Beginning Relational Data Modeling

Second Edition

SHARON ALLEN AND EVAN TERRY

Apress®

Beginning Relational Data Modeling, Second Edition

ISBN (pbk): 1-59059-463-0

Printed and bound in the United States of America 9 8 7 6 5 4 3 2 1

Lead Editor: Tony Davis
Technical Reviewer: Evan Terry
Editorial Board: Steve Anglin, Dan Appleman, Ewan Buckingham, Gary Cornell, Tony Davis, Jason Gilmore, Jonathan Hassell, Chris Mills, Dominic Shakeshaft, Jim Sumser
Assistant Publisher: Grace Wong
Project Manager: Sofia Marchant
Copy Manager: Nicole LeClerc
Copy Editor: Kim Wimpsett
Production Manager: Kari Brooks-Copony
Production Editor: Ellie Fountain
Compositor: Dina Quan
Proofreader: Sue Boshers
Indexer: Kevin Broccoli
Artist: Kinetic Publishing Services, LLC
Cover Designer: Kurt Krames
Manufacturing Manager: Tom Debolski

Distributed to the book trade in the United States by Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013, and outside the United States by Springer-Verlag GmbH & Co. KG, Tiergartenstr. 17, 69112 Heidelberg, Germany.

In the United States: phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders@springer-ny.com, or visit http://www.springer-ny.com. Outside the United States: fax +49 6221 345229, e-mail orders@springer.de, or visit http://www.springer.de.

For information on translations, please contact Apress directly at 2560 Ninth Street, Suite 219, Berkeley, CA 94710. Phone 510-549-5930, fax 510-549-5939, e-mail info@apress.com, or visit http://www.apress.com.

I would like to dedicate this book to the three wonderful women who have become very special to our family, a.k.a. "The Girls": Julie Allen née Herlehy, Leslie Teeter (soon to be Allen), and Christy Bourdaa.
—Sharon Allen

I would like to dedicate this book to our new instance of FamilyMember, Caitlin Elizabeth Terry, who provided much feedback during the book's revision.
—Evan Terry

Contents at a Glance

About the Authors xix
Acknowledgments xxi
Introduction xxiii

CHAPTER 1 Understanding and Organizing Data: Past and Present 1
CHAPTER 2 Introducing Relational Theory 27
CHAPTER 3 Understanding Relational Modeling Terminology 57
CHAPTER 4 Understanding Data Modeling Methods: Graphical Syntax 89
CHAPTER 5 Introducing Object-Oriented Data Modeling 107
CHAPTER 6 Examining Levels of Analysis 131
CHAPTER 7 How Data Models Fit Into Projects 163
CHAPTER 8 Building a Conceptual Model 191
CHAPTER 9 Building a Logical Model 239
CHAPTER 10 Transforming a Logical Model into a Physical Model 303
CHAPTER 11 Designing a Physical Model Only 347
CHAPTER 12 Introducing Dimensional Data Modeling 377
CHAPTER 13 Reverse-Engineering a Data Model 421
CHAPTER 14 Communicating with the Model 467
CHAPTER 15 Improving Data Quality and Managing Documentation 487
CHAPTER 16 Introducing Metadata Modeling 517
CHAPTER 17 Exploring Data Modeling Working Practices 533
APPENDIX A Data Modeling: Resources 563
APPENDIX B Glossary 569

INDEX 589

Contents

About the Authors xix
Acknowledgments xxi
Introduction xxiii

CHAPTER 1 Understanding and Organizing Data: Past and Present . . . 1

What Is Data? 1
Data Management in History 4
What Is Relational Data Modeling? 7
Understanding the Data Life Cycle 9
Need It 9
Plan It 10
Collect It 10
Store It 10
Combine It 11
Act on It 11
Archive It 12
Remove It 12
Need It Again 12
Restore It 13
How a Data Model Helps 13
Who Are the Data Modelers in Most Organizations? 14
Defining the Role 15
A Data Modeler Charter 18
Job Titles 19
Understanding As-Is Support 20
Support Configuration Management 20
Provide Impact Analysis 21
Promote Data Management Departmental Standards 21
Provide Data Integrity Assessments 22
Research Existing Techniques and Tools 22

Understanding To-Be Support 23
Design New Data Structures 23
Provide Expert Advice 23
Provide Alternatives 24
Provide Expectation Assessments 24
Research New Techniques and Tools 24
Summary 25

CHAPTER 2 Introducing Relational Theory 27

Understanding Database Models 28
Hierarchical Databases 28
Network Databases 29
Relational Databases 30
Taking a Relational Approach to Data Modeling 33
Origins of Relational Theory 33
Relational DBMS Objectives 34
What Are Codd's Rules of an RDBMS? 35
Introducing Normalization 40
Universal Properties of Relations 41
First Normal Form (1NF) 46
Second Normal Form (2NF) 48
Third Normal Form (3NF) 50
Boyce-Codd Normal Form (BCNF) 51
Introducing Denormalization 53
Derived Columns 54
Deliberate Duplication 54
Deliberately Removing or Disabling Constraints 55
Deliberate Undoing of Normalization Forms 55
Summary 56

CHAPTER 3 Understanding Relational Modeling Terminology 57

Introducing Conceptual/Logical Modeling Concepts 57
Entities 58
Category Entities 62
Associative, or Intersection, Entity 65
Attributes 68
Keys 70
Relationships 75
Relational Model Business Rules 79

