Database Processing:

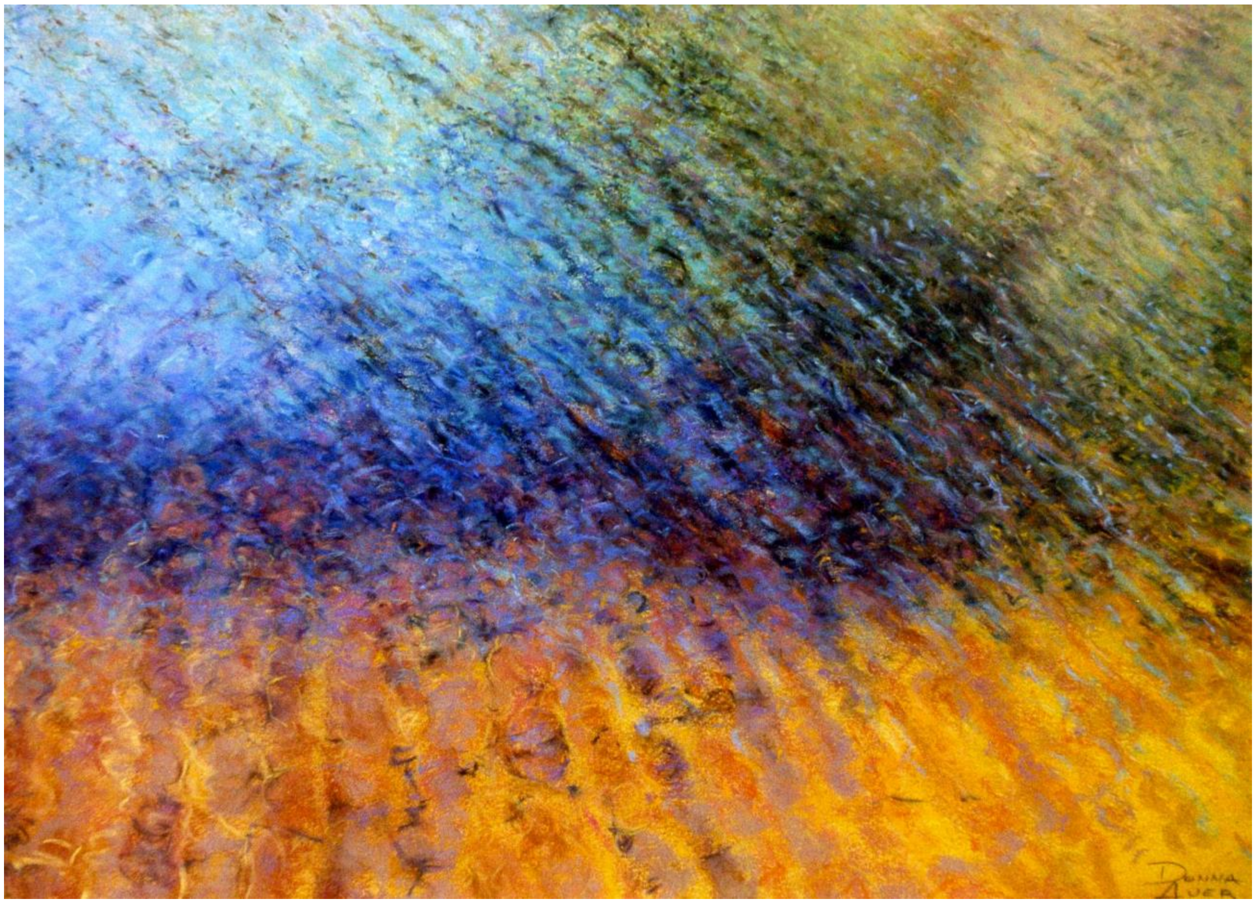
Fundamentals, Design, and Implementation

