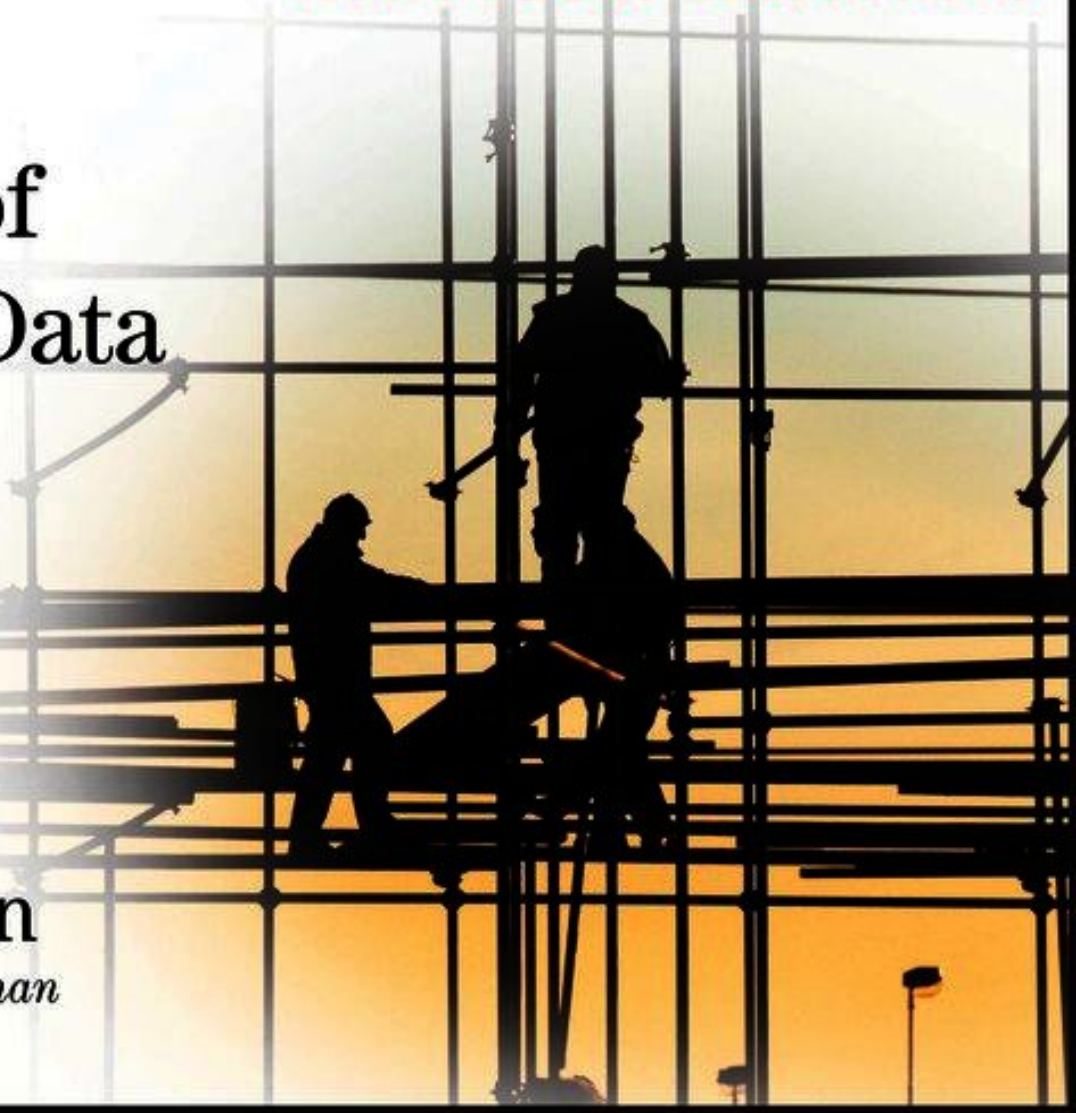
WILEY

The Data Model Resource Book

Revised Edition

Volume 2

A Library of Universal Data Models by Industry Types



Len Silverston

Foreword by John Zachman

Table of Contents

Title Page

Copyright

Dedication

Advance Praise for The Data Model Resource Book, Volume 3

Credits

Foreword

Acknowledgements

About the Authors

Chapter 1: Introduction

- *Why Is There a Need for This Book?*
- *Extending the Discipline of Data Modeling*
- *What Is a Pattern and What Is a Universal Pattern?*
- *What Is the Significance of Patterns?*
- *Approach of This Book*
- *The Different Pattern Levels*
- *Who Is the Intended Audience for This Book?*

What Is in This Book
Other Patterns for Data Modeling
Conventions and Standards Used in This Book

Chapter 2: Setting Up Roles: What Parties Do

What Is the Significance of This Type of Pattern?
What Is in This Chapter?
What Is a Declarative Role?
Level 1 Declarative Role Pattern
Level 2 Declarative Role Pattern
Level 3 Declarative Role Pattern

Chapter 3: Using Roles: How Parties Are Involved

What Is the Significance of This Type of Pattern?
What Is in This Chapter?
What Is a Contextual Role?
Level 1 Contextual Role Pattern, Attributes
Level 1 Contextual Role Pattern, Relationships
Level 2 Contextual Role Pattern
Level 2 Contextual Role Pattern, PARTY Only Alternative
Level 3 Contextual Role Pattern
Hybrid Contextual Role Pattern

Chapter 4: Hierarchies, Aggregations, and Peer-to-Peer Relationships: The Organization of Similar Data

- What Is the Significance of This Type of Pattern?
- What Is in This Chapter?
- What Is a Recursive Relationship and How Is Data Organized by Recursive Relationships?
- Level 1 Recursive Pattern
- Level 2 Recursive Pattern
- Level 2 Expanded Recursive Pattern
- Level 3 Recursive Pattern
- Level 3 Recursive Pattern with Rules

Chapter 5: Types and Categories: the Classification of Data

- What Is the Significance of This Type of Pattern?
- What Is in This Chapter?
- What Are Types, Categorizations, and Taxonomies?
- Level 1 Classification Pattern
- Level 2 Classification Pattern
- Level 3 Classification Pattern
- Level 3 Classification Pattern with Rollups and Schemes

Chapter 6: Status: The States of Data

What Is the Significance of This Type of Pattern?
What Is in This Chapter?
What Is a Status?
Level 1 Status Pattern
Level 2 Status Pattern, Current Status
Level 3 Status Pattern
Level 4 Status Pattern
Status Category Pattern
Status Type with Multi Rollup and Rules Pattern

Chapter 7: Contact Mechanisms: How to Get in Touch

What Is the Significance of This Type of Pattern?
What Is in This Chapter?
What Is a Contact Mechanism?
Level 1 Contact Mechanism Pattern
Level 2 Contact Mechanism Pattern
Level 3 Contact Mechanism Pattern
Level 4 Contact Mechanism Pattern
Contact Mechanism Pattern with Geographic Boundary
Contact Mechanism with Flexible Address Parts Pattern
Other Common Contact Mechanism Data

Chapter 8: Business Rules: How Things Should Work

- What Is the Significance of This Type of Pattern?
- What Is in This Chapter?
- What Is a Business Rule?
- Level 2 Business Rules Pattern
- Level 3 Business Rules Pattern
- Business Rules with Party Roles

Chapter 9: Using the Patterns

- What Is in This Chapter?
- The Scenario
- Prototype Models, Scope Statements
- Application Data Models
- Enterprise Data Models
- Data Warehouse Models
- Master Data Management
- Other Thoughts Regarding Using the Patterns

Chapter 10: Socializing the Patterns

- What Is the Significance of Socializing the Patterns?
- What Is in This Chapter?
- Experiences Using and Socializing These Patterns
- What Makes the Difference In Socializing the Patterns?

Understanding Motivations—Why Would Someone Use or Not Use the Patterns?
Creating a Clear, Common, Compelling Purpose and Vision for Using the Patterns
Developing Trust so People Can Rely on the Patterns
Managing Resistance and/or Conflict Regarding Patterns
Other Comments about Socializing the Patterns

Index



The Data Model Resource Book, Volume 3

Universal Patterns for Data Modeling

Len Silverston
Paul Agnew

WILEY

Wiley Publishing, Inc.

The Data Model Resource Book, Volume 3: Universal Patterns for Data Modeling

Published by

Wiley Publishing, Inc.

10475 Crosspoint Boulevard

Indianapolis, IN 46256

www.wiley.com

Copyright © 2009 by Len Silverston and Paul Agnew.

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-17845-4

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Legal Department, Wiley Publishing, Inc., 10475 Crosspoint Blvd., Indianapolis, IN 46256, (317) 572-3447, fax (317) 572-4355, or online at http://www.wiley.com/go/permissions.

Limit of Liability/Disclaimer of Warranty: The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies

contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or Website is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or Web site may provide or recommendations it may make. Further, readers should be aware that Internet Web sites listed in this work may have changed or disappeared between when this work was written and when it is read.

Library of Congress Cataloging-in-Publication Data is available from the publisher.

For general information on our other products and services please contact our Customer Care Department within the United States at (800) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Trademarks: Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. Universal Data Models is a registered trademark of Universal Data Models, LLC. All other trademarks are the property of their respective owners. Wiley Publishing, Inc. is not associated with any product or vendor mentioned in this book.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

To my amazing and loving wife, Annette, and my wonderful daughters, Danielle and Michaela

—Len Silverston

To my mother, Breda, and in loving memory of my father, Tom

—Paul Agnew

Advance Praise for The Data Model Resource Book, Volume 3

Len and Paul look beneath the superficial issues of data modeling and have produced a work that is a must for every serious designer and manager of an IT project.

Bill Inmon

World-renowned expert, speaker, and author on data warehousing and widely recognized as the "father of data warehousing"

The Data Model Resource Book, Volume 3: Universal Patterns for Data Modeling is a great source for reusable patterns you can use to save a tremendous amount of time, effort, and cost on any data modeling effort. Len Silverston and Paul Agnew have provided an indispensable reference of very high-quality patterns for the most foundational types of data model structures. This book represents a revolutionary leap in moving the data modeling profession forward.

Ron Powell

Cofounder and Editorial Director of the Business Intelligence Network

After we model a Customer, Product, or Order, there is still more about each of these that remains to be captured, such as roles they play, classifications in which they belong, or states in which they change. *The Data Model Resource Book, Volume 3: Universal Patterns for Data Modeling* clearly illustrates these common structures. Len Silverston and Paul Agnew have created a valuable addition to our field, allowing us to improve the consistency and quality of our models by leveraging the many common structures within this text.

Steve Hoberman

Best-Selling Author of "Data Modeling Made Simple"

> The large national health insurance company I work at has actively used these data patterns and the (Universal Data Models) UDM, ahead of this book, through Len Silverston's UDM Jump Start engagement. The patterns have found their way into the core of our Enterprise Information Model, our data warehouse designs, and progressively into key business function databases. We are getting to reuse the patterns across projects and are reaping benefits in understanding, flexibility, and time-to-market. Thanks so much.

David Chasteen

Enterprise Information Architect

> Reusing proven data modeling design patterns means exactly that. Data models become stable, but remain very flexible to accommodate changes.

We have had the fortune of having Len and Paul share the patterns that are described in this book via our engagements with Universal Data Models, LLC. These data modeling design patterns have helped us to focus on the essential business issues because we have leveraged these reusable building blocks for many of the standard design problems. These design patterns have also helped us to evaluate the quality of data models for their intended purpose. Many times there are a lot of enhancements required. Too often the very specialized business-oriented data model is also implemented physically. This may have significant drawbacks to flexibility. I'm looking forward to increasing the data modeling design pattern competence within Nokia with the help of this book.

Teemu Mattelmaki

Chief Information Architect, Nokia

Once again, Len Silverston, this time together with Paul Agnew, has made a valuable contribution to the body of knowledge about data models, and the act of building sound data models. As a professional data modeler, and teacher of data modeling for almost three decades, I have always been aware that I had developed some familiar mental “patterns” which I acquired very early in my data modeling experience. When teaching data modeling, we use relatively simple workshops, but they are carefully designed so the students will see and acquire a lot of these basic “patterns”—templates that they will recognize and can use to interpret different subject matter into data model form quickly and easily. I've always used these patterns in the course of facilitating data modeling sessions; I was able to recognize “*Ah, this is just like*...,” and quickly apply a pattern that I'd seen before. But, in all this time, I've never sat down and clearly categorized and documented what each of these “patterns” *actually was* in such a way that they could be easily and clearly communicated to others; *Len and Paul have done exactly that*. As in the other *Data Model Resource Book*s, the thinking and writing is extraordinarily clear and understandable. I personally would have been very proud to have authored this book, and I sincerely applaud Len and Paul for another great contribution to the art and science of data modeling. It will be of great value to any data modeler.

William G. Smith

President, William G. Smith & Associates,
www.williamgsmith.com

Len Silverston and Paul Agnew's book, *Universal Patterns for Data Modeling*, is essential reading for anyone undertaking commercial data modeling. With this latest volume that compiles and insightfully describes fundamental, universal data patterns, *The Data Model*

Resource Book series represents the most important contribution to the data modeling discipline in the last decade.

Dr. Graeme Simsion

Author of "Data Modeling Essentials" and "Data Modeling Theory and Practice"

Volume 3 of this trilogy is a most welcome addition to Len Silverston's two previous books in this area. Guidance has existed for some time for those who desire to use pattern-based analysis to jump-start their data modeling efforts. Guidance exists for those who want to use generalized and industry-specific data constructs to leverage their efforts. What has been missing is guidance to those of us needing guidance to complete the roughly one-third of data models that are not generalized or industry-specific. This is where the magic of individual organizational strategies must manifest itself, and Len and Paul have done so clearly and articulately in a manner that complements the first two volumes of *The Data Model Resource Book*. By adding this book to Volumes 1 and 2 you will be gaining access to some of the most integrated data modeling guidance available on the planet.

Dr. Peter Aiken

Author of "XML in Data Management" and data management industry leader VCU/Data Blueprint

Credits

Executive Editor
Robert Elliott

Senior Development Editor
Kevin Kent

Technical Editor
Ed Landale

Development Editor
William Bridges

Production Editor
Eric Charbonneau

Copy Editor
Kim Cofer

Editorial Manager
Mary Beth Wakefield

Production Manager
Tim Tate

Vice President and Executive Group Publisher
Richard Swadley

Vice President and Executive Publisher
Joseph B. Wikert

Project Coordinator, Cover
Lynsey Stanford

Proofreader
Publication Services, Inc.

Indexer
Johnna VanHoose Dinse

Cover Image
© Image Source/Jupiter Images

Foreword

When we were younger, my brother and I loved to take apart gadgets to see what made them tick. My grandmother would buy used clocks, radios, and other electronic devices so that we could take a hammer to them, bashing them to bits to see what was inside and how they worked. One of the things we noticed was that even though they were different on the outside, most seemed to have the same parts as other clocks. In fact, once we'd removed the outer covers and taken everything apart, we could no longer tell which part came from which clock, but we could sort all the pieces into similar parts. Cogs, wheels, and springs were sorted into piles of similar shape. If we'd had enough time and will, we probably could have built a new clock out of these components.

I remember asking why these parts looked so similar and why some of them even had the same numbers on them. In fact, some clocks had the same parts that radios did. My grandfather explained to me that it was cheaper and easier for companies to build their products if they could use similar parts. It also made it easier for the builders and fixers to work with the same parts. He showed me how he replaced a component of a radio with a new part to fix it. He was able to do this because the parts followed similar patterns. I thought this was brilliant.

I am delighted to write this foreword for what I believe is the most important volume of the *Universal Data Model* book series. *The Data Model Resource Book, Volume 3: Universal Patterns for Data Modeling* presents highly reusable patterns that could apply to thousands of industries, thousands of projects, and an infinite number of use cases. While the first two volumes focused on template

solutions for common data structures, this one is focused on much more general, fundamental, underlying patterns in data. These aren't industry or functional patterns; they are the cogs and wheels that could fit into any solution. You can create your own parts to make a “clock” for your current project and use those same parts to create other solutions in other projects.

In developing and documenting these patterns, Len Silverston and Paul Agnew have provided to you a set of tools for your entire career. No matter where you work or what business you support, these patterns apply.

All mature professions have identified components of their practices that are highly reusable. Engineers have building standards and patterns and medical professionals have standards of practice. As an emerging profession, Information Technology is still forming and testing patterns for use across many situations. *Universal Patterns for Data Modeling* enables data professionals to raise our practice to the professional level. We can then focus our efforts on those decisions that require tailored solutions.

Consistent use of universal patterns for data modeling frees up team members to focus their efforts on implementing solutions to those business problems that provide competitive advantage, deliver faster services, and reduce costs. Most importantly, it enables users of the models to work faster. Developers who have seen a similar *status* structure many times can quickly tailor their own patterns to make use of it. Test plans and test data can be quickly tailored to support new types of statuses. These economies will be seen by all team members, across many projects.

The authors have provided several levels of generalization for each pattern and it is up to you, as a seasoned professional, to choose the one that makes sense for the costs, benefits, and risks of your designs. I'd like for you to

approach these patterns with a mind toward how they might best fit your current project's context. Every design decision comes down to cost, benefit, and risk, and these are laid out for you for each level. You get to choose which level applies and what the benefits will be. There is no right answer or right pattern for every project, business, and organization, but you will know why your chosen solution is right for your specific design.

As I think back to my childhood and the cogs and wheels of the many clocks we dismantled, the lessons we learned about patterns was one of the most important ones I carried into my professional life. Len and Paul have done the tinkering and sorting of these patterns for you. Your next step is to apply them on your projects so that you can deliver greater business value by saving time, reducing costs, and increasing the quality of your models.

Karen Lopez

Industry thought leader

InfoAdvisors

Acknowledgements

We feel that universal patterns for data models are a significant contribution to the field of data modeling. However, this book would not have been possible without the insights and interaction of our clients and other advocates that have helped to challenge our thinking and advance these patterns. We feel strongly that the relationships that we have with our clients are a mutually beneficial learning experience. As we impart our knowledge of Universal Patterns for Data Modeling and Universal Data Models™, we learn from our clients about the needs and wants of their enterprises. This has been an invaluable input in the evolution of the universal patterns for data models and what you will find in this book. We are extraordinarily grateful to all of our clients, partners, seminar participants, and all those who have provided input and insights, thus helping to advance these patterns. We feel so appreciative that many of these people have become friends and partners with whom we have shared rich experiences.

From among the many people who have contributed to helping us promote, use, and evolve universal models and patterns, we want to thank Aidan Doyle, Ajia Palomaki, Alireza Hasanpour, Andre Boeder, Andy Pozsol, Bongsoo Chong, Cesar Estrada, Chris Nickerson, Craig Rapley, Dan Adler, David Chasteen, Ed Smith, Greg Sorum, Herman Koester, Jagannadha Ghanta and Jan-Erik Osterberg, John Poonnen, John Yelle, Karen Vitone, Ken Bates, Kevin Morris, Kristiina Lammila, Leyla Akgez-Laakso, Lynn Crabb, Marlene Mandt, Mary Mink, Michael Jansen, Milja Karppelin, Radha Krishnan, Ray Serrano, Regina Pieper, Randy Carlson, Robert Hooks, Ron Powell, Rupali Anjaria, Satoshi Matsumoto, Tarja Martti, Ted Kowalski, Teemu Mattelmaki, Tero Leskinen,

Trevor Prusco, Truett Phillips, Vinnie Chintappaly, Vinod Badami, Wes Bennet, and Yang-Young Zhang. This is only a partial list of the many who have contributed to the promotion and advancement of universal models and patterns, and if we have forgotten to mention anyone specifically, we offer our sincerest apologies. We want you to know that your efforts are appreciated. We want to thank the business partners of Universal Data Models, LLC, who have helped to promote the ongoing usage of Universal Data Models and in particular Greg Keller, Josh Howard, Jason Tiret, and Kimber Spradin from our partner Embarcadero Technologies, as well as Ken Hoang and many others from our partner Siperian.

We want to thank all our colleagues in data modeling who have helped to advance this field, and we specifically want to recognize Dr. Graeme Simsion, William G. Smith, and Steve Hoberman, who have helped us on this book and have been great supporters of this work.

We are very thankful to the people who have added to the content to this book. This book would not have been possible without the great contributions for our technical editor Ed Landale. He took time from his busy schedule to scrupulously review every model and every word of this book. His insights and suggestions into each pattern provided valuable feedback and improved the quality of this book significantly. We also thank him for his patience and good humor throughout this whole process. We appreciate the assistance and advice that we received from Karen Lopez. In particular we appreciate her invaluable input on ‘generalization’ as well as specific recommendations she provided to enhance and change some of the data modeling patterns. Karen also helped us to focus on the ‘practical’ nature of the patterns.

There were mentors who helped to make this work possible. Len is extremely grateful to Bill Inmon, who helped

him break into the field of writing and who has been an amazing inspiration as both an industry leader and as a humane person who has helped his career tremendously. He also wants to express huge appreciation to Paul for his amazingly great attitude and contribution throughout this project and throughout the relationship. Paul is grateful to Len as a guide and a mentor and for being a great partner.

We feel honored to have been able to work on this book with Bob Elliot and Kevin Kent at John Wiley & Sons, Inc. We appreciate the vision, management, editing, and support for this book as well as their ongoing encouragement. We want to thank Eric Charbonneau for his help in producing this book also.

From Len Silverston: I am thankful to my wife, Annette Quintana, for being the best life partner I can imagine, for supporting me, and for putting up with the long hours over numerous years to create this book as well as my other books. I want to thank my beautiful daughters, Danielle and Michaela, who are the most amazing gems in my life and who have also supported me on this effort. I am so appreciative to my mom, Dede, and my family and friends including (but not limited to) Steve, Betty, Phil, Janet, Joe, Vicki, LR, Melinda, Les, Leila, and Floyd. Special thanks to my dad, who passed away a few years ago and who inspires me to be caring, decent, and loyal.

From Paul Agnew: I am thankful to my mother and father for all of their guidance throughout my life. Without their support and love I would never have been in a position to complete this book. I wish to thank my brothers (Robert, Tommy, Gerard, Ciaran, Fergus, Declan, and Terry) and my sister (Brenda) for their support over the years. I also want thank my sisters-in-law, brother-in-law, nieces, and nephews. A close family makes things easier. Many of my friends also supported me throughout the writing of this book. Thanks for letting me use your names as examples.

Finally, I am very fortunate to have the support and love of my partner, Neena; there is no way that this book would have been finished without your support. This book is as much yours as mine.

About the Authors

Len Silverston is the best-selling author of *The Data Model Resource Book* series (Volumes 1 and 2) and a speaker and consultant with more than 25 years of experience helping organizations integrate their information and systems. He is regarded as one of the most sought-after experts in data modeling and data integration and is a pioneer in the industry by virtue of publishing and distributing best practice reusable data models that have helped people and organizations develop high-quality data models in very short amounts of time.

Mr. Silverston has published many articles and spoken extensively worldwide as an instructor and as a keynote and an invited speaker on topics such as reusable data models, universal patterns, data integration, and power and politics in data management. He has published hundreds of holistic, reusable data models in his books and articles. His book, *The Data Model Resource Book*, Volume 1, was rated #12 on the Computer Literacy Best Seller List and *The Data Model Resource Book*, Volume 2, which provides universal data models for various industries, has been translated into Chinese. His books and products have been adopted and used globally as a standard by a great number of large and small businesses and government enterprises and by universities as a course text.

Due to his significant, demonstrable contributions to advancing the data management field, he is the winner of the (The Data Management Association) DAMA International Professional Achievement Award for 2004 and the DAMA Community Award for 2006. Mr. Silverston's company, Universal Data Models, LLC, provides consulting, training, publications, and software regarding reusable data models

and data management strategies to help integrate information, systems, and people. Mr. Silverston received his B.S. from SUNY Binghamton and M.S. from Renssellaer Polytechnic Institute.

He can be reached at lsilverston@univdata.com.

Paul Agnew is an author and consultant with more than 17 years of experience in the data management field. He has worked in many industries as an expert in data architecture and data integration, including investment banking firms on Wall Street, telecommunications, insurance, and engineering. In the last 8 years Len Silverston and he have worked together helping many of the top Fortune 500 companies around the world build and integrate information systems using Universal Patterns for Data Modeling, and Universal Data Models.

Mr. Agnew has many years of practical experience working in the data integration and data management fields. He has worked as a database administrator and database developer. He was also a speaker at DAMA International (The Data Management Association) and DAMA Finland.

He is a partner in Universal Data Models, LLC (www.universaldatamodels.com), located in Denver, Colorado, and New York City, providing consulting and training to help enterprises customize and implement Universal Data Models and Universal Patterns for Data Modeling. The company offers many tools to deliver high-quality information systems in a short span of time.

Mr. Agnew was born in Ireland, but has lived in New York City with his partner, Neena, for the past 14 years. He graduated from Dublin Institute of Technology, Kevin Street.

He can be reached at pagnew@univdata.com or pauljagnew@yahoo.com.

Chapter 1

Introduction

Why Is There a Need for This Book?

Based upon our consulting experiences, many companies still develop their data models with very little outside reference materials. There is a large cost associated with either hiring experienced consultants or using internal staff to develop this critical component of the system design. Often there is a need for more objective reference material that an organization can use to test its data models and database designs or from which it can seek alternate options for data models or database structures. This book substantially extends the tools offered in the current *Data Model Resource Book*, Volumes 1 and 2 (Wiley, 2001), providing a comprehensive guide for companies to develop data models with higher quality in a shorter amount of time.

Volume 1 of *The Data Model Resource Book* answered the question "Where can we find a book showing a standard way to model common data model structures?" It provides an extensive library of template data models for common data areas such as people and organizations, products, orders, shipments, invoicing, accounting and budgeting, human resources, and so on. It also provides template models for data warehouse models for sales analysis, human resources, and inventory management analysis among many others.

Volume 2 of *The Data Model Resource Book* continued in the same vein as Volume 1 by extending these template data models and by adding additional data model constructs applicable specifically for certain industries such as manufacturing, telecommunications, health care, insurance, financial services, professional services, travel, and retail e-commerce industries.

Although people and organizations have improved the quality of their data models and saved a great deal of time and effort using the first two books in this series, a question has continued to come up as we have implemented these models. “How can we quickly extend and customize these models for *our* organization and *our* needs to quickly develop *any* data model with higher quality, even if it is specific to *our* enterprise?” Also, many organizations want to adhere to a standard way of creating common data structures. They often say, “We can't be the first people to ask how to extend our data models and/or use the same ideas to construct new types of models. Surely this has been done before.” Volume 3 of the model resource book addresses these questions and concerns. This book looks under the cover of the previous books and examines the common underlying structures that are applicable to all data models.

We have a useful rule of thumb that seems to apply to most data models: One-third of a data model usually consists of common constructs that are applicable to most organizations, one-third of the data model is usually industry-specific, and one-third of the model is specific to an organization. Volume 1 and Volume 2 of *The Data Model Resource Book* address, for the most part, the first and second “thirds” of that rule. What we have also found in our experience with decades of data modeling is that there are very common patterns that apply to well *over 50 percent of most data model constructs* and that can be reused. For

example, a status for an order works in the same way as a status for a person or organization. The classification of product or person follows the same pattern, regardless of the fact that one classification deals with products and the other is about people.

One benefit of this book is that it explains and enhances the underlying patterns that are used in Volumes 1 and 2 of *The Data Model Resource Book*. Yet it goes beyond this because the data model patterns illustrated in this book apply to the common constructs that are applicable to all enterprises, industry-specific data model constructs, and any model constructs specific to an enterprise. This book provides templates that can be used to quickly and consistently model many types of data requirements by reusing these universal data model patterns. This can then have a huge positive impact to help integrate data, share data, and use data as a valuable strategic asset.

The difference between Universal Data Models and Universal Patterns for Data Modeling is that the Universal Data Models apply to very common models, whereas the Universal Patterns can be used to extend and develop just about any type of data model. One way to think of this is in terms of furniture; first think of the design for a dovetail joint, an interlocking technique used to make all sorts of furniture (this is akin to a Universal Pattern), then think of the design for a full set of table and chairs (this is akin to a Universal Data Model), and you have an idea of how the concepts relate. Many of the Universal Data Models are based on Universal Patterns. The first two books provided concrete examples of very common data models that can be reused such as models for shipments, orders, invoices, and so on. In contrast, the Universal Patterns for Data Modeling provide the underlying structural building blocks so that the modelers can reuse these to build any model, even ones that are very unique!

The patterns can be used to quickly develop and/or modify both common models and industry models or to develop brand new models. Each pattern has a real-life example of how to implement it. Any organization can use the Universal Patterns found in this book as a guideline and a set of standards to which their data models can adhere to improve consistency, to save a great deal of time on development and maintenance, and to increase the quality of their models. A data professional in any enterprise can use the template models from Volume 1 or Volume 2 as a data modeling jump-start and then use the patterns in Volume 3 to build upon these common models in a consistent fashion, with the confidence of knowing that the patterns are true and tested common constructs that work in real life. Many of our clients have used these patterns in many different ways, for example:

- To provide a standard that IT professionals can adhere to when modeling data. This has helped them keep a consistent style for data models and subsequent data structures across different databases in their organization.
- As a standard toolkit that data professionals can turn to when building/extending their data models. The patterns cover many of the standard problems that data modelers need to address. Why solve the problem again when the patterns already give you different flavors (levels of generalization) of the solutions, thus providing effective alternatives with their pros and cons?
- As a standard that can be used as the basis for common database structures that allows developers or programmers to create standard interfaces to and from these common structures that are based on the patterns. This means that programmers can “program to the interface” and have less concern about dealing

with many different underlying data semantics and data structures.

- As a useful tool for clients when they buy software or other standard data models. The patterns can set the data requirements that a vendor data model or database must rise to regarding very common data needs. For example, the patterns can specify that a solution needs to support very flexible classification schemes that allow new types of classifications without changing the data model or data structure; does the product you are looking at accommodate this type of flexibility regarding maintaining classifications? Does it use flexible patterns? If it does not, how does it provide the appropriate level of flexibility that you need?
- As an objective source against which an enterprise can evaluate and check its data models from its previous systems development efforts so it can evaluate alternative options.
- As training materials for their data professionals and IT staff in general. The patterns cover a broad range of different structures at different levels of generalization. The patterns are explained in detail with examples that can guide data modelers and other IT professionals in their use.

Many of our clients have used these patterns successfully to save time and increase the quality for a great variety of data modeling efforts, ranging from creating a data model for a prototype, through developing an enterprise-wide data model used to standardize their models worldwide.

Extending the Discipline of Data Modeling

Data modeling has been a discipline that first gained recognition in Dr. Peter Chen's 1976 article that illustrated his approach for describing data structures called Entity-Relationship Modeling.(1) Since then it has become the standard approach used toward modeling and designing databases. By properly modeling an organization's data, the database designer can eliminate data redundancies, which are a key source of inaccurate information and ineffective systems.

There are many books and articles about design patterns, but very little has been written about the underlying patterns for entity relationship modeling (as we are describing in this book). It can be said that the fathers/mothers of patterns were Christopher Alexander, Sarah Ishikawa, and Murray Silverstein, when they wrote *A Pattern Language: Towns, Buildings, Construction*.(2) This is a book about architecture with many patterns that are collected and used as a basis to create solutions for construction problems and town planning. Many programmers liked the concepts in this book and how they simplified the process of creating reusable code. Another seminal piece of work called *Design Patterns: Elements of Reusable Object-Oriented Software* written by the “Gang of Four” (Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides)(3) addressed the common solutions for solving programming problems by using common interfaces.

The first two volumes of *The Data Model Resource Book*(4) and David Hay's excellent book *Data Model Patterns*(5) contain reusable data models for very common data modeling requirements such as how to model data about parties, products, orders/contracts, and so on. Many people think of these books as providing “patterns” in the context of data modeling; however, those books discuss something very different than what is contained in this particular book. In this book, we have a very different meaning when we use

the term *pattern*. We make a distinction between a reusable model for a specific application (reusable models that are covered in these other books) and the underlying, core templates that are independent of any particular application and that we refer to as Universal Patterns for Data Modeling, which are the focus of this book. Although many standards exist for data modeling, we need to take data modeling to the next level: providing accessibility to libraries of universal data patterns with examples in a convenient format so that they can be reused. That is the intention of this book.

What Is a Pattern and What Is a Universal Pattern?

In general, a pattern is “something intended as a guide for making something else.”(6) A pattern in data modeling can be described as a template that can serve as a guide for developing data models. For example, the status patterns in Chapter 6 provide guides or templates for modeling the statuses for any type of entity. Thus, the status patterns may apply to the status of a PARTY, PRODUCT, ORDER, INVOICE, or any other entity that has various states. A PARTY may have states or statuses of “Active,” “Inactive,” and so on, and a PRODUCT may have statuses of “Introduced,” “Support discontinued,” and so on. Both of these entities could use the same “pattern” to model their states.

The word *universal* is defined in Webster's dictionary as “applying to a great variety of uses: comprehending, affecting or extending to the whole.” Thus Universal Patterns for Data Modeling are reusable guides that provide a data modeling template for very prevalent or “universal” themes that occur in data modeling. The intent of these patterns is that they can “apply to a great variety of uses”

and that they can be used by a great number of organizations to save time and effort while offering holistic (that is, universal) perspectives.

What we have found based upon decades of data modeling experience is that the same types of patterns continually occur in data modeling efforts. In this book, we have chosen what we think are the most common, “universal” patterns in data modeling. We have found that a great majority of the data modeling in most organizations have to do with roles (that parties play), hierarchies and recursions, classifications, statuses, contact mechanisms, and business rules. Thus, we have provided patterns for each of these types of data. Though there are other types of patterns for data modeling, we have chosen these because we believe that these patterns are the most common and, therefore, will provide the greatest benefit.

In each chapter that describes a pattern, we provide different alternatives for the pattern and examples of applying the pattern to a specific data requirement. Each alternative provides a pattern for modeling the same type of data at different levels of generalization. For example, in the status chapter, we provide a very specific pattern that models statuses as attributes, then another less specific model that has a STATUS TYPE entity, then a more generalized model that allows any number of status types for an entity, and then an even more generalized model that provides a STATUS APPLICATION entity allowing all entities needing statuses to have a relationship to this common data model structure.

What Is the Significance of Patterns?

The Universal Patterns for Data Modeling are analogous to the blueprints engineers use for building bridges. An engineer has a basic blueprint for building any type of suspension bridge. Every time an engineer has to build a particular suspension bridge, that engineer doesn't try to come up with a new solution; he or she uses the existing design pattern. The facing on the bridge may be different, but many of the underlying structures are the same. For example, Akashi-Kaikyo Bridge in Tokyo and the Brooklyn Bridge in New York are both suspension bridges, and the same basic design patterns were used for both.

The Universal Patterns for Data Modeling represent effective practices and alternatives for modeling very common types of data models. The underlying data model showing how a PARTY is related to an INVOICE is very similar to how PARTY(s) are related to SHIPMENT(s), which is also very similar to how PARTY(s) are related to PAYMENT(s), AGREEMENT(s), or other entities. For example, parties (people or organizations) may have certain roles within the context of a particular transaction or with regards to another entity, and there are very common ways to model this type of data requirement. We call this particular example the Contextual Role Pattern. Another example is that the status of a PROJECT or a financial TRADE is the same basic pattern, just applied to a different category of data. We call this the Status Pattern. Why try creating a new data structure every time you come across a status or a “contextual role” when the blueprint already exists?

As we have said, the first two volumes of the book focus on providing template data model constructs for common and industry purposes that a great number of data modelers and enterprises have used to jump-start their efforts. However, when we are on our consulting engagements, many clients have asked how to extend these models, apply them to additional industries, and/or create their own

examples of reusable models. When we examined this question for a solution, we thought the natural extension of Universal Data Models was to provide Universal Patterns that furnish the underlying building blocks that can be reused to provide a jump-start and alternatives in any data modeling situation and to provide quality and consistency in any data modeling effort.

Approach of This Book

Most data modeling books focus on techniques behind how to data model. This book assumes that the reader has a basic knowledge of data modeling. Data modeling has been around long enough that most information systems professionals are familiar with this concept and will be able to understand this book. By reading this book, data professionals of all kinds will be able to build upon, customize, and refine the existing data model patterns contained within the book in order to develop data models for their organizations and save time while increasing quality to develop new data models. Essentially, it is providing the professional with fundamental tools and building blocks that can be reused. The data professional, or anyone involved in data modeling, can thereby be more productive because we are providing preliminary foundations.

As we mentioned, each chapter contains different variations, or levels, of the same pattern, starting with the most specific version of the pattern and moving toward the most generalized versions of the pattern. Each version, or level, of the pattern may be applicable across a wide variety of information requirement needs for many different organizations. These patterns are the templates that can be reused across a variety of different subject areas. For example, the classification patterns can be used to support

classifications for many different entities, such as PARTY, TRADE, INVOICE, WORK EFFORT, SHIPMENT, and so on. Then we take each version or level of the pattern and show how it can be used in a particular scenario. These scenarios are normally based on our real-life experiences. For example, in one chapter, we describe the different ways to classify products at different levels of generalization for a fictitious computer hardware and software retailer called Euro-Electronics. Although all the people, organizations, sample data, and scenarios throughout this book are fictitious, we often base our examples on data models that we have actually developed in the past.

In each section we have tried to maintain the basic layout of each of the diagrams so that certain entities are in the same place in the diagram. This helps to show the evolution of the different patterns as they go through each level of generalization. This was not always possible, in particular in Chapter 9, where we bring many different patterns together into different models for different information requirements.

The Different Pattern Levels

Different levels of generalization are described in each chapter. Each of the patterns evolves from a specific pattern to a more and more generalized pattern. Within each chapter, each pattern models the same types of data, only with a different data model structure and style. For example, each of the contact mechanism patterns in Chapter 7 handles the data associated with various types of contact information, or as we call them, contact mechanisms (e.g., telephone numbers, postal addresses, email addresses, and so on). Initially, contact mechanisms are handled in a specific manner by modeling them as attributes of a particular role such as the having a **country telephone code, area code**, and **telephone number** attributes in a

CUSTOMER entity. We call this very specific pattern the Level 1 Contact Mechanism Pattern. Subsequently, each of the patterns becomes more and more flexible in its approach by using more and more generalized data model constructs to model this same type of contact information. Thus, we then show the Level 2 Contact Mechanism Pattern, which is a more generalized pattern, then the Level 3 Contact Mechanism Pattern, which is even more generalized, and finally the Level 4 Contact Mechanism Pattern, which is the most generalized version. The level and style of pattern that you may choose to use depends on the needs of the enterprise being modeled and the circumstances involved in the modeling task.

How can you answer the question whether to use a specific pattern or a generalized pattern? You can first ask the question, “What is the purpose of a data model?” We believe that there are two key purposes to a data model:

1. To illustrate and communicate information requirements.
2. To provide a sound foundation for a database design.

These purposes can be at odds with each other. If the purpose is to illustrate and communicate information requirements, the modeler will most probably develop a more specific model showing the specific needs of the business representative. For instance, in order to define what the information requirements are for contact information, the modeler may show attributes of **country telephone code**, **area code**, **telephone number**, **email address**, and so on, within specific entities such as CUSTOMER, SUPPLIER, or EMPLOYEE. Accordingly, this would be considered a specific style of modeling and would be a Level 1 Contact Mechanism Pattern.

Note

We want to emphasize that caution should be exercised with the use of level 1 patterns because these patterns are not generally

an effective foundation for a solid database design. As we stated, data models generally have two purposes: They can be a tool for understanding data requirements, and they also serve as a starting foundation for a database design. The level 1 patterns serve the former purpose very well; however, they are usually very ineffective regarding the latter purpose.

In contrast, if the purpose is to model a sound foundation for a database design, the modeler may need to incorporate more flexibility and use a pattern such as a Level 3 Contact Mechanism Pattern or a Level 4 Contact Mechanism Pattern, where any party (person or organization) can have any number of contact mechanisms that have various types, purposes, usages, and priorities and can be classified any number of ways. Thus the model is very stable and is very unlikely to need changes if there are future requirements for additional types or classifications of contact mechanisms. These types of models tend to be more difficult to understand and do not contain as many specific rules that are enforced in the model. For example, in the generalized form of the contact mechanism, any party may have any number of contact mechanisms, but there may be a rule that a particular person should have only one active pager number. The generalized data model pattern does not enforce this rule, the specific data model pattern does enforce it (because you could have a single attribute for **pager number**).

Thus, we recommend that, especially for generalized data model patterns, you document the relevant business rules. There are numerous robust solutions for documenting these rules in a business rules engine or metadata repository, however, we have found that many enterprises do not have these types of solutions available to them or they may be in the process of creating these solutions. Therefore, you could consider a simpler method of documenting these rules by recording them in a document that is as an adjunct to the data model. Also, some of the patterns in this book,

especially those in chapter 8, provide data structures to capture various business rules. In addition to documenting business rules, we believe that it is very important to illustrate all data models, but especially more generalized data models with data examples/instances of the model.

Figure 1.1 illustrates that as you move from a level 1 to a level 3 or level 4 pattern, you are moving from a more specific style of modeling to a more generalized style of modeling. It also shows that level 1 patterns are used when the data is more "static," and the higher levels of patterns accommodate the need for more flexibility. Thus, if the nature of the data is static and does not change (for example, you need only a single phone number), a more specific modeling style may be appropriate. However, when the nature of the data changes over time (for example, there may be any number of different types of telecommunications numbers that may be needed in the future), a more generalized style of modeling may be more appropriate. Throughout the book, we discuss the pros and cons of each particular pattern. Because each type of pattern will have specific and generalized alternatives, Table 1.1 summarizes both the benefits of using a specific style of modeling and the benefits of using a more generalized style of modeling.

Note

We have used the word *generalization* instead of using the more common term *abstraction*. Many data professionals believe that *abstraction* implies a loss of detail. For example, a map of a roadway is an abstraction because it limits details to a certain level in order to focus attention on roads. *Generalization* implies transforming very specific data model structures to more generalized concepts, in order to more flexibly support data requirements. *Generalization* provides this flexibility by using a less specific data structure and accommodates current and future requirements via adding, changing, or deleting data instances. Another reason for using the term *generalization* is that the object-oriented community uses the term *abstraction* in

a different way that has a different meaning. It should also be noted that dynamic environments require flexible solutions. However, flexible solutions by their nature are more generalized, and generalized solutions are more difficult to understand.(7)

Also, *generalization* should not be confused with *normalization*. They are completely separate concepts. Generalization has to do with using more flexible data model constructs, whereas normalization has to do with eliminating data redundancy by grouping data in a way that it is dependent on “the key, whole key, and nothing but the key.”

Figure 1.1 Levels of generalization

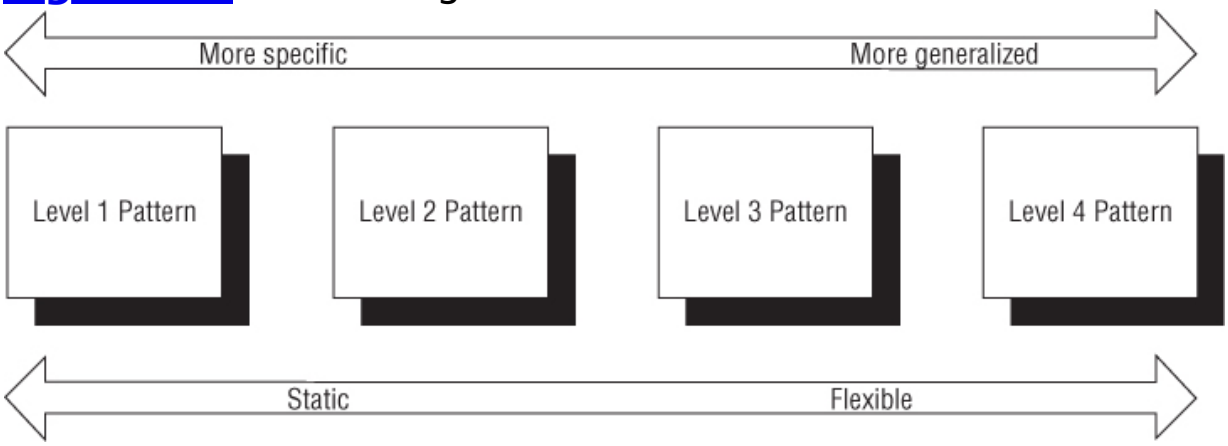


Table 1.1 Benefits of Specific and Generalized Styles of Modeling

BENEFITS OF A MORE SPECIFIC STYLE OF MODELING	BENEFITS OF A MORE GENERALIZED STYLE OF MODELING
Easier to understand model.	More flexible. Can more easily accommodate additional data and/or changes to data, without needing to change the data model. Provides the ability to meet more current and future needs.
Easier to use as a way to communicate with nontechnical audiences, validate and gather requirements, and define scope of the data requirements.	More consistency. Higher-level patterns tend to result in data models that have the same type of structures and can promote consistency and standardization, either within the same data model or across data models.
Good way to start in order to understand the data	Basing the physical database on more generalized data models allows more use of

requirements before generalizing the model.	common routines to manage and access data, because the data structures tend to be more similar.
Can specify and enforce more business rules directly via the data model.	Can sometimes provide more power and capabilities by combining various types of data within the same data model structure. For example, the model allows for powerful analysis capabilities when maintaining all classifications of products together in the same entity.
Easier to implement prototypes.	Provides much more solid and stable foundation when used as a basis for a physical database design, especially when using to develop a robust, production quality database design.

If you are familiar with the Zachman Framework,(8) you may recognize that there may be different audiences for data models, which results in the need for different types and styles of models. For example, a model that is designed for the owner/business representative in order to validate information requirements may look quite different than a model designed for the designer/architect where the model's intention is to be the basis of the database design. The model that one develops for an owner or business representative would most likely be a specific model such as a level 1 or level 2 based model, so that one could use the specific patterns to illustrate and communicate the data needs. The model for a designer or architect would most likely be designed for flexibility and adaptability to change, thus reducing maintenance costs, and so a level 3 or level 4 may be more suitable for this.

Note

John Zachman's framework shows six different rows and six different columns. The six columns correspond to different types of models in IT development, and the six rows correspond to different audiences. In this book, we are focusing on column 1 (the models for data), and we are providing different views by showing different levels. For instance, using more specific patterns such as level 1 and level 2 patterns, would usually

work well for models that correspond to the Zachman Framework row 2 that is designed for an “Owner” view. Using level 3 and level 4 patterns would most likely work well for models that correspond to row 3 in the Zachman Framework, which according to the framework are models for the audience of designers or architects. Depending on how you interpret the Zachman rows and how you intend to use the patterns, you may also make the argument that some of these patterns, such as level 1 patterns, can be used for row 1 (the “planner” view), and some of the more generalized patterns can be used for row 4 (the “builder” view). The key point we are making is that different levels of the patterns are designed to be used by different audiences.

Some data modelers prefer to have different models for different audiences. However, to maintain two models, a “business data model” and an “architecture data model,” and to map and cross-reference them can be quite a bit of work. The patterns can help a great deal in this regard. When developing a data model for business representatives in order to gather and validate data requirements, we will generally use the level 1 and level 2 patterns. Then when developing the “designer” or “architectural” view, we can replace the level 1 or level 2 patterns with level 3 or level 4 patterns. Thus, there is a “plug-and-play” nature of these patterns that can save a great deal of time and help synchronize these types of models. For example, a Level 1 Status Pattern showing the status of an order can be quickly replaced by a Level 3 Status Pattern to show a more flexible approach in the same model. Think of patterns as components that can be substitutes for each other.

Note

Chapter 9 illustrates many examples of how you can use the patterns in a plug-and-play mode for different types of data modeling efforts.

Instead of having two different data models for two different audiences, another possible solution is to

incorporate both specific and generalized patterns into the same model. (This solution is shown in Chapter 9, in the discussion of using the patterns to develop an enterprise data model.) Often, both a specific and a generalized pattern can be used in the same data model for the same data requirement. Then views can be created to show the specific aspects and generalized aspects of the model. For example, if you had a need to model the roles of various parties in a project, you could develop a model of the specific relationships of various roles to the project, namely, sponsors, workers, project manager, and project lead, in order to validate requirements. Then you could include in the model additional entities showing an architectural view of the model where a work effort (which could be a project, activity, task, or any other unit of work) may have any number of parties with any number of roles associated with it.

It is important to keep in mind that this type of “hybrid” modeling solution can be used for any of the patterns in this book. In Chapter 3, we have shown an example of this by providing a hybrid pattern, namely, the Hybrid Contextual Role Pattern. A “hybrid” pattern uses both a specific pattern and a generalized pattern to model a specific type of data requirement. For example, the Hybrid Contextual Role Pattern provides a single pattern that includes both the specific Level 2 Contextual Role Pattern (that models specific roles such as a PROJECT to PROJECT LEAD) and the Level 3 Contextual Role Pattern (that models generalized roles such as a PROJECT to PROJECT ROLE relationship). We could use this same idea to develop a “Hybrid Status Pattern,” “Hybrid Classification Pattern,” “Hybrid Recursive Pattern,” or for any of the other patterns in this book.

Note

The “Hybrid” approach is designed to show alternative ways to model the same type of data: one using a specific method and one using a much more generalized way to model; this is not the same as saying it is okay to model the same data redundantly. We don't consider this to be redundant data modeling, because we don't advocate that you capture the same instances of data in both the specific and the generalized data model structure. We describe this approach in greater detail in chapter 9.

So, why did we describe these patterns using the idea of levels instead of relating them to conceptual data models, logical data models, and physical data models? First of all, the patterns have more to do with the levels of generalization for the model than they have to do with the idea of conceptual, logical, and physical data models. In *Data Model Theory and Practice*(9) Dr. Graeme Simsion points out through extensive studies that the same information requirements within the same scenario are likely to be modeled very differently by different modelers. Furthermore, he points out that three key differences occur between models that are based on the same information requirements. One of these differences that reflect a wide degree of variation in data modeling is the level of generalization. Thus, the levels in this book highlight the degree of generalization, and level 1 patterns have very little generalization whereas level 4 patterns have a high degree of generalization (see Figure 1.1).

Note

In Dr. Simsion's book “Data Model Theory and Practice”, he cites a classification scheme from J. Verelst(10) that shows three major reasons for variability among data models. These are “construct variability” (use of different modeling constructs, such as attributes or entities, to represent the same real world concepts), “vertical variability” (use of different levels of generalization), and “horizontal variability” (different categorization of data at the same level of generalization). While we focus on providing patterns at different levels of

generalization, the patterns that we will be showing you also show alternatives that address “construct variability.”

Another reason that we have, for the most part, stayed clear of comparing these patterns to conceptual, logical, and physical models is that there is great debate in the data management industry regarding what exactly a conceptual data model, logical data model, or physical data model is; what is included in each of these models; and what is the difference between and among these models. Karen Lopez, a well-recognized and prominent industry leader in data management, conducts a seminar called “Data Modeling Contentious Issues.”(11) She has conducted this course for over a decade, polling participants and asking questions such as “What is a conceptual data model?” and has consistently received many widely different views of what concep tual data models, business data models, logical data models, and even physical data models are. As she points out, data modelers often get very heated in discussions about various data modeling contentious issues such as what level of generalization a model should have, if attributes should be part of the conceptual or business data model, if models should use the “party” concept, and so on. This same point, namely that there is a lack of common perspective from data modelers, from novices to gurus, has also been raised by Dr. Simsion, who shares his extensive research about the question and ongoing debate even regarding the very nature and purpose of data modeling in his book.(9) In the data modeling industry, there does not appear to be a common, single, universal understanding regarding the purpose and definitions of conceptual models, business models, logical data models, and physical data models.

Because it is difficult to come to a common definition of conceptual, logical, and physical data models that are broad enough to be generally accepted and specific enough to be

rigorous, we have a classic “Catch-22” situation if we frame the discussion of patterns around these concepts. We believe that taking a stance regarding what we consider to be a conceptual, logical, and/or physical data model or debating the definitions of these models would distract from what we want to offer in this book. We believe that there is another way to categorize data models, namely by specifying the level of generalization, and this can be more helpful in our goals.

As data modelers, we are usually asked to create data models that meet specific business needs. For example, we are asked to create a model that illustrates the required business data by using objects such as entities, relationships, and attributes. The enterprises that need data models want us to create models to support particular functions, and data modelers have tried to segment these models into categories that have meaning primarily to data modelers (conceptual model, business model, logical model, and so on). So, instead of using these categories we have decided to categorize data models by how generalized the model is. In turn, the level of generalization implies suitability of the model for a particular purpose or function. As we already stated, very specific models are generally used to communicate information requirements to business representatives and more generalized models are commonly used as the basis for a flexible foundation for a database design.

Another benefit of this book is that it will show various alternatives at different levels of generalization for each pattern, pointing out the pros and cons for each alternative and allowing the modelers to make intelligent choices for their model extension or new model. For example, for modeling contextual roles (relationships from an entity to a party) we point out five alternatives (level 1, 2, 3, 4, and a hybrid pattern). *The Data Model Resource Book*, Volumes 1

and 2, use these various alternatives in the various Universal Data Models; however, now you can understand the rationale regarding when to apply each alternative and use similar guidelines that were used to create the first two books to know when and how to extend, customize, or create new models.(12)

Who Is the Intended Audience for This Book?

This book is written for data modelers, data architects, data analysts, database administrators, database designers, data stewards, computer science teachers, computer science students, corporate data integrators, as well as anyone involved in any aspect of data modeling. The content of this book is suitable for use by professionals in the fields of data management, data quality, metadata management, master data management, data warehousing, data governance, and any other field where data models are used. Anyone involved in these roles or professional fields can use the data model constructs contained within this book to increase their productivity, to provide a checkpoint to identify potential pitfalls, and to increase the quality of the model by understanding alternatives and by applying patterns.

Aside from being an invaluable toolkit for systems professionals who focus on this area, this book can also be used as a text for organizations such as corporations or universities.

Many people prefer to learn by example so this book is both a tremendous aid for the experienced practitioner as well as a guide for the novice by showing many well-thought-out data model patterns and examples using these patterns.

What Is in This Book

The majority of this book contains chapters with what we believe are the most common and useful Universal Patterns. Chapters 2–8 contain reusable patterns and alternatives for data modeling. These chapters include explanations of the patterns, examples of each of the patterns, sample data, and the pros and cons of each modeling alternative. Chapter 9 describes how to apply the patterns for different types of data modeling efforts. Chapter 10 provides ideas for gaining buy-in regarding using the patterns and/or standardizing on these patterns. Specifically:

- **Chapter 2, “Setting Up Roles: What Parties Do,”** defines what a declarative role is and then provides data model patterns that can be used to model what people and organizations do, or in other words, how they act within the broad context of the overall enterprise. For example, a person or organization may be a customer, supplier, and/or employee. The chapter describes how each pattern supports the data related to each role and the data associated with the party (person or organization), independent of that party's role.
- **Chapter 3, “Using Roles: How Parties Are Involved,”** defines what a contextual role is and provides data model patterns and alternatives that can be used to model what people and organizations do within the context of specific business activities or other entities. For example, this covers the role of a customer within the context of an order (e.g., they may be the “bill to” customer, a “ship to” customer, “end user” customer, or so on for the specific order).
- **Chapter 4, “Hierarchies, Aggregations, and Peer-to-Peer Relationships: The Organization of Similar Data,”** defines the different ways that data may be related to similar types of data, for example, how work

efforts are related to other work efforts, how parts are related to other parts, or how parties are related to other parties. The chapter provides patterns and alternatives to model recursive relationships.

- **Chapter 5, "Types and Categories: The Classification of Data,"** defines taxonomies, types, and classifications, and then provides data model patterns and alternatives that can be used to classify any type of data. For example, these patterns may be used to model classifications for parties, products, orders, work efforts, assets, or any other entity that has classifications.
- **Chapter 6, "Status: The States of Data,"** defines what a status is and provides patterns and alternatives that offer ways of modeling the state of a particular transaction or entity. For example, the state of an order may be "Received," "Pending credit check," "Entered," "Cancelled," or "Fulfilled." The state of a product may be "Conceived," "Introduced," "Discontinued sales," or "Discontinued support." Each pattern supports three fundamental aspects of statuses, that is, the allowable statuses for an entity, what the current status is for the entity, and the history of its statuses, including when each status was first effective and when it was no longer effective.
- **Chapter 7, "Contact Mechanisms: How to Get in Touch,"** describes what a contact mechanism is and provides data model patterns and alternatives that can be used to support the needs of an enterprise when they wish to maintain data about telephone numbers, email addresses, postal addresses, and other types of contact information. This chapter provides various patterns to maintain contact mechanisms and their types, purposes, usages, locations, priorities, and other classifications.

- **Chapter 8, “Business Rules: How Things Should Work,”** describes data model patterns and alternatives that can be used to create a data-centric approach to maintaining the rules that govern how the enterprise operates. The patterns maintain data about three aspects of a rule: data about the rule itself, data about the factors involved in the rule, and data about the outcomes of the rule. For example, a pricing rule may have data about the rule (a rule name and/or rule statement), data about the factors (relationships to GEOGRAPHIC BOUNDARY, QUANTITY BREAK, and so on), and data about the outcome (the price or discount that is to be applied for the specified factors).
- **Chapter 9, “Using the Patterns,”** illustrates how to apply the different patterns for different efforts and circumstances. These types of efforts include applying the patterns for gathering requirements, developing a prototype, developing a specific application data model, develop ing an enterprise data model, developing a relational data warehouse data model, developing a star schema–based data warehouse data model, and developing a master data management data model. We show how these patterns can be used in a plug-and-play fashion by substituting different levels of patterns depending on the type of data modeling effort involved and what level of generalization is needed.
- **Chapter 10, “Socializing the Patterns,”** describes personal, cultural and political factors and the human dynamics involved in gaining acceptance for using these common patterns, based upon our experiences of implementing these patterns at various enterprises. The chapter provides “Universal Principles” that can be used to socialize the patterns, whether you are creating enterprise-wide standards and guidelines or trying to gain acceptance for using these patterns to help jump-

start a data model for a particular effort. Specifically, this chapter describes ways to gain buy-in for using these patterns by understanding motivations as to why people would or would not use them; by creating a common, clear, and compelling purpose and vision for the patterns; by developing trust so people know they can rely on the patterns; and by effectively managing conflict if and when it arises.

> **Note**
>
> To enhance the readers experience each of the figures in this book can be viewed at www.wiley.com/go/datamodelresourcebookvol3.

Other Patterns for Data Modeling

The intent of this book is to show the patterns that relate to the most commonly used data model constructs. Many other patterns for data modeling exist that we have not included in this book, such as the following (to name a few):

- **Name:** This pattern provides alternatives regarding maintaining the names for an entity. This pattern maintains multiple names for an entity, name history, and a flexible approach to maintaining any number of names and name parts, among other things. The pattern could be applied to naming a person, an organization, a product, or any other entity that has many different types of names.
- **Identifier:** This pattern provides a common structure for maintaining identifiers for an entity in multiple ways. For example, if you open your wallet, you can see many different ways that you can be identified: a driver's license, medical card, social security number, and so on.

Products can have many different types of identifiers from many different sources. Investment vehicles may be identified in different ways such as a ticker symbol, a CUSIP number, and an ISIN number. The identifier pattern supports the various methods of identifying data including ways that may come from many different sources.

- **Transactions and events:** Various types of transactions such as orders, shipments, invoices, and accounting transactions (to name a few) have some very common attributes and similar ways, or in other words, patterns, you can use to model them. For example, many transactions are initiated by an event (such as a customer ordering products, a customer complaining, and so on). They often have detailed items (for example, an ORDER ITEM, a SHIPMENT ITEM, an INVOICE ITEM, and so on). They are often related to each other (for example, ORDER ITEM(s) are related to SHIPMENT ITEM(s) in much the same way that SHIPMENT ITEM(s) are related to INVOICE ITEM(s)). Finally, transactions and events often have similar types of roles, statuses, classifications, recursive relationships, and business rules.
- **Authorizations:** This is a common pattern that can address alternative ways to model what permissions are needed to access various types of data, what can be shared, and who has access to what types of data. For example, who is allowed to access what data when logging into a web site, taking cash out of an ATM, using a PIN over the telephone, or under other circumstances where there is a need to provide authorizations and/or permissions to create, access, update, or delete data.

Conventions and Standards Used in This Book

The following section describes the naming standards and diagramming conventions used for presenting the models within this book. The data modeling notation that we use in this book is a slightly modified version of the notation advocated by Richard Barker in his book *Case*Method: Entity Relationship Modelling*.(13) The following sections provide conventions and standards that we use in this book to model entities, supertypes/subtypes, attributes, relationships, and example data (using illustration tables to provide examples of data that could be captured). We then provide some examples of common types of data modeling notations and provide a brief explanation of why we chose the notation used in this book.

Entities

An *entity* is something of significance about which the enterprise wishes to store information. Whenever entities are referenced throughout the book, they are shown in capital letters. For example, ORDER represents an entity that stores information about a commitment between parties to purchase something. When the name of an entity is used in a sentence to illustrate concepts and business rules, it may be shown without capitalization - for example, the word "*order*" is not capitalized in the sentence: "Many enterprises have mechanisms such as a sales order form to record sales order information." The naming conventions for an entity include using a singular noun that is as meaningful as possible to reflect the information it is maintaining. In this book, we have also provided numerous definitions for many of the core entities in each of the different chapters.

Entities are represented by rounded boxes. Figure 1.2 shows an example of the entity ORDER.

Figure 1.2 An entity

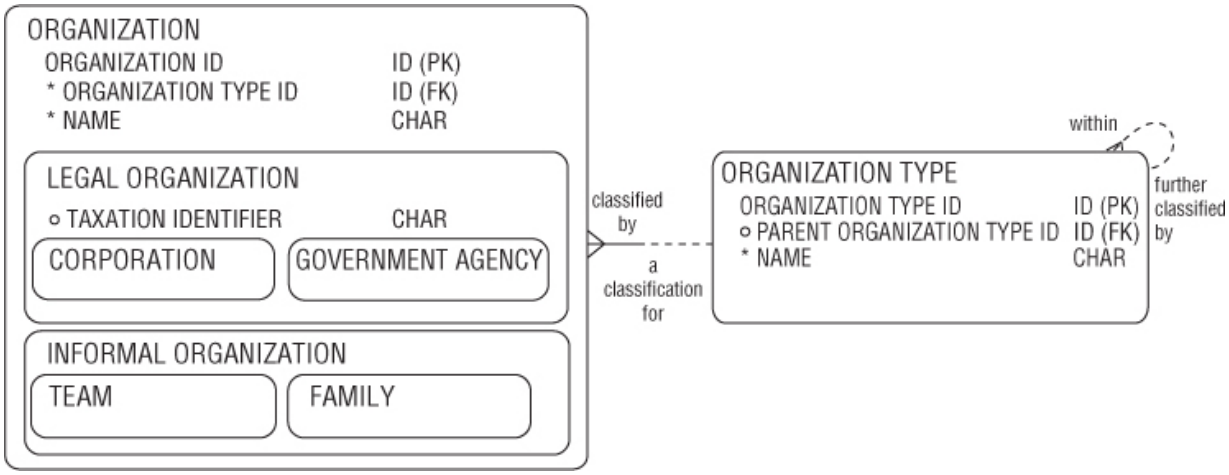


Subtypes and Supertypes

A *subtype*, sometimes referred to as a subentity, is a subdivision of an entity that has characteristics such as attributes or relationships in common with the more general entity (that is, the supertype). Also, the subtypes may have attributes and relationships that are specific to that subtype. LEGAL ORGANIZATION and INFORMAL ORGANIZATION are, for example, subtypes of ORGANIZATION.

Subtypes are represented in the data modeling diagrams by entities inside other entities. The common attributes and relationships between subtypes are shown in the outside entity, which is known as the *supertype*. The attributes and relationships of the supertype are, therefore, inherited by the subtype. Figure 1.3 shows the supertype ORGANIZATION and its subtypes of LEGAL ORGANIZATION and INFORMAL ORGANIZATION. Notice that the **name** applies to the supertype ORGANIZATION and the **taxation identifier** applies only to the LEGAL ORGANIZATION subtype and is, therefore, shown at the subtype level of LEGAL ORGANIZATION because it applies only to that subtype. Both LEGAL ORGANIZATION and INFORMAL ORGANIZATION would have their name maintained in the data model because they inherit the values of the supertype.

Figure 1.3 Subtypes and supertypes



Supertypes may have many levels. Figure 1.3 shows that a CORPORATION and GOVERNMENT AGENCY are subtypes of LEGAL ORGANIZATION, which is also a subtype of ORGANIZATION. Another subtype of ORGANIZATION is INFORMAL ORGANIZATION, which may have subtypes of a TEAM or FAMILY. Thus, boxes may be in boxes down to any level to illustrate which subtypes inherit the attributes and relationships of the parent supertype (its outer box).

Each subtype should be mutually exclusive of each other, and they should represent a complete set of classifications at that level of classification, meaning that the sum of the subtypes covers the possible classifications for that supertype at that level. For example, an ORGANIZATION may be either an INFORMAL ORGANIZATION (one subtype) or a LEGAL ORGANIZATION (another subtype). This complete set of classifications may be at a higher level of classification, and there may be more detailed subtypes that are not included explicitly in the data model; instead, they may be included in a TYPE entity, as seen in Figure 1.3 with ORGANIZATION TYPE. So sometimes, entity classifications are shown in two places on a model: as a subtype and an instance in a TYPE entity that maintains the domain of allowed types for the entity. A reason for modeling subtypes in this way is that there may be attributes and/or relationships about a specific subtype, and thus, it is

modeled as its own entity, and there may also be other attributes and/or relationships that are about the TYPE entity. For example, a subtype of a PARTY ROLE may be a CUSTOMER, which has its own attributes and relationships You may also have a ROLE TYPE that has an instance of CUSTOMER (as well as all the other role types), and this may be related to other entities such as AUTHORIZATION showing what roles are authorized for what permissions. Thus, you need to have both the subtype and the generalized TYPE data model constructs. In this book, we usually show a TYPE entity when we have a subtype supertype structure in a model. At a minimum the TYPE entity contains instances that correspond to each of the subtypes. We further describe this concept in Chapter 5.

> **Note**
>
> Some data models require mutually inclusive subtypes. While we show a notation for this in Volumes 1 and 2, since we don't use mutually inclusive subtypes in this book, we don't provide a convention for them in this book.

Attributes

An *attribute* holds a particular piece of information about an entity, such as the **order date** on an order. Attributes are identified in the text of the book by boldface, lowercase letters such as the previous **order date** example. In the diagrams, the attributes appear in uppercase.

Attributes are shown within entities and have the following parts to them:

- An optional (o) or mandatory (*) indicator for non-primary key attributes. In Figure 1.4 you see ORDER ROLE has a mandatory **from date** and a non-mandatory (that is, optional) **thru date**. Primary key attributes such as **party id** in PARTY, **order id** in ORDER, **order**

role id in ORDER ROLE, and **role type id** in ROLE TYPE, have no optional/mandatory indicator because a primary key is always mandatory.

- The text representing the attribute name, for example, **from date** in ORDER ROLE.
- The data type representing the nature of data that the attribute will maintain. For example, an identifier has the data type of “ID” (identifier), such as the **order id** in the ORDER entity, or the **from date** in the ORDER ROLE entity has the data type of “DATE” (storing a date or datetime value). See Table 1.2 for a list of the data types that we use in this book.
- For primary or foreign keys, a (PK) or (FK). In Figure 1.4 you can see that ORDER ROLE has three foreign keys: one from ORDER (**order id**), one from PARTY (**party id**), and one from ROLE TYPE (**role type id**). All of these attributes are followed by (FK). You can also see that each of the entities have their own primary key: **order id**, **order role id**, **role type id**, and **party id**. By convention, the name for all primary keys used in this book are the entity name followed by id; for example, the primary key for ORDER is **order id**. The values for primary keys in each entity are non-meaningful unique sequence numbers, known as surrogate keys. We use this approach because data may change, and by using a non-meaningful sequence number for our primary key (as opposed to having a primary key that has meaning such as a social security number to identify persons), we can develop a data model where we do not have any problems should the data change (for example, a change to a social security number).

Note

Data modelers have differences of opinions regarding whether the data model should use surrogate keys (keys without any

inherent meaning and are used just to relate entities to each other) or natural keys. We recognize that there is a valid argument for both schools of thought on this issue, but we felt it was more natural to use surrogate keys (no pun intended) in relation to patterns, abiding by the practice of using non-meaningful keys so that if a key changes within a particular instance of an entity, it would not cause data issues or anomalies. (These could occur because keys are the mechanism to link together entities, so changing the value of a primary key can have a rippling effect that can lead to data inconsistencies and other problems.)

- A unique identifier symbol of "(UID)" is used to show when there are alternate identifiers aside from the primary key that uniquely identify the entity. For example, for the entity ORDER ROLE, **party id**, **role type id**, **order id**, and **from date** show (UID) to indicate that this combination of all of the keys can be used to uniquely identify a specific instance. This helps show the nature of the key. For example, in this case it shows that a **party id**, **order id**, and **role type id** alone are not sufficient to make this instance unique because the same order (order #123) for the same party (John Smith) could have the same role (bill-to customer) at two different points in time if that party is first identified as the bill-to customer, then a different party is identified as the bill-to customer, and then the first party is again identified as the bill-to customer. Thus, the **from date** is needed to make the instance unique.

Table 1.2 Data Types Used in the Book

DATA TYPE	TYPE OF VALUES THAT THIS WOULD INCLUDE
ID	A non-meaningful identifier used to specify primary keys. This would normally be a (positive) sequence number that increments. For example, 1, 2, 3, 4, 5, and so on.
DATE	The day, month, and year, for example, Sep. 5, 9/5/2003. We don't specify any particular format for dates. However, in the table

	examples in each chapter we express dates in the form, Mon. DD, YYYY. For example, Feb. 15, 2006.
DATETIME	The date and the time of day that would be provided by a clock, for example, 3/4/10 4:13 p.m. We don't specify any particular format for datetimes such as 12 or 24 hour specifications.
CHAR	Characters, or in other words, a text string.
DESC	A description that expresses information about the nature of the entity and is generally a larger text string than a CHAR data type.
IND	An indicator or flag. This means that there may be only two possible values for the attribute. For example, “Y(es)” or “N(o)” or “M(ale)” or “F(emale)”.
NUMBER	A positive or negative numeric value, including floating-point values as well, for example, 125, 1.0, 9.25, -45 and so on.
MONEY	An amount or sum of money, for example, $1,000,000 or £100 or HK$10,000. In an attribute that has a money data type, the data models will maintain a value such as “1,000,000,” “100,” or “10,000,” and if there is a need to maintain different international currency amounts, there may be a relationship to a CURRENCY TYPE entity that can maintain “US Dollars” “British Pounds” or “Hong Kong Dollars” and the appropriate symbol.

Figure 1.4 Attributes, relationships, and keys

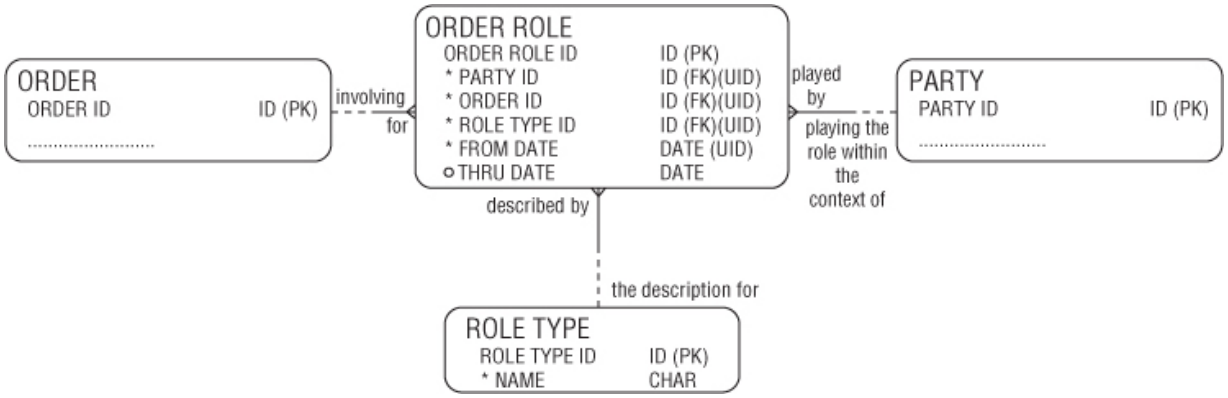


Table 1.2 shows the various data types that we use in this book and an explanation of the values that would be included for each data type.

Certain strings included in an attribute's name have meanings based on the conventions shown in Table 1.3.

Table 1.3 Conventions Used in Attribute Naming

STRING WITHIN ATTRIBUTE NAME	MEANING

ID	System-generated sequential unique numeric identifier (for example, 1, 2, 3, 4, and so on).
NAME	The term by which someone or something is referred to. For example, the ROLE TYPE **name**, PRODUCT **name**, and PARTY CATEGORY **name** signify the name used to refer to a role type (for example, “Customer”), product (for example, “A123 Widget”), or party category (for example, “Income level”).
DESCRIPTION	Text that expresses information about the entity to help describe the nature of that instance. For example, PRODUCT **product description** would describe the nature of the product and may be “This product is a stainless steel, 6-inch device that allows people to core apples much more easily.”
INDICATOR	A binary choice for values (for example, yes/no or male/female).
FROM DATE	Attribute that specifies the beginning date of a date range for which the instance is valid or effective and is inclusive of the date specified. For example, an ORDER ROLE **from date** specifies that the party first started (or will start) playing the role for that order on that **from date** value.
THRU DATE	Attribute that specifies the end date of a date range and is inclusive of the date specified (**to date** is not used because **thru date** more clearly represents an inclusive end of date range). For example, an ORDER ROLE **thru date** specifies that the party no longer played (or will no longer play) the role for that order *after* that **from date** value.
......	The dots mean a continuation of attributes that may be captured in the entity in the pattern but are not germane to the understanding of the pattern. In other words, “etc.”

Relationships

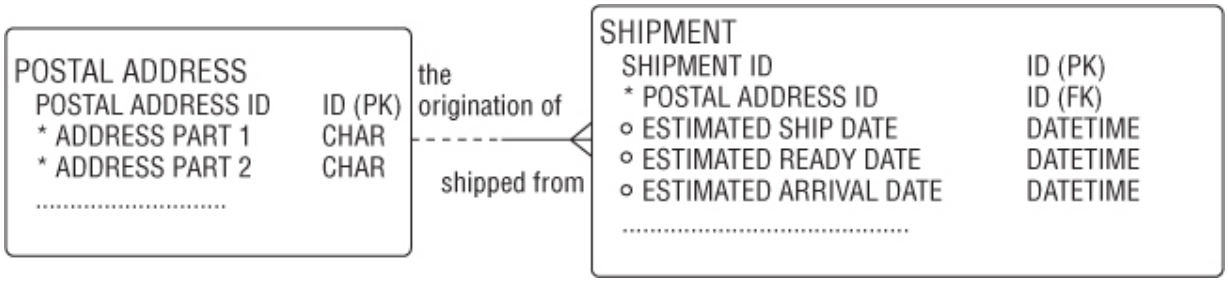
Relationships define how two entities are associated with each other. When relationships are used in the text, they are usually shown in lowercase as a normal part of the text. In some situations, where they are specifically highlighted, they are identified by boldface lowercase letters. For

example, **manufactured by** could be the way a relationship may appear in the text of this book.

Relationship Optionality

Relationships may be either optional or mandatory. A dashed relationship line next to an entity means that the relationship from that entity is optional, and a continuous line means that the relationship is mandatory (the relationship has to exist for all occurrences of each entity). Figure 1.5 shows a relationship that "each SHIPMENT *must be* **shipped from** one and only one POSTAL ADDRESS." This means that the postal address for each shipment must be specified in order to be able to have a SHIPMENT instance. The other side of this relationship is optional: "Each POSTAL ADDRESS *may be* **the origination of** one or more SHIPMENT(s)." Hence, there could be a specific postal address that has not been related to a specific shipment yet.

Figure 1.5 Mandatory versus optional relationships

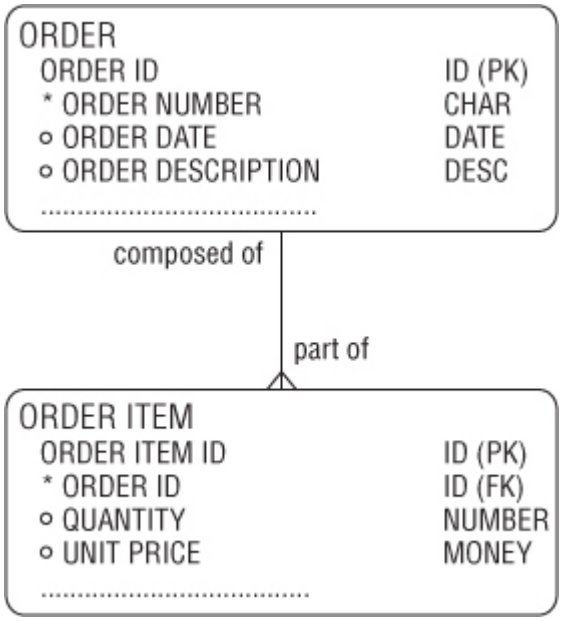


Relationship Cardinality

Relationships may be one-to-one, one-to-many, or many-to-many. This is generally known as the cardinality of the relationship. The presence of a *crowsfoot* (a three-pronged line that looks like a crow's foot) defines whether an entity points to more than one occurrence of another entity. Figure 1.6 shows that "each ORDER must be **composed of** *one or more* ORDER ITEM(s)" because the crowsfoot is at the

ORDER ITEM side. The other relationship side states, “Each ORDER ITEM must be **part of** *one and only one* ORDER.” A one-to-one relationship doesn't have any crowsfeet on the relationship, and a many-to-many relationship has crowsfeet at both ends of the relationship. Sometimes, one-to-many relationships are referred to as parent-child relationships.

Figure 1.6 One-to-many relationship



The data model diagrams do not show many-to-many relationships because many-to-many relationships are broken out into *associative* (that is, intersection) entities. This is a common data modeling practice to break up many-to-many relationships because there could be information that needs to be maintained about the associative entity.

Sometimes the term *over time* needs to be added to the relationship sentence to verify whether the relationship is one-to-many. For instance, an ORDER may appear to have only one ORDER STATUS because there is only one status that the order is in at any point in time; however, if status history is required, each ORDER may be in the state of one or more ORDER STATUS(es) over time.

Foreign Key Relationships

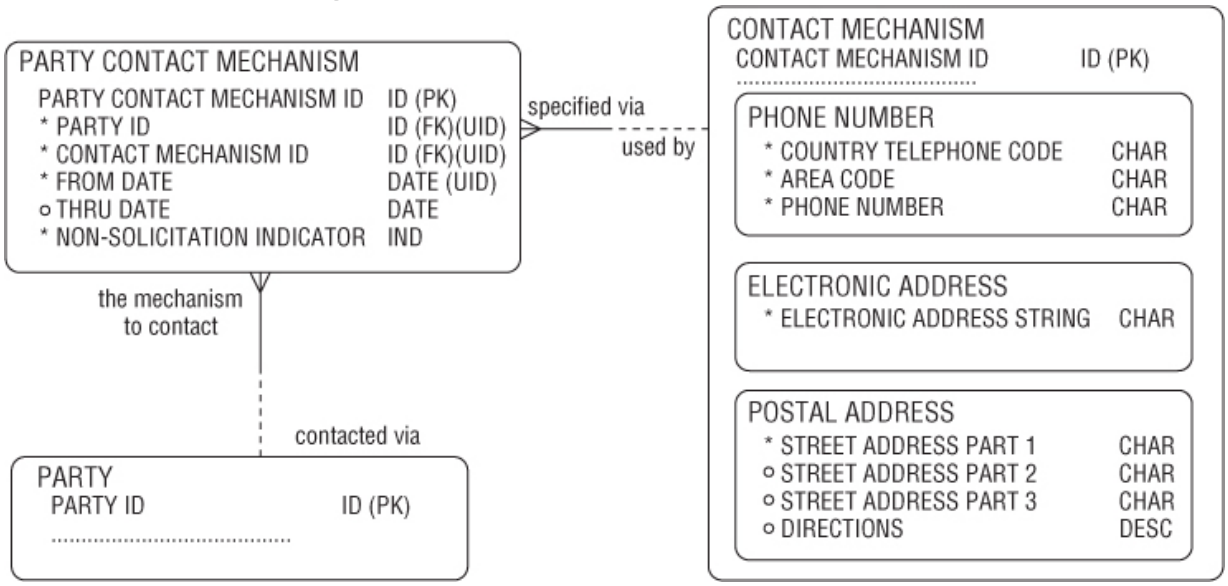
A *foreign key* is defined as the presence of another entity's (or table's) primary key in an entity (or table). For example, in Figure 1.6 the **order id** from the ORDER entity is a foreign key attribute of the ORDER ITEM. Any one-to-many relationship indicates that the primary key of the entity on the *one* side of the relationship is brought into the entity on the *many* (crowsfoot) side of the relationship. You can see that the foreign key attribute has a (FK) beside its data type to indicate that it is the foreign key from a related entity. Some data modelers don't show this foreign key as an attribute of the entity because this is redundant and can be derived from the relationship; in fact, in Volume 1 and Volume 2 of this book series we have done this in the interests of being more concise and being able to show more of the data model on the page. Because the patterns involved in this book generally involve fewer entities and take up less space, and because we received some feedback that perhaps this may be useful to readers, we have chosen to show the foreign keys as attributes within each entity in this volume. For example, in Figure 1.6, the **order id** is shown as an attribute in the ORDER ITEM entity.

Associative Entities to Handle Many-to-Many Relationships

Associative entities are also known as *intersection entities* or *cross-reference entities*. They are used to resolve many-to-many relationships by cross-referencing one entity to another. Often they include additional attributes that may further delineate the relationship. Figure 1.7 shows a many-to-many relationship between a PARTY and a CONTACT MECHANISM that is resolved via a PARTY CONTACT MECHANISM associative entity. Each PARTY may be related to many CONTACT MECHANISM(s) such as the party's

POSTAL ADDRESS, TELECOMMUNICATIONS NUMBER, or ELECTRONIC ADDRESS because people and organizations often have many ways to contact them. Conversely, each CONTACT MECHANISM may be related to more than one PARTY. For example, many people may have the same work address or work phone number. This many-to-many relationship is resolved by the associative entity PARTY CONTACT MECHANISM.

Figure 1.7 Many-to-many relationships resolved by an associative entity



Each associative entity inherits as a foreign key, the key of each of the entities it intersects. For example, the **party id** (inherited from PARTY) and the **contact mechanism id** (inherited from CONTACT MECHANISM) are foreign key attributes of PARTY CONTACT MECHANISM. These foreign key attributes are also parts of the unique key (UID) of the associative entity, and there may be other attributes that are part of the unique key as well. For example, the PARTY CONTACT MECHANISM foreign key attributes of **contact mechanism id** and **party id**, along with the **from date** make up the UID. The **from date** is needed as part of the UID because a party may have the same contact

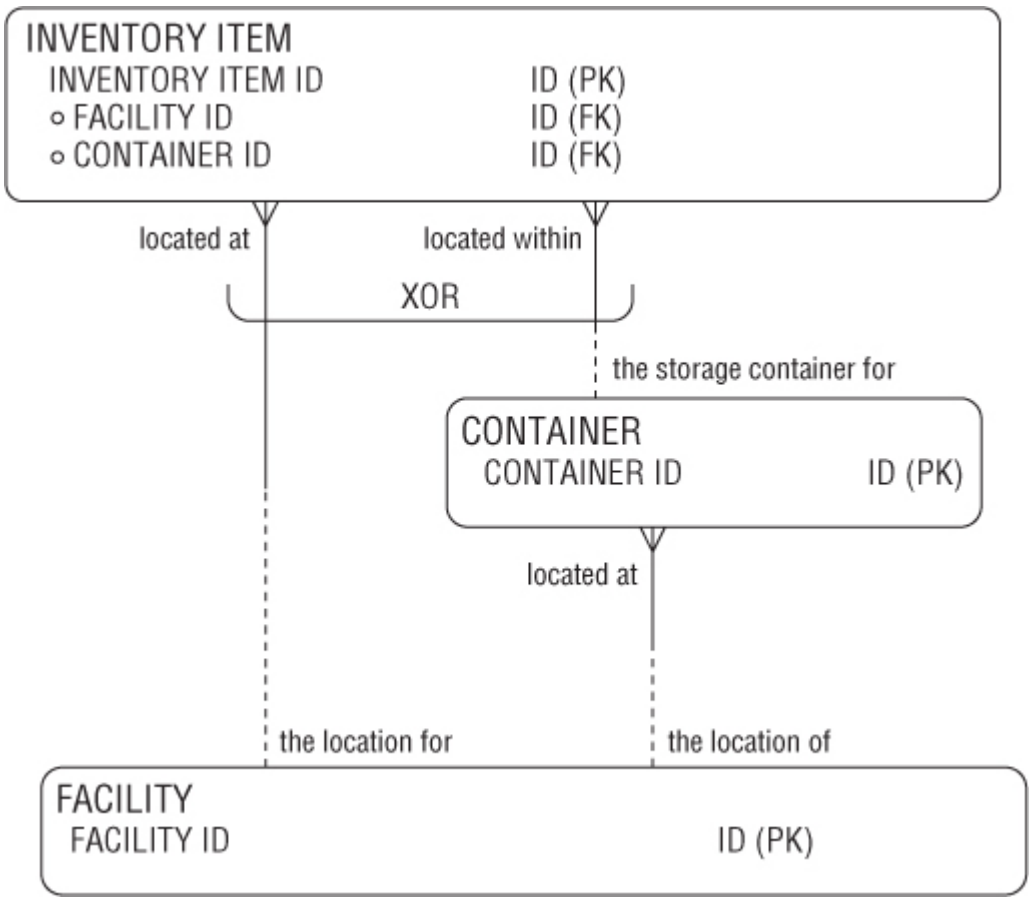
mechanism (same email address) at two different points in time.

Notice that in all the examples given, each relationship has two relationship names associated with it that describe the relationship in both directions. The relationship names should be used so that they read as a complete sentence, as shown in the following format: “Each ENTITY [*must be/may be*] **relationship name** [one and only one/one or more] ENTITY, (over time),” where the choices that are shown in brackets are filled in: for example, “Each PARTY *may be* **contacted via** *one or more* PARTY CONTACT MECHANISM(s) over time.”

Exclusive Arcs

Exclusive arcs are used to identify relationships where an entity is related to two or more other entities, but only one relationship can exist for a specific entity occurrence. The exclusive arc is represented by a curved line going through two or more relationship lines and an “XOR” symbol on the line that connects the relationships. Figure 1.8 shows an example of an exclusive arc. The relationships are read as “Each INVENTORY ITEM *must be* either **located at** one and only FACILITY or *must be* **located within** one and only one CONTAINER, but not both.” This communicates that inventory items are stored at one of two types of levels: They are either located at facilities such as a warehouse or stored within containers such as a bin that is located within a facility.

Figure 1.8 Exclusive Arcs



Example Data in Illustration Tables

Many parts of the data models are illustrated via tables that contain possible values for attributes. Each illustration table is normally defined to show specific worked examples, or in other words, we illustrate the pattern by showing instances of the entities and attributes to provide examples of how a pattern may look when populated. Often one illustration table is not enough and the pattern needs to be explained by two or more illustration tables.

A table illustrating the ORDER ITEM entity is shown in Table 1.4. In order to illustrate the details of an entity, the table may show information from directly related entities. For example, Table 1.4 brings in some attribute information from the ORDER entity even though the illustration table is

primarily used to illustrate instances within the ORDER ITEM entity.

Table 1.4 Order Item

ORDER. ORDER ID	ORDER. ORDER DATE	ORDER ITEM. ORDER ITEM ID	ORDER ITEM. QUANTITY	ORDER ITEM. UNIT PRICE (CURRENCY TYPE)
12930	April 30, 1995	1	120	200 (US Dollars)
12930	April 30, 1995	2	260	100 (British Pounds)

Notice that the entity name is followed by a ".", which is then followed by the attribute name so ORDER.ORDER ID is the column for the **order id** attribute within the ORDER entity. Whenever data from each illustration table is referenced in the text of this book, it is surrounded by double quotes. For instance, the text may refer to a specific order "12930," **order item id** "1," which has a **quantity** of "120" and a **unit price** of "200 (US Dollars)."

Sometimes, a parenthesis is used in a column of an illustration table for data that is closely related and adds context to the value. The last column is ORDER ITEM.UNIT PRICE (CURRENCY TYPE) and the value for the **unit price** in the first row is "200" and the CURRENCY TYPE for this value is "US Dollars" (thus, the price per unit is $200 US). Parentheses are also used to sometimes explain the nature of the column in the illustration tables. For example, when illustrating examples of a recursive relationship, there may be a parent PART and a child PART (showing that one part is made up of numerous other parts) with a recursive relationship around the PART entity. Thus there may be a column in an illustration table called "PART.PART ID (PARENT)" to illustrate that this is the column for the high level part, and "PART.PART ID (CHILD)" to illustrate that this is the column that represents the lower level parts, or in other words, the subcomponents.

Data Modeling Notation

As we have mentioned, we decided to use the Barker's Notation for this book,(13) and Figures 1.2 through 1.8 are examples of using this notation. Barker's Notation refers to the entity relationship diagram (ERD) notation developed by Richard Barker, Ian Palmer, Harry Ellis, et al., and its name came from its being made popular by Richard Barker. This notation is also sometimes referred to as Oracle Designer Notation, because it was used in the Oracle Designer data modeling tool.

Based upon feedback from previous volumes and based upon the types of models that we illustrate in this book, we slightly modified this notation with the intention of helping you, the reader. Thus, we follow Barker's Notation with the exception of the following enhancements and changes:

- The data models explicitly show primary and foreign keys as (PK) and (FK) next to the attributes for your convenience.
- The data models show the data types of the attributes, for example, char, number, and so on, for your reference.
- The "....." is used to indicate when there may be additional attributes that are not shown in the pattern because they were deemed to be not germane to the understanding of the pattern.
- We do not follow the rule that crowsfeet in diagrams must always be pointed from right to left and bottom to top because we find that people can more easily read the diagrams by laying out the relationships in a way that makes the most sense for that data model.

There are many different data modeling notations, and we spent a fair bit of time and effort considering which one of these would be best for this book and our readers. For example, aside from the data modeling notation that we

picked, some of the more popular notations for data modeling include:

- Information Engineering
- IDEF1X
- Unified Modeling Language (UML)

Additionally, there are many other data modeling notations that include the Chen notation, Object Description Language (ODL), and Object-Role Modeling (ORL), but in our experiences, these are less commonly used notations in data modeling at the time of the writing of this book.

In order to give you a better idea of what these modeling notations look like and to make the point that the same data modeling construct may look different depending on the data modeling tool that is used, this next section shows examples of the same data modeling construct, namely the data model shown in Figure 1.7, using different notations.

Figure 1.9 shows the how the data model from Figure 1.7 is modeled using a particular data modeling tool (Computer Associates ERwin tool) using Information Engineering notation.

Figure 1.9 Contact Mechanism Pattern in ERwin from Computer Associates

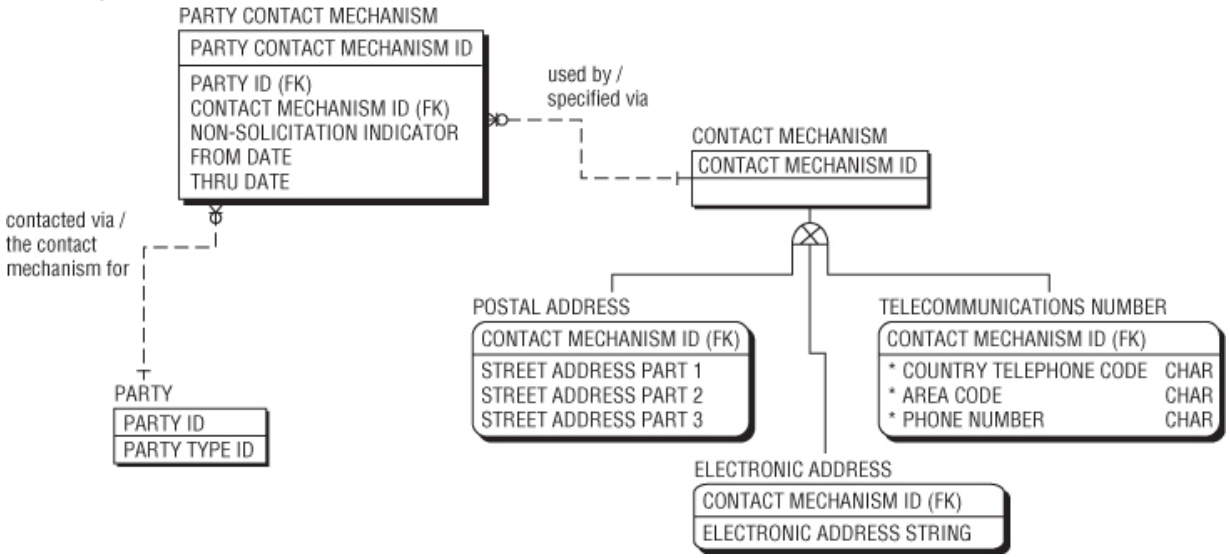
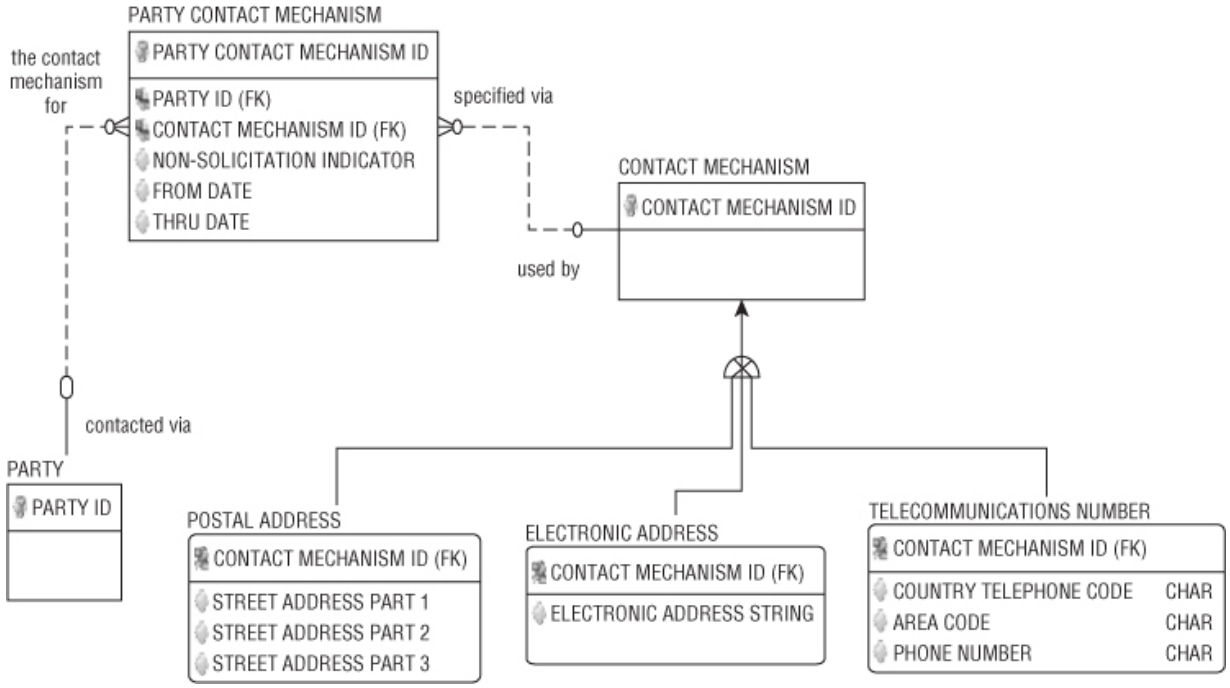


Figure 1.10 shows the same data model also using the Information Engineering notation, however, this time using a different tool, namely Embarcadero's ER/Studio data modeling tool.

Figure 1.10 Contact Mechanism Pattern in ER/Studio from Embarcadero



IDEF1X (Integration Definition for Information Modeling) was introduced as a Federal Information Processing Standard in 1993 and is widely used in the federal sector. “IDEF1X is a method for designing relational databases with a syntax designed to support the semantic constructs necessary in developing a conceptual schema.”(14) Figure 1.11 shows the same data model using IDEF1X notation.

Figure 1.11 Contact Mechanism Pattern shown using IDEF1X notation

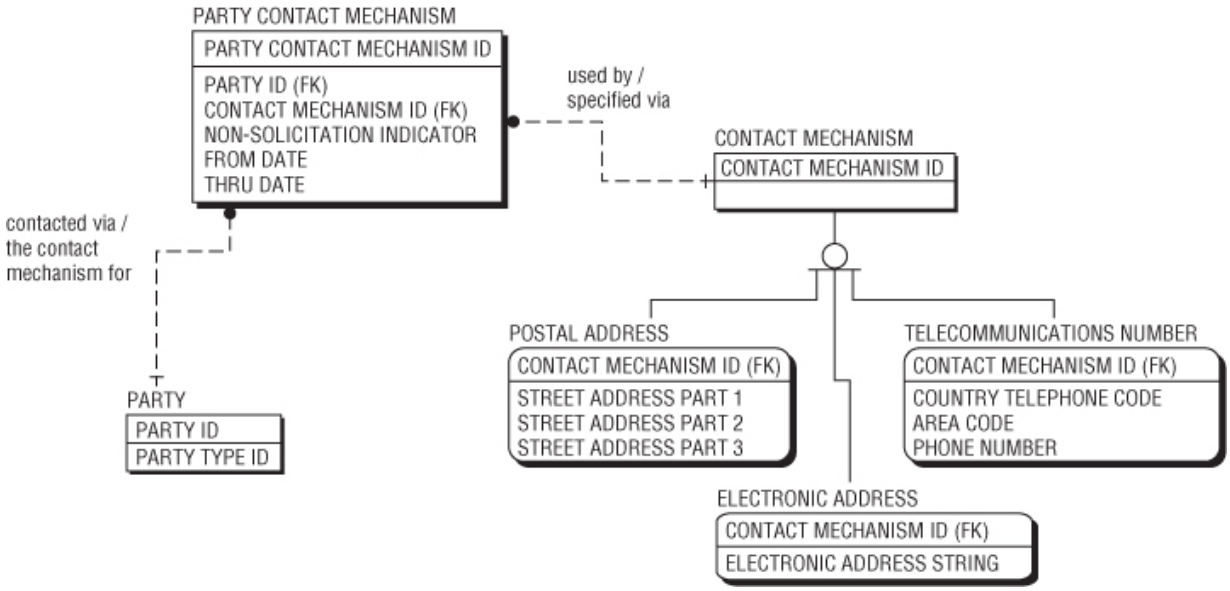
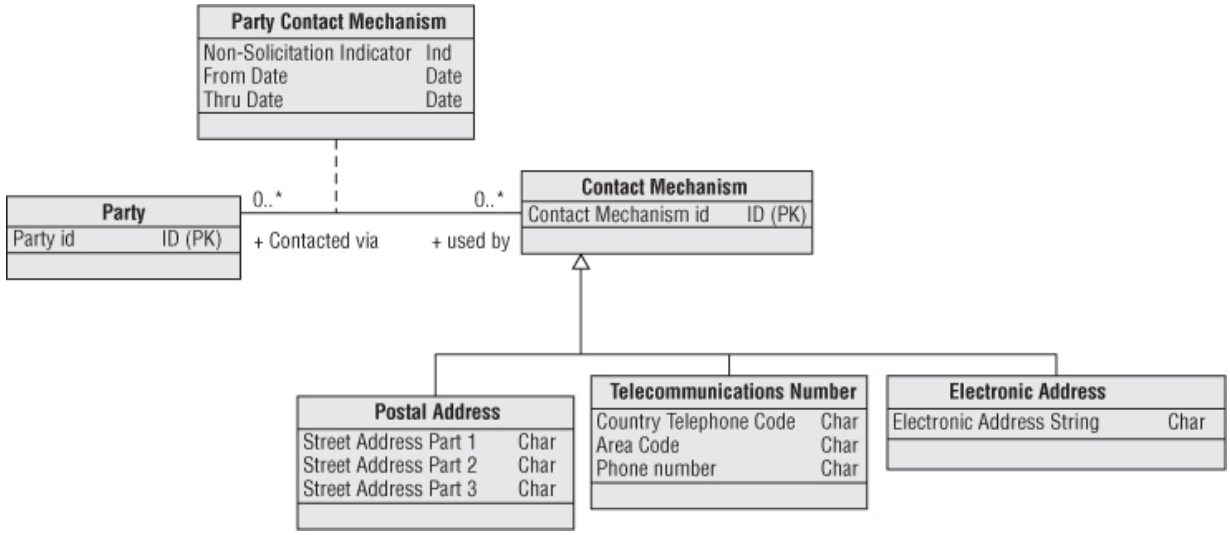


Figure 1.12 shows the same pattern using Unified Modeling Language (UML) notation.

Figure 1.12 Contact Mechanism Pattern shown using UML



We believe that there are pros and cons to each of these data modeling notations and that one is not superior to another notation. If you would like to learn more about the pros and cons regarding these various data modeling techniques, you can refer to David C. Hay's paper “A Comparison of Data Modeling Techniques”(15) for explanations about each of these modeling notations. We

have chosen to use Barker's Notation for the following reasons:

- The notation allows boxes within boxes for subtypes. Thus, subtypes can be displayed on the pages more concisely and elegantly (in our opinion), with less lines, and with more intuitive understanding (in our opinion).
- The convention regarding relationship names translating directly into sentences allows readers to better understand the relationship meanings (in our opinion).
- This notation uses the crowsfoot notation, which is a very commonly understood convention by most data modelers.
- This notation was used in *The Data Model Resource Book*, Volumes 1 and 2, and thus, we are keeping consistent with the previous volumes.

We have included this section to highlight that there are many data modeling notations and what our reasons are for choosing a particular notation. However, the patterns are applicable and will work well for all the aforementioned notations (or any other notation).

We have used the patterns in this book on many engagements with clients that have used many different data modeling notations, and we find that it is very easy and straightforward to translate the patterns from the data modeling notation in this book to any other data modeling notation. Thus, it has been our experience that data modelers and data professionals can very easily follow and use the patterns in this book regardless of the data modeling notation that they are currently using.

Summary

In this introduction our intention was to address the concepts underpinning this book. These include the need for this book, how these patterns enhance the discipline of data modeling, the definition of patterns and universal patterns,

the significance of patterns, the approach of this book, the levels of generalization used to provide alternatives, the audience for this book, a summary of what is in this book, and data modeling conventions used in the book.

What makes this book unique and important is that we are sharing fundamental patterns that can be used as crucial building blocks for most data models. We believe that these patterns and the alternatives offered in this book provide an invaluable tool for producing higher-quality data models in a much shorter time frame.

References

[1] P.P-S. Chen, ‘ The Entity Relationship Model-Towards a Unified View of Data’, ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, pp. 9-36. To see information on this article refer to Dr. Peter Chen's web page at Louisiana State University at http://www.csc.lsu.edu/~chen/chen.html.

[2] A Pattern Language: Towns, Buildings, Construction by Christopher Alexander, Sarah Ishikawa, and Murray Silverstein (Center For Environmental Structure Series, Oxford University Press, 1977).

[3] Design Patterns: Elements of Reusable Object Orientated Software by Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides (Addison Wesley, 1994).

[4] The Data Model Resource Book, Volumes 1 and 2, Revised Edition, by Len Silverston (Wiley, 2001).

[5] Data Model Patterns: Conventions of Thought by David Hay (Dorset House Publishing, 1995).

[6] A definition of *pattern* taken from Wordweb. Accessed at www.wordwebonline.com.

[7] The idea regarding usage of the term *generalization* versus *abstraction* came from comments and suggestions made to us by Karen Lopez, karenlopez@infoadvisors.com.

[8] A Framework for Information Systems Architecture by John A. Zachman, IBM Systems Journal, Vol. 26, No. 3 (1987). Also see http://www.zifa.com/ for diagrams and explanations of the Zachman Architecture.

[9] Data Modeling Theory and Practice by Graeme Simsion Dr. (Technics Publications, 2007).

[10] Verelst, J. (2004) Variability in Conceptual Modeling, University of Antwerp.

[11] At the time of this writing this presentation was available at http://www.infoadvisors.com/ArticlesVideos/InfoAdvisorsPresentations.aspx.

[12] For an alternative evaluation method see “What Makes a Good Data Model? Evaluating the Quality of Entity Relationship Models” by Daniel L. Moody and Graeme G. Shanks in Entity-Relationship Approach—ER ´94: Business Modelling and Re-Engineering, edited by P. Loucopoulos (Berlin/Heidelberg: Springer, 1994).

[13] Case*Method: Entity Relationship Modelling by Richard Barker (Addison-Wesley Professional, 1990).

[14] From the IDEF Data Modeling Method Overview at http://www.idef.com/IDEF1X.html.

[15] “A Comparison of Data Modeling Techniques” by David C. Hay, published in The Database Newsletter, Volume 23, Number 3, May/June, 1995 and updated August, 1999.

Chapter 2

Setting Up Roles: What Parties Do

Enterprises increasingly wish to view their business from an enterprise-wide perspective. One very important way to help with this goal is to understand the relationships that enterprises have with the people and organizations with which they do business.

A person or an organization may play any number of roles, such as a customer, supplier, worker, employee, contractor, or partner. Enterprises track this personal and organizational information in many different applications across the whole of their businesses. Different applications view the person or organization as a customer, partner, or supplier depending how the person or organization is involved in the business life cycle. Capturing roles within the context of discrete transactions or business processes is very important. For example, how is a person involved in a sales transaction—as the customer, as the salesperson? But, what about the roles that are not declared within these limited bounds? What are the roles that can be declared for a person or organization independent of any specific event, transaction, or business process? What are the roles a person or organization plays within terms of the enterprise as a whole?

For example, we declare a party (person or organization) to be playing the role of “customer.” This is one of the ways the enterprise views that party. Salespeople say, “John Smith is a customer of ours.” They don't refer to any particular sale. What they are saying is that a party is a

“customer” in general, or more accurately a party is considered a “customer” to the enterprise as a whole. This is a *declarative role*. However, the people in the shipping department may refer to the same party but within the context of some transaction. They may say “John Smith is the ship-to customer for shipment XYZ.” The “ship-to customer” is within the context of a shipment. This is what we call a *contextual role*. In this chapter we explain the patterns that support the first type of role, that is, the declarative role. We explain the patterns for the contextual roles in Chapter 3. Another way to think of this is that declarative roles involve setting up the role (thus, declaring it), and contextual roles involve using the role (within the context of another entity).

An enterprise would be partially blind if it only viewed parties only as a single role, within the context of a particular process, transaction, or event. For example, blind spots could occur if a particular party (person or organization) was playing a role as a customer and there was a separate entity for this without considering that this party played other roles as well, for example, a supplier and partner. There is a need to define roles that are declared within the context of the enterprise as a whole. This chapter addresses this important issue.

What Is the Significance of This Type of Pattern?

Who are the people and organizations of interest to our enterprise, and how are they related to our enterprise as a whole? These are two fundamental questions that enterprises ask. Senior management in many enterprises wants to manage the relationships an enterprise has with its

core clients to see how it treats people and organizations throughout the business life cycle as a whole.

It is important to capture accurately how persons or organizations may interact, in many different ways, with an enterprise. What are the parts they play, and how do the persons or organizations perceive their roles in general? What is the fundamental information that needs to be captured about these people or organizations? In other words:

- What are the attributes or characteristics of people and organizations that are involved in the course of conducting business?(1)
- What defines the relationships that these people and organizations have to an enterprise in general? What roles do people and organizations play in the course of conducting business?
- Are certain roles only suitable for organizations and not people and vice versa; what roles can both play?
- Are there any categories of roles that need to be managed?
- Can people and organizations play many roles while conducting business?

What Is in This Chapter?

This chapter initially defines what a declarative role is. The chapter then describes the data model patterns used to support the need for any enterprise to model what people and organizations do in terms of their complete business. The chapter describes how each pattern supports the attributes that a particular role needs and the common attributes that all people and organizations have, irrespective of the role they play. The chapter goes on to describe how each pattern may support the demarcation of roles for people and organizations if it is applicable. The

chapter also describes how each pattern handles the situation when the same person or organization plays more than one role.

Like most of the chapters in this book, the style of modeling for each of the patterns starts with the most specific style (Level 1 Declarative Role Pattern) and progresses through the chapter to a more flexible style (Level 3 Declarative Role Pattern). The different levels of generalization may be applicable to different enterprises or styles of modeling.

In summary this chapter includes the following:

- The definition of a declarative role
- The different levels of declarative role patterns
- When to use and not to use different declarative role patterns
- Insights into each pattern
- The relevance of each pattern
- A synopsis of each of the patterns, pros and cons, when to use and not use them

What Is a Declarative Role?

Roles represent the part that an organization or person plays within the context of a particular enterprise. There are two types of roles: declarative and contextual. This chapter deals with the declarative roles. Declarative roles “declare” that a party is playing a particular role such as a person that is identified and declared as an “employee” or an organization that plays the role of “supplier.”

Note

Chapter 3 deals specifically with the other type of role, that is, the contextual roles. In other words, the roles that a person or organization plays within the context of another specific entity

such as an ORDER, SHIPMENT, INVOICE, WORK EFFORT, and so on.

Declarative roles can be defined as “the stated actions and activities assigned to or required of a person or an organization.”(2)

What does this mean in terms of our pattern? What is being modeled in this chapter is the setting in which a person or organization has actions or activities within the context of the enterprise as a whole. This means that a person or organization may be declared as being a “customer” or “supplier” in general. There may or may not be any transactions or other business information associated with that party, but the declarative role is not concerned with this. The declarative role states that the person or organization may have one or more declared roles within the context of the enterprise, not what they are doing within a specific business process or event. This is an important distinction.

Why is this distinction important? Consider this situation: when asked the question “who is this person” (what information do you have available on this person), how does an enterprise respond? The first person operating the web site may say, “It's easy! This person is the guy that visits our web site late in the evenings.” The second person operating the accounting system may say, “This is a person responsible for paying bills.” The third person operating the human resources system may say, “This person is our employee.” The fourth person operating the customer relationship management system may say, “What? This is one of our customers that we service!” If you were able to see a complete profile for people, you would certainly know more about them and be able to service them better.

Note

> Some data modelers may say that you need the context in order to state a declarative role. For example, you only know if someone is a customer if there is an associated order, and you only know if someone is an employee if there is an associated employment agreement. A different perspective is, in many cases you have a need to “declare” that someone plays a role, and you may not maintain the entities that define the context (for example, you may not need to maintain the employment agreement in the data model). Furthermore, there are times when you just need to declare the role without the context, such as this person, in general, plays the role of “doctor” (without maintaining any associated transactions such as their certification document) or simply declare that this organization plays the role of “prospect” (without having to maintain that there was an associated lead).

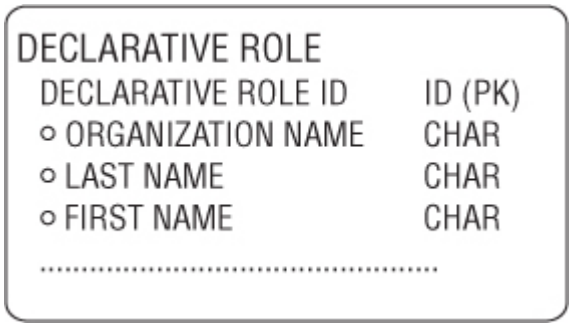
Level 1 Declarative Role Pattern

It is common in database designs (including operational data stores, data warehouse designs, and data marts), to show various types of roles individually. We have often seen separate, unrelated entities like WORKER, CUSTOMER, SUPPLIER, EMPLOYEE, and CONTRACTOR in many data model designs. Each separate role has a common set of attributes (and/or relationships) that need to be declared about a particular person or organization and that are common to all people and organizations, for example, the person's or the organization's name. They also may contain specific attributes for that role, such as a **credit limit** for a CUSTOMER or a **taxation identifier** for a SUPPLIER (as seen in Figure 2.2).

Mixing person and organization information with their role information has a very specific semantic meaning. By using this pattern an enterprise is saying that what a person or organization does is the same as who they are to the enterprise, that is, John Smith or XYZ Ltd is a client, is a supplier, and so on.

Figure 2.1 shows the Level 1 Declarative Role Pattern. Some data professionals may view this pattern as a “how-not-to-do” pattern because it can lead to redundant and inconsistent data if implemented. For example, if John Smith is a customer and a supplier, then you may have his name, date of birth, Social Security number, and everything else stored in two places. In other words, this pattern may capture the same party in two different places. This conclusion is correct if you look at this pattern from the perspective of relational modeling. There can be redundant data if declarative roles are implemented this way. If your enterprise has a shared vision that each party plays numerous roles and it is critical to maintain each party's information once and only once, regardless of the number of roles they play, this pattern may not be for you.

Figure 2.1 Level 1 Declarative Role Pattern



It is also fair to say that dismissing this pattern as “wrong” out of hand ignores some positive traits of this approach. For example, one purpose of this pattern is to show an understandable way to portray data requirements, present scope, and to begin to capture the common terminology for a subject area. It can be a powerful tool for communication with different interested parties. Also, it may help to kick-start the data modeling effort, because it is easy for business people and non–data professionals to understand.

This type of pattern is how the business often views parties; they may not understand or even want the party concept we show in the later patterns. Trying to force a

party concept on the business that does not want or understand it normally fails in our experience. We often use this pattern as the first step in showing the value of the party concept. This pattern illustrates the difficulties with not having a PARTY entity. Under these circumstances this model is a valid tool for data professionals. The data professional should understand its strengths and weaknesses before using it.

Another point is that even though the party models discussed later in this chapter represent a powerful way to model these concepts, some enterprises may be so entrenched in their systems that they will only view that each role is independent and should be its own separate entity (and table). For example, if an enterprise's view is that there should be a “Customer” entity and a “Supplier” and that not only are these roles very different, but that their systems work this way in mind, then even if the same organization is both a customer and a supplier, the systems in the enterprise see these as two independent facts. As Karen Lopez, a noted data management industry leader, said, “Just because you model something in a certain way, doesn't change the business to be that way!”

> **Note**
>
> The party concept is used throughout the Universal Data Models and is discussed in great detail in Volume 1 of *The Data Model Resource Book*. Basically, a party is either a person or an organization (and in some cases perhaps an automated agent(3) or an animal), but just one of these, and each party may play many roles such as customer, partner, supplier, and so on.

We have often seen physical implementations of this type of pattern, with various degrees of success. For example, this pattern works well when implemented as an intermediary database in extract, transform, and load (ETL)

systems or as dimensions in star schemas. But when implemented in a relational database, this pattern can lead to some very undesirable outcomes, for example, creating redundant data and not being able to see the whole picture regarding a party.

Why Do We Need This Pattern?

The purpose of the Level 1 Declarative Pattern is to show a very simple model that can be easily understood by explicitly defining all the different roles as specific entities. Each of the different entities should provide a clear and unambiguous description of the roles for a particular subject area. This pattern also provides a good view of the scope of all the roles in a subject area. Each of the different declarative role entities may contain all of the attributes common to each role and specific attributes for that particular role.

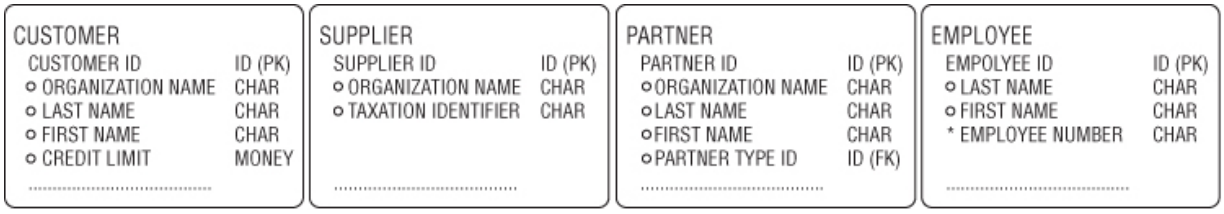
This pattern should also support the basic aspects of a declarative role. First, the definition or representation of the role itself; second, the relationship that these roles have to the enterprise in general; and third, what roles are organizational roles, what roles are people roles, and what roles support both. Finally, the pattern may support the ability of a person or organization to play one or more roles at the same time. When the same person or organization plays more than one role, that person or organization may be maintained in each of the separate role entities. However, details such as name, Social Security number, and date of birth may be repeated in each role. It is important to capture this information requirement, because it helps you understand what attributes are important to each role and what attributes are important to all roles. For example, you may need Social Security number for the role of employee, but not in any other role. This pattern allows us to communicate this fact more clearly.

How Does This Pattern Work?

Figure 2.1 illustrates how an entity is used to describe the declarative role. DECLARATIVE ROLE is the role entity that a data professional is interested in. It would contain the definition of the role and all of the relevant attributes for that role. Though there may be many names for the person or organization playing this role, thus implying a relationship to a NAME entity, from a practical viewpoint we feel that declarative roles under most, if not all, circumstances will have one or more name attributes (e.g., **first name**, **last name** for people or just **organization name** for organizations) to show the primary or current name in this pattern. Name attributes are the designation used to identify the current name of the person or organization playing the role. The DECLARATIVE ROLE entity may have many more generic and specific attributes that further describe and characterize the role.

Figure 2.2 further illustrates how this pattern may support many different types of roles. The scenario is as follows: the CEO of an office-supplies company (Pen Pushers Ltd) characterized their enterprise as a company whose highest priority is in building relationships. As a start in supporting this vision of the firm, the CIO wished to capture all the different roles that the firm considered significant in managing its business. A data professional was employed to examine all the different roles that the office-supplies company declared as critical to its business.

Figure 2.2 Example of using a Level 1 Declarative Role Pattern



After some detailed analysis of business, and while interviewing senior and middle management, the data professional developed an initial model based on the Level 1 Declarative Role Pattern. Figure 2.2 contains the four significant roles that the office-supplies company wished to capture: CUSTOMER, SUPPLIER, PARTNER, and EMPLOYEE. These are the roles that the company wished to maintain regardless of the specific business activities that the firm carried out.

The CUSTOMER role can be defined as a person or organization "that has purchased, been shipped, or used products (either goods or services) from the enterprise. An organization or person may play the role of a customer."(4) The CUSTOMER role entity has different name attributes for organizations and people, that is, **organization name** or **last name, first name**, respectively. CUSTOMER also contains a **credit limit** attribute. The name attributes are used in referring to the CUSTOMER. The **credit limit** is the maximum amount credit that can be extended to a customer, for purchasing goods or services.

In Table 2.1 you see four different CUSTOMER instances, with four different name(s): "Matrix Ltd," "Kantowitz Services," "Card Queen Ltd," and "Caroline Percy" with **credit limit(s)** of "HK$100,000," "$90,000," "$1,000," and "£3,445," respectively. This means that "Matrix Ltd" has an amount of credit under which at any point in time it can owe up to "HK$100,000" worth of office supplies to Pen Pushers Ltd. It is interesting to note that the CUSTOMER entity contains both organizations and people. For example, "Caroline Percy" is a person; all the other instances are organizations. This is normal for a customer role, because customers can often be people or organizations.

Table 2.1 Example of Using a Level 1 Declarative Role Pattern, Customer

CUSTOMER. CUSTOMER ID	CUSTOMER. ORGANIZATION NAME/FIRST NAME, LAST NAME	CUSTOMER. CREDIT LIMIT
101	Matrix Ltd	HK$100,000
104	Kantowitz Services	$90,000
108	Card Queen Ltd	$1,000
107	Caroline Percy	£3,445

The SUPPLIER declarative role entity can be defined as an organization "that has been identified as a potential, current, or future organization that supplies products to the enterprise."(4) The SUPPLIER role entity again contains **organization name**, as did the CUSTOMER entity. SUPPLIER also contains the **taxation identifier**, which can be defined as "a number or string that is used as a way to uniquely recognize and/or distinguish an organization for tax purposes. Sometimes this is referred to as a Tax Identification Number."(4) For example, in Table 2.2, you see three suppliers, "Matrix Ltd," "Card Queen Ltd," and "Kantowitz Services," each of which has their own taxation identifier.

Note

Suppliers are usually organizations. However, sometimes they may be people. For example, a supplier may be a person that supplies training to his or her clients. We make the simplifying assumption in our examples that SUPPLIER(s) can only ever be organizations, which may be true for some enterprises.

Table 2.2 Example of Using a Level 1 Declarative Role Pattern, Supplier

SUPPLIER. SUPPLIER ID	SUPPLIER. ORGANIZATION NAME	SUPPLIER. TAXATION IDENTIFIER
5632	Matrix Ltd	8567891adc456
5634	Card Queen Ltd	5667840978
6700	Kantowitz Services	55 5555 555

The PARTNER declarative role entity “is identified as an ally and with whom mutually beneficial relationships are established.”(4) This definition does not say if an instance of PARTNER must be a person or an organization. If you look at Table 2.3, the instances of PARTNER happen to be organizations, but the definition does not preclude a person from being a PARTNER.

Table 2.3 Example of Using a Level 1 Declarative Role Pattern, Partner

PARTNER. PARTNER ID	PARTNER. ORGANIZATION NAME	PARTNER. PARTNER TYPE ID (PARTNER TYPE NAME)
89	Kantowitz Services	10 (Global Partner)
90	Card Queen Ltd	10 (Global Partner)
100	Matrix Ltd	20 (Asia Partner)

The PARTNER role entity has a name (**organization name (**or **first name, last name)**) that is used to identify the organization (or person) playing that role. The PARTNER entity also has a **partner type id** attribute, which is a foreign key to a partner type entity not shown in the diagram that lists all of the allowable partner types for an instance of PARTNER. You can see from Table 2.3 that “Kantowitz Services” is a “Global Partner” and “Matrix Ltd” is an “Asia Partner.” It is important to note that declarative roles often don't just contain specific attributes but may also have specific relationships that other declarative roles may not have.

Finally, you have the EMPLOYEE role. This is a person-only role. The EMPLOYEE can be defined as “a person working for and hired by an organization for pay and that is not providing their services via an independent business.”(4) The entity has the attribute **employee number** that identifies the EMPLOYEE. EMPLOYEE also contains the **last name** and **first name** for an employee. In Table 2.4 you see that “Rob Gardner,” “Francis Pope,” and “Caroline Percy” are all employees of Pen Pushers Ltd. It is worth noting that

“Caroline Percy” is also a customer of Pen Pushers Ltd, as you saw in Table 2.1.

Table 2.4 Example of Using a Level 1 Declarative Role Pattern, Employee

EMPLOYEE. EMPLOYEE ID	EMPLOYEE. FIRST NAME, LAST NAME	EMPLOYEE. EMPLOYEE NUMBER
4	Rob Gardner	030031
5	Francis Pope	030025
8	Caroline Percy	030077

These examples illustrate some basic features of the Level 1 Declarative Role Pattern.

- First, each of the entities represents a clearly defined role, and each has a clear definition.
- Second, each of the role entities may have attributes such as an **organization name** or **first name, last name** that may be shared among the roles. Also, each of the role entities has specific attributes that apply only to that role. For example, EMPLOYEE has an **employee number**, SUPPLIER has a **taxation identifier**, and so on.
- Third, you see from the instances described in the tables that some persons and organizations play more than one role. “Caroline Percy” is an instance of EMPLOYEE and CUSTOMER, “Matrix Ltd” is a CUSTOMER, PARTNER, and SUPPLIER.
- Finally, you see that some of the roles may be specific to people or organizations. For example, the EMPLOYEE role was only for a person.

Also note that each of the different declarative role instances is implicitly classified. What we mean by this is that any person or organization that is captured as an instance of CUSTOMER, for example, is playing a CUSTOMER “type” declarative role. In other words, “Caroline Percy” and

“Card Queen Ltd”, both may be classified as a customer of Pen Pushers Ltd, as you saw in Table 2.1.

When Should This Pattern Be Used?

We use this data model pattern when:

- **A well-defined set of declarative roles is identified and known by the business** The area under investigation is well known by either senior management or business analysts, who are clear on what roles they consider to be very important. In our example, while interviewing senior and middle management, the data professional identified CUSTOMER, SUPPLIER, PARTNER, and EMPLOYEE as significant declarative roles.
- **An enterprise subscribes to the view that a person or organization is synonymous with what each does or how each acts** For example, if a person in a medical environment is a patient and there is information about this person as a patient, then the person is defined as being a patient, despite the idea that he or she may also play other roles, such as doctor, pharmacist, and is a person. Many doctors think about and refer to the people coming in for treatment as their patients, not about people playing the roles of patients. To a doctor, a patient is synonymous with a person.
- **An enterprise is highly resistant to the party concept** We get asked, from time to time, by our clients to remove PARTY from our models because they believe that it confuses the audience.
- **The data professional needs to describe the business requirements in a simple way and perhaps show a simple, unambiguous diagram to other IT professionals or management** The example shown in this section of the chapter was a straightforward statement of information requirements

(as part of a statement of scope) showing the four different key roles that were of interest to the business and when it was important to illustrate and show different perspectives. We often show level 1 patterns as part of an initial statement of scope.

- **The roles in the business are static and don't change** We have seen implementations like this where the roles were well defined and hardly ever changed, for example, in a business that contained only a few key roles that were very stable. (However, be very careful because we have also seen major problems where the designers thought roles would not change and later on they did.)
- **Each person or organization plays one and only one role, and this is true at the same time (the person or organization does not play more than one role at the same point in time) or over time (the person does not play one role and then switch to playing another role at a later point in time)** If this is truly the case, there is less of an argument that there will be redundant data with this model. (However, this is rarely the case in our experience.)

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern would not be suitable in a dynamic environment where new role types are added frequently, or if the roles are not well understood** —for example, if the data professional was not sure whether he or she had captured all the roles. The roles would have to be updated with the new attributes if they are not well understood.

- **These roles also repeat the same information**, such as the attributes **organization name, last name**, and **first name**, across many different roles. For example, in Tables 2.1 and 2.4 you saw “Caroline Percy” can be duplicated as an EMPLOYEE and as a CUSTOMER. This can lead to two different issues. First, you could be storing information redundantly. Second, and more importantly, redundant information often gets out of sync; this may cause data quality issues—for example, storing “Caroline Percy” as the name CUSTOMER and maintaining “C Percy” in EMPLOYEE. Are they the same person?
- **With this model, it is very difficult to see the whole picture of a person or an organization**. For example, if an organization is both a customer and a supplier, this fact is overlooked. What if you were picking between three potential suppliers, but failed to see that one of those suppliers was actually your biggest customer? Another example is that if your employee was also your customer, you may want to know this fact in order to offer them an employee discount or it may be important to know for compliance or fraud reasons.

Synopsis

In this section we covered a very specific way to model declarative roles. Each role was captured as its own individual entity. For example, in Figure 2.2 you see CUSTOMER, SUPPLIER, PARTNER, and EMPLOYEE. Each entity contained specific attributes for the particular role (for example, **employee number** in EMPLOYEE); each entity also contained shared attributes (for example, **organization name** in CUSTOMER, SUPPLIER, and PARTNER).

Why create a specific model for declarative roles? This is often the first model we draw for our clients. This pattern is helpful for several reasons. It is very easy for business people and nontechnical people to understand. It is a simple statement of data requirements. It is not cluttered with technical entities such as PARTY that may confuse your audience. The chances are that legacy systems in your organization model declarative roles in this way. Finally, people and organizations in the enterprise may only ever play just one role and may not have multiple roles. For all of these reasons it is important to consider this modeling pattern and know the strengths and weaknesses of this pattern.

This section described how people and organizations can play more than one declarative role at the same time, although when this happens, this model maintains redundant data.

Some roles may be defined exclusively as declarative roles of people or organizations. For example, EMPLOYEE is a “person only” role and was defined as “a person working for and hired by an organization for pay and that is not providing their services via an independent business.”(4)

Within the context of a relational model, this pattern has many weaknesses. For example, this pattern may store redundant attributes. Data stored redundantly can quickly create future data quality issues if not managed correctly. The same person or organization may appear twice in two different declarative roles. How do you reconcile them? It is difficult to see the whole picture of a person or organization if they play many roles, and thus you are looking at pieces of the person's or organization's data in different entities. This pattern often helps us highlight the weaknesses of implementing this pattern to business and technical stakeholders.

Finally, this pattern is most suited for illustrating information requirements about declarative roles in a simple way. It is most useful in very well-defined, static environments. If roles change, then implementation of this pattern leads to a design that is not stable, because the underlying data structures will probably require change.

Level 2 Declarative Role Pattern

It is often desirable to model declarative roles with flexibility, but still maintain the specific nature of the pattern. By adding the PARTY entity to the pattern, attributes that are related to people (e.g., **first name, last name** and **date of birth**) and related to organizations (e.g., **organization name**), regardless of their role, can be captured in a PARTY entity. The role-specific information (e.g., **employee number** for EMPLOYEE**, credit limit** for CUSTOMER) may be maintained in specific declarative role entities. This allows you to see a more complete picture of a party because the same party is related to all their roles. Also, if it is assumed that a party can play more than one role at a time, you can reduce redundancy by maintaining a person's or organization's information only once and showing that the party may be playing many roles, either at the same time or over time.

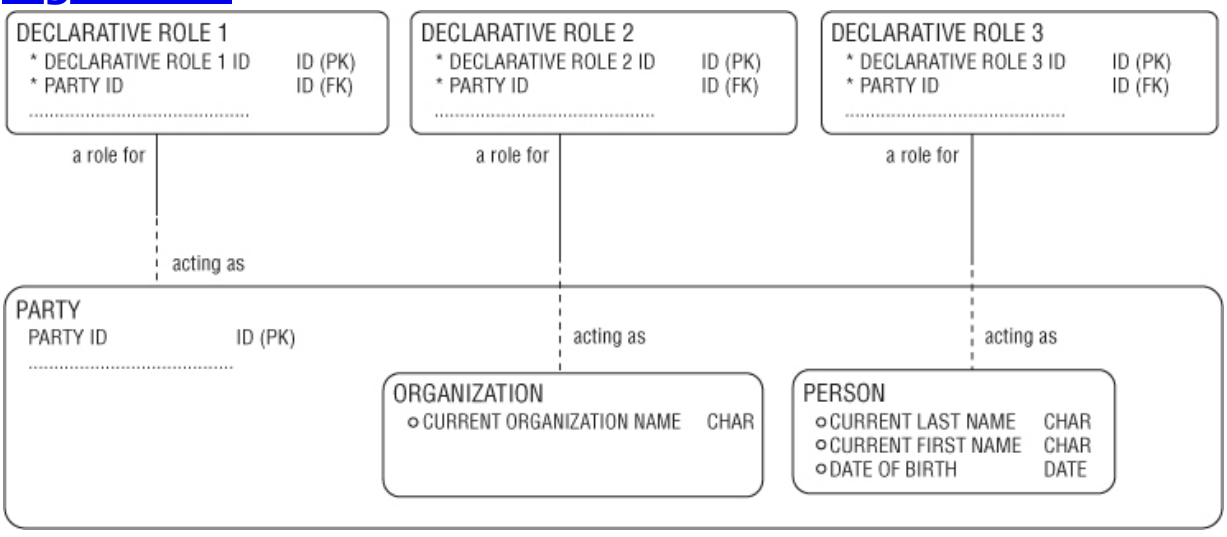
This pattern introduces a different semantic view of roles from the previous pattern. The previous pattern mixed person, organization, and role information together in the declarative role entities. Some enterprises view a party and what role it plays as indistinguishable. This pattern suggests a different semantic view, namely that a party is a person or an organization, and that the party may play one or more different declarative roles. Both semantic viewpoints have merit; neither is completely correct nor completely incorrect. This pattern provides an alternate perspective and

as opposed to “you are what you do,” it offers a different semantic view, namely, “a person or an organization is a party, and parties may play various roles.” Thus, there is information about the person or organization, and the person or organization may play one or more roles such as being a client, employee, partner, and so on.

Why Do We Need This Pattern?

This pattern eliminates the need to capture redundant person or organization data in many different individual declarative roles; in other words, you don't capture the name (or any other attribute) of the party in many different places. Secondly, by having PARTY subtypes of ORGANIZATION and PERSON entities (see Figure 2.3) to which you can relate organization and person declarative roles, you maintain demarcation of organization-only and person-only declarative roles in the data model. In other words, person declarative roles get related only to PERSON and organization roles get related to ORGANIZATION. Roles that can be applied to both people and organizations can be related to PARTY.

Figure 2.3 Level 2 Declarative Role Pattern



Although this pattern is a little more abstract than the level 1 pattern, it can still be used as a part of a statement of scope and help document requirements about the various roles that exist. Data architects, IT professionals, or stake-holders will be able to relate to all the roles that are declared for a PARTY. These roles are still explicitly shown in their own entities, and this pattern has the advantage of being able to maintain the information about the party once and then maintain all the specific information about each role once.

How Does This Pattern Work?

In Figure 2.3 we show that PARTY contains both a PERSON and an ORGANIZATION as subtypes. A PERSON may be defined as "a physical human being, alive or dead." An ORGANIZATION can be considered as "a group of persons organized for a particular purpose; an association"(2) or "a structure through which individuals cooperate systematically to conduct business." The PERSON entity may maintain attributes such as **current first name**, **current last name**, **gender**, and **date of birth**. The ORGANIZATION may maintain attributes such as **current organization name** and so on.

This PARTY supertype allows you to maintain information that is common between a person and organization in one place. For example, people and organizations may both have common data that is related to them, such as objectives, contact information, and credit scores. But most importantly, there may be other entities that may relate to either a person or an organization. For example, by setting up a PARTY entity as this pattern does, this allows you to relate an ORDER to a PARTY instead of having to show that an order may be related to either a person or an organization. This could also be said about so many other

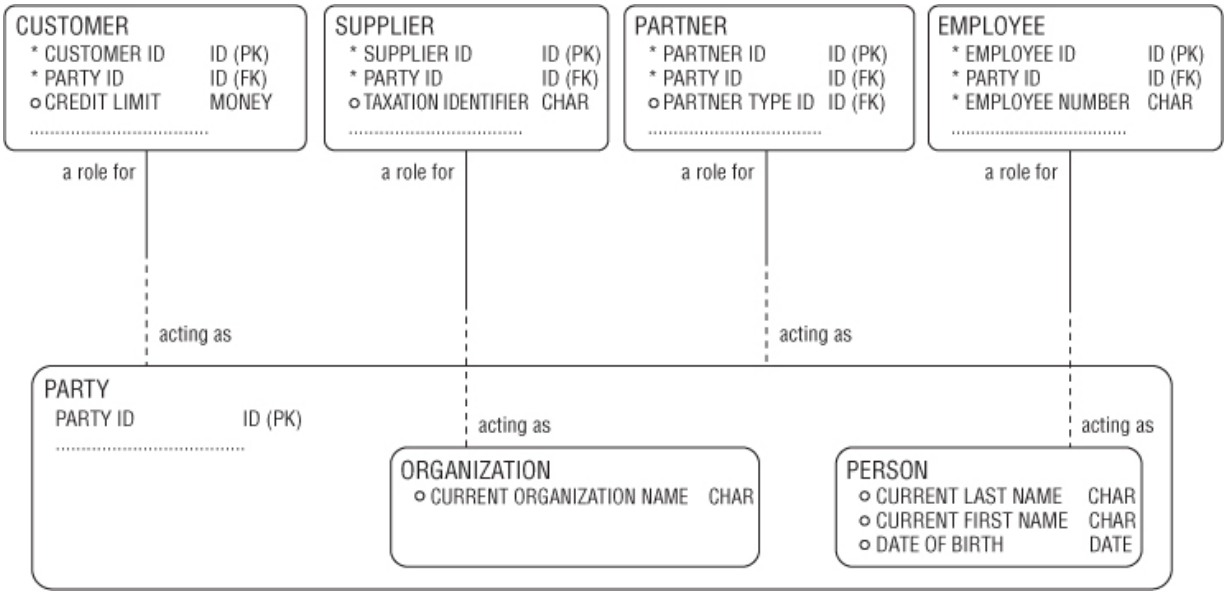
entities such as a SHIPMENT, INVOICE, WORK EFFORT, PAYMENT, CONTACT MECHANISM, and so on.

> **Note**
>
> A variation on this pattern is to show it without the supertype of PARTY and just relate the declarative roles to PERSON and/or ORGANIZATION. This could be used in situations when people and organizations do not share any common attributes and/or relationships.

The declarative role entities (DECLARATIVE ROLE 1, 2, 3) define how a PARTY, PERSON, or ORGANIZATION acts—in other words, what roles the PARTY plays in the enterprise as a whole. Each of the declarative roles must have a **party id** foreign key that is inherited from PARTY. For example, "each DECLARATIVE ROLE 1 *must be* **a role for** one and only one PARTY."

The party-specific attributes are captured in the PARTY, ORGANIZATION, and PERSON supertype subtype structure. Role-specific attributes are captured in their owning declarative role. Any specific one-to-one or one-to-many relationships to a declarative role may be captured via a foreign key in the declarative role to the related entity. Any many-to-many relationships to and from a declarative role may be captured in an associative entity between the declarative role and its related entity. For example, you see in Figure 2.4 that a foreign key **partner type id** in PARTNER specifically relates the PARTNER declarative role to a PARTNER TYPE classification entity.

Figure 2.4 Example of using a Level 2 Declarative Role Pattern



Some declarative roles are person specific, some are organization specific, and some are not specific to either a person or an organization. The business rule specifying if the role is specific to people or organizations is handled by creating the relationships from the subtypes PERSON and ORGANIZATION to their specific declarative roles. In Figure 2.3 you see that DECLARATIVE ROLE 1 is related to PARTY, thus allowing people or organizations to play this role, whereas DECLARATIVE ROLE 2 is a role for organizations only, and DECLARATIVE ROLE 3 is a role for persons only.

Note that the foreign key from PARTY is mandatory. What this implies is that an instance of a declarative role must be a party, that is, you can't have a declarative role instance without first having a PARTY. This is an important rule because it shows that having a role is dependent on first having a PARTY.

If we continue with the scenario of the office-supplies company described in the previous section, you can see that the data professional initially produced Figure 2.2 based on the Level 1 Declarative Role Pattern to show the initial scope of the roles in which the company was interested. Based on some further discussion with stakeholders the data

professional produced Figure 2.4 to show an alternative pattern that would reduce the need to capture redundant data by maintaining information about a PERSON or ORGANIZATION only once, even if they played many roles. Sometimes parties play many roles at the same time, for example, being a customer as well as a supplier, and sometimes parties play many roles over time, for example, when a person is first a customer and then later on in time becomes an employee (perhaps because they are so impressed with the products or services they bought).

In Figure 2.4 you see the PARTY, ORGANIZATION, PERSON subtype supertype structure that was described previously. You also see that the data professional captured the four important roles that Pen Pushers Ltd uses to manage the relationships in its business, that is, CUSTOMER, SUPPLIER, PARTNER, and EMPLOYEE. Each of these declarative roles was defined in the previous section.

This pattern enforces some very powerful concepts. You can see that a CUSTOMER is related to a PARTY. This relationship states that "each PARTY may be **acting as** one and only one CUSTOMER" and "each CUSTOMER *must be* **a role for** one and only one PARTY." Interesting business rules can be inferred from this. First, an instance of a CUSTOMER may be a PERSON or an ORGANIZATION, because both of these are subtypes of PARTY. This is borne out in Table 2.5.

Table 2.5 Example of Using a Level 2 Declarative Role Pattern

PARTY. PARTY ID	PERSON. CURRENT FIRST NAME, CURRENT LAST NAME	ORGANIZATION. CURRENT ORGANIZATION NAME	CUSTOMER. CREDIT LIMIT	SUPPLIER. TAXATION IDENTIFIER	PARTNER. PARTNER TYPE ID (PARTNER TYPE .NAME)	EMPLOYEE. EMPLOYEE NUMBER
1001		Matrix Ltd	HK$100,000			
1001		Matrix Ltd		2334-23		
1001		Matrix Ltd			20 (Asia Partner)	
1002		Card Queen Ltd	$1,000			
1002		Card Queen Ltd		566784-0978		
1002		Card Queen Ltd			10 (Global Partner)	
1003		Kantowitz Services	$90,000			
1003		Kantowitz Services		434232-2422		
1003		Kantowitz Services			10 (Global Partner)	
1004	Caroline Percy		£3,445			
1004	Caroline Percy					030077
1005	Rob Gardner					030031
1006	Francis Pope					030025
1008	Leon Brinkley				10 (Global Partner)	

In this table you see the PARTY, which is an ORGANIZATION, with **party id** "1001," with **current organization name** of "Matrix Ltd." This PARTY has a **credit limit** of "HK$100,000," which is an attribute associated with their role as a CUSTOMER. You also see a PARTY, that is a PERSON, with **party id** of "1004," and **current first name** and **current last name** of "Caroline Percy." This PARTY playing the role of CUSTOMER, has a **credit limit** of "£3,445." PARTNER is also a role that can be played by both a person and an organization. If you examine Table 2.5 you see that the ORGANIZATION(s) "Matrix Ltd," "Card Queen Ltd," and "Kantowitz Services" are PARTNER(s). Also the PERSON "Leon Brinkley" is a PARTNER, illustrating that a PARTNER declarative role is related to a PARTY and thus may be related to an organization or a person.

Note

If numerous types of currencies are needed, there would actually be a **currency type id** foreign key (that is related to a CURRENCY TYPE entity) in the CUSTOMER to signify what type of currency applies to the **credit limit**. We have not included this foreign key in this example, in order to keep our focus on illustrating the core aspects of this pattern.

The SUPPLIER role is related directly to ORGANIZATION. This relationship states that "each ORGANIZATION *may be* **acting as** one and only one SUPPLIER." This means that the SUPPLIER role is applicable only to a PARTY that is an ORGANIZATION. The SUPPLIER has an attribute of **taxation identifier**. Taxation identifiers can only be assigned to an organization playing the role of SUPPLIER. The relationship from SUPPLIER to ORGANIZATION enforces the rule that suppliers must be organizations and not people. In Table 2.5 you see that only "Matrix Ltd," "Card Queen Ltd," and

“Kantowitz Services” have **taxation identifiers**, and they play the declarative role of SUPPLIER.

> **Note**
>
> Often taxation identifiers are captured as an attribute of ORGANIZATION. In other words, an organization, irrespective of the role they play may have a taxation identifier. But, in our example Pen Pushers Ltd had a more specific rule that taxation identifiers where only captured for SUPPLIER(s).

Some declarative roles are related only to people — for example, as in Figure 2.4, EMPLOYEE. From Figure 2.4 you see that “each PERSON *may be* **acting as** one and only one EMPLOYEE.” As was stated in the previous section, Pen Pushers Ltd considers that an “employee” must be a human being (i.e., a person). In Table 2.5 there are three instances of PERSON that are considered EMPLOYEE(s): “Caroline Percy,” who also plays the role of CUSTOMER, “Rob Gardner,” and “Francis Pope.” Each of them has an **employee number**

This pattern also maintains that an instance of PARTY, PERSON, or ORGANIZATION may play one and only one of each of the declarative roles at a time. In this pattern we are showing a very specific case with a specific rule. In other words, a person can only be an employee once, or an organization can only play the role of supplier once. This may cause some data modelers a problem because this is a one-to-one relationship and a useful rule of thumb many modelers use is this: When you see a one-to-one relationship, you are in fact just seeing attributes of a single entity (CUSTOMER, SUPPLIER, PARTNER, or EMPLOYEE) split into two different entities (PARTY and the role).[6] We have used this rule ourselves many times.

Why split out the roles from PARTY? Although we have established in the level 1 pattern that it is not absolutely necessary to distinguish the party from its roles, and

although this is just another semantic view, there are advantages. For one thing, there is information about a party independent of the party's role, such as name, date of birth, and so on, as well as information specific to each role the party plays, such as taxation identifier (SUPPLIER), employee number (EMPLOYEE), and so on. Also, this pattern states that declarative roles are not parties, persons, or organizations. Declarative roles are what a party, person, or organization does! Additionally, by modeling it this way, you avoid storing redundant data. You can now maintain the role-independent data about a party (name, date of birth, and so on) only once and not repeat it for each role they play, as in the level 1 pattern. With this pattern you are also able to see a more complete picture by relating a party to all the roles they play and thus being able to see all the information associated with all their roles for the party.

In the same way as the Level 1 Declarative Roles Pattern, each of the different declarative roles is implicitly classified. What we mean by this is that any person or organization that is recorded as an instance of a specific role is classified by that role. For example, if a person or organization is acting as a CUSTOMER role, then they are playing a CUSTOMER “type” declarative role.

When Should This Pattern Be Used?

We use this data model pattern when:

- **There is common information that exists for people and organizations** For example, people and organizations may both be related to other entities such as ORDER, SHIPMENT, INVOICE, and so on. Thus, the PARTY entity adds value by simplifying the relationships from each of these entities to a single PARTY entity instead of needing to specify relationships to both PERSON and ORGANIZATION.

- **There are situations where the same party plays many roles** It is common that people and organizations play more than one role at a time. This occurs many times because a person or organization becomes better established within the enterprise and then acts in another capacity. For example, often customers become suppliers due to reciprocity; customers become partners because they see the value of the enterprise and want to be part of it; or contacts become employees or contractors. Thus, by modeling a person's or organization's information once, regardless of their roles, the enterprise is able to maintain much higher quality data.
- **There is a need to ensure that only certain roles are for people, that other roles are used by organizations, and that still other roles are used by both people and organizations** You can see from Figure 2.4 that an EMPLOYEE can only be a PERSON, a SUPPLIER can only be an ORGANIZATION, and a PARTNER or CUSTOMER can be either a PERSON or ORGANIZATION (since it related to PARTY).
- **There is a semantic need to distinguish roles from parties, people, and organizations** This pattern maintains a one-to-one relationship from PARTY (PERSON or ORGANIZATION) to the roles. This means that each PARTY may play declarative roles, not that a PARTY *is* a declarative role.
- **The enterprise is interested in using a pattern that works in many situations** We have seen successful implementations of this pattern “as is.” The pattern is simple enough as a design that programmers can quickly grasp the concepts and develop code based on the pattern.
- **There is a need to be somewhat specific about the individual roles in order to use a more**

understandable data model to gain buy-in We feel that this pattern is effective in showing that a subject area not only has a set of declarative roles, but also the relationship that the roles have to parties. This pattern can be a powerful addition to a statement of scope.

- **There is a need to see a more complete picture regarding all the information associated with a party** With this pattern, you can maintain an integrated view of all the information associated with the various roles for a party. It is often useful to know that an ORGANIZATION is playing the role of SUPPLIER as well as CUSTOMER. This may lead you to assign them to a PARTNER role also.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **Some enterprises don't feel that declarative roles are semantically different from parties** Doctors refer to people as patients, salespeople refer to an organization they are selling to as their customer. Some data model professionals believe that the roles that PARTY(s) play are in fact a subtype of the PARTY entity itself. Doing this implies that the roles of a PARTY are inherent and persistent in the definition of the PARTY. This pattern does not support this view because the pattern allows PARTY roles to change over time and the PARTY (and its associated data) is distinct from its related roles.
- **This pattern suggests that a party will play a specific type of role only once** In Figure 2.4 you see that each PARTY may be acting as one and only one PARTNER. Is this always true? We can conceive of some roles that a PARTY, PERSON, or ORGANIZATION may play

many times. For example, the same person may be a contact for more than one organization. Of course, this can be resolved by changing the cardinality of the relationship between PARTY (PERSON or ORGANIZATION) and the declarative role.

- **Much like the Level 1 Pattern, this pattern may not be suitable for a dynamic environment where new role types are added frequently** For this pattern, if new roles are discovered, you will require additional entities, and if this pattern is implemented, it could be costly to have to add new tables when new roles emerge. Imagine you discover a complete new set of marketing roles, such as PROSPECT, SUSPECT, and so on. Each would require a new entity and a new relationship to PARTY, PERSON, or ORGANIZATION.
- **This pattern does not support the management and maintenance of data about the type of role** This pattern may also benefit from having a ROLE TYPE entity attached to the different roles. This would help manage all of the different declarative role types that could be captured for a subject area. Additionally, grouping together all the roles and having a ROLE TYPE entity allows for the capabilities to maintain permissions and authorizations for roles.

Synopsis

In this section you saw that the Level 2 Declarative Role Pattern is a very useful pattern for showing the scope of all of the roles in a well-defined static subject area. The concept of a PARTY (PERSON or ORGANIZATION) was introduced. Each of the roles was shown individually, but also each was shown with its relationship to PARTY, PERSON, and ORGANIZATION. These relationships were used to enforce powerful business rules such as “each PERSON *may be* **acting as** one and only one EMPLOYEE.”

This pattern was significant because the semantic concept of PARTY(s) playing roles was introduced in this pattern. With the addition of the PARTY structure separate from the declarative roles, the data professional is stating that a PARTY plays roles, and not that a PARTY is a role. This pattern does not support the view that a PARTY is synonymous with the role it plays.

This pattern helps you because each of the role entities had common and specific attributes captured as part of that role. This self-contained role can make for easy development if this type of approach is implemented. Also, PERSON and ORGANIZATION data (and relationships) is captured once as part of the PARTY structure. This helps reduce redundancy and have data remain in sync and not have multiple different versions of the same PARTY information. Finally, by relating all the roles to a party, you can see a more complete picture for that party.

Level 3 Declarative Role Pattern

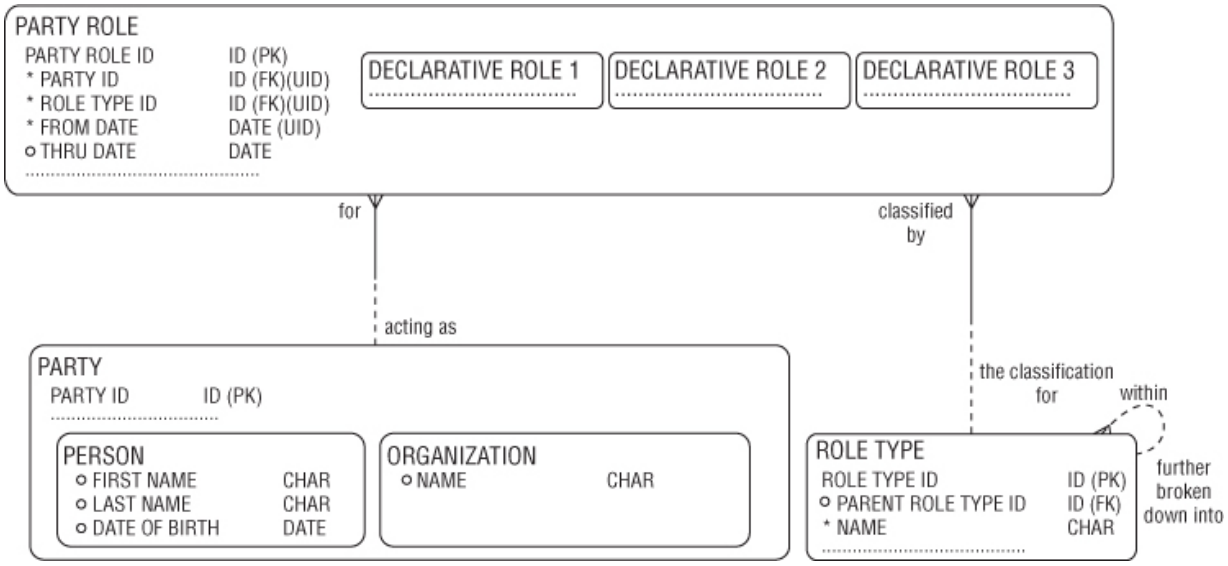
Even mature and well-organized enterprises often don't have a very good handle on the enterprise-wide roles that exist in their enterprise. Frequently, young or dynamic enterprises evolve, change, and create declarative roles when they need them. As business models change, an enterprise's data architecture should be able to support this change. So, how can you lessen the impact of adding and changing declarative roles on your data architecture?

Why Do We Need This Pattern?

The Level 3 Declarative Role Pattern may be used as a flexible approach to adding or changing declarative roles as well as maintaining data about the various role types (e.g., such as permissions, authorizations, and/or classifications

that are applicable for different types of roles). This pattern groups the declarative roles of an enterprise into a supertype called PARTY ROLE, as seen in Figure 2.5. PARTY ROLE provides a convenient way to capture all of the common declarative role attributes and relationships. When a significant declarative role gets identified by an enterprise that has specific data or relationships associated with it, it gets added as a subtype of PARTY ROLE. When a declarative role gets changed it gets changed from within the context of PARTY ROLE.

Figure 2.5 Level 3 Declarative Role Pattern



There is also a ROLE TYPE entity that maintains an instance for each type of role and this provides the ability to maintain data about the type of role, which is different than the data for a specific role. For example, **credit limit** is an attribute of a specific role of CUSTOMER, whereas, AUTHORIZATION(s) (i.e., ability to read, modify, update, and delete) may be related to the ROLE TYPE. There may be some roles that also require a subtype in PARTY ROLE because they may have specific attributes or relationships, whereas there may be other roles that need to have only an instance in the ROLE TYPE entity. The PARTY ROLE and ROLE

TYPE entities provide a buffer to minimize the impact of adding or changing declarative roles.

How Does This Pattern Work?

The model in Figure 2.5 represents a pattern for defining declarative roles in a more flexible manner. Declarative roles (DECLARATIVE ROLE 1, 2, 3) are subtypes of PARTY ROLE. Each of these declarative roles represents the roles that an enterprise defines within the context of all the activities that the enterprise carries out.

PARTY ROLE contains the common attributes and relationships that may exist to support all declarative roles, for example **from date** and **thru date** specifying when the role became effective and when the role was no longer effective. In this diagram you see that "each PARTY *may be* **acting as** one or more PARTY ROLE(s) and each PARTY ROLE *must be* **for** one and only one PARTY." This relationship is important to the pattern because it states that a PERSON or an ORGANIZATION may play many declarative roles at the same time and the declarative roles must be for a PARTY (PERSON or ORGANIZATION).

The subtypes of PARTY ROLE are shown in addition to a ROLE TYPE entity. To some extent, these both model the same type of information, because the instances of ROLE TYPE will generally correspond to a subtype of PARTY ROLE. The reason for modeling both subtypes and ROLE TYPE entity is that there may be data related to a specific subtype of PARTY ROLE—for example, the salary of an employee—and there also may be data related to the ROLE TYPE—for example, the authorizations and privileges allowed for various types of roles. Another reason is that ROLE TYPE supports the classification of PARTY ROLE. In addition to classifying PARTY ROLE, the subtypes of PARTY ROLE (for example, CUSTOMER, PARTNER, and so on) allow

maintenance of specific attributes and relationships for that role. This is a subtle but important distinction.

Like all the entities in this book PARTY ROLE has its own non-meaningful, primary key, **party role id**. But PARTY ROLE can also have a unique identifier (UID) made up of the foreign keys from PARTY and ROLE TYPE, as well as the **from date**. Many data professionals prefer to construct primary keys like this. We have no preference in this matter because there are pros and cons of each approach, and we are choosing this method just to be consistent throughout this book.

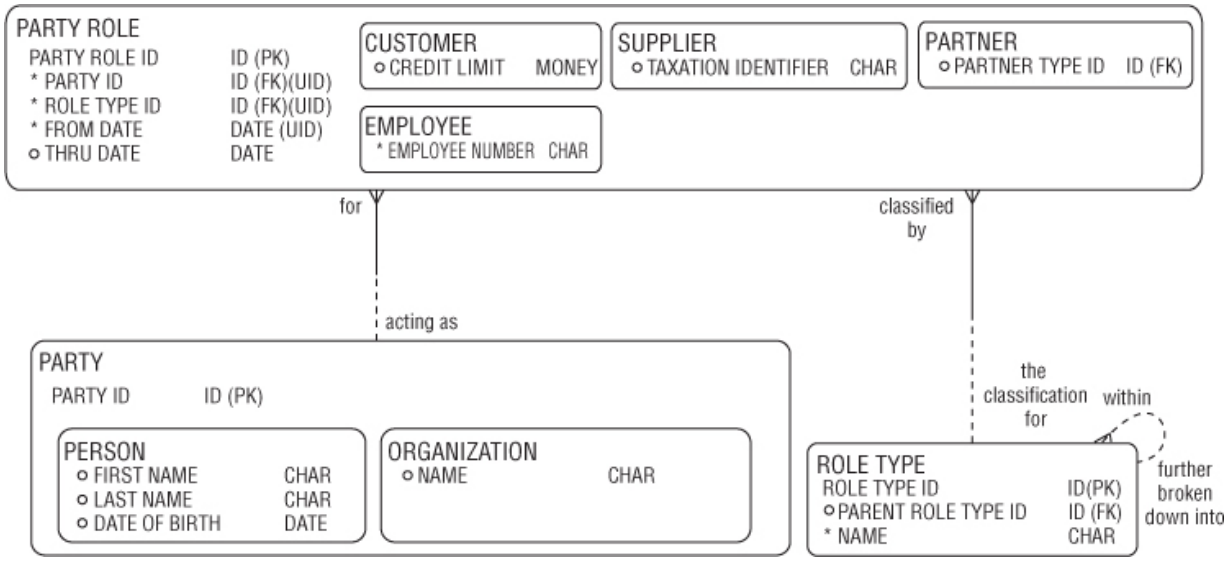
Pen Pushers Ltd described in the previous sections has recognized that it has an expanding and dynamic business. It wishes to ensure that its data architecture can easily support the dynamic nature of its business. Based on this new criterion, the data professional created Figure 2.6. The same four declarative roles defined in the previous sections are easily accommodated as subtypes of PARTY ROLE. Each of the different declarative roles may have its own attributes that are relevant only to it. All of the relationships that the different declarative roles need are handled at the supertype level. You see this with the relationship from ROLE TYPE to PARTY ROLE and the relationship from PARTY to PARTY ROLE.

Note

We are showing this model with the ORGANIZATION and PERSON attributes of **organization name**, **first name**, and **last name** instead of **current organization name**, **current first name**, and **current last name** just to show an alternative naming possibility; however, the meaning of these attributes are the same, and they represent the current names. The convention that you use for the name attributes may depend on if there are separate NAME and NAME TYPE entities that maintain the history of names and if so, you may want to use the convention of using a prefix of “current” to specify that these are the most recent names. However, this also depends

on your or your enterprises design preferences regarding the naming convention that is used.

Figure 2.6 Example of using a Level 3 Declarative Role Pattern



If you look at Table 2.6 you can see the same instances of PARTY playing many different PARTY ROLES. For example, the PARTY "Matrix Ltd" has three different ROLE TYPE(s): "Customer," "Supplier," and "Partner." Each of the ROLE TYPE(s) corresponds to an actual PARTY ROLE instance with some information about that PARTY ROLE. In the case of CUSTOMER, an example of this is the **credit limit** of "HK$100,000"; with SUPPLIER it is the **taxation identifier** of "8567891adc456"; and for the PARTNER declarative role it is a **partner type** of "Asia Partner."

Table 2.6 Example of Using a Level 3 Declarative Role Pattern

PARTY. PARTY ID	PERSON. FIRST NAME, LAST NAME	ORGANIZATION. NAME	ROLE TYPE. ROLE TYPE ID	ROLE TYPE. NAME NAME	PARTY ROLE. PARTY ROLE ID	EMPLOYEE. EMPLOYEE NUMBER	CUSTOMER. CREDIT LIMIT	SUPPLIER. TAXATION IDENTIFIER	PARTNER. PARTNER TYPE ID (PARTNER TYPE.NAME)
1001		Matrix Ltd	1	Customer	200		HK$100,000		
1001		Matrix Ltd	2	Supplier	201			8567891adc456	
1001		Matrix Ltd	3	Partner	202				20 (Asia Partner)
1002		Card Queen Ltd	1	Customer	300		$1,000		
1002		Card Queen Ltd	2	Supplier	301			566784-0978	
1002		Card Queen Ltd	3	Partner	302				10 (Global Partner)
1003		Kantowitz Services	1	Customer	400		$90,000.00		
1003		Kantowitz Services	2	Supplier	401			55 5555 555	
1003		Kantowitz Services	3	Partner	402				10 (Global Partner)
1004	Caroline Percy		1	Customer	500		£3,445		
1004	Caroline Percy		4	Employee	501	030077			
1005	Rob Gardner		4	Employee	602	030021			
1006	Francis Pope		4	Employee	604	030031			
1008	Leon Brinkley		3	Partner	600				10 (Global Partner)

Some of the instances of PERSON (“Caroline Percy,” “Rob Gardner,” and “Francis Pope”) in Table 2.6 have a ROLE TYPE of “Employee.” Each also has an **employee number**. The instance of PERSON, “Caroline Percy,” has more than one declarative role. She also has a ROLE TYPE of “Customer,” with a **credit limit** of “£3,445” in the CUSTOMER declarative role. The PERSON “Leon Brinkley” is a PARTNER.

In Figure 2.6 you see that the PARTY ROLE supertype contains the declarative roles. This can be very useful in discovering complementary roles, roles that overlap, or roles that capture the same data but are named differently. A good single view of the roles that exist in an enterprise can be very powerful as part of a scope statement or as a way of showing duplication of data or different interpretations of the same declarative role data to senior and mid-level management.

Note

Some PARTY ROLE(s) are dependent on the context of another PARTY ROLE in order to fully define them; some roles can stand on their own. For instance, a PARTY ROLE with a ROLE TYPE of “Doctor” may exist on its own without a relationship to another PARTY ROLE, whereas a PARTY ROLE with a ROLE TYPE of “Parent organization” is related to another PARTY ROLE of “Subsidiary” and is useful to identify companies that own other companies. It would be very difficult without creating the supertype of PARTY ROLE(s) to find a common way to capture the different relationships that exist between all the different party roles. The common way to capture these relationships between PARTY ROLE(s) is by using PARTY RELATIONSHIP, which relates PARTY ROLE(s) to other PARTY ROLE(s) via two “one-to-many” relationships from PARTY ROLE to PARTY RELATIONSHIP. This is described in detail in Volume 1 of *The Data Model Resource Book*.(1)

The generalized Level 3 Patterns described in Figures 2.5 and 2.6 seem to have “lost” the fact that some roles are

person-only roles, and other roles are organization-only roles. We can address this issue in part by using the recursion around ROLE TYPE. In Figures 2.5 and 2.6, you see that the ROLE TYPE entity has a recursive relationship that states that “each ROLE TYPE *may be* **classified by** one or more ROLE TYPE(s) and each ROLE TYPE *may be* **within** one and only one ROLE TYPE.” This hierarchical structure allows you to support the classification or organization of the ROLE TYPES. For example, in Table 2.7 you see a hierarchy of different instances of ROLE TYPE. This is illustrated in Figure 2.7. This recursive relationship also allows maintenance of lower-level roles. For example, you could also add additional instances to the ROLE TYPE in order to further break down the roles of customer into “bill-to customer” and “ship-to customer” (although it depends on the circumstances whether these are actually declarative or contextual roles, as is discussed in Chapter 3).