Introducing Physical Modeling Concepts 82
Tables 82
Views 84
Columns 85
Constraints 86
Summary 87

CHAPTER 4 Understanding Data Modeling Methods: Graphical Syntax 89

Integration Definition (IDEF1X) 89
Boxes 90
Lines 93
Terminators 96
Entity-Relationship (ER) or Chen Diagramming 99
Information Engineering (IE) 101
Barker Notation 104
Summary 106

CHAPTER 5 Introducing Object-Oriented Data Modeling 107

Introducing Object-Oriented Design 107
Object-Oriented Models: Class Diagrams 108
Unified Modeling Language (UML) 109
UML Syntax 111
Classes vs. Entities 111
Generalizations vs. Categories 112
Relationships vs. Associations 113
Supporting Object Models with Relational Databases 116
General Transformation Rules 116
Transforming UML Classes into Entities 117
Transforming UML Associations into Relationships 118
Class Hierarchies 120
Example UML Transformation 123
Initial Logical Data Model 123
Transformation to UML Class Diagram 124
UML Class Diagram to Relational Physical Model 126
Lost in Translation: Impedence Mismatch 128
Summary 129

CHAPTER 6 Examining Levels of Analysis 131

Developing Models 132
A Data Model Isn't a Flow Diagram 134
Data Relationship Rules 136
Performing Conceptual Analysis 137
Entities in a Conceptual Model 138
Relationships in a Conceptual Model 138
Conceptual Model Example 139
Performing Logical Analysis 140
Entities in a Logical Model 141
Attributes 142
Logical Analysis Example 146
Performing Physical Analysis 147
Tables 148
Physical Analysis Example 151
Performing Reverse-Engineered Analysis 152
Analysis Is in the Detail 154
Entity-Level Detailing 155
Key-Based (KB) Detailing 156
Fully Attributed (FA) Detailing 158
Summary 160

CHAPTER 7 How Data Models Fit Into Projects 163

The Project 163
Project Management 164
Scope 165
Project Standards 166
Methodologies 166
DACOM Methodology 167
ISO 9001 167
Capability Maturity Model (CMM) 168
Project Team Needs 169
Project Initiators/Sponsor 169
Business Team 170
Development Team 171
Extraction, Transformation, and Loading (ETL) Team 171
Business Intelligence (BI) Team 172
Extended Development Team 172

Budget and Schedule 173
Project Life Cycle 173
Strategize 174
Analyze 175
Design 177
Build and Test 178
Deploy 179
Project Type 179
Enterprise 180
Transactional: OLTP 180
Data Warehouse: Enterprise Reporting 181
Project Style Comparison 182
Model Objective 183
Abstraction Models 183
Data Element Analysis Models 185
Physical Design Models 186
The Right Model 187
Project Type 187
Model Goal 187
Customer Need 188
Model Tips 189
Summary 189

CHAPTER 8 Building a Conceptual Model 191

Modeling the Business Rules 191
Defining the Objectives 192
Defining the Scope 194
Defining the Approach 196
Top-Down Approach 196
Bottom-Up Approach 197
Documenting the Process: Top-Down Approach 198
Deciding on Interview Sources 199
Defining Solitaire Activities 199
Defining the Solitaire Process Steps 199
Building Activity Descriptions 202
Identifying the Important Elements 202
Defining Your Elements 204
Validating Your Work 205
Aggregating into Concepts 206

Documenting the Process Rules: Bottom-Up Approach 208
Documenting the Rules of the Activity 208
Describing the Rules 209
Identifying the Important Elements 210
Defining Your Elements 211
Comparing Methods 213
Building the Conceptual Model 214
Enhancing Conceptual Definitions 215
Adding Relationships 217
Checking the Business Rules 230
Checking the Relationships 231
Publishing the Model 236
Summary 237

CHAPTER 9 Building a Logical Model 239

Reviewing the Conceptual Model As a Guide 240
Validating the Model 242
Using the Feedback 242
Defining the Subject Areas 243
Starting the Logical Data Modeling 244
Modeling the Card Subject Area 245
Analyzing the Card Entity 246
Analyzing the Card Category 247
Analyzing the Card Relationships 249
Analyzing the Card Entity Details 254
Modeling Card Movement Subject Area 275
Analyzing Card Pile 275
Analyzing Play 279
Combining Card Movement 281
Looking at the Attributes, Keys, Definitions, and Relationships . . 284
Card Movement Subject Area Review 287
Modeling the Event Subject Area 288
Analyzing the Event Entity 288
Analyzing the Event Relationships 289
Event Subject Area Review 291
Putting It All Together: The Full Picture 291
Relating Game to Play 292

Performing Quality Assurance Checks 293
Checking for 1NF 294
Checking for 2NF 295
Checking for 3NF 296
Checking for BCNF 296
Checking for Too Many/Too Few Attributes 296
Checking for Accurate Role Names 297
Checking Quality with Instance Tables 297
Performing a Peer Model Walk-Through 299
Applying Finishing Touches 299
Summary 301

CHAPTER 10 Transforming a Logical Model into a Physical Model . . . 303

Checking the Project Status 304
Taking the Next Steps 304
Beginning the Process 305
Physicalizing Names 305
Creating Tables from Logical Categories 310
Introducing Rolled-up Categories 311
Introducing Rolled-down Categories 314
Introducing Expansive Categories 316
Returning to the Solitaire Example 318
Examining Shadow Entities 321
Deciding the Primary Keys 321
Reviewing the Primary Keys 323
Adding Data Types and Sizing 335
Performing Quality Checks and Getting Extra Value 338
Building Instance Tables 338
Double-checking Names and Definitions 338
Reviewing the Requirements 339
Telling the Story 339
Identifying the Data Stewards 339
Building Test DDL 339
Reviewing Other Potential Issues 342
Adding Operational Columns 342
Documenting Population 343
Documenting Volatility 344
Recognizing the Model's Power 344
Summary 345