14th Edition

David M. Kroenke • David J. Auer

Online Appendix D

E-R Diagrams and the UML Standard



Vice President, Business Publishing: Donna Battista
Editor in Chief: Stephanie Wall
Acquisitions Editor: Nicole Sam
Program Manager Team Lead: Ashley Santora
Program Manager: Denise Weiss
Editorial Assistant: Olivia Vignone
Vice President, Product Marketing: Maggie Moylan
Director of Marketing, Digital Services and Products: Jeanette Koskinas
Executive Product Marketing Manager: Anne Fahlgren
Field Marketing Manager: Lenny Ann Raper
Senior Strategic Marketing Manager: Erin Gardner
Product Marketing Assistant: Jessica Quazza
Project Manager Team Lead: Jeff Holcomb
Project Manager: Ilene Kahn
Operations Specialist: Diane Peirano
Senior Art Director: Janet Slowik
Text Designer: Integra Software Services Pvt. Ltd.
Cover Designer: Integra Software Services Pvt. Ltd.
Cover Art: Donna Auer
Vice President, Director of Digital Strategy & Assessment: Paul Gentile
Manager of Learning Applications: Paul Deluca
Digital Editor: Brian Surette
Digital Studio Manager: Diane Lombardo
Digital Studio Project Manager: Robin Lazrus
Digital Studio Project Manager: Alana Coles
Digital Studio Project Manager: Monique Lawrence
Digital Studio Project Manager: Regina DaSilva
Full-Service Project Management and Composition: Integra Software Services Pvt. Ltd.
Printer/Binder: RRD Willard
Cover Printer: Phoenix Color/Hagerstown
Text Font: 10/12 Mentor Std Light

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on the appropriate page within text.

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided "as is" without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services.

The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time. Partial screen shots may be viewed in full within the software version specified.

Microsoft® Windows®, and Microsoft Office® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

MySQL®, the MySQL Command Line Client®, the MySQL Workbench®, and the MySQL Connector/ODBC® are registered trademarks of Sun Microsystems, Inc./Oracle Corporation. Screenshots and icons reprinted with permission of Oracle Corporation. This book is not sponsored or endorsed by or affiliated with Oracle Corporation.

Oracle Database 12*c* and Oracle Database Express Edition 11*g* Release 2 2014 by Oracle Corporation. Reprinted with permission. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Mozilla 35.104 and Mozilla are registered trademarks of the Mozilla Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

PHP is copyright The PHP Group 1999–2012, and is used under the terms of the PHP Public License v3.01 available at *http://www.php.net/license/3_01.txt*. This book is not sponsored or endorsed by or affiliated with The PHP Group.

Copyright © 2016, 2014, 2012 by Pearson Education, Inc., 221 River Street, Hoboken, New Jersey 07030. All rights reserved. Manufactured in the United States of America. This publication is protected by Copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission(s) to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, 221 River Street, Hoboken, New Jersey 07030.

Many of the designations by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

Library of Congress Cataloging-in-Publication Data

Kroenke, David M.
Database processing: fundamentals, design, and implementation/David M. Kroenke, David J. Auer.–Fourteenth edition.
pages cm
Includes bibliographical references and index.
ISBN 978-0-13-387670-3 (student edition)–ISBN 978-0-13-387676-5 (instructor's review copy)
1. Database management. I. Auer, David J. II. Title.
QA76.9.D3K76 2016
005.74–dc23
2015005632

10 9 8 7 6 5 4 3 2 1

PEARSON

ISBN 10: 0-13-387670-5
ISBN 13: 978-0-13-387670-3

Appendix D – 10 9 8 7 6 5 4 3 2 1

Chapter Objectives

- To understand UML-style E-R diagrams.
- To be able to model HAS-A, nonidentifying, identifying, and IS-A relationships using UML symbols.
- To understand the object-oriented programming language constructs used in UML.

What Is the Purpose of This Appendix?

