data type.

### Attribute is Single Valued

An attribute for an entity must be single valued. In more precise terms, an entity instance can have only one value for that attribute at any point in time. This is the most important characteristic of an attribute.

The attribute value, however, may change over time.

## Attribute Examples

Entity	Attribute
EMPLOYEE	Family Name, Age, Shoe Size, Town of Residence, Email, ...
CAR	Model, Weight, Catalog Price, ...
ORDER	Order Date, Ship Date, ...
JOB	Title, Description, ...
TRANSACTION	Amount, Transaction Date, ...
EMPLOYMENT	Start Date, Salary, ...
CONTRACT	

ORACLE

1-14

Copyright © Oracle Corporation, 2002. All rights reserved.

### Attribute Examples

#### Note:

- Attribute Town of Residence for EMPLOYEE is an example of an attribute that is quite likely to change, but is probably single valued at any point in time.
- Attribute Shoe Size may seem to be of no importance, but that depends on the business: if the business supplies industrial clothing to its employees, this may be a very sensible attribute to take.
- Attribute Family Name may not seem to be single-valued for someone with a double name. This double name, however, can be regarded as a single string of characters that forms just one name.

#### Volatile Attributes

Some attributes are volatile (unstable). An example is the attribute Age. Always look for nonvolatile, stable, attributes. If there is a choice, use the nonvolatile one. For example, use the attribute Birth Date instead of Age.



# Relationships

- **Also represent something of significance to the business**
- **Express how entities are mutually *related***
- **Always exist between *two* entities (or one entity *twice*)**
- **Always have two perspectives**
- **Are named at both ends**

ORACLE

## Relationship Examples

**EMPLOYEES *have* JOBS**  
**JOBS *are held by* EMPLOYEES**

**PRODUCTS *are classified by* a PRODUCT TYPE**  
**PRODUCT TYPE *is a classification for* a PRODUCT**

**PEOPLE *make* TICKET RESERVATIONS**  
**TICKET RESERVATIONS *are made by* PEOPLE**

ORACLE

1-16

Copyright © Oracle Corporation, 2002. All rights reserved.

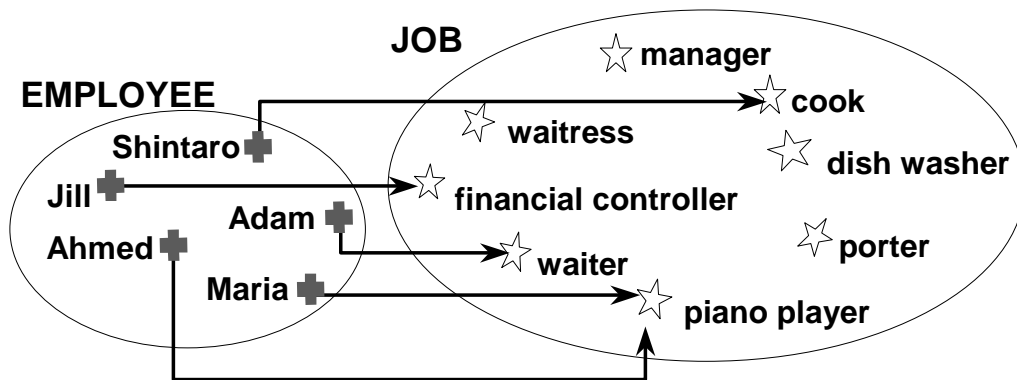
### Relationships

A relationship connects two entities. A relationship represents a significant dependency of two entities—always two entities.

A particular relationship can be worded in many ways: An EMPLOYEE has a JOB, or an EMPLOYEE performs a JOB, or an EMPLOYEE holds a JOB.

An EMPLOYEE applies for a JOB expresses a different relationship. Note that this example shows that two entities can have more than one relationship.

## Employees have Jobs



### ***Numerical observation:***

- **All EMPLOYEES have a JOB**
- **No EMPLOYEE has *more than one* JOB**
- ***Not all* JOBS are held by an EMPLOYEE**
- **Some JOBS are held by *more than one* EMPLOYEE**

ORACLE

1-17

Copyright © Oracle Corporation, 2002. All rights reserved.

## Employees have Jobs

Based on what you know about instances of the entities, you can decide on four questions:

- Must every employee have a job?
- In other words, is this a mandatory or optional relationship for an employee?
- Can employees have more than one job?

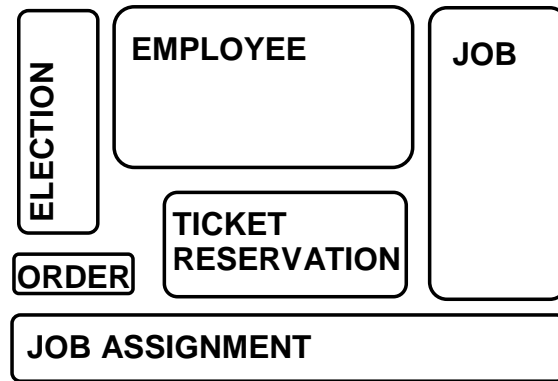
and

- Must every job be done by an employee?
- In other words, is this a mandatory or optional relationship for a job?
- Can a job be done by more than one employee?

Later on we will see why these questions are important and why (and how) the answers have an impact on the table design.

## Entity Representation in Diagram

- Drawn as a “softbox”
- Name singular
- Name inside
- Neither size, nor position has a special meaning



**During design, entities usually lead to tables.**

ORACLE

### Entity Relationship Models and Diagrams

An Entity Relationship Model (ER Model) is a list of all entities and attributes as well as all relationships between the entities that are of importance. The model also provides background information such as entity descriptions, data types and constraints. The model does not necessarily include a picture, but usually a diagram of the model is very valuable.

An Entity Relationship Diagram (ER Diagram) is a picture, a representation of the model or of a part of the model. Usually one model is represented in several diagrams, showing different business perspectives.

## **Graphical Elements**

Entity Relationship diagramming uses a number of graphical elements. These are discussed in the next pages.

Unfortunately, there is no ISO standard representation of ER diagrams. Oracle has its own convention. In this course we use the Oracle diagramming technique, which is built into the Oracle Designer tool.

In an ER diagram entities are drawn as soft boxes with the entity name inside. Borders of the entity boxes never cross each other. Entity boxes are always drawn upright.

Throughout this book, entity names are printed in capitals. Entity names are preferably in the singular form; you will find that diagrams are easier to read this way.

### **Box Size**

Neither the size of an entity, nor its position, has a special meaning. However, a reader might construe a larger entity to be of more importance than a smaller one.

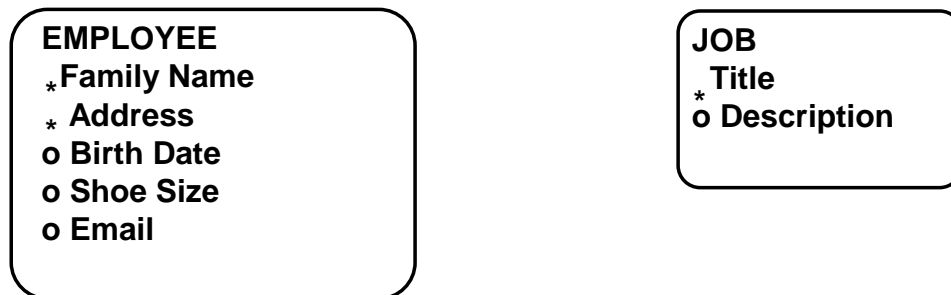
### **Where Entities Lead**

During the design for a relational database, an entity usually leads to a table.

# Attributes in Diagrams

**Mandatory attribute, that is, known *and* available for every instance.**

**Optional attribute, that is, unknown *or* unimportant to know for some instances.**



**During design, attributes lead to columns.**

ORACLE

1-20

Copyright © Oracle Corporation, 2002. All rights reserved.

## Attribute Representation

Attributes are listed within the entity box. They may be preceded by a \* or an O. These symbols mean that the attribute is mandatory or optional, respectively. Throughout this book attributes are printed in Initial Capital format.

### Mandatory:

It is realistic to assume that for every instance of the entity the attribute value is known and available when the entity instance is recorded and that there is a business need to record the value.

### Optional:

The value of the attribute for an instance of the entity may be unknown or unavailable when that instance is recorded or the value may be known but of no importance.

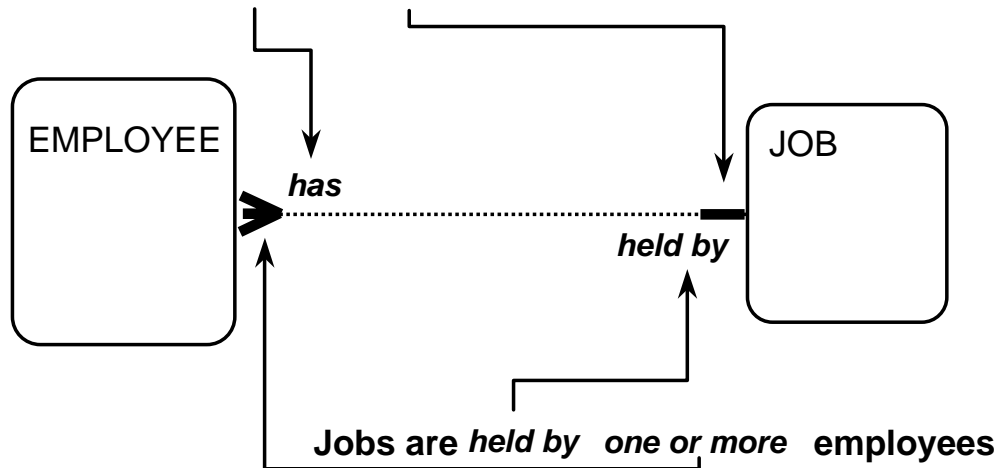
Not all attributes of an entity need to be present in the diagram, but all attributes must be known before making the table design. Often only a few attributes are shown in a diagram, for reasons of clarity and readability. Usually you choose those attributes that help understanding of what the entity is about and which more or less “define” the entity.

### Where Attributes Lead

During design an attribute usually leads to a column. A mandatory attribute leads to a not null column.

## Relationship in Diagrams

An employee *has exactly one* job.



During design, relationships lead to foreign keys.

ORACLE

### Relationship Representation

Relationships are represented by a line, connecting the entities. The name of the relationship, from either perspective, is printed near the starting point of the relationship line.

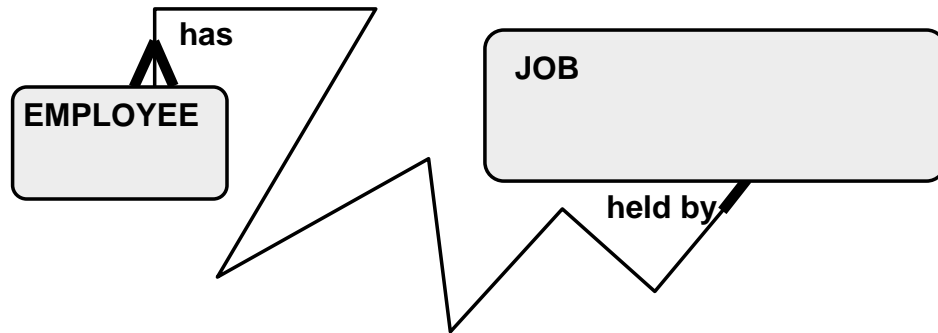
The shape of the end of the relationship line represents the degree of the relationship. This is either one or many. One means exactly one; many means one or more.

In the above example, it is assumed that JOBS are held by one or more EMPLOYEES. This is shown by the tripod (or crow's foot), at EMPLOYEE.

An EMPLOYEE, on the other hand, is assumed here to have exactly one JOB. This is represented by the single line at JOB.

The relationship line may be straight, but may also be curved; curves have no special meaning, nor does the position of the starting point of the relationship line. The diagram below represents exactly the same model, but arguably less clearly.

# Diagrams Are To Communicate





# Characteristics Of The Relationship Line

mandatory:  optional: 

ORACLE

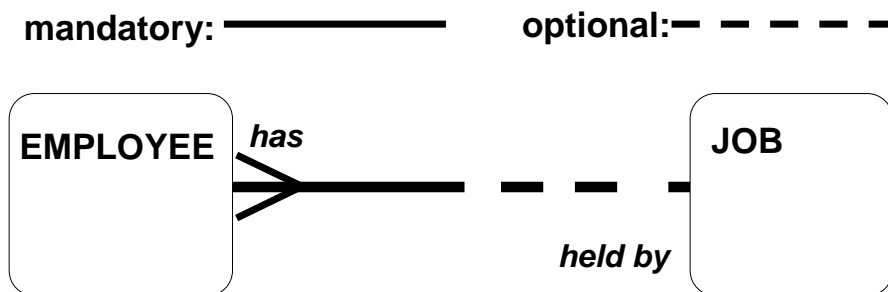
1-23

Copyright © Oracle Corporation, 2002. All rights reserved.

## Mandatory and Optional Relationships

Relationships can be mandatory or optional, in the same way as attributes. Mandatory relationships are drawn as a solid line; optional relationships as dotted lines.

## Two Perspectives

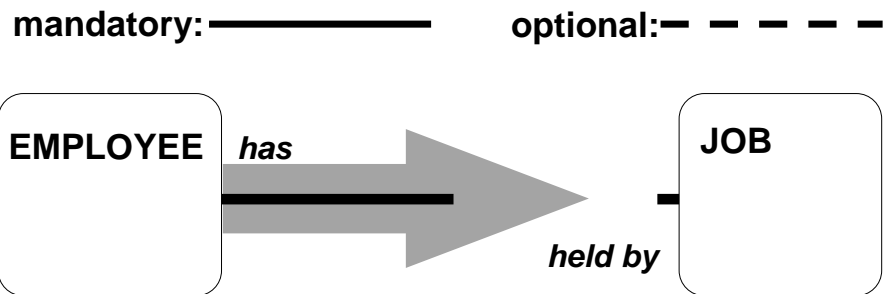


ORACLE

### Relationship and Relationship Ends

Here, the relationship between EMPLOYEE and JOB is modeled using the optional relationship end and mandatory relationship end notation.

## One Way



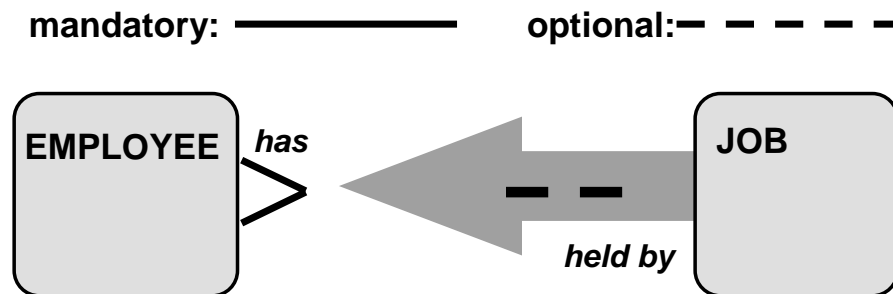
**Every EMPLOYEE *has* exactly one JOB**

ORACLE

### Relationship and Relationship Ends

When you read the relationship, imagine it split into two perspectives:

## The Other Way



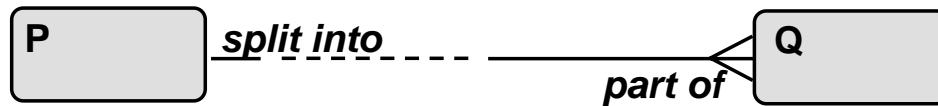
A **JOB** may be *held by* one or more **EMPLOYEES**

ORACLE

### Relationship and Relationship Ends (continued)

- Every **EMPLOYEE** has exactly one **JOB** or, alternatively:
- An **EMPLOYEE** *must have* exactly one **JOB**.

## Reading a Relationship End



ORACLE

1-27

Copyright © Oracle Corporation, 2002. All rights reserved.

### Reading a Relationship End

A relationship from **entity1** to **entity2** must be read:

Each **entity1** {must be | may be}

relationship\_name

{one or more | exactly one} **entity2**

### Where Relationships Lead

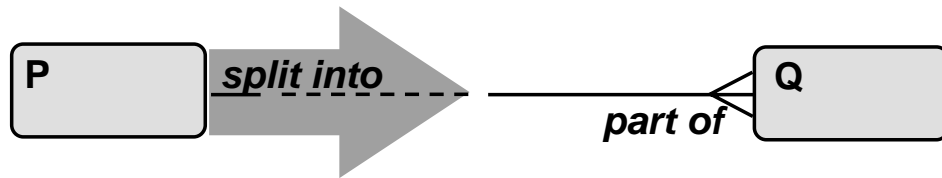
During design relationships lead to foreign keys and foreign key columns. An optional relationship leads to non mandatory foreign key columns.

### Relationship Name in the Diagram

Throughout this book relationship names in the diagrams are printed in lower case italics.

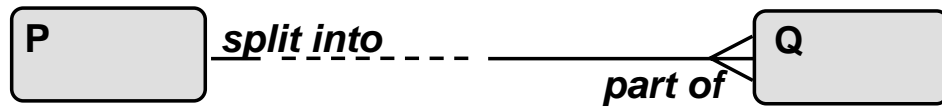
For reasons of space and readability of the diagrams in this book, relationship names are sometimes kept very short, and sometimes only a preposition is used.

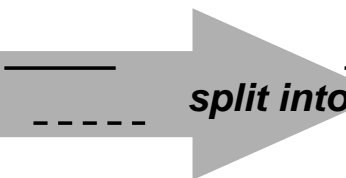
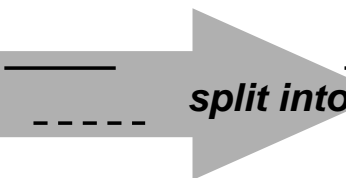
## Reading a Relationship End



ORACLE

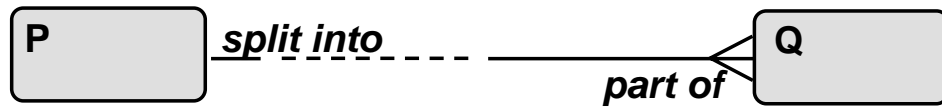
## Reading a Relationship End



“Each P must be  exactly one Q ”  
may be  one or more Qs

ORACLE

## Reading a Relationship End

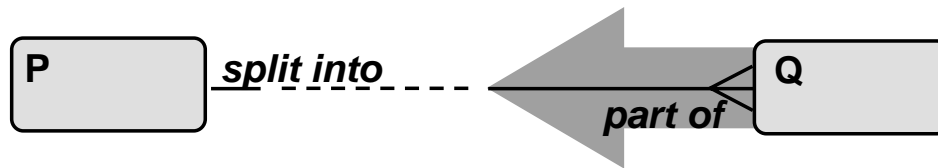


**“Each P may be *split into* one or more Qs”**

ORACLE



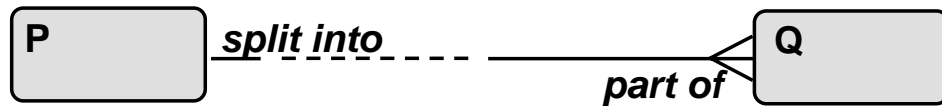
## Reading a Relationship End



***“Each P may be split into one or more Qs”***

ORACLE

## Reading a Relationship End

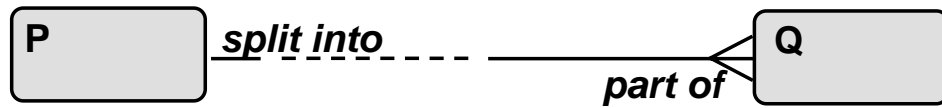


**“Each P may be split into one or more Qs”**

**“Each Q must be ——— exactly one P ”**  
**may be - - - - - one or more Ps**

ORACLE

## Reading a Relationship End



**“Each P may be *split into* one or more Qs”**

**“Each Q must be *part of* exactly one P”**

ORACLE

# Functions Drive Data

- **Business functions are always present.**
  - Explicit
  - Assumed
- **Business functions need data.**
- **An entity, attribute, or relationship may be modeled because:**
  - It is used by a business function.
  - The business need may arise in the near future.

ORACLE

1-34

Copyright © Oracle Corporation, 2002. All rights reserved.

## Functions Drive the Conceptual Data Model

Although this course does not cover the method of function modeling, functions are present at any time, in any discussion on a conceptual data model. You cannot talk about, nor judge a conceptual data model without knowing or assuming the desired functionality of the future system.

Often a conceptual data model discussion may seem to be about the data structure but actually is about functionality, usually unclear or undetermined pieces of functionality. The language used is that of the conceptual data model, the representation used is that of the entity relationship diagram, but the discussion in fact is about functionality.





Functions drive the conceptual data model. The question “Do we need to take Shoe Size for an employee?” can only be answered by answering positively the question “Is there a business function that needs it?”

Consider the conceptual data model as the shadow of the functions of a system.

Most of the time during this course, functionality is only briefly sketched, or merely assumed, to prevent you from reading page after page of functional descriptions.

# Weather Forecast

January 26

København		1/-5	➔ 3
Bremen		0/-3	↙ 4
Berlin		3/-1	↖ 3
München		5/-3	↖ 3
Amsterdam		8/3	↗ 4
Bruxelles		4/0	➔ 2
Paris		4/1	➔ 3
Bordeaux		7/2	↗ 3

ORACLE

1-35

Copyright © Oracle Corporation, 2002. All rights reserved.

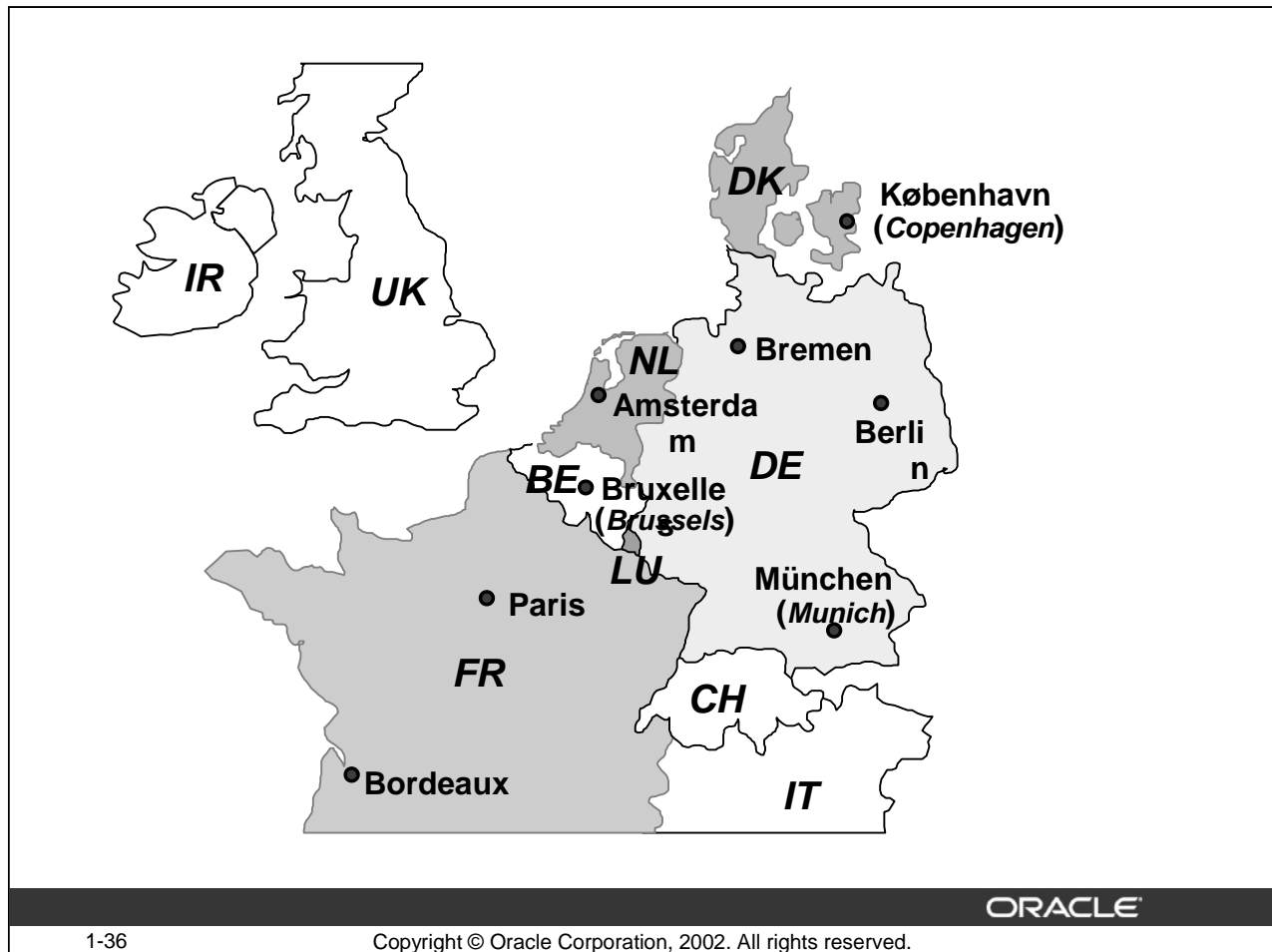
## What Information is Available?

The illustration shows a piece of a weather forecast torn from a European newspaper, showing various types of information. What are the types of information? One of the first things you will see are, for example, “København”, “Bremen”. These are cities, or more precisely, names of cities. The little drawings represent the type of weather; these drawings are icons. The next columns are temperatures, probably maximum and minimum; the arrows indicate wind direction and the number next to it is the wind force. Then there is a date on top which is the forecast date. Therefore we have:

- City
- Name of the city (such as “København”)
- Weather type (such as “cloudy with rain”)
- Icon of the weather type
- Minimum temperature
- Maximum temperature
- Wind direction arrow
- Wind force
- Forecast date

## Is this all?

No, you can find out even more information. To do this you have to have some “business” knowledge. In this case it is geographical knowledge.



## Types of Information

You may notice that the cities in the weather forecast are not printed in a random order. The German cities (Bremen, Berlin and Munchen) are grouped together, just as the French cities are. Moreover, the cities are not ordered alphabetically by name but seem to be ordered North-South. Apparently this report “knows” something to facilitate the grouping and sorting. This could be:

- Country of the city
- Geographical position of the city

and maybe even

- Geographical position of the country

## Next Step

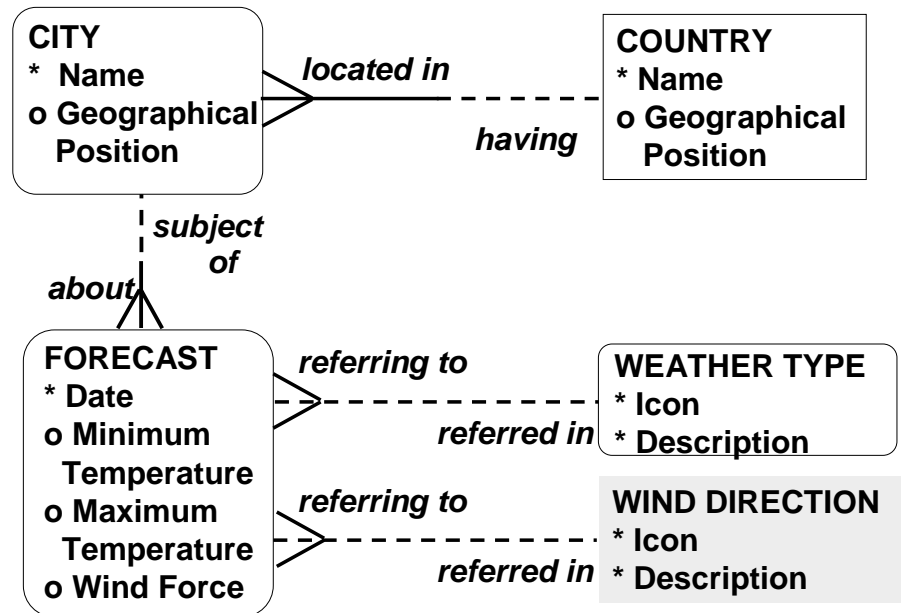
Try to identify which of the above types of information is probably an entity, which is an attribute and which is a relationship.

City and Country are easy. These are entities, both with, at least, attribute Name and Geographical Position. Weather Type could also be an entity as there is an attribute available: Icon. For the same reason there could be an entity Wind Direction. Now, where does this leave the temperatures and forecast date? These cannot be attributes of City as the forecast date is not single value for a City: there can be many forecast dates for a city. This is how you discover that there is still one entity missing, such as Forecast, with attributes Date, Minimum and Maximum Temperature, Wind Force.

There are likely to be relationships between:

- COUNTRY and CITY
- CITY and FORECAST
- FORECAST and WEATHER TYPE
- FORECAST and WIND DIRECTION

## Weather Forecast, a Solution



ORACLE

1-38

Copyright © Oracle Corporation, 2002. All rights reserved.

### Types of Information

In this entity relationship diagram some assumptions are made about the relationships:

- Every FORECAST must be about one CITY, and
- Not all CITIES must be in a FORECAST—but may be in many
- Every CITY is located in a COUNTRY, and
- Every COUNTRY has one or more CITIES
- A FORECAST must not always contain a WEATHER TYPE, and
- Not all WEATHER TYPES are in a FORECAST—but may be in many
- A FORECAST must not always contain a WIND DIRECTION, and
- Not all WIND DIRECTIONS are in a FORECAST—but may be in many

The rationale behind these assumptions is that we consider an incomplete FORECAST still to be a FORECAST, unless we do not know the date or the CITY the FORECAST refers to.



## Graphical Elements of ER Diagram

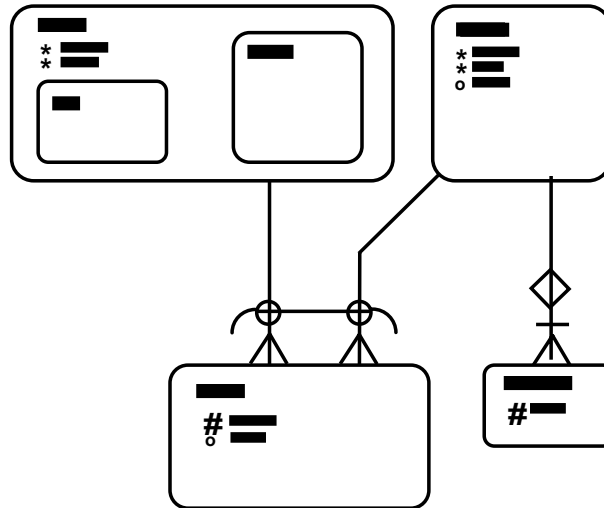
- ✓ Entity
- ✓ Attribute
- ✓ Relationship

Subtype

Unique identifier

Arc

Nontransferability



ORACLE

1-39

Copyright © Oracle Corporation, 2002. All rights reserved.

### Other Graphical Elements

The illustration shows all graphical elements you can encounter in a ER diagram. You saw earlier how to represent an entity, an attribute, and a relationship.

The lessons following this one discuss the remaining four types of elements:

- Subtype, represented as an entity within the boundary of another entity
- Unique identifier, represented as a # in front of an attribute or as a bar across a relationship line
- Arc, represented as an arc-shaped line across two or more relationship lines
- Nontransferability symbol, represented as a diamond across a relationship line

### Limited Set of Graphical Elements

As you can see, the set of graphical elements in ER diagramming is very limited. The complexity of ER modeling is clearly not in the representation. The main complexity of ER modeling lies in the understanding of the business, in the recognition of the entities that play a role in that business, the relevant attributes that describe the entities, and the relationships that connect them.