CHAPTER 11 Designing a Physical Model Only 347

Real-World Constraints 347
Where to Start 348
The Solitaire System 349
Model Exactly What You See 350
Start with Labels 350
Apply Naming Standards 351
Create Lookup Tables 352
Look Again for Important Data Sets 353
Check Out the Text Fields 354
Look at Keys and Data Types 355
Quality and Compromise 357
Something a Bit More Challenging 359
Classifying the Data Elements 361
Text Fields 362
Normalization 364
Other Physical Model–Only Designs 373
Operational Tables 373
Staging Tables 374
Archive Tables 375
Summary 376

CHAPTER 12 Introducing Dimensional Data Modeling 377

Understanding OLAP Database Basics 378
Asking Analytical Questions for Dimensional Models 379
Introducing Dimensional Model Terminology 380
Introducing the Benefits of Dimensional Design 381
Introducing Star Schemas 382
Understanding Facts and Dimensions 383
Avoiding Source System Keys 384
Avoiding Smart Keys 384
Avoiding Multicolumn Keys 384
Understanding Fact Granularity 384
Introducing Snowflake Schemas 385
Understanding Base Dimensions and Subdimensions 386
Revisiting the Solitaire Model 388
Targeting the Facts 389
Creating the Game Data Mart 395

Finishing the Game Data Mart 406
Reviewing the GameFact Data Mart: What Have You Done? 412
Creating the GameMove Data Mart 412
Finishing Up 419
Summary 420

CHAPTER 13 Reverse-Engineering a Data Model 421

Getting Started 422
Determining the Project's Needs 422
Obtaining Resources 423
Analyzing the Data Structure 424
Modeling Tool Support 424
Manual Processing 432
Structure Assessment 436
Analyzing the Data 442
SELECT COUNT 442
SELECT COUNT/GROUP BY 443
SELECT COUNT DISTINCT 443
SELECT MIN 443
SELECT MAX 443
SELECT 444
Assessing the Data 444
Reviewing Data Rules in Code 445
Analyzing the Front End 447
Screen Labels 448
Data Relationship Screen Rules 451
Derived Values 453
Using Historic/Descriptive Information 454
Applying Finishing Touches 455
Building a Logical Model 456
Names 457
Keys 458
Categories 460
Other Rules 463
Relationship Names 464
Finishing It Up 465
Summary 466

CHAPTER 14 Communicating with the Model 467

Why Add More? 467
Arranging Elements 468
Adding Text 470
Adding Titles and Headings 470
Adding Notes 480
Adding Legends 481
Adding Graphics/Images/Icons 482
Publishing Data Models 484
Publishing on the Web 484
Sharing the Files 485
Archiving the Files 485
Summary 486

CHAPTER 15 Improving Data Quality and Managing Documentation 487

Introducing Data Quality 488
Fidelity Analysis (Data Quality Analysis) 489
Criticality Analysis 491
Sensitivity and Privacy Analysis 493
Stewardship 495
Cross-Footing 497
Process Validation 498
Risk and Mitigation Analysis 499
Using the Data Model As a Knowledge Framework 501
Relationships to the Conceptual Model 504
Relationships to the Logical Model 506
Relationships to the Physical Model 508
Pedigree Mappings 511
Role Name Mappings 512
Summary 515

CHAPTER 16 Introducing Metadata Modeling 517

Defining Metadata 517
Technical Metadata 520
Business Metadata 521
Live Metadata 521
Collecting Metadata 523
Building a Metadata Repository 524
Conceptual Metadata Model 525
Logical Metadata Model 527
Physical Metadata Model 528
Understanding a Data Modeler's Metadata 529
Data Modeling Tools and Metadata 529
Data Modeler: A Customer of Metadata 530
Understanding the Future of Metadata 530
Summary 531

CHAPTER 17 Exploring Data Modeling Working Practices 533

Introducing Work Practices 533
Worst Practices 533
Best Practices 541
Understanding Data and Design 548
Logical-to-Physical Transformations 549
Fallacies About Physical Data 549
Understanding Project Lessons 551
Custom Solution Projects 551
Purchased Solution Projects 555
Legacy and Forensic Analysis 557
Closing Thoughts 560
Model Reviews 560
Experience Counts 560
Responsibility vs. Authority 561
Summary 561

APPENDIX A Data Modeling: Resources 563

Organizations 563
DAMA International 563
AITP 563
Books 563
Internet Resources 566
Newsletters 566
Online Magazines 566
Training 566
Courses 566
Conferences 567

APPENDIX B Glossary 569

INDEX 589

About the Authors

SHARON ALLEN has enjoyed learning the intricacies of several careers since 1975, including medical assistant, inventory control supervisor, data administrator, HTML and PL/SQL programmer, and data warehouse architect. This may sound a bit confused, but it has given her a unique perspective and sensitivity to the goals and pressures of both business and information technologies. This has been a huge asset for her as a data modeler; she works as an ambassador and interpreter of both business and information technologies in order to successfully build data requirements and design data solutions.

