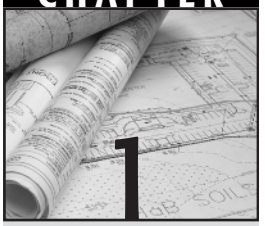
CHAPTER



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.

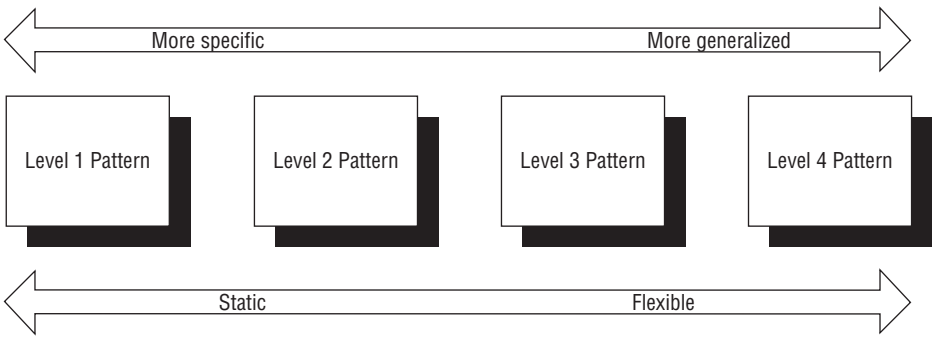


Figure 1-1 Levels of generalization

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 requirements before generalizing the model.	Basing the physical database on more generalized data models allows more use of 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.

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."

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 conceptual 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, developing 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.

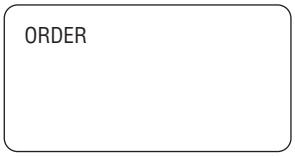


Figure 1-2 An entity

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.

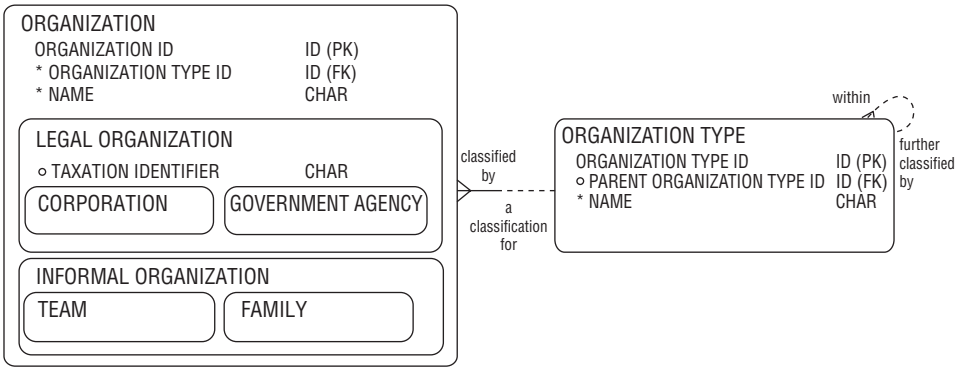


Figure 1-3 Subtypes and supertypes

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.

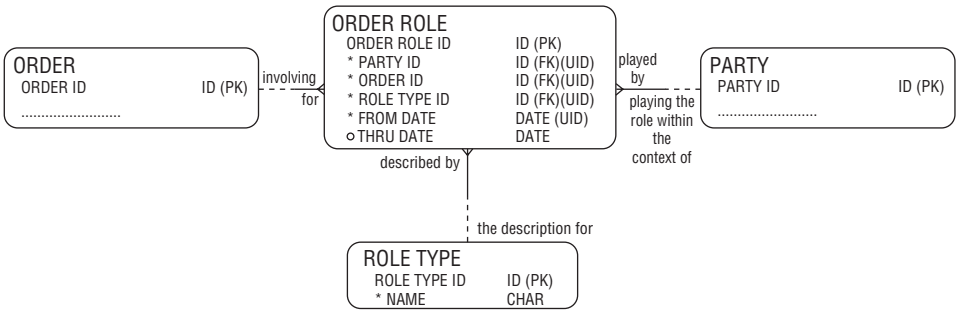


Figure 1-4 Attributes, relationships, and keys

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 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.

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.

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.

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.

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.''

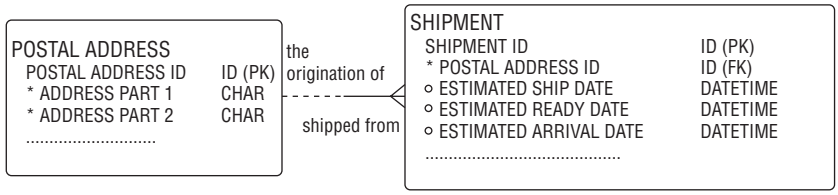


Figure 1-5 Mandatory versus optional relationships

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.

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.