## Summary

- **ER Modeling models information conceptually**
- **Based on functional business needs**
- **“What”, not “How”**
- **Diagrams provide easy means of communication**
- **Detailed, but not too much**

ORACLE

1-40

Copyright © Oracle Corporation, 2002. All rights reserved.

### Summary

Conceptual models are created to model the functional and information needs of a business. These models may be based on the current needs but can also be a reflection of future needs. This course is about modeling the information needs. Functional needs cannot be ignored while modeling data, as these form the only legitimate basis for the data model. Ideally, the conceptual models are created free of any consideration of the possible technical problems during implementation. Consequently the model is only concerned with what the business does and needs and not with how it can be realized.

Entity Relationship modeling is a well-established technique for catching the information needs. The ER model forms the basis for the technical data model. Technical considerations take place at that level.

Entity Relationship diagrams provide an easy-to-read and relatively easy-to-create diagrammatic representation of the ER model. These diagrams initially form the foundation for the discussion of business needs. Later they provide the best possible map of a future system.

The diagrams show a fair amount of detail, but are not too detailed to become cluttered.

# Practices

- **Instance or Entity**
- **Guest**
- **Reading**
- **Hotel**
- **Recipe**

ORACLE®

## Practice: Instance or Entity?

Concept	E/A/I?	Example Instance or Entity
PRESIDENT		
ELLA FITZGERALD		
DOG		
ANIMAL		
HEIGHT		
	E	CAR
	A	CAR
	I	CAR

ORACLE

1-42

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 1-1: Instance or Entity

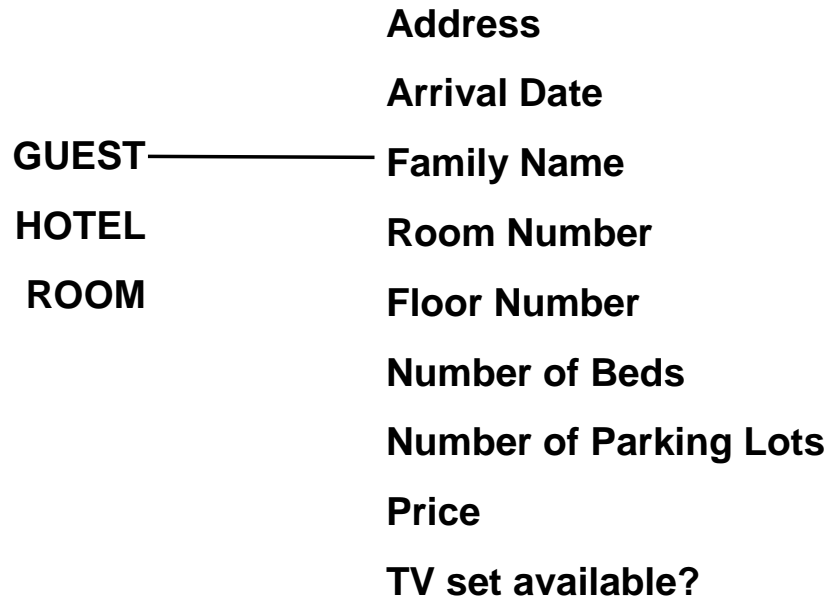
#### Goal

The goal of this practice is to learn to make a distinction between an entity, an attribute, and an instance of an entity.

#### Your Assignment

List which of the following concepts you think is an Entity, Attribute, or Instance. If you mark one as an entity, then give an example instance. If you mark one as an attribute or instance, give an entity. For the last three rows, find a concept that fits.

## Practice: Guest



ORACLE

1-43

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 1-2: Guest

#### Goal

The goal of this practice is to recognize attributes for an entity.

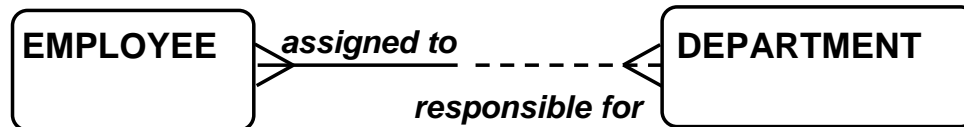
#### Scenario

On the left side of the illustration are three entities that play a role in a hotel environment: GUEST, HOTEL, and ROOM. On the right is a choice of attributes.

#### Your Assignment

Draw a line between the attribute and the entity or entities it describes.

## Practice: Reading



- A** Each EMPLOYEE may be assigned to one or more DEPARTMENTS  
Each DEPARTMENT must be responsible for one or more EMPLOYEES
- B** Each EMPLOYEE must be assigned to one or more DEPARTMENTS  
Each DEPARTMENT may be responsible for one or more EMPLOYEES
- C** Each EMPLOYEE must be assigned to exactly one DEPARTMENT  
Each DEPARTMENT may be responsible for exactly one EMPLOYEE

ORACLE

### Practice 1-3: Reading

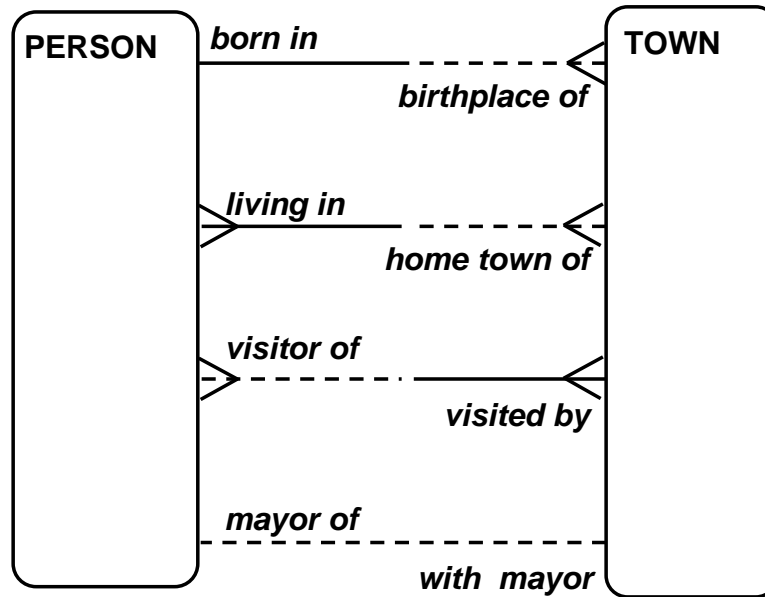
#### Goal

The goal of this practice is to read a relationship.

#### Your Assignment

Which text corresponds to the diagram?

## Practice: Read and Comment



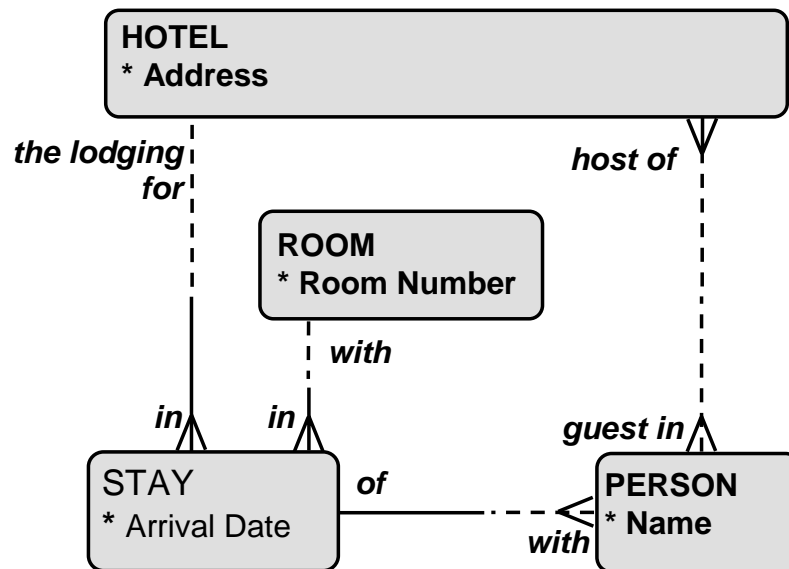
ORACLE

### Practice 1-4: Read and Comment

#### Your Assignment

1. Read each of the relationships in the model presented here.
2. Next, comment on the relationship you just read. Use your knowledge of people and towns.

## Practice: Hotel



ORACLE

### Practice 1-5: Hotel

#### Your Assignment

1. Comment on the relationships of the model presented here.
2. Make up two more possible relationships between PERSON and HOTEL that might be of some use for the hotel business.



Ralph's Raving Recipes	
Soups	<b>Açorda alentejana</b> bread soup from Portugal
vegetarian 15 min easy	For 4 persons: 1 onion 4 cloves of garlic 1 red pepper 1 liter of vegetable broth 4 tablespoons of olive oil 4 fresh eggs 1 handful of parsley or coriander salt, pepper 9-12 slices of (old) bread
Preparation	Cut the onion into small pieces and fry together with the garlic. Wash the red pepper, cut it in half, remove the seeds and fry it for at least 15.
page 127	
ORACLE	
1-47	Copyright © Oracle Corporation, 2002. All rights reserved.

## Practice 1-6: Recipe

### Goal

The goal of this practice is to discover the various types of information that are present in a given source of information.

### Scenario

You work as an analyst for a publishing company that wants to make recipes available on the Web. It wants the public to be able to search for recipes in a very easy way. Your ideas about easy ways are highly esteemed.

### Your Assignment

- Analyze the example page from Ralph's famous Raving Recipes book and list as many different types of information that you can find that seem important.
- Group the various types of information into entities and attributes.
- Name the relationships you discover and draw a diagram.



# 2

## **Entities and Attributes in Detail**

ORACLE®

Copyright © Oracle Corporation, 2002. All rights reserved.

# Overview

- **Data compared to information**
- **Entities and how to track them down**
- **Attributes**
- **Subtypes and supertypes**

ORACLE®

1-2

Copyright © Oracle Corporation, 2002. All rights reserved.

## Introduction

This lesson provides you with a detailed discussion about entities and attributes and how you can track these in various sources of information. The lesson looks at the evolution of an entity definition and the concept of subtype and supertype entity. The lesson also introduces the imaginary business of ElectronicMail Inc. which is used in many examples throughout this book.

## Objectives

At the end of this lesson, you should be able to do the following:

- Track entities from various sources
- Track attributes from various sources
- Decide when you should model a piece of information as an entity or an attribute
- Model subtypes and supertypes

# Data Compared to Information

## Data

- *Facts given from which other facts may be inferred*
- *Raw material*
- **Example: Telephone Directory**

## Information

- *Knowledge, intelligence*
- **Example: Telephone number of florist**

ORACLE

1-3

Copyright © Oracle Corporation, 2002. All rights reserved.

## Data Compared to Information

The words data and information are often used as if they are synonyms. Nevertheless, they have a different meaning.

### Data:

Raw material, from which you can draw conclusions. Facts from which you can infer new facts. A typical example is a telephone directory. This is a huge collection of facts with some internal structure.

### Information:

Knowledge, intelligence, a particular piece of data with a special meaning or function. Often information leads to data. In reverse, information is often the result of the deriving process from data—this may be a particular piece of data. If data is structured in some way, this is very helpful in the process of finding information. To expand the telephone directory data example, information is the telephone number of your dentist or the home address of a colleague.

# Data

- ***Modeling, Conceptual***  
Structuring data concepts into logical, coherent, and mutually related groups
- ***Modeling, Physical***  
Modeling the structure of the (future) physical database
- ***Base***  
A set of data, usually in a variety of formats, such as paper and electronically-based
- ***Warehouse***  
A huge set of organized information

ORACLE

1-4

Copyright © Oracle Corporation, 2002. All rights reserved.

## Data

### Conceptual Data Modeling

Conceptual data modeling is the examination of a business and business data in order to determine the structure of business information and the rules that govern it. This structure can later be used as the basis for the definition of the storage of the business data.

Conceptual data modeling is independent of possible technical implementations. For that reason, a conceptual data model is relatively stable over longer periods of time, as businesses change, often only gradually, over a period of time. Conceptual Data modeling is also called Information Engineering.

### Physical Data Modeling

Physical data modeling is concerned with implementation in a given technical software and hardware environment. The physical implementation is highly dependent on the current state of technology and is subject to change as available technologies rapidly change. A technical design made five years ago is likely to be quite outdated today.

By distinguishing between the conceptual and physical models, you separate the rather stable from the rather unstable parts of a design. This is true for both data models and functional specifications.

## **Database**

A database is a set of data. The various parts of the data are usually available in different forms, such as paper and electronic-based. The electronic-based data may reside, for example, in spreadsheets, in all kinds of files, or in a regular data base. Today, relational databases are very common; but many older systems are still around. The older systems are mostly hierarchical databases and network databases. Systems of more recent date are semantic databases and object oriented databases.

## **Data Warehouse**

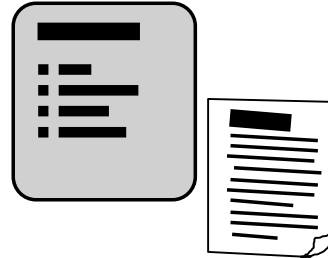
A data warehouse is composed of data from multiple sources placed into one logical database. This data warehouse database, (or, more correctly, this database structure), is optimized for Online Analytical Processing (OLAP) actions.

Often a data warehouse contains summarized data from day-to-day transaction systems with additional information from other sources. An example is a phone company that tracks the traffic load for a routing system. The system does not store the individual telephone calls, but stores the data summarized by hour.

From a data analysis point of view a data warehouse is just a database, like any other, only with very specific and characteristic functional requirements.

# Entities

- **Give the entity a unique name**
- **Create a formal description of the entity**
- **Add a few attributes, if possible**
- **Be aware of homonyms**
- **Check entity names and descriptions regularly**
- **Avoid use of reserved words**
- **Remove relationship name from entity name**



ORACLE

1-6

Copyright © Oracle Corporation, 2002. All rights reserved.

## Tracking Entities

The nouns in, for example, the texts, notes, brochures, and screens you see concerning a business often refer to entities, attributes of entities, or instances of entities.

### Naming an Entity Uniquely

First distinguish an entity by outlining the concept in your mind. Next, try to find a unique and clear name for an entity. This is not always easy as there are far more concepts than clear names. Use your imagination. Use a dictionary. Use a combination of words, use 'X' if necessary, but do not let the lack of a good name stop you from modeling. Good names evolve over time.

Check the names you used every now and then. The implicit definition of an entity may change during analysis, for instance, as a result of adding an attribute or changing the optionality of a relationship.

### Creating a Formal Description

Create a formal description of the entity. This is usually not difficult and the writing helps clarify your thinking about what you are talking about. Check this description regularly. Sometimes concepts evolve during the modeling process. The definitions, of course, should follow that evolution.

### Be Aware of Synonyms

In many business contexts one and the same concept is known under different names. Select one and mention the synonyms in the description: "...also known as ...".



## **Avoid Homonyms**

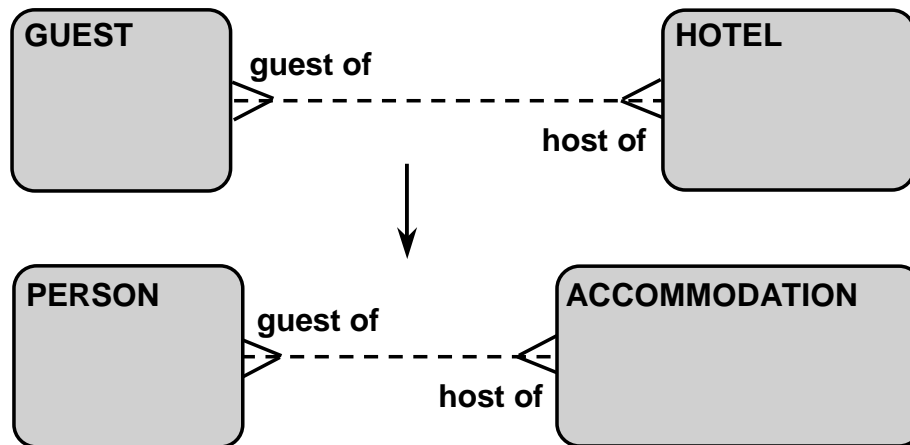
Often in a business one word is used for different concepts. Sometimes even the same person will use the same word but with different meanings as you can see in the next example.

“The data modeling course you attend now was written in 1999 and requires modeling skills to teach.” In this sentence the word “course” refers to three different concepts: a course event (like the one you are attending today), a course text (which was written in 1999) and the course type (that apparently needs particular skills).

## **Avoid Reserved Words**

Although you are free to use any name you want for an entity, try to avoid database and programming terms as entity names if possible. This may prevent naming problems and confusion later on in the design stage.

## Relationship Name in Entity Name



ORACLE

1-8

Copyright © Oracle Corporation, 2002. All rights reserved.

### Remove Relationship Name from Entity Name

Often you can select entity names in a more or less generic way. In the example, both diagrams model the same context. In the first the “guest” aspect is part of the entity name as well as part of the relationship name.

The second model is more general in its naming. There a guest is seen as a **PERSON** playing the role of being a guest.

As a rule, if there is choice take the more general name. It allows, for example, for the addition of a second relationship between the same entities that shows, for example, person is working for or is owning shares in the accommodation. The first model would require new entities.

This subject is closely related to the concept of subtypes and roles. You find more on this later in this lesson and when we discuss Patterns.

## **Some Background Information**

**“ElectronicMail (EM) wants to provide an attractive and user- friendly Web-based e-mail system. Important concepts are user and message.**

**An EM user has a unique address of 30 characters at most and a password supplied by the person who set up the EM user. Who the person really is, we do not know, although we ask for some additional information, such as the name, country, birth date, line of business, and a few more things.**

**ORACLE**

1-9

Copyright © Oracle Corporation, 2002. All rights reserved.

### **Electronic Mail Example**

In this course we investigate various business contexts. One is that of ElectronicMail, a company that supplies an e-mail service. Here is some background information.

## **Some Background Information**

**Users must be able to send and receive mail messages. A mail message is usually a piece of straight text. A message may have attached files. An attachment is a file, like a spreadsheet, that is sent and kept with the message, but not created with our software.**

**Messages are kept in folders. Every user has three folders to start with: Inbox, Outbox, and Wastebasket. Additional folders can be created by the user.”**

**ORACLE**

*advertisement area1*

**EM logo**

**Compose** Template: default

Subject: test Send

To: bipi, giovanni papini@yahoo.com Save Draft

Cc: myself Save Template

Bcc: Cancel

Message text: this is a test as well  
lalala  
pompido

Keep  
Copy

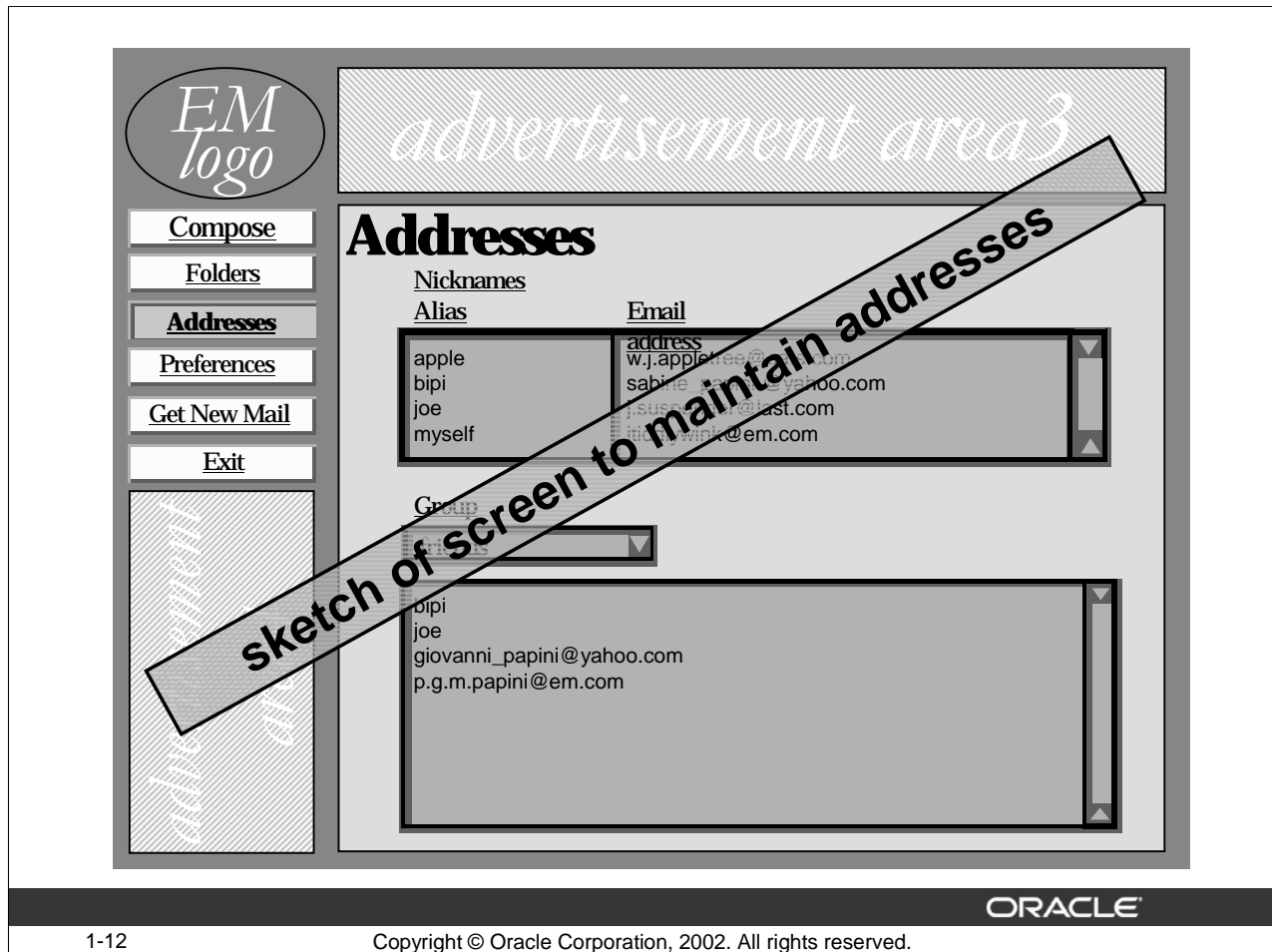
☐ Add Signature

Attachments: Type:  
abc.html Hypertext  
xyz.doc Word document

*sketch of screen to compose mail messages*

*advertisement area2*

ORACLE



### Electronic Mail Example

The screenshots may give an idea of how the Compose a Mail Message screen and the Maintain Addresses screen will look like.

## **Some Desired Functionality**

- **“Users of ElectronicMail must be able to address messages to a mail list, for example, a group of e-mail addresses. The system should only keep one copy of the message sent (to save database space) plus information about whom the message was sent to.**
- **Users must be able to create templates for their messages. A template must be named and may contain everything a real message contains. A template may be used for new messages.**

**ORACLE**

## **Some Desired Functionality**

- **Users must be able to reply to a message. By replying the user creates a new message to which the old message is added.**
- **Users must be able to create an alias for an e-mail address, to hide the often complex addresses behind an easy-to-remember nickname.”**

ORACLE



## Evolution of an Entity Definition

- A message is a piece of text sent by a user.
- A message is a piece of text sent by *an EM* user.
- A message is a note that is sent by an EM user. *A message does not necessarily contain text, nor a subject, etc.*
- A message is a note that is sent by an EM user *or received by an EM user or both*. A message does not necessarily contain text, nor a subject, etc.
- A message is a note that is *received* by an EM user. A message does not necessarily contain text, nor a subject, etc.

ORACLE

1-15

Copyright © Oracle Corporation, 2002. All rights reserved.

### Evolution of an Entity Definition

To illustrate the evolution of a concept, consider ElectronicMail's entity MESSAGE. The first intuitive description of this entity may be:

Any user? Well, no.

Must every message contain text? Should it not be possible to send a message that only transports an attachment, without additional text?

And what about a message that comes from an external source and is received by an EM user? Should those not be kept as well?

Now suppose a message is sent by an EM user to an external e-mail address only. Suppose the EM user does not want to keep a copy of the mail message. In that case there is no need for the system to keep the message as there is no internal EM user that needs the message. In fact, it is not important at all to keep messages that were sent by a EM user; only those that were actually received by an EM user are of interest.

The thinking process shown here is typical for the change of a definition from the first idea to something that is much more well thought-out—though this does not mean that the definition is final.

## **Entity Life Cycle**

It often helps to make things clear if you think about the life cycle of an entity. The life cycle refers to the functional steps of the entity. For example, how can the entity instance come into existence? How can it change? How does it disappear?

In case of entity MESSAGE the questions are:

- When does “something” become a message?
- When does a message change?
- When can a message be removed?

### **Creating a Message**

When I type in some text in the compose screen, is that text a message? You will probably agree that it does not make much sense to consider it as a message until some fields are completed, such as the To or Subject field. The checks must take place after I hit the send key. Only after all checks have been made is the message born.

### **Removing a Message**

When can the system remove a message? When a user hits the delete key? But what should the system do when there are other receivers of that same message? It is better to consider the deleting of a message as the signal to the system that you no longer need the right to read the message. When all users that did receive the same message have done this, then the message can be deleted. Apparently, for a message to exist it must have receivers that still need the message.

### **Changing a Message**

Changing a message? As long as the text is not sent, it is no problem as it is not yet considered to be a message. Changing it after sending it? Changing something that is history? This cannot be done. Changing the text should lead to a new message.

### **Draft**

What about a message that is not yet ready for sending? Suppose a user wants to finish a message at a later date. Is there a place for this? Do we want an unsent, or draft, message in the system? Is a DRAFT a special case of entity MESSAGE, or should we treat a DRAFT as a separate entity?

### **Template**

What about the templates? A template is about everything a message can be, but a template is only used as a kind of stamp for a message. Templates are named, messages are not. Is TEMPLATE a special case of entity MESSAGE, or should we look upon it as a separate entity?

## Business Functions

- “Users of ElectronicMail must be able to *address* messages to a mail list, for example, a group of e-mail addresses. The system should only keep one copy of the message sent (to save data base space) plus information about whom the message was sent to.
- Users must be able to *create* templates for their messages. A template must be named and may contain everything a real message contains. A template may be used for new messages.

ORACLE

1-17

Copyright © Oracle Corporation, 2002. All rights reserved.

### Functionality

In the previous evolution of the entity definition, the definition changes were invoked by thinking and rethinking the functionality of the system around messaging. This illustrates the statement made earlier: functions drive the conceptual data model.

The first idea of the functionality of a system, or desired functionality, can be derived from the verbs in, for example, descriptive texts and interview notes. In the above text the functionality is expressed at a high level, without much detail. Nevertheless, you can probably imagine more detailed functionality.

In this course functionality is always present, often implicitly assumed, sometimes in detail.

## Business Functions

- Users must be able to *reply* to a message. By replying the user *creates* a new message to which the old message is added.
- Users must be able to create an alias for an e-mail address, to hide the often complex addresses behind an easy-to-remember nickname.”

ORACLE

## An Attribute...

- **Always answers “of what?”**
- **Is the property of entity, not of relationship**
- **Must be single valued**
- **Has format, for example:**
  - **Character string**
  - **Number**
  - **Date**
  - **Picture**
  - **Sound**
- **Is an elementary piece of data**

ORACLE

1-19

Copyright © Oracle Corporation, 2002. All rights reserved.

### Tracking Attributes

As discussed earlier, the nouns in, for example, the texts, notes, brochures, and screens you see used in a business often refer to entities, attributes of entities, or instances of entities. You can usually easily recognize attributes by asking the questions “Of what?” and “Of what format?”. Attributes describe, quantify, qualify, classify, specify or give a status of the entity they belong to. We define an attribute as a property of an entity; this implies there is no concept of a standalone attribute.

In the background information text on ElectronicMail that is shown below, the first occurrence of the (probable) entities are capitalized, the attributes are boxed and instances are shown in italics.

## Nouns, Entities, Attributes

- “ElectronicMail (EM) wants to provide an attractive and user friendly Web-based email system. Important concepts are user and message.
- An EM *USER* has a unique *address* of 30 characters at most and a password supplied by the *PERSON* who set up the EM user. Who the person really is, we do not know, although we ask for some additional information, like the *name*, *COUNTRY*, *birth date*, *line of business*, and a few things more.

ORACLE

1-20

Copyright © Oracle Corporation, 2002. All rights reserved.

### Tracking Attributes

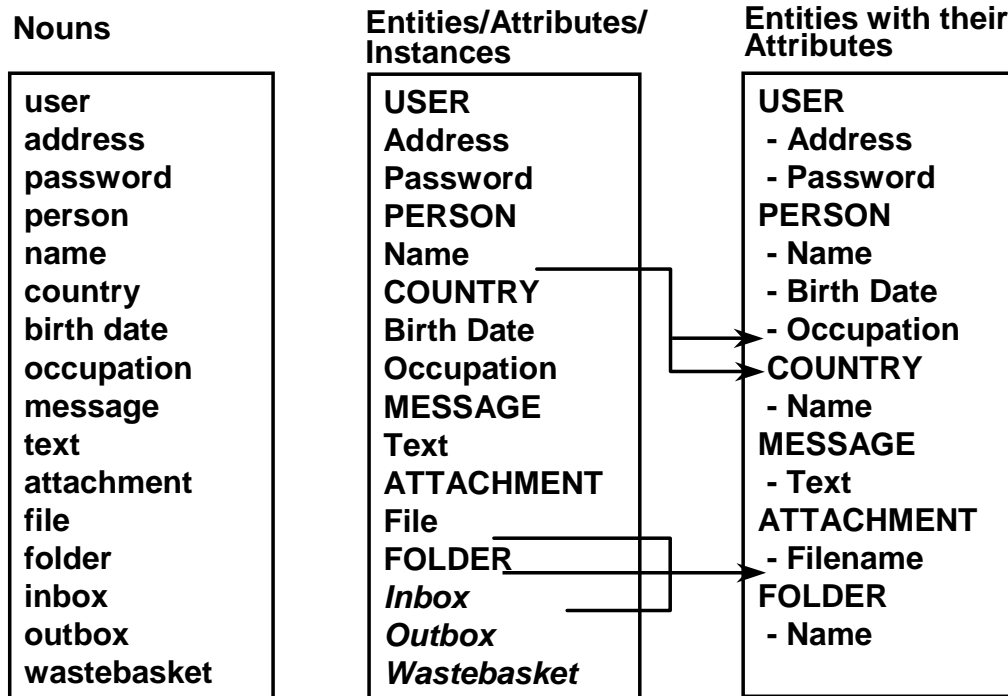
List the types of information, distinguish the probable entities and attributes and group them. Add attributes, if necessary, (like Name of COUNTRY) in the example. Distill one or more attributes from the instances (like Name of FOLDER).

## Nouns, Entities, Attributes

- Users must be able to send and receive mail **MESSAGES**. A mail message is usually a piece of straight *text*. A message may have attached files. An **ATTACHMENT** is a *file*, like a spreadsheet, that is sent and kept with the message, but not created with our software.
- Messages are kept in **FOLDERS**. Every user has three folders to start with: *Inbox*, *Outbox* and *Wastebasket*. Additional folders can be created by the user.”

ORACLE

## EM Entities and Attributes



ORACLE

1-22

Copyright © Oracle Corporation, 2002. All rights reserved.

### Naming Attributes

Attribute names become the candidate column names at a later stage. Column names must follow conventions. Try to name attributes avoiding the use of reserved words.

Do not use abbreviations, unless these were decided beforehand. Examples of frequently-used abbreviations are Id, No, Descr, Ind(icator).

Do not use attribute names like Amount, Value, Number. Always add an explanation of the meaning of the attribute name: Amount Paid, Estimated Value, Licence No.