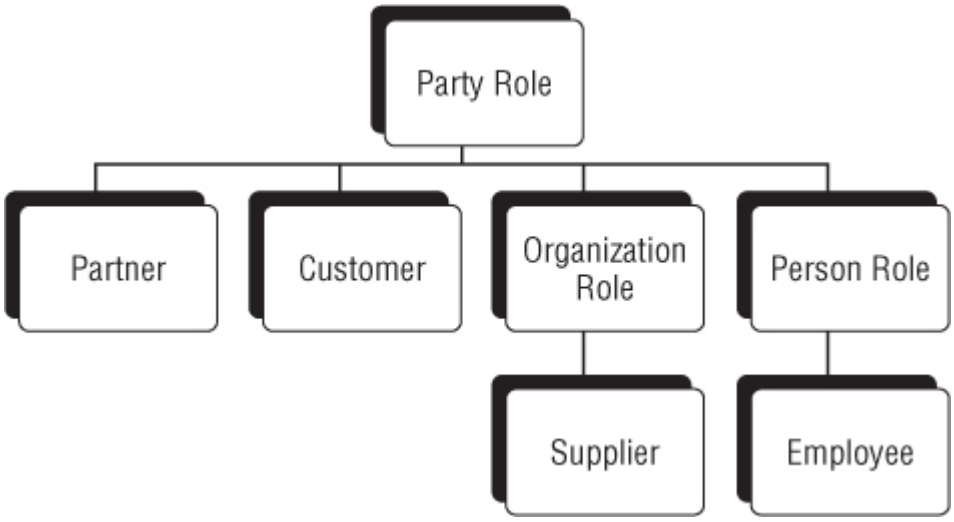
Note

An alternative way to specify which roles are person roles and which are organization roles is to have subtypes within PARTY ROLE of ORGANIZATION PARTY ROLE and PERSON PARTY ROLE and then have the roles that are person-only roles as subtypes of PERSON PARTY ROLE, have the roles that are organization-only roles as subtypes of ORGANIZATION PARTY ROLE, and have roles that are for both as subtypes of PARTY ROLE. Another way of doing this is to have a PARTY TYPE entity (person, organization) and an intersection entity VALID PARTY TYPE ROLE TYPE between ROLE TYPE and PARTY TYPE. This latter method allows more flexibility if the rules change, for example, if a SUPPLIER changes to allow people to be suppliers in addition to having suppliers that are organizations.

Table 2.7 Hierarchy of ROLE TYPE(s)

ROLE TYPE. PARENT ROLE TYPE ID	ROLE TYPE. NAME (PARENT)	ROLE TYPE. ROLE TYPE ID	ROLE TYPE. NAME (CHILD)
100	Party Role	1	Customer
200	Organization Role	2	Supplier
100	Party Role	3	Partner
300	Person Role	4	Employee
100	Party Role	200	Organization Role
100	Party Role	300	Person Role

Figure 2.7 Hierarchy of ROLE TYPE(s)



The instances of ROLE TYPE, “Organization Role” and “Person Role,” don't appear in Figures 2.5 and 2.6. Many subtypes of PARTY ROLE will have a corresponding instance in the ROLE TYPE entity. You can see from Figure 2.6 that this is the case with “Customer,” “Supplier,” “Partner,” and “Employee.” But there are no subtypes for “Organization Role” and “Person Role.” You are not precluded from adding useful categorizations to ROLE TYPE as well as other minor roles (that may not have their own attributes or relationships) that are not specifically captured as subtypes of PARTY ROLE. The rule of thumb that we use is that every subtype should have a corresponding instance in the ROLE

TYPE entity, but not every instance of the ROLE TYPE entity must have a subtype in the PARTY ROLE table. The addition of “Person Role” and “Organization Role” is a way of describing the demarcation of roles that you see in the Level 2 Pattern.

The recursive relationship around ROLE TYPE in Figure 2.6 could also be used to handle categorizations of role types regarding how they are normally used within a certain business context. For example, in table 2.8, there are ROLE TYPE **name**(s) of “Bill to customer,” “Ship to customer,” “End user customer,” and “Salesperson” that are all categorized within a parent ROLE TYPE of “Order roles” (using the foreign key of **parent role type id**). Likewise, there are ROLE TYPE **name**(s) of “Shipment coordinator” (a person responsible for planning, managing and tracking shipments) and “Carrier” (a party that regularly is used to transport shipments) that are within a parent ROLE TYPE of “Shipment roles.”

Table 2.8 Alternate Hierarchy of ROLE TYPE(s)

ROLE TYPE. PARENT ROLE TYPE ID	ROLE TYPE. NAME (PARENT)	ROLE TYPE. ROLE TYPE ID	ROLE TYPE. NAME (CHILD)
500	Order role	50	Bill to customer
500	Order role	51	Ship to customer
500	Order role	52	End user customer
500	Order role	53	Salesperson
600	Shipment role	60	Shipment coordinator
600	Shipment role	61	Carrier

Thus roles may be classified in several ways such as by their context within the business (e.g., all of the roles needed for order processing, product management, or logistics for shipments), as well as whether they are person or organization roles. For this reason there is often a need

for a more flexible structure to classify roles in a many-to-many fashion. However, we have chosen to show a simplified version of this pattern, without complex role type classifications, in order to illustrate the core aspects of the pattern. See chapter 5 for patterns regarding classifications that can be added to this pattern and used to enhance the classifications for ROLE TYPE.

> **Note**
>
> Volumes 1 and 2 of *The Data Model Resource Book* (Wiley, 2001) sometimes use an alternate model that shows PARTY ROLE TYPE as a subtype of ROLE TYPE, thus showing that the declarative roles are one type of ROLE TYPE and other types of ROLE TYPES include ORDER ROLE TYPE (showing the roles normally involved in an order), SHIPMENT ROLE TYPE (showing the roles normally involved in a shipment), and so on.

When Should This Pattern Be Used?

We use this data model pattern when:

- **An enterprise has a dynamic business environment where declarative roles get added and changed as new requirements emerge over time** This pattern is the most flexible pattern of all of the patterns in this chapter. If the data professional were to add a CONTACT declarative role, CONTACT would inherit all of the common attributes and relationships to PARTY and to ROLE TYPE. Furthermore, if a new role were discovered that did not have attributes or relationships of its own, for example, “Webmaster,” then it may be an option to just add it as an instance of ROLE TYPE.
- **There is a need to effectively maintain and manage all roles** With this pattern, all of the different declarative roles in an enterprise are maintained within a single supertype. Therefore, this pattern allows

additional information to be maintained about the types of roles such as authorizations that may apply to types of roles or classifications of roles. This is also very useful as part of a statement of scope to identify the data requirements for the role types. Finally, it is also useful in illustrating potentially complementary roles, roles that overlap, and roles that have redundant attributes.

- **An enterprise wants to allow a flexible way to categorize the different declarative roles contained within PARTY ROLE via the addition of ROLE TYPE** ROLE TYPE also facilitates reporting on party roles and/or relating other entities to the ROLE TYPE. For example, there may be a need to relate ROLE TYPE to entities such as AUTHORIZATION or PRIVILEGE in order to show which roles have what authorizations and/or privileges to access certain data and/or systems.
- **A data professional wants to avoid the repetition of repeated attributes and relationships by capturing common relationships and attributes at the PARTY ROLE level and by maintaining data about a PARTY once even though the party may play many roles.**
- **There is a need to maintain a more complete picture of each party with all the roles that they play along with the associated data about each role.**
- **There is a need to relate other entities to the PARTY ROLE supertype** For example, you can see in Chapter 2 of *The Data Model Resource Book, Revised Edition, Volume 1* (Wiley, 2001) that you can relate one PARTY ROLE to another PARTY ROLE via the entity PARTY RELATIONSHIP. This allows a very flexible way to relate any party role with any other party role.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **Although this pattern provides flexibility and is very adaptable to change, one weakness is that by adding the PARTY ROLE concept you are adding a level of generalization that is not comfortable for many enterprises** This would be a new concept for many organizations and needs to be explained and nurtured in that organization.
- **The pattern obscures that certain roles are for people, others for organizations, and some for parties** In this pattern you "generalize" the roles that parties play into PARTY ROLE. By doing this generalization you obscure the fact that some of the roles are only for PEOPLE and some of the roles are just for ORGANIZATIONS. At first glance, this pattern seems to imply that all roles may be related to any type of PARTY. However, there are other ways to show which roles are applicable to people, organizations, or parties.
- **If parties do not ever play more than one role, then the advantage of reducing redundancy may not be a factor** Thus, if there are not situations where people or organizations play more than one role, the application of this pattern may be overkill. However, in our experience, people and organizations will often be involved in more than one role.

Synopsis

In this section we described a flexible pattern that can be used by many enterprises to describe the different declarative roles that are applicable. This pattern is significant because it captures all of the common attributes and relationships that declarative roles have as part of the

PARTY ROLE supertype and it provides a ROLE TYPE entity to maintain information on role types. By doing this, the enterprise makes it easier to add and change declarative roles. The PARTY ROLE supertype helps by providing a convenient way to capture all the different declarative roles that an enterprise is interested in with a single structure, and it allows this supertype to be related to other entities, for example, allowing PARTY ROLE to be related to a PARTY RELATIONSHIP entity. The ROLE TYPE entity provides a powerful way to maintain data about the type of role and to categorize each of the different types of declarative roles in an enterprise.

The addition of the PARTY ROLE and ROLE TYPE data model structures may be difficult for some enterprises. It may be too abstract for some enterprises, or it may not meet the semantic needs of enterprises that view the roles a party plays as part of the party itself.

This flexible approach lessens the impact of adding declarative roles because new roles may be added or changed much easier and also data professionals can build and re-use declarative role functionality based on the PARTY ROLE and ROLE TYPE instead of on the specific declarative roles.

Summary of Patterns

Table 2.9 contains a synopsis of all the patterns covered in this chapter.

Table 2.9 Synopsis of the Patterns

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 1 Declarative Role Pattern	Defines, describes, and supports the explicit declaration of the declarative role as a distinct, independent entity.	A well-defined set of declarative roles is identified and known by the enterprise. The enterprise subscribes to the view that a person or organization is synonymous with what each does or how each acts. An enterprise is highly resistant to the PARTY concept. As part of a statement of scope. When the data professional wishes to show the differences in perception in an enterprise about the same concept. When the declarative roles for an enterprise hardly ever change. Each person or organization only ever plays one and only one role. (This is not a very common situation.)	If flexibility is needed in the model. There is a need to not repeat the same data for each role. There is a need to see a whole picture of all the roles played by a party.	Easy to understand and use. It can serve as part of a scope statement. It can be useful when there are a small number of stable roles and/or when each person or organization plays one and only one role.	It is not a flexible model. Will contain redundant person and organization data when a party plays more than one role. It can result in an implementation that leads to many data issues. We have seen this pattern implemented in ETL type situations with success and as dimensions in star schemas. But we have also seen it implemented in relational models to disastrous effects because of the redundancy and lack of flexibility. It makes it difficult to see the whole picture (all the roles played) for a person or organization

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 2 Declarative Role Pattern	For the situation where an enterprise wishes to capture the different types of declarative roles in a more specific manner, but does not want to redundantly capture the same person and organization data for each role that they may play.	When you want to avoid capturing redundant party, person, and organization data for each role they may play. When you wish to capture the specific relationships that a person or organization has with the roles they play. When an enterprise views people and organizations differently from the roles they play. When an enterprise needs to see a more complete picture showing all the roles for a person, organization, and/or party.	When the enterprise refuses to accept the idea of PARTY. When the enterprise does not view roles as being semantically different than parties. When people and organizations don't play more than one role. When an even more flexible model is needed that allows roles to be added with additional instances of ROLE TYPE.	Provides a means to capture party information once, even when a person or organization plays more than one role. Good way to capture rules regarding people roles, organization roles, and party roles. It's specific in nature but still has the party structure so it allows other entities to have relationships to a party (such as transactions that may be to either a person or organization). Provides a template for a reasonably simple data model, thus helping to gain buy-in. It maintains a more complete picture because you can see all the information for the roles of a party (person or organization).	It requires acceptance of the more generalized and abstract concept of PARTY. Each new role requires a new entity and therefore limits flexibility. This pattern does not support the management and maintenance of data about the type of role. However, you may consider adding ROLE TYPE to this pattern to address this issue. Only allows a party to play a specific type of role once.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 3 Declarative Role Pattern	For the situation where an enterprise wishes to capture the different types of declarative roles in a very flexible manner, wants to decrease redundancy, and wants the ability to add new roles easier.	When flexibility is needed. In other words, when the scope of declarative roles is changing and/or there is a need to add new roles over time. When common attributes and relationships of party roles need to be captured. When there is a need to effectively maintain, manage and classify all the types of roles in a common data model structure.	As a part of the scope statement for nontechnical people. When the enterprise refuses to accept the idea of PARTY and/or when people and organizations don't play more than one role. If the enterprise cannot or will not buy into a more generalized model and needs to have a more specific data model.	This is a flexible and consistent approach for supporting parties, declarative roles, and role types. It allows new types of roles to be much more easily added via an additional instance of ROLE TYPE. It allows maintenance of data and relationships for the various types of roles. For instance, there may be a need for an AUTHORIZATION entity, which could be related to the ROLE TYPE. It provides a much more complete picture for each party. Avoids repetition of repeated attributes for each specific role by capturing common attributes of roles in a PARTY ROLE supertype.	Can be hard to understand for non–data professionals because it adds a higher level of generalization. If parties do not ever play more than one role, then the advantage of reducing redundancy may not be a factor. It obscures the fact that some roles are played by only PERSON(s), some roles are played by ORGANIZATION(s) and some roles may be played by PARTY(s). It only allows role types to be classified in a single way. However, the classification pattern data model structures may be added to this pattern to provide more flexibility regarding classifying role types.

References

[1] See Chapter 2 of The Data Model Resource Book, *Revised Edition, Volume 1: A Library of Universal Data Models for All Enterprises*, by Len Silverston (Wiley, 2001)

[2] Paraphrased from www.dictionary.com.

[3] For an example of automated agents as subtype of PARTY see Chapter 8 of The Data Model Resource Book, *Revised Edition, Volume 1: A Library of Universal Data Models for All Enterprises*, by Len Silverston (Wiley, 2001)

[5] Universal Data Models Repository. © Universal Data Models 2001–2008.

Chapter 3

Using Roles: How Parties Are Involved

“Who is the ship-to customer on this invoice?” “Who is the project manager of this project?” “Who was the counterparty for this stock trade?” Each of these questions has two things in common. First, they are questions about a party, that is, “who is…?” Secondly, the “who” is always about how a party (person or organization) is involved in some other entity. In other words, what role is the party playing within the context of a business activity or in the context of another entity (for example, the party's role in the context of an order, product, or some other entity)?

In Chapter 2 we discussed the importance of capturing the roles that parties play within the context of the enterprise as a whole. Recording information about customers, suppliers, partners, employees, logistics service providers, health care providers, subsidiaries, and counterparties is important to successfully understanding your total enterprise. But it is not sufficient just to capture these declarative roles. The context in which a person or organization is involved with specific business actions or other entities in your enterprise also needs to be captured. For example, an organization may be declared as a customer in an enterprise in general, as seen in Chapter 2, but within the context of an order the organization may be the “ship-to customer” and/or “bill-to customer.” The declared “customer” role may not be specific enough to be useful in supporting the order and the “ship-to customer” and “bill-to customer” may have little

meaning outside of their involvement with the order. It is for this reason that both declarative roles and contextual roles are needed. The declarative roles described in Chapter 2 support the need for an “enterprise view” of the roles parties play. This chapter supports the need to define how the parties are involved within the context of some business activity or entity.

> **Note**
>
> Though “ship-to customer” and “bill-to customer” are usually contextual roles, there are some cases in which they may be declarative roles as well. For example, you may want to declare who may be a valid ship-to and bill-to customer and therefore set them up as declarative roles before you use them.

What Is the Significance of This Type of Pattern?

It is important to understand the context in which a person or organization is interacting with business activities or entities in an enterprise. This is best described by asking, “How is the PARTY specifically involved in various aspects of our enterprise?” For instance:

- What are the attributes of people and organizations that are involved in the course of conducting business?(1)
- What roles do people and organizations play in the course of conducting business?
- Can people and organizations play many roles while conducting business, or are they precluded from doing so?

As we said in the previous chapter, some people find it useful to think of a declarative role as setting up the role and the contextual role as using a role. You may have declarative roles that are then used in the context of

another entity. For example, you may use the declarative role of SUPPLIER and then may need to relate this to a PRODUCT entity, specifying what products the supplier may be offering. Thus, the declarative role of SUPPLIER is set up using a pattern from Chapter 2, and then you can relate it to the PRODUCT. However, there are many ways that you can relate this declarative role to another entity; this chapter shows patterns and alternatives for doing this.

You also may have contextual roles that do not need to be set up as declarative roles. For example, you may have a role of “received by clerk” for a SHIPMENT. This is not generally a role that you would set up as a declarative role. Nonetheless, this is still a contextual role, and the patterns in this chapter show alternatives for modeling any type of relationship from a PARTY to another entity, whether or not the role of the party is declared in advance.

What Is in This Chapter?

This chapter initially defines what a contextual role is. The chapter then describes the data model patterns used to support the need for any enterprise to model what people and organizations do in terms of specific business activities or entities. The chapter describes how each pattern supports the attributes and relationships involved in modeling contextual roles.

Like most of the chapters in this book the style of modeling for each of the patterns starts with the most specific style (Level 1 Contextual Role Pattern) and progresses through the chapter to a more generalized style (Level 3 Contextual Role Pattern). The different levels of generalization may be applicable to different enterprises or styles of modeling. At the end of the chapter we also describe a “hybrid” pattern that combines both the specific

and generalized style of modeling in a single pattern. In summary, this chapter includes:

- The definition of a contextual role
- The different levels of contextual role patterns
- When to use and not to use different contextual role patterns
- Insights into each pattern
- The relevance of each pattern
- A synopsis of each of the patterns, pros and cons, and when to use and not use them

What Is a Contextual Role?

Context can be defined as “the circumstances in which an event occurs; a setting,”(2) and a role can be defined as “the actions and activities assigned to or required or expected of a person or group.”(3) So, what does this mean in terms of our pattern? What is being described here is the situation in which a person or organization acts or has activities within the context of a business event, transaction, happening, or piece of data. In other words, a contextual role defines how a party is (or was) involved within the context of another entity.

Interpreting the contextual role requires not just assigning a role to the entity but understanding that the role described is relevant within the context of the associated entity. For example, an employee for a large engineering company may be assigned to a particular project as the “project lead.” The contextual role would be “project lead” within the context of a particular “project.”

It is important not to confuse contextual roles with declarative roles. We characterize the difference between them by asking ourselves the following questions:

- **Is this role significant within the context of a specific piece of business information, transaction, or event?** If the answer is yes, the role is normally a contextual role. For example, the person who assured the quality of an ORDER may play a contextual role of “quality assurance person” that doesn't need to be declared as a declarative role.
- **Is this role significant within the context of the enterprise as whole?** If the answer is yes, the role is a declarative role. For example, the role of CUSTOMER may be significant not only for specific transactions but in general across the enterprise. Declarative roles may be used and built upon using the contextual role patterns in this chapter. For example, a CUSTOMER may be declared and then used to relate the CUSTOMER to a particular ORDER.

Level 1 Contextual Role Pattern, Attributes

It is often common to see the specific roles associated with an entity captured as attributes of that entity. For example, an ORDER may have the ship-to and bill-to customers defined as attributes of that order. The names of the bill-to and ship-to customers are captured in those attributes for an instance of a specific order.

Why Do We Need This Pattern?

The Level 1 Contextual Role Pattern, Attributes, shown in Figure 3.1, provides the most specific way to model contextual roles. The roles are maintained via attributes for that particular entity. This pattern also provides a view of the scope of all the roles for a particular piece of data. Also, it is surprising how many data models; physical databases;

star schema dimensions; intermediary ETL (extract, transformation, and load) databases; and legacy systems use this type of pattern. Additionally, there are legitimate reasons to use this pattern; for example, when there is a role that does not require its own record. For example, in the PERSON entity, there may be a need to have an attribute of “mother's name.” Many times, it may not make sense to have another instance of PERSON and show the relationship from one person to their mother. For these reasons we think that it's important to understand the weaknesses and strengths of this pattern.

This pattern supports the basic aspects of a contextual role, namely:

- A representation of the roles for an entity with a definition for each role as captured in the attribute definition
- The relationships that these roles have to the entity itself
- Support for the ability of a person or organization to play one or more contextual roles at the same time

Now some data professionals may view this pattern as a “how-not-to-do” pattern because it can lead to redundant and inconsistent data if implemented. For example, if Kathy Morris is a project lead and a project worker, you may have her name stored in two places in the same entity. In other words, this pattern may capture the same party in two different places. This conclusion is correct if you look at this pattern from the perspective of relational modeling. If your enterprise models in this fashion, this pattern may not be for you.

However, it is also fair to say that dismissing this pattern as “wrong” out of hand ignores some positive traits of this approach. For example, one purpose of this pattern is to portray information requirements, to present scope, and to begin to capture the common terminology for a subject

area. It can be powerful tool for communication with different interested parties. Also, it starts the data modeling effort in a way that is easy for business people and non–data professionals to understand.

An additional and important reason for showing this as an alternative, as we said before, is that there may be cases where you actually need to maintain just a single piece of information on a role, and in this case, it is important to consider having that role as an attribute instead of creating an instance for the party and their associated role. For example, there may be a need to just have an attribute for the **name of last employer** in an EMPLOYMENT APPLICATION entity without maintaining a separate instance for that organization in the role of employer (although depending on the circumstances, you might want to model the organization as a party with their role if the need exists to capture more data about that organization as its own entity).

When just one piece of information is needed about something, it is generally considered an attribute, and thus, this pattern stores these roles as attributes because there is no other information needed about the role. William G. Smith, a noted data modeling industry leader, refers to a "will and means" rule for an entity (this is one of six rules that he has developed about what makes something an entity), which is "the business has to have the will and the means to capture and store at least one non-key attribute about the candidate entity."(4) So in some cases, roles may be modeled as attributes when there is not any other information needed about that role.

Note

An alternative opinion concerning the preceding paragraph is this: If all you need is one piece of information, such as the father's/mother's name, then perhaps this is NOT a role because there is no other PERSON information. This opinion says that

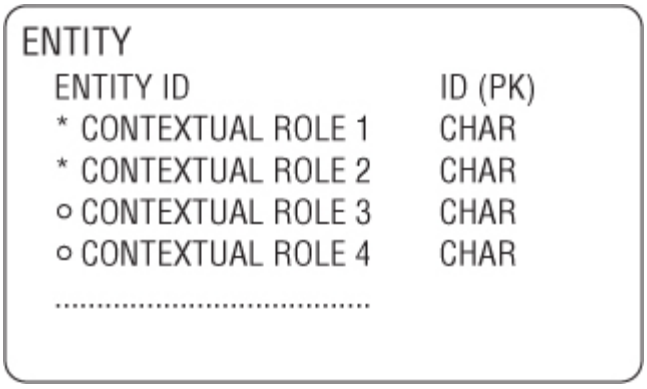
this is just a descriptive attribute of whatever entity for which data is being captured. However, a different perspective is that because a role defines how a party is involved in the enterprise or in another entity, it doesn't matter how much data is maintained if the very nature of it is a role. For example, a person's mother plays the role of “mother” for that person, regardless of how much data we maintain about her.

This type of pattern is how the business often views the people and organizations involved in some transaction, event, or other business action. We often use this pattern as the first step in showing the involved parties in a transaction, event, or other business data. Under these circumstances this model may be used as a valid tool for data professionals to more easily develop data requirements by using a very simple model.

How Does This Pattern Work?

Figure 3.1 illustrates how attributes of ENTITY are used to describe the contextual roles for the ENTITY. The **contextual role 1, 2, 3**, and **4** attributes represent each of the allowable contextual roles for ENTITY. The contextual role attributes have data values of the name of a person or organization that plays the specific contextual role within the context of ENTITY.

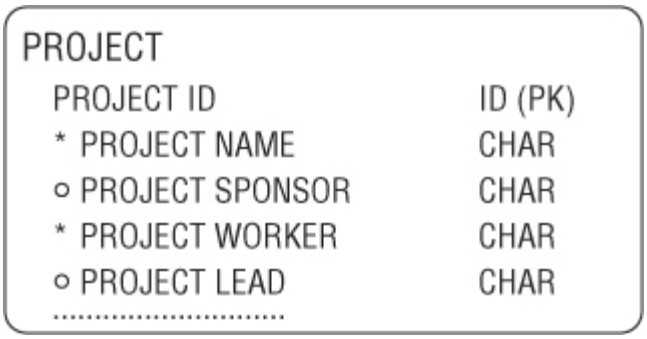
Figure 3.1 Level 1 Contextual Role Pattern, Attributes



This is a very simple structure that supports all the roles that an ENTITY has. It is a very specific and rigid way to show these roles. Only the set of contextual roles that is relevant for that ENTITY is captured—no more, no less. Each of the roles can be designated as mandatory or optional. This means that certain roles must exist for each instance of the ENTITY. Each of the contextual role attributes should have a definition for the role.

Figure 3.2 further illustrates how this pattern may support the specific contextual roles for a piece of data. The scenario is as follows: XYZ Corporation is a large international technology firm that has an ancillary need to maintain data on past projects that its employees worked on before they joined XYZ. As part of an overall data modeling effort, there was a need to have a PROJECT entity showing the name of these past projects and a few of the names of people that were associated with the project.

Figure 3.2 Example of using a Level 1 Contextual Role Pattern, Attributes



After understanding the data needs, the data professional created an initial model based on the Level 1 Contextual Role Pattern, Attributes. Figure 3.2 contains the contextual roles that were needed for the PROJECT entity, that is, **project sponsor**, **project worker**, and **project lead**. These are the roles that the company wished to maintain, without maintaining instances of these parties and/or their roles and without developing an extensive model, because

this was only a minor requirement and there was not any other information needed about these parties other than their name:(5)

- The **project sponsor** can be defined as a person or organization that finances a PROJECT.
- The **project worker** can be defined as a person who performs or is assigned to activities on a particular PROJECT.
- The **project lead** can be defined as the person or organization that provides guidance for a specific PROJECT.

The **project name** attribute is also captured as part of PROJECT. This attribute can be defined as a textual reference by which an instance of a PROJECT is designated and distinguished from other instances of PROJECT. The PROJECT itself is "an undertaking requiring concerted effort."(6)

Notice that **project worker** is a mandatory attribute. This means that a PROJECT has to have a **project worker** assigned to it. Both **project sponsor** and **project lead** are not mandatory. This means that an instance of PROJECT does not have to have an assigned **project sponsor** or **project lead**.

In Table 3.1 you see four different PROJECT instances, with four different **project name(s)**: "Customer Master Project," "Human Resources Database," "Information Architecture Standards," and "Sarbanes-Oxley Reporting" with **project id(s)** of 1001, 1002, 1003, and 1004, respectively. Each instance of PROJECT has an organization that sponsors the project as captured in the **project sponsor** attribute. The definition of **project sponsor** does not preclude a person from being a **project sponsor**. It just appears that in our example all of the project sponsors are organizations. In the case of "Customer Master Project" the project sponsor is the "Master Data Management Dept." This means that the

“Master Data Management Dept.” of XYZ Corporation is using its budget to finance the “Customer Master Project.”

Table 3.1 Example of Specific Contextual Role, Project

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	PROJECT. PROJECT SPONSOR	PROJECT. PROJECT WORKER	PROJECT. PROJECT LEAD
1001	Customer Master Project	Master Data Management Dept.	Neena Davies	Una Corr
1002	Human Resources Database	Human Resources Dept.	Neena Davies	Una Corr
1003	Information Architecture Standards	Information Technology Dept.	Una Corr	Vinnie Chintappaly
1004	Sarbanes-Oxley Reporting	Audit Dept.	Paul Lane	

Each of the different instances of PROJECT also has someone playing the role of **project worker**. You can see from Table 3.1 that “Neena Davies” works on both the “Customer Master Project” and “Human Resources Database.” Each instance of **project worker** contains the name of a person who works on the project. The definition of **project worker** precludes organizations from being project workers.

Finally, you see from Table 3.1 that each of the instances of PROJECT may contain a **project lead**. In the case of “Information Architecture Standards,” that **project lead** is “Vinnie Chintappaly.” The PROJECT “Sarbanes-Oxley Reporting” does not have a **project lead**, and this is allowed because the **project lead** attribute is not mandatory. “Una Corr” leads both the “Customer Master Project” and the “Human Resources Database” projects. A person or an organization may be a **project lead** based on the definition provided by XYZ Corporation.

When Should This Pattern Be Used?

We often use this pattern when:

- **We have a well-defined entity that has a static set of contextual roles** The entity under investigation only needed three and exactly three contextual roles, and this was not expected to change.
- **There are situations where the data professional needs to understand the business requirements more easily and perhaps illustrate a statement of scope for other IT professionals or management** The preceding diagram could be included as part of a statement of scope showing the three different key roles in which the business was interested.
- **We have circumstances where the only piece of data needed is the person's name, and it is not necessary to maintain a complete record or instance of that person** For example, sometimes the data requirement is just for the name of the mother or father of a person (for example, an employment application or passport application in some countries), which may be considered as a contextual role and captured as an attribute, but there is no need for any additional information about the father or mother.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern would not be suitable in a dynamic environment where new role types are added frequently, or if the roles are not well understood** We have seen implementations of the preceding pattern that have caused many problems when new roles arose over time because these new roles would require

changes to the model as well as to any underlying database that was based on the model.

- **The same information, such as the names of people and organizations, may be repeated across many different roles** This can lead to two different issues. First, you could be storing information redundantly. Second, and more importantly, redundant information often gets out of sync. This may cause data quality issues, such as storing “xyz Corporation” as the name in one role and maintaining “XYZ Incorporated” in another role.
- **This pattern does not explicitly enforce the delineation of people roles versus organization roles** In the preceding pattern, the definitions revealed that the **project worker** role was specifically for a person, but the entity relationship model did not enforce this rule.
- **It is easy in this pattern to mix declarative roles with contextual roles and to not even ask if a certain role is declarative** For example, in Figure 3.2, you see the attribute **project lead**. A project leader for a company may be considered a project leader for the enterprise as a whole, not just in terms of a specific project. The definition for **project lead** specified that it was a role for a specific project, but in the next section you can see that this definition gets broadened (See Figure 3.4) when the entity PROJECT LEAD is maintained as its own declarative role entity. It is important to ask if a role is declarative and/or contextual when trying to gather the scope of the roles about an entity.
- **This pattern does not accommodate multiplicity of attributes** It is fair to assume that on many projects more than one project worker would be involved in the project. This would mean that if there were 50 different **project worker(s)** on a project, there would be a need

to capture each of them as individual attributes. This may not be the most efficient way to model data as needs change.

Synopsis

In this section you saw that the Level 1 Contextual Role Pattern is a useful pattern for showing the scope of all the roles for a well-defined static entity. Each of the roles was shown individually as attributes of the entity, and each had a definition. The significance of this pattern is as a way of illustrating the roles that are related to an entity in a very easy to understand modeling style. This pattern helps to start the data modeling effort in a very understandable way, capture common contextual role terminology, and show the scope of the contextual roles in relation to an entity.

In this section you also saw that people and organizations can play more than one contextual role at the same time or over time. We see in Table 3.1, “Una Corr” in one project played the **project lead** role and in another played the **project worker** role, which can lead to data integrity issues because the same party's information can be maintained redundantly and inconsistently. Some roles may be defined exclusively as organization or people contextual roles. For example, **project worker** was described as “a person who performs or is assigned to activities on a particular PROJECT.” The weakness here is that the definition states that the role is a person role but the data model does not enforce this. To some enterprises it may be important to separate organization roles from people roles. This pattern does not enforce the demarcation of people roles versus organization roles. Finally, this pattern is most suited for very well-defined, static environments. If roles change, this pattern leads to models and subsequent implementations that are not stable, because the underlying data structures

require change. This is generally not a pattern that should be implemented in a relational database environment.

We encounter this pattern in many legacy systems, as dimensions in star schemas, or in intermediary databases in ETL environments. We think that it is crucial to understand the strengths and weaknesses of the pattern so you can make an informed decision on when to use it or versus using a different contextual role pattern.

Level 1 Contextual Role Pattern, Relationships

It is sometimes desirable to model contextual roles with a little more flexibility, but still maintain a specific style of modeling. It may also be desirable to avoid capturing data again and again as attributes, when this data could be captured once as a relationship to a specific declarative role that is maintained as an entity.

This pattern decouples the role information from the other entity data. This means semantically that you recognize the fact that contextual role information can be defined outside of its related entity. The previous pattern maintained role information as attributes in the entities. This pattern suggests a different semantic view, namely that a role is a self-contained piece of information that may be captured as an entity in its own right and then related to another entity. We believe that both semantic viewpoints have merit because the pattern that one decides to use depends on the situation. Neither viewpoint is completely correct nor completely incorrect!

Why Do We Need This Pattern?

This pattern eliminates the need to capture role data as different individual contextual role attributes. The pattern

relates a declarative role to an entity via a contextual role entity. The contextual role entity represents the role played by the declarative role (a person or organization) in the context of the entity. For example, a LOGISTICS SERVICE PROVIDER (declarative role) may be the SHIPMENT CARRIER (contextual role) for a SHIPMENT (the entity that has the associated roles).

This level 1 pattern is a little more generalized than the previous pattern. The roles are explicitly shown in their own entities, and this pattern has the advantage of being able to maintain the information about the role once, and then maintain all the specific involvement that instances of that role have with a particular entity. In other words, the pattern builds upon the use of (level 1) declarative roles described in the previous chapter (Chapter 2) and relates them to the an entity via a contextual role.

The significance of this pattern lies in the fact that you use declarative roles to maintain role information such as names, credit rating, taxation identifier, or other information about that role. It then relates the declarative roles to specific entities via a contextual role. For example, you can declare a CUSTOMER (declarative role) to be involved in an ORDER as the SHIP TO CUSTOMER (contextual role).

How Does This Pattern Work?

In Figure 3.3 you see ENTITY, which represents the business data the contextual role has an involvement in. For example, this could represent a PROJECT, ORDER, SHIPMENT, or any other business data. On the far right-hand side of Figure 3.3 you see two specific declarative roles (DECLARATIVE ROLE 1, DECLARATIVE ROLE 2). They could be roles such as customer, supplier, and agent. These declarative roles contain all of the specific attributes for that declarative role, such as a name or an identifier.

Note

The declarative roles shown in Figure 3.3 (and Figure 3.4) were created using the Level 1 Declarative Role Pattern as a template. Please see Chapter 2 for more details on the Level 1 declarative roles. Some may consider this pattern as a level 2 Pattern because it is much more generalized than the other level 1 pattern in this chapter; however, we chose to designate it as a level 1 pattern because it uses level 1 declarative roles.

Figure 3.3 Level 1 Contextual Role Pattern, Relationships

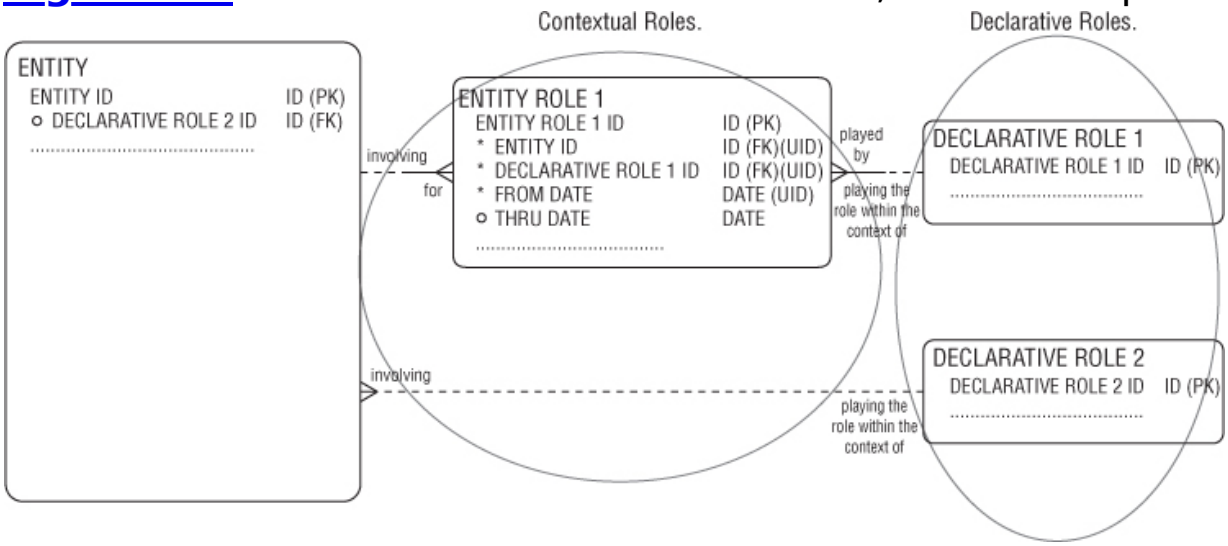
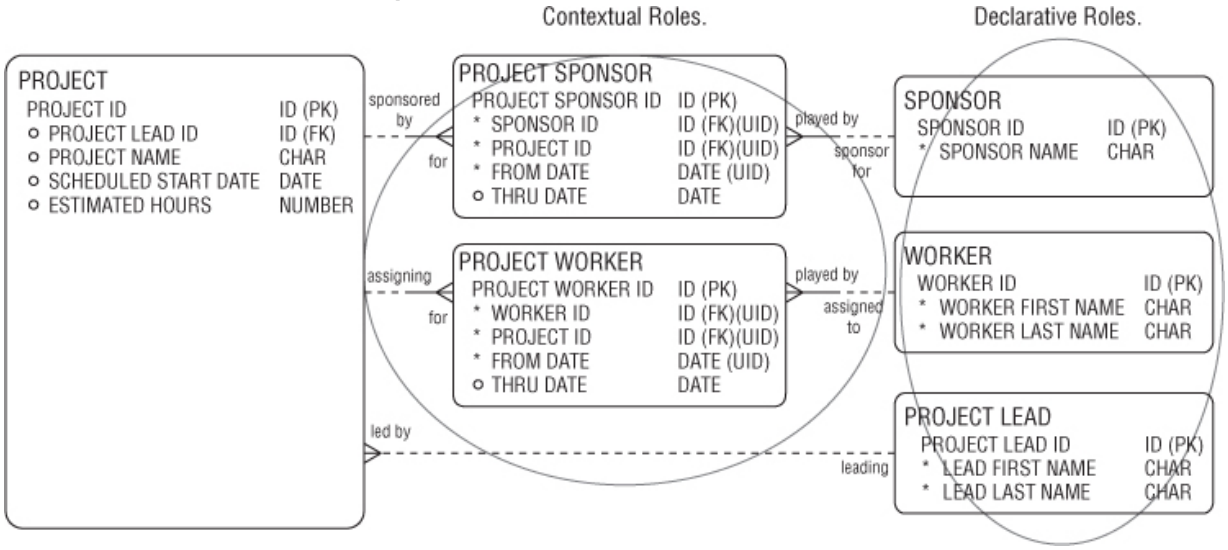


Figure 3.4 Example of using a Level 1 Contextual Role Pattern, Relationships



There are two different alternatives in relating the contextual role to the relevant entity in this pattern. First, there may be a many-to-many relationship between the declarative role and the entity that the role has an involvement in. For example, there may be more than one project worker involved in a project, and a project worker may be involved in more than one project. You see this many-to-many relationship resolved between DECLARATIVE ROLE 1 and ENTITY with the addition of the associative entity ENTITY ROLE 1. Figure 3.3 shows that "each DECLARATIVE ROLE 1 *may be* **playing the role within the context of** one or more ENTITY ROLE 1(s)" and "each ENTITY *may be* **involving** one or more ENTITY ROLE 1(s)." ENTITY ROLE 1 is the entity that captures all of the instances of the contextual role.

You can also capture the contextual role as a relationship between a role and the relevant entity. This is seen in the direct one-to-many relationship in Figure 3.3. This is useful when you have a specific business rule stating that an entity has one and only one instance of a given role. The relationship is defined in Figure 3.3 as "each DECLARATIVE ROLE 2 *may be* **playing the role within the context of** one or more ENTITY(s) and each ENTITY *may be* **involving** one and only one DECLARATIVE ROLE 2."

If we continue with the scenario of the XYZ Corporation described in the previous section, you can see that the data professional initially produced Figure 3.2 based on the Level 1 Contextual Role Pattern to show the initial scope of the contextual roles for PROJECT. Based on some further discussion with stakeholders, XYZ Corporation decided that the project data was actually much more critical than they had originally thought and that they had a need to maintain the information about each of the roles in their own entities so that instances of these roles could be maintained just once in a consistent fashion. The data professional produced

Figure 3.4 based on the alternative Level 1 Contextual Roles Pattern, Relationships, that introduces the declarative roles SPONSOR, WORKER, or PROJECT LEAD. The stakeholders revealed that SPONSOR, WORKER, and PROJECT LEAD were declarative roles because XYZ Corporation could assign a person or an organization to these roles regardless of the specific project. In other words, SPONSOR, WORKER, and PROJECT LEAD had meaning within terms of the enterprise as a whole, regardless of the specific project being worked on by XYZ Corporation. Also, the stakeholders revealed that SPONSOR, WORKER, and PROJECT LEAD played very specific roles within the context of a project, that is, as PROJECT SPONSOR, PROJECT WORKER, and as PROJECT LEAD for a project.

In Figure 3.4 you see that the SPONSOR, WORKER, and PROJECT LEAD declarative roles have been identified as roles that may be involved in a PROJECT. Each of the different declarative role entities contains a name (e.g., **sponsor name, worker first name, worker last name, lead first name, lead last name**) attribute that captures the person or organization playing the role. The data professional has also added some important attributes to the PROJECT entity that the project managers requested, that is, **scheduled start date, estimated hours**.

So, which of the entities described in Figure 3.4 are the contextual role entities? To answer this question we need to refer to the definition of what a contextual role is. What is being described here is an instance in which a person or organization may be captured as a SPONSOR, WORKER, and PROJECT LEAD and the involvement, actions, or activities they have within the setting of PROJECT.

The contextual roles PROJECT SPONSOR and PROJECT WORKER capture this involvement. They define where the person or organization has context within the PROJECT. But where is the contextual role for PROJECT LEAD(s)

involvement with PROJECT? This involvement is defined by the relationship “each PROJECT LEAD *may be* **leading** one or more PROJECTS and each PROJECT *may be* **led by** one and only one PROJECT LEAD.” If you examine Table 3.2, you see that it contains four different PROJECTS with **project name(s)** of “Customer Master Project,” “Human Resources Database,” “Information Architecture Standards” and “Sarbanes-Oxley Reporting.” The “Sarbanes-Oxley Reporting” project has more than one SPONSOR —“Information Technology Dept.” and “Audit Dept.” Also, the SPONSOR “Information Technology Dept.” sponsored both the “Sarbanes-Oxley Reporting” and “Information Architecture Standards.” It is very common for projects to have more than one sponsor and a sponsor may be sponsoring more than one project. The PROJECT SPONSOR contextual role supports this requirement.

Table 3.2 Example of Level 1 Contextual Roles, Project, Sponsor, and Project Sponsor

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	PROJECT SPONSOR. PROJECT SPONSOR ID	PROJECT SPONSOR. FROM DATE	PROJECT SPONSOR. THRU DATE	SPONSOR. SPONSOR ID	SPONSOR. SPONSOR NAME
1001	Customer Master Project	100	Oct. 10, 2007		005	Master Data Management
1002	Human Resources Database	200	Feb. 15, 2008		006	Human Resources Dept.
1003	Information Architecture Standards	300	Jan. 1, 2009		008	Information Technology Dept.
1005	Sarbanes-Oxley Reporting	400	Jan. 28, 2009	Sept. 4, 2009	008	Information Technology Dept.
1005	Sarbanes-Oxley Reporting	500	Mar. 10, 2008		0011	Audit Dept.

Table 3.3 illuminates the very common situation in which a project has many workers and a worker can work on many different projects. You see the same four projects as before. Each instance of WORKER may have a relationship with an instance of PROJECT for a distinct period of time. The **from date** and **thru date** define the range of time for which an instance of the PROJECT WORKER contextual role was valid. For example, for PROJECT “Customer Master Project,” WORKER “Neena Davies” started on “Oct. 10, 2007” and continued playing the role of worker in the project through “Jan. 18, 2009.”

Table 3.3 Example of Level 1 Contextual Roles, Project, Project Worker, and Worker

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	PROJECT WORKER. PROJECT WORKER ID	PROJECT WORKER. FROM DATE	PROJECT WORKER. THRU DATE	WORKER. WORKER ID	WORKER. WORKER FIRST NAME, WORKER LAST NAME
1001	Customer Master Project	101	Oct. 10, 2007	Jan. 18, 2009	001	Neena Davies
1001	Customer Master Project	102	Feb. 15, 2008		007	Yi Lan Tsang
1002	Human Resources Database	103	Jan. 1, 2009		001	Neena Davies
1002	Human Resources Database	104	Jan. 28, 2009	Sept. 4, 2009	003	Paul Lane
1003	Information Architecture Standards	105	Mar. 10, 2008		002	Una Corr
1003	Information Architecture Standards	106	May 16, 2008		009	Steve Toland
1005	Sarbanes-Oxley Reporting	107	Sept. 5, 2008		003	Paul Lane
1005	Sarbanes-Oxley Reporting	108	Sept. 12, 2008		010	John Teevan

Table 3.4 illustrates the specific relationship PROJECT has with PROJECT LEAD. Table 3.4 again contains the same projects as in the previous tables. “Each PROJECT *may be* **led by** one and only one PROJECT LEAD.” In this case “Una Corr” is leading both “Customer Master Project” and “Human Resources Database,” and “Vinnie Chintappaly” is leading the “Information Architecture Standards” PROJECT. The “Sarbanes-Oxley Reporting” PROJECT has no leader yet because no one has been assigned to that role.

Table 3.4 Example of Level 1 Contextual Role, Project, Project Lead

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	PROJECT LEAD. PROJECT LEAD ID	PROJECT LEAD. LEAD FIRST NAME, LEAD LAST NAME
1001	Customer Master Project	1004	Una Corr
1002	Human Resources Database	1005	Una Corr
1003	Information Architecture Standards	1006	Vinnie Chintappaly
1005	Sarbanes-Oxley Reporting		

When Should This Pattern Be Used?

We use this data model pattern when:

- **There are situations where the same person or organization is playing a (declarative) role and may have the same type of involvement many times** For example, an employee is working on more than one project, or a project has more than one employee working on it.
- **There is a need for a rigorous and specific statement of business rules** For example, XYZ Corporation stated that only one person could lead a

project. This rule was described and maintained by this pattern.

- **There is a need to be specific about the individual roles in order to more easily understand the data model** We feel that this pattern is effective in showing that a particular data area not only has a set of contextual roles, but also the relationship that the roles have to the entity in which they are involved. This pattern can be used as part of a statement of scope, to illustrate data requirements.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern repeated the same party information for each declarative role** So, for example, if a person is both a sponsor and a worker, that person's information is maintained in both entities. This pattern only solved the redundancy issue from one perspective —if the party plays the same role for different PROJECT(s), the name (and any other data about this party) is maintained only once but if the party plays two or more roles, then their information is duplicated.
- **This pattern may not be the pattern best suited to a dynamic environment, or to an environment that is not well understood** If new roles are discovered or added, then new entities need to be added to the model, as well as new relationships. For example, in many circumstances, changes in process that occur over time require new roles to be added to a data model. An example of this for projects could be the addition of a quality assurance manager or a project supervisor, and in this pattern, such a change would require new roles to be added.

- **If there is a business rule change, the model needs to change** For example, if the business rule changes so that there can now be more than one project lead, the model (and any underlying database design that is based upon this model) will need to be changed.

Synopsis

In this section you saw that the Level 1 Contextual Role Pattern, Relationships, is a very useful pattern for showing the scope and data requirements of all of the roles in a well-defined static subject area. The significance of this pattern is that the contextual role gets captured as part of the relationships that exist between the declarative role and the entity it is involved in. For example, in Figure 3.4, three contextual roles are captured—two as associative entities PROJECT SPONSOR and PROJECT WORKER and one as the relationship "each PROJECT LEAD *may be* **leading** one or more PROJECTS and each PROJECT *may be* **led by** one and only one PROJECT LEAD."

The pattern resolves the redundancy of repeating data as attributes in an entity as seen in Figures 3.1 and 3.2, but it does not address the fact that party information may be repeated in the role information, because the same party can play more than one role. This is a specific model that should be used in static and well-understood circumstances. If the environment is changing or not well understood, using this pattern can be an issue, because new role entities and relationships will need to be added as they are discovered.

Finally, this pattern has the advantage (and disadvantage) of enforcing business rules in the data model. You see in Figure 3.4 that the rule that a PROJECT can only have one PROJECT LEAD was handled by the pattern. However, if there is a business rule change or new roles are needed, the model would need to be changed to accommodate this.

Level 2 Contextual Role Pattern

One issue with the previous contextual role patterns is that the person and organization information may be repeated redundantly. This may not be an issue when a data professional is modeling a very specific case or just wishes to use this model to show the scope and data requirements of the contextual roles under investigation. If implemented, redundantly capturing person and organization information may cause significant data quality issues, and it may not be the best way to implement a solution for contextual roles in a given subject area.

Why Do We Need This Pattern?

The Level 2 Contextual Role Pattern may be used for associating contextual roles to existing PARTY ROLE information. A ship-to customer may be the company to which you are shipping an order; the bill-to customer may be a different organization to which you send an invoice for that shipment. Both the organizations may have been captured already in your billing and shipment systems. In other words, an instance of the CUSTOMER declarative role may already exist for these customers.(7) Instead of re-creating the information about customers, you can relate the contextual roles they play to the information captured for them in PARTY ROLE. Thus you set up a customer once (or you declare a party as a customer) and then you use this role and relate it contextually to another entity such as a shipment or an invoice. Alternatively, depending on the circumstances, you may also choose to declare BILL TO CUSTOMER and SHIP TO CUSTOMER roles to show that these are valid roles for the party, and then use the contextual role pattern in this section to relate them to another entity, for example, an order.

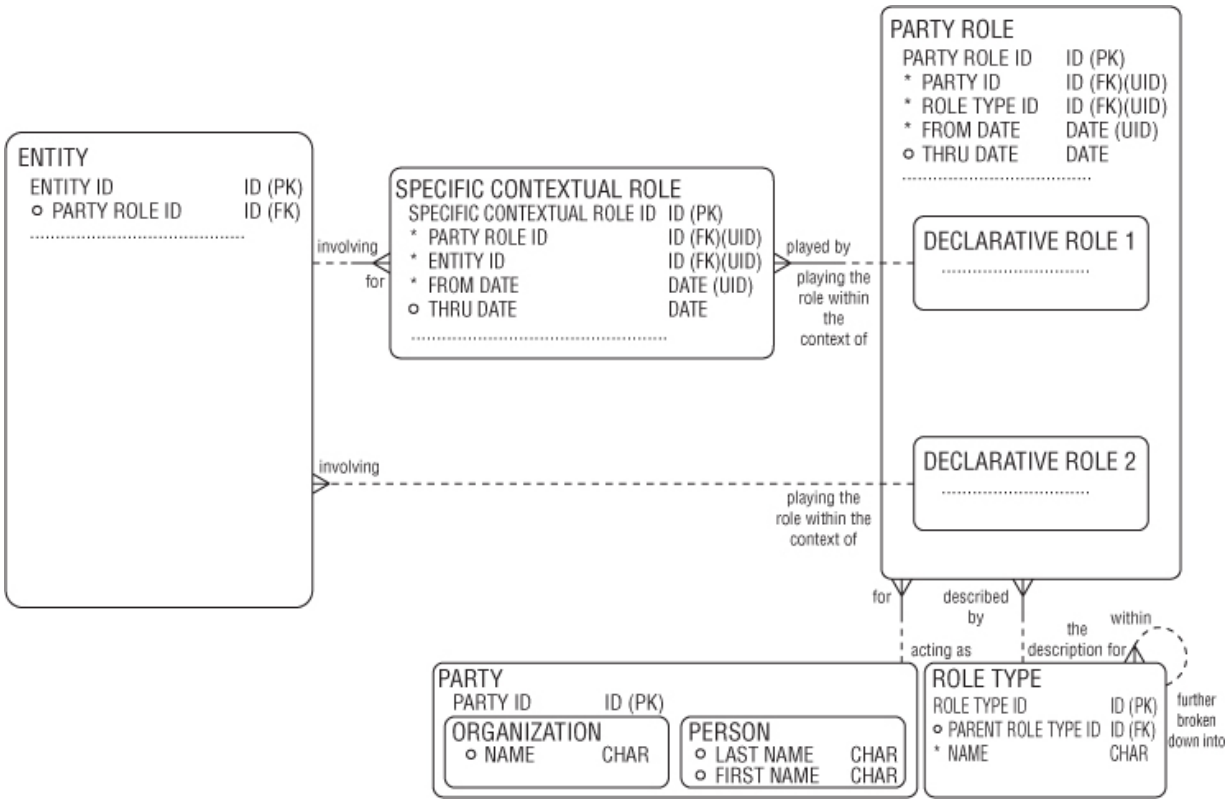
Note

The party and party role concepts are used throughout the universal data models and are discussed in great detail in Volume 1 of *The Data Model Resource Book* and in Chapter 2 of this book. Basically a party is either a person or an organization (and in some cases perhaps an automated agent or even an animal in certain types of enterprises), but just one of these. The party role is a declared role that a party plays in terms of the enterprise as a whole.

How Does This Pattern Work?

In Figure 3.5 you see a more flexible model, which captures each of the roles described in the previous pattern as a subtype of the PARTY ROLE entity. Thus this pattern provides a way to not re-create information for the same party. For example, you can maintain the name (and any other data) of a person once that is related to the PARTY and then show that the party can play many roles such as customer, supplier, or logistics service provider. But be aware that these declared roles are not the contextual roles—they are the roles that you define for the enterprise as a whole, not for a specific business activity.

Figure 3.5 Level 2 Contextual Role Pattern



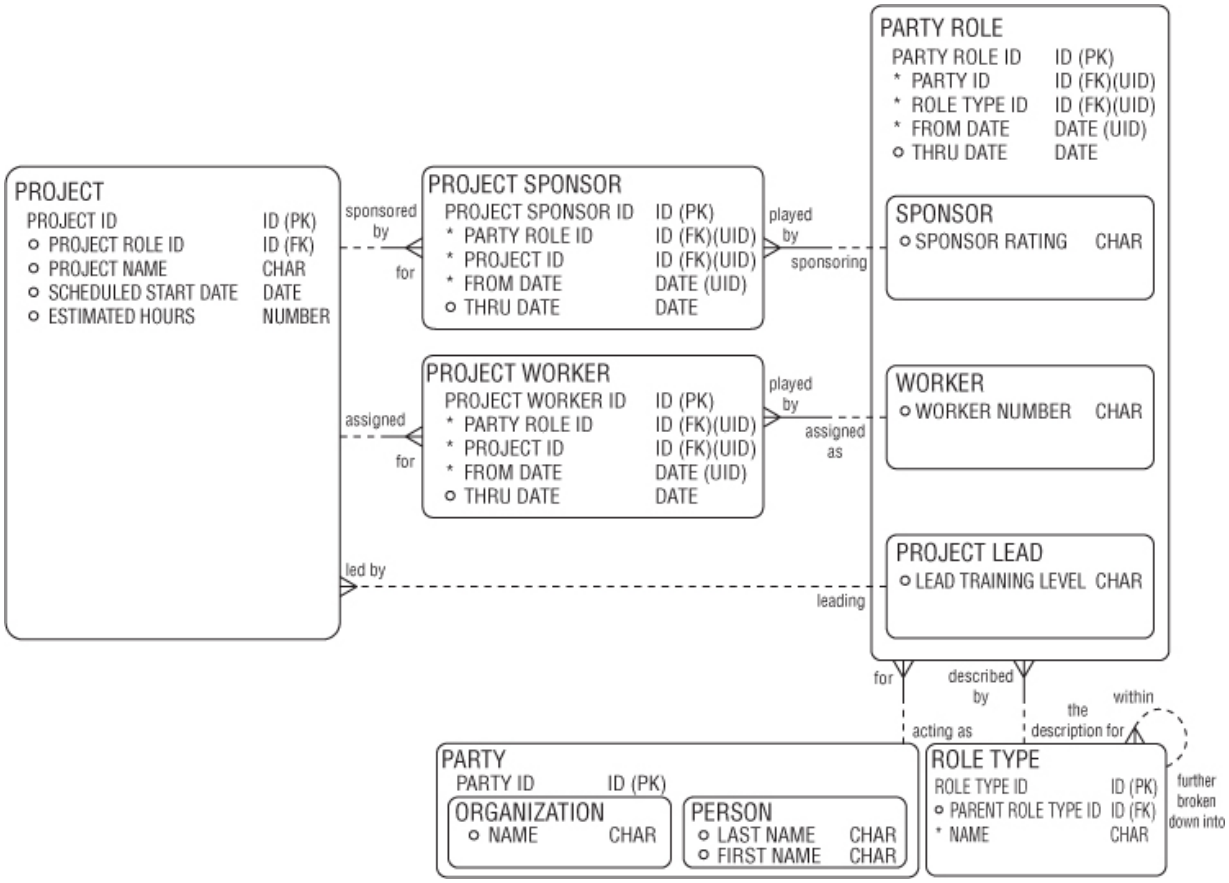
PARTY ROLE contains the common attributes and relationships that may exist to support all declarative roles, for example **from date** and **thru date** specifying when the declarative role became effective and when an instance of the role was no longer valid. In this diagram you see that "each PARTY *may be* **acting as** one or more PARTY ROLE(s) and each PARTY ROLE *must be* **for** one and only one PARTY." PARTY contains the common information about a PERSON or ORGANIZATION such as current name attributes (**first name, last name** and **name**).

The subtypes of PARTY ROLE (DECLARATIVE ROLE 1, 2) are shown in addition to a ROLE TYPE entity. To some extent, these both model the same type of information, because the instances of ROLE TYPE will generally correspond to a subtype of PARTY ROLE. The reason for modeling both subtypes and a ROLE TYPE entity is that there may be data related to a specific subtype of PARTY ROLE, for example, the salary of an employee, and there also may be data

related to the ROLE TYPE, for example, the authorizations and privileges allowed for various types of roles.

ENTITY is the information or data that the data modeler wishes to capture contextual information about. Examples of this type of data are PROJECT (as in Figure 3.6), ORDER, SHIPMENT, and INVOICE.

Figure 3.6 Example of using a Level 2 Contextual Role Pattern



Two different contextual roles are shown in Figure 3.5. The SPECIFIC CONTEXTUAL ROLE entity captures the contextual role that the PARTY acting as DECLARATIVE ROLE 1 is playing. In other words “each PARTY *may be* **acting as** one or more PARTY ROLE(s),” in this case DECLARATIVE ROLE 1, and “each DECLARATIVE ROLE 1 *may be* **playing the role within the context of** one or more SPECIFIC CONTEXTUAL ROLE(s).” For example, a PARTY may play the role of

CUSTOMER (a declarative role) and CUSTOMER may be playing the contextual roles of ORDER BILL TO CUSTOMER, ORDER SHIP TO CUSTOMER, and so on.

The second contextual role described in Figure 3.5 exists in the relationship between DECLARATIVE ROLE 2 and ENTITY. “Each DECLARATIVE ROLE 2 *may be* **playing the role within the context of** one or more ENTITY(s).” For example, a SOLE PROPRIETOR may the one and only “signatory” for a CHECKING ACCOUNT(s).

An important point here is that there are two different ways to capture the contextual roles in this pattern. If ENTITY can have only one PARTY playing the contextual role related to DECLARATIVE ROLE 2, the contextual role is captured in the one-to-many relationship. If ENTITY can have many PARTY(s) playing DECLARATIVE ROLE 1, the contextual role is captured in SPECIFIC CONTEXTUAL ROLE.

Figure 3.6 shows how this pattern can provide a powerful method for capturing very specific business rules. The project managers in XYZ Corporation described some specific business rules that they would like to see captured as part of their data model. They stated that “only parties that were declared as WORKER(s) could be PROJECT WORKER(s) and that a PROJECT could have many PROJECT WORKER(s).” They also stated that “a WORKER could be involved in many different PROJECT(s) at the same time.”

Further, they stated that “only parties that were declared as PROJECT LEAD(s) could lead a PROJECT and that there could be only one PROJECT LEAD for a PROJECT.” A PARTY had to go through a set of training courses before this PARTY could act as a PROJECT LEAD and be set up as this PARTY ROLE. This was to ensure consistency on how the company delivered its projects.

Finally, after some prompting from the data professional, the project managers stated they felt that only a PARTY that was set up as a SPONSOR could be involved in a PROJECT as

a PROJECT SPONSOR. This was a new concept for the firm, but it was felt that it could help them manage the roles that are involved in PROJECTS. Based on this information, the relationships from PARTY ROLE to the different contextual roles and PROJECT were created. In Figure 3.6 you see that "each PROJECT SPONSOR *must be* **played by** one and only one SPONSOR." Also "each PROJECT *may be* **sponsored by** one or more PROJECT SPONSOR(s)." This indicates that only a PARTY playing the role of SPONSOR may be involved in a PROJECT as a PROJECT SPONSOR and that a PROJECT can have more than one SPONSOR. Second, you see that "each PROJECT WORKER *must be* **played by** one and only one WORKER" and that "each PROJECT *may be* **assigned** one or more PROJECT WORKER(s)." This means that many PROJECT(s) can have many different PARTY(s) acting as WORKER(s), via the associative entity, PROJECT WORKER. Finally, "each PROJECT *may be* **led by** one and only one PROJECT LEAD."

Table 3.5 contains examples of all the PARTY(s) and the PARTY ROLES they play. Eleven distinct PARTY(s) are identified: "Neena Davies," "Una Corr," "Paul Lane," "Vinnie Chintappaly," "Master Data Management," "Human Resources Dept.," "Yi Lan Tsang," "Information Technology Dept.," "Steve Toland," "John Teevan," and finally "Audit Dept." You can see from this listing of PARTY(s) that both organizations and people are captured.

Table 3.5 Example of Level 2 Contextual Role Pattern with PARTY ROLE Subtypes, PARTY, ROLE TYPE, and PARTY ROLE

PARTY. PARTY ID	PARTY. FIRST NAME, LAST NAME, NAME	ROLE TYPE. ROLE TYPE ID	ROLE TYPE. ROLE TYPE NAME	PARTY ROLE. PARTY ROLE ID	PROJECT LEAD. LEAD TRAINING LEVEL	WORKER. WORKER NUMBER	SPONSOR. SPONSOR RATING
001	Neena Davies	1	Worker	5001		4412	
002	Una Corr	1	Worker	5004		4455	
002	Una Corr	2	Project Lead	1004	8		
003	Paul Lane	1	Worker	5008		4499	
004	Vinnie Chintappaly	2	Project Lead	1006	3		
005	Master Data Management	3	Sponsor	7002			A++
006	Human Resources Dept.	3	Sponsor	7004			A
007	Yi Lan Tsang	1	Worker	5010		4400	
008	Information Technology Dept.	3	Sponsor	7006			B++
009	Steve Toland	1	Worker	5012		4487	
010	John Teevan	1	Worker	5014		4445	
011	Audit Dept.	3	Sponsor	7008			A++

Each of the PARTY(s) can play one or more PARTY ROLE(s). The PARTY “Una Corr” plays two roles, “Project Lead” and “Worker.” Each of the PARTY(s), depending on which PARTY ROLE(s) they play, has specific information captured about the nature of the PARTY ROLE. For example, “Neena Davies” plays the PARTY ROLE of WORKER and has worker number “4412.” The PARTY “Audit Dept.” plays the PARTY ROLE of SPONSOR and has a sponsor rating of “A++” meaning that it is a SPONSOR with a good track record of sponsoring projects. The PARTY “Vinnie Chintappaly,” playing the PARTY ROLE of PROJECT LEAD, has a lead training level of “3” signifying the current competency of the person who is playing the role of project lead.

Table 3.6 captures the SPONSOR(s) who play the contextual role of PROJECT SPONSOR for PROJECTS. Four different PROJECTS with project name(s) of “Customer Master Project,” “Human Resources Database,” “Information Architecture Standards,” and “Sarbanes-Oxley Reporting” are contained in Table 3.6. There is more than one instance of “Sarbanes-Oxley Reporting,” illustrating that a PROJECT may have more than one SPONSOR. This supports the business rules that XYZ Corporation specified to the data professional, that a project may be sponsored by more than one party. It should also be noted that the SPONSOR “Information Technology Dept.” sponsored both the “Sarbanes-Oxley Reporting” and “Information Architecture Standards.” This illustrates the rule that a sponsor may sponsor more than one project.

Table 3.6 Example of Level 2 Contextual Role Pattern with PARTY ROLE Subtypes, PARTY, ROLE TYPE, PROJECT, SPONSOR, and PROJECT SPONSOR

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	PROJECT SPONSOR. PROJECT SPONSOR ID	PROJECT SPONSOR. FROM DATE	PROJECT SPONSOR. THRU DATE	PARTY ROLE. PARTY ROLE ID	ORGANIZATION. NAME (FOR SPONSOR)
1001	Customer Master Project	100	Oct. 10, 2007		7002	Master Data Management
1002	Human Resources Database	200	Feb. 15, 2008		7004	Human Resources Dept.
1003	Information Architecture Standards	300	Jan. 1, 2009		7006	Information Technology Dept.
1005	Sarbanes-Oxley Reporting	400	Jan. 28, 2009	Sept. 4, 2009	7006	Information Technology Dept.
1005	Sarbanes-Oxley Reporting	500	Mar. 10, 2008		7008	Audit Dept.

Table 3.7 contains examples of PROJECT(s) and the WORKER(s) assigned to a PROJECT. Each WORKER may have a relationship with the project for a distinct period of time, that is, the period of time they were assigned to the project —the **from date** and **thru date** capture the period of time that the person worked on the project. For example, “Human Resources Database” has two WORKER(s) playing the role of PROJECT WORKER. “Neena Davies” started on “Jan. 1, 2009,” the day after the **thru date** on the “Customer Master Project.” “Paul Lane” was assigned to the “Human Resources Database” on “Jan. 28, 2007,” an assignment that ended on “Sept. 4, 2009.”

Table 3.7 Example of Level 2 Contextual Role Pattern with PARTY ROLE Subtypes, PARTY, ROLE TYPE, PROJECT, WORKER, and PROJECT WORKER

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	PROJECT WORKER. PROJECT WORKER ID	PROJECT WORKER. FROM DATE	PROJECT WORKER. THRU DATE	PARTY ROLE. PARTY ROLE ID	PERSON .FIRST NAME, LAST NAME (OF WORKER)
1001	Customer Master Project	101	Oct. 10, 2007	Jan. 18, 2009	5001	Neena Davies
1001	Customer Master Project	102	Feb. 15, 2008		5010	Yi Lan Tsang
1002	Human Resources Database	103	Jan. 1, 2009		5001	Neena Davies
1002	Human Resources Database	104	Jan. 28, 2009	Sept. 4, 2009	5008	Paul Lane
1003	Information Architecture Standards	105	Mar. 10, 2008		5004	Una Corr
1003	Information Architecture Standards	106	Sept. 5, 2011		5012	Steve Toland
1005	Sarbanes-Oxley Reporting	107	Jan. 3, 2011		5008	Paul Lane
1005	Sarbanes-Oxley Reporting	108	June 11, 2010		5014	John Teevan

Table 3.8 contains the four different PROJECTS with **project name(s)** of “Customer Master Project,” “Human Resources Database,” “Information Architecture Standards,” and “Sarbanes-Oxley Reporting” seen previously. The business stated that each PROJECT may have one and only one PROJECT LEAD. In this case “Una Corr” is leading both “Customer Master Project” and “Human Resources Database,” and “Vinnie Chintappaly” is leading the “Information Architecture Standards” PROJECT. The “Sarbanes-Oxley Reporting” PROJECT has no leader; the relationship between the PROJECT and the PROJECT LEAD is non-mandatory. XYZ Corporation told the data professional that they did not need to keep any date information about the time a project lead was assigned to a project. For this reason, there are no from and thru dates (project lead from date, project lead thru date) captured in PROJECT.

Note

The previous tables have from and thru dates, but this one does not because there is a one-to-many relationship from PROJECT LEAD to PROJECT. To account for history, we could have either added the attributes of **project lead from date** and **project lead thru date** to PROJECT or changed this to a many-to-many relationship to support this need. XYZ Corporation decided it did not need this functionality.

Table 3.8 Example of Level 2 Contextual Role Pattern with PARTY ROLE Subtypes, PARTY, ROLE TYPE, PROJECT, and PROJECT LEAD

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	PROJECT. PARTY ROLE ID	PERSON. FIRST NAME, LAST NAME (FOR PROJECT LEAD)
1001	Customer Master Project	1004	Una Corr
1002	Human Resources Database	1004	Una Corr
1003	Information Architecture Standards	1006	Vinnie Chintappaly
1005	Sarbanes-Oxley Reporting		

When Should This Pattern Be Used?

We use this data model pattern when:

- **There is a need to specifically model in order to better understand the information requirements.**
- **There is buy-in regarding using the concept of PARTY ROLE to show that the same party may play many roles and then applying each role within the context of another entity.**
- **There is a decision to integrate the use of declarative roles and contextual roles** Both types of roles are important for different reasons. Declarative roles capture the definition of who parties are in terms of an enterprise as a whole. Contextual roles capture how parties are involved in other entities such as those representing business activities. They are distinct concepts, but they are related. Often a party may be declared but has not yet gotten involved in some business activity. An example would be a new employee who has been set up but has not been assigned to a particular project or is not yet related to other entities.

This pattern shows how these two patterns can support each other.

- **There is a need to specify business rules in the data model** In the example in Figure 3.6 you saw that a PROJECT could have one and only one PROJECT LEAD. This was supported specifically by the relationship in the data model.
- **There is a need to avoid repeating attributes and relationships, by capturing common attributes in PARTY and in PARTY ROLE.**

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **By adding the PARTY ROLE concept you are adding a level of generalization that may be difficult to gain buy-in for from some enterprises** Some enterprises don't view PARTY ROLE as different from PARTY. The view that “you are what you do” is prevalent in many different enterprises. This view is not wrong. It is a valid alternative and is discussed in “Level 2 Contextual Role Pattern, PARTY Only Alternative.”
- **This pattern forces XYZ Corporation to use declarative roles** In other words, if XYZ Corporation discovered a new contextual role, such as “Project risk officer,” they would have to attach to a declarative role. What happens if there are no declarative roles defined that support this role? XYZ Corporation would have to create this new declarative role as well as the new contextual role.

Note

It could be argued that the weakness just mentioned is in fact a strength. In enterprises that have strict data policies it may be

desirable to force the use of declarative roles for all contextual roles. Additionally, this pattern could be expanded to allow the use of contextual-only roles and just relate the entity at hand to a role entity that is not declared as a declarative role. For example, there could be a role entity of DATA ENTRY PERSON that is not a subtype of PARTY ROLE. This would occur if the enterprise decides that it does not want to have to declare the role of “data entry person” because anyone may do this for a specific transaction.

- **This pattern still has flexibility issues** As processes change and new roles are added over time, the underlying data model may need to be changed. What if a new role is needed later such as the quality assurance manager? Thus if you implement a database design based upon this data model pattern, it could be quite expensive to add additional tables for each new role that arises.
- **Another flexibility drawback is that if the cardinality changes over time, the data model (and resulting database structure) may need to be changed** For example, if the XYZ Corporation decided that more than one party could be a PROJECT LEAD for a PROJECT at the same time, this would require a change.
- **This pattern does not distinguish PERSON-only and ORGANIZATION-only roles** For example, in this pattern, it doesn't prohibit an organization from being a PROJECT WORKER. This may be rectified by modifying the PARTY ROLE entity to have PERSON ROLE and ORGANIZATION ROLE subtypes. Please refer to Chapter 2 for an explanation of this.
- **When there are one-to-many relationships from the role to the entity, this pattern does not account for tracking history of who played what role when** For example, what if the project lead changed over time? In this pattern, the old project lead

would just be overridden, unless this relationship was changed to a many-to-many relationship.

> **Note**
>
> It could be argued that history should be handled outside of the data model by other technical means such as with an auditing feature of the database or with history snapshot tables.

Synopsis

In this section we described a specific pattern that can be used by many enterprises to model the different contextual roles that a party may play regarding the party's involvement in some other entity. It is significant that this pattern uses the declarative roles that may be captured for a PARTY as a basis for creating contextual roles that relate to the party's involvement in another entity, such as PROJECTS, ORDERS, or SHIPMENTS.

Although there are benefits of declaring roles using the PARTY and PARTY ROLE patterns, this structure may be too generalized for some enterprises, or it may not meet the semantic needs of enterprises that view the roles a party plays as part of the party itself. It may also be too “rigorous” for some enterprises. In other words they may not feel the need to have a declarative role for every contextual role, although one could extend this pattern to also allow additional contextual role entities that are not subtypes of PARTY ROLE.

This approach helps to capture the specific business rules in the data model. Its specific nature also has the downside of not being flexible enough to easily support changes, such as new roles or changes to cardinality, without changing the data model. This means that if this data model is implemented, this could be costly because the ensuing

database structure may need to be changed as business processes change.

Level 2 Contextual Role Pattern, PARTY Only Alternative

In some situations it makes sense to relate the contextual role to a PARTY directly instead of to a PARTY ROLE, because it may not be appropriate to set up a PARTY ROLE for the involvement that a party plays in the entity. For example, in the entity INVOICE there may be a sending party and a receiving party, so it may not make sense to first declare the role of “sending party” and “receiving party” and then relate these roles to the invoice. Instead, it may make more sense to relate the invoice to a PARTY via two relationships, one for the sending party and one for the receiving party.

Additionally, some enterprises (or data model professionals) don't use the concept of PARTY ROLE. The idea that contextual roles are tied to the declarative roles may be considered cumbersome for some enterprises. Thus if these enterprises view their business in this fashion, the data model may need to support this alternative perspective.

Why Do We Need This Pattern?

This pattern supports the need to support contextual roles and how they are related to people and organizations directly. A PARTY may be a PERSON or an ORGANIZATION, and that party may play the contextual role of ORDER BILL TO CUSTOMER within the context of an ORDER.

There is also a need to have consistent PARTY information across roles. By relating the contextual roles directly to the PARTY, this pattern helps capture information about a PARTY once and only once.

> **Note**
>
> It should be noted that PARTY information is captured only once whether the relationships are directly to PARTY (as in this pattern) or thru PARTY ROLE (as in the previous pattern).

In this pattern, a role doesn't have to be specifically "declared." Because of this fact, this pattern allows more flexibility since any party may play the contextual role. For example, it may an acceptable business rule to identify any party as a project sponsor and not just parties that were declared as a sponsor.

How Does This Pattern Work?

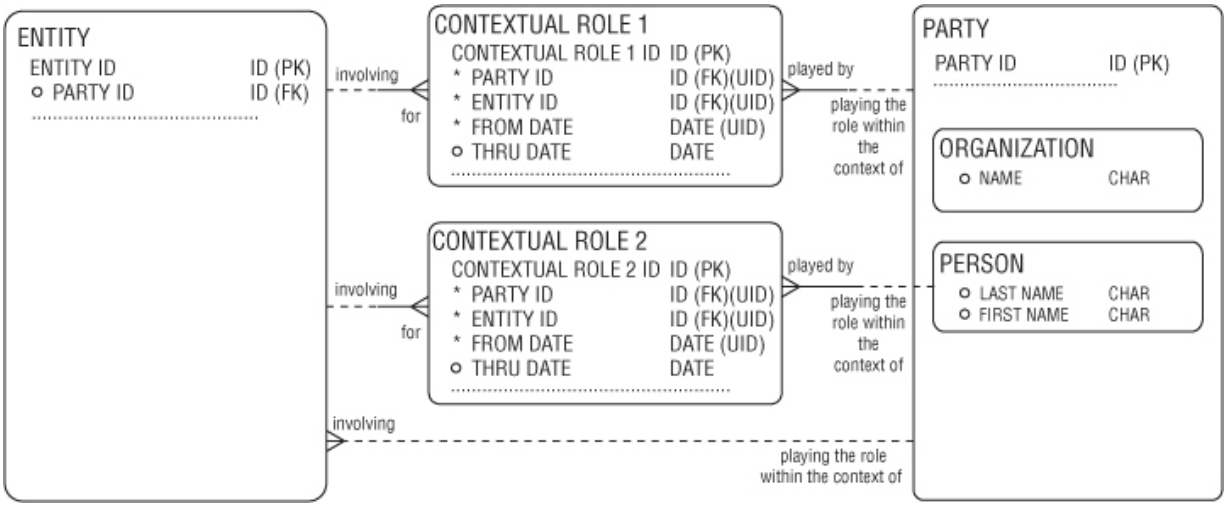
In the previous pattern the contextual roles were not directly associated with a PARTY, but were indirectly related to the PARTY via the PARTY ROLE. This pattern, that is shown in Figure 3.7, uses a direct relationship from the PARTY to the CONTEXTUAL ROLE(s) and then on to ENTITY, or directly to ENTITY. What does this mean? This pattern recognizes the fact that an instance of a PERSON or ORGANIZATION may exist that is involved in some business activity, transaction, or entity that does not specifically have a declared role. Or, the activity, transaction, or entity may be related to a declared role, but there may be cases where there is an exception to using that declared role. For example, you may have a VENDOR role, but once in a while, you might buy from a PARTY not on the vendor list. This opens up a level of flexibility and simplicity that was not seen in the Level 2 Contextual Role Pattern. The downside is that some businesses may consider this pattern to be semantically too lax for them. In other words, they may have the view that a PARTY may only be involved in some business activity when it has a specifically declared enterprise-wide role, that is, it's a declarative role as in the previous pattern.

Similar to the previous patterns, ENTITY represents any entity such as a business activity or transaction that a PARTY (PERSON or ORGANIZATION) is involved in. The ENTITY could be an ORDER, INVOICE, or SHIPMENT. PARTY can be directly related to ENTITY, or it may be related via a contextual role.

The same PARTY may be involved in an ENTITY in many different ways. For example, a PERSON may be the ship-to and bill-to customer for an ORDER. That ORDER may also have more than one PARTY involved with it, such as the PERSON that is paying for the ORDER and the ORGANIZATION that is receiving the ORDER. The CONTEXTUAL ROLE entities capture different instances of these involvements and each CONTEXTUAL ROLE entity (CONTEXTUAL ROLE 1, 2, ...) captures a different type of role that is played in these involvements. This can be seen in the relationship that states that "each PARTY *may be* **playing the role within the context of** one or more CONTEXTUAL ROLE 1(s)" and "each ENTITY *may be* **involving** one or more CONTEXTUAL ROLE 1(s)." What this relationship from ENTITY to CONTEXTUAL ROLE 1 is saying is that the ENTITY can have more than one PARTY playing this particular role at the same time, for example, a PROJECT having more than one SPONSOR(s).

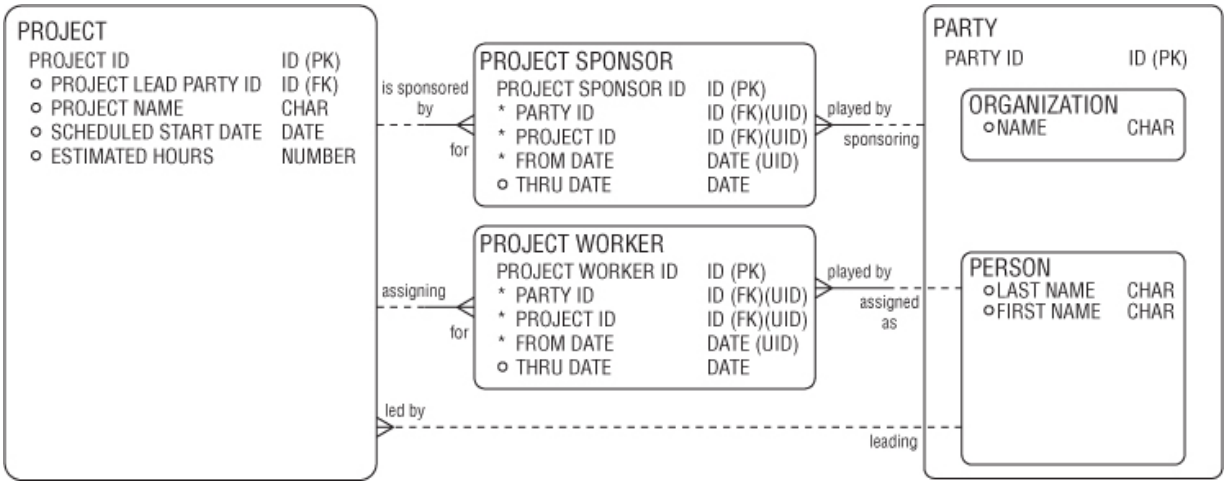
A contextual role also may be modeled as a direct relationship from PARTY to ENTITY if the nature of the relationship is one to many. The direct relationship in Figure 3.7 shows that "each ENTITY *may be* **involving** one and only one PARTY" and contains the contextual role from the PARTY to its related business activity or transaction (that is, the ENTITY). In this way a specific business rule can be shown, namely that only one PARTY can play the specified role for each instance of ENTITY.

Figure 3.7 Level 2 Contextual Role Pattern, PARTY Only Alternative



Notice that there is a relationship between PERSON and CONTEXTUAL ROLE 2. This shows that this pattern may support some specific business rules where a PERSON may be the only type of PARTY that can play a specific type of contextual role. For example, in Figure 3.8, you see a similar type of relationship that "each PERSON *may be* **assigned as** one or more PROJECT WORKER(s)." This makes sense, because normally it is only people who can be assigned as workers on a project.

Figure 3.8 Example of using a Level 2 Contextual Role Pattern, PARTY Only Alternative



This pattern can be further explained by expanding on the scenario you saw in the previous section. XYZ Corporation

may choose to model roles via relationships directly to a PARTY instead of using the PARTY ROLE. Therefore, the data professional produced Figure 3.8 as an alternative way of modeling the applicable contextual roles for the enterprise. Each of the contextual roles was captured—PROJECT SPONSOR, PROJECT WORKER, and the project lead as the relationship "each PERSON *may be* **leading** one or more PROJECT(s) and each PROJECT *may be* **led by** one and only one PERSON." The relationships are much the same as they are in Figure 3.6, except that they are directly from PARTY (or PERSON and ORGANIZATION) and not PARTY ROLE. The attribute of **project lead party id** provides a qualifier to the **party id** foreign key in PROJECT because there may be other relationships in PROJECT over time, and we have found that this is a useful convention when related PARTYs to various entities.

Table 3.9 contains examples of projects and related contextual roles that are specifically for a project sponsor. Table 3.9 contains four different instances of PROJECT with project name(s) of "Customer Master Project," "Human Resources Database," "Information Architecture Standards," and "Sarbanes-Oxley Reporting." There is more than one instance of "Sarbanes-Oxley Reporting" showing two different PARTY(s) involved as PROJECT SPONSOR(s)—"Audit Dept." and "Information Technology Dept." The contextual role of PROJECT SPONSOR also illustrates the common situation where a party may sponsor more than one project, and Table 3.9 shows an example of this where the SPONSOR "Information Technology Dept." sponsors both the "Sarbanes-Oxley Reporting" and "Information Architecture Standards" projects.

Table 3.9 Example of Level 2 Contextual Roles, PARTY Only Alternative with PARTY, PROJECT, and PROJECT SPONSOR

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	PROJECT SPONSOR. PROJECT SPONSOR ID	PROJECT SPONSOR. FROM DATE	PROJECT SPONSOR. THRU DATE	PARTY. PARTY ID	ORGANIZATION. NAME (FOR SPONSOR)
1001	Customer Master Project	100	Oct. 10, 2007		005	Master Data Management
1002	Human Resources Database	200	Feb. 15, 2008		006	Human Resources Dept.
1003	Information Architecture Standards	300	Jan. 1, 2007		008	Information Technology Dept.
1005	Sarbanes-Oxley Reporting	400	Jan. 28, 2009	Sept. 4, 2009	0011	Audit Dept.
1005	Sarbanes-Oxley Reporting	500	Mar. 10, 2008		008	Information Technology Dept.

Table 3.10 contains examples of projects and related contextual roles for workers on a project. Many people today work on multiple different projects in their day-to-day jobs. In the following examples you see “Neena Davies” assigned to both the “Customer Master Project” and the “Human Resources Database” project between Jan 1, 2008 and Jan 18, 2008.

Table 3.10 Example of Level 2 Contextual Roles, PARTY Only Alternative with PARTY, PROJECT, and PROJECT WORKER

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	PROJECT WORKER. PROJECT WORKER ID	PROJECT WORKER. FROM DATE	PROJECT WORKER. THRU DATE	PARTY. PARTY ID	PERSON. FIRST NAME, LAST NAME
1001	Customer Master Project	101	Oct. 10, 2007	Jan. 18, 2007	001	Neena Davies
1001	Customer Master Project	102	Feb. 15, 2008		006	Yi Lan Tsang
1002	Human Resources Database	103	Jan. 1, 2009		001	Neena Davies
1002	Human Resources Database	104	Jan. 28, 2009	Sept. 4, 2009	003	Paul Lane
1003	Information Architecture Standards	105	Mar. 10, 2008		002	Una Corr
1003	Information Architecture Standards	106	Sept. 5, 2011		009	Steve Toland
1005	Sarbanes-Oxley Reporting	107	Jan. 3, 2011		003	Paul Lane
1005	Sarbanes-Oxley Reporting	108	June 11, 2010		0010	John Teevan

A PERSON may have a relationship with the PROJECT for a distinct period of time also, that is, the time they were assigned to a project. The **from date** and **thru date** define the range of time for which an instance of PROJECT WORKER contextual role was valid. For example, for PROJECT "Customer Master Project," Neena Davies started on "Oct. 10, 2007" and was effective in the project thru "Jan. 18, 2009." For the same project "Yi Lan Tsang" started on "Feb. 15, 2008" and, hence, is still considered to be assigned to the project.

The PROJECT "Human Resources Database" has two PERSONS(s) playing the role of PROJECT WORKER. "Neena Davies" started on "Jan. 1, 2009," the day after the **thru date** on the "Customer Master Project." "Paul Lane" was assigned to the "Human Resources Database" on "Jan. 28, 2009," and that assignment ended on "Sept. 4, 2009."

Table 3.11 contains three different PROJECTS with **project name(s)** of "Customer Master Project," "Human Resources Database," and "Information Architecture Standards." Each PROJECT may have one and only one PROJECT LEAD leading the PROJECT, and a PROJECT LEAD may be leading one or more PROJECTS. In this case "Una Corr" is leading both "Customer Master Project" and "Human Resources Database," and "Vinnie Chintappaly" is leading the "Information Architecture Standards" project. As we stated earlier, project lead is always a person. Hence, we capture a PERSON(s) **first name** and **last name**

Table 3.11 Example of Level 2 Contextual Role, PARTY Only Alternative with PARTY(s), PARTY, PROJECT, and PROJECT LEAD

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	PROJECT. PROJECT LEAD PARTY ID	PERSON. FIRST NAME, LAST NAME
1001	Customer Master Project	002	Una Corr
1002	Human Resources Database	002	Una Corr
1003	Information Architecture Standards	004	Vinnie Chintappaly

When Should This Pattern Be Used?

We use this data model pattern when:

- **An enterprise does not use or subscribe to the concept of declarative roles using a PARTY and PARTY ROLE concept and instead defines party roles at the moments that they appear in other entities—for example, the roles for a project, order, or invoice** The declarative roles don't have to be specifically defined for a PARTY to be involved in some business activity. This model allows for more flexibility because any party may play the contextual role and thus there is not a need related the contextual role to a declared role. For example, it may be an acceptable business rule to identify any party as a party sponsor.
- **It makes more sense to have a relationship directly to a PARTY instead of using a PARTY ROLE** For example, in a PAYMENT entity you may say there are two relationships from the PAYMENT to the PARTY, namely, that the PAYMENT may be received by a PARTY and the PAYMENT may be sent by a PARTY. In this case, you may not want to set up declarative roles as a “Receiver” and a “Sender” of payments. Thus, in the same model, there may be some contextual role

relationships to PARTY ROLE and other contextual role relationships to PARTY, depending on the circumstances.

- **You want to specify business rules in the data model** In the example in Figure 3.8 you saw that a PROJECT could have one and only one PARTY leading it. This was supported specifically by the relationship in the data model. The previous pattern also did this.
- **There is a need to maintain information about a person or organization once in a PARTY entity** The previous pattern also did this.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **There are still some flexibility issues with this pattern** Because this pattern has specific relationships to roles, it requires changes to the model if new contextual roles are added over time, if the cardinality of the roles change over time, or if a history of who played what role is needed. For example, a new project role may be needed for "project advisor." Another example is that the business rule may be changed and now two project leads could lead a project. All these changes would require changes to the data model, which is an issue if the data model was the basis for the database implementation, and thus these changes to processes could be very costly because the underlying tables may need to be changed to accommodate these requirements.
- **Some enterprises may like or need the semantic rigor of forcing a new contextual role to be linked to a declarative role** In other words, some enterprises may say that every specific contextual role must be supported by an enterprise-wide role (declarative role).

One reason for doing this is to have a more consistent process for adding roles, which could include always setting up the role first (adding the declarative role) before using it (adding the contextual role).

Synopsis

In this section we described a specific pattern that can be used by many enterprises to model the different contextual roles that a party may play. The significance of this pattern lies in the fact the PARTY is directly related to the contextual roles for another entity such as a PROJECT, ORDER, or SHIPMENT. This allows any PARTY to play a role in another entity such as a business transaction or activity without having to be specifically declared as a declarative role. This approach may suit enterprises that don't subscribe to the PARTY ROLE approach, or it may provide an alternative that allows contextual roles to be related to either a PARTY or a PARTY ROLE, depending on the circumstances."

Specific business rules also are easily implemented in this pattern, such as allowing only a PERSON to be a PROJECT WORKER, or that a PROJECT can be led by only one PARTY as seen in Figure 3.8.

Similar to the previous patterns, this pattern also has some issues with flexibility. As new roles are discovered or added over time, as business rules and processes change, they need to be added as contextual role entities to the model, thus requiring a change to the model. Business rules may also require a change to the cardinality of the relationships, which could also require changes to the model. This pattern may not be suitable for an enterprise that is dynamic or if the contextual roles or business rules surrounding them are likely to change over time. Some enterprises may need the semantic rigor of forcing contextual roles to have associated declarative roles, in

other words of forcing every specific contextual role to have an enterprise-wide role to support it (the declarative role).

Level 3 Contextual Role Pattern

Up to this point all the patterns have supported well-defined static sets of contextual roles. For example, in the previous scenarios XYZ Corporation identified three different roles, PROJECT LEAD, PROJECT WORKER, and PROJECT SPONSOR. It is common, with the fast-moving pace of business today, that the contextual roles a PARTY(s) plays may change or that new roles are discovered and old roles are retired as new business situations arise. Often a new business area is not well enough understood, and the full set of roles a PARTY may play may not have been identified.

Why Do We Need This Pattern?

The Level 3 Contextual Role Pattern supports the need for a very flexible solution that allows the dynamic addition or change in nature (and retirement) of contextual roles for an entity. This pattern does not assume that all the contextual roles related to other entities, such as business transactions or activities, have been specified. Also, contextual roles can change; new role types can take over from old role types to reflect changes in business processes and practices. This pattern supports these needs.

How Does This Pattern Work?

Figure 3.9 describes how this pattern is constructed. In the previous pattern each contextual role was explicitly defined, as in Figure 3.7. In this model all the contextual roles are maintained via the CONTEXTUAL ROLE entity. This CONTEXTUAL ROLE entity serves as a three-way intersection among the ENTITY, PARTY, and ROLE TYPE allowing any

party to play any type of role, any number of times for the entity. Thus a specific entity such as ORDER may have any number of parties, playing any number of roles, of different types, any number of times (with different **from date** and **thru date**s) simply by adding additional instances in the CONTEXTUAL ROLE linking the party with the ROLE TYPE for that entity.

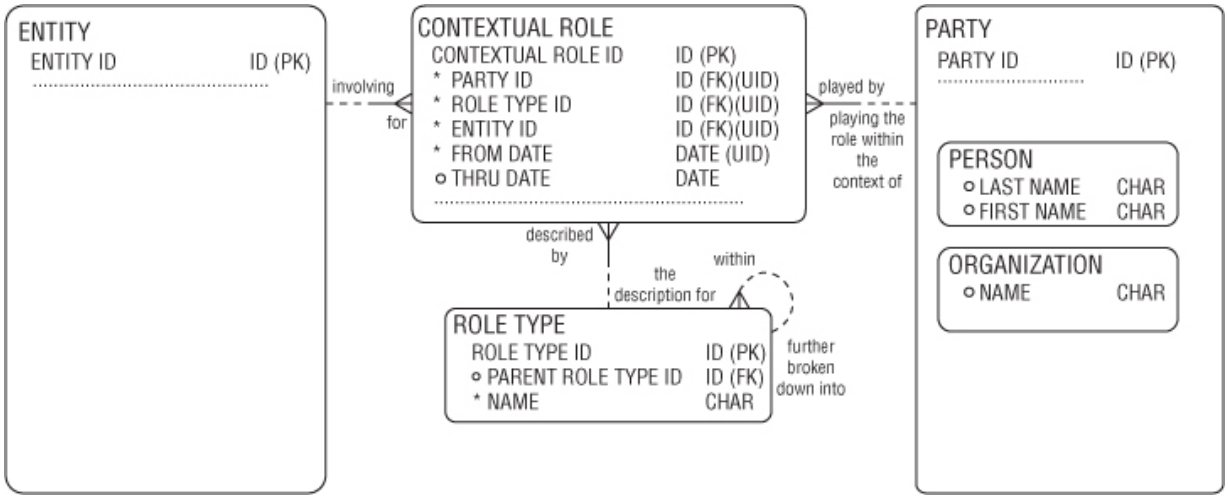
Note

The ROLE TYPE entity here is the same ROLE TYPE entity used to classify PARTY ROLE(s). In other words, ROLE TYPE can support the classification of declarative roles (as seen in Chapter 2) and contextual roles. If you wanted to distinguish declarative type roles from contextual type roles, you could create two subtypes of ROLE TYPE called CONTEXTUAL ROLE and DECLARATIVE ROLE or, you could create subtypes of ROLE TYPE for PARTY ROLE TYPE (for declarative roles) and a subtype for each type of contextual role (for example, ORDER ROLE TYPE, SHIPMENT ROLE TYPE, WORK EFFORT ROLE TYPE, and so on).

Note

Contextual roles and declarative roles may have relationships (and rules about those relationships) between them. For example, a “Provider (Declarative Role)” may be the only type of declarative role that can play the role type of “Medical Service Provider (Contextual Role)”. These types of relationships (and rules around these relationships) between different instances of the same entity are explained in detail in Chapter 4 with the Level 3 Recursive Pattern with Rules.

Figure 3.9 Level 3 Contextual Role Pattern



With this pattern, different types of contextual roles can easily be added, retired, or changed as a business process matures. For example, with the advent of the Sarbanes-Oxley Act in the United States, new types of roles for the oversight of financial transactions were created to support the needs of the act, and other types of roles became defunct. For example, there may be additional roles required for an accounting transaction, such as “quality assurance reviewer,” “auditor,” “compliance supervisor,” and so on. ROLE TYPE allows you to add and remove different types of CONTEXTUAL ROLE(s) without having to change the pattern.

As in all the previous patterns, ENTITY represents something of significance that has a set of business data about it, such as a business transaction or activity that needs to be modeled. The PARTY is the PERSON or ORGANIZATION that is participating in the business activity. “Each PARTY *may be* **playing the role within the context of** one or more CONTEXTUAL ROLE(s)” and “each ENTITY *may be* **involving** one or more CONTEXTUAL ROLE(s).” To support the business requirement of maintaining the history of role changes, the CONTEXTUAL ROLE entity contains **from date** and **thru date**. Finally “each ROLE TYPE *may be* **the description for** one or more CONTEXTUAL ROLE(s).” In other words, the PARTY may have many involvements with

ENTITY by playing CONTEXTUAL ROLE(s) of different ROLE TYPE(s).

If we expand on the scenario from the previous section, you see that the data professional has suggested an alternative pattern, because he feels that XYZ Corporation has a dynamic business environment. Based on interviews with key staff members and using the Level 3 Contextual Role Pattern, he created Figure 3.10.

Figure 3.10 Example of using a Level 3 Contextual Role Pattern

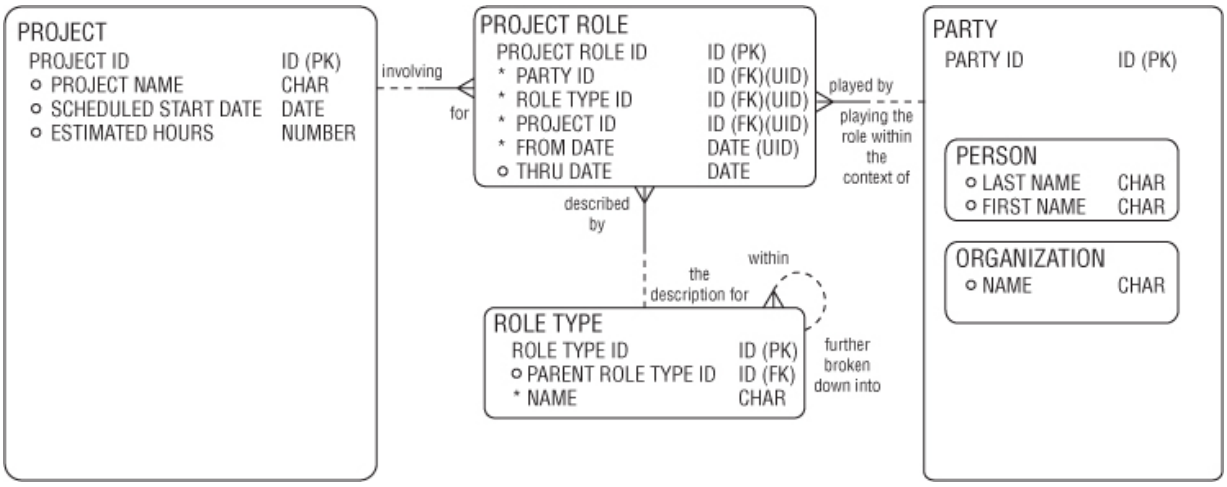


Figure 3.10 illustrates the power of this pattern. The PROJECT entity contains attributes that identify an instance of a PROJECT (**project name**), when it will start (**scheduled start date**), and how long it will take (**estimated hours**). Each PARTY(s) for XYZ Corporation may or may not have an involvement of some kind with an instance of PROJECT via PROJECT ROLE.

In the previous patterns we used the PROJECT WORKER, PROJECT SPONSOR-specific contextual roles (see Figure 3.8), and a relationship that supported project leadership roles. In this pattern each of these contextual roles is supported by PARTY, PROJECT ROLE and ROLE TYPE entities.

Note

> If you wanted to generalize the model more, you could substitute WORK EFFORT for PROJECT. That would allow maintenance of not only projects but also programs, activities, tasks, and any other type of work effort, using the same model. (5)

In Figure 3.9 (and Figure 3.10) you can see that the ROLE TYPE entity has a recursive relationship that states that "each ROLE TYPE *may be* **further broken down into** one or more ROLE TYPE(s) and each ROLE TYPE *may be* **within** one and only one ROLE TYPE." This structure allows you to support the classification or organization of the ROLE TYPE(s). For example, you may classify certain roles to be person-only roles and other roles to be party roles, or alternatively, you may classify certain instances of ROLE TYPE to be **within** a ROLE TYPE instance of "party role type" (for declarative roles), "order role type" (for contextual roles that can be associated with orders), "shipment role type" (for contextual roles that can be associated with shipments), and so on. If there is a need to classify a role with many parent roles (person-only role as well as order role type), there may need to be a many-to-many recursive relationship around the ROLE TYPE.

As an example of ROLE TYPE(s) related to other ROLE TYPE(s), in Table 3.12 you can see that "Project Worker" and "Project Lead" have a parent role of "Person-Only Role," but "Project Sponsor" has a parent role of "Party Role." There are some compelling reasons for capturing these relationships. First, it is often more convenient when reporting on roles to have a way to classify different types of roles into a super-category. For example, the business may ask, "What are all of the roles that can be played only by people?" We could generate the answer for this based on the "Person-Only Role" ROLE TYPE. Second, the generalized Level 3 Patterns described in Figures 3.9 and 3.10 seem to have "obscured" the fact that some roles are person-only roles and other roles are organization-only roles. We address

this issue in part by using ROLE TYPE. Although using ROLE TYPE does not explicitly enforce the “Person-Only” rules stated earlier in this chapter, it does allow you to express that a ROLE TYPE is a “Person-Only Role.”

Table 3.12 Hierarchy of ROLE TYPE(s)

ROLE TYPE. PARENT ROLE TYPE ID	ROLE TYPE. NAME (FOR PARENT ROLE TYPE)	ROLE TYPE. ROLE TYPE ID	ROLE TYPE. NAME (FOR CHILD ROLE TYPE)
100	Party Role	1	Project Sponsor
300	Person-Only Role	2	Project Lead
300	Person-Only Role	3	Project Worker
100	Party Role	300	Person Role

Another interesting aspect of ROLE TYPE classification is that some of the ROLE TYPE(s) may never be directly played by any PARTY involved in any ENTITY. For example, a person should never play the role “Person-Only Role,” but they would play the role “Project Worker,” which is a “Person-Only Role.” In other words, some of the ROLE TYPE instances are purely used for the classification of other ROLE TYPE(s).

Note

Some modelers may see another alternative, and instead of having a three-way intersection between the ENTITY, PARTY, and ROLE TYPE, you could show the pattern with a many-to-many relationship from the ENTITY to PARTY ROLE. A drawback of this option is that it necessitates declaring the role and having it as an instance of PARTY ROLE before it is used. The issue with doing this is that contextual roles do not necessarily have to be declared first. For example, the “Receiver” or “Sender” roles of a PHONE COMMUNICATION (a subtype of COMMUNICATION EVENT as described in Volume 1 of *The Data Model Resource Book*) may not need to be declared, yet they are important contextual roles.

Table 3.13 contains examples of projects and all related project roles for a project. Table 3.13 contains “Customer Master Project,” “Human Resources Database,” “Information Architecture Standards,” and “Sarbanes-Oxley Reporting.” A PARTY may play many PROJECT ROLE(s) described by ROLE TYPE within the context of the PROJECT. For example, the PARTY “Una Corr” plays the ROLE TYPE(s) of “Project Worker” and “Project Lead” for the PROJECT “Customer Master Project.”

Table 3.13 Example of Level 3 Contextual Roles, PARTY, ROLE TYPE, PROJECT, and PROJECT ROLE

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	ROLE TYPE. ROLE TYPE ID	ROLE TYPE. NAME	PROJECT ROLE. FROM DATE	PROJECT ROLE. THRU DATE	PROJECT ROLE. PARTY ID	PARTY. FIRST NAME, LAST NAME, NAME
1001	Customer Master Project	1	Project Worker	Sept. 5, 2009		001	Neena Davies
1001	Customer Master Project	2	Project Lead	Sept. 1, 2009		002	Una Corr
1001	Customer Master Project	1	Project Worker	Sept. 1, 2009		002	Una Corr
1001	Customer Master Project	3	Project Sponsor	June. 1, 2009		005	Master Data Management
1001	Customer Master Project	1	Project Worker	Oct. 9, 2009	Oct. 9, 2010	006	Yi Lan Tsang
1002	Human Resources Database	1	Project Worker	Mar. 1, 2008		001	Neena Davies
1002	Human Resources Database	2	Project Lead	Mar. 2, 2008	Aug. 1, 2010	002	Una Corr
1002	Human Resources Database	2	Project Lead	Aug. 2, 2009		004	Vinnie Chintappaly
1002	Human Resources Database	3	Project Sponsor	Jan. 1, 2009		006	Human Resources Dept.
1002	Human Resources Database	1	Project Worker	Feb. 5, 2008		003	Paul Lane

PROJECT. PROJECT ID	PROJECT. PROJECT NAME	ROLE TYPE. ROLE TYPE ID	ROLE TYPE. NAME	PROJECT ROLE. FROM DATE	PROJECT ROLE. THRU DATE	PROJECT ROLE. PARTY ID	PARTY. FIRST NAME, LAST NAME, NAME
1003	Information Architecture Standards	1	Project Worker	Mar. 8, 2008		002	Una Corr
1003	Information Architecture Standards	2	Project Lead	Mar. 9, 2007	May. 1, 2008	004	Vinnie Chintappaly
1003	Information Architecture Standards	3	Project Sponsor	July 7, 2005		008	Information Technology Dept.
1003	Information Architecture Standards	1	Project Worker	July 8, 2005		009	Steve Toland
1005	Sarbanes-Oxley Reporting	1	Project Worker	June 10, 2006		003	Paul Lane
1005	Sarbanes-Oxley Reporting	1	Project Worker	June 21, 2006		010	John Teevan
1005	Sarbanes-Oxley Reporting	3	Project Sponsor	July 7, 2006		008	Information Technology Dept.
1005	Sarbanes-Oxley Reporting	3	Project Sponsor	July 7, 2006		011	Audit Dept.
1005	Sarbanes-Oxley Reporting	4	Quality Assurance Manager	Aug 10, 2006		010	John Teevan

There is more than one instance of a “Project Sponsor” ROLE TYPE for the “Sarbanes-Oxley Reporting” project, illustrating that a PROJECT may have more than one “Project Sponsor,” in this case “Audit Dept.” and “Information Technology Dept.” “Information Technology Dept.” sponsored both the “Sarbanes-Oxley Reporting” and “Information Architecture Standards.” The contextual role pattern resolves the many-to-many relationship between the PARTY and PROJECT, as well as providing a place to capture the contextual information that exists between the PARTY and the PROJECT.

Thus, this pattern allows the maintenance of any number of parties to play any number of roles for an entity and even allows for the same party to play the same role any number of times (because the same party may play the same role more than once over time). Thus, if new roles are added or discovered, you can add new instances of ROLE TYPE and the model does not need to change. This is illustrated in the last row of Table 3.13, where a new ROLE TYPE of “Quality Assurance Manager” was needed, and the data model easily supported this new required role.

This pattern always supports a many-to-many relationship, and thus, it handles tracking history of roles, for example, if one person was a project lead and then another person took the role of project lead later as shown in the “Human Resources Database” project where the project lead changes from “Una Corr” to “Vinny Chintappaly.” Also, because the pattern allows a many-to-many relationship from the roles to the entity, the need to change cardinality is not an issue with this pattern. Therefore, if the business rule changed and more than one project lead was allowed for a project, the data model would support this requirement. But, because this pattern always supports many-to-many relationships, we have “lost” the rule that a project may have only one project leader at a given point in

time. This rule would have to be supported outside of the data model if you choose to use this pattern.

Another advantage of this pattern is that you maintain all the role types in a single place. This means that business rules can be consistently applied across all the role types. Thus data modelers and architects can use this same pattern so that all roles are handled in the same fashion.

When Should This Pattern Be Used?

We use this pattern when:

- **We wish to provide a very flexible solution for modeling contextual roles** New roles can be added as ROLE TYPE(s). When a PARTY plays that role, it gets captured as an instance of the generalized CONTEXTUAL ROLE, with a ROLE TYPE. This pattern allows new roles to be added or changed without changing the underlying structure, which significantly helps dealing with the impact of change.
- **There is a need for a common way to model all the roles** All types of contextual roles for many entities may be modeled using the generalized CONTEXTUAL ROLE and ROLE TYPE entities, thus capturing all of the role types in one place and contextual roles to be modeled in the same way.
- **There is a dynamic environment for the enterprise where new roles are expected to be added over time or the nature of the roles, such as the cardinality, is expected to change over time.**
- **The enterprise does not have a good handle on the contextual roles it needs, and thus, these may change frequently.**
- **An enterprise wants to capture and maintain all of the ROLE TYPE(s) in a single place** Because this

pattern does not have any specific contextual roles, all of the different types of roles are captured in ROLE TYPE(s).

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern is a more generalized form of modeling and therefore more difficult to understand** With generalization may come confusion regarding the understanding of and the use of this structure. Care needs to be taken to ensure that the structure is well understood by all stakeholders.
- **Because this pattern is more difficult to understand, it is more difficult to use this type of modeling structure as a means to capture information requirements** If one uses this pattern, we recommend showing example data (such as shown in Table 3.13) to help illustrate how this model works.
- **When using a generalized style of modeling, as in this pattern, the model may not capture specific business rules** For example, the specific business rules that the XYZ Corporation used to define its project lead roles have now been lost in the data model. One rule was that only one project lead exists for a PROJECT. With this model there could be more than one project lead at the same time for a PROJECT. When you are designing the physical database design, this type of rule could be implemented using a stored procedure or database constraint, thus allowing the many-to-many relationship to accommodate potential future needs while enforcing current business rules such as not allowing more than one project lead for a project.

- **This pattern cannot capture the optionality of contextual roles** What if a PROJECT *may* have a LEAD, but *must* have a SPONSOR?
- **This pattern does not explicitly distinguish between person-only roles and organization-only roles** We use the ROLE TYPE(s) recursive relationship to address this issue in part, but it does not explicitly support the business rule.
- **This pattern cannot handle different types of contextual roles having different attributes** What if PROJECT SPONSOR has a "sponsorship percent" attribute that does not apply to any other type of contextual role? We could create subtypes of PROJECT ROLE with specific project role attributes, but this is a generalized pattern. By adding the specific attributes, we are affecting the general nature of the pattern.
- **It is easy to mistakenly use an inappropriate role type because of the generalized ROLE TYPE entity** This entity contains all ROLE TYPE(s), some of which may not be relevant to all ENTITY(s). For example, this pattern says it is possible to have a Bill-To Customer for a PROJECT and a Project Sponsor for an INVOICE. We can address this issue by having subtypes of ROLE TYPE (for example, PROJECT ROLE TYPE, INVOICE ROLE TYPE). Another alternative is not to use ROLE TYPE at all and create PROJECT ROLE TYPE, INVOICE ROLE TYPE specifically for the contextual roles for PROJECT(s) and INVOICE(s).

Synopsis

In this section we described a very flexible way to model contextual roles. This pattern would be suitable for enterprises that either have a dynamic business environment or have difficulty anticipating or gaining understanding of the contextual roles that are needed over

time. With this adaptable model, contextual roles may be added easily without changing the model, simply by adding new ROLE TYPE(s) and assigning them to the CONTEXTUAL ROLE instance. Thus if this model is used as the basis for the database design, the database design is much more stable and adaptable, because it will not have to be changed even when business processes change that require changes to roles. Because all the roles are captured using the same type of model, the modeler can establish standard ways to handle business rules around this generalized structure.

Data models based upon this pattern are more generalized and flexible and thus may lead to confusion or lack of understanding for some audiences. Therefore, this type of style is often used as the basis for the physical database design and not generally used as a means to illustrate information requirements to business representatives. The added flexibility that this pattern offers comes with the trade-off of losing the rigorous specification of business rules at the data model level.

Hybrid Contextual Role Pattern

In this section of the chapter, we look at a hybrid pattern that combines some characteristics of Level 2 and Level 3 type patterns. In this pattern it can be shown that a mix of both styles can produce a valid data model that incorporates the best of both worlds.

Why Do We Need This Pattern?

We have found that a very effective strategy in data modeling is to model using both a specific and a generalized style of modeling. One may use a specific style of modeling in order to better understand the information requirements

as well as use a generalized style of modeling in order to serve as a solid, flexible foundation for the database design. It is possible to use these different styles of modeling by creating two different models or, as shown in the following pattern, it is possible to have one model that includes both the specific as well as a generalized style of modeling. When you have a model like this, you can show different views of the data model to different audiences. For example, you may show a view of the specific style of modeling to business representatives and show the generalized style of modeling to architects.

Another advantage of this style of modeling is that it provides possible alternatives that can be chosen at physical database design. The physical database designer may choose to implement this type of model in three ways: the specific way of modeling it, the more generalized way of modeling it, or both the specific way and the generalized way, thus allowing the specific construct for critical data and the generalized construct for any other requirement that may emerge. For example, there may be a few key contextual roles for a project such as project sponsor, project worker, and project lead, and the specific form of the pattern may be used for these. There also may be additional roles that may emerge over time and thus the generalized form of the pattern may be used to accommodate these needs.

Also, large enterprises may have many different business lines at different levels of maturity. The core business of multinational enterprises may be well understood. All the roles in that core business may be defined, and all the rules around those roles may be described. A new business line or area of interest of a multinational enterprise may not be as well understood or described. Hence the contextual roles and rules surrounding it may not be defined in detail or discovered at all. The enterprise may need to handle both

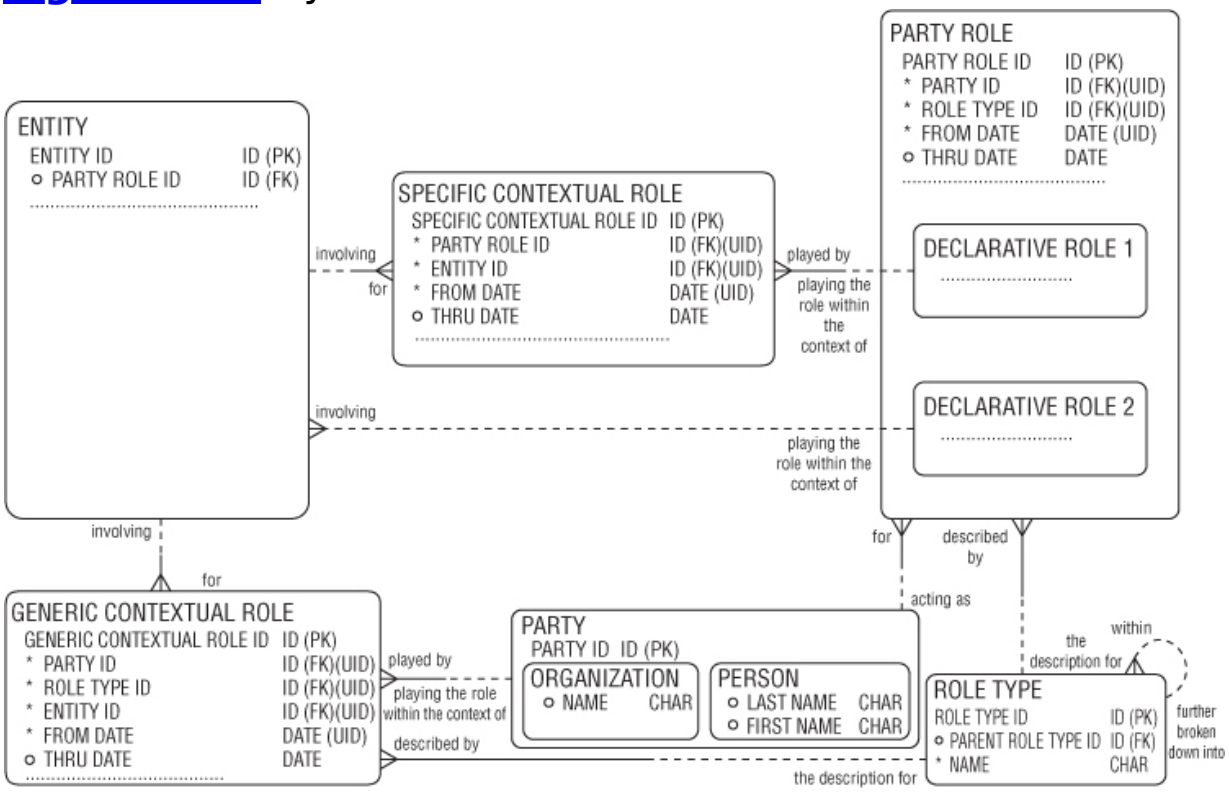
these situations in a consistent fashion. The Hybrid Contextual Role Pattern addresses this need.

> **Note**
>
> The approach of the pattern is to show alternative ways to model the same type of data: one using a specific method, and one using a much more generalized way to model; this is not the same as saying it is okay to maintain the same data redundantly. We don't consider this to be redundant data modeling, because we don't advocate that you capture the same instances of data in both ways.

How Does This Pattern Work?

Figure 3.11 describes how this pattern is constructed. You should recognize elements from both the Level 2 Patterns and the Level 3 Patterns in this figure.

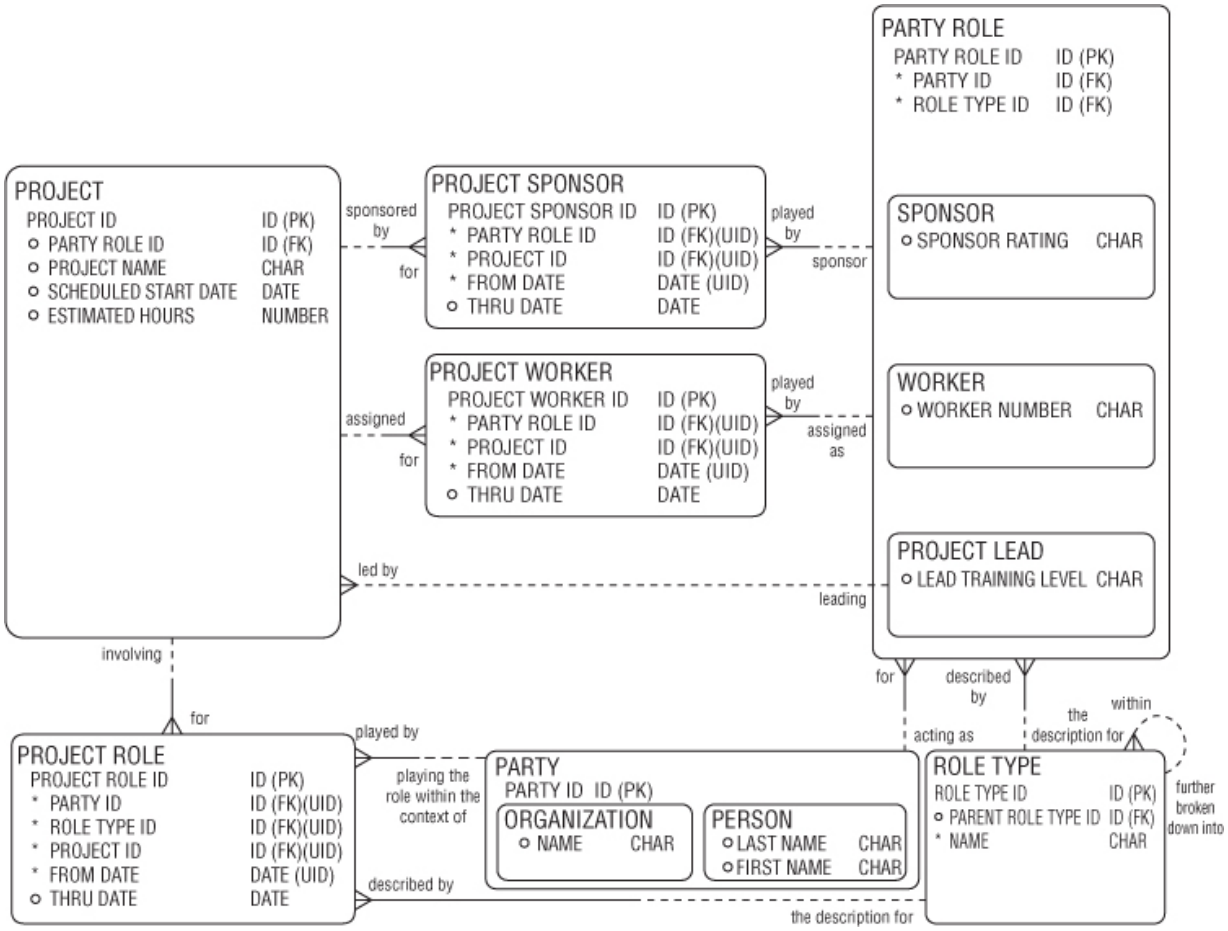
Figure 3.11 Hybrid Contextual Role Pattern



The structure from the Level 2 Contextual Role Pattern can be seen in the top middle part and also on the right-hand side of the figure. This structure supports the specific declaration of contextual roles. “Each PARTY ROLE *may be* **playing the role within the context of** one or more SPECIFIC CONTEXTUAL ROLE(s).” The specific contextual role is related to the declarative role that a party plays. It defines a very specific business rule that each SPECIFIC CONTEXTUAL ROLE *must be* **played by** a DECLARATIVE ROLE 1 (this also may be related to any declarative role or to a PARTY ROLE). In this case, an instance of the SPECIFIC CONTEXTUAL ROLE exists within the context of a declarative role. This type of modeling style is generally used for well-understood entities, business transactions, or events where the business rules and requirements regarding the roles are very stable and unchanging.

Another business rule is seen in the more specific section of the figure. That is where the specific contextual role is captured as a relationship. “Each DECLARATIVE ROLE 2 *may be* **playing the role within the context of** one or more ENTITY(s) and each ENTITY *may be* **involving** one and only one DECLARATIVE ROLE 2.” (An example of this would be a PROJECT having only one PROJECT LEAD as seen in Figure 3.12.) This is a very specific business statement, so one worthwhile question that may affect the style of modeling is, “Will this always be the case?”

Figure 3.12 Example of using the Hybrid Contextual Role Pattern



This pattern also supports a more generalized way of supporting contextual roles. You should be able to recognize the Level 3 Contextual Role Pattern structure at the bottom of the diagram. This pattern creates a flexible structure that allows contextual roles to be added as instances of GENERIC CONTEXTUAL ROLE. “Each GENERIC CONTEXTUAL ROLE *may be* **for** one or more ENTITY(s).” As each contextual role is discovered it can be added as instance of GENERIC CONTEXTUAL ROLE and its type can be captured as a member of ROLE TYPE.

One interesting aspect of this pattern is that there are common entities in the pattern. ENTITY, of course, is common; otherwise, the contextual roles (generic and specific) would have no meaning! PARTY and ROLE TYPE are also common for both generalized and specific contextual roles. PARTY needs to be in both because you don't want to

capture PARTY information again and again for each role the party plays.

ROLE TYPE, interestingly, is also common. ROLE TYPE supports the specific role model via its relationship to PARTY ROLE. It also gives context to the GENERIC CONTEXTUAL ROLE, because this is where the **name** of the role is maintained.

Figure 3.12 provides an example of this pattern. Following on from the previous scenario, XYZ Corporation has decided that it wants the flexibility of the Level 3 Pattern, but also wishes to maintain some specific business rules for core roles as in the Level 2 Pattern. Based on this requirement the data professional created the data model seen in Figure 3.12.

In Figure 3.12 you see a combination of the Level 2 Pattern seen in Figure 3.6 and the Level 3 Pattern seen in Figure 3.10. The core declarative roles SPONSOR, WORKER, and PROJECT LEAD are modeled as specific contextual roles. These are the roles mandated by XYZ Corporation as fundamental to its business.

The PARTY(s) who are acting as these roles have specific relationships with an instance of PROJECT. In other words, only a PARTY who is acting in the role of a WORKER may be involved in a PROJECT as the PROJECT WORKER. That is, a person who is playing the declarative role WORKER is involved in a PROJECT by playing the contextual role of PROJECT WORKER.

> **Note**
>
> A full explanation of the relationships between PARTY ROLE and PROJECT has already been provided in the Level 2 Contextual Role section of this chapter.

If you look at the bottom of the figure you see the generalized structures, as they are in Figure 3.10. In this

structure, “each PARTY *may be* **playing the role within the context of** one or more PROJECT ROLE(s).” This flexible structure allows you to add roles when they are discovered by adding ROLE TYPE instances for the new roles.

> **Note**
>
> A full explanation of the relationships between PARTY and PROJECT ROLE has already been provided in the Level 3 Contextual Role section of this chapter.

The power (and weakness) of this pattern lies in the fact that the role a particular PARTY plays may be modeled in two different ways. For example, in Table 3.14 you see “Customer Master Project” and “Human Resources Database.” “Neena Davies” is playing the role of worker starting on “Oct. 10, 2010” in the “Customer Master Project.” She is also in the role of worker on the “Human Resources Database” project starting on “Jan. 1, 2010.” In both cases her role as a WORKER (declarative role) is linked to her involvement in the PROJECT(s) as a PROJECT WORKER. If you look at Table 3.15, you see that the same PARTY, “Neena Davies,” has another involvement in the “Customer Master Project” as a “Technical Writer.” In this example, the generalized part of the pattern allows you to add new roles easily by adding instances of ROLE TYPE without needing to change the data model or structure.

Table 3.14 Example of the Hybrid Contextual Roles Pattern, WORKER and PROJECT WORKER

PROJECT. PROJECT NAME	PROJECT WORKER. PROJECT ID	PROJECT WORKER. FROM DATE	PROJECT WORKER. THRU DATE	PROJECT WORKER. PARTY ROLE ID	ROLE TYPE. NAME	PERSON. FIRST NAME, LAST NAME
Customer Master Project	1001	Oct. 10, 2010		5001	Worker	Neena Davies
Human Resources Database	1002	Jan. 1, 2010		5001	Worker	Neena Davies

Table 3.15 Example of Hybrid Contextual Roles, PROJECT ROLE

PROJECT. PROJECT NAME	PROJECT ROLE. PROJECT ID	PROJECT ROLE. PARTY ID	ROLE TYPE. ROLE TYPE ID	ROLE TYPE. NAME	PROJECT ROLE. FROM DATE	PROJECT ROLE. THRU DATE	PERSON. FIRST NAME, LAST NAME
Customer Master Project	1001	001	22	Technical Writer	Oct. 10, 2010		Neena Davies

So, does PROJECT ROLE include all contextual roles related to a PROJECT (including specific roles like “Project Worker” and “Project Sponsor”) or only those roles not handled via PARTY ROLE and the specific contextual roles? The answer to that question is that it could contain roles that have been modeled specifically like “Project Worker,” but that we don't recommend that it does. We think that it is important to capture the contextual roles as either a specific contextual role, such as PROJECT WORKER, or by using the generalized PROJECT ROLE/ROLE TYPE structure, for example, “Technical Writer” as seen in Table 3.15. If you capture ROLE TYPE(s) of “Project Worker” and “Project Sponsor” in the generalized PROJECT ROLE/ROLE TYPE structure, we recommend you don't also maintain these in the specific PROJECT WORKER, PROJECT SPONSOR structure. Our feeling is that you should not redundantly capture contextual roles in both the specific way as well as in the generalized way. You could end up double counting the amount of people working on a project,

for example. Or, if implemented, you would have the questions of which is the correct way for programmers to code selecting, inserting, and updating project workers and which is the correct semantic view for the business as a whole.

The strength of the pattern lies in the fact that you get the benefits of both the specific Level 2 modeling style and the generalized modeling style of the Level 3 type pattern. When we needed to capture the specific involvement of “Neena Davies,” she was declared as a WORKER and involved in the PROJECT as a PROJECT WORKER. When we discovered a new role of “Technical Writer,” we wanted to capture this, but did not want to add new declarative role entities because that role was not core to XYZ Corporation business. “Neena Davies” became involved via the PROJECT ROLE with a new ROLE TYPE of “Technical Writer.”

An important point about this pattern is that it allows different views of the data because it is modeling roles specifically as well as generally. In order to understand the information requirements and rules regarding roles, you can use the specific style of modeling. In order to provide a flexible foundation for a database design, you can show the generalized style of modeling. By modeling this both ways, you can offer different views of the model to show to different audiences, and you can also provide different possibilities for the database design. For example, the physical database designer can evaluate if he or she wants to implement the specific model, the flexible model, or both in the database design. This decision could be based upon how much flexibility is needed as well as performance and considerations of how simple the structure should be. It is also important to emphasize that we don't recommend that you capture role data in both ways at the same time.

Note

> This same technique of having a single “hybrid” pattern that combines both a specific and a general style, may be used for many of the other types of patterns in this book, even though we have not specifically illustrated hybrid patterns in many of the other chapters. We illustrate this technique further in Chapter 9, where we apply the patterns for use in developing an enterprise data model.

When Should This Pattern Be Used?

We use this data model pattern when:

- **An enterprise has a core set of contextual roles that it wants to capture and specifically model, but it also wishes to have the flexibility to capture new contextual roles without changing the underlying model.**
- **We wish to show two different views for the contextual roles in an enterprise, but maintained in a single data model** One view could be for the business user and one for the database architect and/or designer. The view for the business user could be the specific model and the view for the database architect and/or database designer could be the generalized structures.
- **We wish to maintain a single logical data model so we don't need to maintain and cross-reference multiple models with different styles** It may make sense to have two distinct models with two different purposes. However, in our experience, it requires a lot of discipline to maintain two or more models (in addition to other models such as physical database design models and process models) as well as cross-referencing these models. It is also fair to say that it takes effort and discipline to maintain both parts of the same model, but it helps if both the generalized part and the specific part are in the same model.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **In this pattern, there are two ways to maintain a PARTY's involvement or role in another entity, and there is a chance that you can capture the same role redundantly under a different name** In the example in Tables 3.14 and 3.15 “Neena Davies” was defined as a PROJECT WORKER and as a “Technical Writer.” Who is to say that this was not the same role, or perhaps a technical writer is a type of project worker?
- **If you tried to answer the question, “What are all of the different role types in our enterprise?,” you would have to check in two places** First you would need to capture all of the specific contextual roles, for example, PROJECT SPONSOR, PROJECT WORKER, and so on, and then you would need to capture all of the instances of ROLE TYPE, for example, “Technical Writer” and so on.
- **The pattern could be confusing** This is the reason that the data professional would need to create different views of this model depending on the audience. For enterprises that like to stick with one style of data model, they may be better off picking either a specific or generalized style of modeling depending on what is needed for the particular situation, instead of using this type of hybrid style.
- **Instead of having one model with two different alternatives, there are some advantages to having two different data models:** a very specific model showing the roles in whatever way can be best communicated and validated with business representatives and another architectural data model for technical audiences that is more generalized and is

designed to be the basis for the database design. By having two different models, there is more leverage to create these models exactly as one needs.

Synopsis

In this section we described a pattern that supports both the flexible modeling of contextual roles and the specific modeling of contextual roles. The pattern created in Figure 3.11 was constructed by merging the Level 2 Contextual Role Pattern with the Level 3 Contextual Role Pattern. This style of modeling is very powerful because it provides a method for understanding information requirements using the specific style of model as well as providing a stable, flexible foundation for a database design using the flexible style of modeling.

This pattern means that a data professional does not have to maintain two different sets of models, one for the business and one for other data architects and/or designers, because the same data model may support different views. But it also means that you have to manage two different styles of modeling in the same model.

With this pattern, there are two possible paths in capturing a PARTY's involvement in another entity, business transaction, activity, or event. This may lead to confusion or the redundant capture of the same role via the different paths if the pattern is misused. Some enterprises may not be comfortable mixing styles of models because the possibility of misuse exists. This pattern has all the strengths of the Level 2 and Level 3 Patterns, but it also has all the weaknesses of both of these pattern styles.

Summary of Patterns

Table 3.16 contains a synopsis of all the patterns covered in this chapter.

Table 3.16 Synopsis of the Patterns

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 1 Contextual Role Pattern, Attributes	The most specific way to model contextual roles. Each of the different entities contains its associated roles, which are captured as attributes.	As part of a statement of scope for gathering and validating data requirements. When there is a well-defined set of static contextual roles. When there is a need for just one piece of data about the role as it relates to another entity and the role doesn't require its own instance (for example, mother's name).	When flexibility is needed in the model, design, or implementation.	Easy to understand and use.	It is not a flexible model. Can contain redundant person and organization data, leading to incorrect data. There may be many of a specific type of role (for example, "worker") and showing attributes for each possible role may be impractical (for example, 30 attributes for 30 possible workers). Doesn't explicitly delineate people versus organization roles. It is easy to mix declarative roles and contextual roles. It doesn't account for tracking history of roles.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 1 Contextual Role Pattern, Relationships	This pattern decouples the role information from the entity that has these roles. This means semantically that you recognize the fact that role information can be defined outside of its related entity.	When the same person or organization is playing a declarative role and may be involved in the same way many times (contextual role). When you need to capture rules about contextual roles in a rigorous fashion. When you need to be specific about the individual roles in a subject area, in other words as a specific scope statement or statement of requirements. When the Level 1 Declarative Role Pattern is used and the enterprise has not bought into the concept of a PARTY.	When a more flexible model is needed. In dynamic environments where roles or rules about roles get added or changed.	It specifically states rules about contextual roles. Begins to tie declarative roles to contextual roles. A strong semantic statement that contextual roles are significant outside of their related entities. They are captured as entities (or relationships) not as attributes.	It is not a very flexible model and does not easily accommodate new roles. It repeats PARTY information redundantly. This may lead to data quality issues. It doesn't account for tracking history of roles.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 2 Contextual Role Pattern	This pattern uses the declarative roles that may be captured for a PARTY as a basis for creating contextual roles that relate to the party's involvement in another entity.	When an enterprise wishes to integrate PARTY and PARTY ROLE data into the contextual roles. When you want to avoid capturing redundant PARTY and PARTY ROLE data. When there is a need to specifically model in order to better understand the information requirements. When you need to capture rules about contextual roles in a rigorous fashion.	Some enterprises don't use declarative roles (PARTY ROLE) and this pattern may not suit them. When you need to distinguish between person-only roles, organization-only roles, and party roles. When you need more flexibility and need a model that is more adaptable toward adding roles or changing the nature of roles.	This is an effective way to specify the scope of data requirements for roles. Specifies business rules in the data model. It does not redundantly capture PARTY and PARTY ROLE data. It integrates declarative roles and contextual roles together. (This can be considered a weakness in very dynamic environments.)	PARTY ROLE structure may be difficult for some enterprises to understand or use. Its specific nature also has the downside of not being flexible enough to allow change without changing the underlying data model. Does not distinguish between person-only, organization-only, and party roles. (However, this can be addressed by using PERSON ROLE and ORGANIZATION ROLE subtypes) You may need to add a new declarative role for a newly discovered contextual role if an existing declarative role does not exist. (This may be considered a strength for some enterprises.) It doesn't account for tracking history of roles for one-to-many relationships between the entity and the role.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 2 Contextual Role Pattern, PARTY Only Alternative	This pattern directly relates PARTY(s) to their contextual roles. This pattern supports the need to support contextual roles and how they are related to people and organizations directly.	When you want to avoid capturing redundant party data. When an enterprise does not use PARTY ROLE, or when an enterprise does not tie contextual roles via PARTY ROLE. When it makes more sense to have a relationship directly to a PARTY instead of using a PARTY ROLE, for example, when the roles are very general such as the "Receiver" and "Sender" of an invoice. When you need to capture specific rules about contextual roles. When you need to distinguish person-only roles from organization-only roles from party roles.	If an enterprise wishes to capture contextual roles via their PARTY ROLE(s). When there is a need for roles to be added or changed based on future requirements. When you need the semantic rigor of attaching contextual roles to declarative roles.	Easy to understand, relatively simple data model. Effective for documenting scope and business rules. Captures contextual roles and the specific nature of their relationship to the entity. Distinguishes person-only roles, organization-only roles, and party roles. Allows a little more flexibility because you can relate any PARTY to the associated entity without needing to specify that the party must be first declared as a particular role.	This pattern again has flexibility issues. When a new role is discovered, a new entity needs to be added. It doesn't account for tracking history of roles.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 3 Contextual Role Pattern	The Level 3 Contextual Role Pattern supports the need for a flexible solution that allows the dynamic addition (and retirement) of contextual roles for an entity.	When the scope of contextual roles may change. When flexibility is needed. Helps in creating a common set of business rules for all roles around the generalized CONTEXTUAL ROLE and ROLE TYPE entities. When modeling in a dynamic changing environment. As an implementation that is very adaptable to change.	As a scope statement or way of gathering and validating data requirements with nontechnical people. Some business rules (for example, cardinality and optionality rules) get lost because of the generalized nature of the pattern. If roles have different attributes, this pattern needs to be modified to be more specific. If an enterprise wishes to capture contextual roles via specific relationships to specific PARTY ROLE(s).	This is a flexible and consistent approach for supporting contextual roles. Accounts for history of who played what role and when.	Can be hard to understand for non–data professionals because it is very generalized. When you are using a generalized style of modeling such as in this pattern, the model does not capture specific business rules, specific cardinalities, and specific attributes. Because this pattern is more difficult to understand, it is more difficult to use this type of modeling structure as a means to capture information requirements. Because it uses a generalized ROLE TYPE it is possible to relate inappropriate roles that should not even be possibilities within a specific context.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Hybrid Contextual Role Pattern	This pattern supports both a specific and flexible way to model contextual roles. It's a single model that combines the level 2 Pattern and the level 3 Pattern.	When there are some core specific roles that need to be modeled as well as accommodating additional roles that may be needed over time. When both specific contextual roles and flexible contextual roles need to be captured. When an enterprise wants a single model to support both flexible and specific contextual roles.	When an enterprise does not subscribe to mixing modeling styles.	A very flexible solution that also provides for capturing specific rules about contextual roles. Provides a way to show different types of views to different audiences, while having to maintain only one model. Offers different options for the physical database design.	This is the most complex of the patterns. Two paths exist for a PARTY to be involved, so roles may be redundantly captured if the pattern is misused. The pattern could be confusing because of the mixed styles of modeling, and some may decide that having two different data models may be more advantageous. More difficult to find out all the possible roles since roles are maintained using two different modeling styles. Not as much freedom as the alternative of having two separate models that allows you to model as specifically as you want in one model and as generalized as desired in another model.

References

[1] See Chapter 2 of The Data Model Resource Book, *Revised Edition, Volume 1: A Library of Universal Data Models for All Enterprises* (Wiley, 2001).

[2] Paraphrased from dictionary.com (http://dictionary.reference.com/browse/context).

[3] Paraphrased from dictionary.com (http://dictionary.reference.com/browse/role).

[4] With permission from William G. Smith, Founder of William G. Smith and Associates, Syllabus for Conceptual Data Modeling Course, 2008.

[5] The project concept is really a part of the WORK EFFORT concept that is defined in Chapter 6 The Data Model Resource Book, *Revised Edition, Volume 1: A Library of Universal Data Models for All Enterprises* (Wiley, 2001).

[6] Paraphrased from dictionary.com (http://dictionary.reference.com/browse/project).

[7] See Chapter 2 of this book for more on declarative roles.

Chapter 4

Hierarchies, Aggregations, and Peer-to-Peer Relationships: The Organization of Similar Data

Account managers want to know their customer organizational hierarchy and how parent companies are related to subsidiaries that are related to lower-level organizations such as divisions and departments. Project managers want to know how programs are broken down into projects and then into activities and tasks, or in other words, how certain types of work efforts are further broken down into lower-level work efforts. Investment banks want to know how certain financial instruments such as options are constructed from other financial instruments. Manufacturers need to know the bill of materials and how parts may make up other parts. All these examples have one thing in common; they refer to a situation where some class of information is related to itself in some way. This chapter describes the patterns that support the organization of similar types of data into a structure. We refer to this type of self-association as a *recursive relationship*.

What Is the Significance of This Type of Pattern?

Most enterprises live and die by the reports they create and analyze. People working in the enterprise need to summarize the information in the reports or drill into the

details of the data.(1) Hierarchies, aggregations, or peer-to-peer relationships of a class of data allow people to drill up and down through the details of a report. For example, a manager may start wondering why a project will take six months to complete. Managers often want to drill into different levels of a project to better understand the details of the phases, activities, and tasks within the 6-month time frame.

Another common example is how a product is broken down into components (other products). Engineers, marketers, and manufacturers create bills of materials (BOM) showing all the components in a product. This is crucial for material resource management and manufacturing. These are often called parts lists and often get supported by recursive relationships.

Risk departments at large financial institutions need to be able to evaluate the different atomic-level securities that go to make up the composite products (e.g., mutual funds, options, derivatives) that reside on their balance sheets. If they can't reconstruct how these products are structured, they are unable to hedge their financial position against market change (market risk) or customer bankruptcy (credit risk).

What Is in This Chapter?

This chapter initially defines the different ways in which an enterprise organizes types of data that are related to other similar types of data. The chapter then describes the data model patterns used to support the needs of an enterprise to organize this type of data.

Like most chapters in this book, the style of modeling for each of the patterns starts with a very specific style (Level 1 Recursive Pattern) and progresses through the chapter to a very flexible style (Level 3 Recursive Pattern). The different

levels of generalization may be applicable to different enterprises or styles of modeling.

This chapter includes:

- The definition of a recursive relationship and the different ways data is organized by recursive relationships
- The different patterns that support recursive relationships
- The relevance of each pattern
- Insights into each pattern
- When to use and not to use different patterns
- A synopsis of each of the patterns including pros and cons

What Is a Recursive Relationship and How Is Data Organized by Recursive Relationships?

One definition of a recursive relationship is “A semantic connection between many objects of the same class.”(2) Another is “Involuted, or recursive, relationships, are self-relationships; relationships from and to the same entity.”(3) Each of the definitions refers to the crucial fact that recursive relationships are relationships where instances of an entity are related with other instances of the same entity. It's this self-reference that makes the relationship recursive.

Note

Data professionals have many names for this type of relationship, and they use those names interchangeably. Self-referencing relationship, involuted relationship, “pig's ear” relationship, recursive association, and recursive relationship

are a few common terms. Recursive relationship seems to be the most popular term, and is probably the term with the most semantic rigor. This is the term we will use in this chapter, for the most part.

Recursive relationships support different ways of collecting data into groups and associating data of the same class with other data of the same class. For example, a bill of materials is the structure that supports a product (or parts) being made up of other products (or parts). Similarly an organizational hierarchy records which customer organizations may be made up of other customer organizations. Three of the most significant types of recursive relationships are hierarchies, aggregations and peer-to-peer relationships, which are described as follows:

- A hierarchy can be described as “an organization with few things or one thing at the top and with several things below each other thing, an inverted tree structure.”(4) An example would be a directory hierarchy in computing.

  Hierarchies imply ownership. The top of the hierarchy owns all the children underneath. This means that if the owning object is destroyed, the hierarchy is destroyed also. For example, a university has several schools (for example, business school, school of sciences, and so on) and each school owns various departments (for example, chemistry, and so on). If the university closes, the schools and departments will no longer exist! Figure 4.1 is a graphical representation of this idea. Another interesting point about hierarchies is that they usually are a series of one-to-many relationships in the same direction. In other words, each “child” has only a single “parent” (or there could be many “parents” if it is a many-to-many hierarchy), and a “parent” may have one or more “child” entities.

- An aggregation can be described as a total, considered with reference to its constituent parts. “An empire is the aggregate of many states under one common head” (Edmund Burke).

  How is this different from a hierarchy? Aggregation differs from hierarchies in that destroying the owner does not destroy the elements of the hierarchy. For example, many countries in Europe are members of an aggregation of states called the EU (European Union). Many of the member states (UK, Ireland, France, Finland, Greece, and so on) of the EU were formally members in the EEC (European Economic Community). The EEC was superseded by the EU; when the EEC no longer existed, none of the states disappeared. They just became members of a new aggregation called the EU. Figure 4.2 describes an aggregation of states in the EU.
- A peer-to-peer relationship can be defined as a connection of persons, things, or ideas by some common factor; a union.(5) This is when data items in the same class are related to each other in a different way than a parent-child relationship. An example would be a shipment for a manufacturing firm where the vendor shipment is needed in order to fulfill a customer shipment.

  How does this type of relationship differ from a hierarchy and from an aggregation? Peer-to-peer relationships are between elements that are at the same level so there is not a higher level element that has numerous lower level elements. They are also different from a hierarchy because a hierarchy implies ownership, whereas a peer-to-peer relationship does not have ownership. Deleting or destroying one element in the peer-to-peer relationship does not cascade and delete all the associated peers. In other words, there is no owning element. It is also worth

noting that peer-to-peer associations support many-to-many, one-to-many, and one-to-one relationships in different directions. Figure 4.3 describes this concept in a diagram.

> **Note**
>
> We concentrated on the three most common ways of collecting data into groups using recursive relationships, but there are other examples of structures that a recursive relationship will support, such as collections (a group of objects accumulated in one location as a result of some process, for example, a stamp collection or a collection of statistics about purchasing(5)) and compositions (the act of combining parts or elements to form a whole,(5) for example, a piece of music made up of its notes). We also note that hierarchies and aggregations don't have to be exclusively for recursive relationships, but they frequently occur in recursive relationships, which is the reason we decided to discuss them in this setting. Peer-to-peer associations should occur only in recursive relationships.

Figure 4.1 A hierarchy of university, schools, and departments

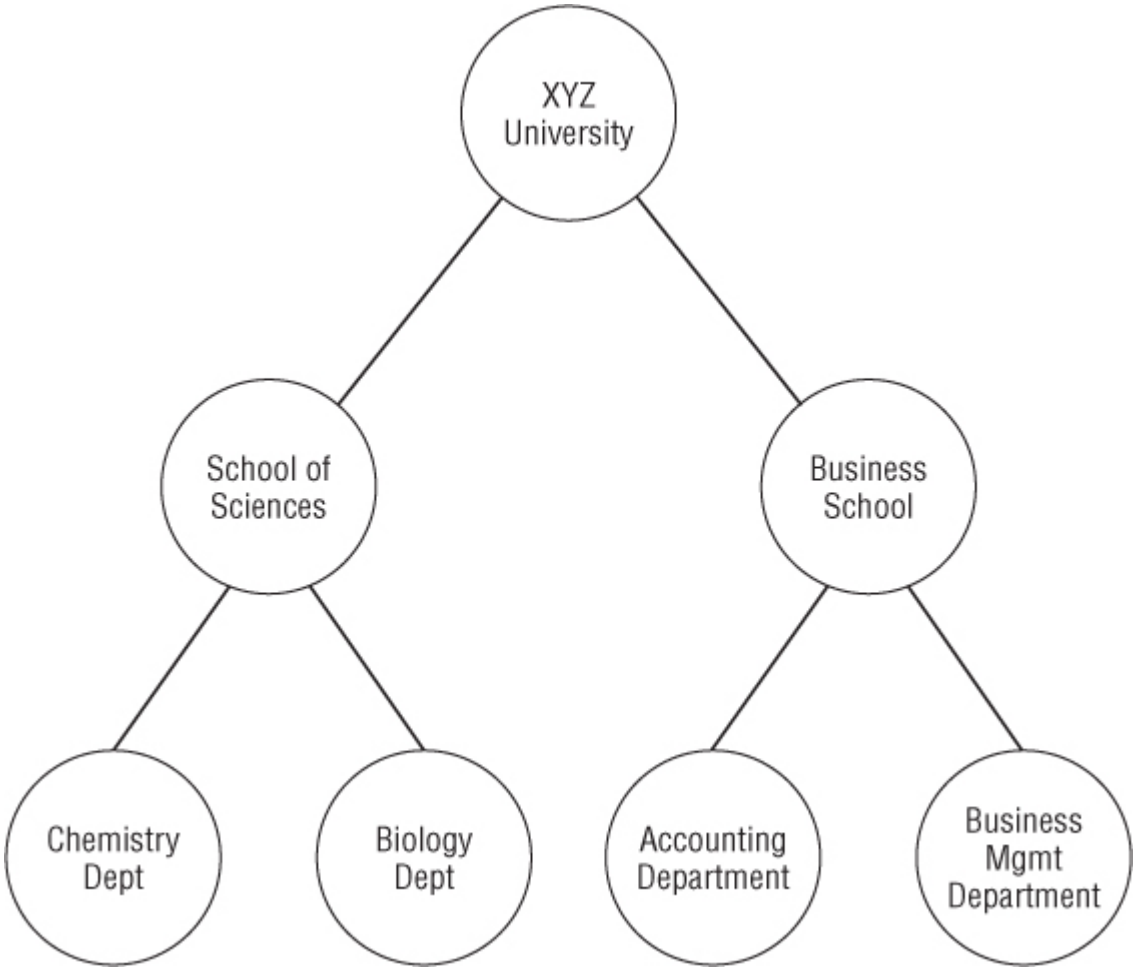


Figure 4.2 An aggregation of states in the European Union

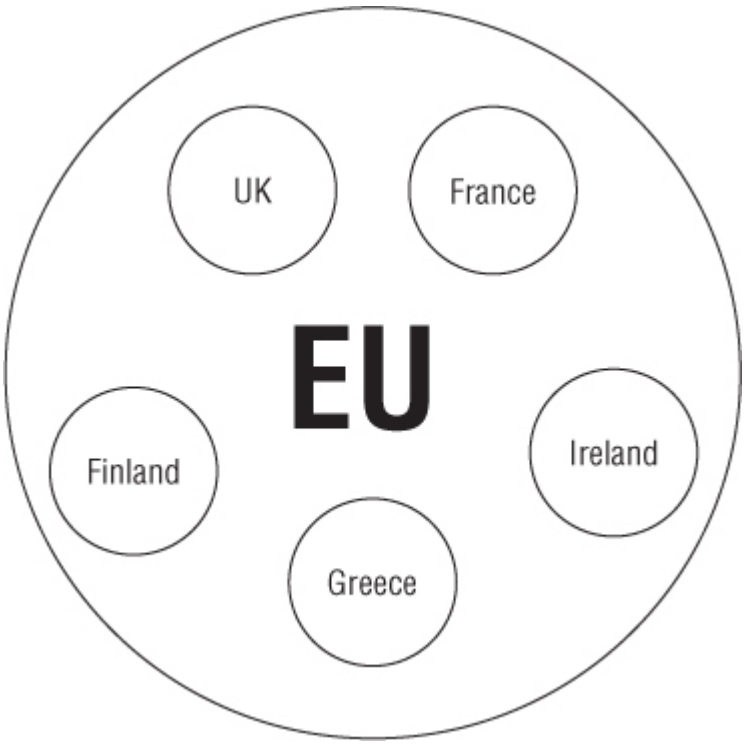
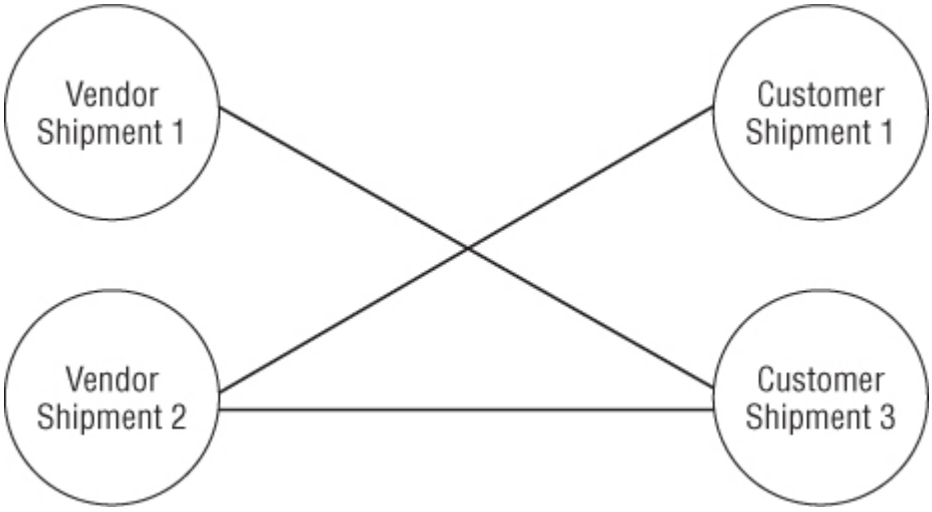


Figure 4.3 A peer-to-peer association of shipments



Why is it important to make the distinction between the different ways a recursive relationship groups data? Understanding the nature of how a class of data is related helps interpret the meaning and significance of the data and how you manage that data. Imagine a mobile phone manufacturer that decides to stop manufacturing a particular model of phone. The design for the creation of the

phone is an aggregation (bill of materials), describing how the phone is made up of different parts. The phone is no longer manufactured so this bill of materials becomes obsolete. Even though the complete hierarchy is now obsolete, each of the parts within the bill of materials may still exist and be used within other bill of materials for other products.

Now examine the example of the hierarchy of university, schools, and departments as seen in Figure 4.1. It can be said that the "Accounting Department" and the "Business Management Department" are members of a hierarchy called the "Business School." As we have previously stated, if the university closes, the schools close and the departments close. In other words if the "owning" entity of "XYZ University" is destroyed, the hierarchy and each of the members in the hierarchy get destroyed. We should be able to manage and understand the different organizations of the data. How else would we know that the consequence of deleting "university" is to cascade through the "schools" and "departments?

Level 1 Recursive Pattern

Some enterprises need to describe very specific hierarchies and aggregations. For example, a U.S.-based company capturing the countries, states, and cities within North America may claim that a country, such as the United States, has a set of states, which is further broken down into set of cities. The important thing about this aggregation is that it is an example of a static structure. In other words, COUNTRY (USA) breaks down into STATES (NY, NJ, CA, and so on) that further breaks down into CITY(s) (New York, Los Angeles, and so on).

Why Do We Need This Pattern?

The Level 1 Recursive Pattern provides the most specific way to model a hierarchy (or aggregation) of data. When an enterprise needs to support unchanging hierarchies or aggregations that have strict business rules about how one level relates to another level, the Level 1 Recursive Pattern may be applicable. This pattern is not generally used for modeling peer-to-peer relationships.

Another reason to use the Level 1 Recursive Pattern is as a visual representation of a requirement for a hierarchy or aggregation. The pattern unambiguously shows how a hierarchy or aggregation is broken down. Business people or nontechnical people can relate to this very specific structure more easily than to a more generic flexible structure.

This pattern supports the basic aspects of a hierarchy or aggregation:

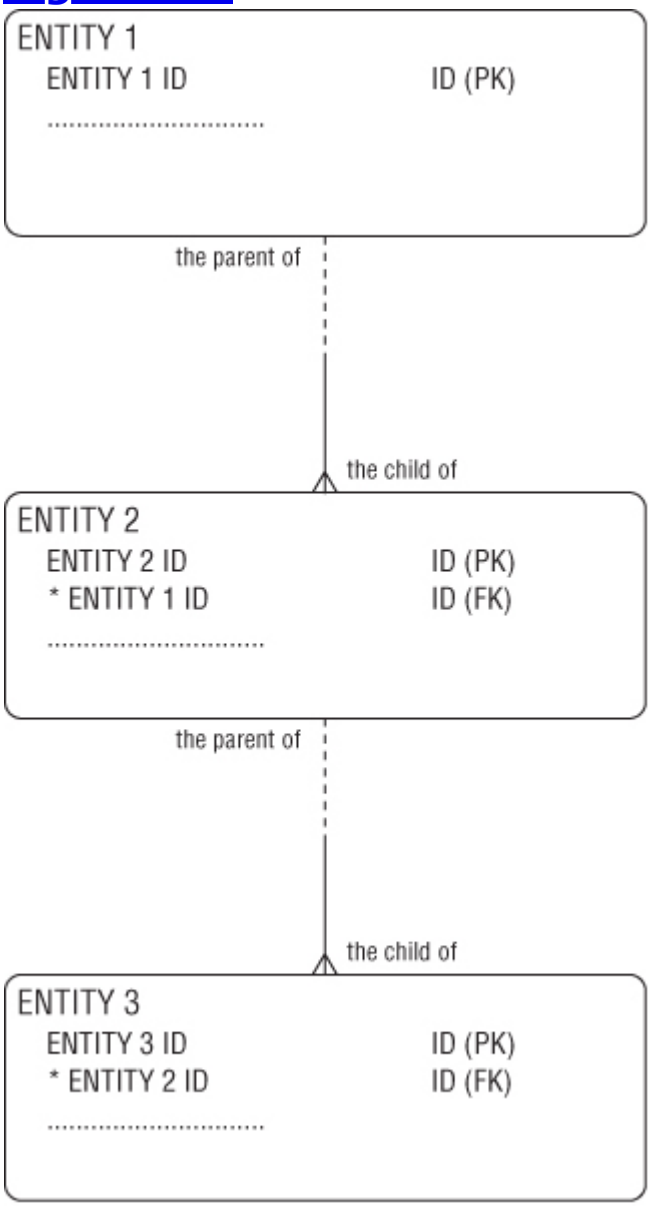
- A definition or representation of the different levels of the hierarchy or aggregation
- The relationships that the levels of the hierarchy have to each other, that is, the specific business rules that define where each entity is in the hierarchy or aggregation

How Does This Pattern Work?

Figure 4.4 describes a pattern that uses a very specific style of modeling a three-level hierarchy/aggregation. Each of the entities represents a different level in the hierarchy/aggregation. ENTITY 1 is the top of the hierarchy/aggregation. ENTITY 2 represents the mid-level of the hierarchy/aggregation, that is, the “child” of ENTITY 1 and the “parent” of ENTITY 3. ENTITY 3 is the lowest level of the hierarchy/aggregation and it has no “children.” An enterprise can expand this pattern to meet the number of levels that it requires just by adding a new entity (or entities) onto the tail of the hierarchy/aggregation (ENTITY

3) or by inserting new entities between levels, or above ENTITY 1.

Figure 4.4 Level 1 Recursive Pattern

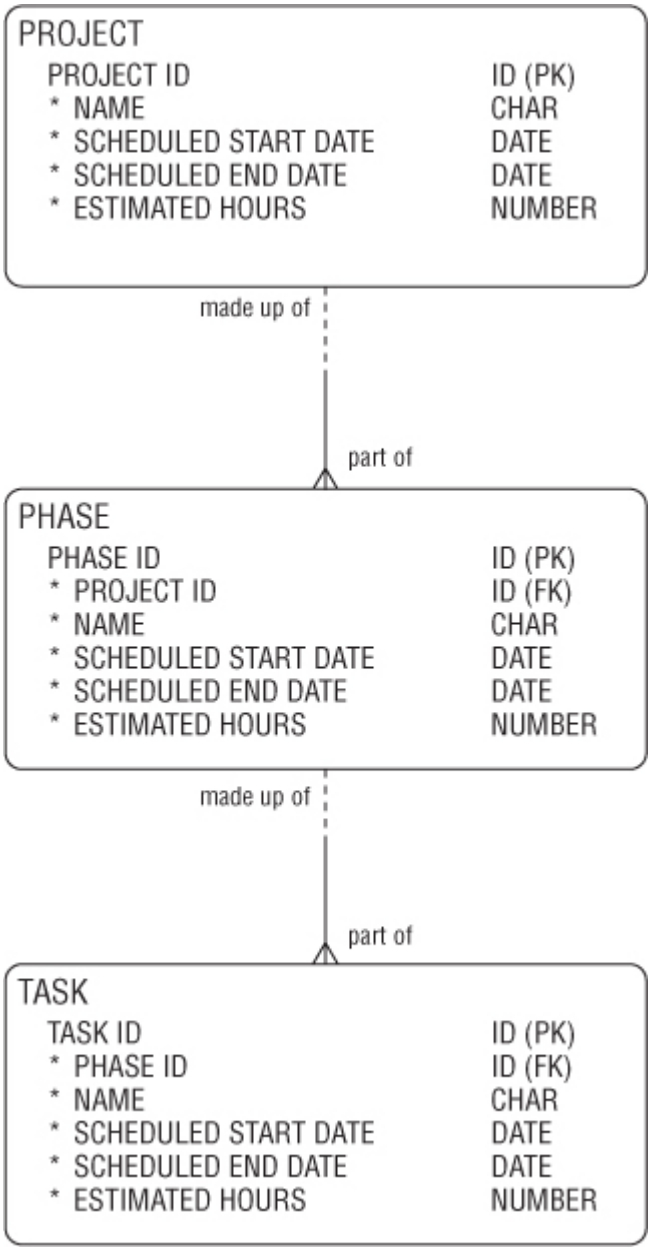


In this model you see that “each ENTITY 1 *may be* **the parent of** one or more ENTITY 2(s) and each ENTITY 2 *must be* **the child of** one and only one ENTITY 1” and “each ENTITY 2 *may be* **the parent of** one or more ENTITY 3(s) and each ENTITY 3 *must be* **the child of** one and only one ENTITY 2.”

Imagine a scenario where XYZ Corporation is a large international technology services firm concerned that it does not capture and control information about the projects it works on in a very organized and comprehensive manner. As part of an effort to rectify this situation, the firm commissioned a study to examine how projects were structured. A data professional was employed to examine all projects that the technology services company worked on.

The data professional created Figure 4.5 to start the discussion with the key project managers in XYZ Corporation. Figure 4.5 has a three-level hierarchy with a PROJECT being the highest level of work effort that is tracked by the enterprise. A PROJECT is defined as “a large or major undertaking, esp. one involving considerable money, personnel, and equipment.”(5) “Each PROJECT *may be* **made up of** one or more PHASE(s).”

Figure 4.5 Example of using a Level 1 Recursive Pattern



At the next level is a PHASE. “Each PHASE(s) *must be* **part of** one and only one PROJECT.” The PHASE can be defined as “a stage in a process of change or development.” And “each PHASE *may be* **made up of** one or more TASK(s).”

Finally, at the lowest level, are TASK(s). A TASK can be defined as “a definite piece of work assigned to, falling to, or expected of a person.”(5) The TASK was the lowest level of work that XYZ Corporation evaluated.

Note

We consider PROJECT, PHASE, and TASK to be WORK EFFORT(s). Volume 1 of *The Data Model Resource Book* contains a comprehensive explanation of this concept.(6) In general a WORK EFFORT can be described as "a planned, in progress, or completed work activity that is performed. It may be the activity related to the fulfillment of a work requirement."(7) This term becomes significant in the following sections.

Table 4.1 further illustrates how this pattern can be applied. In Table 4.1 there is one project, with a **project name** of "Enterprise Data Warehouse." This project has four PHASE(s) with phase **name**(s) of "Mapping," "Systems Analysis," "ETL Design," and "Testing." Each PHASE(s) has its own **scheduled start date(s)** and **estimated hours**. For instance, "Mapping" began on "June 3, 2009" and was scheduled to run for "700" hours. Each of the PHASE(s) has TASK(s). For example, the "Mapping" phase contains two tasks, "Create Mapping Template" and "Create Mapping." Each task has its own **scheduled start date(s)** and **estimated hours** - for example, the "Create Mapping Template" TASK was scheduled to start on "July 20, 2009" and was estimated to take "30" hours."

Table 4.1 Example of a Level 1 Recursive Pattern

PROJECT. NAME	PROJECT. SCHEDULED START DATE	PROJECT. ESTIMATED HOURS	PHASE. NAME	PHASE. SCHEDULED START DATE	PHASE. ESTIMATED HOURS	TASK. NAME	TASK. SCHEDULED START DATE	TASK. ESTIMATED HOURS
Enterprise Data Warehouse	Jan. 1, 2009	2300	Mapping	June 3, 2009	700	Create Mapping Template	July 20, 2009	30
Enterprise Data Warehouse	Jan. 1, 2009	2300	Mapping	June 3, 2009	700	Create Mapping	July 30, 2009	500
Enterprise Data Warehouse	Jan. 1, 2009	2300	Systems Analysis	Feb. 4, 2009	400	Create Initial Scope Statement	Feb. 28, 2009	90
Enterprise Data Warehouse	Jan. 1, 2009	2300	Systems Analysis	Feb. 4, 2009	400	Create Source System Inventory	Mar. 13, 2009	100
Enterprise Data Warehouse	Jan. 1, 2009	2300	Systems Analysis	Feb. 4, 2009	400	Discover Target Requirements	Apr. 30, 2009	100
Enterprise Data Warehouse	Jan. 1, 2009	2300	ETL Design	Sept. 5, 2009	200	Create Common Strategy	Sept. 24, 2009	40
Enterprise Data Warehouse	Jan. 1, 2009	2300	ETL Design	Sept. 5, 2009	200	Create Common Transformations	Oct. 9, 2009	70
Enterprise Data Warehouse	Jan. 1, 2009	2300	ETL Design	Sept. 5, 2009	200	Create One Off Transformations	Oct. 30, 2009	100
Enterprise Data Warehouse	Jan. 1, 2009	2300	Testing	Dec. 1, 2008	110	Create Integration Test Plan	Dec. 27, 2008	40
Enterprise Data Warehouse	Jan. 1, 2009	2300	Testing	Dec. 1, 2008	110	Create System Test Plan	Nov. 3, 2008	40

If the “Enterprise Data Warehouse” PROJECT were cancelled, the four instances of PHASE and all their TASKS would also cease to exist. This is typical of a hierarchical structure. As we discussed earlier, the hierarchy implies ownership, and if the “owning” object is destroyed, the “owned” objects or instances are also destroyed.