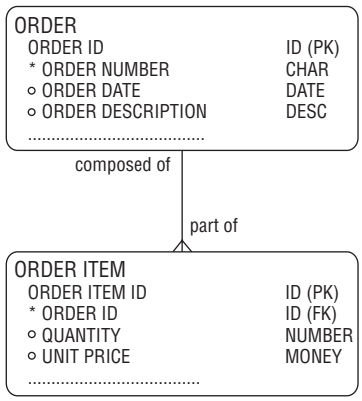


Figure 1-6 One-to-many relationship

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.

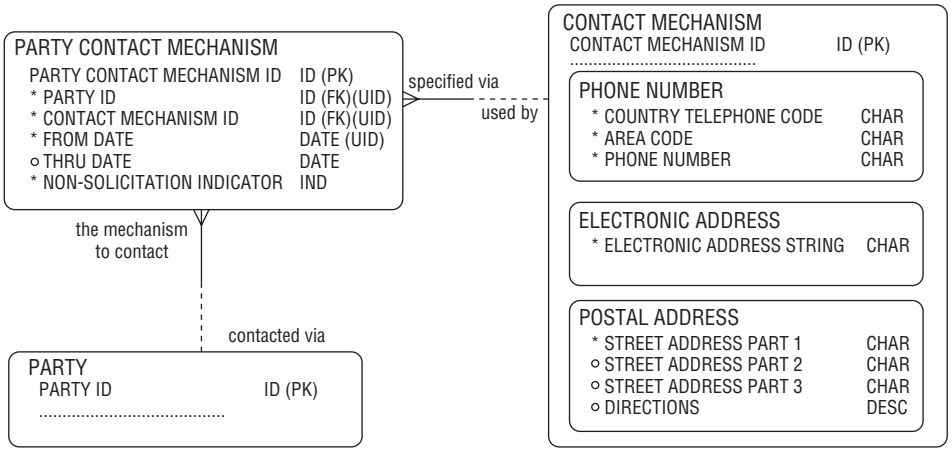


Figure 1-7 Many-to-many relationships resolved by an associative entity

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.

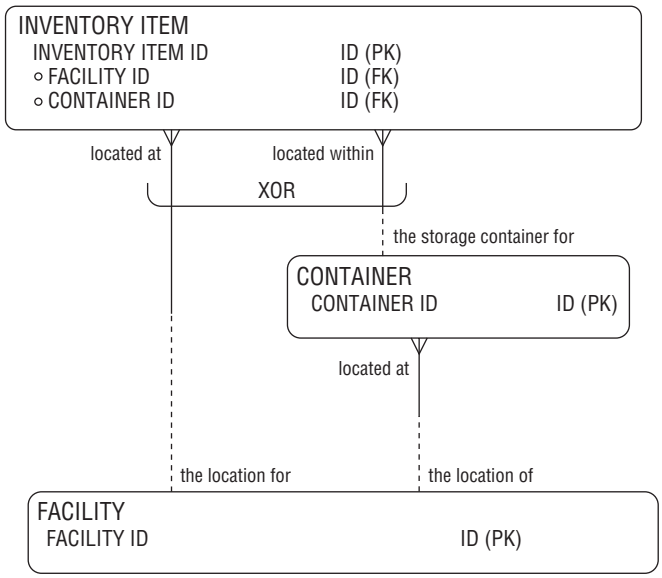


Figure 1-8 Exclusive Arcs

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.

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).''

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)

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 1-10 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.

IDEF1X (Integration Definition for Information Modeling) was introduced as a Federal Information Processing Standard in 1993 and is widely used in the