Always put frequently-used name components, such as “date” or “indicator”, of attribute names in the same position, for example, at the end—Start Date, Creation Date, and Purchase Date.

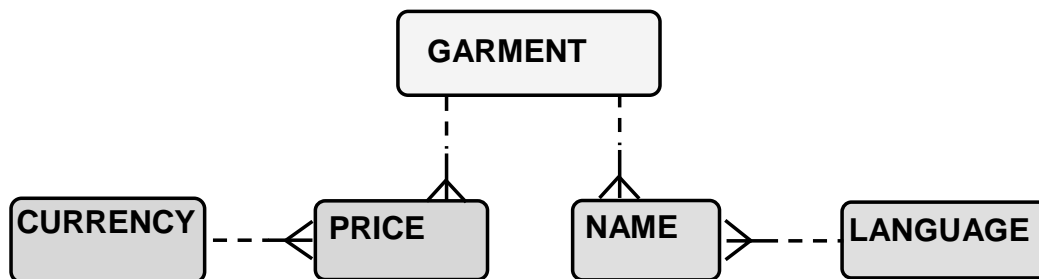
Do not use underscores in attribute names that consist of more than one word. Keep in mind that attribute names, like entity names, must be as clear and understandable as possible.



## Attribute and Entity



**Attributes in one model can be entities in another.**



ORACLE

### Entities Compared to Attributes

Sometimes a piece of information that is an attribute in one context is an entity in another context. This is purely specific to the business. A typical attribute, like Name, may need to be modeled as an entity. This happens, for example, when the model needs an extra dimension, such as the language. If product names must be kept in several languages and prices must be kept in various currencies, you may suddenly find one product has several names. For example: “This particular article of clothing is named ‘Acapulco swimming trunks’ in English, and ‘Akapulko Badehose’ in German.”

A commonly encountered dimension is time. This is discussed later.

# Redundancy

## COMMODITY

- \* Name
- \* Price exclusive VAT
- \* Price inclusive VAT
- \* VAT %

ORACLE

1-24

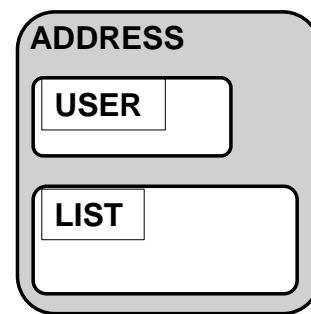
Copyright © Oracle Corporation, 2002. All rights reserved.

## Redundancy

You should take special care to prevent using redundant attributes, that is, attribute values that can be derived from the values of others. An example is shown below. Using derivable information is typically a physical design decision. This is also true for audit type attributes such as Date Instance Created, and User Who Modified.

## A Subtype ...

- Inherits all attributes of supertype
- Inherits all relationships of supertype
- Usually has its own attributes or relationships or business functions
- Is drawn within supertype
- Never exists alone
- May have subtypes of its own
- Is also known as “Subentity”



ORACLE

1-25

Copyright © Oracle Corporation, 2002. All rights reserved.

### Subtypes and Supertypes

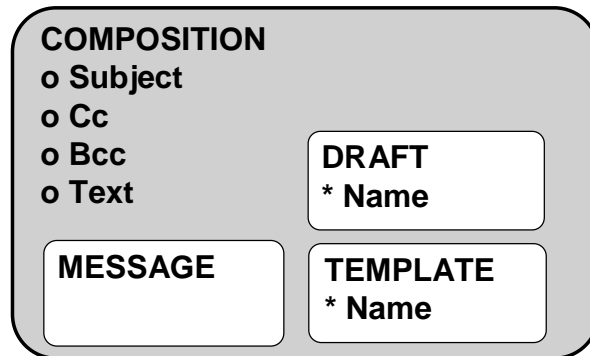
Sometimes it makes sense to subdivide an entity X into subtypes. This may be the case when a group of instances has special properties, such as attributes or relationships that only exist for that group, or a fixed value for one of the attributes, or when there is some functionality that only applies to the group. Such a group is called a subtype of X. Entity X is called the supertype as a consequence. Subtypes are also modeled when particular constraints apply to the subtype only. This is discussed further in the lesson on Constraints.

Subtypes have all properties of X and usually have additional ones. In the example, supertype ADDRESS is divided into two subtypes, USER and LIST. One thing USER and LIST have in common is an attribute NAME and the functional fact that they can both be used in the To field when writing a message.

#### Inheritance

In the next illustration, is a new entity, COMPOSITION, as a supertype of MESSAGE, DRAFT, and TEMPLATE. The subtypes have several attributes in common. These common attributes are listed at the supertype level. The same applies to relationships. Subtypes inherit all attributes and relationships of the supertype entity.

## Subtype: Example



ORACLE

### Subtype: Example

Read the diagram as:

- Every MESSAGE (DRAFT, or TEMPLATE) is a COMPOSITION

and thus has attributes like Subject and Text. Conversely:

- Every COMPOSITION is either a MESSAGE, a DRAFT, or a TEMPLATE

## Subtype: Rules

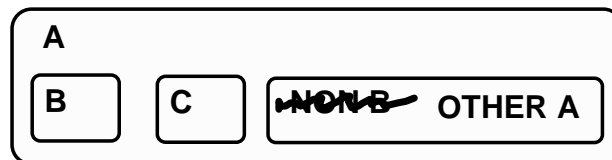
**Subtypes of the same entity must be:**

- **Exhaustive:**  
Every instance of a supertype is also instance of one of the subtypes.

and

- **Mutually exclusive:**  
Every instance of the supertype is of *one and only one* subtype.

**Name subtypes adequately:**

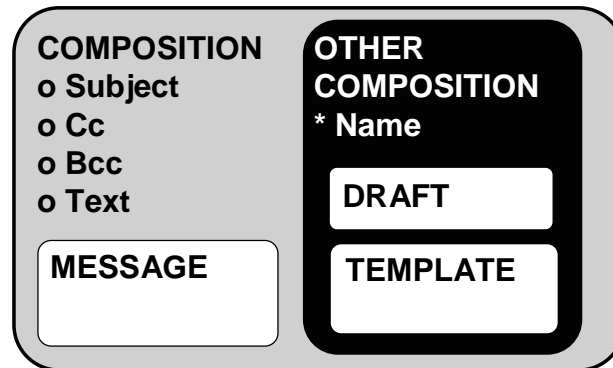


ORACLE

### Always More Than One Subtype

Entity relationship modeling prescribes that when an ER model is complete subtypes never stand alone. In other words, if an entity has a subtype, there should always be at least a second subtype. This makes sense. What use would there be for distinguishing between an entity and the single subtype? This idea leads to the two subtype rules.

## Subtypes: Three Levels



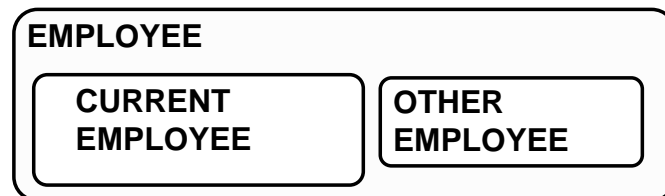
ORACLE

### Nested Subtypes

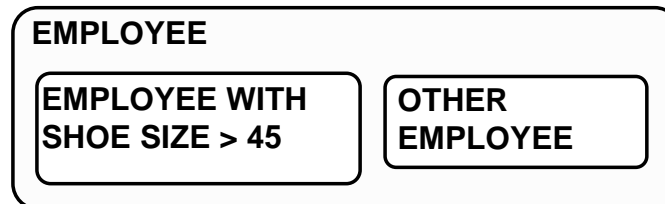
You can nest Subtypes. For readability, you would not usually subtype to more than two levels, but there is no major reason not to do so. Reconsider the placement of the attributes and relationships after creating a new level.

## More on Subtypes

Subtypes *always* exist...



... but do not all make sense



ORACLE

### Subtypes Always Exist

Every entity can always be subtyped. You can always make up a rule to subdivide the instances in groups, but that is not the issue. The reason for subtyping should always be that there is a business need to show similarities and differences at the same time.

### Implementing Subtypes

You can implement subtype entities in various ways, for example, as separate tables or as a single table, based on the super entity.

# Summary

## Entities

- **Nouns in texts**
- **Tangible, intangible, events**

## Attributes

- **Single-valued qualifiers of entities**

## Subtypes

- **Inherit all attributes and relationships of supertype**
- **May have their own attributes and relationships**

ORACLE

1-30

Copyright © Oracle Corporation, 2002. All rights reserved.

## Summary

Entities can often be recognized as nouns in texts that functionally describe a business. Entities can be tangible, intangible, and events. Subtypes of an entity share all attributes and relationships of that entity, but may have additional ones.

Attributes are single-valued elementary pieces of information that describe, qualify, quantify, classify, specify or give a status of the entity they belong to.

Most entities have attributes.

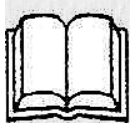
Every attribute can be promoted to a separate entity which is related to the entity the attribute initially belonged to. You must do this when you discover that the attribute is not single valued, for example, when names must be kept in multiple languages or values in multiple currencies.



# Practices

- **Books**
- **Moonlight Coffees**
- **Shops**
- **Subtypes**
- **Schedule**
- **Address**

ORACLE®



1. I have just finished writing a book. It's a novel about justice and power.
2. We have just published this book. The hard cover edition is available now.
3. Did you read that new book on Picasso? I did. It's great!
4. If you like you can borrow my book.
5. I have just started translating this book into Spanish. I use the modern English text as a basis and not the original, which is 16th century.
6. I ordered that book for my parents.

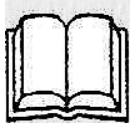
## Practice 2-1: Books

### Goal

The goal of this practice is to differentiate between various meanings of a word used in a text.

### Your Assignment

- In this text the word book is used with several meanings. These meanings are different entities in the context of a publishing company or a book reseller. Try to distinguish the various entities, all referred to as book. Give more adequate names for these entities and make up one or two attributes to distinguish them.
- Create an ER model based on the text. Put the most general entity at the top of your page and the most specific one at the bottom. Fit the others in between. Do not worry about the relationship names.



7. **Yes, we have that book available. You should find it in Art books.**
8. **A second printing of the book War and Peace is very rare.**
9. **I think My name is Asher Lev is one of the best books ever written. Mine is autographed.**
10. **I want to write a book on entity relationship modeling when I retire.**

## **Practice 2-1: Books**

### **Goal**

The goal of this practice is to differentiate between various meanings of a word used in a text.

### **Your Assignment**

- In this text the word book is used with several meanings. These meanings are different entities in the context of a publishing company or a book reseller. Try to distinguish the various entities, all referred to as book. Give more adequate names for these entities and make up one or two attributes to distinguish them.
- Create an ER model based on the text. Put the most general entity at the top of your page and the most specific one at the bottom. Fit the others in between. Do not worry about the relationship names.



## Summary

- **Moonlight Coffees is a fast growing chain of high quality coffee shops with currently over 500 shops in 12 countries of the world. Shops are located at first-class locations, such as major shopping, entertainment and business areas, airports, railway stations, museums. Moonlight Coffees has some 9,000 employees.**

### Products

- **All shops serve coffees, teas, soft drinks, and various kinds of pastries. Most shops sell nonfoods, like postcards and sometimes even theater tickets.**

## Practice 2-2: Moonlight

### Scenario

You work as a contractor for Moonlight Coffees Inc. One of your colleagues, who is a business analyst, has prepared some documentation. Below you find an extract from the summary document.

### Your Assignment

- Make a list of about 15 different entities that you think are important for Moonlight Coffees. Use your imagination and common sense and, of course, use what you find in the summary that is printed below.
- Write a formal definition of the entity that represents:
  - The coffee shops.
  - The Moonlight employees.



## Summary

### Financial

Shop management reports sales figures on a daily basis to Headquarters, in local currency. Moonlight uses an internal exchange rates list that is changed monthly. Since January 1, 1999, the European Community countries must report in Euros.

### Stock

Moonlight Coffees is a public company; stock is traded at NASDAQ, ticker symbol MLTC. Employees can participate in a stock option plan.

ORACLE

1-35

Copyright © Oracle Corporation, 2002. All rights reserved.

## Practice 2-2: Moonlight

### Scenario

You work as a contractor for Moonlight Coffees Inc. One of your colleagues, who is a business analyst, has prepared some documentation. Below you find an extract from the summary document.

### Your Assignment

- Make a list of about 15 different entities that you think are important for Moonlight Coffees. Use your imagination and common sense and, of course, use what you find in the summary that is printed below.
- Write a formal definition of the entity that represents:
  - The coffee shops.
  - The Moonlight employees.

## Shop List

**Shoplist, ordered to date opened page 4**

**181 The Flight, JFK Airport terminal 2, New York, USA,  
212.866.3410, Airport, 12-oct-97**

**182 Hara, Kita Shinagawa,Tokyo, JP, 3581.3603/4,  
Museum, 25-oct-97**

**183 Phillis, 25 Phillis Rd, Atlanta, USA, 405.867.3345,  
Shopping Centre, 1-nov-97**

**184 JFK, JFK Airport terminal 4, New York, USA,  
212.866.3766, Airport, 1-nov-97**

ORACLE

### Practice 2-3: Shops

#### Scenario

You work as a contractor for Moonlight Coffees. Your task is to create a conceptual data model for their business. You have collected all kinds of documents about Moonlight. Below is a page of a shop list.

#### Your Assignment

Use the information from the list as a basis for an ER model. Pay special attention to find all attributes.

## Shop List

**185 VanGogh, Museumplein 24, Amsterdam, NL,  
76.87.345, Museum, 10-nov-97**

**186 The Queen, 60 Victoria Street, London, UK,  
203.75.756, Railway Station, 25-nov-97**

**187 Wright Bros, JFK Airport terminal 1, New York,  
USA, 212.866.9852, Airport, 6-jan-98**

**188 La Lune, 10 Mont Martre, Paris, FR, 445 145 20,  
Entertainment, 2-feb-98**

### Practice 2-3: Shops

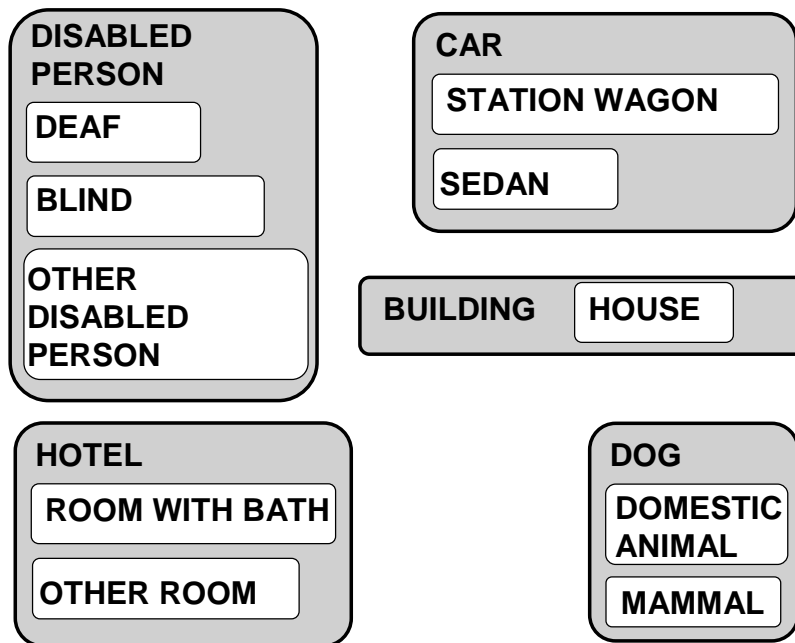
#### Scenario

You work as a contractor for Moonlight Coffees. Your task is to create a conceptual data model for their business. You have collected all kinds of documents about Moonlight. Below is a page of a shop list.

#### Your Assignment

Use the information from the list as a basis for an ER model. Pay special attention to find all attributes.

## Subtypes



ORACLE

1-38

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 2-4: Subtypes

#### Goal

The goal of this practice is to determine correct and incorrect subtyping.

#### Your Assignment

Find all incorrect subtyping in the illustration. Explain why you think the subtyping is incorrect. Adjust the model to improve it.



**van Gogh, Museumplein, Amsterdam**

Schedule Oct 12 - Oct 18				prepared by Janet			
Shift	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Annet S			2		2	2	1
Annet B	1				1	1	
Dennis	2	2	1	2	3		
Jürgen						5	4
Kiri		3			4	4	
Wit							

ORACLE

## Practice 2-5: Schedule

### Scenario

You work as a contractor for Moonlight Coffees.

### Your Assignment

Use the schedule that is used in one of the shops in Amsterdam as a basis for an entity relationship model. The schedule shows, for example, that in the week of 12 to 18 October Annet B is scheduled for the first shift on Monday, Friday, and Saturday.

The scheme suggests there is only one shift per person per day.

## Practice: Address (1/2)

Rheingasse 123  
53111 Bonn  
Germany

34 Oxford Road  
Reading  
Berkshire RG1 8JS  
UK

1020 Maple Drive  
Kirkland WA 98234  
USA

ORACLE

1-40

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 2-6: Address

#### Goal

The goal of this practice is to sort out various ways of modeling addresses.

#### Your Assignment

An entity, possibly PERSON (or ADDRESS) may have attributes that describe the address as in the examples below.

1. How would you model the address information if the future system is required to produce accurate international mailings?

## Practice: Address (2/2)

**P.O. Box 66708  
Nairobi  
Kenya**

**c/o Mrs Smith  
Maude Street  
Sandton  
Johannesburg 2144  
South Africa**

ORACLE

1-41

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 2-6: Address

#### Your Assignment

1. Would your model from the previous practice also accept the addresses below?
2. Check if your model would be different if the system is also required to have facilities to search addresses in the following categories. Make the necessary changes, if any.

#### All addresses:

- In Kirkland
- With postal code 53111 in Bonn
- That are P.O. Boxes
- On:
  - Oxford Road or
  - Oxford Rd or
  - OXFORD ROAD or
  - OXFORD RD
- in Reading



# 3

## Relationships in Detail

ORACLE®

Copyright © Oracle Corporation, 2002. All rights reserved.

# Overview

- **Relationships**
- **Ten different relationship types**
- **Nontransferability**
- **Relationships that seem to have attributes**
- **Rules of Normalization**

ORACLE

3-2

Copyright © Oracle Corporation, 2002. All rights reserved.

## Introduction

This lesson discusses in detail how to establish a relationship between two entities. You meet the ten types of relationship and examples of the less frequent types. This lesson looks at nontransferable relationships and discusses the differences and similarities between relationships and attributes. It also provides a solution for the situation where a relationship seems to have an attribute. Finally, the rules of normalization are discussed in the context of conceptual models.

## Objectives

At the end of this lesson, you should be able to do the following:

- Create a well-defined relationship between entities
- Identify which relationship types are common and which are not
- Give real-life examples of uncommon relationship types
- Choose between using an attribute or a relationship to model particular information
- Resolve a m:m relationship into an intersection entity and two relationships
- Resolve other relationships and know when to do so
- Rules of Normalization

## Establishing a Relationship

- **Determine the existence of a relationship**
- **Choose a name for the relationship from both perspectives**
- **Determine optionality**
- **Determine degree**
- **Determine nontransferability**

ORACLE

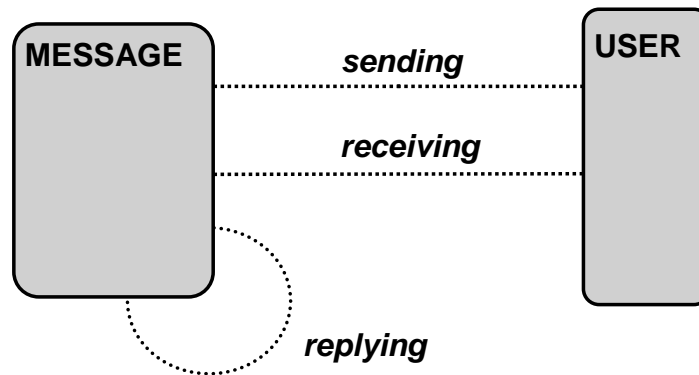
3-3

Copyright © Oracle Corporation, 2002. All rights reserved.

### Determining the Existence of a Relationship

- Ask, for each of your entities, if it is somehow related to one or more of the entities in your model, and, if so, draw a dotted “skeleton” relationship line.
- Usually all entities in a model are related to at least one other entity. Exceptions are rare, but they do exist.
- Two entities can be related more than once. For example, in the Electronic Mail system there are two relationships between entities MESSAGE and USER, one is about who is *sending* a MESSAGE and one about who *receives* a MESSAGE.
- An entity can be related to itself. This is called a recursive relationship. For example, a MESSAGE can be a reply to another MESSAGE. See the paragraph on recursive relationships for more details on this.

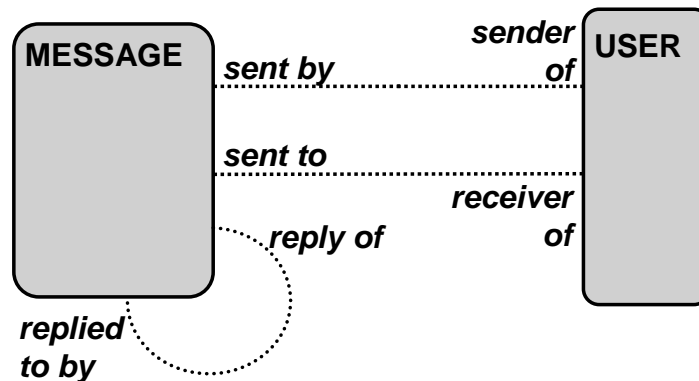
## Establishing the Relationship



ORACLE



## Relationship Names



ORACLE

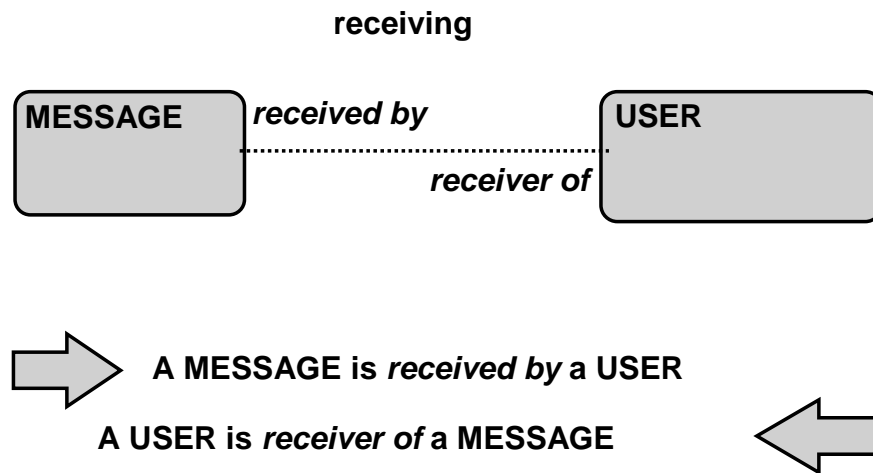
3-5

Copyright © Oracle Corporation, 2002. All rights reserved.

### Choosing a Name for the Relationship

- Sometimes the relationship name for the second perspective is simply the passive tense of the other one, such as *is owner of* and *is owned by*. Sometimes there are distinct words for both concepts, such as *parent of / child of* or *composed of / part of*.
- Try to use names that end in a preposition.
- If you cannot find a name, you may find these relationship names useful:
  - Consists of / is part of
  - Is classified as / is classification for
  - Is assigned to / is assignment of
  - Is referred to / referring to
  - Responsible for / the responsibility of
- Sometimes a very short name is sufficient, for example, *with, in, of, for, by, about, at, into*.

## Naming the Relationship



ORACLE

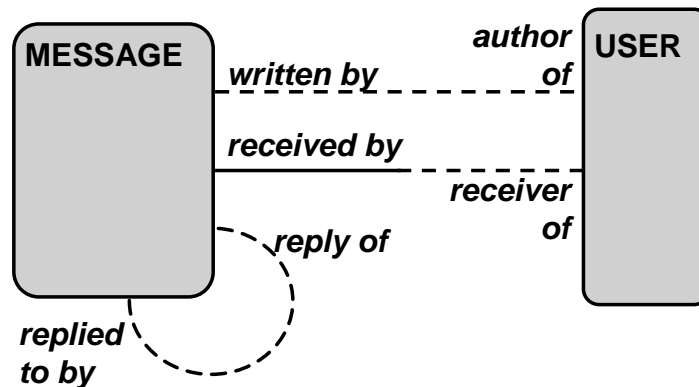
3-6

Copyright © Oracle Corporation, 2002. All rights reserved.

### Naming the Relationship

Are *sent to* and *receiver of* really opposite? If so, the assumption is that if a MESSAGE is sent to a USER, it also arrives. Maybe it is safer to name the relationship *received by* / *receiver of*...

## Optionality



ORACLE

3-7

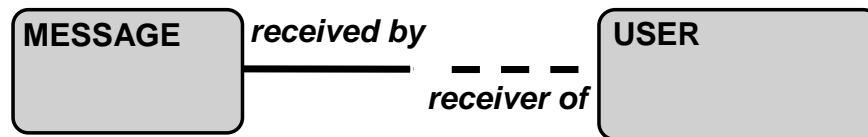
Copyright © Oracle Corporation, 2002. All rights reserved.

### Determining Optionality of Both the Relationship Ends

- Answer the questions:
  - Must every MESSAGE be sent by a USER?
  - Must every USER be sender of an MESSAGE?
  - Must every MESSAGE be sent to a USER?
  - Must every USER be addressed in a MESSAGE?
- When an answer is Yes the relationship end is mandatory, otherwise it is optional.
- Be careful at this point. Often a relationship end *seems* to be mandatory, but actually it is not. In the ElectronicMail example it seems that every MESSAGE *must* be sent by a USER. But a MESSAGE that was sent by an external user to an internal USER has no relationship to a USER, unless the system were to keep external users as well.
- Sometimes a relationship is ultimately mandatory, but not initially. Such a relationship should be modeled as optional.

## Optionality

No: \_ \_ \_ \_ . Yes: \_\_\_\_\_



- **Must every MESSAGE be received by a USER?** *Yes*
- **Must every USER be receiver of a MESSAGE?** *No*

ORACLE

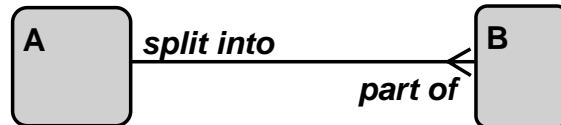
3-8

Copyright © Oracle Corporation, 2002. All rights reserved.

### Determining Degree of Both the Relationship Ends

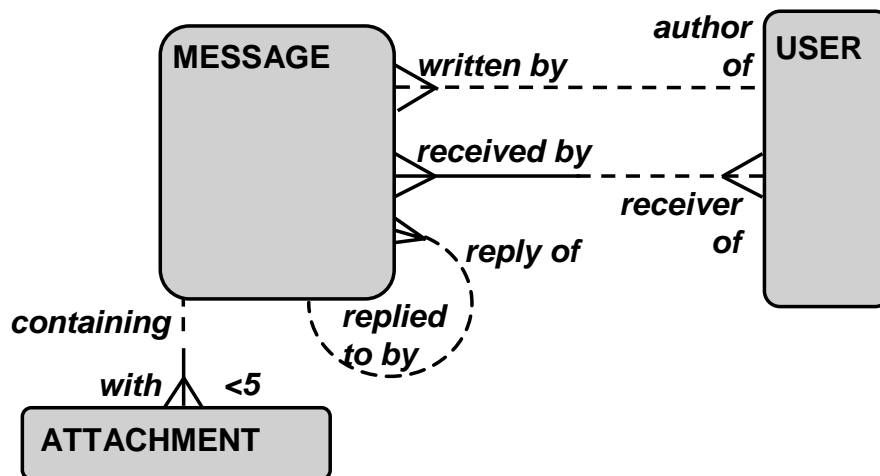
- Answer the questions:
  - Can a MESSAGE be written by more than one USER?
  - Can a USER be author of more than one MESSAGE?
- If the answer is No the degree is called “1”.
- If the answer is Yes the degree is called “many” or just “m”.
- This must be determined for all relationship ends.
- Note that a **mandatory** “*many*” relationship end from A to B does not mean that it is mandatory for A to be *split into more* than on

## Mandatory 1: Mandatory m



- Every A must be *split into* at least one B
- Every B must be *part of* exactly one A

## Degree



ORACLE

3-10

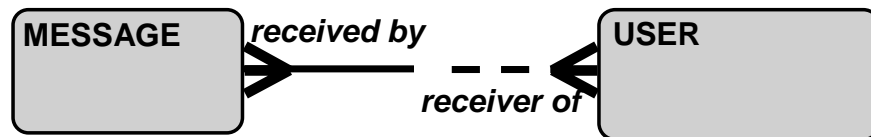
Copyright © Oracle Corporation, 2002. All rights reserved.

### Degree

- An *optional “many”* relationship end means *zero*, one or more. In the e-mail example a USER can be author of 0,1 or more MESSAGES.
- Sometimes the degree is a fixed value, or there is a maximum number. Assume a MESSAGE may be *containing* one or more ATTACHMENTS, but for some business reason, the number of ATTACHMENTS per MESSAGE may not exceed 4. The degree then is *<5*. The diagram, however, shows a crow'sfoot.

## Degree

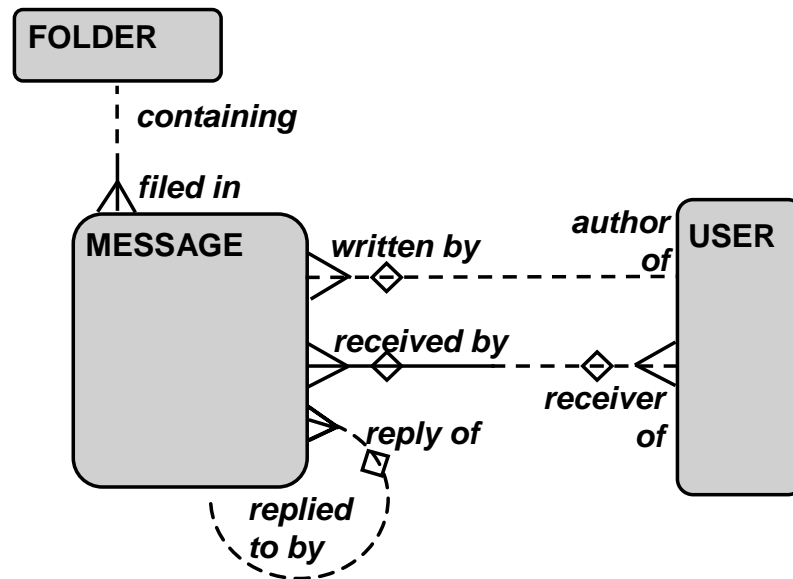
One: — One or more: ≤



- Can a MESSAGE be received by *more than one* USER? Yes
- Can a USER be the receiver of *more than one* MESSAGE ? Yes

ORACLE

# Nontransferability



ORACLE

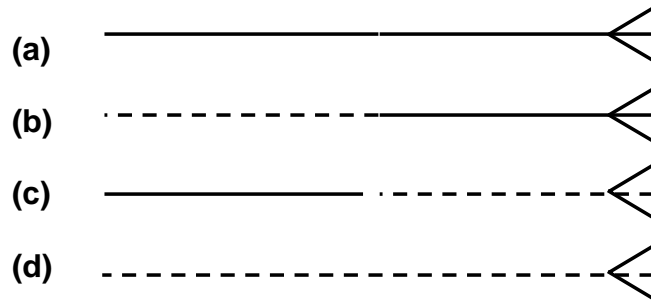
## Determine Nontransferability of Both the Relationship Ends

- When a MESSAGE is created, the USER who is the author of the MESSAGE is a fact. It would be strange if a mail system allowed you to change the author after the MESSAGE is completed.
- Often relationships have the following property: you cannot change the connection, once made. That property is called nontransferability. Nontransferability leads to nonupdatable foreign keys. Nontransferability is shown in the diagram with a little diamond-shaped symbol through the line of the relationship end.
- Not all relationships are nontransferable. Assume the mail system allows a user to file a MESSAGE in a FOLDER. This is only a valuable functionality if the user is allowed to change the FOLDER in which a MESSAGE is filed.