Sharon could be role-named as Wife, Mother, Daughter, Sister, Friend, Author, Data Modeler, Programmer, Stained-Glass Crafter, Crocheter, Playwright, Storyteller, Gardener, Teacher, and World Traveler. Recently added roles include Mother-in-Law, Conference Speaker, Wedding Planner, and (soon to be) Grandmother. She knows she doesn't fulfill any of these roles nearly well enough, but she has a great time trying.

EVAN TERRY has been in the IT industry for more than 15 years as a programmer/analyst, systems engineer, custom software consultant, senior developer, data analyst, and data architect, serving government and the private sector. He firmly believes that in order to succeed at developing complex systems, the IT professional must *truly understand the business processes* he supports. Evan tries to bridge the gap between the technical and the nontechnical by understanding the perspectives of both and helping the two groups communicate effectively.

In addition to a fascination with North American international relations, the following entities would be included in Evan's logical subject area: Charitable Organization, Ice Hockey, Trail Running, Hiking, International Travel, Vocal Music, Foreign Language, and Labor Day Weekend (although this subject area would likely still be considered incomplete).

Data Modeler Blessings

May the Creator gather us into one entity defined as members of the set Humanity.

May our primary key help us to recognize ourselves as irreplaceable and unique for all time.

May all our mandatory attributes be filled with good values and our optional attributes be generously supplied.

May all our relationships be for good, gainful, and positive purposes for building strong connections in our travels as foreign keys both as recursions to ourselves and as intersections with other entities in this universe.

And may any categorization be recognized as less a division, separating us from each other, and more a highlighting of what makes us special.

—Sharon Allen, 2002

A bad design is the bulldozer of ideas. Inelegant and ruthless. It leaves you to sift through the rubble of what might have been. A good design, done with style, takes longer but will always serve you better in the end.

May your teammates be open to new ideas.

May your management allow you the time to do your job.

May your encounters with bulldozers be few and your designs be elegant.

—Evan Terry, 2005

Acknowledgments

To John Baker and Paul Yamor, who let me try data modeling in the first place: thanks for giving me a chance! And to Shan Pao who graciously read through the raw material.

To the gang who helped me learn this craft and how to wield it wisely: Wendell, Ted, Paul, Janet, Karen, Drew, Cindy, Ron, Dennis, Charles, Cory, Camelia, Sanjiv, Geraldo, Richard, Ella, Jan, Paul, David, Stuart, Wendell, Rich, Phil, Michael, John, Dave, Daryl, Danny, Brad, Yamile, Cindy, Rex, Arno, Steve, Rick, Jane, Susan, Arun, Evan, and Les. Thanks for taking the time to explain "one more time" and share your wisdom and experience.

To the people who bought the first edition and critics on Amazon.com: thanks for inspiring me to go through the process of a second edition. I hope you like this one better; I do!

To the IT writers who gave me an encouraging word: Sid Adelman, Larry English, David Marco, and Michael Brackett. Thanks for the vision of possibility!

To the special people who kept me going when the going got rough: Doug, Dan, Julie, Tim, Leslie, and Mike (my gang), as well as Mom, Charlie, Evan, and Suzanne. I wouldn't have gotten here without you!

To Evan Terry, my coauthor, who helped me realize a much better version of my dream.

To everyone who had to put up with me through yet another odyssey: thanks for all the understanding and patience during my roller coaster highs and lows.

—Sharon Allen

To Sharon, for allowing me the opportunity to be a part of this work.

To my wife, Jen, for supporting me in taking on this project.

To Russ Robinson and the gang at CFS, who got me going in the first place.

To Mike Elsesser, who allowed me to grow from programmer to data architect.

To Jay Flynn and Robert Kreznarich, whose friendship and whose input into UML and object-relational mapping techniques were critical to me retaining my sense of humor amid the boots to the head. I really appreciate it.

To those who helped me learn the art and science of data modeling: Chris Dufour, Joe Mayer, Loren Brewster, Christopher Thames, Robert Kreznarich, Jay Flynn, Sharon Allen, and Arun Yarlagadda.

—Evan Terry

To the team who made something out of our first attempts: the editors, reviewers, and development team. Three cheers!

—Sharon Allen and Evan Terry

Introduction

Data modeling is a skill set that has a long history and yet is often not immediately recognized in the IT world. Although jobs for programmers and database administrators are regularly advertised, few companies employ people in a role titled *data modeler*. However, data design, which is an awareness of how data behaves and what it means within an organization, is supremely important in ensuring that company data systems work efficiently. It's often only when systems start misbehaving, or can't cope with the data in a way that the clients want it to, that poor database design and incomplete data understanding is exposed. How many times have you tried to cancel a gas or electric utility account only to find that the phone operators can't erase your records because of constraints in their data system? Instead of modifying or removing your details, you're forced to set up another account. This is not only time consuming but also a poor use of existing data. Accurate data modeling and subsequent database design can minimize these problems and ensure that each system is truly tailored to the needs of the clients.

Relational databases provide a powerful means by which to store and manipulate vast volumes of data. The design of such systems is based on the mathematics of relational theory, which was advanced in its initial stages by E. F. Codd. This mathematical basis has meant that many of the standard database design titles aren't easily accessible to many would-be modelers who want to know how to practically implement the theories they read about. This book focuses on giving the reader a background in the field of relational data modeling. The final physical implementation of these designs is beyond the scope of this book (it merits a title of its own), thanks to the complexities that can result in performance tuning a database application. The aim of this book is to provide a practical guide to the what and how of data modeling in a day-to-day, task-oriented way, rather than from a purely theoretical viewpoint. It has been more than 40 years since Codd's initial work, and yet we're still waiting to see a commercial relational database that can be said to be truly relational. In the real world, theoretical perfection, as prescribed by Codd's own rules for a relational system, has to be compromised by system performance. The most useful aspect of the art of data modeling is understanding the data better so you can suggest compromises and understand the risk factors that accompany them.