The **Unified Modeling Language (UML)** is a set of structures and techniques for modeling and designing object-oriented programs (OOP) and applications. UML has come to prominence due to the Object Management Group, an organization that has been developing OOP models, technology, and standards since the 1980s. It is also beginning to see widespread use among OOP practitioners. In this appendix, we are primarily interested in understanding the UML notation, and how it relates to IE Crow's Foot notation.

Why Should I Learn to Use UML?

Because it is an application development methodology, UML is a subject for a course on systems development and is of limited concern to us. However, some UML diagrams are used to model database designs, and you may encounter them during your career. Accordingly, you should be familiar with their style.

What Will This Appendix Teach Me?

In this appendix you will learn the differences between the extended E-R model and UML-style E-R diagrams, and how to use UML to create data models.

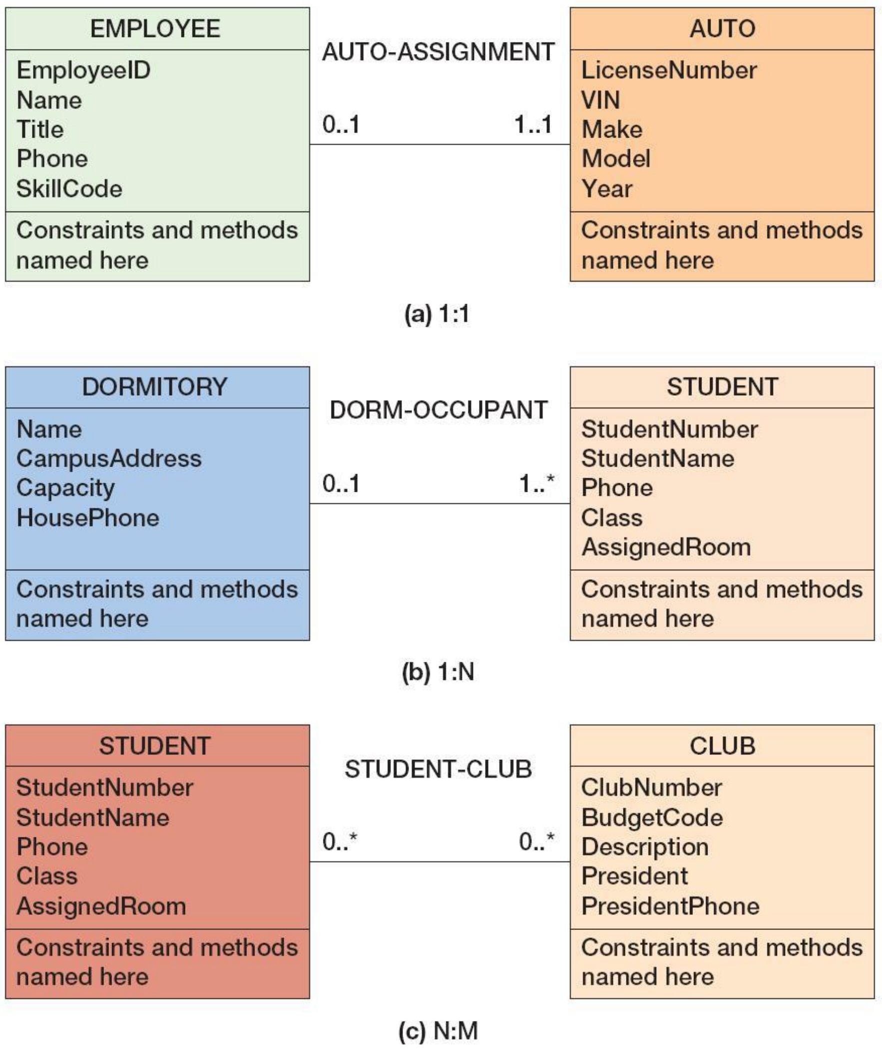


Figure D-1: UML Representation of HAS-A Relationships

How Does UML Represent Entities and Relationships?