## Relationship Types

### 1:m



ORACLE

3-13

Copyright © Oracle Corporation, 2002. All rights reserved.

### Relationship Types

There are three main groups of relationships, named after their degrees:

- One to many (1:m)
- Many to many (m:m)
- One to one (1:1)

This paragraph discusses the various types and gives some examples of their variants.

#### Relationships—1:m

The various types of 1:m relationships are most common in an ER Model. You have seen several examples already.

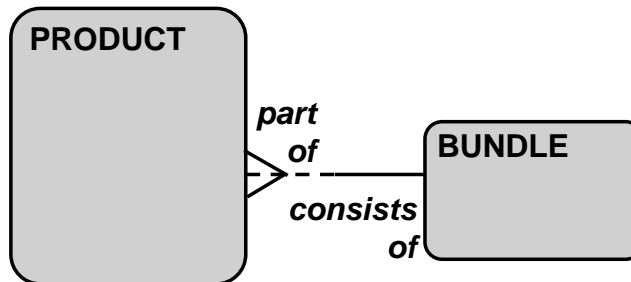
1. Mandatory at both ends. This type of relationship typically models entities that cannot exist without each other. Often the existence of mandatory details for a master is more wishful thinking than a strict business rule. Often the relationship expresses that an entity is always split into details. Seen from the other perspective, it often expresses an entity that is always classified, assigned.

### **Circumventing Mandatory 1 to Mandatory m**

Usually you would try to avoid relationship type (a) in favor of type (b), by taking a different perspective on the subject. For example, suppose an order is defined as something with at least one order item. In other words, an order is regarded as a composed concept. You can avoid modeling order as an entity as you can decide to model a slightly different concept instead, say ORDER HEADER. Next, define an ORDER HEADER to have zero, one or more ORDER ITEMS. An *order* would then be a thing composed of two entities: any ORDER HEADER with one or more ORDER ITEMS. Empty headers would not be considered to be an order.

## Relationship Types

### m:1



ORACLE

3-15

Copyright © Oracle Corporation, 2002. All rights reserved.

## Relationship Types

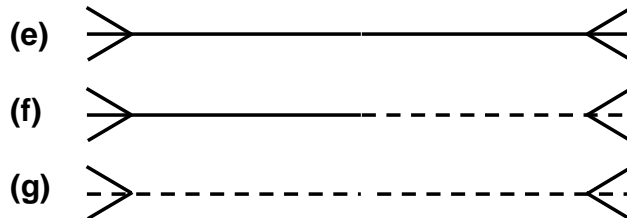
### Why Circumvent?

Implementing a 1:m relationship that is mandatory at both ends causes technical problems. In particular it is difficult to make sure details exist for a newly-created record. In most relational database environments it is even impossible.

2. Optional 1: mandatory m. This is a very common type of relationship, together with (d). Normally, at least 90% all relationships are of type (b) and (d). The relationship expresses that the entity at the 1-end can stand alone, whereas the entity at the many end can only exist in the context of the other entity.
3. Mandatory 1: optional m. This is not common. You will see it only when the relationship expresses that an entity instance only exists when it is a non-empty set, and where the elements of the set can exist independently. In the example below a PRODUCT may be part of one BUNDLE. According to the model, a BUNDLE is of no interest if it is empty.
4. Optional at both ends. See remarks for (b).

## Relationship Types

### m:m



ORACLE

3-16

Copyright © Oracle Corporation, 2002. All rights reserved.

## Relationship Types

### Relationships—m:m

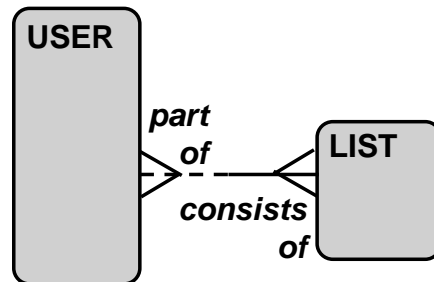
The various types of m:m relationships are common in a first version of an ER Model. In later stages of the model most m:m relationships, and possibly all, will disappear.

5. Mandatory at both sides is very uncommon in normal circumstances. This relationship seems to mean that an entity instance can only be created if it is immediately assigned to an instance of the other entity, as well as conversely. But how can this occur when we do not have an instance of either entity? Enforcing the mandatory rule from scratch leads to a conflict.

The relationship can, however, be part of a model of a theoretical nature, like the mathematical: a LINE always consists of many POINTS and a POINT is always part of many LINES. It can also describe an existing situation: a DEPARTMENT always has EMPLOYEES and an EMPLOYEE is always assigned to a DEPARTMENT. Here the question may arise if it is guaranteed that the situation will always remain this way. A m:m relationship that is mandatory at both sides can occur when the relationship is part of an arc. See the lesson on Constraints for more details.

## Relationship Types

### m:m



ORACLE

3-17

Copyright © Oracle Corporation, 2002. All rights reserved.

### Relationship Types

6. Mandatory at one end is not uncommon in early versions of a model although they usually disappear at a later stage.
7. Optional at both ends is common in early versions of a model. These also usually disappear at a later stage.

## Relationship Types

### 1:1

- (h) \_\_\_\_\_
- (i) \_\_\_\_\_
- (j) \_\_\_\_\_

ORACLE

3-18

Copyright © Oracle Corporation, 2002. All rights reserved.

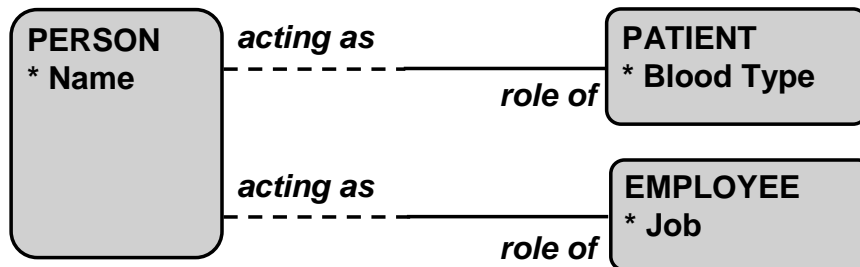
## Relationship Types

### Relationships—1:1

Usually you will find just a few of the various types of 1:1 relationships in every ER Model.

8. A 1:1 relationship, mandatory at both ends, tightly connects two entities: when you create an instance of one entity there must be exactly one dedicated instance for the other simultaneously; for example, entity PERSON and entity BIRTH. This leads to the question why you want to make a distinction between the two entities anyway. The only acceptable answer is: only if there is a functional need.  
If you have this relationship in your model, it is often, possibly always, part of an arc.
9. Mandatory at one end is often in a model where roles are modeled, for example, in this hospital model.

## 1:1 Relationships Roles



ORACLE

3-19

Copyright © Oracle Corporation, 2002. All rights reserved.

### Relationship Roles

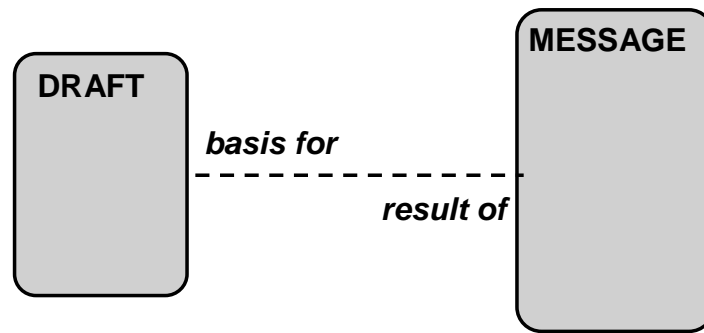
**Note:** These role-based relationships are often named is/is type of or simply is/is.

Both PATIENT and EMPLOYEE are roles played by a PERSON. The attribute BLOOD TYPE is, according to this model, only of interest when this person is a PATIENT. Note that PATIENT and EMPLOYEE cannot be modeled as subtypes of PERSON, as a PERSON may play both roles. You meet the concept of roles again in a later lesson.

10. Optional at both ends is uncommon. However, they can occur, for example, when there is a relationship between two entities that are conceptually the same but exist in different systems. An example of this is entity EMPLOYEE in one system and entity PERSON in a different, possibly a third-party, system.

Many 1:1 relationships (of all three variants) do occur when some of the entities represent various stages in a process, such as in the next example. Relationship names in this case can always be leads to / result of or something similar.

## 1:1 Relationships Process



ORACLE®

3-20

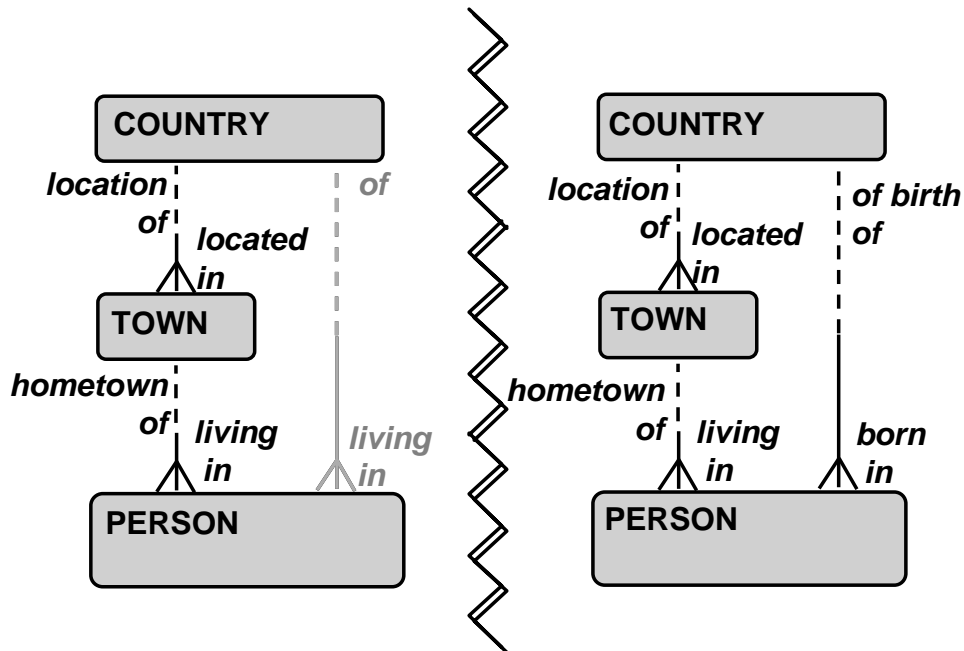
Copyright © Oracle Corporation, 2002. All rights reserved.

### Relationship Roles

If you consider a person to be a process as well, the earlier example of BIRTH and PERSON fit nicely into this general idea.



## Redundant Relationships



ORACLE

3-21

Copyright © Oracle Corporation, 2002. All rights reserved.

### Redundancy

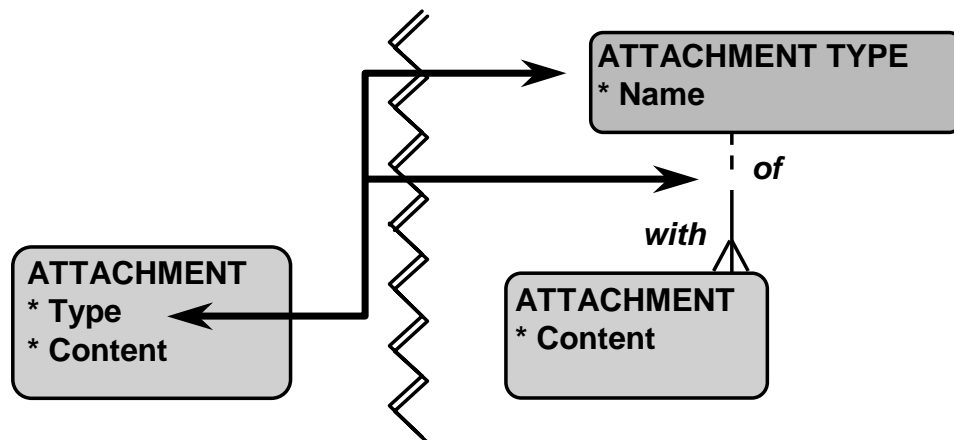
Like attributes, relationships can be redundant.

In the left-hand example you can derive the relationship from PERSON to COUNTRY from the other two relationships and you should remove them from the model.

This is a semantic issue and cannot be concluded from the structure alone, as the right-hand example shows.

## Relationships and Attributes

- An attribute can hide a relationship
- Relationship can be “downgraded” to attribute



ORACLE

3-22

Copyright © Oracle Corporation, 2002. All rights reserved.

### Relationships and Attributes

Attributes can hide a relationship. In fact, any attribute can hide a relationship.

In the example, attribute **TYPE** of entity **ATTACHMENT** can be replaced by an entity **ATTACHMENT TYPE** plus a relationship from **ATTACHMENT** to **ATTACHMENT TYPE**.

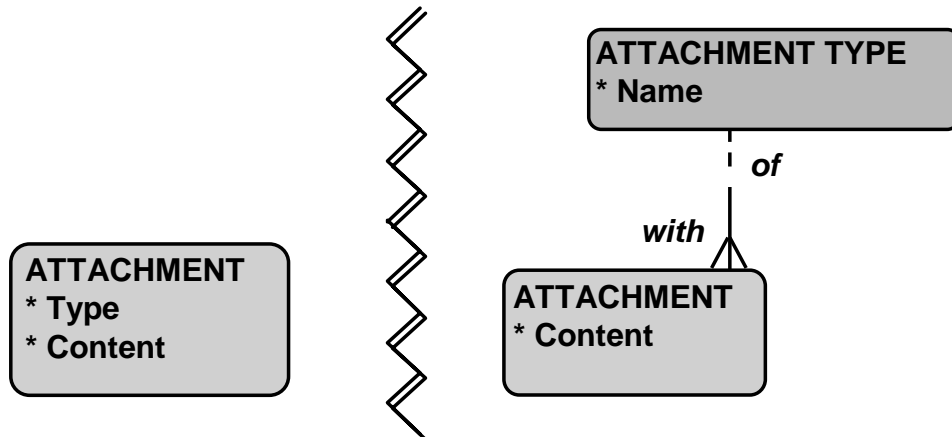
You would have no choice other than to model it this way as soon as you need to keep extra attributes for **ATTACHMENT TYPE**. If there are no important attributes for **ATTACHMENT TYPE** to keep other than the **Name** of the type, you could consider removing the entity and take **Type** as an attribute of **ATTACHMENT**.

You could also consider using the left-hand option when the number of types is a *fixed* and *small* amount, such as in the context of a chain of hotels where there are only three types of rooms: single, double, and suite.

## Attribute Compared to Relationship

- Easy model
- Fewer tables
- No join

- Value control
- List of values
- Other relationships



ORACLE

### Attribute Compared to Relationship

The table based on entity **ATTACHMENT** would contain the same columns in both situations, but the Attachment Type Name column would be a foreign key column in the second implementation. This would mean that an Attachment Type Name entered for an **ATTACHMENT** can only be taken from the types listed in the table based on entity **ATTACHMENT TYPE**. The list serves as a pick list and spelling check.

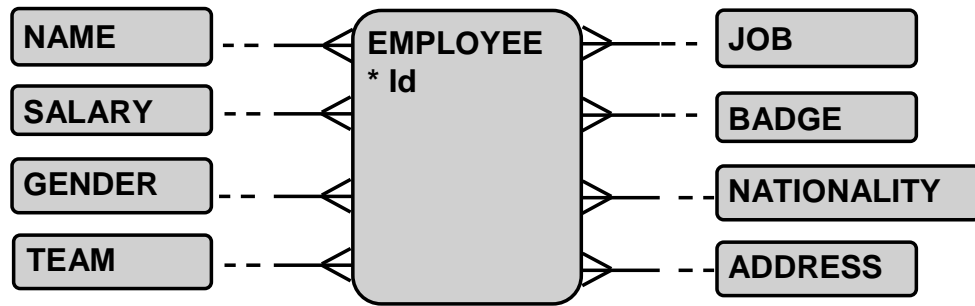
There are advantages and disadvantages for both models.

The one entity model is somewhat easier to read because it is less packed with lines. In the table implementation you would need no joins to get the required information.

However, a two-entity model is usually far more flexible. It leaves the option open to create relationships from other entities to the new entity. You would have control over the values entered as they are checked against a given set. Usually, the two-table implementation takes less (sometimes even much less) space in the database.

Use your common sense when you select the attributes and entities.

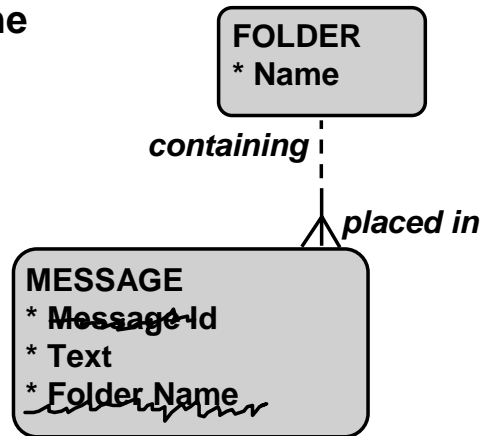
## Attribute or Entity



ORACLE

## Attribute Compared to Relationship

- There is no such thing as a foreign key attribute
- Usually, the attribute name should not contain an entity name



ORACLE

3-25

Copyright © Oracle Corporation, 2002. All rights reserved.

## Attribute Compared to Relationship

### Nonexistence of Foreign Key Attributes

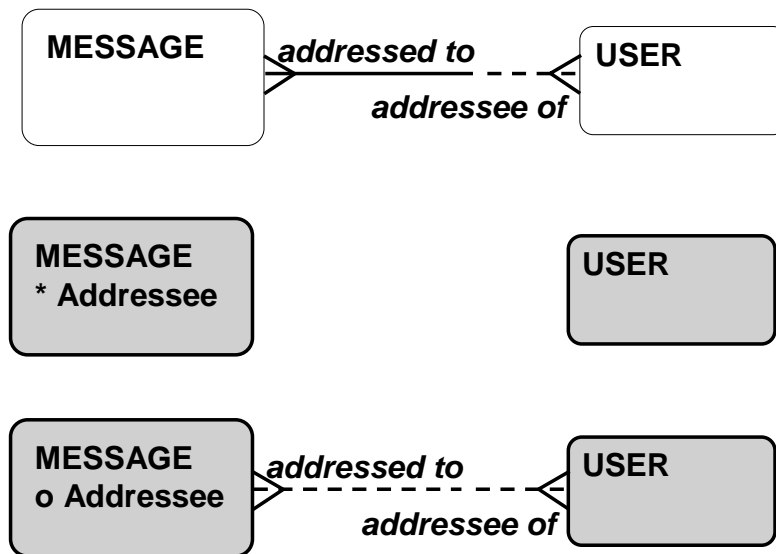
Be aware of foreign key attributes such as attribute Folder Name of entity MESSAGE in the example. In ER modeling there is no such thing as a foreign key attribute. The future foreign key is represented by the relationship between MESSAGE and FOLDER. A foreign key column (or columns) will result from the primary unique identifier of the entity FOLDER. See the lesson on CONSTRAINTS for more details on unique identifiers.

### No Entity Name in Attribute Name

When an attribute name contains an entity name, it usually comes from one of the following situations:

- The attribute hides a relationship to an entity, as in the above example. The second entity was probably added in a later stage.
- The attribute hides an entity. A typical example is an attribute Employment Date of entity EMPLOYEE. This might hide the entity EMPLOYMENT, as there is probably no rule that an employee may be employed by the same company only once.
- The entity name in the attribute name is redundant. A typical example is attribute Message Id of entity MESSAGE. The name “Id” would suffice.
- The attribute is the result of a one-to-one relationship that is not modeled, for example, attributes Birth Date and Birthplace of entity EMPLOYEE. These are in fact attributes of an entity BIRTH that is not (and probably will never be) modeled.

## Relationship Compared to Attribute



ORACLE

3-26

Copyright © Oracle Corporation, 2002. All rights reserved.

### Relationship Compared to Attribute

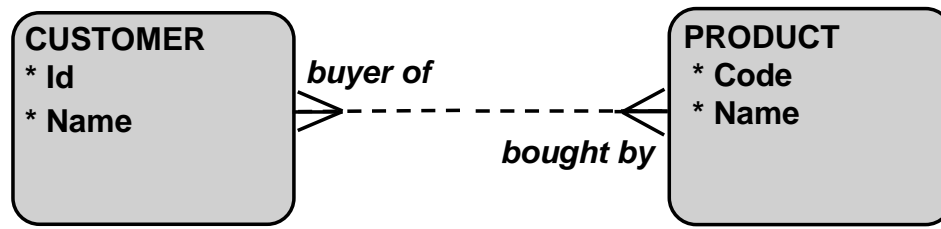
Sometimes a piece of information looks like a relationship between entities, but actually is not a relationship.

In ElectronicMail's Compose Message screen there is a field labeled "To" where the user is supposed to enter the names of the addressees. Initially you may want to model that as a relationship *addressed to / addressee of* between MESSAGE and USER, but this is a questionable approach. If a message is sent to an external user would it make sense for ElectronicMail to keep track of all external user addresses that were used to send messages to, just for the sake of maintaining the relationship? Would this be possible?

In this case it would be a better choice to see the Addressee as an attribute of the MESSAGE. This attribute *may* contain a value that is also known as a USER. In other words, entity USER contains only suggestions for addressees.

Another possibility is to do both—model an optional relationship and an optional attribute that cooperatively handle the addressee. An extra constraint (which cannot be shown in the diagram) must then make sure that *at least one of the attribute and the relationship* is actually given a value for a MESSAGE.

## m:m Relationships May Hide Something



ORACLE

3-27

Copyright © Oracle Corporation, 2002. All rights reserved.

### m:m Relationships May Hide Something

During the process of modeling you will find many relationships to be of type m:m. Often this is a temporary thing. After you have been able to add more details to the model, a lot of the m:m relationships will disappear as, after consideration, they simply do not model the business properly.

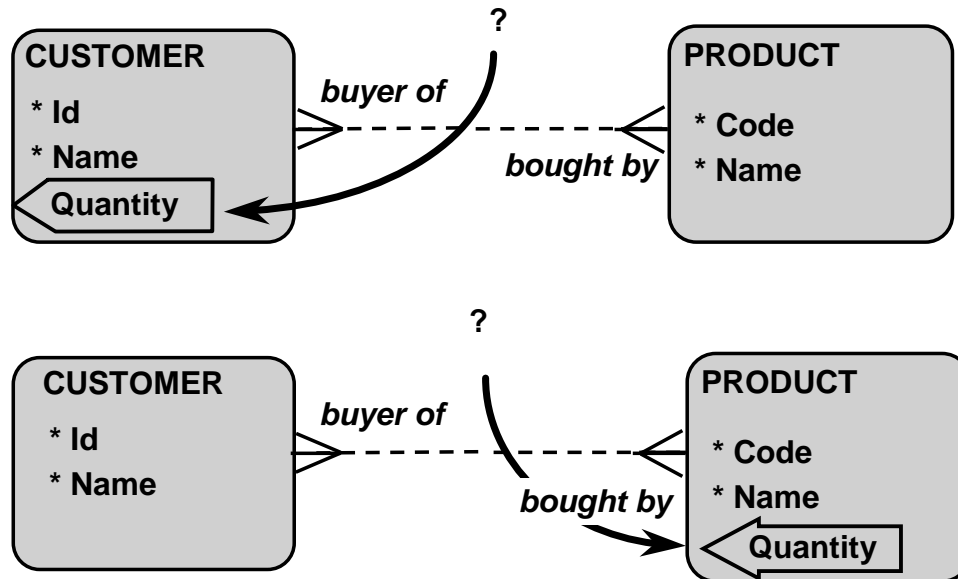
A typical example is about the CUSTOMER/PRODUCT relationship.

Suppose you make a model for a retail company that sells PRODUCTS. A CUSTOMER *buys* PRODUCTS. Suppose future customers are accepted into the system as well. This would mean:

- A CUSTOMER may *buy* one or more PRODUCTS
- A PRODUCT may be *bought by* one or more CUSTOMERS

A typical event for this company would be customer Nick Sanchez buying two shirts. “Nick Sanchez” is a CUSTOMER Name, “shirt” is a PRODUCT Name. This leaves the question of where to put the “two”, the quantity information.

## Quantity Is Attribute of ...



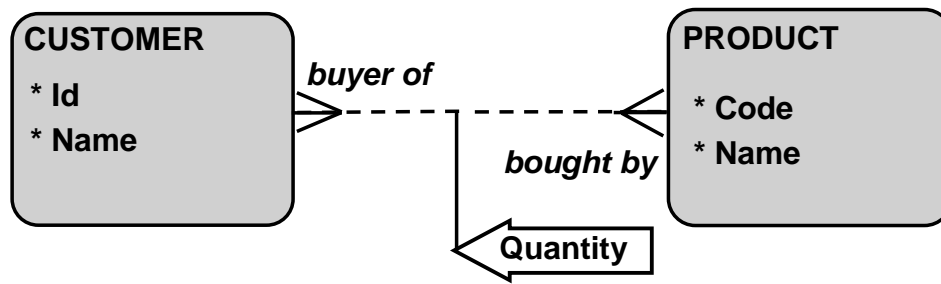
ORACLE

### Quantity

It is clear that Quantity is neither a property of CUSTOMER nor of PRODUCT. Quantity seems to be an attribute of the relationship between CUSTOMER and PRODUCT.



## Attribute of Relationship ?



ORACLE

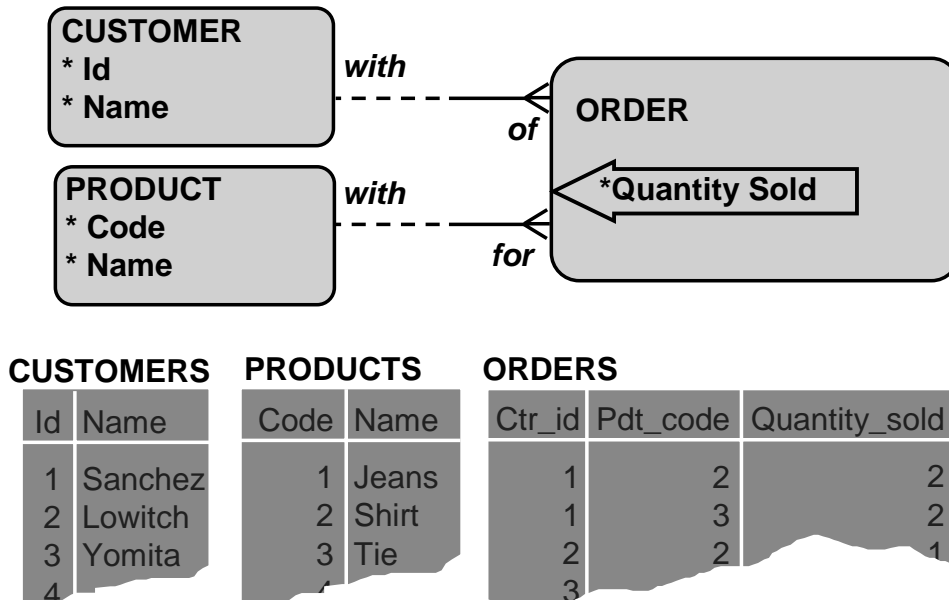
3-29

Copyright © Oracle Corporation, 2002. All rights reserved.

### Attribute of Relationship ?

Relationships do not and cannot have attributes. Apparently an entity of which quantity is a property, is missing. For that reason we need to change the model. Entity ORDER (or SALE or PURCHASE) enters the scene.

## New Entity ORDER



### New Entity Order

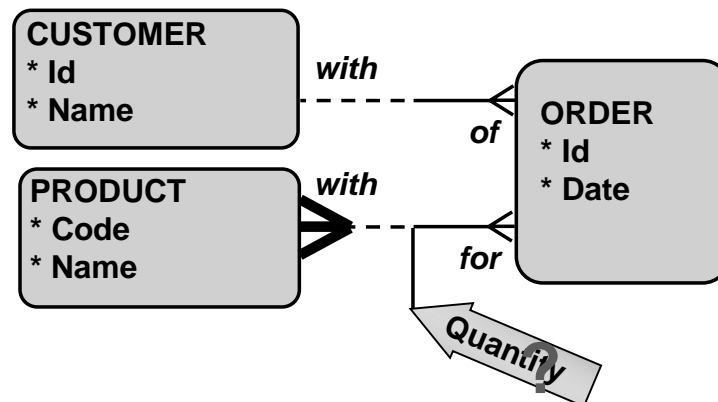
The table design here is the default design for implementing the model. Note the two foreign key columns in the ORDERS table, Ctr\_id (foreign key to CUSTOMERS) and Pdt\_code (to PRODUCTS).

Now suppose Pepe Yomita enters the store and buys one pair of jeans, two shirts, and one silk tie. Given the current model this would mean that Pepe places three orders: one for the jeans, one for the shirts and one for the tie. Three orders, all at the same time, from one and the same customer. No problem so far as the model allows for this.

Now suppose the store wants to automate the billing of the orders. (This is probably one of the reasons for making the model anyway.) Using the above model, this would mean three orders and, as a consequence, three bills, as the system has no way of knowing these three orders somehow belong to each other.

It is better to change the model in such a way that one order can be for more than one product. That means we should have a m:m relationship between ORDER and PRODUCT, which we should investigate next.

## Multiple PRODUCTS for an ORDER

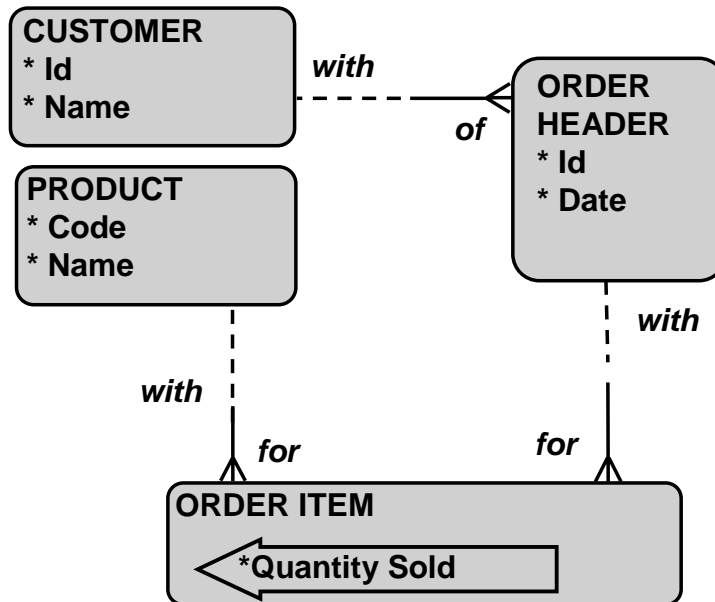


ORACLE

### Multiple Products for an Order

Then there is the question again: where do you put quantity? Quantity can now no longer be an attribute of an order because the attribute must be single-valued and cannot contain three values 1, 2 and 1 at the same time. Quantity has become a property of the m:m relationship between PRODUCT and ORDER.