What This Book Covers

The book is split into three sections. The first seven chapters of this book deal with the basic skill set that a data modeler has to have.

Chapter 1 begins your journey into data modeling by covering the nature of data and data modeling, as well as its use within an enterprise. It recognizes the various guises in which data modeling makes its way into job descriptions within business and gives an overview of the ultimate goals of modeling.

Chapter 2 delves into the theory underpinning the relational model and covers the rules by which a database is assessed as relational. It then covers the process of normalization by which data is broken down so as to adhere to the premises of relational theory. It also briefly touches on the practical nature of modeling and where Codd's rules need to be bent so as to improve the performance of the given application.

Chapter 3 looks at the terminology involved in Conceptual, Logical, and Physical relational data modeling. This is where the mathematical basis of relational theory is left to others to expound; the focus is on what the various concepts equate to in a real-world setting. We also introduce the syntax by which we graphically represent data models, focusing primarily on the notation we'll use throughout the book.

Chapter 4 looks at the world of Logical and Physical modeling using graphics. This is where various database structures are designed to store data for various purposes. Again, we review the physical graphic syntax used throughout the book.

Chapter 5 briefly compares relational modeling and object-oriented data modeling. We cover the differences and overlaps using UML as a basis of graphic syntax comparison. This chapter tries to provide a simple translation technique you can use to deploy an object model to a relational database.

Chapter 6 provides a guide to the various types of analytical models that you can be asked to produce as a data modeler and the level of detail you can choose to deliver. Tailoring your model to the needs of your audience is of key importance since a lack of comprehension of the model makes the exercise itself entirely futile. Knowing what level of detail is appropriate in different circumstances is important for a modeler; by providing the right information at the right time, you make yourself an indispensable part of any development team. For example, while project managers will want an overview of the data model you've devised, programmers will need more detail in terms of the data types and size constraints they need to use in implementing your model. You need to be flexible in providing models for a variety of needs.

Chapter 7 covers how models fit into development projects. The design of a new application passes through several iterations, from the beginning of the project where the data elements that need capturing are outlined at a general level of detail to the final model provided for physical implementation as a database system, which is highly detailed and often complex. Ensuring that the model meets the needs of the project requires careful management as well as effective communication between the modeler and client.

The next portion of the book puts the knowledge gained in the first half of the book to practical use by covering how to build a sample system. And in order to emphasize that the modeling techniques presented here are applicable to whatever system you happen to be working on, the chosen system isn't a sample proprietary database such as Oracle 8*i* or SQL Server 2000 but rather the electronic version of the card game Solitaire.

Chapter 8 begins the process of modeling by capturing the scope of the modeling project. This is where the initial boundaries of the model are drawn and the important concepts of the game are captured in graphical format. By the end of this chapter, we present a general model of the game for further analysis. In the course of drawing up this model, we show how to create a general checklist of tasks that you can apply to any general project.

Chapter 9 picks up this model and iterates through many of the same steps outlined in the previous chapter in developing the level of detail further. It takes the concepts of the previous chapter and converts them into logical entities on the model. This process is something of a "drill down"—the Conceptual model of the previous chapter is fleshed out with more detail regarding the nature of the data in the system.

Chapter 10 illustrates the basic principles needed to take the Logical model from the previous chapter and turn it into a model capable of being implemented as a physical database system. This is where the fine detail must be documented and room for errors is greatly diminished. If you have a poor design here, the implementation problems for the development team may be myriad. It's important at this stage in the model development to be clear about the definitions of the various data elements and how they relate.

Chapter 11 considers the situation where budget and time constraints mean you have to skip the work highlighted in earlier chapters and move straight to a Physical model. This isn't an ideal situation for a modeler, but it's one where you can still apply basic practices to improve the model's integrity and longevity. This chapter outlines how you can take appropriate shortcuts and still provide verifiable detail in the model.

Chapter 12 covers the basics of Dimensional modeling techniques. This presents a different challenge to those of the previous chapters since the data is structured differently to improve the efficiency with which users can sum, aggregate, and manipulate data to provide reports and summary sheets. This chapter also introduces all the necessary terminology along with two sample applications that illustrate how you can develop such ad hoc analysis designs.

Chapter 13 explores a different scenario altogether. While all the work until this point has focused on building systems from scratch, this chapter covers the process of reverse engineering, where you attempt to model an existing system. This, in many ways, is the reverse process of the previous chapters since you have to work your way back along the development chain, starting with the physical elements in the database, in order to return to a Logical or Conceptual model. This is often necessary in the reappraisal of systems, such as when companies merge and need to verify what data is being collected in each system or when looking to expand or enhance an existing system.

The final section of the book covers the added value that the modeler can bring to a development team.

Chapter 14 covers what enhancements a modeler can provide to make the graphical models they produce for their clients more readable. Simple additions of text, abbreviations, icons, and color can greatly enhance the readability of a model and give it a life beyond the point when you're no longer associated with it. The more clarity a modeler can bring to their work, the easier it is for others to both follow and utilize their work.