Figure D-1 shows the UML representation of a 1:1, a 1:N, and an N:M **HAS-A relationship**. Each entity is represented by an **entity class**, which is shown as a rectangle with three segments. The top segment shows the name of the entity and other data that we will discuss. The second segment lists the names of the attributes in the entity, and the third documents constraints and lists methods (program procedures) that belong to the entity.

Relationships are shown with a line between two entities. Cardinalities are represented in the **x..y cardinality format**, where *x* is the minimum required and *y* is the maximum allowed. Thus, 0..1 means that no entity is required and that at most one is allowed. An asterisk represents an unlimited number. Thus, 1..* means that one is required and an unlimited number is allowed.

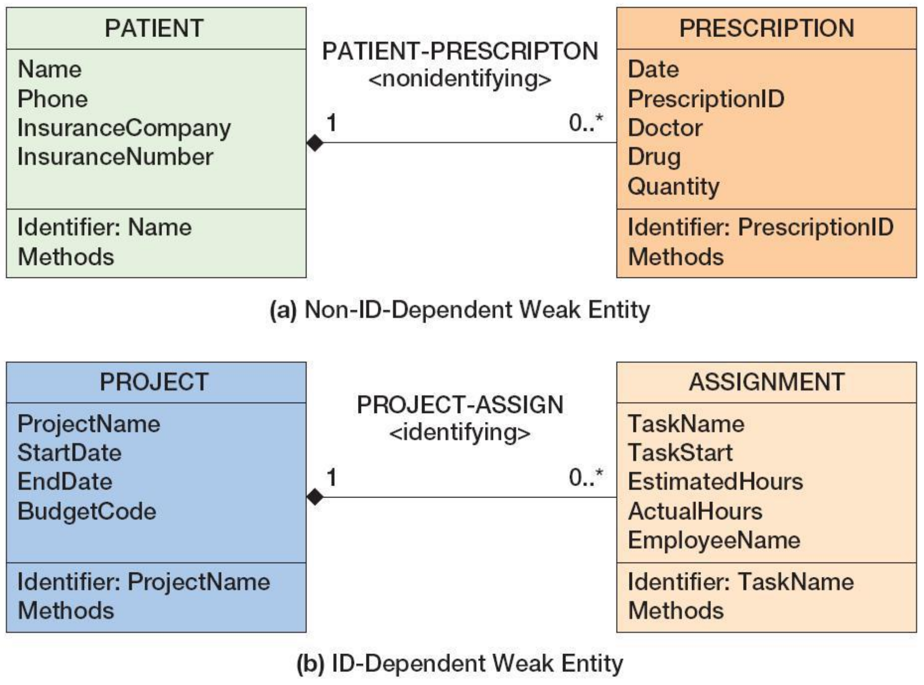


Figure D-2: UML Representation of Weak Entities

Representation of Weak Entities

Figure D-2 shows the UML representation of weak entities. A filled-in diamond is placed on the line to the parent of the weak entity (the entity on which the weak entity depends). In Figure D-2(a), PRESCRIPTION is the weak entity and PATIENT is the parent entity. All weak entities have a parent, so the cardinality on their side of the weak relationship is always 1..1. Because of this, the cardinality on the parent entity is shown simply as 1.

Figure D-2(a) shows a weak entity that is not an ID-dependent entity. It is denoted by the expression **nonidentifying** on the PATIENT-PRESCRIPTION relationship. Figure D-2(b) shows a weak entity that is ID-dependent. It is denoted with the label **identifying**.

Representation of Subtypes

UML represents subtypes and **IS-A relationships**, as shown in Figure D-3. In this figure, INDIVIDUAL, PARTNERSHIP, and CORPORATE subtypes of CLIENT are allowed. According to this figure, a given CLIENT could be one, two, or three of these subtypes. This does not make sense for this particular situation; a CLIENT should be one and only one of these types. The current version of UML does not provide a means to document exclusivity. Such notation can be added to a UML diagram, however.