## Another New Entity: ORDER ITEM



ORACLE

### Another New Entity: ORDER ITEM

Note the name change from ORDER to ORDER\_HEADER, to avoid the 1: m relationship that is mandatory at both ends. The set of tables for this model could be:

# Tables

## CUSTOMERS

Id	Name
1	Sanchez
2	Lowitch Yomita

## ORDER\_HEADERS

Id	Ctr_id	Date_ordered
1	1	25-MAY-1999
2	2	25-MAY-1999
2	2	25-MAY-1999

## ORDER\_ITEMS

Ohd_id	Pdt_code	Quantity_sold
1	2	2
		2
2	2	1

## PRODUCTS

Code	Name
1	Jeans
2	Shirt
3	Tie

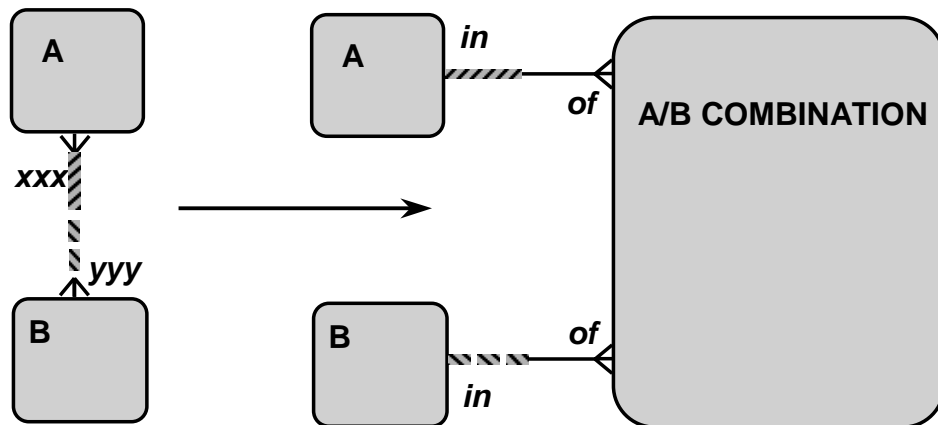
ORACLE

## Tables

Note the name change from ORDER to ORDER\_HEADER, to avoid the 1: m relationship that is mandatory at both ends. The set of tables for this model could be:

## Resolving m:m Relationship

- **Create new intersection entity**
- **Create two m:1 relationships, derive optionality**
- **Remove m:m relationship**



ORACLE

3-34

Copyright © Oracle Corporation, 2002. All rights reserved.

### Resolving Relationships

#### Relationships and Intersection Entities

Earlier in this lesson you saw a typical example of relationships seeming to have attributes. The relationships in the example were many-to-many relationships. You deal with the situation by creating a new entity, an intersection entity, that replaces the relationship and can hold attributes.

This leads to the following questions:

- What are the steps in resolving a relationship in general?
- Should every m:m relationship be resolved?
- Can other relationships than m:m be resolved?

#### Resolving a Relationship

Suppose we want to resolve the m:m relationship between entities A and B.

1. First create a new intersection entity. You will experience that sometimes there is no suitable word available for the concept you are modeling. The new entity can always be named with the neologism “A/B COMBINATION”, or a name that is somehow derived from the name of the original m:m relationship. Do not let the unavailability of a proper name for the entity stop you from modeling it.

2. Next create two new m:1 relationships from entity A/B COMBINATION, one to A and one to B. Initially, draw these as mandatory at A/B COMBINATION, as you will probably only be interested in complete pairs of A and B. If the original m:m relationship was optional (or mandatory) at A's side, then the new relationship from A to A/B COMBINATION is also optional (or mandatory).
3. Name the relationships. You can often name both relationships "in / of".
4. The next step is to remove the m:m relationship you started with.
5. Finally, reconsider the newly-drawn relationships. They may be optional at the A/ B COMBINATION side. Also, they may turn out to be of type m:m and require resolving, as you have seen in the example of customers buying products.

### **Should Every m:m Relationship be Resolved?**

The answer depends on a number of factors.

Given the usual scenario, when you start creating an ER model you will discover that many of the relationships you draw are of type m:m. Most of these will appear to hide entities that you need in a later stage as you need to have a place in which to put specific attributes. Finally, you will have only a few "genuine m:m" relationships left.

#### **No**

Purely from a conceptual data modeling point of view, there is no need to resolve these genuine m:m relationships. The model is rich enough to be the basis for table design. A m:m relationship will transform into a binary table; this is a table that consists of the columns of two foreign keys only. This is exactly the same table as the one that would result from the intersection entity when you resolved the m:m relationship.

A m:m relationship in a conceptual data diagram needs less space than a separate entity plus two relationships. For this reason a diagram with unresolved m:m relationships is more transparent and easier to read.

#### **Yes**

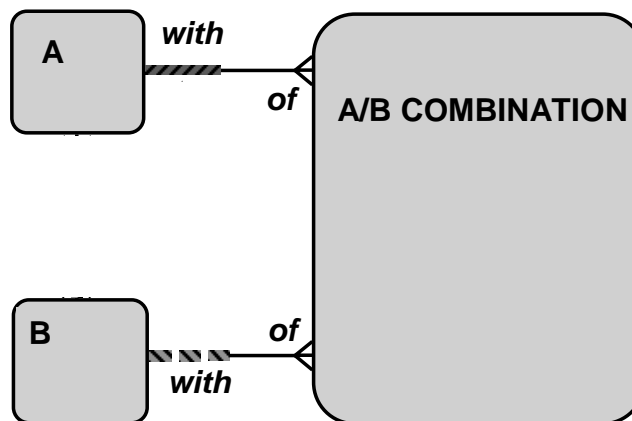
From a function modeling point of view the answer is different. If your model contains a true m:m relationship there is apparently a business need to keep information on the combinations of, say, entity A and B. In other words, the system would contain at least one business function that creates the relationship. This "create relationship" cannot be expressed as a usage of entities or attributes, although this is usually what design tools require of the functional model. Oracle Designer is no exception. This means that when you create an ER model in Oracle Designer you would always resolve the m:m relationships in order to create a fully-defined functional model with all data usages included.

### **Resolving Other Relationships**

Can relationships other than m:m be resolved? Yes. Every relationship, even a 1:1, can be resolved into an intersection entity and two relationships, just like a m:m relationship. When would you want to do this? It is quite rare to find a situation where you have to do this. A typical situation where you may like to resolve a non m:m relationship is when one entity represents something that is outside your system, for example, when the entity is part of a third-party package.

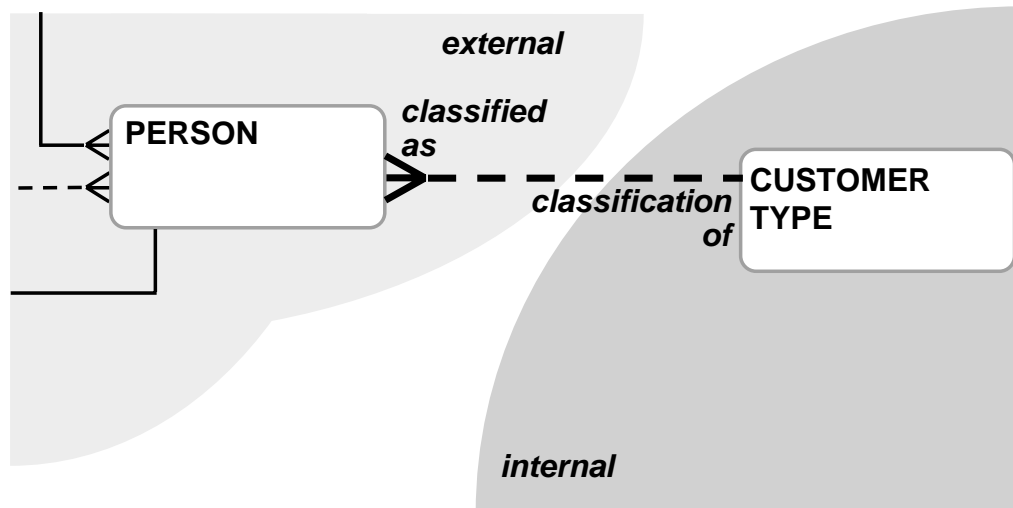
## Resolving m:m Relationship

- Create new intersection entity
- Create two m:1 relationships, derive optionality
- Remove m:m relationship





## Resolving m:1 Relationship



ORACLE

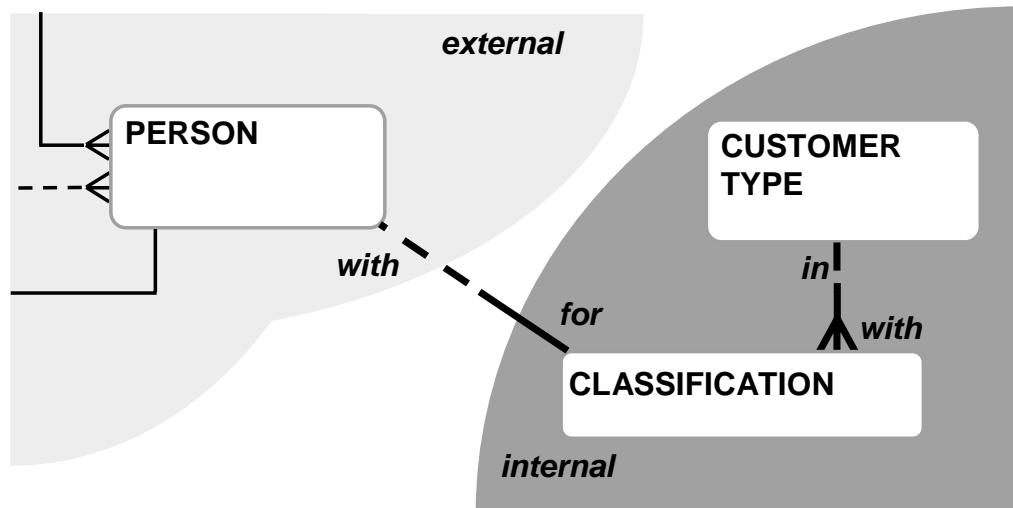
3-37

Copyright © Oracle Corporation, 2002. All rights reserved.

### Resolving m:1 Relationship

Suppose you need your system to create a m:1 relationship from external entity **PERSON** to **CUSTOMER TYPE**, one of your internal entities.

## Resolving m:1 Relationship



ORACLE

3-38

Copyright © Oracle Corporation, 2002. All rights reserved.

### Resolving m:1 Relationship

This would result later on in a change of the table structure of the third-party PERSONS table. This is undesirable (third parties often ask you to you sign a contract that simply forbids you to do that) and sometimes even impossible if you have no authority over that table.

The above model leaves the external entity PERSON as is and does the referencing from inside. The m:1 relationship is replaced by an entity CLASSIFICATION and two relationships.

# Normalization Rules

Normal Form Rule	Description
First Normal Form	All attributes are single valued.
Second Normal Form (2NF)	An attribute must be dependent upon entity's entire unique identifier.
Third Normal Form (3NF)	No non-UID attribute can be dependent on another non-UID attribute.

**“A normalized entity-relationship data model automatically translates into a normalized relational database design”**

**“Third normal form is the generally accepted goal for a database design that eliminated redundancy”**

ORACLE

3-39

Copyright © Oracle Corporation, 2002. All rights reserved.

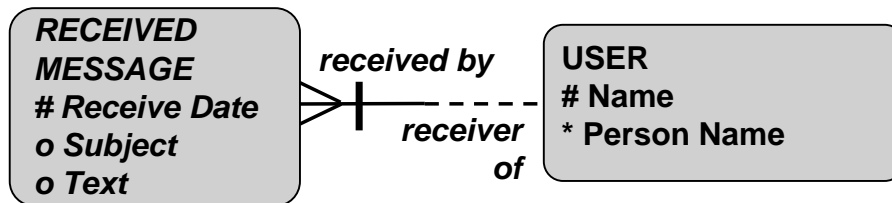
## Normalization During Data Modeling

Normalization is a relational database concept. However, if you have created a correct entity model, then the tables created during design will conform to the rules of normalization. Each formal normalization rule from relational database design has a corresponding data model interpretation. The interpretations which can be used to validate the placement of attributes in an ER Model are as follows.

## First Normal Form in Data Modeling

**USER**  
# Name  
\* Person Name  
\* Message Receive Date  
o Message Subject  
o MessageText

All attributes must be single-valued.



ORACLE

### First Normal Form in Data Modeling

All attributes must be single-valued.

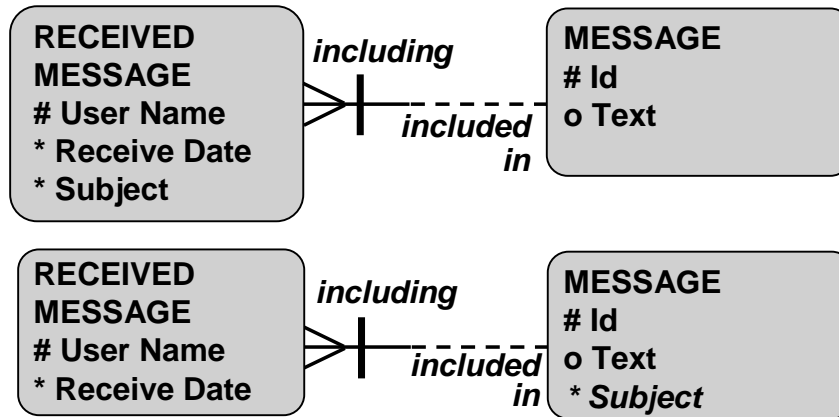
Validate that each attribute has a single value for each occurrence of the entity. No attribute should have repeating values.

You can often recognize the misplaced attributes by the fact that there is the same (entity) name in the attribute name, such as **Message Subject** and **Message Text**.

If the attribute has multiple values, create an additional entity and relate it to the original entity with a m:1 relationship.

## Second Normal Form in Data Modeling

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



ORACLE

### Second Normal Form in Data Modeling

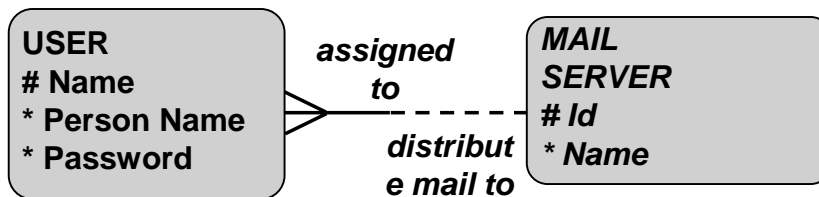
An attribute must be dependent upon its entity's entire unique identifier.

Validate that each attribute is dependent upon its entity's entire unique identifier. Each specific instance of the UID must determine a single instance of each attribute. Validate that an attribute does not depend upon only part of its entity's UID. If it does, then it is misplaced and you must move it.

## Third Normal Form in Data Modeling

**USER**  
# Name  
\* Person Name  
\* Password  
\* Server Id  
\* Server Name

No non-UID attribute can be dependent upon another non-UID attribute.



ORACLE

### Third Normal Form in Data Modeling

No non-UID attribute can be dependent upon another non-UID attribute. If an attribute is dependent upon a non-UID attribute, then move both the dependent attribute and the attribute it is dependent upon to a new, related entity.

## Summary

- **Relationships express how entities are connected.**
- **Initially relationships often seem to be of type m:m.**
- **Finally relationships are most often of type m:1.**
- **Relationships can be resolved into:**
  - **Two new relationships**
  - **One intersection entity**
- **Third Normal form is generally accepted standard.**

ORACLE

3-43

Copyright © Oracle Corporation, 2002. All rights reserved.

### Summary

Relationships connect entities and express how they are connected. There are ten types of relationships, 4 of type 1:m, 3 of type m:m and 3 of type 1:1.

The m:1 relationship that is optional at the 1 side is by far the most common type in finished ER models. This one is very easy to implement in a relational database.

At the beginning of the process of creating an ER model there are often many m:m relationships. Many of these disappear after closer investigation.

Relationships cannot have attributes. If this seems to be the case, you need to resolve the relationship into an intersection entity plus two relationships.

The other types are less common—some express more a desired situation rather than reality, such as the m:1 relationship that is mandatory at both ends.

A normalized data model yields a normalized relational database design. Third normal form is the generally accepted standard.

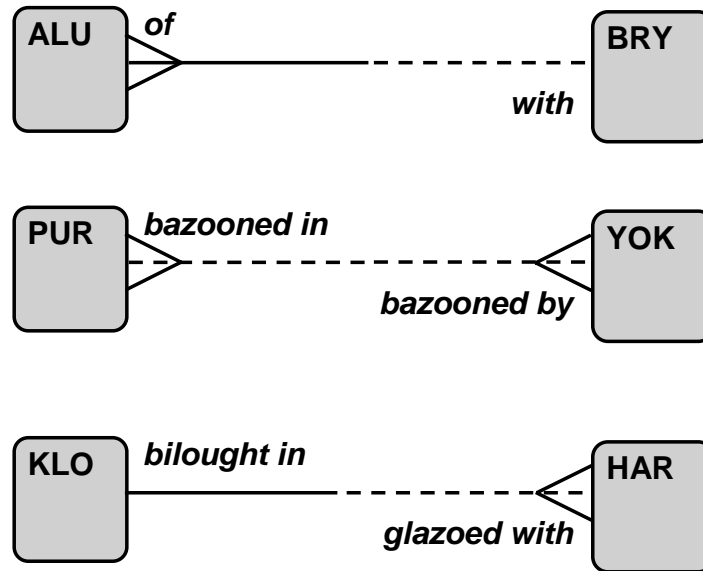
# Practices

- **Read the Relationship**
- **Find a Context**
- **Name the Intersection Entity**
- **Receipt**
- **Moonlight P&O**
- **Price List**
- **EMail**
- **Holiday**

ORACLE®



## Practice: Read the Relationship



ORACLE

3-45

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 3-1: Read the Relationship

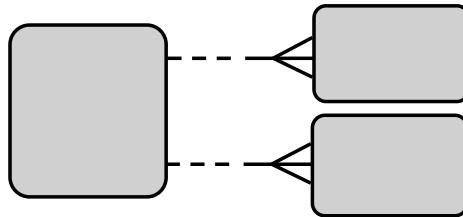
#### Goal

The goal of this practice is to learn to read relationships from an ER diagram.

#### Your Assignment

Read the diagrams aloud, from both perspectives. Make sentences that can be understood and verified by people who know the business area, but do not know how to read ER models.

## Find a Context (1)



ORACLE

3-46

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 3-2: Find a Context

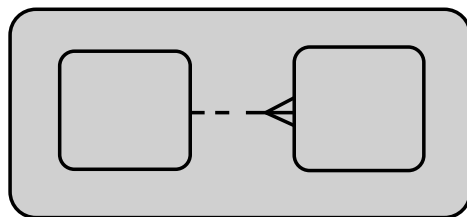
#### Goal

The purpose of this practice is to use your modeling skills.

#### Your Assignment

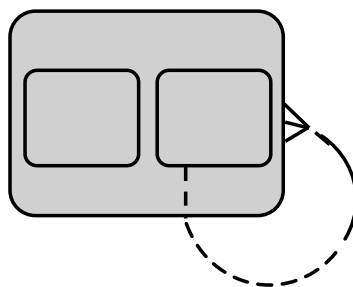
Given the following ER diagrams, find a context that fits the model.

## Find a Context (2)

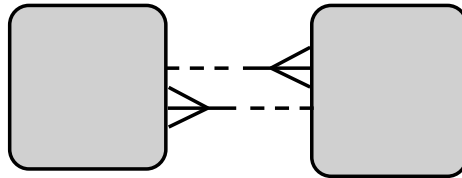


ORACLE

## Find a Context (3)

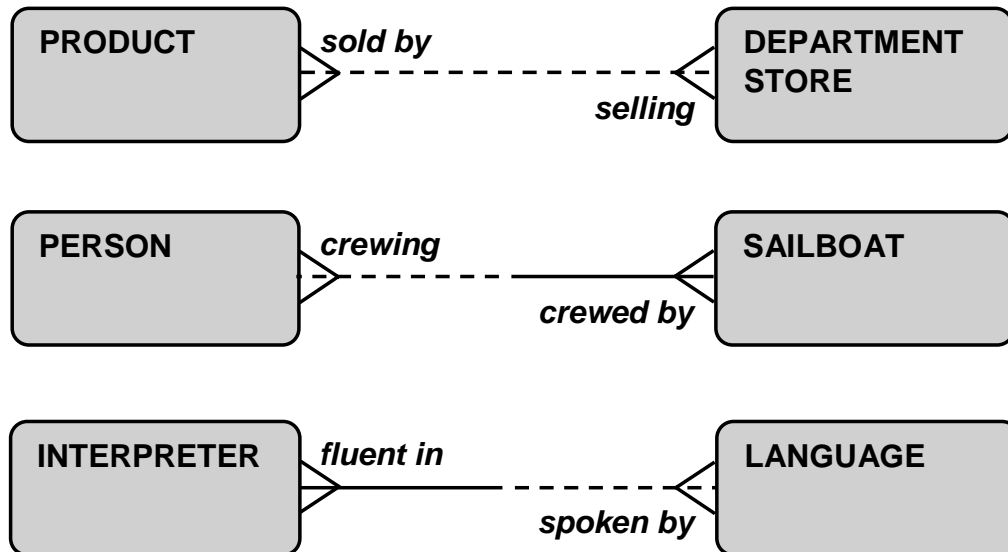


## Find a Context (4)



ORACLE

## Practice: Name the Intersection Entity



ORACLE

3-50

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 3-3: Name the Intersection Entity

#### Goal

The goal of this practice is to find a proper name for the intersection entity after resolving the m:m relationship.

#### Your Assignment

1. Resolve the following m:m relationships. Find an acceptable name for the intersection entity.
2. Invent at least one attribute per intersection entity that could make sense in some serious business context. Give it a clear name.

## Practice: Receipt

Served by: Dennis

Till: 3 Dec 8, 4:35 pm

```
-----
CAPPUCC M   3.60
              * 2   7.20
CREAM        .75
              * 2   1.50
APPLE PIE            3.50
BLACKB MUF         4.50
      <SUB>          16.70
      tax 12%        2.00
      <TOTAL>       18.70
                      =====
      CASH           20.00
      RETURN         1.30
-----
```

Hope to serve you again  
@MOONLIGHT COFFEES  
25 Phillis Rd, Atlanta

ORACLE

### Practice 3-4: Receipt

#### Goal

The purpose of this practice is to use a simple source of real life data as a basis for a conceptual data model.

#### Scenario

You work as a contractor for Moonlight Coffees. Your task is to create a conceptual data model for their business. You have collected all kinds of documents about Moonlight. Below you see an example of a receipt given at one of the shops.

#### Your Assignment

Use the information from the receipt and make a list of entities and attributes.

## Practice: Moonlight P&O

- All Moonlight Coffee employees work for a department such as “Global Pricing” or “HQ”, or for a shop. All employees are at the payroll of one of our country organizations. Jill, for example, works as a shop manager in London; Werner is a financial administrator working for Accounting and is located in Germany.
- All shops belong to one country organization (“the countries”). There is only one country organization per country. All countries and departments report to HQ, except HQ itself.
- Employees can work part time. Lynn has had an 80% assignment for Product Development since the 1st September. Before that she had a full-time position.

ORACLE

3-52

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 3-5: Moonlight P&O

#### Goal

The purpose of this practice is to create a ER model iteratively, based on new pieces of information and new requirements.

#### Scenario

You are still working as a contractor for Moonlight Coffees—apparently you are doing very well!



## Practice 3-5: Moonlight P&O

### Your Assignment

1. Create a entity relationship model based on the following personnel and organization information:
2. Extend or modify the diagram based on this information:
3. And again:
4. Change the model—if necessary and if possible—to allow for the following new information.
  - Jan takes shifts in two different shops in Prague.
  - Last year Tess resigned in Brazil as a shop manager and moved to Toronto. Recently she joined the shop at Toronto Airport.
  - To reduce the number of direct reports, departments and country organizations may also report to another department instead of Headquarters.
  - The shops in Luxembourg report to Belgium.
  - To prevent conflicting responsibilities, employees are not allowed to work for a department and for a shop at the same time.
5. Would your model be able to answer the next questions?
  - Who is currently working for Operations?
  - Who is currently working for Moonlight La Lune at the Mont Martre, France?
  - Are there currently any employees working for Marketing in France?
  - What is the largest country in terms of number of employees? In terms of managers? In terms of part-timers?
  - When can we celebrate Lynn's fifth year with the company? When can we do the same with Tess' fifth year with Moonlight?
  - What country has the lowest number of resignations?

## price list

## Practice: Price List

25 Phillis Road, Atlanta

visit us at [www.moonlight.com](http://www.moonlight.com)

	small	medium	large	
regular coffee	2.25	2.90	3.50	
cappuccino	2.90	3.60	4.20	
café latte	2.60	3.20	3.90	
special coffee	3.10	3.70	4.40	
espresso	2.25	2.90	3.50	
coffee of the day	2.00	2.50	3.00	
<u>decaffeinated</u>	<u>.25</u>	<u>.50</u>	<u>.75</u>	<i>extra</i>
black tea	2.25	2.90	3.50	
infusions	2.60	3.20	3.90	
herbal teas	2.90	3.60	4.20	
tea of the day	2.00	2.50	3.00	
<u>decaffeinated</u>	<u>.25</u>	<u>.50</u>	<u>.75</u>	<i>extra</i>
milk	1.25	1.90	2.50	
soft drinks	2.25	2.90	3.50	
soda water	2.25	2.90	3.50	
mineral water	2.90	3.60	4.20	
apple pie				3.50
strawberry cheesecake			3.50	
whole wheat oats muffin with almonds				3.90
blackberry muffin				4.50
fruitcake				4.50
cake of the day				4.00
additional whipped cream				.75

Sales Tax included  
September 16

ORACLE

### Practice 3-6: Price List

#### Goal

The purpose of this practice is to use a simple source of real life data as a basis for a conceptual data model.

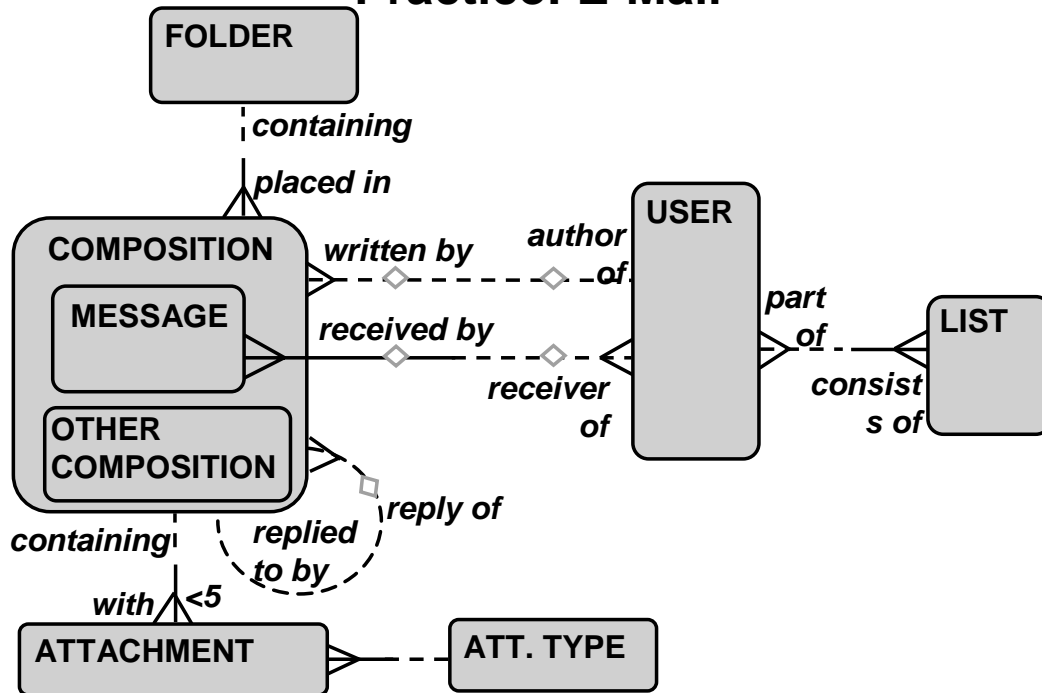
#### Scenario

You work as a contractor for Moonlight Coffees.

#### Your Assignment

Make a ER model based on the pricelist from one of the Moonlight Coffee Stores.

## Practice: E-Mail



ORACLE

3-55

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 3-7: E-mail

#### Goal

The goal of this practice is to extend an existing conceptual data model.

#### Scenario

#### Your Assignment

Take the given model as starting point. Add, delete, or change any entities, attributes, and relationships so that you can facilitate the following functionality:

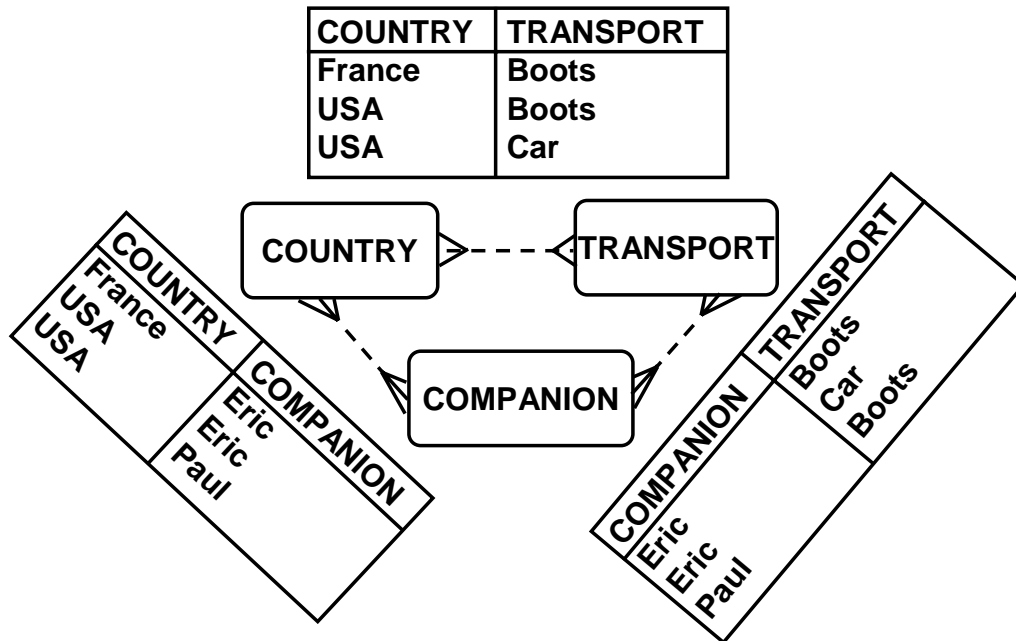
1. A user must be able to create nick names (aliases) for other users.
2. A folder may contain other folders.
3. A user must be able to forward a composition. A forward is a new message that is automatically sent together with the forwarded message.
4. All folders and lists are owned by a user.

#### Challenge:

5. A mail list may contain both users and other lists.
6. A mail list may contain external addresses, like "giovanni\_papini@yahoo.com".
7. A nickname may be an alias for an external address.

## Practice: Holiday

“Paul and I hiked in the USA. Eric and I hiked in France and we rented a car in the USA last year.”



ORACLE

3-56

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 3-8: Holiday

#### Goal (See Page 54)

The purpose of this practice is to do a quality check on a conceptual data model.