Some interesting issues crop up because of the structure seen in Figure 4.5. The **estimated hours** attribute gets enumerated at all three levels. For example, the PROJECT has **estimated hours** of “2300” hours, the total number of **PHASE estimated hours** is “1410” hours (“700”+“400”+“200”+“110”), and finally, the summation of all **TASK estimated hours** works out to be “1110” hours. This is a strength and a weakness of this pattern. Each of the difgferent estimates may be independent of each other. You can define the estimated hours for a PROJECT to be different from the estimated hours for a PHASE. For example, at the beginning of the project you may do the project estimate; then as you progress through each phase, you may estimate the hours on a phase level and so on. If you wanted to maintain hours only at the TASK level, you could delete the **estimated hours** attributes from PHASE and PROJECT. In other words, the Level 1 Recursive Pattern allows you to capture specific attributes at any level in any hierarchy/aggregation. This means that you can support very specific business rules about the hierarchy or aggregation. In our example, we could have included a **percentage complete** attribute at only the TASK level and then used that to estimate the percentage complete for PROJECT(s) and PHASE(s). As we will see later in this chapter, more generalized patterns don't give you this opportunity.

This specific structure of maintaining the same attributes at each level may lead to some misunderstanding. If you are not careful, you could calculate the total estimated project

hours by summing the PHASE **estimated hours** or TASK **estimated hours** and then there is a conflict regarding how many hours were estimated for the project. Which total is correct? If this structure were implemented, where would the developers look for the total number of estimated hours in a project—at the top level, the second tier, or the bottom? Is the discrepancy in hours "overhead" or "other" hours? The programmers should look in PROJECT, but if a PROJECT is made up PHASES, could they not derive the estimated hours for a project based on the total **estimated hours** of all of the PHASE(s) or the total **estimated hours** of all TASK(s)? The relationships between these entities would lead you to believe this to be possible.

Another interesting issue has arisen in Table 4.1 in the last row. For the task with **task name** "Create System Test Plan" there is a **scheduled start date** of "Nov. 3, 2008." But at the phase level, the phase with **phase name** "Testing" has a **scheduled start date** of "Dec. 1, 2008." How is it possible that a task can start before its phase?

This again is a strength of the pattern in one sense because it provides the ability to specify different meanings and definitions for the attributes at each of the levels. Thus, the **scheduled start date** may be defined differently at each of the different levels in the structure, which could mean that it's perfectly possible to specify in the definition that a TASK may start before a PHASE. For example, the **scheduled start date** for the PHASE may be defined as "The date upon which the phase will start. This date may be before, during, or after the scheduled start date for a task". We will see that with the more generalized patterns in this chapter, that all the levels in the structure must have the same definition for attributes such as the **scheduled start date**, regardless of their level.

Other business examples of data structures that may use this pattern are:

- Bill of materials structures where a FINISHED GOOD may be made up of SUBASSEMBLY(s), which is then made up of PART(s)
- Geographic boundary structures where there may be a COUNTRY(s) that consist of STATE(s), which consist of CITY(s), and so on.
- Organizational structures where a PARENT COMPANY is broken down into its SUBSIDIARY(s) and then into DIVISION(s) and then DEPARTMENT(s)

When Should This Pattern Be Used?

The factors that we take into account when deciding to use this pattern are as follows:

- **When there is a well-defined, static hierarchy** If the area under investigation is well understood by XYZ Corporation and is not anticipated to change in the future, this pattern may be applicable.
- **When there is a need to incorporate business rules into the data model that govern the hierarchy or aggregation** There may be specific rules that are important to represent in the data model, for example, “Each PHASE must be the child of one and only one PROJECT” showing that a phase can be in only one project.
- **When there is a need to define specific attributes or relationships at different levels of the hierarchy** Although it is not shown in the example, this pattern allows each level to have different attributes. For example, TASK may have an attribute **percentage complete** that none of the other levels have. Also, if the enterprise decides that the estimated hours for a PROJECT *must* always equal the estimated hours for the related PHASE(s), you can just delete this attribute from PROJECT because it can be derived. Also, as we

discussed, even if the attribute names are the same, there may be different definitions for the attribute at different levels of the structures.

- **As a method to stimulate data analysis** In this case the XYZ Corporation project managers could use the model to confirm the levels of the hierarchy that were needed. Second, the model raised awareness of problems such as evaluating and maintaining **estimated hours** at different levels in the hierarchy. It also started to capture common terms. Finally, it helps start the data modeling effort in a more straight forward manner by providing a very simple structure for modeling hierarchies or aggregations.
- **A situation where the data professional needs to understand the business requirements more easily and perhaps use as part of a statement of scope to other IT professionals and/or business people** The diagrams in this section were a straightforward statement of scope showing the three different levels in the hierarchy that were of interest to the enterprise.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This is a very rigid structure that does not easily allow change** What if the enterprise decides that projects will be managed by programs, and within programs there are projects, then phases, then activities, and then tasks? This would require the data model and underlying data structures to be significantly modified.
- **This pattern does not support many-to-many relationships.(8)** The pattern shows only one-to-many

relationships which is very common for hierarchies or aggregations. Of course, you could modify the pattern to allow many-to-many relationships.

- **Having attributes with similar meaning and similar data in different levels of the same hierarchy may lead to confusion and data quality issues** For example, can the **scheduled start date** of a TASK be before the **scheduled start date** of a PROJECT? Can the **estimated hours** of a PROGRAM be less than the total of all the **estimated hours** of its child PROJECTS? It is worth noting that this is different from saying that you are capturing redundant data. The attributes at the different levels in the hierarchy may not have the same definitions. But in our experience they often do. It is also fair to say that if you do capture similar data at different levels, you can get rid of the "redundant" attributes at the higher level in the hierarchy and derive its value based off the same attributes in the lower levels. For example, you can sum the **estimated hours** in TASK to derive the estimated hours for a PHASE.
- **This structure handles hierarchies and aggregations, but it does not handle peer-to-peer relationships at all.**

Synopsis

In this section you saw that the Level 1 Recursive Pattern can be used by a data professional to model a hierarchy or aggregation in a very specific fashion. It allows the data professional to clearly illustrate the data requirements and concepts that people need to see. This pattern is also significant because you can capture attributes and relationships that may be specific at different levels of the same hierarchy and capture very specific business rules regarding the structure of a hierarchy or aggregation.

It is a risk to assume that the structure of a hierarchy or aggregation will never change. Even what seems to be the most static hierarchy or aggregation can change sometime in the future. When using this pattern, if there are changes in the hierarchy, the resulting data model will no longer be valid and will need to be changed which could create significant rework. The data model may maintain the similar data at the different levels in this pattern. For example, **estimated hours** were captured at PROGRAM, PROJECT, and TASK levels in the hierarchy. This may lead to misunderstanding and data quality issues, because data may be handled differently at different levels and the relationships between these elements may not be handled consistently.

For an enterprise, a significant issue with the specific hierarchy pattern is that it is rigid in structure. This can be viewed as a strength in some circumstances, because the data model and subsequent data structure can describe specific data requirements and enforce specific business rules. However, it is a weakness because if the information requirements change, then there are required changes to the model, which could result in changes to the data structures that are created from the model.

Level 2 Recursive Pattern

Some enterprises need to be very specific and thus use the more specific pattern when handling the recursive relationships. The Level 1 Recursive Pattern helps these enterprises support this need. There are situations when an enterprise needs a more flexible solution. For example, a manufacturing company might have a set of products that are made up of components which are further made up of subcomponents and there could be any number of levels in the hierarchy/aggregation. The engineers in the

manufacturing company might be very inventive and keep discovering better ways to simplify the design of the products that the company manufactures. Thus, the structure of the bill of materials might evolve as time progresses, and the levels needed in the bill of materials structures may continue to change.

Why Do We Need This Pattern?

This pattern supports a flexible solution for the creation of hierarchies, aggregations, or peer-to-peer relationships. This pattern is more generalized than the level 1 pattern and it avoids some issues pointed out at the end of the previous section, including having to change the data model when new associations are discovered.

How Does This Pattern Work?

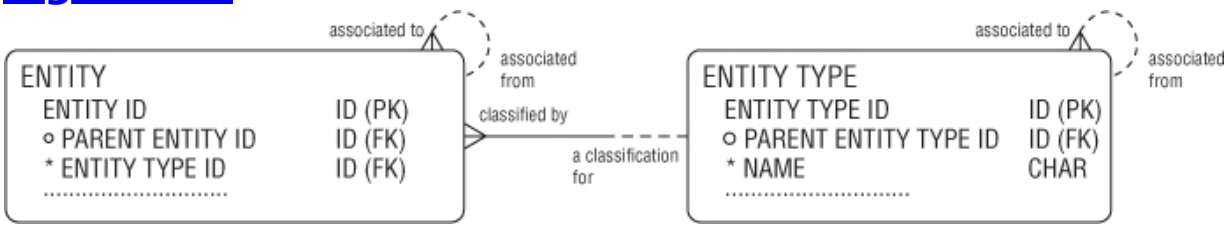
Figure 4.6 shows a pattern for modeling recursive relationships in a more flexible way than the previous pattern. In the diagram you see the one-to-many recursive relationship that supports structures such as hierarchies, aggregations and peer-to-peer relationships on the side of ENTITY. You also see the added ENTITY TYPE. Recursive relationships often support the relationships between different types of the entity. ENTITY TYPE has its own recursive relationship. This supports the need to relate the different entity types into an aggregation or hierarchy. For example, a PROJECT type would be the parent of a PHASE type, which would be the parent of a TASK type.

Note

It is possible that an entity with a recursive relationship does not have a type entity associated with it. In other words, all instances of an entity are of the same type. For example, it may be that the enterprise has a PART entity that is made up of other PARTs and there is not a need to have a PART TYPE. But more

often than not, we have found that recursive relationships in entities create structures (hierarchies, aggregations), and each level of the structure has a “label.” These “labels” are the types. Examples of these labels are “project,” “phase,” and “task” for a generalized entity of WORK EFFORT as well as “finished good,” “subassembly,” and “raw materials” for a generalized entity of PART.

Figure 4.6 Level 2 Recursive Pattern



If you look at Figure 4.6 you see that ENTITY 1 has an optional foreign key to itself, **parent entity id**, which supports the requirement that “each ENTITY *may be* **associated from** one and only one ENTITY and each ENTITY *may be* **associated to** one or more ENTITY(s).” This means that an instance of ENTITY may or may not have a parent.

Note

A very common mistake we have seen in many data models is making recursive relationships (supporting a hierarchy structure) mandatory. If this relationship were mandatory, the “top of the tree” could not be supported. Imagine a large company with many employees. If you went from the mailroom right up through the list of employees via their managers, you would eventually get to the CEO and/or owner(s) of the company. He or she does not have a boss, and thus, there cannot be a mandatory parent relationship for this instance!

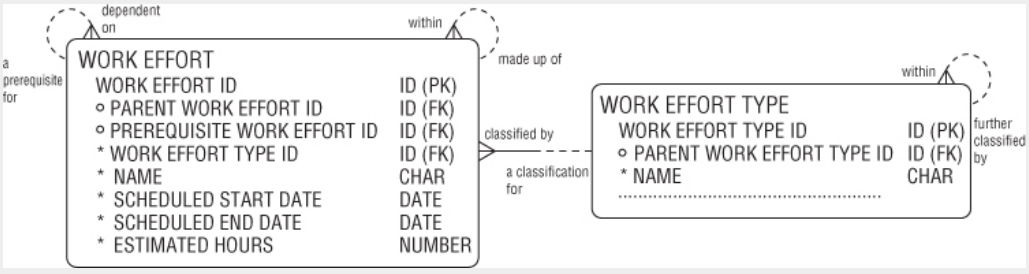
Because “each ENTITY *may be* **associated from** one and only one ENTITY,” in this pattern, each instance of the ENTITY can have only one parent. This may best be explained by example. Suppose a customer is ordering an X25 Flip phone (the name of a type of mobile phone) with

many features, such as a red or blue cover, an extra large memory card or a standard memory card, and so on. The customer picks the features that he/she wishes from the available choices, in other words “An X25 Flip Phone with a red cover and an extra large memory card.” The order items for the order for the “X25 Flip phone” are related to each other. Thus in this example, the pattern could be used to model the recursive relationship between ORDER ITEM(s), that is, the order item for “X25 Flip phone” is related to another order item for a “Red cover” as well as related to another order item for an “Extra large memory card” for that single order. Thus, the order item for a “Red cover” is within one and only one main order item, namely the “X25 Flip phone” order item, and the same goes for the order item for the “Extra large memory card”.

Note

This pattern could include more than one recursive relationship. For example, one recursive relationship may support dependencies (an instance that is dependent on another instance), another may support concurrency (two instances needing to happen at the same time), and so on. For simplicity we put in only one recursive relationship, but you are not limited by this. You see this illustrated in Figure 4.7.

Figure 4.7 Example of using a Level 2 Recursive Pattern



If we continue with the scenario of the XYZ Corporation described in the previous section, you can see that the data professional initially produced Figure 4.5 based on the Level

1 Recursive Pattern to show the initial scope of the levels of WORK EFFORT(s) the company captured. Some further discussion with stake holders established that PROJECT, PHASE, and TASK were in fact just WORK EFFORT(s). In other words, the stake holders stated that PROJECT, PHASE, and TASK had the same attributes and relationships such as scheduled dates, estimated hours, parties assigned, and so on; therefore they could be characterized as WORK EFFORT(s) in general. The different WORK EFFORT(s) were related to each other in many different ways. On that basis, the data professional produced Figure 4.7. In addition to capturing that work efforts may be made up of other work effort, XYZ Corporation also wanted to capture the fact that a project, phase, or task may be dependent on another project, a phase, or a task.

In Figure 4.7 you see that "each WORK EFFORT *may be* **made up of** one or more WORK EFFORT(s)." For example, instances of "projects", which are a type of WORK EFFORT, captured in WORK EFFORT TYPE, may be made up of instances of "Phase", which is another type of WORK EFFORT and therefore an instance of WORK EFFORT TYPE. And those instances of "Phase" are made up of instances of "Task", another instance of WORK EFFORT TYPE. If XYZ Corporation decided that it needed to add additional levels such as "Program" or "Activity", the model accommodates this without having to change. All you need to do is add "Activity" and "Program" to WORK EFFORT TYPE and then add the instances of specific programs or activities to WORK EFFORT.

Table 4.2 illustrates instances of this model. "Enterprise Data Warehouse" with **work effort id** of "9002" was scheduled to start on "Jan. 1, 2009," and was estimated to take "2300" hours. Notice that the instance of WORK EFFORT for "Enterprise Data Warehouse" does not have a value in WORK EFFORT **parent work effort id** or a WORK

EFFORT **name** for the parent. This indicates that “Enterprise Data Warehouse” is the parent or top node of the hierarchy of WORK EFFORT(s). “Enterprise Data Warehouse” has a WORK EFFORT TYPE **name** of “Project.” XYZ Corporation previously stated that the structure it uses to manage its work efforts consisted of projects that were made up of phases and phases that were made up of tasks.

Table 4.2 Example of Level 2 Recursive Pattern, WORK EFFORT(s)

WORK EFFORT.WORK EFFORT ID	WORK EFFORT. NAME	WORK EFFORT TYPE. NAME	WORK EFFORT. PARENT WORK EFFORT ID	WORK EFFORT. NAME (PARENT)	WORK EFFORT TYPE.NAME (PARENT)	WORK EFFORT. SCHEDULED START DATE	WORK EFFORT. ESTIMATED HOURS
9002	Enterprise Data Warehouse	Project				Jan. 1, 2009	2300
9004	Mapping	Phase	9002	Enterprise Data Warehouse	Project	June 3, 2009	700
9005	Create Mapping Template	Task	9004	Mapping	Phase	July 20, 2009	30
9010	Create Mapping	Task	9004	Mapping	Phase	July 30, 2009	500
9006	Systems Analysis	Phase	9002	Enterprise Data Warehouse	Project	Feb. 4, 2009	400
9007	Create Initial Scope Statement	Task	9006	Systems Analysis	Phase	Feb. 28, 2009	90
9008	Create Source System Inventory	Task	9006	Systems Analysis	Phase	Mar. 13, 2009	100
9009	Discover Target Requirements	Task	9006	Systems Analysis	Phase	Apr. 30, 2009	100
9050	Conduct interviews	Subtask	9009	Discover Target Requirements	Task	Apr. 30, 2009	20
9051	Model target Requirements	Subtask	9009	Discover Target Requirements	Task	Apr. 30, 2009	80

Another WORK EFFORT is named here, “Mapping.” “Mapping” has a **work effort id** of “9004,” was scheduled to start on “June 3, 2009,” (i.e., **scheduled start date**) and was estimated to take “700” hours (i.e., **estimated hours**). “Mapping” is a “Phase” type of WORK EFFORT. It has a **parent work effort id** of “9002,” which is the foreign key to the “Enterprise Data Warehouse” WORK EFFORT. This means that the “Mapping” WORK EFFORT is a child of the “Enterprise Data Warehouse” WORK EFFORT. The “Systems Analysis” WORK EFFORT, with **work effort id** of “9006,” also is a child of the “Enterprise Data Warehouse” WORK EFFORT. This structure can be viewed like a directory tree, with “Enterprise Data Warehouse” as the root directory and the “Mapping” and “Systems Analysis” WORK EFFORT(s) as subdirectories of “Enterprise Data Warehouse.”The “Mapping” and “Systems Analysis” WORK EFFORTS have their own WORK EFFORT(s) in the form of individual tasks. In the case of “Mapping” you see the tasks “Create Mapping Template” and “Create Mapping.” In the case of “Systems Analysis,” you see TASK(s) “Create Initial Scope Statement,” “Create Source System Inventory,” and “Discover Target Requirements.”

After starting the effort, XYZ realized that they needed to further break down tasks into subtasks. The last two rows in Table 4.2 illustrate the flexibility of this pattern to accommodate new requirements such as this one, without changing the data model. Thus a new WORK EFFORT TYPE **name** of “Subtask” may be added and the last two rows show that the task “Discover Target Requirements” is made up of “Conduct Interviews” and “Model Target Requirements.”

Some work efforts may depend on others being completed before they can start. Figure 4.7 shows that “Each WORK EFFORT may be dependent on one and only one WORK EFFORT. Table 4.3 illustrates an example of this and shows

that the “Create Mapping” instance of TASK is dependent on the “Create Mapping Template” being completed. This is an example of using this pattern not for a hierarchy or aggregation, but for a peer-to-peer relationship.

Table 4.3 Example of Level 2 Recursive Pattern, DEPENDENCY

WORK EFFORT. WORK EFFORT ID	WORK EFFORT. NAME	WORK EFFORT TYPE. NAME	WORK EFFORT. WORK EFFORT ID (FOR PREREQUISITE)	WORK EFFORT. NAME (FOR PREREQUISITE)	WORK EFFORT TYPE. NAME (FOR PREREQUISITE)
9010	Create Mapping	Task	9005	Create Mapping Template	Task

In Figure 4.7 there are two recursive relationships, one stating that “each WORK EFFORT *may be* **made up of** one or more WORK EFFORT(s)” and the other stating that “each WORK EFFORT *may be* **a prerequisite for** one or more WORK EFFORT(s) and each WORK EFFORT *may be* **dependent on** one and only one WORK EFFORT” (if work efforts are dependent on more than one other work effort being completed, a many-to-many relationship is needed, and this is covered in the next pattern). This illustrates the point that an entity can have more than one recursive relationship.

The following bullet points show other examples where you can generalize other types of specific relationships into a common, generalized entity that has a recursive relationship:

- Bill of materials structures where a FINISHED GOOD may be made up of SUBASSEMBLY(s) that is then made up of PART(s) can be generalized into an entity called PART with a recursive relationship around it and a type entity to classify it as a “finished good,” subassembly,” or “raw material.”
- Geographic boundary structures where there may be a COUNTRY that consists of STATE(s) that consist of

CITY(s) may be generalized into a common entity called GEOGRAPHIC BOUNDARY with a recursive relationship around it as well as a type entity to classify it as a “country,” “state,” or “city.”

- Organizational structures where a PARENT COMPANY is broken down into its SUBSIDIARY(s) and then into DIVISION(s) and then DEPARTMENT(s) may be generalized into a common entity called ORGANIZATION (or ORGANIZATION UNIT) with a recursive relationship around it as well as a type entity to classify it as a “parent organization,” “subsidiary,” “division,” or “department.”

The preceding data modeling structures could all be modeled using the pattern in Figure 4.6 by substituting ENTITY for the applicable generalized entity. For example, you could have entities for PART, GEOGRAPHIC BOUNDARY, and ORGANIZATION with a recursion and a PART TYPE, GEOGRAPHIC BOUNDARY TYPE, and ORGANIZATION TYPE in order to maintain the various levels of the hierarchy or aggregation.

When Should This Pattern Be Used?

We use this data model pattern when:

- **There is a need for a flexible model that supports as many levels as needed without having to change the underlying entities** In the previous pattern, the addition of another level to the hierarchy required an addition of a new entity and relationships. If a new level needed to be inserted (such as an ACTIVITY that is needed between a PHASE and a TASK, or a SUBTASK that is a further breakdown of TASK), this required even more work. In this current pattern, changes to the number or type of levels are much easier. For example, if programs need to be added to the WORK EFFORT hierarchy, you add a new WORK

EFFORT TYPE of “Program” and add the specific instances about programs into WORK EFFORT. The relationships between programs and all of the other types of work efforts are supported by the recursive relationship.

- **There are strictly one-to-many relationships needed between the recursive entity, and there are no many-to-many relationships** For example, where there is a hierarchy in which a child might have more than one parent relationship, this pattern would not be suitable (however, the next pattern in this chapter, the Level 2 Expanded Recursive Pattern, would work for this).
- **The different levels of a hierarchy all have the same or similar attributes and relationships (although this is not required in order to use this pattern)** In the example, XYZ Corporation stated that PROJECT, PHASE, and TASK contained the same attributes and relationships and therefore could be characterized as different types of WORK EFFORT(s). If they did not contain the same attributes and relationships you could add subtypes of PROJECT, PHASE, and TASK to WORK EFFORT and maintain the individual attributes and relationships for each subtype.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern is more generalized and abstract than the level 1 pattern** Therefore it is harder to communicate the meaning of the model to stake holders. For instance, the level 1 pattern showed very clearly each of the different levels in the hierarchy and this pattern obscures this information. Of course, a little

education goes a long way because the pattern can be explained, especially with the use of instance diagrams (diagrams that show the specific instances of each entity and attribute) and/or worked examples.

- **This pattern can still lead to models that repeat the same type of data at different levels of a hierarchy, thus causing data inconsistencies** For example, you can still end up recording **estimated hours** at different levels. As we discussed, this creates the problem that the estimated hours may be inconsistent between a parent instance (e.g., for a PHASE) and the sum of the children (e.g., for the sum of the hours for that phase's TASK(s)). One solution would be to have subtypes of WORK EFFORT, such as PROJECT, PHASE, and TASK, and then have the TASK subtype be the only subtype containing an attribute for **estimated hours**.
- **One-to-many recursive relationships may have limitations** In a hierarchy when there is a need for the child entity to be related to more than one parent, this pattern cannot support this need. For example, if a part were included in more than one parent assembly, this data model would only allow the part to be shown in a single parent assembly. The same applies to tasks, and this would not allow a task to be part of two different projects.
- **This pattern does not enforce the business rules regarding how specific types of work effort may relate to each other** For example, there may only be projects that are made up of phases that are made up of tasks. This pattern allows any number of work effort types, whether they are valid or not, to be related to any other work effort type. When using this pattern, the data professional should not be “lazy” about

documenting rules that may not be apparent from the model.

- **This model cannot maintain that different levels within the structure may have different rules regarding optionality** For example, PROJECT(s) must have a **name**; TASK(s) may have a **name**. What it does show is that WORK EFFORT(s) of any WORK EFFORT TYPE must have a **name**. Also, this pattern cannot enforce the rule that all children except the top must have a parent.
- **This model cannot maintain specific relationships that each level of entity within the structure may have** For example, PROJECT(s) have a SPONSOR; TASK(s) and PHASE(s) do not. Or only TASK(s) are "implemented by" a particular PERSON.
- **This pattern makes it more difficult to handle different types of entities having different attributes** For example, perhaps TASK(s) have an attribute of **actual hours worked** while PHASE(s) and PROJECT(s) do not. However, subtypes could be added to the ENTITY to accommodate this, for example, there could be WORK EFFORT subtypes of PROJECT, PHASE, and TASK that can each maintain specific attributes or relationships to the subtypes.

Synopsis

In this section you saw that the Level 2 Recursive Pattern is a very useful pattern that supports a flexible solution for the creation of hierarchies, peer-to-peer relationships, or aggregations. There are some significant features of this pattern. First, we generalized PROJECT, PHASE, and TASK into WORK EFFORT(s) of different types, as seen in Figure 4.7. We were able to do this because PROJECT, PHASE, and TASK represented the same basic type of data and all had the same attributes and relationships. Because we could

generalize PROJECT, PHASE, and TASK into WORK EFFORT, we were able to generalize the hierarchy relationships seen in Figure 4.5 in the previous section into a recursive relationship. This generalization into a recursive relationship avoids having to change the data model when new levels in a hierarchy are discovered. This can really help enterprises that need change to the structure of their hierarchies or aggregations, such as their reporting structures or “bills of materials.”

The issues with generalization are also illustrated by this pattern. For example, if PROJECT, PHASE, and TASK had specific attributes or relationships, this pattern would have lost these facts. To address this, we could have added subtypes to WORK EFFORT with the specific relationships and attributes. However, the generalized recursive relationship will not support specific optionality between different levels of a hierarchy (or aggregation): for example, if a PROJECT *must* have one or more PHASE(s), but a PHASE *may* have one or more TASK(s).

The pattern is more abstract than the level 1 pattern, but it should be fairly easy to explain to technical audiences. In general, more documentation is needed in the data model, such as showing instance diagrams, worked examples, and/or spreadsheets and databases with example data, because the model is not as self-explanatory as the level 1 pattern.

Another key limitation of this pattern is that it provides only one-to-many relationships. Therefore, if there is a hierarchy where there is need for a child to be related to more than one parent, this pattern does not support this need. This may be referred to as either a “many-to-many” or a “matrix” relationship structure and is covered in the next section.

Level 2 Expanded Recursive Pattern

Most enterprises need flexibility when dealing with hierarchies, aggregations, or peer-to-peer relationships. The Level 2 Recursive Pattern supports this flexibility, but with some loss of specific understanding. This is the classic balance of flexibility versus understanding. An enterprise sometimes requires that the data model retain the specific business knowledge, such as specific one-to-many relationships, and still be able to maintain a level of flexibility that allows different ways of creating hierarchies, aggregations, or peer-to-peer relationships.

Why Do We Need This Pattern?

An enterprise should consider using the Level 2 Expanded Recursive Pattern when there is a need for many-to-many recursive relationships.(9) For example, a manufacturer of electronics may need to capture each of the different ways that it can create a finished good and the parts that may be included, knowing that the same part may be included and related to many parent subassembly parts in different circumstances. Additionally, there may be many other types of many-to-many recursive relationships, such as what parts can the manufacturer substitute when a stock of parts is low, which parts are incompatible with each other, or what parts complement each other.

Another example we will use for this section is that of an IT services firm wishing to capture the structures in a project (work effort), such as which tasks (work efforts) are earlier versions of other tasks (work efforts), which tasks have precedence (need to be done before another task), and what is the work breakdown structure, knowing that a work effort may be part of more than one parent work effort.

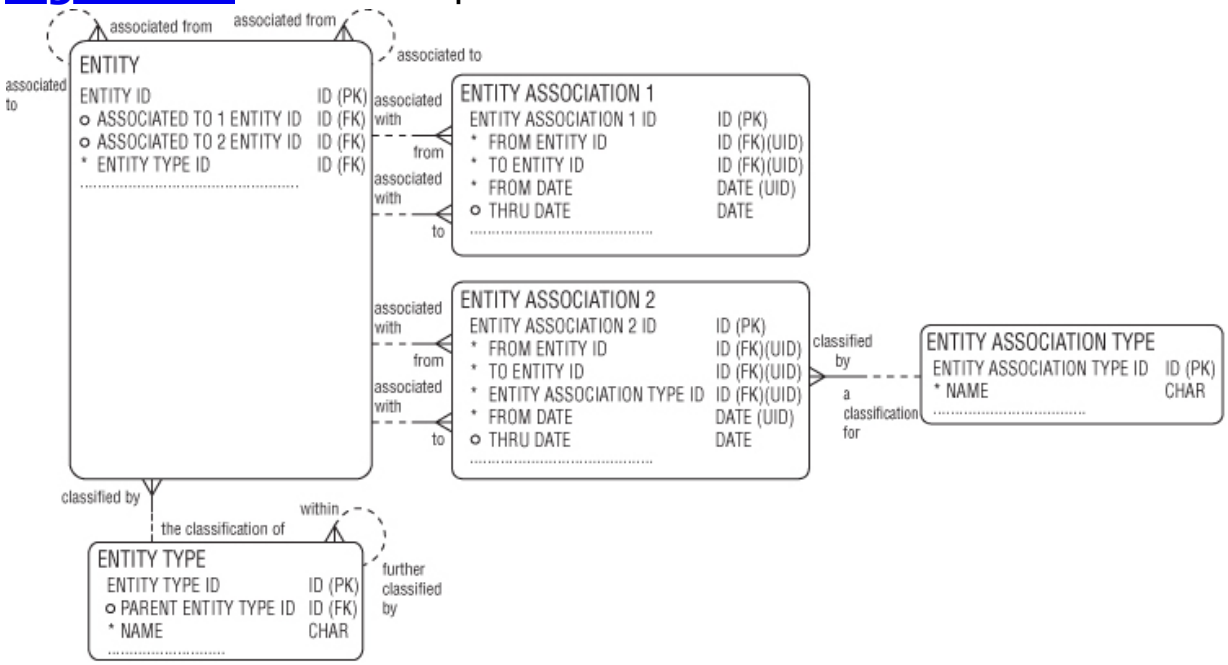
Note

Precedence can be defined as “the fact, state, or right of preceding; priority” and *precede* is defined as “to go before, as in place, order, rank, importance, or time.” Dependency can be defined as “conditioned or determined by something else; contingent.”(5)

How Does This Pattern Work?

Figure 4.8 illustrates a pattern that maintains either one-to-many and/or many-to-many recursive relationships. This pattern will also support any one-to-many recursive relationship from the last pattern; however, it will go further to support situations in which there are more complex many-to-many relationships between instances.

Figure 4.8 Level 2 Expanded Recursive Pattern

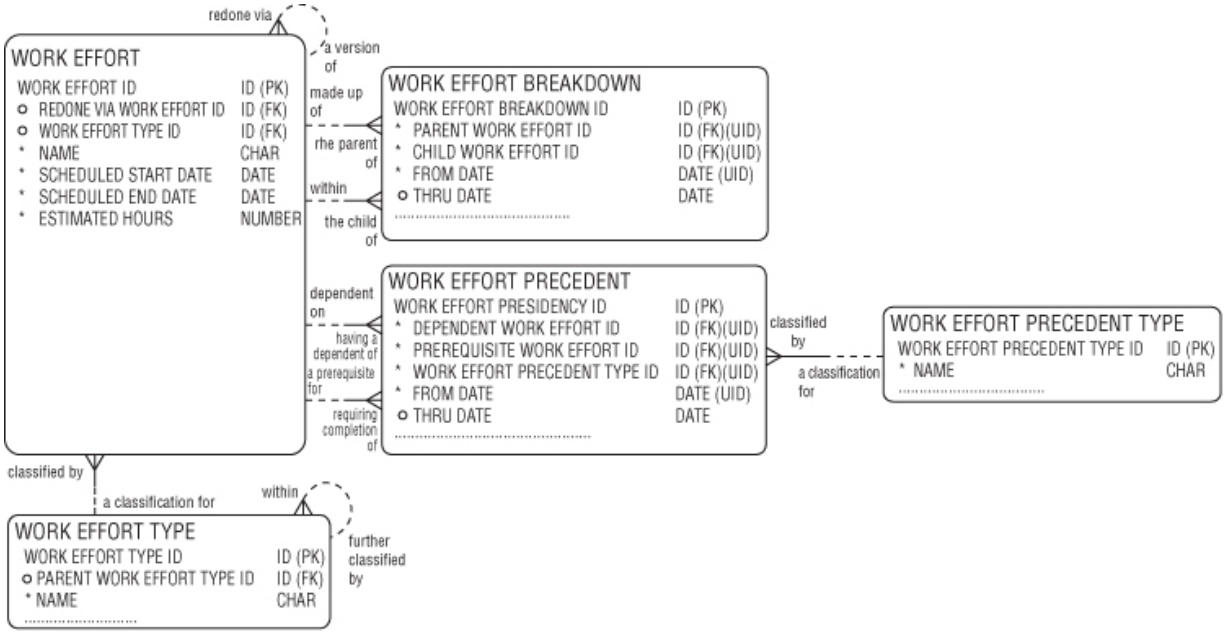


In Figure 4.6 you saw a one-to-many recursive relationship. If you need a many-to-many relationship, you need an associative or intersection entity to allow instances of the entity to be related to other instances of the entity in

a many-to-many fashion. In Figure 4.8, you see these associative entities, ENTITY ASSOCIATION 1 and 2.(9) These associative entities specifically resolve the different ways that the ENTITY may be related to itself. The ENTITY ASSOCIATION entities may or may not have any data attributes of their own. They may exist purely to capture the many-to-many relationship that exists in the recursive entity.

Another interesting addition to this pattern is that ENTITY ASSOCIATION 2 has an ENTITY ASSOCIATION TYPE entity. Some recursive associations may be classified, for example, a dependency association between work efforts may be either a precedence type of dependency (one effort needs to be done before another) or a concurrency type of dependency (two work efforts need to be done together). We further illustrate this concept, with a different example of this, in Figure 4.9.

Figure 4.9 Example of using a Level 2 Expanded Recursive Pattern



To illustrate this pattern we continue with the scenario described in the previous sections. XYZ Corporation was

concerned that some tasks that needed to be accomplished were actually part of two different projects. For example, “Create source system inventory” may be a work effort that is part of the systems analysis for the data warehousing project, but the same work effort may also be part of a companywide configuration management project. Hence, the nature of some of the work efforts are that they are recursive in a many-to-many fashion and thus a “parent” has many “child” work efforts and the “child” may have many “parents.” XYZ Corporation also wanted to retain the one-to-many recursive relationship that described how it manages versioning of work efforts. Finally, XYZ project managers stated that there were different ways to interpret the precedence of tasks in projects—for example, if a task can be started only if the preceding task has been completed fully, or if it can it be started once the preceding task has been started.

Based on the requirements stated by the project managers in XYZ Corporation and by using the Level 2 Expanded Recursive Pattern, the data professional produced Figure 4.9. In this figure the data professional has maintained a similar structure seen in Figure 4.6, but has allowed for a many-to-many relationship that supports the additional requirements of XYZ Corporation.

A specific relationship between WORK EFFORT(s) is captured by the one-to-many recursive relationship on the top right-hand of the WORK EFFORT entity. This relationship states that “each WORK EFFORT *may be* **redone via** one or more WORK EFFORT(s) and each WORK EFFORT *may be* **a version of** one and only one WORK EFFORT.” This captures the relationships between WORK EFFORT(s) where a new iteration or version of a WORK EFFORT has replaced another WORK EFFORT. For example, imagine XYZ Corporation started the task of “Create Initial Scope Statement” and then this task was redefined and replaced with a newer

version of this type of work (with different scheduled dates, hours, and so on) and called “Develop Statement of Work and Scope.”

In Table 4.4 you can see that “Create Initial Scope Statement” was redone via “Develop Statement of Work and Scope”, which is the newer version of the work effort. Additionally, “Discover Target Requirements” was redone via “Develop Requirements Specifications,” which was the new version of that work effort.

Table 4.4 Example of Level 2 Expanded Recursive Pattern, Versions of WORK EFFORT(s)

WORK EFFORT. WORK EFFORT ID (REDONE VIA)	WORK EFFORT.NAME (REDONE VIA)	WORK EFFORT. WORK EFFORT ID (A VERSION OF)	WORK EFFORT NAME (A VERSION OF)
9007	Create Initial Scope Statement	9015	Develop Statement of Work and Scope
9009	Discover Target Requirements	9011	Develop Requirements Specifications

WORK EFFORT BREAKDOWN describes how the various WORK EFFORT(s) are composed of each other within a hierarchy. This often manifests itself as a project plan. XYZ Corporation was very clear in its desire that there be only one type of WORK EFFORT BREAKDOWN. In other words, it captures the way that a project breaks down into a master project plan, and there aren't different ways to break down work efforts. This is the reason there is no WORK EFFORT BREAKDOWN TYPE. XYZ Corporation also stated that some tasks are part of more than one phase and some phases may be part of more than one project.

Note

Work breakdowns may actually have specific types of breakdown structures—work breakdown according to project

management or according to the vendor and so on. It happens also for PARTS, such as a BOM for engineering, marketing, and manufacturing. If this is the case, it is easy just to add a WORK EFFORT BREAKDOWN TYPE in much the same way as we show a WORK EFFORT PRECEDENT TYPE.

In Table 4.5 you see that the “Create Source System Inventory”, with **work effort id** 9008, task is part of both the “System Analysis” phase of the “Enterprise Data Warehouse” project and the “Systems Inventory” phase of the “Configuration Management” project. This is a common occurrence in that the same task may be performed for more than one phase within different projects. It is situations like this that require the pattern to support many-to-many recursive relationships.

Table 4.5 Example of Level 2 Expanded Recursive Pattern, WORK EFFORT BREAKDOWN

WORK EFFORT.WORK EFFORT ID	WORK EFFORT. NAME	WORK EFFORT TYPE.NAME	WORK EFFORT. WORK EFFORT ID (PARENT)	WORK EFFORT.NAME (PARENT)	WORK EFFORT TYPE.NAME (PARENT)	WORK EFFORT. SCHEDULED START DATE
9002	Enterprise Data Warehouse	Project				Jan. 1, 2009
9006	Systems Analysis	Phase	9002	Enterprise Data Warehouse	Project	Feb. 4, 2009
9008	Create Source System Inventory	Task	9006	Systems Analysis	Phase	Mar. 13, 2009
9022	Configuration Management	Project				Mar. 3, 2009
9023	Systems Inventory	Phase	9022	Configuration Management	Project	Mar. 3, 2009
9008	Create Source System Inventory	Task	9023	Systems Inventory	Phase	Mar. 13, 2009

WORK EFFORT PRECEDENT maintains that one work effort must be started before another work effort. One rule that project managers in XYZ Corporation stated was that the prerequisite WORK EFFORT must be 100 percent complete before the dependent WORK EFFORT could start. This was the most common situation. After some investigation, the data professional discovered another type of WORK EFFORT PRECEDENT where a dependent WORK EFFORT may be started before the prerequisite WORK EFFORT is fully complete. XYZ Corporation's project managers refer to the different types as “Total” and “Partial” precedent tasks. In Table 4.6 you see that the “Create Mapping Template” must be complete before the “Create Mapping.” The systems analysts need to finalize 100 percent of the template before any mapping takes place. In the case of the “Create Initial Scope Statement” and “Create Source System Inventory,” the system analysts are allowed to create the scope statement outline and introduction, and then begin gathering all the source system documentation. Finally, you see that the “Create Mapping” also needs to wait for the “Create Source System Inventory” task to start, because the company needs to know what source systems should be mapped. In other words, “Create Mapping” has two different precedent tasks—one, “Create Mapping Template,” must be completed totally, and one, “Create Source System Inventory,” need only be completed partially.”

Table 4.6 Example of Level 2 Expanded Recursive Pattern, WORK EFFORT PRECEDENT

WORK EFFORT. WORK EFFORT ID (DEPENDENT ON)	WORK EFFORT.NAME (DEPENDENT ON)	WORK EFFORT ID (THE PREREQUISITE)	WORK EFFORT. NAME (THE PREREQUISITE)	WORK EFFORT PRECEDENT. FROM DATE	WORK EFFORT PRECEDENT TYPE.NAME
9005	Create Mapping Template	9010	Create Mapping	July 24, 2009	Total
9007	Create Initial Scope Statement	9008	Create Source System Inventory	Mar. 7, 2009	Partial
9008	Create Source System Inventory	9010	Create Mapping	July 24, 2009	Partial

The WORK EFFORT PRECEDENT **from date** is defined as the date when this association between work efforts went into effect. Some modelers may think we need an additional attribute to show the end date at which the "parent" WORK EFFORT was totally or partially complete in order to see when the other work effort can begin. This is information that should be maintained in the WORK EFFORT entity with either an additional attribute of **percentage complete** or better yet by using the status pattern showing a WORK EFFORT STATUS of "completion status" that maintains the **percentage complete** and the **status datetime** (see Chapter 6 in this book on the status pattern).

WORK EFFORT PRECEDENT is a type of peer-to-peer relationship. What we mean by this is that all of the instances of WORK EFFORT PRECEDENT are peers of each other within the context of WORK EFFORT PRECEDENT because there is not a parent-child relationship and one work effort needs the other work effort to be partially or fully completed before it is started. So in this example "Create Mapping Template" must be totally complete before XYZ Corporation can begin "Create Mapping." Notice that both of these WORK EFFORTS are tasks. Would it be possible, for example, that "System Analysis" (a phase) would have to be completed before "Create Mapping" (a task) was started? In other words can a Project, a Phase, and a Task be peers of each other? The answer is yes because they are all considered work efforts. In this

example this makes perfect sense. It should be possible to show that a project must be completed before you can start a task, phase, or another project and so on.

Because WORK EFFORT PRECEDENT is a peer-to-peer relationship, you know that if you delete a member of a peer-to-peer relationship, you don't have to cascade that delete so that all members in that chain are also deleted or marked as invalid. But this raises a problem. Is the peer-to-peer association of precedent transitive? In other words, by saying “Create Initial Scope Statement” must precede “Create Source System Inventory” and “Create Source System Inventory” must precede “Create Mapping” (as seen in Table 4.6), does it mean that “Create Initial Scope Statement” must precede “Create Mapping?” The answer is yes, due to transitive logic which is very similar to the concept of transitive dependency in relational theory.(10) This means that if you delete “Create Source System Inventory,” you should be careful to maintain that “Create Initial Scope Statement” precedes “Create Mapping.” Otherwise, you could lose important data about which work efforts preceded other work efforts. This points out that when you have peer-to-peer relationships (or a hierarchy or an aggregation), you need to define how you manage the data structure as a whole. For example, when you delete one instance or relationship, you need to consider the effect this has on other instances or relationships.

When Should This Pattern Be Used?

We use this data model pattern when:

- **There is a need for many-to-many recursive relationships** You need to evaluate if a many-to-many recursion is needed for each type of WORK EFFORT and if it is, then the Level 2 Recursive Pattern will not work and this pattern is needed.

- **When there are variations of the same type of many-to-many recursive relationship** The ENTITY ASSOCIATION 2 and ENTITY ASSOCIATION TYPE allow for these variations. For example, variations of WORK EFFORT PRECEDENT may be “partial” or “complete” which may be maintained in a WORK EFFORT PRECEDENT TYPE entity.
- **There is a need to flexibly support peer-to-peer relationships** This pattern can support one-to-one, one-to-many or many-to-many peer-to-peer relationships.
- **The different levels of a hierarchy all have the same or very similar attributes and relationships** In the example, XYZ Corporation stated that PROJECT, PHASE, and TASK contained the same attributes and relationships; therefore, they could be characterized as different types of WORK EFFORT(s).

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern explicitly declares each of the different recursive associations that exist for an ENTITY, but what happens if you discover another association that you have not captured** This means that you need to add a new entity or new recursive relationship for every recursive association that you discover. This might not be the most efficient way to model if there is a dynamic environment where new types of recursive relationships may be discovered. For example, if a need emerged to also record which work efforts needed to be done at exactly the same times, then another type of recursive relationship, namely a

"concurrency" relationship, would need to be added to the model.

- **This pattern does not capture certain types of business rules** For example, what if the same two tasks in the same project are maintained as both a concurrent relationship and a precedent relationship? This would cause confusion because these relationships do not logically make sense. This would need to be resolved by recording and implementing a business rule outside the data model, by using the Level 3 Recursive Pattern with Business Rules, or by using our Business Rule Pattern described in chapter 8.
- **This pattern cannot maintain that different levels in the hierarchy may have different rules regarding optionality** For example, there may be a rule that TASK(s) must have **estimated hours**, but PROJECT(s) and PHASE(s) *may* have **estimated hours** because they may be able to be derived from the associated TASK(s). This pattern shows that WORK EFFORT(s) of any WORK EFFORT TYPE must have **estimated hours**. Also, this pattern cannot enforce the rule that all children except the top must have a parent.
- **This pattern cannot maintain specific relationships that each level in the hierarchy may have** For example, PROJECT(s) have a SPONSOR; TASK(s) and PHASE(s) do not; or only TASK(s) are "implemented by a particular person".
- **This pattern makes it more difficult to handle different types of entities having different attributes** For example, perhaps TASK(s) have an attribute of **actual hours worked** while PHASE(s) and PROJECT(s) do not. However, subtypes could be added to the ENTITY to accommodate this, for example, there could be WORK EFFORT subtypes of PROJECT, PHASE,

and TASK that can each maintain specific attributes (or relationships) to the subtypes.

Synopsis

In this section we discussed a pattern that provides even more flexibility than the last pattern by allowing numerous types of either one-to-many or many-to-many recursive relationships. This pattern retains some business knowledge (such as specifying one-to-many recursive relationships); however, the real strength of this pattern is that it maintains a level of flexibility that allows multiple different ways of creating hierarchies, aggregations or peer-to-peer relationships. The significance of this pattern is that it explicitly addresses resolving many-to-many recursive relationships. The pattern supports hierarchies, aggregates, and peer-to-peer relationships. In this section we described why it is important to understand the underlying rules that support peer-to-peer relationships, such as transitive dependency. Finally, this pattern is also significant because we show how you can mix both many-to-many and one-to-many relationships.

This pattern helps you when you need to support a complex environment where you need to capture many different ways that an entity is related to itself. This approach often provides a good basis for implementing databases in fairly dynamic environments with the exception of not easily accommodating new types of recursive relationships that may emerge (we will address this in the next pattern). If you need the flexibility of having multiple many-to-many recursive relationships and you know all of the different recursive relationships that you wish to capture, their nature (M:M or 1:M), and that they will not change, this pattern is an appropriate choice.

In most circumstances, this pattern would be a little too abstract to use as part of a scope statement or presentation

of data requirements, although it depends on the level of data modeling experience of the audience. In this pattern, each of the different specific recursive relationships was explicitly described. This pattern can help jump-start your modeling effort and help you begin to define the common terms used for the recursive relationships in your enterprise.

Unless you create specific subtypes in WORK EFFORT(s) (for example, PROJECT, PHASE, and TASK), this pattern will not support specific relationships or attributes for different types of WORK EFFORT(s). For example, what if PROJECT(s) was the only work effort type that captured **booked versus burned ratio** (for professional services firms, “booked” is the amount of revenue that was estimated and “burned” is the amount of revenue that was earned) or what if TASK(s) was different from the other types of work effort in that it had a specific relationships to PERSON?

Level 3 Recursive Pattern

Certain situations require a lot of flexibility when modeling recursive associations. In a dynamic environment, all the different types of recursive relationship that exist or may emerge for an entity may not be apparent. Sometimes, the data professionals may have identified only core recursive relationship types but know that the enterprise is expanding, and there is a need to add other recursive relationship types and rules to govern those associations.

For example, in the case of recursive relationships for a PRODUCT, the data professionals may identify the need for a PRODUCT BREAKDOWN (products made up of other products) and PRODUCT SUBSTITUTION (which product may be substituted for another product). They may use the previous Level 2 Expanded Recursive Pattern to accommodate these needs. Then additional data requirements may arise that also involve other types of

recursive relationships such as PRODUCT OBSOLESCENCE (which product supersedes another product), PRODUCT INCOMPATIBILITY (which product should not be sold with another product), PRODUCT COMPLEMENT (which product should be recommended as a complementary product to the one they ordered), or any number of additional recursive relationships. The previous patterns may not be the best choice when there are additional recursive relationships that may emerge over time.

Why Do We Need This Pattern?

This pattern supports the need for flexibility when adding or updating the recursive associations that an entity may have. This pattern is ideal for situations where a data professional is working in a dynamic environment or in a new business area where the number of different types of recursive associations that are needed is not clear. How do data professionals declare these associations and thus have the flexibility to add new associations without changing the underlying model?

This pattern also provides a stable and common data model structure that can help facilitate standard services, functions, procedures, and rules. For example, this same pattern may be applied to PARTY, PRODUCT, WORK EFFORT, PART, ORDER ITEM, INVENTORY ITEM, or any other entity that has a self association, and thus the model can be consistent across different parts of the model. Thus, common services, functions, procedures and rules can all refer to the same type of data structure. This pattern is also flexible enough that the underlying data model does not get changed as you discover new types of associations.

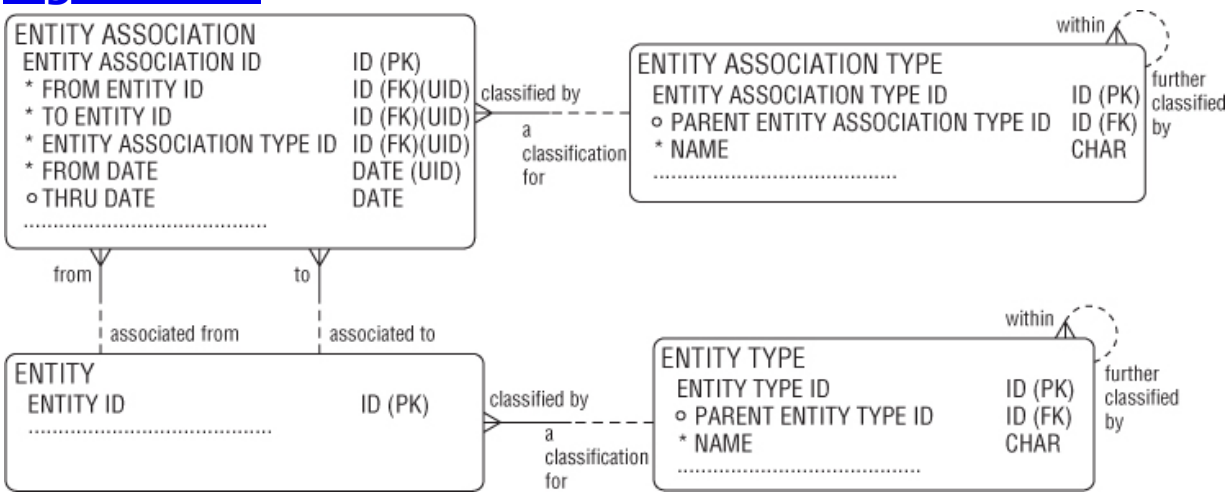
Finally, this pattern provides a robust implementation. We have implemented this pattern in many different environments with success.

How Does This Pattern Work?

This pattern takes the associations that were described in the Level 2 Expanded Recursive Pattern and generalizes them into a common ENTITY ASSOCIATION entity and adds ENTITY ASSOCIATION TYPE to define each of the different association's types that can be captured by this pattern. This allows additional recursive relationships that emerge to be easily added by adding an additional instance of an ENTITY ASSOCIATION TYPE. In other words, ENTITY ASSOCIATION allows any type of many-to-many relationships that an ENTITY may have currently or in the future and classifies them by ENTITY ASSOCIATION TYPE.

In Figure 4.10 “each ENTITY *may be* **associated from** one or more ENTITY ASSOCIATION(s)” and “each ENTITY *may be* **associated to** one or more ENTITY ASSOCIATION(s).” These generic relationship names allow for any type of recursive relationships to be maintained.

Figure 4.10 Level 3 Recursive Pattern



The ENTITY ASSOCIATION gets classified by ENTITY ASSOCIATION TYPE. You can create new association types by adding an instance to ENTITY ASSOCIATION TYPE. “Each ENTITY ASSOCIATION TYPE *may be* **a classification for** one or more ENTITY ASSOCIATION(s) and each ENTITY

ASSOCIATION *must be* **classified by** one and only one ENTITY ASSOCIATION TYPE."

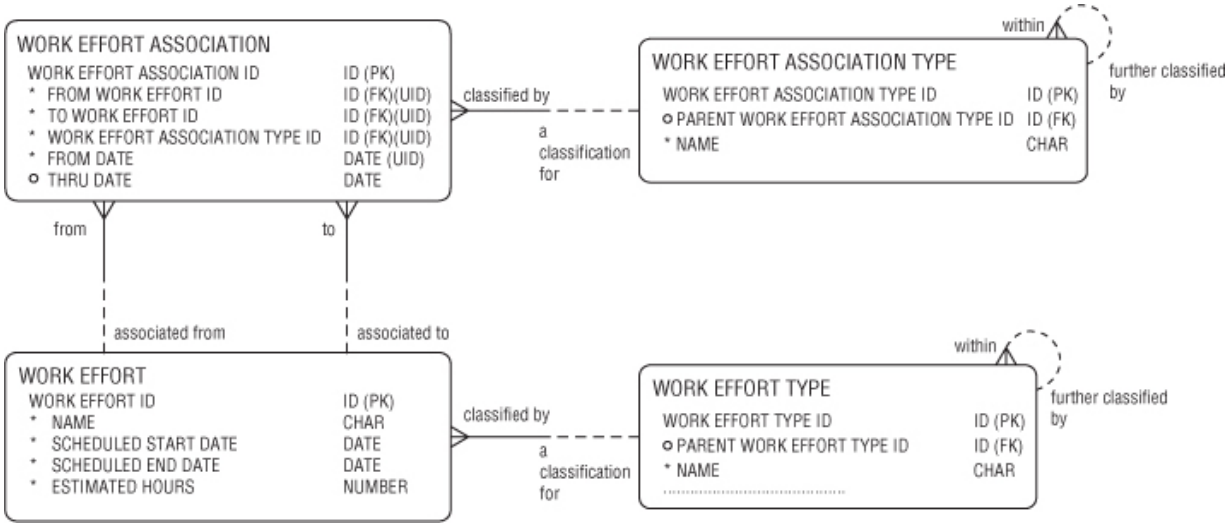
> **Note**
>
> The ENTITY ASSOCIATION TYPE entity here is a supertype of the ENTITY ASSOCIATION TYPE entity in the previous pattern. In the previous pattern it had values like "Partial" and "Total," which were only a subset of instances for a precedent recursion. Here, it could have those values and also include additional types of recursions such as "Dependent," "Concurrent," and "Breakdown." Therefore, in the previous pattern the ENTITY ASSOCIATION TYPE maintained only instances for a specific type of recursion whereas here it is much more broad.

Note that the ENTITY ASSOCIATION TYPE has a recursive relationship to itself saying that "Each ENTITY ASSOCIATION TYPE *may be* **within** one and only one ENTITY ASSOCIATION TYPE." This would support a hierarchy of types that may show, for example, that "Precedent" and "Concurrency" are two types of recursive relationships that are within the ENTITY ASSOCIATION TYPE of "Dependency."

An example of this pattern is shown in Figure 4.11. In Figure 4.9 you saw WORK EFFORT BREAKDOWN and WORK EFFORT PRECEDENT as well as a one-to-many recursive relationship supporting versions of different WORK EFFORT(s). In this pattern, you can maintain this type of data by adding instances of ENTITY ASSOCIATION TYPE ("Breakdown," "Precedent," and "Version"), and the relationships that were captured in WORK EFFORT BREAKDOWN, WORK EFFORT PRECEDENT, and the one-to-many recursive relationship for different versions of a work effort, would all be captured in WORK EFFORT ASSOCIATION. Any new association types could be captured in the same way—for example, if XYZ Corporation wanted to capture "Incompatibility," where the two work efforts cannot both exist together in the same project, or "Superseding," where one work effort replaced the other work effort. Finally, you

can use the recursion around the WORK EFFORT ASSOCIATION TYPE entity in order to capture a hierarchy of classifications such as “Total” and “Partial” being association types with the association type “Precedent.”

Figure 4.11 Example of using a Level 3 Recursive Pattern



In Table 4.7 you see examples of how you can capture different types of recursive relationships; all are instances of WORK EFFORT ASSOCIATION. In other words this pattern supports a generalized way to capture all recursive relationships.

Table 4.7 Example of WORK EFFORT ASSOCIATION and WORK EFFORT ASSOCIATION TYPE

WORK EFFORT. WORK EFFORT ID (FROM)	WORK EFFORT. NAME (FROM)	WORK EFFORT. WORK EFFORT ID (TO)	WORK EFFORT. NAME (TO)	WORK EFFORT ASSOCIATION. FROM DATE	WORK EFFORT ASSOCIATION TYPE.NAME
9007	Create Initial Scope Statement	9015	Develop Statement of Work and Scope	Feb. 28, 2009	Version
9009	Discover Target Requirements	9011	Develop Requirements Specifications	Feb. 28, 2009	Version
9005	Create Mapping Template	9010	Create Mapping	July 24, 2009	Precedent
9002	Enterprise Data Warehouse	9004	Mapping	June 3, 2009	Breakdown
9004	Mapping	9005	Create Mapping Template	July 20, 2009	Breakdown
9004	Mapping	9010	Create Mapping	July 30, 2009	Breakdown

In the first two rows of Table 4.7 you can see two “version” relationships of WORK EFFORTS, where “Create Initial Scope Statement” has now become a version of the work effort called “Develop Statement of Work and Scope” and “Discover Target Requirements” was changed to a version of this work effort called of “Develop Requirements Specifications.” You also see in Table 4.7 that the WORK EFFORT ASSOCIATION can have a “Precedent” association type also, with “Create Mapping Template” needing to happen before “Create Mapping.”

In Table 4.5 you saw the work breakdown structure for the “Enterprise Data Warehouse” project. This work effort structure can also easily be accommodated by the WORK EFFORT ASSOCIATION entity. You see in Table 4.7 that “Mapping” is a lower level work effort of “Enterprise Data Warehouse,” and you also see that “Mapping” has two work efforts within it, namely, “Create Mapping Template” and “Create Mapping.”

Some issues do arise because of the “generalized” nature of this pattern. In Figure 4.9 in the previous section, you saw that the relationship names to the WORK EFFORT BREAKDOWN and WORK EFFORT PRECEDENT were very descriptive. For example, the relationships between WORK EFFORT and WORK EFFORT PRECEDENT stated that “each WORK EFFORT *may be* **dependent on** one or more WORK EFFORT PRECEDENT(s) and each WORK EFFORT PRECEDENT *may be* **having a dependent of** one and only one WORK EFFORT; each WORK EFFORT *may be* a **prerequisite for** one or more WORK EFFORT PRECEDENT(s) and each WORK EFFORT PRECEDENT *may be* **requiring completion of** one and only one WORK EFFORT.” In this generic pattern these rich relationship names need to be more abstract to include many different types of recursive relationships. For example, “each WORK EFFORT *may be* **associated from** one or more WORK EFFORT ASSOCIATION(s).” Because of this, and as is

the case with most generalized patterns in this book, it is a little more difficult to understand. this pattern

Aside from using this pattern for WORK EFFORT, there are many other data modeling scenarios where the Level 3 Recursion Pattern works well, such as for:

- Bill of materials structures where you could substitute PART, PART ASSOCIATION, and PART ASSOCIATION TYPE for ENTITY, ENTITY ASSOCIATION, and ENTITY ASSOCIATION TYPE. There may be PART subtypes of FINISHED GOOD, SUBASSEMBLY, and RAW MATERIAL, and there may be PART ASSOCIATION TYPE instances of "part breakdown," "part substitute," "part obsolescence," "part complement," and "part incompatibility," as well as other possible types of part associations.
- Geographic boundary structures where you could substitute GEOGRAPHIC BOUNDARY, GEOGRAPHIC BOUNDARY ASSOCIATION, and GEOGRAPHIC BOUNDARY ASSOCIATION TYPE for ENTITY, ENTITY ASSOCIATION, and ENTITY ASSOCIATION TYPE. There may be GEOGRAPHIC BOUNDARY subtypes of COUNTRY, STATE, TERRITORY, PROVINCE, CITY, COUNTY, and many more subtypes for different parts of the world (see Chapter 7 on contact mechanisms for more discussion on this). There may be GEOGRAPHIC BOUNDARY ASSOCIATION TYPE instances of "within" (for example where a city is within a state), "overlapping," and "bordering," as well as other possible types of geographic boundary associations.
- Organizational structures where you could substitute ORGANIZATION, ORGANIZATION ASSOCIATION, and ORGANIZATION ASSOCIATION TYPE for ENTITY, ENTITY ASSOCIATION, and ENTITY ASSOCIATION TYPE. There may be ORGANIZATION ASSOCIATION TYPE instances of "parent relationship" (to capture an organizational

association that records which organization is subordinate to another organization), “merged” (for organizations that have merged), and “partner,” as well as many other possible types of organization associations. This could also be expanded to PARTY(s) instead of ORGANIZATION(s) and include all the different types of associations of people and organizations.

> **Note**
>
> In the models used in Volume 1 and Volume 2 of *The Data Model Resource Book*, and in Chapter 9 within the enterprise data model section, we have used this pattern with the term *relationship* for PARTY RELATIONSHIP and PARTY RELATIONSHIP TYPE instead of using the term “*association*” because the former is a more familiar term in the industry and it seems to fit well semantically for parties; however, either term can work. If your enterprise has a different way to describe these type of associations we encourage you to use that term.

When Should This Pattern Be Used?

We use this data model pattern when:

- **There is a need to provide a very flexible solution for modeling hierarchies, peer-to-peer relationships, and aggregations** New types of associations can be easily added as ENTITY ASSOCIATION TYPE(s) without needing to change the model.
- **There is a need to capture all of the associations using a consistent data modeling approach** This pattern provides more consistent recording and management of the various types of associations. Also, a common set of business rules for all associations can be created around the generic ENTITY ASSOCIATION and ENTITY ASSOCIATION TYPE.
- **There is a need to accommodate any type of recursive relationship** The pattern will accommodate

hierarchies, aggregations and peer-to-peer relationships. Because all recursive relationships are defined as many-to-many, the pattern will accommodate one-to-one and one-to-many relationships as well and allow for change in case the relationship cardinality changes.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern may be characterized as follows:

- **It is harder to understand** Some people may find this more difficult to understand and manage.
- **All types of recursive relationships are handled in this pattern, so the specific data and relationships for each type of recursion is not maintained in the pattern** Perhaps different types of recursive relationships may have different properties. For example, some recursive relationships may need to be modeled only as one-to-many, and using this many-to-many pattern may be overkill as well as not enforcing the one-to-many rule that may apply (for example, a child in a hierarchy may be able to belong to only one parent).
- **Because you are generalizing more, it is less clear what type of recursion each association type represents and what should happen in different situations** In the Level 2 Pattern, you had a "breakdown" type structure that was a hierarchy. A hierarchy implies ownership, so if you deleted the "Enterprise Data Warehouse" owner from the hierarchy, the whole project should be deleted. Even if you update its name to "EDW XYZ Corp" you have affected the whole hierarchy. In this pattern, for the "Version" structure you are capturing a peer-to-peer type

structure. This does not imply ownership. By deleting one of the tasks you don't delete the other. In other words, you have to be careful about how you manage and interpret the different structures that you capture in the ENTITY ASSOCIATION. If you see a hierarchy, when you delete the top node, should you delete all the children? If you see a peer-to-peer relationship, you should not delete all of the related elements. If you see an aggregate, you can delete the top node, but you don't delete the children. Different rules apply to different structures. You may need either another “type” entity or an attribute to identify the type of structure you are supporting, such as hierarchy, peer-to-peer, or aggregation

- **ENTITY ASSOCIATION and ENTITY ASSOCIATION TYPE do not capture rules information about the recursive relationships** The examples in Table 4.7 showed that two instances of an entity are related and the relationship is of a certain type, that is, “Enterprise Data Warehouse” is the parent of “Mapping” in a “breakdown.” This may not be enough in some circumstances. For example, there may be a behavior that exists between the two instances in the relationship. In product breakdowns, you see that a set of components may make up a product hierarchy. But some members of the product hierarchy may be captured, not because they are used to make up the ultimate product but because they have to be “excluded” from being used to create the product. Or they are “complementary” or there is an “implication,” that is, if you include product A it implies you are including product B. In this pattern, you don't explicitly capture these rules governing the behavior of the associations. You only capture the type of association that exists.

- **This pattern cannot maintain the optionality rule within a hierarchy or aggregation.** For example, it cannot enforce the rule that all children except the top entity must have a parent.
- **This pattern cannot maintain that different levels of entities may have different rules regarding optionality.** For example, there may be a rule that TASK(s) *must* have **estimated hours**, but PROJECT(s) and PHASE(s) *may* have **estimated hours** because they may be able to be derived from the associated TASKS. This pattern shows that WORK EFFORT(s) of any WORK EFFORT TYPE must all have **estimated hours**.
- **This pattern cannot maintain specific relationships that each level of the hierarchy may have** For example, PROJECT(s) have a SPONSOR; TASK(s) and PHASE(s) do not; or only TASK(s) are "implemented by a particular person."
- **This pattern makes it more difficult to handle different types of entities having different specific attributes** For example, if TASK(s) have **actual hours worked** and PHASE(s) and PROJECT(s) do not.

Synopsis

In this section, we described a very flexible way to maintain and create hierarchies, aggregations, and peer-to-peer relationships. This pattern would be suitable for enterprises that have a dynamic business environment or do not have a good understanding of the different types of recursive associations that may be needed for the entity at hand.

The significance of this pattern lies in the fact that all recursive relationships are handled in exactly the same way. Because of this feature, it is easier to add additional data model structures, such as rules, or to build common services or routines. Also, it is easier to build competency around a single way to handle recursive relationships in

your data community. Those in your data community don't have to worry about dealing with one-to-many structures. Also they will go to only one place to access all the recursive structures they need for any specific entity.

This pattern is also significant because it is expressly built to be "change tolerant." New types of recursive relationships can easily be added.

The pattern helps you because once implemented, it rarely needs to be changed. It supports all different types of recursive structures—hierarchies, aggregations, peer-to-peer relationships, and any new types of associations you may need to use, such as compositions and collections-in exactly the same way.

The structure is more generalized, which may lead to confusion in interpretation even for data professionals. You would not generally use it as part of a scope statement. This pattern also allows the data modeler to "catch all" situations. However, this also means that a modeler could become lazy and not capture all of the business rules that should be recorded for the different recursive relationships.

By capturing all of the different types of recursive relationship structures in one place in the model, you may also need to "tag" each structure, showing what type of structure it is. After all, the rules for maintaining a hierarchy are different from those maintaining a peer-to-peer relationship.

Unless you create specific subtypes in WORK EFFORT(s) (for example, PROJECT, PHASE, and TASK), this pattern will not support specific relationships or attributes for different types of WORK EFFORT(s). For example, what if PROJECT(s) was the only work effort type that captured **booked versus burned ratio** (described in the previous section) or what if TASK(s) had a specific relationship to PERSON?

Level 3 Recursive Pattern with Rules

An alternative way to support a flexible solution for recursive associations is described here in the Level 3 Recursive Pattern with Rules. In this solution we bring in a new way to view the relationships that exist between the different associations in a recursive relationship. In the previous section we classified the relationship using ENTITY ASSOCIATION TYPE. For example, in Table 4.7 we captured association types “Precedent,” “Version,” and “Breakdown.” For most enterprises it may be sufficient to do this, but for other enterprises this “classification” structure may not be enough.

Why Do We Need This Pattern?

The Level 3 Recursive Pattern with Rules supports the need for a flexible solution that allows for the classification of recursive relationships as well as the creation of rules that govern the behavior of those recursive relationships. In Table 4.7 we created the WORK EFFORT ASSOCIATION TYPE(s), “Precedent,” “Version,” and “Breakdown.” Are they really categorizing the association or are they describing a behavior or are they doing both?

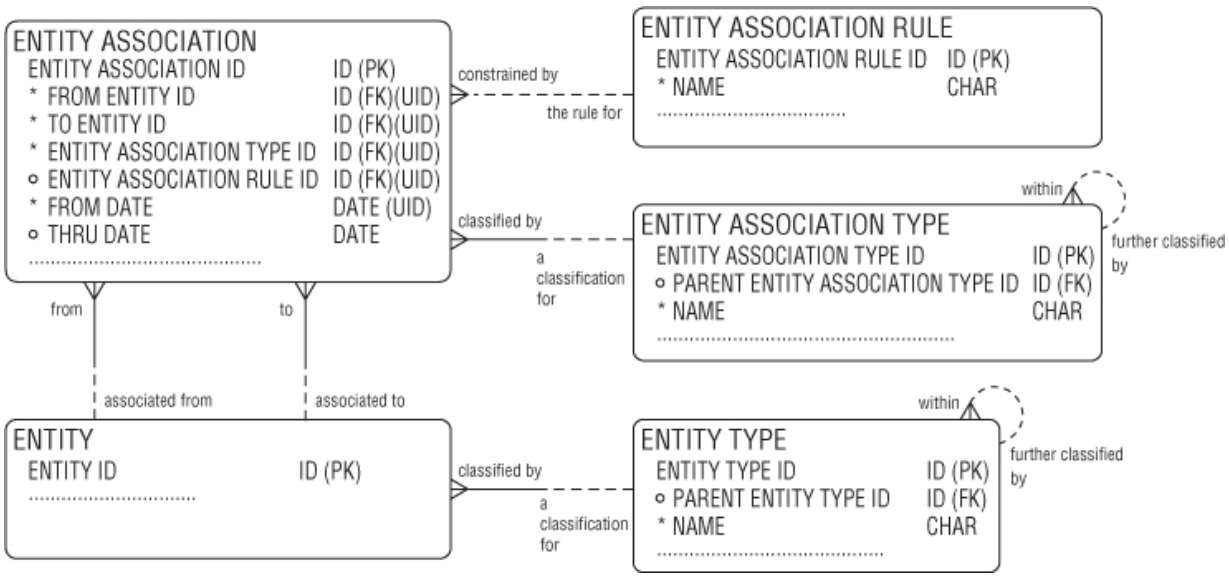
The “Breakdown” type describes a behavior in one way, that is, “Enterprise Data Warehouse” decomposes into “Mapping,” but another way to interpret it would be that “Breakdown” is saying this association can be categorized as a member of a work effort breakdown structure. Both ways of interpreting the WORK EFFORT ASSOCIATION TYPE are plausible and have merit. But which is it, a classification or a rule, or is it both? This pattern addresses the need to distinguish a classification of a recursive relationship from a rule that governs the behavior of the recursive relationship.

For example, two parts may be alternate parts. There may be a rule that says they can be both be used as a part in manufacturing a particular mobile phone. The “Alternate parts” association type may be the classification of the recursive relationship. Even though these are alternate parts, we have not specified a rule to say that for this mobile phone, that one part may be substituted for the other. The fact that one part can be substituted for the other could be considered a rule that governs the behavior between these two parts, within the setting of manufacturing the phone. If the enterprise made a different choice and specified a different rule, these same two parts may be explicitly “excluded” from being substituted for each other. Again, the “Alternate parts” association type classifies the recursive relationships. An “Exclusion” or “Substitution” could be considered the rule about the behavior of the two parts within the setting of the enterprise's manufacturing process.

How Does This Pattern Work?

Figure 4.12 is basically the same pattern described in Figure 4.10 with the addition of ENTITY ASSOCIATION RULE. As was seen in the previous section, the ENTITY ASSOCIATION TYPE was defined as the classification of the ENTITY ASSOCIATION. The ENTITY ASSOCIATION RULE can be defined as the “principle or regulation governing the conduct, action, or procedure of an association.”

Figure 4.12 Level 3 Recursive Pattern with Rules



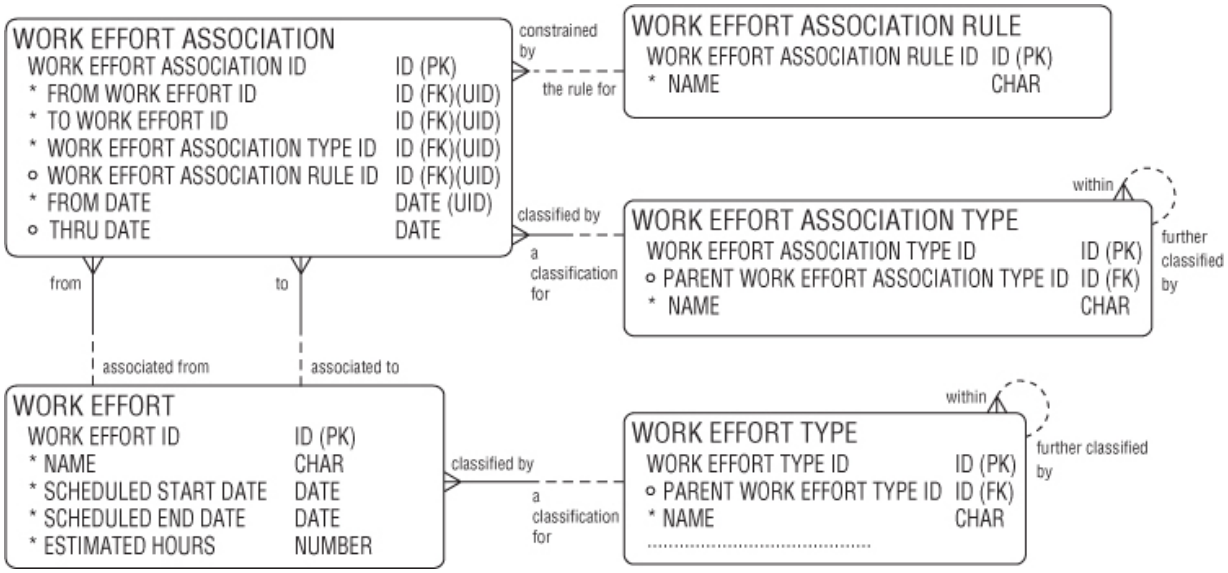
In the Level 3 Recursive Pattern you saw that “each ENTITY *may be* **associated from** or **associated to** one or more ENTITY ASSOCIATION(s), and each ENTITY ASSOCIATION *may be* **classified by** one and only one ENTITY ASSOCIATION TYPE.” In Figure 4.12 you now see that “each ENTITY ASSOCIATION *may be* **constrained by** one and only one ENTITY ASSOCIATION RULE.”

So, how is ENTITY ASSOCIATION TYPE different from ENTITY ASSOCIATION RULE? For many enterprises they don't have to be. In the previous section you saw how XYZ Corporation classified WORK EFFORT ASSOCIATIONS with “Breakdown,” “Precedent,” and “Version.” “Breakdown” classified the WORK EFFORT by a type of thing, that is, a breakdown structure. “Precedent” classified the WORK EFFORT by a behavior, that is, that one work effort must precede another. However, some enterprises may choose to model the classifications as something that is distinct from the rules. The benefit of this is that by doing this enterprises can be very precise about how the different instances of an association interact with each other and also can capture the classification of how those instances are related.

The data professional produced Figure 4.13 to show an alternative way for XYZ Corporation to manage its WORK

EFFORT(s). This is very similar to Figure 4.11, however, the WORK EFFORT ASSOCIATION RULE adds the capability to define how the work efforts interact with other and what types of rules exist that govern the behavior of these instances.

Figure 4.13 Example of using a Level 3 Recursive Pattern with Rules



You can capture each of the different types of associations with WORK EFFORT ASSOCIATION TYPE. This you have seen in the previous section. Examples of instances of WORK EFFORT ASSOCIATION TYPE could be:

- **Work Breakdown Structure:** When a work effort at a higher level is further broken down into its constituent work efforts. For example, the breakdown of projects into phases, phases into tasks, and so on
- **Program Aggregation:** The aggregation of a set of projects under one program.
- **Peer Work Efforts:** The peer-to-peer relationship of different work efforts, such as a set of data warehousing work efforts that are at the same level within the work breakdown structure. For example, this is useful when

peer work efforts are being benchmarked or compared with each other regarding budget and performance

ENTITY ASSOCIATION RULE defines how both sides of the association interact with the other via the association. So in Figure 4.13 you see a WORK EFFORT ASSOCIATION RULE containing different behaviors that an association between two WORK EFFORTS can have, such as:

- **Substitution:** For example, when a task can be substituted for another task
- **Exclusion:** For example, when a specific project is not allowed to be part of a specific program, or a specific task is "excluded" from being part of a specific phase or project
- **Concurrent:** Where two work efforts must happen at the same time
- **Precedent:** Where one work effort needs to be completed (partially or fully) before another work effort can start
- **Complementary:** When work efforts help each other, for example, when two projects may help each other, such as a metadata project helping a data governance project and vice versa. In this example, the more progress that is accomplished on the metadata project, the more progress for the data governance project, and vice versa.

Does this help XYZ Corporation manage its business better? It may help the business a great deal. If you examine Table 4.8 you see example instances of how Figure 4.13 could be populated. These instances illustrate a very comprehensive method of maintaining both classifications and rules for work effort associations, that XYZ Corporation has used to structure its WORK EFFORT ASSOCIATION(s).

Table 4.8 Example of WORK EFFORT ASSOCIATION TYPE and WORK EFFORT ASSOCIATION RULE

WORK EFFORT. WORK EFFORT ID (FROM)	WORK EFFORT. NAME (FROM)	WORK EFFORT. WORK EFFORT ID (TO)	WORK EFFORT. NAME (TO)	WORK EFFORT ASSOCIATION. FROM DATE	WORK EFFORT ASSOCIATION TYPE.NAME	WORK EFFORT ASSOCIATION RULE.NAME
9006	Systems Analysis	9007	Create Initial Scope Statement	Feb. 28, 2009	Work Breakdown Structure	
9006	Systems Analysis	9008	Create Source System Inventory	Feb. 28, 2009	Work Breakdown Structure	
9006	Systems Analysis	9009	Discover Target Requirements	Feb. 28, 2009	Work Breakdown Structure	
9008	Create Source System Inventory	9009	Discover Target Requirements	Feb. 28, 2009	Peer Work Efforts	Concurrent
9008	Create Source System Inventory	9009	Discover Target Requirements	Feb. 28, 2009	Peer Work Efforts	Complementary
9012	Discover Source Systems Requirements	9009	Discover Target Requirements	Jan. 1, 2009	Peer Work Efforts	Substitution
9150	Data Management Program	9002	Enterprise Data Warehouse	June 3, 2009	Program Aggregation	
9150	Data Management Program	9070	Customer Master Project	Feb. 28, 2009	Program Aggregation	
9002	Enterprise Data Warehouse	9100	Hardware Purchase	Jan. 1, 2009	Work Breakdown Structure	Exclusion

First, you see that “Create Initial Scope Statement,” “Create Source System Inventory,” and “Discover Target Requirements” are all part of the “Systems Analysis” work effort in the “Work Breakdown Structure” (a WORK EFFORT ASSOCIATION TYPE). There is no specific behavior that is captured with a WORK EFFORT ASSOCIATION RULE for these first three roles. In other words, “Systems Analysis” has three child work efforts, “Create Initial Scope Statement,” “Create Source System Inventory,” and “Discover Target Requirements.” This structure can be classified as a “Work Breakdown Structure.” We did not assign any rule to the relationships between them. WORK EFFORT ASSOCIATION RULE is not mandatory, and it is not necessary to have a rule for every association.

If you look at the fourth row of the table you see that “Create Source System Inventory” and “Discover Target Requirements” are of the association type of “Peer Work Efforts” and have a rule of “Concurrent,” meaning they need to be performed at the same time. Instead of just saying that “Create Source System Inventory” and “Discover Target Requirements” are related to each other as “Peer Work Efforts,” we are also storing the WORK EFFORT ASSOCIATION RULE of “concurrent” to show that they must be done together.

You might ask, “Couldn't you do this with the previous pattern, have two rows: one association type for “Peer Work Efforts” and one for “Concurrency”?” Yes, you could do this in the previous pattern. But are “Concurrency” and “Peer Work Efforts” the same thing? “Peer Work Efforts” could be considered a classification of the type of association. But is “Concurrency” a classification or does it more closely resemble a rule that affects the behavior between two tasks? Some enterprises feel that there is a significant difference between classifying a relationship and how the instances interact which affects behavior in some way.

The fifth row of the table shows that the same two work efforts are related to each other and have the WORK EFFORT ASSOCIATION TYPE of “Peer Work Efforts,” meaning they are work efforts that are at the same level within the work breakdown structure. You can also see from this row that they are “Complementary.” This means that the two work efforts may help each other in some way. This goes beyond saying that two work efforts are related to each other as peers. It also says that they complement each other as peers because we can learn something from each work effort that can help with the other work effort. Two work efforts could be peers of each other but don't complement each other in any way.

Note

An alternative to Figure 4.12 is to allow for many rules and thus replace the one-to-many relationship from the ENTITY ASSOCIATION RULE to the ENTITY ASSOCIATION with a many-to-many relationship. For example, “Create Source Systems Inventory” and “Discover Target Requirements” could be classified as a “Peer Work Efforts” association type, and then that could have two rules of “Concurrent” and “Complementary,” thus allowing many rules for the same WORK EFFORT ASSOCIATION.

In table 4.8, you see a new WORK EFFORT with parent id of “9012” described as “Discover Source Systems Requirements.” This work effort is associated with “Discover Target Requirements.” This association has a WORK EFFORT ASSOCIATION TYPE of “Peer Work Efforts.” This means that these two work efforts are two tasks that can be grouped together as work efforts that are at the same level; in other words, they are peers of each other. They also have WORK EFFORT ASSOCIATION RULE(s) of “Substitution.” XYZ Corporation decided that as an alternative approach to creating a source system inventory that it would be acceptable to substitute “Discover Source System

Requirements.” They created this rule on the **from date** of “Jan. 1, 2009.”

In Table 4.8 you also see that the work effort “Data Management Program” has two associated work efforts that have been aggregated under it: “Enterprise Data Warehouse” and “Customer Master Project.” There are no rules associated with these instances.

The last row in Table 4.8 shows that “Hardware Purchase,”, within its context as a “Work Breakdown Structure” element of “Enterprise Data Warehouse”, has a WORK EFFORT ASSOCIATION RULE of “Exclusion.” This means that XYZ Corporation has explicitly stated that this work effort should not be part of the “Enterprise Data Warehouse” project because it has decided that it will not purchase any hardware in the data warehouse project. Would it have been better just to not include or associate “Hardware Purchase” to the “Enterprise Data Warehouse” project? By excluding the “Hardware Purchase,” XYZ Corporation is explicitly declaring that it will not be allowed to include the work effort of purchasing hardware. This is different from just not capturing the association at all.

Note

This specialized pattern is particularly useful when creating PRODUCT or PART designs. There are many rules about the interactions that exist between different components of a product, such as substitution, incompatibility, complementary, and obsolescence.

Note

Association rules are not just limited to self associations (recursive relationships). Any association between one, two, or more entities may have rules that govern their behavior beyond just categorizing the association. But the rules do seem to occur more frequently when dealing with recursive relationships.

When Should This Pattern Be Used?

We use this pattern when:

- **There is a need to define the behavior that affects recursive associations** As we saw with the last pattern, the Level 3 Recursive Pattern, this can be accomplished with an association type, however this pattern has a different semantic view that explicitly distinguishes a classification for an association from an association rule.
- **There is a need to create and drive applications from the data using this pattern** For example, an application can check to ensure that work efforts with a "precedent" rule cannot start until their associated prerequisite work effort has been completed, or the application may make sure that any calculations do not include numbers for work efforts that have been "excluded" via work effort association rules.
- **There is a need to capture the rules of an enterprise to better understand the enterprise** If an enterprise is very rule-driven when managing data, this pattern will support its ethos.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern is a more generalized form of modeling** With generalization comes confusion on how to use this structure. Care needs to be taken to ensure that the structure is well understood by all stake holders.
- **This is quite a new concept and takes time to gain buy-in with many enterprises** Many enterprises will consider this overkill. However, we have implemented

this pattern in rule-driven enterprises with great success.

> **Note**
>
> This pattern has similar strengths and weaknesses to those of the previous pattern. This past section only focused on the strengths and weaknesses of adding WORK EFFORT ASSOCIATION RULE to the pattern.

Synopsis

In this section you saw another way to view the relationships that exist between the different associations in a recursive relationship. You saw how we defined the behavior that exists between different associations. The Level 3 Recursive Pattern with Rules supports the need for a flexible solution that allows the categorization of an association and the creation of rules that govern the behavior of that association.

The ENTITY ASSOCIATION TYPE was defined as the classification of the ENTITY ASSOCIATION, such as “Work Breakdown Structure,” “Program Aggregation,” and “Peer Work Efforts.” The ENTITY ASSOCIATION RULE can be defined as the “principle or regulation governing the conduct, action, or procedure” of an association, such as “Substitution,” “Exclusion,” “Concurrent,” and “Precedent.”

This specialized pattern is particularly useful for enterprises that are business-rule driven. However, it is a more abstract pattern, with new concepts. This may be overkill or too complicated for many enterprises.

Summary of Patterns

Table 4.9 contains a synopsis of all the patterns covered in this chapter.

Table 4.9 Synopsis of the Patterns

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 1 Recursive Pattern	A very specific way to model recursive relationships. Each of the different entities represents a different level in the hierarchy. The relationships between the entities show how the different levels in a structure are related to each other.	As a part of a statement of scope. As a model to develop or validate data requirements with nontechnical business representatives. When a static structure exists for hierarchies or aggregations. To help more simply start the data modeling effort, illustrate terms being used, and raise issues with data. When there is different data and relationships that exist at different levels in the structure.	When flexibility is needed in the design, for example, when additional levels in the structure may be needed. When you need to model peer-to-peer relationships.	Easy to understand and use. It can help as part of a scope statement. Provides business rules regarding the relationships between levels in the hierarchy. More specifically addresses the business requirements. It allows specific attributes and relationships to be captured at different levels in the hierarchy or aggregation. Allows different optionality to be used between different levels of the aggregation or hierarchy.	It is not a flexible model. May contain similar information at each different level (e.g., **estimated hours** at PROJECT, PHASE and TASK level) which can cause confusion if the data is defined inconsistently. This pattern does not support peer-to-peer relationships. This pattern does not support many-to-many recursive or enforce one-to-one relationships.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 2 Recursive Pattern	This pattern supports any type of one-to-many recursive relationship and provides a level of flexibility that allows new levels to be added to hierarchies or aggregations.	When you need a flexible solution that allows you to add new levels in a hierarchy or aggregation without having to change the model. When each of the levels in a hierarchy or aggregation has the same or similar attributes and relationships. When you only need to maintain one-to-many recursive relationships.	When you need to maintain specific attributes and relationships at different levels in the hierarchy or aggregation. (However, some of this could be done by using subtypes in this pattern.) When there is different optionality between different levels of a hierarchy or aggregation. When there is a need to maintain many-to-many recursive relationships.	A flexible way to capture recursive relationships in that it allows any number of additional levels to be added. Can be used as a common way to model recursive relationships, and thus, the enterprise can create common routines and can have consistent ways of handling recursions. Can maintain some rules in that it captures one-to-many recursive relationships.	It is more abstract than the level 1 pattern and therefore more difficult to use to develop or validate data requirements. It does not allow the flexibility to add additional types of recursive relationships that may emerge without changing the model. For example, if a new type of recursion such as "incompatibility" was needed, there would need to be another recursive relationship added. Can create redundant data at different levels in the hierarchy. For example, **estimated hours** may be able to be derived based on the lower levels, but this is included at all levels. Does not maintain specific data requirements for each level (different relationship optionality at different levels, different attributes at different levels, and different relationships at different levels). Forces all levels in the structure to have the same attributes even if there are differences between the level in terms of their attributes (although this could be accommodated using subtypes) Does not model the business rules that manage what is allowed such as what associations are mutually exclusive (and so on). Does not maintain specific business rules regarding the relationships between the levels. (For example, does not show the specific relationships and rules such as that a PROJECT is made up of one or more PHASE(s)).

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 2 Expanded Recursive Pattern	This pattern retains the one-to-many relationship seen in the Level 2 Recursive Pattern, and it also allows many-to-many recursive relationships.	When there is a need for flexibility where new levels may be added to a hierarchy or aggregation. When an enterprise has a need to capture many-to-many in addition to one-to-many recursive relationship. When there are variations of the same type of many-to-many recursive relationship, for example, variations of "partial" or "complete" for a WORK EFFORT PRECEDENT relationship. When the different levels of the hierarchy, aggregation, or peers have the same attributes and relationships. When there is a need to maintain all different types of recursive relationships.	If there is a need for more flexibility to add new types of recursive relationships if they emerge. When there are specific attributes and relationships at different levels in the hierarchy or aggregation. (This could be resolved by using subtypes.) When there is different optionality between different levels of a hierarchy or aggregation.	Provides the flexibility of maintaining any number of recursive relationships with any types of cardinality (1:M or M:M). Provides some business rules by showing each recursive relationship. Can be used as a common way to model recursive relationships and thus the enterprise can create common routines and have consistent ways of handling recursions. This can be a good implementation when you need some flexibility but the number of recursive relationships is stable and unchanging.	It is somewhat abstract and difficult to understand. Does not model the business rules that manage what is allowed such as what associations are mutually exclusive (and so on). Can create redundant data at different levels in the hierarchy. For example, **estimated hours** may be able to be derived based on the lower levels, but this is included at all levels. Does not allow the flexibility to add additional types of recursive relationships without changing the model. Does not maintain specific data requirements for each level (e.g., relationship optionality, different attributes at different levels, different relationships needed at different levels).

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 3 Recursive Pattern	The Level 3 Recursive Pattern supports the need for a very flexible solution that allows the dynamic addition (and retirement) of all different types of recursive relationships that could emerge over time.	When a lot of flexibility is needed because new types of recursive association are needed over time or additional types of recursive associations are discovered. When there is a need to support any type of recursive relationship by generalizing it into a many-to-many relationship. When there is a need to use a very consistent approach to capturing various types of recursive associations. This helps in creating a common set of business rules for all roles that can be created around the generic ENTITY ASSOCIATION and ENTITY ASSOCIATION TYPE. As the basis for an implementation that accommodates change very easily, for example, being able to add new association types without changing the model. When the different levels of the hierarchy, aggregation, or peers have the same or similar attributes and relationships.	As part of a scope statement for nontechnical business representatives. As the basis for a model to develop or validate data requirements with nontechnical business representatives. There is a need to capture specific business rules for each level in the structure. When you have specific attributes and relationships at different levels in the hierarchy or aggregation. (This could be resolved by using subtypes.) When there is different optionality between different levels of a hierarchy or aggregation.	Provides the flexibility of maintaining any number of recursive relationship types. Provides the capability of being able to add new types of recursive relationships as they emerge without changing the model. Can be used as a very common way to model recursive relationships, and thus the enterprise can create very common routines and have consistent ways of handling recursions. This pattern serves as an effective basis for an implementation in a changing dynamic environment.	Can be hard to understand for non–data professionals because it adds another new level of abstraction. This more generalized pattern can lead to confusion, for example, in misinterpreting what each association type is meant to represent. This pattern does not maintain the business rules for the associations such as which instances are mutually exclusive (and so on). Can create redundant data at different levels in the hierarchy. For example, **estimated hours** may be able to be derived based on the lower levels, but this is included at all levels. Does not maintain specific data requirements for each level (e.g., relationship optionality, different attributes in different entities, different relationships needed at different levels).

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 3 Recursive Pattern with Rules	This pattern supports the generalized way to maintain recursive associations as in the Level 3 Recursive Pattern but also brings in the concept of maintaining rules that exist for associations. This pattern addresses the need to distinguish between classifications of a recursive relationship from the rules that govern the behavior of the recursive relationship.	When an enterprise wants to flexibly capture different recursive associations and accommodate additional types of recursive associations that may emerge without needing to change the model. When the enterprise wants to capture the rules that govern the behavior of the elements in the structure. For enterprises that have a desire to move towards rule-based data structures.	When an enterprise does not subscribe or need the rules between associations or when the enterprise decides to capture these rules as association types as the previous pattern did. When you have specific attributes and relationships at different levels in the hierarchy or aggregation. (This could be resolved by using subtypes.) When there is different optionality between different levels of a hierarchy or aggregation.	Allows the capability to maintain the rules about what is allowed regarding the recursive relationship instances. Facilitates data driven applications that can refer to these data structures instead of hard coding the rules in the application. Allows for better understanding of the rules in the enterprise. Allows you to capture a rich vein of information about the different associations and rules in recursive structures. This is a good implementation in a changing dynamic environment, for example, when there are additional types of recursive relationships and additional rules emerge over time.	This is the most complex and abstract of the patterns in this chapter, thus there may be more confusion and it may be more difficult to gain buy-in. May be overkill and may not be needed depending on the circumstances. Does not maintain specific data requirements for each level (e.g., relationship optionality, different attributes at different levels, different relationships needed at different levels). Can create redundant data at different levels in the hierarchy. For example, **estimated hours** may be able to be derived based on the lower levels, but this is included at all levels.

References

[1] For more about the organization of data see Chapter 5 of this book.

[2] This definition was taken from Ananda Amatya (March 1999). Available at http://www.dcs.warwick.ac.uk/~ananda/umlNotes/node155.html.

[3] This definition was taken from Alf A. Pederson's article “Entity Relationship Modeling” (April 2005). Available at http://www.devarticles.com/c/a/Development-Cycles/Entity-Relationship-Modeling.

[4] This definition was taken from the Free On-line Dictionary of Computing (© 1993–2008 Denis Howe). Available at http://foldoc.org/index.cgi?query=hierarchy&action=Search.

[5] Definition taken from Dictionary.com.

[6] Please refer to Chapter 6 of The Data Model Resource Book, *Revised Edition, Volume 1, A Library of Universal Data Models for all Enterprises*, by Len Silverston (Wiley. 2001) for a discussion of this specific example.

[7] Source for this definition is the Universal Data Models Repository. © Universal Data Models 2001–2008.

[8] You can find a very interesting discussion on many-to-many recursive relationships at the Steve Hoberman & Associates web site at http://www.stevehoberman.com/DesignChallenge/challenge11response.htm.

[9] You can find good guides for the creation and resolution of many-to-many relationships in CASE*METHOD: Entity Relationship Modeling by Richard Barker (Addison Wesley, 1990) or Data Modeling Made Simple by Steve Hoberman (Technics Publications, 2008).

[10] See The Relational Database by John Carter (International Thomas Computer Press, 1995) and Database Systems: A Practical Approach to Design, Implementation, and Management, 3rd Edition, by Thomas M. Connolly and Carolyn E. Begg (Addison-Wesley, 2004).

Chapter 5

Types and Categories: the Classification of Data

Since the 1980s we have seen the amazing growth in data content around the world, in part due to the World Wide Web being part of people's lives. How do we manage this data content, how do we begin to get our minds around this myriad of information? Biologists have come to the rescue. They have been using classifications, also known as taxonomies, since the days of Carl Linnaeus,(1) the great Swedish naturalist.

It seems to be part of human nature to create classifications. It helps people to organize complex sets of data into categories. Classifications give humans a common vocabulary for grouping data sets. For example, most enterprises need reports and analytics to manage their business. Senior management needs to see revenue reports categorized by business line, and a technical call center's work gets driven by exception reports sorted by various types of errors. Manufacturing companies need to report on available inventory by product type and so on. People working in these enterprises need to analyze the data in the reports, summarize the report information into classes, or drill into the detail of a class to the underlying data. The categories in the reports are a common vocabulary between departments and people in an enterprise. This common language requires that various types of data can be grouped together into different classifications.

What Is the Significance of This Type of Pattern?

Data models often capture various categories, types, groups, families, and other ways to classify entities. For example, there may be product groups, product types, product families, customer types, facility types, industry classifications, and many other classifications of entities. Often classifications are modeled independently or on-the-fly without a firm understanding of common ways to model them. It is crucial, for data-driven enterprises, to a have a consistent and robust way to classify data.

What are the fundamental pieces of information that need to be supported about classifications? The types of information about classifications that we will discuss in this chapter are:

- How are the classifications related to the data they are classifying? For example, how is product type related to product?
- How are classifications related to each other, are there different types of categories, and can one category be a member of another category?
- What are the various ways that an entity may be classified? For example, can a mobile phone model be classified as a “Luxury good” and “Electronic device” at the same time?

What Is in This Chapter?

The terms type, categorization, and taxonomy are sometimes used interchangeably when talking about classifying data. Do they mean the same thing? To address this, we initially define what classifications, types, categorizations, and taxonomies are and discuss the

differences between them. The chapter then describes the data model patterns used to support the needs of an enterprise when working with classifications. These patterns describe the different ways in which we organize classification data.

Like most of the chapters in this book, the style of modeling for each of the patterns starts with the most specific style (Level 1 Classification Pattern) and progresses through the chapter to a very flexible style (Level 3 Classification Pattern). The different levels of generalization may be applicable to different enterprises or styles of modeling. For example, an enterprise wanting to create a simple scope statement for understanding data requirements may choose a level 1 pattern. For a more flexible solution that may be more suitable for an implementation, they may choose a level 2 or level 3 Pattern(2).

In this chapter we use the entity PRODUCT to illustrate how to use the different versions of this pattern. It is important to note that PARTY, ASSET, SHIPMENT, POLICY, PAYMENT, WORK EFFORT, or any other entity that needs to be classified, can use these patterns in a similar way.

This chapter includes:

- The definition of classifications
- The different patterns that support classifications
- The relevance of each pattern
- Insights into each pattern
- When to use and not to use different patterns
- A synopsis of each of the patterns, pros, and cons

What Are Types, Categorizations, and

Taxonomies?

Types, categories, and taxonomies are closely related concepts, but each has a slightly different meaning. Each of the patterns described in this chapter may support a type, category, and/or a taxonomy.

Types may be defined as “a number of things or persons sharing a particular characteristic, or set of characteristics that cause them to be regarded as a group, more or less precisely defined or designated.”(3) In our patterns, a type entity classifies the ‘base’ entity we are trying to classify in a very simple way. For example, if we are classifying an ORDER, then there may be a related ORDER TYPE that would contain instances of “Sales Order” or “Purchase Order” which could be considered types of orders.

Categorization can be defined as “any general or comprehensive division.”(4) The key difference between a categorization and a type is that a categorization is more comprehensive than a type, and it involves different ways to classify an entity, that is, a categorization includes many different “types of types”. For example, a person may be categorized many different ways (e.g., by level of income, by gender, by race, and so on), and the categorization encompasses all these various types.

Taxonomy may be defined as any classified collection of elements. There are two different aspects of taxonomies that need to be supported. The first aspect is the scientific rigor of a classic taxonomy that you would see in natural history. “Taxonomy is the science of identifying and naming species and organizing them into systems of classification.”(5) Second, taxonomies have become a very important aspect to web-site content management and design. Taxonomies are meant to support “intuitive familial groups.”(6) Taxonomy refers to a hierarchy or aggregation of categories (and types). From the perspective of this book,

our patterns are able to support different ways to manage content by classification. For example, if we are classifying a “Party,” a taxonomy may include various types of party classification (for example, “Party type,” “Industry classification,” “Size classification,” and so on) and any relationships between the instances within these types such as how the “Party types” of “Person” and “Organization” are related to “Size classification” instances of “Large,” “Medium,” and “Small.”

Types, categories, and taxonomies also may have classifications that are within (or related to) other classifications.(7) For example, if you are classifying PRODUCT you may have PRODUCT TYPE instances of “Good” and “Service.” Then you may further break each of these down into other classifications. For example, you may break down the “Service” into “Time and materials service” (for services that are billed by the hour) and “Deliverable based service” (for services that are billed based upon the delivery of a product).

Note

Although we have defined types, categories, and taxonomies, this chapter mostly focuses on modeling “types” for the level 1 and level 2 patterns and “categories” for the level 3 pattern because we consider a “category” to be more comprehensive and encompassing than a simple “type.” When we refer to a classification, this includes “types” and/or “categories.” Taxonomies are made up of “types” and/or “categories,” so each of the patterns support taxonomies to different degrees of complexity.

Level 1 Classification Pattern

A very simple way to capture categories and types of an entity is to use attributes—for example, an attribute **order**

type maintaining that an ORDER is in fact a “sales order,” “purchase order,” “work order,” or a “manufacturing order.”

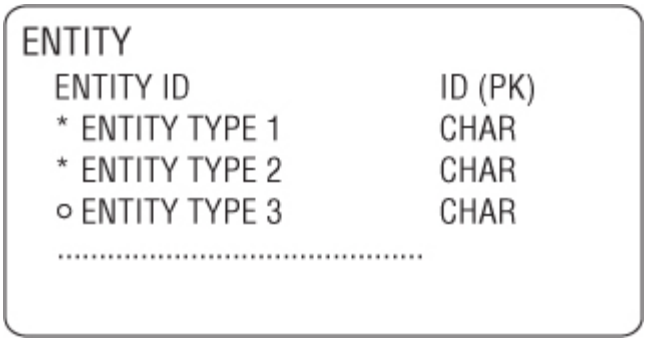
Why Do We Need This Pattern?

Under specific circumstances enterprises often need a very simple and easy way to model classifications. The Level 1 Classification Pattern provides this very specific way to model classifications. All the classification information is contained inside the entity as attributes. In other words, the entity captures the classifications of the entity as attributes (for example, **order type**), and the instances of each of these attributes maintains the type of entity it is (for example, “sales order,” “purchase order,” “work order,” or a “manufacturing order”).

How Does This Pattern Work?

Figure 5.1 is a pattern that provides a very specific model for an entity with different types. ENTITY contains three different classifications, **entity type 1, entity type 2**, and **entity type 3**. This pattern could include any number of type attributes and we are just showing three of them for illustration purposes. Each of these classifications may, or must, contain the different “types” for the ENTITY. For example, ENTITY must have values for **entity type 1** and **entity type 2**, but may be classified by **entity type 3**.

Figure 5.1 Level 1 Classification Pattern



Imagine that a data professional is examining how a large computer hardware and software retailer called Euro-Electronics is classifying its data as part of a data warehouse project. Assume the data professional has been asked to provide a scope statement of all the different possible classifications that Euro-Electronics has for its products. Based on interviews with management and using the Level 1 Classification Pattern, the data professional produced Figure 5.2.

Figure 5.2 Example of using a Level 1 Classification Pattern

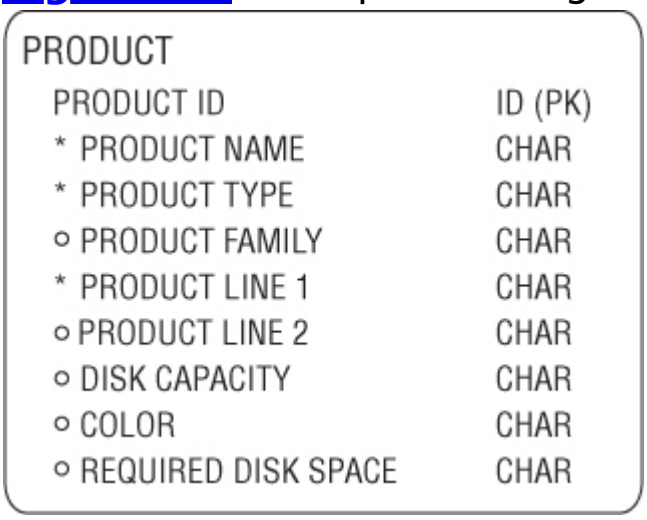


Figure 5.2 shows an example of how to use this pattern. Euro-Electronics has defined PRODUCT as "hardware, software, or accessories that is, was, or will be offered for sale to a customer". **Product type**, **product family**, **product line 1**, and **product line 2** contain the different "types" of classifications that classify an instance of PRODUCT in different ways. The reason that there are **product line 1** and **product line 2** attributes is to allow for a product being in more than one product line (for example, classified as for both "commercial use" and "home use").

Note

A special case of classifications are indicators, for example, "M(ale)" and "F(emale)," to indicate gender, or "Y(es)" versus

"N(o)," for indicating if a person is a smoker. The indicator is maintaining a specific piece of knowledge about that instance of an entity by offering a choice between two values, for example, that a particular PERSON is male (versus female) or that the PERSON is a smoker (by choosing a value of "Y(es)" for the **smoker indicator** attribute). Most classifications support many possible data values that can classify an entity, whereas indicators can be used to classify data into two specific possibilities. However, the modeler needs to be very careful when using an indicator attribute because the number of possible value could expand to more than two. For example, instances of a **gender indicator** could expand to be "M(ale)", "F(emale)", and "N(ot given)."

An alternative view is that indicators are semantically different from classifications, because an indicator should only be used to capture a specific piece of knowledge about that instance of an entity (or that row in a table) e.g., if a PERSON is male (versus female) or that the PERSON is a smoker or a nonsmoker. This view has some merit, but even if you subscribe to this view, people in your organization will, more often than not, still use the indicator to classify an entity.

Table 5.1 provides examples of the same product being classified many different ways. You can see in the first row that the "Save Disk 2000" instance of PRODUCT is classified into **product type** ("Hardware"), **product family** ("Disk Drives"), **product line 1** ("Home Use"), and **product line 2** ("Commercial Use"). As we said, **product line 1** and **product line 2** accommodate the need to classify a product into more than one product line; for example, "Save Disk 2000" is classified into the **product line 1** of "Home Use" and the **product line 2** of "Commercial Use." In the last row you see that the "Standard Memory Card" also has multiple classifications of "Hardware" (the **product type**), "Computer" (the **product family**), "Home Use" (**product line 1**), and "Commercial Use" (**product line 2**).

Table 5.1 Examples of Level 1 Classification Pattern

PRODUCT. PRODUCT ID	PRODUCT. PRODUCT NAME	PRODUCT. PRODUCT TYPE	PRODUCT. PRODUCT FAMILY	PRODUCT. PRODUCT LINE 1	PRODUCT. PRODUCT LINE 2	PRODUCT. DISK CAPACITY	PRODUCT. COLOR	PRODUCT. REQUIRED DISK SPACE
102	Save Disk 2000	Hardware	Disk Drives	Home Use	Commercial Use	200 GB	Black	
103	Carry All Case	Accessory	Carrying Case	Commercial Use			Green	
105	Hyper Sales Software Package	Software		Home Business				154 MB
109	Standard Memory Card	Hardware	Computer Memory	Home Use	Home Business	1 GB		

The PRODUCT has three other attributes. **Disk capacity** maintains the amount of available space on a disk or memory device and is only applicable for hardware. **Color** maintains the color of an accessory and is only maintained for accessories such as carrying cases. **Required disk space** maintains the footprint (or needed space in MB) and is only maintained for software products.

Note

The PRODUCT attributes of **color**, **disk capacity**, and **required disk space** may also be considered to be ways of classifying various products. However, for the purposes of this chapter and to simplify the examples, we will focus our discussions on the classifications of **product type**, **product family**, and **product line**. **Disk capacity**, **color**, and **required disk space** could also have been modeled as features. For more on this concept, see Chapter 3 of *The Data Model Resource Book*, *Revised Edition, Volume 1* (Wiley, 2001).

From Table 5.1 (as well as Figure 5.2) you can see that not all of the categories are mandatory. For example, “Hyper Sales Software Package” is not classified by any product family. The data professional also discovered that the same product may be classified into multiple product lines; hence, the repeating attributes of **product line 1** and **product line 2** accommodate classifying a product into two different product lines. This repeating group highlights some data issues that the data professional needs to investigate further because perhaps there could be even more product lines that may be needed for a product. (For example, what if a product is classified as “Home Use,” “Business Use,” and “Home Business”?)

If you are familiar with data modeling, it is probably evident that this pattern is not normalized. It fails first normal form (no repeating groups), as just discussed regarding **product line 1** and **product line 2**, because

there could be many (repeating) product line classifications for a product. This model also fails third normal form (transitive dependency) because the values of each “type” attribute is actually not dependent on the key of a PRODUCT but is really dependent on its own key. For example, the value in the first row of “Disk Drives” is not dependent on the PRODUCT key of “102” since many products may have this same value of “Disk Drives.” Therefore, the value of “Disk Drives” should, according to normalization rules, be maintained in its own entity (e.g., a PRODUCT FAMILY entity), which could have a primary key of “111” with a **name** attribute, for example, “Disk Drives.” In this case, the value of “Disk Drives” is dependent on and can be determined by the value “111.” Then, if this was split out into its own entity, there would be a foreign key with the value “111” in the PRODUCT entity to signify if something is classified as “Disk Drives.” This would eliminate the redundancy of maintaining the same data of “Disk Drives” in several instances of PRODUCT.

Thus, this pattern can create huge data redundancies and inconsistencies if it is implemented in a relational database. For example, the value “Disk Drives” may be repeated for many PRODUCT instances and could be inconsistently spelled, or when this lookup value changed to “Hard Disk,” it could cause data integrity issues with all the duplicate and redundant instances of this value.

So you may be thinking, “If this model is *wrong*, why would you show a wrong data model as a possible pattern?” The reason for showing this pattern is that it can be used as a way to identify the possible types of data that may exist in a very simple format in order to communicate and validate data requirements with nontechnical audiences. Many modelers in the industry refer to this as a conceptual or business data model whose purpose is to illustrate data needs in a very simple way to nontechnical audiences. In

1975, ANSI (American National Standards Institute) came up with the classifications of “Conceptual Schema,” “Logical Schema,” and “Physical Schema” to allow different perspectives of the data requirements, which is the same idea that we are illustrating.

Thus, we want to emphasize that we would almost always not recommend to use this pattern (or for that matter, most of the level 1 patterns) as a basis for any long-term or significant implementation because there are great pitfalls that exist in implementing this pattern, such as redundant and inconsistent data. However, instead of having to show a “normalized” data model to a business representative (who doesn't care if the model is normalized or not and only cares that you have captured their needs), we have found that this type of pattern can be an effective way to illustrate data requirements to communicate with business representatives in order to better understand their needs.

We have emphasized not to use this pattern for implementations; however, there may be rare cases where the classifications and instances of classifications are completely unchanging or static and where redundant data does not cause major issues. You could make an argument that if the classification required only a simple attribute, you knew that the entity would only ever have this one classification, and the physical implementation benefited from a non-normalized design, this may be a possible implementation. For example, perhaps it could be a possibility to use a **gender type** attribute for PERSON where the values are and always will be “M(ale),” “F(emale),” and “N(ot Given).”

It is also worth noting that many legacy systems implement classifications as attributes (or columns). One reason for this is that many legacy systems were not relational originally. They may have been moved onto a relational platform, but they may not have been

transformed using normalization rules. For this reason we think that it is important to understand this pattern, its strengths, and its weaknesses as it represents the reality of how many systems implement classifications.

When Should This Pattern Be Used?

We use this data model pattern:

- **There is a need for a very simple model to gather and validate data requirements:** This type of model can be used to illustrate the scope of the classifications for an entity in a very easy-to-understand format. This simple format may have more resonance with business people.
- **As a simple implementation for a prototype:** Having the classifications as attributes makes it easier to build a quick prototype. But beware that the prototype does not become the end product.
- **As a way to stimulate analysis of the different classification structures:** It helps expose the nomenclature that the business uses to categorize its data. It raises issues they have with the data, such as repeating **product line** attributes. This pattern often helps to start the data modeling process in a simple, very understandable way.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **Maintains redundant data and not meant to be used as the basis for a significant or long-term implementation:** This pattern maintains redundant data because each value for a classification may be repeated many times inconsistently across different instances of the entity and cause data integrity issues if

implemented. *This is not meant to be the basis for implementation, except in rare cases*. For example, if "Manufacturing" is maintained in an industry-type attribute that classifies a PARTY, each instance of PARTY may have inconsistent values for the same industry type (for example, "Manufacturing" and "Mfg.").

- **Very inflexible:** With any rigid structure, change may cause a data model based on this pattern to be reworked. Thus if you implement the database based on this pattern and changes occur, the implemented database structure may need to be changed, which could be quite expensive over time. For example, when you need to add another category, you need to add a new attribute. A PARTY may have a great number of classifications such as size, industry, income, race, gender, market segment, and many more may emerge over time.
- **This method may not be practical when an entity may be classified into the same "type" many times:** For example, consider the example where a PARTY may be classified into several industries at the same time. (The same company that operates in different markets may be classified into manufacturing, health insurance, disability insurance, telecommunications, retail, and so on.) This pattern requires that there be an attribute for each industry that a party may be in (industry type 1, industry type 2, and so on). Thus, this pattern may not be practical for this type of modeling requirement. You saw in the example in this section that two **product line** attributes were needed to support the business needs.
- **Does not maintain data about the classification:** Often categories have many different attributes such as a description, from and thru dates, and so on. To accommodate this, each of these attributes would have

to be added to the entity, and even so, it becomes very awkward to add these attributes for each type. With this pattern, attributes are maintained in the overall entity even though some attributes may apply to a particular classification.

- **Does not maintain many business rules about the classifications:** This pattern can handle only the most basic business rules such as mandatory classifications versus optional classifications.

Synopsis

In this section you saw that the Level 1 Classification Pattern provides a very specific way to model classifications. This pattern is significant because of its simplicity. This pattern uses attributes to capture which instances of an entity are classified into which “types.” The level 1 pattern is useful as part of a statement of scope and visual representation of the different classifications for a particular entity. This pattern can help you start your data modeling effort in a very straightforward fashion, exposing the terminology a company uses to categorize its types. It may even be useful as basis for a prototype.

This pattern is not meant to be used as the basis for implementing databases for almost all circumstances because there are usually serious pitfalls regarding data redundancy and data integrity issues in this pattern. Also, if new categorizations need to be added or if the nature of the classification changes, new attributes need to be added, and thus, this pattern does not provide much flexibility for future needs.

Level 2 Classification Pattern

Many enterprises need a more flexible and normalized approach when modeling classification data. In the previous pattern, each of the entities was classified into types that were captured as attributes. In this pattern the types are not captured as attributes, but in their own entities. This is a significant semantic leap from the previous pattern. This means that classifications may exist independently and can be managed independently from their related entity.

Why Do We Need This Pattern?

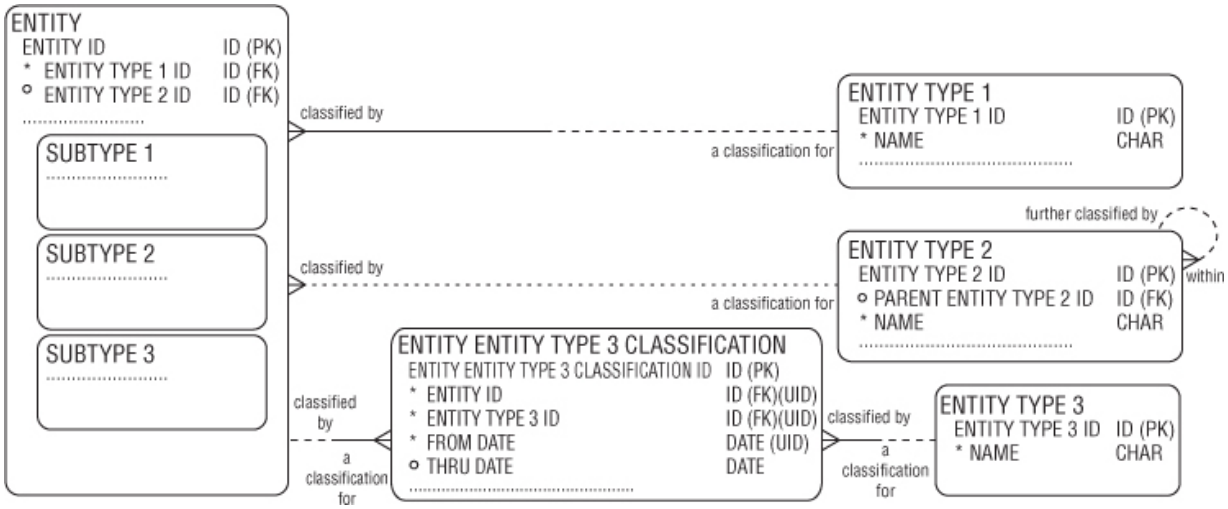
In this pattern, by modeling classifications as entities, you can create, delete, and manage the “types” independent of the entity they are classifying. This also provides for additional flexibility in that the same instance of a type entity can classify many instances of an entity, and an entity can be classified by many instances of the same type. For example, a PARTY may be classified by an INDUSTRY TYPE, which is maintained in its own entity. Thus, an instance of a PARTY may be related to many instances of an INDUSTRY TYPE (e.g, “Manufacturing” and “Distribution”) and the same INDUSTRY TYPE (e.g., “Manufacturing”) may be used to classify many instances of PARTY.

In the previous pattern we saw a very specific way of modeling classifications as attributes. Many legacy systems have implemented classifications this way. The Level 2 Classification Pattern provides an alternative that can be used to convert Level 1 Classification Pattern structures, with all their inherent weaknesses, to a more robust relational way to support classifications. A real strength of the patterns, in general, is that they illustrate how specific patterns can be converted into more flexible generalized alternatives (and vice versa).

How Does This Pattern Work?

Figure 5.3 contains a pattern for modeling classifications with more flexibility and less redundancy than the level 1 pattern. ENTITY represents any data that the data professional is modeling, for example, PARTY, PRODUCT, ASSET, WORK EFFORT, or ORDER. ENTITY may be classified into multiple different subtypes as represented by SUBTYPE 1, SUBTYPE 2, and SUBTYPE 3. You see that ENTITY may also be classified in different ways via ENTITY TYPE 1, ENTITY TYPE 2, and/or ENTITY TYPE 3, plus additional entity types if needed.

Figure 5.3 Level 2 Classification Pattern



Although it may seem redundant to classify an entity with a subtype as well as via “type” entities, there is often a need to model types both ways. Subtypes accommodate additional attributes and relationships that may be needed for a specific subtype. For example, if a PRODUCT was subtyped into GOOD or SERVICE, there may be specific attributes and/or relationships for a GOOD, such as its related INVENTORY ITEM(s), that are not there for a SERVICE. And there may be a type entity such as PRODUCT TYPE with values of “Good” and “Service,” which also may have specific attributes and/or relationships. For example, the PRODUCT TYPE may be related to a DISCOUNT that

stores a particular discount off certain product types for a specified time period. Thus, the “type” entity is needed also.

In this pattern you see that “each ENTITY *must be* **classified by** one and only one ENTITY TYPE 1,” and “each ENTITY *may be* **classified by** one and only one ENTITY TYPE 2.” This illustrates that some classifications may be mandatory for the ENTITY and some may not be. For example, there may be an example of this pattern where a WORK EFFORT *must be* **classified by** a WORK EFFORT TYPE (such as a “Project,” “Task,” or “Activity”). In the same model, perhaps there may also be an optional relationship to a “type” entity where a WORK EFFORT(s) may be (or may not be) classified by a WORK EFFORT PURPOSE TYPE, (such as “Maintenance,” “Research,” and so on). Because the WORK EFFORT may not have an assigned purpose at the beginning of the WORK EFFORT(8), the relationship to WORK EFFORT PURPOSE TYPE may not be mandatory.

The ENTITY TYPE 3 provides for the possibility of a many-to-many relationship between an ENTITY and its “type” entity. It is often possible for an instance of ENTITY to be classified by more than one instance of the same ENTITY TYPE 3, and it's also possible that an instance of ENTITY TYPE 3 can classify more than one instance of ENTITY. For example, an organization may be classified by several industry types such as being a manufacturing company, a telecommunications company, and a services company all at the same time. In this case, INDUSTRY TYPE is an example of the ENTITY TYPE 3, and a specific ORGANIZATION could have three instances of INDUSTRY TYPE (manufacturing, telecommunications, and services). Each instance in INDUSTRY TYPE (for example, “manufacturing”) could be applied to more than one instance of ORGANIZATION. Figure 5.3 resolves this many-to-many relationship with ENTITY ENTITY TYPE 3 CLASSIFICATION. This (resolved) many-to-many relationship

allows for an instance of ENTITY to be classified by more than one instance of ENTITY TYPE 3 and each ENTITY TYPE 3 instance may classify more than one instance of ENTITY.

Though the pattern only shows three types with one mandatory, one nonmandatory, and one many-to-many relationship, the intention of this pattern is that you may have any number of 'type' entities to classify the ENTITY—some mandatory, some optional, and some many-to-many, depending on the needs.

Note

From one perspective, there are two types of classifications:

- Classifications that are mutually exclusive—A product may be "Hardware," "Software," or an "Accessory," and the product cannot have more than one classification at the same time.
- Classifications that may have multiple classifications at the same time—A product may be in several product lines at the same time (for example, a product is classified to be a "Home Use" and "Commercial Use" product).

The first type will generally lead to a one-to-many relationship from the "type" to the related "entity," and the second type will involve a many-to-many relationship.

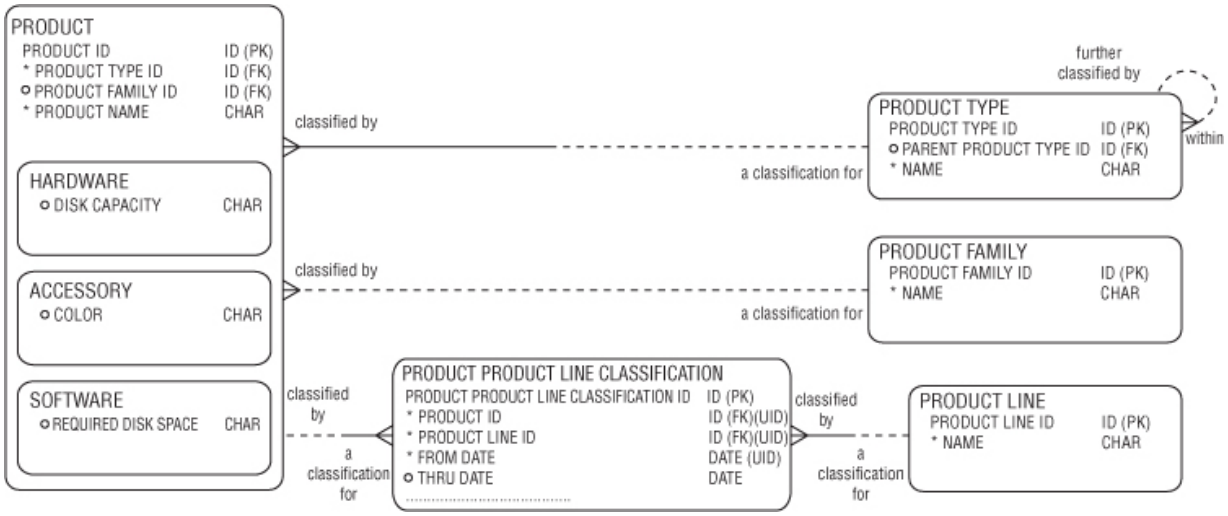
Some "type" entities may be recursive, in other words there may be types of types. For example, a WORK EFFORT TYPE entity may have instances of "Project" and "Activity," and then an "activity" instance may be further broken down into a "task" or "job" (or any way that the enterprise sees the breakdown), and so on. So the recursive relationship shows how a "type" entity instance is related to other "type" entity instance. Some "type" entities may only have

a flat set of values and not a hierarchy (or aggregation) of the various types, so they may not need a recursive relationship on the “type” entity. Therefore, to signify this as an option, ENTITY TYPE 2 has a recursive association using the **parent entity type 2 id** foreign key that relates one instance to another instance of **ENTITY TYPE 2**.(9)

> **Note**
>
> We capture only one attribute in each “type” entity (for example, “name”), in Figure 5.3. Many other attributes could also have been captured, such as **description**, **short name**, **code**, **from date**, **thru date**, **current indicator**, and so on.

Figure 5.4 further illustrates how this pattern would be used. We can continue with the scenario seen in the previous section. The hardware and software retailer called Euro-Electronics is classifying its data as part of a data warehousing project. Based on interviews with management and sales, and using the Level 2 Classification Pattern, the data professional produced Figure 5.4.

Figure 5.4 Example of using a Level 2 Classification Pattern



The data professional discovered that the PRODUCT entity has three main subtypes, namely, HARDWARE, ACCESSORY,

and SOFTWARE. Each of these subtypes has data that is specific to that subtype.

The data professional previously discovered that PRODUCT(s) may be classified in three different ways. You can see this in Figure 5.4, with the entities PRODUCT TYPE, PRODUCT FAMILY, and PRODUCT LINE. Each of these different ways to classify PRODUCT represents a different view that Euro-Electronics uses to manage, analyze, and report on its PRODUCT(s). Each of the different classifications in Figure 5.4 follows the same basic pattern as seen in Figure 5.3. Thus, a particular PRODUCT may be classified into a PRODUCT TYPE such as “Hardware,” a PRODUCT FAMILY such as “Disk drives,” or a PRODUCT LINE such as “Commercial Use.”

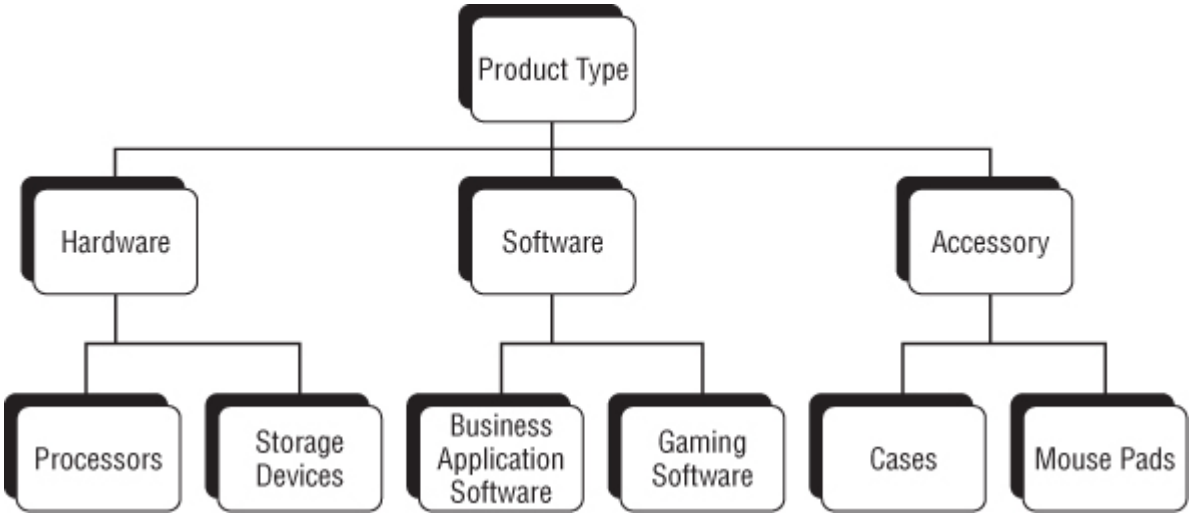
The PRODUCT TYPE(s) represent the most common way that different divisions of Euro-Electronics classify products. You can see in Figure 5.4 that in this model, PRODUCT has three subtype(s) (HARDWARE, SOFTWARE, and ACCESSORY) that are used by Euro-Electronics to classify (and manage) its PRODUCT information. We have already stated that these subtypes (HARDWARE, SOFTWARE, and ACCESSORY) are captured because they have their own attributes and relationships. But it also makes sense to capture “Hardware,” “Software,” and “Accessory” as instances of PRODUCT TYPE. This allows other entities to be related to a PRODUCT TYPE. For example, there may be a PRICE COMPONENT RULE entity or a REGULATION that is related to and dependent on the PRODUCT TYPE (see Chapter 8 for more details on PRICE COMPONENT RULE). Figure 5.5 shows additional instances that may be included in PRODUCT TYPE.

Note

A rule of thumb we use is this: If an entity has subtypes and is also classified by another “type” or “category” entity, those

subtypes will nearly always be instances in the “type” or “category” entity.

Figure 5.5 Product Type



In Figure 5.4, PRODUCT TYPE has a recursive association around it. In other words, “each PRODUCT TYPE(s) *may be* **further classified into** one or more PRODUCT TYPE(s) and each PRODUCT TYPE *may be* **within** one and only one PRODUCT TYPE.” Euro-Electronics had a need to classify products at multiple levels. For example, it wants to roll up all lower level types such as “Processors,” “Storage Devices,” “Business Applications Software,” and so on, into higher level types such as “Hardware,” “Software,” or “Accessory.” This allows senior management to answer questions such as “What were last year's sales for my hardware products?” or “What are next year's forecasted sales for storage devices as a percentage of all hardware sales?”

In Figure 5.5 and Table 5.2 you see three different hierarchies of PRODUCT TYPE(s) where you can classify HARDWARE products as “Processors” or “Storage Devices”; classify SOFTWARE products as “Business Application Software” or “Gaming Software”; or classify ACCESSORY(s) into “Cases” or “Mouse Pads.” In other words, this pattern

supports the organization of classifications into these types of hierarchies or aggregations.(9)

Table 5.2 Examples of Hierarchy of Types

PRODUCT TYPE. PRODUCT TYPE ID (CHILD)	PRODUCT TYPE. NAME (CHILD)	PRODUCT TYPE. PARENT PRODUCT TYPE ID	PRODUCT TYPE. NAME (PARENT)
10	Processors	1	Hardware
13	Storage Devices	1	Hardware
14	Business Application Software	2	Software
16	Gaming Software	2	Software
12	Cases	4	Accessory
17	Mouse Pads	4	Accessory
1	Hardware	100	Product Type
2	Software	100	Product Type
4	Accessory	100	Product Type

You might ask if this means that products must be classified by the lowest level of PRODUCT TYPE. For example, a PRODUCT should not be directly classified by "Hardware," but instead by "Processors" or "Storage Devices." No, it does not mean that products *must* be classified at the lowest level of PRODUCT TYPE(s) according to the data model. However, we believe that, as a general rule, they generally *should* be. For example, if you classified a product "Acme Video Card" as "Hardware" instead of "Processors," would you be able to answer this question: "What percentage of hardware sales were processors?" There is also the case where a new product does not have a suitable PRODUCT TYPE. For example, imagine a new product called "Laser Pen 4000." This product is an accessory, but it is not a case or a mouse pad and is not an appropriate PRODUCT TYPE instance for this new product.

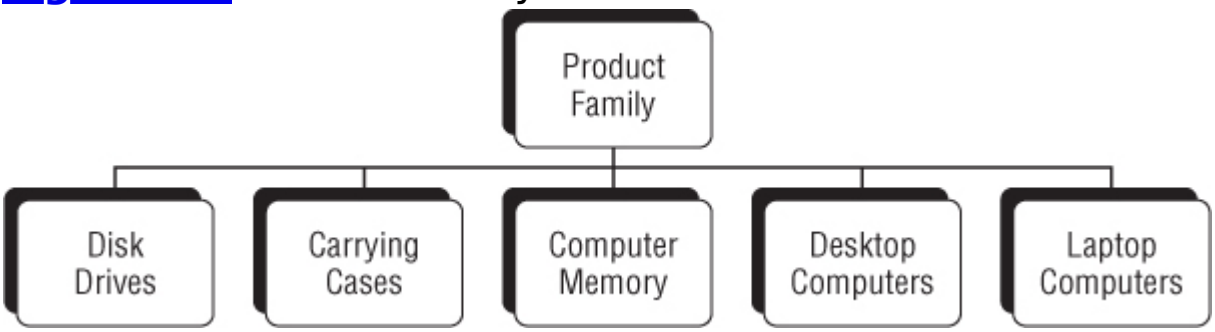
Should you classify it by “Accessory”? You may do this, but we recommend that you create a new lower level PRODUCT TYPE of “Laser pens” and then create the relationship from “Laser pens” to “Accessory” by having the instance of “Laser pens” have a **parent product type id** that relates it to the instance of “Accessory”. Then you can assign “Laser Pen 4000” to this new PRODUCT TYPE instance.

You may also wonder if the PRODUCT subtypes of HARDWARE, SOFTWARE, and ACCESSORY should have lower level subtypes such as PROCESSOR and STORAGE DEVICE for HARDWARE. Our rule of thumb is that if the type does not have any attributes or relationships specific to it, then we do not include it as a subtype and instead we include it as a “type” entity instance. In this example, HARDWARE, SOFTWARE, and ACCESSORY have their own attribute(s) and/or relationship(s), whereas “Processors,” “Storage Devices,” and other lower-level types don't have any attributes or relationships. They are just a way Euro-Electronics classifies its products. For this reason, we do not show them as subtypes of HARDWARE, SOFTWARE, or ACCESSORY.

PRODUCT FAMILY (see Figure 5.6) is a different way that Euro-Electronics wants to classify products. This classification categorizes the product according to physical and functional similarities. For example, products may be classified into product families such as “Disk Drives,” “Carrying Cases,” “Computer Memory,” “Desktop Computers,” and “Laptop Computers.” The sales force uses PRODUCT FAMILY classifications to segment PRODUCT(s) into the different needs that customers have. For example, the sales force may have a lead on a company looking for disk drives. Therefore, they could look for all the PRODUCT(s) that are classified as a PRODUCT FAMILY of “Disk Drives” to address this need, and they may recommend the “Save Disk 2000,” which is a popular

PRODUCT that has the PRODUCT FAMILY classification of "Disk Drives."

Figure 5.6 Product Family



The PRODUCT FAMILY classification is optional, as you can see from Figure 5.4, and therefore it is modeled with an optional foreign key of **product family id**. The reason for this is that sometimes there is no current relevant product family for a PRODUCT.

PRODUCT LINE (see Figure 5.7) is another way that Euro-Electronics wants to classify its products, according to how the products are used. For example, with "Home Use," "Commercial Use," "Home Business," and "Government" you can see that PRODUCT LINE(s) is closely aligned with the profit centers that are often used for financial reporting within Euro-Electronics. The interesting thing about PRODUCT LINE(s) is that the same instance of PRODUCT can be classified by many PRODUCT LINE(s). For example, the "Save Disk 2000" may be used in both "Home Use" and "Commercial Use," and the "Carry All Case" may also be classified for "Home Use" and "Home Business." In Figure 5.4 this many-to-many relationship gets resolved with the associative entity of PRODUCT PRODUCT LINE CLASSIFICATION. "Each PRODUCT *may be* **classified by** one or more PRODUCT PRODUCT LINE CLASSIFICATION and each PRODUCT LINE *may be* **a classification for** one or more PRODUCT PRODUCT LINE CLASSIFICATION(s)."

Figure 5.7 Product Line

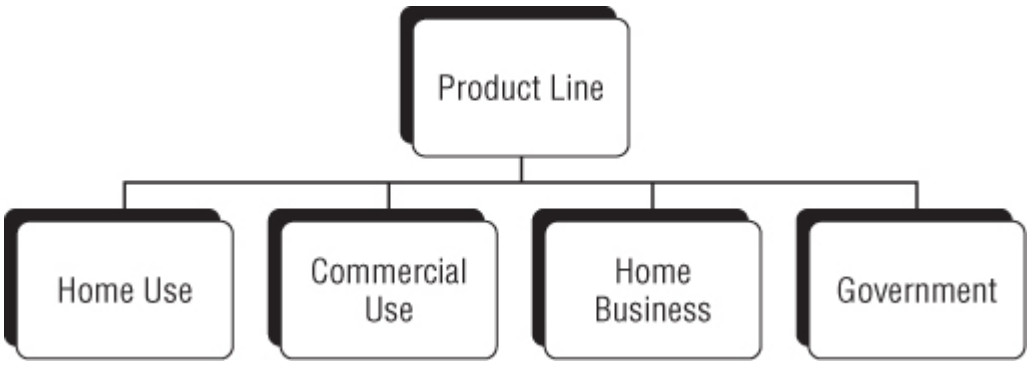


Table 5.3 further illustrates how this pattern would be used. The PRODUCT “Save Disk 2000” is classified in three different ways—first by PRODUCT TYPE, “Storage Devices”; then by PRODUCT FAMILY, “Disk Drive”; and finally by the PRODUCT LINE(s) of “Home Use” and “Commercial Use.” “Scanner Disk Fob” is classified as “Storage Device” and “Home Use” in a similar way as the “Save Disk 2000,” except that “Scanner Disk Fob” is not classified by any PRODUCT FAMILY. If the inventory manager wanted to see all the different “Hardware” products, the manager could ask for all products that are classified as “Hardware” and this could be derived by using the recursive relationship for the PRODUCT TYPE where “Hardware” is the parent PRODUCT TYPE of “Storage Devices” and “Processors,” as seen in Table 5.2. Another example is “Carry All Case.” It was classified in three different ways—first by PRODUCT TYPE, “Cases”; then by PRODUCT FAMILY, “Carrying Cases”; and finally by PRODUCT LINE(s), “Home Use” and “Home Business.”

Table 5.3 Examples of Level 2 Type Pattern

PRODUCT. PRODUCT NAME	PRODUCT TYPE.NAME	PRODUCT FAMILY.NAME	PRODUCT LINE.NAME
Save Disk 2000	Storage Devices		
Save Disk 2000		Disk Drive	
Save Disk 2000			Home Use
Save Disk 2000			Commercial Use
Scanner Disk Fob	Storage Devices		
Scanner Disk Fob			Home Use
Carry All Case	Cases		
Carry All Case		Carrying Cases	
Carry All Case			Home Use
Carry All Case			Home Business

When Should This Pattern Be Used?

We use this data model pattern:

- **As a model to understand data requirements and use as a part of a statement of scope:** We find this pattern useful in helping to document the scope of a data modeling effort and in gathering and validating data requirements because it illustrates the different ways an entity gets classified in a relatively straightforward fashion. It exposes each of the different classifications as their own entities and shows the nature of the relationships to the relevant entities. For example, in this pattern some classifications may be optional, some may be mandatory, some may have a one-to-many recursion, and some may have many-to-many associations with the entity that they are classifying.
- **When it is important to have a normalized yet specific way of modeling classifications:** Unlike the first pattern in this chapter, there is no redundant data

in this pattern (that is, it is normalized), and this is still a very specific way of modeling classifications (in the next pattern, you will see how you can model this in a more generalized way).

- **For independently maintaining classification data:** Enterprises often refer to classification data as reference data. Many enterprises have procedures and policies around managing reference data. By decoupling the classification data from and pulling it from inside another to its own entity, it's easier to manage.
- **When each of the different classification categories has its own attributes and/or relationships:** Because this type of pattern specifically decouples classification data from the data it classifies, we can add more semantics to the model. For example, PRODUCT LINE(s) may have a **short name** but PRODUCT TYPE does not, or a PRODUCT *must* be classified by one and only one PRODUCT TYPE but *may* be classified by one or more PRODUCT PRODUCT LINE CLASSIFICATION(s) (each of which is related to a PRODUCT LINE).
- **When the different classification types are well understood and static:** If the classifications are very stable and will not require changes, this may be a good option.
- **When there are relationships between the different classification types:** For example, marketing may wish to correlate its PRODUCT LINE(s) with PRODUCT TYPE(s). By exposing them as their own entities, one can create a relationship between PRODUCT LINE(s) and PRODUCT TYPE(s).

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **It is not very flexible:** It does not accommodate new types of classifications that may be added in the future or changes to the relationships from the type to the entity. For example, if the data professional had discovered a new type, such as PRODUCT GROUP, the professional would have had to create a new entity (with relationships and attributes) to support that classification type. Additionally, if the nature of the relationship from PRODUCT to PRODUCT TYPE changed to many-to-many (or became optional instead of mandatory), the data model would need to be changed to accommodate this change. For this reason, this pattern may not be suitable as a basis for implementation of a database design in a very dynamic environment.
- **Since classifications are maintained in separate entities as opposed to a single entity for all classifications, it can sometimes limit analysis capabilities:** This pattern does not provide a way to relate data to all classifications within a single entity. It is often useful to have all types of classifications in the same entity, in order to answer questions such as what types of customers are interested in what types of products (including all different ways to classify products).
- **There is not a standard way to maintain types:** Each type may have different attributes and relationships, and there may be inconsistent conventions used such as different attributes maintained for each type. (This can be considered a strength, too!)
- **Specific relationships between the various 'type' entities can lead to a model that is more difficult to manage:** If there are many relationships that exist

between various types, this can get much more complicated than having a common, flexible way to store relationships between types, such as by generalizing the different types into a single entity and then having a recursive relationship that allows all types to be related to each other (you will see this in the next pattern).

- **If an entity may be classified many ways, then classifications are harder to manage with this pattern:** Some entities (such as parties, products, assets, and work efforts) may have a great number of classification types. The same entity may be classified in many different ways depending on many factors, such as how various external data providers classify it, how it is classified in different geographic areas, and how different parts of the enterprise classify it. This pattern would require separate entities and relationships for each of these different ways to classify it.

Synopsis

In this pattern, by decoupling classification information from an entity, we make a strong argument for maintaining classification data for an entity independent of, yet related to, that entity. This means that an enterprise can manage classification data, maintain specific data and relationships for each ‘type’ entity, and the same classification entity can be related to various other entities in the data model.

Some classification data may have its own structure. You saw this in the recursive relationship around PRODUCT TYPE in Figure 5.4. This ability to provide aggregations or hierarchies of types is often important, because many enterprises require that data can be “rolled up and down” based on their classifications. You also saw that instances of an entity may or may not have a classification, and that an

instance of data often has many different classifications that may be used for different purposes.

You discovered in this section some limitations with this pattern. If new classifications are discovered, new classification entities have to be added along with their relationships. If the nature of a classification changes (for example, from a one-to-many (1:M) to a many-to-many (M:M) relationship), this also requires a data modeling change. These can have a negative impact in a dynamic environment. Finally, this pattern does not capture the interrelationships between the different classification types, although these may be added to the pattern, allowing the modeler to relate different classification types to each other.

Level 3 Classification Pattern

Many enterprises need a very flexible approach for managing and capturing their classifications, because new classifications for an entity may appear over time and the nature of the relationships from the classification to an entity may change over time. For example, the cardinality or optionality may change. In the previous pattern you saw specific classification entities. This pattern generalizes the specific classification entities into a single categorization entity that includes all the various types and allows for much more flexibility and adaptability when changes occur.

Why Do We Need This Pattern?

Many enterprises add and change classifications as they change how they manage and view their businesses. Many enterprises compete by using alternative analytical approaches to their advantage. To do this they need to analyze their data in new and innovative ways.(10) For example, some enterprises' marketing and sales divisions

continuously think of new ways to categorize products and services in order to better understand their customers and the marketplace. Many enterprises have needs for many diverse ways to classify entities. For example, on one Universal Data Model assignment, we were working with a car distributorship, and there were dozens of ways that cars were classified (make, model, year, engine type, transmission type, and so on). Also, there was a strong likelihood that new classifications would be added in the future.

Enterprises may need an easy way to add, update, and delete different types of classifications without having to add new entities, attributes, or relationships. For most environments, it is very helpful to have minimum impact on existing databases, analytics, programs, and reports when changing categorizations. The Level 3 Classification Pattern provides this flexibility.

How Does This Pattern Work?

Instead of providing a separate entity for each new “type,” the Level 3 Classification Pattern has a single “category” entity that allows for any number of classifications of an entity and a “category type” that maintains the kinds of classifications that may exist or that may emerge over time. For example, an entity such as PRODUCT may have many different ways to classify it, or in other words different “category types,” that is, by “product type” (“Processors,” “Business Application Software,” or “Mouse Pads”); by “product family,” (“Disk Drives,” “Carrying Cases,” and so on); by “product line,” (“Home Use,” “Commercial Use,” and so on); or by many other types of classifications that may exist or may appear over time.

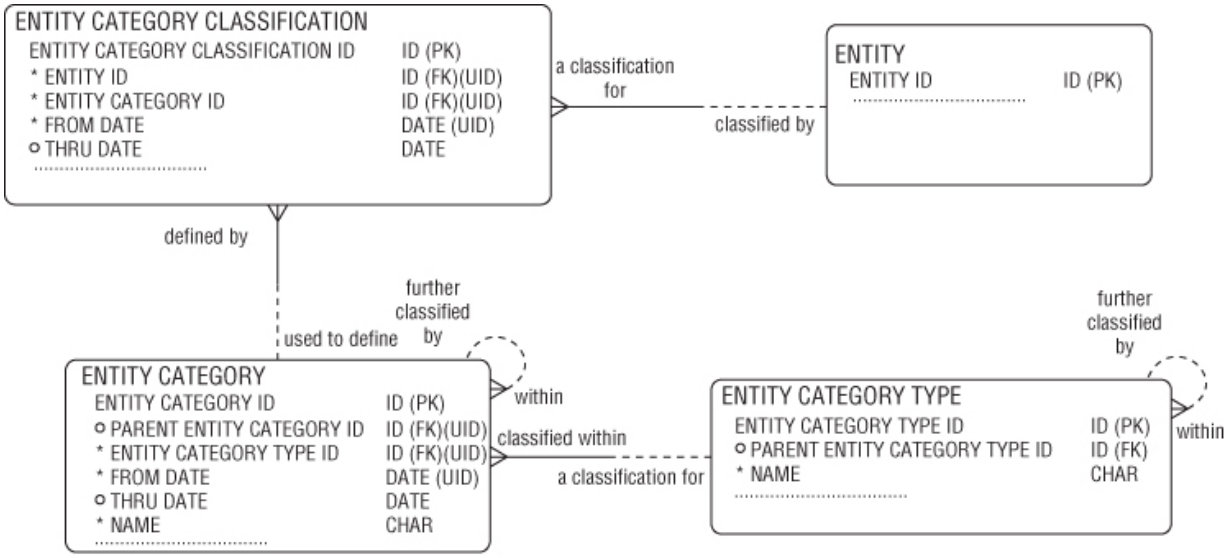
These categories may also roll up or roll down into various higher or lower-level categories, and thus this pattern provides the capability to have hierarchies or aggregations

of categories and of category types. For example, the product category of “Hardware” may be further classified into “Storage Devices,” and “Input/Output Devices,” and the product category of “Software” may be further classified into “Business Application Software” and “Gaming Software.” Many businesses like to evaluate their products at higher levels and then drill into the detail to support analytical needs. Thus, this pattern needs to support the rolling up and down of higher and lower-level categories. Likewise, category types may also have hierarchical structures (or aggregation structures), for example, maintaining that the category type of “Product type” may be further classified into “Product subcategory.”

In the previous pattern, each of the different categories was captured specifically as an entity in its own right, such as ENTITY TYPE 1, ENTITY TYPE 2, and ENTITY TYPE 3, as was seen in Figure 5.3. In Figure 5.8 these different classification types are captured as instances of ENTITY CATEGORY TYPE. In other words, ENTITY CATEGORY TYPE contains the instances “Entity Type 1,” “Entity Type 2,” and “Entity Type 3” (and any entity types that may occur in the future) or more specifically in our example, “Product Type,” “Product Family,” and “Product Line.” Each of the different instances of those classification types is then captured as an instance of ENTITY CATEGORY. For example, the PRODUCT CATEGORY TYPE instance of “Product Type” may be **a classification type** for the PRODUCT CATEGORY instances of “Hardware,” “Software,” “Storage Devices,” “Gaming Software,” and other instances of product types that are shown in Figure 5.5. This provides the ability to dynamically add additional types of classification types and classifications if and when new needs arise. For example, if Euro-Electronics discovered a new classification category called “Product Market Segment” with values “Mass Market” and “Luxury Market,” it would add “Product Market

Segment” as a PRODUCT CATEGORY TYPE and add “Mass Market” and “Luxury Market” as instances of PRODUCT CATEGORY.

Figure 5.8 Level 3 Classification Pattern



ENTITY CATEGORY CLASSIFICATION cross-references an ENTITY to an ENTITY CATEGORY. It captures instances of an ENTITY (the data you are trying to categorize, such as a particular product) and an instance of an associated ENTITY CATEGORY (a classification such as “Hardware”). An instance of an ENTITY (for example, a product of “Save Disk 2000”) may be classified in many different ways (for example, “Storage Device,” “Disk Drive,” and “Home Use”) and an ENTITY CATEGORY may be used to classify many different instances of the ENTITY (for example, the category “Hardware” may apply to many different products). The ENTITY CATEGORY CLASSIFICATION resolves this many-to-many relationship between ENTITY and ENTITY CATEGORY. The model illustrates that “each ENTITY *may be* **classified by** one or more ENTITY CATEGORY CLASSIFICATION(s) and each ENTITY CATEGORY *may be* **used to define** one or more ENTITY CATEGORY CLASSIFICATION(s).”

Notice that ENTITY CATEGORY(s) gets classified by ENTITY CATEGORY TYPE. So, why do you need to maintain the type

of classification? In other words, what is the difference between ENTITY CATEGORY(s) and ENTITY CATEGORY TYPE(s)? ENTITY CATEGORY(s) are the different classifications that an ENTITY may have, and ENTITY CATEGORY TYPE(s) are the labels or the names describing what type of classification this is. For example, a WORK EFFORT (an ENTITY) may be classified as a “Project” or “Task.” “Project” and “Task” are WORK EFFORT CATEGORY(s) (ENTITY CATEGORY(s)). These are within a classification type (one way to classify work efforts) of “Work Effort Type,” which is the WORK EFFORT CATEGORY TYPE (the ENTITY CATEGORY TYPE). There could be other WORK EFFORT CATEGORY TYPE instances, such as “Work Effort Purpose Type,” which represents a different way to classify work efforts by its purpose such as whether it is a “Research” work effort or a “Manufacturing” work effort. “Research” and “Manufacturing” would be instances of WORK EFFORT CATEGORY(s).

A useful way to imagine the difference between ENTITY CATEGORY and ENTITY CATEGORY TYPE(s) is to visualize a report. The ENTITY CATEGORY(s) could be the columns of a report and the ENTITY CATEGORY TYPE(s) could be the heading above those columns (also known as the break points in the report). The ENTITY is therefore directly classified by the ENTITY CATEGORY (the columns of the report) and not the ENTITY CATEGORY TYPE. For instance, there may be a PRODUCT CATEGORY TYPE of “Product Family” and the columns for this section of the report may be the PRODUCT CATEGORY(s) of “Disk Drives,” Carrying Cases,” “Computer Memory,” “Desktop Computers,” and “Laptop Computers.” If the rows are for each product and there are values within the report for the number of units of each product sold for the various columns, this illustrates that the PRODUCT CATEGORY(s) are what is used to classify products, not the PRODUCT CATEGORY TYPE(s). Thus, the

ENTITY CATEGORY TYPE represents the way that something is classified, and the ENTITY CATEGORY maintains the possible values that directly classify something. The PRODUCT may be classified as a PRODUCT CATEGORY of "Laptop Computer," and the way that this is classified is within the PRODUCT CATEGORY TYPE of "Product Family." Thus, you would *not* say that the product is classified as a "Product Family" but rather as a "Laptop Computer" with the category type of "Product Family."

Another way to visualize it would be to imagine WORK EFFORT "Data warehouse project #123" being classified by a "Work effort type" of "Project," and also classified by a "Work effort purpose type" of "Development Effort." You would never say that "Data warehouse project #123" is classified as a "Work effort purpose type" or a "Work effort type" (both of these are WORK EFFORT CATEGORY TYPE(s)). But you could say that "Data warehouse project #123" is a "Project" (WORK EFFORT CATEGORY) and "Project" is a "Work effort type" (WORK EFFORT CATEGORY TYPE). You could also say the "Data warehouse project #123" is a "Development effort" (WORK EFFORT CATEGORY) and a "Development effort" is a "Work effort purpose type" (WORK EFFORT CATEGORY TYPE). Thus, the ENTITY CATEGORY is related to an ENTITY (through the ENTITY CATEGORY CLASSIFICATION) and the ENTITY CATEGORY TYPE contains the type of classification you are maintaining.

As you saw in the previous section (in Figure 5.5 and Table 5.2), category types may have their own structure. ENTITY CATEGORY has a recursive association around it to support this need. ENTITY CATEGORY TYPE may also have a recursive structure around it to support a hierarchy of ENTITY CATEGORY TYPE(s).

Why would you need a hierarchy of ENTITY CATEGORY TYPE(s)? Sometimes it is important to show the corresponding "labels," or in other words, the category

types, for each of the category levels. It is often true that ENTITY CATEGORY TYPE(s) don't have a hierarchical structure, but sometimes they do. For example, if you wish to create a hierarchy of ENTITY CATEGORY(s) instances, you may want to look up to see if a hierarchy of ENTITY CATEGORY TYPE(s) existed that formed a basis to validate which ENTITY CATEGORY(s) can roll up to each other. In other words the ENTITY CATEGORY hierarchy (or aggregation) may need to mirror an ENTITY CATEGORY TYPE hierarchy (or aggregation). For example, if the PRODUCT CATEGORY TYPE of “Product Type” was further classified into “Product Subcategory,” there could be a business rule stating that the recursive relationship of ENTITY CATEGORY instances must each relate to appropriate ENTITY CATEGORY TYPE instances that also roll up in a similar way. An example would be that you want to check if it is valid to have a recursive relationship on PRODUCT CATEGORY and relate an instance of “Cases” rolling up to “Accessories.” “Accessories” may be related to a PRODUCT CATEGORY TYPE of “Product Type,” and “Cases” may be related to a PRODUCT CATEGORY TYPE of “Product Subcategory.” Finally, the recursion on PRODUCT CATEGORY TYPE shows that the instance of “Product Type” is a parent of the instance of “Product Subcategory**.”** Thus, you can conclude that it is valid to roll up a product category that is of type “Product Subcategory” (for example, “Cases”) to a product category that is of type “Product Type” (for example “Accessories”). This can help to prevent rolling up categories that are of completely different types of classifications, for example, rolling up “Mouse Pads” (a product type) to “Commercial Use” (a product line), thus mixing the proverbial apples and oranges. This can also be used to make sure that a lower-level category appropriately rolls up to the next level and does not skip a level.

Note

Another reason for the hierarchy of instances in ENTITY CATEGORY TYPE may be for creating labels that are useful in reporting or queries: for example, show me the “Product Types” and their associated “Product Subcategories.”

Although the hierarchy (or aggregation) of ENTITY CATEGORY TYPE(s) can be used as data to validate appropriate ENTITY CATEGORY(s), the data model does not specifically enforce a rule stating that “an ENTITY CATEGORY hierarchy must follow a specific hierarchy in ENTITY CATEGORY TYPE.” This would have to be achieved by a business rule and could be implemented by some external code, or by using the Business Rule Pattern described in Chapter 8 of this book. What the data model does do is store the parameters that show what ENTITY CATEGORY TYPE(s) are supposed to roll up to each other, which can guide what ENTITY CATEGORY(s) are supposed to roll up to each other. In other words you can verify a hierarchy (or aggregation) of ENTITY CATEGORY(s) by checking if their related ENTITY CATEGORY TYPE(s) roll up also.

Note

The two recursive relationships in Figure 5.8 are one-to-many relationships. The one-to-many relationships support hierarchies and aggregations where the child instance may only be related to one parent instance. Many-to-many hierarchies, aggregations, and peer-to-peer relationships may be needed and if they are, the next section of the chapter will enhance this pattern to accommodate this need (see Figure 5.12). Another alternative is to substitute the one-to-many recursions with any of the recursive patterns that exist in Chapter 4.

To further illustrate this pattern, we continue with the scenario that we described in the previous section. The CIO of Euro-Electronics has stated that one of the biggest IT

issues the company has is that sales and marketing people keep changing the way they want to slice and dice product information. For example, they continually come up with new ways to analyze product information, such as various product groupings, subgroupings, usage, types of materials, and so on. Each new type of category that they add, change, or make obsolete may need new (or changed) entities and eventually new (or changed) tables. This has become a headache for the overstretched programmers in the IT department. Sales and marketing have threatened to employ their own in-house programmers to get their reports up and running. The CIO wants to have a flexible architecture in place to alleviate the pressure on his programmers so that he can meet the needs of sales and marketing. Based on this need and using the Level 3 Pattern, the data professional produced Figure 5.9.

Figure 5.9 Example of using a Level 3 Classification Pattern

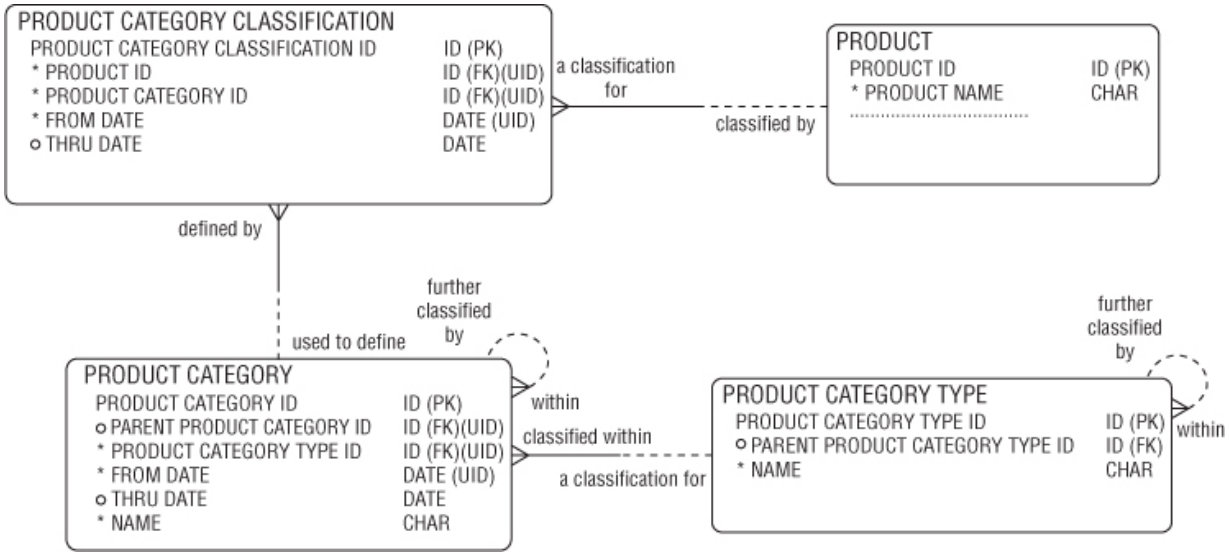


Figure 5.9 captures the same type of data that Figure 5.4 does, however, in a much more flexible manner. In Figure 5.4 you saw how PRODUCT had three different classification types: PRODUCT LINE, PRODUCT TYPE, and PRODUCT FAMILY. Each of these classification types captured the different ways Euro-Electronics classified a PRODUCT. For

example, the same product may be classified as a PRODUCT TYPE of “Storage Devices,” a PRODUCT LINE of “Home Use,” and a PRODUCT FAMILY of “Disk Drives.” In this pattern the different types of classifications (PRODUCT LINE, PRODUCT FAMILY, and PRODUCT TYPE) are captured as instances of PRODUCT CATEGORY TYPE.

In Figure 5.9 you see that each of the different PRODUCT(s) may get classified directly into many different PRODUCT CATEGORY(s) via PRODUCT CATEGORY CLASSIFICATION. And each of the different PRODUCT CATEGORY(s) represents a classification that is within the context of a parti cular type of classification, namely within a PRODUCT CATEGORY TYPE.

For example, in Table 5.4 you see in the first four rows that “Save Disk 2000” is classified into four different PRODUCT CATEGORY(s), of different category types, with **name(s)**: “Storage Devices” for the PRODUCT CATEGORY TYPE of “Product Type,” “Disk Drives” for the PRODUCT CATEGORY TYPE of “Product Family,” and “Home Use” and “Commercial Use” for the PRODUCT CATEGORY TYPE of “Product Line.” You also see in the next four rows that the “Carry All Case” PRODUCT gets classified into “Accessory,” “Carrying Cases,” “Commercial Use,” and “Home Use” PRODUCT CATEGORY(s), and each of these corresponds to various category types.

Table 5.4 Product Category and Category Type

PRODUCT. PRODUCT NAME	PRODUCT CATEGORY CLASSIFICATION. FROM DATE	PRODUCT CATEGORY CLASSIFICATION. THRU DATE	PRODUCT CATEGORY. NAME	PRODUCT CATEGORY. NAME(PARENT)	PRODUCT CATEGORY TYPE. NAME
Save Disk 2000	Jan. 1, 2008		Storage Devices	Hardware	Product Type
Save Disk 2000	Jan. 1, 2008		Disk Drives		Product Family
Save Disk 2000	Jan. 1, 2008		Home Use		Product Line
Save Disk 2000	Jan. 1, 2008		Commercial Use		Product Line
Carry All Case	Mar. 1, 2009		Cases	Accessory	Product Type
Carry All Case	Mar. 1, 2009		Carrying Cases		Product Family
Carry All Case	Mar. 4, 2008	Feb. 3, 2009	Commercial Use		Product Line
Carry All Case	Feb. 4, 2009		Home Use		Product Line
Save Disk 2000	Dec. 5, 2009		Portable Device		Product Group
Carry All Case	Feb. 5, 2010		Semi-Portable Device		Product Group

The marketing and sales departments wanted the ability to easily change the categorizations of PRODUCT(s). For example, marketing realized that the “Carry All Case” was actually being sold and used mainly for noncommercial customers and therefore wanted to classify in the PRODUCT LINE of “Home Use.” In other words they no longer classified the “Carry All Case” in the PRODUCT LINE of “Commercial Use” after Feb. 3, 2009. To support this business functionality, PRODUCT CATEGORY CLASSIFICATION has **from date** and **thru date** attributes to maintain when a specific product was first classified into a particular PRODUCT CATEGORY and when it is no longer classified into that PRODUCT CATEGORY. In Table 5.4, for example, the “Carry All Case” was in the “Commercial Use” PRODUCT CATEGORY (with PRODUCT CATEGORY TYPE of “Product Line”) until and including the **thru date** of “Feb. 3, 2009,” and then on Feb. 4, 2009, it was considered to be in the “Home Use” product line.

The marketing and sales departments of Euro-Electronics also wanted the flexibility to “roll up” the different PRODUCT CATEGORY(s) into other PRODUCT CATEGORY(s). For example, in Table 5.4 you see that the “Save Disk 2000” is classified as “Storage Devices,” which is within the parent PRODUCT CATEGORY of “Hardware,” which is classified within the PRODUCT CATEGORY TYPE of “Product Type” (“Storage Devices” would be within the PRODUCT CATEGORY TYPE of “Product Subcategory”). Likewise, the “Carry All Case” product is classified into “Cases,” which is within the parent PRODUCT CATEGORY of “Accessory.” Having multiple levels of categorization is a very normal occurrence in business. Often managers approach analytics from many different perspectives. The managers may want to ‘drill down or up (or across)’ to the ‘detail/summary’ depending on their requirements. In this example, sales and marketing may ask “what are the annual sales for

accessories?" and then specifically for "Cases" and the data model supports this type of query.

The key distinction of this pattern is that it allows the addition of new types of categories much more easily by just adding additional instances and without needing to change the data model. The last two rows in Table 5.4 show that marketing and sales wanted an additional way to categorize products, namely by a new classification type called "Product Group," and this pattern accommodates this new category type and any other new category types that may emerge without any need to change the model. Table 5.4 shows that the "Save Disk 2000" can be classified into the new PRODUCT GROUP category as a "Portable Device" and the "Carry All Case" is classified as a "Semi-Portable Device." Thus, the data model (and any database structures that are based on this model) supports adding new ways of categorizing the products as additional needs appear over time.

This pattern also provides the capability to add additional hierarchies or aggregations of categories. For example, you can see from Figure 5.10 and Table 5.5 that PRODUCT CATEGORY(s) are aggregated(9) into Profit and Loss Reporting Categories of "Computers," and "Computer-Related Equipment," as well as "Government Business," and "Non-Government Business."