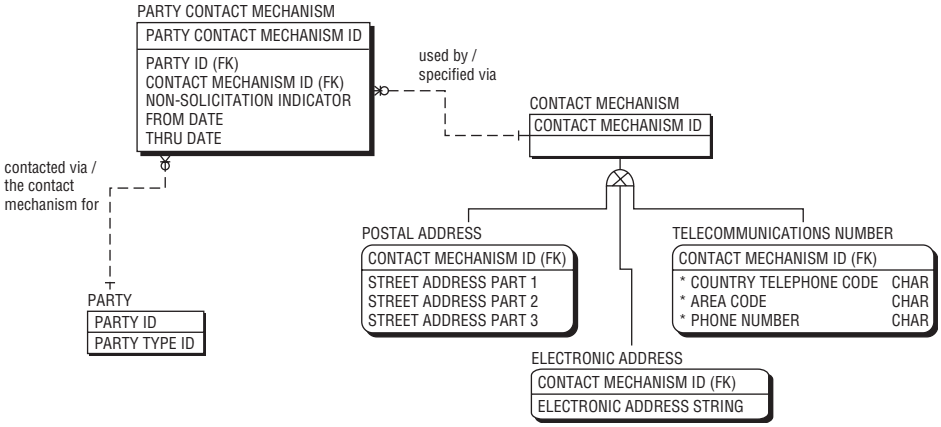


Figure 1-9 Contact Mechanism Pattern in ERwin from Computer Associates

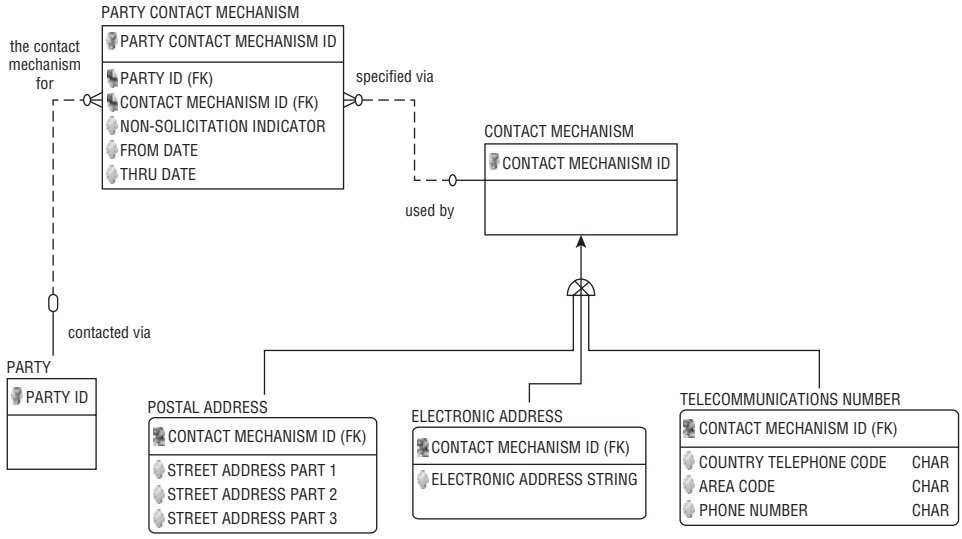


Figure 1-10 Contact Mechanism Pattern in ER/Studio from Embarcadero

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 1-12 shows the same pattern using Unified Modeling Language (UML) notation.

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.

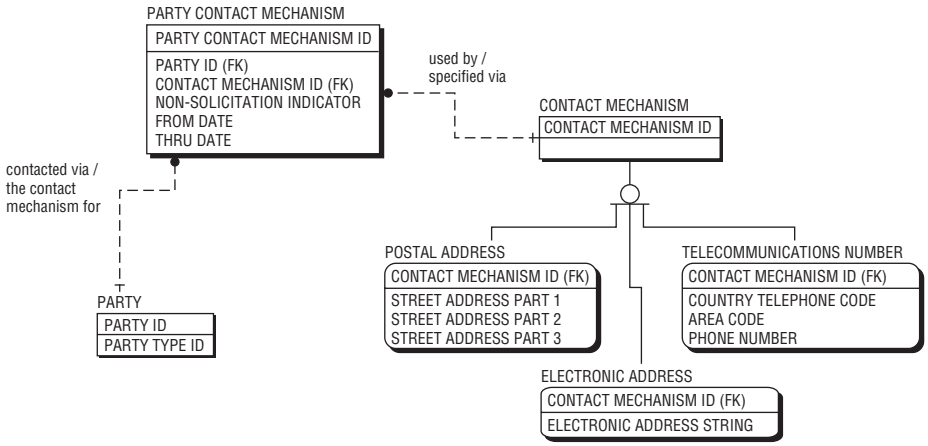


Figure 1-11 Contact Mechanism Pattern shown using IDEF1X notation

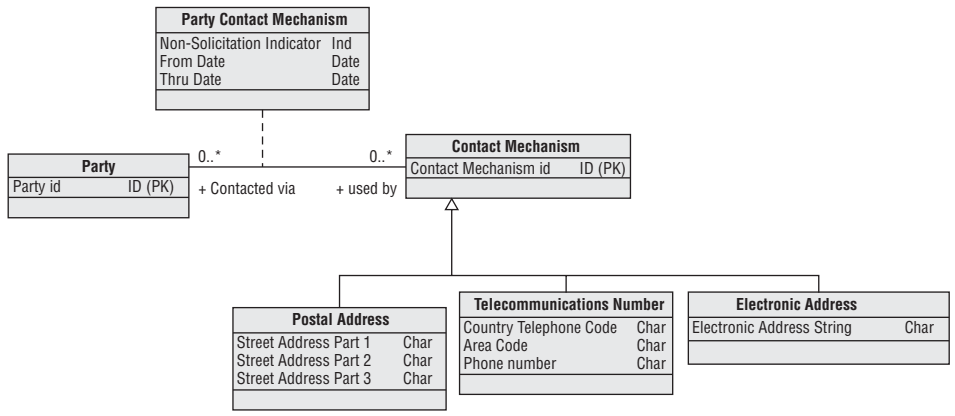


Figure 1-12 Contact Mechanism Pattern shown using UML

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 Dr. Graeme Simsion (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.