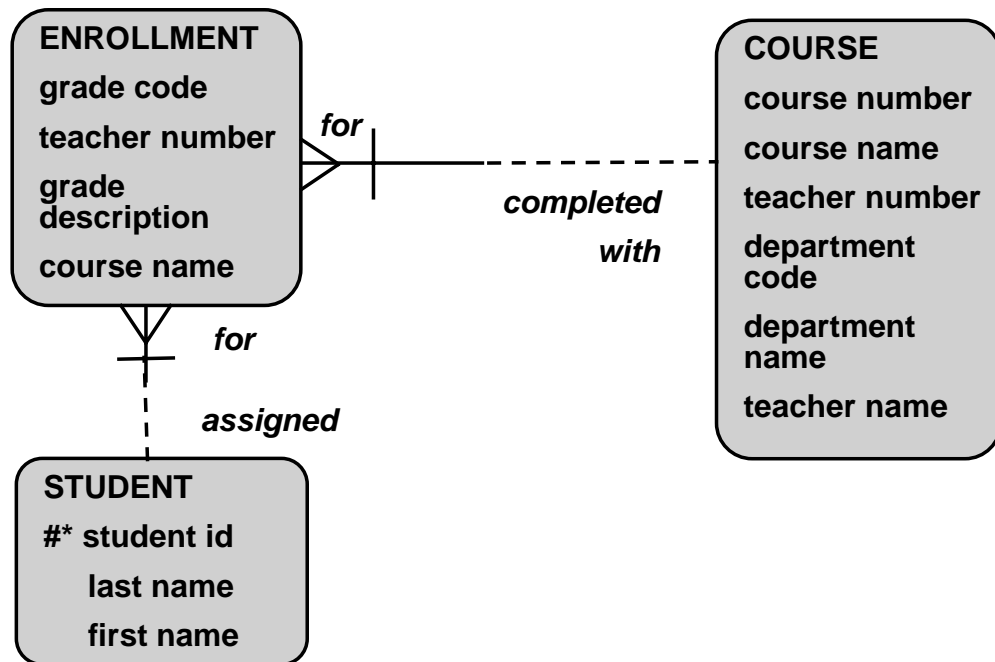
#### Scenario

“Paul and I hiked in the USA. Eric and I hiked in France and we rented a car in the USA last year”.

#### Your Assignment

Comment on the model given below that was based on the scenario text.

## Practice: Normalize an ER Model



ORACLE

3-57

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 3-9: Normalize an ER Model

#### Goal

The purpose of this practice is to place an unnormalized ER Model into third Normal Form.

#### Your Assignment

1. For the following ER Model, evaluate each entity against the rules of normalization, identify the misplaced attributes and explain what rule of normalization each misplaced attribute violates.
2. Optionally, redraw the ER diagram in third normal form.



# 4

## Constraints

ORACLE®

Copyright © Oracle Corporation, 2002. All rights reserved.

# Overview

- **Unique Identifiers**
- **Arcs**
- **Domains**
- **Various other constraints**

ORACLE®

4-2

Copyright © Oracle Corporation, 2002. All rights reserved.

## Introduction

This lesson is about constraints that apply to a business. Constraints are also known as business rules. Some of these constraints can be easily modeled. Some can be diagrammed but the resulting decreased clarity may not be acceptable. Some constraints cannot be modeled at all. These should be listed in a separate document.

## Objectives

At the end of this lesson, you should be able to do the following:

- Describe the problem of identification in the real world
- Add unique identifiers to your model and know how they are represented
- Recognize correct and incorrect unique identifiers
- Decide when an arc is needed in your model
- Describe the similarities between arcs and subtypes
- Describe various types of business constraints that cannot be represented in an ER diagram



# Rembrandt



## Identification

### What Are We Talking About?

It is not unreasonable to assume everybody knows Rembrandt was born in the Netherlands. What most people probably do not know is that Rembrandt was born on a farm as the son of Pajamas and an unknown father. Rembrandt had a twin sister. Although Rembrandt never married, he was the father of numerous children. You can easily recognize Rembrandt and his offspring as they all have four white stripes at the end of their tails.

Identification is about knowing what or who you are talking about. Obviously, the name Rembrandt is not unique to the famous painter; other human beings and even cats have the same name.

In day-to-day conversations, you can usually assume that you and the people you talk to share enough of the same context and know enough about each other's jobs and interests, to understand what you are both talking about. Language is always a rather nonspecific way to communicate, with lots of ambiguities, but people are very capable of interpretation.

## **Identification**

Computers must communicate in a more specific way that is not open to much interpretation. It would help a system to be told “Rembrandt the painter” or “Rembrandt van Rijn, born in 1606” or maybe even the combination of all: “Rembrandt van Rijn, the painter, born in 1606”, to distinguish this Rembrandt from the other famous creatures with the same name.

### **The Problem of Identification**

There are three sides to the problem of identification. One is identification in the real world—how do we distinguish two real world things that have very similar properties? This is the most difficult side. The second is identification within a database system— how do we distinguish rows in tables? This one is far less complex. A third issue deals with representation: how do we know what real world thing a row in a table represents?

### **Identification in the Real World**

Many things in the real world are difficult, if not impossible, to identify—distinguishing between two cabs, two customers, two versions of a contract, or two performances of the fourth string quartet by Shostakovich. As a general rule, real world things cannot be identified with certainty. You have to live with a substantial level of ambiguity. For example, how can I be sure that the car at the other side of the street with license plate MN4606 is the same car as the one I saw last week with that number? I cannot even be sure it is the same license plate. In normal circumstances there is no reason for doubt, but that is not the same as certainty. Sometimes people have their reasons for creating confusion.

Fortunately, some things in the real world are easier as they are within your reach. There you can define the rules. When a company sends out, for example, invoices, it can give every single invoice a unique number. When a business lets people create ElectronicMail usernames (identities), they can force these names to be unique.

### **Identification Within a Database**

Usually, database systems can make sure that a row is not stored twice, or, to be more exact, that a particular combination of values is not stored twice, within the same table. The technical problem is solved for you by the standard software you use.

### **Representation**

The remaining problem is to make sure that you can always know what real world thing is represented by a particular row in a table. The solution to this problem depends highly on the context. How likely do you consider it to be that two different employees for the same company have the same family name, or the same family name plus initials, or the same family name plus initials plus birthdate?

# Identification and Representation

**EMPLOYEES**

Name	Initials	Birthdate
PAPINI	G.	02-FEB-1954
HIDE	T.M.	11-JUN-1961
PAPINI	G.	02-FEB-1945
BAKER	S.J.T.	24-SEP-1958

G. Papini, please?



## Identification and Representation

Clearly, the answer could be different when your company employs five or 50,000 employees.

Be aware that adding a new identifying attribute for EMPLOYEE, say, Id, only partially solves the above problem. It would be very useful within the database. It would not help much in the real world where employees usually would not know their IDs, let alone the IDs of others. This kind of Id attribute often works only as an internal, but not as an external identification.

## Unique Identifier Examples

	<b>JOB</b>	<b>Name</b>
<b>COMPUTER IN NETWORK</b>		<b>IP Address</b>
<b>TELEPHONE</b>		<b>Country code, Area code, Telephone number</b>
<b>EMPLOYEE</b>		<b>Employee number <i>or</i> Name, Initials, Birth Date</b>
<b>MAIL LIST</b>		<b>Name, Owner</b>

ORACLE

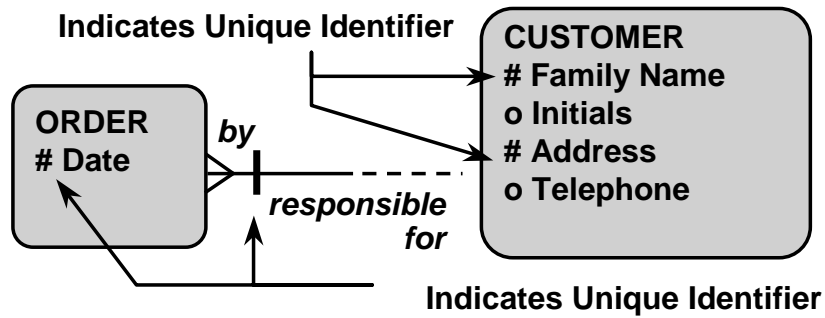
4-6

Copyright © Oracle Corporation, 2002. All rights reserved.

### Unique Identifier

To know what you are talking about, you need to find, for every entity, a value, or a combination of values, that uniquely identifies the entity instance. This value or combination is called the Unique Identifier for the entity.

## Unique Identifier



ORACLE

### Unique Identifier

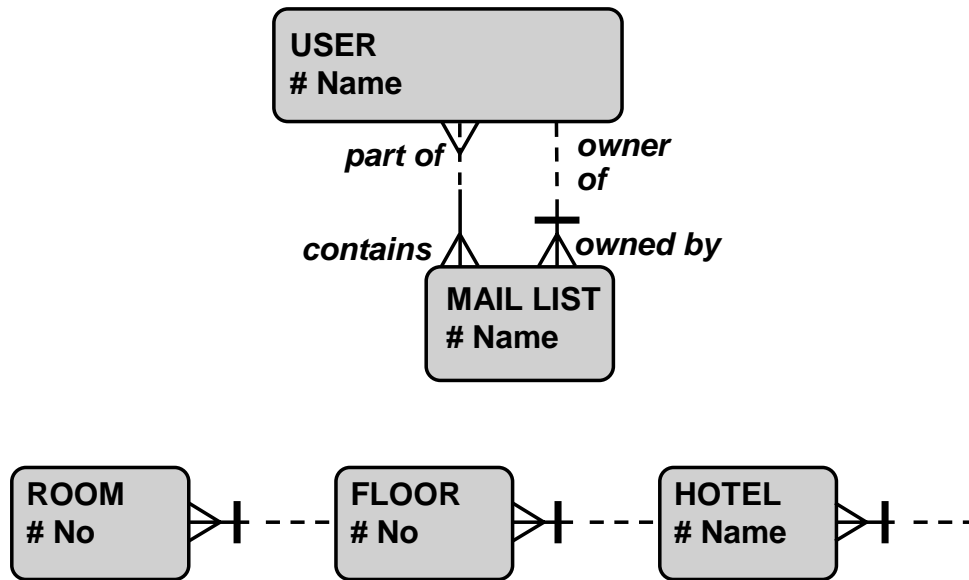
The MAIL LIST example shows that a unique identifier is not necessarily a combination of attributes: the *owner* of a MAIL LIST is actually represented by a *relationship*.

### UID Representation

In an ER diagram, the components of the UID of an entity are marked:

- # for attributes.
- With a small bar across the relationship end for relationships (a barred relationship).

## Unique Identifiers



ORACLE

### Unique Identifier

#### Single Attribute UID

The model shows that a **USER** of **ElectronicMail** is identified by attribute **Name** only. Many entities will be identified by a single attribute. Typical candidate attributes, if available, for single attribute UIDs are: **Id**, **Code**, **Name**, **Description**, **Reference**.

#### Multiple Attribute UID

An entity may have a UID that consists of multiple attributes; for example, a software package can usually be identified by its **Name** and its **Version**, such as **Oracle Designer**, version *9i*.

## **Unique Identifier**

### **Composed UID**

A MAIL LIST, illustrated above, is identified by the Name of the LIST plus the USER that owns the LIST. That means that the combination of OWNER and a Name of a list must be a unique pair.

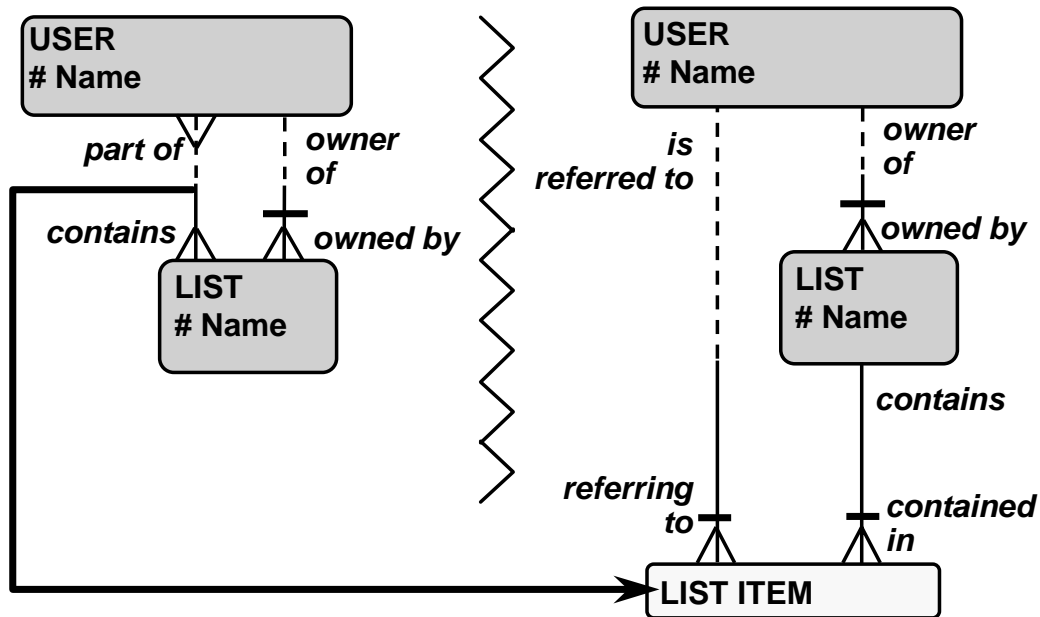
This means that every USER must name their LIST instances uniquely, but need not worry about names given by other users. It also means that the system may have many LIST instances with the same name, as long as they are owned by different USERS.

You may argue that a USER also has a composed UID, as the Name must be unique, within this mail system. To show this, you could add an extra high level entity, MAIL PROVIDER, plus a relationship from USER to PROVIDER. The relationship then is part of the UID of a USER.

### **Cascade Composed UID**

It is not uncommon that an entity has a barred relationship to another entity that has a barred relationship to a third entity, and so on.

## Multiple Relationship UID



ORACLE

4-10

Copyright © Oracle Corporation, 2002. All rights reserved.

### UID: Relationships Only

A Unique Identifier can also consist of relationships only.

At the lower right side of the diagram, entity **LIST ITEM** is shown, which resulted from the resolved m:m relationship between **LIST** and **USER**.

The model shows that a **LIST ITEM** is identified by the combination of the **USER** and the **LIST**. In other words, the model says that a **LIST** may contain as many **ITEMS** as you like, as long as they refer to different **USERS**.

This results in the next definition:

A Unique Identifier (UID) of an entity is a constraint that declares the uniqueness of values; a UID is composed of one or more attributes, one or more relationships, or a combination of attributes and relationships of the entity.

Consequently, not all components of the UID may be optional.

### Indirect Identification

Identification regularly takes place using an indirect construction, that is, when the instance of an entity is identified only by the instance of another entity it refers to.



## UID: Relationships Only

### Examples

- In many office buildings employees are identified by their badge, which is identified by a code.
- Around the world a person is identified by the picture on their passport.
- All cows in the European Community are identified by the number of the tag they are supposed to wear in their ear.
- When you park a car at Amsterdam International Airport you enter the parking lot by inserting a credit card into a slot at the gate. The parking event is identified by the credit card of the person that parked the car. This is a double indirect identification.

Clearly, these identification constructions are not 100% reliable, but are probably as far as you can go in a situation.

The model of these indirect identifications is shown in the next illustration, at the right bottom corner. An instance of S is identified by the single instance of T it refers to. In other words, the UID consists of one relationship only.

### Multiple UIDs

Entities may have multiple UIDs. Earlier, you saw the example of entity EMPLOYEE that can be identified by an Employee Number, and possibly by a combination of, for example, Name, Initials and Birth Date.

At some point in time, usually at the end of your analysis, you promote one of the UIDs to be the primary UID. All the other UIDs are called secondary UIDs.

You would usually select the UID that is most compact or easy to remember to become primary UID. The reason, of course, is that the UID leads to one or more foreign key columns in related tables. These columns should not be too sizeable. Preferably, the primary UID of an entity does not consist of optional elements.

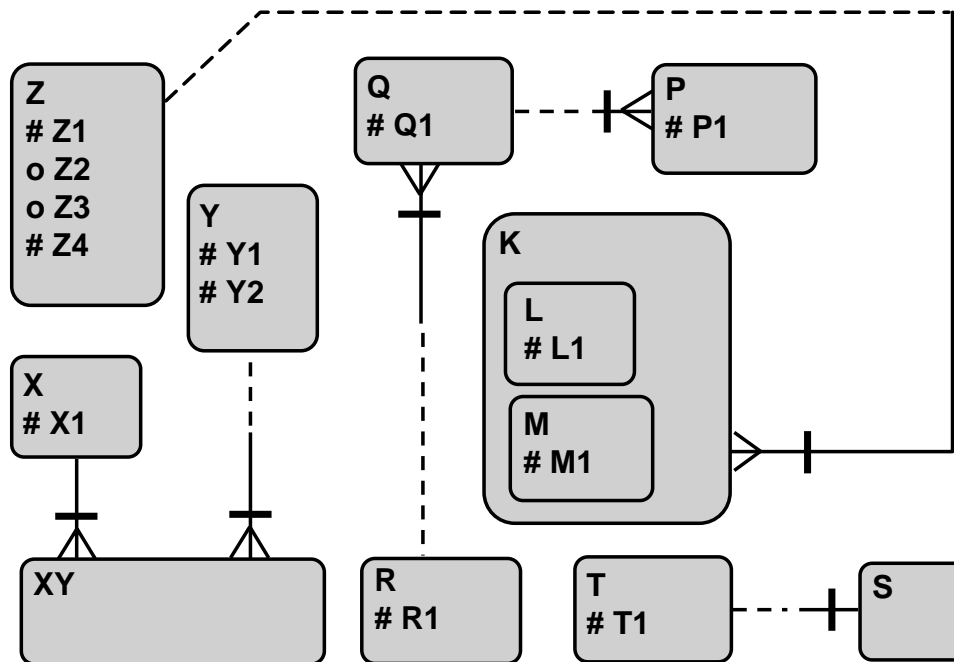
### UID in Diagram

*Only* the primary UID is shown in ER Diagrams.

### Where UIDs Lead

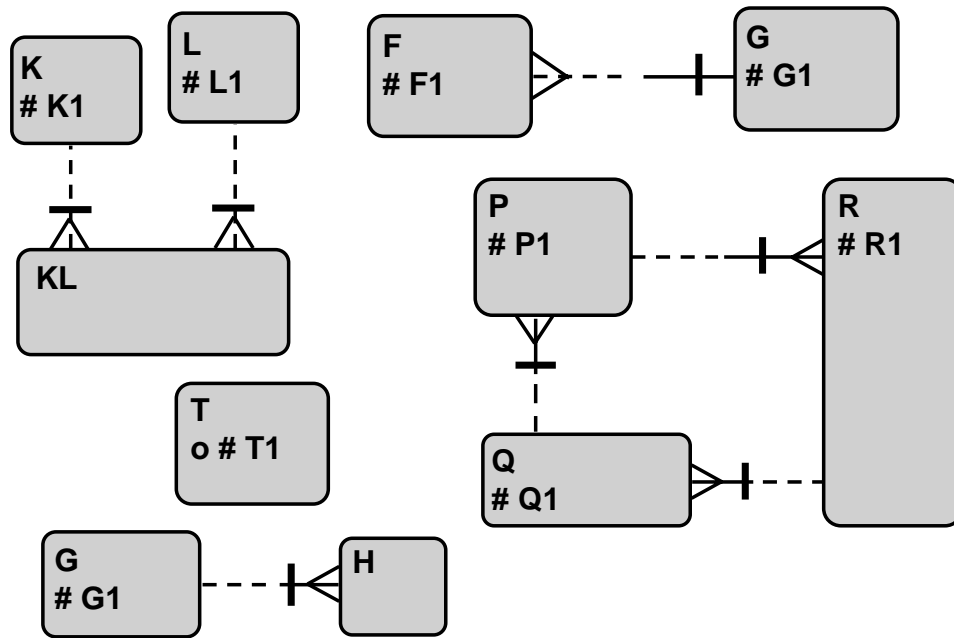
Unique Identifiers lead to Primary Key and Unique Key constraints.

## Well-defined Unique Identifiers

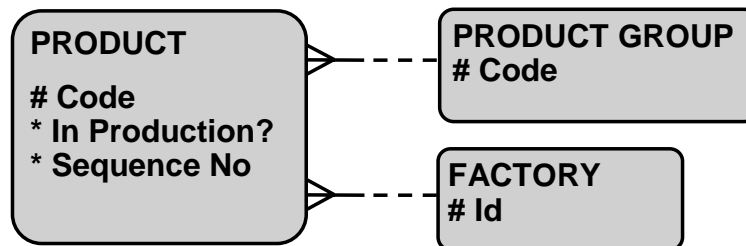
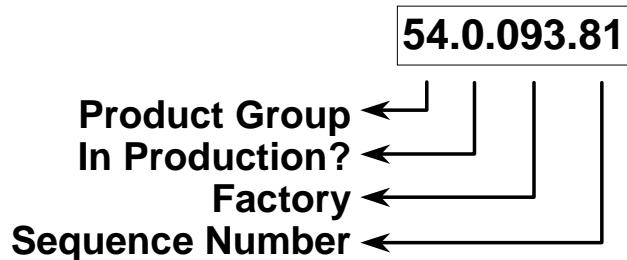


ORACLE

## Incorrect Unique Identifiers



## Information-Bearing Codes



### Information-Bearing Identifiers

When things in the real world are coded, you need to be especially careful. Codes that have been used for some time are often information bearing. An example is a company that uses product codes like 54.0.093.81, where 54 refers to the product group, 0 shows that the product is still in production, 093 identifies the factory where the product is made and 81 is a sequence number. These codes come from the time when a maximum amount of information had to be squeezed into a minimum number of bits.

The example above would be modeled conceptually:

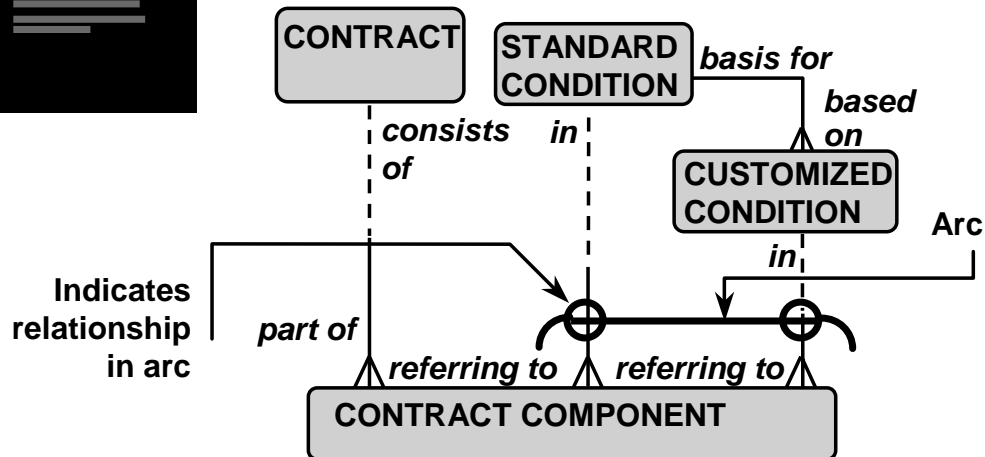
The Code attribute would contain the same codes, for reasons of compatibility, but now without meaning, as the old meaning is transferred to the attributes and relationships.

Product 54.0.093.81 may now be produced by factory 123 and may no longer be in Product Group 54.

# Arcs



**“A contract consists of contract components; these are standard conditions or customized conditions”**



ORACLE

4-15

Copyright © Oracle Corporation, 2002. All rights reserved.

## Arcs

Suppose ElectronicMail rents the Advertisement Areas that are located in their various mail screens on the Web. This renting is controlled by contracts; contracts consist of one or more standard conditions and customized conditions. This can be modeled with four entities: CONTRACT, CONTRACT COMPONENT, STANDARD CONDITION and CUSTOMIZED CONDITION. See the model below. How do we model the following constraint: every instance of CONTRACT COMPONENT refers to either a STANDARD CONDITION or a CUSTOMIZED CONDITION, but not to both at the same time?

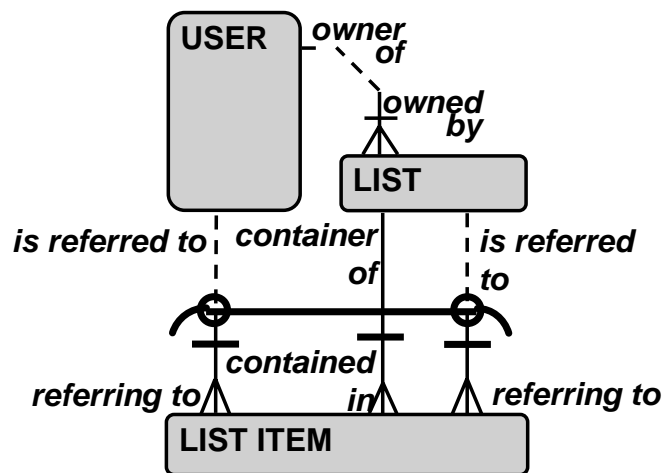
An *arc* is a constraint about two or more relationships of an entity. An arc indicates that any instance of that entity can have only one valid relationship of the relationships in the arc at any one time. An arc models an *exclusive* or across the relationships. An arc is therefor also called exclusive arc.

There is no similar constraint construct for attributes of an entity.

### Arc Representation

The arc is drawn as an arc-shaped line, around an entity. Where the arc crosses a relationship line a small circle is drawn, but only if the relationship participates in the arc.

## Exclusive Arc



ORACLE

### Mandatory Arc

Suppose a MAIL LIST may contain USERS as well as other MAIL LISTS. This means that a particular LIST ITEM may refer to a USER or a LIST. To be more precise, it must be a reference to a USER or to a LIST, but not to both at the same time.

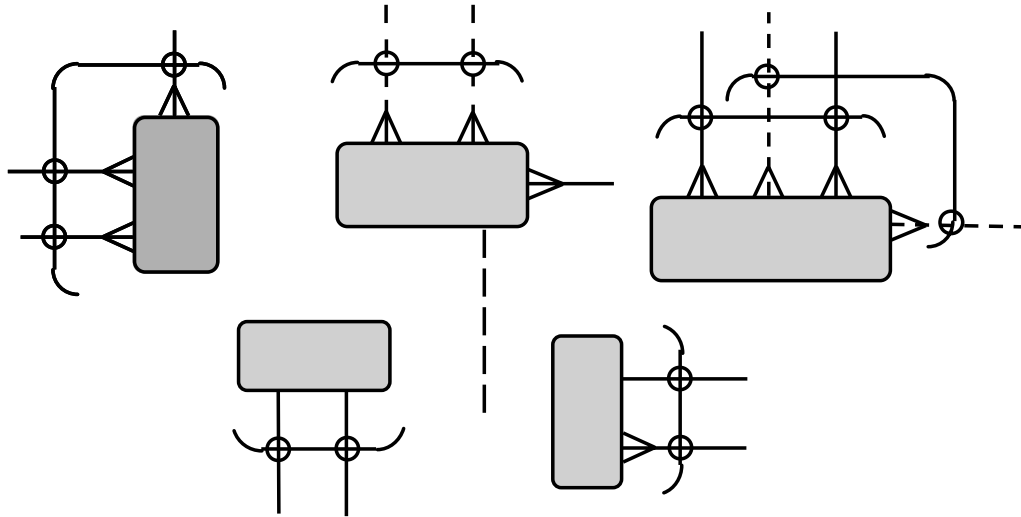
#### Note

- The relationship *contained in/container of* from LIST ITEM to LIST (the one that is printed in gray) is *not* part of the arc as there is no small circle at the intersection with the arc.
- A relationship that is part of a UID may also be part of an arc.
- The constraint that a LIST may only contain LISTS other than itself cannot be shown in the model.

### Mandatory Compared to Optional Relationships in an Arc

When the arc is drawn across two mandatory relationships, as in the example above, it means that every instance of CONTRACT COMPONENT must have one valid relationship. When the arc is drawn across two optional relationships, it would mean that an instance may have one valid relationship.

## Possible Arc Constructs



ORACLE

4-17

Copyright © Oracle Corporation, 2002. All rights reserved.

### Where Arcs Lead

An arc is normally implemented as a *check constraint* in an Oracle database. Note that a check constraint is not an ISO standard relational database object. In other words, an arc must be implemented differently in other database systems.

#### Some Rules About Arcs

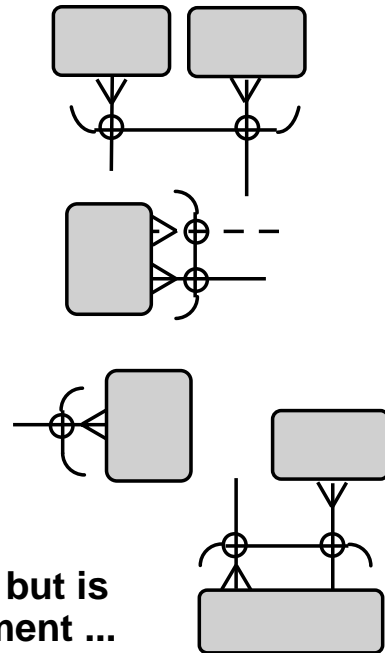
- An arc always belongs to one entity.
- Arcs can include more than two relationships.
- Not all relationships of an entity need to be included in an arc.
- An entity may have several arcs.
- An arc should always consist of relationships of the same optionality: all relationships in an arc must be mandatory or all must be optional.
- Relationships in an arc may be of different degree, although this is rare.

#### Tips About Arcs

- Do not include a relationship in more than one arc, for clarity reasons.
- Consider modeling subtypes instead of arcs (see the next paragraph).

## Some Incorrect Arc Constructs

- The arc “belongs” to one entity
- Relationships in the arc must be of the same optionality
- Arcs must contain at least two relationships



**An arc may be correct, but is quite difficult to implement ...**

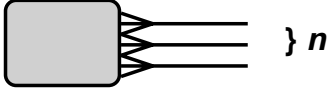
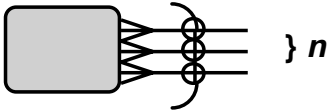
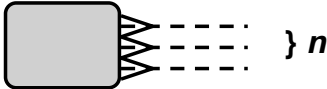
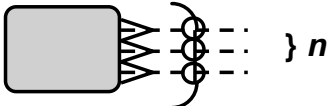
ORACLE

### Incorrect Arcs

You cannot capture all possible relationship constraints with arcs. For example, if two out of three relationships must be valid, this cannot be represented. The table below shows what an arc can express.