Figure 5.10 Profit and loss categories

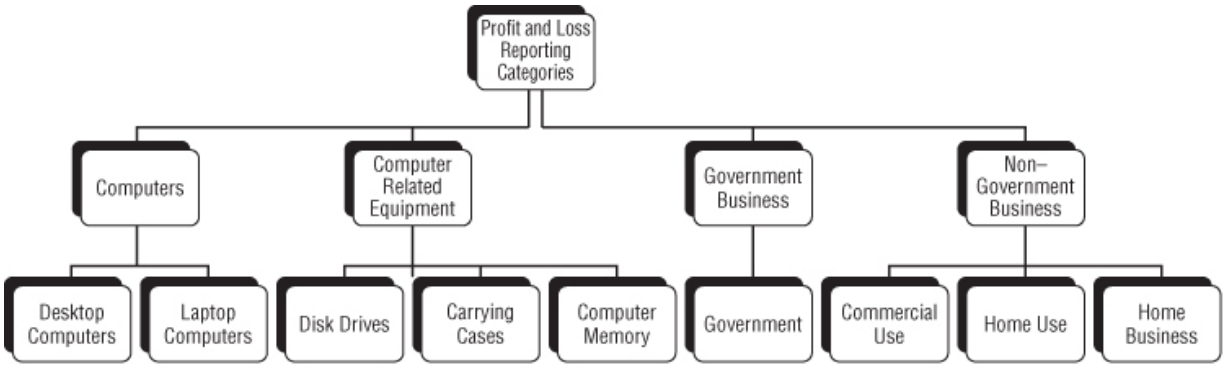


Table 5.5 Examples of Level 3 Classification Pattern, PRODUCT CATEGORY(s) Aggregation

PRODUCT CATEGORY. PRODUCT CATEGORY ID (PARENT NAME)	PRODUCT CATEGORY. PRODUCT CATEGORY ID (CHILD NAME)	PRODUCT CATEGORY TYPE. NAME*	PRODUCT CATEGORY. FROM DATE	PRODUCT CATEGORY. THRU DATE
	20 (Computer-Related Equipment)	Profit and Loss Reporting Category	Apr. 1, 2008	
20 (Computer-Related Equipment)	3 (Disk Drives)	Product Family	Jan. 1, 2005	
20 (Computer-Related Equipment)	4 (Carrying Case)	Product Family	Jan. 1, 2005	
20 (Computer-Related Equipment)	7 (Computer Memory)	Product Family	Jan. 1, 2005	
	30 (Computers)	Profit and Loss Reporting Category	Apr. 1, 2008	
30 (Computers)	9 (Desktop Computers)	Product Family	Feb. 4, 2005	
30 (Computer)	11 (Laptop Computers)	Product Family	Mar. 3, 2005	
	40 (Non-Government Business)	Profit and Loss Reporting Category	Apr. 1, 2008	
40 (Non-Government Business)	190 (Home Use)	Product Line	Jan. 1, 2005	
40 (Non-Government Business)	199 (Commercial Use)	Product Line	Jan. 1, 2005	
40 (Non-Government Business)	230 (Home Business)	Product Line	Jan. 1, 2005	
	50 (Government Business)	Profit and Loss Reporting Category	Apr. 1, 2008	
50 (Government Business)	240 (Government)	Product Line	Jan. 1, 2005	

*This is the PRODUCT CATEGORY TYPE **name** for the "child," or in other words, for the second column, because both the "parent" category and the "child" category could be related to a different PRODUCT CATEGORY TYPE.

Why did Euro-Electronics want to do this? Senior managers of Euro-Electronics wanted to capture profit and loss information based on how much was earned from "Computers" sales versus "Computer-Related Equipment"

sales, and they also wanted to find out earnings based on “Government Business” versus “Non-Government Business.” Figure 5.10 shows how this can be accomplished by combining some of the categories from “Product Lines” and “Product Families” and creating new categories for this need. Thus, by simply adding new categories and then creating aggregations, the team was able to give management exactly the classifications they needed without changing the data model. Table 5.5 shows how you would capture these different “Profit and Loss Reporting” categories and their relationships to other categories using the model in Figure 5.9. You can see in the table that the new categories of “Computer-Related Equipment,” “Computers,” “Government Business,” and “Non-Government Business” were added on April 1, 2008 and then related to other existing categories to provide the capabilities they wanted without changing the data model.

What this pattern enables you to do is add categories that may never be associated directly to an instance of PRODUCT but aggregates the categories that are directly related to PRODUCTS into useful classifications for reporting and analysis. In other words, by creating these aggregations that are of type “Profit and Loss Reporting Category,” the enterprise can provide a convenient way to support general reporting and analytics about its products, as needed. This means that Euro-Electronics can answer questions such as, “How much profit/loss did we make on computers versus computer-related equipment?” or “What was the profit/loss on products sold to the government versus non-government business?” PRODUCTS(s) can be classified by these categories as well as other categories such as “Product Family” and “Product Line,” and therefore the enterprise can also answer questions like “How much profit/loss did we make from computers sold that are designed for home use?” This question can be answered by aggregating the

profit/loss from “computers” (which aggregates the “Laptop Computer” and “Desktop Computer” categories) and then filtering this by products that were classified in the PRODUCT CATEGORY CLASSIFICATION of “Home Use.”

> **Note**
>
> While some additional hierarchies and aggregations may be easily added within this pattern, a key limitation to this is that there may only be one parent for each child instance. This limits the variations in hierarchies and aggregations that can be maintained with this pattern. For example, if Euro-Electronics wanted to add the “Product type” of “Processors” as another child of “Computer-Related Equipment,” this would not be possible because “Processors” already has a parent instance of “Hardware” and according to the data model there can only be one parent. This capability is addressed in the next pattern, the Level 3 Classification Pattern with Rollups and Schemes.

Notice in Table 5.5 that we were very careful not to mix categories of “Product Line” with categories of “Product Family” into the same PRODUCT CATEGORY data model structure. For example, we did not mix “Home Business” (a “Product Line”) with “Laptop Computers” (a “Product Family”) within the same PRODUCT CATEGORY. Because an instance of PRODUCT may be classified in both ways, by “Product Line” and by “Product Family,” we could double count them if we mixed “Product Line” and “Product Family” into a single PRODUCT CATEGORY.

How did we enforce this? Before creating these aggregations of PRODUCT CATEGORY(s) seen in Figure 5.10 and Table 5.5, a “template” structure was created as seen in Figure 5.11 and Table 5.6. In this structure you see that “Profit and Loss Reporting Category” may be an aggregation of either “Product Line” or “Product Family(s)”. In other words, Euro-Electronics used the aggregation seen in Figure 5.11 as a template to ensure the validity of the profit and loss categories as seen in Figure 5.10 and Table 5.5.

Figure 5.11 The aggregation of Financial Reporting Category Types

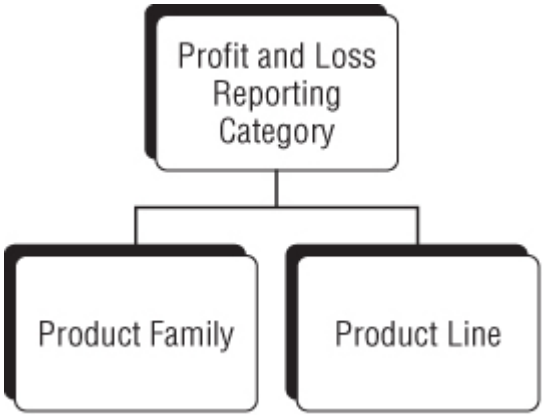


Table 5.6 Examples of Level 3 Classification Pattern, PRODUCT CATEGORY TYPE(s) Aggregation

PRODUCT CATEGORY TYPE.PRODUCT CATEGORY TYPE ID (CHILD)	PRODUCT CATEGORY TYPE. NAME (CHILD)	PRODUCT CATEGORY TYPE.PRODUCT CATEGORY TYPE ID (PARENT)	PRODUCT CATEGORY TYPE. NAME (PARENT)
55	Product Line	60	Profit and Loss Reporting Category
56	Product Family	60	Profit and Loss Reporting Category

Why do this? The flexibility of this pattern allows Euro-Electronics to create valid hierarchies or aggregations of PRODUCT CATEGORY(s). On the one hand, it allows the freedom to mix and combine various categories together for the various types of analysis or reporting it needs. But some control may be needed. Table 5.6 shows that only product lines and product families may be aggregated together to form a "Profit and Loss Reporting Category" (and not "Product Types"). Thus, when the enterprise wants to add a new "Profit and Loss Reporting Category" to Figure 5.10, it first checks the template as seen in Figure 5.11 to make sure that the category hierarchy is allowable. In other words, the current template says that only product lines or

product families can be aggregated to create a new profit and loss reporting category.

When Should This Pattern Be Used?

We use this data model pattern when:

- **When an enterprise requires a flexible data model that does not have to be changed when:**
  - New types of classifications are needed or discovered
  - The nature of the classification type changes in relation to an entity (such as the cardinality or optionality of a classification in relation to the entity)
  - Old classification types need to be deleted
- **When an enterprise has many ways to classify an entity:** Some entities (such as parties, products, assets, and work efforts) may have a great number of classification types. The same entity may be classified in many different ways depending on many factors such as how various external data providers classify it, how it is classified in different geographic areas, and how different parts of the enterprise classify it.
- **There is a need to have a common model to manage the various categories and category types of an entity:** This pattern can serve as a common method to classify entities, and thus, the enterprise can consistently model categorizations for various entities. Additionally, there could be common routines for modeling categorizations because the data models (and ensuing database designs) could be very similar.
- **There is a need to combine categories in order to provide more powerful analytic capabilities:** Combining similar categories sometimes allows more

powerful analytics. For example, by combining all the various categories of parties into a PARTY CATEGORY entity and also combining the various categories of products into a PRODUCT CATEGORY entity, you can then create a cross-reference entity between PARTY CATEGORY and PRODUCT CATEGORY called MARKET INTEREST that maintains which types of PARTY(s) are interested in which types of PRODUCT(s) (for example, “high-income” parties may be more interested in “high-end, deluxe” products).

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **More difficult to understand:** This pattern is much more generalized and is harder to understand and communicate than the Level 2 Classification Pattern. Because this model is very generalized, it is hard to effectively use this pattern on its own to define the scope or to gather and validate data requirements such as what types of classifications are needed.
- **Does not enforce business rules directly in the data model:** This model does not capture specific business rules such as the cardinality or optionality of the relationship from the classification to the related entity. The model says only that the related entity can be classified into any number of categories, the entity may be classified into different instances of a category, and each category and category type may have any number of levels of hierarchy or aggregation. This allows data modelers a lazy way to ‘catch all’ the different categories that could exist for an entity without having to examine which categories are needed and what the requirements are for each categorization.

Different categories may have different rules, relationships, or attributes associated with them. These need to be analyzed and understood even if one models using this pattern.

- **There are no explicit business rules maintained in the data model about how the categories roll up:** Although the model allows you to store data about the ENTITY CATEGORY rollups and the ENTITY CATEGORY TYPE rollups, there is not an explicit rule the data model enforces to ensure that the ENTITY CATEGORY recursion rolls up the same way that the ENTITY CATEGORY TYPE recursion rolls up. So it would be possible to maintain that a “Product Subcategory” rolls up to a “Product Type” in the PRODUCT CATEGORY TYPE recursion but have a “Product Type” roll up to the “Product Family” in the PRODUCT CATEGORY recursion. This means that if business rules and processes are not defined around this relationship the two hierarchies could go “out of sync.” (This could also be addressed by using the Business Rule Pattern described in Chapter 8).
- **The ways that categories and category types may be rolled up is limited:** Since this pattern only provides a one-to-many recursive relationships for categories and category types, the various types of hierarchies and aggregations is limited. Also, various schemes may need to need to be maintained and this pattern does not address this need.
- **May not need this pattern if there are only a few ‘types’ or if the number of ‘types’ is stable:** This pattern may be overkill if there are only one or two different ways that are needed to classify the entity, or if the number of ways to classify the entity is very stable and unchanging.
- **The pattern does not maintain different attributes or specific types of relationships for different**

TYPE(s)/CATEGORY(s): A specific type of category may have an attribute or relationship that is specific to that category type. However, you may choose to address this by creating subtypes of ENTITY CATEGORY and ENTITY CATEGORY TYPE.

Synopsis

There are many reasons why this pattern is significant; however, there are two key reasons. First, it provides the flexibility to add or change new types of categories without having to change the data model. Second, it provides a single standard way to deal with all classifications for an entity in a single consistent fashion.

In Table 5.4 you saw that different PRODUCT(s) could be classified into various types of classifications. For example, they could be classified into product lines of “Home Use” or “Commercial Use,” and/or they could be classified into the product family of “Disk Drives” and/or “Laptop Computers,” as well as classified into any number of other types of classifications including new ones that may emerge over time. With this pattern, the enterprise can add or delete new ENTITY CATEGORY(s) and ENTITY CATEGORY TYPE(s) without changing the model, leading to a very flexible implementation solution.

Because of the generalized nature of the pattern, we would not use it as a part of a statement of scope for nontechnical audiences, or as a way to gather and validate data requirements. In this pattern, the different category types are captured as instances, not as entities and attributes. We would also warn data professionals not to be lazy when using this pattern. This may be a ‘catch all’ for different types of categories, but that isn't an excuse for not doing proper analysis to understand what categories are needed and the nature of the categorizations. Because you capture all of the classifications in the same way, you lose

the ability to capture specific attributes and relationships that some classification types may have. You could address this by having subtypes of ENTITY CATEGORY and ENTITY CATEGORY TYPE.

Level 3 Classification Pattern with Rollups and Schemes

There may be schemes or sets of classifications that are designated by internal or external bodies. For example, a financial services data provider, such as Bloomberg or Reuters, provides classification schemes regarding how to classify financial securities. Another example would be governments that provide ways to classify industry types such as the Standard Industrial Classification (SIC) scheme, and it is common that different governments provide different classification schemes.

Furthermore, classifications are often related to each other in more than one way, and there is often a many-to-many recursive relationship for classifications as well as classification types.(9) For example, there may be a product classification of “Office product” that has subclasses of “Computer parts,” “Office supplies,” and “Office machines.” Another classification of “Retail product” may also be further classified into some of the same categories of “Computer parts” and “Office supplies” but not “Office machines.” Thus, “Computer parts” and “Office supplies” roll up to different parent classifications, namely “Office product” and “Retail product.” These multiple rollups (or breakdowns) of classifications are the paths that enterprises use to drill into, out of, and across the details of their data within their enterprise and thus are critically important.

Why Do We Need This Pattern?

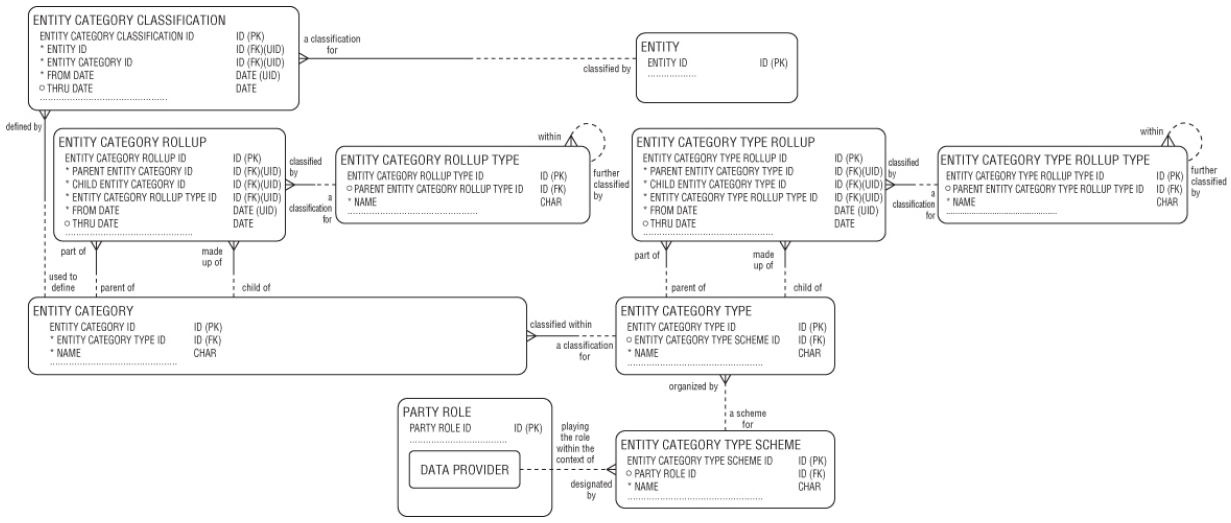
The pattern within this section supports the management of standard schemes of classifying data. The pattern supports the relationship of the originator of a classification scheme with those classification types that are members of that scheme. For example, a financial institution may need to know that a classification of financial securities came from a particular data provider, or that a certain rating classification came from a particular rating agency.

This pattern also supports the need to create many-to-many recursive associations between categories and between category types. This pattern also supports the ability to create “template” classification structures that can help in conforming classification data.

How Does This Pattern Work?

In Figure 5.12 you see that “each ENTITY CATEGORY TYPE *may be* **organized by** one and only one ENTITY CATEGORY TYPE SCHEME and each ENTITY CATEGORY TYPE SCHEME *may be* **a scheme for** one or more ENTITY CATEGORY TYPE(s).” The ENTITY CATEGORY TYPE SCHEME can be defined as “a combination of elements … that are connected, adjusted, and integrated by design”(11) for the classification of ENTITY CATEGORY TYPE(s).

Figure 5.12 Level 3 Classification Pattern with Rollups and Schemes



An ENTITY CATEGORY TYPE SCHEME captures the **name** of the scheme, or in other words what the scheme is commonly referred to as, for example, the “Standard Industry Classification (SIC)” scheme that is used to classify an organization by the industries that it is in. The scheme structure and all the elements of the scheme may not be explicitly captured or even used in this pattern. The model associates a certain scheme with one or more ENTITY CATEGORY TYPE(s). “Each ENTITY CATEGORY TYPE SCHEME *may be* **designated by** one and only one (internal or external) DATA PROVIDER.” For example, an enterprise-wide scheme of general ledger account category types may be provided by the financial accounting department of a company, and this is an internal scheme. Another example of a scheme that is external is a set of rating category types defined and provided by Standard & Poor's for financial instruments and companies. The financial accounting department and Standard & Poor's are the PARTY(s) who play the PARTY ROLE of DATA PROVIDER.(12) It is often crucial for enterprises to know the source of a categorization scheme, especially if it is sharing data with other enterprises. These schemes may provide a common taxonomy where different enterprises can share and compare data.

Note

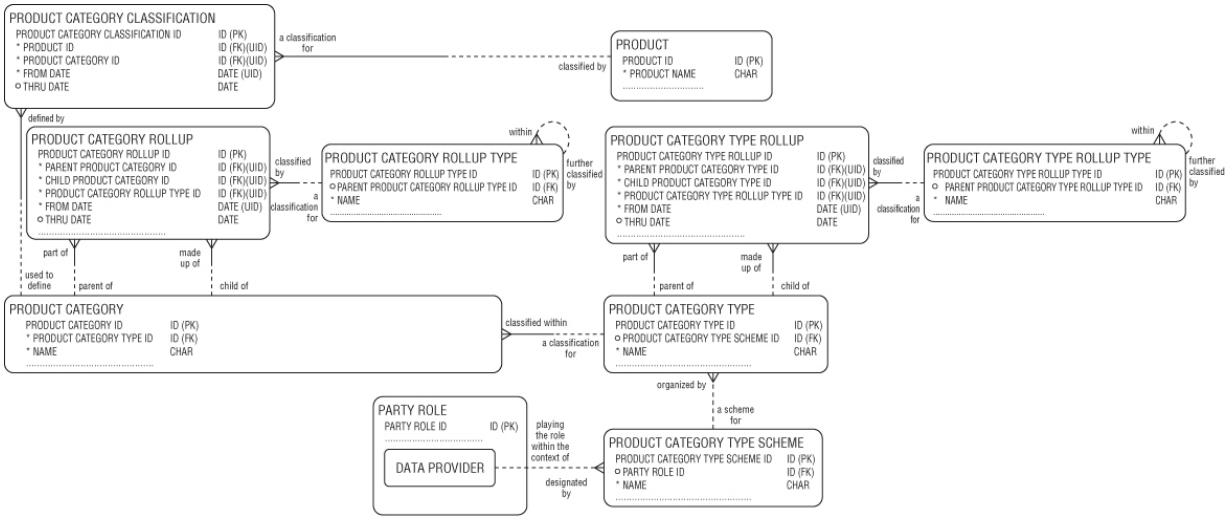
Two different ENTITY CATEGORY TYPE(s) may have the same name, and look like the same ENTITY CATEGORY TYPE, just repeated. But this is not always the case if the ENTITY CATEGORY TYPE is provided by different DATA PROVIDER(s) and is within a different scheme. For example, Reuters and Bloomberg may both provide a fixed income securities scheme with a category named “Government Bond Types.” Are they the same SECURITY CATEGORY TYPE? Generally they are not even though their schemes may have very similar semantic meanings and they refer to the same set of government bonds. The schemes may even be substituted for each other but they are not exactly the same. You should not create, for example, one “Government Bond Classification” scheme that is related to two different data providers (Bloomberg and Reuters).

The other change to this pattern is that although Figure 5.12 has the same basic structure as the Level 3 Classification Pattern described in the previous section, ENTITY CATEGORY ROLLUP and ENTITY CATEGORY TYPE ROLLUP were substituted for the one-to-many recursive relationships around ENTITY CATEGORY and ENTITY CATEGORY TYPE. These entities support multiple different ways to rollup (or breakdown) the ENTITY CATEGORY and ENTITY CATEGORY TYPE. For example, you may want to classify PRODUCT CATEGORY(s) of “Hard disks,” “Mouse,” and “Keyboard” within the PRODUCT CATEGORY of “Computer parts” and also categorize them within the PRODUCT CATEGORY of “Hardware” as well. In other words “each ENTITY CATEGORY *may be* **the parent of** one or more ENTITY CATEGORY ROLLUP(s) as well as **the child of** one more ENTITY CATEGORY ROLLUP(s),” and the same is true regarding the relationship between ENTITY CATEGORY TYPE and ENTITY CATEGORY TYPE ROLLUP. This form of recursion also supports tracking of how classifications are maintained (and changed) over time via the **from date** and

thru date attributes in ENTITY CATEGORY ROLLUP and ENTITY CATEGORY TYPE ROLLUP.

The concepts in this pattern get further illustrated in Figure 5.13. Following on from the scenario described in the previous sections, Euro-Electronics has stated that it wishes to have a single master repository for product reference data. As part of that repository it wants to maintain the parties that provide the types of classifications and are therefore sources of PRODUCT CATEGORY TYPE(s), if these types of classification indeed came from a specific party.

Figure 5.13 Example of using a Level 3 Classification Pattern with Rollups and Schemes



As a part of this effort, the data professional created Figure 5.13. PRODUCT, PRODUCT CATEGORY CLASSIFICATION, PRODUCT CATEGORY, and PRODUCT CATEGORY TYPE all serve the same functions described in the previous section. The data professional added PRODUCT CATEGORY TYPE SCHEME and attached it to PARTY ROLE of DATA PROVIDER to maintain the party that designated the PRODUCT CATEGORY TYPE(s).

Additionally, the PRODUCT CATEGORY ROLLUP and PRODUCT CATEGORY TYPE ROLLUP allow many-to-many recursive relationships, and the PRODUCT CATEGORY

ROLLUP TYPE describes what type of rollup it is (for example, “Sales reporting rollup” versus “Service reporting rollup”). Thus, there could be different variations of PRODUCT CATEGORY ROLLUP(s) that have different PRODUCT CATEGORY ROLLUP TYPE(s). Similarly PRODUCT CATEGORY TYPE ROLLUP(s) show how the ‘labels’ roll up (for example, “Product Subcategory” rolls up to “Product type”), and PRODUCT CATEGORY TYPE ROLLUP TYPE maintains how types of classifications roll up to each other. For example, “Sales reporting rollup” and “Service reporting rollup” may both roll up to a PRODUCT CATEGORY TYPE ROLLUP TYPE of “Reporting rollup”. In this recursive structure, different areas of the enterprise may also have different ‘label’ hierarchies.

In Table 5.7 you see two external sources of PRODUCT CATEGORY TYPE(s), “The World Customs Organization” and the “U.S. Office of Management and Budget.” Each of these are instances of a PARTY (not shown in the diagram or table) and they each play the role of DATA PROVIDER, with party role ids of “34” and “36” respectively. These DATA PROVIDER(s) provide two very similar schemes, “Harmonized System (HS or HTS)” and “Schedule B.” These are used as export and import product classifications. What is interesting here is that there seem to be two very similar instances of the same PRODUCT CATEGORY TYPE, that is, “Computer Disks” and “Disks, magnetic, recorded (sound, video, or computer).” One refers to the scheme produced to export goods, “Harmonized System (HS or HTS)” and the other is for importing goods, “Schedule B.” This is often the case with different classification schemes. Schemes often classify the same type of thing in a different way using different semantics. This is where you must be careful; two different classifications within two different schemes may refer to the same type of data, but because they are two different schemes, there are two different PRODUCT CATEGORY TYPE(s), provided by two different data providers

that are different and unique.(13) Although it is technically possible to use the same PRODUCT CATEGORY or PRODUCT CATEGORY TYPE for two different schemes, we have found that it is very confusing and therefore dangerous because the category or category type may have different meanings and possibly different elements in each of them.

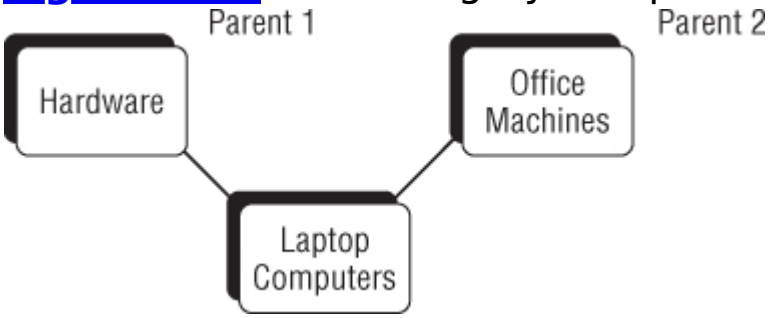
Table 5.7 Example of Using the Level 3 Classification Schemes

PARTY ROLE. PARTY ROLE ID (NAME OF PARTY FOR DATA PROVIDER ROLE)	SCHEME. SCHEME ID	SCHEME. NAME	PRODUCT CATEGORY TYPE. PRODUCT CATEGORY TYPE ID	PRODUCT CATEGORY TYPE.NAME
34 (The World Customs Organization)	67	Harmonized System	23	Computer Disks
36 (U.S. Office of Management and Budget)	69	Schedule B	24	Disks, magnetic, recorded (sound, video or computer)

At Euro-Electronics, the data professional also added PRODUCT CATEGORY ROLLUP and PRODUCT CATEGORY TYPE ROLLUP to capture many-to-many relationships that might exist between the different PRODUCT CATEGORY(s) and PRODUCT CATEGORY TYPE(s), respectively. For example, in Figure 5.14 you see the simple case where the "Laptop Computers" category may be a child of both the "Hardware" and "Office Machines." This is a very common occurrence in various enterprises. Types and categories can be members of different taxonomies. For example, in Euro-Electronics the sales department views the "Laptop Computers" category as rolling up to "Hardware," whereas the maintenance and support department rolls up "Laptop Computers" into "Office Machines." Thus, different departments may use the PRODUCT CATEGORY ROLLUP TYPE in order to define the way that their product categories are rolled up to each other. For example, there may be instances of "Sales department rollup" and "Support

department rollup” within PRODUCT CATEGORY ROLLUP TYPE to maintain the type of rollup.

Figure 5.14 Multicategory rollups for Laptop Computers



In Table 5.8 you see how Figure 5.14 would be captured in the Level 3 Classification Pattern with Rollups. “Laptop Computers” is a child of both “Hardware” and “Office Machines.” Thus, each child may have many parents and each parent may have many children and the many-to-many relationships between PRODUCT CATEGORY and PRODUCT CATEGORY ROLLUP accommodates this need. You can also see from the **from date** and **thru date** that each of these PRODUCT CATEGORY ROLLUP(s) was effective from “Jan. 1, 2009” and is still current because neither has a **thru date**.

Note

PRODUCT CATEGORY TYPE ROLLUP functions in a similar way as PRODUCT CATEGORY ROLLUP. Notice that we are using the Level 2 Expanded Recursive Pattern from Chapter 4, allowing more flexibility so that categories and category types may be rolled up to each other in many different ways. You could chose to substitute the Level 3 Recursive Pattern and substitute ASSOCIATION for ROLLUP, allowing categories to be associated with other categories in any possible way (for example, allowing substitutions, breakdowns, incompatibilities, and so on) for those enthusiasts that need an even more generalized pattern. For more on multiple hierarchies, aggregations, or peer-to-peer relationships, see Chapter 4.

Note

Another possible enhancement to this pattern would be the addition of an ENTITY CATEGORY ROLLUP RULE and ENTITY CATEGORY TYPE ROLLUP RULE to manage the behavior of the associations between ENTITY CATEGORY ROLLUP(s) and ENTITY CATEGORY TYPE ROLLUP(s) respectively. For example, you could exclude “Commercial Use” product family from being in the “Government” category. For more on this please refer to Chapter 4, Level 3 Recursive Pattern with Rules.

Table 5.8 Example of using the Level 3 Classification Pattern with Rollups

PRODUCT CATEGORY. PRODUCT CATEGORY ID	PRODUCT CATEGORY. PRODUCT CATEGORY ID	PRODUCT CATEGORY ROLLUP	PRODUCT CATEGORY ROLLUP.	PRODUCT CATEGORY ROLLUP.
1 (Hardware)	11 (Laptop Computers)	Sales Reporting Aggregation	Jan. 1, 2009	
70 (Office Machines)	11 (Laptop Computers)	Maintenance Reporting Aggregation	Jan. 1, 2009	

When Should This Pattern Be Used?

We use this data model pattern when:

- **When there is a need for a very comprehensive and flexible classification model:** This pattern should be used when there are many ways to classify an entity, when there is a need for a common way to model categories, when flexibility is needed, and when there is a need to combine categories for powerful analysis. These are the same reasons for using the previous pattern, the Level 3 Classification Pattern.
- **When there is a need to capture the source DATA PROVIDER for an ENTITY CATEGORY SCHEME that**

designates the ENTITY CATEGORY TYPE(s) and how they are associated to each other: This "scheme" aspect of this pattern is meant to be used when there is a need to capture the external or internal PARTY that supplies a set of classifications, or in other words, a classification scheme.

- **There is a need to capture many-to-many hierarchies, aggregations, or peer-to-peer relationships that may exist for classifications or classification types:** This pattern allows categories to roll up in different ways, for example, allowing different departments to have individual rollup structures.
- **The business requires a very flexible pattern that can withstand changing classification needs:** This pattern provides a great deal of flexibility in allowing any number of categories and/or category types to be added, allowing there to be any number of roll ups of categories and/or category types, and any number of schemes from data providers, all without changing the data model.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **It has the same weaknesses as the Level 3 Classification Pattern:** For example, it is not effective for using as a part of the statement of scope, or for gathering and validating data requirements for nontechnical audiences. It does not capture specific business rules, data or relationships for specific types of categories or types. It also allows data modelers to use "catch all" categories without doing detailed analysis.
- **It is very complex and difficult to understand:** It is a complex pattern that introduces many-to-many

recursive relationships and schemes. It can be difficult even for seasoned data professionals to understand and communicate this pattern to the business or other IT professionals.

- **It may be overkill if only one-to-many recursions are needed or if schemes are not needed:** Many times there is only a need to have one-to-many recursive relationships for categories and category types, so many-to-many relationships are not necessary. Also, there may not be a need to capture schemes and the related data providers.

Synopsis

In this pattern we introduced the idea of an ENTITY CATEGORY TYPE SCHEME. This is where a set of ENTITY CATEGORY TYPE(s) has been designated by an external or internal organization that we described as the DATA PROVIDER. This allows you to capture the source of a set of ENTITY CATEGORY TYPE(s) and how they are related to each other as prescribed by the DATA PROVIDER. When enterprises communicate with each other using common taxonomies, it is crucial to know what the source of that taxonomy is. This pattern also supports the need to create aggregations or hierarchies between categories and category types in a many-to-many fashion.

This is a complex pattern that can be difficult to understand. It is important that the data professional provides detailed explanations of this pattern along with worked examples and instance diagrams. Also, this pattern may not be suitable to be used in a scope statement for nontechnical audiences. It is used when there are comprehensive needs for many categories and category types with needs for flexible hierarchical or aggregation structures among categories, and that are used within schemes. This pattern can be the basis for a very flexible

database design that can withstand many changes to classifications over time.

Summary of Patterns

Table 5.9 contains a synopsis of all the patterns covered in this chapter.

Table 5.9 Synopsis of the Patterns

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 1 Classification Pattern	A specific way to model classifications generally used to help develop and validate data requirements with nontechnical audiences. All the classification information is contained inside the entity as attributes.	When you wish to have a simple model that can be used to gather and that captures the ways that an entity may be classified, as attributes of that entity. As a simple implementation, perhaps for prototypes; it could also be used for very static, unchanging entity classifications. To stimulate the analysis of the different classifications, to expose the classification nomenclature, expose problems with the classifications, and start the modeling in a straightforward manner.	As the basis for a long term database implementation because there are serious data redundancy and data integrity issues that could occur. When a flexible approach to classifications is needed. Any new classification would require new attributes. When there is more information that needs to be maintained about the classification such as description, short name, and so on. When there are multiple occurrences of the same classification for an entity (for example, a PARTY is classified into many industries). When there are complex business rules about the classifications that need to be maintained in the data model.	A simple way to show how the entity is classified for use with nontechnical audiences. A simple way to stimulate the analysis	Not a normalized data model and thus there are data redundancies that can cause data integrity issues if this is used as the basis for implementation. Very inflexible because it does not easily accommodate new classifications or when an entity needs more occurrences of the same type of classification (for example, having a party classified into many industries). It also has repeating classification attributes and therefore is very unstable since new classification attributes may be needed over time. Does not allow maintenance of other data about the classification. Impractical if an entity has many possible classifications of the same type (e.g., product may be classified into may product lines). Does not support complex business rules.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 2 Classification Pattern	A normalized, yet specific approach for capturing and managing classification data as entities and then relating them to the relevant entities. This pattern maintains classification data independently from (but related to) the data it classifies.	As a method of understanding data requirements or as a part of a statement of scope that is easy to understand. When there is a need to model and maintain data about each classification type. As a basis for implementing a very specific database design. When you have a static set of classification types that are well understood and their nature does not change (for example, from a 1:M to a M:M or from optional to mandatory), as well as when new types of classifications will not be needed in the future. When each of the classifications has its own set of attributes and/or relationships. When some classification types have a hierarchy or aggregation structure that needs to be modeled.	When an even simpler model is needed as a way to communicate with nontechnical audiences. When a more flexible approach is needed. In this pattern, new types of classification or changes in the relationships of classification type to their related entity (for example, if changing the cardinality or optionality) require data model changes. When the different classifications are highly interrelated and it is important to capture the relationships between classification types in a common, standard way. When you need a single standard way to handle types/categories. When an entity has a large number of ways to classify it.	This pattern represents a normalized data model and therefore does not lead to redundant data if used properly. This pattern allows for a strong semantic statement of data requirements. Classification data is independent from, but related to, the data it classifies. The data professional can create a hierarchy or aggregation of classifications. It is a strong statement of scope because it is relatively easy to understand and it can depict a lot of information about the nature of each classification entity. It captures specific business rules such as whether each classification is optional or mandatory as well as whether each classification is 1:M or M:M. You can capture specific attributes/relationships for each individual classification entity.	This pattern is not as simple to understand as the level 1 pattern. The pattern is not very flexible. Every time a new classification type is needed, then a new entity needs to be created, and every time the nature of the relationship from the classification to the entity changes, the data model needs to be changed. If there are many relationships between different classification types, this pattern can become very complicated and more difficult to manage, as opposed to having a single standard way to relate classification types to each other. It can limit analysis capabilities because it does not provide for a way to relate something to all classifications in a single entity (for example, to answer questions such as which types of customers are interested in which types of products). There is not a standard way to maintain types (for example, a single entity that includes all the possible types for the related entity).

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 3 Classification Pattern	A flexible approach for capturing and managing classification data where new types of classifications can be easily added as instances.	In an environment where new types of classifications often get added, changed, or updated. As a basis for a flexible implementation. When there are many classifications needed for an entity. When there is a need for a very common way to model various types of categorizations. When there is a need for more powerful analysis that can result from combining various types into a single 'category' entity. When classification type data has a hierarchy or aggregation structure for its classifications.	As an explicit statement of scope or as a tool in gathering and validating data requirements. When there are only a very few ways to categorize an entity and those ways are stable and unchanging. As a "catch all" for classifications. It is easy to use this pattern and not discover all the types of classifications that may be needed. When each type of classification has very specific attributes and relationships. When there are specific rules for each category, category type, or rollup such as when a specific classification type data has a hierarchy or aggregation structure and other data does not. When an entity has only a small number ways to classify it.	A very flexible pattern that does not have to be changed as new classifications are discovered. A pattern that provides a very standard way to classify any type of entity that has many classifications. A way to be able to manage all the types of classifications for an entity together and consistently. Provides powerful analytical capabilities such as cross-referencing which types of parties are interested in which types of products. Provides flexible hierarchies or aggregations of category and category types. Maintains data about which category types roll up to each other, which can be used to validate category structures.	This is a generalized pattern that may be hard to understand and communicate. Does not capture specific business rules about each classification such as the optionality or cardinality of each relationship from the classification to another entity. Does not capture specific attributes and relationships for different classifications. Supports only limited numbers and types of hierarchies and aggregation because there are only one-to-many recursive relationships for categories and category types. Also does not maintain classification schemes. Does not specifically enforce the business rule about which categories may be rolled up according to the category type structures (although it helps with this by maintaining the valid types of recursive relationships).

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 3 Classification Pattern with Rollups and Schemes	A flexible approach for capturing and managing classification data that allows many-to-many classification rollups as well as classification scheme data.	The same types of scenarios as the Level 3 Classification Pattern. When an enterprise needs to capture the source of classification data and various types of classification schemes. When classification and classification type data have multiple different ways to "roll up."	The same types of scenarios as the Level 3 Classification Pattern. When there is no need to capture scheme information. When there is no need for many-to-many classification rollups.	Similar pros as the Level 3 Classification Pattern. A comprehensive classification pattern that allows companies to capture the source of the data classifications as well as information about the classification scheme. Allows many-to-many relationships for accommodating very flexible category and category type rollups. Allows multiple different classifications to be part of different taxonomies.	Similar cons as the Level 3 Classification Pattern. It is complex and difficult to understand. Schemes as well as many-to-many recursive relationships may not be needed, and thus this pattern may be overkill.

References

[1] Carl Linnaeus was known as the father of taxonomy. See http://www.ucmp.berkeley.edu/history/linnaeus.html.

[2] For more on specific modeling versus generalized modeling see Chapter 1 of this book.

[3] Taken from Dictionary.com at http://dictionary.reference.com/browse/type.

[4] Taken from Dictionary.com at http://dictionary.reference.com/browse/category.

[5] Definition taken from the web site of The Natural History Museum in London, England, at http://www.nhm.ac.uk/nature-online/science-of-natural-history/taxonomy-systematics/what-is-taxonomy/what-is-taxonomy.html.

[6] See Michael Barnwell's article "What Is Taxonomy: Organizing Content for Better Site Performance (June 2005). Available at the Avenue A/Razorfish web site at http://www.avenuea-razorfish.com/articles/TaxonomyInsight_Barnwell.pdf.

[7] This characteristic may be referred to as morphism. Morphism describes the mapping between a domain and a codomain, which is very similar to describing a category mapped to another category. See http://planetmath.org/?op=getobj&from=objects&id=8114.

[8] For more on WORK EFFORT(s), see The Data Model Resource Book, *Revised Edition, Volume 1, A Library of Universal Data Models for All Enterprises*, by L. Silverston (Wiley, 2001).

[9] For more on recursive relationships, refer to Chapter 4 of this book.

[10] See Competing on Analytics: The New Science of Winning by Thomas H. Davenport and Jeanne G. Harris (Harvard Business School Press, 2007).

[11] Taken from Dictionary.com at http://dictionary.reference.com/browse/scheme.

[12] For more on contextual roles, see Chapter 3 of this book.

[13] The two schemes we referred to are from http://www.census.gov/foreign-trade/.

Chapter 6

Status: The States of Data

With the increase in sophistication of the world of commerce, processes across enterprises have grown even more complicated. Programs or projects to provide services have become more involved and complex, and the state of these work efforts need to be managed in greater detail. Financial information, such as options prices or interest rate changes, is affected by many different bits of information that have many different states. When you add to this the proliferation of computer systems in enterprises, the need for structures to capture and manage statuses becomes crucial in modern enterprises today.

As data flows naturally through business processes, a great number of statuses may be created, updated, or deleted in many different ways. For example, imagine a large electronic retailer receives an order for 1,000 mobile phones. After this order is received, it may be submitted into the order entry system. A supervisor may validate the order. If the order has been submitted correctly, the order entry system checks if the inventory is available for the customer; if the inventory is available, the order triggers the logistics system to set up the shipment. After the order is shipped, the billing system generates a request for payment, or in other words, an invoice. In this scenario, the order for the mobile phones went through several states such as “Received,” “Entered,” “Confirmed,” and “Pending” (waiting for an inventory check). Do “Shipped” and “Invoiced” represent statuses for the order? Or are they statuses for the associated shipments and invoices? There may be many

other statuses for orders such as “Cancelled,” “Failed credit check,” and so on. As processes change, additional statuses may be needed for orders such as “Possible fraud detected” or “Waiting for supervisor approval.”

In the preceding scenario, many transactions were generated as the order transitions from one state or status to another, and the status of the order changes as different events happen. At different relative times in all of the processes, various entities may have one or more status. These statuses indicate some business standing or condition, such as “Entered,” “Cancelled,” and so on. Having a clear strategy for managing statuses is crucial for any enterprise. How else would it know what was happening inside its business?

A very common problem for data professionals when interviewing subject matter experts is the difficulty in unraveling and understanding the various statuses as various transactions or events occur. For example, operations staff may say that their job is to close “Open orders.” To operations people orders may be either “Open orders” or “Closed orders.” They see only one small part of the life cycle. So, the orders are always open when they do their work, and when they are finished, the orders become closed and are someone else's problem. The finance department may consider an order to be paid, or unpaid. In other words, an order only has two statuses of interest to the finance department, “Paid” or “Unpaid.” To a data professional, the order may be in all of the statuses we have mentioned, plus many more statuses, as it goes through a complete order fulfillment life cycle.

On data modeling engagements, when we have reviewed existing database designs, we have found that a large percentage of fields are dates or datetime and many of these fields represent statuses. For example, there may be fields for product introduction date, person employment

date, invoice date, phone call start datetime, and a plethora of many other date and time fields. What we also find is that these dates are usually modeled quite differently without regard for the idea that there are huge advantages for consistent management of this very common and significant data within enterprises. This chapter will address the requirements for modeling status data in a consistent fashion.

What Is the Significance of This Type of Pattern?

What has happened to my order? How long does it take to expedite an order? How many invoices are overdue? What is the current state of a product; for example, was it introduced, is it going to be discontinued, and so on? These are all examples of common questions that an enterprise may ask itself regarding statuses. Knowing the status of crucial business data and the dates and/or times for the statuses is the only way to accurately answer these types of business questions. Knowing when a status is created, changed, or deleted is not always applicable; in some circumstances just knowing the status of a piece of business information is enough. For example, you may just want to know that an inventory item is in the state of “Damaged beyond repair,” and it may not matter when the inventory item became damaged, although depending on the situation this may be important data. More often than not, a crucial time aspect that maintains either a date, a date and a time, or from and thru dates, is associated with a status. Therefore, the fundamental questions that need to be answered concerning statuses are:

- What are the allowed statuses of entity, that is, what are the possible values for statuses? For example, what are

the possible statuses for parties, products, inventory items, orders, shipments, invoices, payments, accounting transactions, budgets, and many, many other entities in the data model?

- What is, and was, the status of a particular entity? For example, is an order “opened,” “closed,” “pending,” “expected to be shipped”? For this question, you may want to choose from the allowable statuses and assign a particular status to the entity.
- What time-specific data do you need to capture about the statuses? Do you need to capture the date (or the date and time) that the status occurred, or is expected to occur? For example, *when was* the order “opened,” “closed,” “pending,” and *when will* the order be “expected to be shipped”? Do you need to capture the date the status started and ended (from and through date for a status) or do you simply capture one specific date and time (and not a range of date and times) that an event occurred? In other words, do you need to know when the status came into existence and then ceased to exist, or just to know when a status occurred or changed? For example, you may need to know that an order was “Opened” on “March 1, 2009, at 2 p.m.” and “Closed” on “March 4, 2009, at 3 p.m.,” or for other order statuses such as “Received,” you might need to know just the specific date and time that the order was received.

What Is in This Chapter?

- This chapter first defines what a status is. The chapter then describes different levels of data modeling patterns that support the need for enterprises to model statuses in a consistent fashion. Each pattern supports the three fundamental aspects of statuses:

- The allowable statuses for a particular entity
- What the status *is* for that entity (or what the statuses *are* for the entity if there are many at the same time)
- When did that status become effective, change, or cease to exist if needed

The chapter starts with the most specific style (Level 1 Status Pattern) and moves to a very generalized style (Level 4 Status Pattern).

Two patterns at the end of this chapter, the “Status Category Pattern” and the “Status Type with Multi Rollup and Rules Pattern,” are provided as a way to enhance the level 2, level 3, or level 4 patterns. These patterns enhance the status pattern by bringing in patterns from chapter 5 (on classifications) and chapter 4 (on recursions). The “Status Category Pattern” supports the need to classify statuses into various sets and can help when an entity has more than one set of statuses. For example, an ORDER may have one set of statuses for order processing (“Received,” “Entered,” and “Confirmed”) and another set of statuses for scheduling (“On Schedule,” “Behind Schedule,” and “Overdue”). The “Status Type with Multi Rollup and Rules Pattern” supports the need to maintain rules about how statuses can (or can't) be related to each other. In other words, you may want to maintain that you cannot have an order that is “Entered” before you have “Received” that order.

Note

The same concept that is used in the “Status Category Pattern” and the “Status Type with Multi Rollup and Rules Pattern,” namely of using other patterns to enhance a pattern and, in this case, specifically adding flexible self-associations (recursions) and classifications, may be used as a technique to enhance other patterns in this book as well.

The data model patterns within this chapter can be used by most enterprises to build consistent data models that support a great variety of needs regarding maintaining data about statuses.

In summary, this chapter includes the following:

- A definition and introduction to the concept of status
- The relevance and significance of each of the patterns
- The different levels of status patterns
- When to use, and not to use, different status patterns
- Insights into each pattern
- A synopsis of all the patterns, pros and cons, and when to use and not use them

> **Note**
>
> Status is known as “state” in Unified Modeling Language (UML). In other contexts, statuses represent a core concept, such as for finite state machines, directed graphs, and other types of models. Triggering transactions are often referred to as transitions in UML.(1)

What Is a Status?

Status can be defined as “a state at a particular time; for example ‘a condition (or state) of disrepair’.”(2) Status indicates some state of affairs or situation that applies to the data. For example, an account may be “Open” or “Closed.” A status can indicate some legal condition that the data must conform to, such as balance sheet account that is in a state of “Satisfies Sarbanes-Oxley Compliance” or a financial account may be put under “Suspension” because of an ongoing criminal investigation. Computer programs often use statuses to decide on a particular path to follow. CEOs want to know how many “New Account” customers were generated in a period of time or how many projects are “Overdue.” So, what does this mean in terms of data

model patterns? This means that an instance of an entity may be in (or was in) a status of some type and this often has important business consequences.

Status types may be grouped into particular classifications. For example, an account may have “Account Opening” status types such as “Requested,” “Entered,” and “Opened.” The account may also have “Account Maintenance” status types, such as “Dormant,” “Active,” “Inactive,” or “Under Investigation.” Interestingly, this means that an account can have more than one status at a time. In other words an account may be “Opened” and “Active” (or “Opened” and “Inactive”). So, what does this mean in terms of data model patterns? This means that each type of status may be classified into one or more status categories.

Can an instance of an entity have more than one status type from within a single classification? For example, can an account be both “Entered” and “Opened” (these are both “Account Opening” statuses) at the same time? Surprisingly, the answer is yes; it is possible. That's because the “current status” of an entity is subjective. It depends on who is looking at the status of that entity. For example, if an account is “Requested,” then “Entered,” and finally “Opened,” it would seem that the account has only one current status of “Opened.” But to the account entry manager the account has been “Entered,” which means his or her staff has done its job, and he or she can report on how many accounts were entered that day. To the account manager the account has been “Opened,” which means he or she can report on the revenue for this account. In other words, the account entry manager is interested in “Entered” statuses, and the account manager is interested in “Opened” statuses for accounts. Thus, the account can have both statuses at the same time! Both statuses are ‘current’ for each of the different managers and their respective

views. Thus, you must be careful that to recognize that an entity may have many specific ‘current statuses’ (as well as having a history of statuses) to specific people and/or groups in an enterprise, and across the enterprise as a whole, you should recognize that the entity may have many different ‘current’ statuses at the same time. So, what does this mean in terms of these patterns? This means that you may need to support the fact that an ENTITY may be associated with one or more statuses at the same time and/or over time.

> **Note**
>
> In some circumstances, there may only be the need to maintain a single ‘current’ status of an entity, which is ‘current’ to all parts of an enterprise. We support this with the Level 2 Status Pattern, Current Status.

There are circumstances when different status types in the same status classification (or across different status classifications) may have rules that govern the behavior between those statuses. For example, a customer may be active or inactive,(3) but the customer can't be active and inactive at the same time! In other words, some status types are mutually exclusive. Other status types may be substitutes for each other. For example, if an order is “Entered” and if the order status is “Entry Complete” this may in fact be two status types for orders from two different applications that mean the same thing. What this means is that status types may be related to each other, and it is important to capture the nature of that relationship between statuses (for example, are these statuses synonyms for each other, are they mutually exclusive, and so on?). Thus, the patterns may need to support some rules that dictate the behavior between different statuses. We address this issue with Status Type with Multi Rollup and Rules Pattern in the final section of this chapter.

Many statuses are derived. For example, an ORDER typically has many ORDER ITEM(s), and these ORDER ITEM(s) can be in various statuses (“Backordered,” “Inventory Assigned,” and so on). You may think that there is also a “Shipped” status of either the ORDER or the ORDER ITEM. However, the ORDER (and ORDER ITEM(s)) is generally related to a SHIPMENT (and SHIPMENT ITEM) and the “Shipped” status can be derived based upon the status of the associated shipment. Rather than maintain these derived values as instances, we recommend that the application derive the status of the ORDER (in this case “Shipped” can be derived by looking at the associated SHIPMENT(s)). It is important that you do *not* model redundant status attributes that may need to change based upon the status of another entity. For example, when you change the SHIPMENT status to “Delivered,” you don't want to have to synchronize this with an ORDER status of “Shipped” because you can derive this and the statuses could potentially be out of sync if you maintain the status in more than one place. Because we try not to maintain data redundantly, it may mean that certain statuses are *not* directly modeled anywhere, but must be derived.(4)

Another way to look at this is to imagine that an ORDER, SHIPMENT, PAYMENT, or any other entity, can be seen as supporting a significant business transaction. Inside of this transaction there may be many events that change, update, or amend that transaction. For example, a SHIPMENT may have to support a “Shipment scheduled” event, a “Shipment packed” event, and a “Shipment delivered” event. You can model the SHIPMENT in your data model using the shipments data models from Chapter 5 of *The Data Model Resource Book*, Volume 1, Revised Edition (Wiley, 2001). This will help to capture the complete transaction. But you may not specifically want to have an entity in your model for all of the events that update, change, or amend the

SHIPMENT. Would you have a SHIPMENT SCHEDULED entity or a SHIPMENT DELIVERY entity in your “Shipments” data model? You may or you may not. If you do specifically model these events as entities, you could derive the status of your SHIPMENT based on the SHIPMENT SCHEDULED or SHIPMENT DELIVERY entities in your data model. If you don't model the events as entities, you can use the status pattern to capture these events and, hence, capture the status of the SHIPMENT.

In this chapter we refer to statuses that may happen in the future. For example, there may be a **shipment scheduled date**, which is the date that a shipment was scheduled to be sent, or an **order expected payment date**, which is the date that you expect to receive payment for a sales order. Some data modelers have valid reservations about modeling these expected dates as statuses. These modelers may have the view that statuses should be captured only as they happen. In other words, a SHIPMENT has a status only when some shipping event has taken place and not when you are expecting something to happen. Similarly, an INVOICE has a status only when something happens regarding the invoice, such as when you have received the check (and it has cleared) from your customer, and not when something is expected to happen. This perspective may view that **shipment scheduled date** and **order expected payment date** are not statuses; they are information about the SHIPMENT and INVOICE entities and should be modeled as attributes of those entities. This view is supported by the fact that the expected date that an event occurs is often different from when it actually occurred. For example, the date a SHIPMENT was “Shipped” may be later (or earlier) than the **shipment scheduled date**.

The alternative view (that expected dates do actually represent a status), is based upon the idea that when an

expected date is recorded, it can change the status. For example, when an **expected scheduled date** for a SHIPMENT is changed from Mar.1, 2010 to Jan. 1, 2010, some modelers would say that this is an important status of the SHIPMENT and it changes the “state” of the shipment because the shipment is expected much earlier.

Whether you subscribe to the view that an expected date is a status or not, the status patterns in this chapter are equally valid. If you subscribe to the view that an expected date is not a status, then you could use the status patterns only to capture statuses that have happened, and model “future date” statuses as attributes. Alternatively, if an enterprise decides that it wants to maintain “future date” statuses, it would include these as well as statuses that have happened in the patterns. We see both views as valid and we don't have a preference for either viewpoint—only that you use a consistent view throughout your modeling efforts. In this chapter we decided to include “future date” statuses because we wanted to illustrate the broadest range of statuses that this pattern can support.

A status pattern may maintain the following information:

- **The allowable statuses that data can have** A status pattern may allow for a set of attributes for a status type such as name, description, effective from date, effective thru date, and so on.
- **The status(es) that are applied to an entity** For example, that an order was “Entered.”
- **The time aspect of the statuses, if that aspect exists** When was an order “Opened,” and when was the shipment “Shipped”?
- **The classification of the status types** This allows statuses to be grouped into sets of similar classifications such as an ORDER having one set of statuses for order processing and another set of statuses for scheduling.

- **Rules that may exist between the different types of statuses** For example, there may be a rule stating that an order first has to have a status of “Opened” before it can be in a state of “Closed.”

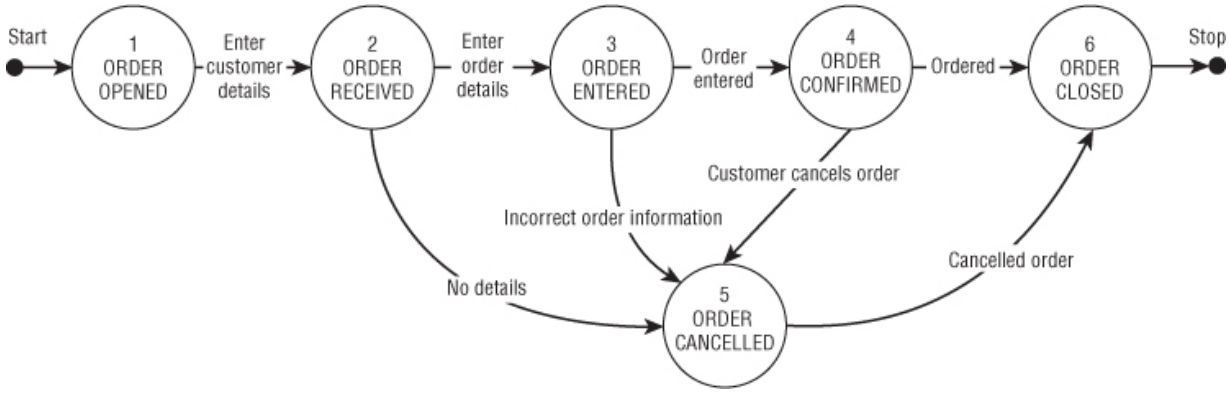
Level 1 Status Pattern

Statuses can be modeled in a very specific fashion using a Level 1 Status Pattern. This pattern maintains each status using “event” attributes of the entity, for example, **order received date, order entered date, order confirmed date**, and so on. The statuses are updated as an entity goes through some predefined set of steps that make up the business process, for example, an order may be “Opened,” “Received,” “Entered,” “Confirmed,” and “Canceled,” or “Closed” as it goes through its life in an order fulfillment process. This is illustrated by the state diagram in Figure 6.1. The basic status pattern must support the list of all the different statuses that the data can have over its entire life cycle.

Note

We have used a relatively simple process model for the statuses of orders to illustrate use of this pattern. We recognize that there may be many other statuses in an order process and the flow may be different. For example, there may be statuses of “Order data needs correction” instead of moving to “Order Cancellation” or in some environments, the “Order Cancellation” may be the last status instead of moving to an “Order Closed” status afterwards.

Figure 6.1 Order fulfillment state diagram



The Level 1 Status Pattern supports the three important characteristics described in the previous section.

- First, the allowable statuses are recorded as potential attributes. For example, ORDER could have status attributes of **order received**, **order entered**, **order confirmation**, **order cancelled**, and **order opened from date, order closed thru date.**
- Second, the entity is recorded as being in a status by updating an attribute value. For, example, an ORDER may be considered in the status of “Opened” when there is a value in the **order opened from date** attribute
- Third, the status may have an associated time or time span. For example, the order was “Received” on Jan. 4, 2 p.m., 2010 and this may be maintained by recording the date and time in an **order received datetime** attribute.

Note

Some events happen at a point in time and other events happen over a period of time. For example, an order may have been “Received” at 12:32 a.m. on January 5 and the same order was opened from “January 5” through “January 10.” This means that some statuses are maintained with a single date (or datetime) and other statuses with a date range.

Note

On occasions you may not need to capture a time element for a status. For example, an enterprise may decide that it only needs to state that a shipment was “Planned” and not the date on which the shipment was planned.

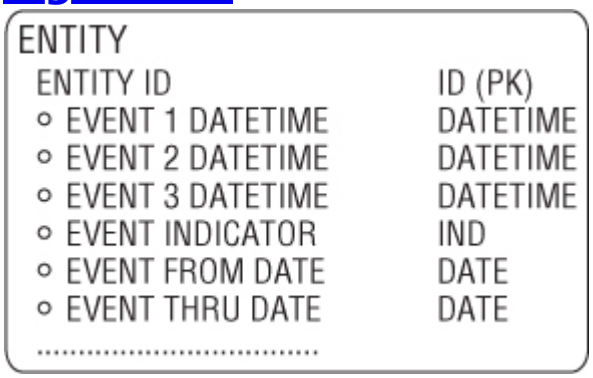
Why Do We Need This Pattern?

The purpose of the specific Level 1 Status Pattern is to explicitly maintain all the statuses for an entity with the “event” attributes of that entity in order to create a very simple, specific, and understandable model. This type of model can be effective in showing business representatives and subject matter experts (SME) the scope of the statuses that are being defined. A SME can see that an order can be opened, closed, and confirmed if there are specific attributes for each of these in an ORDER entity. Also, the pattern captures the date (or date and time) that a status was activated (or scheduled), if applicable; for example, **order opened from date**, **order closed thru date**, and **order confirmation date**. This pattern provides an illustration for the SMEs to see a listing of the events from which they can derive statuses for an entity and the significance of the dates (or dates and times) associated with the statuses.

How Does This Pattern Work?

Figure 6.2 illustrates how attributes are used to maintain the basic status for an ENTITY. ENTITY represents the relevant data, transaction, or event for which the data professional is interested in maintaining statuses. For example, this could be PARTY, PRODUCT, TRADE, RESERVATION, ORDER, SHIPMENT, and so on (in many data models, there are dozens of entities that maintain status data).

Figure 6.2 Level 1 Status Pattern



The ENTITY may have one or more different statuses because of one or more events that occurred within the context of the entity. For example, an order may be "Opened," "Confirmed," or "Closed." Each status attribute (**event 1 datetime**, **event 2 datetime**, **event from date**, **event thru date**, and so on) represents a status corresponding to when that event happened. ENTITY may have multiple different statuses at the same time; for example, **event 1 datetime**, **event 2 datetime**, and **event 3 datetime** may all have values at the same time. If an event attribute has a value, you consider that the event has happened or will happen if it is a scheduled event, such as **shipment scheduled date**. If the event does not have a value, the event has not happened (or is not planned or scheduled to happen), and the status has not been set.

In this pattern, there are two things that you are maintaining when you add a value to the attributes **event 1 datetime**, **event 2 datetime**, **event 3 datetime**, and so on. You are recording that status (or state) of an entity as well as when an event occurred. We normally would not encourage data modelers to have two purposes for an attribute because some may consider this "overloading" an attribute and bad data modeling form. However, in the case of this status pattern, this can be justified because the date when the event occurred implies the status of the ENTITY. The alternative of creating an **event indicator** attribute in

addition to an **event status datetime** attribute is usually redundant. For example, you could model an **order received indicator** to record that the order was received and then have another status attribute for **order received datetime** to record when it was received. The attribute **order received indicator** does not have a time value; it is just an indicator that says that an event has occurred or has been scheduled. Rather than have two attributes, one for capturing whether the status did or is expected to occur and one for capturing *when* it occurred (for example, a **status datetime** attribute), we think that it is more advantageous to capture the status of the ENTITY along with the time(s) that it occurred in the same attribute because putting in a time for the event implies that the status did actually occur (or is scheduled to occur).

Some statuses may not have any time component at all. For example, if you have an attribute of a SHIPMENT called **shipment overdue indicator**, you can know that the status of a shipment is “Overdue,” but it may be a status that may not have a ‘time’ element (although it depends on the circumstances if this attribute needs a time element also). Finally, it should be noted that not all statuses refer to a point in time; some statuses happen over time. For example, **event from date** and **event thru date** capture statuses that exist within a date range. In Figure 6.3 **order opened from date** and **order closed thru date** are examples of statuses that exist over a period of time. Put another way, some statuses happen at a point in time, such as a shipment being delivered. Other statuses happen over a time period, such as an order being open for a week and then being closed when the order is fulfilled. For this reason, some statuses are ‘point in time’ statuses, and other statuses are ‘range of time’ statuses. Quite often, statuses that are ‘range of time’ are interested in just the date and not the date and the time; thus the data type shown in the

pattern is DATE, not DATETIME. However, a datetime data type could be used for them if that level of time granularity is needed. Capturing the proper ‘date’ data type for a status is important because the business often requires a specific level of granularity to be maintained when capturing a status. For example, a company may be interested that most of its orders get entered before midday, or it may want to know the average number of hours that orders are open on a month-by-month basis.

> **Note**
>
> Although the pattern shown in Figure 6.2 (and implemented in Figure 6.3) shows that there may be ‘point in time’ statuses that are DATETIME data types and ‘range of time’ statuses that are DATE data types, either DATE or DATETIME data types may be used for either type of status and what data type to use depends on what is needed for each particular circumstance.

> **Note**
>
> Some data modelers may prefer to show ‘range of dates’ attributes using the same event type name. For example, instead of having attributes of **order opened from date** and **order closed thru date**, alternative names for these attributes could be **order opened from date**, **order opened thru date**, signifying that the order was opened from a certain date and thru (and including) another date. Yet another alternative is to regard each of these as “point in time” statuses and have an **order opened date** and **order closed date**. Each of these alternatives is valid and it is largely a matter of defining consistent semantics that you will use.

Figure 6.3 Example of using a Level 1 Status Pattern

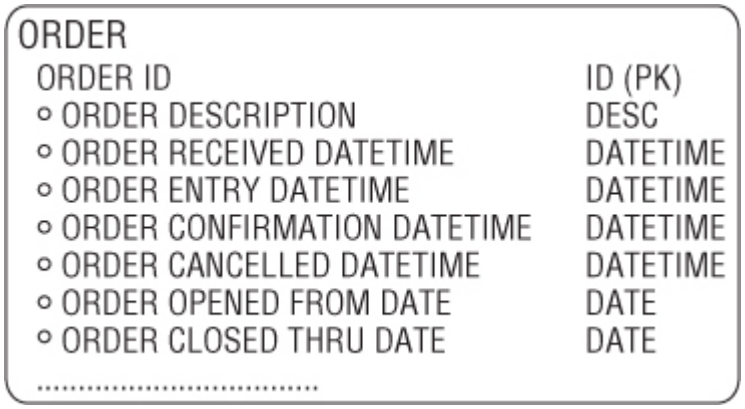


Figure 6.3 further illustrates how to use this pattern. The scenario is as follows: imagine that a large mobile phone manufacturer wishes to review the different states that an order goes through as it gets expedited through its order fulfillment work flow. The data professional has been employed to examine the different statuses of orders and then report on all of the different states that an order can have to the head of operations and IT.

After some detailed analysis of the order fulfillment work flow, as seen in Figure 6.1, and after interviewing different experts in the company the data professional created Figure 6.3, based on the Level 1 Status Pattern. Figure 6.3 contains the ORDER entity, where ORDER(s) are a commitment to pay money for the delivery of goods or services.(5) After doing some analysis, the data professional captured the different events that the order goes through as attributes of ORDER. When certain events occur, each of the statuses may be updated for any instance of ORDER.

In this scenario, the data professional discovered that when there is any data at all about an order it is regarded as “Opened.” This could be when a salesperson has contacted the customers and has a verbal commitment to order some goods and services. The salespeople enter some very basic data into the order entry system at this point and get an order number. The customers may then email, fax, or send in the order details for their goods and/or services, and at

this point it is considered “Received.” So the order may be “Opened” before it is “Received.” For example, in Table 6.1 you see an instance of ORDER with **order id** “12560” for “Deluxe Mobile Phone” with an **order received datetime** of “Feb. 2, 2010, 2 p.m.” and that has an **order opened from date** of “Feb. 1, 2010.” In other words, the **order opened from date** is the date when the company recognizes that an order is in the order fulfillment work flow (perhaps by some verbal commitment), and the **order received datetime** is the date and time that the order information was formally sent by the customer.

Table 6.1 Example of Using the Level 1 Status Pattern

ORDER. ORDER ID	ORDER. ORDER DESCRIPTION	ORDER.ORDER RECEIVED DATETIME	ORDER.ORDER ENTRY DATETIME	ORDER.ORDER CONFIRMATION DATETIME	ORDER.ORDER CANCELLED DATETIME	ORDER.ORDER OPENED FROM DATE	ORDER.ORDER CLOSED THRU DATE
12560	Deluxe Mobile Phone	Feb. 2, 2010, 2 p.m.	Feb. 2, 2010, 3 p.m.	Feb. 3, 2010, 9 a.m.		Feb. 1, 2010	Feb. 12, 2010
23000	Mobile Phone Standard	Feb. 2, 2010, 2 p.m.	Feb. 2, 2010, 6 p.m.	Feb. 3, 2010, 9 a.m.	Feb. 4, 2010, 2 p.m.	Feb. 2, 2010	Feb. 14, 2010
32999	Mobile Phone Accessories	Jan. 3, 2010, 3 p.m.	Jan. 3, 2010, 5 p.m.	Jan. 12, 2010, 4 p.m.		Jan. 3, 2010	

The data professional also discovered that there was a time stamp for the time an order gets “Entered” into the order entry computer system and captured this time stamp in the **order entry datetime** attribute of ORDER. From Table 6.1, the order with **order id** “32999” for “Mobile Phone Accessories” has an **order entry datetime** of “Jan. 3, 2010, 5 p.m.” The data professional discovered three different statuses that indicate three potential order creation dates for an order. If senior management wanted to know the average length of time an order took from creation to fulfillment, they could have three different answers. To illustrate this, consider order “32999,” which is shown in the third row in Table 6.1.

- The sales subject matter expert (SME) considers an order created at the time that there is any notice of a commitment such as when the customer calls in his or her verbal commitment. In the case of order “32999” that was when the order status **order received datetime** was set to “Jan. 3, 2010, 3 p.m.”
- From the perspective of the data entry staff, an order is created at the **order entry datetime** “Jan. 3, 2010, 5 p.m.,” 2 hours later than sales considers it was created.
- From the perspective of the logistics staff, the order begins its life cycle when it enters the order fulfillment work flow (as seen in Figure 6.1), represented by the status **order opened from date** of “Jan. 3, 2010,” so in this case, they measure the life cycle in days and not hours.
- The accounting staff would book a percentage of revenue of an order when an order had a status of confirmed. In order “32999” for “Mobile Phone Standard” the **order confirmation datetime** was on “Jan. 12, 2010, 4 a.m.” From the accounting department's perspective, this is where an order's life cycle begins.

This is very valuable information from the mobile phone company's perspective. It may indicate a misunderstanding between different departments, it may indicate that the enterprise has overlapping statuses that could be consolidated, or it may indicate a valid set of different perspectives that the company wishes to maintain.

In Table 6.1 you can see that the order with **order id** "23000" for a "Mobile Phone Standard" got cancelled on "Feb. 4, 2010, 2 p.m." The status of that order was cancelled on that date, but if you look at the **order closed thru date**, the last day on which the order was opened, it was "Feb. 14, 2010." It seems that an order can be cancelled, but still open at the same time. This is quite normal. Many different transactions may still have to be completed that support the cancellation of an order and turn it into an order that is closed. In the case of order "23000," the order revenue was booked because the order was confirmed on "Feb. 2, 2010, 2 p.m." and that now had to be reversed. When the necessary reversals and other processes were completed, the order was closed on the **order closed thru date** of "Feb. 14, 2010."This example illustrates some very useful things. First, the data professional captured order events and maintained the states that spanned the life cycle of an order. The statuses were not limited to one particular group of experts in the company, and therefore, many different perspectives were captured. These different perspectives about the states of an order get neatly grouped together into a single construct. Finally, the pattern shows that the time component of a status can be for a point in time or for a range of time.

Note

Many data modelers do not like to capture class words in attribute names. For example **event 1 datetime** would be captured as **event 1**. There are some very good reasons for not having class words as part of the attribute name. For example, it

can be considered as redundant to capture the fact that **event 1 datetime** has a data type of datetime in the attribute name, the data type datetime is captured in the entity already. What happens if the data type changes to date? The attribute name is no longer valid. For the purposes of this book we capture the class words in the attribute name as it helps when reading explanations of the patterns, and it's often useful to help identify the nature of the attribute without having to look up the data type. This is a common data modeling issue; modelers and will often feel strongly for either solution, and both perspectives have strengths as well as weaknesses. We don't have a particular preference, only that it's consistently applied in you enterprise.

When Should This Pattern Be Used?

We use this data model pattern:

- **When there is a well-defined set of specific statuses that are static and when additional statuses are not expected:** The area under investigation was considered to be static and new types of statuses were not expected. In the case of the preceding scenario, it specified particular statuses such as an **order received datetime**, **order entry datetime**, **order confirmation datetime**, **order cancelled datetime, order opened from date**, and **order closed thru date** that were perceived to be static and were not anticipated to change in the future.
- **When there are very few statuses:** For example, in some applications, the entity PERSON may have only two different statuses of “Alive” or “Deceased.” This pattern may be appropriate because the flexibility of allowing any number of statuses (provided by other patterns in this chapter) may not be needed or it may be considered to be overkill. Another option is to use a combination of this pattern for the critical statuses and use the level 2, level 3, or level 4 patterns for the other statuses. For example, you may choose to maintain just

the **order entry datetime** in the ORDER entity using this pattern and maintain the other statuses with a more flexible pattern.

- **If a status is only ever specific to a particular entity, you may wish to capture it as an attribute of that entity:** There is a stronger argument for capturing this type of entity-specific status as an attribute, as it will never be reused for any other entity the data model. For example a SHIPMENT may have **shipment lost date** status. This status is only ever used for shipments, and it is not used by any other entity. If this is the case, the pattern may even be used as an implementation, for example, in the case of a prototype.
- **When the data professional needs a simple way to represent the business data requirements, perhaps as part of a basic statement of scope to nontechnical audiences or management:** The diagram was given to the head of operations and business representatives as a simple statement of the statuses of the orders and as a way to communicate and validate the data requirements. The diagram showed the states (as derived from the event attributes) for ORDER. This pattern is very easy to understand because it uses a specific style of modeling, and therefore, it is effective as a tool to help understand data requirements. This model reveals the terminology that was used across different groups in the enterprises. It can be used to kick-start the discussion on statuses among the different groups involved with orders.
- **When it is important to illustrate and show different perspectives:** Different groups in the enterprise had very different views on when an order is actually created. Each of the different stakeholders in

the preceding scenario had very specific ideas about the order statuses that were important to them.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern would not be suitable in a dynamic environment in which new status types emerge and are needed over time:** As processes change, new statuses are often needed. For example, new processes may lead to new statuses that are needed such as “Pending credit check,” “Approved by supervisor,” or “Accepted by legal” statuses. The entity would have to be updated with the new attributes and use of this pattern may require the model to be changed, which could be very expensive.
- **When there is a great number of statuses for an entity, and therefore, use of this pattern may make it more difficult to consistently manage statuses:** If a very specific style of modeling was used, then for many circumstances, an ORDER entity may have many more than the six status attributes shown in Figure 6.3, and many of these attributes could be dates and indicators. It may not be clear which of these attributes are related to statuses and which are not (for example, if “Overdue Indicator” is considered a status field, but “Priority Indicator” is not).
- **The pattern doesn't provide classifications of statuses:** As we discussed, there may be sets of statuses; for example, an ORDER may have one set of statuses for order processing (“Received,” “Entered,” and “Confirmed”) and another set of statuses for order scheduling (“On Schedule,” “Behind Schedule,” and

"Overdue"), and this pattern does not provide a way to distinguish different sets of status types.

- **This pattern does not maintain rules about the statuses:** The pattern groups all of the statuses together related to an order, but does not show the rules about these statuses, such as if some statuses are needed in order to record other statuses. For example, there may be a rule that an ORDER cannot have a "Cancelled" status if there is not an entry for the "Received" status. This pattern does not support rules such as this.

Synopsis

The Level 1 Status Pattern is a very specific way to maintain statuses by maintaining "event" attributes that show the states that a particular entity may go through as well as when those states occurred or are scheduled to occur. In the case of orders, you saw **order received datetime**, **order entered datetime**, **order confirmation datetime**, **order cancelled datetime**, **order opened from date** and **order closed thru date** and each may get set at various points in time. It is significant to point out that the pattern derives what the status is from the date (or date and time) events, and doesn't redundantly capture them as attributes. For example, instead of maintaining an indicator to capture that an event occurred and then a status datetime attribute to record when it happened (or when it is scheduled to happen), we can accomplish this with a single attribute capturing the event that occurred as well as when it happened.

In this pattern you also saw an **order opened from date** and **order closed thru date** that show statuses that have a time span. An event may happen at a point in time, such as the date and time that an order is received, and some events happen over time, such as an order being opened

and subsequently closed at a point in time. Hence, some statuses are ‘point in time,’ such as **order received datetime**, and others are valid over a ‘range of time.’ It is possible that a status has no time component; for example a **shipment overdue indicator** may indicate that a shipment is overdue, but has no time associated with it.

When you maintain all of the related statuses for an entity in a single place, it is easier for interested parties in an enterprise to see the different perspectives different groups have of the statuses for an entity. It draws out the different vocabulary that an enterprise has for similar concepts. In the example from this section of the chapter, the various departments of logistics, sales, data entry, and accounting all used a different status to indicate the beginning of the life cycle of an order from each of their viewpoints. Thus, this pattern is very effective for use in gathering and validating the data requirements related to statuses.

In conclusion, this pattern can be used to help define the scope of an effort regarding statuses, and it provides an initial way to begin to collect the requirements for those statuses. It is simple to understand and simple to implement. But beware—if the environment is dynamic and statuses change, or get added or deleted, this pattern has the major disadvantage of requiring data model changes, which generally are very expensive.

Level 2 Status Pattern, Current Status

We have encountered the situation where enterprises are concerned only with the current status of an entity and where there is only one status for the entity because one status is replaced by another status as the entity goes through its life cycle: What is the state of my order now? Is

it “Received,” “Opened,” “Confirmed,” or “Closed”? What is the status of my shipment now? In some situations, enterprises are not particularly concerned with the multiple states that an entity may be in, or with capturing the history of those states. For example, a credit card clearing business may only be concerned with the current state of a transaction. It may not have the need to capture all of the different statuses a credit card transaction has throughout its complete life cycle. To support the need to capture just the current status, we use the Level 2 Status Pattern, Current Status.

This pattern provides a flexible strategy for data professionals when they wish to create a specific current status pattern solution. The nature of this solution is that it is used when an entity will have one and only one status. This means that throughout the whole enterprise every department will see only this one status for an entity. For example, an “Account” with the status “Opened” will appear “Opened” to data entry people, marketing people, and salespeople alike.

This pattern differs from the previous pattern in that it provides a more flexible solution because it allows for additional status types to be easily added as instances, when processes change over time and require these additional statuses. Although this pattern supports maintaining possible status types that an entity either has or could have in the future, there may be only one current status for the entity. For example, there could be only one status maintained at a particular point in time for an ORDER.

Why Do We Need This Pattern?

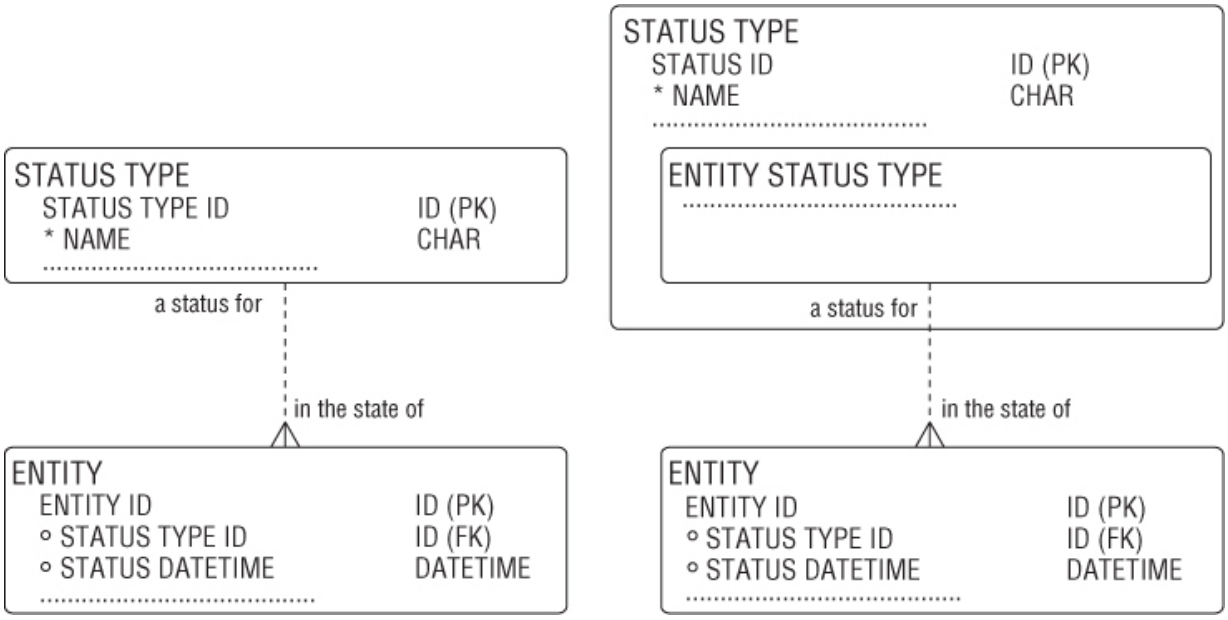
This pattern is needed for enterprises that may require additional valid statuses over time; however, one and only status may be assigned at any particular point in time to the

relevant entity. In the previous section you saw that the Level 1 Status Pattern explicitly defined each of the different status types for an entity as attributes of that entity. For example, in Figure 6.3 you saw that the ORDER entity had the attributes **order received datetime**, **order entry datetime**, and so on. If an enterprise wished to capture only the current status, all of the status attributes would be empty, except for the current one. This pattern provides a more elegant solution for capturing just the current status. Additionally, new statuses may be needed in the future, and the pattern accommodates this without any need to change the model. For example, an additional need for a status of "Order Approved" may arise that is due to an additional process to make sure that the order is acceptable before it is confirmed, and this pattern would accommodate this need by simply adding an instance of a STATUS TYPE for "Order Approved."

How Does This Pattern Work?

Figure 6.4 illustrates the pattern with the addition of a STATUS TYPE entity. The STATUS TYPE entity can be defined as a set of states or condition of affairs that share some or all common characteristics or sets of common characteristics.(6) The data professional can capture all of the known set of status types for an entity and have the capability to add new status types when they are discovered. This is achieved by "generalizing" the status types into their own STATUS TYPE entity. As was the case in the previous section, ENTITY represents a set of data, transaction, or event that the data professional is interested in, such as PRODUCT, INVOICE, PURCHASE ORDER, PAYMENT, TICKET, and many other possible entities.

Figure 6.4 Level 2 Status Pattern, Current Status



You can see from Figure 6.4 that there are two variations of this pattern:

- First, the different status types of an enterprise (for example, party status types, order status types, invoice status types, and so on) can be consolidated into a single entity STATUS TYPE. This works to simplify the model by managing all status type data in the same entity. Also, it is important to use this pattern when the different status types for all entities in an enterprise have the same attributes (although it is not shown in the model, the STATUS TYPE may include other attributes such as **description**, **short name**, **long name**, **effective from date**, and so on). STATUS TYPE contains the **name** for the different status types, such as “Opened,” “Received,” and so on.
- Second, a specific ENTITY STATUS TYPE may be created as a subtype of the STATUS TYPE entity. Using the subtype is a way of grouping statuses allowable for a specific entity into its own structure, thus allowing for the enforcement of a rule in the data model stating that an ENTITY can have only statuses that are in that entity's status subtype (ENTITY STATUS TYPE). For

example, the modeler may want an ORDER STATUS TYPE that maintains the specific statuses allowed for an order. Also, this variation is suitable for an enterprise that has the circumstance where the data for the various status types are not the same, that is, some may have different attributes and/or different specific relationships.

Note

In chapter 5, we implied that if an entity has subtypes there is usually an associated ‘type’ or ‘category’ entity. In the second variation of this pattern, we have subtypes for STATUS TYPE, and we could consider adding a STATUS TYPE TYPE entity that can be used to manage the classifications of status types. For example, we may want to inventory and classify all the different types of status types, such as which status types are for transactions versus products or parties. Alternatively, for a much more flexible way to provide classifications, we can enhance this pattern with the Status Category Pattern, which we will discuss later in this chapter.

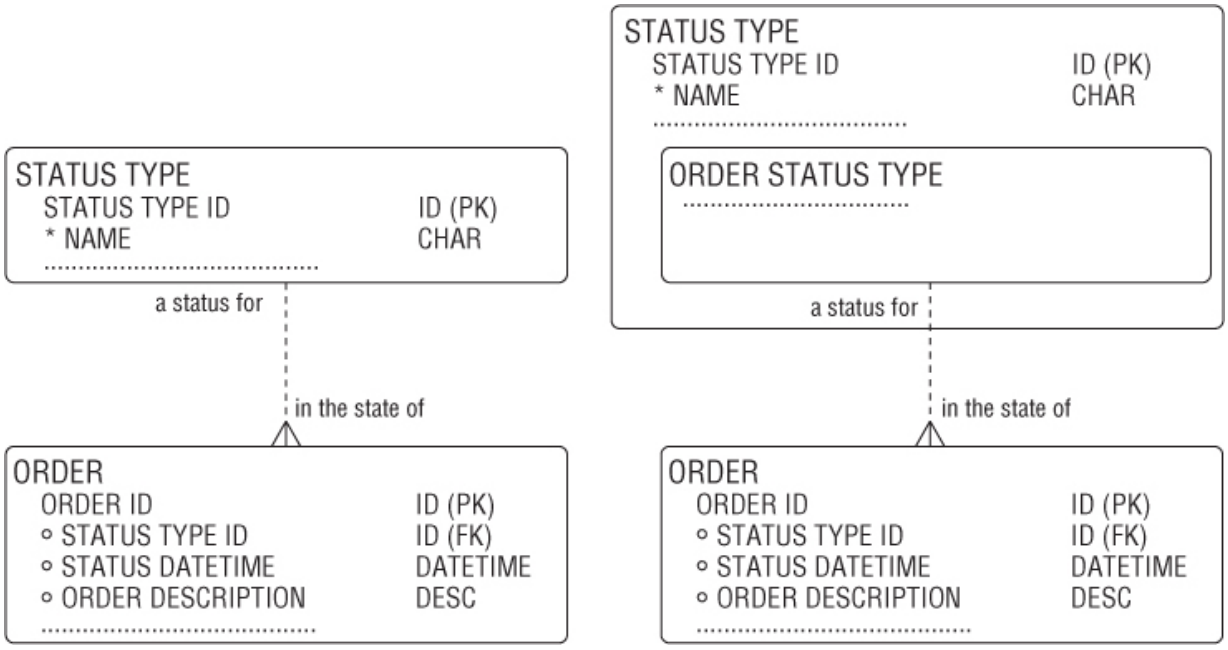
Both of the variations work in the same way, for the most part. Instead of capturing the statuses explicitly as attributes, as in the Level 1 Status Pattern, the data professional captures the different status types as instances of a STATUS TYPE entity, and the ENTITY inherits only the foreign key to maintain the entity's current state. As the status is captured, the foreign key attribute **status type id** in ENTITY is related to and migrated from STATUS TYPE. A time stamp may be maintained in a **status datetime** within the ENTITY in order to capture a time component showing when the status occurred.

It should be noted that an ENTITY may not have a particular status recorded at all, and this is why the relationship between STATUS TYPE (or ENTITY STATUS TYPE) and ENTITY is optional. This is a common enough occurrence in business. What is the status of a shipment? The answer

might be unknown, or it might not be recorded yet. Also, not all statuses have a time component. For example, statuses such as “Account Overdue” or “Order Pending” may just indicate that the entity is in a certain state. This is why the **status datetime** attribute is optional.

Figure 6.5 further illustrates how this pattern works. If you further expand on the scenario from the previous section, a large mobile phone manufacturer receives orders and wishes to have a full range of possible statuses that an order may have, but only wishes to capture the current status of an order. The mobile phone manufacturer wishes to create a prototype solution to handle orders, based in part on the solution provided by the data professional for handling the current status.

Figure 6.5 Example of using a Level 2 Status Pattern, Current Status



Based on the analysis the data professional did in the previous section, and using Figure 6.4 as a template, the data professional created the model on the left hand side of Figure 6.5 to meet the needs of the prototype (the model on the right hand side of Figure 6.5 is an alternate model that

could have been developed). The data professional captured each of the possible valid statuses for an order as instances of the STATUS TYPE. In Table 6.2 you see different types of statuses as instances of STATUS TYPE and then related to specific orders, for example, “Order Opened,” “Order Received,” “Order Entered,” “Order Confirmed,” “Order Cancelled,” and “Order Closed.” These statuses correspond to the states captured as attributes in Figure 6.1.

Table 6.2 Example of Using the Level 2 Status Pattern, Current Status

ORDER. ORDER ID	ORDER. ORDER DESCRIPTION	ORDER. STATUS DATETIME	STATUS TYPE. STATUS TYPE ID	STATUS TYPE. NAME
12560	*Deluxe Mobile Phone*	*Feb. 2, 2010, 2 p.m.*	*10*	*Order Opened*
12560	*Deluxe Mobile Phone*	*Feb. 2, 2010, 3 p.m.*	*55*	*Order Received*
12560	*Deluxe Mobile Phone*	*Feb. 2, 2010, 4 p.m.*	*20*	*Order Entered*
12560	*Deluxe Mobile Phone*	*Feb. 3, 2010, 9 a.m.*	*30*	*Order Confirmed*
12560	**Deluxe Mobile Phone**	**Feb. 19, 2010, 6 p.m.**	**60**	**Order Closed**
23000	*Mobile Phone Standard*	*Feb. 2, 2010, 2 p.m.*	*10*	*Order Opened*
23000	*Mobile Phone Standard*	*Feb. 2, 2010, 2 p.m.*	*55*	*Order Received*
23000	*Mobile Phone Standard*	*Feb. 2, 2010, 6 p.m.*	*20*	*Order Entered*
23000	*Mobile Phone Standard*	*Feb. 3, 2010, 9 a.m.*	*30*	*Order Confirmed*
23000	*Mobile Phone Standard*	*Feb. 4, 2010, 2 p.m.*	*70*	*Order Cancelled*
23000	**Mobile Phone Standard**	**Feb. 14, 2010**	**60**	**Order Closed**
32999	**Mobile Phone Accessories**	**Mar. 3, 2010, 3 p.m.**	**10**	**Order Opened**

Let's examine the life cycle of an order as it passes through the different statuses of the order fulfillment process. In Table 6.2 the order for “Deluxe Mobile Phones” with **order id** “12560” was opened (the status of “Order Opened”) on “Feb. 2, 2010, 2 p.m.” This was when the sales staff got a call from the customer and at that point in time “Order Opened” was the current status; then at 3 p.m. on

the same day the order was faxed in by the customer and then the status changed to “Order Received” and the ORDER **status datetime** was changed to “Feb. 2, 2010, 3 p.m.” The order was passed to the order operations department and was fully entered into the order entry system on “Feb. 2, 2010, 4 p.m.,” and the order got a new current status of “Order Entered.” The following day the salespeople rang the customer who placed the order and confirmed with her that the order details were correct. The current status of the order changed to “Order Confirmed” on “Feb 3, 2010, 9 a.m.” Finally, the order was closed successfully and its final current status was “Order Closed” as of “Feb. 19, 2010, 6 p.m.” One interesting thing to note is that the status datetimes do not have to be sequential regarding the order of time; the time component for a status is just the effective time of that particular status.

Note

To help in illustrating the pattern in Table 6.2 we have shown “old” current statuses. For example, order “12560” has five different statuses, four of which should/would have been overwritten with the next current status if this was implemented in a relational database table. We have displayed the “old” current statuses in italics and the latest current statuses in bold to differentiate them.

In the case of the instance of ORDER with **order id** of “23000” the order life cycle encompasses six different states: “Order Opened,” “Order Received,” “Order Entered,” “Order Confirmed,” “Order Cancelled,” and the final status “Order Closed.” This again complies with Figure 6.1. The last row in the table shows an instance of ORDER with **order id** “32999” for “Mobile Phone Accessories” that has a current state of “Order Opened.” This is an order that has just been opened.

When Should This Pattern Be Used?

This pattern should generally be used under the following circumstances:

- **When an enterprise only needs to capture one and only one state for an entity:** This pattern assumes that an entity can only be in one status at a given point in time. For example, an ORDER is in the status of “Received” at a point in time, then it moves beyond “Received” into the status of “Entered” and so on. This is not the norm because many times there is a need to capture more than one status for an entity, but this situation is not all that uncommon and we have come across this need many times.
- **When the enterprise does not care about capturing the history of statuses for an entity:** This pattern is used when the enterprise cares only about the current state and does not want to maintain that the entity was first in one state (for example, the order was received on Feb. 2, 2010, 3 p.m.) and then it went to another state (the order was entered on Feb. 2, 2010, 4 p.m.). Thus, using this pattern would not support getting the information about the average time between receiving an order and entering an order because one status replaces another.
- **When there is a need to provide for any number of possible status types that may emerge over time:** There may be either hidden or obscured requirements that may be revealed or new processes that require new statuses, and this pattern allows new statuses to be added simply by adding an instance of STATUS TYPE. Thus, the data model does not need to be changed when new types of statuses emerge.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern enforces the rule that an entity can have only one current status and that each previous status is overwritten:** Thus, not only do you lose the ability to maintain when the sequence of statuses occurred over time, but also with this pattern you can see that an entity cannot maintain two statuses that could occur at the same time (an order being “Opened” as well as in the state of “Confirmed”).
- **When you have only one current status, such as “Order Opened,” this can lead to different interpretations across your enterprise as a whole and it does not have the ability to record different statuses for different parts of the enterprise:** “Order Opened” may mean one thing to the shipping department and another thing to accounting. Because we can only maintain a single status we can't represent the different views for statuses across various parts of the enterprise.
- **This pattern is not as effective as the Level 1 Status Pattern at graphically illustrating the scope of the different statuses that an entity may have:** The statuses are captured as instances of the STATUS TYPE and not specifically as attributes. Therefore, they are not visible in the diagram or as easy to understand for nontechnical audiences.

Synopsis

In this section you examined how Level 2 Status Pattern, Current Status, meets the specific needs that some enterprises have.

First, this pattern captures all the different statuses that an entity has by generalizing all of the different types of status into a STATUS TYPE (and alternatively may include various subtypes) as seen in Figure 6.4 and implemented in Figure 6.5.

Because the status types are captured in the STATUS TYPE entity and not as specific attributes, the pattern accommodates change more easily because if a new status type is needed, you can simply add a new instance to STATUS TYPE and need not change the model.

This pattern has a somewhat narrow focus, to meet the specific need of only capturing the current status of an entity. However, in one aspect, it provides more flexibility than the Level 1 Pattern because it introduces the STATUS TYPE for the first time in this chapter. This allows you to add new instances of statuses when they are discovered.

You saw in this section that because of the specific nature of this pattern, it should not be used by an enterprise needing to capture the history of the different statuses of an entity or for an entity that has more than one current status at a given time.

Level 3 Status Pattern

Rapidly changing environments with new processes, new business rules, and new statuses is a common situation for many enterprises these days. For example, imagine a manufacturer wishes to get its ISO 9000 certification. This would require that it upgrade its entire business process to reflect a new level of excellence. As the business process gets changed to meet the new standard, so the data will probably have to go through different states to comply with the new standards.

Why Do We Need This Pattern?

Flexibility is an important consideration in modeling. Imagine a situation where an enterprise needed to capture all of the statuses related to a particular area of business, but it is not sure what statuses it currently captures. Imagine also that an enterprise knows that its business model will change, but not how it will change. This situation is common, in particular with enterprises that are not traditional, such as online video downloading or gaming, or an Internet search management enterprise. Most large enterprises have made major changes to their business process; for example, many enterprises need to change their processes in reaction to new regulations, such as the introduction of the Sarbanes-Oxley Act(7), or new banking regulations enacted after the “Credit Crunch” crises of 2008. The enactment of the Sarbanes-Oxley Act has required that enterprises put new accounting and compliance processes in place, at great cost in time, effort, and finance. A flexible approach to statuses would have helped enterprises to minimize the impact of the new government requirements.

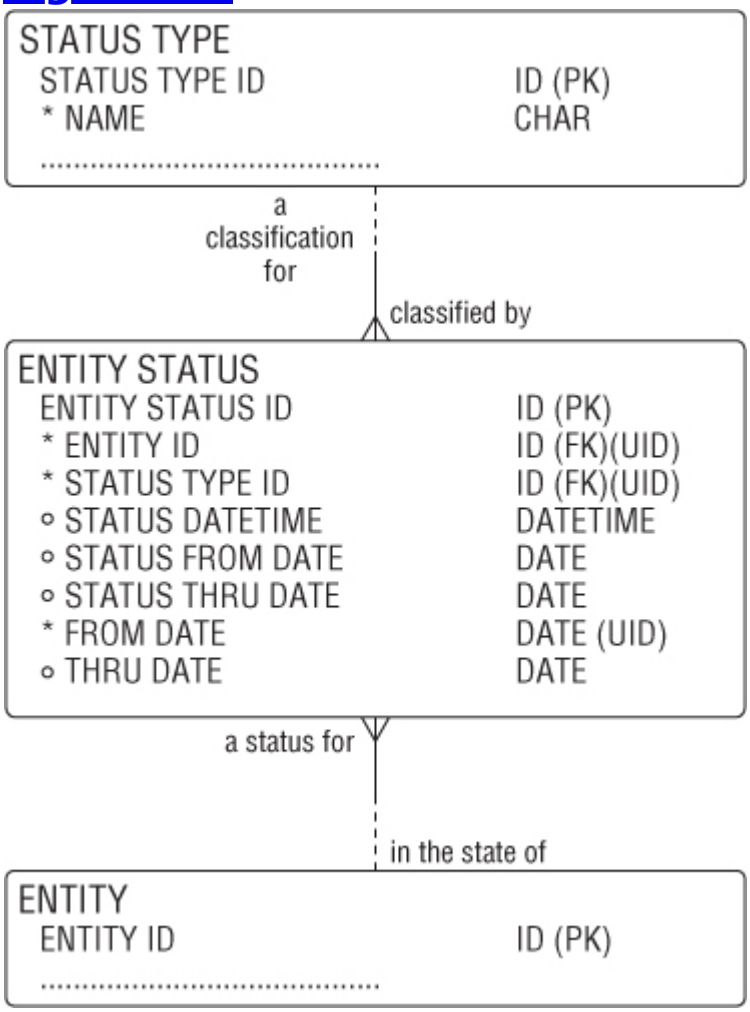
This pattern also addresses the needs of enterprises that wish to capture the history of different statuses for an entity and allows entities to have more than one status at the same time.

How Does This Pattern Work?

Figure 6.6 describes a more flexible status pattern. This pattern, like the Level 2 Status Pattern, Current Status, uses the STATUS TYPE entity to capture all of the different statuses an entity can have. Unlike the previous pattern, this pattern assumes that an ENTITY may have more than one status, and this pattern may also capture all of the statuses that ENTITY has had. This many-to-many

relationship between STATUS TYPE and ENTITY is resolved by the addition of the ENTITY STATUS entity.

Figure 6.6 Level 3 Status Pattern



In the Level 3 Status Pattern, types of statuses are more flexibly maintained as instances of ENTITY STATUS. “Each ENTITY *may be* **in the state of** one or more ENTITY STATUS(es), and each ENTITY STATUS *may be* **classified by** one and only one STATUS TYPE.” This allows new STATUS TYPE(s) to be easily added and any number of status types to be recorded for an ENTITY, without changing the data model.

ENTITY STATUS uses the optional attributes **status datetime** or **status from date** and **status thru date** to

capture different types of time components related to the status. Therefore, this pattern allows for statuses that occur at a specific point in time, such as “Order Cancelled,” to be recorded with the **status datetime** attribute, as well as statuses that require a date range with the **status from date** and **status thru date** attributes (for example, the order was open from a certain date and through another date). Remember that it is possible that a status does not have a time component, such as statuses like “In Abeyance” or “Overdue” where there is no need to say when they became in the state of “Abeyance” or “Overdue.”

Note

In this pattern, there could be many alternatives regarding the time-related attributes. For example, the model could include attributes of **status date** or, alternatively, **status datetime** if a date and time is needed. For the ‘range in time’ status attributes, the pattern could be either **status from date** or **status from datetime** (and the same for the **status thru date**) depending on your needs. This also applies to the **from date** and **thru date** attributes.

Similar to all the other associative entities in this book, this pattern includes the attributes **from date** and **thru date** to capture when an instance of an ENTITY STATUS is effective from and effective through. These attributes have a very different meaning than the **status datetime**, **status from date**, and **status thru date** attributes. The **from date** and **thru date** attributes capture when a status was enacted or set and when it is effective until, while the other ‘status date’ attributes capture information about the status itself. For example, if we are recording an expected shipment scheduled date of “Jan. 5, 2010 2 p.m.” in a SHIPMENT STATUS entity, then there may be a STATUS TYPE of “scheduled” for a SHIPMENT STATUS that has a **status datetime** of “Jan. 5, 2010 2 p.m.” However, this instance was enacted or set when it was discovered, which may have

been on Jan. 1, 2010 and therefore the **from date** of this instance would be Jan 1, 2010 and the instance would be effective up until the **thru date.** If on Jan. 4, the expected shipment date changed to Jan. 7, 2010 4 p.m. then the **thru date** of the first instance would be recorded as “Jan 4, 2010” and a new instance would be created that has a **from date** of “Jan 4, 2010” with a **status datetime** of “Jan. 7, 2010 4 p.m.” signifying the new expected shipment date.

The **from date** and **thru date** are also used for ENTITY STATUS instances that do not have a status time component, such as the need to record a **shipment overdue** STATUS TYPE without any associated date for the status, such as specifying how long it was overdue. The **from date** would still record the date that the shipment first was recorded as being overdue and the **from date** would record when the shipment was no longer marked as overdue. Similar to other associative entities, the **from date** (or **from datetime**, if needed) attribute is part of the unique identifier (UID). The **status datetime** (or **status date)** is not part of the UID as it is possible for a status not to have a time component, such as “Overdue” indicators.

This section continues with the scenario of the large mobile phone manufacturer. Imagine a new requirement has been asked for after the data professional presented the previous pattern to the IT steering committee. A member of the compliance department was on the steering committee and raised the point that a new policy was being enacted that requires orders over a certain amount of money must be held until the credit worthiness of the customer was ascertained. The compliance team knew that other additional processes would be needed, but those processes had not been finalized. The data professional was asked to create a model that would support the current known set of statuses and the new, yet unknown statuses; this model would minimize the impact of that change.

By using the Level 3 Status Pattern as a template and after interviewing key staff members, the data professional created Figure 6.7, which shows that “each ORDER *may be* **in the state of** one or more ORDER STATUS.” The ORDER STATUS entity cross-references an ORDER with a STATUS TYPE, and thus, the **name** of a particular status could be looked up via the foreign key **status type id** to STATUS TYPE. The STATUS TYPE **name** attribute maintains the values for the **status type id**s such as “Order Opened” and “Order Closed.” So, when the head of compliance in this scenario stated that a new status would need to be added to show if an order was under credit review, this status would be added as an instance of STATUS TYPE with a **name** of “Credit Hold”, and specific ORDER(s) could have any number of status types related to them.

Figure 6.7 Example of using a Level 3 Status Pattern

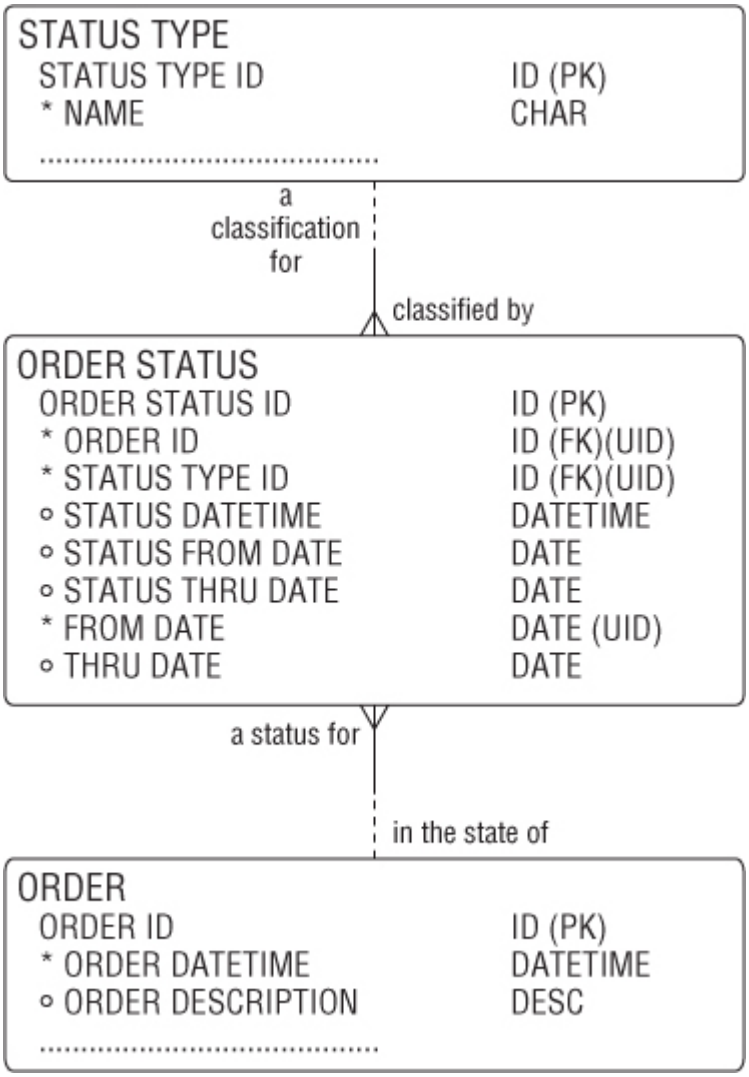


Table 6.3 further illustrates how this pattern works. Order "12560" for "Deluxe Mobile Phone" had six different statuses while it flowed through the order fulfillment process. On Feb. 1, 2010, at 2 p.m., there was a phone notification from the customer about the intention to place a specific order for many Deluxe Mobile Phones. Thus, for the ORDER "12560," there was an ORDER STATUS with a **status datetime** of "Feb. 1, 2010, 2 p.m." with the STATUS TYPE of "Order opened." The "12560" order was then received and entered at the same time ("Feb. 2, 2010, 2 p.m.); this may have been because the order was submitted via the Internet. Therefore, this order had three different ORDER STATUS instances of "Order Opened," "Order Received," and

“Order Entered,” all with the same **status datetime** of “Feb. 2, 2010, 2 p.m.” Unlike what happens with the Level 2 Pattern, one status does not replace another status, and they are all able to exist at the same time. On “Feb. 3, 2010,” the order was put on “Credit Hold” by the credit department. The order was for a large enough value that it needed credit checking. This was the new process the compliance stakeholder was talking about at the IT steering committee meeting. The order had a “Credit Hold” STATUS TYPE from Feb. 3 (with a **status from date** of “Feb. 3, 2010”) until Feb. 11(with a **status thru date** of “Feb. 11, 2010.”) The “Credit Hold” status type is a range status showing the complete length of time that the order was being credit checked. Finally, on “Feb. 12, 2010, 9 a.m.” order “12560” was closed, and one new status was set, “Order Closed.”

Table 6.3 Example of Using the Level 3 Status Pattern

ORDER. ORDER ID	ORDER. ORDER DESCRIPTION	ORDER STATUS. STATUS DATETIME	ORDER STATUS. STATUS FROM DATE	ORDER STATUS. STATUS THRU DATE	STATUS TYPE. STATUS TYPE ID	STATUS TYPE. NAME
12560	Deluxe Mobile Phone	Feb. 1, 2010, 2 p.m.			10	Order Opened
12560	Deluxe Mobile Phone	Feb. 2, 2010, 2 p.m.			55	Order Received
12560	Deluxe Mobile Phone	Feb. 2, 2010, 2 p.m.			20	Order Entered
12560	Deluxe Mobile Phone		Feb. 3, 2010	Feb. 11, 2010	25	Credit Hold
12560	Deluxe Mobile Phone	Feb. 12, 2010, 9 a.m.			60	Order Closed
23000	Mobile Phone Standard	Feb. 2, 2010			10	Order Opened
23000	Mobile Phone Standard	Feb. 2, 2010, 2 p.m.			55	Order Received
23000	Mobile Phone Standard	Feb. 2, 2010, 6 p.m.			20	Order Entered
23000	Mobile Phone Standard	Feb. 3, 2010, 9 a.m.			30	Order Confirmed
23000	Mobile Phone Standard	Feb. 4, 2010, 2 p.m.			70	Order Cancelled
23000	Mobile Phone Standard	Feb. 14, 2010			60	Order Closed

This illustrates the flexibility and power of this pattern. As processes, rules, regulations, and enterprise change, a data model based upon this pattern can accommodate these changes without the need to change the model. Any number of new types of statuses may emerge such as “Order data needs corrections,” “Order awaiting legal review,” “Order complaint issued,” “Order fulfilled,” and so on. This type of data model structure can adapt and stand up to a great number of changes in statuses for an enterprise.

When Should This Pattern Be Used?

We use this data model pattern:

- **When a flexible solution is needed:** This pattern provides a great deal of flexibility in that any number of possible statuses may be added as STATUS TYPE(s), the ENTITY may have any number of statuses either at the same time or over time, the same status may even occur more than once for an ENTITY (for example, the ORDER may be opened, cancelled, reopened, and cancelled again), and statuses may be recorded either without a time component or with either a status date (and time) or with a range of dates via **status from date** and **status through date** attributes. We have implemented this pattern for many enterprises, and our experience is that it works well for many scenarios because if more new status types are needed over time, the pattern can accommodate these needs.
- **When the statuses of a subject area under examination are not very well defined or known:** There are cases where it is unclear what types of statuses are needed, even in the short term. For example, in the scenario in this section, the compliance stakeholder requested a new status; this was easily

accommodated by adding a new instance of “Credit Hold” for STATUS TYPE.

- **When the enterprise wants to consolidate all statuses for easier management of status types:** For example, all the statuses for an enterprise could be captured within a STATUS TYPE entity and attached to the relevant entity via a relevant ENTITY STATUS entity. This allows for much easier management of status types because they can be used, classified and maintained with the same type of structure.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern does not enforce specific business rules:** With this pattern, the entity can have any number of statuses of the same type or of different types, whether or not this makes sense. For example, an ORDER can have many statuses of “Order Confirmed” even though there may be a business rule that an order is really only confirmed once. A way to handle this situation is to specify the specific rules not in the data model itself, but as an adjunct to the data model, or by using a Business Rules Pattern described in Chapter 8 of this book.
- **This pattern does not accommodate when specific types of statuses have specific attributes and/or relationships:** The pattern maintains all types of statuses in the STATUS TYPE entity with the same attributes and relationships. However, if specific types of statuses required specific attributes and/or relationships, it would be possible to slightly modify the pattern and add a subtype to the STATUS TYPE, much the same way as seen on the right side of Figure 6.5.

- **This pattern is more abstract and more difficult to understand and, therefore, is not as effective for use in gathering or validating requirements:** The pattern does not explicitly show the different statuses that an ENTITY could have, as does the Level 1 Status Pattern and, therefore, is less effective at illustrating the scope required for statuses to nontechnical audiences because the different status types are captured as instances of the STATUS TYPE.

Synopsis

This pattern is significant because it allows for the common scenario where entities can have many statuses either at the same time or over time. The Level 3 Status Pattern addressed this issue by maintaining the relationships between an ENTITY and its different allowable statuses, as represented by STATUS TYPE. This many-to-many relationship is resolved via the associative entity ENTITY STATUS, which maintains the current statuses and historical statuses for ENTITY. In the example, you saw that ORDER STATUS may maintain the date and time of the statuses and what the status is/was via the foreign key to reference the associated STATUS TYPE(s) and its lookup name.

Most of the allowable statuses captured in STATUS TYPE happen either at a given point in time or over a range of dates. You see this with “Order Received” and “Order Opened” statuses for order “12560” in Table 6.3.

This pattern can be used by most enterprises that want a consistent and flexible model for handling statuses. As new status types are discovered they can be added to STATUS TYPE without having to change the underlying data model. Additionally, each ENTITY may have any number of statuses and even have the same status recorded multiple times at different points in time. However, this pattern does not enforce specific business rules. For example, there may be a

rule that a particular status should occur only once (for example, “Failed Credit Check”). These rules would need to be developed in conjunction with the data model. This pattern also assumes that all types of statuses have the same attributes, although if there were different attributes needed for different types of statuses, there could be subtypes of STATUS TYPE, for example, PRODUCT STATUS TYPE, ORDER STATUS TYPE, SHIPMENT STATUS TYPE, and so on.

Level 4 Status Pattern

The Level 4 Status Pattern is even more flexible than any of the preceding patterns in this chapter. We consider it a “plug-and-play” type pattern. What we mean by this is that with this pattern any ENTITY that needs to maintain statuses just needs to plug itself into this structure, without adding any new entities or attributes (except a foreign key). When a new entity needs to capture statuses, all that is necessary is to create a relationship from the ENTITY to STATUS APPLICATION and include a foreign key in the STATUS APPLICATION entity.

Why Do We Need This Pattern?

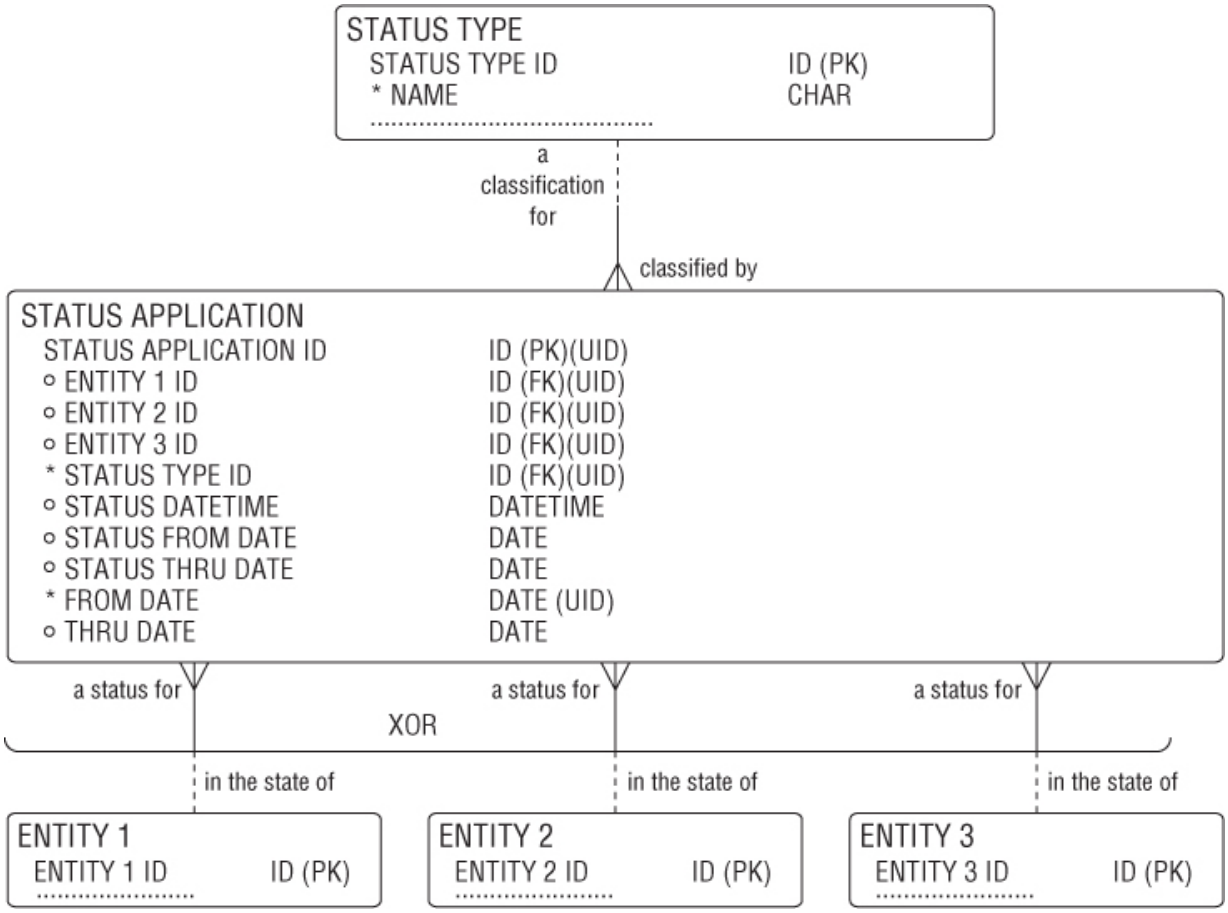
This pattern is quite similar to the Level 3 Pattern, but it differs in one significant way: instead of having an individual ENTITY STATUS connected to STATUS TYPE, a single consolidated entity called STATUS APPLICATION supports all relationships to the STATUS TYPE entity. This is almost the equivalent of an interface in programming. In other words, if any new or existing entity needs to maintain status information, all you need to do is attach to the interface STATUS APPLICATION by creating a relationship to it. This is very useful for enterprises that have a very dynamic data

environment and that want to have a standard, modular approach for maintaining status data.

How Does This Pattern Work?

Figure 6.8 illustrates the pattern. ENTITY 1, ENTITY 2, and ENTITY 3 are the entities that have statuses. Each of these entities connects to the STATUS APPLICATION in the same way. The relationships state that each instance of an ENTITY 1 (or ENTITY 2 or ENTITY 3) *may be* **in the state of** one or more STATUS APPLICATION(s) and one or more STATUS APPLICATION(s) *must be* **a status for** one and only one ENTITY 1 (or ENTITY 2 or ENTITY 3). There is an “exclusive or” (XOR) spanning the relationships from STATUS APPLICATION to each ENTITY. This means that a STATUS APPLICATION *must be* **a status for** one and only one entity, and that the entity may be ENTITY 1 or ENTITY 2 or ENTITY 3. In other words, an instance of STATUS APPLICATION can't be applied to more than one entity at a time, but it must be applied to one entity.

Figure 6.8 Level 4 Status Pattern



"Each STATUS TYPE *may be* **a classification for** one or more STATUS APPLICATION(s), and each STATUS APPLICATION(s) *must be* **classified by** one and only one STATUS TYPE." These relationships are standardizing the many-to-many relationships to STATUS TYPE via the STATUS APPLICATION from all entities. As you can see in the diagram, STATUS APPLICATION carries three non-mandatory foreign keys from the entities (**entity 1 id**, **entity 2 id**, **entity 3 id**). For each instance of STATUS APPLICATION, one of these foreign keys may have a value.

Why would we use this pattern? The answer lies in the fact that most statuses have a very common set of attributes, that is, **status datetime** or **status from date/status thru date**, and **from date/thru date**, and the relationships to STATUS TYPE via an associative entity (ENTITY STATUS in Figure 6.6) are mostly the same. This pattern also assumes

that STATUS TYPE entities such as ORDER STATUS TYPE, SHIPMENT STATUS TYPE, and WORK EFFORT STATUS TYPE have the same attributes. If this is the case, why not consolidate!

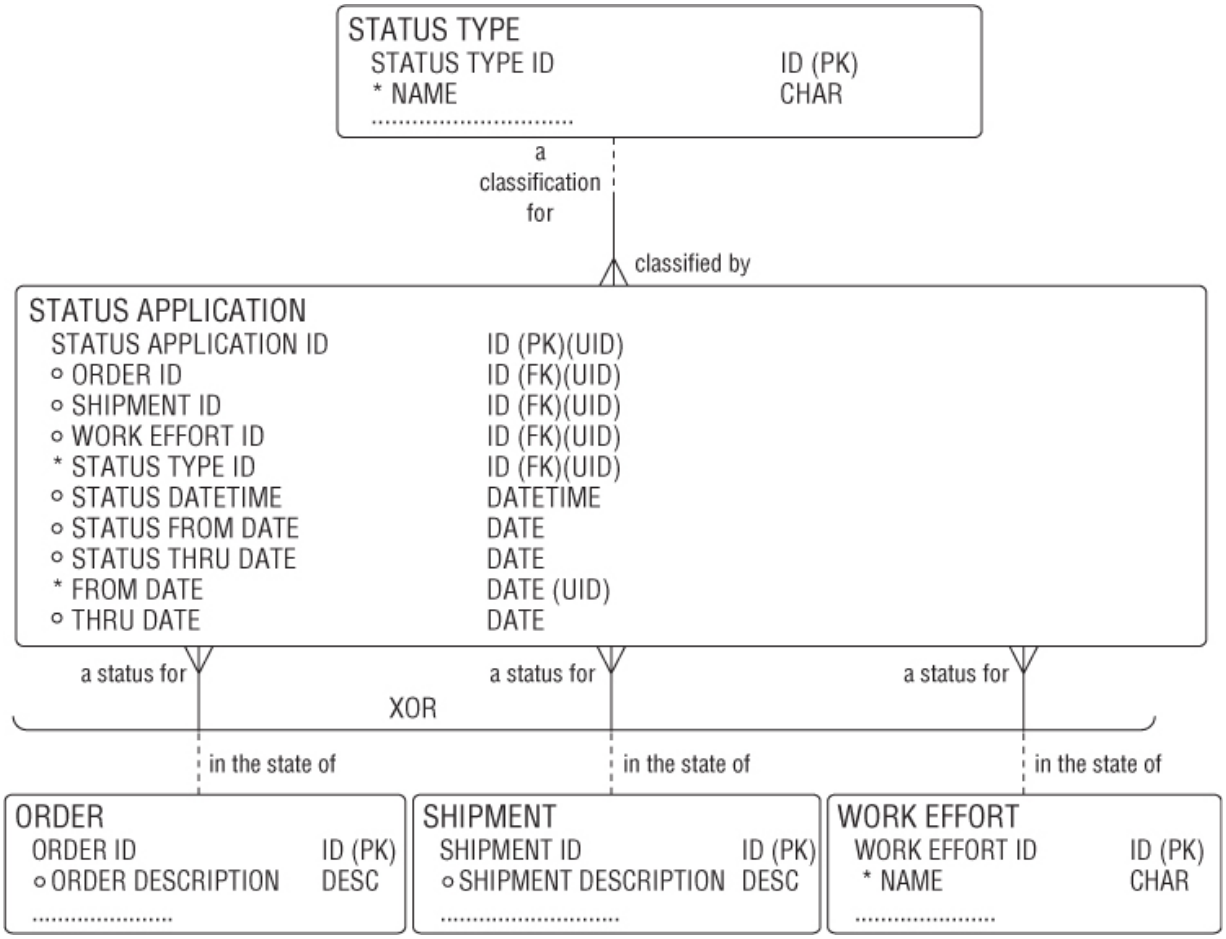
> **Note**
>
> If the different status types have different attributes, you could have subtypes in STATUS TYPE such as WORK EFFORT STATUS TYPE, SHIPMENT STATUS TYPE, and ORDER STATUS TYPE.

By creating this interface entity, STATUS APPLICATION, the data professional simplifies the task of ensuring every entity is connected to the STATUS TYPE in the same way. There is no need to create a new ENTITY STATUS entity for every new ENTITY that needs access to STATUS TYPE, as was seen in Figures 6.6 and 6.7. By connecting to STATUS APPLICATION, ENTITY 1, ENTITY 2, and ENTITY 3 get access to a common associative entity that provides flexibility and consistency as well as the complete set of STATUS TYPE(s) that were defined.

Finally, if the data model/data structure of either the APPLICATION STATUS or STATUS TYPE changes, all of the connected entities are aware of it at once, instead of having to update and make consistent each and every ENTITY STATUS entity.

Figure 6.9 further illustrates how the pattern would work. Continuing with the scenario described in all the previous sections, say the mobile phone manufacturer wanted to create a consistent approach to all statuses across the enterprise as a whole that fits into its master data strategy. The data professional created Figure 6.9 based on the Level 4 Status Pattern.

Figure 6.9 Example of using a Level 4 Status Pattern



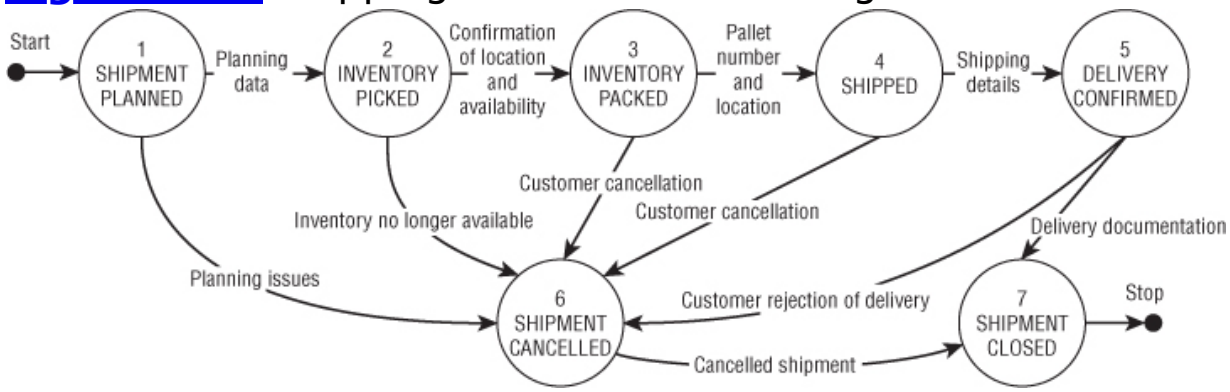
You see from Figure 6.9 that the data professional has taken three different entities and integrated them into the Level 4 Status Pattern: ORDER, SHIPMENT, and WORK EFFORT. ORDER you have seen already from the previous pattern. If you examine Table 6.4, you see it acts in exactly the same way as it did in the Level 3 Status Pattern, except instead of ORDER STATUS **status datetime**, **status from date**, and **status thru date**, the pattern has a STATUS APPLICATION **status datetime**, **status from date**, and **status thru date.** For example, order “12560” for “Deluxe Mobile Phone” has a STATUS TYPE **name** of “Order Opened” with a STATUS APPLICATION **status datetime** of “Feb. 2, 2010, 2 p.m.” and a STATUS TYPE **name** of “Order Closed” with a STATUS APPLICATION **status datetime** of “Feb. 12, 2010, 9 a.m.”

Table 6.4 Example of Using the Level 4 Status Pattern, Order

ORDER. ORDER ID	ORDER. DESCRIPTION	STATUS APPLICATION. STATUS DATETIME	STATUS APPLICATION. STATUS FROM DATE	STATUS APPLICATION. STATUS THRU DATE	STATUS TYPE. STATUS TYPE ID	STATUS TYPE NAME
12560	Deluxe Mobile Phone	Feb. 2, 2010, 2 p.m.			10	Order Opened
12560	Deluxe Mobile Phone	Feb. 2, 2010, 2 p.m.			55	Order Received
12560	Deluxe Mobile Phone	Feb. 2, 2010, 2 p.m.			20	Order Entered
12560	Deluxe Mobile Phone		Feb. 3, 2010	Feb. 11, 2010	25	Credit Hold
12560	Deluxe Mobile Phone	Feb. 3, 2010, 9 a.m.			30	Order Con-firmed
12560	Deluxe Mobile Phone	Feb. 12, 2010, 9 a.m.			60	Order Closed

Now take a look at SHIPMENT. A shipment is “the movement of materials, goods, and/or items that are delivered from one location to another location. The destination and target locations may both belong to the same party or they may each be from different parties.”(8) Each SHIPMENT goes through the complete life cycle for picking, packing, and shipping as described in Figure 6.10. Each of the different circles in this figure represents a state for a shipment. Each translates to an instance of STATUS TYPE.

Figure 6.10 Shipping fulfillment state diagram



If you examine Table 6.5, you see the shipment for the order described previously. The interesting thing to note is that the structure of the table is the same as Table 6.4. The SHIPMENT statuses are handled in the same way as the ORDER statuses. The set of status values that the SHIPMENT uses are different from ORDER, that is “Shipment Planned,” “Inventory Picked,” “Inventory Packed,” “Shipped,” “Delivery Confirmed,” “Shipment Cancelled,” and “Shipment Closed,” but they are accommodated in the same way as the order statuses using STATUS TYPE and then applying them to SHIPMENT(s) via the STATUS APPLICATION entity.

Table 6.5 Example of Using the Level 4 Status Pattern, Shipment

SHIPMENT. SHIPMENT ID	SHIPMENT. DESCRIPTION	STATUS APPLICATION. STATUS DATETIME	STATUS APPLICATION. STATUS FROM DATE	STATUS APPLICATION. STATUS THRU DATE	STATUS TYPE. STATUS TYPE ID	STATUS TYPE. NAME
32	Deluxe Mobile Phones to Telephone Warehouse in London	Feb. 3, 2010, 9 a.m.			100	Shipment Planned
32	Deluxe Mobile Phones to Telephone Warehouse in London	Feb. 4, 2010, 9 a.m.			120	Inventory Picked
32	Deluxe Mobile Phones to Telephone Warehouse in London	Feb. 6, 2010, 3 p.m.			125	Inventory Packed
32	Deluxe Mobile Phones to Telephone Warehouse in London	Feb. 6, 2010, 8 p.m.			300	Shipped
32	Deluxe Mobile Phones to Telephone Warehouse in London	Feb. 8, 2010, 9 a.m.			310	Delivery Confirmed
32	Deluxe Mobile Phones to Telephone Warehouse in London	Feb. 8, 2010, 5 p.m.			500	Shipment Closed

The shipment in Table 6.5 happens to be a shipment for order “12560” described in Table 6.4. This instance of SHIPMENT has a **shipment id** of “32” with the **name** “Deluxe Mobile Phones to Telephone Warehouse in London.” You see that it had a state of “Shipment Planned” on “Feb. 3, 2010, 9 a.m.” The availability of the order was checked in the order confirmation process; now the inventory allocated to the order is picked (the necessary items are pulled from inventory) and the shipment has a state of “Inventory Picked” on “Feb. 4, 2010, 9 a.m.” The shipment is put into crates and the shipment gets a status of “Inventory Packed” on “Feb. 6, 2010, 3 p.m.” This means that the shipment has been packed into crates and put in a shipping container. The shipment left the warehouse on “Feb. 6, 2010, 8 p.m.,” and the shipment attained the status of “Shipped” at that point. It arrived at the customer's warehouse at “Feb. 8, 2010, 9 a.m.” and was electronically signed for by the customer's receiving department. The status of the shipment was set to “Delivery Confirmed” at that point in time. Later that day at 5 p.m., the shipment was closed and got the status of “Shipment Closed” when the delivery personnel confirmed that there was nothing left to do with this shipment.

It is interesting to note that shipping is normally a very time-sensitive process because parties are awaiting their deliveries. Thus, each of the different statuses in this process has a time component that reflects this sensitivity. Also, there was not a shipment status instance that had a ‘range of dates’ using the **status from date** and **status thru date** attributes. The life span of the shipment can be enumerated by the difference between “Shipment Planned” and “Shipment Closed.” One possibility that the data professional may want to suggest is a status type, for example, “Shipment Open,” that uses a range of dates with a **status from date** and **status thru date** for this life span.

Finally, take a look at another data entity, WORK EFFORT. A work effort can be defined as a “planned, in progress, or completed work activity that is performed. It may be the activity related to the fulfillment of a work requirement.”(9) Orders may either be fulfilled by a shipment (usually for products or parts) or may be fulfilled by performing work (usually for services). This entity can be handled in exactly the same manner as both of the previous entities of ORDER and SHIPMENT. Again the structure of Table 6.6 is the same as Table 6.5 and Table 6.4, although the set of statuses is different (that is, “Planning,” “Budget Estimation,” “Committee Review,” “No Go,” “Go”).

Table 6.6 Example of Using the Level 4 Status Pattern, Work Effort

WORK EFFORT. WORK EFFORT ID	WORK EFFORT. NAME	STATUS APPLICATION. STATUS DATETIME	STATUS APPLICATION. STATUS FROM DATE	STATUS APPLICATION. STATUS THRU DATE	STATUS TYPE. STATUS TYPE ID	STATUS TYPE. NAME
904312	Data Warehouse Project	Sept. 3, 2010			1000	Planning
904312	Data Warehouse Project	Oct. 4, 2010			1201	Budget Estimation
904312	Data Warehouse Project	Nov. 4, 2010			1202	Committee Review
904312	Data Warehouse Project				1203	No Go

The work effort “go”/“no go” process is described by Figure 6.11. This process describes how when a work effort is proposed the enterprise decides if the work effort should be continued based on planning, budget estimates, and committee reviews. Each of the circles describes a state that the work effort can be in at each stage of the go/no go process.

Figure 6.11 Work effort go/no go state diagram



You can see from Table 6.6 a proposal for a “Data Warehouse Project” with **work effort id** of “904312.” On “Sept. 3, 2010,” the work effort began “Planning.” Notice that this instance showed only the date and not the time, and thus it did not match the full format of **status datetime** in STATUS APPLICATION. For this specific work effort, it was deemed that the time component was not needed; however, the data model accommodated this by just leaving out the specific time. Thus depending on your conventions, a **datetime** attribute may mean that the time component is optional. After about a month the proposer of the project estimated the cost of the project by Oct. 4, 2010, based on the planning. The work effort “Data Warehouse Project” attained a status of “Budget Estimation” at that point. The budget and plan for the project then were reviewed by an oversight committee. The work effort went under “Committee Review” on “Nov. 4, 2010.” Unfortunately

the committee did not like the budget estimate and cancelled the work effort, thus giving it a “No Go” status. The “No Go” state had no time component at all. All it stated was that the project was considered cancelled and while the team could have entered the date of this “No Go” decision, they decided not to record any time for this status.

What is interesting here is that a completely different set of data, with a different set of statuses, can be handled in the same fashion by the Level 4 Status Pattern. This pattern could be used in place of many other status associative entities (such as PARTY STATUS TYPE, PARTY RELATIONSHIP STATUS, PRODUCT STATUS TYPE, INVENTORY ITEM STATUS, ACCOUNT STATUS, COMMUNICATION EVENT STATUS, INVOICE STATUS TYPE, PAYMENT STATUS TYPE, and many other status types that may be applicable depending on the circumstances).

When Should This Pattern Be Used?

We use this data model pattern:

- **When an enterprise has made a commitment to create very flexible data models that will more easily accommodate changes to the data model:** It is important to use this type of model only if the enterprise fully understands the value of a consistent “plug-and-play” pattern and is willing to put in the effort with a more generalized model (including gaining buy-in) in order to create a very flexible and consistent data model strategy.
- **When an enterprise wishes to consolidate common entities, attributes, and relationships to facilitate easier management of these objects as a part of their data model strategy:** We have seen more and more enterprises make the conscious decision to model using this approach, which minimizes additions of new entities and attributes as new types of entities

need to record status data or as information needs change.

- **When an enterprise is committed to the creation of a consistent approach to the management of data, and in particular, status information, for example, for enterprises that are committed to data-driven data management strategies:** Usage of this pattern can also facilitate the creation of consistent software, data architecture, and application architecture. It offers a very powerful strategic benefit that can save a great deal of money and time.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **Because there is more flexibility, there is also more "rope" and less enforcement of business rules:** For example, when you attach a new entity to the Level 4 Status Pattern, you get every available status type whether or not it relates to the current entity, and therefore it does not restrict the choices to the statuses applicable to that entity. This means that an entity like WORK EFFORT that normally does not have a "Credit Hold" status would have access to that status. Business rules about which status types are available for an entity can be maintained in addition to the model and built into either a rule layer or the application in order to add restrictions to this pattern.(10) We feel that this is a weakness because most entities don't use every different type of status type, and status types can be very specific to a particular entity, business process, or work flow.
- **Different types of statuses may actually have different information requirements:** For example,

perhaps there is an entity that needs only a datetime status attribute. Also an entity may need to have only one current status and have no need to maintain a history of statuses. So this pattern requires each entity to have available in its data structure all the possible attributes of the pattern, whether or not they are needed.

- **Modeling things in a very generalized fashion gives you more flexibility, but at the cost of less clarity:** Non–data professionals will generally have a hard time understanding this pattern, what data is captured, and where. But if you commit to producing and maintaining state transition diagrams (or finite state machines, or some other description of data flows and state transition) for each set of statuses, this would go a long way toward helping with the understandability that is lost through this type of very generalized model.
- **Consolidating all status information in two entities may lead to performance issues with if the pattern is not implemented correctly:** The idea of having all the statuses' information in just two consolidated entities, STATUS APPLICATION and STATUS TYPE, is a highly generalized model that when implemented could have performance implications because searches for available statuses would always have to be filtered by the particular type of status that is needed at the time (for example, order status, shipment status, and so on). Although performance considerations are not really a part of data modeling, some would argue that it is a consideration to look downstream and make sure that the data model can be practically implemented. If this pattern is implemented correctly in a robust physical architecture, you should have no issues with performance.

Synopsis

In this section we described the Level 4 Status Pattern that is an even more flexible “plug-and-play” pattern and that can be used by an enterprise to automatically connect new entities that need statuses to a very comprehensive status data model structure.

The pattern uses an “interface”-like entity called STATUS APPLICATION, to which any new or existing entity can attach. STATUS TYPE already supports the different types of statuses an entity may need, so by connecting to STATUS APPLICATION, any entity can gain access to a complete set of statuses as well as a comprehensive data structure that accounts for initial and future needs.

This approach has many advantages. It helps enterprises provide a consistent approach across all status types, providing not only a highly consistent data model regarding the handling of statuses, but also allowing an enterprise to benefit by developing consistent processes and architectures that are also in sync with this pattern. It takes the guesswork out of decisions regarding what to do with new entities that need status information. It is a very flexible, stable pattern that can withstand the addition of new status types and additional requirements that arise. Comprehensive status capabilities get automatically included for any entity needing status information. Management of the data model (and database design) is simplified because there are many fewer entities and attributes to manage and keep consistent. If the enterprise decides to change the way it manages statuses and this change affects the data model (such as by always specifying that data and time is needed for all statuses instead of just date), the model can be changed in one place. The disadvantages are that fewer business rules are enforced by the model, it does not maintain different status data requirements for different entities, and there could be

downstream database performance consequences. This pattern does require that an enterprise is committed to making investments in a flexible data architecture.(11)

Another disadvantage is the lack of understandability. If an enterprise is using a generalized model like this, we recommend that the enterprise make a strong commitment to supplementing the data model with other documentation, such as state transition diagrams, instance diagrams, and/or worked examples. Also, this pattern does not maintain different attributes, relationships and/or rules for different status types.

Status Category Pattern

In the previous sections we referenced different state diagrams that showed basic states for ORDER (Figure 6.1), SHIPMENT (Figure 6.10), and WORK EFFORT (Figure 6.11). Each of these different diagrams illustrated a flow where the different states were connected to each other. Not only were the states connected to each other but they were also members of a finite set. In other words, each state may have been directly related to one or more other states, but all of the states were contained in a grouping (aggregation or hierarchy) of states. For example, “Order Received,” “Order Entered,” “Order Confirmed,” “Order Cancelled,” and “Order Closed” were all members of an order fulfillment process. However, it is possible that orders have other processes in addition to the order fulfillment process. For example, orders may have an order scheduling process with status types of “On Schedule,” “Behind Schedule,” and “Overdue”. In fact, orders (or any entity) may have many different processes supporting them and thus may have different sets, or categories of status types. The Status Category Pattern captures these and other types of status categories.(12)

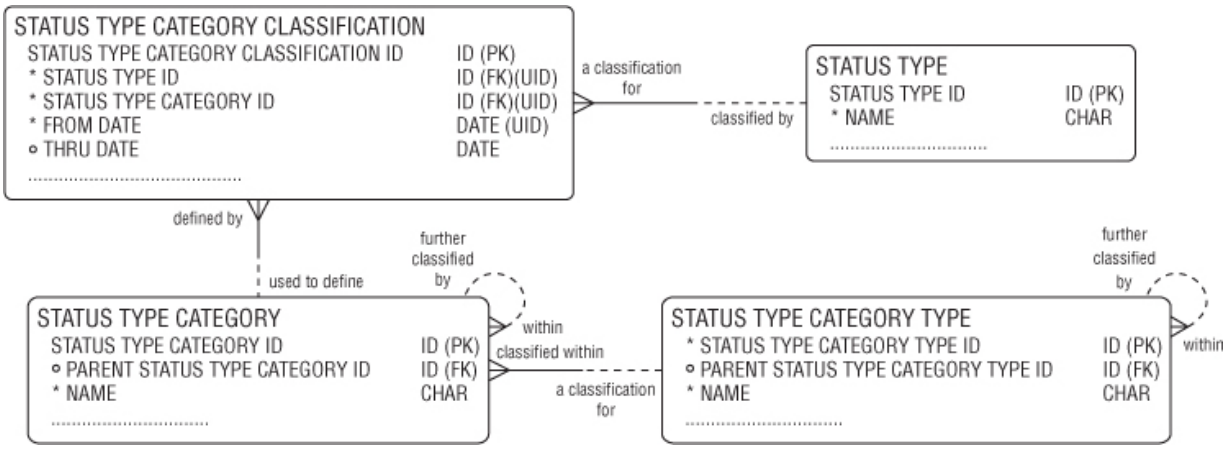
Why Do We Need This Pattern?

We often find that there are multiple sets of status types, and therefore, it is important to classify the status types into their appropriate sets. This pattern supports the need for a status type to have one or more sets of status classifications that may be applied to it.

How Does This Pattern Work?

Figure 6.12 illustrates how we create the categorization for statuses. This pattern applies the Level 3 Classification Pattern from chapter 5 and can be used to enhance the level 2, level 3, or level 4 status patterns. Thus, with this addition, any status type may be classified any number of ways. The figure shows that "Each STATUS TYPE *may be* **classified by** one or more STATUS TYPE CATEGORY CLASSIFICATION(s) and each STATUS TYPE CATEGORY CLASSIFICATION(s) *may be* **defined by** one and only one STATUS TYPE CATEGORY." For example, a SHIPMENT may have STATUS TYPE(s) of "Shipment Planned," "Inventory Picked," and so on. These STATUS TYPE(s) may be classified as members of a "Shipping Fulfillment" STATUS TYPE CATEGORY. Shipments may also have other STATUS TYPE(s) of "Scheduled," "Shipped," "In Route," and "Cancelled." These statuses may refer to a categorization of "Shipping Transportation" statuses, which is another STATUS TYPE CATEGORY.

Figure 6.12 Status Category Pattern



You can also aggregate both the “Shipping Fulfillment” and “Shipping Transportation” into a “Shipping Status” STATUS TYPE CATEGORY by using the recursive relationship “each STATUS TYPE CATEGORY *may be* **further classified by** one or more STATUS TYPE CATEGORY(s).” This can be useful when reporting on all of the statuses a shipment (or set of shipments) has.

The STATUS TYPE CATEGORY can also be classified by STATUS TYPE CATEGORY TYPE. For example, you might classify “Order Fulfillment” and “Shipping Fulfillment” status categories as a “Transaction Processing” STATUS TYPE CATEGORY TYPE and classify other status types into a STATUS TYPE CATEGORY TYPE of “Reference data status.” This can become important when managing master reference data. It may be convenient for you to manage, maintain, and report on similar categories of statuses. You must also be careful not to confuse the difference between a STATUS TYPE CATEGORY such as a “Shipping Status” and creating a STATUS TYPE CATEGORY TYPE classification like “Transaction Processing.” The STATUS TYPE CATEGORY instances are directly applied to the STATUS TYPE(s) and are used to used to classify these statuses; the STATUS TYPE CATEGORY TYPE is used to create very general classifications for the STATUS TYPE CATEGORY(s) that are not directly applied to status types.

Note

As we mentioned, the Status Category Pattern is an example of using a classification pattern to enhance the status patterns. This illustrates the power of using patterns. You may use the patterns from each of the chapters and sections to build your data model to meet your specific needs. Instead of applying the Level 3 Classification Pattern to the status patterns you may alternatively consider using the Level 3 Classification Pattern with Multi Rollups and Schemes, the Level 2 Classification Pattern, or even the Level 1 Classification Pattern to meet your specific needs.

In Table 6.7 you can see how the Status Category Pattern supports the grouping of all of the statuses seen in the state diagrams for ORDER (Figure 6.1), SHIPMENT (Figure 6.10), and WORK EFFORT (Figure 6.11). For example, you see that the “Order Opened” status has a **status type category id** of “5000,” which corresponds to the “Order Fulfillment” status category. Not all of the order statuses are members of this category. For example, you can also see “On Schedule,” “Behind Schedule,” and “Overdue” that are members of the “Order Schedule” status category. What is also interesting is that the “Order Confirmed” status type was also a member of the “Order Schedule” status category. This is very common in complex processes. Often the same status may affect or be affected by different processes. Hence, “Order Confirmed” may be categorized into “Order Fulfillment” and “Order Schedule.” This is supported in the pattern by the associative entity STATUS TYPE CATEGORY CLASSIFICATION that allows a STATUS TYPE to be classified into many STATUS TYPE CATEGORY(s) (and, of course, allows many STATUS TYPE(s) within a STATUS TYPE CATEGORY as well).

Table 6.7 Example of Categorization of Status Types

STATUS TYPE. STATUS ID	STATUS TYPE. NAME	STATUS TYPE CATEGORY. STATUS TYPE CATEGORY ID	STATUS TYPE CATEGORY. NAME	STATUS TYPE CATEGORY TYPE.NAME
10	Order Opened	5000	Order Fulfillment	Transaction Status
55	Order Received	5000	Order Fulfillment	Transaction Status
20	Order Entered	5000	Order Fulfillment	Transaction Status
25	Credit Hold	5000	Order Fulfillment	Transaction Status
30	Order Confirmed	5000	Order Fulfillment	Transaction Status
60	Order Closed	5000	Order Fulfillment	Transaction Status
30	Order Confirmed	6000	Order Schedule	Transaction Status
71	On Schedule	6000	Order Schedule	Transaction Status
72	Behind Schedule	6000	Order Schedule	Transaction Status
74	Overdue	6000	Order Schedule	Transaction Status
100	Shipment Planned	7000	Shipping Fulfillment	Transaction Status
120	Inventory Picked	7000	Shipping Fulfillment	Transaction Status
125	Inventory Packed	7000	Shipping Fulfillment	Transaction Status
300	Shipped	7000	Shipping Fulfillment	Transaction Status
310	Delivery Confirmed	7000	Shipping Fulfillment	Transaction Status
500	Shipment Closed	7000	Shipping Fulfillment	Transaction Status

STATUS TYPE. STATUS ID	STATUS TYPE. NAME	STATUS TYPE CATEGORY. STATUS TYPE CATEGORY ID	STATUS TYPE CATEGORY. NAME	STATUS TYPE CATEGORY TYPE.NAME
1000	Planning	8000	Work Effort Go/No Go	Transaction Status
1201	Budget Estimation	8000	Work Effort Go/No Go	Transaction Status
1202	Committee Review	8000	Work Effort Go/No Go	Transaction Status
1203	No Go	8000	Work Effort Go/No Go	Transaction Status
1204	Go	8000	Work Effort Go/No Go	Transaction Status
15001	Alive	8000	Person Life Status	Reference Data Status
15002	Deceased	8000	Person Life Status	Reference Data Status
26001	Product Sales Support Discontinued	8000	Product Support Status	Reference Data Status
26002	Product Technical Support Discontinued	8000	Product Support Status	Reference Data Status

If you look at the “Shipment Planned” status, it is a member of “Shipping Fulfillment,” as are all of the other shipping statuses. This categorization can be very useful for reporting on all shipping activity. Instead of accessing each individual status, you can access them as a group by using “Shipping Fulfillment.”

All the statuses related to the go/no-go work effort evaluation process, described in Figure 6.11 were grouped into the “Work Effort Go/No Go” status category.

While most of the rows in Table 6.7 maintain status types that are “transaction status” STATUS TYPE CATEGORY

TYPE(s), the last four rows of Table 6.7 show just a few “Reference Data Status” STATUS TYPE CATEGORY TYPE(s), illustrating that this pattern may be used to capture all the statuses of the enterprise.

When Should This Pattern Be Used?

We use this pattern:

- **When an enterprise wants to enhance the Level 2, Level 3, or Level 4 Status Patterns, when an enterprise needs to categorize the different status types and status type categories:** This pattern provides the capabilities to classify status types any number of ways. Of course, some enterprises don't have the need to manage their statuses in this fashion.
- **When an enterprise needs a convenient way to report on the different status classifications that exist and are used across entities, processes, or work flows:** It makes it easier to report on all related statuses by breaking down the statuses from the highest-level status classifications to lower-level status classifications.
- **When an enterprise wishes to manage its data on statuses in a very comprehensive manner:** An important step toward data management of statuses is to begin to categorize statuses into manageable sets or clusters that are related to each other in some way. Each of the statuses referenced in this section may also be members of a set that are the available statuses for a process.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are:

- **It does not effectively illustrate the specific data needs regarding what types of status classifications are required:** It says only that status types can be classified into any number of categories and that there may be any number of different ways to categorize the status types and status type categories. This allows data modelers a lazy way to “catch all” the different status categories that could exist for a status type without having to examine which categories are needed and what the requirements are for each categorization. Different categories may have different rules or attributes associated with them. These need to be analyzed and understood even if you model using this pattern.
- **This pattern does not specify different attributes or relationships for specific STATUS TYPE CATEGORY(s):** If this is needed, you could use the Level 2 Classification Pattern instead of the Level 3 Classification Pattern.

Synopsis

This pattern is significant because it describes how STATUS TYPE(s) may be members of one or more different categories. For example an “Order Confirmed” status type may be a member of an “Order Fulfillment” STATUS TYPE CATEGORY. The same “Order Confirmed” status may also be a member of an “Order Schedule” STATUS TYPE CATEGORY.

It can be very useful for an enterprise to categorize its statuses, and we have shown how to apply the Level 3 Classification Pattern from Chapter 5 to this status pattern. This categorization is supported by a recursive relationship for STATUS TYPE CATEGORY entity as well as by the recursive relationship for STATUS TYPE CATEGORY TYPE, thus allowing categories to roll up and down. This provides a comprehensive and convenient way for enterprises to report

and manage their different status types. This pattern can be an important step toward an enterprise-wide approach to effectively managing statuses, such as in a master data management program.

This pattern does not effectively illustrate the specific data needs of the different classifications of STATUS TYPE, and because it is generalized, it is not very effective as a tool to gather and validate data requirements with non–data modelers. We strongly recommend that if you use this pattern, you use additional documentation such as transition diagrams, finite state machine diagrams, instance diagrams, and worked examples to show the different classifications of the STATUS TYPE(s) and to verify that there is clear understanding of how this pattern would work. The pattern also assumes that the different categories have the same attributes and relationships.

Status Type with Multi Rollup and Rules Pattern

As the business process of enterprises become more complicated, data can travel many paths through an enterprise and may be required to go through various statuses, rules, phase gates, or audit regulations that require data to behave in a particular way. Regulatory requirements such as Sarbanes-Oxley may require that a piece of data should never be in a particular state without having passed through other processes, for example, that budgeting information should not be in a status of “Approved” before it passes though a state of “Regulatory Compliance Review Completed.” A regulation or requirement may require that one state is mutually exclusive of another state, or in other words, one status excludes another status. For example, a balance sheet with

many general ledger account balances may have a status of “Audited” or “Failed Audit Compliance” but never both of these at the same time; thus, one of these states must be excluded if another state is current. It can also be the case that one status has superseded another status over time, for example, a status of “Complied” may be superseded by a status of “Passed Compliance.”

These business rules are often captured in procedure guides, or passed down only through the experience from one employee to the next (the worst case scenario). Sometimes they are captured as process models. For some advanced organizations they may even captured in metadata repositories, wiki pages and implemented in a rules engine. These business rules may also be captured and supported in the data model. Capturing these rules in the data model has many benefits (and some drawbacks) as you will see.

Why Do We Need This Pattern?

It is the purpose of this pattern to capture rules that govern how statuses are related to each other, such as how one possible status type is related to another status type, within the context of a status type category. For example, can an “Open Order” status and an “Order Closed” status both be current at the same time in the framework of “Order Fulfillment” status type category, or is there some rule that says that this is not allowed?! The pattern has to be flexible enough to allow for multiple status types, allow for multiple different status type categories, describe different ways to classify the relationships between statuses (other than status type category) and rich enough in structure to allow for different rules.

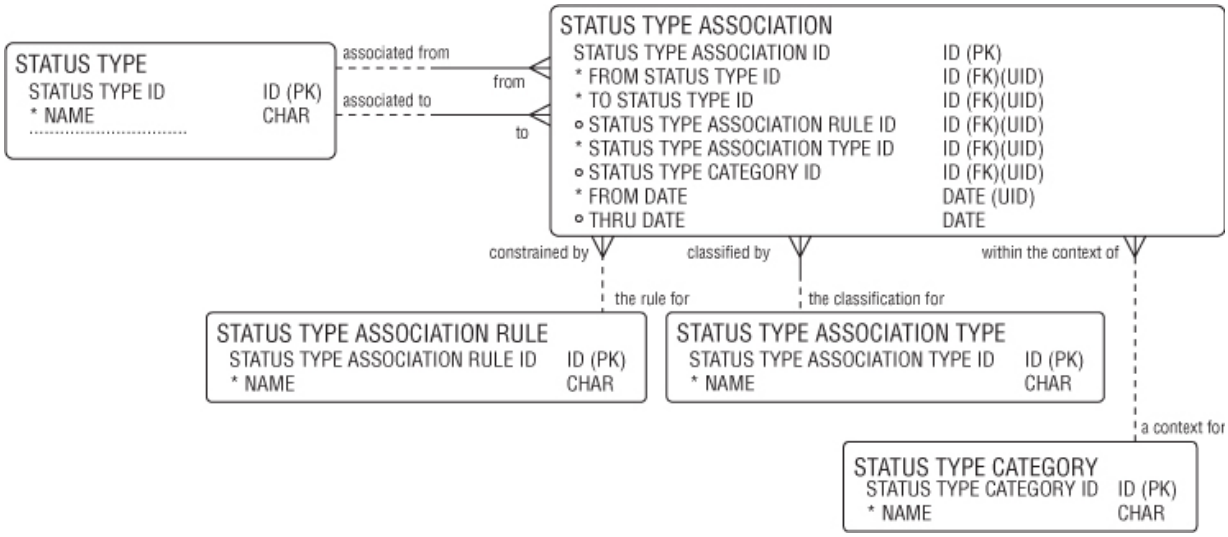
Within this chapter you saw three different state transition diagrams that described Order fulfillment (Figure 6.1), Shipment fulfillment (Figure 6.10), and Work effort go/no go

(Figure 6.11). All of the different statuses were represented as circles in each of the diagrams. In the previous section of this chapter (“Status Category Pattern”), we described how you can classify each of the different statuses into their specific categories. For example, in Table 6.7 “Order Opened” was classified by the STATUS TYPE CATEGORY **name** “Order Fulfillment.” In other words, we captured the fact that “Order Opened” was classified as an “Order Fulfillment” status type category. But what about the association (or transition) you saw between “Order Opened” and “Order Received” and the association between “Order Received” and “Order Entered” and so on in Figure 6.1? Each of the different statuses may be related to each other, as we see with the different associations between “Order Opened” and “Order Received” and so on. You may need to capture these associations. It is also possible that you need to capture the context in which these associations exist. In other words, within what context (or contexts) is an “Order Opened” status related to “Order Entered” status? Or put another way, you want to know that “Order Opened” is associated with “Order Entered” within the context of the “Order Fulfillment.” To do this you must be able to associate one STATUS TYPE to another STATUS TYPE within the context of a STATUS TYPE CATEGORY.

How Does This Pattern Work?

Figure 6.13 illustrates the entities, key attributes, and relationships used to support the need to capture the relationships between different STATUS TYPE(s), the rules about the relationships between STATUS TYPE(s), the STATUS TYPE ASSOCIATION, and if the association is within the context of a STATUS TYPE CATEGORY.

Figure 6.13 Status Type with Multi Rollup and Rules Pattern



To allow for multiple different peer-to-peer associations, aggregations, or hierarchies for which statuses can be members, a many-to-many recursive relationship exists for STATUS TYPE.(12) The STATUS TYPE ASSOCIATION entity in this pattern resolves the many-to-many relationship between STATUS TYPE(s). The **from status type id** and **to status type id** are the foreign keys, each representing an instance of a STATUS TYPE. Instead of using a convention such as **parent status type id** and **child status type id**, we use **from status type id** and **to status type id** as foreign keys because we are allowing for not only parent-child relationships in this pattern, but also many other relationship types. For example, one status may be a peer of another status.

Each STATUS TYPE ASSOCIATION may have a foreign key from STATUS TYPE ASSOCIATION RULE, **status type association rule id.** Also each STATUS TYPE ASSOCIATION has a foreign key from a classification entity called STATUS TYPE ASSOCIATION TYPE, **status type association type id** (to classify the association into a “peer-to-peer association,” “aggregation,” or “hierarchy”). Finally each STATUS TYPE ASSOCIATION may have a foreign key from an entity, STATUS TYPE CATEGORY, **status type category id**; this is because the association may be within the context of a

categorization of status types. These represent instances of rules and types of associations the STATUS TYPE ASSOCIATION is supporting.

> **Note**
>
> The STATUS TYPE CATEGORY entity in this pattern is the same STATUS TYPE CATEGORY entity we used for the Status Category Pattern in Chapter 5. Our more "Eagle Eyed" readers may have noticed that we used a Level 2 Classification Pattern to classify the STATUS TYPE ASSOCIATION. In other words we explicitly captured STATUS TYPE ASSOCIATION TYPE and STATUS TYPE CATEGORY as classifications of STATUS TYPE ASSOCIATION. We did *not* combine these different classifications together with a Level 3 Classification Pattern.

The STATUS TYPE ASSOCIATION RULE **name** may maintain, but is not limited to, values such as:

- **Precedence** indicates that one status type must occur before another status type. For example, there may be a rule that "Order Entered" must occur before "Order Confirmed."
- **Compatible** indicates that both the status types can exist at the same time and thus the association captured in the STATUS TYPE ASSOCIATION is valid and allowable.
- **Implied** indicates that one status type implies that another status type can be inferred. For example, if there was a status of "Cancelled Sales Order," this may imply that another status of "Order Cancelled" also exists. This rule says that two statuses have similar meanings, but it does not mean they can replace each other. This is useful when trying to map statuses across the enterprise. You can say that the order entry process in China sets a status of "Open Order." This would imply an "Order Opened" status in the European order entry process.

- **Exclusion** indicates that if the first status category exists, then the second status category is not allowed to exist at the same time. This is useful in operational systems. For example, an “In Production” status for a PRODUCT may have an “Exclusion” rule if an “In Development” status currently exists for the PRODUCT. This does NOT mean that every conceivable pair of statuses must have an “Exclusion” rule if they are not governed by another rule. What it means is that if you need to explicitly state that two statuses are incompatible, then you can use this type of rule.
- **Substitution** indicates that the first status category and the second status category can be replacements for each other. Both statuses can be current at the same time. For example, if two processes for shipments in the different countries have “Delivered” and “Shipment Delivered,” they may be substitutes for each other.
- **Obsolescent** indicates that the first status category has been replaced and has been superseded by the second status category. For example, if a status has changed its name, then the old (from) status name can be substituted with the new (to) status. The difference between “Obsolescent” and “Substitution” is that statuses that are substitutes for each other are both currently used. Obsolescence implies that one of the status types is no longer used. This is very useful when you are migrating data from one system to a replacement system.

Each STATUS TYPE CATEGORY has a **name** that provides a context for the association between two STATUS TYPE(s). For example, in Table 6.8 you see STATUS TYPE CATEGORY of “Order Fulfillment” as described in Figure 6.1. Thus, a rule of “Compatible” that exists between the “Order Opened” and “Order Received” statuses may be valid within the context of a STATUS TYPE CATEGORY of “Order Fulfillment,” but it

may not be valid within the context of another STATUS TYPE CATEGORY of “Order Schedule.” However, there may be a different rule in that context.

Table 6.8 Example, of Status Type With Multi Rollup and Rules Pattern

STATUS TYPE ASSOCIATION. FROM STATUS TYPE ID	STATUS TYPE. NAME (FROM)	STATUS TYPE ASSOCIATION. TO STATUS TYPE ID	STATUS TYPE. NAME (TO)	STATUS TYPE CATEGORY. NAME	STATUS TYPE ASSOCIATION RULE. NAME	STATUS TYPE ASSOCIATION TYPE. NAME
20	Order Entered	30	Order Confirmed	Order Fulfillment	Precedence	Peer-to-Peer Association
10	Order Opened	55	Order Received	Order Fulfillment	Compatible	Peer-to-Peer Association
55	Order Received	20	Order Entered	Order Fulfillment	Compatible	Peer-to-Peer Association
20	Order Entered	30	Order Confirmed	Order Fulfillment	Compatible	Peer-to-Peer Association
30	Order Confirmed	60	Order Closed	Order Fulfillment	Compatible	Peer-to-Peer Association
55	Order Received	70	Order Cancelled	Order Fulfillment	Compatible	Peer-to-Peer Association
20	Order Entered	70	Order Cancelled	Order Fulfillment	Compatible	Peer-to-Peer Association

STATUS TYPE ASSOCIATION. FROM STATUS TYPE ID	STATUS TYPE. NAME (FROM)	STATUS TYPE ASSOCIATION. TO STATUS TYPE ID	STATUS TYPE. NAME (TO)	STATUS TYPE CATEGORY. NAME	STATUS TYPE ASSOCIATION RULE. NAME	STATUS TYPE ASSOCIATION TYPE. NAME
30	Order Confirmed	70	Order Cancelled	Order Fulfillment	Compatible	Peer-to-Peer Association
70	Order Cancelled	102	Order Revenue Recorded	Order Fulfillment	Exclusion	Peer-to-Peer Association
70	Order Cancelled	37	Order Cancelled Due to Backordered Inventory	Order Fulfillment	Exclusion	Aggregation
88	Open Order	10	Order Opened	Order Fulfillment	Obsolescent	Peer-to-Peer Association
77	Cancelled Purchase Order	70	Order Cancelled	Order Fulfillment	Implied	Peer-to-Peer Association
20	Order Entered	66	Order Inputted	Order Fulfillment	Substitution	Peer-to-Peer Association

Each STATUS TYPE ASSOCIATION TYPE has a **name** that provides a special classification for the association between two STATUS TYPE(s). This classification type describes the "structure" of the STATUS TYPE ASSOCIATION. For example, in Table 6.8 you see STATUS TYPE ASSOCIATION TYPE of "Peer-to-Peer" for the most part and "Aggregation" in one instance (four rows from the bottom of the table). How you manage the different associations between STATUS TYPE(s) depends on their type of structure. For example, if several STATUS TYPE(s) are in a "hierarchy," deleting the top or owning status type, means you should delete all of the other STATUS TYPE(s) also. For more information on this topic please refer to Chapter 4, of this book.

Because this pattern supports rules with STATUS TYPE ASSOCIATION RULE, you can define the behavior of the relationship between two associated status types. For example, in the first row of Table 6.8, "Order Entered" and "Order Confirmed" within the context of the "Order Fulfillment" work flow, have a rule of "Precedence" meaning that the "Order Entered" status type must precede the "Order Confirmed" status. The fourth row of this table shows that "Order Entered" and "Order Confirmed" are also "Compatible" (along with many other pairs of status types that are also compatible). This means that the association between "Order Entered" and "Order Confirmed" status types may exist within the "Order Fulfillment" set of statuses, and thus the association captured in the STATUS TYPE ASSOCIATION is valid and allowable within the context of "Order Fulfillment." Some other interesting associations *not* seen in the state diagram described in Figure 6.1 are also captured by this pattern. Imagine that the data professional working at the large mobile phone manufacturer has interviewed some subject matter experts about the order fulfillment process and discovered some very interesting rules and associations between the different

status types that were not captured as part of the state diagrams. For example, a subject matter expert described how the financial reporting system takes data about cancelled orders from the order entry systems and reports on it. The financial reporting system refers to the “Order Cancelled” state as “Cancelled Purchase Order.” So, “Cancelled Purchase Order” is the status the financial reporting system uses. The data professional can capture this important relationship in the pattern. You can see this in the second to last row of Table 6.8 where “Cancelled Purchase Order” is associated with “Order Cancelled” within the context of “Order Fulfillment” as an “Implied” rule.

Another subject matter expert mentioned that the previous order entry process used the status “Open Order,” which was replaced by “Order Opened” in the current process. Many of the older staff still referred to the old status. Table 6.8 captures the fact that “Open Order” is obsolete and has been replaced by “Order Opened.”

A third subject matter expert mentioned that the order fulfillment process in Asia was the same as it is described in Figure 6.1 except they use different names for the status types. For example, in the last row of Table 6.8 we see that “Order Entered” may be substituted for “Order Inputted”.