Chapter 15 looks at additional value issues in terms of the integrity and security issues surrounding the system data. Data integrity is of key importance to a system since nonsensical values naturally reduce the effectiveness of the application and user confidence in its viability. A modeler can do a number of simple tests to consider the nature of the data in the system and flag any obvious problems for the rest of the development team.

In cases where company-sensitive information is involved, you may need to modify the model design to protect the data from being publicly exposed. While the database administrator (DBA) is often the first port of call in restricting access to the databases being used, the modeler can help protect the data.

Chapter 16 looks at the nature of metadata, or data about data. It discusses the need to document descriptive information about data, which often appears of little value within the development team and which is generally lost over time since it often exists only in the heads of the team and not in paper or electronic format. This chapter also covers how to distribute data, and its use in repositories, for easy reuse. As the number of database systems grows, the number of model designs also increases. While each system will have its own special requirements, often previously modeled structures can be reused. This requires both easy access to previous models and good documentation of such models, which is where metadata repositories can prove to be valuable assets.

Chapter 17 is the final chapter in the book and summarizes some of the best practices modelers should aim for in their working environment. It highlights what traits a modeler should avoid in order to ensure that the project succeeds, and the chapter reinforces this with some best practices. It then covers some further project details, such as how long each of the various modeling tasks will take and some of the pitfalls to look for as you work your way through a project.

Finally, Appendix A shows where you can find more information about the topics in the book, and Appendix B includes a glossary of terminology related to data modeling.

Who Is This Book For?

This book assumes no prior knowledge of relational modeling or programming constructs. It's aimed at anyone who has a need for an easy-to-read, practical guide to relational data modeling. To that end, it has a broad appeal not only for developers and DBAs taking their first steps into database design but also for people such as data architects, information officers, and knowledge managers who simply want a greater understanding of how to analyze their data efficiently. To follow along with this book, there are no specific software needs since it shies away from a specific database platform. Access to a computer with a version of Solitaire, or even a pack of playing cards, may aid your understanding of the Solitaire example, but it isn't essential.

Conventions

We've used certain styles of text and layout in this book to help highlight special kinds of information. For example, program code will appear in a block formatted like the following:

```
ALTER TABLE Suit
    ADD ( FOREIGN KEY (ColorNm)
                REFERENCES Color ) ;
```

In addition, we've made every attempt to distinguish entity names for Physical data models, table names, and attribute names using a `monospaced font`.

Finally, *notes* (shown here) consist of incidental information of one type or another that defines, explains, or elaborates upon the main discussion. *Tips* contain information that should make your life easier. *Cautions* indicate a potential danger.

Note Remember that you're developing models for clients, not to suit your own needs.

CHAPTER 1

Understanding and Organizing Data: Past and Present

In this chapter you'll look at a bit of history of how humans have strived through the ages to find a good way to organize, locate, and understand the knowledge they gather. We'll show how data modeling has become the current industry standard in documenting what we know about data, its intrinsic nature, and business uses. And we'll review what data modeling involves and who, within any given organization, actually does the modeling. You'll look at the following:

- The nature of data vs. information vs. knowledge
- The ancient history of data management
- What data modeling is today
- How data modeling helps in dealing with the life cycle of enterprise data
- Who actually performs data modeling tasks in the organization
- In what activities the data modeler participates

What Is Data?

We're sure many term papers and master's theses attempt to answer this very question. The nature and importance of specific data elements are being challenged every day. What we've discovered over time is that we can't architect a good storage solution that supports creation, movement, management, integration, and storage functionalities without knowing more than the basic values, shapes, and sizes generally found in the element. You need to understand its meaning, use, importance, and security aspects as well.

If we were to describe the various characteristics of data in terms of its physical syntax (which you need to understand to design a basic "holder" for it), we would probably list something like this:

- **Data type**: Binary, numeric, textual, video, and so on
- **Size**: Maximum length or volume of memory required to store it
- **Precision**: Decimal allowance

- **Necessity**: Null or not null
- **Language**: English, Spanish, Japanese, and so on
- **Restricted domain**: The day names in a week, the numbers 0–9, Yes or No

You probably recognize these characteristics as ways to describe the rules for a data holder (column, field) in a screen or database. Describing how a data holder needs to be built is one way for you to understand data. Unfortunately, it's often the only aspect of data on which any emphasis is placed.

But it isn't hard to grasp that data is more than that. Each value means something to a person who understands its use. In other words, *data*, as we use the term, is both a logical building block that we call a *data element* with a shape and size as well as a bit of communication conveying a meaning that represents a single measurement, fact, or concept. You can't map data values as being the same data element with the same meaning based on shape and size alone. You can do that only if you're completely confident of what the value is trying to communicate. You can always change, or *transform*, data values to look the same if they can be counted on to mean the same thing.

This book is all about understanding and documenting data mostly from the perspective of a single theoretical abstract notion of meaning. We often analyze data values to gain understanding about that theoretical concept. Here are two definitions that note the differences between data and data elements:

- *Data* is a unit of readable language (such as numbers, characters, images, or other methods of recording) on a durable medium. Data on its own may carry no (or little) meaning. It needs an interpretation to fully process meaning.
- A *data element* is the smallest autonomous communication definition that imparts meaningful information, generally corresponding to a field in a database column of a spreadsheet or to a blank on a paper or electronic form. For example, names, addresses, work titles, Social Security numbers, record series titles, and record group numbers all represent data elements.