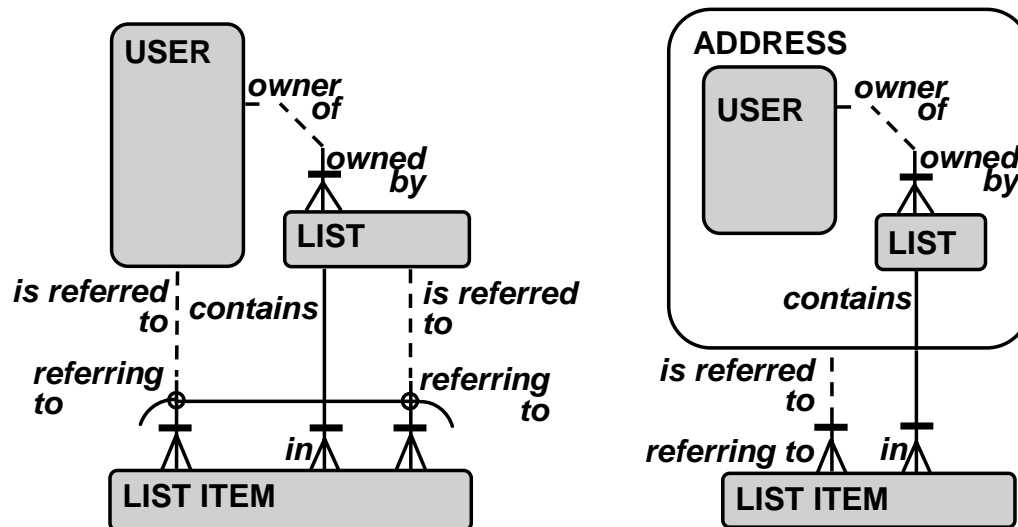


## Some Incorrect Arc Constructs

Number of Valid Relationships in Arc Per Entity Instance	Minimum	Maximum
 } <i>n</i>	n	n
 } <i>n</i>	1	1
 } <i>n</i>	0	n
 } <i>n</i>	0	1

ORACLE

## Arc or Subtype



ORACLE

4-20

Copyright © Oracle Corporation, 2002. All rights reserved.

### Arc or Subtypes

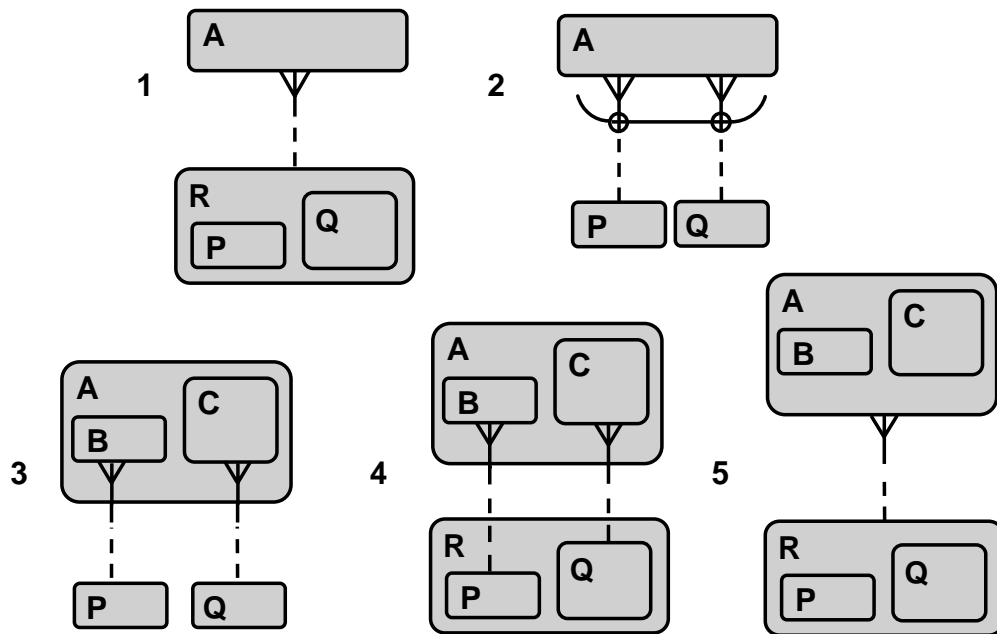
Relationships within an arc are often of a very similar nature. They frequently carry exactly the same names. If that is the case, an arc can often be replaced by a subtype construction, as the illustration shows. On the left you see the arc that contains both *referring to* relationships of **LIST ITEM**. In the model on the right there is only one relationship left, now connected to an entity **ADDRESS**, a new supertype entity of **USER** and **LIST**.

Both models are equivalent.

The model on the left emphasizes the difference between **USER** and **LIST**, which clearly exists; the other model emphasizes the commonality. This commonality is mainly a functional issue. Both **USERS** and **LISTS** can be part of a **LIST** and both can be used as the address in the To, Cc or Bcc field in the screen for composing a message.

Generally speaking, you can replace every arc with a supertype/subtype construction and every supertype/subtype construction with an arc.

## Arc and Subtypes



ORACLE

4-21

Copyright © Oracle Corporation, 2002. All rights reserved.

### More About Arcs and Subtypes

Arcs and Subtypes are similar notions. The five models that are printed below all show the same context.

Model 1 and 2 are equivalent models to what you have seen before.

If every instance of **A** is related to a **P** or a **Q**, then you could say there are **P-related-A's** and **Q-related-A's**. These two subtypes of **A** are shown in model 3.

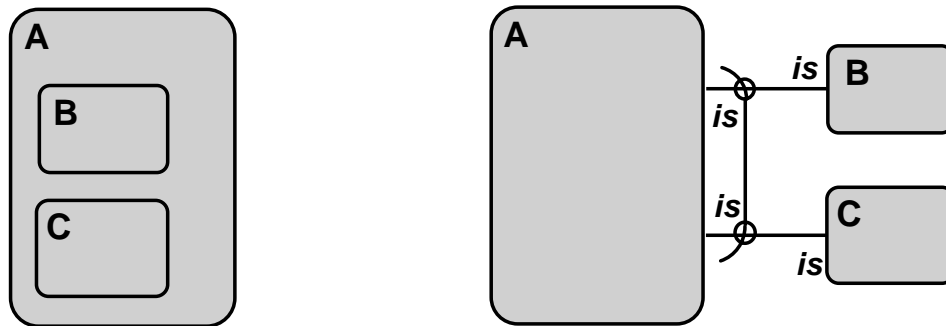
Model 4 goes one step beyond this and shows subtypes of entity **A** and a supertype **R** of **P** and **Q**.

Though models 3 and 4 are completely correct, it is likely they both model something twice.

Note that only model 5 does not present the same information. In model 5, an instance of **B** may be related to an instance of **Q**, unlike that which is modeled in 3 and 4.

## Subtypes Hide Relationships in Arc

- Every A is either a B or a C
- Every B is an A
- Every C is an A
- Every A must *be* a B or *be* a C
- Every B must *be* an A
- Every C must *be* an A

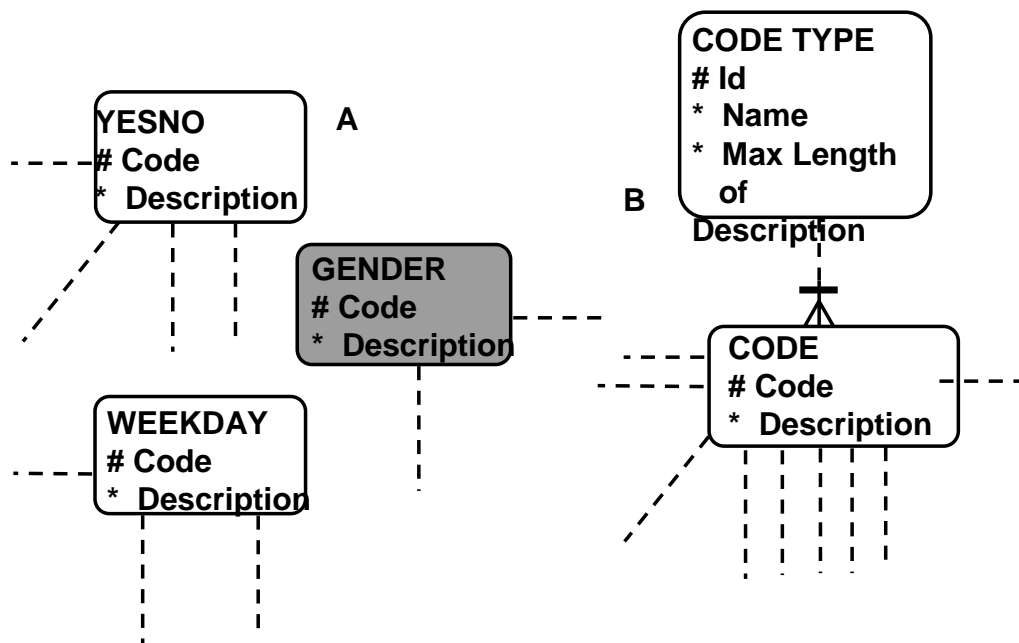


ORACLE

### Hidden Relationships

Every subtype hides a relationship between the subtype and its supertype. Moreover, the relationships are in an arc, as the next illustration shows. Both relationships are mandatory 1:1 *is/is* relationships.

## Value sets



ORACLE

4-23

Copyright © Oracle Corporation, 2002. All rights reserved.

## Domains

A very common type of *attribute constraint* is a set of values that shows the possible values an attribute can have. Such a set is called a *domain*.

Very common domains are, for example:

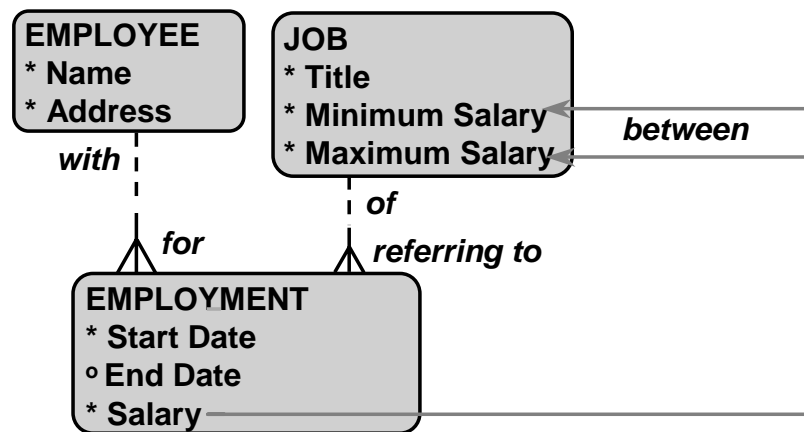
**YesNo:** Yes, No - **Gender:** Male, Female, Unknown - **Weekday:** Sun, Mon, Tue, Wed, Thu, Fri, Sat

In a conceptual data model you can recognize these as entities with, usually, only two attributes: Code and Description. These domain entities are referred to frequently but do not have any “many” relationships of their own, (see model A below). Typically, you would know all the values before the system is built. The number of values is normally low. Often you would deliver such a system with non-empty code tables

An alternative model for the (sometimes many) code entities is a more generic, two- entity approach: CODE and CODE TYPE, model B. Model A has the advantage of fewer relationships per entity as well as easy-to-understand entities; B has obviously fewer entities and therefore will lead to fewer tables.

Domains that have a large number of values, such as all positive integers up to a particular value, are usually not modeled. You should list and describe such a constraint in a separate document.

## Other Constraints: Range Check



ORACLE

### Some Special Constraints

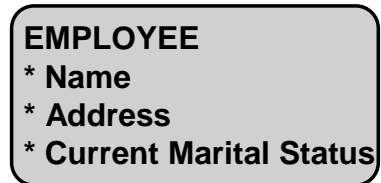
Although an entity relationship model can express many of the constraints that are not too complex, there are many types of constraints left that cannot be modeled. These constraints must be listed on a separate document and often need to be handled programmatically.

#### Categories: Examples

- **Conditional domain:** The domain for an attribute depends on the value of one or more attributes of the same entity.
- **State value transition:** The set of values an attribute may be *changed to* depends on the current value of that attribute.
- **Range check:** A numeric attribute must be between attribute values of a related instance.
- **Front door check:** A valid relationship must only exist at creation time.
- **Conditional relationship:** A relationship must exist or may not exist, if an attribute (of a related entity) has a special value.
- **State value triggered check:** A check must take place when an attribute is given a value that indicates a certain state.
- There are also combinations of the above.

Constraint: Employee salary must be within the salary range of the job of the employee.

## Other Constraints: State Value Transition



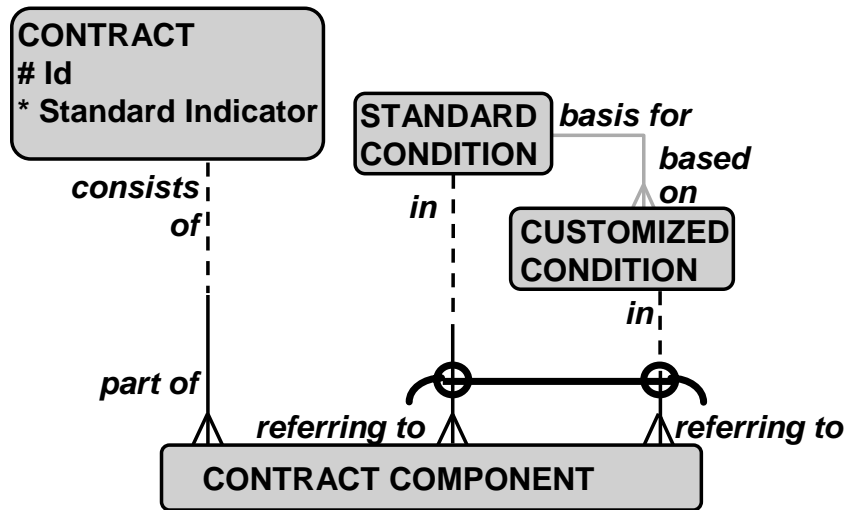
Possible Marital Status Transitions	to				
	Sin	Mar	Wid	Div	DP
Single	m	✓			✓
Married			✓	✓	
Widowed		✓			✓
Divorced		✓			✓
Domestic Partnership	✓	✓			

ORACLE

### Other Constraint: State Value Transition

Constraint: Marital Status of employees cannot change from any value to all other values.

## Conditional Relationship



ORACLE

4-26

Copyright © Oracle Corporation, 2002. All rights reserved.

### Conditional Relationship

Constraint: If a CONTRACT has Standard Indicator set to Yes, the CONTRACT COMPONENT may not refer to a CUSTOMIZED CONDITION.

#### Derived Attribute?

You may argue that the attribute Standard Indicator of CONTRACT is derivable. If the contract contains CUSTOMIZED CONDITIONS, it is, by consequence, not a standard CONTRACT. This may be true, but it is not necessarily so. Suppose the contract is created in various steps, by various people with different responsibilities. Then, the creation of a CONTRACT is a process that may take days. The Standard Indicator, then, is an attribute of that process. Only when the CONTRACT is finalized, should a check be made that the Indicator corresponds with the actual STANDARD and CUSTOMIZED CONDITIONS. In those situations, the entity CONTRACT will usually have an attribute Completed Indicator that triggers the check when set to Yes.



## **Conditional Relationship**

### **Rules May Lead to Attributes**

If you cannot capture a constraint in the model, the best you can do within the model is make the model rich enough so that a program for constraint checking performs well.

Consider the rule:

If the Standard Indicator is set to No, and there is no CUSTOMIZED CONDITION, then the CONTRACT is not yet ready for being sent to the CUSTOMER.

This rule deals with a procedure and cannot be modeled as such, but it calls for an indicator at entity CONTRACT to indicate something like a Ready To Send status.

### **Model for Overview**

An analyst often runs into constraints that cannot be modeled and thus must be documented separately. This is not a weakness of the model. An important goal of a diagram is to give an overall picture, not all the details. The model should let you view the key areas clearly.

# Boundaries

unrelated entity

**EXTERNAL**  
# Id  
\* Description  
\* Value

and possible implementation

## EXTERNALS

Id	Description	Value
1	Value added tax %	15
2	Maximum available Space per Mail User in Mbyte	500
3	Maximum level of Nested Mail Folders	3
4	Maximum level of Nested Mail Lists	16

ORACLE

## Boundaries

More than once the checking of constraints or special rules needs to use information that is not directly related to one of the entities in the model.

Typical examples are rules and boundaries set by external sources, like a mother company or national legislation. If reasonably possible, these rules should be part of your conceptual data model, and should not be hard coded in your programs. The reason is obvious: if the rule changes, which is beyond your power, there is a chance you do not have to make changes to your programs. Only an update of a value in a table would be necessary. The time spent developing a complete model is fully justified by the programming time saved.

## Summary

- **Identification**
  - Can be a real problem in the real world
  - Models cannot overcome this
- **Entities must have at least one Unique Identifier**
- **Unique Identifiers consist of attributes or relationships or both**
- **Arcs**
- **Many types of constraint are not represented in ER model**

ORACLE

4-29

Copyright © Oracle Corporation, 2002. All rights reserved.

### Summary

Entities in the real world must be individually identified before they can be represented in a database. You would not know what you are talking about, otherwise. Some entities are really difficult to identify, such as people and paintings. Some are more easy, especially when they are part of the domain as you can make up the rules, such as a unique number for each of the invoices you send to your customers. Some unique identifiers are already present in the real world, often as a combination of attributes and relationships of the entity.

Arcs in a diagram represent a particular type of constraint for the relationships of one entity. Many business constraints cannot be represented in a diagram and must be listed separately. This way the model remains clear and not too full of graphical elements.

# Practices

- **Identification Please**
- **Identification**
- **Moonlight UID**
- **Tables**
- **Modeling Constraints**

ORACLE®

## Practice: Identification Please

- A city
- A contact person for a customer
- A train
- A road
- A financial transaction
- An Academy Award (Oscar)
- A painting
- A T.V. show

ORACLE

4-31

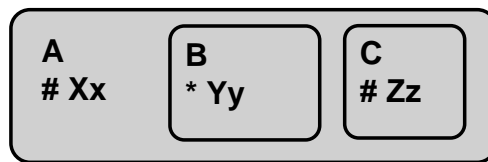
Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 4-1: Identification Please

#### Your Assignment

Describe how you would identify the following entities, making up any attributes and relationships you consider appropriate.

## Practice: Identification 1



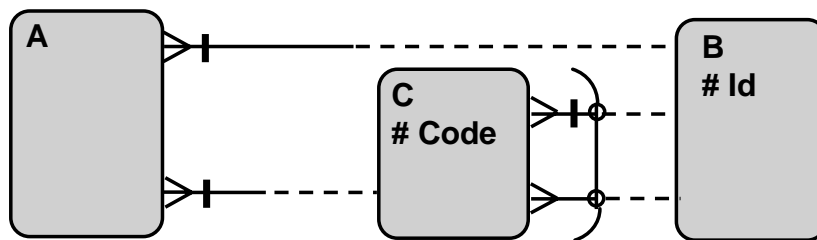
ORACLE

### Practice 4-2: Identification

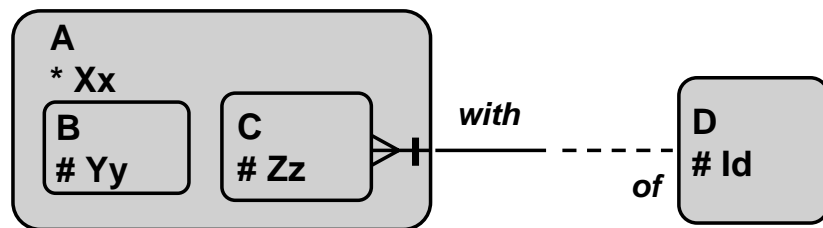
#### Your Assignment

Are the entities in the next diagrams identifiable?

## Practice: Identification 2

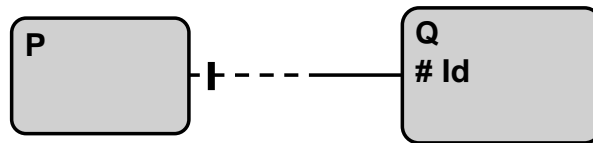


## Practice: Identification 3



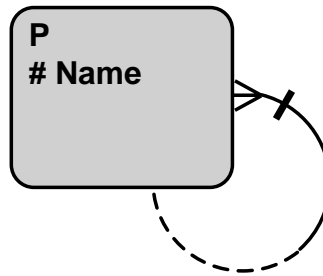


## Practice: Identification 4



ORACLE

## Practice: Identification 5

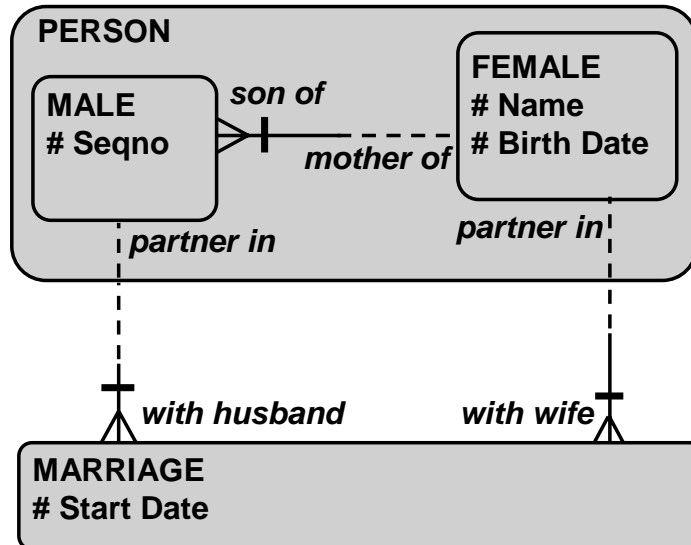


ORACLE

### Practice 4-2: Identification

**Note:** the next model describes a context that may be different from the world you are familiar with.

## Practice: Identification 6

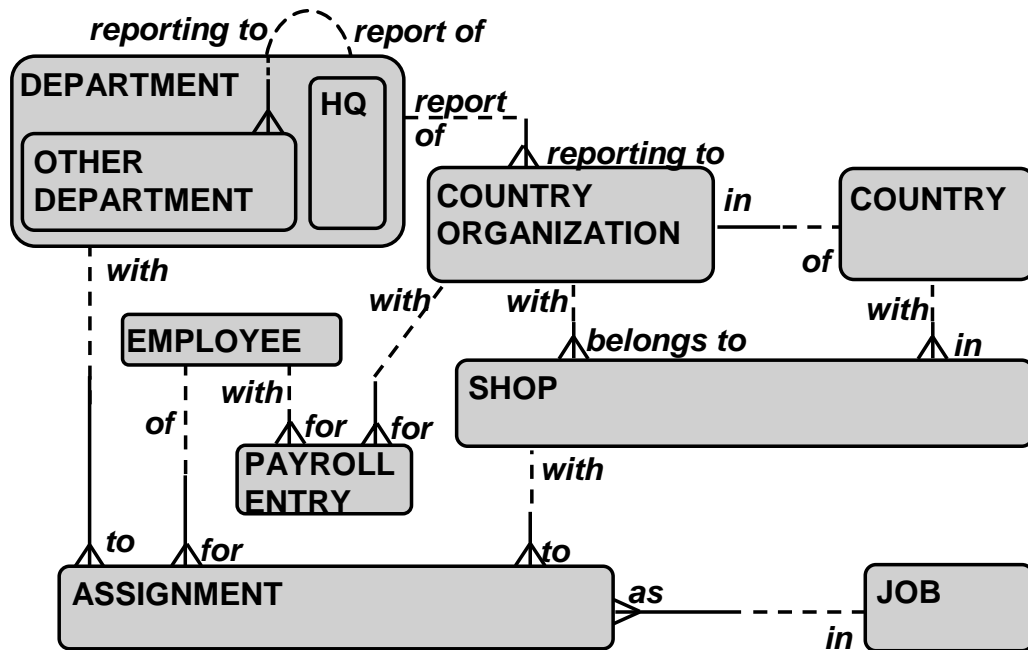


### Practice 4-2: Identification

Given the above model, answer the following questions.

1. Can person A marry twice?
2. Can person A marry twice on the same day?
3. Can person A marry with person B twice?
4. Can person A marry with person B twice on the same day?
5. Can person A be married to person B and person C simultaneously?
6. Can person A be married to person A?

## Practice: Moonlight UID



### Practice 4-3: Moonlight UID

#### Goal

The purpose of this practice is to define UIDs for given entities.

#### Scenario

Moonlight Coffees, organization model.

#### Your Assignment

Use what you know about Moonlight Coffees by now, and, most importantly, use your imagination.

1. Given the model below, indicate UIDs for the various entities. Add whatever attributes you consider appropriate. Country organizations have a unique “tax registration number” in their countries.
2. Are there any arcs missing?

## Practice: Table 1

- **“In a relational database system, data is stored in tables. Tables of a database user must have a unique name. A table must have at least one column. A column has a unique name within the table. A column must have a data type and may be Not Null.**
- **Tables can have one primary key and any number of unique keys. A key contains one or more columns of the table. A column can be part of more than one key.**

ORACLE

4-39

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 4-4: Tables

#### Goal

The purpose of this practice is to match a given context with a ER model.

#### Your Assignment

Read the text on ISO Relational tables.

Do a quality check on the ER model based on the quoted text and what you know about this subject. Also list constraints that are mentioned in the text but not modeled.

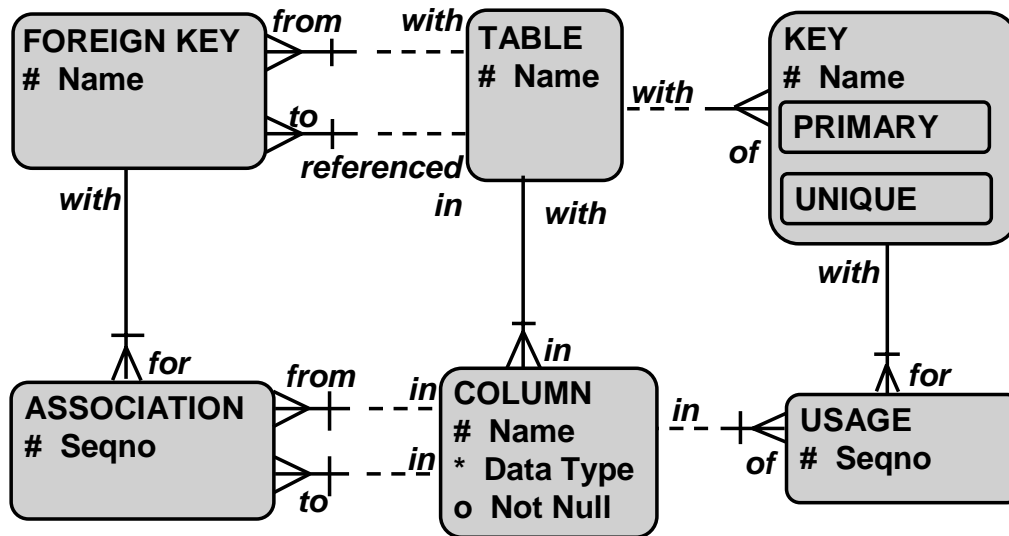
## **Practice: Table 1**

**A table can have foreign keys. A foreign key always connects one table with another. A foreign key consists of one or more columns of the one table that refers to key columns of the other table.**

**“The sequence of columns within the key and foreign key is important.”**

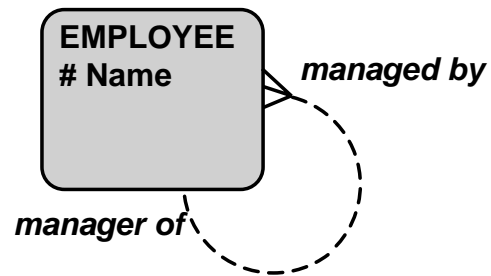
**ORACLE**

## Practice: Table 2



ORACLE

## Practice: Constraints 1



### Practice 4-5: Modeling Constraints

#### Goal

The purpose of this practice is to learn what constraints can be modeled and how, and which cannot be modeled.

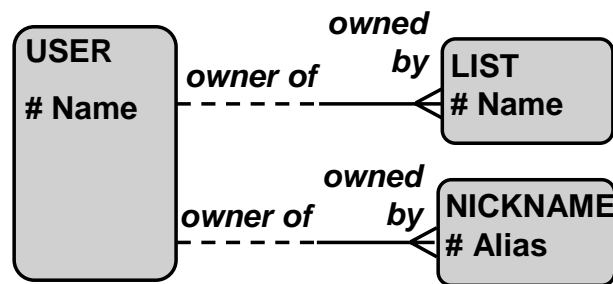
#### Your Assignment

Change the diagrams to model the constraint given.

1. Every EMPLOYEE must have a manager, except the Chief Executive Officer.



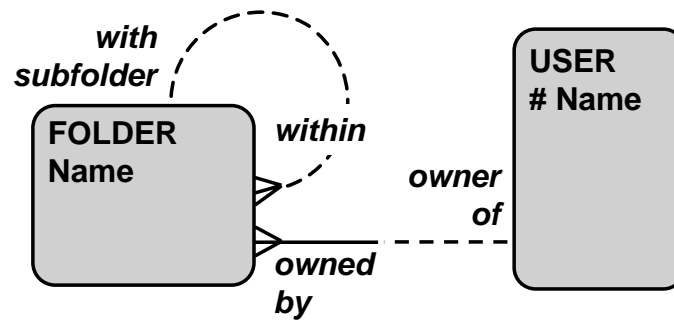
## Practice: Constraints 2



### Practice 4-5: Modeling Constraints

2. A user may not use the same name for both NICKNAME and LIST name.

## Practice: Constraints 3



### Practice 4-5: Modeling Constraints

3. A top level FOLDER must have a unique name per user; sub folders must have a unique name within the folder where they are located.

# 5

## Modeling Change

ORACLE®

Copyright © Oracle Corporation, 2002. All rights reserved.

# Overview

- **Date and time**
- **Modeling change over time**
- **Prices change**
- **Journaling**

ORACLE

5-2

Copyright © Oracle Corporation, 2002. All rights reserved.

## Introduction

Every update of an attribute or transfer of a relationship means loss of information. Often that information is no longer of use, but some systems need to keep track of some or all of the old values of an attribute. This may lead to an explicit time dimension in the model which is usually quite a complicated issue.

Time is often present in a business context, as many entities are in fact a representation of an event. This lesson discusses the possibilities and difficulties that arise when you incorporate time in your entity model.

## Objectives

At the end of this lesson, you should be able to do the following:

- Make a well considered decision about using entity DATE or attribute Date
- Model life cycle attributes to all entities that need them
- List all constraints that arise from using a time dimension
- Cope with journaling

## Change and Time

- **Every update means loss of information.**
- **Time in your model makes the model more complex.**
- **There are often complex join conditions.**
- **Users can work in advance.**
- **When do you model date/time as an entity?**
- **What constraints do arise?**
- **How do you handle journaling?**

ORACLE

5-3

Copyright © Oracle Corporation, 2002. All rights reserved.

### Modeling Time

In many models time plays a role. Often entities that are essentially *events* are part of a model, for example, PURCHASE, ASSIGNMENT. One of the properties you record about these entities is the date or date and time of the event. Often the date and time are part of a unique identifier.

A second time-related issue often helps to increase the usability of a system dramatically. By adding dates like Start, Expiry, End Date, to data in the system, you allow users to work in advance. Suppose a particular value, say the price of gas or diesel, will change as of January 1. It is very useful to be able to tell the system the new value long before New Year's Eve. By adding a time dimension to the model you make the system independent of the *now*.

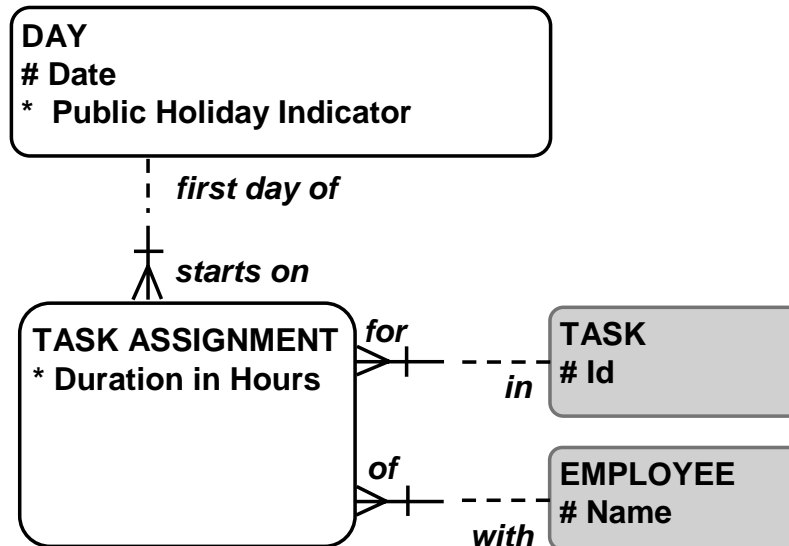
As always, there is a price for adding things such as this. Adding a time dimension to your conceptual data model makes the model considerably more complex. In particular, the number of constraints and business rules that must be checked will increase.