Another subject matter expert found out that the “Order Cancelled” status types could not exist at the same time as an “Order Revenue Recorded” state. In other words, you can't cancel an order and still have revenue recorded for it. This relationship pertained to concerns about misstating revenue and possibly being out of compliance. Accordingly, in the fifth row from the bottom in Table 6.8 you can see that “Order Cancelled” has an “Exclusion” rule to “Order Revenue Recorded.”

Finally, the enterprise has instituted a business rule that for the “Order Fulfillment” category of statuses, the status type of “Order Cancelled Due to Backordered Inventory” is

not allowed as a child of “Order Cancelled” and, therefore, has a rule of “Exclusion.” In this case, the enterprise decided that it will not allow this status within the order fulfillment process and will always wait for the inventory so that it will not lose any orders for this reason, although this status type may be applicable in another status category type. This rule is an “aggregation” STATUS TYPE ASSOCIATION TYPE because it involves a parent-child aggregation relationship between these two status types. You might say, “If the association between status types is not allowed, then just don't capture it!” This is true, but from a data management perspective, managing exclusions (or any other rule that governs behavior) by omission is not as rigorous as explicitly defining the behavior of a relationship by a rule.

You might also ask, “Aren't rules just types of associations? Why have separate rules entities?” In our opinion, there is a valid reason for making a distinction between the rules and types of associations. The STATUS TYPE ASSOCIATION TYPE categorizes the association between status types, for example, “Peer-to-Peer Association,” “Hierarchy,” or “Aggregation.” The STATUS TYPE ASSOCIATION RULE defines how the association between the two different status types behaves. A second reason for separating the rules from types is that it is a good policy to make sure that a rule that governs a behavior between different statuses is maintained in an entity that is distinct from an entity maintaining possible classifications of the association, such as “Peer-to-peer association.” This is similar to the idea expressed by Ron Ross when he stated in general that “rules should exist independently of procedures and workflows.”(13)

Note

An alternative model to Figure 6.13 could be to have a many-to-many relationship between STATUS TYPE ASSOCIATION and a

RULE TYPE resolved by an associative entity of STATUS TYPE ASSOCIATION RULE (which also may be related to and within the context of a STATUS TYPE CATEGORY). This would allow the same STATUS ASSOCIATION instance to be related to many rules. For example, there could be a STATUS ASSOCIATION between an “Ordered Entered” and an “Order Confirmed” status that has (is related to) RULE TYPE(s) of “Precedence” and “Compatible.”

This pattern includes a way to model some of the information that is in a state diagram as instances of STATUS TYPE ASSOCIATION. This is quite useful from a data management and a metadata management perspective. Many state diagrams get published as diagrams, but the logic that is shown gets implemented only via the application architecture or a rules engine. This is fine, but there is also the possibility to capture the relationships between states at the data level. This allows for the documentation of rules in a data structure that can be modified and added to by changing or adding additional instances of STATUS TYPE ASSOCIATION. This also means that rules engines and applications may be more ‘data driven.’ As you know, it is easier to change the value of a piece of data in a database than to change lines of application code.

When Should This Pattern Be Used?

This pattern is quite complex, but very versatile, and should be used:

- **When an enterprise wishes to maintain rules about status types in order to have more rigorous data management:** The rules capability of this pattern allows an enterprise to explicitly define the behavior between the status types. If the intention of the enterprise is to maintain information about statuses in data structures as opposed to putting them within

applications, this pattern allows you to build intelligence into the data and better manage status data.

- **For more mature data enterprises that are committed to making an investment in data management:** A certain amount of discipline is needed to use this pattern. This pattern is intended for enterprises that have a strong intention and commitment to be data-driven.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern is complex:** It is easy even for experienced data professionals to have difficulty understanding this pattern. The implementation and use of this pattern should ideally be within the context of a broader data management program. It requires training and understanding for programmers and application architects to use this pattern, and commitment from the enterprise to invest in an integrated data management strategy. That said, once this pattern is understood and utilized, it is a very powerful tool for management of statuses.
- **Rules may be already maintained in a metadata repository or rules engine:** If your enterprise is already using a metadata repository or rules engine, then these types of rules may be alternatively maintained there.

Synopsis

In this section you examined how the associations between different instances of STATUS TYPE may be managed and described by rules. This pattern allows status types to be associated with each other in many different ways and

allows for new association rules and association types to be added over time. This pattern is significant because it supports the ability to manage various ways that status types are or may be related to each other within the context of a status type category. For example, these rules can maintain “Exclusion” associations and can ensure that associations are “Compatible,” among many other rules, depending on the process in which the context of the status exists. This is a powerful tool to explicitly manage data about status categories.

This pattern is complex and requires commitment from an enterprise. This pattern is complex and requires commitment from an enterprise. However, it is an incredibly powerful way to manage status data in a data-driven manner.

Summary of Patterns

Table 6.9 contains a synopsis of all the patterns covered in this chapter.

Table 6.9 Synopsis of the Patterns

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 1 Status Pattern	Defines, describes, and supports maintaining statuses very specifically as "event" attributes in the specified entity.	When there is static set of statuses and new statuses are not expected to emerge for the entity. When there are not a large number of statuses. When there is a need for a relatively simple model to gather and validate the data requirements about statuses. When the data professional wishes to show the differences in perception in an enterprise about statuses.	When flexibility is needed in the model, for instance, having the ability to add statuses as needed without changing the model. When you need to manage the rules between statuses. If there are a large number of statuses, it may be more difficult to manage all the various attributes (and potential changes to the attributes) in a consistent fashion. When you need multiple different classifications of the statuses for an entity.	It is an easy way to start modeling basic business data requirements and to illuminate common terminology and issues with statuses. It specifically shows the allowable statuses as attributes of an entity.	It is not a flexible model, and if new statuses emerge, new attributes need to be added and thus the data model needs to be changed. Does not handle status classifications or rules between status types.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 2 Status Pattern, Current Status	For the situation where an enterprise wishes to flexibly maintain any number of different allowable types of statuses an entity may have, but wishes to capture only the current status at any given time.	When the enterprise wishes to capture only the current status of an entity and wishes to capture the different types of statuses an entity may have. An enterprise does not care about capturing the history of statuses that an entity had. When there is a need to provide for any number of types of possible statuses that may emerge over time.	If an entity can have more than just one status at a given time. This is a specific model when the circumstance requires just maintaining one current status. If you wish to capture the history of the statuses an entity had. If you need to capture different types of statuses for different parts of the enterprise.	Fits a specific need very well (when the enterprise only wants to capture a single status at a time for an entity). It captures the allowable status types that an entity may have and allows for additional types of statuses to be added if they emerge.	It is not a very flexible model outside of its specific need. A single status across the enterprise does not accommodate different interpretations of a status by different parts of the enterprise and there is possibly the need to store different status types. Not a very effective way to illustrate, gather, and/or validate the data requirements for statuses. Does not accommodate maintaining the history of statuses for an entity because one status overrides another.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 3 Status Pattern	This is a very flexible pattern for maintaining status data, which supports any number of statuses for an entity, allows for any number of allowable statuses, and maintains various ways to maintain status oriented dates.	When a flexible solution for handling statuses is desired, for example, allowing any number of possible status types, any number of statuses assigned to an entity, and various options regarding the recording of time for the status. If the types of statuses needed are not well defined or known. If the enterprise needs a very consistent model for handling statuses for all entities.	As a part of a scope statement or model for gathering and validating requirements with nontechnical people. If the circumstance requires recording only the current status of an entity. When different status types have different attributes or relationships (although this pattern may be modified by adding subtypes of STATUS TYPE and recording the attributes of each subtype of status). If an even more flexible pattern is needed that further modularizes status, or that requires categories or rules for statuses.	This is a very flexible model for handling statuses and accommodates any number of status types, any number of status assignments, and accommodates additional status types to be added over time if needed. This provides a consistent approach for supporting statuses and status types for all types of entities.	Can be hard to understand for non – data professionals. This pattern does not support specific business rules and allows any number of possibilities for statuses even if there are restrictions. This pattern does not accommodate specific attributes and relationships for specific types of statuses. This is not as effective a pattern for using to define scope or for gathering and validating data requirements about statuses.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 4 Status Pattern	This supports a plug-and-play capability to categorize statuses. When an entity is added to a data model that needs to maintain statuses, all that needs to be done is to attach a relationship to the STATUS APPLICATION, and all of the STATUS TYPES become available to it.	Use this pattern when an enterprise needs a very flexible approach for modeling STATUS(es). When the enterprise wants a very consistent approach toward managing status data. If the enterprise has made a commitment to data management of statuses in a very integrated fashion.	When an enterprise has a very limited view and tactical view of status data. This pattern would be overkill for those enterprises. When there is a need to model specifically in order to help define scope and to gather and validate data requirements regarding statuses. When there is a need to specify particular attributes, relationships of business rules for specific status types.	It is very flexible, allows new statuses to be added over time, and allows various time elements for statuses. It allows consistent management of statuses; it takes the guesswork out of modeling statuses for new entities because all statuses use the same modeling structure. It allows data driven architecture that facilitates common software, services, and application architecture.	It may be too complicated for some enterprises. This is a difficult pattern to understand because it is a highly generalized model. There are less enforcement of business rules. It is not well suited for defining scope and/or gathering and validating data requirements. We would not show this model to most nontechnical stake holders. It does not provide for differences in attributes and relationships between different status types. (unless you model subtypes for STATUS TYPE). May lead to performance issues.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Status Category Pattern	This pattern classifies the STATUS TYPE(s) in a very flexible manner by applying the Level 3 Classification Pattern from Chapter 5 to the STATUS TYPE entity in this chapter. This pattern can be used to enhance the Level 2, 3, and 4 Patterns in this chapter and will provide comprehensive management of the different status categories.	When an enterprise needs to capture the classification of the different status types in a very flexible fashion. This will help in the management of the status categories. When one needs to enhance the level 2, 3, and 4 status patterns with status type classifications. For master data management efforts where categorization of statuses is important.	When an enterprise has a very limited view and tactical view of status category data. This pattern would be overkill for those enterprises.	This pattern is very flexible and allows any number of status type categorizations. It helps support the reporting and management of status types and status categories. It provides a very comprehensive way to manage statuses as an enterprise-wide level.	May be too complex for some enterprises that are either not as sophisticated in data modeling or that simply do not have the need to be this comprehensive about categorizing status types. The different status types and status type categories must have the same attributes and relationships (unless you create subtypes of STATUS TYPE and STATUS TYPE CATEGORY). Because this is a very generalized model, use of this pattern is not effective for scoping or for gathering and validating requirements about statuses.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Status Type with Multi Rollup and Rules Pattern	This pattern can be used to enhance the Level 2, 3, and 4 Patterns and will provide better management of the different statuses by maintaining the rules for how STATUS TYPE(s) are related to each other.	When an enterprise needs to capture the rules that govern the behavior between different status types and needs to be able to relate status types to each other in many different ways. When an enterprise is committed to a data-driven way to manage status types and the rules that exist between status types. When an enterprise is committed to investing in a more sophisticated data management approach.	When an enterprise less complicated view of status category data. This pattern would be overkill for those enterprises. If an enterprise has limited skills in data management or lacks commitment to making an investment in data management of statuses. When an enterprise already has a meta data environment or has rules engine.	This pattern can enhance the management of status type data by maintaining all the various ways that status types may be related to each other and the rules that govern the behavior between status types. It enhances the Level 2, 3, and 4 Patterns.	Is too complex for some enterprises. Because this is a very generalized model, use of this pattern is not effective for scoping or for gathering and validating requirements about statuses. Rules may already be maintained in a metadata repository or rules engine.

References

[1] See www.wikipedia.org/wiki/State_diagram.

[2] This definition is taken from WordNet (r) 2.0 (August 2003).

[3] This is sometimes characterized in finite state machines as acceptors and recognizers having a binary output. See UML In a Nutshell by Sinan Si Alhir (O'Reilly, September 1998).

[4] This insight was provided by our technical editor (Ed Landale).

[5] We paraphrased this definition from Dictionary.com. See http://dictionary.reference.com/browse/order.

[6] Paraphrased from www.dictionary.com.

[7] For common changes needed in relation to the Sarbanes-Oxley Act, see http://www.sec.gov/divisions/corpfin/faqs/soxact2002.htm.

[8] Please refer to Chapter 5 of The Data Model Resource Book, *Revised Edition, Volume 1, A Library of Universal Data Models for All Enterprises*, By L. Silverston, (Wiley, 2001) for more detailed shipment models.

[9] Please see Chapter 6 of The Data Model Resource Book, *Revised Edition, Volume 1, A Library of Universal Data Models for All Enterprises*, By L. Silverston, (Wiley, 2001) for more detailed work effort models.

[10] Please see Chapter 8 of this book for more information about business rules.

[11] Please look at Chapters 9 and 10 of this book for more information about how an enterprise can leverage the patterns to create a consistent data strategy.

[12] Please see Chapter 4 of this book for more information on hierarchies, aggregations, and peer-to-peer associations and

Chapter 5 of this book for more information about the classification of data.

[13] See Principles of the Business Rules Approach by Ronald G. Ross (Addison-Wesley, 2003).

Chapter 7

Contact Mechanisms: How to Get in Touch

A colleague of ours has two cell phones, one work phone, one home phone, three different email addresses, a homepage, a MySpace account, a BlackBerry, a home address, a work address, a holiday home address, and finally a fax number. Our colleague also has quite a few old numbers, addresses, and email accounts! Each of the different contact mechanisms mentioned may have one or more different purposes, such as the work address being used as a 'Bill to' address or the home phone number being used for a secondary office number. It is not okay to contact this person at his holiday home, as some of his suppliers found out to their detriment. Is this person all that unusual? Maybe 20 years ago this person would be, but now this type of highly connected individual can be seen using his PDA as he takes the ski lift up to the top of the slopes. Imagine now how many contact points enterprises have if an individual can have so many. This chapter addresses the need to capture and manage the different ways a person or an organization can be contacted, what we refer to as *contact mechanisms*.

What Is the Significance of This Type of Pattern?

A major pre-occupation with many businesses is how to track information about addresses, phone numbers, and other forms of contact information. For example, sales applications capture ‘Bill to’ addresses, stock trading applications maintain counter party information, and marketing applications use address information for campaigns and marketing analysis.

Two of the most frequently asked business questions about contact information are:

- What are the addresses, phone numbers, and other contact mechanisms for people or organizations, or in other words, how can they be contacted? We may want to also know the contact mechanisms for other types of entities, such as facilities, orders, shipments, and so on.
- What information is available about each contact mechanism, for example, what is the purpose and usage of each of the different contact mechanisms? You may want to know the phone number that should be used for billing inquiries, the postal address that should be used for returns, or the email address that is specified as the ‘work’ email address.

We believe that there is a great advantage in maintaining all of the different types of contact information (postal addresses, telephone numbers, email addresses, and so on) in a consistent way throughout your enterprise, and the patterns in this chapter can help a great deal toward this. From our experiences working with many enterprises, we have found that contact mechanism information represents a very large percentage of the fields that are maintained in various legacy systems. We have also found that this type of data is maintained in a wide variety of inconsistent formats between databases. If there were a more standard and consistent way of modeling this type of data, we believe that this would result in much easier database maintenance

and much better data quality and would provide much easier access to this data.

What Is in This Chapter?

The first four patterns described in this chapter have similar basic characteristics and answer the two previously mentioned questions about contact mechanisms, namely that:

- Each pattern can support the different ways of knowing the contact information for a party, facility, or any other entity that has contact information associated with it: that is, the various telephone numbers, fax numbers, mobile numbers, postal addresses, email addresses, blog addresses, and other types of contact mechanisms.
- Each pattern has data that is associated with the contact mechanisms. For example, you may want to know the intended specific purpose or reason for that contact mechanism, such as if a contact mechanism should be used for shipment notification. Or you may want to know the intended type of usage for a contact mechanism, for example, if it is for business or personal use.

Each of these characteristics is catered to in different ways, depending on the style of model. For example, the first of the four patterns handles the basic characteristics of contact mechanisms in a very specific manner. Subsequently each of the patterns becomes more and more generalized in their approach. Whatever flavor or level of pattern is chosen depends on the needs of your enterprise and the modeling task that is taking place. It should be noted that dynamic environments require flexible solutions; flexible solutions by their nature are more generalized; and generalized solutions are more difficult to understand.

The last two patterns in this chapter are different from the other patterns in that they cover additional common requirements that many enterprises face when dealing with contact mechanisms.

- First, there is a **geographic boundaries pattern**, which supports the way a postal address may be categorized into physical and logical geographical areas.
- Second, there is a **postal address parts pattern**, which supports the needs of many enterprises to have many different ways to construct the different parts of a postal address, for example, in order to accommodate various international address structures.

Finally, there is a section at the end of the chapter that deals with some other common contact mechanism data (non-solicitation, directions, instructions, and telephone extensions) that can be used to enhance the basic contact mechanism information. This section discusses possibilities for modeling these other aspects of contact mechanisms. They were not added as part of each of the patterns because, in general, we do not view them to be as central in influencing the structure of contact mechanisms in each pattern, but adding them is easy once they are explained.

The data model patterns within this chapter can be used for most enterprises and applications to build consistent data models that support a wide variety of contact mechanism data. This chapter includes:

- A definition and introduction to the concept of a contact mechanism
- The different contact mechanism patterns
- Insights into each pattern
- When to use and not use different contact mechanism patterns
- The relevance of each of the patterns
- A synopsis of all the patterns, pros and cons, and when to use and not use them

What Is a Contact Mechanism?

A *contact mechanism* is an agency or means by which two or more persons, groups (parties), or other item (facility) are placed in communication with each other.(1) This is distinguished from a *device*, which may represent a physical instrument. For example, a telephone number is a contact mechanism because it is the means by which a salesperson and customer can get in contact with each other. The physical telephone is a conduit for the communication, but the means by which the physical telephone is used for communication is the telephone number. The number is the contact mechanism, not the telephone.

Said another way, a contact mechanism is the contact information that is used in order to get in touch with a party (for example, a person or an organization) or another entity (for example, a facility) or that is associated with a transaction (for example, an order, a shipment, or an invoice). Contact mechanisms may include phone numbers, fax numbers, mobile numbers (cell numbers), pager numbers, postal addresses, or email addresses. There are also many other types of contact mechanisms, such as PDA numbers, blog addresses, chat room addresses, Skype name, as well as many methods of contact that are yet to be invented!

A common example of a contact mechanism is a postal address. Modelers sometimes view postal addresses as very separate entities from various telephone numbers or emails because at first glance it seems that there is very different information associated with a postal address compared to other types of contact mechanisms. However, they share a number of common data requirements, for example:

- **They are all methods to get in contact with a party:** Imagine that you can pull up in a system all the

ways to get a hold of a party, including their postal address(es) if you need to write to that party.

- **They are all methods that can be used to authorize a transaction:** Depending on the situation, you may use a phone, fax, email, or postal address for the bill-to address as a means to secure billing authorization so that a party may place an order. On many transactions, it may be necessary to enter some form of contact mechanism (phone, fax, email, postal address, and so on), whether it is for an application, a reservation, an order, or a financial transaction.
- **They all may be used for identification:** It is common to use a variety of contact mechanisms as a means to identity a party. For example, there may be more than one "John Smith" in a system, and the various contact mechanisms (postal address, telephone numbers, and so on) are ways to see if one "John Smith" is actually the same person as another "John Smith."

A postal address, which is a type of contact mechanism, is very different from a FACILITY (that is, a warehouse, a building, factories, rooms, and so on), which is a physical structure that is used to accommodate people or organizations. These facilities may have contact mechanisms, but they *are not* contact mechanisms. For example, a company, XYZ Corporation, is based at a facility, or in other words a site, in Tokyo, Japan, and this facility has a phone number, fax number, and email address. The facility may have its own attributes that are different than that of a contact mechanism. For example, a facility may have attributes maintaining its square area (square footage or square meters) and whether or not it is handicap accessible, and it may have a physical location that has a longitude and latitude, but try putting that on a letter and see if it arrives! The postal address (a type of contact mechanism) has different attributes and is the method by

which something may be contacted via postal mail. Another way to view this concept is that the contact mechanism is a *virtual* method for getting in touch, or in other words, a label that allows someone to be contacted, whereas a facility is a *physical* structure.

Some of the needs in maintaining contact mechanism data that we address in various patterns in this chapter are:

- To support all the contact information associated with parties, orders, facilities, and other entities. For example, it is important that for each entity needing it you are able to maintain telephone numbers, fax numbers, postal addresses, electronic address, and so on.
- To be flexible enough to support new means of contact. For example, in the past couple of years blogs have become a new way to contact someone. “Post something to my blog!” We imagine there will be other ways of contacting parties in the future and it is important to account for this when modeling.
- To allow parties or other entities to have many different contact mechanisms. Most people and organizations have at least a phone number and a postal address; however, in these times, there are often many different ways to get in touch with a party. Likewise, an order may have numerous contact mechanisms, such as a ‘Bill to’ address, ‘Ship to’ address, ‘Shipment notification’ fax number (to alert the customer when their order was shipped), and a ‘Payment follow up’ telephone number (in case there is a need to follow up on the payment that is due).
- To support many ways to classify the contact mechanism. For example, you may need to maintain the type of usages for the contact mechanisms (‘Business’ or ‘Personal’); the purpose(s) of the contact mechanism (‘Bill to’ or ‘Ship to’); the type of contact mechanism

(‘Fax number,’ ‘Mobile number,’ ‘Pager number’); the priority of the contact mechanism (‘Primary,’ ‘Secondary’); the location of a contact mechanism (‘Home,’ ‘Office’); and other classifications that we discuss in this chapter.

Note

Modelers may debate if capturing the ‘Location’ of a contact mechanism, for example, a home phone number, an office phone number, or car phone, is relevant. This is different than the ‘Usage’ (‘Business,’ ‘Personal’) in that the location specifies *where* the contact mechanism is (at their home or at their office). However, this may be of use to your enterprise. For example, you often hear people say, “Don't call your clients at home!” We don't capture the location in the preliminary patterns for two reasons. First, we believe, going forward, that the location of a contact mechanism is less relevant in the new mobile world. Fewer and fewer people have ‘home phones,’ ‘office phones,’ and ‘car phones.’ Second, we think that “Don't call your clients at home!” may mean “Don't call your clients ‘personal’ phone number for ‘business’ reasons.” We take care of this by capturing the usage of a contact mechanism, and the usage may indicate ‘personal’ use or ‘business’ use. Nonetheless, if there is a need to capture the location of the contact mechanism, this is accommodated in the Level 3 and Level 4 Patterns by providing the flexibility to have as many classifications of a contact mechanism as is needed.

Level 1 Contact Mechanism Pattern

A common approach for data models (and data modelers) is to model the specific contact mechanism needs for an enterprise in order to better understand these requirements. This type of approach should show all of the different types of contact mechanisms for an entity, as well as the different classifications of the contact mechanism, such as the purposes, usages, and/or priority of the contact

mechanisms. One option for accommodating this need is to create a very specific data model of contact mechanisms, and you can use the Level 1 Contact Mechanism Pattern as a template for this approach. This pattern will provide a very easy approach to understanding the data requirements to help start creation of a model where each of the contact mechanisms is defined as an attribute of the containing or owning entity. The purpose or usage is explicitly shown in the attribute name, for example, **ship to postal address part 1**, where “postal address” is the type of contact mechanism, “ship to” is the purpose, and “part 1” represents the first line of the address.

As we have pointed out in other chapters, many data modelers find it very difficult to reconcile this style of data modeling with more normalized styles of logical data modeling that may view these attributes as repeating groups (because there could be many “ship to” addresses) that need to be broken out into their own entity in order to allow any number of contact mechanisms. Thus, you may view this type of model as bad modeling that needs to be unlearned and discarded. But is this really the case? Yes, this style of model has flaws and weaknesses that need to be understood; for example, when more contact mechanisms are required or additional types of contact mechanisms are required, the data model needs to be changed to accommodate these needs. However, this type of model also has benefits that should also be understood. They can be useful under some circumstances, for example, if the number and type of contact mechanisms are very stable and unchanging or if you need a means to understand the data requirements by modeling them very specifically. Furthermore, this type of specific modeling appears in many legacy implementations. For this reason, a data professional should know its strengths and weaknesses.

In this type of data modeling the data professional explicitly captures the contact mechanism information as attributes of the entity. For example, a data professional may interview a salesperson and hear her say something like, “We capture the street name, an apartment number, the zip code of the address, the city, the state, an email address, and a contact number. This is enough for us to process the order!” This leads the data professional to capture attributes of **order address part 1**, **order apt address part** (for the apartment number), **order postal code** (for the zip code), **order city**, **order state, order email address,** and **sales contact number.**

But as a data modeler you may ask yourself, “Is this the wrong way to model this because these data requirements may change?” Not always—it states the needs of the salesperson in a very direct and unambiguous way. Is it a complete representation of the data needed for contact mechanisms? It may or may not be; that can only be determined with the analysis of the subject area under investigation by the data modeler. Would it be a good way to implement the model under normal circumstances? Maybe not, but we have seen implementations built in the style of this pattern, so we should know its relative strengths and weaknesses. This type of modeling may be useful as part of a statement of scope where a very specific model is used in order to gather and validate information requirements. However, we usually find that this type of modeling has severe drawbacks regarding its lack of flexibility if it is used as the basis for implementing a database design.

Note

We want to emphasize that caution should be exercised with the use of this level 1 pattern (as is the case with most of the level 1 patterns) because this pattern is not generally an effective foundation for a solid database design. Data models generally

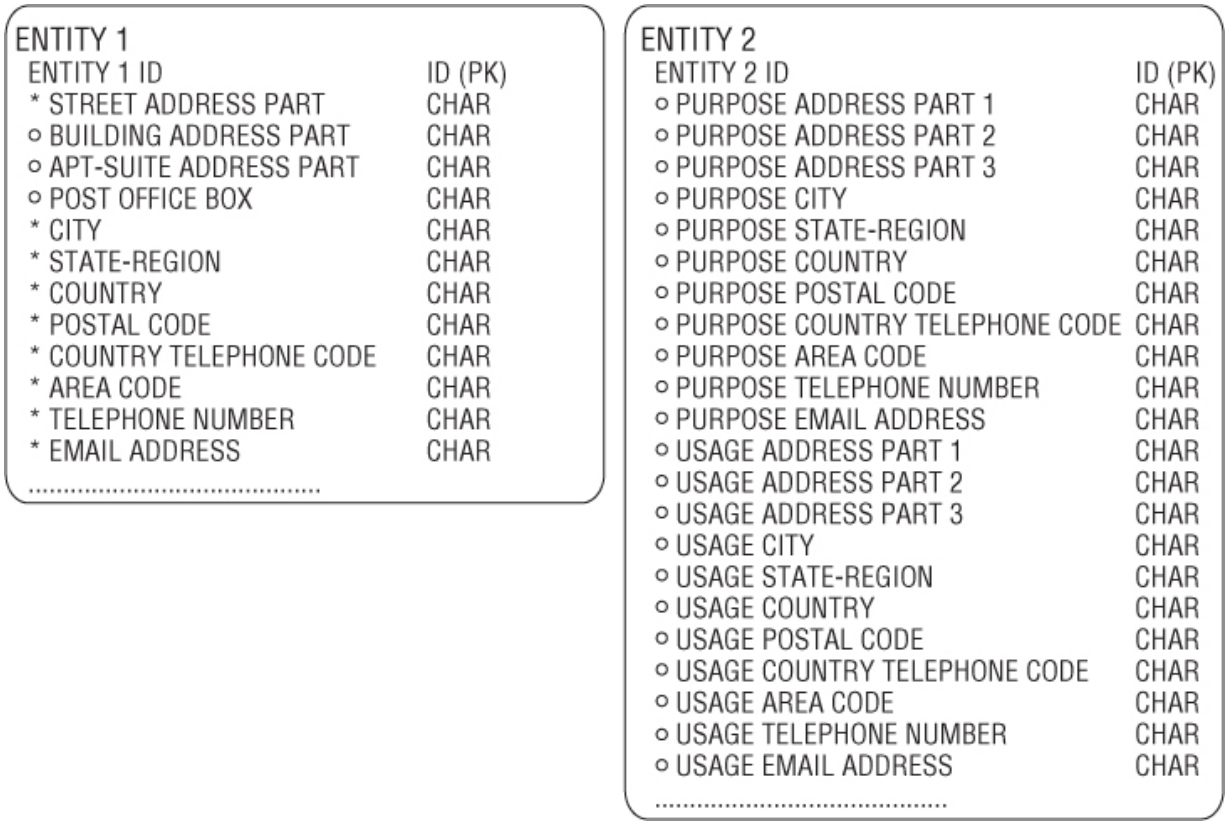
have two purposes: They can be a tool for understanding data requirements, and they also serve as a starting foundation for a database design. This pattern (as with most level 1 patterns) serves the former purpose very well; however, it is usually very ineffective regarding the latter purpose.

Why Do We Need This Pattern?

The reason why contact mechanisms may be modeled using this specific pattern is that enterprises sometimes need a simple, unambiguous way to model and illustrate the data regarding contact mechanisms. There may be situations where this pattern is relevant, such as where there is the need to model only a single postal address, a phone number, and an email address and nothing else regarding contact mechanisms.

The strength of the pattern is that it provides stake holders and subject matter experts a means to determine the scope of data requirements for contact mechanisms in a clear and detailed way. Previously, we mentioned the salesperson who needed to capture the telephone numbers, postal addresses, and email addresses of his or her customers. The Level 1 Contact Mechanism Pattern shown in Figure 7.1 shows each of these contact mechanism types explicitly as attributes that also sometimes include the purpose of the contact mechanism or the usage of the contact mechanism.

Figure 7.1 Level 1 Contact Mechanism Pattern



Finally, this pattern can be a very powerful tool in understanding the differences in perception that an enterprise has for the same or similar concepts. It starts to illuminate the common (and different) terminology that people in an enterprise may use for the same contact mechanism concepts. It also shows the weaknesses of implementing a rigid model if you have a dynamic, changing environment.

How Does This Pattern Work?

Figure 7.1 illustrates a specific way of modeling contact mechanisms. ENTITY 1 and ENTITY 2 represent entities that the data modeler has determined require contact information. ENTITY 1 and ENTITY 2 represent any entity such as PARTY, ORDER, FACILITY, SHIPMENT, CUSTOMER and so on.

Each of these entities contains attributes that maintain the specific types of contact mechanism needed for that entity. For example ENTITY 1 is composed of, but not limited to, **street address part**, **building address part**, **apt-suite address part**, **post office box**, **city**, **state-region**, **postal code**, and **country** that make up the postal address for the entity. Also, in ENTITY 1 you can see **country telephone code**, **area code**, and **telephone number** that make up the full telecommunications number for that entity. Finally, you can see that ENTITY 1 contains the **email address** that supports electronic communications via email. This satisfies the first need of contact mechanism patterns, that is, that each pattern may support various types of contact.

Note

Some modelers may object to the idea of maintaining attributes that have values that may be looked up, such as city, state (or state-region), country, and postal code. These are shown for the sake of simplicity in this diagram for a level 1 pattern. A more normalized model would be to have foreign keys to entities of CITY, STATE, COUNTRY, and POSTAL CODE, as shown in Figure 7.5 (an example of a Level 2 Contact Mechanism Pattern), so that the same instance (for example, the same city of “New York” or the same country of “USA”) can be reused across various addresses. Another alternative would be to have a many-to-many relationship to a GEOGRAPHIC BOUNDARY entity, as illustrated in Figure 7.9 (an example of a Level 4 Contact Mechanism Pattern). We also discuss this possibility later in this chapter in the section “Contact Mechanism Pattern with Geographic Boundary.”

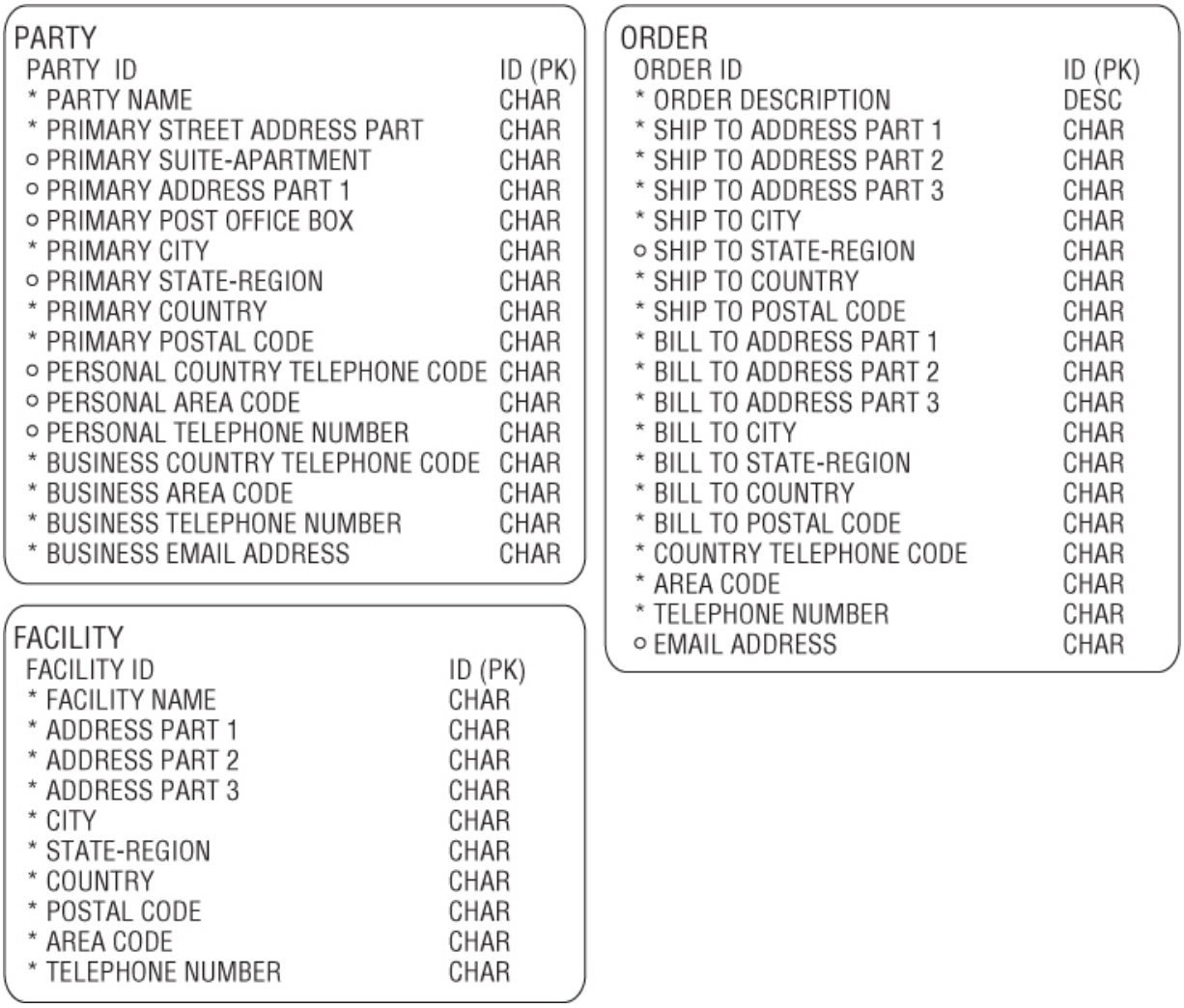
ENTITY 1 does not assign a purpose or usage to any of the attributes for the contact mechanisms. This is perfectly fine in many situations; the purpose or usage may or may not be needed, depending on the entity and the circumstance. In ENTITY 2 you see that different contact mechanisms have a **purpose** or **usage** as part of the attribute name. This **purpose** represents the specific reason that the contact

mechanism is used. For example, the **purpose** may be “Ship to” or “Bill to” for an address, and this results in attribute names such as **ship to country** or **bill to address part 1.** This supports the need that a contact mechanism may have other data associated with it, such as a contact mechanism purpose.

ENTITY 2 also contains usages that represent the common customary use of a contact mechanism. For example, a usage may be business, personal, or even emergency to make attributes like **business email address**, **personal mobile number**, or **emergency phone number.**

Figure 7.2 further illustrates how to employ this pattern. The scenario is as follows: A medium-sized building supply firm named ABC Building Corporation in New York wants to manage the contact information it has for people, organizations, places, and orders. It wishes to buy an off-the-shelf contact management software to accomplish this. ABC Building Corporation has grown very aggressively over the past 5 years with a building boom. It has many domestic (ABC is based in the USA) and international customers. The CEO realizes that they have many different systems where they must look for telephone numbers and contact details for the parties with which they do business. For this reason the CEO wants to know all of the different ways that they contact different people and organizations, the address and telephone numbers of their customers, how to get in touch with various facilities, and finally, any contact information associated with their orders, such as ‘Ship to’ and ‘Bill to’ information.

Figure 7.2 Example of using a Level 1 Contact Mechanism Pattern



Using the data model pattern described in Figure 7.1, based on the analysis of the current systems in place and by conducting detailed interviews with key staff, the data model professional creates Figure 7.2 to model the different contact mechanisms that the contracting firm needs. This diagram could be used as part of a statement of scope of the contact mechanism data requirements and used to help evaluate third-party applications from vendors that are selling contact management systems. The data model professional identified three major data entities on which the building supply company wanted to focus: PARTY (people and organizations), ORDER (a commitment to purchase goods or services), and FACILITY (a physical

structure that is used to accommodate people or organizations).

> **Note**
>
> Though ORDER(s) are defined as “a commitment to purchase goods or services,” there are many different types of orders, such as sales orders (where the customer orders goods and/or services from the enterprise), purchase orders (where the enterprise orders goods and/or services from a supplier), or work orders (where a part of the enterprise orders work to be performed). In this chapter, the examples given are sales orders where customers are ordering from the enterprise (ABC Building Corporation); however, all of the contact mechanism patterns would work for any of these types of orders as well. Please see Chapter 4 on “Ordering Products” in *The Data Model Resource Book*, Volume 1, Revised Edition (Wiley, 2001) for more information on subtypes of Orders.

Based on detailed analysis and interviews with management and secretarial staff, the data professional identified the primary contact address, personal telephone number, business phone number, and business email addresses for the different people and organizations with which the company does business. The data professional also discovered that the company captures domestic and international phone numbers. Therefore, the data professional added **personal country telephone code** and **business country telephone code** for the telephone numbers to accommodate international telephone numbers. It is interesting to note that the data professional identified the usages of “personal” and “business” for the telecommunications numbers. There may be very specific rules around when to use these different numbers. For example, the building supply company may not want its salespeople to use “personal” numbers for sales calls.

The optionality of each of the attributes in PARTY reflected the requirements of the building firm. For example, members of the secretarial staff were very specific about

how they wanted the address of a party constructed. They insisted on having mandatory **primary street address part** and optional **primary suite-apartment.** The reason for this is that they created most of the mass mailing marketing and needed to ensure consistent formatting for those mass mailings.

In Table 7.1, you can see examples of how this entity describes the needs of the enterprise. For example, you see six different sets of contact mechanism information for six different parties. Three of the parties are people: “Nadine Gerard,” “Ed Smith,” and “Manu Collet”; the other three are organizations: “XYZ Corporation,” “Toms Building Materials,” and finally “Lingsat Ltd.” The parties have addresses in various different parts of the world. For example, “Nadine Gerard” has a **primary country** of “USA” and “Toms Building Materials” is in “China.”

Table 7.1 PARTY Contact Mechanisms—Address, Phone Number, and Email

PARTY. PARTY ID, PARTY. NAME[2]	PARTY. PRIMARY STREET ADDRESS PART	PARTY. PRIMARY SUITE-APARTMENT	PARTY. PRIMARY ADDRESS PART 1	PARTY. PRIMARY CITY	PARTY. PRIMARY STATE-REGION[3]	PARTY. PRIMARY COUNTRY	PARTY. PRIMARY POSTAL CODE	PARTY. PERSONAL TELEPHONE NUMBER[4]	PARTY. BUSINESS TELEPHONE NUMBER[4]	PARTY. BUSINESS EMAIL ADDRESS
1001, XYZ Corporation	100 Main Street	Suite 819	The Coalman Building	New York	NY	USA	10019		1 917 555 2100	xyz@ xyzcorp. com
1005, Nadine Gerard	55 Right Road		The Foundry	Los Angeles	CA	USA	90210	1 805 555 4534	1 805 555 3434	ngirard@ ms.com
2002, Ed Smith	55 Charing Cross Road	Flat 4	Shaldon Mansions	London		UK	WC2H 0LA	44 20 5555 1234	44 20 5555 7983	jsmith@ vca.co.uk
2004, Manu Collet	Andheri Kurla Road	Apt. 604	Marol	Mumbai	Andheri East	India	400099	91 11 5555 665	91 11 5555 665	Mcollet@ mtln. net.in
3000, Toms Building Materials	710 Dong Fang Road	Suite 1502	Thompson Commer-cial Bldg	Shanghai	Fudong	China	202100		86 21 555 1001	tom@nails. com
3010, Lingsat Ltd	Dorego 1708			Buenos Aires		Argentina	1412		54 11 4777 1221	noticas@ lingsat. com.ar

Initially, we want to examine the postal address contact mechanism information for PARTY(s) that are people. In Table 7.1, “Manu Collet,” in the fourth row of the table, has a **primary street address part** of “Andheri Kurla Road,” **primary suite-apartment** of “Apt. 604,” **primary address part 1** of “Marol” (a suburb of Mumbai), **primary city** of “Mumbai,” and **primary state-region** of “Andheri East” in a **primary country** of “India” with a **primary postal code** of “400099.” The address contact mechanism was broken into this set of attributes based on the needs specified by the secretarial staff for mass mailings. This structure allows them to format the address information for mailing in a more structured manner—the **primary street address part** is the first part of the mailing label, the second part is the **primary suite-apartment**, and the **primary address part 1** was specified to accommodate other parts to the address, such as a building or suburb (they left room for additional address parts, just in case they needed to add them). This is a very common way to segment address information, as can be seen from Table 7.1. This structure supports the different address contact mechanism needs for many different international addresses. For example, in Table 7.1 you have addresses in India (for “Manu Collet”), The People's Republic of China (for “Toms Building Materials”), USA (for “XYZ Corporation” and “Nadine Gerard”), UK (for “Ed Smith”), and Argentina (for “Lingsat Ltd”).

Note

It's important to remember that the usage of a contact mechanism, such as ‘business’ for a phone number, is not the same as an ‘office’ phone number or a ‘primary’ phone. “Office” refers to a location where a device may be located; ‘primary’ refers to a level of importance tied to the contact mechanism (we address this as another possible classification called a “Priority type” classification in the Level 4 Pattern). For example,

the physical phone is at a place of work. But an ‘office’ phone may be used for ‘business’ and ‘personal’ reasons. We don't capture the location type or priority type classifications until the Level 3 and Level 4 Patterns for reasons we explained earlier in the chapter.

Next, examine the telephone numbers for two different parties, one a person, “Ed Smith,” and the other an organization, “Toms Building Materials,” again in Table 7.1. “Ed Smith” has two telephone numbers. First, a personal telephone number made up of **personal country telephone code**, **personal area code**, and **personal telephone number** (“44 20 5555 1234”). Second, a business telephone number made up of **business country telephone code**, **business area code**, and **business telephone number** (“44 20 5555 7983”). “Toms Building Materials” has only a business telephone number made up of **business country telephone code**, **business area code**, and **business telephone number** (“86 21 555 1001”).

The data model professional did notice that this model would allow any PARTY, including organizations, to have personal phone numbers. He/she then raised the question with the business representatives, “If you're going to have ‘business’ and ‘personal’ as contact mechanism usages, should we state that these apply only to PERSON(s), not ORGANIZATION(s)?” So, the data model pattern illuminated this possible issue with the enterprise view of contact mechanism usages. The business representatives said that they were nearly sure that all formal organizations, such as suppliers and customers, have only ‘Business’ contact mechanisms, but they also have informal organizations, such as certain prominent families in the building trade, that have ‘Personal’ phone numbers and addresses that should not be used for business purposes.

Note

Why have different attributes for the different parts of the phone number in the entity? The reason to split them out for this pattern is twofold:

- First, to record each of these pieces of data independently because each of these may have specific properties; for example, the business country telephone code of 44 corresponds to the UK, and there may be a need to search by country telephone code.
- Second, to ensure that the diagram is an accurate and rigorous statement of scope. The entity specifically states we need to accommodate international phone numbers for PARTY(s), whereas for FACILITY, country telephone codes are not needed.

Finally examine the business email address for “Lingsat Ltd” and “Nadine Gerard.” Both have a business email address, “noticas@lingsat.com.ar” and “ngirard@ms.com,” respectively. Both organization and individual email contact mechanisms can be accommodated in this attribute. The interesting thing about the business email address is that a usage has been tied to the contact mechanism attribute, namely, that this email is used for ‘Business’ related emails. This raises the question of whether there should be another attribute of personal email address.

ABC Building Corporation also had a need to relate contact mechanisms to their orders (as seen in Figure 7.2). After interviewing the accounting staff and operations staff, the data professional identified two significant types of postal addresses related to the ORDER: ‘Ship to’ and ‘Bill to’ addresses, and these represented different purposes for

order-related addresses. The ‘Ship to’ address is the address of the location where an order is to be delivered. Unlike the primary address attributes in the PARTY entity, the ‘Ship to’ address is broken into **ship to address part 1**, **ship to address part 2**, and **ship to address part 3**, as well as **ship to city**, **ship to state-region**, **ship to country**, and **ship to postal code.** The operations staff and accounting staff did not feel that it was necessary to rigorously structure the address into suite-apartment or street address parts. For example, you can see in Table 7.2 an order with **order id** “47742” and **order description** for “Cement” being shipped to “100 Main Street, Suite 819, The Coalman Building” in **ship to city** “New York,” **ship to state-region** “NY” in **ship to country** “USA” at **ship to postal code** “10019.” There is also another order in Table 7.2 with **order id** “47799” and **order description** for “Gold electrical wiring” being shipped to “Andheri Kurla Road, no. 604, Marol” in **ship to city** “Mumbai,” **ship to state-region** “Andheri East” in **ship to country** “India” at **ship to postal code** “400099.”

Table 7.2 ORDER Contact Mechanisms, Ship to Address, Bill to Address, Phone Number, and Email

ORDER. ORDER ID	ORDER. ORDER DESCRIPTION	ORDER.SHIP TO ADDRESS PART 1, 2, 3[4]	ORDER.SHIP TO CITY, SHIP TO STATE-REGION,[3] SHIP TO COUNTRY, SHIP TO POSTAL CODE[4]	ORDER.BILL TO ADDRESS PART 1, 2, 3[4]	ORDER.BILL TO CITY, BILL TO STATE-REGION,[3] BILL TO COUNTRY, BILL TO POSTAL CODE[4]	ORDER.TELEPHONE NUMBER (COUNTRY TELEPHONE CODE, AREA CODE, TELEPHONE NUMBER)[4]	ORDER. EMAIL ADDRESS
47742	Cement	100 Main Street, Suite 819, The Coalman Building	New York, NY, USA, 10019	100 Main Street, Suite 819, The Coalman Building	New York, NY, USA, 10019	1 917 555 2100	Info@ XYZ.com
47790	Steel girders	55 Right Road, The Foundry	Los Angeles, CA, USA, 90210	100 Boulder St, Suite 23	Santa Monica, CA, USA, 90212	11 805 555 8999	
47799	Gold electrical wiring	Andheri Kurla Road, no. 604, Marol	Mumbai, Andheri East, India, 400099	55 Charing Cross Road, Suite 233, Shaldon Mansions	London, UK, WC2H 0LA	44 20 5555 1234	Mcollet@ mtln. net.in

Notice that because the attributes are not as tightly structured as in Table 7.1, there is room for inconsistencies. For example, Order “47790” for “Steel girders” shows that “The Foundry” (which is a building) is used for **ship to address part 2**, yet a suite (“Suite 819”) is maintained in their **ship to address part 2** for the order of “47742”; that means the same attribute is maintaining two different types of data. Furthermore, the last order, “47799”, maintains building information in **bill to address part 3**. This will cause some integration issues, and it is important to capture the fact that different departments (such as the accounting and operations staff) don't have a de facto standard for address structure. Also, notice that some postal address data is abbreviated, such as “NY” (for the state of New York). It is also possible that the same postal address data is not abbreviated. In other words, instead of “NY,” the operations and accounting staff could enter “New York” in the same attribute. This can also cause some data consistency issues, and we discuss other data modeling solutions for capturing these types of geographic boundaries later in this chapter.

Figure 7.2 provides a model that accommodates the accounting and operations departments' data needs regarding contact information. It is very useful to capture the difference in needs so that the data professional can identify integration problems up front. In other words, the format for the contact mechanism data in PARTY is quite different from the format of that data in ORDER, and this can create issues when integrating addresses.

In Figure 7.2 you also see ‘Bill to’ address attributes. The ‘Bill to’ address is the address that is to receive the invoice for goods or services. In Table 7.2, in the case of the order “47742” for “Cement,” there is an instance where the **bill to address** and **ship to address** are exactly the same. However, you can also see from Table 7.2 that the **bill to**

address for the order “47799” for “Gold electrical wiring” is “55 Charing Cross Road, Suite 233, Shaldon Mansions, London, UK, WC2H 0LA,” different from the **ship to address** of “Andheri Kurla Road, no. 604, Marol” in **ship to city** “Mumbai,” **ship to state-region** “Andheri East” in **ship to country** “India” at **ship to postal code** “400099.” This is a very common occurrence in business, that an order is billed to one address yet shipped to another.

Is it true to say that for order “47742” with **order description** “Cement” the ‘Ship to’ and ‘Bill to’ addresses are the same? Looking at it from a purely data standpoint the answer is yes, but in fact if you look at the purpose of the addresses, you can see that the meaning of the addresses is very different. Both of the addresses are related to each other because they are addresses for an instance of an ORDER, but both addresses have a very different significance to different departments in ABC Building Corporation. The operations staff is responsible for the logistics for an order, but may not care about the billing issues. The accounting staff is required to pay the bills; therefore, the ‘Bill to’ address is crucial to them. This raises the issue of keeping the addresses in sync. Again, the pattern helps raise awareness regarding a possible issue with data integration. The same address may be perceived in two different ways by two different parts of the building firm. Who is correct, are both organizations correct, and ultimately, who is responsible for making sure that the address data is up to date, consistent, and correct?

The accounting and operations staff did not specify if the ORDER **country telephone code**, **area code**, and **telephone number** were or were not for a specific purpose. This shows that it is possible that a contact mechanism may or may not have any associated purposes. Enterprises may not know the reason for maintaining some contact mechanisms, and this pattern highlights this

possible gap in knowledge. Not capturing the specific purpose for the ORDER telephone number could also lead to confusion. For example, different people may have different ideas about the purpose for the ORDER telephone number or what number to use for the ORDER telephone number. Someone in operations may use the customer organization's headquarters telephone number, someone in accounting may enter the customer contact number, and someone else may enter the telephone number of the ABC Building Corporation's person that is responsible for supporting the customer with this order. Which type of contact number should be maintained here or are they all allowed? This pattern helps to highlight situations where purposes may be needed to avoid confusion.

Also notice that the order for “Steel girders” does not have an email address associated with it. In Figure 7.2 you see that PARTY must have a **business email address**(es). In the entity ORDER, the **email address** was optional. Different types of contact mechanisms may be optional or mandatory, depending on the needs of that entity and the requirements of the enterprise.

Note

You may ask, “Why only maintain contact mechanisms for an ORDER instead of just relating the ORDER to a party's contact mechanisms?” This could be modeled with a relationship between the PARTY CONTACT MECHANISM (described in the next section) and the ORDER in order to accommodate the contact mechanisms for the ORDER. Yes, it is usually necessary to maintain both the contact mechanism and the parties involved in orders. However, these are two independent facts. The contact mechanisms for an ORDER are a distinct fact from the contact mechanisms for a PARTY. If you relate the ORDER to a PARTY's contact mechanisms, this implies that first you need to set up the PARTY with their contact mechanisms. This is not always done because the ORDER contact mechanism may be provided just for use with that order. It is a highly sensitive issue these days to capture contact mechanisms for a PARTY when the intended purpose was to just use it for that ORDER. This

chapter is dealing with relationships to contact mechanisms, and though the parties involved in the order also need to be modeled, this can be accomplished by adding the contextual role patterns from Chapter 3. By modeling the relationship from contact mechanisms to an order separately from the order's contact mechanisms, you are also allowed the flexibility to sometimes maintain only the contact mechanism for the order and sometimes only the party relationships to the order. For example, there may be a need to just maintain only the person that gave the order without necessarily having to maintain that person's contact information.

Finally, after interviewing the senior management, the data professional discovered, unlike the situation with ORDERS, that there was no significant purpose or usage to any of the contact mechanisms for a FACILITY. The senior managers just wanted to capture a full address and domestic telephone number; no **email address** or international dialing code (**country telephone code**) was needed for FACILITY. For example, in Table 7.3 you see three different **facilities**, “XYZ Corporation head office,” “MS warehouse,” and “Charing building site.” Each may have its own **address part 1**, **address part 2**, **address part 3**, **city**, **state-region**, **country**, and **postal code**.

Table 7.3 FACILITY Contact Mechanisms, Address, Phone Number, and Email

FACILITY. FACILITY ID	FACILITY. FACILITY NAME	FACILITY. ADDRESS PART 1, 2, 3[4]	FACILITY.CITY, STATE-REGION,[3] COUNTRY, POSTAL CODE[4]	FACILITY. TELEPHONE NUMBER[4]
10901	XYZ Corporation head office	100 Main Street, The Coalman Building	New York, NY, USA, 10019	212 555 1234
10903	MS warehouse	55 Right Road, Warehouse, The Foundry	Los Angeles, CA, USA, 90210	805 555 4534
10906	Charing building site	55 Charing Cross Road, Flat 4, Shaldon Mansions	London, UK, WC2H 0LA	020 5555 1234

Senior management stated that they believed that a facility does not have an email address. Is this true? Is there a flaw in this assumption? Some facilities may have an informational email address, such as info@188ludlowstreet.com. The pattern has shown a weakness or inflexibility in the requirements of senior management and a major gap in their understanding!

Each of the facilities had its own telephone number, but with no country telephone code. For example, **facility** “XYZ Corporation head office” has an **area code** of “212” and a **telephone number** of “555 1234.” Senior management believed because the address of the facility was related to the phone number, the address would indicate what the country telephone code for the phone number was. For example, “MS Warehouse” is in **country** “USA”; this means its related facility telephone number must be a United States telephone number and therefore did not have to be explicitly captured. Is this a rigorous way to capture phone numbers? This could lead to issues in that the telephone number could conceivably be for one country and the postal address could be in another country! Of course, having a telephone number in one country and a postal address in

another country may not in fact be an issue at all. Many people or organizations maintain telephone numbers in one country but have moved abroad. The pattern again may have illustrated another difference in understanding between different groups in the same enterprise.

> **Note**
>
> For simplicity, many attributes and relationships that would normally exist for ORDER, FACILITY, and PARTY have been left out. If the reader wishes to see a more definitive set of attributes and relationships for these entities, he or she can refer to Volume 1 of *The Data Model Resource Book*.

When Should This Pattern Be Used?

We use this pattern:

- **When there is a need to understand the data requirements, help define scope and provide a simple way to facilitate discussions about requirements (and issues) with other business representatives** The pattern will show the breadth of the area of where the contact mechanisms are applicable. It helps to gather the common contact mechanisms terminology and to provide a very simple way to start the data modeling effort in order to gain better understanding regarding the needs for contact mechanisms.
- **When there is a well-defined and known set of specific data needs for contact mechanisms** Our building supply firm may have had contact mechanism data spread about three different business subject areas, but perhaps they knew their subject areas well. But beware, needs often inadvertently pop up and/or change!
- **When the contact mechanism attributes of an entity are specific to that entity:** When the contact

mechanism attributes are specific to an entity, there is less of an argument for splitting out the attributes into their own entity, because that specific type of contact mechanism will not be reused in other entities. For example, a “Skype name” may only be applicable for a single entity, such as PERSON. However, this is tricky because things may change and the “Skype name” may be needed for other entities in the future.

- **When it is known that we are dealing with only a few very specific types of contact mechanisms:** This again is an indication of a very limited specific subject area. If there are only a couple of contact mechanisms needed, then this pattern may provide a simple and effective way to capture them. For example, if a specific entity, such as a FACILITY, only needed to maintain one postal address and phone number, then this pattern may be suitable.
- **When showing the different requirements and needs of different stake holders and the different perceptions that different stake holders may have:** For example, in the scenario discussed in the preceding section, the secretarial staff had a more rigorous set of requirements than any other group of stake holders. This pattern raised awareness of the different problems and perceptions that different parts of the enterprise had in relation to contact mechanisms.
- **When contact mechanism entities, attributes, and relationships are static and there are not anticipated future changes:** This means they will rarely, if ever, change, and new contact mechanisms will not be needed in the future. This may or may not have been the case in the preceding scenario, but it's an issue a data professional should consider when using a very specific pattern.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **The pattern is not flexible at all, and it does not stand up well to the addition (or changes) of new contact mechanisms that may emerge over time:** For example, the addition of another contact mechanism type, such as a "Skype name," would require the addition of at least one new attribute to each entity. If additional purposes, usages, or priorities for contact mechanisms are needed, these will also necessitate additional attributes. For example, what if "Installation contact email address" or "Secondary (or tertiary) business phone number" is needed?
- **When there is additional information that needs to be maintained about the contact mechanism:** In addition to the type of contact mechanism, purpose, usage, and other ways to classify it, there are many types of data that may be maintained about a contact mechanism, such as non-solicitation indicator, telephone extension, contact mechanism preference times, and directions (for a postal address). When there is additional data, it is quite awkward to use this pattern because each type of data about the contact mechanism would have to be added as another attribute for each type of contact mechanism. Also, by doing this, you are also breaking normalization principles because the new attributes are really related to the attribute that maintains the contact mechanism.
- **This pattern does not accommodate relating contact mechanisms to other entities (or to itself recursively):** Because this pattern does not maintain contact mechanism data in its own entity, it does not allow relationships from contact mechanisms to other

entities. For example, the same contact mechanism, such as a postal address, may be related to a PARTY as well as to an ORDER, a SHIPMENT, an INVOICE, and other entities. Another example is that contact mechanisms may be linked to other contact mechanisms; for example, a person's fax may be linked to his email so that when he gets a fax it automatically sends him an email.

- **This pattern does not handle multiplicity of attributes very well:** If a person has ten telephone numbers, you should have at least ten different attributes for a PARTY. If each of these telephone numbers has two different purposes, you would have twenty attributes for the entity! So you can see that attributes can multiply rather easily with this pattern. Therefore, when you have a lot of the same type of contact mechanisms for an entity, using this pattern has great disadvantages. For example, when there are many telephone numbers for a PARTY, use of this pattern is violating the first normal form of normalization (repeating groups), and this could cause more and more attributes to be added as new phone numbers are needed.
- **In our experience, basing the database design on this pattern usually results in a poor implementation choice because it is difficult to manage over time:** We have seen a few cases where this pattern has been implemented as is and has satisfied the needs of the enterprise using it. But we have also seen many more implementations of this pattern that have cost large amounts of money, time, and effort to manage or replace.

Synopsis

The Level 1 Contact Mechanism Pattern supports basic aspects for contact mechanisms using a very specific style of data modeling. First, it supports the explicit definition of all of the different contact mechanism types for an entity. In Figure 7.2 you see three different entities: PARTY, ORDER, and FACILITY. Each has postal addresses and telephone numbers. Also, PARTY and ORDER have email addresses. Second, the pattern supports the purpose, and usage of the contact mechanism where applicable. For example, in Figure 7.2 there are ORDER 'Ship to' and 'Bill to' address attributes. The purpose of those contact mechanism attributes was to specify where goods for ORDER(s) should be delivered as well as where the bill should be sent. You also saw that a PARTY had telephone numbers commonly used for 'Personal' and/or 'Business' reasons.

The Level 1 Contact Mechanism Pattern may be used to better understand and validate the specific data requirements. It is very useful at explicitly showing all of the different types (purpose and usage) of contact mechanisms for an area of interest. It provides two useful aids for describing a subject area. First, it provides a clear and unambiguous view of the contact mechanism types for a subject area. Second, it highlights the different ways that the same concept may be viewed by different areas of the business. For example, Figure 7.2 accommodates the PARTY contact mechanism attributes of the secretarial staff, which were very different from the ORDER contact mechanism attributes described by accounting and operations staff. This can be a strong indicator of problems of understanding and consistency in an enterprise. The pattern also helps provide a very easy way to start the modeling process and begins to uncover the needs and terminology that different parts of the enterprise use when talking about contact mechanisms.

The Level 1 Contact Mechanism Pattern can be implemented, and has been, but beware! Usage of this pattern for implementations has several drawbacks: It does not accommodate changes to information requirements very easily, attributes may get repeated redundantly, and there may be data about each contact mechanism that the pattern doesn't easily accommodate. Although there are some exceptions, in our opinion, this pattern should generally not be used as the basis for a database design; however, it can be used effectively as a model to help gather and validate information requirements.

Level 2 Contact Mechanism Pattern

The Level 2 Contact Mechanism Pattern provides a way to create a fairly specific data model to maintain contact mechanism data, yet also provides a more flexible data modeling solution than the Level 1 Contact Mechanism Pattern. This pattern provides a method for handling any number of contact mechanisms, types, purposes, and usages in a flexible manner while modeling each major type of contact mechanism (telecommunications number, electronic address, and postal address) in its own data modeling entities and structures. It is not nearly as specific as the level 1 pattern but it may be even more effective for using as part of a statement of scope or to effectively gather and validate data requirements in situations where the audience is comfortable with some generalization, such as for an audience of data professionals or power users. It may be more effective as part of a statement of scope because it allows the data modeler to illustrate specific relationships.

This pattern still needs to be able to support the fundamental concepts for a contact mechanism, that is, capture all of the different types of contact mechanisms in a subject area and support other data about the contact mechanism, such as its purposes and its usages.

Many data modelers find it easier to use this style of data model than the Level 1 Contact Mechanism Pattern because this pattern uses a more normalized logical data modeling approach. In this pattern, instead of capturing contact mechanisms as attributes, you capture them as entities and relate them to other entities, such as PARTY, ORDER, and FACILITY.

Why Do We Need This Pattern?

This pattern provides the flexibility for many entities (such as PARTY, FACILITY, and ORDER) to maintain as many contact mechanisms that are needed, as well as the various types, purposes, and usages that are needed to specify contact mechanisms. The pattern provides a way to add new contact mechanism types, purposes, or usages without needing to change the model. Also, this pattern provides a way to model different types of contact mechanisms (for example, telecommunication numbers, electronic addresses, and postal addresses) with different data model structures.

How Does This Pattern Work?

Figure 7.3 illustrates the Level 2 Contact Mechanism Pattern. ENTITY represents an area of interest that the data modeler is examining; for example, it could be used to maintain the contact mechanisms of a PARTY or an ORDER as seen in Figures 7.4 and 7.5. This pattern handles the contact mechanisms for the ENTITY in two different ways. Either or

both styles may be used, depending on the data requirements:

- One style captures the contact mechanism, such as a telecommunication numbers and postal addresses, in their own self-contained entities, such as CONTACT MECHANISM 1. For example, a specific contact mechanism, such as an address of “100 Main Street, NY, NY, 10001,” is maintained as its own instance, with its own attributes, and may then be related to many parties (because many parties may have the same postal address). The **contact mechanism data** attribute is how we represent all of the attributes that go to make up a contact mechanism. For example, the street address, the apartment number, a PO box, and so on if the contact mechanism is for a postal address or a country telephone code, area code, and telephone number for a telecommunications number. The other style captures the contact mechanism information, such as an electronic address, as part of a dependent entity of the ENTITY, such as in ENTITY CONTACT MECHANISM 2. In this style, the attribute **contact mechanism data** maintains the actual contact mechanism value(s) (for example, ngirard@xyz.com) in an attribute instead of as its own entity. This is used where there is not the need to relate the same contact mechanism data to many parties (or to many instances of another entity), and thus there is not a need to maintain it in its own entity. This is often the case for emails because email addresses are usually not shared.

Figure 7.3 Level 2 Contact Mechanism Pattern

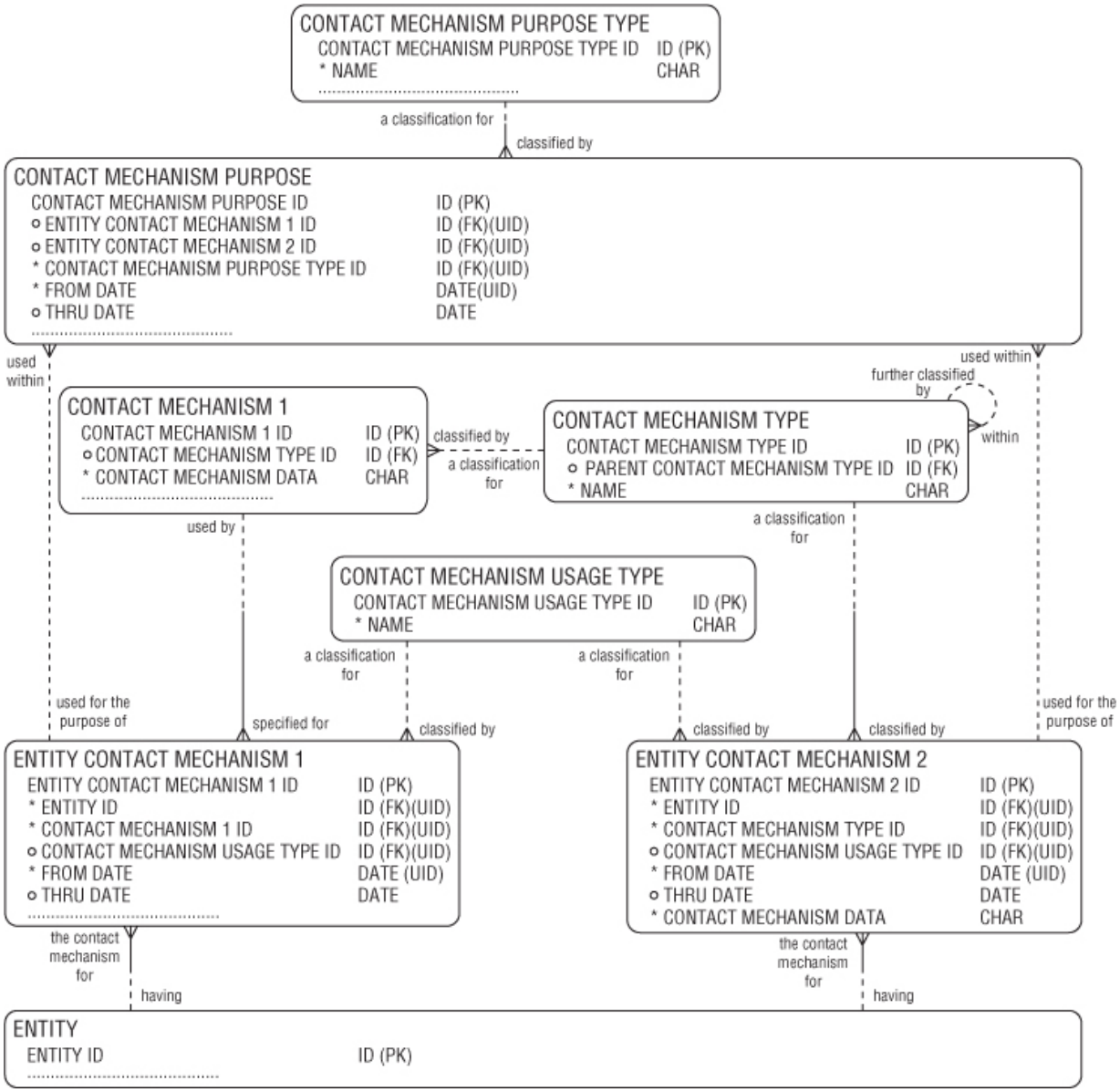


Figure 7.4 PARTY example of using a Level 2 Contact Mechanism Pattern

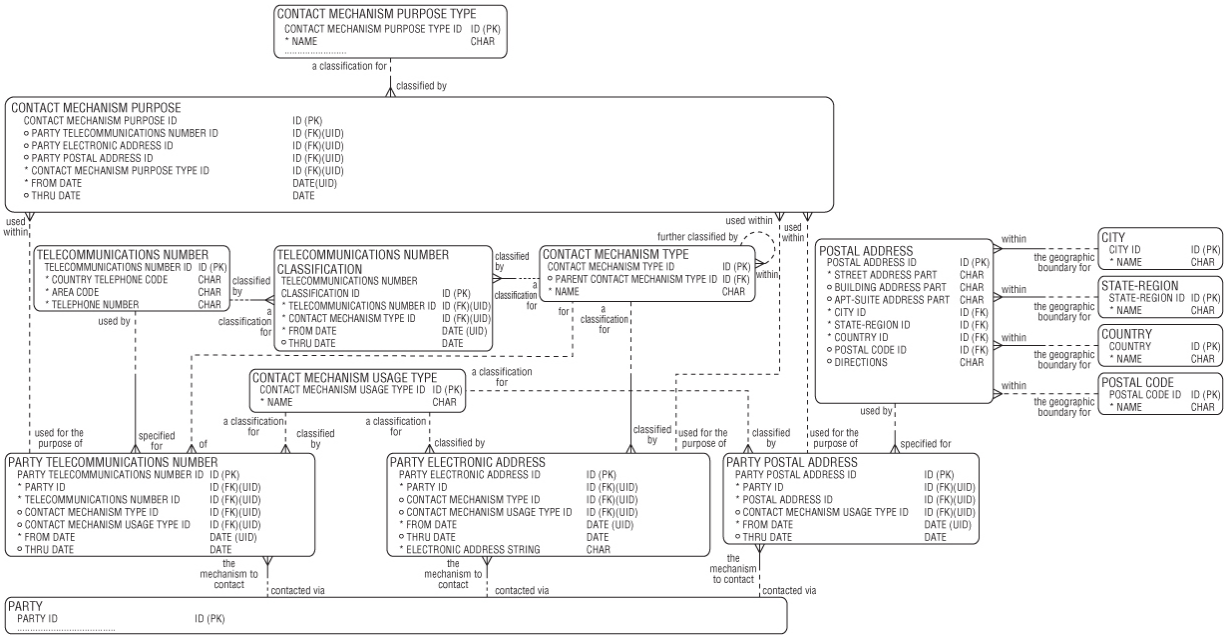
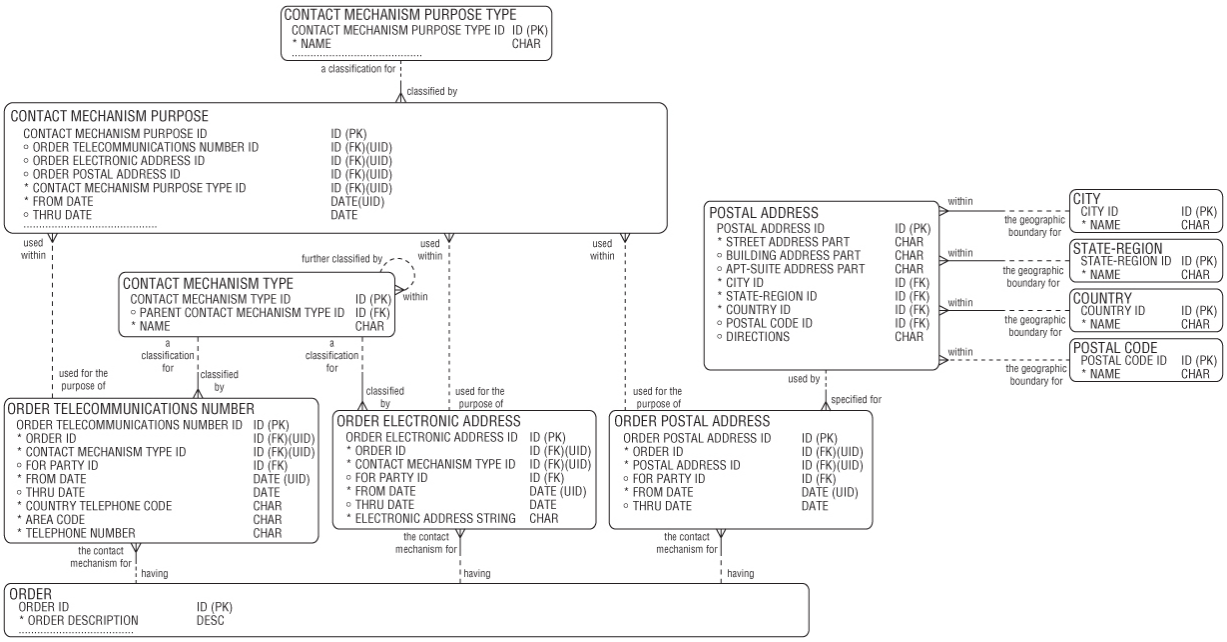


Figure 7.5 ORDER example of using a Level 2 Contact Mechanism Pattern



CONTACT MECHANISM TYPE contains a list of classifications of different contact mechanisms. For example, the **name** attribute in CONTACT MECHANISM TYPE would capture types like “Telephone number,” “Fax number,” and “Mobile number” for telecommunications

numbers and “Email address,” “Blog address,” and “Chat room address” for electronic addresses.

CONTACT MECHANISM TYPE also has a recursive relationship around it. In other words, “each CONTACT MECHANISM TYPE *may be* **further classified by** one or more CONTACT MECHANISM TYPE(s).” This means that the model can maintain an instance of CONTACT MECHANISM TYPE, such as “Telecommunications number,” that may have sub-classifications of “Fax number,” “Pager number,” and “Mobile number.” The model may also have a CONTACT MECHANISM TYPE of “Electronic address” that may have sub-classifications of “Email address,” “Blog address,” and so on.

In Figure 7.3, “each CONTACT MECHANISM 1 *may be* **classified by** one and only one CONTACT MECHANISM TYPE.” There may also be a many-to-many relationship as seen in Figure 7.4 (an example of how to use this pattern) between CONTACT MECHANISM TYPE and TELECOMMUNICATIONS NUMBER, and thus the pattern is slightly modified to accommodate this need. For example, the same phone number may be classified as both a “Telephone number” and a “Fax number.”

“Each ENTITY *may be* **having** one or more ENTITY CONTACT MECHANISM 1(s) and each ENTITY CONTACT MECHANISM 1 *may be* **specified for** one and only one CONTACT MECHANISM 1.” In other words, ENTITY CONTACT MECHANISM 1 resolves the many-to-many relationship between an ENTITY, such as PARTY, ORDER, SHIPMENT, or FACILITY, and a way to contact it (CONTACT MECHANISM 1), such as TELECOMMUNICATIONS NUMBER or POSTAL ADDRESS. The model needs to resolve this many-to-many relationship because one instance of an ENTITY may have many instances of CONTACT MECHANISM 1, and the same instance of CONTACT MECHANISM 1 may be related to the many instances of ENTITY. For example, the same instance

of ORDER may be related to many POSTAL ADDRESS(es) (one being the ‘Ship to’ and one being the ‘Bill to’), and the same instance of POSTAL ADDRESS may be used for many different ORDER(s).

What about the situation where a contact mechanism is not shared and is only applicable to one instance of the related entity? For example, what if an email address is a contact mechanism for only one instance of a PARTY? We address this need by using ENTITY CONTACT MECHANISM 2. In the previous example, the contact mechanism, such as a telephone number, was stored in CONTACT MECHANISM 1. This meant that you could reuse the same contact mechanism for many different instances of ENTITY, and an instance of ENTITY could use many different contact mechanisms (that is, CONTACT MECHANISM 1). In this example, the contact mechanism information, such as an electronic address, is stored in ENTITY CONTACT MECHANISM 2, not in its own entity.

Each of the ENTITY CONTACT MECHANISM 2(s) may also be classified by a CONTACT MECHANISM TYPE. For example, an electronic address may be classified as an email address, blog address, and so on. The entity, ENTITY CONTACT MECHANISM 2, contains the actual contact mechanism data, and it gets classified by CONTACT MECHANISM TYPE.

ENTITY CONTACT MECHANISM 1 or ENTITY CONTACT MECHANISM 2 may have one or more CONTACT MECHANISM PURPOSE(s). This supports the possible need to specify an intention(s) or reason(s) for which a particular contact mechanism is used. For example, a contact mechanism (such as a postal address or an email address) may be used as the designated place to send bills (“Bill to” purpose) and/or send products (“Ship to” purpose). A phone number or an email address may be used for the purpose of “Payment follow up,” “Billing inquiries,” or “Technical support.” Each CONTACT MECHANISM PURPOSE(s) must

have a CONTACT MECHANISM PURPOSE TYPE that has a **name** attribute that maintains the available purpose types.

An alternative approach to this design would be to have multiple contact mechanism purpose entities related to each different entity, instead of consolidating all of the contact mechanism purposes into a single CONTACT MECHANISM PURPOSE entity. For example, you could create ENTITY 1 CONTACT MECHANISM PURPOSE and ENTITY 2 CONTACT MECHANISM PURPOSE instead of the consolidated CONTACT MECHANISM PURPOSE. Or you could create subtypes in CONTACT MECHANISM PURPOSE related to each individual entity. This is a more specific way of modeling and can add to understanding of the subject area. However, it also adds a bit more complexity and a little less flexibility because associative entities are needed to maintain the purposes for each ENTITY CONTACT MECHANISM.

Each of the associative entities (ENTITY CONTACT MECHANISM 1 or ENTITY CONTACT MECHANISM 2) may have one and only one CONTACT MECHANISM USAGE TYPE. This supports the possible need to specify the common and customary use for a particular contact mechanism. For example, an email address may be used for “Business,” “Personal,” or “Business and Personal” reasons, or a mobile number may be used for an “Emergency” purpose.

Note

Some data modelers may have an issue with mixing types “Business and Personal.” There are some legitimate reasons not to mix these types together. For example, is “Personal and Business” the same as “Business and Personal”? An alternative model could be to have a many-to-many relationship between CONTACT MECHANISM USAGE TYPE and ENTITY CONTACT MECHANISM 1 (and ENTITY CONTACT MECHANISM 2) instead of a one-to-many relationship. This would allow the contact mechanism to have multiple usages of “Business,” “Personal,” and “Emergency” all at the same time. Generally, we find that the one-to-many relationship is sufficient because there are very few possible instances (“Business,” “Personal,” “Business

and Personal," or "Emergency"). Technically, you may conclude that there really are many usages for a CONTACT MECHANISM (the contact mechanism could be both "Business" and "Personal"), so this may be a valid alternative model. Another alternative is to use the Level 3 Contact Mechanism Pattern where we support a many-to-many relationship in this pattern between CONTACT MECHANISM USAGE TYPE and ENTITY CONTACT MECHANISM(s). Yet another alternative is to use the Level 4 Contact Mechanism Pattern.

Both CONTACT MECHANISM USAGE TYPE and CONTACT MECHANISM PURPOSE are related to the entities ENTITY CONTACT MECHANISM 1 and ENTITY CONTACT MECHANISM 2. Modeling the purpose and the usage in addition to the CONTACT MECHANISM TYPE provides flexible combinations for maintaining contact mechanisms. For example, imagine a person who works from home. This person (a PARTY) may have a telecommunications number of "Fax number" (CONTACT MECHANISM TYPE) that is used for "Business" (CONTACT MECHANISM USAGE TYPE) for the purpose of "Technical support" (CONTACT MECHANISM PURPOSE TYPE). This same telecommunications number may be used as a "Telephone number" (CONTACT MECHANISM TYPE) by that person's husband for personal use (CONTACT MECHANISM USAGE TYPE) with no purpose. Usage and purpose are relevant within the context of the PARTY's relationship to the CONTACT MECHANISM.

Figure 7.4 and Figure 7.5 further illustrate how to employ this pattern. For illustration, we can build upon the scenario described in the Level 1 Contact Mechanism Pattern section of the chapter. Initially, ABC Building Corporation wished to buy an off-the-shelf contact management software to manage its contact mechanism. They used Figure 7.2, based on the Level 1 Contact Mechanism Pattern, to evaluate third-party vendor contact management packages; however, the CIO (Chief Information Officer) did not like the options due to price more than function. He asked the data professional to come up with a solution that could be built

by the in-house IT staff. To do this the data professional suggested using the Level 2 Contact Mechanism Pattern as the basis for a database design for a prototype to maintain contact mechanisms for PARTY and ORDER.

The requirements of the building supplier were that the prototype solution captures all different types of potential contact mechanisms for ORDER and PARTY. Also, each of the different contact mechanisms may (or may not) have a purpose or usage associated with them. Managers wanted to be able to report on the various contact mechanisms for parties and/or orders by the different contact mechanism types, purposes, and usages.

Based on the requirements, by conducting interviews with staff and by using the Level 2 Contact Mechanism Pattern, the data professional created Figures 7.4 and 7.5. Figure 7.4, which shows the PARTY example of the Level 2 Contact Mechanism Pattern, states that one or more PARTY(s) can be related to one or more TELECOMMUNICATIONS NUMBER(s) (via PARTY TELECOMMUNICATIONS NUMBER).

This is illustrated in Table 7.4, where you see the **party** “1005 (Nadine Gerard)” with a “Mobile number” (CONTACT MECHANISM TYPE) of “1 805 555 4534” and a “Telephone number” (a different CONTACT MECHANISM TYPE) of “1 917 555 2100.” The CONTACT MECHANISM PURPOSE TYPE for the “1 805 555 4534” number is empty. In other words, no specific purpose is associated with this mobile number. The CONTACT MECHANISM USAGE TYPE for this number is “Personal.” In other words, this number is used for personal reasons only. This gives a clue to why you don't have any specific purpose associated with the number. The “1 917 555 2100” number has a CONTACT MECHANISM USAGE TYPE of “Business.” This means that the number can be used for commercial reasons. The specific purpose for this number is “Technical Support.” Nadine happens to be an engineering consultant who can be contacted via this

number when there is a technical issue. This is a very common situation—many people have a least two phone numbers, one for business and one for personal reasons.

Table 7.4 PARTY Telecommunications Numbers and Contact Mechanism Purposes

PARTY. PARTY ID (NAME)[2]	TELECOMMUNICATIONS NUMBER.COUNTRY TELEPHONE CODE, AREA CODE, TELEPHONE NUMBER[4]	CONTACT MECHANISM TYPE. NAME	CONTACT MECHANISM USAGE TYPE. NAME	CONTACT MECHANISM PURPOSE TYPE. NAME
1001 (XYZ Corporation)	1 917 555 2100	Telephone number	Business	General inquiries
1005 (Nadine Gerard)	1 805 555 4534	Mobile number	Personal	
1005 (Nadine Gerard)	1 917 555 2100	Telephone number	Business	Technical support
2002 (Ed Smith)	44 20 5555 1234	Mobile number	Personal	
2002 (Ed Smith)	44 20 5555 1234	Mobile number	Emergency	
2002 (Ed Smith)	44 20 5555 7983	Telephone number	Business	Invoice inquiries
2004 (Manu Collet)	91 11 2623 665	Fax number	Business	Sales fax number
2004 (Manu Collet)	91 11 2623 665	Telephone number	Business and Personal	General inquiry number
2004 (Manu Collet)	91 11 2623 665	Telephone number	Business and Personal	Sales inquiries
3000 (Toms Building Materials)	86 21 555 1000	Telephone number	Business	Shipping inquiries
3000 (Toms Building Materials)	86 21 555 1000	Telephone number		Billing inquiries
3010 (Lingsat Ltd)	54 11 4777 1221	Telephone number		

It should also be noted that the telecommunications number “1 917 555 2100” is also the “Business” (CONTACT

MECHANISM USAGE TYPE) “Telephone number” (CONTACT MECHANISM TYPE) for “XYZ Corporation” with **party id** “1001” (for the purpose type of “General inquiries”), as well as the “Telephone number” for “Nadine Gerard.” Nadine happens to contract to XYZ Corporation and therefore she uses the XYZ Corporation telephone number. Again, this is a common enough situation where a phone number is related to more than one party. Another example of this would be the “Telephone number” located at someone's home. This number may be used for all of the people in the house.

Examine the PARTY “Manu Collet”, with **party id** “2004”, in Table 7.4, with a telecommunications number of “91 11 2623 665.” The same telecommunications number has a CONTACT MECHANISM TYPE of both “Telephone number” and “Fax number.” When it is used as a fax (i.e., when it is of contact mechanism type “Fax number”), it has a usage of “Business”; when it is used as a telephone number, it can be used for both “Business and Personal” reasons. The same contact mechanism appears once in the TELECOMMUNICATIONS NUMBER entity and twice in TELECOMMUNICATIONS NUMBER CLASSIFICATION, once as a combination of “91 11 2623 665” and “Telephone number” and once as a combination of “91 11 2623 665” and “Fax number.” It is very common to have a telecommunications number that can be of several types, such as a fax, pager, and telephone all at the same time.

The same telecommunications number, “91 11 2623 665” (TELECOMMUNICATION NUMBER), has two associated instances of PARTY TELECOMMUNICATIONS NUMBER(s) for the party “Manu Collet” because the purpose depends on whether it is used as a “Telephone number” or a “Fax number.” Thus the PARTY TELECOMMUNICATIONS NUMBER may be of one and only one CONTACT MECHANISM TYPE in order to allow different purposes for different contact mechanism types. One of these instances of PARTY

TELECOMMUNICATIONS NUMBER has a CONTACT MECHANISM TYPE of “Fax number” and has a CONTACT MECHANISM PURPOSE TYPE **name** of “Sales fax number.” The other instance of this PARTY TELECOMMUNICATIONS NUMBER has a CONTACT MECHANISM TYPE of “Telephone number” and has two CONTACT MECHANISM PURPOSE TYPE **name(s)**, “General inquiry number” and “Sales inquiries.” Notice that this type of relationship (from the ENTITY CONTACT MECHANISM 1 to the CONTACT MECHANISM TYPE) is not in the pattern shown in Figure 7.3 and there is only a one-to-many relationship from the CONTACT MECHANISM TYPE to the contact mechanism (CONTACT MECHANISM 1), so this is a slight deviation from the pattern based on specific needs.

Finally, a PARTY can have a PARTY TELECOMMUNICATIONS NUMBER with no associated purpose or usage. The final example in Table 7.4, PARTY “Lingsat Ltd,” has a “Telephone number” of “54 11 4777 1221” with no purpose or usage associated with it. This is also quite normal; sometimes there is either not a need to maintain the specific purpose and usage of a contact mechanism, or perhaps they are both unknown.

The style for capturing of electronic address information is different from telecommunications numbers. In Figure 7.4, telecommunications numbers are captured in their own entity of TELECOMMUNICATIONS NUMBER, while electronic addresses are captured as part of the entity PARTY ELECTRONIC ADDRESS. Thus, each electronic address is not maintained in its own entity and is maintained just for a specific party. In Table 7.5 you can see the PARTY “XYZ Corporation” has a PARTY ELECTRONIC ADDRESS of “xyz@xyzcorp.com” with a CONTACT MECHANISM TYPE **name** of “Email address.” There are also two different CONTACT MECHANISM PURPOSE(s), “General info” and “Billing inquiries” purposes, respectively. Capturing the

contact mechanism data for electronic address in the entity PARTY ELECTRONIC ADDRESS works well for this type of data because an electronic address string is normally only for a single party and is not normally shared among parties.

Table 7.5 PARTY, Electronic Addresses and Contact Mechanism Purposes

PARTY. PARTY ID (NAME)[2]	PARTY ELECTRONIC ADDRESS. ELECTRONIC ADDRESS STRING	CONTACT MECHANISM TYPE. NAME	CONTACT MECHANISM USAGE	CONTACT MECHANISM PURPOSE
1001 (XYZ Corporation)	xyz@xyzcorp.com	Email address	Business	General info
1001 (XYZ Corporation)	xyz@xyzcorp.com	Email address	Business	Billing inquiries
1005 (Nadine Gerard)	ngirard@xyz.com	Email address	Business	
1005 (Nadine Gerard)	Barry@mylife.com	Email address	Personal	
2004 (Manu Collet)	Mcollet@mtln.net.in	Email address	Business	Technical support
2004 (Manu Collet)	www.my_space.com/mcollet/blog	Blog address	Personal	
2004 (Manu Collet)	www.techexperts.com/techchat	Chat room address	Business	Technical forum
3010 (Lingsat Ltd)	noticas@lingsat.com.ar	Email address	Business	Shipping inquiries

Alternatively, we could have shown the contact mechanism of electronic address in its own entity to allow parties to share the same electronic addresses. Storing the contact mechanism (whether it is an electronic address, postal address, or telecommunications number) in its own entity means that more complex logic is required to change a party's contact mechanism because you would need to check to see if all the parties with that contact mechanism should also be changed. However, it is very common that

many people share a postal address or phone number, and if this is so, then it is often beneficial to model a many-to-many relationship to accommodate this. For instance, a company postal address or telephone number may be shared by many people working for that company and if the data model accommodates shared addresses and telephone numbers, then this can help with data quality because there would only be one instance for these 'shared' contact mechanisms.

So, why use *electronic address* and not just *email address*? Electronic address is a more encompassing entity that includes many types of electronic addresses. For example, it accommodates the advent of 'Blogs,' which is a personal diary space online, and 'Interactive chat rooms.' These contact mechanism types are different from emails in terms of their usage and purpose. Emails are more analogous to physical mail; chat rooms are more like telephone conversations, that is, more interactive. A relatively new communications phenomenon is interactive help done via 'Chat' or "Instant Messaging (IM)" as opposed to over the telephone. Discussion groups have also sprung up that allow for general community communications, in other words, posting messages on an internet page for all those subscribed. Each of these types of electronic contact mechanisms can be handled with this structure. In Table 7.5 the **party** "2004 (Manu Collet)" has three different electronic address instances. "Mcollet@mtln.net.in" is a standard "Email Address" with a usage type of "Business" and a purpose of "Technical support." "www.my_space.com/mcollet/blog" is where this person posts a weekly personal diary; hence, this CONTACT MECHANISM TYPE **name** is "Blog address" with a CONTACT MECHANISM USAGE TYPE of "Personal." Lastly "www.techexperts.com/techchat," techexperts, is a chat room that this party attends for technical discussions on

products and services in his industry; hence, the contact mechanism type is “Chat room address” for the purpose of “Technical forum” with a usage type of “Business.”

Referring back to the scenario we've been discussing, ABC Building Corporation also wanted to make sure that it captured POSTAL ADDRESS(es) for people and organizations. In Figure 7.4 you see that POSTAL ADDRESS(es) are modeled in a similar fashion as TELECOMMUNICATIONS NUMBER(s), that is, by maintaining them in their own entity. Why do this? One good reason is that each POSTAL ADDRESS has data that you may not want to repeat. For example, you may maintain a standard set of **directions** for a POSTAL ADDRESS and you would not want to repeat this 1000 times if 1000 people had this same address (please see the end of this chapter for more discussion on modeling “Directions”). Also, whereas TELECOMMUNICATIONS NUMBER had CONTACT MECHANISM TYPE(s), such as “Fax number,” “Pager number,” and “Telephone number,” POSTAL ADDRESS has no need for types other than postal address, and therefore, there is not a CONTACT MECHANISM TYPE for this contact mechanism.

In Figure 7.4 you also see that POSTAL ADDRESS is related to CITY, STATE-REGION, COUNTRY, and POSTAL CODE. In the Level 1 Pattern this data was captured as attributes, such as **ship to city**, **ship to state-region**, **ship to country**, and **ship to postal code** in the entity ORDER in Figure 7.2. As was mentioned in the previous section, this may have led to data quality issues. For example, the **ship to city** attribute could maintain the same city inconsistently (for example, a **ship to city** of “NY” in one instance and “New York” in another instance). To accommodate this issue, the data professional related the POSTAL ADDRESS to entities of CITY, STATE-REGION, COUNTRY, and POSTAL CODE, that captured this reference data once and only once.

In Table 7.6 “Nadine Gerard” with **party id** “1005” has two different addresses “55 Right Road…” and “100 Boulder Street….” The first address is for “Personal” use (CONTACT MECHANISM USAGE TYPE) and has one purpose of “Home residence” (CONTACT MECHANISM PURPOSE TYPE). The second address is used for “Business” and has a single purpose of “Sales office.” This is Nadine's office in Santa Monica that she uses for sales purposes. If you look at “Manu Collet” with **party id** “2004,” you see his address “Andheri Kurla Road, Marol, no. 604…” is used for both “Business and Personal.” This is a common situation; many people have a home office location that is for “Business and Personal” use. There are three purposes (CONTACT MECHANISM PURPOSE) for this PARTY POSTAL ADDRESS, namely, “Home residence,” “Home office location,” and “Sales office”; thus, this illustrates that a party's postal address (for a specific type of usage) may have any number of purposes with this pattern.

Table 7.6 PARTY Contact Mechanisms, Address

PARTY. PARTY ID (PARTY NAME)[2]	PARTY POSTAL ADDRESS. POSTAL ADDRESS ID (FK)	POSTAL ADDRESS. STREET ADDRESS PART, BUILDING ADDRESS PART, APT-SUITE ADDRESS PART[4]	POSTAL ADDRESS. CITY, STATE-REGION,[3] COUNTRY, POSTAL CODE[4]	USAGE TYPE. NAME	PURPOSE TYPE. NAME
1001 (XYZ Corporation)	100	100 Main Street, Suite 819, The Coalman Building	New York, NY, USA, 10019	Business	Headquarters
1005 (Nadine Gerard)	210	55 Right Road, The Foundry	Los Angeles, CA, USA, 90210	Personal	Home residence
1005 (Nadine Gerard)	220	100 Boulder Street, Suite 23	Santa Monica, CA, USA, 90212	Business	Sales office
2004 (Manu Collet)	320	Andheri Kurla Road, Marol, no. 604	Mumbai, Andheri East, India, 400099	Business and Personal	Home residence
2004 (Manu Collet)	320	Andheri Kurla Road, Marol, no. 604	Mumbai, Andheri East, India, 400099	Business and Personal	Home office location
2004 (Manu Collet)	320	Andheri Kurla Road, Marol, no. 604	Mumbai, Andheri East, India, 400099	Business and Personal	Sales office

Figure 7.5 illustrates an ORDER example of the Level 2 Contact Mechanism Pattern for our building supply company based off the pattern described in Figure 7.3. This diagram looks and acts in much the same way as the previous diagram with two differences. In this model, the telephone numbers are maintained not in TELECOMMUNICATIONS NUMBER, but in the entity ORDER TELECOMMUNICATIONS NUMBER. Why the change? After interviewing staff about PARTY telephone numbers and ORDER telephone numbers, the data professional discovered a different set of needs and uses for ORDER telephone numbers. The PARTY TELECOMMUNICATIONS NUMBER(s) were used all over the company to generate business and to contact subcontractors, customers, architects, city planners, and so on. The telephone numbers on an ORDER were only used if there was a query or problem with the order. The phone numbers for an ORDER would also change for every ORDER. This means that the ORDER telephone numbers were tightly coupled to the ORDER and not re-used.

Second, this example of the pattern does not utilize CONTACT MECHANISM USAGE TYPE(s). The different usage classification helped distinguish between “Business,” “Personal,” “Emergency,” and “Business and Personal.” However, all of the ORDER(s) in our scenario are used for “Business” reasons. Adding the CONTACT MECHANISM USAGE TYPE would have been superflous. The operations staff already had a list of PARTY(s) telephone numbers if they had a problem with an order. However, because the ORDER contact mechanisms were sometimes different than the PARTY's usual contact mechanisms, the data professional created a relationship from ORDER to ORDER TELECOMMUNICATIONS NUMBER with the attributes **country telephone code**, **area code**, and **telephone number.**

Figure 7.5 shows that there is also an optional foreign key **for party id** in the ORDER TELECOMMUNICATIONS NUMBER, ORDER ELECTRONIC ADDRESS, and ORDER POSTAL ADDRESS entities to capture the associated party of the contact mechanism. Thus, if there was a telephone number, email, or postal address for the order, it is often helpful to associate this with a party who is using this contact mechanism for the order. For example, if an order had a telephone number for "Payment follow up," that number may be for a specific person, which is useful to know. This is very different than relating the ORDER to a PARTY TELECOMMUNICATIONS NUMBER (the entity in Figure 7.4), which would require setting up a party with a telecommunications number. In Figure 7.5, we are specifying that there are contact mechanisms specifically for an order (hence, not to be used for other purposes) and there may be a need to specify a party that is related to that contact mechanism. Thus, parties may provide a telephone number or an electronic address specifically for that order and that is what we are capturing with the **for party id** foreign key attributes in Figure 7.5

While we have modeled the contact mechanism party with an optional foreign key in Figure 7.5, there are numerous ways to model the parties associated with the order and the order contact mechanisms. For instance, there may be a need for a many-to-many relationship if there could be multiple parties for an order's contact mechanism. Another modeling option would be to use a contextual roles pattern from Chapter 3 to maintain the various parties involved in various roles, such as "Payment follow up contact," "Shipment notification coordinator," and so on. You could then relate these contextual roles for parties to the order contact mechanism entities. Finally, as yet another option, you could relate the order to PARTY CONTACT MECHANISM; however, this has the drawbacks that we mentioned

previously of having to first set up the contact mechanism for the party and also there may be cases where you only need a contact mechanism without a party.

Table 7.7 further illustrates how this pattern works. The ORDER with **order id** “47742” and **order description** “Cement” was an order that XYZ Corporation placed (the party names are not part of the related Figure 7.5 and not in Table 7.7; however, to set the scene we are just giving you some background information). There is a telecommunications number of “1 917 555 2100” that is associated with this order that ABC Building Corporation requires just in case it needs to follow up regarding payment for the order or to call if any shipping problems arise. The third row shows that this number is also used as a “Fax Number” for “Shipment notification,” or in other words, to notify the party about the shipment details when the shipment occurs for the order. According to the pattern in Figure 7.5, there are two instances of the ORDER TELECOMMUNICATIONS NUMBER entity, because this number may be either a “Telephone number” or a “Fax number”. When this number is a “Telephone number” of “1 917 555 2100,” it has two different purposes: (CONTACT MECHANISM PURPOSE(s)) of “Payment follow up” (to follow up regarding payment for the order) and “Shipping instruction inquiries” (to contact the party if any shipping problems arise). When it is used as a “Fax number” of “1 917 555 2100,” it has only one purpose, namely, “Shipment notification.” This illustrates a potential weakness with this pattern. If the same ORDER TELECOMMUNICATIONS NUMBER “1 917 555 2100” was also a “Pager number” and a “Mobile number,” ABC Corporation's order entry staff would have captured this number four times, each time with a different CONTACT MECHANISM TYPE. This is not a very efficient use of their time, as they are putting in the same data again and again, instead of just once. Also, capturing

the number four times, redundantly, may lead to data quality issues. We can imagine that ABC Corporations order entry staff may make a simple mistake entering one of the numbers. This weakness can be addressed by using the first flavor of the Level 2 Contact Mechanism Pattern seen illustrated in Figure 7.4.

Table 7.7 ORDER, Telecommunications Numbers and Contact Mechanism Purposes

ORDER. ORDER ID	ORDER. ORDER DESCRIPTION	ORDER TELECOMMUNICATIONS NUMBER.COUNTRY TELEPHONE CODE, AREA CODE,	CONTACT MECHANISM TYPE. NAME	CONTACT MECHANISM PURPOSE TYPE.
47742	Cement	1 917 555 2100	Telephone number	Payment follow up
47742	Cement	1 917 555 2100	Telephone number	Shipping instruction inquiries
47742	Cement	1 917 555 2100	Fax number	Shipment notification
47790	Steel girders	11 805 555 8999	Telephone number	Payment follow up
47799	Gold electrical wiring	44 20 5555 1234	Mobile number	Shipping instruction inquiries

Examples of electronic addresses for ORDER(s) are shown in Table 7.8. You see that **order id** “47742” for “Cement” has two electronic address contact mechanisms. The first contact mechanism for the ORDER is given by customers in order to receive confirmation details of their order. The next “email address” associated with this order is “accountspayable@xyzcorp.com,” which has a purpose of “Payment inquiries” that can be used by ABC Building Corporation if it needs to ask the customer anything regarding their payments. It is also worth noting that many

more firms are using interactive chat rooms for inquiries. In Table 7.8 you see **order id** "47790" with an electronic address of "www.Inquiries_abcspace.com/techchat" with a type of "Chat room" that the customer can use for the purpose of "Technical support."

Table 7.8 ORDER, Electronic Addresses and Contact Mechanism Purposes

ORDER. ORDER. ID	ORDER.ORDER DESCRIPTION	ORDER ELECTRONIC ADDRESS. EMAIL ADDRESS STRING	CONTACT MECHANISM TYPE. NAME	CONTACT MECHANISM PURPOSE TYPE. NAME
47742	Cement	info@ xyzcorp.com	Email address	Order confirmation
47742	Cement	accountspayable@ xyzcorp.com	Email address	Payment inquiries
47790	Steel girders	www.Inquiries_ abcspace.com/techchat	Chat room	Technical support

Take a look at ORDER(s) and POSTAL ADDRESS(es) in Table 7.9. Each ORDER can be related to many POSTAL ADDRESS(es). For example, an order could have a "Ship to" address and a "Bill to" address. In Table 7.9 you see order "47790" for "Steel girders" has two associated addresses: "55 Right Road..." and "100 Boulder Street... ." Each of these addresses may have one or more CONTACT MECHANISM PURPOSE TYPE(s). As you can see order "47790" with address "55 Right Road..." has a CONTACT MECHANISM PURPOSE TYPE **name** of "Ship to," and "47790" with address "100 Boulder Street..." has a CONTACT MECHANISM PURPOSE TYPE **name** of "Bill to."

Note

Depending on the business rules, orders may actually have their "Ship to" address specified at the ORDER ITEM detail so that different items may go to different places. However, for illustration purposes, we show the "Ship to" address related to the ORDER. This pattern still works if the "Ship to" relationships

are at the ORDER ITEM detail, and the ORDER ITEM would then have an associated contact mechanism(s).

Table 7.9 ORDER Contact Mechanisms, Address

ORDER. ORDER ID	ORDER.ORDER DESCRIPTION	ORDER POSTAL ADDRESS. POSTAL ADDRESS ID (FK)	POSTAL ADDRESS (STREET ADDRESS PART, BUILDING ADDRESS PART, APT-SUITE ADDRESS PART)[4]	POSTAL ADDRESS (CITY, STATE-REGION,[3] COUNTRY, POSTAL CODE)	PURPOSE TYPE. NAME
47742	Cement	100	100 Main Street, Suite 819, The Coalman Building	New York, NY, USA, 10019	Ship to
47742	Cement	100	100 Main Street, Suite 819, The Coalman Building	New York, NY, USA, 10019	Bill to
47790	Steel girders	210	55 Right Road, The Foundry	Los Angeles, CA, USA, 90210	Ship to
47790	Steel girders	220	100 Boulder Street, Suite 23	Santa Monica, CA, USA, 90212	Bill to
47799	Gold electrical wiring	320	Andheri Kurla Road, Marol no. 604	Mumbai, Andheri East, India, 400099	Ship to
5000	Paint	320	Andheri Kurla Road, Marol no. 604	Mumbai, Andheri East, India, 400099	Ship to

An instance of a POSTAL ADDRESS (maintained in ORDER POSTAL ADDRESS) may have more than one purpose for a specific ORDER. This can be seen for order “47742.” This order was for “Cement” to be shipped to “100 Main Street, Suite 819,....” The bill for the order was to be sent to the same address, and therefore there is another purpose for this address, namely, “Bill to.”

Finally, you can see that the same POSTAL ADDRESS can have many different ORDER(s) associated with it. The postal address beginning with “Andheri Kurla Road” in Mumbai has two associated orders—“47799” for “Gold electrical wiring” and “5000” for “Paint”—that are both being shipped to this address.

When Should This Pattern Be Used?

We use this pattern:

- **When there is a need for more flexibility than seen in the level 1 pattern when modeling contact mechanisms. This pattern maintains any number of contact mechanisms, purposes, usages, or types for contact mechanisms** This pattern is much more flexible than the Level 1 Contact Mechanism Pattern in that it accommodates as many contact mechanisms as are needed and as many contact mechanism types, purposes, and usages as are needed. It also allows additional contact mechanisms, types, purposes, and usages to be added over time.
- **When there are a large number of different types of contact mechanisms with different usages and purposes:** When there are needs for many different types of telecommunications numbers, electronic addresses, and postal addresses with various purposes and usages, then this pattern can accommodate these needs.

- **When there is a need to balance a specific style of modeling with a generalized style:** Although this pattern provides a lot more flexibility than the Level 1 Contact Mechanism Pattern, it is still a specific model (in comparison to some other styles of modeling, for example, Level 3 and Level 4 models). If the attributes, entities, or relationships are not static, well understood, and/or well defined, using this pattern will accommodate a lot of change. As usual, when choosing the correct pattern, it's a case of picking the “right horse for the right course.” This pattern balances flexibility with specificity because it has specific entities for different types of contact mechanisms rather than one generalized contact mechanism entity. This provides the ability to use different data model structures for different types of contact mechanisms in order to provide more specific rules and ways of handling certain types of contact mechanisms. For example, Figure 7.4 provided an example of the pattern where electronic addresses were not maintained in their own entity and instead they were specifically tied to a party (via PARTY ELECTRONIC ADDRESS) because it was deemed that electronic addresses, such as email addresses, were not re-used for a party.
- **When showing the different requirements and needs of different stake holders:** This pattern is an effective way to show the scope and requirements for contact mechanisms to data professionals or power users. It is not quite as simple as the Level 1 Pattern because it requires understanding of relationships in data modeling. But it explicitly shows different ways to classify contact mechanisms (CONTACT MECHANISM USAGE TYPE, CONTACT MECHANISM PURPOSE TYPE, and CONTACT MECHANISM TYPE) and different specific contact mechanism entities (POSTAL ADDRESS,

ELECTRONIC ADDRESS, and so on). It also captures the requirement to add new CONTACT MECHANISM USAGE TYPE(s) and CONTACT MECHANISM PURPOSE TYPE(s).

- **To provide alternative ways to model contact mechanisms:** For example, in Figure 7.4, we show an email address was not shared between parties; however, this requirement could change when the same email address is used by multiple members of a family and the enterprise wishes to capture the email address in its own entity and then be able to relate it to multiple parties. To support this requirement we could have modeled emails in the same way as we modeled telecommunications numbers. This pattern allows both alternatives.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **Maybe more difficult to understand for some audiences:** This pattern may lose some of the advantages of the simplicity of the Level 1 Contact Mechanism Pattern. It may not be as visually clear to some nontechnical people as the specific pattern. For example, we can easily imagine how some nontechnical people may have difficulty dealing with the generalized concept of contact mechanisms by just looking at the diagrams.
- **This pattern does not include all possible ways to classify contact mechanisms:** There may be other ways to classify contact mechanisms, such as by location (home, work, car), priority (primary, secondary), and technology (fax machine, PDA device). In this pattern we only concentrated on what we consider to be the most common classifications, i.e.,

CONTACT MECHANISM PURPOSE and CONTACT MECHANISM USAGE.

- **More difficult to manage because different types of contact mechanisms are modeled using different structures:** Although this pattern offers the benefit of being more specific than the level 3 and level 4 patterns, this is also a disadvantage because each type of contact mechanism may have different data model structures and may need to be handled differently. Thus, there is not a consistent way of managing contact mechanisms, and each type of contact mechanism may require different data structures, rules, and routines. Thus, it may be more costly and difficult to manage contact mechanisms with this pattern than by using a common, consistent structure for contact mechanisms, such as the ones in the level 3 or level 4 patterns.
- **When other entities are related to contact mechanisms, there are several places to look (several entities for contact mechanisms), and this can result in more complexity in systems development:** Contact mechanisms are referenced together frequently, and when you model different types of contact mechanisms in their own entities, it makes it more difficult to answer questions like: “What are all the ways to contact this person?” or “What are all the contact mechanisms for an order?”

Synopsis

The Level 2 Contact Mechanism Pattern is a much more flexible pattern than the Level 1 Contact Mechanism Pattern because it provides for any number of contact mechanisms, as well as types, purposes, and usages of contact mechanisms. There are two different styles of capturing the

contact mechanism and each style could be used for different types of contact mechanisms:

- One style captures the contact mechanism with its own entity, such as POSTAL ADDRESS or TELECOMMUNICATIONS NUMBER.
- The other style captures the contact mechanism as part of a dependent entity, such as PARTY ELECTRONIC ADDRESS.

This pattern supports all different types of contact mechanisms. CONTACT MECHANISM TYPE classifies contact mechanisms. For example the **name** attribute in CONTACT MECHANISM TYPE includes “Telephone number,” “Fax number,” “Pager,” and “Email address.” CONTACT MECHANISM TYPE(s) may be **further classified by** other CONTACT MECHANISM TYPE(s). For example, “Electronic address” may be made up of “Email address,” “Blog address,” and so on, or “Telecommunications number” may be made up of “Fax number,” “Mobile number,” “Pager Number,” and “Telephone number.”

The pattern also supports the reasons the contact mechanism is being used and its context (business or personal), that is, its purpose and its usage. Each of the associative entities (ENTITY CONTACT MECHANISM 1 and ENTITY CONTACT MECHANISM 2) may have one or more CONTACT MECHANISM PURPOSE(s) and one and only one CONTACT MECHANISM USAGE. This supports the possible need to specify an intention(s) or a reason(s) for which a particular contact mechanism is used. For example, a telephone number may be used for the CONTACT MECHANISM USAGE TYPE of “Business,” and addresses may have multiple CONTACT MECHANISM PURPOSE TYPE(s) of “Bill to” (the address to send bills) and “Ship to” (the address to send products).

The strength of this pattern lies in the fact that it goes some way toward modeling contact mechanisms in a

specific manner and yet still provides a reasonable amount of flexibility to add new contact mechanisms, contact mechanism types, purposes, and/or usages. We would not have much problem showing a model like this to stake holders with only a modicum of data modeling experience.

The weaknesses of the pattern are that it is specific in some ways and thus includes business rules that may change and that in other ways it is not as flexible as the level 3 and level 4 patterns. For instance, each type of contact mechanism (telecommunications number, electronic address, and postal address) has its own data model structure and may be managed in different ways. If the enterprise is somewhat mercurial in nature but desires some degree of specificity when modeling contact mechanisms, this may be the way to go.

Level 3 Contact Mechanism Pattern

Common attributes (if any exist) and common relationships are consolidated in this pattern by an entity known as CONTACT MECHANISM. CONTACT MECHANISM encapsulates all of the different types of contact mechanisms, such as telecommunications numbers, postal addresses, and electronic addresses. This means instead of independently defining each of the different contact mechanisms, they are defined as part of the CONTACT MECHANISM entity.

So, why do this? The Level 3 Contact Mechanism Pattern is used where an enterprise wishes to create a structure that supports the encapsulation of any common attributes and relationships for contact mechanisms. Though there are usually not many common attributes for different types of CONTACT MECHANISM(s), there are usually many common relationships. For example, all types of contact mechanisms

are usually related to PARTY, FACILITY, ORDER, SHIPMENT, INVOICE, PAYMENT, and so on. Having the supertype CONTACT MECHANISM helps simplify the relationships that different contact mechanisms have to other entities. This enables the grouping of all contact mechanisms under a single umbrella in order to manage and simplify the relationships to contact mechanisms. This pattern also allows for a more consistent way to maintain contact mechanism data. For example, by providing the supertype of CONTACT MECHANISM, we can apply the classification pattern to the CONTACT MECHANISM and categorize all contact mechanisms instead of having to categorize each type of contact mechanism entity.

Why Do We Need This Pattern?

Like all level 3 patterns throughout the book, this pattern is more generalized and flexible than the previous patterns in this chapter. It is not a colossal leap from the Level 2 Contact Mechanism Pattern to this pattern, but there is one very significant change: the addition of the CONTACT MECHANISM supertype. Encapsulating the different types of contact mechanisms in this generalized supertype allows the designer to avoid managing multiple relationships (and potentially repeated attributes, such as **non-solicitation indicator**) for each different contact mechanism. The addition of CONTACT MECHANISM also provides a handy place holder for all new contact mechanism types that may be discovered over the life cycle of a project. So the main reason you need this pattern is to simplify the relationships to contact mechanisms, because many relationships may be needed to reference a complete set of contact mechanisms.

Without this entity of CONTACT MECHANISM, it would be necessary to show relationships to each type of contact mechanism entity. For example, the level 2 patterns example would require a PARTY to be related to PARTY

TELECOMMUNICATIONS NUMBER, PARTY ELECTRONIC ADDRESS, and PARTY POSTAL ADDRESS instead of just having a PARTY related to PARTY CONTACT MECHANISM. This is also the case regarding the whole data model, and there would be needs for several additional associative entities when relating CONTACT MECHANISM(s) to FACILITY(s), ORDER(s), SHIPMENT(s), INVOICE(s), PAYMENT(s), and so on. Using the level 2 pattern may lead to a lot more relationships and associative entities than this level 3 pattern. Thus, using this pattern helps keep the model more manageable, helps with consistent handling of contact mechanisms, and helps reduce the number of additional associative entities when contact mechanisms are related to other entities.

How Does This Pattern Work?

In Figure 7.6, CONTACT MECHANISM 1 and CONTACT MECHANISM 2 represent different types of contact mechanisms that need to be modeled. These are wrapped within the supertype CONTACT MECHANISM. CONTACT MECHANISM is the generalized concept that can be defined as a method or way to get in touch with a party or other entity using some type of label, string, or identifier, such as a phone number, email address, postal address, and so on. CONTACT MECHANISM 1 and CONTACT MECHANISM 2 may be a template for entities like POSTAL ADDRESS, ELECTRONIC ADDRESS, or TELECOMMUNICATIONS NUMBER, as seen in Figure 7.7, or you may choose to have a different set of subtypes that is suitable for your enterprise's terminology and needs, for example, PHONE NUMBER, EMAIL, and ADDRESS.

Figure 7.6 Level 3 Contact Mechanism Pattern

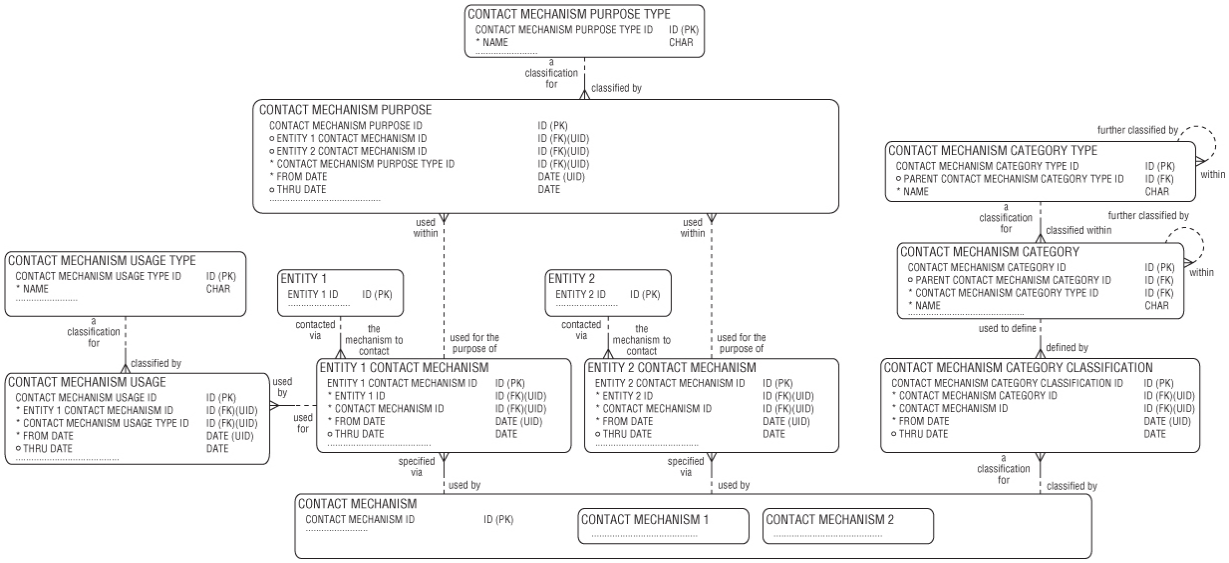
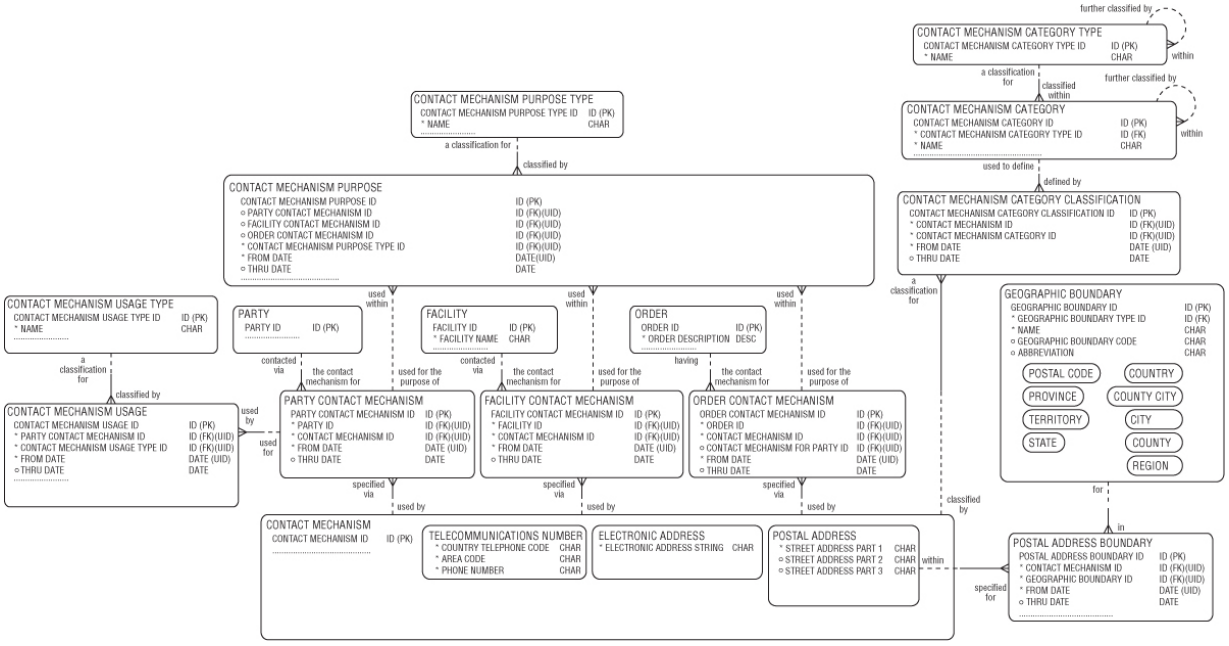


Figure 7.7 Example of using a Level 3 Contact Mechanism Pattern



With the addition of the CONTACT MECHANISM entity, common attributes and relationships for the contact mechanism can be handled in a simple and elegant fashion. In the previous pattern, the Level 2 Contact Mechanism Pattern, you see from Figure 7.4 that TELECOMMUNICATIONS NUMBER, ELECTRONIC ADDRESS, and POSTAL ADDRESS each had their own relationship to CONTACT MECHANISM via

PARTY TELECOMMUNICATIONS NUMBER, PARTY ELECTRONIC ADDRESS, and PARTY POSTAL ADDRESS. This can now be consolidated using a **used by, specified via** generalized relationship from CONTACT MECHANISM to PARTY CONTACT MECHANISM, as seen in Figure 7.7. This consolidated relationship works for all of the different types of CONTACT MECHANISM(s).

By adding a CONTACT MECHANISM entity, a data professional also creates a place holder for all newly discovered types of contact mechanisms. If a new type of contact mechanism is needed, all that needs to be done is to add a subtype and an instance of CONTACT MECHANISM CATEGORY. The relationships from CONTACT MECHANISM to various entities already exists, the primary key attributes already exist, and other common attributes, if any, are already available.

This pattern adds the Level 3 Classification Pattern that allows for the classification of each of the different contact mechanisms. Each of the different subtype entities in CONTACT MECHANISM will have an instance categorizing it in the CONTACT MECHANISM CATEGORY entity. In other words, if a subtype of CONTACT MECHANISM is a TELECOMMUNICATIONS NUMBER, you would have a "Telecommunications number" instance in CONTACT MECHANISM CATEGORY. Additionally, each contact mechanism may be broken down into multiple levels via the relationship "Each CONTACT MECHANISM CATEGORY *may be* **further classified by** one or more CONTACT MECHANISM CATEGORY(s)." In other words you can break down "Telecommunications number" into "Fax number," "Mobile number," "Pager number," and so on. Or you could break down "Electronic address" into "Email address," "Blog address," and so on. The CONTACT MECHANISM would be related to a CONTACT MECHANISM CATEGORY CLASSIFICATION that is related to a lower-level CONTACT

MECHANISM CATEGORY. For example, a contact mechanism may be classified as a “Fax number,” which is within the classification of “Telecommunications number.”

CONTACT MECHANISM CATEGORY *must be* **classified within** one and only one CONTACT MECHANISM CATEGORY TYPE. This flexible data model structure allows contact mechanisms to be categorized in as many ways as is needed, such as by the contact mechanism type, purpose, usage, priority, location, and so on. For example, “Telecommunications number,” “Electronic address” and “Postal address” may be classified as “Contact Mechanism Types.” CONTACT MECHANISM CATEGORY may contain many more ways to categorize a contact mechanism. For example, earlier in this chapter we mentioned that some enterprises may wish to capture “Location types” of contact mechanisms. They may wish to capture the fact that a “Telephone number” is the “Home” telephone number or “Office” telephone number (this is different than “Business” or “Personal” usage in that it maintains that the physical location of the device is at their home or office). The Level 3 Classification Pattern allows you to easily do this by adding the categories “Home” and “Office” to CONTACT MECHANISM CATEGORY and classifying these categories as “Location type” in CONTACT MECHANISM CATEGORY TYPE. Another example would be a CONTACT MECHANISM CATEGORY TYPE of “Technology type” with CONTACT MECHANISM CATEGORY(s) of “Mobile phone device,” “PDA device,” “Fax machine,” “Pager Device,” and so on. This would allow an enterprise to classify its contact mechanism based on the physical device that enables the contact mechanism. We go into this topic in more detail in the Level 4 Contact Mechanism Pattern later in the chapter. Another possible CONTACT MECHANISM CATEGORY TYPE could be “Priority type,” and associated values in CONTACT MECHANISM CATEGORY could be “Primary,” “Secondary,”

and “Tertiary” and could be used to describe which contact mechanism to use first, second, third, and so on.

The addition of CONTACT MECHANISM CATEGORY does have one significant drawback. Not all CONTACT MECHANISM CATEGORY(s) are applicable to all CONTACT MECHANISM(s). By generalizing the classification of CONTACT MECHANISM with the Level 3 Classification Pattern you allow CONTACT MECHANISM to be classified in any way. For example, a “Technology type” of “Fax machine” could be misused and applied to a POSTAL ADDRESS.

You might ask, “If you have subtypes of contact mechanism, do you also need a Level 3 Classification Pattern?” The point of having a CONTACT MECHANISM CATEGORY(s) is to support a wide range of categorizations, while subtypes support a very basic classification. CONTACT MECHANISM subtypes allow specific attributes and relationships for subtypes of contact mechanism. For example, the contact mechanism subtype of TELECOMMUNICATIONS NUMBER has attributes of **country telephone code**, **area code**, and **phone number**.

Note

For more detail on the Level 3 Classification Pattern see Chapter 5. This section also highlights how you can integrate different patterns from the different sections in the book to create a comprehensive solution. The Level 3 Classification Pattern can be considered a component that we attached to the Level 3 Contact Mechanism Pattern to provide a flexible solution for classifying contact mechanisms.

Much like you saw in Figure 7.3, you can see that “each of the ENTITY CONTACT MECHANISM(s) *may be* **used for the purpose of** one or more CONTACT MECHANISM PURPOSE(s) and each CONTACT MECHANISM PURPOSE *must be* **classified by** one or more CONTACT MECHANISM PURPOSE TYPE(s).” In other words, you may say that an organization

with a telecommunications number of “353 1 8555 209” may use that contact mechanism for the purpose of “Order confirmation” and “Payment follow up,” and the same organization may have a postal address “323 Howth Road…,” which could be used for the purposes of “Ship to” and “Bill to.”

Each ENTITY 1 CONTACT MECHANISM *may be* **used for** one or more CONTACT MECHANISM USAGE(s), each of which *must be* **classified by** a CONTACT MECHANISM USAGE TYPE.” In the previous section you saw that an ENTITY CONTACT MECHANISM(s) may be classified by the type of usage. For example, a telecommunications number may be used for “Business,” “Personal,” “Business and Personal,” or “Emergency” reasons. In this pattern we use a more flexible way (and technically, a more normalized way) to capture the usages of an ENTITY CONTACT MECHANISM. In this pattern an ENTITY CONTACT MECHANISM may have more than one CONTACT MECHANISM USAGE TYPE. In other words, a person or an organization may use the same telecommunications number for both “Business” and “Personal” uses. This helps avoid creating mixed types, such as “Business and Personal,” in CONTACT MECHANISM USAGE TYPE. Also, notice that ENTITY 2 CONTACT MECHANISM does not have any usages. It is also common that some ENTITY(s), such as ORDER, SHIPMENT, WORK EFFORT, or INVOICE, may only ever have one usage of “Business.” So there is no need to explicitly capture it.

Note

In Figure 7.7, there are entities for CONTACT MECHANISM USAGE TYPE and CONTACT MECHANISM PURPOSE TYPE (as well as associative entities for these). There are also entities for CONTACT MECHANISM CATEGORY. A variation on this pattern would be to maintain the “Purpose type” and “Usage type” as other CONTACT MECHANISM CATEGORY TYPE(s). This would also require relationships from PARTY CONTACT MECHANISM, ORDER CONTACT MECHANISM and FACILITY CONTACT MECHANISM to

CONTACT MECHANISM CATEGORY CLASSIFICATION, as well as exclusive arcs (similar to the exclusive arc shown in the Level 4 Contact Mechanism Pattern). You could then remove the CONTACT MECHANISM PURPOSE TYPE and the CONTACT MECHANISM USAGE TYPE from this pattern because these could also be considered ways of classifying the associations between contact mechanisms and entities.

Figure 7.7 was created using the pattern in Figure 7.6 as a template based on a change to the needs of ABC Building Corporation, namely, that it wanted a more flexible and comprehensive solution for a production system. As we discussed in the previous section, the building company viewed the vendor packages based on its needs and decided to build its own prototype. This was successfully completed, as was seen in the previous section. The CEO liked the prototype solution. At this point, ABC Building Corporation decided to build a very robust solution for contact mechanisms for its production environment, and thus, it needed a very flexible data model. The CIO wanted to ensure that the data model would support ABC Building Corporations current and future contact mechanism needs, with a minimum of impact.

The three contact mechanism entities picked for this phase of the solution were the same types of contact mechanisms needed in the prototype. Thus, the data model uses a supertype of CONTACT MECHANISM in Figure 7.7 and has subtypes of TELECOMMUNICATIONS NUMBER, ELECTRONIC ADDRESS, and POSTAL ADDRESS. Some of the CONTACT MECHANISM CATEGORY instances are “Postal Address,” “Electronic Address,” and “Telecommunications number,” one for each of the subtypes of CONTACT MECHANISM (and there are also many lower-level categories, such as “Fax number,” “Pager number,” and so on). These categories are classified as “Contact Mechanism Type” in the CONTACT MECHANISM CATEGORY TYPE. Of course, there are many other CONTACT MECHANISM

CATEGORY TYPE(s) that may be maintained, such as “Priority type,” that may maintain CONTACT MECHANISM TYPE(s) of “Primary,” “Secondary,” and “Tertiary,” which could maintain which contact mechanism to try first, second, third, and so on. Another example of a CONTACT MECHANISM CATEGORY TYPE is “Location type,” which could maintain CONTACT MECHANISM CATEGORY(s) of “Home” or “Office” to specify where the contact mechanism is directed (showing if you are calling someone at their home or at their office). This is different from the usages of “Personal” and “Business” because you could call someone at their “Home” but the telephone number might be specified for “Business” usage.

Note

Notice in Figure 7.7 there are entities for POSTAL ADDRESS BOUNDARY and GEOGRAPHIC BOUNDARY that allow a flexible approach to allowing any type of geographic boundary. This is especially useful for international addresses where there may be a region, territory, prefecture, and so on. This is not in the pattern from Figure 7.6, and we do not discuss this here because we describe this in the section later on this chapter, “Contact Mechanism Pattern with Geographic Boundary.” However, we included this in Figure 7.7 because this level of generalization is applicable to a Level 3 Pattern.

As we stated earlier “each CONTACT MECHANISM CATEGORY *may be* **further classified by** one or more CONTACT MECHANISM CATEGORY(s).” In other words, you can break down “Telecommunications number” into “Fax number,” “Mobile number,” “Pager number,” “Telephone number,” and so on. Or you could break down “Electronic address” into “Email address,” “Blog address,” and so on. For example, in Table 7.10 you see that “XYZ Corporation” has a “Telephone number” of “1 917 555 2100.” Using the recursive relationship on CONTACT MECHANISM CATEGORY in Figure 7.7, you can maintain that this “Telephone

number” is **within the** CONTACT MECHANISM CATEGORY of “Telecommunications number.” Similarly, the party “Nadine Gerard” has a “Telephone number” “1 805 555 4534” and the party “Ed Smith” has a “Mobile number” “44 20 5555 1234” that are also “Telecommunication number(s).” The recursive relationship around CONTACT MECHANISM CATEGORY allows you to assign specific (or more general) types to CONTACT MECHANISM(s). This is very useful because initially you may be able only to say that a contact mechanism “1 212 555 9999” is just a “Telecommunications number,” but after some time, you may find out it is in fact a “Mobile number” and a “Pager number.”

Table 7.10 PARTY Contact Mechanisms, Address, Phone Number, and Electronic Address

PARTY. PARTY ID (PARTY NAME)[2]	PARTY CONTACT MECHANISM. CONTACT MECHANISM ID	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)[4]	USAGE TYPE. NAME	CONTACT MECHANISM PURPOSE TYPE. NAME	CONTACT MECHANISM CATEGORY. NAME
1001 (XYZ Corporation)	100	100 Main Street, Suite 819, The Coalman Building New York, NY, USA, 10019	Business	Ship to	Postal address
1001 (XYZ Corporation)	200	xyz@xyzcorp.com	Business	General inquiries	Email address
1001 (XYZ Corporation)	200	xyz@xyzcorp.com	Business	Billing inquiries	Email address
1001 (XYZ Corporation)	300	1 917 555 2100	Business	General inquiries	Telephone number
1001 (XYZ Corporation)	300	1 917 555 2100	Emergency	General Emergency number	Telephone number
1005 (Nadine Gerard)	400	1 805 555 4534	Personal		Telephone number
1005 (Nadine Gerard)	400	1 805 555 3434	Business		Telephone number
2002 (Ed Smith)	500	44 20 5555 1234	Personal		Mobile number
2002 (Ed Smith)	500	44 20 5555 1234	Emergency		Mobile number
2002 (Ed Smith)	550	44 20 5555 7983	Business		Telephone number
2004 (Manu Collet)	600	91 11 2623 665	Business		Fax number
2004 (Manu Collet)	600	91 11 2623 665	Business		Telephone number
2004 (Manu Collet)	600	91 11 2623 665	Personal		Telephone number
2004 (Manu Collet)	320	Andheri Kurla Road Marol no. 604, Mumbai, Andheri East, India, 400099			Postal address

PARTY. PARTY ID (PARTY NAME)[2]	PARTY CONTACT MECHANISM. CONTACT MECHANISM ID	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)[4]	USAGE TYPE. NAME	CONTACT MECHANISM PURPOSE TYPE. NAME	CONTACT MECHANISM CATEGORY. NAME
2004 (Manu Collet)	700	Mcollet@mtln.net.in	Business		Email address
2004 (Manu Collet)	800	www.my_space.com/ blog Mcollet/	Personal		Blog address
2004 (Manu Collet)	900	www.techexperts.com/ techtalk		Technical support	Email address
3000 (Toms Building Materials)	210	55 Right Road, The Foundry, Los Angeles, CA, USA, 90210		Ship to	Postal address
3000 (Toms Building Materials)	220	100 Boulder St, Suite 23, Santa Monica, CA, USA, 90212		Bill to	
3000 (Toms Building Materials)	1000	86 21 555 1000		Business inquiries	
3000 (Toms Building Materials)	1000	86 21 555 1000		Billing inquiries	
3010 (Lingsat Ltd)	1100	54 11 4777 1221		General inquiries	
3010 (Lingsat Ltd)	1200	noticas@lingsat.com.ar		Sales inquiries	
3010 (Lingsat Ltd)	1300	Dorego 1708, Buenos Aires, Argentina, 1412		Ship to	

In Figure 7.7, “each PARTY and FACILITY *may be* **contacted via** one or more CONTACT MECHANISM(s), and each ORDER *may be* **having** one or more CONTACT MECHANISM(s).” Notice we have changed the relationship names from the pattern to be appropriate for the type of entity that has the contact mechanisms. Each of the different associative entities, PARTY CONTACT MECHANISM, FACILITY CONTACT MECHANISM, and ORDER CONTACT MECHANISM, *may be* **used for the purpose of** one or more CONTACT MECHANISM PURPOSE(s) that *must be* **classified by** one and only one CONTACT MECHANISM PURPOSE TYPE. This means that the association between a FACILITY and a CONTACT MECHANISM of POSTAL ADDRESS may be used for a “Ship to” purpose, as well as other purposes, such as “Order confirmation” specifying that the confirmation of an order be sent to that facility's address.

You can also see that “each PARTY CONTACT MECHANISM *may be* **used for** one or more CONTACT MECHANISM USAGE(s) and each CONTACT MECHANISM USAGE *must be* **classified by** one and only one CONTACT MECHANISM USAGE TYPE.” This means that there may be two or more PARTY CONTACT MECHANISM USAGE(s) for a PARTY CONTACT MECHANISM, such as “Business” and “Personal,” at the same time. ORDER and FACILITY have no CONTACT MECHANISM USAGE(s). The reason for this is that ORDER and FACILITY contact mechanisms are only ever used for business usage in our scenario. This means there is no need for a CONTACT MECHANISM USAGE TYPE in this example.

In Table 7.10, there are numerous examples of capturing many different contact mechanisms in different ways. For example, PARTY “XYZ Corporation” has three different contact mechanisms: a POSTAL ADDRESS beginning with “100 Main Street Suite 819...,” an ELECTRONIC ADDRESS of contact mechanism type “Email address” of “xyz@xyzcorp.com,” and a TELECOMMUNICATIONS NUMBER

of contact mechanism type “Telephone number” of “1 917 555 2100.” The “xyz@xyzcorp.com” PARTY CONTACT MECHANISM has two different CONTACT MECHANISM PURPOSES of “General inquiries” and “Billing inquiries.” All of these contact mechanisms have a CONTACT MECHANISM USAGE TYPE of “Business.”

Another example of this in Table 7.10 is the PARTY “Manu Collet”; this PARTY has a CONTACT MECHANISM “91 11 2623 665” that is both a “Telephone number” and a “Fax number.” Thus, in this model one instance of “91 11 2623 665” is created in TELECOMMUNICATIONS NUMBER, a subtype of CONTACT MECHANISM, with two CONTACT MECHANISM CATEGORY(s) of “Telephone number” and “Fax number.” It is worthwhile stating that both “Telephone number” and “Fax number” may roll up to the CONTACT MECHANISM CATEGORY of “Telecommunications number,” and it is classified as a CONTACT MECHANISM CATEGORY TYPE of “Contact mechanism type.”

Another interesting aspect of the pattern is again illustrated by the PARTY “Manu Collet.” This PARTY can be contacted via the POSTAL ADDRESS, another subtype of CONTACT MECHANISM, beginning with “Andheri Kurla Road, Marol…,” which is of CONTACT MECHANISM CATEGORY “Postal address.” This PARTY CONTACT MECHANISM instance has no related CONTACT MECHANISM PURPOSE or CONTACT MECHANISM USAGE. It is not mandatory to have a usage or a purpose. In fact, the usage or purpose for a specific contact mechanism may not have been discovered yet.

Table 7.11 shows four different FACILITY CONTACT MECHANISM instances. Most of the CONTACT MECHANISM(s) are of CONTACT MECHANISM CATEGORY “Postal address.” This is not surprising if you consider that a FACILITY can be defined as “a physical structure that is used to accommodate people or organizations.” It is also true to say that a FACILITY may also have telephone numbers, such as

how the FACILITY “XYZ Corporation head office” has a telephone number that is used as a “General emergency number” (a CONTACT MECHANISM PURPOSE TYPE) in case of fire.

Table 7.11 FACILITY Contact Mechanisms, Address, Phone Number, and Email

FACILITY. FACILITY ID	FACILITY. FACILITY NAME	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)[4]	CONTACT MECHANISM CATEGORY. NAME	CONTACT MECHANISM PURPOSE TYPE. NAME
10901	XYZ Corporation head office	100 Main Street, Suite 819, The Coalman Building New York, NY, USA, 10019	Postal address	
10901	XYZ Corporation head office	212 555 7111	Telephone number	General Emergency number
10903	MS warehouse	55 Right Road, The Foundry, Los Angeles, CA, USA, 90210	Postal address	Ship to
10906	Charing building site	55 Charing Cross Road, Flat 4, Shaldon Mansions, London, UK, WC2H 0LA	Postal address	

Finally, examine the ORDER CONTACT MECHANISM, as seen Figure 7.7. This entity captures the CONTACT MECHANISM(s) used by an ORDER. Most orders have a “Ship to” address (that is, a delivery address), and a “Bill to” address (that is, where the invoice is to be sent). As we discussed in the level 2 pattern, there may also be a need to specify the party associated with any order's contact mechanism, and thus there is an optional relationship from ORDER CONTACT MECHANISM to PARTY supported by the optional attribute **contact mechanism for party id**. In Table 7.12 you see the case of the ORDER “47742” having contact mechanisms with four different CONTACT MECHANISM PURPOSE TYPE(s), “Ship to,” “Bill to,” “Shipping instruction inquiries” (allowing for any questions regarding shipping), and “Payment follow up” (allowing a number to call if there is a need to follow up on payment). ORDER “47742” for “Cement” has a CONTACT MECHANISM beginning with “100 Main Street, Suite 819…” maintained in the CONTACT MECHANISM TYPE as a “Postal address.” This postal address has two different CONTACT MECHANISM PURPOSE(s) of “Ship to” and “Bill to” CONTACT MECHANISM PURPOSE TYPE(s). This is the same address, but with two different reasons for using it.

Table 7.12 ORDER Contact Mechanisms, Address, Phone Number, and Email

ORDER. ORDER ID	ORDER. ORDER DESCRIPTION	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)[4]	CONTACT MECHANISM TYPE. NAME	CONTACT MECHANISM PURPOSE TYPE. NAME
47742	Cement	100 Main Street, Suite 819, The Coalman Building New York, NY, USA, 10019	Postal address	Ship to
47742	Cement	100 Main Street, Suite 819, The Coalman Building New York, NY, USA, 10019	Postal address	Bill to
47742	Cement	xyz@xyzcorp.com	Email address	Shipping instruction inquiries
47742	Cement	1 917 555 2100	Telephone number	Payment follow up
47790	Steel girders	55 Right Road, The Foundry, Los Angeles, CA, USA, 90210	Postal address	Ship to
47790	Steel girders	55 Right Road, The Foundry, Los Angeles, CA, USA, 90210	Postal address	Project work site
47790	Steel girders	100 Boulder Street, Suite 23, Santa Monica, CA, USA, 90212	Postal address	Bill to

ORDER. ORDER ID	ORDER. ORDER DESCRIPTION	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)[4]	CONTACT MECHANISM TYPE. NAME	CONTACT MECHANISM PURPOSE TYPE. NAME
47790	Steel girders	100 Boulder Street, Suite 23, Santa Monica, CA, USA, 90212	Postal address	Accounting office
47790	Steel girders	86 21 555 1000	Telephone number	Shipping instruction inquiries
47790	Steel girders	86 21 555 1000	Telephone number	Payment follow up
47799	Gold electrical wiring	Andheri Kurla Road, Marol no. 604, Mumbai, Andheri East, India, 400099	Postal address	Ship to
47799	Gold electrical wiring	91 11 2623 665	Telephone number	Payment follow up
5000	Paint	Andheri Kurla Road, Marol no. 604, Mumbai, Andheri East, India, 400099	Postal address	Ship to
5000	Paint	91 11 2623 665	Telephone number	Shipment instruction inquiries

It is common for orders to be delivered to one address and billed to another. That is certainly the case with the order for steel girders because they want their orders delivered to the site they are working at, but the billing would go to their “Accounting office” (another purpose), not the “Project work site” (another purpose). For example, in Table 7.12, ORDER “47790” for “Steel girders” has a “Ship to” address of “55 Right Road, The Foundry, Los Angeles, CA, USA, 90210” and a “Bill to” address of “100 Boulder Street, Suite 23, Santa Monica, CA, USA, 90212.”

ORDER(s) don't just have POSTAL ADDRESS(es) associated with them; they may also have TELECOMMUNICATION NUMBER(s), such as how ORDER “47799” for “Gold electrical wiring” has a telephone number “91 11 2623 665” with a CONTACT MECHANISM PURPOSE TYPE of “Payment follow up” used to contact the customer in case there is a need to follow up on their payment for the order. More and more the preferred CONTACT MECHANISM TYPE for ORDER(s) is “Electronic addresses” in modern enterprises. It is often easier to use electronic mail than it is to conduct telephone calls. In ORDER “47742” for “Cement” the “Shipment notification” (alerting the party ordering that their shipment was sent) was to be sent to an “xyz@xyzcorp.com” email address, with the CONTACT MECHANISM PURPOSE TYPE **name** of “Shipment notification.”

Finally, it is worth noting that a CONTACT MECHANISM can have more than one ORDER associated with it. ORDER “47799” for “Gold electrical wiring” has a “Ship to” address of “Andheri Kurla Road, Marol no. 604, Mumbai, Andheri East, India, 400099” and so does ORDER “5000” for “Paint.”

When Should This Pattern Be Used?

We use this pattern:

- **When an enterprise wishes to have a consistent way to model all types of contact mechanisms** By having the supertype of CONTACT MECHANISM, you can manage all types of contact mechanisms in a similar manner, and whenever there is a need to relate an entity to contact mechanisms, instead of relating it to specific types of contact mechanism entities (such as TELECOMMUNICATIONS NUMBER, POSTAL ADDRESS, or ELECTRONIC ADDRESS), you can relate it to a consolidated entity of CONTACT MECHANISM. This reduces the need for many associative entities in the model, thus simplifying the data model. Also, instead of having separate classifications for different types of contact mechanism entities, you can use a generalized CONTACT MECHANISM CATEGORY to provide a very flexible classification structure.
- **When there is a need to classify contact mechanisms in many different ways:** The flexible CONTACT MECHANISM CATEGORY data model structure in this pattern allows for great flexibility in managing contact mechanism data. Contact mechanisms may be categorized in any number of additional ways, such as by “Location type” (“Home,” “Office”), by “Priority type” (“Primary,” “Secondary,” “Tertiary”), or even by new classifications that may emerge over time.
- **When flexible management of contact mechanisms is needed:** The addition of the supertype CONTACT MECHANISM means that if any new type of contact mechanism is needed, it can be added as another type of contact mechanism. This provides more flexibility than having different entities for different types of contact mechanisms as in the level 2 pattern. The addition of the Level 3 Classification Pattern helps to support a very flexible way to manage the allowable list of CONTACT MECHANISM CATEGORY(s) and the

classification of those categories within CONTACT MECHANISM CATEGORY TYPE.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern is more generalized and abstract, and thus, it is more difficult to use as a means to identify scope or for use in gathering and validating data requirements:** This pattern is usually difficult for nontechnical people to understand. Still, this may not always be the case, as more and more nontechnical people are able to grasp generalized concepts used in data modeling.
- **Because this is more generalized, there can be a loss of specificity and business rules in the data model:** For example, in the level 2 pattern we were able to model a one-to-many relationship from PARTY to PARTY ELECTRONIC ADDRESS showing that there was not a need for a many-to-many relationship or for maintaining each electronic address in its own entity. With this pattern, all contact mechanisms are treated similarly, so when relating a PARTY to a CONTACT MECHANISM, the PARTY CONTACT MECHANISM provides for a many-to-many relationship, whether or not it is needed. Also, a CONTACT MECHANISM may be classified by any CONTACT MECHANISM CATEGORY. For example, a POSTAL ADDRESS could conceivably be classified with a "Technology type" of "Fax machine" if the pattern is misused.
- **Though we have consolidated contact mechanisms for a more elegant data modeling solution for easier management of the model, there is still a method for consolidating further:**

For example, although we have consolidated CONTACT MECHANISM(s), what about a consolidated way to connect any entity to the CONTACT MECHANISM instead of creating many different ENTITY CONTACT MECHANISM variants? Can a single CONTACT MECHANISM APPLICATION entity be created? This approach is covered in the next pattern.

Synopsis

The Level 3 Contact Mechanism Pattern encapsulates the different types of contact mechanisms within a generalized CONTACT MECHANISM supertype. This approach can simplify the model so that there are not repeated relationships (and attributes, to a lesser extent) for each type of contact mechanism type needed for an entity (as was the case in the level 2 pattern). The CONTACT MECHANISM also provides a place holder for all new types of contact mechanism types that may be discovered over the life cycle of a project. The main reasons to use this pattern are to provide flexibility to maintain any contact mechanism classifications, such as type, usage, purpose, location, and priority, as well as to have a common structure for all types of contact mechanisms. This helps to minimize the effect of adding new types of contact mechanisms to your data model.

The addition of the Level 3 Classification Pattern allows for any CONTACT MECHANISM to be classified any number of ways using the CONTACT MECHANISM CATEGORY CLASSIFICATION, which is related to the CONTACT MECHANISM CATEGORY entity. CONTACT MECHANISM CATEGORY(s) can be within higher-level categories. For example, the “Telecommunications number” category is a higher-level category for a “Telephone number,” “Fax number,” or “Pager number.” The CONTACT MECHANISM CATEGORY may also be classified within CONTACT

MECHANISM CATEGORY TYPE(s). For example, “Electronic address,” “Telecommunications number,” and “Postal Address” may be classified within a CONTACT MECHANISM CATEGORY TYPE of “Contact mechanism types,” and “Primary,” “Secondary,” and “Tertiary” may be classified within a CONTACT MECHANISM CATEGORY TYPE of “Priority type.” This allows great flexibility in categorizing a contact mechanism. For example, earlier in this chapter we mentioned that some enterprises may wish to capture “Home phone” and “Office phone” as “Location types” of contact mechanisms, or “Mobile device” and “Pager device” as “Technology types.” With this flexibility comes the danger of classifying CONTACT MECHANISM(s) within CONTACT MECHANISM CATEGORY(s) that make no sense. For example, a POSTAL ADDRESS should not be classified as a “Fax number.”

The pattern should be used when the enterprise desires to have a consistent way to manage contact mechanisms, when there is a need to have many ways to flexibly classify contact mechanisms, and when a great degree of flexibility is needed to be able add any new types of contact mechanisms. This pattern can also help if the subject area is not very specific or well understood or the full list of contact mechanisms (or types) is not known. Or you could use the pattern if contact mechanisms are known, but you believe that processes or circumstances could change in the future. In the scenario described in the section, ABC Building Corporation was not very specific about the types of contact mechanisms that it wanted to add later, only that it needed a robust data model that would support its future needs with a minimum of impact. If an enterprise wishes to consolidate contact mechanisms into a single entity so that any other entity that has contact mechanisms (such as a PARTY, ORDER, or FACILITY) can reference all their contact mechanisms with a single relationship (instead of using

multiple relationships for each type of contact mechanism), this pattern can help.

One possible weakness of this pattern lies in the fact that a generalized concept (CONTACT MECHANISM) has been introduced and this may be difficult for nontechnical people to understand. Also, because the pattern uses more generalization, there is a certain amount of specificity that can be lost. For example, there can be a tendency to treat all contact mechanisms similarly, when certain types of contact mechanisms may need a different data model structure. Of course, this may be accommodated by relating data to the CONTACT MECHANISM subtypes of TELECOMMUNICATIONS NUMBER, ELECTRONIC ADDRESS, and POSTAL ADDRESS. Although this pattern offers consolidation of contact mechanism data, there is still a way to consolidate even more, as we discuss in the next section.

Level 4 Contact Mechanism Pattern

The Level 4 Contact Mechanism Pattern is even a more flexible solution. We consider it a ‘plug-and-play’ type pattern. What we mean by this is that once this pattern has been added, any ENTITY that needs access to all CONTACT MECHANISM(s) and CONTACT MECHANISM CATEGORY(s) (including type, purpose, usage, location, priority, and any other classification) just needs to plug itself into (that is, create the **having**, **the contact mechanism for** relationship) the CONTACT MECHANISM APPLICATION, and it has access to everything it needs. This pattern suits enterprises that require highly flexible and consistent data solutions that minimize the impact of change to their underlying data architecture.

Why Do We Need This Pattern?

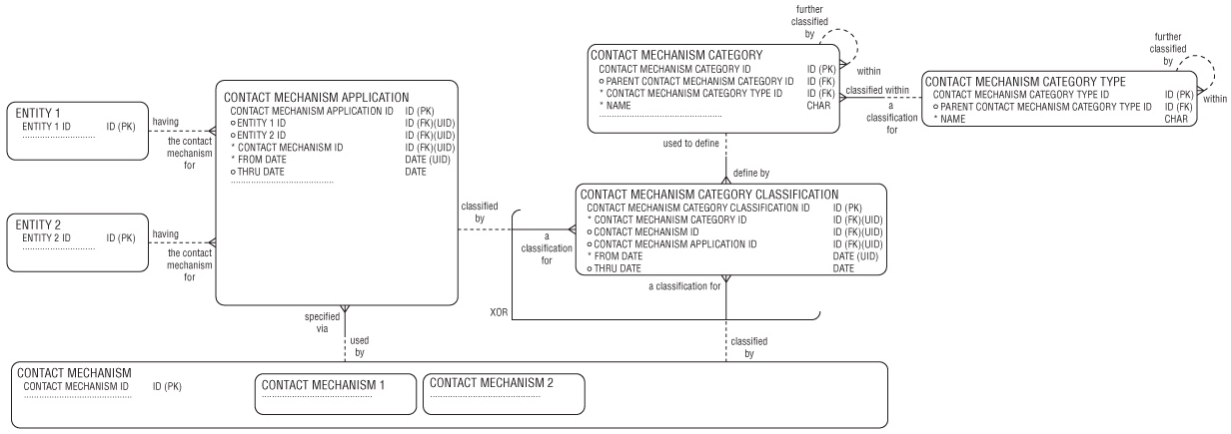
This pattern is quite similar to the level 3 pattern, but it differs in two significant ways. First, it encapsulates all of the relationships that every entity has with CONTACT MECHANISM into a single consolidated entity called CONTACT MECHANISM APPLICATION. Second, it maintains all contact mechanism usages and purposes as CONTACT MECHANISM CATEGORY(s) that are CONTACT MECHANISM CATEGORY TYPE(s) of “Purpose type” and “Usage type.”

This pattern is almost the equivalent of an interface in programming. In other words, by attaching to the interface, CONTACT MECHANISM APPLICATION, any new or existing entity can have access to all of the data for CONTACT MECHANISM(s), including the various ways to classify contact mechanisms and the associations that are needed between the entity and the contact mechanism. This is very useful for an enterprise that has a dynamic data environment and doesn't want to have to re-investigate what is needed to maintain the contact mechanism data for every new entity that needs it. By using this pattern, an enterprise can standardize all CONTACT MECHANISM data and all relationships that are needed by entities that require contact mechanism data.

How Does This Pattern Work?

Figure 7.8 describes the pattern. ENTITY 1 and ENTITY 2 are the entities that the data professional is investigating. Each of these entities connects to the CONTACT MECHANISM APPLICATION in the same way. Those relationships state that “each ENTITY 1 (or ENTITY 2) *may be* **having** *one or more* CONTACT MECHANISM APPLICATION(s), and each CONTACT MECHANISM APPLICATION(s) may be **the contact mechanism for** one and only one ENTITY 1 (or ENTITY 2).”

Figure 7.8 Level 4 Contact Mechanism Pattern



Each CONTACT MECHANISM (and all of its subtypes) *may be* **used by** one or more CONTACT MECHANISM APPLICATION(s). Also, the CONTACT MECHANISM APPLICATION *may be* **classified by** one or more CONTACT MECHANISM CATEGORY(s) (via the associative entity CONTACT MECHANISM CATEGORY ASSOCIATION) that classify the contact mechanism in any number of ways, such as by contact mechanism type, purpose, usage, priority, location, and so on. For example, there could be CONTACT MECHANISM CATEGORY(s) of “Personal” and “Business” that are within a classification of CONTACT MECHANISM CATEGORY TYPE of “Usage type,” which refers to how contact mechanisms are customarily utilized.” Another example would be CONTACT MECHANISM APPLICATION having one or more CONTACT MECHANISM CATEGORY(s), such as “Bill to” and “Ship to,” that are categories that are within a classification of CONTACT MECHANISM CATEGORY TYPE of “Purpose type.” The beauty of this pattern is that contact mechanisms can be classified very flexibly in any number of ways and new types of classifications may be added if they emerge. For example, in Figure 7.2 you saw attributes like **primary street address part** and **primary city**. The “primary” designation refers to a level of importance or “Priority type” tied to the contact mechanism. Therefore, you could capture the CONTACT MECHANISM CATEGORY(s) of “Primary,”

“Secondary,” and so on within a “Priority type” CONTACT MECHANISM CATEGORY TYPE, and this would dictate which contact mechanism to call first. For example, there could be the two telephone numbers for a person, and the first telephone number could be related to a CONTACT MECHANISM CATEGORY of “Primary” and the second telephone number could be related to a CONTACT MECHANISM CATEGORY of “Secondary.” Thus, this would mean to first try the person at their primary number and then at the secondary number.

The CONTACT MECHANISM APPLICATION works in the same fashion as all of the associative entities do for the Level 3 Contact Mechanism Pattern (for example, FACILITY CONTACT MECHANISM, PARTY CONTACT MECHANISM, and ORDER CONTACT MECHANISM in Figure 7.7). However, all the relationships from any entity to a CONTACT MECHANISM is supported via a single associative entity, CONTACT MECHANISM APPLICATION.

Creating this interface entity of CONTACT MECHANISM APPLICATION simplifies the task of ensuring every entity is connected to the CONTACT MECHANISM, the classifications of those CONTACT MECHANISM(s), and the classification of the CONTACT MECHANISM APPLICATION(s) all in a consistent fashion. There is no need to create a new ENTITY CONTACT MECHANISM entity (such as PARTY CONTACT MECHANISM or FACILITY CONTACT MECHANISM) for every new ENTITY that needs access to CONTACT MECHANISM, as you saw in Figures 7.6 and 7.7; you need only add a foreign key within the CONTACT MECHANISM APPLICATION referencing the new entity. By connecting to CONTACT MECHANISM APPLICATION, ENTITY 1 and ENTITY 2 get access to every different subtype of CONTACT MECHANISM (CONTACT MECHANISM 1, CONTACT MECHANISM 2, and so on) that were defined. Also, every different CONTACT MECHANISM CATEGORY that has been captured is available as a way to classify ENTITY 1 and

ENTITY 2 via the CONTACT MECHANISM APPLICATION(s) relationship to CONTACT MECHANISM CATEGORY CLASSIFICATION.

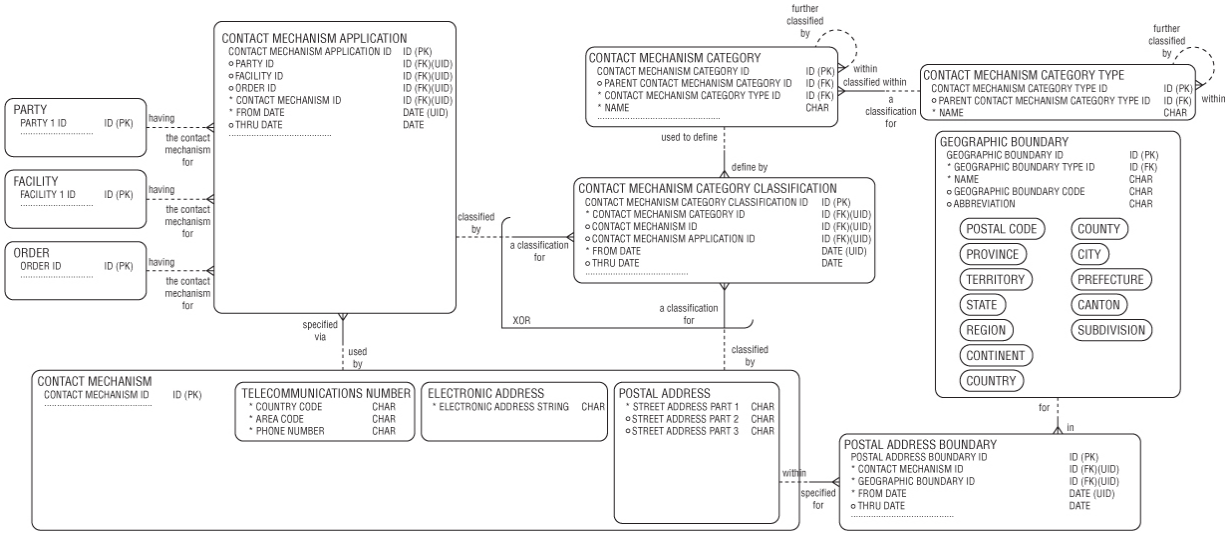
Notice that CONTACT MECHANISM CATEGORY CLASSIFICATION has an exclusive arc over the relationships between it and CONTACT MECHANISM APPLICATION and CONTACT MECHANISM(s). We are using the same Level 3 Classification Pattern and allow the same types of classification structures to support both entities. For example, you may want to relate a CONTACT MECHANISM CATEGORY TYPE of “Usage type” to either the contact mechanism or the contact mechanism application, depending on the need. Thus, you may choose to relate a “Usage type” of “Business” to the instance of a person's telephone number of “212 919 9999” (because that person uses it for business) or to the telephone number in general for any party that has that same telephone number (because that telephone number is used for business regardless of the party involved). The exclusive arc illustrates the rule that you classify either the contact mechanism or the contact mechanism application, but not both, so you don't cause inconsistencies.

Figure 7.9 illustrates how the pattern would work. We will finish the scenario that we have described in all the previous sections of the chapter by saying that ABC Building Corporation wants to go ahead and build the contact mechanism system in house, but in the most flexible manner possible. Senior management has decided to standardize all of the CONTACT MECHANISM(s) across the enterprise as a whole to be POSTAL ADDRESS, ELECTRONIC ADDRESS, and TELECOMMUNICATIONS NUMBER. They want to make sure they can classify each contact mechanism by its type, purpose, usage, technology, location, and priority. They also want to make sure that if any new type of contact mechanism is discovered, they can easily add it and also

that every entity that maintains contact mechanism data has access to the new type of contact mechanism.

> **Note**
>
> In Figure 7.9, the model has entities of POSTAL ADDRESS BOUNDARY and GEOGRAPHIC BOUNDARY in order to accommodate all different types of geographic areas that help form domestic and international postal addresses. This data model structure is further developed and discussed in the next section called “Contact Mechanism Pattern with Geographic Boundary.”

Figure 7.9 Example of using a Level 4 Contact Mechanism Pattern



Based on this need, the data professional starts the process of data modeling by creating Figure 7.9. The first phase of the development is the integration of PARTY, FACILITY, and ORDER; each of these entities gets attached to the CONTACT MECHANISM APPLICATION using the same type of relationship.

Table 7.13 further illustrates how this pattern works for PARTY. PARTY “XYZ Corporation” with **party id** “1001” has four different ways to contact it, or in other words, four different “Contact mechanism types.”

- A “Postal address,” “100 Main Street, Suite 819,” (The first row in the table) used for the “Purpose type” of “Ship to” (The second row in the table).
- An “Email address,” “xyz@xyzcorp.com,” used for the “Usage type” of “Business”
- A “Telephone number,” “1 917 555 2101,” that has a “Location type” of “Reception phone”
- A “Fax number,” “1 917 555 2100,” that has a “Technology type” of “Fax machine.” Notice that just because this is a contact mechanism type of “Fax number” does not mean that it is connected to a fax machine because it could be tied to a PC or telephone line, and the technology type helps you know what type of technology is being used for this contact mechanism.

Table 7.13 CONTACT MECHANISM APPLICATION, PARTY Contact Mechanisms, Address, Phone Number, and Electronic Address

CONTACT MECHANISM APPLICATION.PARTY ID (PARTY NAME)[2]	CONTACT MECHANISM APPLICATION.CONTACT MECHANISM ID	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)[4]	CONTACT MECHANISM CATEGORY. NAME	CONTACT MECHANISM CATEGORY TYPE. NAME
1001 (XYZ Corporation)	100	100 Main Street, Suite 819, The Coalman Building New York, NY, USA, 10019	Postal address	Contact mechanism type
1001 (XYZ Corporation)	100	100 Main Street, Suite 819, The Coalman Building New York, NY, USA, 10019	Ship to	Purpose type
1001 (XYZ Corporation)	200	xyz@xyzcorp.com	Email address	Contact mechanism type
1001 (XYZ Corporation)	200	xyz@xyzcorp.com	Business	Usage type
1001 (XYZ Corporation)	350	1 917 555 2101	Telephone number	Contact mechanism type
1001 (XYZ Corporation)	350	1 917 555 2101	Reception phone	Location type
1001 (XYZ Corporation)	300	1 917 555 2100	Fax number	Contact mechanism type
1001 (XYZ Corporation)	300	1 917 555 2100	Fax machine	Technology type
1005 (Nadine Gerard)	400	1 805 555 4534	Mobile number	Contact mechanism type
1005 (Nadine Gerard)	400	1 805 555 4534	Business	Usage type
1005 (Nadine Gerard)	400	1 805 555 3434	Technical support	Purpose type
2002 (Ed Smith)	500	44 20 5555 1234	Mobile number	Contact mechanism type
2002 (Ed Smith)	500	44 20 5555 1234	Personal	Usage type
2002 (Ed Smith)	550	44 20 5555 7983	Business	Usage type
2004 (Manu Collet)	600	91 11 2623 665	Telephone number	Contact mechanism type
2004 (Manu Collet)	600	91 11 2623 665	Fax number	Contact mechanism type

CONTACT MECHANISM APPLICATION.PARTY ID (PARTY NAME)[2]	CONTACT MECHANISM APPLICATION.CONTACT MECHANISM ID	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)[4]	CONTACT MECHANISM CATEGORY. NAME	CONTACT MECHANISM CATEGORY TYPE. NAME
2004 (Manu Collet)	600	91 11 2623 665	Business	Usage type
2004 (Manu Collet)	600	91 11 2623 665	Office phone	Location type
2004 (Manu Collet)	600	91 11 2623 665	Landline	Technology type
2004 (Manu Collet)	600	91 11 2623 665	Fax machine	Technology type
2004 (Manu Collet)	320	Andheri Kurla Road, Marol no. 604, Mumbai, Andheri East, India, 400099	Postal address	Contact mechanism type
2004 (Manu Collet)	320	Andheri Kurla Road, Marol no. 604, Mumbai, Andheri East, India, 400099	Primary	Priority type
2004 (Manu Collet)	700	Mcollet@mtln.net.in	Email address	Contact mechanism type
2004 (Manu Collet)	700	Mcollet@mtln.net.in	Business	Usage type
2004 (Manu Collet)	800	www.my_space.com/blog/Mcollet	Blog address	Contact mechanism type
2004 (Manu Collet)	800	www.my_space.com/blog/Mcollet	Personal	Usage type
2004 (Manu Collet)	900	www.techexperts.com/techtalk	Chat room address	Contact mechanism type
2004 (Manu Collet)	900	www.techexperts.com/techtalk	Technical support	Purpose type
3000 (Toms Building Materials)	210	55 Right Road, The Foundry, Los Angeles, CA, USA, 90210	Postal address	Contact mechanism type
3000 (Toms Building Materials)	210	55 Right Road, The Foundry, Los Angeles, CA, USA, 90210	Ship to	Purpose type
3000 (Toms Building Materials)	220	100 Boulder Street, Suite 23, Santa Monica, CA, USA, 90212	Postal address	Contact mechanism type

CONTACT MECHANISM APPLICATION.PARTY ID (PARTY NAME)[2]	CONTACT MECHANISM APPLICATION.CONTACT MECHANISM ID	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)[4]	CONTACT MECHANISM CATEGORY. NAME	CONTACT MECHANISM CATEGORY TYPE. NAME
3000 (Toms Building Materials)	220	100 Boulder Street, Suite 23, Santa Monica, CA, USA, 90212	Bill to	Purpose type
3000 (Toms Building Materials)	1000	86 21 555 1000	Telephone number	Contact mechanism type
3000 (Toms Building Materials)	1000	86 21 555 1000	Shipment instruction inquiries	Purpose type
3000 (Toms Building Materials)	1000	86 21 555 1000	Payment follow up	Purpose type
3010 (Lingsat Ltd)	1100	54 11 4777 1221	Telephone number	Contact mechanism type
3010 (Lingsat Ltd)	1100	54 11 4777 1221	Headquarters phone number	Purpose type
3010 (Lingsat Ltd)	1200	noticas@lingsat.com.ar	Email address	Contact mechanism type
3010 (Lingsat Ltd)	1200	noticas@lingsat.com.ar	Business	Usage type
3010 (Lingsat Ltd)	1300	Dorego 1708, Buenos Aires, Argentina, 1412	Postal address	Contact mechanism type
3010 (Lingsat Ltd)	1300	Dorego 1708, Buenos Aires, Argentina, 1412	Ship to	Purpose type

Another example is the party “Manu Collet” with **party id** “2004,” who has a “Telephone number,” “91 11 2623 665,” that doubles as a “Fax number” (and thus is related to two CONTACT MECHANISM CATEGORY (s) within the same CONTACT MECHANISM CATEGORY TYPE of “Contact mechanism type”). This telecommunications number has a “Usage type” of “Business” and a “Location type” of “Office phone” (meaning that if you ring this number you will be put through to Manu's office location). Interestingly the same number has two technology types, one of “Landline” and one of “Fax machine,” because this number may be connected to a physical (non-wireless) telephone line or a fax machine.