As a simple example, the meaning *yes* or *positive* in data could come in many data value forms: different human languages (ja, sí, da, yes), different computer languages (0, 1, ASCII codes), abbreviations (Y, N), full words (true, false), and even graphical symbols (+, –). A data value, such as these, when used in a well-defined data element, represents a meaning to someone using it. You need both halves in conjunction to communicate, which underscores the importance of understanding the data vs. accommodating the necessary means to store it. One of the things often missing in information technology (IT) is the desire to *understand* the nature of the data being used (and the business processes that use the data). Data modeling is the process we use to review and capture the data element and their relationships to each other. Frequently we review data values in order to gain understanding.

You should note that the terms *data*, *information*, and *knowledge* all have different meanings. *Data* are the values describing facts, *information* is the useful interpretation of data, and *knowledge* implies the ability to act on decisions based on that interpretation. Because the ultimate goal of data is to provide useful information that can be built into working knowledge, it must be modeled in such a way to aid that aim.

- *Information* is data that has been organized in a coherent and meaningful manner.
- *Knowledge* is information associated with rules, which allow inferences to be drawn automatically so that the information can be employed for useful purposes.

Data values used to be represented simply by text and numbers in a database; today data is captured as abstractions in digital format. It has expanded to include pictures, sounds, virtual 3D objects, and complex multimedia that blends all those things. The evolutionary growth of the very nature of data is challenging us all to figure out how to use technology to its maximum effectiveness, including how to properly use technology as a laborsaving device and as a means to enhance our information-processing ability.

Knowledge—that blend of familiarity, awareness, and understanding gained through pattern recognition—could be defined as the supposition that information will always support a specific conclusion. You may "know" it takes ten minutes to bring water to boiling using your current method of heating it. Therefore, you may stop measuring that time duration of a process because you "know" it to be true. You may know that political unrest in the Middle East will result in the price of crude oil rising, which leads you to believe gasoline prices will be going up next, which again leads you to include political climates as a variable for next year's budgets. Knowledge is an expected extrapolation based on collections of information.

Data is the raw material for information, and information is the raw material of knowledge.

We're pointing this out because data management (DM), information management (IM), and knowledge management (KM) are different, but they do have connections and dependencies, most notably to the quality of the data and the ability to decipher it back into its underlying meaning.

Data subjects, focus areas, and methods of data and data capture have changed. We used to think of data mostly in terms of inventory and accounts payable systems. We've added dynamic assessment of complex spatial data using dynamic uplinks to satellites that help us find our way around matching urban map data, noting locations of gas stations with the global positioning system (GPS) coordinates of our car to keep us from running out of gas. Online shopping applications remember either in databases or as cookies on customer systems exactly who we are and what we might be interested in buying. We even find ourselves under the scrutiny of cameras on public buildings that match our features with face recognition software to see if our location may need extra attention.

While people seem to want to collect ever-increasing amounts of data values, they also want to have it manipulated into information and knowledge in order to provide aggregates, trends, logic-driven prioritizations, derivations, and associations to other aspects of importance about the data (metadata). To support those desires, we need to analyze and model the targeted data in order to document meaning and build storage structures that support different styles of software design and functionality, such as the following:

- **Online transactional processing (OLTP)**: Event-by-event-driven data capture
- **Online analytical processing (OLAP)**: Specially organized data sets to support analytical pattern assessment
- **Operational data stores (ODS)**: Timely data collections usually used for reporting
- **Data warehousing (DW)**: Historic data collections used for any number of uses

To support these increasingly complex analyses, we need to better understand, manage, anticipate, and control data. The software that allows all these increasingly complex algorithms requires that the data is highly consistent and predictable. The understanding, management, anticipation, and control of data are all enhanced by proper data modeling and proper database design.

Our current view of data value management tells us that we need to collect and store data in the most elemental, atomic, basic, and reusable parts possible. Only by doing that can we isolate data meaning and build relationships that support the creation of newly defined information. And since we frequently need to combine those basic data elements, we need to design appropriate storage and processing facilities that allow this to be accomplished. For instance, look at any date we use. The data value is made up of three pieces combined: the day number, month number, and year number. We hardly even consider that all dates are really three distinct data elements. Data modeling is concerned with recognizing and separating the meanings of those data elements so that you can easily relate, group, locate, or just recognize all the 2004 dates without having to separate that part of the meaning from the full data value of a date.

The process we use to discover those basic data elements, determine how they relate to each other today, and define them so that they can be recognized and interpreted in the future is called *data modeling*.

Data Management in History

Let's look at a severely abbreviated history of data to show our drive to find more efficient ways of recording, collecting, ordering, searching, and protecting data. People have been working on these tasks for more than 5,000 years, and these tasks are still relevant today. Data management isn't just about computers and electronic information. It's also about our need to find and reuse information and to build knowledge that can be shared with future generations.

Language: The Origin of Data

Language, in its most generic interpretation as a useable communication medium, is the most recognized raw material from which data is created. Spoken language is much older and more mysterious than written language. It took quite some time for humans to discover a method of capturing the content and meaning of what was said in a form that would support gathering collections of persistent "data." In other words, it was a while before we started writing things down.

Writing became an important tool for dealing with the practical day-by-day economic, social, and administrative affairs of urban collections of people around 3500 BC. To continue to survive and flourish, people realized they needed better records and data management capabilities than the soft tissue between their ears permitted.

Sumerians developed the first widely used written language. Called *cuneiform*, it consisted of a series of little arrowhead shapes pointed in different directions. A shocking 95 percent of the document samples from that era, accumulated by archeologists, is economic in nature. Existing cuneiform records, in the form of stone tablets, are largely tallies, accounts, and records relating to personal property and temple business—data values that needed collection and maintenance to allow assessment and reporting on the local economy.