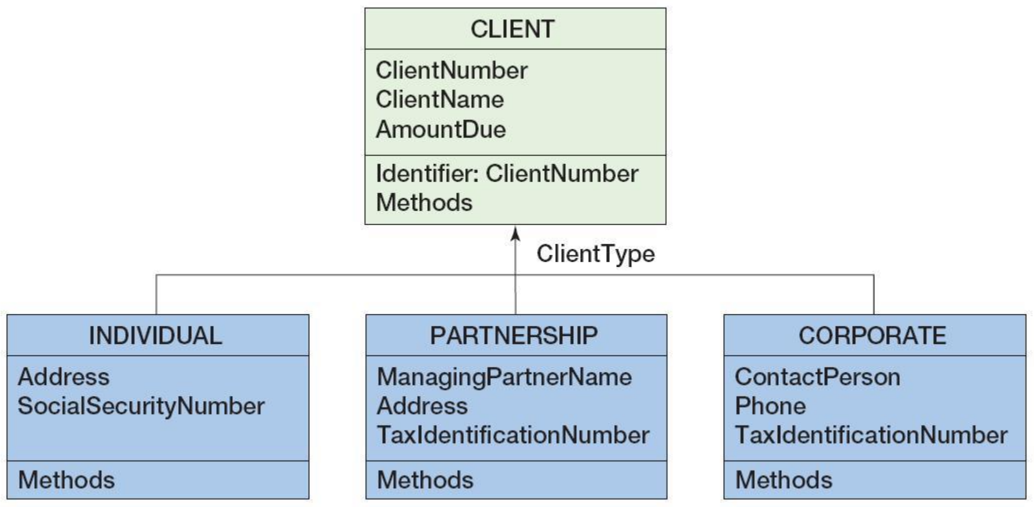


Figure D-3: UML Representation of HAS-A Relationships

What OOP Constructs Are Introduced by UML?

Because UML is an object-oriented technology, several OOP constructs have been added to UML entity classes. We will touch on these ideas here; you will learn more about them when you study object-oriented systems development. Consider the UML diagram in Figure D-4, which shows an E-R diagram with the full UML decoration.

First, the classes of all entities that are to be stored in the database are labeled with the keyword **<<Persistent>>**. This simply means that data should continue to exist even if the object that processes it is destroyed. In simpler terms, it means that the entity class is to be stored in the database.

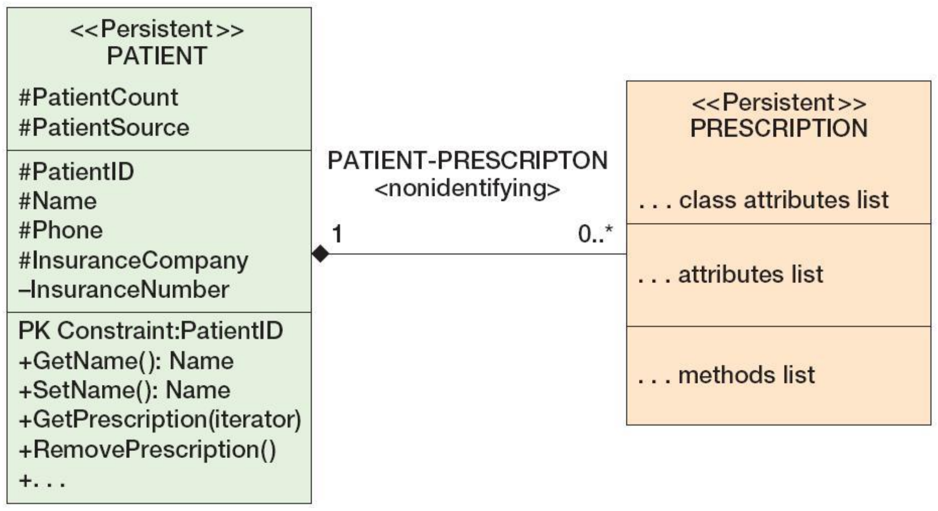


Figure D-4: UML Representation of HAS-A Relationships

Next, UML entity classes allow for **class attributes**. Such attributes differ from entity attributes because they pertain to the class of all entities of a given type. Thus, in Figure D-4 PatientCount of PATIENT is an attribute of the collection of all PATIENTs in the database. PatientSource is an attribute that documents the source of all of the PATIENTs in the database.