Note

There is a limitation on this classification pattern of not being able to specify that if something is in one category, then it also belongs to another category. For example, the pattern will not accommodate the following scenario: if there is a telecommunications number that doubles as a “Fax number” and a “Telephone number” and when it is used as a “Fax number,” it is for the purpose of “Order confirmation,” but if the same telecommunications number is used as a “Telephone number,” then it is used for the purpose of “General inquires.” If this is needed, it could be handled by using the Level 3 Recursive Pattern with Rules (see Chapter 4) and specifying a rule that for this contact mechanism, the purpose type of “Order confirmation” is only valid if the contact mechanism type is “Fax.” Alternatively, this could be handled by enhancing this pattern with one of the business rules patterns in chapter 8, or you may even consider using both the Business Rules Patterns and the Level 3 Recursive Pattern with Rules.

The party “Manu Collet” also has a few different electronic addresses—an “email address,” “Mcollet@mtln.net.in,” that he uses for “Business”; a “Blog address” for “Personal” usage, “www.my_space.com/blog/mcollet”; and finally a tech expert's “Chat room address” of

“www.techexperts.com/techtalk” used for “Technical support” that he uses to answer technical questions. This party also maintains a POSTAL ADDRESS at “Andheri Kurla Road …,” which is classified as his “Primary” address.

If you examine Table 7.14, you should see that FACILITY has mostly the same structure and types of data as found in Table 7.13 for PARTY. The reason for this is that the way you connect FACILITY to the contact mechanism information is exactly the same as how you do it for PARTY. So the structure shown in Figures 7.8 and 7.9 is beneficial for two reasons:

- First, it takes away any ambiguity of how to connect new entities to their contact mechanisms.
- Second, a consistent structure allows programs to create reusable software. Selects, updates, and deletes for a party's contact mechanism would be very similar if not the same as the selects, updates, and deletes for a facility's contact mechanisms.

Table 7.14 CONTACT MECHANISM APPLICATION, FACILITY Contact Mechanisms, Address, Phone Number, and Electronic Address

CONTACT MECHANISM APPLICATION.FACILITY ID (FACILITY NAME)	CONTACT MECHANISM APPLICATION.CONTACT MECHANISM ID	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ ELECTRONIC ADDRESS/ POSTAL ADDRESS)[4]	CONTACT MECHANISM CATEGORY.NAME	CONTACT MECHANISM CATEGORY TYPE.NAME
10901 (XYZ Corporation head office)	100	100 Main Street, Suite 819, The Coalman Building New York, NY, USA, 10019	Postal address	Contact mechanism type
10901 (XYZ Corporation head office)	350	1 917 555 2101	Telephone number	Contact mechanism type
10901 (XYZ Corporation head office)	350	1 917 555 2101	Emergency	Usage type
10903(MS warehouse)	210	55 Right Road, The Foundry, Los Angeles, CA, USA, 90210	Postal address	Contact mechanism type
10903 (MS warehouse)	210	55 Right Road, The Foundry, Los Angeles, CA, USA, 90210	Ship to	Purpose type
10906 (Charing building site)	644	55 Charing Cross Road, Flat 4, Shaldon Mansions, London, UK, WC2H 0LA	Postal address	Contact mechanism type

If you examine Table 7.14, you see that a FACILITY “XYZ Corporation head office” has a POSTAL ADDRESS of “100 Main Street, Suite 819,...,” much in the same way as the PARTY “Manu Collet” had a POSTAL ADDRESS of “Andheri Kurla Road....” You can also see that this FACILITY has a “Telephone number” that is used for an “Emergency” purpose. This number is “1 917 555 2101” and is the internal emergency number used in the building. The point is that the same information about contact mechanisms can be maintained in the same manner for both FACILITY and PARTY.

Maybe a richer example of this phenomenon can be seen by looking at ORDER(s). An ORDER may be make use of any number of contact mechanism types, such as a postal address, email address, telephone number, fax number, and so on.

If you examine Table 7.15, you see one particular ORDER “47742” for “Cement” that has a postal address of “100 Main Street, Suite 819,...” for the purpose of “Bill to,” an “Email address” of “xyz@xyzcorp.com” for the purpose of “Shipment notification” (to alert customers when their order was shipped), and a “Telephone number” of “1 917 555 2100” for “Payment follow up,” which is the number that may be called if ABC Building Corporation wants to call to follow up on payment due for the order.

Table 7.15 CONTACT MECHANISM APPLICATION, ORDER Contact Mechanisms, Address, Phone Number, and Electronic Address

CONTACT MECHANISM APPLICATION. ORDER ID	ORDER.ORDER DESCRIPTION	CONTACT MECHANISM APPLICATION.CONTACT MECHANISM ID	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)[4]	CONTACT MECHANISM CATEGORY. NAME	CONTACT MECHANISM CATEGORY TYPE. NAME
47742	Cement	100	100 Main Street, Suite 819, The Coalman Building New York, NY, USA, 10019	Postal address	Contact mechanism type
47742	Cement	100	100 Main Street, Suite 819, The Coalman Building New York, NY, USA, 10019	Bill to	Purpose type
47742	Cement	200	xyz@xyzcorp.com	Email address	Contact mechanism type
47742	Cement	200	xyz@xyzcorp.com	Shipment notification	Purpose type
47742	Cement	350	1 917 555 2100	Telephone number	Contact mechanism type
47742	Cement	350	1 917 555 2100	Payment follow up	Purpose type
47790	Steel girders	210	55 Right Road, The Foundry, Los Angeles, CA, USA, 90210	Postal address	Contact mechanism type
47790	Steel girders	210	55 Right Road, The Foundry, Los Angeles, CA, USA, 90210	Ship to	Purpose type
47790	Steel girders	220	100 Boulder Street, Suite 23, Santa Monica, CA, USA, 90212	Postal address	Contact mechanism type
47790	Steel girders	220	100 Boulder Street, Suite 23, Santa Monica, CA, USA, 90212	Bill to	Purpose type

CONTACT MECHANISM APPLICATION. ORDER ID	ORDER.ORDER DESCRIPTION	CONTACT MECHANISM APPLICATION.CONTACT MECHANISM ID	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)[4]	CONTACT MECHANISM CATEGORY. NAME	CONTACT MECHANISM CATEGORY TYPE. NAME
47790	Steel girders	1000	86 21 555 1000	Telephone number	Contact mechanism type
47790	Steel girders	1000	86 21 555 1000	Business inquiries	Purpose type
47790	Steel girders	1000	86 21 555 1000	Billing inquiries	Purpose type
47799	Gold electrical wiring	320	Andheri Kurla Road, Marol no. 604, Mumbai, Andheri East, India, 400099	Postal address	Contact mechanism type
47799	Gold electrical wiring	320	Andheri Kurla Road, Marol no. 604, Mumbai, Andheri East, India, 400099	Ship to	Purpose type
47799	Gold electrical wiring	600	91 11 2623 665	Telephone number	Contact mechanism type
47799	Gold electrical wiring	600	91 11 2623 665	Business	Usage type
5000	Paint	320	Andheri Kurla Road Marol no. 604, Mumbai, Andheri East, India, 400099	Postal address	Contact mechanism type
5000	Paint	320	Andheri Kurla Road Marol no. 604, Mumbai, Andheri East, India, 400099	Ship to	Purpose type
5000	Paint	600	91 11 2623 665	Fax number	Contact mechanism type
5000	Paint	600	91 11 2623 665	Business	Usage type

The ORDER has exactly the same needs that a PARTY or a FACILITY does. It is interesting to note that two orders (“47799” for “Gold electrical wiring” and “5000” for “Paint”) have a telephone number of “91 11 2623 665” with a “Usage type” of “Business.” This illustrates one issue with the flexibility provided by this pattern. In the previous patterns ORDER(s) did not need a CONTACT MECHANISM USAGE because orders are only used in business. What then is the point in having a “Usage type” for orders? Depending on the circumstances, most things can have different usages, even orders. For example, you may order a technical book for “Business” use, and you may also order a spy novel for “Personal” reasons. This pattern lets you support these circumstances if needed.

Notice that with this pattern, the CONTACT MECHANISM APPLICATION already has an optional **party id** attribute that allows a party to be associated with the order's contact mechanism, if needed (or it can be used for any other entity's contact mechanism, such as facility). Thus, there could be an order id and also a party id in some CONTACT MECHANISM APPLICATION(s) that are for specific orders. For example, in an order, if there was a need to associate a particular person's name to the telephone number of “1 917 555 2100” for the purpose type of “Payment follow up” (the sixth row in Table 7.15), then this could be done by updating the optional foreign key of **party id** in CONTACT MECHANISM APPLICATION. As we discussed previously, this is very different than recording the party's contact mechanism (which would be in Table 7.13) for an order, because when we are recording the **party id** for an order's CONTACT MECHANISM APPLICATION, we are creating a separate relationship from the order contact mechanism to a party, signifying that this contact information recorded is for use within a specific order. This could be important for parties that only want to give their contact information for

the order and keep that separate from their ongoing contact mechanisms, which would be maintained in the CONTACT MECHANISM APPLICATION that just has a **party id**.

When Should This Pattern Be Used?

We use this pattern:

- **If an enterprise has made a commitment to develop very flexible data models:** It is important to use this type of model only if the enterprise fully understands the value and effort that needs to be put into the creation of a fully flexible data strategy. This pattern needs the enterprise to commit to a standard way of dealing with all contact mechanisms.
- **When an enterprise wishes to consolidate common entities and relationships in order to more easily and consistently manage the data model:** This style of modeling will result in far fewer entities and relationships, and it will allow handling all contact mechanisms in a very consistent way. We have seen more and more enterprises make the conscious decision to model in a very consistent manner. This consistent approach can help in the creation of consistent software and data architecture. There is money and time to be saved by using this ‘plug-and-play’ approach.
- **If the subject area is not very specific or well understood or the full list of contact mechanisms (and classifications of contact mechanisms) are not known:** In the preceding scenario the contracting firm was not very specific about the types of contact mechanisms that it wanted to add later. This pattern accommodates future needs in uncertain environments.
- **When the nature and the type of data maintained about various contact mechanisms is deemed to be very similar:** This pattern manages all types of

contact mechanisms in a very similar fashion and does not maintain different types of relationships and classifications for specific types of contact mechanisms.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern is not specific at all and thus does not enforce business rules in the data model:** When you attach a new entity to the Level 4 Contact Mechanism Pattern, you have access to the same type of contact mechanism data and every contact mechanism classification, including types, purposes, usages, and so on. Thus the pattern does not show the specific classifications or relationships that are appropriate to that type of contact mechanism. This means that an entity that normally does not have an email address would have the capability to store email addresses. An entity would also have available every purpose type and usage type. For example, a SHIPMENT entity would be able to associate a “Technical support” purpose or a “Personal” usage type when these classifications may not make sense for a SHIPMENT. To accommodate this, you could enhance this pattern with one of the rules patterns in this book and there could be a business rules layer to enforce rules, such as what purposes and usages are allowed for specific types of contact mechanisms.(6)
- **Because it is a generalized approach, it is easier for data modelers to become ‘lazy.’** Because of the flexible nature of this pattern it is easy for data modelers to forego the analysis of new contact mechanisms and contact mechanism classifications and

just ‘attach’ any new entity that needs contact mechanisms to this data model structure.

- **This pattern is generalized and abstract and thus more difficult to understand:** This pattern may be difficult for many to understand because it is abstract, and in our opinion, it would not be a very effective model for gathering and validating data requirements or to help define scope.

Synopsis

The Level 4 Contact Mechanism Pattern is the most flexible ‘plug-and-play pattern’ that can be used by an enterprise to automatically connect to all aspects of contact mechanisms and contact mechanism purposes.

The pattern creates an ‘interface-like’ entity called CONTACT MECHANISM APPLICATION, to which each and every entity, whether it is a current and/or future entity, can attach by using a **having, the contact mechanism for** relationship. By connecting to CONTACT MECHANISM APPLICATION, any entity can gain access to all the different contact mechanisms and contact mechanism classifications. In this pattern, either CONTACT MECHANISM(s) or CONTACT MECHANISM APPLICATION(s) may be classified by CONTACT MECHANISM CATEGORY CLASSIFICATION and thus the pattern allows classifications of a contact mechanism regardless of its application, or a specific contact mechanism that is associated with an entity, for a example, a party contact mechanism.

There are many advantages with this approach. First, it provides a consistent data modeling structure across all contact mechanisms. With this consistency an enterprise gets the benefit of building consistent application architecture by building on data models that use this pattern. It eliminates the guesswork of what to do with new entities that need contact mechanisms and contact

mechanism purposes. It is a very flexible, stable pattern that can withstand the addition of any new types of contact mechanism categorizations. If there is a new entity that needs to maintain contact information, these new contact mechanism capabilities get automatically ‘pushed’ to the entity attached via the CONTACT MECHANISM APPLICATION. This pattern does require that an enterprise is committed to a flexible data architecture.

Contact Mechanism Pattern with Geographic Boundary

Anywhere administrations have been set up, the boundaries of their jurisdiction must be defined, including government jurisdictions, such as city, state, postal code, and country, and other geographic boundaries, such as territory, canton, and prefecture, that are very useful for establishing domestic and international contact mechanism.'(7) Most enterprises and the people who work within them function within some jurisdiction that was either created by some third party (a government), like a region, or by themselves, like a sales territory. These jurisdictions, or as we call them, *geographic boundaries*, are crucial for organizing enterprises and reporting on those enterprises. The geographic boundary provides enterprises a framework from which to manage all types of geographical data and report on their effectiveness across various geographical areas.

Note

Some of the geographic boundaries, such as a sales territory or service region, have nothing to do with contact mechanisms. We don't address these geographic boundaries in this section or in this book. However, if these are applicable, they could be valid subtypes of GEOGRAPHIC BOUNDARY. We are addressing

geographic boundaries only as they relate to contact mechanisms. For example, take sales territories - we could not put a sales territory on a postcard, send it off, and have a reasonable expectation of it arriving (unless it was via internal company mail). So in this chapter, we will try to limit our discussion to the geographic boundaries that have to do with contact mechanisms.

The Contact Mechanism Pattern with Geographic Boundary is an “add on” to the previous patterns by expanding the model to support geographic boundaries to capture the jurisdictions enterprises use to manage their business.

Note

We have used some of this pattern already in the examples in Figures 7.7 (a level 3 example) and 7.9 (a level 4 example) because this pattern is at a level 3 or level 4 level of generalization.

Why Do We Need This Pattern?

Geographic boundaries are fundamentally related to contact mechanisms, especially postal addresses and to lesser extent telephone numbers and electronic addresses. For example, a postal address may contain regions, states, cities, countries and so on. Each of these areas can be considered a geographic boundary. Different third parties (governments, postal unions, or standards organizations) set up different jurisdictions. For example, in Brazil addresses have states (*estados*, for example, AC for Acre is a Brazilian state), Japan has *ku*, *shi*, and *ken* for Ward, City, and Prefecture but no concept like states or territories (as in Australia).

Geographic boundaries allow all forms of data to be categorized by their jurisdiction, which is crucial to effective data management, and also to reporting and data mining environments. For example, a CEO may ask for all of our

customers who have “Primary” postal addresses in the People's Republic of China. The geographic boundary type is COUNTRY with a value of “People's Republic of China.”

Why not model geographic boundaries as attributes of contact mechanisms, such as modeling **city**, **state**, **postal code**, and **country** as attributes of POSTAL ADDRESS, as we did in the Level 1 Contact Mechanism Pattern? Aside from the issues of redundancy that we discussed previously in this chapter, international addresses have many different types of geographic boundaries, and thus there could be many different types of attributes (**state**, **territory**, **prefecture**, **province**, and so on).

Another alternative would be to have CITY, STATE, POSTAL CODE, and COUNTRY entities with foreign keys of **city id**, **state id**, **postal code id**, and **country id**, as we did in the Level 2 Contact Mechanism Pattern. This reduces the redundancy by maintaining each city, state, postal code, and country in only one place. However, international addresses often require many other types of geographic boundaries, such as CANTON, PROVINCE, TERRITORY, PREFECTURE, and so on. Also, geographic boundaries can change! This happened to the former Soviet Union and to the former Czechoslovakia. Thus, by using a more flexible GEOGRAPHIC BOUNDARY entity for contact mechanisms, such as POSTAL ADDRESS, you can much more effectively model international addresses and also accommodate change much easier.

In Table 7.16, the South Korean address of “981 ponji, Yaum-dong, Ulsan-si, Kyongsangnamdo, South Korea” does not have a state, but instead includes a subdivision and a province. We are able to relate the address to the subdivision (Yaum-dong), which is in the city (Ulsan-si), which is in the province (Kyongsangnamdo), which is in the country (South Korea). This illustrates how with this pattern, you can take any international address and relate it to its

appropriate geographic boundary type (which would be in the GEOGRAPHIC BOUNDARY TYPE **name** attribute).

Table 7.16 GEOGRAPHIC BOUNDARIES, Geographic Boundary Associations, Contact Mechanisms

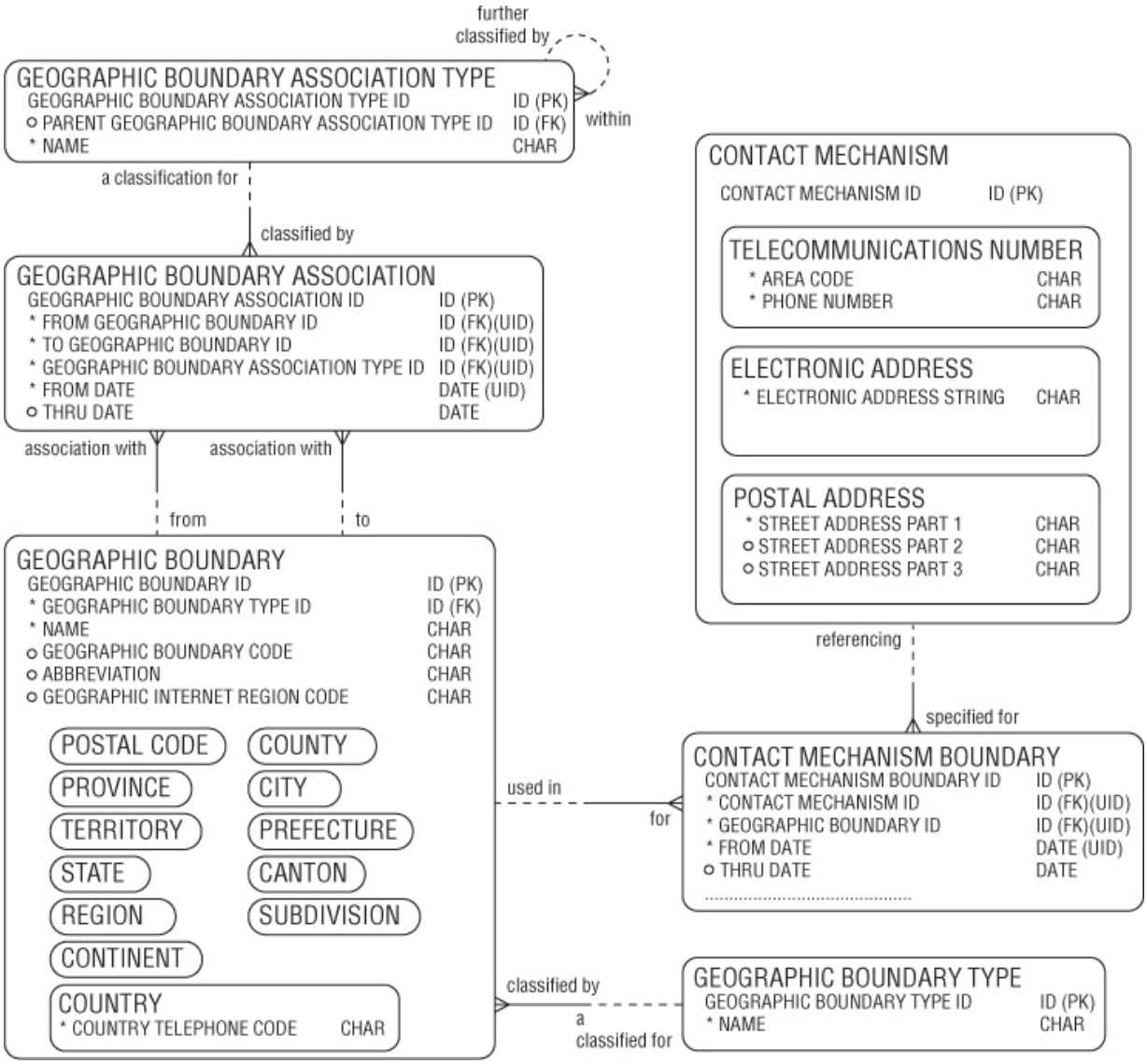
CONTACT MECHANISM. CONTACT MECHANISM ID	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)	CONTACT MECHANISM BOUNDARY. GEOGRAPHIC BOUNDARY ID (FK)	GEOGRAPHIC BOUNDARY. NAME (TYPE)	GEOGRAPHIC BOUNDARY. NAME (TYPE)	GEOGRAPHIC BOUNDARY ASSOCIATION TYPE. NAME	COUNTRY. COUNTRY TELEPHONE CODE	GEOGRAPHIC BOUNDARY. GEOGRAPHIC INTERNET REGION CODE
100	100 Main Street, Suite 819, The Coalman Building	20002	New York (city)	New York (state)	City State relationship		
			New York (state)	USA (country)	State Country relationship		
			United States of America (country)	North America (continent)	Country Continent relationship		
100	100 Main Street, Suite 819, The Coalman Building	20011	10019 (postal code)	New York (state)	Postal Code State relationship		
320	Andheri Kurla Road, Marol, no. 604	20220	Mumbai (City)	Maharashtra (State)	City State relationship		
			Maharashtra (State)	India (Country)	State Country relationship		
			400099 (Postal Code)	Maharashtra (State)	Postal Code State relationship		
4555	981 ponji	20288	Yaum-dong (Subdivision)	Ulsan-si (City)	Subdivision City relationship		
			Ulsan-si (City)	Kyongsangnamdo (Province)	City Province relationship		

CONTACT MECHANISM. CONTACT MECHANISM ID	CONTACT MECHANISM (TELECOMMUNICATIONS NUMBER/ELECTRONIC ADDRESS/POSTAL ADDRESS)	CONTACT MECHANISM BOUNDARY. GEOGRAPHIC BOUNDARY ID (FK)	GEOGRAPHIC BOUNDARY. NAME (TYPE)	GEOGRAPHIC BOUNDARY. NAME (TYPE)	GEOGRAPHIC BOUNDARY ASSOCIATION TYPE. NAME	COUNTRY. COUNTRY TELEPHONE CODE	GEOGRAPHIC BOUNDARY. GEOGRAPHIC INTERNET REGION CODE
			Kyongsangnamdo (Province)	Country (South Korea)	Province Country relationship		
			Country (South Korea)	Asia (Continent)	Country Continent relationship		
300	917 555 2100	40230	United States of America (country)	North America (continent)	Country Continent relationship	1	
600	11 2623 665	32003	India (country)	Asia (continent)	Country Continent relationship	91	
700	Mcollet@mtln.net.in	32003	India (country)	Asia (continent)	Country Continent relationship		.in
1200	noticas@lingsat.com.ar	32006	Argentina (country)	South America (Continent)	Country Continent relationship		.ar
1809	info@eservices.ca.gov	35000	California (state)	United States of America (Country)	State Country relationship		.ca

How Does This Pattern Work?

In Figure 7.10, the GEOGRAPHIC BOUNDARY entity maintains any type of encompassing area, such as (but not limited to) a county, city, state, postal code, province, canton, prefecture, subdivision, or territory. For instance, customers might have several addresses that have many different types of geographic boundary(s), depending on its location in the world. Notice that there are attributes of GEOGRAPHIC BOUNDARY for **name** and **abbreviation**, and this helps with consistency so that the name (such as “New York”) is not haphazardly interchanged with its abbreviation (such as “N.Y.”).

Figure 7.10 Contact Mechanism with Geographic Boundary Pattern



GEOGRAPHIC BOUNDARY contains many different subtypes, including COUNTRY, POSTAL CODE, CITY, COUNTY, PREFECTURE, REGION, CANTON, and so on. Each of these GEOGRAPHIC BOUNDARY(s) has an associated entry in the GEOGRAPHIC BOUNDARY TYPE entity. The GEOGRAPHIC BOUNDARY subtypes are related to each other via the GEOGRAPHIC BOUNDARY ASSOCIATION entity. For example, COUNTRY (“United States of America,” “Australia”) may contain STATE (“New York”, “New South Wales”), or COUNTRY (“Italy”) may contain REGION (Calabria). These associations can be typed, for example, the “New York” to

“United States of America” association could be considered a “State Country relationship” in the GEOGRAPHIC BOUNDARY ASSOCIATION TYPE entity. Another example would be how the “Calabira” to “Italy” association could be a “Region Country relationship.”

> **Note**
>
> The subtypes within GEOGRAPHIC BOUNDARY may or may not be subtypes in your data model, depending on your application. If there are no attributes or relationships that are specific to these subtypes, then we recommend not including them as subtypes and only as instances of GEOGRAPHIC BOUNDARY TYPE.

Each CONTACT MECHANISM *may be* **referencing** one or more CONTACT MECHANISM BOUNDARY(s), which are each **for** a GEOGRAPHIC BOUNDARY. Then each GEOGRAPHIC BOUNDARY *may be* **from** and **to** other GEOGRAPHIC BOUNDARY(s) through the GEOGRAPHIC BOUNDARY ASSOCIATION that is classified by a GEOGRAPHIC BOUNDARY ASSOCIATION TYPE. For example, Table 7.16 shows that the CONTACT MECHANISM (postal address subtype) of “100 Main Street, Suite 819, The Coalman Building” is related to a GEOGRAPHIC BOUNDARY of “New York” (a “city”) via the CONTACT MECHANISM BOUNDARY associative entity. Once we know the city, the GEOGRAPHIC BOUNDARY ASSOCIATION can relate the city to the state (“New York” city is related to “New York” state), the state to the country (“New York” state is related to “United States of America” country), and, if needed, the country to the continent (“United States of America” is related to “North America” continent). Because the postal code cannot be derived from the city (or vice versa), there is also another instance of CONTACT MECHANISM BOUNDARY that relates the address of “100 Main Street, Suite 819, The Coalman Building” to the postal code of “10019.”

An alternative way to maintain instances of a contact mechanism's association boundary is to relate all relevant GEOGRAPHIC BOUNDARY(s) to the CONTACT MECHANISM (via CONTACT MECHANISM BOUNDARY) instead of using the GEOGRAPHIC BOUNDARY ASSOCIATION. For instance, you could have four instances of CONTACT MECHANISM BOUNDARY that relate the specific city, state, country, and postal code to a postal address for United States addresses. This requires more instances to be maintained; however, it has the advantage of allowing more flexibility in the geographic boundaries that may be captured for a postal address. This may be required to handle postal address instances that do not neatly conform to what ‘should’ be. For example, we may need to capture an address that has “Sydney” as a city in the “United States,” and even though this is not correct, this is the data that we have for this postal address!

Note

We do not view postal addresses as being the same as geographic locations. A postal address is a way to send mail via some postal service, and it represents a ‘label’ that may correspond to a geographic location. A geographic location represents geographic coordinates for a boundary, point, or pathway. An alternative model that is discussed in *The Data Model Resource Book*, Volume 2, Revised Edition (Wiley, 2001), in Chapter 3, the “Telecommunications” chapter, is to have the supertype of GEOGRAPHIC LOCATION with subtypes of GEOGRAPHIC BOUNDARY (for a geographic area), GEOGRAPHIC POINT (for a specific GPS coordinate with specific latitude and longitude in degrees, minutes, and seconds), and PATHWAY (for the route between two points). Thus, depending on your needs, GEOGRAPHIC BOUNDARY may be a subtype of GEOGRAPHIC LOCATION; however, in terms of contact mechanisms, GEOGRAPHIC BOUNDARY is the only one of these subtypes that is needed and that is why we are not showing the GEOGRAPHIC LOCATION supertype in this section.

Geographic boundaries may apply to more than just postal addresses. For example, phone numbers may be applicable for certain COUNTRY(s). In Table 7.16, you see that a telephone number of “917 555 2100” is related to a GEOGRAPHIC BOUNDARY of “United States of America.” According to Figure 7.10, the GEOGRAPHIC BOUNDARY subtype of COUNTRY has a **country telephone code** attribute. The CONTACT MECHANISM telephone number of “917 555 2100” (notice the TELECOMMUNICATIONS NUMBER in Table 7.16 no longer has a “1” prefix for the country telephone code) is within a CONTACT MECHANISM BOUNDARY that is in the GEOGRAPHIC BOUNDARY **name** of “United States of America,” which has a **country telephone code** of “1” (which is the code used to access this country) in the COUNTRY entity. Thus, in this pattern, we removed the TELECOMMUNICATIONS NUMBER attribute of **country telephone code** that we have shown in previous figures, because it is an attribute of the associated country for the contact mechanism. Similarly we have related the COUNTRY “India” with **country telephone code** “91” and associated it (via the CONTACT MECHANISM BOUNDARY) to the telephone number of “11 2623 665” (this was “91 11 2623 665” in previous tables in this chapter because it included the country telephone code).

Note

In most cases, it is enough to relate a CONTACT MECHANISM of type “Telecommunications number” to a GEOGRAPHIC BOUNDARY of type “Country” (through CONTACT MECHANISM BOUNDARY) in order to get the country telephone code for the number. In some cases, the enterprise may also want to relate the CONTACT MECHANISM of type “Telecommunications number” to a different type of GEOGRAPHIC BOUNDARY, such as a “City” or “State” in order to find out which telecommunication numbers are within those boundaries, for example, the number “303 211 1111” is in the state of “Colorado.” We can accomplish this with this pattern. However, there would be some redundancy because the area code is an attribute of

TELECOMMUNICATIONS NUMBER and thus we are redundantly maintaining many instances of the same area code that is related to the same geographic boundary. For example, many instances of an area code of “303” (for each TELECOMMUNICATION NUMBER instance) would be redundantly related to a state of “Colorado” (the same GEOGRAPHIC BOUNDARY instance). A solution to this is to remove the **area code** attribute from CONTACT MECHANISM and just maintain an attribute in GEOGRAPHIC BOUNDARY for an “area code” (or you could have even a more generic attribute of “telephone code”). However, from a practical perspective, most enterprises probably would not want to look up area codes for a telecommunications number in the GEOGRAPHIC BOUNDARY entity.

Electronic addresses may also have different geographic boundaries. For example, “info@XYZ.com” may be an electronic address for XYZ Corporation, but so could “info@XYZ.co.uk,” “info@XYZ.co.hk,” and “info@XYZ.ie.” The “.co.uk” designates the United Kingdom, “.co.hk” designates the Hong Kong region, and “.ie” designates Ireland. This is illustrated in Table 7.16 as CONTACT MECHANISM “Mcollet@mtln.net.in” is related to the GEOGRAPHIC BOUNDARY for “India,” which has an attribute of **geographic internet region code** of “.in.” You may conclude that we are maintaining redundant data because we are maintaining the “.in” in the **electronic address string** of ELECTRONIC ADDRESS as well as in the **geographic internet region code** of GEOGRAPHIC BOUNDARY. However, to leave out the “.in” as part of the **electronic address string** causes issues as well. Notice the last instance of the email address CONTACT MECHANISM “info@eservices.ca.gov” that we are relating to the state GEOGRAPHIC BOUNDARY. If we remove the “.ca” from the **electronic address string** of “info@eservices.ca.gov,” thus making it “info@eservices.gov,” it would be difficult to ascertain the content of the email address and where to insert the “.ca.”

Note

It is interesting to note that there is a different type of relationship to geographic boundary for electronic addresses than there is for postal addresses or telecommunications numbers in that the email address is not really “within” this geographic boundary because it is virtual. Thus, unlike the relationship name in GEOGRAPHIC BOUNDARY from Figures 7.7 and 7.9, we have changed the relationship name from GEOGRAPHIC BOUNDARY to **referencing** in order to cover all types of contact mechanisms.

Note

The designations (for example, “.uk,” “.hk”) may have less significance going forward after the freeing up of restrictions on domain names, and thus there is great advantage to flexibly allowing contact mechanisms to be related to geographic boundaries that may have any type of **geographic internet region code**.

Note

We have shown a fairly generic version of this geographic boundary pattern. Each of the different subtypes of GEOGRAPHIC BOUNDARY may have specific relationships between them. For example, a COUNTRY may be composed of STATE(s). You may explicitly model these relationships. For an example of this, see Figure 2-8 in of *The Data Model Resource Book, Revised Edition, Volume 1* (Wiley, 2001). Also, you can use the Level 3 Recursive Pattern with Rules Pattern (see Chapter 4 of this book) to manage the relationships in the GEOGRAPHIC BOUNDARY TYPE and GEOGRAPHIC BOUNDARY ASSOCIATION entities.

A benefit of relating one GEOGRAPHIC BOUNDARY to another GEOGRAPHIC BOUNDARY via the GEOGRAPHIC BOUNDARY ASSOCIATION is that all of the different postal hierarchies could be pre-created here and used to reduce the number of invalid addresses. Addresses could be

checked off against the data stored in the GEOGRAPHIC BOUNDARY and GEOGRAPHIC BOUNDARY ASSOCIATION entities to make sure the geographic boundaries are valid for an address, for example, that a specific postal code that is entered actually makes sense for the city that is associated to it. The GEOGRAPHIC BOUNDARY ASSOCIATION entity could maintain the specific valid relationships that exist between different instances of GEOGRAPHIC BOUNDARY. For example, GEOGRAPHIC BOUNDARY ASSOCIATION could contain the association that shows that "10019" is a valid postal code for the city of "New York."

Note

In addition to maintaining how specific geographic boundaries are related to each other, a lot of repeated energy goes into deciding what *type* of geographic boundaries should be related to other *types* of geographic boundaries, for a particular jurisdiction or for a particular part of the world. For example, does the United Kingdom have counties or does it have regions, and if so, how are regions related to counties? Does Australia have states or territories or both? How are subdivisions, cities, provinces, and regions related to each other in Korea? This can be resolved by applying the Level 3 Recursive Pattern with Rules to GEOGRAPHY BOUNDARY TYPE, and by adding a relationship to GEOGRAPHIC BOUNDARY (e.g., "Australia" in the context our example above). This relationship will give a context of the association between GEOGRAPHIC BOUNDARY TYPE(s). See Chapter 4 for more discussion on modeling these types of 'rule' entities in relation to hierarchies, peer-to-peer associations, and aggregations.

Note

Aside from what is in Table 7.16, there could be other GEOGRAPHIC BOUNDARY ASSOCIATION TYPE(s), such as "Overlapping area," "Adjacent area," and/or "Contained within." There could also be a parent of the association types shown in Table 7.16 that would use the recursive relationship around GEOGRAPHIC BOUNDARY ASSOCIATION TYPE to show that sub-classifications, such as "City State relationship" or "State

Country relationship," are within the parent association type of "Contained within." Another example is that the countries Portugal and Spain could have a GEOGRAPHIC BOUNDARY ASSOCIATION TYPE of "Adjacent area".

When Should This Pattern Be Used?

This pattern is useful:

- **When a flexible solution is needed to store contact mechanism and jurisdiction information:** This pattern provides a structure to support any type of GEOGRAPHIC BOUNDARY and any type of associations between GEOGRAPHIC BOUNDARY(s). Each CONTACT MECHANISM may be related to many instances and types of GEOGRAPHIC BOUNDARY(s).
- **When there is a strong need to support domestic and international addresses with all different types of jurisdictions:** This pattern accommodates a wide variety of domestic and international addresses because various types of geographic boundaries may be used (for example, province, territory, state, region, and so on).
- **When there is a need to maintain the valid relationships between GEOGRAPHIC BOUNDARY(s) in order to help improve data quality of contact mechanisms:** With this pattern, geographic boundaries can be associated together in advance of creating the postal address, by using GEOGRAPHIC BOUNDARY ASSOCIATION. For example, this can be used to maintain that a specific CITY has specific valid POSTAL CODE(s). In fact, the interrelationships between each different geographic boundary can create complex hierarchies of different geographic boundaries. These complex hierarchies may be captured as part of the GEOGRAPHIC BOUNDARY ASSOCIATION and GEOGRAPHIC BOUNDARY TYPE ASSOCIATION, and additional rules may be

captured by using the Level 3 Recursive Pattern with Rules or a GEOGRAPHIC BOUNDARY TYPE RULE entity with one of the business rules patterns described in Chapter 8 of this book.

- **When there is a need to classify not only postal addresses but all types of contact mechanisms by geographic boundary:** For example, there may be a need to find out which emails you receive from India or how many telephone calls you are making to the United Kingdom.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This pattern is one of the more complex patterns in terms of content:** This pattern uses highly generalized structures and thus is more difficult to understand. Also, there are many different ways of maintaining the associated geographic boundaries for contact mechanisms with this pattern. For example, you can maintain the related city for the postal address and then find out the state and country by using the GEOGRAPHIC BOUNDARY ASSOCIATION, or you can relate the city, state, and country as three instances of CONTACT MECHANISM BOUNDARY.
- **Because it is a generalized approach, it is easier for data modelers to become 'lazy.'** It is easy for a data modeler to add new instances of geographic boundaries, geographic boundary associations, or geographic boundary types and not model and/or analyze the specific data requirements.
- **This pattern is more generic and, therefore, does not specify certain business rules as part of the data model:** For example, a telephone number may

have a business rule that there is only one geographic boundary, namely a country, related to it; however, the pattern allows for a many-to-many relationship between any contact mechanism and any geographic boundary (thus, you could actually relate two countries to a telephone number with this pattern).

Synopsis

In this section we examined how most enterprises and the people who work within them function within some jurisdiction. These jurisdictions, or as we call them, geographic boundaries, are crucial for managing the enterprise's contact mechanisms and for many reporting and management functions. A flexible solution is needed to store jurisdiction information. This pattern provides a structure to allow CONTACT MECHANISM(s) to maintain any number and any type of GEOGRAPHIC BOUNDARY(s) needed in an enterprise and to maintain how GEOGRAPHIC BOUNDARY(s) are related to each other via the GEOGRAPHIC BOUNDARY ASSOCIATION entity.

Geographic boundaries are fundamental elements in contact mechanisms, especially postal addresses and to lesser extent telephone numbers and electronic addresses. It is crucial that a pattern exists to support the needs of all of these types of boundaries in a flexible manner. There are great disadvantages in hard coding geographic boundaries like **city**, **state**, **postal code**, and **country** as attributes of a CONTACT MECHANISM like POSTAL ADDRESS, or even in creating separate entities for each of these. The GEOGRAPHIC BOUNDARY entity maintains any type of encompassing area, such as, but not limited to, a COUNTY, CITY, STATE, POSTAL CODE, PROVINCE, CANTON, PREFECTURE, or TERRITORY.

Each CONTACT MECHANISM may have several GEOGRAPHIC BOUNDARY(s) via the associative entity

CONTACT MECHANISM BOUNDARY. For example, a POSTAL ADDRESS may need a GEOGRAPHIC BOUNDARY of both a postal code and a city. Geographic boundaries may apply to more than just postal addresses. Phone numbers have a **country telephone code** that is an attribute of a COUNTRY subtype of GEOGRAPHIC BOUNDARY. Electronic addresses may also have different geographic boundaries. For example "info@XYZ.com" may be an electronic address for XYZ Corporation, but so could "info@XYZ.co.uk," "info@XYZ.co.hk," and "info@XYZ.ie." The ".co.uk" designates the United Kingdom, ".co.hk" designates the Hong Kong region, and ".ie" designates Ireland. Thus, each ELECTRONIC ADDRESS subtype of CONTACT MECHANISM may be related to a GEOGRAPHIC BOUNDARY attribute of **geographic internet region code** in order to show that an email is for a country, such as "India," which has an ".in" **geographic internet region code.**

This pattern is a very flexible data model solution that accommodates international addresses effectively, provides a way to improve data quality by maintaining valid geographic boundary relationships in advance, and allows all types of contact mechanisms to be classified by their associated geographic boundaries. However, it is complex, more difficult to understand, and hides many business rules. Many of these complex rules and associations may be accommodated by adding to this pattern the Level 3 Recursive Pattern with Rules Pattern or by maintaining rules another way, such as by using a business rules pattern or having a business rules layer.

Contact Mechanism with Flexible Address Parts Pattern

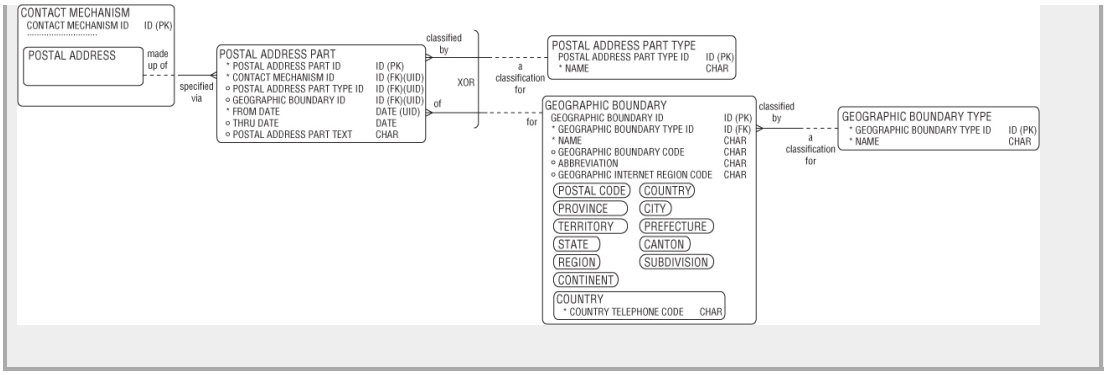
We have often come across the common problem that various postal services throughout the world have very different formats for maintaining addresses. There are two different problems with postal addresses, of which we address only one (no pun intended).

- Different postal services have different standards about the presentation of the address on a piece of mail. We *don't* completely address this problem, but we do help in making the solution to this problem somewhat easier.
- Those different postal services have different 'parts' that make up the address. How do we create a flexible structure to suit all postal services with their different parts? This is the main issue that we address.

Note

It is worth noting that address presentation is beyond the scope of this section. For a definitive look at this idea, see http://www.columbia.edu/kermit/postal.html, one of the most useful and comprehensive address resources out there. It is also worth noting that the standards for address presentation differ widely, and there is no real standard. The ISO International Standard 11180 "Postal Addressing" was withdrawn on the 15th of January 2004 because of difficulties with it. This section deconstructs the address parts into its subcomponents so that they can be stored. How they are presented is part of the standards of the sending and receiving countries. Nonetheless, it is possible to create a data model structure to help maintain the formatting structure of address parts by creating a number of additional entities, attributes, and relationships. For example, one could create a POSTAL ADDRESS FORMATTING RULE entity that is related to many POSTAL ADDRESS PART TYPE(s) (to define how the parts are presented) and COUNTRY (to define the specific country the formatting is for) entities (see Figure 7.11 for an example and see Chapter 8 for more discussion on creating this type of business rules entity).

Figure 7.11 Contact Mechanism with Flexible Address Parts Pattern



The first problem can be characterized by the example in Table 7.17. The order of presentation is ‘minor to major,’ that is, from the person's name to the country. This is the same in the following Japanese example (Table 7.18), but the city part has been replaced with “ku,” “shi,” and “ken.” The part that gives trouble is the city part. This is where most presentation problems occur because the address parts or elements are often different at this level from jurisdiction to jurisdiction. We are not so concerned with this presentation problem, that is, in what sequence things go on the page. What concerns us is being able to capture all of the different constituent parts of the address. How those parts are fitted together for presentation is dependent on the different postal governing bodies in each jurisdiction, and as we mentioned, could be handled with additional rules oriented data model structures.

Note

The person's name, company name, and department are not part of the contact mechanisms as we see it. They are related to the contact mechanisms (as PARTY(s)) and could be related via the PARTY CONTACT MECHANISM entity shown in Figure 7.7. For example, Mr. Taro Tanaka of the Company Fujitsu Limited in the department Optical Network Systems Development may not be the only person (company or department) residing at 4-1-1 Kamikodanaka in Nakahara–ku, the Street, Ward, City, Prefecture, Postal Code, and Country that comprise the contact mechanism! The person's name, company name, and department can be maintained with the PARTY, PERSON,

ORGANIZATION, and PARTY ROLE entities, and they can be related to each other via PARTY RELATIONSHIPS(s).[9]

Table 7.17 Addresses Presentation Structure Example, Australia

Joe Bloggs	Person's name
Computer Center	Department (if any)
Curtin University of Technology	Institution or Company (if any)
309 Kent Street	Street Address (or Post Office Box)
Bentley, WA 6102	City Part (city, territory [WA = Western Australia], and postal code)
Australia	Country Name

Table 7.18 Addresses Presentation Structure Example, Japan

Mr. Taro Tanaka	Person's name
Fujitsu Limited	Company name
Optical Network Systems Development	Department
4-1-1 Kamikodanaka	Street
Nakahara-ku	“ku” = Ward
Kawasaki-shi	“shi” = City
Kanagawa-ken	“ken” = Prefecture
211-8588	Postal code
Japan	Country

Why Do We Need This Pattern?

Different postal services have different ‘parts’ that make up the address. Although Tables 7.17 and 7.18 show the address presentation format, this pattern will focus on maintaining the constituent parts of an address. This pattern creates a very flexible structure to suit all postal services that may have very different address parts. Based on the preceding example in Table 7.18, you see that for the country “Japan,” you need to maintain a POSTAL CODE and PREFECTURE. PREFECTURE contains CITY, and CITY contains

WARD. These geographic boundaries and their associations to each other are already handled by the Contact Mechanism with Geographic Boundary Pattern, as seen in Figure 7.10, and in that pattern, the street address part is maintained in the CONTACT MECHANISM. However, to allow more flexibility, you can maintain the various parts of the POSTAL ADDRESS so each of the different parts of Table 7.18 (Street, “ku” = Ward, “shi” = City, “ken” = Prefecture, Postal code, and Country) and Table 7.17 (Street or Post Office Box, City, State, Postal Code, and Country) are handled in exactly the same manner. This means that an enterprise can maintain the parts for any address it wishes for anywhere in the world.

How Does This Pattern Work?

The nuts and bolts of this pattern lie in the associative entity of POSTAL ADDRESS PART entity (see Figure 7.11). In this pattern, CONTACT MECHANISM has a subtype of POSTAL ADDRESS and “each POSTAL ADDRESS *may be* **made up of** one or more POSTAL ADDRESS PARTS(s).”

A POSTAL ADDRESS is a composition of all of the various POSTAL ADDRESS PART(s). It would be possible to have a derived attribute in this entity called **postal address string** that could contain all of the text of the POSTAL ADDRESS PART(s) concatenated together to form a single address string. Some enterprises find this useful; some don't, so we have left it out, subscribing to the general practice of not including derived data in models, especially in our more generic patterns.

A POSTAL ADDRESS PART is a single atomic piece of postal address information. For example “APT 5A, The Foundry” has two different distinct POSTAL ADDRESS PART(s). “APT 5A” is the apartment number of an abode. This POSTAL ADDRESS PART has a POSTAL ADDRESS PART TYPE of “Apartment.” “The Foundry” is the building name for the

address and has a POSTAL ADDRESS PART TYPE of “Building Name.”

> **Note**
>
> POSTAL ADDRESS PART TYPE is important to some enterprises. For example, catalog companies and marketing agencies - if you have an address part with a type of “Apartment,” they probably should not send a catalog of lawn-care equipment, but might want to send you a catalog of compact washer/dryers.

POSTAL ADDRESS PART is the confluence of different foreign keys:

- First, the foreign key from POSTAL ADDRESS to POSTAL ADDRESS PART is **contact mechanism id.**
- Second, it maintains either the optional foreign key from POSTAL ADDRESS PART TYPE, **postal address part type id**, or the optional foreign key, **geographic boundary id**, from GEOGRAPHIC BOUNDARY. There is an exclusive or (XOR) relationship over the two foreign keys from POSTAL ADDRESS PART TYPE and GEOGRAPHIC BOUNDARY. This means that an instance of a POSTAL ADDRESS PART must have either a POSTAL ADDRESS PART TYPE (like “Apartment” or “Street address”) to identify the type of **postal address part text** it is or else have a foreign key to a GEOGRAPHIC BOUNDARY like “Kawasaki-shi” with a GEOGRAPHIC BOUNDARY TYPE of “CITY.”

> **Note**
>
> You could expand this pattern to also record the rules about the allowable ways that POSTAL ADDRESS PART(s) and GEOGRAPHIC BOUNDARY(s) are related to each other within a COUNTRY. This would provide a data model that could be used to validate that addresses have a valid structure. For instance, in the United States of America, the postal address always uses cities that are within states that are within countries. You could use the rule patterns that are discussed in Chapter 8 to add a data model

structure (with a POSTAL ADDRESS RULE entity that has relationships to GEOGRAPHIC BOUNDARY TYPE(s) and POSTAL ADDRESS PART TYPE(s)) to accommodate this or use the Level 3 Recursive Pattern with Rules described in Chapter 4.

To illustrate this pattern take a look at Table 7.19. If you look at the postal address with the **contact mechanism id** of “901” (the first seven rows of the illustration table), you can see that it is made of many different POSTAL ADDRESS PART(s): “91001” (“100 Main Street”), “91002” (“Suite 819”), “91003” (“The Coalman Building”), and a number of other parts that are related to various geographic boundaries. In Table 7.19, you see that the postal address part of “91001” is “100 Main Street” of type “Street address” meaning it is a number (and/or text, such as 100A) and street (or something similar, such as a boulevard, road, highway, and so on). With this pattern, it would be possible to maintain this as two separate POSTAL ADDRESS PART(s) of “Street number” (“100”) and “Street name” (Main Street) if desired. However, this may be overkill because countries all over the world have some type of “Street address.” In the case of POSTAL ADDRESS PART **postal address part id** “91002” has a value of “Suite 819” (the **postal address part text**) of POSTAL ADDRESS PART TYPE “Suite.” The table illustrates that if the part of the address is a POSTAL ADDRESS PART, it can't be a GEOGRAPHIC BOUNDARY because of the ‘exclusive or’ (XOR) in Figure 7.11. The POSTAL ADDRESS PART “91003” is “The Coalman Building” with a POSTAL ADDRESS PART TYPE of “Building.”

Table 7.19 CONTACT MECHANISM, Postal Address and Postal Address Parts

POSTAL ADDRESS PART. CONTACT MECHANISM ID	POSTAL ADDRESS PART. POSTAL ADDRESS PART ID	POSTAL ADDRESS PART. POSTAL ADDRESS PART TEXT	POSTAL ADDRESS PART TYPE. NAME	GEOGRAPHIC BOUNDARY. NAME (ABBREVIATION)	GEOGRAPHIC BOUNDARY TYPE. NAME
901	91001	100 Main Street	Street Address		
901	91002	Suite 819	Suite		
901	91003	The Coalman Building	Building		
901	20032			New York	City
901	20042			New York (NY)	State
901	20052			10019	Postal Code
901	34001			United States of America	Country
1001	10101	4-1-1 Kamikodanaka	Street Address		
1001	10203			Nakahara-ku	Ward
1001	10305			Kawasaki-shi	City
1001	10407			Kanagawa-ken	Prefecture
1001	10609			211-8588	Postal Code
1001	32003			Japan	Country

The first three parts of this address are POSTAL ADDRESS PART(s), and the next four parts are GEOGRAPHIC BOUNDARY(s). For the geographic boundaries, the first part is the GEOGRAPHIC BOUNDARY “New York” of GEOGRAPHIC BOUNDARY TYPE “City.” Next is “New York (NY),” which is of GEOGRAPHIC BOUNDARY TYPE “State.” Notice we can use either the GEOGRAPHIC BOUNDARY **name**, or **abbreviation** to create the postal address, depending on the presentation rules of a countries postal service. Then the next part is “10019” for the GEOGRAPHIC BOUNDARY TYPE “Postal code,” and finally “United States of America” with GEOGRAPHIC BOUNDARY TYPE “Country.” This creates a complete POSTAL ADDRESS as illustrated in Table 7.20.

Note

Instead of relating the POSTAL ADDRESS PART(s) to all of its GEOGRAPHIC BOUNDARY(s), an alternative way of maintaining the instances of POSTAL ADDRESS PART(s) is to maintain instances only for the lower-level GEOGRAPHIC BOUNDARY instance and then use the GEOGRAPHIC BOUNDARY ASSOCIATION to derive the higher-level instances of GEOGRAPHIC BOUNDARY, as we did in the “Contact Mechanism with Geographic Boundaries Pattern” section earlier in this chapter. Thus, for the address “100 Main Street, Suite 819, The Coalman Building,” you may choose to maintain relationships only to the GEOGRAPHIC BOUNDARY(s) of “City” and “Postal Code” and then derive the relationships to “State” and “Country” using the GEOGRAPHIC BOUNDARY ASSOCIATION data model structure. Which approach to choose depends on the specific circumstances and whether the enterprise decides to maintain relationships between geographic boundaries in order to validate addresses or decides to just maintain the geographic boundary relationships that appear for the address.

Table 7.20 POSTAL ADDRESS PART(s), Creating a Complete Postal Address from Table 7.19

POSTAL ADDRESS PART	POSTAL ADDRESS PART TYPE.NAME OR GEOGRAPHIC BOUNDARY.NAME
100 Main Street	Street Address

Suite 819	Suite
The Coalman Building	Building
New York	City
NY	State
10019	Postal Code
United States of America	Country

When Should This Pattern Be Used?

We use this pattern for:

- **Enterprises that need flexibility when dealing with postal addresses:** Many enterprises, such as magazine subscription firms and enterprises that do mail shots or direct marketing, would find this pattern to be very useful and flexible. Any postal enterprise could use this pattern. In fact it is possible that any and every enterprise that has postal addresses or uses postal addresses should use this pattern, especially enterprises that need to maintain flexible, international address structures.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are:

- **It may be overkill for many enterprises that need a much for more simple address structure:** This pattern would be an added complication to them. This pattern has a level of complexity and flexibility that can be confusing, and this can lead to poor implementations that in turn lead to data quality issues. This pattern needs to be explained in detail before use.
- **The pattern does not accommodate maintaining various presentation formats for different**

countries: Although the pattern accommodates a very flexible way to maintain address parts, many additional entities, attributes, and relationships are needed to expand this pattern to also maintain valid postal address presentation formats.

Synopsis

In this section we examined a major issue with postal addresses across the world, that is, the problem that different jurisdictions require a great variety of different parts in the address. What concerns us is being able to capture all of the different constituent parts of the address. This pattern creates a flexible structure to suit various countries regardless of the different address parts they use.

The nuts and bolts of this pattern lie in the POSTAL ADDRESS PART entity. A POSTAL ADDRESS PART is a single atomic piece of postal address information. POSTAL ADDRESS PART is the confluence of four pieces of information: the related POSTAL ADDRESS, the POSTAL ADDRESS PART TYPE, the GEOGRAPHIC BOUNDARY, and the actual text of the postal part (for example, “Apt. 5E”). A POSTAL ADDRESS PART may be related to either a GEOGRAPHIC BOUNDARY or POSTAL ADDRESS PART TYPE, but never both at the same time.

This pattern may be useful to any enterprise that needs a flexible approach to managing postal addresses and their parts. It might be considered ‘overkill’ for an enterprise that has a simple or stable address structure.

Other Common Contact Mechanism Data

This section deals with some other common data items that many enterprises may need to capture as part of the

contact mechanism patterns. These common data needs were not added to the patterns because we did not consider them to be integral to the structure of the patterns. In other words, including this type of data in the patterns does not significantly change the basic structure of each pattern, and because many of these patterns cover a lot of data modeling structures already, we chose to just describe this other type of data so that we do not illustrate too much in the diagrams. Nevertheless, these additional types of data enhance the knowledge about contact mechanisms and are often considered to be very important for maintaining contact mechanism information. The four different types of additional contact mechanism data described here are as follows:

- **Non-solicitation**—This is data that specifies if a contact mechanism could (or should not) be used to solicit business.
- **Instructions**—This data supports any knowledge or information imparted about how to use the contact mechanism, for example, operating hours, preferred times to contact, and so on.
- **Directions**—This maintains any guidance on how to get to a particular contact mechanism of type POSTAL ADDRESS.
- **Telephone extensions**—This is a particular number, such as “Ext. 112,” that refers to a specific telephone set and that is added onto an organization's telephone number in order to contact a specific person's phone within a switchboard.

Non-Solicitation

Certain contact mechanisms may be tagged as contact information that is not to be solicited (or in other words not to be used for sales reasons). Because there are many laws that regulate that a party has the right not to be solicited

when asked, it is important data that enterprises need to ensure that certain contact mechanisms are not used to solicit business. For example, many people get junk mail credit card applications delivered to their home addresses. If you choose to stop receiving prescreened offers of credit, you can call a toll-free number and your contact mechanism (POSTAL ADDRESS) will be marked for 'non-solicitation.'

One way to model this is to add an attribute of **non-solicitation indicator**. If the non-solicitation indicator is recorded as "Y(es)," this would indicate not to use this contact mechanism for solicitation purposes. When people register at a web site by putting in details about their contact mechanisms (POSTAL ADDRESS, TELECOMMUNICATIONS NUMBER, and ELECTRONIC ADDRESS) there will more than likely be a check box at the bottom of the screen indicating if they wish to receive information (normally set to "Y(es)"). The federal government in the United States (and governments in other countries) have enacted legislation to stop people from 'cold calling' telephone numbers. Companies who do this can suffer stiff penalties for not keeping their solicitation data up to date and for subsequently cold calling someone who expressly requested that they do not receive these cold calls.

In the level 1 pattern you can accommodate this need by adding an attribute for each different contact mechanism, that is, **email non-solicitation indicator**, **telephone non-solicitation indicator**, and **bill to postal address non-solicitation indicator**. This is interesting in two ways: first, you need a solicitation indicator for each type of contact mechanism, and second, you need an indicator for each different type of contact mechanism with a contact mechanism purpose.

Where does this attribute go in the level 2, 3, and 4 patterns? This depends on the circumstances. Consider a

situation where PARTY(es) called “Manu Collet” and “Rupali Anjaria” share the same contact mechanism of “91 11 2623 665.” The **non-solicitation indicator** should be “N(o)” for “Manu Collet” and “Y(es)” for “Rupali Anjaria” for this telephone number. This means that in this case, the **non-solicitation indicator** needs to be in an associative entity that is related to it. Therefore, the **non-solicitation indicator** would be in the PARTY TELECOMMUNICATIONS NUMBER for the level 2 pattern, the PARTY CONTACT MECHANISM for the level 3 pattern, and in the CONTACT MECHANISM APPLICATION in the level 4 pattern. Although less common, there may be other circumstances where the contact mechanism needs to have a **non-solicitation indicator**, regardless of the parties involved or its context, and in this case, the non-solicitation indicator would be maintained within the CONTACT MECHANISM entity.

Furthermore, if the non-solicitation is based upon many factors, such as the purpose type, usage type, contact mechanism type, or other factors, you may need a more robust solution where non-solicitation is maintained with additional entities to show the rules behind the non-solicitation. Thus, there may be a NON-SOLICITATION RULE entity that is related to the various contact mechanism classification entities (don't solicit when the “Usage type” is “Personal”) as well as a FACTOR TYPE entity (maintaining the factors involved in solicitation; for example, don't solicit on weekends). Please refer to Chapter 8 for more information on the rules pattern. However, more often than not we have found that putting the non-solicitation data as a non-solicitation indicator in the associative entities works for most enterprises.

Instructions

Instructions capture any knowledge the enterprise wishes to impart about how to use a contact mechanism, such as “Use

this address for deliveries only between 9 A.M. and 5 P.M. seven days a week." It could be modeled as a free-form text attribute that imparts some concepts or knowledge. A similar problem that exists for the **non-solicitation indicator** exists for **instructions**. Some **instructions** may be attributes of a contact mechanism or they may be attributes of an associative entity because the instructions may be in the context of a particular party, order, purpose, usage, type, or other factor.

An example of when **instructions** are directly related to CONTACT MECHANISM could be "This address is accessible for business deliveries between 9:00 A.M. and 5:00 A.M. weekdays", and this could be related to the address "Andheri Kurla Road, Marol no. 604, Mumbai, Andheri East, India, 400099," regardless of context. Or perhaps the **instructions** may be related to an associative entity, such as PARTY CONTACT MECHANISM, for example, "Please leave all packages at the front desk" could be tied to "Andheri Kurla Road, Marol no. 604, Mumbai, Andheri East, India, 400099" when it is used for the party "Manu Collet." Or finally, for a specific order using a contact mechanism, "We need a signed order confirmation sent to this fax number" could be attached to the association between the ORDER and the CONTACT MECHANISM, ORDER CONTACT MECHANISM (or CONTACT MECHANISM APPLICATION in the level 4 pattern). In fact all these situations could be true and CONTACT MECHANISM, PARTY CONTACT MECHANISM (or CONTACT MECHANISM APPLICATION), and ORDER CONTACT MECHANISM could all have an **instructions** attribute, or a relationship to an INSTRUCTION entity, depending on what is needed.

For a more comprehensive data modeling solution, **instructions** may be modeled as various additional entities. For example, the preferred times to contact may be modeled via an intersection entity, PREFERRED CONTACT

TIME, which links the PARTY CONTACT MECHANISM with another entity of STANDARD TIME PERIOD, showing specific preferred time periods to call, such as weekdays from 9 A.M. to 5 P.M. Additionally, similar to non-solicitation data, there could be rules entities to maintain the instructions for different circumstances.

Directions

Directions are how to get somewhere. They are related to POSTAL ADDRESS; you don't need directions to get to a telephone number or an email address. (There may be a valid exception for this, and maybe we will get an email from someone with the valid exception!) A very simple solution is to have a **directions** attribute for the POSTAL ADDRESS. However, the problem with directions is the classic ‘from where to here’ problem. If anyone has been to a wedding, they will probably have gotten the directions in the form of “by train from London, by train from Scotland, by road from London, coming from the north, coming from the south,” and so on. For directions to be handled in a truly rigorous fashion a DIRECTIONS entity could be created, with DIRECTIONS TYPE (“By rail,” “By road,” “By air,” “Coming from the South,” “Coming from the East,” and so on). The DIRECTIONS entity would have a one-to-many relationship to POSTAL ADDRESS and a many-to-one relationship to GEOGRAPHIC BOUNDARY for the ‘from where’ part of the question. It is usually sufficient for many if not most enterprises to have a free-form text attribute in POSTAL ADDRESS.

Telephone Extensions

Another piece of data regarding contact mechanisms is the telephone extension of a person. For example, a person may have a telephone number of “1 917 555 2100,” which is the

telephone number for their company, and their telephone extension may be “256,” which can be accessed via a company switchboard to get to their phone. You may think the telephone extension is just an attribute of TELECOMMUNICATIONS NUMBER. However, if the same telecommunications numbers may be used for many parties, which is often the case when telephone extensions are involved, we recommend modeling it as an attribute of an associative entity between a party and the contact mechanism (except for the level 1 pattern, which does not have an associative entity). This allows the general number to be shared (for example, many people may have the number “1 917 555 2100” for the company) and the **telephone extension** to be specific to that party and contact mechanism.

In the level 1 pattern, the **telephone extension** can be maintained as an attribute of a PARTY. In the level 2 pattern it can be maintained as an attribute of the PARTY TELECOMMUNCATIONS NUMBER. In the level 3 pattern it can be maintained as an attribute of the PARTY CONTACT MECHANISM and in the level 4 pattern, it can be maintained as an attribute of the CONTACT MECHANISM APPLICATION. There is one wrinkle in that if multiple people share an extension, there could be some redundancy in having to maintain that same extension for each person. Therefore, this can be maintained as another attribute of the telecommunications number or another solution could be to have a TELEPHONE EXTENSION entity tied to the association entity for a party and a contact mechanism; however, this is usually overkill for most enterprises.

Synopsis

This section covered possible ways to model four additional aspects of contact mechanism data: non-solicitation, instructions, directions, and telephone extensions.

Non-solicitation data may be maintained as attribute(s) for the PARTY entity in the level 1 pattern and as an attribute in the contact mechanism associative entities in the Level 2, 3, and 4 Patterns (such as PARTY TELECOMMUNICATIONS NUMBER, PARTY CONTACT MECHANISM, or CONTACT MECHANISM APPLICATION, respectively). If a more comprehensive solution is needed, numerous additional entities may be needed.

Instructions may also be maintained as an attribute of the contact mechanism, associative entities, or if a more robust solution is needed, then it may be modeled using additional entities, such as PREFERRED CONTACT TIME related to a STANDARD TIME PERIOD, or the business rules pattern from Chapter 8.

Directions for how to travel to a postal address may be maintained as an attribute of POSTAL ADDRESS, unless there is a real need for a robust directions structure whereby an entity of DIRECTIONS (that is of a DIRECTION TYPE) could be related to POSTAL ADDRESS.

Telephone extensions may also be related to either the PARTY entity in the Level 1 Pattern or to the contact mechanism associative entities for the Level 2, 3, or 4 Patterns.

Summary of Patterns

Each different pattern has its own strengths and weaknesses. Each different pattern has its own uses. Table 7.21 describes each pattern.

Table 7.21 Synopsis of the Patterns

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 1 Contact Mechanism Pattern	This pattern uses a very specific style of using attributes with specific purposes, usages, priority, and types in order to model contact mechanisms and the classifications of contact mechanisms.	As a part of statement of scope and for use in gathering and validating data requirements. When there is a well known set of data needs for contact mechanisms. When there are contact mechanism types that are only related to a specific entity. When it is known that there are a few very specific types of contact mechanisms. When showing the differences in perception in an enterprise about the same concept and to start the data modeling effort in a simple, understandable model. When contact mechanism attributes, relationships, and entities are very static and hardly ever change.	When flexibility is needed in the model. This pattern does not stand up well to changing or adding contact mechanisms. When there is a need to relate contact mechanisms to other entities or to each other. When there is a need to maintain other data about the contact mechanisms. When there is need to manage many contact mechanisms or many context mechanism types, purposes, usages, or contact mechanism classifications. As a basis for implementation in most cases. Basing the database design on this pattern usually results in a poor implementation choice because it is difficult to change as requirements change over time. When you have many contact mechanism attributes.	Easy to understand and use. Good as part for a simple scope statement and to gather and validate data requirements. Provides very specific rules for contact mechanisms. A good way to start analysis, or highlight differences in perception about contact mechanisms because of its simplicity.	Not a flexible model in any way. This model should generally not be used as the basis for an implementation. Does not allow contact mechanisms to be related to other entities or to other contact mechanisms. Does not provide a very effective way to maintain additional data about the contact mechanism, such as non-solicitation indicator. Does not easily support handling many contact mechanisms or handling many contact mechanism types, purposes, usages, or other contact mechanism classifications. May be impractical and awkward if there are many possible contact mechanisms.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 2 Contact Mechanism Pattern	A pattern that accommodates any number of contact mechanisms, types, usages, and purposes, but it is still a specific model that captures the specific attributes, relationships and rules for contact mechanisms.	When there is a need for more flexibility in modeling contact mechanisms. This pattern maintains any number of contact mechanisms, purposes, usages, or types for contact mechanisms. As a statement of scope in a more traditional logical data model format. This pattern may be more suitable as part of a statement of scope when presenting to other data professionals. When there is a need to balance a specific style of modeling with a generalized style. When there is a need to show different requirements for different stakeholders. When there is a need to show different modeling structures for different types of contact mechanisms. When a simple model is needed for a prototype.	If your enterprise has additional ways to classify contact mechanisms aside from contact mechanism type, purpose type, and usage type. This pattern specifically concentrates on what we believe to be the three key classifications of contact mechanism. If your enterprise wishes to manage all contact mechanisms in the same way. This pattern allows alternative ways to model contact mechanisms (for example, as entities in their own right using an associative entity or as dependent entities of an entity). When other entities have several relationships to various contact mechanism. When this happens there are several places to look (several entities for contact mechanisms), and this can result in more complexity in systems development.	Relatively easy for relational data modelers and IT professionals to understand. This can be used as scope statement for data professionals or users familiar with data modeling. Allows different alternative ways to model different types of contact mechanisms (for example, as entities in their own right using an associative entity or, alternatively, as dependent entities of an entity). Supports the flexibility of adding as many contact mechanisms as needed as well as adding new contact mechanism types, purposes, and usages as requirements change. A good balance between a specific model, supporting specific requirements, specific business rules and the need to flexibly support new classifications of contact mechanisms.	Although it is more flexible than the level 1 pattern, it still lacks some flexibility (for example, adding new types of contact mechanism classifications require new entities). Has more complexity because there are separate data modeling structures for each type of contact mechanism. For example, if an entity uses all contact mechanism types, then many relationships and associative entities are needed (as opposed to the level 3 pattern that has a supertype for all contact mechanisms). Loses some of the clarity of the Level 1 Contact Mechanism Pattern, especially for audiences with little data modeling experience. Does not include all possible ways to classify contact mechanisms. We have described what we believe to be the three key specific classifications of contact mechanism, that is, type, purpose, and usage. There may be many other classifications that we address in later patterns. Different types of contact mechanisms are modeled using different structures, and this requires different management of the contact mechanisms. Thus, this may lead to very different data access routines for different types of contact mechanisms versus a consistent method for maintaining contact mechanism data. Specific rules are included in some of the data model structures, such as a contact mechanism having only one usage type or an electronic address being only for one party. These rules could change over time resulting in data model changes.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 3 Contact Mechanism Pattern	This pattern supports the encapsulation of common attributes and relationships for all contact mechanisms by using a supertype of CONTACT MECHANISM. This pattern also provides a common approach for all different types of contact mechanisms.	When an enterprise wishes to have a consistent way to model all types of contact mechanisms. When there is a need to classify contact mechanisms in many different ways. When flexible management of contact mechanisms is needed.	As a part of a scope statement or to gather and validate data requirements. This pattern is more generalized and subsequently more difficult to understand, especially for nontechnical people. Because this pattern is more generalized, there can be a loss of specificity and business rules in the data model. There is still a way to provide even more consolidation (for example, as in the level 4 pattern) since multiple associative entities are needed for entities that require contact mechanism data.	This is a flexible and consistent approach for supporting contact mechanisms. Because this is a more flexible/generalized pattern, you can use this pattern as the basis for a very stable implementation. It is flexible enough to support a great deal of new and changing requirements for contact mechanisms. This pattern supports a flexible approach for the classification of contact mechanisms, by using the Level 3 Classification Pattern.	Can be hard to understand for non–data professionals and even for some data professionals. Because all contact mechanisms are treated similarly, there may be a loss of some business rules that may be specific to certain types of contact mechanisms. Although we have consolidated contact mechanisms for a more elegant data modeling solution for easier management of the model, there is still a method for consolidating further.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 4 Contact Mechanism Pattern	This supports the 'plug-and-play' concept for contact mechanisms. When an entity is added to a data model that has contact mechanism data needs, all that needs to be done is to attach a relationship to the CONTACT MECHANISM APPLICATION and it uses and inherits a common data model structure.	When an enterprise needs a very flexible approach for all contact mechanisms. When an enterprise wishes to consolidate common entities and attributes to facilitate management and consistency of contact mechanism data. When an enterprise is committed to a fully flexible and integrated data strategy. When the nature of data maintained about various types of contact mechanisms is deemed to be very similar.	When an enterprise has a very limited view and tactical view of contact mechanisms data. This pattern would be overkill for those enterprises. When contact mechanism data has very specific relationships and attributes and is unchanging in its nature, and/or when there are very few types of contact mechanism data.	Very flexible and consistent. Takes the guesswork out of creating contact mechanisms for new data items. Facilitates common routines for contact mechanism data. Simplifies the data model greatly by reducing the number of relationships and associative relationships needed. Provides a similar generalized data model structure for all types of classifications, including purpose types and usage types.	May be too complicated for some enterprises. We would not show this model to most nontechnical stake holders. Somewhat abstract and difficult for some to work with; for example, technical developers may not want to work with this level of abstraction/generalization. Very generic and, therefore, does not incorporate business rules into the data model. Modelers may use this pattern and avoid discovering the specific requirements for contact mechanisms. This pattern does not maintain many of the specific attributes and relationships associated with specific types of contact mechanisms.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Contact Mechanism Pattern with Geographic Boundary	Provides a very flexible data model structure for maintaining all different types of geographic boundaries, the associations between geographic boundaries, and the relationships that geographic boundaries have to contact mechanisms. This pattern can be used to enhance the level 2, 3, and 4 contact mechanism patterns.	When there is a need to have a very flexible solution for maintaining jurisdiction data. When there is a need to maintain the various types of geographic boundaries that exist for domestic and international addresses. When there is a need to maintain which geographic boundaries are related to each other in order to improve data quality. For example, to know the valid postal codes for a certain city. When there is a need to classify telecommunications numbers and electronic addresses by various geographic boundaries.	When there is a need for a specific data model to understand and validate the various relationships between geographic boundaries that are needed. When there is a need to capture specific relationships between specific geographic boundaries. This pattern hides the complex relationships between geographic boundaries, for example, that there may be a one-to-many relationship between COUNTRY and TERRORITY for a certain part of the world. When the enterprise will not buy into this type of more complex data model structure.	Supports all types of domestic and international geographic boundaries. Easy to add new types of geographic boundaries. Easy to add and maintain any type of relationships that exist between geographic boundaries. Can help data quality via a data model structure that can maintain which geographic boundaries should be related in a postal address. Can provide insight by relating all types of contact mechanisms to geographic boundaries including telecommunications numbers and electronic addresses.	Can be hard to understand for non-data professionals. Does not support rules specifically in relationship in the data model, for example, that a CITY is within a STATE (however, this could be added by specifically relating the various subtypes of GEOGRAPHIC BOUNDARY to each other). Can lead to avoidance of understanding specific data requirements by using this generalized data model structure without proper analysis.

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Contact Mechanism with Flexible Address Parts Pattern	Supports a very flexible address part structure in order to maintain various parts of an address in a way that supports a great number of postal address needs, such as international address formats.	When a very flexible address part structure needs to be defined and maintained. When various international address formats are needed. When it is important to be able to search on various parts of the address, for example, when queries are needed based upon the parts of an address, such as what addresses are in buildings, what addresses are for apartments, and so on.	When a much more simple address structure is needed. When the various types of address parts are known and relatively static.	Supports flexible address parts and can create a great variety of address structures including supporting various international address formats.	Somewhat complex. May be overkill in many circumstances, for example, when a simple, static address structure is needed.

References

[1] Paraphrased from http://dictionary.reference.com/browse/contact and http://dictionary.reference.com/browse/mechanism.

[2] In this illustration table, as well as other tables throughout this chapter, we have used party name as a convenient way to refer to parties; however, technically this data may be maintained in several different attributes in the ORGANIZATION and PERSON subtypes of PARTY. Please see Chapter 2 for more details and examples of how a party's name is modeled.

[3] In this illustration table, as well as other tables throughout this chapter, the **state-region** attribute is meant to represent any province, county, district, territory, land, shire, department, canton, prefecture, oblast, autonomous region, and so on.

[4] In this illustration table, as well as other tables throughout this chapter, we have combined multiple attributes for either a telephone number or postal address into a single column in order to aid in formatting. There are, in fact, separate attributes for this column value in the entity.

[5] Please look at Chapter 8 about business rules for more information about this subject.

[6] Please look at Chapters 9 and 10 for more information about how an enterprise can leverage the patterns to create a consistent data strategy.

[7] You can find a comparison of the theories of administrative boundaries at http://www.rev.net/~aloe/boundary/.

[8] See The Data Model Resource Book, *Revised Edition, Volume 1, A Library of Universal Data Models for All Enterprises*, by L. Silverston (Wiley, 2001), Chapter 2.

Chapter 8

Business Rules: How Things Should Work

> *"The central idea behind the concept of business rules is that any organization has logic that it uses to carry out its operational and managerial tasks."*
>
> —Malcolm Chisholm[1]

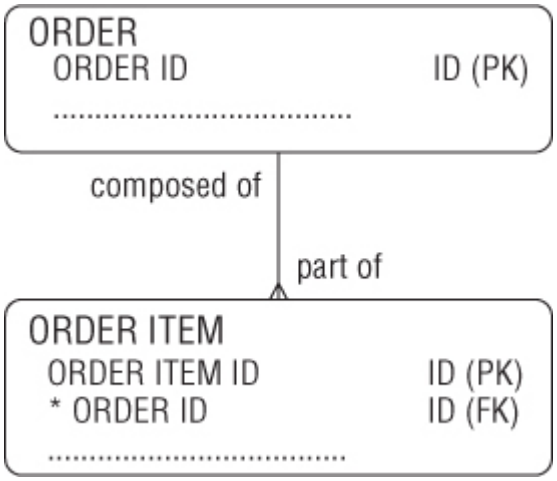
> *"The first and most basic principle in rule management is that your rules should be databased."*
>
> —Ronald G. Ross[2]

Every enterprise has processes, rules, and logic that define how the enterprise functions. For example, armies have standard protocols for command and control, most firms adhere to General Accepted Accounting Principles (GAAP), and investment banking institutions have limit and risk criteria to which they need to adhere. These business rules are expressed in many different ways; for example, as process models, standards documents, and books and other documents that record these rules. Often business rules are not formally expressed at all, but are almost ethereal and may exist only in the minds and experience of employees (the worst-case scenario). Data and business rules have a symbiotic relationship. Data gets affected by rules, and the rules are only guidelines unless they affect the data in some way. Given the importance of business rules, it is crucial to have a set of formal patterns for data models to enable the expression of business rules in a consistent and rigorous manner.

What Is the Significance of This Type of Pattern?

Data models capture the relationships that exist between different entities. These relationships can be regarded as one type of business rule. For example, in Figure 8.1 you can read the data model as "each ORDER *must be* **composed of** one or more ORDER ITEM(s) and each ORDER ITEM *must be* **part of** one and only one ORDER."(3) This is a useful statement that specifies how ORDER(s) and ORDER ITEM(s) need to behave for an enterprise. For instance, this model suggests the rule that an order must contain at least one item that is ordered. But what of a business rule such as "A person is a woman if the person is female and the person's age is 21 or over"?(4) This business rule is much more difficult to capture explicitly as relationships in a data model, yet you may still want to have a structure to capture this business rule and relate it to PERSON(s) or PARTY(s). In this case, instead of trying to model that specific rule statement, this chapter provides a pattern that can be used to develop a data model that maintains business rules such as this as well as the factors that determine when someone is considered a woman, a man, a child, a teenager, and so on. Thus, you can develop a model that "databases" these rules and includes each specific rule statement, the relationships the rule has to other entities in the data model, the factors that affect that rule, and the different possible outcomes of the rule.

Figure 8.1 Order, Order Item



This chapter is a specialized topic; hence, it's the last patterns chapter. The patterns in this chapter are different from the other patterns in this book in one very significant way. These business rules patterns model ‘metadata’ as distinguished from ‘business data.’ What do we mean by this? Business rules describe how data in entities are influenced or guided. This can be considered metadata because we are further describing the data. Some modelers may ask “Why are you addressing business rules in this book when this is related to metadata and not business data?” We see metadata as data, just a different type of data.

In our experience, the question of modeling rules occurs quite often in many scenarios where we are modeling business data requirements. In these scenarios, there is often an expectation from business representatives that business rules and business data are both part of the same problem you are trying to solve. In other words, the business does not distinguish between metadata and business data. For this reason we believe that it is important to have a way to model these rules and not just say, “Oh, that is metadata and out of the scope of our data modeling effort!”

For example, a common data modeling requirement is to model the price for a product. This seems like a business

requirement as opposed to a metadata requirement. However, when you look a little deeper, you notice that the price may vary based on the geographic area (United States versus European pricing), the type of party (government versus commercial), the quantity or volume that is bought, and many, many other factors. Thus, what you are really modeling are the various factors that affect the price of a product, or in other words, the business rules for pricing and how the price of a product may be influenced by different factors that may already be captured as entities in the model. Whenever you are modeling a situation where various factors influence the data that is being maintained (for example, the price or percentage discount of a product), it is an indication that you are modeling rules.

The need to model rules happens in many scenarios where we want to model requirements such as the following:

- What price to charge for what product or service.
- How to prioritize and/or rank customers or suppliers and what factors to use in doing this. For example, customers who order over a certain monetary amount for goods and services over a specific period of time may be given a higher priority.
- Which manufacturing plant or which logistics carrier to use under different conditions. For example, if you have an order for goods in Hong Kong, you should use your Shanghai manufacturing plant to make these goods, not your plant in New Zealand.
- How to follow up with customers based on various conditions. For example, what is the recommended action when someone first becomes a customer (such as to send out a thank you letter) or when someone files a complaint (such as to call that person)?
- How international postal addresses should be formatted in different countries. For example, in the United States you present the address from the most specific

information first (the name of the recipient) to the most general information last (the name of the country). This is not the same in Iran or Russia.

- How and when to replenish inventory. For example, you may want to replenish inventory only when you have ‘pipelined’ a certain amount of orders.

Another clue that you are modeling a rule is when you notice conditional statements, such as ‘if, then’ statements or ‘while’ statements. For example, if the geographic boundary of the ship-to address is in New York, and if the quantity is greater than 100 items being ordered, and if the type of customer is a governmental customer, then the price is $100 for this product. Or another example, if a customer files a complaint, and if this customer is a gold loyalty member, and if the customer has ordered more than $10,000 from you this year, then you have a vice president call that customer.

The beauty of this pattern is that when you recognize that you are dealing with data requirements involving business rules, you can choose to model this and other types of business rules consistently with one of the business rules patterns that we will share in this chapter. We have been in many situations where we have modeled something and then realized later that this was in fact a ‘rule’ data modeling structure. By identifying types of conditional statements (such as ‘if, then,’ ‘while,’ or ‘in case’ statements) or data requirements that are factors and/or outcomes, you can recognize rules up front. Then, using the rules pattern, you can create more consistent data models by combining various rules in the same data modeling structure when appropriate, and also develop higher quality models by using data model constructs that have already been well thought out.

Because business rules are crucial to many enterprises, this chapter provides patterns that we use to model these

business rules in a consistent fashion and in a manner that is integrated with other entities in a data model.

What Is in This Chapter?

The chapter describes data model patterns that can be used to support the needs of an enterprise when it wishes to maintain business rules in its data model. The chapter starts by defining business rules and then provides two very powerful patterns for modeling business rules using data model structures.

Like most of the chapters in this book, the style of modeling for each of the patterns starts with a more specific style and moves to a more generalized style. However, this chapter is different from the other chapters in the book in that it does not describe a Level 1 Business Rules Pattern. It is possible to create a Level 1 Business Rules Pattern where the rule factors and rule outcomes could be captured as attributes in an ENTITY RULE (where entity is the subject of the business rule). Because in most cases, but not all cases, factors and outcomes are maintained as entities as we see in the Level 2 Business Rules Pattern, we chose not to show a level 1 pattern for business rules.

Different levels of generalization may be applicable to different enterprises or situations. For example, an enterprise wanting to maintain specific business rules might use the Level 2 Business Rules Pattern, whereas an enterprise wanting a single, generalized structure for all business rule models might use the Level 3 Business Rules Pattern.

In this chapter we illustrate this pattern with the following two business rules scenarios:

- The rules regarding how to respond, under different conditions, to certain EVENT TYPE(s) (an *event* is an activity that happens at a given place and time and that

may trigger other activities, for example, a phone call responding to a complaint from a customer).
- The rules, based upon various factors, regarding pricing for a PRODUCT or PRODUCT FEATURE.

Although we focus on these two scenarios, please note that the patterns in this chapter work for all different types of business rules.

This chapter includes the following:

- The definition of business rules
- The different patterns that support business rule definition
- The relevance of each pattern
- Insights into each pattern
- When to use and not to use different patterns
- A synopsis of each pattern's pros and cons.

What Is a Business Rule?

Business rules may be defined in many different ways. For example, they can be defined broadly such as “a directive intended to influence or guide business behavior”(2) or narrowly such as “a business data rule is a constraint on the data beyond the constraints implied by the data model.”(5) This last definition is interesting because it implies that business (data) rules are constraints that are beyond the data model. We explore this idea further in the section describing the Level 2 Business Rules Pattern.

This chapter concerns itself with business rules that guide business behavior and also that maintain constraints. For example, we are concerned with business rules such as:

- **We must send an apology letter to a customer if we get a valid customer complaint:** This rule defines how to act when a particular type of event (EVENT TYPE)

occurs, in this case, getting a valid complaint from a customer.

- **We must give a 2 percent discount on all products for minority-owned businesses:** This rule has to do with how we price PRODUCT(s) under various conditions.
- **Normal tax return due date must be set to “April 15” in the United States of America:** This business rule is specifically related to TAX RETURN(s), the attribute **return due date**, and the GEOGRAPHIC BOUNDARY “United States of America.”

In the patterns in this chapter, we capture data about the business rules, the different types of factors that affect the business rules, and the outcomes that are the result of a business rule.

The following concepts need to be supported in the business rules patterns:

- **The patterns need to be able to capture the core data about the business rules:** For example, they should be able to capture a business rule statement such as “We must give a 2 percent discount on all orders for minority-owned businesses.” The patterns also need to be able to classify business rules.
- **The business rules are affected by factors, which are circumstances that affect the outcome. These factors need to be captured by the pattern** Factors may be related to specific entities; for example, a factor may be the geographic boundary that one is in. For example, the price of a television in Europe, a GEOGRAPHIC BOUNDARY, may be different from the price of the same television in Asia, a different GEOGRAPHIC BOUNDARY. Or, the price of a PRODUCT may have business rules such as “We must give a 20 percent discount on all orders for Disk Drives for over $999,999 and over 1000 units sold” that may in turn be

affected by different factors that are represented by entities of PRODUCT CATEGORY, ORDER VALUE, and QUANTITY BREAK.

- **The patterns need to be able to capture all the outcomes (and outcome types) of that business rule:** For example, the pattern may maintain an outcome of what the price is under a set of specific conditions, if the price will be reduced by 2 percent based on various factors, or if a surcharge of 1 percent will be added to the price.

Note

The business rules pattern does not capture information on what happened, only what is supposed to happen in different situations. In other words, we capture the statement of the rule, all the different factors that affect a rule, and all the different outcomes of the rule. When a specific business rule is applicable for a specific set of factor values, the rule states that a specific outcome or set of outcomes should occur. For example, a rule stating that a discount should affect a price for a specific product under certain conditions or a rule that specifies that when a certain event occurs, it requires that a specific communication event such as a telephone call be made to a valued client.

Often people regard process modeling (where many business rules are captured) as completely distinct from data modeling. Enterprises often capture business rules independent of data models in process models, rule dictionaries, metadata repositories, or rules engines. However, we feel that while these rules may be captured in other types of models, you can achieve great advantages by maintaining many business rules in the data model. Rules and data are often tightly linked, and there are many situations where it is beneficial to maintain them in the same model. We believe data modelers should have templates for how to capture data about rules in the data model, and use this for the business rules they come across

in the process of doing their analysis. When business rules are maintained in a data model and then implemented in a subsequent database design, they can then be dynamically changed. For instance, if you use a data model to maintain all the various rules regarding how a product is priced (for example, 2 percent discount when more than 100 are sold), then when that rule changes (for example, to 3 percent discount when more than 100 are sold), you can often simply change the instances of the data model (and the database that was based on the data model) without changing the structure of the data model (or database) or even the application. Thus, you can be more ‘data-driven’ and dynamic.

Level 2 Business Rules Pattern

Business rules can be modeled and managed in a specific way using the Level 2 Business Rules Pattern. Using this pattern, you can capture data for a particular type of business rule and then re-use the pattern for each additional type of business rule. The data captured includes the business rules name, business rule statement, factors, outcomes, and the different ways to classify the business rule. This pattern allows you to use the specific factors that already exist in your data model to influence a business rule. This pattern also allows you to capture generalized factors that may not exist in your data model already. This pattern captures each of the different outcomes that may be the result(s) of the business rule. This pattern provides a flexible strategy for data professionals when they wish to develop a specific data model for each type of business rule.

Why Do We Need This Pattern?

The purpose of the Level 2 Business Rules Pattern is to maintain business rules data such as the business rule

statement, business rule factors, and business rule outcomes for a particular situation, problem area, or business subject area. For example, imagine a retail firm is interested in defining the stock replenishment business rules, including the factors affecting that rule, and all possible stock replacement rule outcomes. For this example, the factors may be the product, the reorder level, the location, and the time of year. Based on the combination of these factors, a “reorder quantity” may be assigned, and this represents the ‘outcome.’ To illustrate this, for the product “A123,” if the stock level falls below 100 units (that is, the reorder level) at the Paris plant (the location) in the fall/autumn (the time of year that is busy), then reorder 200 of these units (the reorder quantity, which is also the outcome of the rule for this set of circumstances).

Many situations such as this one occur while data modeling, and this pattern allows data professionals to address each situation with a common template to model rules, and their associated factors and outcomes, and integrate these rules into their data model.

Note

Some enterprises prefer to maintain business rules by recording the data about the rules in a document or spreadsheet that is an adjunct to the data model.(5) A possible format could be to provide:

- The name of the business rule. For example, “Maximum Course Enrollment Rule.”
- A business description of the rule. For example, “When enrolling students, the number of students enrolled in a course must not be greater than the maximum allowed seats for that course.”

- A ‘Model Definition’ of the rule. This is a SQL-like (or any other language) definition of the rule, using the entities and attributes in the model. Our example of a college course where the number of students enrolled must not exceed the “Maximum Course Seats” might have a Model Definition of “if count (COURSE ENROLLMENT) > COURSE. Maximum Allowed Seats where COURSE ENROLLMENT. Course Id = COURSE. Course Id, then send ERROR MESSAGE. name of “Course maximum enrollment exceeded.” It should be noted that this assumes there is a COURSE entity with a Maximum Allowed Seats attribute and a COURSE ENROLLMENT entity in the model.
- Outcomes (such as a success or an error message). For example, either a success outcome with a message “Successful enrollment” or an error message such as “Course maximum enrollment exceeded.”
- Factors influencing the rule. For example, the maximum allowed seats for the course and the number of students already registered.

Although this may be a good option in certain circumstances, instead of just having business rules captured in a document that is an appendix to the data model, we show in this chapter how to maintain many business rules in a data model with its related entities (which could then eventually be implemented in a database). This brings the business rules front and center. Instead of just saying that a business rule is metadata only, we believe that a business rule may be integrated to its related entities at the data model and database levels.(6)

Note

Many enterprises believe business rules go into a metadata repository (or a business rules engine) related to the data model. We don't disagree, but in our experiences, many enterprises have a very difficult time implementing a completely integrated metadata repository (or a business rules engine). This pattern provides a useful alternative, which is to maintain many business rules directly within the data model. Even if you are implementing a metadata repository (or business rules engine) these patterns may be useful as a stopgap measure until you get your implementation up and running.

How Does This Pattern Work?

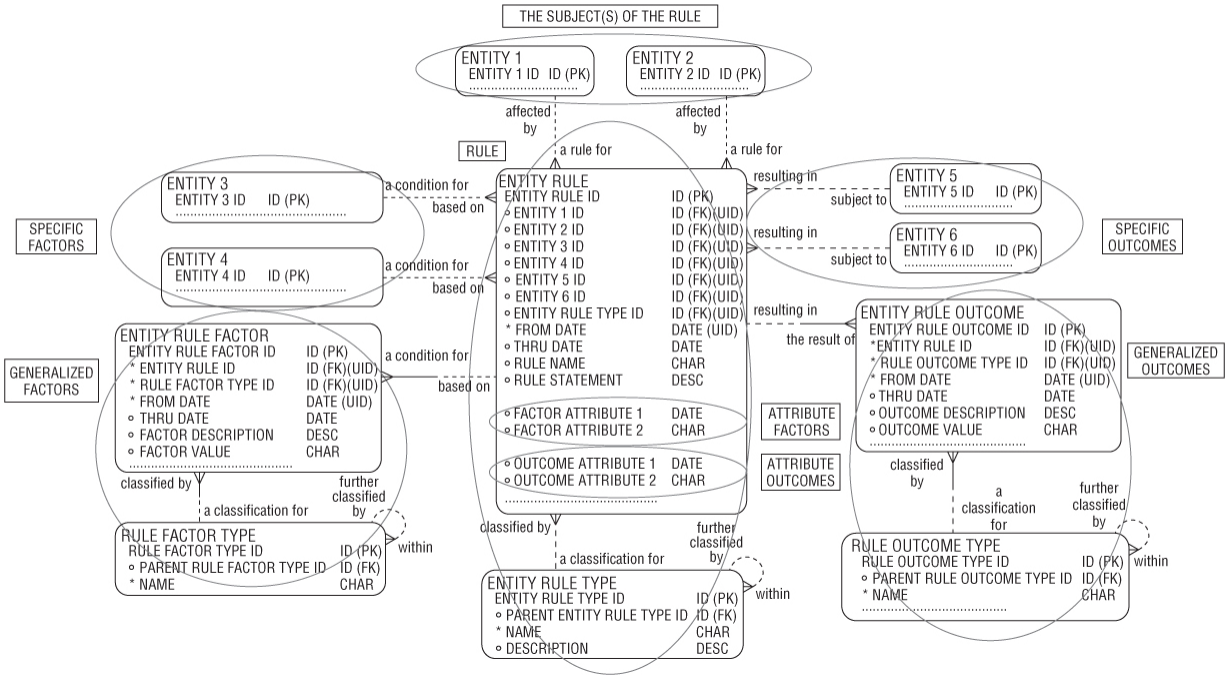
Figure 8.2 shows a level 2 pattern for maintaining business rules data. This figure displays several parts of the business rules pattern, namely:

- **The subject(s) of the rule (shown at the top of the diagram)** An entity or entities that are affected by a rule. For example, a business rule about pricing may affect the PRODUCT and PRODUCT FEATURE entities, which would be the subjects of the pricing rule.
- **The rule (entity) itself (shown in the middle of the diagram)** The rule entity is the central entity that is related to its subject(s), factors, and outcomes. The rule also includes **rule name**, the **rule statement**, and how it is classified (via the ENTITY RULE TYPE entity).
- **The factors (shown on the left side and in the middle of the diagram)** These are the conditions that affect the outcome of the rule. For example, the geographic boundary or the type of party involved may influence the outcome of the rule. Factors may be classified as attribute factors, specific factors, or generalized factors.

    - **Attribute factors** are maintained as attributes of the rule entity. These are illustrated in Figure 8.2 as **factor attribute 1** (a “DATE” data type) and **factor attribute 2** (a “CHAR” data type), however, factor attributes may be any data type and often these attribute factors are **effective from date** and **effective thru date** (an example of this is later on in Figure 8.4 on page 443) to show that a condition of the rule is that it is only effective between these dates (of course, there may be any number of attribute factors).
    - **Specific factors** relate to specific entities that already exist in the data model. In Figure 8.2, these are ENTITY 3 and ENTITY 4.
    - **Generalized factors** are used for other factors that are not stored as attributes and that are not related to existing entities. In Figure 8.2, these are captured in ENTITY RULE FACTOR and RULE FACTOR TYPE.
- **The outcomes (shown on the right side and in the middle of the diagram)** These are the results of the rule. For example, there may be a rule that has a price as an outcome, based on factors such as geographic boundary and the type of party that is buying. Similar to factors, outcomes are also classified as attribute outcomes, specific outcomes, or generalized outcomes.
    - **Attribute Outcomes** are maintained as attributes of the rule entity. In Figure 8.2, these are **outcome attribute 1** and **outcome attribute 2** (of course, there may be any number of attribute outcomes).
    - **Specific Outcomes** relate to specific entities that already exist in the data model that may be the outcome of a rule. In Figure 8.2 these are ENTITY 5 and ENTITY 6.
    - **Generalized Outcomes** are used for other outcomes that are not stored as attributes and that

are not related to existing entities. In Figure 8.2 these are captured via ENTITY RULE OUTCOME and OUTCOME RULE TYPE.

Figure 8.2 Level 2 Business Rules Pattern



In Figure 8.2, at the top of the diagram you see ENTITY 1 and ENTITY 2, which represent any entities that you wish to capture ENTITY RULE(s) about, or in other words, the subjects of the business rule. You see that “each ENTITY 1 (and/or ENTITY 2) *may be* **affected by** one or more ENTITY RULE(s) and each ENTITY RULE *must be* **a rule for** one and only one ENTITY 1 (and/or ENTITY 2).” In other words, the ENTITY RULE is the business rule that affects ENTITY 1 (and/or ENTITY 2), and ENTITY 1 (and/or ENTITY 2) is the subject of ENTITY RULE. These entities (ENTITY 1 or ENTITY 2) could be PRODUCT, ORDER, SHIPMENT, PARTY, or any number of other entities that have rules. Thus, depending on the subject, there may be PRODUCT RULE(s), ORDER RULE(s), SHIPMENT RULE(s), or PARTY RULE(s). For example, SHIPMENT is the subject of SHIPMENT RULE(s). SHIPMENT RULE(s) would contain rules such as “a shipment of

computer parts must not be sent to North Korea" or "a shipment is not considered delivered without a valid signature from our clients."

Unlike other patterns in this book, the ENTITY RULE does not always have to use the convention of replacing ENTITY with the entity at hand. There may be a PRODUCT that is the subject of a PRODUCT REPLENISHMENT RULE (that determines how and when to replenish inventory of that product) and the subject of a PRICE COMPONENT RULE (that determines pricing based on various factors). Or there may be a CUSTOMER entity that is the subject of a PRIORITY RULE that determines how to prioritize or rate customers based on various factors.

Often, there is a single entity that is the subject of the business rule; however, there may be more than one entity that is the subject of a business rules structure. For example, as you can see in Figure 8.4 later in the chapter, there may be a PRODUCT and a PRODUCT FEATURE (a variation in the product such as color, size, and so on) that may be the subject of a PRICING RULE, which may maintain the prices for either a PRODUCT or PRODUCT FEATURE. Additionally, an entity may sometimes have multiple relationships to business rules entities. For example, there may be a PRODUCT entity that has a relationship to a PRICE COMPONENT RULE, PRODUCT REPLENISHMENT RULE, and PRODUCT REGULATION RULE specifying what types of regulations must be enforced under various circumstances. This latter rule may maintain information about how certain products may (or may not) be sold based on different factors. For example, according to security regulations, a financial securities company may not be allowed to sell a particular hedge fund (an investment with high yields, but also with high risk of losses) to a party that does not have a certain level of net worth, or the amount of their portfolio

that a party is allowed to have in a hedge fund may be limited to a certain percentage.

Note

We have found that many of the core entities in an enterprise may have business rules associated with them. For example, for logistics firms, SHIPMENT RULE(s) and PARTY RULE(s) often need to be captured in the data model. For investment banks, TRADE RULE(s), LIMIT RULE(s), and PRICING RULE(s) are often captured explicitly as entities (and eventually as tables). This may leave you with the impression that every entity in your data model must have an ENTITY RULE for its business rules. This is not the case. Normally we find that only core entities are the subject of business rules, and the other entities in the data model may be used as factors (or possibly outcomes) of the business rule entity (ENTITY RULE), or they may have no relationship to a business rule at all.

ENTITY RULE captures the attributes and relationships for the business rule. Thus, ENTITY RULE is related to the subject(s), factors, outcomes, classification, and any other data about this business rule. For example, a business rule stating “We must give a 2 percent discount to all minority-owned businesses” would be captured in the **rule statement** attribute of a PRICE COMPONENT RULE. This rule may have a **rule name** such as “Pricing rule 1280—minority business discount.”

Note

There may be other attributes or entities that supplement the modeling of business rules and that could be included in this pattern (or in subsequent patterns). For example, there may be attributes or entities for:

- A **rule note,** an explanatory or critical comment.
- A **rule reference** that points to a reference to some quoted authority. For example, the rule “Congress shall make no

law respecting an establishment of religion, or prohibiting the free exercise thereof …" is Amendment 1 of the Bill of Rights for the Constitution of the United States of America, which is referenced from "The Library of Congress." We could have other quoted authorities for this rule, such as "Cornell Law School" or "Annals of Congress 434 (June 8, 1789)."

- A **rule source** specifying where the rule came from (rules should be single sourced).(2) The source of "Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof …" is "The Constitution of the United States of America, Amendment 1."
- A **rule specified by** stating who crafted or defined the rule. There may also be a rule that "Rules should be specified directly by those people who have relevant knowledge." For example, "Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof …" was specified by "James Madison."
- A **rule managed by** specifying the party(s) who are responsible for managing the rules. For example, "Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof …" is managed by "The Supreme Court (a custodian of the rule, the arbiter of the rule, and the interpreter of the rule), and the President (who swears to preserve, protect, and defend it)."

- An **external reference id** that could capture an identifier to a source for business rules such as a centralized business rules engine or metadata database. For example, "http://www.usconstitution.net/const.html" for "Congress shall make no law respecting an establishment of religion, or prohibiting the free exercise thereof …"

Each of these pieces of data may be maintained as attribute(s) or in its own entity. For example, if a **rule reference** is reused (for example, if Cornell Law School is referenced many times) or if there is additional data or relationships about the reference (for example, there is a long name and abbreviated name for the reference), then the **rule source, rule specified by,** and **rule managed by** data could be handled by using one of the contextual role patterns from Chapter 3 that relate various PARTY(s) to each rule via different roles (see Figure 8.7 and the final section of this chapter for an explanation of this).

The pattern shows that that "each ENTITY RULE *may be* **classified by** one and only one ENTITY RULE TYPE." For example, you might classify the instance of a PRICE COMPONENT RULE, "We must give a 2 percent discount to all minority-owned businesses" as a PRICE COMPONENT RULE TYPE **name** of "Discount." Another PRICE COMPONENT RULE instance having a **rule statement** of "The price for Product A467 in Europe during 2009 is €15" may be classified as a PRICE COMPONENT RULE TYPE **name** of "Price." There may be other classifications of price component rules such as "Surcharge," "Recurring charge," and "Utilization charge." The recursive relationship around the ENTITY RULE TYPE allows rule types to roll up to each other. For example, you might classify the rule "We must give a 2 percent discount to all minority-owned businesses"

as a “Minority discount,” which is **within** the rule type of “Discount.”The **description** attribute in ENTITY RULE TYPE is important to be able to explain the nature of the rules that are maintained in ENTITY RULE. For example, there may be a number of ENTITY RULE(s) that are classified as a “Discount,” which may be the ENTITY RULE TYPE **name**. The **description** may be “This type of rule specifies the factors when a price is reduced and results in an outcome of either a percentage or an amount regarding the price reduction.”

Note

An alternative way to classify rules would be to classify them based on the BRS Rule Classification scheme or based on Rule Speak Templates.(2) If you need to classify business rules in more than one way, you can use the Level 3 Classification Pattern, as seen in Figure 8.5 later in the chapter, to classify a business rule in multiple different ways. This illustrates the strength of patterns. When you need a more flexible solution to classify business rules, you may simply plug in the Level 3 Classification Pattern for a more generalized solution. If you needed a more specific solution you could swap out the Level 3 Classification Pattern for the level 2 or even the level 1 pattern.

A factor may be defined as one of the elements contributing to a particular result or situation, or in other words, conditions that affect the outcome of the rule. For example, for a PRICE COMPONENT RULE instance that has a **rule statement** of “if a customer resides in North America, that customer must get a 10 percent discount on any order between $500,000 and $999,999,” one factor in this business rule is if the customer residence is in the geographic boundary of “North America.” Another factor is if the value of an order is in a particular range, that is, between “$500,000” and “$999,999.”

Factors may be classified into three different categories: first, factor attributes that exist as attributes of the business rule; second, entities that already exist in your data model,

or are needed as entities for this rule (we refer to these as specific factors because they are rules that are specific to entities in the model); and finally, other generalized factors.

The first category is attribute factors that exist as attributes of the business rule and are represented as **factor attribute 1** and **factor attribute 2** in the ENTITY RULE entity in Figure 8.2. These factor attributes are often a range of dates such as a time period when a rule can be applied. For example, a discount rule for “fur coats” may be valid only for the summer, that is, between June 1 and September 1, so these factor attributes could be **effective from date** and **effective thru date** when the pattern is applied.

The second category, specific factors, includes factors that are entities that exist already in your data model. These are represented by ENTITY 3 and ENTITY 4, as you can see on the left-hand side of Figure 8.2. In our example, “if a customer resides in North America, they must get a 10 percent discount on any order between $500,000 and $999,999,” “North America” would be captured as a relationship to the instance of “North America” in the GEOGRAPHIC BOUNDARY entity. GEOGRAPHIC BOUNDARY could have already been defined in the data model.

It is important to point out that business rules don't exist independently. They are often connected to many different factors that may already exist as values from entities in your data model. When you are capturing business rules in this pattern, it is important to use existing entities to capture the different existing factors that influence the outcome of the business rule and relate them to ENTITY RULE. By doing this, you can avoid redundantly capturing data in the model. For example, if a factor is geography, instead of entering in all of the geographic areas in a RULE FACTOR TYPE entity, or creating geography attributes in the ENTITY RULE, you can relate the ENTITY RULE to an existing

GEOGRAPHIC BOUNDARY entity. Additionally, by relating these factors to existing entities, you often realize the true scope of business rules for a particular entity; in other words, you can see the entities that are conditions for a business rule. Also, if the business rule is integrated into your data model and the entity that provides the factor data changes in some way, this helps capture the effect of the change on the business rule. For example, if the rule says that there is a 10 percent discount for “North American” sales of a certain amount, and you update the GEOGRAPHIC BOUNDARY instance to “N. America,” then the rule and the rest of the data model must be in sync regarding that data value.

The last category of factors includes factors that are captured as RULE FACTOR TYPE(s). These are generalized factors that are not explicitly captured as relationships to existing entities in the data model, or as specific factor attributes in the business rule, but as instances of RULE FACTOR TYPE. For example, in the rule “Passengers that have made 3 or more international trips during the current year must get preference for flight upgrades over passengers with 0 to 2 international trips,” “0 to 2,” and “3 or more” are examples of **factor value(s)** in ENTITY RULE FACTOR(s) of RULE FACTOR TYPE “International Trips For Current Year.” The reason that these factors are not modeled as specific entities (by having an entity called INTERNATIONAL TRIPS FOR CURRENT YEAR) is that this may be the only place where this data is used. Does it make sense to create a new entity to support this very specific need? There may be many such specific factors that may not be modeled as entities in their own right. ENTITY RULE FACTOR(s) and RULE FACTOR TYPE(s) support these factors, which we refer to as ‘generalized’ factors.

So, when should one use these ‘generalized’ factors as opposed to modeling them as attributes? Specific attributes,

such as **effective from date** and **effective thru date**, could alternatively be modeled using the ENTITY RULE FACTOR **factor value** (which would maintain the relevant date) and RULE FACTOR TYPE(s) name (which would maintain “Effective from date” and “Effective thru date”). We use the following guideline; when the factor is a single atomic piece of data (it does not represent a repeating group of values), and there is no other data or relationships to this specific piece of data, and if the same value is not reused over and over (and thus a lookup entity would be used), it may be maintained as an attribute.

Another consideration is whether the attribute is common for most of the rule instances. **Effective from date** and **effective thru date** are examples of **factor attribute 1** and **factor attribute 2** that may apply to almost every instance of an ENTITY RULE, and therefore they are generally modeled as specific factor attributes. The ‘generalized’ factors may be specific only to certain business rules captured in ENTITY RULE; hence, they are captured in ENTITY RULE FACTOR. For example, the factor of “Maximum number of miles that furthest customer location may be from distribution center” may apply only to a few of the rule instances in the PRICE COMPONENT RULE and thus is empty most of the time if it is modeled as an attribute. Furthermore, if you model all these generalized factors as attributes, you lose flexibility and have to manage a lot of attributes.

A big advantage of using the ENTITY RULE FACTOR and RULE FACTOR TYPE to capture generalized factors is flexibility. If rules have additional factor types over time, they can be added as instances of RULE FACTOR TYPE. For example, if there were additional factors about upgrades that arose over time, such as the number of trips within the passenger's native country or the total number of trips,

these could be added as additional ENTITY RULE FACTOR(s) and RULE FACTOR TYPE(s).

It is up to you (who should reflect the needs of the business) as to how you model different types of factors (as attributes, specific factors, or generalized factors). In other words, should you model factors as attributes because they apply to every ENTITY RULE, as specific factors that already exist in your data model as entities (such as GEOGRAPHIC BOUNDARY(s) for pricing rules), or as generalized factors in ENTITY RULE FACTOR(s) because there are no related entities in your data model, such as “International Trips For Current Year”? While we have given some guidelines and options, the answer often depends on the specific circumstances involved.

Note

We often find that generalized factors are related to derived data, because derived data is not stored in the data model. For example, the number of international trips for a customer in a period of time would be derived data and, therefore, not directly maintained in the data model. Average amount of time that it takes customers to settle their bills would also be derived data. When this type of derived data is involved, this can be a good indication that it is appropriate to use generalized factors.

Notice that each of the different specific factors related to existing entities is optional for the ENTITY RULE. For example, “each ENTITY RULE *may be* **based on** one and only one ENTITY 3 and each ENTITY 3 *may be* **a condition for** one or more ENTITY RULE(s).” Is it possible that an ENTITY RULE could have mandatory factors or that all factors are mandatory? The answer is yes, but we have found more often than not, business rules mix and match factors to give different results. For example, a pricing rule may use various combinations of GEOGRAPHIC BOUNDARY(s), ROLE TYPE, PRODUCT CATEGORY, QUANTITY BREAK, and/or ORDER VALUE as different factors, and other

rule instances may use any combination of these factors. Thus, there may be an instance of a PRICE COMPONENT rule that records that there is a 5 percent discount for the GEOGRAPHIC BOUNDARY of “United States of America” and the PRODUCT CATEGORY of “Accessories” for a certain time period. Another instance may record that there is a 3 percent discount for the GEOGRAPHIC BOUNDARY of “Canada” for an ORDER VALUE that is over “CAN$2,000” for a certain time period. Thus, any instance of the RULE ENTITY may be related to any combination of the factors. That is why we have optional relationships from the factors to the business rule and optional factor attributes in ENTITY RULE.

The outcomes of a business rule may be maintained in three different ways:

- First, the outcome may be modeled as an attribute. If the outcome(s) of a business rule is a single atomic piece of data (if there is not a repeating group of values), there are no other data or relationships to this specific piece of data, and if the values of the outcome(s) are not reused over and over (and thus a lookup entity would be used), it can be captured as an attribute of the ENTITY RULE. In Figure 8.2 you see the attributes **outcome attribute 1** and **outcome attribute 2** as two possible attributes that may meet this criteria. For an additional example, look ahead to Figure 8.4 later in the chapter. There in the PRICE COMPONENT RULE, the outcomes may be one or all of the following; **price amount**, **discount amount**, **surcharge amount**, **discount percentage**, and **surcharge percentage**. These are examples of using the template attribute of **outcome attribute 1** and **outcome attribute 2**. These values are atomic in that there is no additional data about the outcomes. Also, the values for each outcome are not generally reused

for other instances of the rule. In other words, an instance of the PRICE COMPONENT RULE will generally produce a **price amount**, **discount amount**, **surcharge amount**, **discount percentage**, or **surcharge percentage** just for that instance. Thus, they are modeled as attributes.

- Second, an outcome of a rule may be an instance of an entity that exists in the data model. This is illustrated in Figure 8.2 via ENTITY 5 and ENTITY 6. For example, ENTITY 5 may be a WORK EFFORT TYPE, which may be the object of a rule for monitoring and auditing the total spend of projects. If a project goes over budget by a certain amount, then the rule may result in the need for an "Audit project" WORK EFFORT TYPE that should be set up to investigate why a project is overspending.
- Finally, you can handle the outcomes of a business rule in a more flexible way as one or more ENTITY RULE OUTCOME(s). On the bottom right-hand side of the Figure 8.2, you can see the entities for the generalized outcomes of the ENTITY RULE. Based on combinations of the different factors (ENTITY 3, ENTITY 4, ENTITY RULE FACTOR, **factor attribute 1** and **factor attribute 2**), a business rule may result in different outcomes. The business rule, "Make a phone call within one day to any gold member loyalty customer that files a valid complaint and offer some compensation" might have several different outcomes, for example, "Call the customer within one day" and "Offer the customer a gift coupon." Thus, there may be several outcomes to maintain, and each outcome may have additional data, such as the value of "1" day or the monetary amount of the coupon offered.

So, why are these outcomes important? The outcomes maintain the result(s) of the rule, or in other words, what should happen under various conditions, and thus, they are

an integral part of the rule. Also, these outcomes of the rules may be inputs into other rules, or they may trigger a transaction or a process. For example, if the outcome of a LIMIT RULE for a TRADE is “Limit exceeded by more than $100 US,” this may trigger another rule that results in an email to the credit risk department of the bank. In terms of the data model, a LIMIT RULE OUTCOME entity (an example of an ENTITY RULE OUTCOME) may be a factor for a COMMUNICATION EVENT RULE, and if a ‘Limit exceeded by more than $100 US’ outcome happens, the COMMUNICATION EVENT RULE OUTCOME is “Email credit department.” Thus the LIMIT RULE OUTCOME could be a specific factor and a condition for the COMMUNICATION EVENT RULE.

Note

While the pattern in Figure 8.2 shows one-to-many relationships between the ENTITY RULE and the factors, and one-to-many relationships between the ENTITY RULE and the outcomes, there may be many-to-many relationships for each of these and you may need to modify the pattern to accommodate this. For example, a pricing rule may be based on several GEOGRAPHIC BOUNDARY(s), such as when a certain discount percentage is offered to the states of “New York,” “New Jersey,” and “Connecticut.” Thus there may be a many-to-many relationship from the PRICE COMPONENT RULE to the specific factor of GEOGRAPHIC BOUNDARY. The Level 3 Business Rules Pattern shown later in this chapter explicitly shows these many-to-many relationships for both specific factors and outcomes.

To further elaborate on this pattern, imagine a computer hardware retailer called Kantowitz Electronics. As part of its effort to understand its business, the company employed a data modeler with experience in process modeling to examine two distinct and different parts of its business. As part of the assignment, the data modeler had to produce a consistent approach to handling business rules across different aspects of the business. Based on these

requirements and using the Level 2 Business Rules Pattern, the data modeler produced Figure 8.3 and Figure 8.4.

Figure 8.3 Example of using the Level 2 Business Rules Pattern, Communication Event

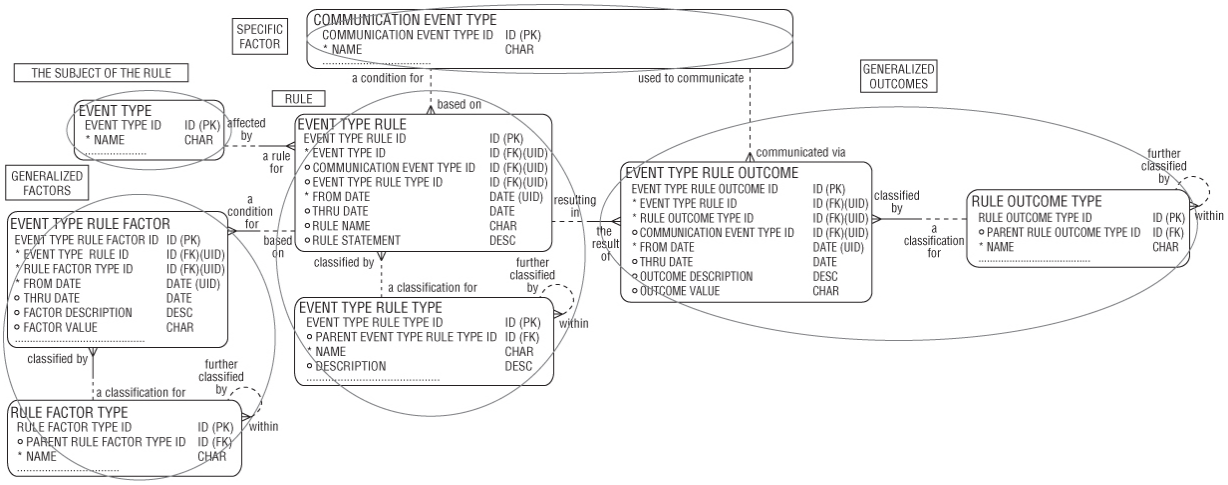
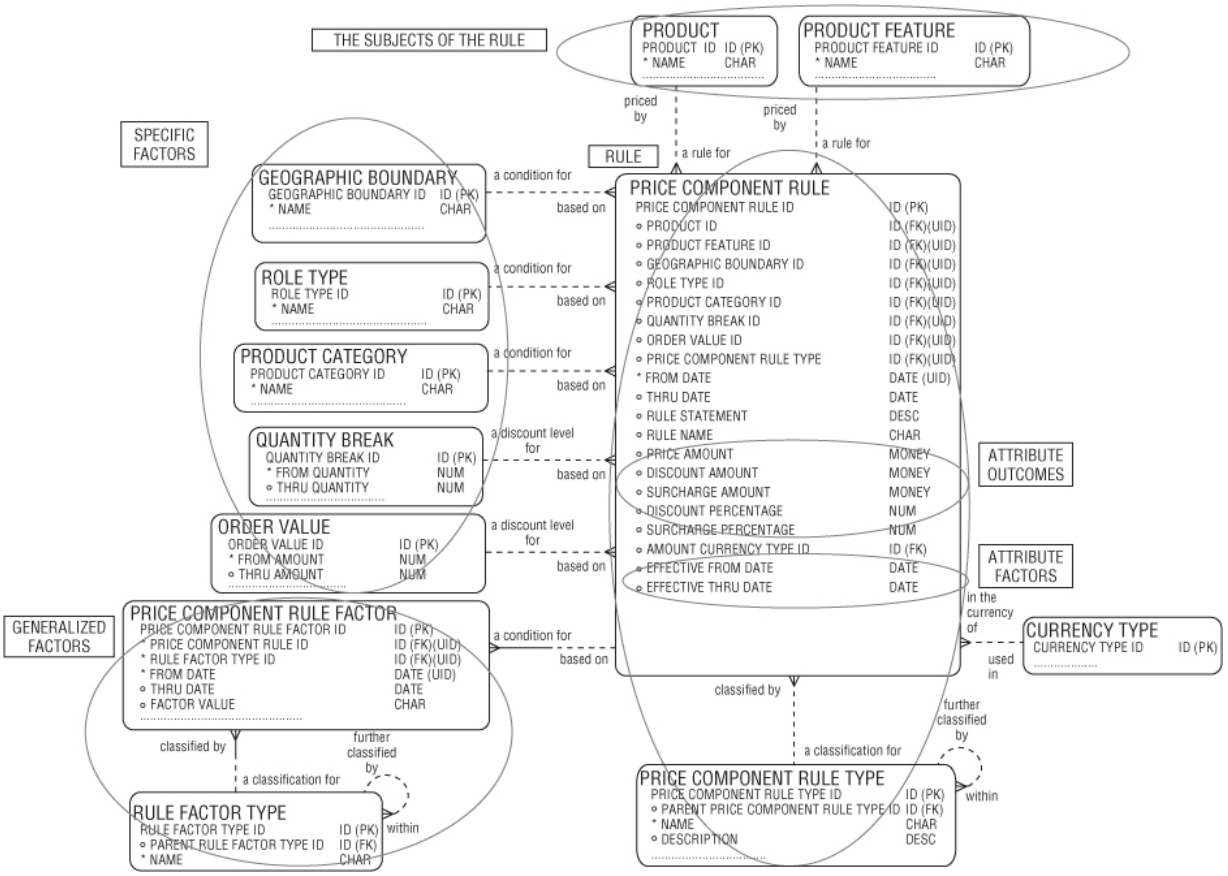


Figure 8.4 Example of using the Level 2 Business Rules Pattern, Pricing



Take a look at Figure 8.3. The data professional was asked to examine the business rules around different types of events such as getting telephone calls from complaining customers. As indicated earlier in the chapter, an event is defined as “an activity that happens at a given place and time and that may trigger other activities.” EVENT TYPE is therefore the subject of the EVENT TYPE RULE because there will be rules for different types of events that may occur.

The data professional quizzed the service managers on how employees should deal with each of the different types of events. He initially got a response saying that they handled each different EVENT TYPE depending on what COMMUNICATION EVENT TYPE occurred. The data professional knew that COMMUNICATION EVENT TYPE(s) (“Email,” “Phone Call,” “Face to Face,” and so on) already existed in the enterprise data model, and so decided to reuse these values as specific factors for the EVENT TYPE RULE. Hence, “each EVENT TYPE RULE *may be* **based on** one and only one COMMUNICATION EVENT TYPE.” For example, a complaint from a customer is an event, and how one should normally respond to an event might depend on the type of communication event received (“Phone call,” “Fax,” or so on) and on many other factors.

If a customer submits a complaint by phone or email and the complaint is valid, the customer care staff calls the customer with an apology. This is an example of an outcome that would be maintained in EVENT TYPE RULE OUTCOME that has an associated RULE OUTCOME TYPE of “Apology.” If there is a valid customer complaint that is given “Face to face,” the rule may be that the customer care staff apologizes in person and sends the customer a coupon with a credit that they can use on their next purchase (two EVENT RULE OUTCOME instances). Customer care may feel that a customer who complained “Face to face” needed

more care and attention than a customer who phoned in a complaint or who emailed in a complaint.

> **Note**
>
> In this example, we don't have any specific factors or outcomes that we capture as attributes. In the next example, which you can see in Figure 8.4 later in this chapter, there are attributes that capture both outcomes and factors. This illustrates that the business rules pattern can be tailored to meet different needs.

Table 8.1 provides example data for possible instances of Figure 8.3. In the first column of the table each instance of the rule is described. Keep in mind that **rule statement** is an optional attribute and that the enterprise may not choose to maintain this data because the other data maintained for the rule actually describes the rule by maintaining the parameters for the rule such as its factors and outcomes.

Table 8.1 Example of Using the Level 2 Business Rules Pattern

EVENT TYPE RULE. RULE STATEMENT	EVENT TYPE RULE TYPE. NAME	EVENT TYPE. NAME	COMMUNICATION EVENT TYPE. NAME (OF FACTOR)	RULE FACTOR TYPE. NAME	EVENT TYPE RULE FACTOR. FACTOR VALUE	RULE OUTCOME TYPE. NAME	COMMUNICATION EVENT TYPE. NAME (OF OUTCOME)
Phone call with an apology must be made when a customer calls on the phone with a valid complaint	Customer Service	Customer Complaint	Phone Call	Valid		Apology	Phone Call
Phone call with an explanation must be made when a customer calls on the phone with a complaint that is not valid	Customer Service	Customer Complaint	Phone Call	Not valid		Explanation	Phone Call
Phone call with an apology must be made when a customer emails a valid complaint	Customer Service	Customer Complaint	Email	Valid		Apology	Phone Call
Email with an explanation must be sent when a customer emails an invalid complaint	Customer Service	Customer Complaint	Email	Not valid		Explanation	Email

EVENT TYPE RULE. RULE STATEMENT	EVENT TYPE RULE TYPE. NAME	EVENT TYPE. NAME	COMMUNICATION EVENT TYPE. NAME (OF FACTOR)	RULE FACTOR TYPE. NAME	EVENT TYPE RULE FACTOR. FACTOR VALUE	RULE OUTCOME TYPE. NAME	COMMUNICATION EVENT TYPE. NAME (OF OUTCOME)
A personal meeting with an apology as well as a credit coupon should be given when a customer has a valid complaint that they communicate face to face.	Customer Service	Customer Complaint	Face to Face	Valid		Apology	Face to Face
						Credit Coupon (EVENT TYPE RULE OUTCOME **outcome value** of "50 (US Dollars)")	Package
Customers must be sent an acknowledgment letter and a gift by mail on their 10th customer anniversary	Customer Incentive	Customer Anniversary		Customer Anniversary Number	10	Acknowl-edgment and thank you.	Letter
						Pen set gift	Package

The first row of the table maintains an instance of EVENT TYPE RULE and has a **rule statement** of “Phone call with an apology must be made when a customer calls on the phone with a valid complaint.” This is classified as an EVENT TYPE RULE TYPE **name** of “Customer Service.” This is useful when someone wants to see all of their “Customer Service” rules, or all their “Pricing Rules” and so on.

The subject of this rule is EVENT TYPE, because the rule is about what to do when a certain type of event occurs. Thus, the EVENT TYPE **name** is a “Customer Complaint.” This instance of the rule is based on two factors, namely when a “Phone call” is received and when it is a “Valid” complaint as judged by the person answering the call. Thus, one of the factors is a specific factor because it is related to an existing entity and it is related to the instance COMMUNICATION EVENT TYPE **name** “Phone call.” Because there is not an existing entity to maintain the other factor that this rule instance refers to a “Valid” complaint, the generalized factor entities are used and the RULE FACTOR TYPE **name** of “Valid” is related to this rule. Thus, this rule is based on two factors, that it is a “Phone call” and also that the complaint is judged as being “Valid.” The outcome, or result of the rule, is that when these factors occur, there is a RULE OUTCOME TYPE name of “Apology,” which is communicated via a “Phone call.”

Note

There may be various ways of classifying rules. For example, the rule statement of “Phone call with an apology must be made when a customer calls on the phone with a valid complaint” may be classified as a “Customer Service” rule type as shown in Table 8.1, or it may be classified as a “Complaint” rule type that is within a classification of “Customer Service” rule which could be accommodated via the recursive relationship on EVENT TYPE RULE TYPE”. Another option is that it may be classified as a “Communications” rule type. Because some rules such as this

require more than one classification, we address a more flexible way to accommodate rule classifications in the level 3 pattern.

Note

It may seem that the “rule statement” is redundant because we are maintaining the various factors and outcomes and relating them to the rule entity. However, it is often helpful to state the rule in ‘plain English’ (or whatever language is appropriate) so that it is clear what the rule actually is. More sophisticated enterprises may derive the rule statement from the factors and outcomes. In the pattern we show the rule statement as optional because in some situations the enterprise may decide to maintain the rule statement and in other situations, for instance where there may be tens of thousands of rule entity instances such as for product pricing, the enterprise may elect not to maintain the rule statement.

The other rows in Table 8.1 show what the rules are under different conditions. For instance, the second row maintains the rule that if a customer has an EVENT TYPE **name** of “Customer Complaint” and they communicate it via the COMMUNICATION EVENT TYPE **name** of “Phone call” and the generalized factor of RULE TYPE FACTOR **name** is “Not valid” (meaning the complaint is judged as invalid), then the specified outcome for this situation is that an “Explanation” (RULE OUTCOME TYPE **name**) is communicated via a COMMUNICATION EVENT TYPE of “Phone call.”

Notice the fifth row shows that more than one outcome can occur. When the factors for the complaint are that it was communicated “Face to Face” and that it is a “Valid” complaint, there are two outcomes that the enterprise has specified: An “Apology” (RULE OUTCOME TYPE **name**) should be communicated “Face to Face” (COMMUNICATION EVENT TYPE **name**) and a “Credit Coupon” (RULE OUTCOME TYPE **name**) with an EVENT TYPE RULE OUTCOME **outcome value** of “50 (US Dollars)” is sent via a “Package” that is sent to the customer. (There could be a relationship to

another entity of CURRENCY TYPE to look up the value of “US Dollars.”)

The data modeler discovered numerous factors that were not related to existing entities in the data model and used the EVENT TYPE RULE FACTOR and RULE FACTOR TYPE entities (generalized factors) to model these data requirements. For instance, we just discussed that one type of factor that was not related to an existing entity was whether the complaint was “Valid” or “Invalid.” Another factor that was not contained in an existing entity was the number of years that a customer has been with the enterprise, or in other words, their “Customer Anniversary Number,” for example, if it is their 1st, 2nd, 3rd, or 10th anniversary. The last row in the table shows that when a customer has an anniversary (EVENT TYPE **name** of “Customer Anniversary”) and it is their 10th anniversary (RULE FACTOR TYPE **name** of “Customer Anniversary Number” and EVENT TYPE RULE FACTOR **factor value** of “10”), the RULE OUTCOME TYPE is that the customer receives an “Acknowledgment and thank you” that is communicated via letter (COMMUNICATION EVENT TYPE “Letter”) by the customer care staff from sales and marketing. This illustrates that a generalized rule factor may have a parameter that is applied to it, and you can use the **factor value** attribute to maintain any additional parameters to describe the outcome. Another outcome for when a customer has a 10th anniversary is that they receive a “Pen set gift” that is sent to them via a “Package” (COMMUNICATION EVENT TYPE **name**).

In this example, there could have been other additional generalized factors or other factors may come up over time. For example, the enterprise may decide later that how it handles complaints is dependent on the nature of the complaint. If someone has a complaint about a “Billing” issue, there may be a different response than there is to a

complaint of “Product did not work,” and these could be additional factors that would influence the rule outcome and they could be maintained as additional instances of RULE FACTOR TYPE **name**.

Using the recursive relationship around RULE FACTOR TYPE allows multiple levels of classification. The RULE FACTOR TYPE(s) of “Billing” and “Product did not work” may be classifications **within** another RULE FACTOR TYPE **name** instance of “Nature of the complaint.” Thus, various factors may be grouped together into other higher-level RULE FACTOR TYPE(s) such as “Complaint validity” (that is **further classified by** “Valid,” “Invalid”), “Severity” (that is **further classified by** “High,” “Low”), and so on. This allows a great number of additional business rules to be maintained that may depend on other types of factors that may emerge over time.

The same data professional also captured business rules about pricing. After meeting with the sales, marketing, and accounting heads, and using the business rules pattern, the data professional produced Figure 8.4. In Figure 8.4 you see a different example of how to implement the same pattern. In this example “each PRODUCT and/or PRODUCT FEATURE *may be* **priced by** one or more PRICE COMPONENT RULE(s).”(7) Most enterprises have different ways to price their product and product features (a product feature is a variation in the product such as color, size, and so on). In this example you see that Kantowitz Electronics uses several different factors to create its PRICE COMPONENT RULE(s). You see several entities in this model that are part of the enterprise data model for this enterprise and used as factors, for instance:

- GEOGRAPHIC BOUNDARY may be a postal area, city, state, country, continent, sales territory, or many other geographic areas. You can see in Table 8.2 that GEOGRAPHIC BOUNDARY contains “North America.”

- ROLE TYPE contains different classifications of roles a person or organization may play that affects the price of a product. In this example Kantowitz Electronics gave better prices to roles of “Partner.” Another example could be price breaks if someone plays the role of “Employee.”
- PRODUCT CATEGORY(s) allows pricing to be associated to different product classifications, such as product families, product lines, or product types. For example, there could be a discount for a certain product line, like fur coats, during a specific time period, such as the summer.
- QUANTITY BREAK allows special pricing levels based on the total number of items that are ordered. For example, if 1000 units are ordered, this could be a factor in a reduction in the price of a product.
- ORDER VALUE contains a range from the low end to the high end of a monetary amount for an order, which when met, would be a factor that could influence the price. For example, if an order was between $500,000 and $999,999, a price reduction might be applied. This is different from the monetary amount of an ORDER, which would be an attribute of ORDER (or derived from ORDER ITEM amounts), because the ORDER VALUE represents a range in values and is used just as a way to specify this factor. Alternatively, ORDER VALUE could be captured as a generalized factor in PRICE COMPONENT RULE FACTOR if it was not captured in your data model.
- Other generalized factors are maintained in PRICE COMPONENT RULE FACTOR. For example, how far the customer's delivery location is from the distribution center or what the customer's average number of days taken to pay for past orders is. This type of data is not maintained in the data model (other than for this rule)

and, therefore, is not tied to a specific entity that is already in the model.

Note

Why is this called PRICE COMPONENT RULE rather than PRODUCT RULE? As we stated earlier, the name of the ENTITY RULE (as seen in Figure 8.2) may not always have the name of the subject(s) of the rule in it. Many times an entity such as PRODUCT will have many business rule entities, and therefore, it is necessary to have more specific business names for each type of business rule. Another alternative is to generalize all the rules about PRODUCT so that it includes PRICE COMPONENT and other product rules such as PRODUCT REPLENISHMENT RULE (that determines how and when to replenish inventory of that product). The business rule pattern supporting PRODUCT RULE would have many of the same factors and look very similar in structure to PRICE COMPONENT RULE, although it would be more generalized and more encompassing since it would need to incorporate the factors, outcomes, and classifications for all types of PRODUCT rules.

Table 8.2 Example of Using the Level 2 Business Rules Pattern, Pricing

PRICE COMPONENT RULE .RULE NAME	PRICE COMPONENT RULE. RULE STATEMENT	PRODUCT. NAME/ PRODUCT FEATURE. NAME	GEO BOUNDARY, ROLE TYPE, PRODUCT CATEGORY, QUANTITY BREAK, ORDER VALUE FROM AMOUNT/ THRU AMOUNT, PRICE COMPONENT RULE. EFFECTIVE FROM DATE/EFFECTIVE THRU DATE	RULE FACTOR TYPE. NAME	PRICE COMPONENT RULE FACTOR. FACTOR VALUE	PRICE COMPONENT RULE. PRICE AMOUNT, DISCOUNT AMOUNT, SURCHARGE AMOUNT (CURRENCY TYPE.NAME)	PRICE COMPONENT RULE. DISCOUNT PERCENTAGE, SURCHARGE PERCENTAGE	PRICE COMPONENT RULE TYPE. NAME
		A23 Widget	Geographic Boundary of "North America" Valid From Date of "Jan 1, 2009"			Price amount of "10" Discount amount of "1" Surcharge amount of "0" (US Dollars)	Discount percentage of "0" Surcharge percentage of "0"	Net Price
		A23 Widget	Geographic Boundary of "Europe" Valid From Date of "Jan 1, 2009" Valid Thru Date of "June 30, 2009"			Price amount of "12" Discount amount of "0" Surcharge amount "0" (Euros)	Percentage discount of "0" Surcharge percentage of "8"	Net Price
		A23 Widget Accessory #1	Geographic Boundary of "Europe" Quantity Break. From quantity "100" and Thru Quantity "500"				Discount percentage of "15"	Discount
Summer Saver Program			Product Category of "Hardware" Valid From Date of "June 1, 2009" Valid Thru Date of "Sept 30, 2009"			Discount amount of "10" (US Dollars)		Discount

PRICE COMPONENT RULE .RULE NAME	PRICE COMPONENT RULE. RULE STATEMENT	PRODUCT. NAME/ PRODUCT FEATURE. NAME	GEO BOUNDARY, ROLE TYPE, PRODUCT CATEGORY, QUANTITY BREAK, ORDER VALUE FROM AMOUNT/ THRU AMOUNT, PRICE COMPONENT RULE. EFFECTIVE FROM DATE/EFFECTIVE THRU DATE	RULE FACTOR TYPE. NAME	PRICE COMPONENT RULE FACTOR. FACTOR VALUE	PRICE COMPONENT RULE. PRICE AMOUNT, DISCOUNT AMOUNT, SURCHARGE AMOUNT (CURRENCY TYPE.NAME)	PRICE COMPONENT RULE. DISCOUNT PERCENTAGE, SURCHARGE PERCENTAGE	PRICE COMPONENT RULE TYPE. NAME
Pricing Rule 124	A 25 percent price reduction only if **all** of the following are true: 1. The customer is in North America. 2. The customer is a partner. 3. The Product Category is Hardware or Accessory. 4. Over 1000 items are purchased. 5. The Order Value is $1,000,000 or greater. 6. All delivery locations for customer are within 500 miles of the closest distribution center. 7. Price is effective anytime from August through September of 2009.		Geographic Boundary is "North America" Role Type is "Partner" Product Category is "Hardware" and "Accessory" Quantity Break From Quantity "1000" items purchased Order Value is "$1,000,000–$10,000,000" Valid From Date is "Aug. 1st, 2009"; Valid Thru Date is "Sept. 30th, 2009"	Maximum distance of miles that furthest delivery location is from distribution center	"500"		Discount percentage of "25"	Discount

In Figure 8.4, you see that the outcomes of PRICE COMPONENT RULE(s) are captured as attributes, that is, **price amount**, **discount amount**, **surcharge amount**, **discount percentage**, and **surcharge percentage**.

- The **price amount** maintains either a price for a product or product feature, after having applied any discounts or surcharges.
- The **discount amount** maintains a flat amount of a discount that can be applied to reduce the price.
- The **surcharge amount** maintains a flat amount for a surcharge that can be applied to increase a price. The CURRENCY TYPE allows for the price, discount, and surcharge amounts to be for “US Dollars,” “Euros,” and so on.
- The **discount percentage** and **surcharge percentage** maintain some adjustment to a **price amount** as a portion of the **price amount**. The value of the amounts and/or percentages is based on the combination of factors that are related to that instance.

For example, as shown in Table 8.2, the **price amount** is “10” (US Dollars), **discount amount** is “1” (US Dollars), and **surcharge amount** is “0” (US Dollars) for the product of “A23 Widget” and is based on the combination of factors that the geographic boundary is “North America” and that the date ordered is after “Jan. 1, 2009.”

The PRICE COMPONENT RULE TYPE **name** that is the last column in Figure 8.4 and Table 8.2 classifies the rule instance. The first two instances are classified as “Net Price” because the rule is regarding what price, discount, and surcharge to charge based upon different factors. The last three instances are classified as “Discount” because the rules describe only a discount rule, and the outcome of the rule is only a **discount percentage** or a **discount amount**.

Note

For a more in-depth look at pricing, please see Chapter 3 of *The Data Model Resource Book*, Revised Edition, Volume 1 (Wiley, 2001).

Notice that in this pattern all outcomes and some factors are maintained as attributes. Because there are not several prices or percentages for the same type of attribute in any one specific instance and these are atomic pieces of data whose values are generally not re-used, we do not capture the amount or percentage in a separate entity as PRICE COMPONENT RULE OUTCOME(s), as seen in the pattern in Figure 8.2. However, you could make the argument that because the amount also has a currency, the **amount** and **amount currency type id** could be broken out into its own rule outcome entity. Also note that some factors are captured as attributes, that is, **effective from date** and **effective thru date** (which are examples of **factor attribute 1** and **factor attribute 2** in the pattern in Figure 8.2). These denote the length of time that a particular PRICE COMPONENT RULE is effective. The first two rows in Table 8.2 show that the price is based on and dependent on certain dates. Another example shown in Table 8.2 is that the PRICE COMPONENT RULE that applies a discount amount of “10 (US Dollars)” might be part of a “Summer saver” program valid from June until September, 2009. Because business rules often have time frames showing when they are effective, these timeframe factors are often captured as attributes of the rule.

Note

The **effective from date** and **effective thru date** are very different from the **from date** and **thru date** attributes. The **from date** and **thru date** attributes are attributes that show the length of time an instance of PRICE COMPONENT RULE was

valid from and through, or in other words, when the instance first existed and through when the instance was valid. The **effective from date** and **effective thru date** show when an instance of PRICE COMPONENT RULE was effective or operational. For example, fur coats may have a discount rule applied to them that is operational for the summer (the **effective from date** and **effective thru date**), but the discount rule itself may exist, and be a valid instance, months before the summer and for many years afterwards (the **from date** and **thru date**).

You can also see that PRICE COMPONENT RULE FACTOR may contain other factors that are not part of the enterprise data model for Kantowitz Electronics. In the last row of Table 8.2 you see a RULE FACTOR TYPE **name** of "Maximum distance of miles that furthest delivery location is from distribution center," which represents an instance of an additional factor that is present for the rule. This factor is an example of a factor that is not directly related to a value from any existing entity as was the case with the other factors. This factor says that the PRICE COMPONENT may also be valid only for the condition that deliveries cannot be too far away from the distribution center. Some RULE FACTOR TYPE(s) require a value that corresponds to the factor, and in this case, the value that needs to be specified is the number of miles that the furthest delivery location needs to be from the distribution center. For example, you see the PRICE COMPONENT RULE FACTOR **factor value** is "500," showing that all delivery locations must be within 500 miles of the distribution center to get the discount. This allows any number of other factors to be maintained and then applied to the business rule.

Some PRICE COMPONENT RULE instances may contain the actual **rule statement**. The last row of Table 8.2 shows such an instance. It has the **rule name** "Pricing Rule 124," and this particular business rule states (in the **rule statement**) "A 25 percent price reduction only if **all** of the following are true:

1. The customer is in North America.

2. The customer is a partner.

3. The Product Category is Hardware or Accessory.

4. Over 1000 items are purchased.

5. The Order Value is $1,000,000 or greater.

6. All delivery locations for customer are within 500 miles of the closest distribution center.

7. Price is effective anytime from August through September of 2009."

The rule specifies that if all of these conditions are met, a PRICE COMPONENT RULE **discount percentage** (an outcome) of "25" is applied.

You could have a great number of rule instances that are maintained in PRODUCT COMPONENT RULE because there may be amounts or percentages for any combination of factors that influence prices. For example, there could be a price (**price amount**) for product "A245," indicating you charge $9.99 (US Dollars) for this product in New York State, $10.99 (US Dollars) in California, £9.99 in the UK, and HK$99.99 in Hong Kong. You could adjust these prices by applying any specific or generalized factors. So you could say that the price is $8.99 in New York State for a "Partner" (ROLE TYPE) when the ORDER VALUE is over "$1,000 (US Dollars)." There may be a PRICE COMPONENT RULE that has a PRICE COMPONENT RULE TYPE of "Surcharge" and is for an amount of "$100.00" for any delivery location outside of the 500 mile delivery from distribution center limit (see point 6 of the **rule statement** example).

Note

Notice that in this example, not all instances of PRICE COMPONENT RULE relate to a PRODUCT or PRODUCT FEATURE. For example, Pricing rule 124 does not relate to a particular PRODUCT or PRODUCT FEATURE. There could be other discounts that apply just to a particular PRODUCT CATEGORY, ROLE TYPE, and/or a particular GEOGRAPHIC BOUNDARY without a specific

PRODUCT or PRODUCT FEATURE involved. So in these instances, does the PRODUCT CATEGORY, ROLE TYPE, or GEOGRAPHIC BOUNDARY become the subject? From one point of view, the answer is yes. However, another way to view this pattern is by seeing all the subjects as potential factors, and all of the factors as potential subjects. Thus in this example, PRODUCT and PRODUCT FEATURE may be viewed as factors that influence the outcome just like any of the other factors that influence the outcome.

When Should This Pattern Be Used?

We use this model:

- **When an enterprise is interested in capturing how business rules for specific subject areas are integrated to the other entities in their data models (or enterprise data models):** Often, business rules are interrelated with the rest of the data model. It may be awkward to say that product pricing represents business rules data, and therefore, you exclude it from the data model because it is covered in the metadata repository. In the first example, you saw in Figure 8.3 that EVENT TYPE RULE(s) affect what happens for EVENT TYPE(s), such as a “Customer Complaint” or a “Customer Anniversary,” and that COMMUNICATION EVENT TYPE(s) such as “Phone Call” or “Email” were factors in the possible outcomes of “Apology” or “Explanation.” This knowledge can be important to many businesses. For example, if a business adds a new EVENT TYPE or COMMUNICATION EVENT TYPE, it can maintain the associated business rules and integrate them within the context of the data model.
- **There is a need to have a flexible data model structure capturing rules:** The pattern accommodates a great number of possible outcomes based upon various combinations of factors, as well as additional types of factors that may emerge for the rule.

The pattern uses the ENTITY RULE FACTOR and RULE FACTOR TYPE entities to allow any number of additional generalized factors that could be applied. For example, in Figure 8.3 you could add additional rules depending on additional factors such as the "Nature of the complaint" or the "Severity" of the complaint.

- **The enterprise does not have a metadata environment or business rules engine to capture rules:** These patterns can provide a place holder for capturing rules, factors, and outcomes. We have implemented these patterns at many companies that have used them to directly capture business rules and integrate directly into their databases.
- **The enterprise wishes to capture additional business rule factors that are not directly tied to entities that exist in the data model, or when there is a need to accommodate future types of factors that may emerge:** This is why we also capture other factors in ENTITY RULE FACTOR(s) and RULE FACTOR TYPE(s). In the second example as seen in Figure 8.4 and Table 8.2 you saw a RULE FACTOR TYPE of "Maximum number of miles that furthest customer location is from distribution center" that captures a factor that is not directly maintained in an existing entity.
- **As an intermediary solution toward a business rules engine or metadata repository environment:** If the enterprise (Kantowitz Electronics) is creating a centralized rules engine or metadata environment, this pattern may be a useful stopgap that allows people to create a rules-driven environment without having to wait for the metadata environment to be implemented. It can then be easily integrated (as a source or target for business rules) into the metadata environment when it does get implemented.

- **There are situations where the data professional needs to show a statement of scope to other IT professionals:** The pattern shows the breadth of the area of where the business rules are applicable. Each new type of business rule may require a different set of factors, outcomes, and so on. It is important to show these differences.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **Many data modelers and nearly all nontechnical people will probably have difficulties understanding this pattern:** This pattern is for data professionals who have a good understanding of business rules and who are looking for an alternative way to capture and implement business rules as part of a data model and database, respectively.
- **If a new factor is discovered, it is often easy for a data professional just to add that factor as a RULE FACTOR TYPE and not try to discover if the enterprise data model captures this factor as an entity in its own right:** It is easy for data modelers to be ‘lazy’ and not integrate the factors as relationships from existing entities. It is crucial to do the due diligence on each and every factor that affects a business rule to see how it integrates into the existing data models.
- **There may need to be a more flexible way to classify the rules:** This pattern used the Level 2 Classification Pattern, and if the rules have many different types of classifications, then you may need to use the Level 3 Classification Pattern. This could be viewed as a weakness in the classification pattern as

opposed to a weakness of the business rules pattern. Because, you can substitute a more specific or more generalized version of the classification pattern within the business rules pattern to suit your specific needs.

- **There are many types of data that are common to various types of business rules and thus could be consolidated into a BUSINESS RULE super-type:** We discuss this option in the next section.

Synopsis

The Level 2 Business Rules Pattern allows a data professional to capture many types of business rules as part of a data model. The pattern can be broken into three different major components: rules, factors, and outcomes, as you saw in Figure 8.2.

In the rules component of the pattern, you maintain the **rule name**, **rule statement**, classification of the business rule, the subject(s) of the rule, factors, and outcomes. You may also maintain other attributes that enhance the definition of the rules such as **rule source**, **rule note**, **rule reference**, and so on.

The factors are defined as "the conditions that affect the outcome of the rule." Factors may be maintained as attributes of the rule entity, relationships to existing entities, or generalized factors that accommodate factor instances that are not currently maintained in existing entities. For example in Figure 8.4, the PRICE COMPONENT RULE captured the factor attributes of **effective from date** and **effective thru date** showing the range of time when an outcome such as a **price amount** or **discount percentage** was in effect. Thus, they may be used to indicate a summer discount or a winter special.

Factors may be related to existing entities, such as a factor that has to do with the existing entity of GEOGRAPHIC BOUNDARY, and thus, you can reuse values that are already

maintained in this entity. For example, you can specify a rule that there is a 2 percent discount for orders from the GEOGRAPHIC BOUNDARY of “South America.”

Also, you may capture factors that do not relate to existing entities in the RULE FACTOR TYPE and ENTITY RULE FACTOR entities. These are factors that have significance within the context of specific business rules, but are not explicitly captured via values for entities or attributes in their own right. Data professionals should not get ‘lazy’ and capture all factors as FACTOR TYPE(s). We recommend you take care in understanding each of the factors that affect a rule.

These factors may evolve over time as your enterprise expands. So as factors and other data changes in the data model, this evolution should be reflected in the business rules. Hence, capturing business rules in the data model means the business rules should evolve as the rest of your data model evolves (and vice versa). Also, by integrating your business rules into your data model you can assess the impact of changes to your data model. For example, what happens if you remove GEOGRAPHIC BOUNDARY from your data models? What rules will this affect?

In the outcomes component, this pattern maintains the different possible results of the business rule. You need to understand and capture the outcomes of a business rule because this outcome may be used to affect what should be done in different situations, or in other words, based upon different factors. Similar to factors, the pattern can capture outcomes in three ways: as attributes of the business rule, as entities that exist already in your data model, or as instances of the generalized ENTITY RULE OUTCOME.

This pattern is a more complex pattern that may be of use to more advanced data professionals who are trying to solve business rules issues within the context of integrating it into their data modeling efforts. This pattern is not meant to take the place of a metadata solution or business rules engine in

an enterprise, but we have found that many enterprises have not implemented a full metadata solution or business rules engine solution and/or don't intend on doing so. This pattern may help these enterprises understand and integrate their business rules into their data models (and databases). Even if an enterprise is creating a metadata solution (or centralized business rules engine), you can apply this pattern as a very useful way to model business rules because many business rules, such as product pricing, are so integral to data modeling efforts. We have found that data modelers will often model rules without even knowing they are rules. Thus, if you can recognize that this is an 'if/then' situation, you can use this pattern to effectively and consistently model this type of data requirement.

Level 3 Business Rules Pattern

In Figures 8.3 and 8.4 of the previous section you saw two examples of the Level 2 Business Rules Pattern. In these figures you saw that each of the 'rule' entities (EVENT TYPE RULE and PRICE COMPONENT RULE) had their own distinct entities to maintain the rules, factors, outcomes, and classifications. However, many of these data model structures had similarities and there are great advantages to maintaining rules in a single consolidated model. The essence of the Level 3 Business Rules Pattern is to maintain all types of rules across an enterprise, or rules for a certain domain area within the enterprise, using the same data model.

Why Do We Need This Pattern?

You can imagine that by using the Level 2 Business Rules Pattern, there may be a great number of business rules entities throughout the data model. The Level 3 Business Rules Pattern addresses the need for an enterprise to

capture many types of business rules within the same data model using a very flexible format to accommodate many types of business rules and changes to business rules that may occur over time. This may be needed in order to capture the business rules across the entire enterprise or for a certain domain such as all the rules for transactions or the various business rules for parties. When capturing these rules, the enterprise may need to capture the common aspects of different business rules, such as the business rule statement, the generalized factors, classifications, and outcomes, and also capture the specific aspects for a type of rule, such as the different specific factors that affect that type of rule. This pattern provides a consistent and flexible way to do this.

Note

We find this pattern particularly useful when capturing all of the business rules for a specific domain. Many of the business rules in a domain use the same factors. For example, in an investment bank, trading environment you can capture all LIMIT RULE(s), MARKET RISK RULE(s), and CREDIT RISK RULE(s) for TRADE(s) by using this common structure. It gives a good indication of how the rules may be related and what different factors commonly affect different rules in the same domain.

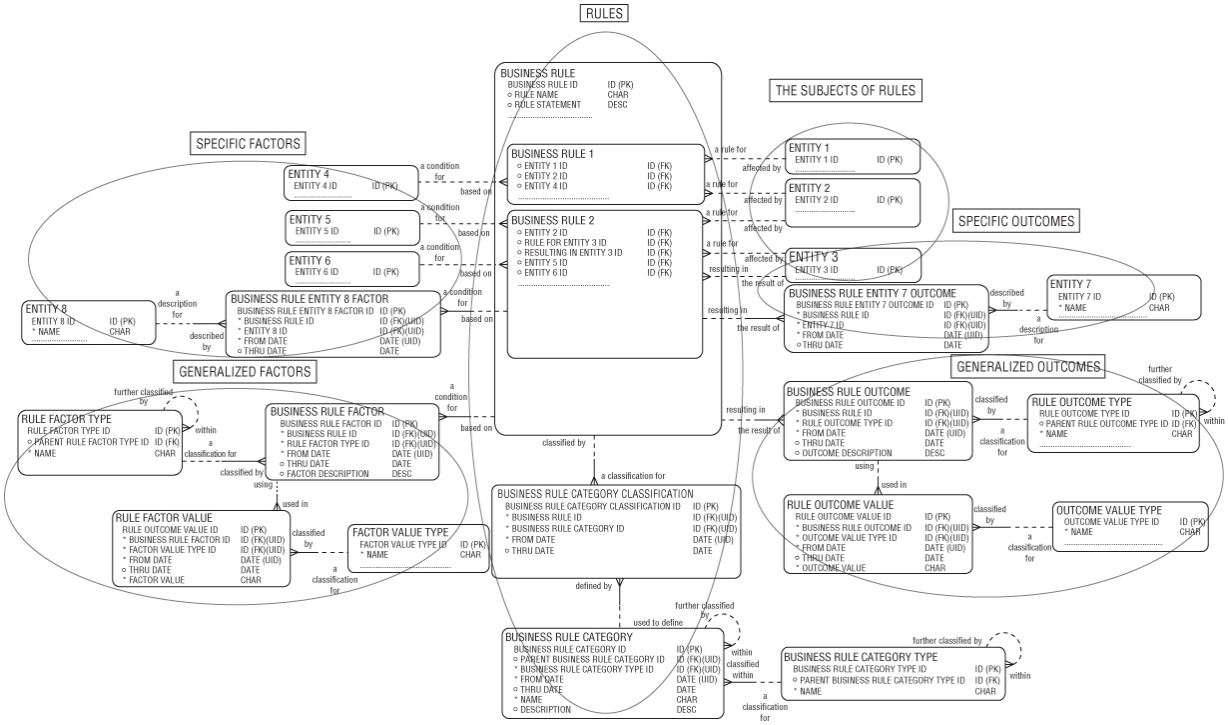
How Does This Pattern Work?

As you saw in the last section, most rules have numerous common parts that need to be modeled: the name and statement of the rule, the classification of the business rule, the specific and generalized factors, and the outcome(s) of that rule. In this pattern we model these parts in a more generalized fashion to allow more flexibility and to be able to maintain many types of rules in the same data model. Most rules also have specific entities or attributes that are applicable only to a certain type of rule. For example, the PRICE COMPONENT RULE is based on ORDER VALUE, and

this may be a relationship that is very specific to this type of rule. In this pattern we provide a method to model both the specific aspects of a particular business rule as well as the common aspects that apply to all business rules.

In Figure 8.5 you see a supertype called BUSINESS RULE. This BUSINESS RULE entity supports the data requirements of all the types of rules that may apply across the enterprise or across a domain within an enterprise. There could be any number of subjects for these rules and this pattern illustrates subjects, such as ENTITY 1, ENTITY 2, and ENTITY 3. BUSINESS RULE 1 and BUSINESS RULE 2 are subtypes of BUSINESS RULE that correspond to rules about ENTITY 1, ENTITY 2, and ENTITY 3. These could, for example, be the entities shown in the level 2 pattern, namely, EVENT TYPE RULE, which is a rule for EVENT TYPE(s), or PRICE COMPONENT RULE, which is a rule for PRODUCT(s) and PRODUCT FEATURE(s). There could also be any number of BUSINESS RULE subtypes, for example, ORDER RULE, WORK EFFORT RULE, SHIPMENT RULE, PARTY RULE, PRODUCT REPLENISHMENT RULE, PRODUCT REGULATIONS RULE, SALES RESTRICTION RULE, and so on. Each of these subtypes may be related to various core business entities in the data model.

Figure 8.5 Level 3 Business Rules Pattern



In Figure 8.5 you also see that ENTITY 2 is a subject for both BUSINESS RULE 1 and BUSINESS RULE 2 subtypes to illustrate that an entity to be related to more than one rule. For example a PRODUCT or PRODUCT FEATURE may be related to PRICE COMPONENT RULE(s) and SALE RESTRICTION RULE(s). In other words, a PRODUCT may have rules that dictate its price and rules that dictate to what countries and/or to what age groups you can sell a product.

In Figure 8.5 you also see that ENTITY 3 is related to BUSINESS RULE 2 in two different ways. First, it is the subject of the rule. It can also be a specific outcome of a rule. For example, a project, a WORK EFFORT TYPE, may be subject of a rule for auditing the total spend of all projects. If a project goes over budget then the outcome of the rule may be to set up an “Audit project,” another WORK EFFORT TYPE, that investigates why a project is overspending. In fact we can imagine that an entity could be the subject of a rule, a factor in a rule, and also a specific outcome for a rule.

Note

This pattern may include many types of business rules that correspond to many entity subjects. This may be very useful depending on the size and complexity of your enterprise, but it can get very large with many complex relationships between factors, outcomes, and business rule entity subtypes. To keep this more manageable, we sometimes use this data model structure for each major domain or area of the model. For example, we could have business rules entities for PARTY RULE, PRODUCT RULE, and TRANSACTION RULE. The PARTY RULE may cover all the rules for parties, party roles, party relationships, and all other party-oriented rules, the PRODUCT RULE may cover any rule that relates to the product subject data area. The TRANSACTION RULE may cover all rules about orders, shipment, invoices, payments, and all other types of transactions. This approach can help balance having a more specific style for this level 3 pattern while reducing the number of entities to manage in the model (as compared with the Level 2 Business Rules Pattern). However, when a more generalized style of modeling is preferred, the enterprise can use this pattern to maintain all of the rules within an enterprise.

BUSINESS RULE contains all of the common attributes and relationships that business rules may have. For example, some attributes of BUSINESS RULE may be **rule name**, capturing a designation for the rule such as “Pricing rule 124,” and **rule statement**, capturing the complete business rule in plain language such as “Apology must be sent when a customer has a valid complaint.” Inside BUSINESS RULE you see two subtypes (BUSINESS RULE 1 and BUSINESS RULE 2) that represent specific types of business rules that you wish to capture. These subtypes may have their own specific relationships and attributes. For example, if the ORDER and SHIPMENT were the subjects that corresponded to ENTITY 1 and ENTITY 2, then BUSINESS RULE may have subtypes of ORDER RULE and SHIPMENT RULE.

Note

We have found through experience that usually, the data that is specific to the subtypes of BUSINESS RULE are the relationships to entities that are its specific factors, specific outcomes, and the entities that are the subjects of the rule (such as PRODUCT or PRODUCT FEATURE in Figure 8.4). However, this is not always the case. Some BUSINESS RULE subtypes may also have their own attributes, in particular their own factor attributes. One example would be a **quantity attribute** for a one-time UTILIZATION CHARGE RULE (this shows how many times the usage occurred, for example, **number of text messages**, to know the price for this level of usage), which could be a subtype of PRICE COMPONENT RULE. This also illustrates that there could be subtypes of subtypes within BUSINESS RULE. See Chapter 3, Figure 3.7, of *The Data Model Resource Book*, Volume 1, Revised Edition.

On the left-hand side of Figure 8.5 you see two different types of factors: "Specific Factors" and "Generalized Factors." The specific factors, on the top left of the diagram, are related to entities that already exist in your data models. As we already stated, it is important to capture how specific existing entities are related to the business rules so that you do not redundantly maintain data that is already in the data model. For example, if a factor is related to a geographic boundary, "New York," you should relate the business rule to data that is already in the GEOGRAPHIC BOUNDARY instance of "New York" instead of redundantly re-creating this data in another entity (e.g., in a RULE FACTOR TYPE). Thus, specific factors are related to existing entities, and Figure 8.5 shows numerous examples of this where "each ENTITY 4 *may be* **a condition for** one or more BUSINESS RULE 1." Another example would be "each BUSINESS RULE 2 *may be* **based on** one and only one ENTITY 5 and/or *may be* **based on** one and only one ENTITY 6."

The relationship from BUSINESS RULE 2 to BUSINESS RULE ENTITY 8 FACTOR shows how this pattern can support a

many-to-many relationship to a specific factor. For example, suppose that a price component rule was based on two instances of the same specific factor such as a discount percentage that is given to a party that is either a “Partner” or a “Supplier” (the rule was based on two instances of ROLE TYPE). If BUSINESS RULE 2 was PRICE COMPONENT RULE and ENTITY 8 was ROLE TYPE, this pattern could support this condition via a PRICE COMPONENT RULE being **based on** two instances of a BUSINESS RULE PRICE COMPONENT FACTOR associative entity for a ROLE TYPE of “Partner” and another ROLE TYPE of “Supplier.”

On the bottom left, you see the “Generalized Factors.” These are the factors that are not related to existing entities in the data model and are needed to maintain other types of factors and possibly values for the rule. They are captured as instances of BUSINESS RULE FACTOR, RULE FACTOR TYPE, RULE FACTOR VALUE, and FACTOR VALUE TYPE. Notice that BUSINESS RULE FACTOR is related to the BUSINESS RULE supertype and not the specific subtypes of BUSINESS RULE, because this is a generalized structure to maintain factors for any type of business rule. The model states, “each BUSINESS RULE *may be* **based on** one or more BUSINESS RULE FACTOR(s)” and “each BUSINESS RULE FACTOR *must be* **a condition for** one and only one BUSINESS RULE.” These BUSINESS RULE FACTOR(s) are **classified by** RULE FACTOR TYPE(s), and they *may be* **using** one or more RULE FACTOR VALUE(s), each of which is classified by a FACTOR VALUE TYPE. For example, an instance of a RULE FACTOR TYPE may be the length of time a party has been a customer of an enterprise (“Number of years that party has been a customer”). This is an example of a generalized factor that may not be maintained (or needed) somewhere else in the data model, and thus, you can maintain it here. This BUSINESS RULE FACTOR may have an associated RULE FACTOR VALUE **factor value** of

“10,” signifying that the rule applies if the customer has been a customer for 10 years. There may be several values (RULE FACTOR VALUE(s)) that are applicable for a rule factor (BUSINESS RULE FACTOR). For example, you may need to influence the price based on a factor of whether the party has been a customer for between 5 and 10 years. For this factor, you can maintain a RULE FACTOR TYPE **name** instance of “Number of years that party has been a customer” with two values: one RULE FACTOR VALUE **factor value** of “5” with a FACTOR VALUE TYPE **name** of “Minimum number of years” and another RULE FACTOR VALUE **factor value** of “10” with a FACTOR VALUE TYPE of “Maximum number of years.”