But early languages, such as cuneiform, weren't sophisticated enough for people. Symbols and pictograms weren't flexible enough to represent all the data people wanted to capture.

So written language transformed further, from pictograms to hieroglyphs and ultimately to symbols that could represent phonetic sounds. With these phonetic symbols, a spoken language and written language became much more closely tied. Now people could record songs, stories, and descriptions of life processes in subject areas from medicine to metallurgy, and they could develop sophisticated cultural artifacts such as poetry and literature.

Around 3200 BC the process of writing further evolved with the invention of a flexible writing material, made from the pith of the papyrus plant. This new material (the first type of paper) was much lighter than stone tablets but could still be awkward since it was generally used in the form of long strips, rolled up into scrolls. This meant you'd basically have to search an entire document to locate the single piece of information for which you were looking. Every request for data was pretty much a full document search!

Collecting and Organizing

As certain civilizations thrived, and in their growth absorbed other groups, reading and writing became even more important as a critical communication tool. As more and more data was recorded, people started to create larger and larger collections of it. The gathering piles of scrolls containing an ever-increasing quantity of information became rather like sand dunes. Each fact was a grain of sand that grew over time into mountains of great width and depth. They became proof of the human need to remember—and the databank came into being.

In the dictionary, a *databank* is defined as "a large collection of stored records from which data may be extracted or organized as desired." The operative words here are *extracted* and *organized*. Great collections of data weren't enough. There needed to be a way to easily extract key pieces of information from the mounds of scrolls.

In early 600 BC, King Ashurbanipal of Assyria wanted to accumulate and organize written records of Babylonian history in his library in Nineveh. When he finished, his library supposedly contained at least 5,000 stone tablets and scrolls, physically arranged by subject or type and organized in alcoves. A summary list of contents of each alcove was painted or carved on the alcove's entrance to serve as a catalog. While this was obviously not a very flexible design, he knew that having a large collection of data wasn't useful if you couldn't find the information for which you were looking.

Having data and being able to use it are two different things. Book and periodical catalogs, tables of contents, bibliographies, and cross-referencing indexes sprang into being almost as soon as collections of data were created, probably as a defensive measure by the information curators. Customers have always wanted data retrieved as quickly and successfully as possible.

Even within an individual scroll there were problems finding specific data. Locator structures such as book name or number, chapter number, and verse numbers were created to provide an address of sorts for specific information. These techniques were constantly evolving and being fine-tuned so that a librarian could accurately reference specific data using a location notation.

At the height of the Roman Empire, data storage evolved. The Romans found reading and writing those long, rolled-up scrolls to be troublesome, so they cut the parchment into pages and bound the pages together into the book form we still use today. They called it a *codex*. The design of the storage media (pages in a book) created a structure for the written material captured upon it. A page number could be used as a locator number for data values. You could navigate straight to the general location of the data without having to unroll the whole scroll.

In the tenth century, the Grand Vizier of Persia took his 117,000-volume library on the road with him. He insisted that the camels carrying his precious library be trained to walk in alphabetical order to preserve his ability to find his favorite volumes, even during the trip. Again, organization and knowledge of what was in the collection was of paramount importance to him.

After the Jesuits were dissolved in 1773, all the books they had collected were moved to a vacant church for storage. When the mice that plagued the church kept chewing on the books, the books were sorted by importance, with the most important being placed on shelves in the center of the storage room. All the other books were then stacked on the floor encircling the center shelves. The strategy was to let the mice gnaw away at the books on the floor, leaving those on the central stacks, which had been identified as being more important, alone.

Data organization and management has evolved throughout history. It has become easier and easier to discover the answer to our ever-expanding list of questions. Our insatiable need for information has increased as the media and processes that are used to capture and copy data become faster and less expensive. How many of us use the text search functions on the Internet to find information about just about anything? Data collection and storage have become part of our daily lives, and we're not just interested in mission-critical data but also in little details once thought unimportant. We're now able to capture, store, and retrieve detailed data, such as what products are bought together at grocery stores at 10 a.m. on Mondays or how many adult tickets were purchased for G-rated movies. We admit to having difficulty in subjectively calling any data unimportant—importance is in the eye of the data customer. We collect, organize, and correlate data about almost everything. It seems to be part of human nature.

■ **Note** Human data needs are getting broader and richer all the time.

Libraries and the library sciences have been at the forefront of data and information management. They were the first to tackle the challenges of managing collections of data, information, and knowledge. Even today many terms from the library sciences and the book publishing communities are used to describe facets of computer science and computerized data management, such as *coding*, *paging*, *indexing*, *cataloging*, and *referencing* data. Thus, today's media (electronic storage) and techniques (database management systems) are really just the tools and techniques of a point in time of the evolution of data management.

Data values, which began as a physical record used to remember something important about an event or thought, were handwritten, painstakingly reviewed, and artistically illuminated. What were at first rare and special artifacts of the educated and elite have became natural and unavoidable products of our lifestyles. We struggle now because so much data exists that it is hard to care about all of it. This makes the ability to target specific values or filter large collections an increasingly important task.

In today's world of data collecting, not all data passes through human hands anymore. Sometimes this is good, since it eliminates transcription errors and inherent subjectivity or bias. On the other hand, in doing this we have removed the natural quality checks such as human reasonability factors and judgment.