A third time-related issue in conceptual data models is connected to the concept of *logging* or *journaling*. Suppose you allow values to be updated, but you want to keep track of some of the old values. In other words, what do you do when you need to keep a record of the history of attribute values, of relationships, of entire entities?

The following issues arise:

- When do you model date/time as an entity, and when as an attribute?
- How do you handle the constraints that arise in systems that deal with time-related data?
- How do you handle journaling?

## Entity DAY



### Entity DAY

It is not only systems that deal with historical information that struggle with dates. Sometimes a system needs to know more about a day than can be derived from its date. A planning system, for example, often needs to know if a particular day is a public holiday. Many data warehouse systems use a calendar that is different from the normal one, for example, where a year is divided into four-week periods or 30 day Months or Quarters where Q1 starts in the middle of May.

Some warehouses need weather information about days in order to do statistical analysis about the influence of the weather on, for example, their sales. In these cases a day has attributes or relationships of its own and should be modeled as entity DAY.

The above model shows part of a planning system where tasks are assigned to employees. Tasks may take from a few hours to, at maximum, several days.

Based on this model, table TASK\_ASSIGNMENTS will contain a date column that is a foreign key column to the DAYS table.

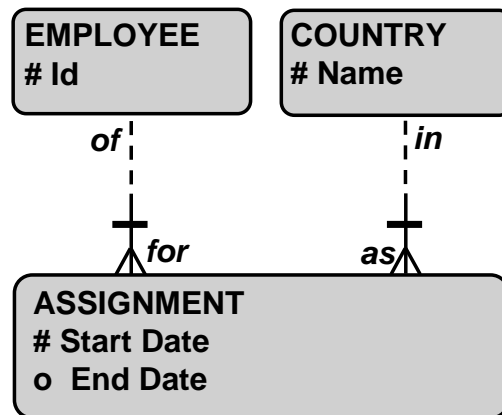
## **Modeling Days Over Time**

### **Date and Time**

As stated earlier, an Oracle DATE column always contains date and time. This needs some special attention as two DATE columns may apparently contain the same date but they are not equal because of a difference in their time component.

While modeling, always make explicitly clear when time of the day is an issue, for instance, by naming the attribute DateTime. As soon as hours and minutes play a role, the concepts of “time zone” and “daylight saving time” may become important.

## Modeling Change



ORACLE

### Modeling Changes Over Time

Date and Time in your models may substantially increase the complexity of your system, as the next example shows.

The context for this example is that of an Embassy Information System, but could have been chosen from almost any business area.

Embassy employees have an assignment for a country, but, of course, the assignments may change over time. Therefore, the model would need an entity **ASSIGNMENT** with a mandatory attribute **Start Date** and an optional **End Date**. **Start Date** is modeled as part of the UID for **ASSIGNMENT**. This means that the model allows an employee to have two assignments in the same country, as long as they start on different days. It also allows the employee to have two assignments that start on the same day, as long as these are for different countries.

Suppose we know today that Jacqueline will switch from Chili to Morocco on the first of next month. This fact can be fed into the system immediately, by creating a new instance of **ASSIGNMENT** with a **Start Date** that is still in the future at create time. The future users will appreciate this kind of functionality.



## **Modeling Changes Over Time**

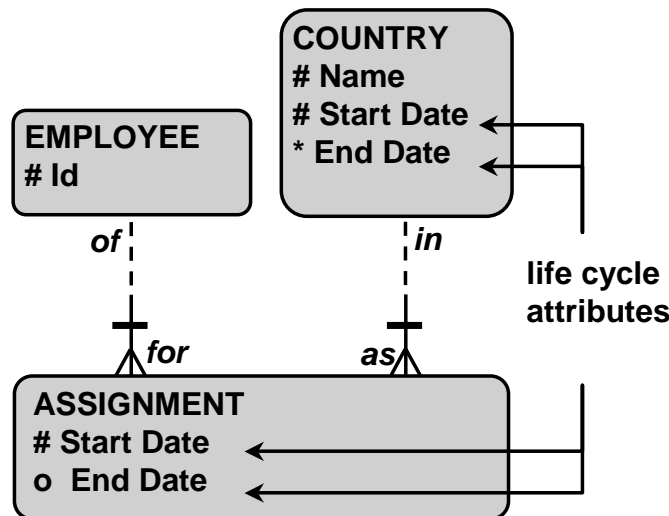
### **End Date Redundant?**

You may argue that attribute End Date of ASSIGNMENT is redundant because Jacqueline's assignments follow each other: the End Date of Jacqueline's assignment in Chili matches the Start Date of the one in Morocco. This may be true, but it does not take into consideration that embassy people may take a leave and return after a couple of years. In other words, if you do not model attribute End Date you ignore the possibility that the assigned periods of a person are not contiguous.

Note that the model does allow an employee to have two assignments in, for example, Honduras, that overlap! The unique identifier does not protect the data against overlapping periods. Adding End Date to the UID does not help.

You would need a whole series of extra constraints to cope with this.

## Even a Country Has a Life Cycle



ORACLE

### Modeling Changes Over Time

#### Countries Have a Life Cycle Too

Suppose the Embassy Information System contains data that goes back to at least the late eighties. In those days the USSR and Zaire were still countries. Suppose there are ASSIGNMENTS that refer to the USSR and Zaire. In the case of Zaire, you could consider an update of the Name of the COUNTRY: Democratic Republic Congo is essentially just the new name for Zaire. In case of the USSR this would not make sense. There is not a new name for the old country. The old country simply ceased to exist when it broke into several countries. Although the concept of a country seems very stable, countries may change fundamentally during the lifetime of the information system.

This leads to the next model.

## Modeling Changes Over Time

### Time-related Constraints

Be aware of the numerous constraints that result from the time dimension! Here is a selection:

- An ASSIGNMENT may only refer to a COUNTRY that *is valid* at the Start Date of the ASSIGNMENT.
- The obvious one: End Date must be past Start Date.
- A business rule: ASSIGNMENT periods may not overlap. The Start Date of an ASSIGNMENT for an EMPLOYEE may not be between any Start Date and End Date of an other ASSIGNMENT for the same EMPLOYEE.
- As for the previous constraint, but for End Date.
- You would probably not allow an ASSIGNMENT to be transferred to another COUNTRY, unless the ASSIGNMENT has not yet started, that is, the Start Date of the ASSIGNMENT is still in the future.
- This is an example of *conditional nontransferability*.

*For updates of the attribute Start Date here are some possible constraints:*

- *A Start Date of an ASSIGNMENT may be updated to a later date*, unless this date is later then the End Date (if any) of the COUNTRY it refers to.
- A Start Date of an ASSIGNMENT may be updated to a *later* date, if the current Start Date is still in the future.
- A Start Date of an ASSIGNMENT may be updated to an *earlier* date, unless this date is earlier than the Start Date of the COUNTRY it refers to.
- A Start Date of an ASSIGNMENT may be updated to an *earlier* date, if this new date is still in the future.
- A Start Date of a COUNTRY may be updated to a *later* date, if there are no ASSIGNMENTS that would get disconnected.

Similar constraints apply to attribute End Date.

### Referential Logic

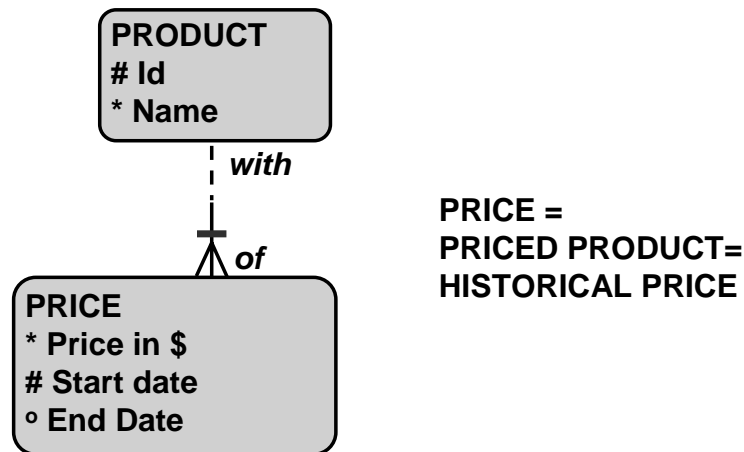
Note that, except for two, these constraints result from *referential logic* only. There may be more additional business constraints.

Imagine the sheer number of constraints if a time-affected entity is related to several other time-affected entities! Fortunately, these constraints all have a similar pattern; these result from the referential, time related, logic.

### Implementation

In an Oracle environment, one of these constraints can be implemented as a check constraint, (End Date must be later than Start Date). All the others will be implemented as database triggers.

## Products and Prices



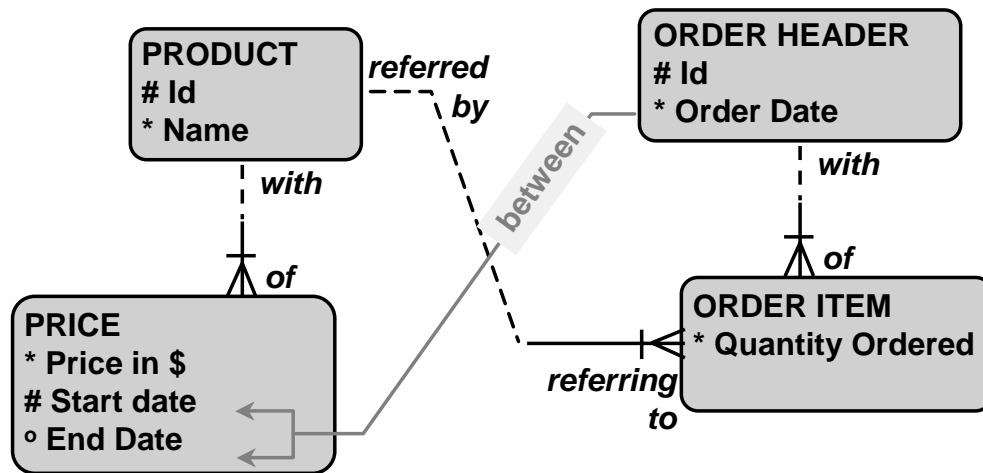
ORACLE

### A Time Example: Prices

Products have a price. Prices change. Old prices are probably of interest. That leads to a model with entities **PRODUCT** and **PRICE**. The latter entity contains the prices and the time periods they are applicable. In real-life situations you find the concept of **PRICE** also named **PRICED PRODUCT**, **HISTORICAL PRICE** (and less appropriate: price list or price history); all these names more or less describe the concept.

You may argue the need for an **End Date** attribute. If the various periods of a product price are contiguous, **End Date** is obsolete. If, on the other hand, the products are not always available, as in the fruit and vegetable market, the periods should have an explicit **End Date**.

## What Price to Pay?



ORACLE

5-11

Copyright © Oracle Corporation, 2002. All rights reserved.

### A Time Example: Prices

#### Introducing Order Header and Order Item

Here, entities ORDER HEADER and ORDER ITEM are introduced. An ORDER HEADER holds the information that applies to all items, like the Order Date and the relationship to the CUSTOMER that placed the order or the EMPLOYEE that handled it. (For clarity, these relationships are not drawn here.) The ORDER ITEM holds the Quantity Ordered and refers to the PRODUCT ordered. The price that must be paid can be found by matching the Order Date between Start Date and End Date of PRICE. Note that you cannot model this “between relationship”.

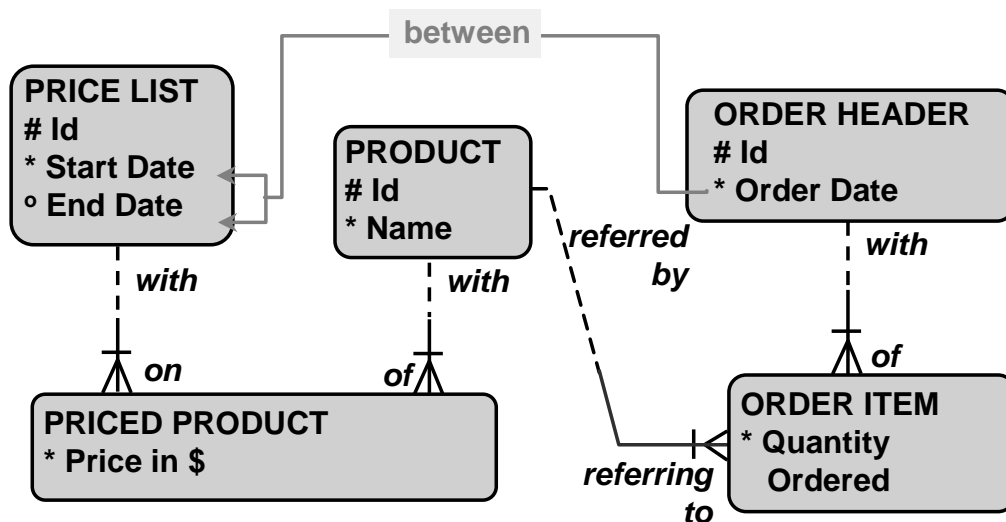
This model is a fairly straightforward product pricing model and is often used.

#### Order

Note that the concept of an order in this model is composed of ORDER HEADER and ORDER ITEM.

To find the order total for an order, it would need a join over four tables.

## Price List Search



ORACLE

5-12

Copyright © Oracle Corporation, 2002. All rights reserved.

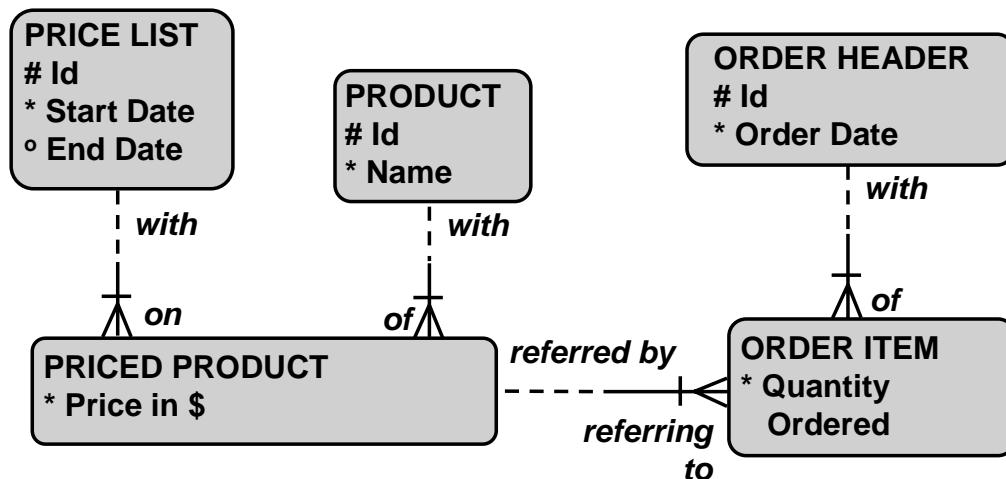
### A Time Example: Prices

#### Price List

A variant on the above model is often used when prices as a group are usually changed at the same time. The period that prices are valid is the same for many prices; that would lead to this model:

Entity PRICE LIST represents the set of prices for the various products; PRICED PRODUCT represents the price list items. To know the price paid for an ordered item, you take the Order Date of the ORDER HEADER, and take the PRICE LIST that is applicable at that date. Next, you go from ORDER ITEM to the PRODUCT that is referred to and from there to the PRICED PRODUCT of the PRICE LIST you have just found. To find the order total for an order, it would need a join over five tables.

## Order for Priced Products



ORACLE

5-13

Copyright © Oracle Corporation, 2002. All rights reserved.

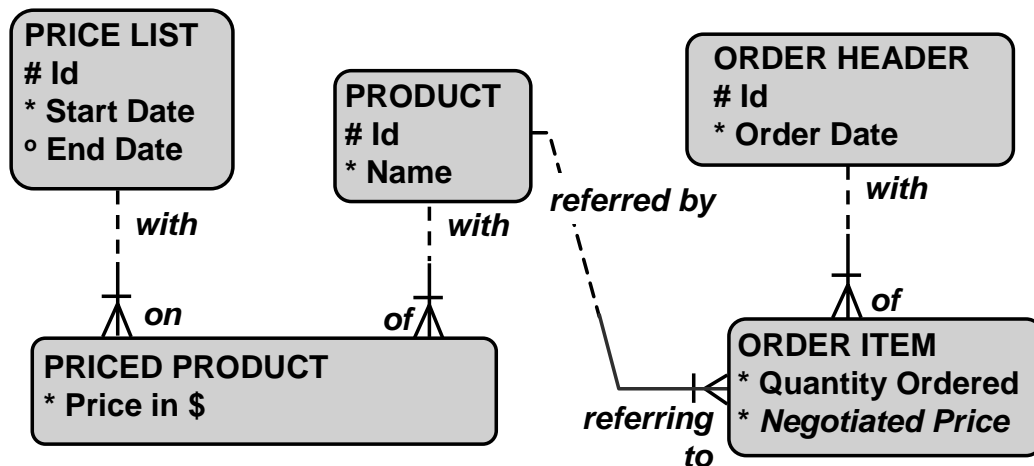
### A Time Example: Prices

#### Buying a **PRODUCT** or a **PRICED PRODUCT**?

Another variant of a pricing model is shown here.

Here an **ORDER ITEM** refers directly to a **PRICED PRODUCT**. At create time of the **ORDER ITEM** the constraint is applied that the Order Date must match the correct **PRICE LIST** period. To find the order total for an order now only requires three tables.

## Negotiated Prices



ORACLE

5-14

Copyright © Oracle Corporation, 2002. All rights reserved.

### A Time Example: Prices

#### Negotiated Prices

When prices are subject to negotiation, the model becomes simpler. Negotiated Price is now an attribute of entity ORDER ITEM; ORDER ITEM refers to PRODUCT. Every referential constraint can be modeled.

This model may seem to hold derivable information, but this is not true. Even in the case that almost all Negotiated Prices are equal to the current product price, you have to model Negotiated Price at ORDER ITEM level, just because of the small chance of an exception. To find the order total you require only two tables. You can imagine that many analysts choose this variant of the model as a safeguard, even if there is nothing to negotiate at present.



## **A Time Example: Prices**

### **Which Variant to Use and When?**

Typically, the model with the negotiated prices will occur where the number of ORDER ITEMS per ORDER HEADER is low, often just a single one, and where the value is high, as, for example, in the context of a used car business.

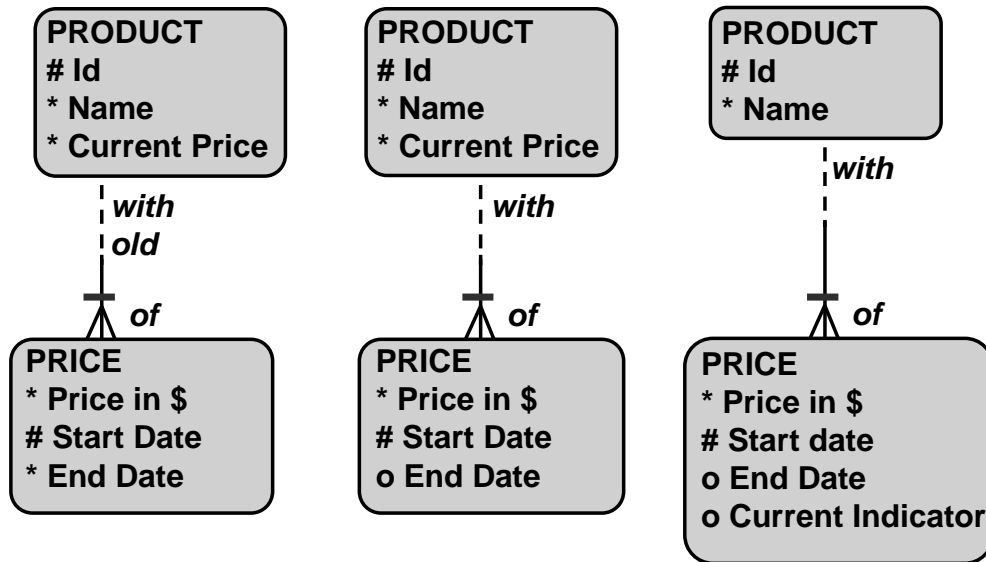
You see ORDER ITEM referring to a PRODUCT most often in the situation where prices do not change frequently. The number of items per ORDER HEADER is often well over one, and the overall value limited. Typical examples are the fashion industry and grocery stores.

The model with ORDER ITEM referring to PRICED PRODUCT is often used in businesses where prices often change, as in the fresh fruit and vegetable markets. Prices there may even change during the day.

The model with attribute Current Price for a PRODUCT is typically the model for the supermarket environment where instant availability of prices at the checkouts is vital.

As stated earlier, the best model for a particular context depends on functional needs. See more on this in the chapters on Denormalized Data and Design Considerations.

## Current Prices



ORACLE

5-16

Copyright © Oracle Corporation, 2002. All rights reserved.

### Current Price

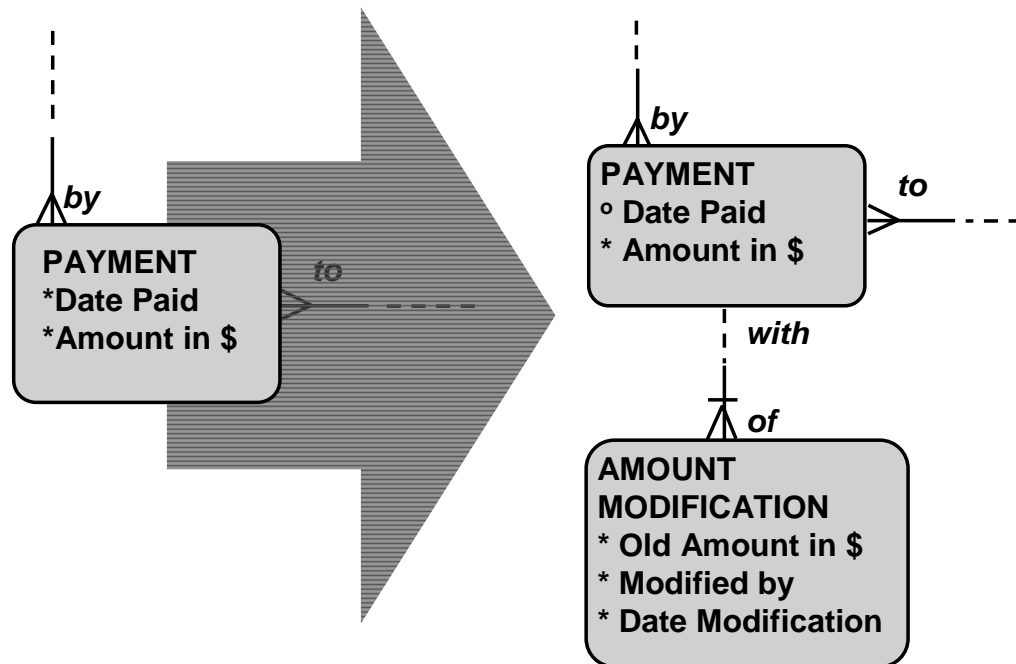
These models are variants on the **PRODUCT-PRICE** model you have seen before.

In the left-hand model the 1:m relationship between **PRODUCT** and **PRICE** shows the real historical prices only. You can guess that only historical prices are kept because attribute **End Date** is mandatory; an additional constraint is that this value should always be in the past. The **Current Price** of a **PRODUCT** is represented as an attribute. This model does not have any redundancies.

In many situations it would be a good *design* decision to keep the current product prices as well as the old prices in one table based on entity **PRICE**. The middle model is an ER representation of that situation. Note that **End Date** is now optional.

The right-hand model is another model that contains a subtle redundancy. See more on this type of redundancy in the lesson on **Denormalized Data**.

## Journaling



ORACLE

## Journaling

When a system allows a user to modify or remove particular information, the question should arise if the old values must be kept on record. This is called *logging* or *journaling*. You will often encounter this when the information is of a financial nature.

### Consequences for the Model

A journal usually consists of both the modified value and the information about who did the modification and when it was done. This extra information can, of course, be expanded if you wish. Apart from the consequences for the conceptual data model, the system needs special journaling functionality: any business function that allows an update of Amount In should result in the requested update, plus the creation of an entity instance **AMOUNT MODIFICATION** with the proper values. Of course, the system would need special functions as well in order to do something with the logged data.

## **Journaling**

### **No Journal Entity**

When several, or all, attributes of an entity need to be journalled, it is often implemented by maintaining a full shadow table that has the same columns as the original plus some extra to store information about the who, when, and what of the change. This table does not result from a separate entity; it is just a second, special, implementation of one and the same entity.

### **Journaling Registers Only**

Note that logging does not prevent a user from making updates. Preventing updates entirely is a functional issue and is invisible in the conceptual data model. Be aware that preventing updates altogether would also block the possibility to change typos or other mistakes.

At this stage, decisions must be made about the behavior of the system with respect to updates; sometimes this leads to modifications in the conceptual data model.

For example, suppose that in a particular business context a certain group of users is allowed to create instances of PAYMENT but is not allowed to change them. Changes can only be made by, say, a financial manager. Suppose you just created a PAYMENT instance and you discover you made a mistake. For those cases the business would need some mechanism to stop the erroneous instance. One mechanism would be to ask one of the financial managers to make the change. A far better mechanism would be to add functionality so that a payment can be neutralized. This may be represented in the model as an attribute Neutralized Indicator that users can set to Yes.

## Summary

- **Consider the need for keeping old values**
- **Time in your model is complicated:**
  - **Implicit versions**
  - **References**
- **Journaling**

ORACLE

5-19

Copyright © Oracle Corporation, 2002. All rights reserved.

### Summary

Every update in a system means loss of information. To avoid that you can create your model to keep a history of the old situations. Sometimes relationships refer to a time-dependent state of an entity. In other words, the updated entity is in fact a new instance of the entity and not an updated existing instance. If this is the case, the time-dependent referential constraints cannot be modeled by a relationship only.

Time in your model is a complicated issue. Many models have some time-related entities.

# Practices

- **Shift**
- **Strawberry Wafer**
- **Bundles**
- **Product Structure**

ORACLE®

## Practice: Shift

Museumplein, Amsterdam, March 21					
Shift	1	2	3	4	5
Mon	6:30 11:30	11:30 16:00	16:00 20:30	20:30 23:00	-
Tue	7:00 11:30	11:30 16:00	16:00 20:30	20:30 23:00	-
Wed	7:00 11:30	11:30 16:00	16:00 20:30	20:30 23:00	-
Thu	7:00 11:30	11:30 16:00	16:00 20:30	20:30 23:00	-
Fri	7:00 11:30	11:30 16:00	16:00 20:30	20:30 24:00	-
Sat/Sun	8:00 11:30	11:30 15:00	15:00 18:00	18:00 21:00	21:00 24:00

ORACLE

5-21

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 5-1: Shift

#### Goal

The purpose of this practice is to model various aspects of time.

#### Scenario

Some shops are open 24 hours a day, seven days a week. Others close at night. Employees work in shifts. Shifts are subject to local legislation. Below you see the shifts that are defined in one of the shops in Amsterdam.

#### Your Assignment

List the various date/time elements you find in this Shift scheme and make a conceptual data model.

## Practice: Strawberry Wafer

- **Prices are at the same level within a country; prices are determined by the Global Pricing Department. Usually the prices for regular, global products are re-established once a year.**
- **Prices and availability for local specialties are determined by the individual shops. For example, the famous Norwegian Vafler med Jordbær (a delicious wafer with fresh strawberries) is only available in summer. Its price depends on the current local market price of fresh strawberries.**

ORACLE

5-22

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 5-2: Strawberry Wafer

#### Scenario

You have modeled a price list in an earlier lesson. Now some new information is available.

#### Your Assignment

Revisit your model and make changes, if necessary, given this extra information.



# prijslijst

de Keyzer, Keyzerlei 15, Antwerpen  
bezoekt ons op 't Web: [www.moonlight.com](http://www.moonlight.com)

Practice: Price list

inclusief BTW  
16 September

	klein	middel	groot	
gewone koffie	60	90	120	
cappuccino	90	110	140	
koffie verkeerd	75	100	130	
speciale koffies	99	125	150	
espresso	60	95	110	
koffie van de dag	45	75	100	
<i>caffeine vrij</i>	<i>5</i>	<i>10</i>	<i>15</i>	<i>toeslag</i>
zwarte thees	60	100	120	
vruchten thees	75	110	130	
kruiden thees	80	120	140	
dag thee	50	85	100	
<i>caffeine vrij</i>	<i>5</i>	<i>10</i>	<i>15</i>	<i>toeslag</i>
frisdranken	60	100	130	
diverse sodas	60	100	130	
mineraal water	75	120	140	
appel taart				180
brusselse wafel				150
portie chocolade bonbons				150
koekje van eigen deeg				120
portie slagroom				30

ORACLE

## Practice: Bundles(1)

**A SweetTreat(tm) consists of a large soft drink plus cake of the day.**

**A BigBox(tm) consist of a large coffee of the day plus two cakes of the day.**

**A SuperSweetTreat(tm) consists of a SweetTreat(tm) plus whipped cream (on the cake).**

**A FamilyFeast(tm) consists of two BigBoxes(tm) plus two SweetTreats™ plus a small surprise.**

**A DecafPunch(tm) consists of a regular decaffeinated coffee or a regular decaffeinated tea, plus a blackberry muffin.**

ORACLE

5-24

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 5-3: Bundles

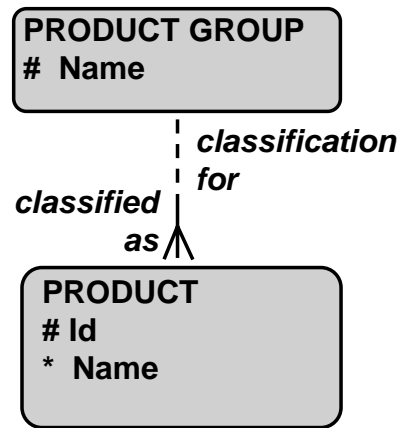
#### Goal

The purpose of this practice is to expand the concept of an old entity.

#### Scenario

As a test, Moonlight sells bundled products in some shops, for a special price. Here are some examples.

## Practice: Bundles(2)



ORACLE

5-25

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 5-3: Bundles

Bundles sell very well; all kinds of new bundles are expected to come.

The system should know how all these products are composed, in order to complete various calculations.

#### Your Assignment

1. Modify the product part of the model in such a way that the desired calculations can be completed.
2. Change the model in such a way that it allows for:

## Practice: Product Structure



ORACLE

5-26

Copyright © Oracle Corporation, 2002. All rights reserved.

### Practice 5-4: Product Structure

#### Goal

The purpose of this practice is to model a hierarchical structure.

#### Scenario

Moonlight needs to make sales information available as a tool to optimize its business. A hierarchical product structure is being developed to be able to report on different summary levels. This hierarchical structure should replace the single level product group classification. Below you see the current idea about a product structure. This structure is far from complete, but it should give you an idea of the shape the structure will take. The + signs mean that the structure will be expanded at that point.

#### Your Assignment

1. Create a model for a product classification structure.
2. (Optional) How would you treat the bundled products?