Note

In our experience, many generalized factors have a single value. Also, some factors that have multiple values can be managed by specifying multiple factor instances. For example, in the preceding example, we could have had two RULE FACTOR TYPE(s) of “Minimum number of years as a customer” and “Maximum number of years as a customer.” Thus, an alternative data model structure is to have a **factor value** attribute in the BUSINESS RULE FACTOR entity (as we did in Figure 8.2), instead of maintaining the entity of RULE FACTOR VALUE. However, the way that we have modeled generalized factors in Figure 8.5 provides much more flexibility to accommodate many different scenarios.

Note

An issue with having generalized RULE FACTOR TYPE(s) and BUSINESS RULE FACTOR(s) is that not all of the different RULE FACTOR TYPE(s) are applicable to all BUSINESS RULE(s). In this pattern you are juggling the flexibility that generalized RULE FACTOR TYPE(s) and BUSINESS RULE FACTOR(s) provides versus how specific you need to be when associating RULE FACTOR TYPE(s) with BUSINESS RULE(s). For example, the RULE FACTOR TYPE “Number of years that party has been a customer” may have nothing to do with SHIPPING RULE(s) or BILLING RULE(s). At one enterprise we worked at, two IT business rules were

enacted that stated, “All specific rule factors must be using entities that are in the data model in their own right” and “All generalized rule factors must affect ALL business rules.” This resulted in a lot of additional entities that were just used to capture rule factors and a very sparse RULE FACTOR TYPE and BUSINESS RULE FACTOR entity! Not necessarily the best solution.

Similar to factors, on the right-hand side of the diagram you see two types of outcomes: “Specific Outcomes” and “Generalized Outcomes.” Just as factors may apply to specific entities that already exist in the data model, there may be outcomes that are related to data that is already in existing entities in the model. For example, a common outcome is that a certain type of work effort should be set up under certain conditions. So there could be a business rule that states that when there is a very significant complaint, a work effort of WORK EFFORT TYPE “Manage the complaint” should be set up and tracked to help ensure that the complaint is handled through resolution. In this case, the outcome of a rule (such as an EVENT TYPE RULE) may be that there is a specific outcome, which is maintained by relating the rule to the existing data model entity of WORK EFFORT TYPE.

Similar to the idea of specific factors having a many-to-many relationship with the rule, specific outcomes may also have a many-to-many relationship with the rule. For instance, when there is a very significant complaint, there may be a business rule that has an outcome with multiple WORK EFFORT TYPE(s) that should be created, for instance, “Manage the complaint” and “Assess the process (that led to the complaint).” The BUSINESS RULE ENTITY 7 OUTCOME associative entity illustrates a data model structure that accommodates a rule to have more than one of the same type of specific outcome.

On the bottom-right side of Figure 8.5, business rules generalized outcomes are maintained with the entities of

BUSINESS RULE OUTCOME, RULE OUTCOME TYPE, RULE OUTCOME VALUE, and OUTCOME VALUE TYPE. This mirrors the data model structure of the generalized factors and allows maintaining different results of business rules, each of which may have multiple values associated with them. Thus, BUSINESS RULE is directly related to the BUSINESS RULE OUTCOME that allows the flexibility to add any number of generalized outcomes, that is, "each BUSINESS RULE *may be* **resulting in** one or more BUSINESS RULE OUTCOME(s)." It's important to emphasize that the outcome of a rule only says what *should* happen and not what *actually* happened. Thus, just stating or describing a rule such as that an "Apology" should happen for a valid complaint, does not necessarily guarantee that this actually happened. Said another way, the purpose of maintaining the rule is just to state what should occur under various circumstances and not what happened or necessarily will happen.

Again, similar to factors, for each rule instance, there may be many outcomes, and each outcome may have many values. For example, as shown in the second rule instance in Table 8.3, there may be a business rule stating that when the customer celebrates their 10th anniversary (a factor), there are two outcomes: an acknowledgment/thank you letter and a gift. For the gift outcome, there may be two values for the gift, such as receiving "2" Pen sets and a discount coupon for "50" US Dollars. Thus, there would be a BUSINESS RULE OUTCOME that is **classified by** a RULE OUTCOME TYPE **name** of "Acknowledgment and thank you" and another BUSINESS RULE OUTCOME that is **classified by** a RULE OUTCOME TYPE of "Gift." For the BUSINESS RULE OUTCOME related to "Gift," there could be two RULE OUTCOME VALUE(s), one that has a RULE OUTCOME VALUE **outcome value** of "2" of OUTCOME VALUE TYPE "Pen sets" and another RULE OUTCOME VALUE **outcome value** of "50" for OUTCOME VALUE TYPE "US Dollar coupon discount." With

this structure, if the parameters of the outcomes change over time, the enterprise can keep the same type of outcome, but can modify the values of the outcomes (changing the amount of the coupon or the type of gift).

Table 8.3 Example of Using the Level 3 Business Rules Pattern, Pricing

BUSINESS RULE SUBTYPE	BUSINESS RULE. RULE	BUSINESS RULE. RULE STATEMENT	PRODUCT. NAME	GEO BOUNDARY, ROLE TYPE PRODUCT CATEGORY, QUANTITY BREAK, ORDER VALUE, EFFECTIVE FROM DATE/EFFECTIVE THRU DATE	EVENT TYPE. NAME	COMMUNI-CATION EVENT TYPE. NAME (FACTOR)	RULE FACTOR TYPE. NAME	RULE FACTOR VALUE. FACTOR VALUE (FACTOR VALUE TYPE. NAME)	RULE OUTCOME TYPE. NAME (VIA COMMUNI-CATION EVENT TYPE. NAME)	RULE OUT-COME VALUE. OUTCOME VALUE (OUTCOME VALUE TYPE) NAME)	WORK EFFORT TYPE. NAME
EVENT TYPE RULE	Complaint Rule 100	Phone call with an apology must be made when a customer calls on the phone with a valid complaint and a work task must be set up to manage this			Customer Complaint	Phone Call	Valid		Apology (via phone call)		Manage Complaint
EVENT TYPE RULE	Incentive Rule 305	Customers must be sent an acknowl-edgment letter and a gift by mail on their 10th customer anniversary			Customer Anniversary		Customer Anniversary Number	10 (Number of years)	Acknowledg-ment and thank you (via letter)		Manage Anniver-sary Event
									Gift (via package)	2 (Pen sets) 50 (US Dollar coupon discount)	

BUSINESS RULE SUBTYPE	BUSINESS RULE. RULE	BUSINESS RULE. RULE STATEMENT	PRODUCT. NAME	GEO BOUNDARY, ROLE TYPE PRODUCT CATEGORY, QUANTITY BREAK, ORDER VALUE, EFFECTIVE FROM DATE/EFFECTIVE THRU DATE	EVENT TYPE. NAME	COMMUNI-CATION EVENT TYPE. NAME (FACTOR)	RULE FACTOR TYPE. NAME	RULE FACTOR VALUE. FACTOR VALUE (FACTOR VALUE TYPE. NAME)	RULE OUTCOME TYPE. NAME (VIA COMMUNI-CATION EVENT TYPE. NAME)	RULE OUT-COME VALUE. OUTCOME VALUE (FACTOR VALUE TYPE) NAME)	WORK EFFORT TYPE. NAME
PRICE COMPO-NENT RULE			A23 Widget	Geographic Boundary of "North America"			Effective From Date	Jan 1, 2009 (Effective Date)	Price	10 (US Dollars)	
PRICE COMPO-NENT RULE	Pricing Rule 125	A 25 percent price reduction only if **all** of the following are true: 1. The customer is in North America 2. The customer is a Partner 3. The Product Category is "Hardware" 4. Over 1000 items are purchased 5. Customer pays within an average of 60 - 90 days 6. The date is later than October 1st, 2009		Geographic Boundary is "North America" Role Type is "Partner" Product Category is "Hardware" Quantity Break is "Over 1000 items bought"			Average payment history range Effective From Date	60 (lowest allowable average days of payment) 90 (highest allowable average days of payment) Oct 1, 2009 (Effective Date)	Discount	25 (Percentage)	

Notice also that the same instance of a BUSINESS RULE may have both specific and generalized outcomes, much like it can also have specific and generalized factors. For example, a “Customer Complaint” event type may lead to rule with the specific outcome of a WORK EFFORT TYPE of “Manage Complaint” as well as a generalized outcome of “Send a discount voucher to the client.”

Notice that because this is a more generalized data model solution, we have decided not to capture factors or outcomes as specific attributes such as **effective from date**, **price amount**, or **discount percentage** as seen in Figure 8.4. Instead we use the generalized factor entities (BUSINESS RULE FACTOR, RULE FACTOR TYPE, RULE FACTOR VALUE, FACTOR VALUE TYPE) and/or generalized outcome entities (BUSINESS RULE OUTCOME, RULE OUTCOME TYPE, RULE OUTCOME VALUE, OUTCOME VALUE TYPE). This allows for more consistency and flexibility because we then have the same format for maintaining factors and/or outcomes, and we can use this same structure to accommodate any additional factors or outcomes that may emerge over time. For example, the **effective from date** of “Jan 1, 2009” from the first row of Table 8.2 (that is in Figure 8.4 Level 2 Business Rules Pattern) is now captured in the third business rule of Table 8.3 as a RULE FACTOR TYPE **name** of “Effective From Date,” and RULE FACTOR VALUE **factor value** of “Jan. 1, 2009.” Another example would be the taking rule that had many factors resulting in a **percentage** discount of “25” that you saw in Figure 8.4 and Table 8.2 and maintaining this in BUSINESS RULE OUTCOME, with a RULE OUTCOME TYPE **name** of “Discount” and a RULE OUTCOME VALUE **outcome value** of “25” that has an OUTCOME VALUE TYPE **name** of “Percentage.”

Note

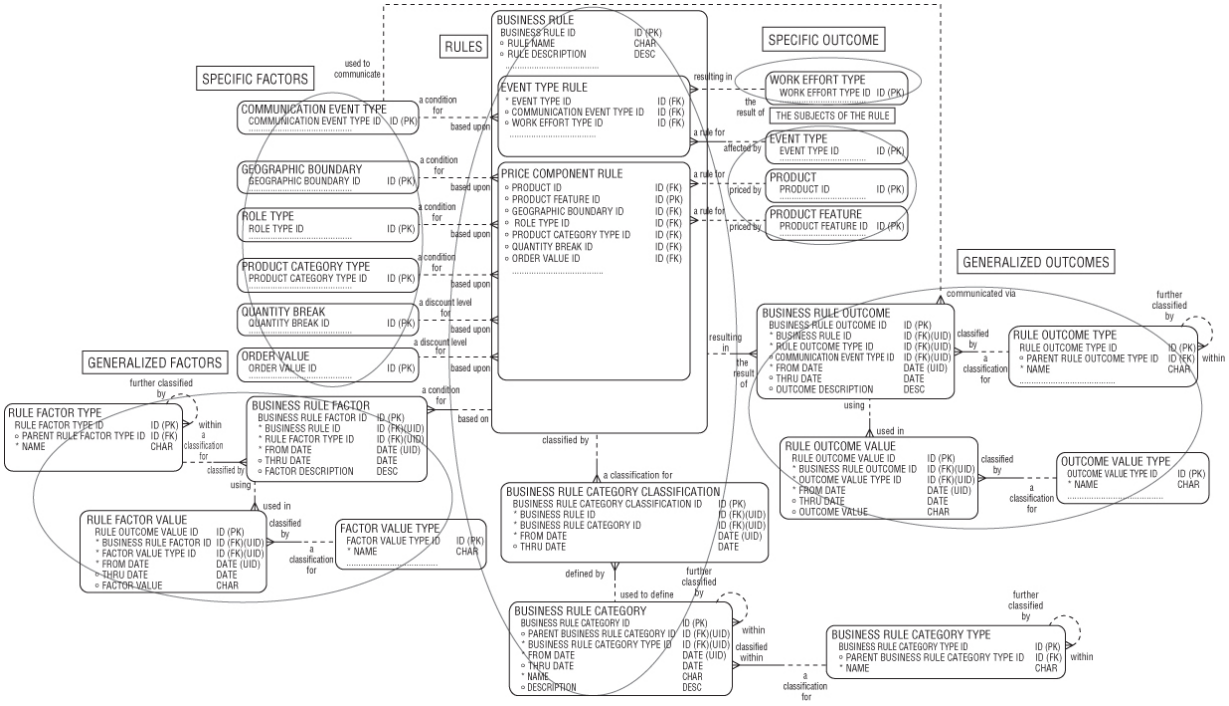
It is also possible with this pattern to keep specific attribute outcomes such as **price amount** (and any other factor or outcome attributes) within the BUSINESS RULE subtypes (such as PRICE COMPONENT RULE) and use the generalized outcomes for other outcomes. However, this means that there is an inconsistent way to handle all factors and outcomes across the enterprise. Some business rule outcomes may be maintained in BUSINESS RULE OUTCOME, and some may be attributes of the BUSINESS RULE supertype or subtypes. Likewise, some factors may be maintained in BUSINESS RULE FACTOR. This may be confusing to developers. But it may also be more semantically correct. As always it is a balancing act between flexibility and semantic rigor. The same logic can be applied to very specific factors like **effective from date**.

Another change that we have made from the Level 2 Business Rules Pattern to the Level 3 Business Rules Pattern is to apply the Level 3 Classification Pattern, as shown by the Figure 8.5 entities of BUSINESS RULE CATEGORY CLASSIFICATION, BUSINESS RULE CATEGORY, and BUSINESS RULE CATEGORY TYPE. This allows rules to be classified in any number of different ways. In the level 2 pattern, there was just one classification, namely, the ENTITY RULE TYPE. Table 8.2 showed that the instances of PRICE COMPONENT RULE may be “Net Price” (for maintaining the prices net of discounts and surcharges under various circumstances) and “Discount” (for maintaining a discount that may be either a flat amount or a percentage). However, what if you also wanted to maintain a different categorization for each instance regarding if it was a “One time price,” a “Recurring charge,” or a “Utilization charge”? For example, there could be a need to classify PRICE COMPONENT RULE instances in several different ways, such as by having a “Price component nature” BUSINESS RULE CATEGORY TYPE with “Price,” “Discount,” and “Surcharge” as instances of BUSINESS RULE CATEGORY and a “Price component frequency” BUSINESS RULE CATEGORY TYPE with “One time charge,” “Recurring charge,” and “Utilization charge” as instances of BUSINESS RULE CATEGORY. This would allow

you to specify that a particular rule that specified the monthly charge for a PRODUCT was classified both as a “Price” BUSINESS RULE CATEGORY and as a “Recurring charge” BUSINESS RULE CATEGORY. Thus, there would be two instances of BUSINESS RULE CATEGORY CLASSIFICATION to maintain both of these classifications.

Consider the example from the previous section where a data professional was asked to capture business rules as they related to EVENT TYPE(s) and PRICE COMPONENT RULE. After the data professional produced Figures 8.3 and 8.4, Kantowitz Electronics wanted to consider capturing all of its business rules in the same way, across the whole of its enterprise. To show the viability and the effects of doing this, the data professional merged Figure 8.3 and 8.4 into Figure 8.6, based upon the Level 3 Business Rules Pattern (illustrated in Figure 8.5).

Figure 8.6 Example of using the Level 3 Business Rules Pattern, Communication Event Type and Pricing



In Figure 8.6 you see that EVENT TYPE RULE and PRICE COMPONENT RULE are now subtypes of BUSINESS RULE. So

the first two rows of Table 8.3 are rules instances of the EVENT TYPE RULE subtype, and the next two rows are instances of the PRICE COMPONENT RULE subtype. The BUSINESS RULE entity contains the common attributes of **rule name** and **rule statement** and these are shown in the second and third columns of Table 8.3. You see that the EVENT TYPE RULE subtype has two rule names and rule statements. For instance, the first row shows the **rule name** of “Complaint Rule 100” and a **rule statement** of “Phone call with an apology must be made when a customer calls on the phone with a valid complaint and a work task must be set up to manage this.” Then there are two rules for the PRICE COMPONENT RULE subtype. The first of these (second to last row in the table) does not have a **rule name** or **rule statement**. This is because the enterprise decided that it will not have a **rule name** or **rule statement** for the rules dealing with the prices for products because there are many thousands of instances. The second rule for the PRICE COMPONENT RULE subtype does have a **rule name** of “Pricing rule 125” and a **rule statement** of “A 25 percent price reduction only if **all** of the following are true:

1. The customer is in North America
2. The customer is a Partner
3. The Product Category is “Hardware”
4. Over 1000 items are purchased
5. Customer pays within an average of 60 – 90 days
6. The date is later than October 1, 2009”

Note

“Pricing Rule 125” is a new iteration of the business rule “Pricing Rule 124” we described in Table 8.2, and it changes the rule in three ways: It removes the restriction of a minimum order value, it changes the generalized factor from a delivery location factor to a factor having to do with the average payment time for the customer, and it changes the effective date. This change raises an interesting question: Is this a new rule or is it a revision of an

existing rule? We suggest that enterprises have parameters to define when a rule is an update of data to the same instance of a rule, when it is another instance that is considered a revision of a rule, and when it constitutes a completely new rule. Also, if it is a revision, perhaps there is a need for a recursion around business rules. Another enhancement to this model could be to apply one of the recursive patterns from chapter 4 to this pattern to show how rules are related to each other (for example, when a rule is a revision of another rule).

Table 8.3 shows the specific and generalized factors and the specific and generalized outcomes for the EVENT TYPE RULE and the PRICE COMPONENT RULE. For example, the first row dealing with an EVENT TYPE RULE with the **rule name** of "Complaint Rule 100" shows a specific factor, namely that it applies to a COMMUNICATION EVENT TYPE **name** of "Phone Call" (because this relates to a specific entity that is in the data model). The factor of "Valid" (that is, that this is a valid complaint) is maintained as a generalized factor because this does not relate to data from an existing entity and is used only for rules processing.

The last row in Table 8.3 shows a PRICE COMPONENT RULE with the **rule name** of "Pricing rule 125" and has several specific factors that are based on data from various entities that already exist the data model such as GEOGRAPHIC BOUNDARY, ROLE TYPE, and so on. The **rule statement** for this instance shows that we need to accommodate the new factor regarding "Customer pays within an average of 60–90 days." In this case, a single factor has two values, "60" and "90," and thus, the BUSINESS RULE FACTOR (of RULE FACTOR TYPE "Average payment history range") has RULE FACTOR VALUE **factor value(s)** of "60" and "90" that represent FACTOR VALUE TYPE(s) of "Lowest allowable average days of payment" and "Highest allowable average days of payment," respectively.

Similarly, the EVENT TYPE RULE and PRICE COMPONENT RULE each have specific and generalized outcomes. In Table 8.3, you see the first examples of specific outcomes of

“Manage Complaint” and “Manage Anniversary Event” because they are outcomes that specifically relate to data in an existing entity in the data model, namely the entity, WORK EFFORT TYPE. The rules state that if the factors are met, there will be outcomes to set up a work effort (such as a project) of the WORK EFFORT TYPE(s) of “Manage Complaint” and “Manage Anniversary Event” to help ensure full resolution that all the necessary steps are taken when there is a customer complaint or customer anniversary.

Table 8.3 also shows many generalized outcomes that are maintained such as “Apology,” “Acknowledgment and thank you,” and “Gift,” for EVENT TYPE RULE(s) and “Price” and “Discount” for PRICE COMPONENT RULE(s). Also notice that some BUSINESS RULE OUTCOME(s) may have relationships to specific entities. Figure 8.6 and Table 8.3 show that the EVENT TYPE RULE instances have BUSINESS RULE OUTCOME(s) that are **communicated via** COMMUNICATION EVENT TYPE(s). For example, the first row shows a rule that has a BUSINESS RULE OUTCOME of “Apology” that is communicated via “Phone Call” COMMUNICATION EVENT TYPE (this is shown in parenthesis in the table).

Why maintain different types of rules using the same data model? Why handle BUSINESS RULE(s), BUSINESS RULE FACTOR(s), and BUSINESS RULE OUTCOME(s) using the same data model? By capturing all of the different specific factors and outcomes for each BUSINESS RULE subtype, you may get some insight into how different sets of rules may be related to each other. For example, if you found that the PRICE COMPONENT RULE and COST COMPONENT RULE (two different possible BUSINESS RULE subtypes) had GEOGRAPHIC BOUNDARY, PARTY TYPE, PRODUCT CATEGORY TYPE, and ORDER VALUE all in common, this may indicate that similar rules should or could be used in both places. Also by capturing all of the BUSINESS RULE FACTOR(s) and RULE FACTOR TYPE(s) in a single place you may discover

synergies between different rule factors. For example, if you discovered that the “Valid” and “Not valid” RULE FACTOR TYPE(s) were used for many different BUSINESS RULE subtypes, this may indicate that they should actually be an entity in their own right, due to their importance across different set of business rules in different parts of the enterprise.

There are also great advantages in having a consistent way to maintain factors and outcomes and also in having a very flexible data model structure that easily accommodates any new factors or outcomes that may emerge. Capturing all of the business rules in a single place is the equivalent of capturing a dictionary of different rules, irrespective of what they apply to, in the BUSINESS RULE supertype. This may make it more efficient to catalog and define the rules as part of a metadata repository effort (or business rules engine effort).

It is also fair to say that capturing all of the business rules in a single model can be abstract and difficult to understand for some audiences. Many BUSINESS RULE subtypes may reuse the same types of factors PRICE COMPONENT RULE and COST COMPONENT RULE. Imagine that four other rules also shared these factors. GEOGRAPHIC BOUNDARY(s), ROLE TYPE, and PRODUCT CATEGORY(s) are often factors that are shared by a great number of types of business rules. The model could become quite complex with many relationships from each factor to each business rule subtype. There are also many common types of outcomes as you saw in the example. A case in point is that many business rules may have outcomes of WORK EFFORT TYPE. However, you can show a separate view of the data model for each BUSINESS RULE subtype to aid in understanding.

Another thing to consider: In contrast to modeling very specific factor attributes and outcome attributes, when you use the generalized factors and generalized outcome

entities to maintain these values, the model is much harder to follow. For example, recording a simple **effective from date** attribute (as we did in the example of the level 2 pattern in Figure 8.4) requires much more navigation through the various generalized factor entities in this pattern.

Thus, although this pattern is more difficult to understand, the power that comes from managing all your business rules using a single data model can lead to better understanding of your rules, the discovery of synergies between rules, the reduction of guesswork on how to capture rules, and finally, a saving in time and effort in developing a rules solution for every business domain that has business rules.

When Should This Pattern Be Used?

We use this pattern when:

- **An enterprise is interested in capturing business rules using the same data model for consistency and ease in maintaining the data model:** Capturing all different types of business rules in a standard data modeling format provides consistency that allows easier management of business rules and the model, and reduces the likelihood that different types of business rules will be modeled with different conventions. Because there is a single structure for handling business rules within this pattern, there are far fewer entities to maintain. If and when a new change is needed to the business rules model that affects all rules, then the model can be updated in one place, as opposed to the level 2 pattern that would necessitate updating each business rules modeling construct.
- **An enterprise is building a metadata repository or rules engine:** Capturing all of the business rules together in this fashion can be an effective way to catalog each of the different business rules, their

factors, and their outcomes. Also by capturing all of the rules, factors, and outcomes together, you show the breadth and depth of the types of rules that are needed for creating a metadata repository. You may also expose common factors and/or outcomes that may apply to more than one business rule entity.

- **There is a need to handle all business rules in a common fashion in order to help with the implementation of business rules:** Imagine a programmer creating a standard interface to manage the BUSINESS RULE FACTOR(s) and BUSINESS RULE OUTCOMES that can be reused. This can be done only when there is a common, stable structure underlying that code. By using a single data model for all rules, this can help facilitate common architecture, rules, processes, and services.
- **There is a need for flexibility in maintaining business rules:** When there are needs for additional business rule types, factor types, outcome types, and category types, these may be added much more easily and without needing to change the data model. In this pattern, use of the generalized factor and generalized outcome entities instead of modeling using attribute factors and attribute outcomes, results in more flexibility to add new types of factors and outcomes when needed, without model changes.

What Are the Weaknesses of the Pattern?

The weaknesses of this pattern are as follows:

- **This is even more difficult to understand than the previous level 2 pattern for many data modelers and nearly all nontechnical people:** This difficulty in understanding can be addressed by showing data

examples, providing training, and using this chapter as a training aid.

- **It can be unwieldy to model all business rules in a single model:** When all of the BUSINESS RULE subtypes are shown in a single model with an all encompassing BUSINESS RULE(s) entity, you will quickly find that the model gets very large and complex. However, this can be managed by having different data model views for each BUSINESS RULE subtype, as well as one view showing the key entities in the overall business rule model.

Synopsis

In this section we showed how you can ‘wrap’ different types of business rules into a single supertype BUSINESS RULE entity. The reason this was done is that most business rules maintain the same basic types of data. For example, all BUSINESS RULE subtypes have:

- Subject(s) of the business rule
- Common business rule attributes such as **rules name** and **rules statement**
- Specific and generalized factors
- Specific and generalized outcomes
- Business rule classifications

In this pattern, by creating a BUSINESS RULE supertype, we show the Level 3 Business Rules Pattern that models common business rule data in the same data model structure, and also accommodates the specific aspects of each business rule subtype. This can help in the development of a metadata repository or business rules engine in many ways. First, the pattern can help show the scope of the different rules that need to be captured and provide insight into common factors and outcomes for various types of business rules. Second, it can catalog all of

the rules, factors, and outcomes that need to be examined. Finally, it can be a good basis for the design to develop an implementation for business rules. This is a complex and generalized pattern that can be used by data professionals who are addressing the problem of integrating various types of business rules, for example, integrating business rules within a subject area or across the enterprise as a whole. It provides the great advantage of a very consistent and flexible way to model all different types of business rules.

Business Rules with Party Roles

For business rules to be effectively used and managed, the various roles that parties play in business rules need to be supported. In this final section we address one important supporting structure that data professionals may find useful in supporting the various roles involved in business rules such as who specifies, manages, is the source of, or is authorized to use business rules.

Why Do We Need This Pattern?

There are certain principles that support business rules that were stated by Ron Ross in his excellent book *Principles of the Business Rule Approach* (Addison-Wesley, 2003). It is with some of these in mind that we show this next pattern. The principles that we support with this pattern are as follows:

- Rules should be managed.
- Rules should be accessible to authorized parties.
- Rules should be specified directly by those people who have relevant knowledge.
- Rules should be single sourced. In other words, the enterprise should not be defining the same rule

differently in various parts of the enterprise. One definitive source for a rule is crucial.

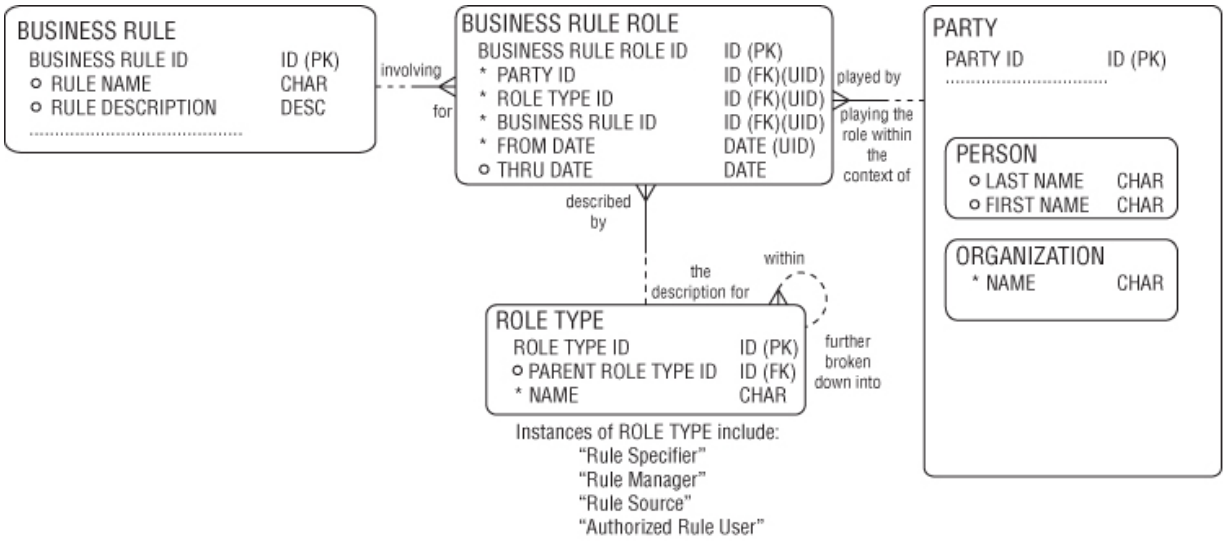
How Do These Patterns Work?

These four principles address the management of business rules, the authorized parties, specification by authorized people, and the single sourcing of business rules, respectively. Each of these principles refers to how a PARTY is involved with the BUSINESS RULE. In Chapter 3, we showed a Level 3 Contextual Roles Pattern that used a three-way associative entity (CONTEXTUAL ROLE) to flexibly relate any ENTITY, PARTY, and ROLE TYPE. This allowed any PARTY(s) to be involved in any ROLE TYPE(s) in any ENTITY for any time frame. In Figure 8.7, we have applied the Level 3 Contextual Role Pattern in relation to BUSINESS RULE(s).

Note

As we have illustrated in this and other chapters, patterns can often be applied to other patterns. In this section, we have chosen to illustrate how the contextual role pattern may be applied to the business rules pattern, and in the last section in this chapter, we showed how the classification pattern could be applied. We could have also shown how other patterns could be applied. For instance, applying the Level 3 Recursive Pattern with Rules could help maintain how rules interact with each other and applying the Level 3 Status Pattern could help maintain what state(s) the rules are in, such as “Proposed,” “In force,” or “Discontinued.” We recommend that if you are developing any new data models, that you check to see which of the patterns may be applicable and then consider applying them.

Figure 8.7 Business Rules with Their Involved Parties



In Figure 8.7, each BUSINESS RULE may be **involving** one or more BUSINESS RULE ROLE(s), each of which may be **played by** one and only one PARTY and **described by** one and only one ROLE TYPE. This means that this model accommodates the four different roles (as well as any number of other roles) that apply to the four different business rule principles. These roles, namely, "Rule manager," "Authorized rule user," "Rule specifier," and "Rule source," are maintained as instances of ROLE TYPE and are explained as follows:

- **"Rule manager":** The rule manager is responsible for the upkeep of the BUSINESS RULE data including its statement, classification, factors, and outcomes. In other words, this person must make sure that the people who specify the rules keep those specifications up to date. This could be the manager of the metadata system or a responsible business representative.
- **"Authorized rule user":** This would be any PARTY that has rights to use this rule in some fashion. The model in Figure 8.7 can be expanded to show which types of users have what types of rights, by adding entities to record authorizations and permissions under various

circumstances. (You could model authorizations by applying the business rules patterns in this chapter!)

- **“Rule specifier”:** These are the “people who have relevant knowledge” who can directly specify the business rule. They could be business people who know how to effectively state the rule and define the classifications, factors, and the outcomes for the rule. IT professionals may also be considered a type of rule specifier in the sense that they may specify the database and/or application design in order to implement the rule.
- **“Rule source”:** The originator of the business rule. The person or organization (PARTY) that is the definitive supplier of the rule.

By applying this flexible contextual role pattern, there is an issue regarding this last role. The business rule principle states that “rules should be single sourced.” This pattern allows for more than one source. We could have applied the Level 2 Contextual Role Pattern and directly related a RULE SOURCE to BUSINESS RULE in a one-to-many fashion to explicitly show that a BUSINESS RULE may be sourced by one and only one RULE SOURCE to support this requirement. However, we decided that it may also be possible over time that a business rule may have more than one source. For example, foreign trade rules may have been sourced originally from the “Common Market,” which was superseded by the “EEC (European Economic Community),” which itself was superseded by the “EU (European Union).” All three of these may be the source of the same rule, depending on what time frame you look at the definition of the rule.

Note

There are also other alternate ways of maintaining the various roles for BUSINESS RULE(s), and we suggest you see Chapter 3

for a discussion of some of these alternatives.

This pattern supports the creation of any other business rule roles that may be defined by your enterprise. You are not just limited to the roles that we have defined, based on Ron Ross's business rule principles. But Ron Ross's business rules roles are a good guide for the type of roles involved in business rules.

When Should This Pattern Be Used?

We use this pattern when:

- **There is a need to capture the different PARTY(s) that are involved with BUSINESS RULE(s) in various ROLE TYPE(s) over time:** Possible ROLE TYPE(s) include “Rule manager,” “Authorized rule user,” “Rule specifier,” and Rule source”; however, the pattern supports any number of roles, parties involved, at many different times for any BUSINESS RULE(s).

What Are the Weaknesses of the Pattern?

The weakness of this pattern is as follows:

- **Some enterprises may not need this amount of detail:** They may be interested in the specification of only the business rules and perhaps one or two key roles. Thus, an alternative is to apply a different Contextual Role Pattern instead of the Level 3 Contextual Role Pattern shown in this chapter.

Synopsis

In this section we described how PARTY(s) playing different ROLE TYPE(s) are involved with BUSINESS RULE(s) at different points in time. We described how the ROLE TYPE instances of “Rule manager,” “Authorized rule user,” “Rule specifier,” and “Rule source” support the principles that business rules need to be accessible to authorized parties

only, specified by the right people (or organization), managed, and single sourced. This alternative was based on concepts described in Chapter 3 of this book.

Summary of Patterns

Table 8.4 contains a synopsis of all the patterns covered in this chapter.

Table 8.4 Synopsis of the Patterns

PATTERN	DESCRIPTION	WHEN TO USE	WHEN NOT TO USE	PROS	CONS
Level 2 Business Rules Pattern	A more specific way to model business rules, rule factors, and rule outcomes, where each type of business rule is modeled in its own distinct data model structure.	There is a need to integrate business rules into the data model because they are an integral part of the data requirements and interrelated with other data model entities. As a statement of scope for the different types of business rules There is a need to have a flexible and consistent data model structure for capturing business rules. When the enterprise wants to be data driven and maintain business rules in a dynamic fashion so that when the parameters for rules change, they can be easily updated. When an enterprise is interested in capturing how business rules for specific subject areas are related to the other entities in its data models (or enterprise data models). If an enterprise does not have a metadata environment or rules engine to capture rules, or has not implemented a metadata environment, these patterns can provide a place holder for capturing those rules, factors, and outcomes. When an enterprise wishes to have a flexible way to maintain additional business rules factors and outcomes that may emerge over time and that are not related to existing data model entities.	When an enterprise already has a centralized rules engine or integrated metadata environment and decides to maintain all their business rules there, or captures business rules in process models and rules dictionaries that are not related to the data model When there is a need for a more consolidated approach of maintaining all business rules in a single data model	A flexible way to maintain rules data, factors, and outcomes for rules. Accommodates any number of new factor or outcome types that may emerge over time via the generalized entities for factors and outcomes. A consistent method to model all types of conditional statements ('if/then' or 'while') data requirements. A way to be much more data driven by capturing the factors and outcomes of business rules in a data model. Integrates business rules into your data models, which enhances your understanding of data, business rules, and how they are related	This model is complex and may not be the most effective way of describing business rules to the business. However, we have found some business representatives relate to these models very well. Some data modelers would find this model difficult to use if they had no experience with business rules. There may be many rules entities that could be consolidated into a single rules entity for consistency and manageability. Provides only a single classification for a rule; however, this can be easily changed by substituting the Level 3 Classification Pattern. Data professionals may mistakenly use the generalized factor and outcome entities instead of relating the factor or outcome to an existing data model entity and this can lead to redundantly and inconsistently capturing data.

Level 3 Business Rules Pattern	An approach for capturing and managing all types of business rules in the same data model in a very flexible way.	When an enterprise is interested in capturing business rules in a consistent fashion using a single, common data model for the enterprise When there is a need for flexibility in modeling business rules for many different areas in their enterprise. When an enterprise is working toward a more full-blown metadata repository or rule engine. Capturing all of the business rules together in this fashion can be an effective way to catalog each of the different business rules, their factors, and their outcomes. As a statement of scope for a metadata repository. When a single data model structure is needed to help implement common architecture and services.	As a statement of scope or a method for gathering and/or validating data requirements about business rules from nontechnical business representatives. When a more specific style of modeling business rules is needed.	This provides a very flexible model that allows for a great number of factors, outcomes, and classifications, without your needing to change the data model when new factors, outcomes, and/or classifications are needed. By wrapping different types of business rules into a single supertype you create a single, consistent way to manage all business rules and data model maintenance is easier. Handling all business rules in a common fashion can help with the implementation of business rules by allowing very similar routines for accessing business rules. Integrates business rules into your data models, which enhances your understanding of data, business rules and how they are related	This is a very generalized model and may be difficult to understand for some audiences. Using the generalized factors and outcomes requires a lot more data model navigation than maintaining factors and outcomes as attributes. If a new business rules classification, factor, or outcome is discovered, it is often easy for a data professional to assume that it is covered and not do proper analysis If all of the rules are shown in a single diagram as subtypes of an all encompassing BUSINESS RULE(s) entity, the model can become quite large and unwieldy. We recommend breaking this up into data model diagrams for each subtype of business rule.
Business Rules with Party Roles	A flexible approach for capturing the roles involved in business rules.	When you need to capture many roles for business rules such as who manages, specifies, is authorized to use, and is the source of a business rule.	When you don't need to capture many roles that people or organizations play in business rules.	Easy and flexible way to support any number of roles for any number of parties, for any time frames. Examples of these roles include the authorization, specification, management, and sourcing roles for business rules.	If there is only a need to capture only one or two roles for the business rule or if the number and types of roles is not expected to change over time. If this is the case, this pattern may be overkill and it may be more appropriate to use the level 1 or level 2 contextual role pattern.

References

[1] From How to Build a Business Rules Engine: Extending Application Functionality through Metadata Engineering by Malcolm Chisholm (Morgan Kaufman, 2004).

[2] From Principles of the Business Rule Approach by Ronald G. Ross (Addison-Wesley, 2003).

[3] For more information on data models for orders, see Chapter 4 of The Data Model Resource Book, *Revised Edition, Volume 1, A Library of Universal Data Models for All Enterprises*, by L. Silverston (Wiley, 2001).

[4] From “Sentence Patterns for Rule Statements” in Principles of the Business Rule Approach by Ronald G. Ross (Addison-Wesley, 2003).

[5] From the ever-excellent data modeler, Ed Landale.

[6] A good reference for physical implementations for business rules is How to Build a Business Rules Engine: Extending Application Functionality through Metadata Engineering by Malcolm Chisholm (Morgan Kaufman, 2004).

[7] An earlier version of this pricing model can be seen in The Data Model Resource Book, *Revised Edition, Volume 1, A Library of Universal Data Models for All Enterprises*, By L. Silverston (Wiley, 2001).

Chapter 9

Using the Patterns

After reviewing the patterns, you are likely at the point where you are asking, “How do I use, apply, and/or implement these patterns in my enterprise?” This question is hard to answer because the answer often includes the words “it depends.” The answer is different if it is asked by a person developing a data model for a prototype system than by someone developing a data model for a data warehouse. Even two people both developing data models may use the patterns in different ways because both of the people have different views of what a data model is or have different audiences to whom they are presenting them!

In each of the preceding chapters, we showed the strengths and weaknesses of each pattern at different levels of generalization and let the user of the patterns decide which pattern to use given their special circumstances. For example, someone creating a conceptual data model and presenting it to business people may choose to use the level 1 patterns because the strengths of those patterns suit that circumstance. Someone creating a data model for a data warehouse that needs to have a flexible design may apply level 3 patterns because the strengths of that level of patterns suits that particular need. For example, that type of pattern can generally accommodate the wide variety of data requirements of the source systems feeding the data warehouse.

This chapter shows how the patterns that we discussed in the previous chapters can be used to create different data models that meet different needs. A good way to imagine

this process is to think of each of the different levels of patterns you saw in the previous chapters as interchangeable components. In this chapter, we take each of the component patterns and combine them together to create data models that solve common data modeling challenges that most enterprises need to address.

What Is in This Chapter?

First, we describe a scenario that will be repeated and built upon throughout the chapter. This scenario evolves as circumstances change in each section. Then in the rest of this chapter, our intention is to address what we believe are some of the most common circumstances in which people could use the patterns to build data models. These include the following:

- **Prototyping and scoping data models:** These patterns could be used for prototypes, as part of a scope statement, and in other circumstances where there is a need to specify information requirements. These are examples of using the patterns to develop very specific data models and generally are used for gathering and validating data requirements with business-oriented audiences.
- **Application data models:** When there is a need to develop an application system, these patterns can be used to develop a data model for the application.
- **Enterprise data models:** This section addresses the need for using the patterns to develop data models that reflect the needs of the enterprise as a whole. Also, we show a specific approach to enterprise data modeling that provides alternative data model constructs to model the same type of data. Thus, some data requirements are modeled using both specific and

generalized patterns, providing a choice to applications using the enterprise data model.

- **Data warehouse data models:** This section has two subsections for showing how these patterns can be used for developing two different types of approaches regarding data warehouse models, namely:
    - **Relational-based data warehouse models:** The patterns can be used to start development of a relational-oriented data warehouse data model. This model may be used as the basis for the design of a data warehouse or operational data store.
    - **Star schema-based data warehouse models:** Each of the patterns can be used to better understand the data and uncover complexity that would need to be resolved in a star schema. Also, the patterns can be used to start the design of dimensions and facts.
- **Master data management (MDM) data models:** These models show how to use the patterns to accommodate the flexibility needed when capturing master data and reference data with rigorous rules on how to use that master data and reference data.

> **Note**
>
> In this chapter we don't delve into the details of each pattern because we have covered this material in other chapters. The purpose is to show how we integrate the patterns together to develop models that fit a purpose and why we choose to use certain levels of patterns in different situations.

The Scenario

To illustrate the thought process in choosing which level of pattern to use, we have created a scenario that evolves with

the changing needs of each type of model. Our example shows what patterns to use and how we would use these patterns for various types of efforts, for example, for a prototype, an application, an enterprise data model, a data warehouse relational model, a data warehouse star schema, and a master data management model. In each of these sections we explain our approach, why we chose to use one level of pattern over another, and why we mixed different levels of patterns.

Imagine a large equipment distributor for the petrochemical industry based in the Emirate of Dubai called Sands Distribution. This firm has grown significantly over the past few years with the increase in oil revenues and the need for its equipment in the Middle East and all over the world. Its main data issues stem from its growth. It has acquired smaller distribution firms in many different countries that have data of their own. The enterprise realizes that data may be one of its greatest potential assets, but how can it best use this asset? It wants a consistent data strategy for managing and integrating all its data, including the various data formats it receives from acquired companies.

The chief executive officer's vision includes developing Sands Distribution into a customer-centric business. To do this, the enterprise needs to manage its customer data in a better way, but it is not sure how to do this. For example, it is not sure if its own existing customers are already doing business with these newly acquired companies. It can't easily find out customer trends, evaluate risks regarding losing customers, or even know how many customers it has because it doesn't have integrated customer data. It is under these circumstances that the enterprise has employed a team of data professionals to help manage this valuable customer data.

The customer data it wishes to manage includes the following:

- The name of the customer and its contact information, such as telephone numbers, postal addresses, and email addresses.
- The status of the customer, for example, if the customer is active, inactive, has approved credit, or is under investigation.
- Customer hierarchies and organizational structures—that is, who are the parent companies of their customers and who are their subsidiaries and divisions?
- The different roles that the parties play in general, for example, customer, salesperson, and partner.
- The different specific roles that a customer plays within the context of important transactions, such as being the bill-to or ship-to customer for orders.
- The different classifications of customer, such as customer type, customer size, industry type, and so on.

With this in mind let us begin by examining the first part of our scenario: Prototype models, Scope statements.

Prototype Models, Scope Statements

As we have discussed throughout this book, each of the patterns in this book can have a different level of generalization. You can mix different levels of generalization in data models to achieve your specific needs. Thus, we suggest using the appropriate patterns that are suitable for the type of effort and circumstances involved.

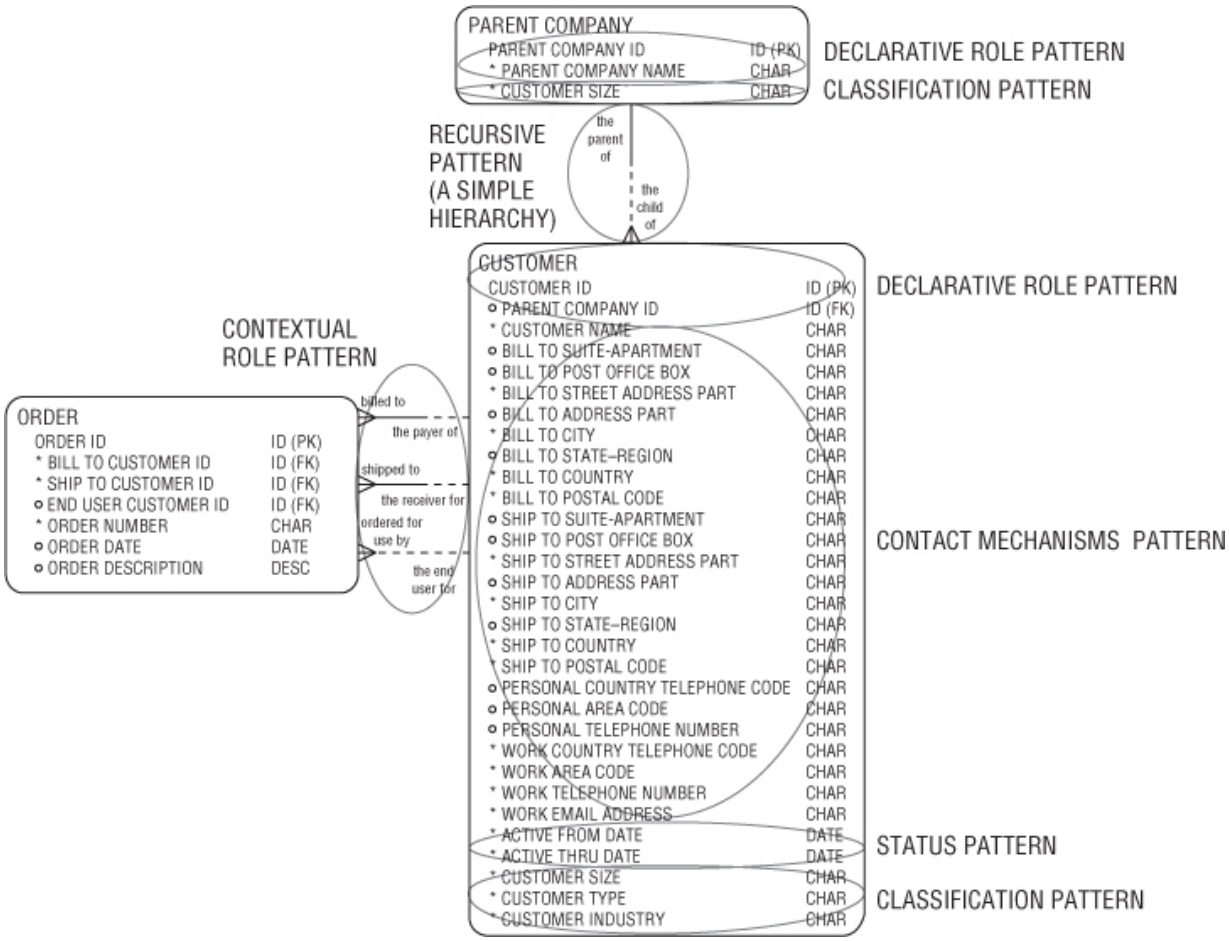
In this section, we describe how to use the different patterns to help illustrate scope and to show concepts, business data needs, and data issues. We then show how the patterns may be used to create a simple data model

used for a prototype. The patterns described in this section can also be used as a starting point to establish a baseline for communication across functional boundaries as they capture common core terminology.

The Scenario for These Models

In the previous section we introduced you to Sands Distribution. As you saw, it has employed a group of data professionals to help shape its data strategy that mirrors its desire to become a more customer-centric company. As part of this effort, the data professionals proposed creating a strategy statement for their future data requirements. As part of this statement, they created a scope statement of Sands Distribution customer data. However, they also wanted to create a prototype set of reports, screens, and analytic routines to illustrate where Sands Distribution could be in the future. The senior data professional felt that these working prototypes would be a powerful way to convince senior management of the benefits of integrated customer data as well as a way to ensure the scope of required data was captured. With this in mind, the data professionals created Figure 9.1 as an initial scope statement.

Figure 9.1 Data model for use in scope statements and understanding data requirements



How Do These Models Work?

You can see in Figure 9.1 a very specific data model made up mostly of level 1 patterns. The CUSTOMER entity shows an example of applying a Level 1 Declarative Role Pattern in which you capture the entity CUSTOMER as its own separate entity (independent of other roles) and include **customer name** as an attribute of a person or organization playing the role of a customer. A declarative role can be defined as "the stated actions and activities assigned to or required of a person or an organization." Notice that we call the declarative role CUSTOMER and not CLIENT or some other name. This is an important point that may seem trivial, but it is crucial to capture the right terminology for declarative roles. A company that Sands Distribution has acquired may

use the term CLIENT or PARTNER. Are they the same as CUSTOMER? Sands Distribution decided to use the entity name CUSTOMER for any person or organization “that has purchased, been shipped, or used products (either goods or services) from the enterprise. An organization or person may play the role of a customer.” The PARENT COMPANY entity is another example of applying the Level 1 Declarative Role Pattern for companies that play the role of a higher-level umbrella organization that controls the activities of the customer in some way.

> **Note**
>
> You may capture other terms such as CLIENT that may be a synonym of CUSTOMER, or they may even have a different meaning and be declarative role entities in their own right.

In Figure 9.1 you also can see the Level 1 Contact Mechanism Pattern from Chapter 7. In this model, this pattern illustrates the various attributes that capture different ways to contact customers. In this figure you see **bill-to street address part**, **bill-to city**, **work telephone number**, **work email address**, and so on. In addition to capturing the actual contact mechanism details, such as a telephone number, the model also captures the type of contact mechanism and the purpose or usage of the contact mechanism, such as **bill-to address part** (a purpose and contact mechanism type) or **personal telephone number** (a usage and contact mechanism type). So, you can see that in this case the data professionals used a model to capture the specific methods for contacting customers. First, they wanted to capture the scope of all of the different possible ways to contact a CUSTOMER, such as postal addresses, telephone numbers, and email addresses. Second, they also needed to capture the type, purpose, or

usage for each of the different contact mechanisms. Figure 9.1 also captures whether a customer is active or inactive.

A customer is considered active if they have completed any transactions with Sands Distribution in the last 2 years. This is captured using the Level 1 Status Pattern as the attributes **active from date**, **active thru date** of CUSTOMER. There may be more statuses for a customer, but the data professionals have not identified them as of yet.

Each instance of CUSTOMER may be classified in different ways. Sands Distribution initially wanted to capture the CUSTOMER(s) by **customer size**, **customer type**, and **customer industry**. These classifications are the 'bread and butter' of reporting and analytics for Sands Distribution. The reports get broken down or rolled up to these core set of classifications. Sands Distribution indicated that it may wish to capture the organization structures of its customers. Each of the CUSTOMER(s) may be part of a larger enterprise. To support this requirement, the data professionals created PARENT COMPANY to capture the organization structure of their customers. PARENT COMPANY captures the name of the parent company (**parent company name**) and the size, such as "Large," Medium" or "Small," of the parent company (**customer size**). Sands Distribution said that it only needed to classify the PARENT COMPANY in terms of its size, thus PARENT COMPANY contains only the **customer size** attribute (and not the other classification attributes).

"Each PARENT COMPANY *must be* **the parent of** one or more CUSTOMER(s), and each CUSTOMER *may be* **the child of** one and only one PARENT COMPANY." This leaves open the possibility that a CUSTOMER may not have a PARENT COMPANY, but if you capture a PARENT COMPANY, it must have at least one child CUSTOMER. Thus, in this model, they

applied the Level 1 Recursive Pattern from Chapter 4 to model a very simple, two-level hierarchy structure.

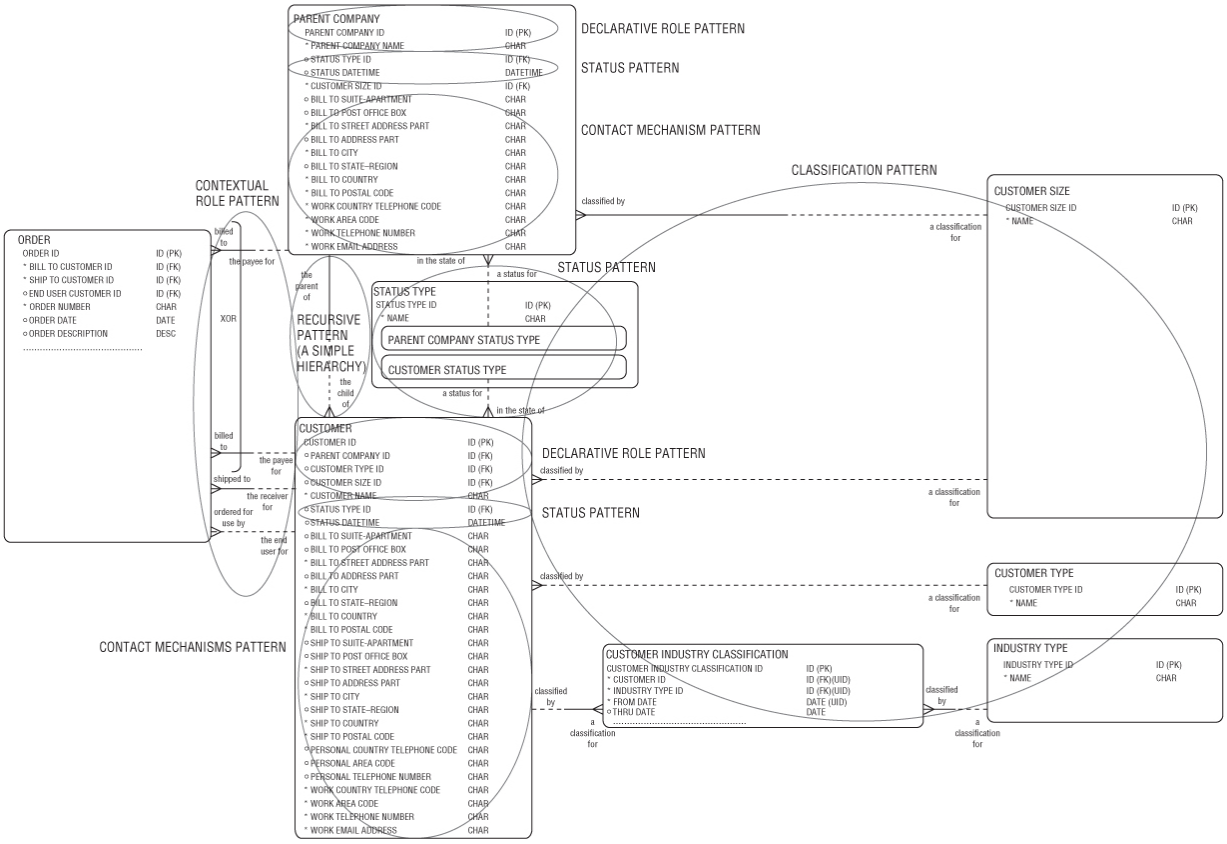
In Figure 9.1, you can also see the capture of the contextual roles for a CUSTOMER as they are related to ORDER. Three relationships (and three foreign keys) each capture a role that CUSTOMER plays within the context of ORDER. In Figure 9.1, you see three contextual roles in the relationships: "each ORDER *must be* **billed to** one and only one CUSTOMER, **shipped to** one and only one CUSTOMER, and *may be* **ordered for use by** one and only one CUSTOMER." This captures the specific contextual roles (bill-to, ship-to, and end user customer) that a CUSTOMER has in relation to an ORDER. Notice that the data professionals applied the Level 2 Contextual Role Pattern here because they deemed that this would serve the purpose of better understanding the data requirements, versus using a Level 1 Contextual Role Pattern. In Figure 9.1, you can see that most of the CUSTOMER data was captured as attributes of CUSTOMER. In the application of the level 1 patterns, the model tends to contain many attributes. For example, the Level 1 Contact Mechanism Pattern captures telephone numbers via the attributes **work country telephone code**, **work area code**, and **work telephone number**, and the Level 1 Status Pattern was applied, resulting in attributes of **active from date** and **active thru date** to capture if an instance of CUSTOMER has conducted some business with Sands Distribution in the last 2 years. As we mentioned, there are also specific attributes to classify the CUSTOMER by **customer size**, **customer type**, and **customer industry**.

The data professionals used the Figure 9.1 data model to validate the data requirements and as part of the scope statement with subject matter experts in Sands Distribution. They got feedback that included the following:

- CUSTOMER(s) may have many more statuses than just active and inactive.
- The subject matter experts also wanted to capture contact information about the PARENT COMPANY because it may be different from the CUSTOMER contact information. However, they also stated that PARENT COMPANY(s) did not have contact mechanisms that were for “personal” use, such as **personal telephone number**.
- Sands Distribution wanted to capture the statuses, contact mechanisms, and size classification for PARENT COMPANY(s) as well as CUSTOMER(s).
- The subject matter experts in Sands Distribution stated if there is a PARENT COMPANY for the CUSTOMER, then the PARENT COMPANY for their CUSTOMER(s) is the payee for any order. In other words, if a parent company exists, then all bills need to be sent to the PARENT COMPANY, not to the CUSTOMER, and if there is no PARENT COMPANY, then the bills are sent to the CUSTOMER.
- The subject matter experts in Sands Distribution also stated that all ORDER(s) get delivered to the CUSTOMER, never to the PARENT COMPANY, even though the PARENT COMPANY paid for the order. In other words, all shipments are directed to the CUSTOMER, not the PARENT COMPANY.
- Sands Distribution classified PARENT COMPANY only by size, whereas CUSTOMER(s) are classified by size, type, and industry. It also stated that it did not know all of the different customer types, sizes, or industries it wished to capture going forward. Additionally, the company stated that the same customer could be classified into many industries at the same time (for example, a customer could be in both manufacturing and distribution).

The data team created a working prototype that showed the possibilities that integrated customer data can provide. With this in mind, and based on the feedback from subject matter experts in Sands Distribution, the data team created Figure 9.2 as part of a scope statement and as a basis for the database design for a prototype. Notice that in Figure 9.2, in order to accommodate the additional requirements, the Level 1 Status Pattern and Level 1 Classification Pattern were replaced by a Level 2 Status Pattern and Level 2 Classification Pattern. Also, the PARENT COMPANY entity now employs the Level 1 Contact Mechanism Pattern to record contact information associated with the PARENT COMPANY. The data model shows that each ORDER must be billed to either the CUSTOMER or to the PARENT COMPANY. The rule describing that if a parent company exists, then the order must be billed to that parent company was recorded in a business rules document that served as a supplemental document to the data model. They thought about applying the Level 2 Business Rules Pattern using an ORDER RULE entity; however, they deemed that was too complex for a prototype.

Figure 9.2 Data model for prototypes



In order to meet the additional requirements that were discovered, the team converted the model described in Figure 9.1 into the prototype model seen in Figure 9.2. This was a relatively straightforward process consisting of:

- A find and replace of the classification and status level 1 patterns for the level 2 patterns. This met the requirement of being able to add new instances of classifications and statuses more easily. In the Level 2 Classification Pattern, the team also used a many-to-many relationship from CUSTOMER to INDUSTRY TYPE to accommodate the need for a customer to be classified in multiple industries.
- The addition of the Level 1 Contact Mechanism Pattern to PARENT COMPANY, thus including the needed contact mechanism individual attributes.
- The addition of another contextual role from ORDER to PARENT COMPANY (and an exclusive arc) because it was

discovered that the ORDER would be billed to the PARENT COMPANY, if one existed, or else it would be billed to the CUSTOMER if there was not a PARENT COMPANY.

> **Note**
>
> The level 1 patterns (or level 2 pattern) in Figure 9.1 could also have been replaced with level 3 or even level 4 patterns if more flexibility was needed for the prototype. For example, the team may have used a Level 3 Classification Pattern if they thought that they needed to demonstrate how new types of classifications could be added on the fly. That is the beauty of the data model patterns; they essentially support the same core function to a greater or lesser extent of flexibility. This means that they are interchangeable. You can see this illustrated in detail as you go through each evolution of the models in this chapter.

Why Do We Do It This Way?

These particular models are very specific. Level 1 patterns were mostly used to create Figure 9.1 (with the exception of one level 2 pattern for contextual roles), and a combination of level 1 and level 2 patterns were used to create Figure 9.2.

So, why did the data team choose to go mostly with the level 1 patterns for Figure 9.1? The data requirements of Sands Distribution were very specific, and the audience for these models was nontechnical and wanted to see how their requirements were met, so the team used very specific patterns to start the analysis process, using very understandable data model constructs. The primary purpose of the first diagram is to communicate and validate data requirements and to be used as part of a statement of scope. The data team wanted to make it as clear as possible and to represent the requirements in an easily understood fashion. To do this, they wanted to specifically name each of

the important data elements of interest to the business. In Figure 9.1 you see the major data entity(s) PARENT COMPANY, CUSTOMER, and ORDER. They then drilled into much of the specific business data and captured them as attributes or relationships.

An important purpose of the Figure 9.1 diagram was to help facilitate the communication process by using the model as a tool to help understand the data requirements. Figure 9.1 is unambiguous with a minimum of technical elements. It can be easily explained to almost any audience, especially when accompanied by supporting material such as use cases and instance diagrams.

> **Note**
>
> We left the primary key attributes and foreign key attributes in the Figure 9.1 and 9.2 diagrams. If you think that these attributes could confuse your audience, it's very easy to remove them. We often do this ourselves when we are presenting to nontechnical audiences.

This diagram also helps to start the data modeling process in a very simple fashion in order to better understand the data requirements. This point is often forgotten. This diagram also helps members of the data modeling team visualize the terms of reference for the area under examination. Some senior data modelers may feel that this is unnecessary and want to jump right to a generalized and normalized data model. However, in our experience, we feel that data modelers can benefit from specific modeling in order to get clarity on the information requirements before generalizing the model.

This model was easy to put together in a short period of time. The data team did not want to invest a lot of time building a complex model as part of a statement of scope. They felt they needed to ‘get the ball rolling’ with the analysis of Sands Distribution customer data issues. This

model helped them quickly uncover core terminology, core issues, and core concepts, while using the models as an effective tool and not investing too much time “modeling in isolation.” In other words, they used this approach instead of using highly generalized models that they could not use effectively to gather and validate data requirements.

The data team was careful in capturing the correct terms that the business uses (that is, PARENT COMPANY, CUSTOMER, and ORDER). This may seem trivial, but it's important that everyone ‘speaks the same language’ in data modeling efforts. The level 1 patterns help identify the appropriate terms to use when discussing the data requirements with business representatives.

Because the model is specific, it is easy to see mistakes or misunderstandings regarding the information requirements. In other words, the data team can put this model up as a ‘straw man’ so the business representatives can point out where the data team got the analysis wrong. For example, after reviewing the relatively straightforward data model shown in Figure 9.1, the business representatives saw that they wanted PARENT COMPANY to be captured with its relevant contact mechanisms, they pointed out that a CUSTOMER and a PARENT COMPANY can have statuses other than “active” and “inactive,” and so on.

> **Note**
>
> We could also have used Figure 9.1 as a basis for a prototype. In fact, it is *initially* easier to build prototypes using level 1 patterns because they are so specific and therefore easier to understand and use for programming.

Why then did we choose to go with more level 2 patterns for Figure 9.2? Interestingly, the reason we replaced some of the level 1 patterns in Figure 9.1 with level 2 patterns in Figure 9.2 was based on the “unknowns.” For example, the subject matter experts of Sands Distribution stated that

they knew that there could be many status types, but they did not know what they were yet, and the Level 2 Status Pattern allowed new types of statuses to be added much more easily. They also stated that they did not know all the various instances of size, type, and industry classifications that may exist. Therefore, by using the Level 2 Classification Pattern and maintaining these classification instances in their own entities, new customer types, customer size types, and industry types could be much more easily managed and kept consistent. The new model also provided the flexibility to classify customers by multiple industries types. Thus, we had captured the additional needs and also provided more flexibility regarding status types and classification types, which was needed in the prototype. The team could also list out possible status and classification types in a table similar to the tables that we have used throughout this book to show examples of data.

The contextual roles applied to both PARENT COMPANY and CUSTOMER. The subject matter experts stated that either PARENT COMPANY(s) or CUSTOMER(s) could be “Bill to” customer for ORDER(s), and only CUSTOMER(s) could be the “Ship to” customer or the “End user” customer for an ORDER. The data team captured these very specific rules as relationships using the Level 2 Contextual Role Pattern.

In short, this solution maintains many aspects of the specific nature of the model seen in Figure 9.1, especially regarding contact mechanisms, declarative roles, and contextual roles, but it also provides two important benefits to the data team. First, it allows the data team to build a prototype from Figure 9.2 that provides the capability to maintain and catalogue new types of statuses and classifications in their own entities as they are uncovered (or deleted as they become defunct). Second, it provides a natural path to move toward a more flexible data model as the project matures.

Note

We want to emphasize that caution should be exercised with the use of level 1 patterns because this type of pattern is not generally an effective foundation for a solid database design. Data models generally have two purposes: they can be used as a tool for understanding data requirements, and they also serve as a starting foundation for a database design. This pattern (as with most level 1 patterns) serves the former purpose very well; however, it is usually very ineffective regarding the latter purpose.

What Are the Strengths of Using Patterns for the Solution?

The strengths of using the specific patterns (such as level 1 and level 2) for understanding data requirements and for prototyping (as shown in Figure 9.1 and Figure 9.2) are as follows:

- **Easy to understand:** The specific nature of these models that use the level 1 and level 2 patterns allows you to show, in a relatively straightforward manner, all of the key entities and attributes within the scope of the subject area under examination. These diagrams may be useful to gather and validate data requirements as well as to use them as part of a scope statement.
- **Easy to use:** The level 1 and level 2 patterns are easy to apply, easy to use in development of data models, and easy to communicate in a short period of time. They allowed the data team to quickly 'get the ball rolling' with the subject matter experts of Sands Distribution.
- **Easy to implement prototype:** It is generally easier and quicker to build a prototype based off these simple structures.
- **Good starting point:** These types of models provide you with a good starting point to start the data

development process. In other words, you have to start somewhere, so why not start with the clearest and simplest models? These models jump-start the scoping process, the prototyping process, and the process of extending the data model to become more stable and flexible, such as by adding level 2 patterns in certain places, as we did in this scenario.

- **Effective for communicating to business representatives:** We often use models like this as part of our communications kit when talking to business representatives. The common business terminology is shown in this model, which helps to involve businesspeople in the modeling exercise. We are constantly pleased to find that businesspeople find these models easy to understand and fun to work with.
- **Simple to change to a higher-level pattern:** When we discover an 'unknown,' such as "we know we capture the status of an order, but we don't know what those statuses are," it is very straightforward to replace the level 1 patterns with their level 2 equivalents (or even level 3 equivalents). The higher-level patterns (levels 2, 3, and 4) provide more 'place holders' for unknowns. Of course, this does not excuse us from doing the analysis to discover these 'unknowns.'

What Are the Weaknesses of Using Patterns for the Solution?

The weaknesses of applying patterns for requirements gathering and prototyping are as follows:

- **Using level 1 patterns does not generally provide a good basis for implementations:** Granted, in some very specific circumstances where a set of data is very static and the values are not generally reused, you may use a level 1 pattern solution for an implementation, but

these situations are very rare indeed. Also, if level 1 patterns are implemented in a database design, they would often cause redundant data. The level 1 patterns often include many attributes that are not normalized (that is, they don't follow 1st, 2nd, or 3rd normal form). For example, if you used the Level 1 Classification Pattern to implement a **customer size** field where the values are "Large," "Medium," and "Small," this would lead to many instances that redundantly had the value "Large," as opposed to having a foreign key that pointed to a CUSTOMER SIZE table, which is how it would be modeled using the Level 2 Classification Pattern.

- **When level 2 patterns are used instead of Level 1 Patterns, you lose some specific understanding:** For example, we created STATUS TYPE and CUSTOMER STATUS in Figure 9.2 to replace **active from date** and **active thru date** that were attributes of CUSTOMER in Figure 9.1. It is obvious from Figure 9.1 that there is an "Active" status (**active from date**, **active thru date**). This is not apparent in Figure 9.2. In other words, the specific attributes (**active from date**, **active thru date**) became generalized into STATUS TYPE.
- **Lack of ability to reuse common routines:** When you are using very specific patterns, the data model structures tend to be very different across a model, and thus, unlike when you use some of the generalized structures, there are usually not common routines that can be used for different parts of the model.
- **Lack of flexibility:** Though the level 1 and level 2 patterns are much easier to understand, there are a great number of drawbacks regarding flexibility. For example, the data model will need to be changed for many different situations, such as when new types of classifications are discovered, multiple customer statuses at the same time are needed, new types of

contextual roles are required, new types of contact mechanisms are used, or you have more than two levels for the customer organization hierarchy.

Synopsis

This section dealt with the beginning of the development of a customer information system for Sands Distribution. The data team created a simple, specific data model as part of a scope statement to start the discussions and the data modeling around customer data. The data model captured the common terminology and the attributes that were important to Sands Distribution.

After the data modelers used the first data model to get feedback from subject matter experts, additional requirements emerged. The data modelers then expanded the model in Figure 9.1 and created the data model that would be used for the initial prototype (Figure 9.2). This was a little more flexible model than the model shown in Figure 9.1.

The enhanced data model was quickly created by replacing a couple of the level 1 patterns with level 2 patterns. For example, in Figure 9.2, the classification attributes of Figure 9.1 became entities in their own right, a many-to-many relationship was created to allow multiple industry classifications for a customer, an additional contextual role entity was added to allow the parent company to be the bill-to customer, and statuses were maintained in their own STATUS TYPE entity. This created a good baseline for communication with the business and IT, and a more flexible initial data model, which could be used as the basis for building some prototype screens and reports that illustrated the need for customer information.

As a general rule, we believe that even specific data models such as those shown in Figures 9.1 and 9.2 should have additional supporting documentation, such as instance

diagrams and use cases. Although it might be effective for communications, a rule of thumb is that you should not use an initial level 1–based model (such as that shown in Figure 9.1) for implementations because it could cause many data anomalies. Likewise, you should also be careful not to use prototype models as full-scale implementations because they don't have the flexibility that is usually needed to support major production applications.

Application Data Models

The ‘bread and butter’ work for many data professionals is the creation of data models for applications. This section discusses how to use the patterns in this book to develop robust, flexible, consistent, and practical models that can form the foundation for a solid application database design.

The Scenario for This Model

After presenting the prototype to senior management and re-examining the scope of the project, the data team was asked to produce a full-blown data model to support an application that included the entering, updating, access, and management of customer data. Interestingly, Sands Distribution wanted this application built but still did not have a good idea about the different customer statuses, customer contextual roles, or declarative roles that could exist. This is not an uncommon situation with many application development projects.

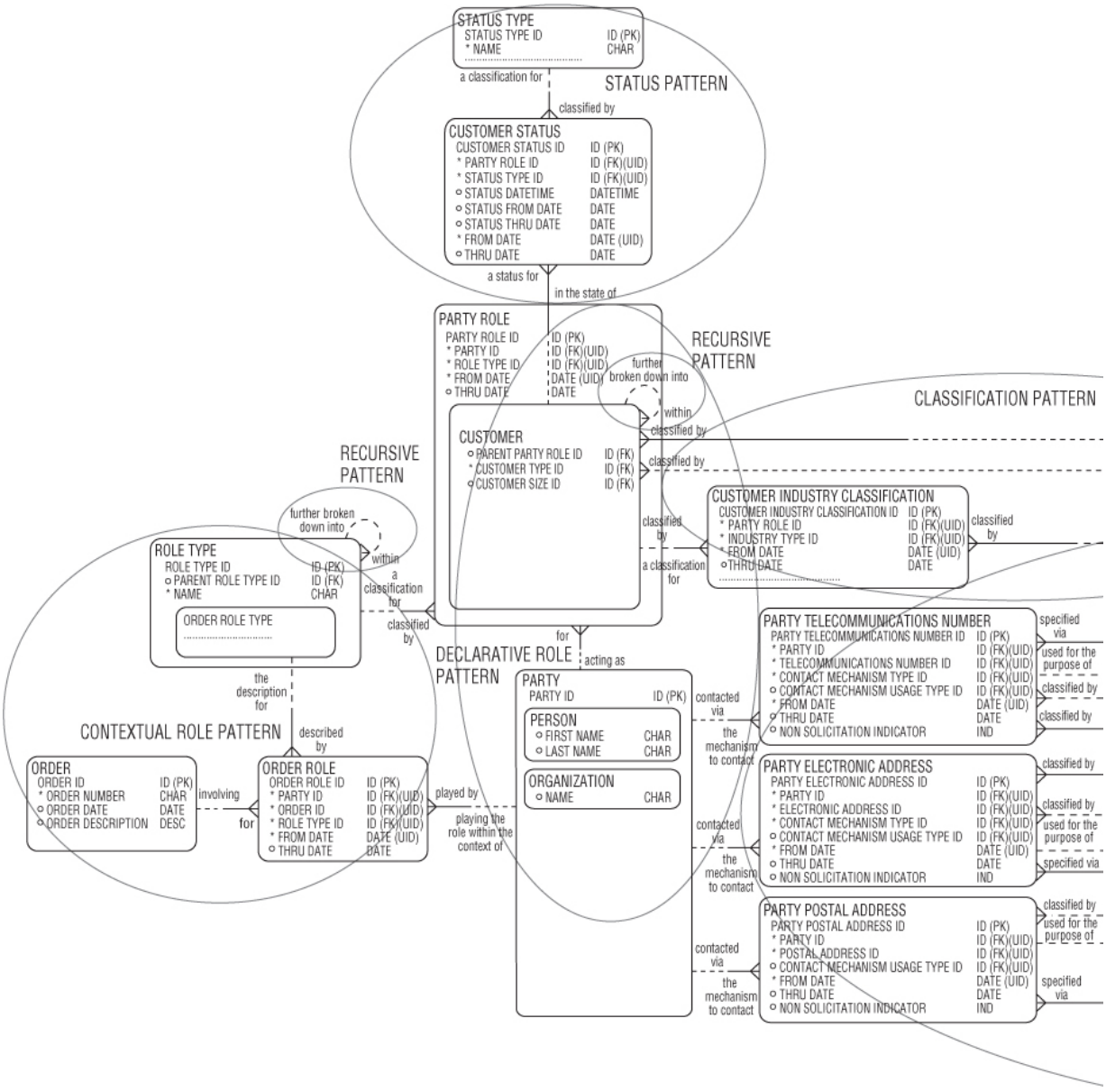
How Do These Models Work?

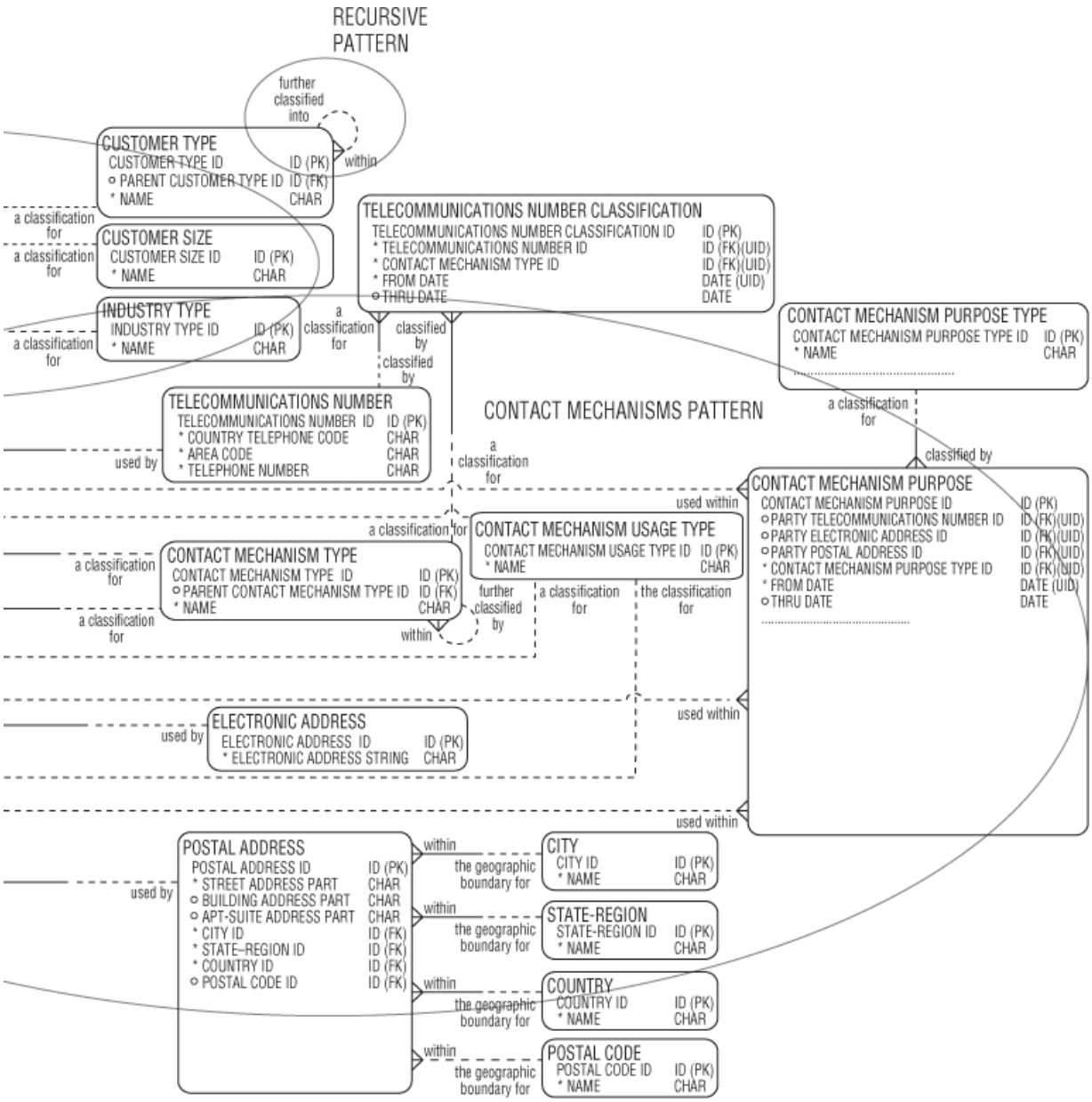
After looking at the model described in Figure 9.2, the data team decided to apply a more flexible approach to the application data model. They first confirmed the scope of the data based on Figure 9.2. Second, they discussed with

subject matter experts (both business and IT) the functionality that was needed for developing a full-blown production application. Based on this discussion, they decided to exchange some simplicity with flexibility. To do this, they replaced some of the level 1 and level 2 data model pattern structures seen in Figure 9.2 with the level 2 and level 3 data model structures that are shown in Figure 9.3 to create a much more flexible application data model.

The subject matter experts agreed that a CUSTOMER may have more than one status at a given point in time. For example, a CUSTOMER may be “Active” and “Under investigation” at the same time. To support this need, the data team replaced the Level 2 Status Pattern seen in Figure 9.2 with the Level 3 Status Pattern seen in Figure 9.3, which can be seen at the top of the diagram. This new structure supported the need for a CUSTOMER to have multiple STATUS TYPE(s) at the same time.

Figure 9.3 Application data model


STATUS TYPE
STATUS TYPE ID ID (PK)
* NAME CHAR
a classification for
STATUS PATTERN
classified by
CUSTOMER STATUS
CUSTOMER STATUS ID ID (PK)
* PARTY ROLE ID ID (FK)(UID)
* STATUS TYPE ID ID (FK)(UID)
STATUS DATETIME DATETIME
STATUS FROM DATE DATE
STATUS THRU DATE DATE
* FROM DATE DATE (UID)
THRU DATE DATE
a status for
in the state of
PARTY ROLE
PARTY ROLE ID ID (PK)
* PARTY ID ID (FK)(UID)
* ROLE TYPE ID ID (FK)(UID)
* FROM DATE DATE (UID)
THRU DATE DATE
further broken down into
RECURSIVE PATTERN
within
classified by
CUSTOMER
PARENT PARTY ROLE ID ID (FK)
* CUSTOMER TYPE ID ID (FK)
CUSTOMER SIZE ID ID (FK)
classified by
CLASSIFICATION PATTERN
CUSTOMER INDUSTRY CLASSIFICATION
CUSTOMER INDUSTRY CLASSIFICATION ID ID (PK)
* PARTY ROLE ID ID (FK)(UID)
* INDUSTRY TYPE ID ID (FK)(UID)
* FROM DATE DATE (UID)
THRU DATE DATE
classified by
classified by
a classification for
RECURSIVE PATTERN
further broken down into
ROLE TYPE
ROLE TYPE ID ID (PK)
PARENT ROLE TYPE ID ID (FK)
* NAME CHAR
ORDER ROLE TYPE
within
a classification for
classified by
for
DECLARATIVE ROLE PATTERN
acting as
the description for
CONTEXTUAL ROLE PATTERN
described by
PARTY
PARTY ID ID (PK)
PERSON
FIRST NAME CHAR
LAST NAME CHAR
ORGANIZATION
NAME CHAR
contacted via
the mechanism to contact
PARTY TELECOMMUNICATIONS NUMBER
PARTY TELECOMMUNICATIONS NUMBER ID ID (PK)
* PARTY ID ID (FK)(UID)
* TELECOMMUNICATIONS NUMBER ID ID (FK)(UID)
* CONTACT MECHANISM TYPE ID ID (FK)(UID)
CONTACT MECHANISM USAGE TYPE ID ID (FK)(UID)
* FROM DATE DATE (UID)
THRU DATE DATE
NON SOLICITATION INDICATOR IND
specified via
used for the purpose of
classified by
classified by
PARTY ELECTRONIC ADDRESS
PARTY ELECTRONIC ADDRESS ID ID (PK)
* PARTY ID ID (FK)(UID)
* ELECTRONIC ADDRESS ID ID (FK)(UID)
* CONTACT MECHANISM TYPE ID ID (FK)(UID)
CONTACT MECHANISM USAGE TYPE ID ID (FK)(UID)
* FROM DATE DATE (UID)
THRU DATE DATE
NON SOLICITATION INDICATOR IND
classified by
classified by
used for the purpose of
specified via
PARTY POSTAL ADDRESS
PARTY POSTAL ADDRESS ID ID (PK)
* PARTY ID ID (FK)(UID)
* POSTAL ADDRESS ID ID (FK)(UID)
CONTACT MECHANISM USAGE TYPE ID ID (FK)(UID)
* FROM DATE DATE (UID)
THRU DATE DATE
NON SOLICITATION INDICATOR IND
classified by
used for the purpose of
specified via
ORDER
ORDER ID ID (PK)
* ORDER NUMBER CHAR
ORDER DATE DATE
ORDER DESCRIPTION DESC
involving
for
ORDER ROLE
ORDER ROLE ID ID (PK)
* PARTY ID ID (FK)(UID)
* ORDER ID ID (FK)(UID)
* ROLE TYPE ID ID (FK)(UID)
* FROM DATE DATE (UID)
THRU DATE DATE
played by
playing the role within the context of



Another point raised by the data team was that the hierarchical structure described in Figure 9.2 may not support the future needs of their enterprise. They pointed out that many of their customers have more than two levels in their customer hierarchies, such as parent company, subsidiaries, divisions, and departments. To support this need, they replaced the Level 1 Recursive Pattern (a simple hierarchy) that was shown in Figure 9.2 ("each PARENT COMPANY *may be* **the parent of** one or more

CUSTOMER(s)") with the Level 2 Recursive Pattern in Figure 9.3 that shows that "each CUSTOMER *may be* **further broken down into** one or more CUSTOMER(s) (who may then also be broken down into one or more customers and so on)."

Because the data team has now captured the hierarchical relationship between PARENT COMPANY and CUSTOMER with the recursive relationship around CUSTOMER, they no longer need to capture the PARENT COMPANY entity or any of its attributes or relationships because they have captured them in the CUSTOMER entity. This not only helps to allow any number of levels in the customer hierarchy, but it also consolidates data about customers and parent companies. All classifications, statuses, and contextual roles can now be maintained for just the CUSTOMER entity instead of for both the PARENT COMPANY and CUSTOMER. This allows the data team to consolidate some of the relationships that were in Figure 9.2, such as the two relationships to CUSTOMER SIZE and the two relationships to the ORDER that specify the bill-to role. A significant benefit in using the recursive relationship in Figure 9.3 is that if there is a need to change the hierarchy (for example, to parent company, subsidiary, and division), then all the classifications, statuses, and contextual roles can remain intact without the need to change the data model.

With an eye to the future, the data team also added the concept of PARTY and PARTY ROLE to support declarative roles. This is to standardize capturing common PARTY, ORGANIZATION, and PERSON information (such as **name** [the name of an organization], **first name**, and **last name** [the name of a person]) in one place, and to allow a party to play any number of roles without redundantly capturing the same data for each role that a party plays. So, for instance, the same party that plays the role of a customer may also play the role of partner and/or supplier, and the party data

can be captured just once. Though this is not a requirement of the current application, the team recognized that this could be a consideration for the future. For example, their enterprise may want to give special consideration to customers that were also partners. Therefore, the model includes CUSTOMER and any other declarative roles within PARTY ROLE with a ROLE TYPE classification. This Level 3 Declarative Role Pattern allows the data team to capture the common attributes and relationships for all roles in PARTY ROLE.

Another major change to the model is the addition of the Level 3 Contextual Role Pattern. The IT subject matter experts of Sands Distribution agreed that a flexible way to capture all of the involved PARTY(s) in ORDER(s) would benefit the application in the long run. They were unsure of what new order roles may exist in the future, but they were sure that the enterprise would be creating new roles as new processes, rules, and regulations developed over time. With this in mind, the data team replaced the three relationships in Figure 9.2 that followed the Level 2 Contextual Role Pattern (**billed-to**, **shipped-to**, and **ordered for use by**) with the Level 3 Contextual Role Pattern made up of the associative entity ORDER ROLE that is **played by** a PARTY, **for** an ORDER, and is also **described** by an ORDER ROLE TYPE, which is a subtype of ROLE TYPE. Thus, if there were additional roles for the order that emerged such as “Customer contact,” “Person placing order,” “Person taking order,” “Installer,” “Salesperson,” “Quality assurer,” and so on, these could be accommodated without changing the model.

Still another change between the prototype data model and the application data model is the replacement of the Level 1 Contact Mechanism Pattern with the Level 2 Contact Mechanism Pattern. The subject matter experts for Sands Distribution pointed out that the same CUSTOMER instance

could be related to many different TELECOMMUNICATIONS NUMBER(s), POSTAL ADDRESS(es), or ELECTRONIC ADDRESS(es) (for example, email addresses). Also, they found that some of the phone numbers, mailing addresses, and email addresses were shared by multiple customers, especially when they were individual customers that were within a subdivision of the same company. The data team concluded that they needed to capture the CONTACT MECHANISM PURPOSE(s) and CONTACT MECHANISM USAGE TYPE for each telecommunications number, electronic address, and postal address for each party. The subject matter experts told a story of a salesperson who called a senior partner of a client on his personal phone number (which the client had asked not to use), and the salesperson tried to solicit new business; this caused Sands Distribution to nearly lose this account. Hence, they had the need to capture the proper purpose and usage for various contact mechanisms. Notice that aside from being able to classify contact mechanisms with purposes (“Bill to”) and usages (“Personal”), the data team also used a **non solicitation indicator** attribute (this was discussed at the end of Chapter 7) for PARTY TELECOMMUNICATIONS NUMBER, PARTY ELECTRONIC ADDRESS, and PARTY POSTAL ADDRESS to identify any party's contact mechanism that should not be called for sales purposes.

Why Do We Do It This Way?

Building an application data model requires a data professional to balance expressed requirements with future requirements, some of which may not be obvious to the enterprise or the application development team. In this application data model, the data team had an intention of balancing the use of more specific data model patterns with more generalized models when needed. This section

explains many of the reasons why we used the approach in Figure 9.3.

A full-blown application for a customer information application usually requires a flexible model that can accommodate many current and future needs. Therefore, the data team needed to replace some of the lower-level patterns with higher-level patterns that offered this flexibility. For example, the hierarchical structure shown in Figure 9.2 for capturing the PARENT COMPANY(s) relationship to CUSTOMER was very specific. It stated that "each PARENT COMPANY may be the parent of one or more CUSTOMER(s)." What happens if a CUSTOMER has more than two levels in their organization structure? Most large enterprises are multilayered. To ensure that the data model can handle this, the data team used the Level 2 Recursive Pattern, which provided some flexibility in creating the various multilayered hierarchies (and aggregations) needed for many Sands Distribution customers. Likewise, the flexible data model patterns used in Figure 9.3 for declarative roles, contextual roles, statuses, and contact mechanisms provides a means to meet a great variety of current and future needs.

One consideration for developing an application data model in a consistent fashion is the ability for the application data to integrate with other data in the enterprise (as well as with external data). The team can facilitate integration by using consistent patterns among applications and by using the same type of patterns in the enterprise data model and in a data model that supports master data. For example, the data team decided that they needed the Level 3 Declarative Role Pattern instead of the specific level 1 pattern seen in Figures 9.1 and 9.2. Though there are many good arguments for using this pattern (which we discussed in this chapter and that are discussed in detail in *The Data Model Resource Book*, Volume 1,

Revised Edition [Wiley, 2001]), a compelling reason for using this pattern is that many of the other data models in the enterprise use the PARTY, PERSON, and ORGANIZATION with PARTY ROLE(s) concept. In fact, many third-party tools use it (for example, as of the time of this writing, many of the most popular Enterprise Resource Planning (ERP) packages use this 'party' concept and have a PARTY entity in their logical data model). Thus, if you use the PARTY/PARTY ROLE concept, you might be able to more easily integrate external data into your system and exchange data more easily among various systems.

A consideration that is often overlooked is building up a knowledge base among various people in your enterprise. Once a data professional, programmer, or power user understands a pattern, they can apply it in many places in the enterprise, and people can more easily understand the underpinnings of various systems. For example, if a Level 3 Classification Pattern is used in your order entry system, and the same Level 3 Classification Pattern is used in your logistics reporting system and financial management system, then it is easier to build the knowledge base about how these systems work. The functionality for each of these systems may be different, but the building blocks of the system can be the same.

The application data model needed to meet the specific needs for that customer information application. The business subject matter experts said they were very sure that they needed to categorize a CUSTOMER only by size, type, and industry. They knew with relative certainty that there were no other needs for categorizing a CUSTOMER in this specific application. Because of this certainty, the data team stuck with the level 2 pattern that was created for the initial prototype and modeled in Figure 9.2 and decided not to use the more generalized (and complex) Level 3 Classification Pattern. However, we suggest that this type of

decision be approached with caution because we have experienced scenarios in which it seemed that the classifications were known, then later other classifications were needed for the application. For example, there may be needs in the future for other classifications, such as minority–owned businesses and customer market segment classifications.

A data model for applications should ideally be built with consideration for future requirements so that it can handle the changing needs of the enterprise. In our experience, application development is only a small fraction of the cost when compared to application maintenance, and a significant application development cost occurs when the database design needs to change. For example, the data team discovered that the business and the IT subject matter experts knew about three different contextual roles (shipped to, billed to, and ordered for end user). But they were also sure that there would be new roles that they had not discovered yet. This was why the data team replaced the specific Level 2 Contextual Role Pattern in the prototype with the flexible Level 3 Contextual Role Pattern for the production application data model.

What Are the Strengths of Using Patterns for the Solution?

The strengths of using patterns in application data models are as follows:

- **Balance of specific needs of the application with generalized patterns to provide flexibility:** The application data model seen in Figure 9.3 provides a nice balance between the various needs of the business. It uses level 2 patterns for classifications and contact mechanisms because the data team felt that this addressed the specific needs for these data areas. For

example, by using the Level 2 Classification Pattern, the model maintains the specific need that there may be only one customer size for a customer and that the customer may be classified into only one customer type. It uses level 3 patterns for declarative roles, contextual roles (to ORDER), and statuses to accommodate more flexibility for these data areas because the subject matter experts were not sure what future requirements would be needed in these areas. By using mostly level 3 patterns (and a couple of level 2 patterns), the model provides the ability to add new types of declarative roles, new contextual roles for orders, new types of statuses, and new types, purposes, and usages of contact mechanisms, without needing to change to model.

- **Common, integrated solution:** This solution can serve as a basis for integrating data into a relatively common data structure; for example, this data model has entities for PERSON and ORGANIZATION that are most likely very consistent with the enterprise data model and perhaps other applications.
- **Consistency:** The model is using the patterns in a plug-and-play way that promotes consistency. The beauty of using patterns (as data model components) is that when flexibility is needed, the data team can replace one pattern for a more flexible version of the same pattern, and when the business requires a more specific solution, they can retain (or use a lower-level pattern to model) the more specific pattern. Thus, re-using the same types of constructs in patterns leads to very similar types of data model structures and more standardization.

What Are the Weaknesses of Using Patterns for the Solution?

The weaknesses of applying patterns for applications are as follows:

- **Inflexibility in some parts and complexity in other parts:** There could be drawbacks regarding inflexibility when more specific patterns are used and complexity when more generalized models are used. The data team had to find the right balance between all of the different factors that affect that application data model. This is not easy to do and requires experience.
- **Multiple levels of patterns and therefore lack of one common style of modeling:** Because the different levels of patterns are interchangeable, these types of patterns can result in multiple styles of data modeling—that is, more specific level 2 patterns and more generalized level 3 patterns. Some data modelers may be uncomfortable with what seems like mixed styles of modeling. Data modelers may make the point that by using the same style and level of pattern in the models, reusable routines can more easily be developed across applications. For example, in Service-Oriented Architecture (SOA), there could be very common and similar services built to call various types of data classifications, and if they are all modeled using a level 3 pattern, this could help create consistency in how the services are developed.
- **Lack of understandability when using more generalized patterns:** These types of patterns may be too complicated for business representatives, depending on their experience working with data models. Specific models are normally a better way to communicate the data requirements to nontechnical audiences than are generalized models because specific models are easier to understand.
- **Lack of some business rules specifications when using more generalized patterns:** Specific patterns

result in models that show many more business rules and specifications than generalized patterns, in the sense that they can capture much more of the data requirements inherent to the entities and relationships, such as relationship optionality, relationship cardinality, and more specific attributes and relationships. For example, using a level 3 pattern for contextual roles via the ORDER ROLE entity does not maintain if a specific role can only exist once for an order, such as allowing one and only one “bill to” customer. Maintaining certain business rules as part of the data model needs to be balanced against the need for data model solutions that can support the changing requirements of the business.

Synopsis

In this section, you saw the creation of an application data model to support the needs of Sands Distribution in relation to its customer information application. You saw how the data team evolved the prototype model seen in Figure 9.2 into a full-blown application data model. They did this by replacing some of the specific patterns (level 1 and level 2) with more flexible patterns (level 2 and level 3).

The data team decided to balance accommodating specific needs with a flexible style of modeling. Meeting specific needs allowed the data model to be customer–tailored to the application. Using a flexible style of modeling allowed new data requirements to be added more easily. This meant that they used different levels of patterns depending on current and any anticipated future requirements. For example, they added the PARTY and PARTY ROLE level 3 concept to allow new declarative roles to be added more easily in the future and also to facilitate capturing common PERSON and ORGANIZATION information only once, regardless of the number of roles that a party may play. On the other hand, they maintained more specific

classifications of CUSTOMER because the business said that these were the only way that the customer application would classify customer data.

Some data modelers may object to using different styles and levels of patterns within a single model and argue that if the same types of patterns are used, this can promote reusability of common functions to access data and a consistent way to manage data. If this is the case, the modeler can alternatively choose to develop the application data model all at the same level to promote even more consistency.

Enterprise Data Models

Enterprise data models have become much more important artifacts over the past 10 years, coinciding with the realization by many enterprises that information is a valuable asset. Companies such as Netflix and Wal-Mart use their data to gain a competitive advantage.(1) A core part in their gaining this advantage is having a common, consistent view of their data landscape.

Enterprise data models help enterprises comprehend their data landscape. An enterprise data model can provide:

- **A standard way to view core entities, attributes, and relationships in the enterprise:** For example, in a particular enterprise data model, such as the example shown in Figure 9.4 on page 496, some of the core entities are PARTY, PARTY ROLE, PARTY RELATIONSHIP, CONTACT MECHANISM, ORDER, and PRODUCT, and the enterprise data model shows some of the main attributes and relationships for these entities. There also may be other core entities not shown in Figure 9.4 that may be a part of an enterprise data model, such as SHIPMENT, INVOICE, PAYMENT, and WORK EFFORT. The enterprise data model can illustrate common ways to

handle relationships, such as showing that a many-to-many relationship may exist between an ORDER ITEM and SHIPMENT ITEM. Though some specific applications may have different terms for these or different needs regarding their relationship, the enterprise data model provides a common model that is applicable for many applications, thus providing a guide for developing consistent data models throughout the enterprise. Using patterns helps this approach because patterns provide a standard way for constructing common data structures such as classifications, statuses, declarative roles, contextual roles, and so on. These patterns can be used to create a very consistent enterprise data model for your enterprise.

- **The same flexible data model construct may be used for many different applications:** Enterprise data models tend to use many generalized structures so that they may be applicable for a wide range of applications. These generalized data model structures are often examples of patterns. For example, a specific application may model customers and their parent enterprises via a CUSTOMER entity with a foreign key that relates to a PARENT CUSTOMER entity. However, the enterprise model may show a more generalized data model construct allowing any party in any role to be related to any other party in any other role via a PARTY RELATIONSHIP construct (Level 3 Recursive Pattern). Thus, the enterprise data model meets the need for that specific application while also meeting the needs of many other applications.
- **A method to guide and assure the quality of entities, attributes, and relationships in an application and to show where they may not be correct:** The enterprise data model can be used as way to assure the quality of and verify an application data

model. In this way, the enterprise data model serves as a checkpoint. For example, an application may show a model where a PRODUCT is maintained at a FACILITY. The enterprise data model can point out that according to that enterprise's view of the data, a PRODUCT is actually related to INVENTORY ITEM (representing a specific instance of a product—for example, product XYZ, serial number #1458), and the INVENTORY ITEM is the item that is actually stored in a FACILITY, whereas the PRODUCT is just the catalogued item and, as such, is not really physically stored at all. Again, patterns help greatly when used to cross–check models. The patterns provide standard ways to do common things. For example, the enterprise data model can point out the enterprise's view of managing classifications or contextual roles.

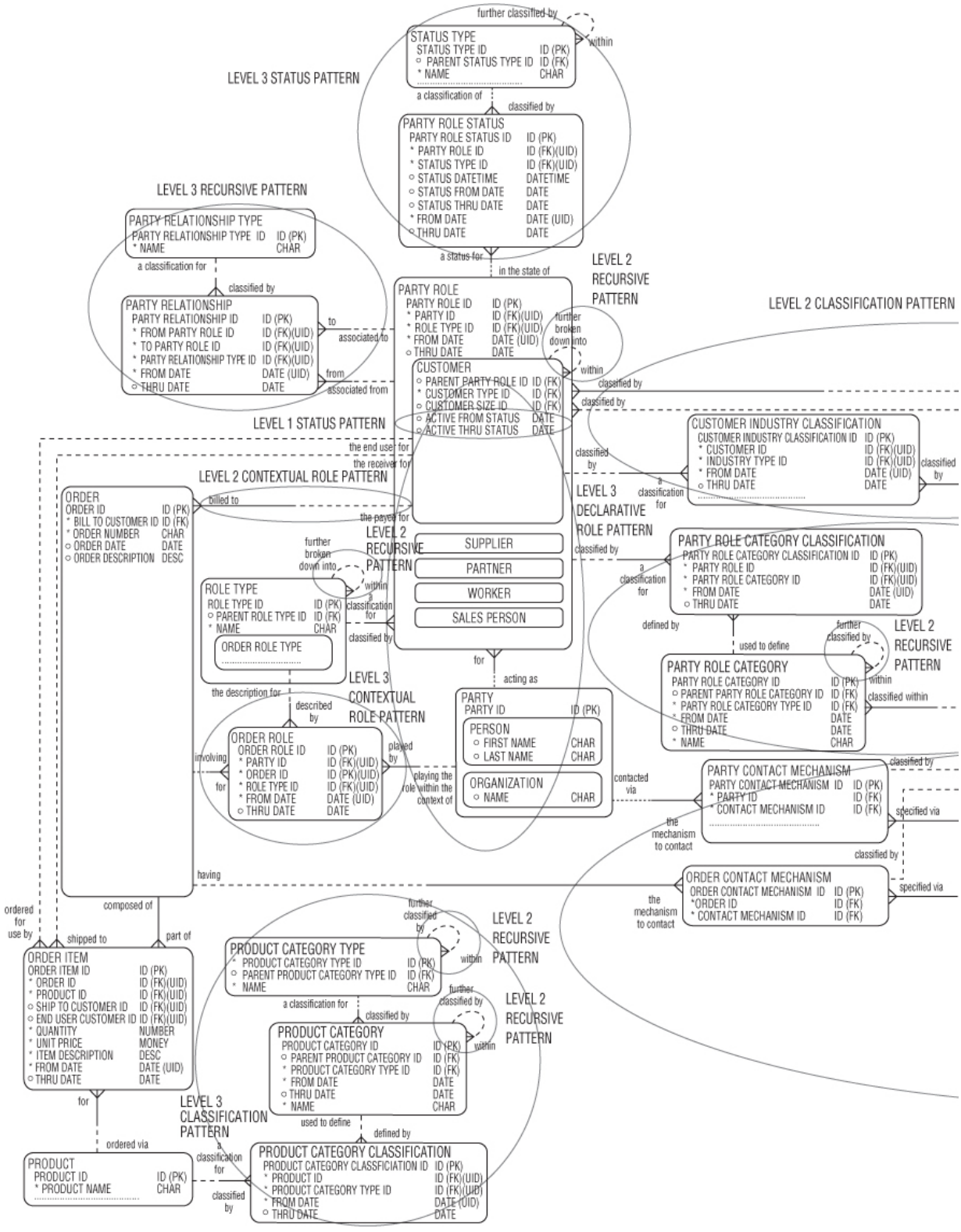
- **Different alternative ways to model the same concept (if the enterprise data model uses the technique of showing data model alternatives within the same model):** One approach that we have sometimes used in developing enterprise data models is to model the same data model constructs in both a generalized fashion (using level 3 or 4 patterns) as well as model it specifically (using level 1 or 2 patterns). For example, the enterprise data model may show a generalized way to support the classification of all PARTY ROLE(s) and also show specific classification entities such as CUSTOMER SIZE, INDUSTRY TYPE, and CUSTOMER TYPE for the CUSTOMER (a PARTY ROLE). Thus you can have various views of the enterprise data model that can be shown to different audiences. For example, a data architect may want to view a generalized style for a part of the data model in order to make sure it is flexible enough, whereas a business representative may want to see a specific view of that

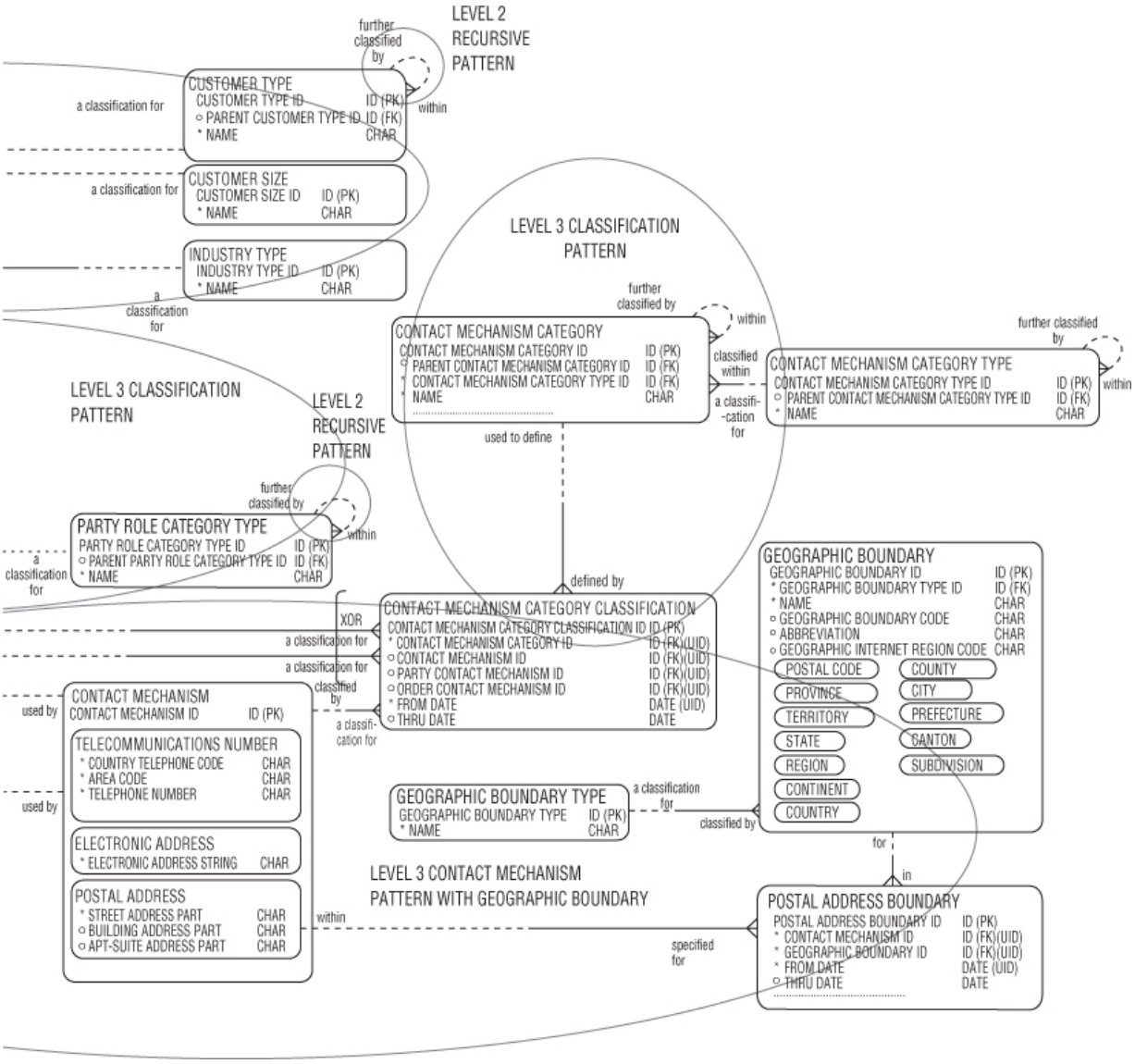
same portion of the model in order to make sure their requirements are met. This technique also provides multiple alternatives for reusing constructs from the enterprise data model in an application because different applications have different needs.

Note

The approach for an enterprise data model that we are illustrating is to show alternative ways to model the same type of data: one using a specific method to model and one using a much more generalized way to model. This is not the same as saying it is okay to maintain the same data redundantly. Consider the enterprise data model as providing standards for the enterprise but also offering alternatives. For example, if you need a specific solution for classification of CUSTOMER, you may choose either a specific or a more generalized classification data model structure. A third alternative would be to use both the specific and generalized data model structures by using the specific pattern for certain critical data and the more generalized model for other data, especially for new types of data that may emerge over time. If you do this, you should be very careful to specify what data and under what circumstances you use the specific data model structure versus the generalized data model structure. For example, we may use the Level 1 Status Pattern to model an **order date** (a critical piece of data) and the Level 3 Status Pattern for all the other order statuses. Another way to put this would be to say that the enterprise data model may sometimes show a specific and a generalized data model construct to accommodate the same type of data requirement, such as modeling classification of CUSTOMER or PARTY ROLE(s). We don't consider this to be recommending the maintenance of redundant data, because we don't advocate capturing the same instances of data in both ways. We feel that providing valid alternative data modeling options can be extremely useful in offering standardization while allowing for options in different situations.

Figure 9.4 Subsection of the enterprise data model


LEVEL 3 STATUS PATTERN
STATUS TYPE
STATUS TYPE ID ID (PK)
PARENT STATUS TYPE ID ID (FK)
NAME CHAR
further classified by
within
a classification of
classified by
PARTY ROLE STATUS
PARTY ROLE STATUS ID ID (PK)
PARTY ROLE ID ID (FK)(UID)
STATUS TYPE ID ID (FK)(UID)
STATUS DATETIME DATETIME
STATUS FROM DATE DATE
STATUS THRU DATE DATE
FROM DATE DATE (UID)
THRU DATE DATE
a status for
in the state of
LEVEL 3 RECURSIVE PATTERN
PARTY RELATIONSHIP TYPE
PARTY RELATIONSHIP TYPE ID ID (PK)
NAME CHAR
a classification for
classified by
PARTY RELATIONSHIP
PARTY RELATIONSHIP ID ID (PK)
FROM PARTY ROLE ID ID (FK)(UID)
TO PARTY ROLE ID ID (FK)(UID)
PARTY RELATIONSHIP TYPE ID ID (FK)(UID)
FROM DATE DATE (UID)
THRU DATE DATE
to
associated to
from
associated from
PARTY ROLE
PARTY ROLE ID ID (PK)
PARTY ID ID (FK)(UID)
ROLE TYPE ID ID (FK)(UID)
FROM DATE DATE (UID)
THRU DATE DATE
further broken down into
LEVEL 2 RECURSIVE PATTERN
LEVEL 2 CLASSIFICATION PATTERN
CUSTOMER
PARENT PARTY ROLE ID ID (FK)
CUSTOMER TYPE ID ID (FK)
CUSTOMER SIZE ID ID (FK)
ACTIVE FROM STATUS DATE
ACTIVE THRU STATUS DATE
within
classified by
classified by
LEVEL 1 STATUS PATTERN
the end user for
the receiver for
CUSTOMER INDUSTRY CLASSIFICATION
CUSTOMER INDUSTRY CLASSIFICATION ID ID (PK)
CUSTOMER ID ID (FK)(UID)
INDUSTRY TYPE ID ID (FK)(UID)
FROM DATE DATE (UID)
THRU DATE DATE
classified by
a classification for
LEVEL 2 CONTEXTUAL ROLE PATTERN
ORDER
ORDER ID ID (PK)
BILL TO CUSTOMER ID ID (FK)
ORDER NUMBER CHAR
ORDER DATE DATE
ORDER DESCRIPTION DESC
billed to
the payer for
LEVEL 2 RECURSIVE PATTERN
SUPPLIER
PARTNER
WORKER
SALES PERSON
LEVEL 3 DECLARATIVE ROLE PATTERN
PARTY ROLE CATEGORY CLASSIFICATION
PARTY ROLE CATEGORY CLASSIFICATION ID ID (PK)
PARTY ROLE ID ID (FK)(UID)
PARTY ROLE CATEGORY ID ID (FK)(UID)
FROM DATE DATE (UID)
THRU DATE DATE
classified by
a classification for
defined by
used to define
further classified by
LEVEL 2 RECURSIVE PATTERN
PARTY ROLE CATEGORY
PARTY ROLE CATEGORY ID ID (PK)
PARENT PARTY ROLE CATEGORY ID ID (FK)
PARTY ROLE CATEGORY TYPE ID ID (FK)
FROM DATE DATE
THRU DATE DATE
NAME CHAR
within
classified within
further broken down into
within
ROLE TYPE
ROLE TYPE ID ID (PK)
PARENT ROLE TYPE ID ID (FK)
NAME CHAR
ORDER ROLE TYPE
a classification for
classified by
for
acting as
LEVEL 3 CONTEXTUAL ROLE PATTERN
the description for
described by
ORDER ROLE
ORDER ROLE ID ID (PK)
PARTY ID ID (FK)(UID)
ORDER ID ID (PK)(UID)
ROLE TYPE ID ID (FK)(UID)
FROM DATE DATE (UID)
THRU DATE DATE
involving
for
played by
playing the role within the context of
PARTY
PARTY ID ID (PK)
PERSON
FIRST NAME CHAR
LAST NAME CHAR
ORGANIZATION
NAME CHAR
contacted via
PARTY CONTACT MECHANISM
PARTY CONTACT MECHANISM ID ID (PK)
PARTY ID ID (FK)
CONTACT MECHANISM ID ID (FK)
classified by
specified via
the mechanism to contact
having
ORDER CONTACT MECHANISM
ORDER CONTACT MECHANISM ID ID (PK)
ORDER ID ID (FK)
CONTACT MECHANISM ID ID (FK)
classified by
specified via
the mechanism to contact
composed of
ordered for use by
shipped to
part of
ORDER ITEM
ORDER ITEM ID ID (PK)
ORDER ID ID (FK)(UID)
PRODUCT ID ID (FK)(UID)
SHIP TO CUSTOMER ID ID (FK)(UID)
END USER CUSTOMER ID ID (FK)(UID)
QUANTITY NUMBER
UNIT PRICE MONEY
ITEM DESCRIPTION DESC
FROM DATE DATE (UID)
THRU DATE DATE
for
ordered via
PRODUCT
PRODUCT ID ID (PK)
PRODUCT NAME CHAR
LEVEL 2 RECURSIVE PATTERN
PRODUCT CATEGORY TYPE
PRODUCT CATEGORY TYPE ID ID (PK)
PARENT PRODUCT CATEGORY TYPE ID ID (FK)
NAME CHAR
further classified by
within
a classification for
classified by
LEVEL 2 RECURSIVE PATTERN
PRODUCT CATEGORY
PRODUCT CATEGORY ID ID (PK)
PARENT PRODUCT CATEGORY ID ID (FK)
PRODUCT CATEGORY TYPE ID ID (FK)
FROM DATE DATE
THRU DATE DATE
NAME CHAR
further classified by
within
used to define
defined by
LEVEL 3 CLASSIFICATION PATTERN
a classification for
classified by
PRODUCT CATEGORY CLASSIFICATION
PRODUCT CATEGORY CLASSFICIATION ID ID (PK)
PRODUCT ID ID (FK)(UID)
PRODUCT CATEGORY TYPE ID ID (FK)(UID)
FROM DATE DATE (UID)
THRU DATE DATE



The creation of an enterprise data model is often a very large and difficult task. Collecting and understanding all of the different data artifacts in an enterprise require organization and methodology. We have often seen large enterprise data models that lack consistency across various parts of the model. The patterns can help because they remove the guesswork from deciding how to model very common constructs that apply to any form of data, and they provide a consistent way and style of modeling similar types of data. For example, data modelers can use the same pattern when modeling a classification for a PRODUCT as

the classification for a CUSTOMER, ASSET, WORK EFFORT, or any other entity that requires many types of classifications; they just need to change the names because they are dealing with a different type of data. A status for an ORDER can be modeled using the same pattern as the status for SHIPMENT, INVOICE, PAYMENT, WORK EFFORT, and/or PRODUCT. The patterns allow the data modelers to concentrate on more unique aspects of the data model for their enterprise, such as how to model their product-costing structures, rather than on how to best model recursive relationships or contextual roles. The modeler can also decide which level of pattern to use in various circumstances and recognize where similar data model constructs are needed. Thus the patterns can provide a great deal of consistency by allowing modelers to use consistent data model structures across the enterprise data model and across various application models when modeling the same types of data.

The Scenario for This Model

The Sands Distribution operating committee realized that they have various needs across the enterprise and that their information is a key corporate asset to be managed. They appreciated that in developing the customer information application, many other applications also have the same types of data requirements regarding modeling customer data. By using consistent data modeling structures, they could improve communications, reduce maintenance costs, simplify interfaces, and improve data quality. So they asked the data team to provide a strategy for managing their data effectively and consistently.

As part of the enterprise data strategy, the data team decided to create an enterprise data model showing the nature of the data assets in their enterprise. Two purposes for the enterprise data model were:

- First, to describe the current and future data landscape as Sands Distribution sees it across the enterprise as a whole.
- Second to provide reusable alternatives for any effort developing data models, thus providing a jump-start to other efforts, providing more consistency to help integration, and providing a quality assurance checkpoint for other data modeling efforts.

Based on the requirements of the business and given a mandate from the operating committee of Sands Distribution, the data team started the development by using patterns to help provide a broad-brush enterprise data model. Figure 9.4 shows part of their enterprise data model that focuses on customers as well as customer-related orders and products.

How Do These Models Work?

In the previous sections of this chapter, you saw that different levels of patterns are used depending on the requirements of the business. For example, in the first section of the chapter when we created a simple scope statement, we utilized level 1 and level 2 patterns because they showed the specific attributes and entities that were of interest to the business. In contrast, in the next section where we needed to create a full-blown application, we used level 2 and level 3 patterns for the most part because the business required the data production application to be more flexible. Now, in the enterprise data model, we need to take a different approach.

In Figure 9.4, you can see a hybrid or mixed approach, modeling the same type of data multiple ways! So, what does this mean? In this approach you may use level 1, level 2, and level 3 patterns and use them in the same model, sometimes to even show two different alternatives for the same construct. For example, you can see that the

enterprise data model captured CUSTOMER TYPE, CUSTOMER SIZE, and CUSTOMER INDUSTRY for classifying CUSTOMER. You might recall this as the approach that the application development team used when creating the customer information application as seen in Figure 9.3. The enterprise has said that it is very common to classify customers by their type, size, and industry, and thus, the model shows a standard way of modeling this. You can also see in Figure 9.4 the PARTY ROLE CATEGORY CLASSIFICATION (and its associated category entities). This is the Level 3 Classification Pattern. This data model pattern provides a much more generalized way of modeling classifications that includes all the capabilities that the Level 2 Classification Pattern provides; however, it is much more comprehensive. It provides the capability to maintain any current or future type of categorization not only for customers, but for any type of PARTY ROLE. It also provides the capability to have hierarchies of categories and category types to be able to maintain higher and lower levels of classifications.

You can see a similar example in which the enterprise data model applies two different levels of a pattern. In Figure 9.4 you can see that “each CUSTOMER *may be* **further broken down into** one or more CUSTOMER(s).” This captures the CUSTOMER(s) internal organization structure using a Level 2 Recursive Pattern. You can also see that “each PARTY ROLE *may be* **associated from** one or more PARTY RELATIONSHIP(s) and also *may be* **associated to** one or more PARTY RELATIONSHIP(s).” This may then be related to a PARTY RELATIONSHIP TYPE of “Parent subsidiary relationship” or “Subsidiary division relationship.” Thus this Level 3 Recursive Pattern applied to PARTY RELATIONSHIP(s) can capture the hierarchy of CUSTOMER(s), as well as any other relationships between any other PARTY ROLE(s). The enterprise data model uses an approach to allow an

application to have two choices: a simple recursive relationship around customer or a very comprehensive way to relate any party within the context of one role to any other party role within the context of another role. Again, it is worth stating that we don't advocate redundantly capturing the same data. We are just showing different possible alternatives of modeling the same type of thing.

You can also see in Figure 9.4 an example of a mixed contextual role pattern. We have captured the bill-to customer for the ORDER using the Level 2 Contextual Role Pattern. We have also used the Level 3 Contextual Role Pattern using the ORDER ROLE, which can capture any number of parties involved in any manner with the role. This could maintain the salespeople for the order, the person taking the order, the person entering the order, the party quality assuring the order, and any number of additional roles that may be needed as time passes and as processes change.

What is interesting about this example is that the enterprise data model is really showing three different alternatives for implementation in this example:

- First, a development group can implement just the foreign key relationships to CUSTOMER of **billed to**, **shipped to**, and/or **ordered for end use by** (the level 2 pattern) if that is all they are interested in. (Incidentally, there could also be other level 2 contextual roles from the ORDER ITEM to the CUSTOMER such as **ship to customer** to show where specific order items were designated to be shipped.)
- Second, if a development group needs to capture every contextual role possible, they can consider implementing the Level 3 Contextual Role Pattern using the ORDER ROLE and associated relationships to PARTY ROLE TYPE and ORDER.

- Third, if they need to specifically capture the **billed to** customer relationship because this is a key relationship and also other ORDER ROLE(s) (and associated relationships), they can apply both of these data model structures, or in other words, use a hybrid pattern.(2)

The enterprise data model does not always have to show every alternative. Suppose that after some investigation across the enterprise, the data modeling team discovers that, as a whole, the enterprise wants to standardize on the more generalized Level 3 Contact Mechanism Pattern. In other words, they find a consensus within the enterprise that this is the way the enterprise views contact data. Therefore, there is no need to include the other alternatives in the enterprise data model.

There are different modeling styles and ways that the patterns may be applied to an enterprise data model. This scenario has illustrated one method of using a 'hybrid' approach that may maintain more than one level of pattern, even for the same data modeling construct. Other alternatives include the following:

- **Consistently using the same level of pattern (using level 1, level 2, or level 3 patterns) across the enterprise data model.**
- **Consistently modeling all data modeling constructs with both a specific and generalized pattern:** You can view this as a way to keep the enterprise data model more consistent; however, it may be overkill for data areas where the enterprise has decided that there will be one way to implement certain types of data model constructs. In the example, we illustrated the CONTACT MECHANISM pattern as one area that was already agreed upon enterprise-wide. Thus, creating two alternatives in the enterprise data model might create confusion because the enterprise

may not be advocating looking at alternatives for this type of data.

- **Creating different versions of the enterprise data model, for example, a version with level 2 patterns and another version with level 3 patterns:** Some would refer to the level 2 enterprise data model for a prototype or as a business data model designed for viewing by business representatives. The key purpose of this type of model would be to illustrate information requirements in an understandable fashion. Then there could be a separate (but synchronized) level 3 enterprise data model that serves as an architectural data model designed for an audience of data professionals such as architects. The key purpose of this type of model is to serve as a very flexible foundation for developing stable and reusable database constructs.

Note

Some data professionals also view a business data model as synonymous with a conceptual data model based on the 1970s ANSI Committee discussions regarding conceptual models. Unfortunately, there is no agreement in the data management industry regarding what a conceptual data model is. That is why we have chosen to use a new concept of levels 1, 2, and 3 to categorize these models. In our minds, the key purpose for a business data model, or in other words, a conceptual data model, is to describe information requirements in a very understandable fashion, and thus we will normally advocate using level 1 and level 2 patterns to accomplish this. But we don't discount using level 3 patterns also; it depends on your audience. Our mantra with business data models/conceptual data models is use whatever pattern or technique it takes to get the message across.

An ‘architectural’ data model is designed for a technical audience, employs normalization techniques, and is sometimes referred to as a logical data model. However, there is even disagreement in the industry on what a logical data model is and/or should be. Thus, in this book, by referring to models using levels, we can categorize the models according to level of

generalization and avoid misinterpretation of what we mean by a conceptual and/or logical data model.

Why Do We Do It This Way?

In a way, the enterprise data model has to be all things to all people. Therefore, including various views of the data landscape is helpful. The ideal situation for standardization is when consensus can be reached across the enterprise as whole; such was the case with the contact mechanism pattern in the previous example. Often that consensus can't be reached for many different reasons. (3) In those cases, it is often better to show alternative ways to model a concept. This is not the same as redundantly modeling the same data, which we do not recommend.

The patterns provide an effective tool to consistently model common structures such as recursive relationships, roles, or statuses. As you see in Figure 9.4, this enterprise data model also provides alternatives for modeling the same type of data, such as ways to classify by using different levels of the same pattern.

What Are the Strengths of Using Patterns for the Solution?

The strengths of using patterns in enterprise data models are as follows:

- **Quality data model constructs for common requirements:** Creating an enterprise data model is hard enough without having to worry about how to create effective ways to capture status, classifications, roles, and so on. By providing reusable patterns that have been field–tested, data modelers are free to concentrate on the hard work of modeling unusual information requirements or finding out how various types of core entities are related.

- **Consistency across the enterprise data model:** By using the same types of patterns for various data requirements, the enterprise data model is much more consistent. For example, if the same classification pattern is used throughout the enterprise data model it, provides a common, stable approach that people over time can trust more and more.
- **Leverage in providing alternatives in enterprise data modeling:** Regardless of the approach you choose in enterprise data modeling, we feel that it is especially important in this context to be able to offer alternative ways of modeling the same type of data requirement. Once familiar with the patterns, a data professional can weigh the strengths and weaknesses of the alternatives shown in the enterprise data model when deciding which alternative to apply for their application.
- **Time saved in the enterprise data modeling effort:** An enterprise data model can be a very time-consuming effort, and we have found that using patterns can significantly reduce this amount of time.

What Are the Weaknesses of Using Patterns for the Solution?

The weaknesses of using patterns for enterprise data models are as follows:

- **It could, at first, cause some confusion to have alternative patterns for the same type of data:** Reading the enterprise data model with its alternative patterns can be difficult initially, even for experienced data professionals, regardless of how patterns were used in the model. However, we have found that over time, people generally get used to and like the idea of being able to choose from a couple of options instead of

being forced into one way to model a specific type of data. Alternatively, you may chose not to mix different alternative patterns in your enterprise data model.

- **Different levels of patterns in the same data model can lead to inconsistency:** Because there are a lot of options for modelers, unless you use an approach that uses the same level of pattern across the enterprise data model, you may have inconsistencies in the ways that various types of data are modeled.

Synopsis

In this section, we described how you can use different patterns to start an enterprise data modeling effort. The patterns provide alternative ways to create common structures, such as roles, classifications, recursive relationships, statuses, and so on. This allows the data professionals to concentrate more on their unique data requirements such as their product cost structure or the relationship between ORDER(s) and SHIPMENT(s) for their enterprise.

The enterprise data model should show the landscape of the data structures that support the needs of the business. One possible approach for developing a data model is to maintain different alternatives that model the same data requirement(s). For example, classifications for customers were modeled two different ways in the enterprise data model: the Level 2 Classification Pattern and the Level 3 Classification Pattern. This approach to developing an enterprise data model provides a choice of one of the alternatives, thus providing some standardization without being too rigid because different applications have different needs. There are times that an application may choose to implement both of the styles shown in the enterprise data model. For example, an application may use a level 1 style of modeling for a critical attribute such as **order date** and

use the level 3 style for the many other status types of data. This is not the same as advocating you maintain the same data redundantly in both alternative data model structures, which we do not recommend at all. You also saw that some structures in the enterprise data model were consistent and agreed upon across the enterprise as a whole, such as contact mechanisms. In this case, there may not be a need to show different alternatives.

The enterprise model needs to be used properly. Think of it as a means to show generally accepted and standard data model structures, but sometimes it can also show alternatives. In these cases, data modelers can look at the enterprise data model and pick the alternative that best meets the needs of their application.

Data Warehouse Models

Each enterprise has its own unique requirements regarding its decision support and reporting needs. In general, the decision support environments provide information to illustrate trends, depict performance, and provide key business indicators so that the enterprise can make informed decisions. Many enterprises look to their data warehouses to provide these solutions. The question is, “How can patterns help you create, understand, and validate your decision support environment?”

This section of the chapter contains two subsections where we look at the two most common styles of modeling for data warehousing: a relational modeling approach and a star schema approach. The first subsection is based upon the approach that is advocated by Bill Inmon(4) and to which Claudia Imhoff has also contributed to a great deal.(5) The second subsection is based upon the approach advocated by Ralph Kimball and Margy Ross.(6) Of course, both of these approaches have their strengths and weaknesses, and

both approaches have many supporters and some detractors. We have successfully used both when building data warehouses, and we feel that the approach to use is very dependent on the goals and circumstances. We also feel that the patterns can help with either of the approaches, so the next two sections address how patterns can be used to help within each of these different data warehouse styles.

The Scenario for This Model

Sands Distribution successfully implemented the application database for customer information based on the application data model produced in Figure 9.3. It realized that there was a need for standardization and consistency across various applications, and thus it created the enterprise data model shown in Figure 9.4. It further realized that it could use the enterprise data model for its next effort, which was a data warehouse. The enterprise knew that using an enterprise data model as the basis for its data warehouse was a well-accepted industry practice.

The customer information application as well as many other applications had needs for operational and decision report reporting, and the application developers became overburdened with requests from many different business areas for reports and analysis of their customers. “What products are most popular, to whom, in what locations, and when?” and “What types of customers are buying what types of products?” are just two common questions that they were expected to answer. The enterprise felt there was a lack of response by the Information Technology (IT) department to its decision support needs. Senior IT and business management asked the data team to come up with solutions to address the growing decision support needs of the business, specifically in relation to customer data.

The data team decided to present two different solutions for the decision support needs of the business. One is based on a more relational approach (the ‘Inmon’ approach) for the core data warehouse model (although even in this approach star schemas are used for data marts), and one is based on a star schema style model (the ‘Kimball’ approach) for the core data warehouse model as well as for use in the data marts.

One of the key sources for the data warehouse application was the customer information application data model as represented by Figure 9.3. A physical database was created based on this data model where each entity and subtype translated directly into a relational table. The other input was the order entry system where the enterprise maintained customer orders and had entities such as CUSTOMER, ORDER, ORDER ITEM, PRODUCT, and PRODUCT CATEGORY, and their model (and database design) closely resembled the same type of entities, attributes, and relationships in the enterprise data model shown in Figure 9.4.

In addition to the application data model in Figure 9.3 and the order entry system, the key input for this model was the new decision support needs for the enterprise and for IT, as captured in focus groups and one-on-one meetings with business and IT subject matter experts. In the meetings with the business and with IT, the data team uncovered some requirements that went beyond the scope of what is captured in their customer information, order entry applications, and the enterprise data model. These new requirements were as follows:

- The business wanted to be able to report on order amounts and quantities by customer, customer category, customer status, product, product category, geographic boundaries, time, or any combination of these.

- The business wanted to the ability to classify its products in different ways and report on any future classifications. At the moment it classified products by product line, product type, and product family, but in the future it knew there would be other ways to classify products, such as “Product price range type” (for example, high priced, medium priced, low priced) and “Product usage type” classifications (for example, heavy duty, light duty).
- Other areas of the business felt that there were many more ways to classify a customer than just size, industry, and type. For example, they wanted to classify the customers by market sectors (for example, oil and gas, refining, and exploration companies) and by number of employees (for example, 0–1000, 1001–50,000, and greater than 50,000).
- They wanted to classify the order data by geographic boundary. For example, “Where are we getting the largest order volumes, and how do the various countries rate regarding the total amounts of orders?”
- IT senior management specified that whatever solution the data team came up with had to be flexible enough to meet future needs of the business (even if the business was not sure what those needs would be) and simple enough that the application team could quickly build reports and analytics.

The next two sections describe how the data team could use these patterns for two different approaches, namely:

- **Data Warehouse Data Models—Relational Approach** This approach uses the enterprise data model as the basis for a data warehouse data model. This data warehouse data model is then used as the basis for implementing an integrated data warehouse database design. After the data is integrated into a very flexible database structure, the data is then passed to

various data marts that are usually based on star schema designs. The data warehouse data model is used as the basis for a design that integrates the data from various sources, and the star schema structures are used for most of the reporting.

- **Data Warehouse/Data Mart Data Models—Star Schemas** Some may call this alternative a “Bus architecture,” “Constellation,” or “Conformed dimension” approach. This approach also advocates an enterprise-wide data warehouse structure that integrates the data from several sources and then passes the data to several data marts that are structures used for specific application reporting needs. In this approach, both the enterprise-wide data warehouse structure and the data marts are designed using multidimensional analysis that included designing facts, measures, dimensions, levels, and a combination of various star schemas (integrated star schemas are sometimes called a “constellation”). In the enterprise-wide design, the dimensions are “conformed,” allowing each dimension to be reused in many star schema designs within the constellation. For example, the same customer dimension may be used in an order fact, a shipment fact, a payment fact, and so on to maintain consistency.

Note

In the star schema subsection, we refer to many multidimensional modeling terms and concepts that are quite different than many of the other logical data modeling concepts that we have been using throughout the book. We felt that explaining these terms in great detail was outside the scope of this book because our intention was just to show how to apply the patterns if you are working in a multidimensional modeling effort. If you are not familiar with multidimensional modeling, please refer to a multidimensional modeling book such as *The*

Data Warehouse Toolkit(6) by Ralph Kimball and Margy Ross to explain these terms and concepts.

Data Warehouse Data Models—Relational Approach

In this section, we address the need for the creation of data warehouse models based upon a more relational data modeling approach, an approach sometimes related to a “hub and spoke” architecture (used in Bill Inmon's approach) as opposed to a “bus architecture” (used in Kimball's Data Warehouse Toolkit approach).

One of the reasons for choosing this approach is that Sands Distribution felt that it needed a very flexible model for its central data warehouse design so that it can meet different unexpected requirements. It felt that the star schema approach involved a more specific design, and though it viewed it as an easier approach to implement, it viewed it as not being as flexible.

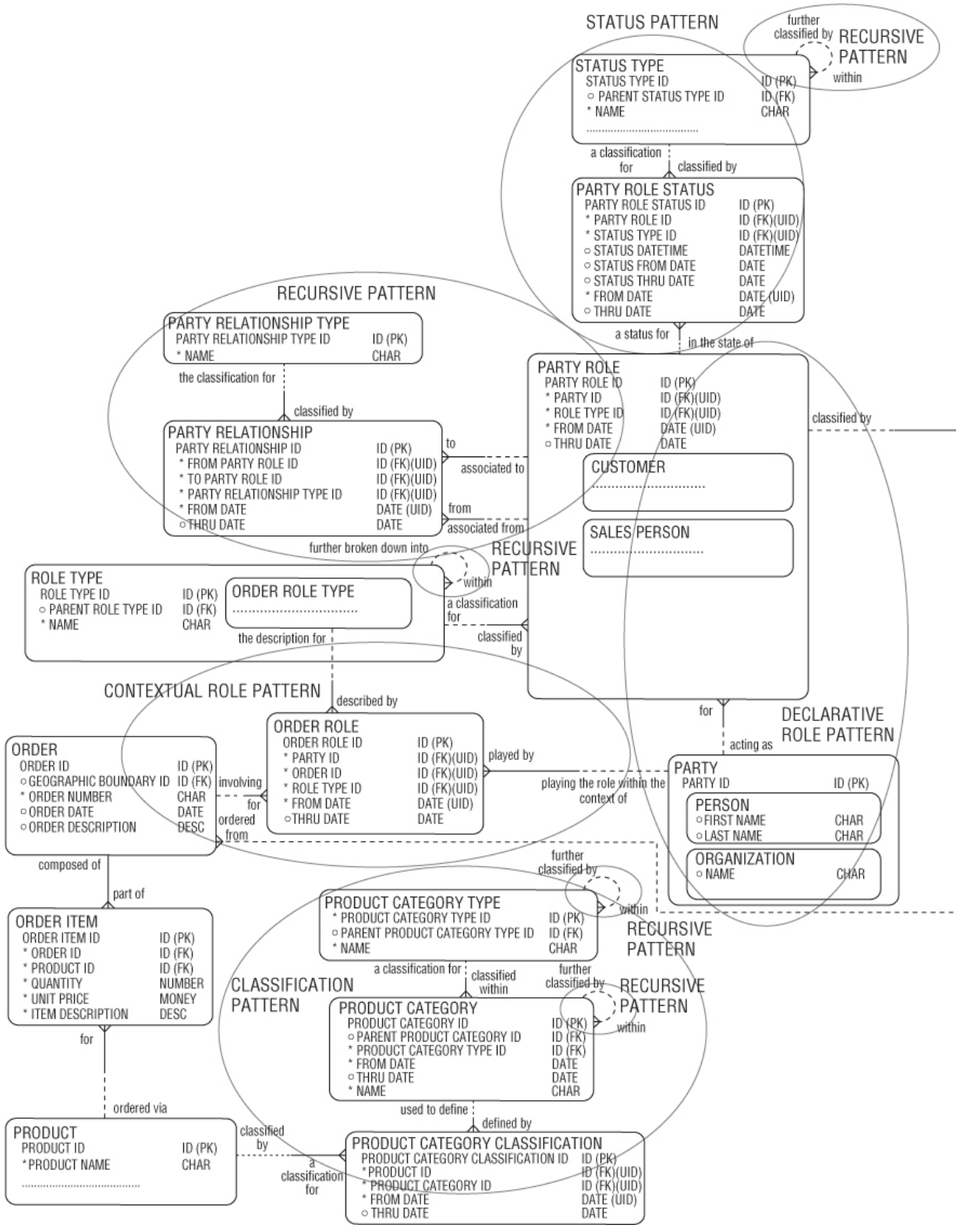
Although the approach in this section does not use a multidimensional approach for the enterprise-wide data warehouse structure, this approach does advocate a star schema design approach (that is, a multidimensional approach) for the subsequent development of their data marts. After developing the enterprise-wide data warehouse structure, the data team planned on using star schema designs to develop data marts for specific reporting needs for various groups and/or applications. A technique for using patterns for star schemas is discussed in the next subsection.

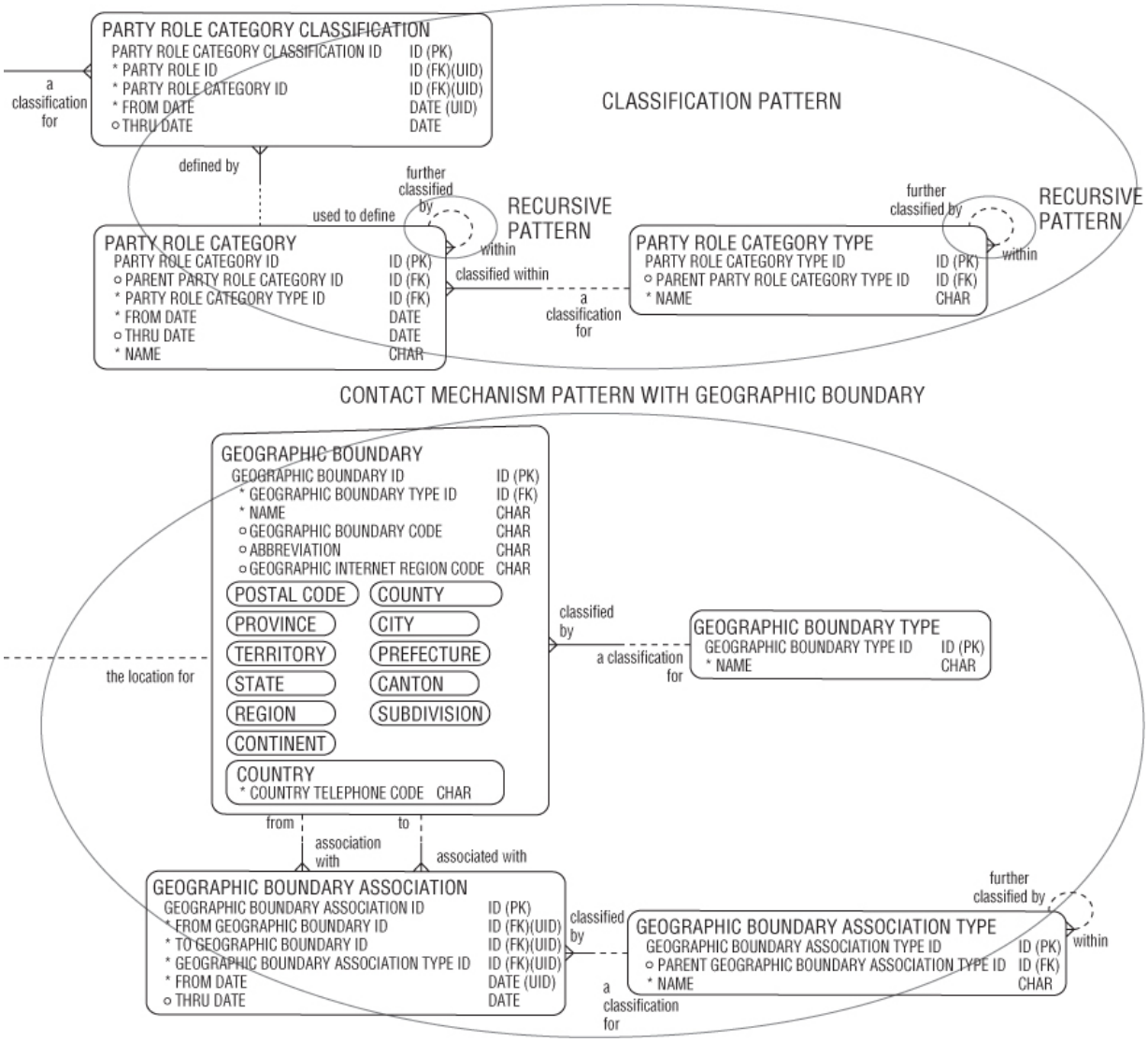
Using the approach of a relational-based data warehouse, the data team produced the diagram shown in Figure 9.5.

Note

When you are creating a decision support environment using a relational approach to data warehousing, there are some steps that should be taken to convert the data model to a physical database design. For example, removing operational data, adding an element of time, adding derived data, creating relationship artifacts, accommodating levels of granularity, merging tables, separating based on stability, and so on. We don't go into the details of these steps here, because they are described in detail in Chapters 10 and 11 of *The Data Model Resource Book, Volume 1, Revised Edition* (Wiley, 2001). The following section's purpose is to show how the patterns can form the basis to develop a data model for creating a data warehouse. In this section, we limit our discussion to how to use the patterns as part of the creation of data models and do not delve into physical database design (with the exception of a short discussion on this at the end of this chapter) because we feel that physical database design considerations are covered in depth in many books and do not need to be reiterated here(10). Also, there could be many variations of the data model in the physical database design, and these variations are based upon volumes, frequencies, and performance considerations.

Figure 9.5 Relational data warehouse data model


STATUS PATTERN
further classified by
RECURSIVE PATTERN
within
STATUS TYPE
STATUS TYPE ID ID (PK)
○ PARENT STATUS TYPE ID ID (FK)
* NAME CHAR
a classification for
classified by
PARTY ROLE STATUS
PARTY ROLE STATUS ID ID (PK)
* PARTY ROLE ID ID (FK)(UID)
* STATUS TYPE ID ID (FK)(UID)
○ STATUS DATETIME DATETIME
○ STATUS FROM DATE DATE
○ STATUS THRU DATE DATE
* FROM DATE DATE (UID)
○ THRU DATE DATE
a status for
in the state of
RECURSIVE PATTERN
PARTY RELATIONSHIP TYPE
PARTY RELATIONSHIP TYPE ID ID (PK)
* NAME CHAR
the classification for
classified by
PARTY RELATIONSHIP
PARTY RELATIONSHIP ID ID (PK)
* FROM PARTY ROLE ID ID (FK)(UID)
* TO PARTY ROLE ID ID (FK)(UID)
* PARTY RELATIONSHIP TYPE ID ID (FK)(UID)
* FROM DATE DATE (UID)
○ THRU DATE DATE
to
associated to
from
associated from
further broken down into
RECURSIVE PATTERN
within
PARTY ROLE
PARTY ROLE ID ID (PK)
* PARTY ID ID (FK)(UID)
* ROLE TYPE ID ID (FK)(UID)
* FROM DATE DATE (UID)
○ THRU DATE DATE
classified by
CUSTOMER
SALES PERSON
ROLE TYPE
ROLE TYPE ID ID (PK)
○ PARENT ROLE TYPE ID ID (FK)
* NAME CHAR
ORDER ROLE TYPE
the description for
a classification for
classified by
CONTEXTUAL ROLE PATTERN
described by
for
DECLARATIVE ROLE PATTERN
acting as
ORDER
ORDER ID ID (PK)
○ GEOGRAPHIC BOUNDARY ID ID (FK)
* ORDER NUMBER CHAR
○ ORDER DATE DATE
○ ORDER DESCRIPTION DESC
involving
for
ordered
from
ORDER ROLE
ORDER ROLE ID ID (PK)
* PARTY ID ID (FK)(UID)
* ORDER ID ID (FK)(UID)
* ROLE TYPE ID ID (FK)(UID)
* FROM DATE DATE (UID)
○ THRU DATE DATE
played by
playing the role within the context of
PARTY
PARTY ID ID (PK)
PERSON
○ FIRST NAME CHAR
○ LAST NAME CHAR
ORGANIZATION
○ NAME CHAR
composed of
part of
further classified by
within
RECURSIVE PATTERN
PRODUCT CATEGORY TYPE
* PRODUCT CATEGORY TYPE ID ID (PK)
○ PARENT PRODUCT CATEGORY TYPE ID ID (FK)
* NAME CHAR
a classification for
classified within
CLASSIFICATION PATTERN
further classified by
within
RECURSIVE PATTERN
PRODUCT CATEGORY
PRODUCT CATEGORY ID ID (PK)
○ PARENT PRODUCT CATEGORY ID ID (FK)
* PRODUCT CATEGORY TYPE ID ID (FK)
* FROM DATE DATE
○ THRU DATE DATE
* NAME CHAR
ORDER ITEM
ORDER ITEM ID ID (PK)
* ORDER ID ID (FK)
* PRODUCT ID ID (FK)
* QUANTITY NUMBER
* UNIT PRICE MONEY
* ITEM DESCRIPTION DESC
for
ordered via
used to define
defined by
PRODUCT
PRODUCT ID ID (PK)
*PRODUCT NAME CHAR
classified by
a classification for
PRODUCT CATEGORY CLASSIFICATION
PRODUCT CATEGORY CLASSIFICATION ID ID (PK)
*PRODUCT ID ID (FK)(UID)
* PRODUCT CATEGORY ID ID (FK)(UID)
* FROM DATE DATE (UID)
○ THRU DATE DATE



How Do These Models Work?

The data team decided to create a very flexible data model because it is very common that data requirements change often in decision support environments. The data team decided to use the enterprise data model in Figure 9.4 as the basis for their data warehouse model in order to reuse standard, enterprise ways of modeling while accommodating their current and future decision needs. To do this, the data team looked at the patterns utilized in the enterprise data model from Figure 9.4 and how they could adapt them for their decision support data model.

When examining the enterprise data model and the decision support needs expressed by the enterprise, they felt that the data warehouse needed to accommodate many different current and future source systems; thus they had a strong need for flexibility. They also felt that if they could keep the data warehouse structures at a consistent level of generalization, this would help a great deal in being able to reuse routines to bring the data in and out of the data warehouse. For example, the classification pattern for PARTY ROLE(s) is the same pattern used to classify PRODUCT (that is, the Level 3 Classification Pattern). When the same type and levels of patterns are applied consistently in various parts of the date warehouse data model, the routines for managing any data can be very similar and easier to develop and manage.

With this in mind, they looked at the various parts of their model, starting with how to model declarative roles. They decided to use the same type of data model structure as the flexible Level 3 Declarative Role Pattern structure, as seen in Figure 9.4. This is the pattern that maintains a single PARTY, that is, a PERSON or an ORGANIZATION, and several roles for that party that are recorded as PARTY ROLE(s), such as CUSTOMER. They felt that this was an ideal structure that allowed them to add new ROLE TYPE(s) that may be needed in the future and maintain a single instance for each party, regardless of the number of roles they play. They also felt that many internal (and some external) IT professionals employed by Sands Distribution were very familiar with this pattern, and they could reuse this expertise in their data warehousing effort.

The data team then noticed that the enterprise data model maintains both a Level 2 and Level 3 Recursive Pattern that is applied to CUSTOMER and PARTY ROLE, respectively. The Level 2 Recursive Pattern captures the CUSTOMER(s) organizational structure, and the Level 3 Recursive Pattern

allows any PARTY ROLE to be related to any other PARTY ROLE via the PARTY RELATIONSHIP and PARTY RELATIONSHIP TYPE. For example, the CUSTOMER that is the parent company may be related to the CUSTOMER that is the subsidiary, and this relationship can be maintained in the PARTY RELATIONSHIP with a PARTY RELATIONSHIP TYPE of "Parent subsidiary relationship." This level 3 pattern provides the ability for any role (such as customer, supplier, partner, and so on) to be related to another other party role via different types of relationships. They noticed the new customer information application, which would be one of their sources for data, used the Level 2 Recursive Pattern to handle their customer hierarchies. Though this worked for the limited requirements of the customer information system, they decided that this was not flexible enough for their data warehouse that needs to support the needs of all sources of data. For example, perhaps other applications or external sources of data captured other types of customer relationships such as which customer was merged into another customer, which customer was acquired by another customer, and/or which customer partnered with another customer. Additionally, they felt they needed to be able to accommodate future needs in the data warehouse to maintain the hierarchies, aggregations, and peer-to-peer relationships for all the different roles. For example, they could use the same structure to model the relationship of SUPPLIER hierarchies, the relationship from SALES PERSON to CUSTOMER, the relationship of WORKER to CUSTOMER, and so on.

With this in mind, the data team used the Level 3 Recursive Pattern PARTY RELATIONSHIP and PARTY RELATIONSHIP TYPE. This pattern captures all of the associations of CUSTOMER(s), such as a customer organizational structure. It also flexibly captures all of the other relationships that may exist between parties, such as

between each of their own internal organizations and partner organizations, between a customer contact and the enterprise for which they work, or between a supplier contact and the supplier.

The data warehouse team then looked at the enterprise data model's ways of classifying CUSTOMER data because this was an important decision support requirement. They noticed that the enterprise data model offered two ways to classify customers. The first option was to use the Level 2 Classification Pattern to classify CUSTOMER(s) by CUSTOMER TYPE, CUSTOMER SIZE, or INDUSTRY TYPE (allowing many industry types for a customer). The other option was to classify CUSTOMER(s), as well as any other PARTY ROLE, in a very flexible manner by relating PARTY ROLE to PARTY ROLE CATEGORY CLASSIFICATION (and other classification entities) using the Level 3 Classification Pattern.

The data warehouse team knew they already had requirements to categorize customers by many classifications, such as customer type, customer size, industry type, market segment, number of employees (as determined by an external source), minority customer status (needed by their human resources application), customer valuation (as determined by another application), and so on. The data team wanted the flexibility to add classifications for any PARTY ROLE (for example, classifications for suppliers, employees, partners, and so on) whenever they were needed by the business without having to add new classification entities.

So, as seen in Figure 9.5, they decided to use the Level 3 Classification Pattern and relate PARTY ROLE to PARTY ROLE CATEGORY CLASSIFICATION in order to accommodate current and future classification needs, to provide a very flexible data model structure that allows any number of ways to classify customers and allows any number of

hierarchies for classifications, and additionally to track the history of how customers were classified over time.

Notice that the same classification pattern used to classify the PARTY ROLE(s) is also used for PRODUCT. This provides the data warehouse the flexibility to create new classifications for PRODUCT(s) whenever they are needed. Because many different types of classifications (for example, product type, product line, product family) are all maintained together (in the PRODUCT CATEGORY TYPE entity), it also provides capabilities to much more easily relate different types of categories together (such as relating PARTY ROLE CATEGORY to PRODUCT CATEGORY) and thus provides some very powerful analytics capabilities. For example, the team may decide at a later point in time to have a cross-reference entity between PRODUCT CATEGORY and PARTY ROLE CATEGORY showing what types of customers are typically interested in what types of products (customer categories could be distinguished by having them within a PARTY ROLE CATEGORY TYPE of “customer categories”).

The data warehouse team also decided to use other level 3 patterns for the data warehouse model for the similar reasons of providing flexibility, accommodating future needs, and maintaining history. For example, the Level 3 Contextual Role Pattern and the Level 3 Status Pattern are flexible enough to support the addition of new contextual roles and new status types that the business may need in the future and to track when status and contextual roles changed at any time in the past.

The data warehouse team also noticed that the enterprise data model may not easily accommodate the business requirement for reporting on the locations of customers (plus other roles) and which orders were associated with different geographic boundaries (for example, a country, state, sales region, canton, or region). The enterprise data

model relates PARTY(s) and ORDER(s) to GEOGRAPHIC BOUNDARY via their CONTACT MECHANISM(s), as seen in Figure 9.4. Is this sufficient for the data warehouse to accommodate the need to answer questions such as, “Where are we getting the largest order volumes from, and how do the various countries rate regarding the total amounts of orders”? To answer this, the data team clarified the requirement with business representatives. The business representatives stated that for each order, they needed to know which COUNTRY (a GEOGRAPHIC BOUNDARY) corresponded to the ‘Billed to’ party headquarters of the order. For example, if an order came from XYZ Corporation, and its headquarters was in London, the COUNTRY associated with that order in the data warehouse would be the “United Kingdom.”

To support these specific requirements, the data team had two choices. First, they could use the structure in the enterprise data model in which an ORDER has an ORDER ROLE with ROLE TYPE of “Bill to customer” that is related to a PARTY that has a CONTACT MECHANISM(s) of type POSTAL ADDRESS that has the location type of “Headquarters” and which is related to a GEOGRAPHIC BOUNDARY. They felt that this would require too much complicated development and would go against the requirements stated by IT that all data warehouse development should be simple enough that the application team could quickly build reports and analytics. Second, they could create a relationship from GEOGRAPHIC BOUNDARY to ORDER, where an “ORDER *may be* **ordered from** one and only one GEOGRAPHIC BOUNDARY, and a GEOGRAPHIC BOUNDARY *may be* **the location for** one or more ORDER(s).” They decided to go with the latter option, as seen in Figure 9.5. This relationship between ORDER(s) and GEOGRAPHIC BOUNDARY(s) may be relevant to the enterprise as a whole, or it may be just a specific need in the data warehouse. The data team decided that they would

investigate this relationship as a candidate for inclusion in the enterprise data model. It is interesting to note that the data warehouse solution was based in large part on the enterprise data model, but the enterprise data model also may be updated based on the requirements of the data warehouse, if those requirements are applicable to other parts of the enterprise.

In conclusion, the data team created a very flexible data model that attempted to support the future needs of the business and maintain a history of changes in relationships. In fact, they also decided in this data warehouse data model to use the same level of generalization throughout the model, namely, level 3 patterns. In doing this, they believed that they would give up some simplicity of implementation, but in practice, because the models were very consistent, there was a great benefit in being able to consistently manage the data in the data warehouse and reuse routines. For instance, they noticed that the part of the data model that supported flexible classification of customer information was the same pattern that supported product classification (or any other classification). This meant that the routines and code supporting product classifications could be reused, with some modification, to support any classification. They also noticed that some of the constructs that were reused, such as the status pattern and the contextual role pattern, had useful code and expertise surrounding them in the customer application that they could leverage. Finally, developing the data warehouse model based on the enterprise data model is a well known ‘best practice’ and seemed quite natural and intuitive. Some of the data team was involved in the application and enterprise data model efforts and had built up knowledge and expertise that could be easily transferred to the data warehouse data model. This, along with the ability to reuse

code for the same types of patterns, can help in the simplification of development of the data warehouse.

Why Do We Do It This Way?

As we mentioned, there are two common approaches and methodologies for developing an enterprise data warehouse. In this scenario, the data team wanted to show a relational approach to the data warehouse data model (as opposed to using a star schema approach). They felt this approach may suit Sands Distribution because it is a dynamic environment that needs a more flexible model and additionally does not have substantial expertise in developing enterprise-wide, multidimensional models.

Star schemas have many advantages and, of course, a few disadvantages, but one disadvantage is that once you have created the star schema, it can be more difficult to modify. Many times, we find that star schemas are completely revamped. In other words, if you want to add new classifications for products, or new contextual roles such as **person installing product**, the model needs new attributes. New attributes mean a rebuild of the data warehouse dimension tables and significant work changing the extract, transformation, and load (ETL) routines.

Another reason for using the approach outlined in this section is that it allows you to maintain very comprehensive types of data that can help with analytics. For example, with this type of data model, you can comprehensively maintain all the various types of classifications, including classifications from external sources, and then do various comparisons across classifications, for example, which types of customers are interested in which types of products and/or are located in which geographic boundaries.

Another reason the data team wanted to provide a relational data warehouse solution was to reuse the expertise that was in-house. The application development

team had already built code and competencies around the customer application system. Because the data team re-used the same types of patterns that were used in other parts of the enterprise, development of the data warehouse data model in Figure 9.5 was not a huge leap for them in terms of expertise. However, in data warehousing, we do not normally recommend using the same data structures as in the source systems because it is much more important to have flexible and robust data model structures in order to accommodate the current and future needs of the data warehouse.

What Are the Strengths of Using Patterns for the Solution?

The strengths of using patterns for a relational-based data warehouse data model are as follows:

- **The more generalized patterns (such as level 3 patterns) provide a very flexible solution for maintaining decision support data:** For example, if the data warehouse needs to add a new way of classifying products, a new instance of PRODUCT CATEGORY TYPE may be added, or if there is a completely new type of status, a new instance of STATUS TYPE may be added. In both of these cases, the underlying data structures don't have to be changed.
- **Consistency:** The same level of pattern (level 3) is used throughout the data warehouse data model. This provides the ability to view and manage data consistently and to reuse common routines for moving data in and out of the warehouse. Also, if source applications also use similar patterns, this can simplify the mapping and ETL process between the source system and the data warehouse. Thus, if the enterprise is able to standardize the use of patterns, this simplifies

communications between systems, reduces maintenance costs, and allows the enterprise to leverage common routines for managing data. However, in a data warehouse design, we caution against defaulting to the same type of structures as found in the source systems to simplify ETL routines, because it is much more important to have a solid, stable data warehouse data model and design.

- **Capturing of history:** One of the critical data warehouse requirements is to track history. For example, a customer may have been classified as a “Small” enterprise, and over time they changed the classification to a “Medium”-sized organization. The associative entities in many of the more generalized patterns already capture much of the data warehouse history using the **from date** and **thru date** attributes. For example, in the Level 3 Classification Pattern, the PARTY ROLE CATEGORY CLASSIFICATION entity can maintain when a party was a small enterprise and when they became a medium-sized enterprise. The other level 3 patterns capture history in a similar fashion.

What Are the Weaknesses of Using Patterns for the Solution?

The weaknesses of using patterns for a relational-based data warehouse data model are as follows:

- **Using the more generalized patterns results in a more complex model:** When you are using generalized patterns, you should generally not present the resulting data warehouse model to business representatives. Even experienced data professionals may find models based on this type of pattern more difficult to follow if they don't understand patterns and/or they prefer a more specific style of modeling.

- **The patterns do not accommodate all history needed for data warehousing:** Though the patterns accommodate some of the history needs in data warehousing via the **from date** and **thru date** attributes in associative entities, there are other aspects of history that are not captured in the patterns, for example, the history for when an attribute value changes, such as when an ORDER ITEM **quantity** changes. The data warehouse may need to accommodate this as well. We do not address all of the different ways to accommodate history in the universal pattern because they are usually considered a physical database design consideration (for example, snapshot tables).

Synopsis

In this section, you saw how we can use patterns to create a relational-based data warehouse data model. This approach consisted of using level 3 patterns consistently across the data warehouse data model. This approach results in a very flexible and robust data model structure that can support the many needs of a data warehouse such as being able to meet needs from many source systems, accommodating many new data requirements without needing to change the data model, and accommodating many requirements regarding tracking history. Having flexible structures, such as those that are based upon level 3 patterns, allows a data warehouse to withstand the impact when there are new sources of data (without having to change the underlying structure).

The data warehouse structures used many of the data model constructs that were in the enterprise data model and thus helped move the enterprise toward using more consistent data models. There were also some similarities to the data models in the customer information application,

and this approach allowed Sands Distribution to reuse routines and expertise that had been created building that application system to create the ETL and programs for the data warehouse. However, while there is some benefit to using similar data model structures to the source systems, the first priority should be to develop a solid, stable model that accommodates current as well as future needs, and this will often look very different from the data structures in most source systems.

This data warehouse model is a relatively complex model, and it is more difficult to understand for those not familiar with more generalized data models or patterns. Also, loading into and/or reporting off a database that is based on this type of data model is more difficult, unlike the star schema approach that we discuss in the next section. However, the data model in this approach is intended to be the basis for creating a centralized, flexible data structure that can maintain data from many various source systems. After bringing this data together into a robust, flexible data warehouse structure, you can then distribute the data into data marts based on star schema designs.

Data Warehouse/Data Mart Data Models—Star Schemas

In this approach, we address the same data requirements based on the same scenario as the previous section (this was explained at the beginning of the data warehouse section of the chapter). However, in this scenario, the data team developed the data warehouse using a star schema approach.

The data team wanted a practical, simple approach to modeling and designing a data warehouse, and they knew that many enterprises successfully have used a star

schema, multidimensional approach to develop their enterprise-wide data warehouses.

They knew that they could use this approach to develop both the enterprise-wide data warehouse model and the models for the data marts. This approach greatly simplifies the synchronization and passing of data between the enterprise-wide data warehouse and the data marts because it uses the same types of model, design, and implementation for both of them.

As we discussed, the following approach may also be used in the relational data warehouse approach (Inmon approach) for the development of the data marts that are fed by the enterprise-wide data warehouse.

How Do These Models Work?

The first thing the data team decided to do when developing the model for the data warehouse application was to analyze the business requirements. Based upon these requirements, they realized that the core aspect of this reporting application was to be able to report on customer orders in various ways. They knew that there would be a fact table based on the amount and quantities for order items, and thus, they knew that there would be a fact based on the ORDER ITEM entity, with **order item amount** and **order item quantity** as measures, that would have several key dimensions such as customer, product, geographic boundaries, and time.