As you will learn, such class attributes have no place to reside when using the relational model. Instead, in some cases, attributes such as PatientCount are not stored in the database but are computed at run time. In other cases, a new entity is introduced to contain the class attributes. For the entity in Figure D-4, a new entity called PATIENT-SOURCE could be defined to hold both PatientCount and PatientSource attributes. In this case, all of the entities in PATIENT are connected to PATIENT-SOURCE.

A third new feature is that UML uses object-oriented notation for the visibility of attributes and methods. Attributes preceded by a + are public, those with a # are protected, and those with a – are private. Thus, in Figure D-4 Name in PATIENT is a protected attribute. These terms arise from the discipline of object-oriented programming. A **public** attribute can be accessed and changed by any method of any object. A public method can be invoked by any method of any object. **Protected** means that the attribute or method is accessible only by methods of this class or of its subclasses, and **private** means that the attribute or method is accessible only by methods of this class.

Finally, UML entities specify constraints and methods in the third segment of the entity classes. In Figure D-4, a primary key constraint is placed on PatientID. This simply means that PatientID is a unique identifier. Additionally, Figure D-4 documents that GetName() is to be created to provide public access (note the + in front of GetName) to the Name attribute; SetName() is to be used to set its value; and the method GetPrescription() can be used to iterate over the set of Prescription entities related to this PATIENT entity.

What Is the Role of UML in Database Processing Today?

The ideas illustrated in Figure D-4 lie in the murky water where database processing and object-oriented thinking merge. Such object-oriented notation does not fit with the practices and procedures of commercial database processing today. The notion that an entity attribute can be hidden in an object does not make sense unless only object-oriented programs are processing the database; even then, those programs must process the data in conformance with that policy. This is almost never done.

Instead, most commercial DBMS products have features that allow all types of programs to access the database and process any data that they have permission to access. Moreover, with facilities such as SQL, there is no way to limit access to attribute values to a single object.

So, the bottom line is that you should know how to interpret UML-style E-R diagrams. They can be used for database design just as extended E-R diagrams can. At present, however, the object-oriented notation they introduce is of limited practical value to database practitioners. The extended E-R model using the crow's foot notation is far more common and useful for database (as opposed to OO) design.

Key Terms

<<persistent>>	**<identifying>**
<nonidentifying>	**class attributes**
entity class	**HAS-A relationship**
IS-A relationship	**private**
protected	**public**
Unified Modeling Language (UML)	**x..y cardinality format**

Review Questions

D.1 Why is UML important? Why is it of concern to database designers?

D.2 Show a 1:1 relationship in UML format.

D.3 Show a 1:N relationship in UML format.

D.4 Show an N:M relationship in UML format.

D.5 Explain how UML documents minimum cardinality.

D.6 Show identifying and nonidentifying weak entities in UML format.

D.7 Show subtypes in UML format.

D.8 What are class attributes? How are they documented in UML?

D.9 How would class attributes be represented in the extended E-R model described in Chapter 5?

D.10 Explain the significance of the +, #, and – signs in a UML diagram.

D.11 Give an example of a constraint on an entity in a UML diagram.

D.12 Redraw the E-R diagram in Figure 5-52 using UML.

D.13 Describe the ways in which UML and commercial database processing are misfits. How do you think this situation will be resolved?

D.14 Answer Review Question 5.56, but use UML instead of the crow's foot model.

D.15 Answer Review Question 5.59, but use UML instead of the crow's foot model. Note that this question asks for exclusive subtypes, which are not supported in UML – explain how you will handle this problem (*Hint:* you may use application logic if the database design itself is not sufficient). Redraw Figure 5-57 using UML notation.