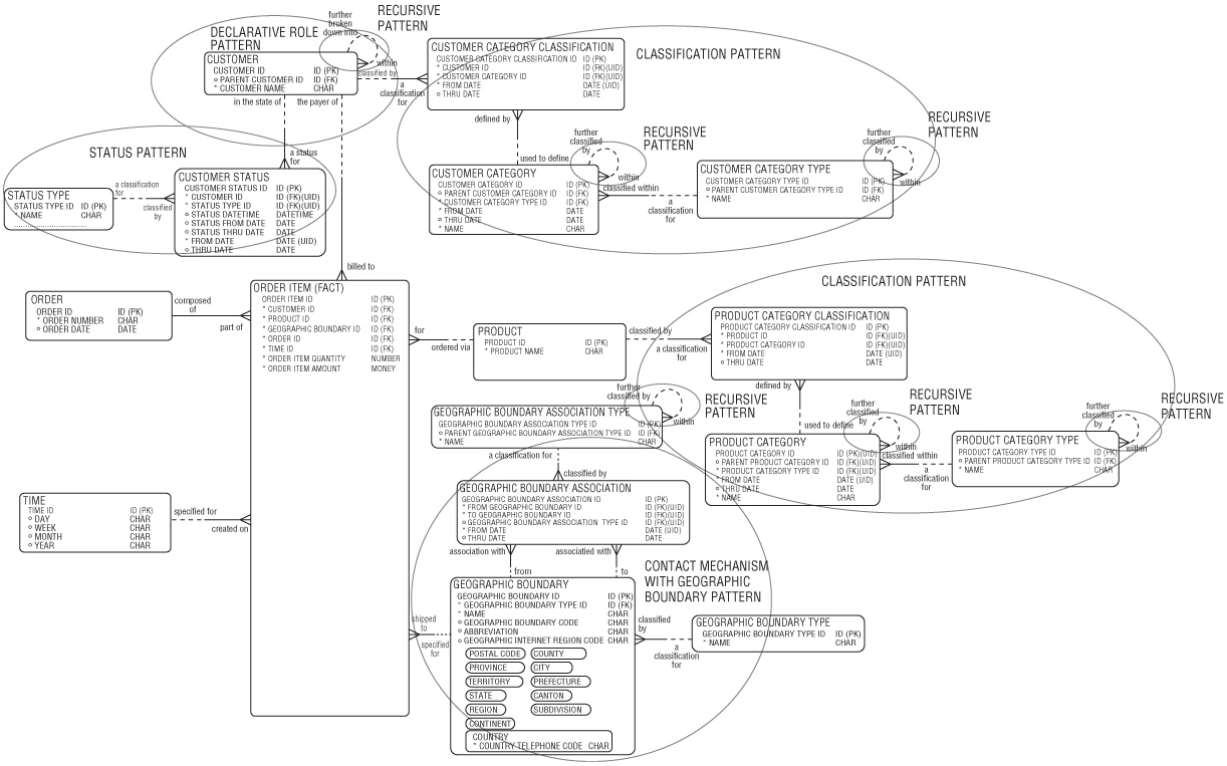
So why not just go directly to the step of creating a star schema from here? There is a great amount of complexity in each of the dimensions that the enterprise needed. For example, there could be any number of product categories and product category types. There could be many different types of geographic boundaries. There could be many different types of customer classifications that each contain sub-classifications. Thus, there are many different ways to

design the dimensions for the star schema. Although using a relational data model is not often advocated with the star schema approach, *we believe that it is critical to understand the nature of the data before designing any type of physical design*. Because a star schema is a physical design for reporting, we highly recommend understanding the complex relationships in the data, and the patterns in this book can help develop a data model that can help you understand the nature of the data.

For this effort, the data team used an approach of first understanding the data requirements, then identifying the fact table, which in this case, is fairly apparent, and then modeling the potential dimensions using the patterns to help understand the complexity involved in the various dimensions and the many possible ways that dimensions could be designed. Because the team wanted to have conformed dimensions that could be reused and integrated with other star schemas, they believed it was critical to develop a data model that helped them understand the data better so they could make more informed decisions about the design of the star schema and in particular helped with the design of the dimensions.

Thus, the data team created the data model in Figure 9.6 as a precursor to a star schema. This model uses many of the same types of patterns that are shown in the enterprise data model in Figure 9.4 and the customer information application data model in Figure 9.3. The data team used these models and the new requirements stated by the enterprise as a basis for the star schema design. In other words, they asked themselves, “What are the data requirements, what are the facts, measures, and dimensions needed to support the data requirements, and what patterns and models can we reuse to understand the nature of the data in order to develop a solid data warehouse design (using a star schema approach)?”

Figure 9.6 Star schema–based data warehouse data model



Based on the needs of reporting the order amount and quantity for orders by several dimensions, such as customer, product, geographic boundary, and so on, it was easy for the team to identify that there would be a fact table based on the ORDER ITEM entity, and thus they added an ORDER ITEM (FACT) entity in the data model, indicating that this was the candidate fact table, which had **order item quantity** and **order item amount** as attributes that could be candidate measures.

Because the business need was to report on this data by customer, customer category, customer status, product, product category, geographic boundaries, and time, the data team identified the main dimensions that they knew were needed, namely, CUSTOMER, PRODUCT, GEOGRAPHIC BOUNDARY, ORDER, and TIME. By relating the ORDER ITEM to these entities, they knew they could derive all the other requirements; for example, if they knew the customer for the order item, they could find out the customer status or

customer category by relating the customer to their applicable status(es) and/or categories.

The next question was then, “How should we model the dimension data of CUSTOMER, PRODUCT, GEOGRAPHIC BOUNDARY, ORDER, and TIME?” There are many ways that these entities could be modeled and designed to meet their needs. They could be modeled specifically versus more generally, and there could be several different dimensions related to order items, such as separate dimensions for each type of customer or product categorization. In order to decide how to create the most appropriate dimensions, they decided to first model each dimension using the data model patterns. They also compared the decision support requirements with the patterns used in the application data model and the enterprise data model. When they explored the requirements for the decision support application, they realized that they needed to consider the following:

- Business representatives stated the need to have many ways to classify products, such as by product type, line, family, product price range, and product usage. They also said that there may be new types of product categorizations needed in the future. Business representatives also stated that products may be classified into several different categories at the same time. For this reason, the data team applied the Level 3 Classification Pattern to the PRODUCT dimension.
- The enterprise needed a flexible way to classify customers. The business representatives felt that there were many more ways to classify a customer than just size, industry, and type, as shown in Figure 9.3. Notice that in the previous sections, the business was sure that a customer would be classified only by CUSTOMER TYPE, CUSTOMER SIZE, and CUSTOMER INDUSTRY. This illustrates that care is needed when you are deciding what level of pattern to use. However, the patterns

allow you the option to replace the more specific level 2 patterns with the more flexible level 3 patterns, which is what they did in Figure 9.6, using the Level 3 Classification Pattern.

- Though the initial decision support application required reporting on orders only by city, state (or region), and country, in the future there would be a need to report on any of the international geographic boundaries, including provinces, territories, prefectures, cantons, and so on. Thus, the data team decided to use the Contact Mechanism with Geographic Boundary Pattern to better understand the GEOGRAPHIC BOUNDARY dimension.

Thus, the data team created Figure 9.6. This model is meant to be *a precursor to a star schema design* that allows the data professionals to first understand the nature of the data. In other words, it is not meant to be the implementation model, but a way to get to a design that is similar to a star schema, where you can analyze the fact and key dimensions. Thus, this model allows you to see key data requirements and complexities, such as the need for complex classifications. Based upon this model, you can then choose the implementation model that includes the exact facts, measures, dimensions, and levels. After discussing the model in Figure 9.6, we will discuss a model that is based on Figure 9.6 and is a true star schema.

While going through the steps of creating the initial model, the data team kept two key ideas in mind.

First, each of the dimensions for a star schema may have a lot of complexity surrounding it, such as hierarchies, numerous statuses, various classifications, and so on. Thus, they need a way to look at this complexity so that they can resolve it before they develop the star schema.

The next idea that the data team used is that the complexity of the dimensions can be better understood by

applying the appropriate patterns. The pattern that describes these various dimensions (for example, product, customer, geographic boundary) may already have been captured in either the enterprise data model from Figure 9.4 and/or the application system as seen in Figure 9.3, or they may be new data requirements that are not in the current data models. As an example of a requirement that was already captured in the application data model, the dimension CUSTOMER as shown in Figure 9.6 has retained its related CUSTOMER STATUS and STATUS TYPE pattern that was seen in the application data model in Figure 9.3. However, you can see from Figure 9.6 that a CUSTOMER can have more than one CUSTOMER STATUS.

This raises interesting questions. For example, when we are reporting on orders by customer status, is there one primary customer status associated with the order, and if so, which customer status is associated with the ORDER? It also raises the question of whether we are only interested in knowing the customer status at the time the order was placed, when the order was closed, or for multiple times. The subject matter experts said they needed to capture whatever statuses existed for a CUSTOMER throughout the order lifecycle. For instance, a customer may have had “Active” and “Approved for credit” status types for an order. Would both customer statuses be valid when reporting on that order? The subject matter experts said “yes.” If the data team did not see the complexity of the underlying relationship between CUSTOMER, CUSTOMER STATUS, and STATUS TYPE, they may not have known to ask the subject matter experts this question. Thus, they may make an incorrect assumption, such as to use only one customer status (such as the latest status), when in fact, all of the customer statuses that existed throughout the order lifecycle are applicable.

The data team noticed in the customer application based on Figure 9.3 that a Level 2 Classification Pattern was used to capture CUSTOMER TYPE, CUSTOMER SIZE, and CUSTOMER INDUSTRY. But the enterprise felt that there were many more ways to classify a customer than just size, industry, and type. Also, the enterprise data model based on Figure 9.4 shows that either the Level 2 Classification Pattern or Level 3 Classification Pattern could be used. So, this requirement led the data team to use the more flexible Level 3 Classification Pattern and apply this to CUSTOMER so that they can easily support the existing classifications and any new classifications. This can be seen on the top right-hand side of the Figure 9.6. However, this raised a similar question to the one regarding statuses: if a CUSTOMER has more than one category at any one point in time, which one is associated with an ORDER? Said another way, if you were asking the question, "What is the total order amount for each customer category?," then if a customer is in more than one category, the sum for the report would add up to more than the total order amount because customer orders are being recounted for each customer category that applies. The business subject matter experts said that this was okay, and they just wanted to be able to analyze by customer category, knowing that customers could be in more than one category.

The business wanted to capture the order information as it related to its customers and the type of products that they ordered. Therefore, PRODUCT is one of the dimensions that classify the ORDER ITEM (FACT). However, PRODUCT also has classifications of its own. This is expressed by the Level 3 Classification Pattern containing PRODUCT CATEGORY CLASSIFICATION, PRODUCT CATEGORY, and PRODUCT CATEGORY TYPE. For example, PRODUCT may be classified by categories that are within category types of "Product line," "Product family," and "Product type." The model

accommodates any number of different instances of each different PRODUCT CATEGORY(s) that are within PRODUCT CATEGORY TYPE(s). New PRODUCT CATEGORY TYPE(s) could also be added as the enterprise sees fit. Why choose the level 3 pattern? The IT senior management specified that whatever solution the data team came up with had to be flexible enough to meet future needs of the business (even if the business was not sure what those needs would be). Again this raised the question: if a PRODUCT is in more than one category, which one is associated with an ORDER? The business representatives gave a similar answer to the status and customer category questions and said that if the ordered product was in multiple product categories, they would all be applicable when reporting on that order item.

In Figure 9.6, an ORDER is related to the ORDER ITEM (FACT), and each ORDER ITEM is related to a PRODUCT. Without having the understanding of the underlying data model structure that ORDER(s) must be composed of ORDER ITEM(s), which are specifically for a single PRODUCT, the designer could make mistakes in designing the star schema, such as a mistake about the level of granularity needed in the ORDER ITEM (FACT). For instance, if the data team puts in an **order amount** attribute in the ORDER ITEM (FACT) entity, which represents the sum of all the order items on an order, it could lead to confusion because there is only one PRODUCT that is related to an ORDER ITEM.

Two additional dimensions are also added to Figure 9.6 to support the needs of the business. First, GEOGRAPHIC BOUNDARY is added so the business can classify the order data by where the order was shipped. For example, you might need to answer a question such as, “For this period, what were the top three countries to which we shipped the most quantity of products?” The data team used the Contact Mechanism with Geographic Boundary Pattern as seen in Chapter 7 of this book as a basis for this model. This

data model structure allows reporting on different international geographic boundaries such as PROVINCE(s), STATE(s), CANTON(s), TERRITORY(s), COUNTRY(s), and any other possible geographic area. It also shows how various geographic boundaries are related to each other so that business representatives may drill up and drill down on any international geographic boundaries to analyze data at different levels of detail.

The other dimension the data team added was the TIME dimension. This has been comprehensively explained and expanded upon in many publications.(6) In short, the TIME dimension provides a way of specifying various time frames upon which one can report and then drill up or down to get a more detailed or broader view. In this case, the TIME dimension specifies the day for the ORDER ITEM (FACT) and the corresponding week, month, or year for each day. This allows for reporting on the amount and quantity ordered for any number of specified day(s), week(s), month(s), or year(s). There are also other aspects of time that appear in the data model in Figure 9.6 such as CUSTOMER STATUS (**status from date**, **status thru date, from date**, and **thru date**), and CUSTOMER CATEGORY CLASSIFICATION, PRODUCT CATEGORY CLASSIFICATION, and GEOGRAPHIC BOUNDARY ASSOCIATION that all have **from date** and **thru date** attributes that maintain when these associations were valid. The data team provides these in the model to highlight to the business that they need to capture decision support dates regarding changes of customer statuses, customer classifications, product classifications, and geographic boundary association changes. This may seem trivial, but it is often one of the most contentious issues regarding how to capture time when building a star schema. And it is often addressed under the topic of “slowly changing dimensions(6)”; however, the patterns can help to identify what type of history is needed.

Note

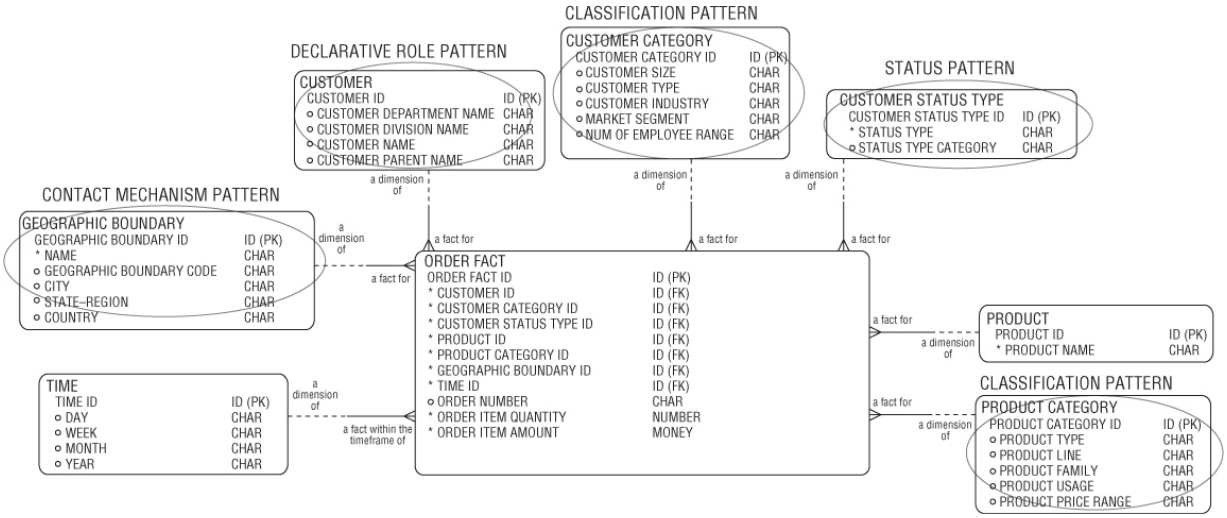
We have modeled a generalized TIME dimension in the models in Figure 9.6 (and in the star schema design that we will discuss shortly) that could have alternatively been named "TIME BY DAY." In many models, we have used multiple TIME dimensions, each with different levels to accommodate different granularities. Thus, depending on the circumstances and modeling style, there may be a TIME BY DAY dimension, TIME BY WEEK dimension (having only **week**, **month** and **year**), and/or TIME BY MONTH dimension (having only **month** and **year)**. *The Data Model Resource Book, Revised Edition, Volume 1*, provides examples of these variations on TIME dimensions.

Figure 9.6 can be viewed as a stepping stone to understand the nature of the data before creating a star schema. Many dimensional modelers skip this type of effort and create a star schema directly from the requirements and the source system databases. If you have the experience (and many dimensional modelers do), you may feel confident enough to skip this type of precursor modeling. However, we find that regardless of the level of experience, a lot of time can be saved and many potential errors may be avoided by modeling the data first, and then using the patterns to highlight and understand potential complexity so you can appropriately address the issues.

A star schema is an implementation structure that resembles a star in that it has a single fact table (the center of the 'star') with dimension tables (the protrusions of the star) that are one-to-many relationships between the dimension and the associated fact table. The star schema in Figure 9.7 is what the physical tables could look like, or if implemented, these would be the direct inputs for storing the fact, measures, dimensions, and levels. The star schema resolves the complexity of the relationships between data entities up front so that programmers don't have to write complex joins to create useful reports. It is the job of the data professionals to resolve that complexity. To do this they

need to see and understand the complexity first before resolving it, and thus, we have found that using the patterns to see this complexity before jumping directly into designing a star schema is very useful. We have been involved in efforts in which a star schema was developed, and only after significant amounts of testing had been conducted did the team realize that there was a fundamental misunderstanding in how data was related.

Figure 9.7 Star schema design, based on star schema–based data warehouse data model



The star schema is a very common and useful technique to create a relatively simple and fast-performing data analysis and reporting application. Also, many OLAP and multidimensional analysis tools are designed to deal exclusively with star schemas and ask the designers for the respective facts (the central table), measures (metrics in the fact table), dimensions (ways to slice, dice, and filter data), and levels (used for hierarchies in a dimension, such as a PRODUCT that is within a PRODUCT CATEGORY and these both could represent levels within the dimension of PRODUCT).

In Figure 9.7, you can see how the data team converted Figure 9.6 into a star schema structure and designed the various dimensions. A key observation that you should

notice is that because dimensions are very specific, they tend to look very similar to level 1 or level 2 patterns. For example, the CUSTOMER, CUSTOMER CATEGORY, PRODUCT CATEGORY, and GEOGRAPHIC BOUNDARY dimensions look very similar to the level 1 patterns. The CUSTOMER dimension is similar to the Level 1 Declarative Role Pattern because CUSTOMER is specified independent of other roles (this also could be considered a very basic and specific recursive structure since there are different levels of the customer organization that are within this dimension). The CUSTOMER CATEGORY is similar to the Level 1 Classification Pattern because each classification is maintained in its own level (attribute). The GEOGRAPHIC BOUNDARY is similar to the Level 1 Contact Mechanism Pattern in that the **city**, **state-region**, and **country** are specified distinctly within the dimension as levels (we are using the term ‘levels’ as a common star schema term and not like to mean the other type of levels that we have discussed in this book). The CUSTOMER STATUS dimension resembles a Level 2 Status Pattern. Creating a star schema often involves ‘flattening out’ more generalized patterns and using a much more specific style of modeling.

Now it's time to take a closer look at the CUSTOMER, CUSTOMER CATEGORY, and CUSTOMER STATUS dimensions in Figure 9.7. In Figure 9.6, there is a CUSTOMER entity, and it shows that customers *may be* **further broken down into** other customers, which in turn *may be* **further broken down into** other customers, and so on, thus maintaining a customer organizational hierarchy structure. This data model indicates that a customer may have any number of levels in their hierarchy. In the star schema design, because a dimension can only specify levels and not any other related dimensions, the team decided to accommodate four levels of hierarchy allowing for a **parent company name**, **customer name**, **customer division**

name, and **customer department name** because they felt that this would accommodate the current needs. This highlighted that if additional levels (or types) in the hierarchy are needed, the star schema would need to be changed.

In Figure 9.6, the CUSTOMER is related to a Level 3 Classification Pattern, and there is a CUSTOMER CATEGORY CLASSIFICATION entity (and other related classification entities) to support a flexible approach to classifying customers. This was needed because the business representatives said that there were many different types of categorizations for customers. In Figure 9.7, you see that there is a CUSTOMER CATEGORY dimension that now contains the levels of **customer size**, **customer type**, **customer industry**, **market segment**, and **num of employees range** to support the five different classifications that the business confirmed they captured (each of these attributes correspond to instances of CUSTOMER CATEGORY as seen in Figure 9.6). By modeling this as a separate dimension from CUSTOMER, this addresses the question, “If a CUSTOMER has more than one category, which one is associated with the ORDER?” According to the business representatives, they are all associated with that order item. For example, if the customer of an order item was in several different industries at the same time, the ORDER ITEM would be counted for each of the different industries. You may be tempted to record these customer classifications in the CUSTOMER dimension; however, because a customer may be in several categories, it may be confusing to have the customer and their classifications in the same dimension. Also, done, it could lead to incorrect or misinterpreted results.

In the data model from Figure 9.6, CUSTOMER STATUS was captured as a Level 3 Status Pattern. The subject matter experts stated earlier that they wished to capture several

statuses of the CUSTOMER. By modeling the CUSTOMER STATUS TYPE as its own dimension, it addresses the question, “Which customer status is associated with the ORDER?” The answer is whatever statuses the customer has had throughout the lifecycle of the order. Therefore, if a customer has had both the “Active” and “Credit approved” statuses during the order, both are counted. So if you ask for the total **order item amount** for “Active” customers and then ask for the total amount for the total **order item amount** for “Credit approved” statuses, if a customer has had both statuses for an ORDER ITEM, the amounts would apply to both questions. By modeling the CUSTOMER STATUS TYPE as its own dimension, rather than having it as a level of CUSTOMER, it avoids the possible confusion of thinking that there is only one status for a customer.

A similar process was applied to the PRODUCT dimension where the flexible Level 3 Classification Pattern was replaced with both a PRODUCT dimension and a PRODUCT CATEGORY dimension because the same product can be classified in multiple categories and classified in the same category type many times (a product may be considered to have a product usage of both “Heavy duty” and “Light duty” at the same time). Thus, instead of having a single dimension for PRODUCT, separating PRODUCT and PRODUCT CATEGORY into two different dimensions avoids any confusion by asserting that knowing the product doesn't mean we necessarily know the product category (because there may be many). The PRODUCT CATEGORY maintains the various classifications such as **product type**, **product line**, **product family**, **product price range**, and **product usage** (each of these attributes corresponds to instances of PRODUCT CATEGORY as seen in Figure 9.6). When the subject matter experts were asked, “If a PRODUCT has more than one category, which one is associated with an ORDER?,” their answer was that all of a product's categories

apply, and the data team accommodated this requirement in the star schema design.

The TIME dimension data was derived by using the **order date** in ORDER from Figure 9.6 and then applying it to the star schema design and relating the various measures of the fact table to the TIME dimension. This provides a way of reporting, slicing, and rolling up the order data by day, week, month, and year.

The GEOGRAPHIC BOUNDARY dimension was flattened, and each subtype, supertype structure seen in Figure 9.6 was implemented using a very specific structure in the dimension shown in Figure 9.7. For example, in this case, the data team maintained **city**, **state-region**, and **country** in the dimension (as well as **geographic boundary code**), allowing the data to be reported at any of these levels. The data model in Figure 9.6 showed that there may be future needs to maintain international address boundaries such as territory, province, canton, and so on; however, the data team decided to simplify the model because the current requirements could be met by just capturing the three levels of city, state, and country for the GEOGRAPHIC BOUNDARY dimension.

The ORDER FACT contains two different measures for the applicable dimensions.

- First, **order item quantity**, the quantity - for example, a quantity of “235” items ordered may be an instance that would be applicable for the specified combination of dimensions. This number would be the quantity of products ordered that are associated with specific product(s), product category(s), customer(s), customer status(es), customer category(s), or any combination of these dimensions.
- Second, **order item amount**, the monetary value of the ORDER ITEM: for example, “200,000” that would be applicable to the specified dimension values. This star

schema assumes that there is only one currency involved. If numerous currencies may be recorded, there could also be a CURRENCY TYPE dimension, or alternatively, there could be several measures, one for each type of currency, for example, **order item US dollars amount**, **order item Euros amount**, and so on.

> **Note**
>
> What we described here was a subsection of a data warehouse model. In a data warehouse there would normally be other star schemas that are combined into a constellation format, and they would each probably have shared or, in other words, “conformed” dimensions. For example, the same CUSTOMER dimension may also be used in another star schema, such as in a CUSTOMER PAYMENT FACT star schema that shows analysis of a customer's payment history within the data warehouse.

Why Do We Do It This Way?

Similar to other types of data models or data designs, developing a star schema design requires a data professional to understand the nature of the data and then balance current versus future requirements, some of which may not be obvious to the business or the application development team. The data model patterns can help develop star schemas in the following ways:

- Patterns may be used to develop an initial data model before the star schema design in order to show how various types of common data may be modeled. Allowing the data modeler to pick from various patterns and incorporate the appropriate amount of flexibility in the data model provides clues regarding an effective structure for the dimensions. For example, if there is a strict hierarchy where a product is within a product line, which is within a product family, it may be more suited to put them in as levels (‘levels’ is used here as a star

schema term) within the same dimension. If products may be in multiple categories and they don't roll up neatly in a hierarchical fashion, it is still possible to put them in the same dimension; however, it may be more clear to have separate dimensions for PRODUCT and PRODUCT CATEGORY because there is a many-to-many relationship between them.

- We can get hints about how to structure dimensions from the patterns. Dimensions tend to usually resemble level 1 patterns and sometimes level 2 patterns (for example, when we showed the customer status pattern). The complexity that resides in a logical data model is often ‘flattened out’ in order to provide a very simple way to report on the data.
- The IT professionals of Sands Distribution were very familiar with patterns and relational data modeling. This method of expressing the dimensions as patterns helped them grasp the complexity of the dimensions and gave them a stepping stone into a dimensional model.
- The patterns can help provide a bridge between the enterprise data model (or source application data models) and decision support requirements for a star schema. In other words, patterns were expressed in the star schema data warehouse data model (Figure 9.6) to illustrate complexity that needed to be understood and then resolved in the dimensions of the star schema (in Figure 9.7).

What Are the Strengths of Using Patterns for the Solution?

The enterprise had some specific requirements in relation to decision support, for example, it felt that there were many more ways to classify a customer than just size, industry,

and type. Also, the requirements specified that the solution that the data team came up with had to be flexible enough to meet future needs of the business (even if the enterprise was not sure what those needs would be) and simple enough that the application team could quickly build reports and analytics. The strengths of using patterns for star schema design are as follows:

- **A more solid and stable star schema design with fewer mistakes:** By using patterns to develop a data model before developing the star schema, you can better understand the nature of the data. This helps you to make more intelligent decisions when designing the star schemas and to develop more stable and solid designs. When the data is neither modeled nor understood well, we have found that costly mistakes can happen that either result in a delayed schedule for implementing the star schema, or worse, misinterpreted data when people are using it.
- **Consistent designs for more consistent ways to develop dimensions:** By having the patterns available, you can develop more consistent ways of designing dimensions. For example, the customer and product classification levels within a dimension may be similar and may use the Level 1 Classification Pattern as a guide for any type of dimensions having to do with classifications. You can use the patterns to convert what may initially seem to be a complex level 3 pattern to a structure that resembles a simple Level 1 or level 2 pattern. Thus, the complexity involved in the data can be first understood and then resolved by creating a simple (flattened) dimension table, while understanding potential tradeoffs of doing this.
- **Better understanding of the data:** If you don't develop a data model before jumping into the star schema, you may misunderstand how the data is

related, and these patterns can help you to understand the nature of the data better. For example, if you just have the product type and other product classifications as levels within the PRODUCT dimension without understanding the complexity in classifying the product data, the results from the star schema may be unpredictable and misunderstood. If you use a pattern to create a PRODUCT CATEGORY CLASSIFICATION data model structure, such as that shown in Figure 9.6, you can see that products may be in multiple classifications at the same time and that there are different sets of classifications with multiple ways to rollup product classification data. This helps with the design and usage of the star schema.

What Are the Weaknesses of Using Patterns for the Solution?

The weaknesses of using patterns for a star schema design are as follows:

- **The data model that is used as the basis for the star schema sometimes does not show true data relationships:** For instance, the data model in Figure 9.6 shows a relationship from ORDER ITEM to CUSTOMER, and there is really a relationship from ORDER to CUSTOMER. We did this because we were working under the assumption that the ORDER ITEM was the basis for the fact table, and we knew that there would be a relationship from the ORDER ITEM (FACT) to the CUSTOMER that was the “billed to” party. However, in this approach, we are making certain assumptions, such as what the fact table will be, and therefore we are not always modeling the true nature of the relationships.

- **Some consider using patterns for the development of a relational data model an extra step that is unnecessary for star schema development:** Many star schema designers and advocates feel that they can use multidimensional analysis to understand the nature of the data and that relational data modeling is not necessary and is an extra step.

Synopsis

In this section, you saw how the data team utilized the different levels of patterns to create a data model that could be used as the basis for a star schema. A data model that is a precursor to a star schema design can be used to better understand the data requirements. When using this approach, you can develop the data model using the patterns in this book to represent the appropriate amount of flexibility needed for various requirements. This helps create a bridge between the requirements and the star schema by first modeling the requirements using the appropriate patterns before developing a star schema.

Some experienced dimensional data modelers may feel that it is unnecessary to develop a precursor data model before developing the star schema. We have found that whether you are an experienced modeler or an inexperienced modeler, modeling and understanding the data requirements helps a great deal in developing effective designs, including the design of star schemas.

Master Data Management

In *The Data Administrator Newsletter*, David Friedland writes, “Good master data management (MDM) is required to keep this data clean, and to standardize master data models amid the relational taxonomy of facet data.”(7)

Master data management supports the integration of data and the dissemination of consistent reference and master data across an enterprise as a whole. In this section, we use patterns to create a standardized master data management data model for customer data.

The Scenario for This Model

Sands Distribution became the target of a hostile takeover bid from a large Middle Eastern conglomerate. After fending off the hostile bid, the CEO decided that he is going to send all his customers a letter explaining the situation and thanking them for the continued support. He presented his letter to the CIO and asked him to have it mailed to each customer. The CIO in turn asked for the definitive list of all customers from the heads of each of his application systems. What he discovered is that the billing system has one list, the customer information application had another list, and the general ledger payments system had yet another list. Further, there were overlaps and inconsistencies in all of the customer lists! “What company does not know its customers?” cried the CEO, who was naturally irate.

Given this situation, the CEO recalled his highly acclaimed data team to give him a solution for this customer master data problem. The data team described to the CEO the nature of their customer data problem as follows:

- First they told the CEO that the customer data did not reside in one single system; that is, they don't have OTOB (one thing in one box).(8)
- The structure of the customer data was different in each of the different systems where it was stored, and each system uses different semantics and terminology to describe various pieces of customer data. For example, the customer application captured some contact

information, and the general ledger captured some payment addresses but no phone numbers.

- The data stored in each system was inconsistent. So for the same piece of data, for example, a customer headquarters postal address, different systems often captured a different address.
- Each of the application managers of each of the systems that have customer data believed that his or her system should be the central source (that is, the system of record) of all customer data.
- They started moving toward more consistent data and more consistent semantics by using patterns for their data models. They used these patterns to develop an enterprise data model, customer information system, and data warehouse application. However, the enterprise had dozens of applications that use customer data, and though there is a movement toward more consistent data, there were still a great many inconsistencies in data and different semantics among systems.

The head of the data team suggested the following alternatives to the CEO:

- **Clean up the mess:** In other words, start a data cleanup program to make sure that the data in each of the systems is consistent. Each of the application groups would need to devote resources to clean up data for their systems, and the enterprise would need to initiate a data cleansing process to coordinate these efforts. This needed to be followed by a system of checks to ensure the data is clean and in sync. All these measures would cost time and resources. Furthermore, this effort could then be expanded to a data quality program so the mess would not only be cleaned up, but an ongoing program would be established, including

developing processes to measure and continually improve the quality of data.

- **Create a new master data management (MDM) database that integrates and synchronizes all master data for customer information:** The master data management system would coordinate all the system, synchronize all the systems, and move customer data from the primary sources in order to keep customer data consistent. As part of this effort, the enterprise would decide which data under various circumstances should be the primary source (system of record) for customer data. This effort would require new technology and an investment of significant resources, including time and money.
- **Develop a data governance program:** This program would establish governing bodies, rules, policies, decision rights, procedures, standards, accountabilities, and data stewardship so all can manage their data assets more effectively.
- **Do all of the above:** Clean up the data (and develop a data quality program), create a new master data management database, and create a data governance program. A cleanup of the data would help in the short term and could also help set up the data before a master data management application started synchronizing and integrating data across systems. Each of the existing systems would identify where the sources of master data existed, the rules would be identified for appropriately matching, consolidating, and synchronizing data, and then they would develop (and/or buy) the technology to manage master data such as customer data. The enterprise could also establish a data governance program to support the master data management effort and data integration

and management in general. This alternative would require the largest investment in time, effort, and cost.

> **Note**
>
> Of course, there are many other solutions that Sands Distribution could have evaluate to solve this problem, such as implementing a full-blown data quality program, implementing a metadata management program, reengineering its business processes that control the data, and many other alternatives. For the purposes of this example, those were discounted by the data group after an initial assessment.

The data group produced straw man project plans for each alternative solution with some initial costing of the time and money for each project. Based on the data group's analysis, the CEO and CIO, as part of the operating committee, decided to take the last and most expensive option.

> **Note**
>
> The overriding rationale for going with this solution is that Sands Distribution realized that data is an asset, and that all assets need to be managed. They felt this solution was the best way to manage this data asset. It also recognized that some of the key drivers for MDM were compliance issues, mergers and acquisitions, Service-Oriented Architecture, better sales, and service of Sands Distribution customer needs. Finally, it decided it wants to be an analytical competitor, and to do this it recognized that MDM provides a very solid foundation for having the quality data it needs, when it is needed.

The data team concentrated on some key issues for the first iteration of the MDM solution. One key issue was how to provide a more complete view of the customers as well as other parties involved in their business. They also realized that there are many sophisticated packaged solutions for identity management and matching, business rules, determining system of record, and other aspects of master data management. They knew that to develop and

implement this MDM solution from scratch was going to be a very difficult task, so they decided to use the data model as a way to communicate their data needs to potential MDM solution vendors. The team wanted to start small and think big with their data model, so they concentrated initially on a very small subset of customer master data. This included capturing semantics of a subset of customer data and describing the level of control and flexibility that Sands Distribution needed to manage data that classifies customers. The data they decided to initially concentrate on was as follows:

- The status data for a customer.
- The relationships customers had/have with each other, such as the parent company/subsidiary relationship.
- The classifications for a customer.
- The different ways to contact a customer, how to manage those contacts and, in particular, how to manage addresses in different parts of the world.
- The business rules involved with customers, for example, defining the factors that dictate which system may be the system of record or defining the factors that result in providing a valuation (that is, an estimate of how valuable a customer is to Sands Distribution) of customers.

A significant aspect to master data management is identity management, and there was some debate concerning whether this should have been addressed as part of the data modeling effort. Identity management focuses on the need to look at multiple sources of the same data and determine where they refer to the same data instance. For example, if there is a record in one database that says “Steve Jones,” is located at “101 Main Street, Denver, Colorado,” is this the same instance as another record coming from another database that says “Steven Jones” is located at “123 Main Street, Denver, Colorado”? Or

could it be that there just happens to be two different people, one person named “Steve Jones” and another person named “Steven Jones,” and they both live on Main Street? There are several strategies and methods for identity management, and the data team decided that they would not address this in the data model at this time because this required cross-referencing the source systems and specifying the systems of record (which system would dictate the master record). They decided they can capture this information in their metadata model. Another reason for not addressing this issue as part of the data model is that there are a great number of design considerations that impact how to handle this cross-referencing (for example, master data management solutions often have widely different approaches to the way that they handle identity management).

How Does This Model Work?

The customer master data solution needed to integrate and synchronize the following types of data:

- The various statuses for customers, the different classifications for these customer statuses, and the structure and rules that manage and describe those groups of classifications. For example, “Active,” “Inactive,” “In abeyance,” “Credit approved,” and “Under investigation” may all be possible customer statuses, and “Active status” may be further broken down into additional statuses such as “High level of activity” and “Low level of activity.”
- The relationships that the customers have to various parties, including the relationships that exist between various parts of customer organizations (the internal structure of the customers), the contacts that work for a customer enterprise, the relationship between salespeople and the customer contacts, or any other

relationships that exist between a customer and other parties.

- All of the different ways that customers may be classified, such as by industry, size, number of employees, and/or any other classification that may be needed for existing or future applications, including data that is provided by external parties. Additionally, because this type of data is maintained in a great variety of formats from different systems, there is a need to be able to map these various data formats to a common format in order to synchronize and integrate the data.
- The different contact mechanisms that a customer may have, such as phone numbers, cell numbers, emails, postal addresses, and/or any other type of contact information. The customers of Sands Distribution are in many different locations around the world, so the address information must accommodate international requirements that reflect the different address structures around the world.
- The different business rules applicable to master data management. These may include rules specifying how matching of records between different source systems occurs and rules that can be applied to customers or other party roles.

So, to support these different needs for customer master data management, we will describe how the data team applied more generalized patterns to model the above types of data, namely how they modeled:

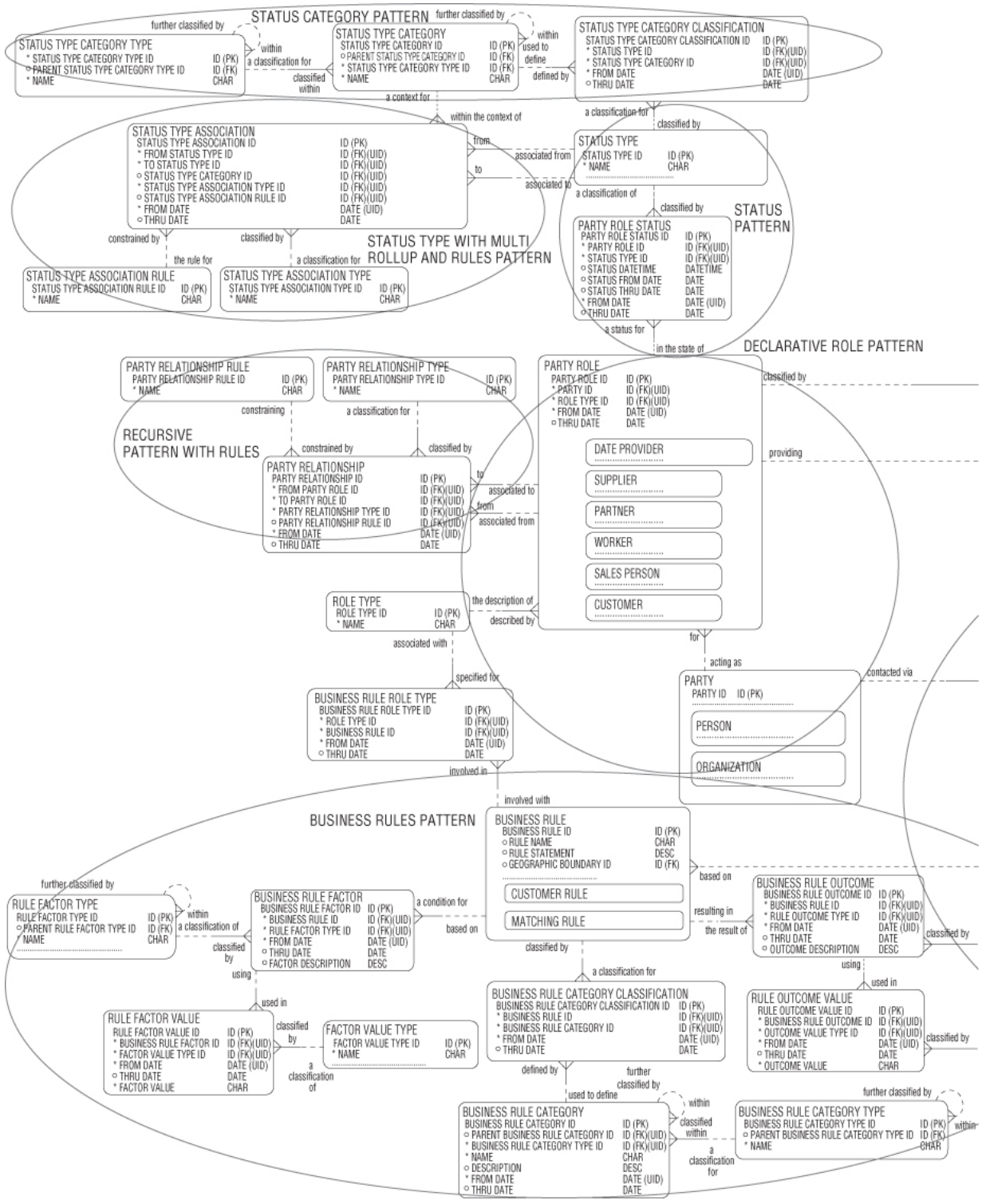
- Customer statuses
- Customer relationships
- Customer classifications
- Customer contact information (contact mechanisms)
- Customer business rules (especially for determining system of record and customer valuation)

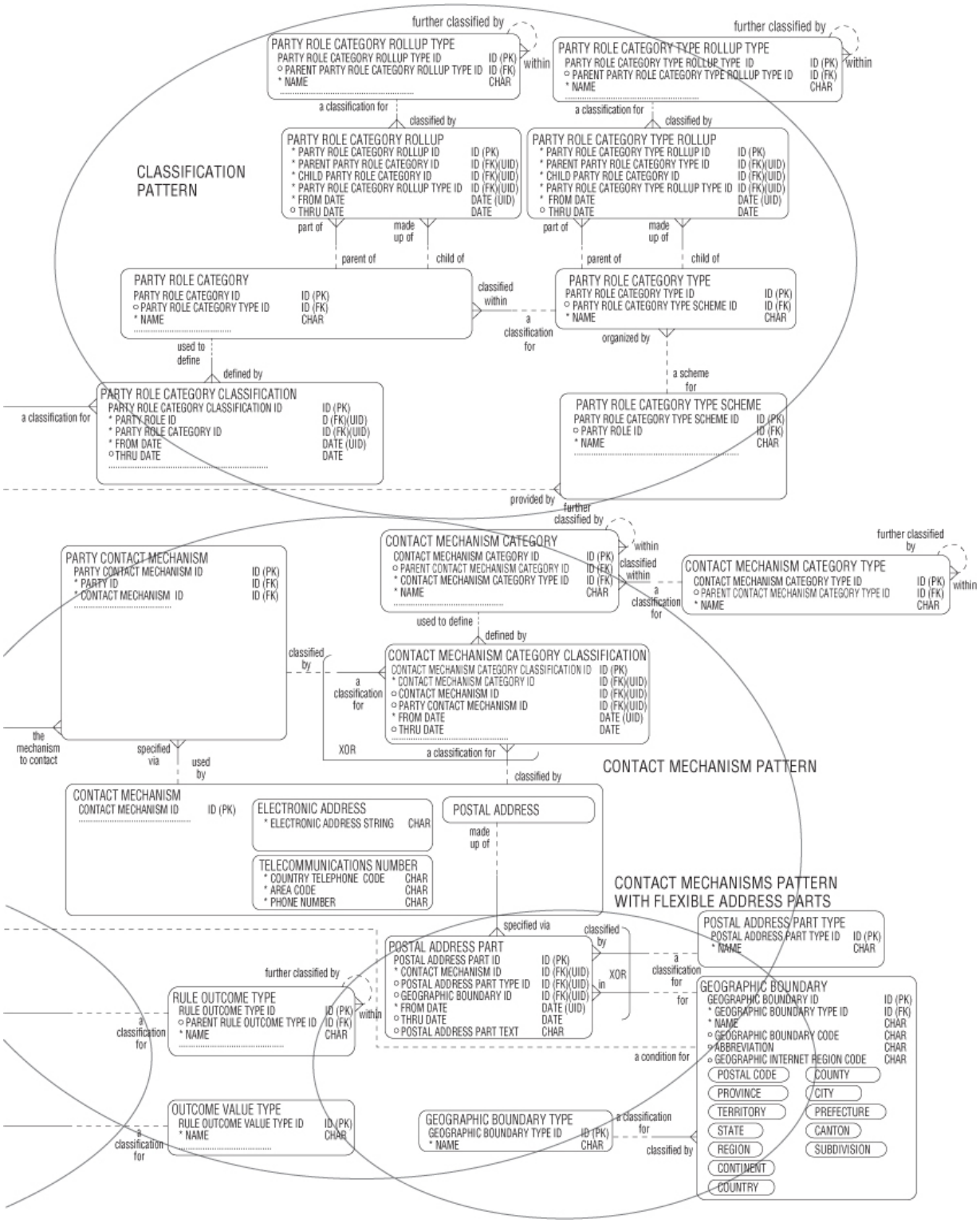
The data team developed the model that is in Figure 9.8 to accommodate the preceding needs for master data management. You can see that the Level 3 Status Pattern was applied, and this allowed any PARTY ROLE to be related to any number of STATUS TYPE(s), and any STATUS TYPE may apply to more than one PARTY ROLE. For example, customer master data involves many different types of customer statuses such as “Active,” Credit approved,” “In Abeyance,” and “Under Investigation.” Thus, a customer may have any number of these statuses at the same time, or a party may have many types of statuses over time. This data model structure also works for other types of roles, such as for SUPPLIER, PARTNER, and WORKER, because they each involve a great number of types of statuses that are also important for other areas of master data management.

Figure 9.8 has applied the Status Category Pattern that allows statuses to be flexibly categorized so that the same STATUS TYPE(s), such as “Active” and “Inactive,” may be classified into multiple STATUS TYPE CATEGORY(s). For example, when a customer is purchasing products, there could be a STATUS TYPE CATEGORY of “Customer purchasing statuses” that may be a classification for many STATUS TYPE(s) such as “Active,” “Credit approved,” and “Credit denied.” A different set of statuses could be used to capture various credit statuses. For example, there could be a STATUS TYPE CATEGORY of “Customer credit statuses” that may be a classification for STATUS TYPE(s) of “Credit denied,” “Credit approval pending,” or “Credit approved.” Notice that the same STATUS TYPE(s) of “Credit approved” and “Credit denied” may be classified by more than one STATUS TYPE CATEGORY, and the pattern accommodates this via the associative entity of STATUS TYPE CATEGORY CLASSIFICATION. Furthermore, there could be more general (that is, higher level) types of categories maintained in the STATUS TYPE CATEGORY TYPE. For example, each of the two

categories just mentioned, “Customer credit statuses” and “Customer purchasing statuses,” *may be* **classified within** a STATUS TYPE CATEGORY TYPE of “Customer statuses.” Another example of a STATUS TYPE CATEGORY TYPE could be “Transaction statuses” that may include STATUS TYPE CATEGORY(s) of “Order statuses,” “Shipment statuses,” or “Invoice statuses.” The model also has used the Status Type with Multi Rollup and Rules Pattern, which maintains the different ways that statuses are related to each other and the rules about how these statuses may be related. Statuses may relate to each other in various ways. For example, one status may be a further breakdown of another status in a hierarchy, one status may be mutually exclusive of another status, or one status may be substitutable for another status. For example, there could be hierarchies of statuses with the highest level of one status hierarchy being “Blacklisted.” Then within the “Blacklisted” status you might have a status of “Blacklisted due to unpaid bills written off” and another status of “Blacklisted due to ethical issues.” Thus, the STATUS TYPE ASSOCIATION and associated entities use the recursive pattern, allowing you to relate statuses to each other in different ways.

Figure 9.8 Master data management data model


STATUS CATEGORY PATTERN
STATUS TYPE CATEGORY TYPE
STATUS TYPE CATEGORY TYPE ID
PARENT STATUS TYPE CATEGORY TYPE ID
NAME
STATUS TYPE CATEGORY
STATUS TYPE CATEGORY ID
PARENT STATUS TYPE CATEGORY ID
STATUS TYPE CATEGORY TYPE ID
NAME
STATUS TYPE CATEGORY CLASSIFICATION
STATUS TYPE CATEGORY CLASSIFICATION ID
STATUS TYPE ID
STATUS TYPE CATEGORY ID
FROM DATE
THRU DATE
STATUS TYPE ASSOCIATION
STATUS TYPE ASSOCIATION ID
FROM STATUS TYPE ID
TO STATUS TYPE ID
STATUS TYPE CATEGORY ID
STATUS TYPE ASSOCIATION TYPE ID
STATUS TYPE ASSOCIATION RULE ID
FROM DATE
THRU DATE
STATUS TYPE
STATUS TYPE ID
NAME
STATUS TYPE WITH MULTI ROLLUP AND RULES PATTERN
STATUS PATTERN
STATUS TYPE ASSOCIATION RULE
STATUS TYPE ASSOCIATION RULE ID
NAME
STATUS TYPE ASSOCIATION TYPE
STATUS TYPE ASSOCIATION TYPE ID
NAME
PARTY ROLE STATUS
PARTY ROLE STATUS ID
PARTY ROLE ID
STATUS TYPE ID
STATUS DATETIME
STATUS FROM DATE
STATUS THRU DATE
FROM DATE
THRU DATE
DECLARATIVE ROLE PATTERN
PARTY RELATIONSHIP RULE
PARTY RELATIONSHIP RULE ID
NAME
PARTY RELATIONSHIP TYPE
PARTY RELATIONSHIP TYPE ID
NAME
PARTY ROLE
PARTY ROLE ID
PARTY ID
ROLE TYPE ID
FROM DATE
THRU DATE
RECURSIVE PATTERN WITH RULES
PARTY RELATIONSHIP
PARTY RELATIONSHIP ID
FROM PARTY ROLE ID
TO PARTY ROLE ID
PARTY RELATIONSHIP TYPE ID
PARTY RELATIONSHIP RULE ID
FROM DATE
THRU DATE
DATE PROVIDER
SUPPLIER
PARTNER
WORKER
SALES PERSON
CUSTOMER
ROLE TYPE
ROLE TYPE ID
NAME
BUSINESS RULE ROLE TYPE
BUSINESS RULE ROLE TYPE ID
ROLE TYPE ID
BUSINESS RULE ID
FROM DATE
THRU DATE
PARTY
PARTY ID
PERSON
ORGANIZATION
BUSINESS RULES PATTERN
BUSINESS RULE
BUSINESS RULE ID
RULE NAME
RULE STATEMENT
GEOGRAPHIC BOUNDARY ID
CUSTOMER RULE
MATCHING RULE
RULE FACTOR TYPE
RULE FACTOR TYPE ID
PARENT RULE FACTOR TYPE ID
NAME
BUSINESS RULE FACTOR
BUSINESS RULE FACTOR ID
BUSINESS RULE ID
RULE FACTOR TYPE ID
FROM DATE
THRU DATE
FACTOR DESCRIPTION
BUSINESS RULE OUTCOME
BUSINESS RULE OUTCOME ID
BUSINESS RULE ID
RULE OUTCOME TYPE ID
FROM DATE
THRU DATE
OUTCOME DESCRIPTION
RULE FACTOR VALUE
RULE FACTOR VALUE ID
BUSINESS RULE FACTOR ID
FACTOR VALUE TYPE ID
FROM DATE
THRU DATE
FACTOR VALUE
FACTOR VALUE TYPE
FACTOR VALUE TYPE ID
NAME
BUSINESS RULE CATEGORY CLASSIFICATION
BUSINESS RULE CATEGORY CLASSIFICATION ID
BUSINESS RULE ID
BUSINESS RULE CATEGORY ID
FROM DATE
THRU DATE
RULE OUTCOME VALUE
RULE OUTCOME VALUE ID
BUSINESS RULE OUTCOME ID
OUTCOME VALUE TYPE ID
FROM DATE
THRU DATE
OUTCOME VALUE
BUSINESS RULE CATEGORY
BUSINESS RULE CATEGORY ID
PARENT BUSINESS RULE CATEGORY ID
BUSINESS RULE CATEGORY TYPE ID
NAME
DESCRIPTION
FROM DATE
THRU DATE
BUSINESS RULE CATEGORY TYPE
BUSINESS RULE CATEGORY TYPE ID
PARENT BUSINESS RULE CATEGORY TYPE ID
NAME



The team also deemed that the rules regarding how statuses were related were important. For example, there may be a status type that needs to occur before another

status is possible. Therefore, the data team decided to use the Status Type with Multi Rollup and Rules Pattern (which is based on the Level 3 Recursive Pattern with Rules). This pattern allows the data team to create different hierarchies, aggregations, or peer-to-peer associations of different status types. Also, by using the STATUS TYPE ASSOCIATION and STATUS TYPE ASSOCIATION RULE, they are able to maintain rules about the behavior between different instances of STATUS TYPE(s). Thus, a rule may be maintained about two statuses that cannot be true at the same time within a particular STATUS TYPE CATEGORY. For example, there may be a rule within the STATUS TYPE CATEGORY of “Customer purchasing statuses” that you can't have an “Active” status and an “In abeyance” status at the same time. Another example could be that a certain status may imply another status. For example, if a customer is “Waiting for shipment” within the context of the “Shipping process statuses,” then the customer may be considered “Active” within that context. This solution provides flexibility coupled with control to manage the reference data for all of the different STATUS TYPE(s).

One of the most important aspects of customer master data is capturing the various relationships that a customer has. These relationships may include relationships between the customer and Sands Distribution, for example, “Who is the salesperson for this customer?” It may also include relationships concerning the internal organizations of that customer, such as “Who is the parent company for this customer?” Other relationships that you may need are the customer contact for the customer organization and the relationship between salespeople and customer contacts (which may be different than the relationship between the salesperson and the customer). These relationships are captured by using the Level 3 Recursive Pattern with Rules, as modeled with the entities PARTY RELATIONSHIP, PARTY

RELATIONSHIP TYPE, and PARTY RELATIONSHIP RULE.(9) The different relationships have different types that can be maintained in PARTY RELATIONSHIP TYPE such as “Customer contact–customer organization relationship,” “Sales person–customer contact relationship,” or “Parent subsidiary relationship.” There may also be rules that are specified between different types of relationships, such as whether some relationships are disallowed or implied. For example, a customer may request that a particular salesperson not work with their account (exclusion). That would be an example of a PARTY RELATIONSHIP RULE.

> **Note**
>
> The pattern described for PARTY RELATIONSHIP, PARTY RELATIONSHIP TYPE, PARTY RELATIONSHIP RULE and STATUS TYPE ASSOCIATION, STATUS TYPE ASSOCIATION TYPE, and STATUS TYPE ASSOCIATION RULE is, in fact, the same pattern applied to different data sets in the data model, that is, the Level 3 Recursive Pattern with Rules.

How an enterprise classifies its customers is crucial reference data. Most reports and decision support applications use these classifications to drill into, out of, and across the different information about a customer. At different stages of a business life cycle, a customer may get classified in many different ways, or different parts of the same business may classify a customer differently depending on their context. For example, in the customer application database you saw in a previous section of this chapter, the customer was specifically classified by CUSTOMER SIZE, CUSTOMER TYPE, and CUSTOMER INDUSTRY (see Figure 9.3). The customer master data application must be flexible enough to deal with all of the different classifications a customer may have from the all of the different perspectives within (and from outside) the enterprise.

You can see on the right-hand side of Figure 9.8 that the data team decided to use the Level 3 Classification Pattern with Rollups and Schemes from Chapter 5 to support the needs of customer master data. For example, the marketing department segmented customers via a classification scheme where a customer was a “Core customer,” “Key customer,” or “Strategic partner.” The “Core customer” was defined as a customer whose loss would have a global impact on Sands Distribution; a “Key customer” loss would have a local impact on business. The “Strategic partner” customers are the most important customers to Sands Distribution business. These are the customers that Sands Distribution could not lose under any circumstances. Thus “Core customer,” “Key customer,” and Strategic partner” are each PARTY ROLE CATEGORY(s), and these different categories are of a PARTY ROLE CATEGORY TYPE of “Customer segment type.” The marketing department also had other PARTY ROLE CATEGORY(s) of “High value customers” and “Mid value customers” that were of the PARTY ROLE CATEGORY TYPE “Customer valuation.” Both of these PARTY ROLE CATEGORY TYPE(s) were considered part of a “Marketing Customer Classification Scheme” (PARTY ROLE CATEGORY TYPE SCHEME) that was **provided by** the “Marketing department” (the DATA PROVIDER).

Of course, this example describes only one portion of the enterprise. The sales department, accounting department, and IT department will also have their own way of classifying and re-classifying reference data to meet their needs. The structure and semantics of each area of the enterprise also needs to be captured by the MDM solution, and the flexible PARTY ROLE data model structures accommodate these future requirements well. Also, external DATA PROVIDER(s) may supply Sands Distribution with classification schemes for PARTY ROLE CATEGORY TYPE(s). For example, OPEC may have a classification scheme for all

various oil CUSTOMER(s), or Dunn and Bradstreet may have its own external classification schemes based on industry types that you wish to capture.

Most master data management solutions need to have several different hierarchies and/or aggregations for these classifications. Thus, both the PARTY ROLE CATEGORY and PARTY ROLE CATEGORY TYPE have multiple rollups associated with them (PARTY ROLE CATEGORY ROLLUP and PARTY ROLE CATEGORY TYPE ROLLUP, respectively). These entities support any type of hierarchy, aggregation, or peer-to-peer associations that may exist between the different PARTY ROLE CATEGORY(s) and/or PARTY ROLE CATEGORY TYPE(s). The PARTY ROLE CATEGORY of “Strategic customer” may be related to lower-level classifications such as “Strategic global customer” and “Strategic national customer.” These would be related via an instance of PARTY ROLE CATEGORY ROLLUP that could be of PARTY ROLE CATEGORY ROLLUP TYPE “Marketing reporting classification.” The PARTY ROLE CATEGORY(s) of “Core customer,” “Key customer,” and “Strategic partner” may be **classified within** the PARTY ROLE CATEGORY TYPE of “Customer segment type.” These different types of classifications may also have a hierarchical structure. For example, “Customer segment type” and “Customer industry type” may be a **within** a PARTY ROLE CATEGORY TYPE of “Customer classification.”

A core requirement of customer master data is capturing the different ways to contact a customer, and the different types, purposes, usages, and other contact mechanism classifications for CONTACT MECHANISM(s). In Figure 9.8 you see that the data team used the Level 3 Contact Mechanism Pattern and the Contact Mechanism with Flexible Address Parts Pattern to achieve a great deal of flexibility when dealing with customer contact information. Contact information is a very important aspect of master data

management because it helps with identity management; therefore, a lot of flexibility is usually needed. First, with this level 3 pattern, you can capture any ELECTRONIC ADDRESS, TELECOMMUNICATIONS NUMBER, or POSTAL ADDRESS that any PARTY may have. People and organizations both get contacted many different ways. These people or organizations may be customers, suppliers, partners, or competitors. They may play all four of these roles (plus many more roles) all at the same time. But their contact information may be the same, regardless of role. For example, “K Morris Corporation” is a customer and a supplier for Sands Distribution. Its head office address is in Riyadh in the Kingdom of Saudi Arabia. The address is the same regardless of whether it is playing the role of supplier or customer. Thus, this flexible data model structure in Figure 9.8 accommodates this and many other scenarios.

The CONTACT MECHANISM CATEGORY CLASSIFICATION, CONTACT MECHANISM CATEGORY, and CONTACT MECHANISM CATEGORY TYPE provide a way to maintain the type, purpose, usage, priority, location, or any other type of classification for either a CONTACT MECHANISM or for a PARTY CONTACT MECHANISM. The model shows that each PARTY *may be* **contacted via** one or more PARTY CONTACT MECHANISM(s), each of which may be associated with various CONTACT MECHANISM CATEGORY(s) (via the CONTACT MECHANISM CATEGORY CLASSIFICATION associative entity). Either a CONTACT MECHANISM or a PARTY CONTACT MECHANISM may be related to the CONTACT MECHANISM CATEGORY CLASSIFICATION, and thus either one may be classified by any number of purposes, usages, priorities, or other classifications. For example, a customer contact may have a contact mechanism (PARTY CONTACT MECHANISM) that has CONTACT MECHANISM CATEGORY(s) of “Postal address,” “Bill to,” “Primary,” and “Business.” Each of these categories corresponds to

CONTACT MECHANISM CATEGORY TYPE(s) of “Contact mechanism type,” “Purpose type,” “Priority type,” and “Usage type,” respectively. A marketing application might want only contact information that has a purpose type of “Sales” and might not want to use any CONTACT MECHANISM that is solely for “Personal” use.

The different POSTAL ADDRESS structures around the world cause most enterprises problems. For example, an address in Moscow is very different from an address in Bangkok. Each different postal organization in each different country seems to have a slightly different addressing structure for their jurisdiction. To deal with this fact, the data team uses the Contact Mechanism with Flexible Address Parts Pattern from Chapter 7 to allow great flexibility when maintaining international addresses that may cover many different jurisdictions. Imagine the board of directors wishing to send their annual report to all of the shareholders around the world and not being able to put the correct address structure on the envelopes. This pattern enables the business to handle international addresses in a very flexible manner.

A very significant aspect of master data management is the ability to maintain business rules. Thus, the data team decided to use the Level 3 Business Rules Pattern from Chapter 8 to have a great amount of flexibility to maintain business rules. A common business rule for master data management is how to know when a customer from one source system (for example, Kathy Morris at 100 Main St., New York, NY) is the same as a customer from another source system (for example, Kathy Moris at 100 Main Street, New York, NY). For example, there may be a MATCHING RULE (a subtype of BUSINESS RULE) instance, maintaining that if the customer has the same exact name (one generalized factor maintained RULE FACTOR TYPE) or the name can be matched using a common set of synonyms

(another generalized factor), and the contact mechanisms are the same (another generalized factor), then the customer is considered the same (an outcome maintained in RULE OUTCOME TYPE).

Other business rules in the master data management model may be applicable to particular types of roles, such as customer, supplier, partner, worker (including employees and contractors), and so on. For example, this structure can maintain business rules such as each customer must have their credit rating re-evaluated every year, or each customer in Europe must have a valid EU tax id. These could be examples of CUSTOMER RULE(s) (a subtype of BUSINESS RULE). Another important set of rules about customer data is what data can be shared as master data to whom. For example, some customer master data may not be shared between different parts of the business due to specific regulations.

Note

As we noted earlier in the chapter, capturing and mapping the sources of data was beyond the scope of this initial data model, although we recognize that this is important. While we can capture aspects of this requirement using the Level 3 Business Rules Pattern, there are specific data models that may be developed for this requirement, such as a cross-reference entity between a master primary key and source systems keys. There are many different approaches for this, and it is usually considered a physical database design consideration. The data team is interested in capturing the master data management requirements and ensuring that the MDM solution is flexible enough to meet the needs of the enterprise. In other words, Figure 9.8 doesn't directly address the modeling of ‘sourcing and tagging’ of data back to the original source systems, cross-referencing of data among the source systems, and which data should be considered the system of record in this section.

Why Do We Do It This Way?

When the data modeling team modeled the customer master data for Sands Distribution, they needed to take several different considerations into account.

First, they needed to ensure that they had a flexible enough solution to capture all the current and future data that made up the customer master data and to accommodate formats from all different types of source systems. They achieved this by using very flexible data model patterns (level 3 patterns, for the most part). For example, you see in Figure 9.8 the Level 3 Declarative Role Pattern, the Level 3 Status Type Pattern (for PARTY ROLE), the Level 3 Contact Mechanism Pattern, the Level 3 Classification Pattern (for PARTY ROLE), the Level 3 Recursive Pattern with Rules (for PARTY ROLE), and the Level 3 Business Rules Pattern (for ROLE TYPE). These patterns were supplemented with additional patterns offering more flexibility, such as the Status Category Pattern, the Status Type with Multi Rollup and Rules Pattern, and the Contact Mechanism with Flexible Address Parts Pattern.

Second, the customer master data needed to capture various data integrity rules to facilitate better data quality. For example, customer status data was modeled using the Status Type with Multi Rollup and Rules Pattern, allowing a robust set of status data where data integrity rules could be applied to ensure that appropriate statuses were applied. In this model, each STATUS TYPE ASSOCIATION may be **constrained by** a STATUS TYPE ASSOCIATION RULE, such as the statuses “Active” and “In abeyance” should not exist at the same time within the STATUS TYPE CATEGORY of “Customer purchasing status.” Another way that flexible associations and rules were maintained was to use a Level 3 Recursive Pattern with Rules to model the PARTY RELATIONSHIP structures (hierarchy, aggregations, and peer-to-peer associations). This allowed the ability to find out who is related to whom in any way, such as for use in

social networking analysis. It also allowed them to maintain constraints on relationships, such as a certain salesperson should not be associated with a particular customer. Finally, the application of the Level 3 Business Rules Pattern allowed many other business rules to be maintained in the data model.

Figure 9.8 can be used as part of a statement of the needs of the business regarding what it needs for its master data management solution. For example, the classification data model structures indicate that it is important to have a highly flexible way to classify and re-classify customer data into different categories and types of categories. Furthermore, this model shows that each of these categories and types of categories may have multiple levels and hierarchies (or aggregations) that may be part of a specific scheme and/or may come from a certain provider. Also, Sands Distribution could decide to directly use this model (within an MDM prepackaged solution or for a custom MDM implementation) as the basis to implement its master data management database solution.

What Are the Strengths of Using Patterns for the Solution?

The strengths of using patterns for a master data management data model are as follows:

- **Accommodates many different types of source system data structures:** The generalized patterns that were illustrated in this example can accommodate a great variety of source system database structures. For example, source system data structures may maintain contact information in many different ways, such as having particular fields for phone number, having a PHONE NUMBER table, having a CONTACT INFORMATION table, and so on. By using a very

generalized pattern for contact mechanisms in the master data management data model, you can accommodate all these various structures from various source systems. This could be considered a “least common denominator” model that addresses the needs of almost any type of source system data.

- **The use of more generalized patterns offers a great amount of flexibility to accommodate current and future needs:** For example, if any new types of classifications, roles, statuses, rules, or relationships need to be captured for the customer master data, these can be accommodated, and the underlying model can remain the same.
- **The various rule patterns offer a way to dynamically maintain many business rules that can help control the integrity of data:** For example, business rules were captured that maintain the criteria or factors regarding how matching of similar data should occur (this is the MATCHING RULE). This data model structure can maintain a great number of other rules, such as rules about the requirements for customers or other party roles. Also the rules governing status classifications or relationships were maintained. This helps facilitate data quality and data governance.
- **The same level of patterns may be used throughout a model, allowing for a more consistent approach to managing master data:** A model can reapply the same type of patterns regarding classifications, recursive relationships, and rules in many areas of the same data model. For example, the data team saw that they could consistently use the Level 3 Classification Pattern for the classification of party roles (including customers), contact mechanisms, statuses, and business rules.

- **Standardization and potential for reducing costs:** If these patterns become standardized throughout the enterprise, the model can help with development of common code, routines, and services, thus reducing the time and cost of development and systems maintenance. The data team used the same type of patterns that they used in application systems, the enterprise data model, and data warehousing.

What Are the Weaknesses of Using Patterns for the Solution?

The weaknesses of using patterns for a master data management data model are as follows:

- **Difficult to understand when more generalized patterns are used:** If a data professional is not familiar with the patterns, reading the data model can be difficult initially, even for experienced data professionals. It can seem complicated to the uninitiated or untrained. The model produced by the data team was meant to describe the data requirements for the purchase of a MDM solution. The generalized patterns in this model may not be the most effective way to describe specific requirements that Sands may have for this solution. Also, the data team would have had to 'walk' each of potential vendors through the model to describe the data requirements that they needed fulfilled. These requirements may not be self-evident from the model unless the vendor is also familiar with patterns.
- **Mapping from the various source systems to this model can be more complex when using more generalized patterns:** Mapping, integrating, and synchronizing data from many sources can be more challenging using these generalized patterns because

they are more complex (unless the enterprise has standardized on the use of these types of patterns in its other applications and then it can actually be easier to map).

- **Many specific rules are not inherent in the relationships and attributes of the data model when using more generalized patterns:** Generalized data models offer flexibility, but sometimes at the expense of capturing specific requirements. By using the generalized patterns to create these models, you may not capture specific attributes and/or relationships. For example, a PRODUCT FAMILY classification (Level 2 Classification Pattern) may have some specific attributes or relationships that apply only to PRODUCT FAMILY. By using a Level 3 Classification Pattern, you may not capture the need for these specific attributes or relationships. Or there may be a rule that a CUSTOMER may only be classified into one CUSTOMER SIZE. By using the generalized pattern, you do not enforce this rule directly in the data model because you allow customers to be any number of sizes.

Synopsis

Master data management solutions need flexibility to manage data from multiple different sources and also have the need to create rigorous rules around the master data in order to improve data quality.

The master data management data model mainly used the level 3 patterns to create flexibility for maintaining data about the customer (as well as other various parties involved), party classifications, contact information, relationships between parties, and statuses of parties.

The master data management data model also included a way to maintain BUSINESS RULE(s), which may help to manage the master data in many different ways, such as

who is allowed to see certain types of master data, how parties are matched, and rules about customers and other party roles. Other rules are maintained by using the Level 3 Recursive Pattern with Rules for STATUS TYPE(s) and PARTY RELATIONSHIP(s). Thus, the flexibility of the patterns also gets constrained by rules that manage the relationships that may exist in the reference data.

This is a complex data model that utilizes many flexible data patterns. Because the models use many generalization concepts, some audiences may have a harder time understanding this type of model. Some data professionals may feel that modeling rules within the master data management data model is overkill. Though that may be the case in some enterprises, other enterprises (rules-driven enterprises) can use these powerful constructs to help ensure the integrity of their master data while providing a very flexible foundation for master data management that can support a very wide variety of needs from various source systems.

Other Thoughts Regarding Using the Patterns

In this section, we address some issues that may also affect which patterns you choose by discussing some common physical database design concepts and some other ways that patterns may be used.

Physical Database Design

Although some of this chapter used some physical database concepts, such as the star schema section, the majority of this chapter did not address translating these models into a physical database design. There are many considerations

for translating these models into physical database designs, such as:

- **Volumes of data:** How many instances are there of the various entities?
- **Frequencies of data access:** What types of create, read, update, and delete activities are expected?
- **Physical database design standards:** Similar to data modeling standards, physical database design standards are subject to different styles, such as different naming conventions; for example, some physical design standards limit the name lengths of tables or columns, which is often based on the database platform used.
- **Implementation of subtypes:** Subtypes can be converted into tables in four basic ways:

  1. The supertype and subtypes become two tables: one table for the ‘data’ and one table for a ‘type’ table. However, in most of our patterns, the ‘type’ entity already exists, so this would mean that the attributes and relationships of the subtypes would be inherited in the supertype (or ‘data’ table), and there would only be one table.

  2. The supertype becomes a table, and each of the subtypes becomes a table.

  3. Only the subtypes become tables, and the attributes of the supertype are passed down into each subtype.

  4. A hybrid or combination of these, for example, creating a single combined table for the supertype and one subtype, and then tables for each of the other subtypes.

We do not go into detail concerning these or other physical database design issues because this is beyond the intended scope of this book; it is independent of the usage of universal patterns, and database design principles are

applicable regardless of whether or not patterns are used. Additionally, a plethora of books cover this subject.(10)

Other Applications for Patterns

We have covered many common applications where these patterns may be used. However, the patterns are useful in numerous other applications. For example, these patterns could be used to develop standard ways of passing data and/or to create common or ‘universal’ XML schemas. These patterns can also be helpful in establishing common data semantics regarding data governance programs. Within the data governance program, the data team could develop one set of common data semantics for business representatives using level 1 patterns and another set of data semantics for technical representatives using level 3 and 4 patterns. Wherever there is a need to understand and model data, modelers can use these patterns to jump-start models, compare alternatives, apply consistent constructs, and quality assure their models.

Other Considerations When Using Generalized Patterns

You have seen in this chapter and in this book many of the great advantages in flexibility, consistency, reusability, and power that generalized patterns provide. We provide both specific and generalized patterns in this book because, as we have stated throughout this book, there are pros and cons when choosing to use generalized patterns versus specific patterns. There are some other considerations that we thought were worth mentioning regarding using generalized patterns, such as(11):

- Generalized structures move the change process from the typical ‘data architect to DBA to developer to tester’ process to a data change process, and often

organizations have no formal process for data change like they have for application change. This can be addressed with organizational commitment to flexible data modeling structures. In fact, we believe that using generalized patterns is an important step in empowering users to take control of their data. The typical data architect-DBA-developer-tester process is often too cumbersome and slow for users' needs. Generalized patterns force a paradigm shift away from this traditional development process.

- Data stewardship is important when using generalized data patterns because new instances can have a large effect on the model. For example, there may be new instances that could result in new roles, classifications, statuses, contact mechanism types, or business rules. Data stewardship is about assigning parties to be accountable for the data, and generalized models require more management of the instances in the data model. An advantage to using more generalized patterns in data stewardship is that they often provide the ability to add or change new types of data without changing the model, and therefore can empower the data steward. If data stewardship is planned and budgeted, and if an enterprise invests in data stewardship, it can realize more of the benefits of using flexible data structures.
- The main benefits of using a more generalized approach are realized when business changes happen, but typically project teams are not compensated for meeting future needs. This disconnect can lead to frustration and a feeling of ‘overdesign.’ With guidance and a commitment to communicate, you can address this disconnect and show the enterprise the benefits of generalized data structures.

- Sometimes DBAs or developers may raise the question of whether a generalized data model performs as well as a specific data model. In our experience, the main reasons for good or bad performance have to do with factors other than the level of generalization.

Summary of Using the Patterns

Table 9.1 contains a synopsis concerning using the patterns.

Table 9.1 Synopsis of Using the Patterns

MODEL	DESCRIPTION	STRENGTHS	WEAKNESSES
Data model for scope statements and understanding data requirements\ Data model for prototype	Very specific data models using level 1 and level 2 patterns. The objective here is twofold. First, to create a very understandable model that can be used to gather and verify data requirements. Second, to create a model from which it is easy to build a quick prototype in order to illustrate the data requirements so that changes can be addressed.	Easy to understand. Easy to use in developing data models. Easy to implement for a prototype. Good starting point for developing data models. Effective for communicating to business representatives. Simple to change to a higher-level pattern.	Using level 1 patterns does not generally provide a good basis for implementations. Lack of ability to reuse common routines. Lack of flexibility.
Application data model	A data model that is designed to meet the needs of a full-blown production application. This is a data model that needs to meet current and future application needs. It generally uses level 2 patterns when there are well-known static data requirements and level 3 patterns for parts of the application that require more flexibility.	Balance of specific and generalized patterns Flexibility Meets specific needs of application Common, integrated solution Consistency by re-using the same types of data model structures provided by the patterns	Inflexibility in some parts and complexity in other parts Multiple styles of patterns and, therefore, multiple styles of modeling Lack of understandability when using more generalized patterns Lack of some business rules specifications when using more generalized patterns

MODEL	DESCRIPTION	STRENGTHS	WEAKNESSES
Enterprise data model	In this section, we described an approach in developing an enterprise data model that allows the same type of data to be modeled multiple ways, using different levels of the patterns. Thus, the patterns can provide standardized data model structures and sometimes allow an alternative because different applications sometimes have different needs.	Quality data model constructs for common requirements. The patterns provide consistency across the enterprise data model. Leverage in providing alternatives in enterprise data modeling. Saves time in the enterprise data modeling effort.	Could, at first, cause some confusion to have alternative models for the same type of data. Different levels of patterns in same data model can lead to inconsistencies.
Relational data warehouse data model	This is a very flexible approach for the creation of enterprise data warehouse models that would be considered more relational. The enterprise data warehouse design is based on a relational data model, usually an enterprise data model. This is sometimes related to the idea of applying a "hub and spoke" architecture (as used in the "Corporate Information Factory") as opposed to a "bus architecture" (used in Kimball's "Data Warehouse Toolkit" approach).	The level 3 patterns provide a very flexible solution for maintaining decision support data. Consistency - The same level of patterns are used throughout this model. Capturing of history - The level 3 patterns accommodate tracking of how relationships have changed over time.	Using the more generalized patterns results in a more complex model. The patterns do not accommodate all types of history needed for data warehousing.

MODEL	DESCRIPTION	STRENGTHS	WEAKNESSES
Star schema–based data warehouse data model \ Star schema design	This is an approach that uses a data model as a precursor to a star schema design. By using the patterns in this data model, you can better understand the data and possible complexities that may be inherent in the dimensions. Thus, this approach allows you to see the complexity that needs to be resolved in the star schema design.	More solid and stable star schema design with fewer mistakes. Consistent designs for more consistent ways to develop dimensions. Better understanding of the data by first developing a data model using common patterns.	The data model that is used as the basis for the star schema sometimes does not show true data relationships. Some consider using patterns for the development of a relational data model an extra step that is unnecessary for star schema development.
Master data management	In this section, we created a master data management data model that provided the flexibility to deal with multiple different sources (and destinations) for master data and also maintained parameters about rules regarding the master data. The master data was captured mainly using the level 3 patterns and maintained information about the customer (as well as other parties), contact information, relationships between parties, classifications of parties, and statuses of parties.	Accommodates many different types of source system formats. The use of more generalized patterns offer a great amount of flexibility to accommodate current and future needs. The various rule patterns offer a way to dynamically maintain many types of business rules that can help control the integrity of data. The same level of patterns may be used throughout a model, allowing for a more consistent approach to managing master data. Standardization and potential for reducing costs via common services and routines, if the patterns are consistently used and standardized throughout the enterprise.	Difficult to understand when more generalized patterns are used. Many specific rules are not inherent in the relationships and attributes of the data model when using more generalized patterns.

References

[1] See Competing on Analytics: The New Science of Winning by Thomas H. Davenport and Jeanne G. Harris (Harvard Business School Press, 2007).

[2] See Chapter 3 of this book for more on the Hybrid Contextual Role Pattern.

[3] See Chapter 10 of this book for more information on how to socialize and gain buy-in for these patterns.

[4] See Building the Data Warehouse, Fourth Edition, by William H. Inmon (Wiley, 2005).

[5] See The Corporate Information Factory, 2nd Edition, by W. H. Inmon, Claudia Imhoff, and Ryan Sousa (Wiley, 2000).

[6] See The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, Second Edition, by Ralph Kimball and Margy Ross (Wiley, 2002).

[7] "Considerations for Managing Risk in a Post-SOX Environment" by David Friedland in The Data Administrator Newsletter, published April 1, 2007. Accessed at http://www.tdan.com/view-articles/4937.

[8] A term used in Demystifying Master Data Management by Tony Fisher, CIO Magazine (CXO Media Inc., 2008).

[9] See Chapter 2 of The Data Model Resource Book: *Revised Edition, Volume 1: A Library of Universal Data Models for All Enterprises*, by L. Silverston (Wiley, 2001).

[10] An example of a useful book on physical database design considerations is Physical Database Design: The Database Professional's Guide to Exploiting Indexes, Views, Storage, and More, 4th Edition, by Sam S. Lightstone, Toby J. Teorey, and Tom Nadeau (Morgan Kaufmann, 2007).

[11] Some of this material regarding other considerations for generalizations was kindly provided by industry thought leader, Karen Lopez.

Chapter 10

Socializing the Patterns

Okay, you now have some pretty powerful patterns to help develop quality and consistent data models that can be used for many purposes. Some of these purposes, such as using them for enterprise-wide models in enterprise data management, enterprise data warehousing, data governance, and master data management, deal with enterprise-wide data integration.(1) To be successful at enterprise-wide data integration, you need to gain buy-in and get people to move in the same direction. Likewise, if the patterns are used at more of a specific application level, you still need to gain buy-in so that these patterns can be adopted and used on specific projects. The patterns are useful tools to help you build your data models and data architecture. However, after you've made the decision to understand and use patterns, you still need to address some important questions about socializing the patterns in your enterprise; questions such as the following:

- What is an appropriate balance between requiring adherence to the patterns and allowing them to be used completely optionally and used if and when they are helpful to the modeler/designer?
- How do you get your enterprise and various people in the enterprise to adopt these patterns? For example, how can you show the value of these patterns to your senior management or how do you show their value to project managers who are constrained by tight deadlines?

- What types of policies or principles regarding use of the patterns would be most appropriate to get the most benefit from these patterns? For example, should the patterns be part of an initial training program for new IT employees? Should the patterns be part of your standards or methodology documentation? What is the policy regarding how much leeway a project has on whether they need to use or not use these patterns?

What Is the Significance of Socializing the Patterns?

When we say *socializing the patterns*, we are referring to how you get the patterns accepted and used appropriately in your enterprise. The intent of this chapter is to provide some principles and suggestions that we believe have been extremely helpful to other enterprises in socializing these patterns. The ideas in this chapter are applicable for socializing both the patterns in this book and the models in *The Data Model Resource Book*, *Revised Edition*, Volumes 1 and 2 (Wiley, 2001).

The principles and recommendations discussed in this chapter may seem like common sense; however, in our experience they are not so common! We have seen many cases where using these principles and suggestions have greatly helped enterprises gain much more benefits from the patterns. There may be different levels at which these principles may be followed, and our intent with this chapter is to sharpen your skills in this critically important area of human dynamics, which may have as much to do with the success of applying these patterns as the intrinsic value and strength of the patterns themselves.

What Is in This Chapter?

This chapter provides suggestions regarding the human dynamics of socializing the patterns along with scenarios illustrating how these human dynamics influence the success or failure of these patterns with various enterprises. This chapter includes:

- Different scenarios regarding our experiences using the patterns with a focus on two main narratives: one about an enterprise that had difficulty in adopting and standardizing the use of the patterns and another enterprise that very successfully socialized the patterns, resulting in adoption and standardization of them across the enterprise, which led to great benefits.
- A discussion of four principles that we think make the biggest difference in socializing the patterns and getting them adopted. These are:
  - Understand motivations and work toward meeting them
  - Develop a clear, common, compelling vision
  - Develop trust
  - Manage conflict effectively

  Sometimes we refer to these as “Universal Principles” because just like Universal Data Models and Universal Patterns they “apply to a great variety of uses: comprehending, affecting or extending to the whole” (from a Webster's dictionary definition of *universal*). In this chapter, we focus on applying these principles to patterns; however, the principles could be used to socialize or gain buy-in for anything.
- Additional comments on socializing the patterns such as discussing the need for the patterns to be socialized for many types of circumstances, the role of upper-level management commitment, and the estimation of the return on investment for using the patterns.

Experiences Using and Socializing These Patterns

Even with all the benefits that patterns provide, some of our clients have had difficulties getting their enterprise to adopt these (or any) patterns. Why is this? Perhaps it is best to illustrate the answer to this question with two experiences of enterprises that wanted to socialize the patterns across their enterprise. Though we describe real-life scenarios, we will use fictitious names for these enterprises for confidentiality reasons.

First, we want you to examine the experiences of a large, international enterprise trying to socialize the patterns across the enterprise. For this example we refer to the company as *Company A*; however, you would probably recognize the company's name if we stated it. This enterprise's data management staff embraced these patterns and recognized their value. This enterprise had huge data issues. Those involved with each project developed and viewed their data and databases very differently, and this created many data inconsistencies, data quality problems, communication problems (because each project had completely different semantics concerning some data), and high maintenance costs because the enterprise spent a lot of time reconciling data across systems. This company also acquired smaller companies, whose data had to be integrated into the enterprise. The data management staff saw that the patterns could be used to promote consistency across the enterprise's data and could help develop more commonality regarding how they defined, maintained, passed, and viewed data. They also knew that the patterns could help improve quality by pointing out effective ways to model in various systems and by pointing out the pros and cons of various modeling approaches. They knew the patterns could help simplify data interfaces,

reduce maintenance costs, improve communication, and facilitate better understanding of the data, all of which could help to provide better data for better decision making.

However, they faced many difficulties when they tried to get these patterns adopted across the enterprise. Some of these were as follows:

- When the enterprise architects identified projects that could use these patterns, the project members often wanted to use their own ‘patterns’ and their own expertise and did not want to be bound by any predefined solutions coming from the enterprise architecture area.
- Members on individual projects and applications looked at their data, analysis, and modeling from their own specific perspectives. They resisted many of the concepts and ideas that form the basis for the patterns. For example, many involved with the projects resisted the idea that ‘parties’ could have many ‘roles,’ and just said that ‘their’ data was about “their” customers or ‘their’ suppliers. That was all that was important to them. They often looked at their specific data including the application database they were using to identify attributes. So if they identified an attribute such as **order confirmed date**, they did not want to think of this as a type of status. If they identified an attribute such as **customer size**, they did not want to think about this as a potential classification, because they felt that these generalizations were not needed for their specific application. If there was a hierarchy between a parent and subsidiary customer, they definitely did not want to think about this as a recursive relationship that could accommodate other types of relationships between customer enterprises.
- Those involved with individual projects and applications were motivated to be on time, within budget, and meet

their specific requirements for the project. They saw the enterprise architects and their patterns as roadblocks that created project risk by slowing the project down. This is much like the dynamic where someone building a house views the housing architectural committee as a roadblock to doing what they want with ‘their’ house. In other words, the standards of enterprise architects, including patterns, was at best only paid lip-service, at worst completely circumvented.

- Some project members viewed these patterns as theoretical and abstract. The enterprise architectural team tried to show that the patterns could be either very specific or abstract and that they provided choices; however, project members often did not believe this. They thought the enterprise architects were removed from reality and the specific deliverables that were needed for projects.
- Even if project members did not view the patterns as too abstract, many felt that their project data requirements were unique and they did not want to be bound by any preconceived notions or ideas.
- Many of those involved with projects said that because they buy off-the-shelf application packages, they did not need patterns because they did not even need to model their data. They also said that they did not have control over the data structures and they needed to just understand and work with the data in their packages.
- Some involved with projects said they were focused on a specific type of project such as data governance, master data management, data warehousing, and XML development and that these patterns were not specifically designed for that type of effort. Therefore, they did not want to use the patterns.

Ultimately, the management of the enterprise increased the pressure for the enterprise data group to show real

business value (sound familiar?). With the difficulties that the enterprise data group was facing, management made decisions to cut funding in the enterprise data group because they had to cut somewhere, and they perceived that the enterprise data group was not delivering as much value as other groups. Even though the patterns did offer great benefit to the enterprise, after a number of years, this enterprise was still not able to socialize the patterns and standardize data constructs in a significant way across the organization. Many of these data issues have continued to plague this enterprise.

Now take a look at another enterprise (one we call ‘Company B’) that had a different experience. This enterprise also embraced these patterns and saw their value from the start. They were also experiencing many of the same data issues as ‘Company A.’ Because the enterprise was an international, fast-paced, entrepreneurial enterprise with many branches and locations, each new application developed its own data with its own data stores causing many data silos. Because the enterprise was growing rapidly, they needed to get things done quickly and did not make time for many controls or standards when they first developed systems. Later on, when there were very specific, high-priority needs, the enterprise developed programs to synchronize or coordinate data between specific applications, and these programs kept on growing in number. They bought a lot of third-party application packages, and most of the time they did this without a lot of consideration for how the data was integrated across the enterprise. Thus, these issues caused data inconsistencies, the inability to see a complete view of the data (for example, customer data), more complex processes, increased maintenance costs, confusion, communication issues, and many mistakes based upon incorrect data.

The data management area of this enterprise saw the need for Universal Data Models and Universal Patterns and adopted many of the models described in the first volume of this series(2) and many of the patterns that are described in this book. The enterprise stated that the intended purpose of the universal data models and patterns was to be of service to the various projects and to help them save time and increase data model quality, while also helping to integrate data at the enterprise level. This enterprise faced many of the same challenges as the previous enterprise we described. They also had some additional challenges such as the following:

- The industry drastically changed during the time that the enterprise was attempting to use these universal models and patterns to integrate data. These changes adversely affected the business, and they had major financial cutbacks and were under a great deal of pressure to survive.
- Due to these pressures, there was a short-term focus and a project focus in their information technology (IT) areas.
- Projects were given a great deal of latitude and autonomy in developing their systems.

This enterprise was able to change their situation and create significant movement toward a standard, agile, integrated data landscape with universal models and patterns. As the years have progressed, the organization has gained more and more momentum toward integrating their data, improving the quality of data, simplifying their systems, reducing costs, and facilitating information that allows better service to their customers. This has helped the enterprise not only to survive but also to prosper, as it is much easier for this enterprise to effectively compete using analytics, integrate data from various systems, and meet the growing data demands of its business.

What Makes the Difference In Socializing the Patterns?

What were the differences between the two efforts to standardize and socialize these patterns we just presented? Why do some efforts to use common models and patterns produce significant business value and others falter?

For the past several years, we have looked at many organizations attempting to standardize and socialize these patterns. We have asked ourselves, “What makes the difference between successfully standardizing to produce effective business results and unsuccessfully attempting to standardize and in the process, even hinder progress?” We offer the following four principles that, based upon our experiences and research, seem to be key to successfully using these patterns and to successfully socializing and standardizing these patterns either across an enterprise, or within a particular area of an enterprise:

- **Understand motivations and work toward meeting them:** Really understand the motivations of people who would use the patterns, your motivations, and the enterprise's motivations as well as the underlying culture of the enterprise, which often drives these motivations. We model data to understand it, and in this chapter we will discuss ‘modeling’ motivations, and share other techniques to understand motivations. By modeling these motivations and fine tuning our understanding of these motivations, we can keep these in the forefront and develop solutions that meet everyone's needs.
- **Develop a clear, common, compelling vision:** Develop and communicate the purpose for these patterns as well as a vision of what it would be like if various parts of the enterprise used these patterns. Create the purpose and vision in a way that is well

understood (clear), commonly agreed upon (common), and that is very motivating (compelling). This vision should resonate with all levels of the enterprise, technical and non-technical people, and echo the vision of the enterprise as a whole.

- **Develop trust:** Why should people in various parts of the enterprise use these patterns? Are they proven? Can people trust the patterns? Can people trust the intentions of the people who are promoting these patterns? You need the people in the enterprise to feel they can rely on the patterns and that they can also rely on the people who are advocating using the patterns.
- **Manage conflict effectively:** Conflicts are not necessarily bad and this is good because conflict is bound to happen. If the people who are socializing the patterns (as well as others using the patterns) are skilled in how to work through conflicts, there is a much greater chance for success in socializing the patterns.

Understanding Motivations—Why Would Someone Use or Not Use the Patterns?

Some enterprises, such as the one that we first described at the beginning of this chapter, try to socialize these patterns by declaring that these patterns *will* be used by any application that can use them and in all situations across the enterprise. They work within their organization to get architectural policies instantiated that declare that all data modeling efforts must use the common models or patterns from the data architecture group (or else…!).

Although this sounds like it could be an effective strategy to socialize these patterns and although these policies can possibly help, we have seen this type of strategy yield only

limited success. Why is this? David Hawkin's book *Power Versus Force*(3) provides a definitive and comprehensive argument that effective and healthy results come from *real* power, or in other words, from holistic, healthy, positive, service-oriented actions, rather than from trying to control and force things to happen.

When people and teams feel that they are forced into solutions, there is often pushback. One of our clients mandated that project teams use the patterns "*no matter what!*" Project teams then used them to create logical data models for the project just to pass the architectural requirement and to show compliance; then teams developed their own physical models and/or physical implementations that didn't look anything like the logical data models or patterns. The project teams felt that they could meet these architectural standards that were imposed on them and then could develop their database designs the way that they wanted. If policies requiring adherence to the patterns are dictated without regard for the needs of projects or other parties, there are often many loopholes, and project teams will generally find ways around the policy to meet their own needs.

A different approach is to focus on understanding the motivations of people who may use the patterns. Typically, these patterns would be used in data modeling for some type of effort such as a data warehousing, master data management, data governance, specific application, or data interface effort or some other effort where there is a need to model data. What are the motivations of the prospective audience? The answer to the question varies based upon the culture of the enterprise as well as the individual people.

In the book *The Sedona Method*(4) Hale Dwoskin talks about four basic wants: control (I want it my way), approval (seeking love and acceptance), security (survival), and separateness (proving that I am different, better, or special).

Many times people are driven by these core wants, and it is important to understand what is driving each person when socializing the patterns. For example, a data modeler on the team may want to just do his or her own thing and have creativity regarding developing the data model, so he or she does not want to use patterns or even be bothered by them. The core, underlying motivations may be to be separate (that is, I am very creative, and I can do it better, which is also the 'not invented here syndrome') and also control (that is, I want to do it my way). So it may be appropriate in this case to encourage creativity and stress that the patterns do not replace creativity or proper information requirements gathering expertise. They just provide useful tools to help when they can, just like reusing standard application functions in programming to increase productivity. You may also stress to the data modeler on this team that the enterprise data team values his/her input and welcomes new and creative ways to use the patterns and/or suggestions to enhance the patterns.

Note

The book *The Sedona Method* explains that each of the four basic wants have built-in opposing forces that are also potential motivations for people. For example, people may have the motivation of "wanting to be controlled" so they are off the hook (opposite of control); of "wanting to give away love and/or approval" when people just can't accept approval (opposite of approval); of "wanting to die," which may be similar to giving up on the job because they are fed up (opposite of security); and of "wanting to be one," which is about being connected (opposite of separation). It is important to also consider if these are motivations that are also at play.

Another example may be that there is resistance to using the patterns from a long-time employee who does not want any major change. In these days of outsourcing, there may be an underlying motivation that if patterns simplify the job

of data modeling, then perhaps the employee may feel that this could be another reason to outsource data modeling, and therefore, it could jeopardize his or her job (thus relating to the core want of security or survival). If this is the case, it may be wise to point out that understanding patterns in data modeling can greatly strengthen one's data modeling skills and, thus, provide much more security to continue performing a job. Or you could suggest that knowing how to successfully implement the patterns enhances their importance to the enterprise.

Yet another example may be that a database administrator on a specific project wants to make sure that the database structure performs well because he or she is evaluated based upon the performance of the database. Here, the idea of approval as well as survival (because his or her job may depend on a good evaluation) may come into play. A key point that may be stressed in this situation is that generalized patterns do not equate to poor performance and a great deal depends on many other factors. Also, the patterns could be positioned as a method to show alternatives that can be evaluated, in order to help to produce a better outcome (for the system as well as for the database administrator's evaluation).

Rather than forcing people to reuse patterns, we suggest clearly understanding, identifying, and articulating the motivations of others. Doesn't it make sense to identify clearly what others want? We have perceived that, quite often, this step is missed, and people are not completely aware of other people's true motivations. If you can see the various motivations at work, you are in a much more powerful place to take appropriate actions. It sounds so incredibly simple; however, in our experience, many people either do not take the time or they are not able to comprehend other people's motivations.

Though it is important to understand an individual's motivation, it is also important to understand a group's collective motivation. We find that groups, such as a system development teams, have the same four basic wants also (control, approval, security, and separateness). For example, a group of very skilled data professionals may consider themselves as the ‘A team’ that is different from the rest of the development teams in an organization (that is, separate and special). You could stress how the patterns are cutting-edge technology for your enterprise, and they will be trailblazing for the rest of the IT professionals in your enterprise. Groups of people may also feel under threat (security) by this new approach. Often, in a tight-knit group, the feeling of insecurity gets magnified as it feeds off each individual's insecurities. It is crucial that you understand a group's motivation and address the group as a whole, not just some of the individuals in the group. Your good work with one individual may be undone by another member of the group. It is also worth noting that the motivations of the individuals may be very different than the motivations of the group. Using the example we just gave, the ‘A team’ group's motivation was to differentiate themselves; however, individuals on the team may be motivated by security, control, and approval. It is important to address each of their core motivations individually, but also address the motivations of the group as a whole.

Author Don Herald states “Unhappiness is in not knowing what we want and killing ourselves to get it.”(5) This points out that it is also important to understand your own motivations. We often think we understand our own motivations; however, based upon our experiences, which includes conducting many workshops and seminars on this topic, it seems that we often do not even understand our own motivations. For example, what is your motivation for using patterns? The answer may not be as obvious as you

think, and it may be more important to first understand our own motivations. During Universal Data Model workshops and seminars on this topic, many data modelers and data architects have revealed that they thought they knew their motivations and then realized that their motivations were actually something else.

The following paragraphs in this section will describe some techniques that can help better understand our and others' motivations. Specifically we will now share three techniques to better understand motivations: “the five whys,” “motivation modeling,” and “three ways to understand motivations.”

“The five whys” is a very powerful technique for understanding the root cause of something or understanding motivations at a deep level, which is where we really need to focus to be successful. When you first describe a motivation, there are often hidden motivations underneath. For example, if a project team member refuses to use common patterns because he or she says that his or her project's requirements are unique, there may be a hidden motivation underneath about feeling in charge and in control of the effort. One technique for discovering hidden motivations is to continually ask “Why?” In the 1970s, Toyota Corporation used a technique called “the five whys” to improve production. (This is now a incorporated as a tool within the “Six Sigma” strategy and methodology.) The technique involves continuously asking “why,” in order get to the core motivation or issue. People at Toyota found that they were able to really understand the root issues involved, only after digging deeper by consecutively asking variations of this question and they found that, on the average, they got to the root issue on the fifth time that they asked “why.”

To apply this technique to the socialization of patterns, we could start with the question, “Why would we be interested in socializing these patterns?” If the first response to the

question is “to use these patterns to facilitate consistency for any data modeling effort and move toward data integration,” then ask “why” do you want to facilitate data integration? The answer may be because by facilitating data integration, things in our company can be more organized. Then ask “why” do you want things more organized? The answer might be because if things were more organized, then our company would be more efficient. “Why” do you want more efficiency? Because if I can help our organization become more efficient, then I can help our company to grow and prosper. “Why” do you want the company to grow and prosper? Because if the company grows and prospers, then I can grow and prosper also. Okay, so the core motivation, in this case, is really to help the company grow and prosper and in turn help *you* grow and prosper. Knowing this motivation provides clarity and helps to keep one's eye on the *real* reasons that people do what they do.

Understanding the motivations of people and organizations helps you tailor how you socialize the patterns. For example, in the previous example, it would be easy for you to socialize the patterns by saying that they will help you facilitate data integration, when in fact the more powerful message is that patterns will help you and your organization grow and prosper. By understanding the core motivations of your audience, you can make your vision for patterns resonate more deeply with that audience. We address this more deeply in the next section on purpose and vision in this chapter.

Another technique that provides a great deal of insight is “motivational modeling.” When you consider the question “What do you want and why?” you may think, “Of course, I know what I want!” If that is the case, then write it down! Usually, the first thing that someone thinks or writes about their own motivations is not the core motivation. Thus, if you write it down, you can clarify it, refine it, and in a sense,

‘model’ both your and others' motivations while continuing to understand them.

During some Universal Data Model workshops, in addition to teaching data modeling, participants also learn how to do “motivational modeling.” Participants are given template motivation modeling worksheets where they ‘model’ and fine tune their own and other's motivations. They include questions (and of course, boxes and lines) such as what are your motivations, what perspectives do they have (e.g., professional, personal), and how does your program/project help you with these motivations? We then model other people's motivations and show how to ‘model’ the relationships between people's motivations. This is very similar to data modeling! To understand data, we model it, so why not do this with motivations? We talked about the Zachman Framework in Chapter 1 and essentially, while the data model is column 1 and answers the ‘What’ question in Zachman's Framework, the “motivational model” is a model for column 6 in the Zachman Framework(6), answering the ‘Why’ question!

If you are game, try this: Write down your motivation for working on your program/project on a piece of paper. Then ask why you want this. Then write down the answer and again ask why five times and see where you end up. If you want to go further with this, try ‘modeling’ someone else's motivations.

So what if you don't know other people's motivations? Three ways to find out motivations are to ask, observe, and test.

- For example, you may simply ask a data modeler on a project what his or her motivation is, they may answer, “To save time.” Then you can ask a specific question that speaks to that motivation, such as whether he or she would view a jump-start using the patterns as a benefit toward saving time.

- Just as you are not always aware of your own motivations, sometimes other people aren't aware of what their core motivation is, or they may not want to share it. Thus, you can often observe motivations. For example, if a project data modeler says he or she is really interested in supporting enterprise-wide architecture, but his or her actions are to consistently do whatever is necessary to save time for the project, then you have more insight to his or her primary motivation, which is probably to save time and make sure that the project is on time.
- A third technique is to initiate an action to ‘test’ other people's motivations. Say a project data modeler says that the project team really wants to help with enterprise-wide data integration. Ask if they could schedule time to work with you on this and see what their response is. If they put off investing time in this, you have some clue as to their motivations.

In Company A, there was pressure to implement these common patterns and models across the enterprise. The enterprise team felt this pressure and was motivated to be successful in their venture toward enterprise integration. When the enterprise team explained the needs of the overall enterprise and referred to the mandates to reuse common patterns, specific project teams felt this pressure. Because there were mandates to do this, many of the project teams looked at the common patterns and politically went along with them. However, it seemed that there was a sense of resentment. People generally don't want to have another department's will forced upon them, and the natural human pattern is to resist. The more the resistance, the more the ongoing conflict, hence, the adage, “What we resist, persists.” So there was a motivation of not wanting to be controlled by another group. Though the patterns helped with consistency and integration, data modeling is a

creative task and requires the ability to make subjective judgments, so there can always be a justification for deviating from the common patterns. The enterprise architecture team did not really understand the motivations of the development teams. Instead the enterprise architecture team just said that the patterns were mandated from ‘above.’ This led to people only paying lip-service to using the patterns for political reasons, and then ignoring them when they could, so they could accomplish the ‘real work’ at hand. This was a major obstacle in socializing these patterns.

In Company B, the enterprise data management team recognized that projects were under the gun to produce results, and thus, project team members had motivation to be within budget, to be on time, and to meet requirements (as is often the case in many projects). Similar to Company A, Company B also had mandates from top-level management to reuse and standardize on the patterns and common constructs. However, the data management team did not lead with this when they went to project teams. Instead they focused on how they could service the individual projects and help them be successful. One of their core strategies was to implement the patterns and create a data services layer that offered new projects the ability to call common data routines. This actually saved time on many projects while still meeting the goal of data standardization and integration. Thus, the data management team provided services to each project and project team, focusing on helping them meet their objectives, because they really understood their motivations. In return the projects and teams, in general, cooperated with the data management team, because they felt that the data management team understood their needs and wants and the patterns addressed those needs and

wants. So the organization had great success in socializing the patterns.

A key observation in these two experiences is that Company A did not seem to understand the underlying motivations of the people, and they used a strategy of mandating the patterns. They also did not recognize that people did not want to feel controlled and forced to do something. Company B very clearly understood the motivations for the people to whom they were advocating use of the patterns. They knew that the project members using the patterns needed to be on time, within budget, and meet the project requirements so that they could be successful. By focusing on helping the project members with these needs, first and foremost, they had great success socializing the patterns throughout the enterprise. One of the cardinal rules of sales is to first understand the customer's needs. In many ways, when we are socializing the patterns, we are ‘selling’ the use of the patterns, and when we do this, it is important to first understand people's needs, which allows us to be of greater service.

Key Recommendations

Gain clarity on your own and other's motivations (as individuals and collectively) and focus on those when socializing the patterns rather than just focusing on policies to mandate use of the patterns. Also realize that not everyone's motivations are the same and quite often, the underlying core motivations are not apparent. You may need to tailor how you socialize the patterns in different ways to meet different core motivations.

Creating a Clear, Common, Compelling Purpose and Vision for Using the Patterns

“Seeing within, changes one's outer vision.”

—Joseph Chilton Pearce

What is the purpose for the patterns in your enterprise? What is your vision for how these patterns will be used? What is the corporate purpose and vision for the patterns? What are the core motivations of your enterprise and how do they affect its purpose and vision?

We believe that it is critically important to define, in writing, what the purpose and vision is for using these patterns and relate it to the purpose and vision of the enterprise. Understanding the various motivations for using patterns in your organization can provide key input into the purpose and vision.

Defining a purpose statement is a common and well-recognized principle advocated in many leadership and management texts and courses. A purpose statement is a short statement representing the core intention of something or someone. We suggest creating a purpose statement for these patterns in your organization. For example, it may be something like this: “The purpose for the patterns is to jump-start data modeling efforts, to facilitate consistency, and to offer class, reusable alternatives for those efforts.” Ideally the purpose for the patterns should support the purpose or mission statement of the enterprise. For example, the preceding purpose may be relevant for an enterprise that emphasizes consistency and quality. The purpose should also address the key motivations of your enterprise. For example, using the term *world-class* may come from a group motivation of ‘separateness,’ that is, of differentiation.

Another common and effective leadership and management technique is developing a vision. What is a vision? It is how a person and/or organization sees what will happen in the future. A vision usually represents a detailed description and portrays a picture of the desired future

state. Sometimes, enterprises have a short vision statement that describes the future state. When one reads the vision, it should be clear what the desired state will look like. In addition to a vision, it is helpful to have stated goals, benefits, and a plan for using the patterns.

As an example, a vision for using the patterns may look something like the following (although keep in mind that it is dependent on the culture and environment of the organization).

> *Any new project that needs to develop a data model will have a toolkit of these data model patterns available as a source for jump-starting their efforts and maintaining a consistent style of modeling throughout the enterprise. These patterns will help people save time in data model development by fostering reuse of solid data model constructs, and they will also save time and cost in maintenance because the subsequent database designs will be more similar and consistent. Data modeling efforts will also benefit from having alternative patterns available. The explanations of the patterns will help data modelers to more clearly see the pros and cons of various ways to model common types of data. The patterns will be provided as a living tool that will be updated, and various parts of our enterprise will continually provide feedback to let us know what is working and what suggestions or improvements are appropriate for our environment. The patterns will offer approved guidelines that are encouraged for use throughout the enterprise. There will be positive incentives for appropriately reusing the patterns, and disincentives when these common templates or patterns are not used in situations where it is appropriate to use them.*

> *However, there will be latitude regarding if and when to use the patterns based upon specific needs and circumstances. The patterns will serve as an effective aid to data model developers on projects and programs across the enterprise.*

Again, as we already stated, the type of industry that you are in, the motivations of your enterprise as a whole, as well as the cultural values may affect your vision and purpose. For example, if you are working in an insurance organization where one of the primary values is providing security and lowering risk, then the purpose of the patterns may be to "lower the risk of data modeling mistakes by reusing proven constructs." Another example could be a financial investment enterprise whose principle value is increasing shareholder value. In this scenario, a stated purpose for the patterns could be "the patterns help provide consistent, integrated data, enabling a strategic analytical advantage to increase profits."

We suggest that the vision and purpose have as qualities the following three Cs: that they are clear, common, and compelling.

- The vision and purpose need to state what you see happening **clearly** so that people understand it. For example, in the sample vision stated earlier, the intent is that people understand the vision and purpose of the patterns are about saving time, increasing quality, and serving as a tool that is encouraged but not too harshly pushed within the enterprise.
- The vision and purpose need to be **common**, meaning that general buy-in and common agreement exists among potential users. Getting input, involvement, and communication are key aspects to getting buy-in. One way to do this is to survey and ask people for input. Another way to do this is to communicate the patterns

and ask for feedback. What we have noticed is that when enterprises have sincerely asked for feedback and input and made changes to the patterns based upon requests, they were more easily able to socialize the patterns. For example, in Company B, the patterns and reusable universal data models were presented to the entire development team, and people were given opportunities to express their ideas, concerns, and suggestions. Thus, they were part of the process. This helped tremendously in gaining buy-in and making the purpose and vision 'common.' At Company A, the general approach was to mandate use of the patterns. Thus, in many situations, people and organizations were resistant. This is why it is critical to understand the motivations of your enterprise and the people working in your enterprise.

- **Compelling** is the third ingredient for an effective purpose and vision. You need to provide sufficient motivation for using the patterns. The motivations of people and organizations vary. That is why it is critical to understand the motivations, to work toward satisfying these motivations, and to communicate how the patterns can help accomplish the desired results. For example Company B, like many companies, provided great incentives to be on time and on budget and focused on project success, because this provides security and sustainability for the company. Understanding this, the enterprise data management team focused their efforts not only on offering the patterns but also on having prebuilt, reusable data services, for example, "get basic customer information" or "update customer address." Project teams perceived a compelling reason to use these services because it helped them achieve their goals. At Company A, there was also a desire to help project teams achieve their

goals; however, Company A did not seem to have the same vision regarding first understanding and then working toward the key things that were important to the projects and their teams.

Note

In the scenarios we described previously, both ‘Company A’ and ‘Company B’ initially had senior management support for the patterns concept. What happens in the case where you are selling the idea to senior management, trying to get their support? If this is the case, the purpose and vision statement takes on even greater significance. This stated purpose and vision statement encapsulates what you are trying to convince management to buy into.

It is important to reiterate how difficult it is to create a purpose and/or vision without understanding the motivations of the organization (or people) that will use the patterns. If the purpose and/or vision does not represent the motivations of the organization (and the majority of people within the organization), it is probably not an effective purpose and/or vision.

Key Recommendations

Have a clear, common, and compelling purpose and vision, in writing, for reusable patterns; one that supports the motivations of your enterprise.

Developing Trust so People Can Rely on the Patterns

Although it is extremely worthwhile to understand motivations and establish a clear, common, and compelling purpose and vision, the success of socializing the patterns depends greatly on how much trust there is. For example,

many times people have the following questions and make the following statements when they are presented with the patterns:

- Why should we use these patterns?
- We can develop better patterns ourselves!
- How do you know if these really work? Are they proven? Can we trust them?
- We are under the gun on this project, so please let us just do our own thing so we can be successful! We don't know if we can trust your patterns not to blow our deadlines!

Why is trust important to socializing the patterns? What is trust? Merriam-Webster Online defines it as “Assured reliance on the character, ability, strength, or truth of someone or something.”(7) If people do not have confidence or do not feel that they can rely upon these patterns, then they will not be adopted.

In the book *The Speed of Trust*(8) author Stephen M. R. Covey points out that there are two aspects to trust: character and competence. Character has to do with the intent and integrity of someone or something. Intent is about if someone is thinking just about themselves or if they are willing to help others outside of themselves. Integrity has to do with whether someone or something is in alignment (integration is derived from the word integrity). For example, does one walk the talk? Competency has to do with capabilities and results.

These two aspects of trust (character and competency) may apply to the patterns themselves and/or to the people advocating the patterns. Let's first discuss character, and specifically, intent and integrity. Regarding the patterns themselves, what is the intent of the patterns? This should ideally be stated within the purpose and/or the vision for the patterns, but even more important is the perceived intent based upon actions. If people perceive that the intent of the

patterns is to help people save time and to be of service, they are bound to be more trusted. If the intent (stated or perceived) of the patterns focuses on dictating a ‘mandate’ in order to control how people do things, this could negatively influence the use of the patterns. Is the intent of the people socializing the patterns to help us or is the intent to control us? Regarding integrity, do the people proposing the use of these patterns say what they do and do what they say? In other words, do they have integrity and can we count on them and, thus, can we count on the patterns that they are advocating? Are the stated benefits of the patterns in sync with the realized benefits of the patterns (i.e., can we count on the integrity of the patterns themselves)?

The other part of trust has to do with competency. According to Covey and as we mentioned, competency has to do with capabilities and results. Capabilities are about the skills and potential of someone or something and results are about the outcomes that have actually occurred over time. This also applies to the patterns and to the people involved in advocating use of the patterns. Regarding capabilities, do I trust that these patterns are a very high quality? Do I trust the capabilities of the people who are socializing the patterns? What is their skill level in data modeling, data management, project management, and other relevant skills? How do I know if these patterns will lead to stable, solid data models and database designs? Have the patterns successfully delivered positive results (within our company or within other enterprises)?

In Company A, it seemed to project members that the intention of the patterns was mostly to control how project members did things. The enterprise data management team did not feel that they had to prove anything to anyone about the patterns, and thus, they did not focus on the results and track history of the patterns. There was also some suspicion about the motives or intent of the enterprise data

management team. Whether this mistrust was misplaced or not is of no relevance. It existed, and therefore adversely affected the socialization of the patterns in Company A. One of the mistakes that the enterprise data management team made was not addressing this mistrust head on, and alleviating the concerns of the project teams.

In Company B, the intention of the patterns was to be of service and to help the projects save time and increase quality. This was not only stated but people could tell that these intentions were sincere by the way the enterprise data team acted. They took the patterns to the next level by creating data services based on the patterns, and they took the time to prove and show that the patterns worked well. The enterprise data management team went out of their way to focus on serving and helping the projects and project teams. Yes, they also had the goal of standardization for the benefit of the whole enterprise; however, their approach also focused on servicing their audience, in this case, the individual projects and project members.

On one effort where we were looking to standardize common models, one of the Universal Data Model consultants focused on this principle of trust, which yielded positive outcomes. His job was to standardize data model constructs across the enterprise, and when he approached each project, his first question was “How can I help?” His intention came across clearly and sincerely. One project team said, “We need better performance in our database.” Although this was not his primary objective, he did have skills in performance tuning so he used these skills to help out the project. After this, the project members were unusually helpful toward using common patterns and models and helping him with his team's objectives of standardizing data models and thus facilitating integrating data.

How does one develop trust? There seem to be three keys to developing trust:

- **Openness:** When people are open, vulnerable, and/or transparent and say what is really happening for them without hiding things, the barriers go down, and you move toward integration.
- **Caring about others:** When people's motivations and intent are sincerely about others, people can feel this, and there is more trust.
- **Earning it:** Trust is usually earned when you produce results.

How does this relate to these patterns? People need to trust in the patterns themselves and the people advocating the patterns, in order for the patterns to be socialized and openness has a great deal to do with this. So be open with the patterns. Share them across your organization and invite input and suggestions. Invite scrutiny when it occurs and make sure that people feel that they can test the patterns if needed. If someone points out a potential error or a mistake, be careful to stay open because there is a potential for being defensive, shutting down, and thus creating barriers to trust. It seemed that Company B invited a lot more involvement and input regarding the common models from the start, and there were large group meetings and individual communications allowing for input and feedback on the patterns and reusable models. Thus, people felt like they were part of it and bought into them.

Regarding the second point, “caring about others,” use the patterns as a vehicle for service to others, keeping that foremost in mind. It is common that there are temptations to do whatever is needed to be successful, claim credit, and focus on reaching one's own objectives; however, if you keep your own motivations appropriately balanced with the idea of helping others and show this, trust develops. In Company A, it seemed like people knew at some level that,

even though the patterns had a stated objective of being of service to data modeling efforts, mandating use of the patterns had more to do about the goals of the people in charge of providing the patterns. The application teams felt this, and therefore, there was resistance. In Company B, the mindset of service was so prevalent and came through so strongly, that people trusted the people that were providing the common data patterns and models. They also trusted that the patterns were designed to be of service, which was true.

Perhaps the most important factor in developing trust is earning it and proving that these patterns can help through actual results. We have had many enterprises use these patterns successfully. However, if you were not involved with one of these enterprises, you have probably not experienced this, and neither have the people and projects that are potential users of the patterns in your enterprise. Our observation is that by taking incremental, small steps and delivering results in short periods of time, it is easier to show value and then gain trust and momentum.

> **Note**
>
> In our experience the most important step is the first step. If you fail to deliver on the first project, most goodwill and trust goes out the window. So be very careful in picking the first incremental step. It should be useful enough that it gets noticed and easy enough that it can be done quickly and without too much difficulty. First impressions are the strongest!

Another way to gain confidence in the patterns is to thoroughly review them, scrutinize them, change them if need be, and make sure that people feel that they are stable and solid foundations. We have been through many iterations of these patterns and have implemented them many times, so we have a lot of confidence in the patterns. If people can spend time going through the patterns in

detail, this can help with trust. Also, as you successfully use the patterns in your enterprise, we recommend that you document how the patterns helped, including how much effort, time, and money was saved by using the patterns. Collect anecdotal evidence that will help you gain trust for your next project. We find word of mouth and positive reviews help build up trust.

In Company A, there was an expectation that people in the organization should just use the patterns and reusable models because the enterprise data management team *said* that they needed to use them and because this was the policy. The potential users of the patterns did not scrutinize the patterns, ask questions about them, or suggest changes to them because they felt that they did not have the latitude to do this.

In Company B, the enterprise data management team honored anyone who had the perspective of "prove/show it and I will believe it." The enterprise data management team physically instantiated a couple of the patterns and implemented them as common data services. Then they were able to get some cooperation and collaboration from one project where they could use them, and it saved time, increased quality, and provided a consistent interface to the data; it also helped to standardize and integrate data. People and other project teams saw this and were less resistant. Afterwards, many other efforts used the patterns, models, and common data services.

Key Recommendations

Develop trust in character and in competency regarding the patterns and the people who are socializing the patterns. Be open to suggestions, have a mindset of service first and foremost, and earn the trust of others by showing results. Choose your first step carefully!

Managing Resistance and/or Conflict Regarding Patterns

Our experiences have shown us that you are bound to have conflicts in socializing any idea across an organization. Conflict may be a good thing, when managed properly, because people often have differences of opinion. In fact, some of the most valuable insights regarding the patterns in this book have been developed because of a difference of opinion between the authors. However, when people are not able to effectively work through conflict, it often creates difficulties regarding getting people on the same page. This section provides ideas and suggestions for helping when conflicts arise, which will probably happen at some point in time.

A classic conflict that we see all the time is the conflict between those advocating the use of patterns across an enterprise (such as an enterprise architecture team) and those who are involved in a specific project or task (such as a project team). The enterprise architecture team is often focused on trying to gain consistency so that data can be more easily integrated. The project team is generally focused on how they can get their specific project completed on time and within budget and meet their specific business requirements. The enterprise-wide team may be evaluated and given incentives for successful enterprise-wide integration. The project team may be motivated and incentivized with bonuses, career advancement, a sense of accomplishment, and reputation for delivering their solution on time, within budget, and meeting their specific requirements.

Can you see the inevitable conflict? Some people may be operating from a perspective that looking at the whole is most important. Others that are working on a project may say that they have a specific job to do and they are going to

focus on doing their job. Who is right, or who is *more* right? Is it possible that both of these perspectives have equal validity? It is akin to the old elephant parable, where four blind people bump into various parts of the elephant and each claim that an elephant is something different. Only in this case each person is bumping into his or her own perspective, and thus, if you can see the whole elephant, or all the perspectives involved, you are in a better position to solve the problem. However, when one is in a specific position or role, or on one side of the argument, it often feels like that is the *right* perspective. We have personally been in situations where we have felt that our roles as enterprise data architects were more important than the role of designing a specific data structure because of the impact our role had to the entire enterprise, yet even though at the time we felt this way, it is usually not true that one role or one perspective is more important than another.

Appreciating the other perspective is very difficult to do if one feels attacked. For example, we have been in situations where someone wants to push forward their own agenda and has tried to discount the benefits of the patterns or invalidate them. As a big proponent of using the patterns, our first inclination may be to be defensive. When this happens, as hard as it seems, it is helpful to not take it personally! Thus, if someone knocks the patterns or the people socializing the patterns, sometimes it may be appropriate just to acknowledge and understand the complaints, appreciate the feedback, see what you can do to address this, and then move on. Michael Moore, the movie producer and director, exemplifies this in his movie *Sicko*.(9) At the end of the movie after his arguments about the benefits of socializing health care in the United States, he shares what happened regarding a web site that vehemently attacks him and his ideas. The producer of the

web site ironically was having financial difficulties because of his wife's medical bills not being covered under the insurance policy (which was a key topic for the movie). The producer announced that they would have to close the anti-Michael Moore site. Moore not only acknowledged this site, but he sent a check for $12,000 saying that he thought that the producer of the web site should not have to lose his first amendment rights to free speech, including the right to bash Moore and his ideas. Acknowledging and appreciating perspectives, even when they are attacking and aggressive, can be wise.

Dr. Graeme Simsion(10) once asked one of us in an interview, “Do you see your model for a given subject area constituting the one right answer or only one possible answer?” The response was “Yes, I have the only one right answer and that makes everyone else's model wrong. I'm of course being facetious, but we've all seen this type of attitude in data modeling efforts, and it poses a disservice to our community. I believe the data management community is largely about integration which involves working together, not proving each other ‘right’ or ‘wrong.’”

It is interesting that Dr. Simsion would even ask this question. Later on he published a book called *Data Modeling Theory and Practice*(10) where he interviewed a number of data modeling practitioners and thought leaders, many of whom had a perspective that there was one correct answer for a particular data modeling situation.

When we have done workshops on the patterns, we sometimes get feedback like “Just tell me the best practice solutions instead of continuing to say that there are many good answers and that it depends on the situation. You are the expert(s).” We appreciate that there's another perspective even regarding this principle of whether or not there is a right answer. Nevertheless, it seems that one of the most powerful conflict management principles is to

appreciate different perspectives including the ‘opposite side of the coin.’

Some of our clients have said one of the greatest benefits that the patterns offer is that we offer a third-party, industry perspective regarding possible options for data modeling to help them resolve data modeling conflicts. They have said that when data modelers fight over the ‘right way’ to model things (in our experience this is common), that one of the greatest benefits to the Universal Data Models and Universal Patterns is that they can use them as a third-party, industry approach to help resolve conflicts. Sometimes a data modeling conflict can become polarized, and when the participants are able to look at a different view, it sometimes helps to keep the debate more objective. We want to emphasize that, in our opinion, these Universal Data Models and Patterns, while offering effective alternatives, are not the ‘*right*’ way to model things because we believe there is no definitive ‘*right*’ way to model something. However, they offer solutions that have proven to work well, and they provide possibilities and ideas that you may not have considered.

Regarding socializing patterns, if you do find yourself in a conflict, this core principle of acknowledging and even appreciating the other perspective can be powerful. If you find yourself in the role of trying to get enterprise-wide acceptance of the patterns and then encounter resistance on a specific project, stop and consider the plight of the other side. For example, the project *is* under pressure to deliver. So a key question comes back, “How can you help them meet their goals?” This requires moving from your position to the other side of the argument.

Common objections or perspectives that you may face regarding using the patterns include the following:

“Our data modeling needs are unique. We don't want to use template patterns because it is important for us to

model our own unique requirements as well as develop models."

"These patterns are theoretical and abstract. We only work with the practicalities of developing a good data model for our project."

"We need to develop the data model based on our specific application database. If we use a whole new way to model data, then our data conversions will be quite expensive when we move data from the old system to the new system."

"We do things differently than the patterns. We have different conventions, a different style, and different approach to modeling."

"We simply do not have time to learn and apply these patterns. We have to just focus on our own project needs and there is tremendous pressure to be on-time, on-budget, and meet our requirements."

How would you answer these questions? Before reading on, we invite you to either write down or verbalize what your response would be to each of these questions.

William Ury bases his book *Getting Past No: Negotiating with Difficult People*(12) on five steps to negotiating toward win-win outcomes. They are as follows:

"Step 1. Don't react—Stay objective."

This step suggests that before reacting to resistance or to a potential conflict, just stop and don't take anything personally; just look at it objectively.

"Step 2. Disarm—Step to their side."

This is a key step where you understand the other perspective. You already know your own perspective so if you can really understand the other perspective and place yourselves in the other person's shoes, you are much better positioned for a win-win outcome. For example, is it possible that the project member has a legitimate need to be on time, on budget, and meet his or her project

requirements and that enterprise-wide architectural considerations could be an obstacle to his or her success?

“Step 3. Change the game. Don't reject. Reframe.”

Once you understand both sides, you are exploring and expanding, looking for holistic and common goals. Ury suggests asking questions to reframe that focus on solving the problem toward a solution instead of digging into your position. For example, if someone on the project says do not do anything that will impede my project time frame and deliverables, then instead of debating, reframe the conversation with a question that moves toward a solution. In this situation, you can ask, “How can we help?” or “What if we could help you get some of the data you need for the project?”

“Step 4. Make it easy to say yes. Build a golden bridge.”

In this step, you take a look at a solution or solutions that are truly win-win. Ury talks about getting people involved, continually referring to interests (versus positions), helping people save face, and going slow. After common interests are identified (for example, saving time), you could ask, “If we could help save time on the data model for the project, could you then support using these common patterns?”

“Step 5. Bring them to their senses, not their knees.”

This step suggests using power, not force. The emphasis is on moving toward solutions that address the various motivations and not on one party coercing the other or trying to control the outcome using leverage. Ury points out what could happen if there is no negotiation and if you forget that, while you can warn, you should not threaten. For example, if a project member still refuses to even consider using patterns, instead of saying “You must follow the policy of our organization that says to reuse patterns where applicable or else I will escalate this matter,” you

can point out some of the consequences of not using patterns such as “We want to serve data modeling efforts by pointing out potential pitfalls, and if data modeling efforts do not even look at these patterns, there could be potential mistakes that could have been avoided.”

So, how can you use this knowledge to help you when you receive resistance or when conflict arises regarding use of the patterns?

If any of the previously mentioned objections are raised, using Ury's model, the first step would be “don't react.” In many cases, this is such a huge step because we, as humans, often have many filters and automatic reactions to things. On some efforts, we have noticed that when someone gives the objection that the patterns are abstract and theoretical, our first reaction was to defend the patterns and say, “No, they are not! They are used by many, many enterprises throughout the world and are proven!” The next thing that generally happens is that the argument becomes polarized, and it continues over a long period of time with both sides arguing for their position.

The next step is to “Disarm—Step to their side.” This also is a very difficult step to master in many situations. We have observed so many scenarios where there is resistance to the idea of using patterns or reusable models and seen first-hand the strong tendency to debate. There is great wisdom in truly stepping to the other side and really listening to what the other person is saying even if they seem to be completely against what you are trying to do. On one effort, when we tried to standardize and socialize common patterns and models, the enterprise team shared all the common patterns and models, and when people said that they did things differently and used different conventions, the first thing the enterprise data team did was to ask, with respect and sincerity, “How do you do things differently?” Simply by our asking this type of question, with an intent to

serve as opposed to engage in debate, the energy changed, and there was more of a chance to develop a win-win solution. This step is very much in line with understanding other people's motives. For example, if you skip this step people may feel like you are taking their job or perhaps that you don't understand their needs. Sometimes, if you demonstrate this quality of not only stepping to the other side to help but also actively helping, the other party follows suit! This also helps gain trust, which is critical as we discussed in the last section.

Next, a good idea is to “Change the game. Don't reject. Reframe.” After you understand both sides, you can then expand the possibilities and see what solutions can help meet the needs of both sides. Ury in his book talks about the concept of *position* versus *interest*. A *position* is a narrowly defined scenario regarding what should happen, whereas an *interest* is a broad reason for wanting something. When each party takes a position, there is usually no room for mutual beneficial solutions because the parties each have a narrowly defined idea of what should happen. Conversely, when parties are thinking about their interests, there is much more room for many solutions, and interests generally overlap. It is like trying to find similarity between two small points that are far apart versus expanding them into very big circles that overlap like a Venn diagram.

For example, take the classic enterprise data perspective versus a project data perspective dilemma—if you approach it from a position view, the enterprise team may come from a perspective that everyone must strictly adhere to the patterns and standardize. The project team may fight this and explain that they just don't have time to standardize and must get the project done. Thus, each can argue their sides and polarize the discussions. However, if you approach it from a viewpoint of interests, the enterprise data team is

interested in helping the company be more efficient and so is the project team. When we have done exercises in seminars where we ask people to role play using this technique, sometimes the people role playing realize that they actually all work for the same company! Wouldn't it be great if the people in the organizations realized sooner that they are actually working for the same organization when these types of conflicts happen?

Then, you can move to the next step of “Make it easy to say yes. Build a golden bridge.” If you are in the classic conflict where an enterprise data team wants to standardize and the project team just wants to focus on their effort, you can offer solutions that satisfy the underlying interests of both sides. For example, in Company B, they recognized that project teams were under pressure, and instead of forcing the projects to adopt these patterns, they made it easy for the teams by offering data services that standardized on the patterns. Thus, the application teams were being offered prebuilt programs to access data, and this reduced their workload. This helped the individual project teams meet their deadlines and commitments, while helping to standardize data across the enterprise.

Finally, Ury suggests, “Bring them to their senses, not to their knees.” At one enterprise data management group in New York City, when this idea was presented, many of the practitioners felt that this was idealistic and that an architecture group really needs to ‘lay down the law’ and just demand that people follow standards consistently. We appreciate this perspective, and there is some merit in this approach in some environments and in some situations. However, our experiences and research indicates that forced solutions rarely stick, and even if they do, they create other issues. How do you feel about being forced to do something? Though there is an advantage of having policies and rules stating what should happen, it seems that it is

more effective to lead with powerful win-win solutions rather than try to use leverage, control, or force to bring about outcomes.

Although most situations can ideally lead to a win-win solution, there are some situations where this may not be possible. Ury suggests being prepared with a "Best Alternative To a Negotiated Agreement" (BATNA) in these circumstances. What if a person in the enterprise claims that the patterns that the enterprise is using are poor tools or, even worse, claims that the people who are trying to socialize the patterns are incompetent and then continues to condemn the patterns and the people promoting them. On one of our consultant engagements, there was a person who expressed that the enterprise-wide choices we were making regarding data modeling patterns were the *wrong* choice and that this effort would be unsuccessful. This person was socializing this in various parts of the enterprise, even though the person was on the enterprise-wide data team. What do you do? We heard out this person, actively listened, and tried to incorporate suggestions from this person. However, this still did not work, and the person continued to evangelize the *wrongness* of what we were doing. This effort turned out to be very unsuccessful, and this situation was a factor. Perhaps in situations like this, it may be appropriate to look at other alternatives, such as removing the person from the team (or from the enterprise).

In our case examples, it seemed that Company A did not employ these principles of conflict management as skillfully as Company B, and hence, the result was not as effective. Company B used many of these techniques in a humorous way. For example, using the idea of "bring them to their senses, not to their knees," Company B often led with an intent to be of service and to help each project, and in the rare circumstances where project teams simply did not want to use the enterprise-wide data constructs, they jokingly

pulled out a bat to illustrate what they would do if these project teams didn't follow these enterprise-wide standards. This alleviated any stress in the situation, and then after a chuckle, they reinforced the message that this was absolutely not what they were about, and they went back to understanding the other side, creating a win-win situation, and using power by getting back to the motivations. By the way, we don't recommend using this technique because it could be easily misinterpreted. However, what was most important was the underlying intent and the actions that took place. Thus, the project team members knew that the enterprise data team was truly interested in win-win solutions and was not about forcing outcomes.

Are there situations where one has to use force and lay down the law? Perhaps; however, in our opinion, as we stated earlier, this is the last resort. You might recall that earlier in the chapter we mentioned one enterprise-wide effort where the enterprise mandated use of the patterns, and the project team obliged at the data modeling level but implemented completely different physical database designs. The core reason was that using the common patterns and models would have caused a lot more data conversion work and, from their perspective, would have created a project risk. Thus, they actually lost time on the project because they did extra work to use the patterns in the analysis phase just to accommodate the enterprise data team, even though they didn't use them in the physical implementation. Perhaps if another tack had been taken where both parties looked at the data conversion issue and investigated where and how the patterns could be used in conjunction with data conversion needs, then some of the patterns could have been implemented. In general and in most situations we recommend striking an appropriate balance between strict adherence to these patterns and an

‘anarchy’ perspective, where there is no accountability at all for using common patterns.

Note

We have worked in many different organizations around the world, and you should also be aware of the cultural differences that may exist when it comes to managing conflict. William Ury's five steps can be applied anywhere, but be aware that the ‘tone’ of how you use the five steps may need to be modified depending on where and with whom you are working. In other words, dealing with conflict in Dallas, Texas, versus dealing with conflict in Bangalore, India, may require a modification in approach.

Note

You may want to explore other techniques and methods that can be used to manage conflicts, for example, the book *Crucial Conversations: Tools for Talking When Stakes are High*(13) is an excellent guide and Edward DeBono's *Six Hats* tool (14) may be useful as well. It may be worthwhile for you to ask around in your organization if you have a standard methodology for conflict resolution. However, the ideas illustrated in this section may be useful in conjunction with whatever methodology you use to manage conflict.

Key Recommendations

Understand and appreciate other perspectives as well as the whole picture and find solutions that are truly win-win.

Other Comments about Socializing the Patterns

This final section of the chapter covers some additional information about socializing the patterns relating to three particular areas:

- The need for the patterns to be socialized in many types of circumstances
- The role of upper-level management commitment
- The estimation of the return on investment for using the patterns

Patterns May Need to Be Socialized in Many Types of Circumstances

In general throughout this chapter we have mentioned the enterprise architecture team as being the initiator of the use of the patterns in order to illustrate a common scenario. Though this is one scenario, please keep in mind the patterns can be used in a wide variety of circumstances by people in a wide variety of roles such as project team members, data modelers, data analysts, people developing messaging solutions such as XML, and others. Thus, you may not be coming from an enterprise-wide perspective, and the patterns are still extremely useful. In fact, we often find that these patterns are needed by someone trying to find a way to shorten development time or to increase the data model quality for a particular project.

We have chosen in this chapter to concentrate on enterprise-wide use of the patterns because this is one of the most difficult aspects of socializing the patterns. However, the same principles will apply to socializing the patterns for a very specific project or for other scenarios where it is important to gain buy-in for these patterns. Thus, we want to stress that these patterns are for use by anyone and everyone involved in finding effective data model options and solutions.

What about Upper-Level Management Commitment?

When we do seminars on the topic on the human dynamics involved in data management and socializing the patterns, participants will sometimes ask, “What about the role of upper management in adopting these patterns? Upper management sets the culture, provides the incentives, sets the vision, develops atmospheres of trust (or distrust), and decides how to manage conflict. These ideas are good, but you need to do this seminar for our upper-level management!”

The interesting thing is that we also have had various upper-level management executives at our seminars and they often say “Yes, these ideas are good, but the people working for us need to know and follow these ideas!”

With that said, yes, it is greatly advantageous to have high-level management commitment. Yes, it is true that simply having a company big shot in the room nodding his/her head when the universal patterns idea is introduced may have a positive impact. If upper management sets the vision and direction to use patterns and model more consistently, this can greatly help with momentum. For example, there are great benefits if upper management reinforces the message that using patterns should be part of the systems development life cycle. It is also advantageous if upper-level management supports the idea that project data models will be reviewed and compared with patterns and sets incentives for projects that appropriately use patterns to create enterprise-wide consistency.

However, we have also seen failures even though there was huge management commitment! On one effort, there was huge management commitment from the top of the enterprise and at many levels. They had allocated a great deal of time, money, expertise, and resources, to enterprise-wide data integration and using standardized models. However, they lost this commitment when motivations were not understood, when the vision was not clear, when there

was an environment of mistrust, and/or when there were no effective ways of managing conflict. In our experience, the principles we've discussed seem to be more significant than management commitment in achieving success, and when they are followed, they usually lead to management commitment. They are applicable for all levels of people in an enterprise, and in our experience, the best results are when people at all levels within the enterprise follow these principles.

What Is the Return on Investment Regarding Using These Patterns?

A common question that is asked is “What is the return on investment (ROI) for using these patterns?” In his article “What is the Enterprise Data Model ROI?”(15) Steve Hoberman provides some clues to this question by suggesting two ways of addressing the question of the ROI for an enterprise data model:

1. Find some way to calculate ROI.
2. Refine the question (that they are asking regarding ROI).

He then provides an Enterprise Data Model (EDM) ROI calculation that was provided by Jan Kamil (senior data modeling consultant) and Yvonne Balditt (data analyst):

```
EDM ROI = (EDM contribution to profit) - (cost of the EDM)
```

If we extend this approach to estimate the ROI of using patterns in data modeling, the first question is this: “How do the patterns contribute to profit?” A few major areas where these patterns can contribute to the profit are:

- Savings in time and effort in data modeling.
- Higher-quality data models resulting in much lower systems maintenance cost.
- A reduction of mistakes by quality assuring models against field tested patterns for data models, leading to fewer redevelopment costs.

- Accommodating future needs when using the more generalized patterns. This can be a huge factor considering that changes to data structures represent a very large cost in systems development.
- Consistent, powerful, integrated data structures resulting in higher-quality data for better decision making and operations.

How much time and cost is saved by using these patterns? Many of our clients report months or years of savings in time and almost all our clients have saved a great deal of time and effort. The answer frequently depends on many factors such as how often the patterns are used, the current skill levels, and the enterprise's cultural environment. One way to estimate the return is by trying the use of the patterns on a sample data modeling problem with one group that has been trained in the use of the patterns and another group that does not have knowledge of the patterns, and then comparing the amount of time it takes for each and the quality of the models produced. Then, you can estimate the amount of data modeling that needs to be done for a certain time period and multiply the estimated percentage of time saved to calculate the savings for a certain period of time. However, the amount of time saved usually increases as an organization uses the patterns. Also, there are integration benefits as more and more efforts use the same type of patterns.

Perhaps the largest benefit of the patterns is in reducing data modeling errors and developing stable, flexible, and quality data structures. What is the benefit of developing a quality data model that accommodates current and future needs? On one consulting engagement at a client, just one field that was mistakenly modeled and implemented cost the organization several million dollars in redevelopment costs! On several engagements, we have seen tremendous costs in redevelopment and maintenance because data

requirements changed. Mistakes in a data model or mistakes in not accommodating future needs are often like mistakes in the foundation of a building and are hard to correct. Thus, to calculate the ROI for using the patterns to improve the quality of a data model, one approach would be to use a percentage of the total systems development and maintenance costs because the prevention of errors will substantially lower these costs.

What are the costs of using these patterns? The costs involve investing time in reading and understanding the patterns, spending time and effort in socializing the patterns, and possibly investing in training and/or additional tools regarding the patterns. We believe that in the vast majority of cases, when we have used these patterns, the benefits of using the patterns far outweigh costs associated with investing in these patterns.

The second point that Steve Hoberman makes is to “Refine the question.” He suggests that perhaps when people ask about ROI what they are really expressing is that although you may be pointing out all the great benefits, where is the proof? Thus, this question may relate to the principle regarding developing trust and understanding motivation. We think that it is important to evaluate the ROI on your data modeling efforts when you use these patterns. We highly recommend that an ROI task be added to the ‘close-out’ phase for these efforts. Estimating the ROI has two very significant benefits. First, having ROI numbers will help describe the benefit of using patterns to your organization in concrete terms. Senior management often asks “why should we invest or time and resources in this?” An ROI can help you answer this question in financial terms. Second, as you gather ROI numbers for projects where you have used patterns, the ROI numbers should give an indication of where your costs (and savings) are highest. This can give

you an indication of where the greatest benefits lie and where to focus your efforts.

We have had many clients use these patterns effectively and receive great benefits from them. We suggest that you use the patterns on efforts in your enterprise to create your own proof, and we sincerely hope that this book will help produce a huge return on investment for you and that these patterns will be extremely beneficial to you and your organization.

Summary

This chapter suggested four key principles that we believe can help socialize these patterns and gain buy-in for effective use of the patterns in this book as well as for the reusable data models found in *The Data Model Resource Book, Revised Edition, Volumes 1 and 2* (Wiley, 2001). These principles are as follows:

- Understand motivations. Why would someone use or not use the patterns?
- Create a clear, common, compelling purpose and vision for using the patterns.
- Develop trust so people can rely on the patterns and the people socializing the patterns.
- Effectively manage resistance and/or conflict regarding patterns.

Our observations are that organizations that are successful in socializing the patterns and reusable data models do so by:

- Developing clarity regarding yours, others, and the organization's motivations and working toward satisfying these motivations.
- Having a very clear, common, and compelling purpose and vision for the patterns and putting this in writing.
- Developing trust regarding the character and competency of both the people involved in socializing

the patterns as well as the patterns themselves.

- Understanding and appreciating the other perspectives and the whole picture when resistance or conflict arises and working to find solutions that are truly win-win.

Finally, we covered other topics regarding socializing these patterns. We discussed how the patterns may be used not only for enterprise-wide standardization but also for many other circumstances such as jump-starting the development of a data model for a specific project. We recognized that upper-level management support helps, but in our opinion, the universal principles we discussed in this chapter are core to not only gaining management support but also to achieving success. We provided some suggestions regarding estimating return on investment of the patterns, and we hope these patterns yield a great return on investment for you and your organization.

References

[1] See Chapter 9 for more information on how to use the patterns for these types of efforts.

[2]The Data Model Resource Book: *Revised Edition, Volume 1, A Library of Universal Data Models for All Enterprises* by L. Silverston (Wiley, 2001).

[3]Power Versus Force: The Hidden Determinants of Human Behavior by David R. Hawkins (Hay House, 2002).

[4]The Sedona Method by Hale Dwoskin (Sedona Press, 2003).

[5] From the Routledge Dictionary of Quotations, edited by Robert Andrews (Routledge, 1987).

[6]A Framework for Information Systems Architecture by John A. Zachman, IBM Systems Journal, Vol. 26, No. 3 (1987). Also see http://www.zifa.com/ for diagrams and explanations of the Zachman Framework.

[7] Merriam-Webster Online at www.merriam-webster.com.

[8]The Speed of Trust: The One Thing That Changes Everything by Stephen M. R. Covey (Free Press, 2008).

[9] *Sicko* was produced by, directed by, and starred Michael Moore (Dog Eat Dog Films, 2007).

[10] "Data Discussions," an interview of Len Silverston by Graeme Simsion Dr. on Universal Data Models, 2003; hosted by Wilshire Conferences. Available at http://www.wilshireconferences.com/interviews/silverston.htm.

[11]Data Modeling Theory and Practice by Graeme Simsion Dr. (Technics Publications, LLC, 2007).

[12]Getting Past No: Negotiating with Difficult People by William Ury (Bantam, 1991).

[13]Crucial Conversations: Tools for Talking When Stakes are High by Kerry Patterson, Joseph Grenny, Ron McMillan, Al Switzler, Stephen R. Covey (McGraw-Hill, 2002).

[14] For more on Edward de Bono and his "Six Thinking Hats" technique and methodology please refer to http://www.edwdebono.com/

[15]What Is the Enterprise Data Model ROI? by Steve Hoberman, DM Review Magazine, May, 2006. From Steve Hoberman's "Design Challenge" article series. Jan Kamil and Yvonne Balditt were the people that were mentioned in the article that responded to the "Design Challenge" and provided the EDM ROI calculation suggestion.

Index

A

A Pattern Language: Towns, Buildings, Construction
abstraction
addition of categories
addition of hierarchies
addresses
  capturing
  formats
  parts
  synchronization
adherence to patterns
aggregations
  categories
  creating
  ENTITY 1 and
  ENTITY 2 and
  ENTITY 3 and
  hierarchy comparison
  Level 1 Recursive Pattern
Alexander, Christopher
application data models
architecture data model
arcs, exclusive arcs
area attribute
associative entities
  many-to-many relationships
attributes
  capturing, Level 1 Recursive Pattern
  class words, capturing

consolidation
enterprise data models
ENTITY
event attributes
Level 1 Contextual Role Pattern, Attributes
name attributes, declarative roles
organizations
outcomes
PARTY entity
PARTY ROLE supertype
people
PERSON entity
range of dates
as repeating groups
role-specific, capturing
roles as
statuses and
statuses as
Authorizations pattern
authorizing transactions

B

Barker, Richard
Barker's Notation
exceptions
Bill to address
bill-to address part attribute
BILL TO CUSTOMER role
billing address, vs shipping address
blog address
BOM (bills of materials)
business country telephone code attribute
business data model

business email address attribute
BUSINESS RULE
BUSINESS RULE CATEGORY
BUSINESS RULE CATEGORY CLASSIFICATION
BUSINESS RULE CATEGORY TYPE
BUSINESS RULE FACTOR
BUSINESS RULE OUTCOME
business rules
  capturing
  contextual roles
  independence
  Level 3 Business Rules Pattern
  Level 1 Recursive Pattern
  metadata
  outcomes, maintaining
  party roles
  recording data
  relationships as
  rule entity
  statement
Business Rules with Party Roles

C

canton
capturing
  attributes, Level 1 Recursive Pattern
  business rules
  class words in attributes
  contact mechanism
  electronic address information
categories
  addition
  aggregation

attributes
relationships
rolling up/down
Status Category Pattern
statuses
types
structure
categorization, definition
Chen, Dr. Peter
CITY
foreign keys
class words, capturing in attributes
classifications
adding
deleting
ENTITY CATEGORY TYPE AND
flexibility
independently maintaining data
indicators
Level 1 Classification Pattern
Level 2 Classification Pattern
Level 3 Classification Pattern
Level 3 Classification Pattern with Rollups and Schemes
many-to-many recursive relationship
modeling as entities
multiple
mutually exclusive
PRODUCT entity
redundancy
status, reporting on
types
relationships
static

updating
common information for people and organizations
COMMUNICATION EVENT RULE
COMMUNICATION EVENT TYPE
components
conflict regarding patterns
contact information
CONTACT MECHANISM
CONTACT MECHANISM BOUNDARY
FACILITY and
PARTY and
relationships
CONTACT MECHANISM APPLICATION
party id attribute
CONTACT MECHANISM CATEGORY
CONTACT MECHANISM CATEGORY CLASSIFICATION
Contact Mechanism Pattern with Geographic Boundary, as add on
CONTACT MECHANISM PURPOSE, ENTITY CONTACT MECHANISM and
CONTACT MECHANISM PURPOSE TYPE
CONTACT MECHANISM PURPOSE(s)
CONTACT MECHANISM TYPE
CONTACT MECHANISM USAGE
CONTACT MECHANISM USAGE TYPE
ENTITY CONTACT MECHANISM and
Contact Mechanism with Flexible Address Parts Pattern
Contact Mechanism with Geographic Boundary
contact mechanisms
business phone
capturing
Contact Mechanism with Flexible Address Parts Pattern
Contact Mechanism with Geographic Boundary

data needs
electronic addresses
entities
FACILITY and
geographic boundaries
identification
Level 2 Contact Mechanism Pattern
Level 3 Contact Mechanism Pattern
Level 4 Contact Mechanism Pattern
location
office phone
ORDER
overview
PARTY
placeholders
primary phone
relationships
types
adding new
as attributes
as virtual method for getting in touch
context, definition
CONTEXTUAL ROLE
ENTITY
PARTY
ROLE TYPE
Contextual Role Pattern
contextual roles
business rules
capturing as relationship between role and entity
compared to declarative roles
customers
definition

ENTITY
Hybrid Contextual Role Pattern
Level 1 Contextual Role Pattern, Attributes
Level 1 Contextual Role Pattern, Relationships
declarative roles and
Level 2 Contextual Role Pattern
Level 2 Contextual Role Pattern, PARTY Only Alternative
Level 3 Contextual Role Pattern
multiple
overview
parties and
PARTY
PROJECT entity
relating to entities
role definitions
using role
CONTRACTOR
COUNTRY
foreign keys
country telephone code attribute
credit limit attribute
CREDIT RISK RULE
crowsfoot
currency
CURRENCY TYPE
CUSTOMER
CUSTOMER declarative role
Sands Distribution
ORDER and
CUSTOMER role
PARTY
customers
contextual roles

setup
ship-to

D

data as asset
Data Model Patterns
Data Model Resource Book Volumes 1 and 2
data modeling, discipline of, extending
data modeling notation
data models
reusable
data types
data warehouse/data mart data models, star schemas
Sands Distribution
data warehouse data models
relational approach
data warehouse models, Sands Distribution
relational approach
database physical design
databases
DECLARATIVE ROLE
PARTY AND
DECLARATIVE ROLE 1
DECLARATIVE ROLE 2
declarative role entities
ORGANIZATION
PARTY
PERSON
declarative roles
compared to contextual roles
context and
defining flexibly
definition

Level 1 declarative role pattern
Level 2 declarative role pattern
Level 3 declarative role pattern
name attributes
organization specific
PARTY ROLE supertype
patterns and
person specific
redundant data
relationships
setting up role
DEPARTMENT
Design Patterns: Elements of Reusable Object-Oriented Software
devices, definition
differences in socializing patterns
directions
disk capacity attribute
DIVISION
DIVISION entity

E

effective from date attribute
effective thru date attribute
electronic address
email address
geographic boundaries
electronic address information, capturing
ELECTRONIC ADDRESS subtype
Ellis, Harry
email address
electronic address
EMPLOYEE

EMPLOYEE role
EMPLOYMENT APPLICATION
enterprise data models
  Sands Distribution
entities
  associative
      many-to-many relationships
  classifying
  consolidation
  enterprise data models
  self association
  status, number of
  status types
  in statuses
  subentities
  subtypes
ENTITY
  attributes
  classifications
  contextual roles
  instances, roles
  parents
  status type id attribute
  SUBTYPE 1
  SUBTYPE 2
  SUBTYPE 3
ENTITY 1
  aggregation and
  foreign key
  hierarchy and
  parent entity id foreign key
ENTITY 2
  aggregation and

hierarchy and
ENTITY 3
aggregation and
hierarchy and
ENTITY 1 CONTACT MECHANISM
ENTITY ASSOCIATION
ENTITY ASSOCIATION 1
ENTITY ASSOCIATION 2
ENTITY ASSOCIATION RULE
ENTITY ASSOCIATION TYPE
ENTITY CATEGORY
recursion
ENTITY CATEGORY CLASSIFICATION
ENTITY CATEGORY ROLLUP
ENTITY CATEGORY ROLLUP RULE
ENTITY CATEGORY TYPE
classification and
hierarchy
schemes
ENTITY CATEGORY TYPE ROLLUP
ENTITY CATEGORY TYPE ROLLUP RULE
ENTITY CATEGORY TYPE SCHEME
ENTITY CONTACT MECHANISM
Entity-Relationship Modeling
ENTITY STATUS
status datetime attribute
status from date attribute
status thru date attribute
ENTITY STATUS TYPE
ENTITY TYPE, recursion
ERD (entity relationship diagram)
estimated hours attribute
ETL databases

event attributes
EVENT RULE OUTCOME
EVENT TYPE RULE
EVENT TYPE RULE FACTOR
EVENT TYPE RULE TYPE
Events pattern
example data in illustration tables
exclusive arcs

F

FACILITY
  as physical structure
FACILITY, CONTACT MECHANISM AND
facility attribute
FINISHED GOOD
five whys
FK (foreign keys)
flexibility
foreign key relationships
  STATUS APPLICATION
foreign keys (FK)
  CITY
  COUNTRY
  ENTITY 1
  PARTY entity
  POSTAL ADDRESS PART
  POSTAL CODE
  STATE
  from status type id
  to status type id
Friedland, David
from date attribute
from status type id foreign key

G

GAAP (General Accepted Accounting Principles)
Gamma, Erich
Gang of Four
general styles of modeling
  benefits
generalized outcomes
generalized styles of modeling, specific in same model
GENERIC CONTEXTUAL ROLE
geographic boundaries
  contact mechanisms
  data and
  electronic address
  postal addresses
geographic boundaries pattern
GEOGRAPHIC BOUNDARY
  many-to-many relationship to
  subtypes
GEOGRAPHIC BOUNDARY ASSOCIATION
GEOGRAPHIC BOUNDARY ASSOCIATION TYPE

H

Hawkins, David
Hay, David
Helm, Richard
Herald, Don
hierarchies
  adding
  aggregation comparison
  creating
  ENTITY 1 and
  ENTITY CATEGORY TYPE
  Level 1 Recursive Pattern

ownership
ROLE TYPE entity
static, Level 1 Recursive Pattern
unchanging
hierarchy
ENTITY 2 and
ENTITY 3 and
how-not-to-do pattern
Hybrid Contextual Role Pattern
hybrid modeling solutions

I

IDEFIX (Integration Definition for Information Modeling)
identification, contact mechanisms
Identifier pattern
illustration tables, example data
inconsistencies
inconsistent data
indicators
INDUSTRY TYPE
instructions
intermediary ETL databases
INTERNATIONAL TRIPS FOR CURRENT YEAR
INVOICE
Ishikawa, Sarah

J

Johnson, Ralph

L

legacy systems
Level 3 Business Rules Pattern
Level 1 Classification Pattern

Level 2 Classification Pattern
Level 3 Classification Pattern
Level 3 Classification Pattern with Rollups and Schemes
Level 1 Contact Mechanism Pattern
  static data
Level 2 Contact Mechanism Pattern
Level 3 Contact Mechanism Pattern
  as foundation for database design
Level 4 Contact Mechanism Pattern
  as foundation for database design
Level 1 Contextual Role Pattern, Attributes
Level 1 Contextual Role Pattern, Relationships
  declarative roles and
Level 2 Contextual Role Pattern
Level 2 Contextual Role Pattern, PARTY Only Alternative
Level 3 Contextual Role Pattern
Level 1 declarative role pattern
  business requirements description
  DECLARATIVE ROLE
  declarative roles, identifying
  entities
    attributes
    definitions
  implementations
  multiple roles
  organizations
  party concept
  persons
  roles as specific entities
  specific roles
  static roles
  weaknesses
Level 2 declarative role pattern

Level 3 declarative role pattern
Level 1 patterns
  as foundation
  Sands Distribution
Level 2 patterns, Sands Distribution
Level 1 Recursive Pattern
Level 2 Recursive Pattern
Level 3 Recursive Pattern
Level 3 Recursive Pattern with Rules
Level 1 Status pattern
Level 2 Status Pattern, Current Status
Level 3 Status Pattern
Level 4 Status Pattern
LIMIT RULE
location of contact mechanism
LOGISTICS SERVICE PROVIDER declarative role
Lopez, Karen
lower level subtypes

M

many-to-many relationships, recursive
MARKET RISK RULE
MDM (master data management) data models
  Sands Distribution
metadata, business data and
motivations in using patterns
multiple classifications
mutually exclusive classifications

N

NAME
name attributes, declarative roles
Name pattern

need for book
non-solicitation data
normalization
normalized patterns
notation

O

ODL (Object Description Language)
one-to-many relationships
Oracle Designer Notation
ORDER
  contact mechanism
  FACILITY and
  PARTY and
  Sands Distribution
ORDER CONTACT MECHANISM
order description attribute
ORDER ELECTRONIC ADDRESS
order entry system
order expected payment date attribute
ORDER POSTAL ADDRESS
ORDER RULE
ORDER STATUS
ORDER TELECOMMUNICATIONS NUMBER
orders
  Entered
  Entry Complete
ORDER(s)
  as a commitment to purchase goods or services
  classifications
  order id attribute
  ORDER ITEM(s)
ORGANIZATION

  definition
  PARTY subtype
  supertype/subtype structure, party-specific attributes
ORGANIZATION UNIT
organizations
  attributes
  common information with people
  roles
OTOB (one thing in one box)
OUTCOME VALUE TYPE
outcomes
  generalized
  specific
owned objects
ownership, hierarchies
owning objects

P

Palmer, Ian
PARENT COMPANY
  Sands Distribution
parties
  contextual roles and
  definition
  forcing concept
  many roles
  as organization
  as person
  roles, distinguishing
PARTNER declarative entity
  partner type id foreign key
PARTY
  attributes

business email address attribute
classifications
contact mechanism
CONTACT MECHANISM AND
contextual roles
CUSTOMER
foreign key
INDUSTRY TYPE entity and
multiple roles
ORDER entity
ORGANIZATION
ORGANIZATION subtype
PARTY ROLE and
PERSON
PERSON subtype
supertype/subtype structure, party-specific attributes
PARTY ROLE
attributes
from date attribute
relationships
SPECIFIC CONTEXTUAL ROLE
subtypes
thru date attribute
party role
party role id primary key
PARTY ROLE supertype
attributes
declarative roles
primary key
subtypes
PARTY RULE
PARTY TELECOMMUNICATIONS NUMBER
pattern levels

patterns
adherence to
adopting
applications
Authorizations
Business Rules with Party Roles
as client tool for software or data models
conflict
Contact Mechanism with Flexible Address Parts Pattern
Contact Mechanism with Geographic Boundary
database physical design
declarative roles and
Events
geographic boundaries
how-not-to-do
Hybrid Contextual Role Pattern
Identifier
Level 2 Business Rules Pattern
Level 3 Business Rules Pattern
Level 1 Classification Pattern
Level 2 Classification Pattern
Level 3 Classification Pattern
Level 3 Classification Pattern with Rollups and Schemes
Level 1 Contact Mechanism Pattern
Level 2 Contact Mechanism Pattern
Level 3 Contact Mechanism Pattern
Level 4 Contact Mechanism Pattern
Level 1 Contextual Role, Attributes
Level 1 Contextual Role, Relationships
Level 2 Contextual Role Pattern
Level 2 Contextual Role Pattern, PARTY Only Alternative
Level 3 Contextual Role Pattern
Level 1 declarative role

Level 2 declarative role
Level 3 declarative role
Level 1 Recursive Pattern
Level 2 Recursive Pattern
Level 3 Recursive Pattern
Level 3 Recursive Pattern with Rules
Level 1 Status pattern
Level 2 Status Pattern, Current Status
Level 3 Status Pattern
Level 4 Status Pattern
motivations in using
Name
normalized
postal address parts
purpose for
resistance to
return on investment
significance
socializing
circumstances
experiences
as source for evaluation for enterprises
as standard for database structures
as standard for IT professionals
Status Category Pattern
Status Type with Multi Rollup and Rules Pattern
as toolkit for data professionals
as training materials
Transactions
trust in
Universal Patterns
upper-level management and
Pearce, Joseph Chilton

peer-to-peer relationships
  creating
people
  attributes
  common information with organizations
  roles
percentage complete attribute
PERSON
  attributes
  definition
  PARTY subtype
  supertype/subtype structure, party-specific attributes
personal country telephone code attribute
personal telephone number attribute
perspectives, illustrating
PHASE(s)
physical database design
PK (primary keys)
placeholders for contact mechanisms
POSTAL ADDRESS
  POSTAL ADDRESS PART(s)
postal address, entities
POSTAL ADDRESS BOUNDARY
POSTAL ADDRESS PART
  foreign keys
POSTAL ADDRESS PART(s), GEOGRAPHIC BOUNDARY(s)
postal address parts pattern
POSTAL ADDRESS subtype
postal addresses
  as contact mechanisms
  geographic boundaries
POSTAL ADDRESS(es)
POSTAL CODE

  foreign keys
Power Versus Force
prefecture
PRICE COMPONENT RULE
PRICE COMPONENT RULE FACTOR
PRICE COMPONENT RULE OUTCOME(s)
pricing
  PRODUCT
  PRODUCT FEATURE
primary keys, party role id
primary keys (PK)
  PARTY ROLE supertype
primary street address part attribute
primary suite-apartment attribute
Principles of the Business Rule Approach
PRIORITY RULE
process modeling
PRODUCT
  categorizations, changing
  classifications
PRODUCT BREAKDOWN
PRODUCT CATEGORY
  aggregations
  hierarchies
PRODUCT CATEGORY ROLLUP
PRODUCT CATEGORY TYPE
PRODUCT CATEGORY TYPE ROLLUP
PRODUCT COMPLEMENT
PRODUCT entity
  ACCESSORY subtype
  classifications
  color attribute
  disk capacity attribute

GOOD subtype
HARDWARE subtype
pricing and
product family attribute
product line 1 attribute
product line 2 attribute
product type attribute
required disk space attribute
SERVICE subtype
SOFTWARE subtype
product family attribute
PRODUCT FAMILY entity
PRODUCT FEATURE entity, pricing and
PRODUCT INCOMPATIBILITY entity
product line 1 attribute
product line 2 attribute
PRODUCT LINE entity
PRODUCT OBSOLESCENCE entity
PRODUCT REPLENISHMENT RULE entity
PRODUCT RULE entity
PRODUCT SUBSTITUTION entity
product type attribute
PROJECT
contextual roles
project lead attribute
project name attribute
project sponsor attribute
project worker attribute
project id(s) attribute
project lead attribute
PROJECT LEAD contextual role
PROJECT LEAD declarative role
project name attribute

PROJECT SPONSOR
project sponsor attribute
PROJECT SPONSOR contextual role
project worker attribute
PROJECT WORKER contextual role
prototyping data models
purpose for patterns

R

range of dates
receiving parties
recursion
  ENTITY CATEGORY
  Level 1 Recursive Pattern
  Level 2 Recursive Pattern
  Level 3 Recursive Pattern
  Level 3 Recursive Pattern with Rules
  many-to-many relationships
  ROLE TYPE entity
  type entities
recursive relationships
  data organization
  groups
  mandatory
  modeling
  multiple
  self-references
  type entity
redundancies
redundant data
relational modeling
relationships
  as business rule

cardinality
consolidation
CONTACT MECHANISM
contact mechanisms
data types
declarative roles
enterprise data models
foreign key
Level 1 Contextual Role Pattern, Relationships
declarative roles and
many-to-many, associative entities
one-to-many
optionality
over time
PARTY ROLE
peer-to-peer
recursive
STATUS APPLICATION
reports
resistance to patterns
return on investment with patterns
reusable data models
reusable models
ROLE TYPE
authorized rule user
CONTEXTUAL ROLE
hierarchy
PARTY TYPE subtype
recursion and
rule manager
rule source
rule specifier
subtypes

roles
  as attributes
  data as indivisual contextual role attributes
  individual
  organizations
  people
  same involvement many times
  specific uses
Ross, Ron
RULE FACTOR TYPE
RULE FACTOR VALUE
RULE OUTCOME TYPE
RULE OUTCOME VALUE
RULE SOURCE
rule statement
rules
  accessibility
  classifying
  external reference id
  managing
  notes
  references
  sources
  specifying
  statuses, capturing

S

sales territory
Sands Distribution
  application data models
  CUSTOMER
  customers, active
  data warehouse/data mart data models-star schemas

  data warehouse models
    relational approach
  Level 1 patterns
  Level 2 patterns
  MDM (master data management) data models
  ORDER
  PARENT COMPANY
  scope statement
scheduled start date attribute
scheduled start date(s) attribute
schemes
  ENTITY CATEGORY TYPE
  ENTITY CATEGORY TYPE SCHEME
  name
scope
scoping data models
sending parties
service region
SHIP TO CUSTOMER contextual role
SHIP TO CUSTOMER role
ship-to customers
ship to postal address part 1 attribute
SHIPMENT
  statuses, ORDER statuses and
shipment, life span
SHIPMENT CARRIER contextual role
SHIPMENT DELIVERY
Shipment Planned status
SHIPMENT RULE
shipment schedule date attribute
SHIPMENT SCHEDULED
SHIPMENT STATUS
shipping, time sensitivity

shipping address, vs billing address
SIC (Standard Industrial Classification) scheme
Silverstein, Murray
SME (subject matter experts)
socializing patterns
  circumstances
  differences in
  experiences
SPECIFIC CONTEXTUAL ROLE
  DECLARATIVE ROLE
  PARTY ROLE
specific outcomes
specific styles of modeling
  benefits
  generalized in same model
SPONSOR declarative role
star schema dimensions
STATE
  foreign keys
state
STATE-REGION
statement of business rules
static hierarchies, Level 1 Recursive Pattern
status
  as attribute of entity
  attributes
  as attributes
  categorization
  classifications
    reporting on
  current, old
  definition
  dynamic

entities
flexibility
great number for entity
Level 1 Status pattern
Level 2 Status Pattern, Current Status
Level 3 Status Pattern
Level 4 Status Pattern
ORDER ITEM(s)
overview
rules
capturing
static
time
types, rules
updating
STATUS APPLICATION
foreign key
Status Category Pattern
status datetime attribute
status from date attribute
Status Pattern
status thru date attribute
STATUS TYPE
capturing statuses
description attribute
effective from date attribute
ENTITY STATUS TYPE subtype
long name attribute
many-to-many recursive relationship
relationships, capturing
shipment overdue attribute
short name attribute
STATUS TYPE ASSOCIATION

  foreign keys
STATUS TYPE ASSOCIATION RULE
  values
STATUS TYPE ASSOCIATION TYPE, STATUS TYPE, associations
STATUS TYPE CATEGORY
  Order Fulfillment
  STATUS TYPE associations
STATUS TYPE CATEGORY CLASSIFICATION(s)
STATUS TYPE CATEGORY TYPE
Status Type with Multi Rollup and Rules Pattern
statuses, types, multiple sets
straw man project plans
SUBASSEMBLY
subentities
SUBSIDIARY
subtypes
  lower level
supertypes
  levels
SUPPLIER
SUPPLIER declarative role entity
  attributes
  ORGANIZATION
suppliers

T

TASK(s)
taxation identifier attribute
taxonomy
  definition
TELECOMMUNICATIONS NUMBER
TELECOMMUNICATIONS NUMBER CLASSIFICATION
TELECOMMUNICATIONS NUMBER subtype

telephone extensions
telephone number attribute
telephone numbers, capturing
templates
territory
The Data Administrator Newsletter
The Sedona Method
thru date attribute
time, status and
to status type id foreign key
TRADE
transactions, authorizing
Transactions pattern
trust in patterns
type entities
  entity instances
  recursion
types
  definition

U

UID (unique identifier)
UML (Unified Modeling Language)
  state
universal, definition
Universal Patterns
Universal Patterns for Data Modeling
  blueprints and
updating status
upper-level management and patterns

V

Vlissides, Johns

W

WORK EFFORT
  classifications
  definition
  go/no go
WORK EFFORT ASSOCIATION
WORK EFFORT ASSOCIATION RULE
WORK EFFORT ASSOCIATION TYPE
WORK EFFORT BREAKDOWN
work effort id attribute
WORK EFFORT PRECEDENT
  from date attribute
WORK EFFORT STATUS
WORK EFFORT TYPE
WORKER
WORKER declarative role

Z

Zachman, John
Zachman Framework