



5

Data Modeling with the Entity-Relationship Model

Chapter Objectives

- To understand the two-phase data modeling/database design process
- To understand the purpose of the data modeling process
- To understand entity-relationship (E-R) diagrams
- To be able to determine entities, attributes, and relationships
- To be able to create entity identifiers
- To be able to determine minimum and maximum cardinalities
- To understand variations of the E-R model
- To understand and be able to use ID-dependent and other weak entities
- To understand and be able to use supertype/subtype entities
- To understand and be able to use strong entity patterns
- To understand and be able to use the ID-dependent association pattern
- To understand and be able to use the ID-dependent multivalued attribute pattern
- To understand and be able to use the ID-dependent archetype/instance pattern
- To understand and be able to use the line-item pattern
- To understand and be able to use the for-use-by pattern
- To understand and be able to use recursive patterns
- To understand the iterative nature of the data modeling process
- To be able to use the data modeling process

In this chapter and the next, we consider the design of databases that arise from the development of new information systems. As you will learn, such databases are designed by analyzing requirements and creating a data model, or blueprint, of a database that will meet those requirements. The data model is then transformed into a database design.

This chapter addresses the creation of data models using the entity-relationship data model, the most popular modeling technique. This chapter consists of three major sections. First, we explain the major elements of the entity-relationship model and briefly describe several variations on that model. Next, we examine a number of patterns in forms, reports, and data models that you will encounter when data modeling. We then illustrate the data modeling process using the example of a small database at a university. Before starting, however, you need to understand the purpose of a data model.

Data modeling is a part of the systems analysis and design process. For an introduction to systems analysis and design, see Appendix B.

The Purpose of a Data Model

A **data model** is a plan, or blueprint, for a database design. By analogy, consider the construction of your dorm or apartment building. The contractor did not just buy some lumber, call for the concrete trucks, and start work. Instead, an architect constructed plans and blueprints for that building long before construction began. If, during the planning stage, it was determined that a room was too small or too large, the blueprint could be changed simply by redrawing the lines. If, however, the need for change occurs after the building is constructed, the walls, electrical system, plumbing, and so on will need to be rebuilt, at great expense and loss of time. It is easier, simpler, and faster to change the plan than it is to change a constructed building.

The same argument applies to data models and databases. Changing a relationship during the data modeling stage is just a matter of changing the diagram and related documentation. Changing a relationship after the database and applications have been constructed, however, is much more difficult. Data must be migrated to the new structure, SQL statements will need to be changed, forms and reports will need to be altered, and so forth.

The Entity-Relationship Model

Dozens of different tools and techniques for constructing data models have been defined over the years. They include the hierarchical data model, the network data model, the ANSI/SPARC data model, the entity-relationship data model, the semantic object model, and many others. Of these, the entity-relationship data model has emerged as the standard data model, and we will consider only that data model in this chapter.

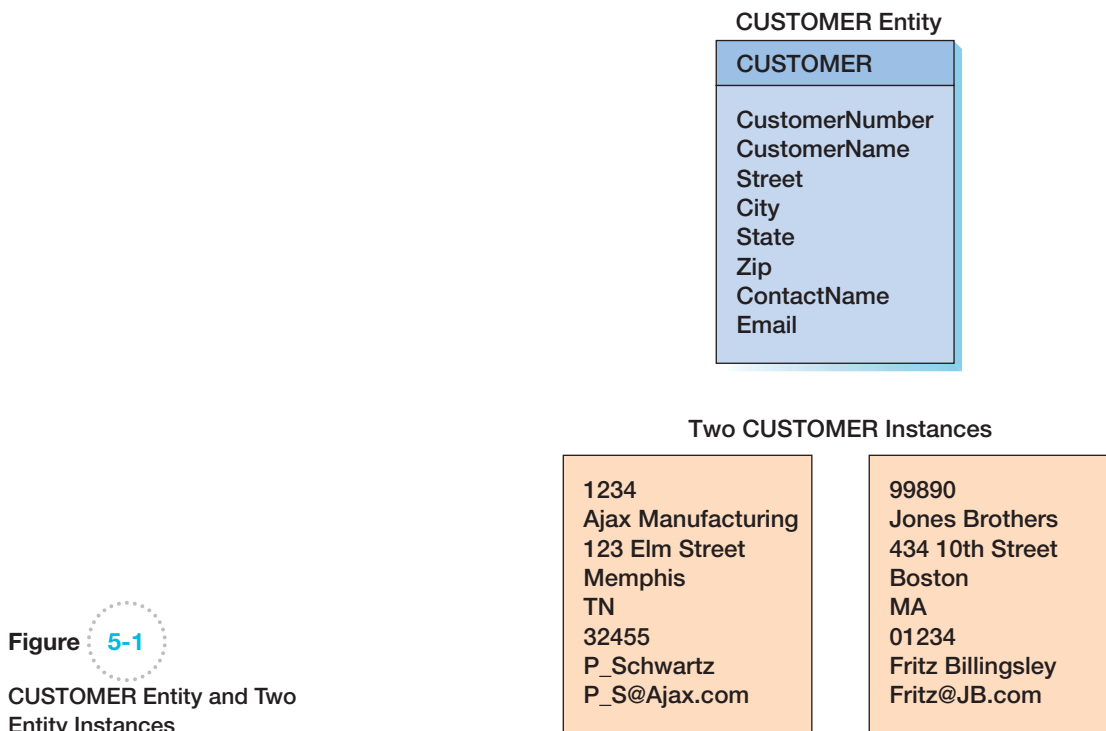
The **entity-relationship (E-R) model** was first published by Peter Chen in 1976.¹ In this paper, Chen set out the basic elements of the model. Subtypes (discussed later) were added to the E-R model to create the **extended E-R model**,² and today it is the extended E-R model that most people mean when they use the term *E-R model*. In this text, we will use the extended E-R model.

Entities

An **entity** is something that users want to track. It is something that is readily identified in the users' work environment. Example entities are EMPLOYEE Mary Lai, CUSTOMER 12345, SALES-ORDER 1000, SALESPERSON Wally Smith, and PRODUCT A4200. Entities of a given type are grouped into an **entity class**. Thus, the EMPLOYEE entity class is the collection of all EMPLOYEE entities. In this text, entity classes are shown in capital letters.

¹ Peter P. Chen, "The Entity-Relationship Model—Towards a Unified View of Data," *ACM Transactions on Database Systems*, January 1976, pp. 9–36. For information on Peter Chen see http://en.wikipedia.org/wiki/Peter_Chen, and for a copy of the article see <http://csc.lsu.edu/news/erd.pdf>.

² T. J. Teorey, D. Yang, and J. P. Fry, "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model," *ACM Computing Surveys*, June 1986, pp. 197–222.



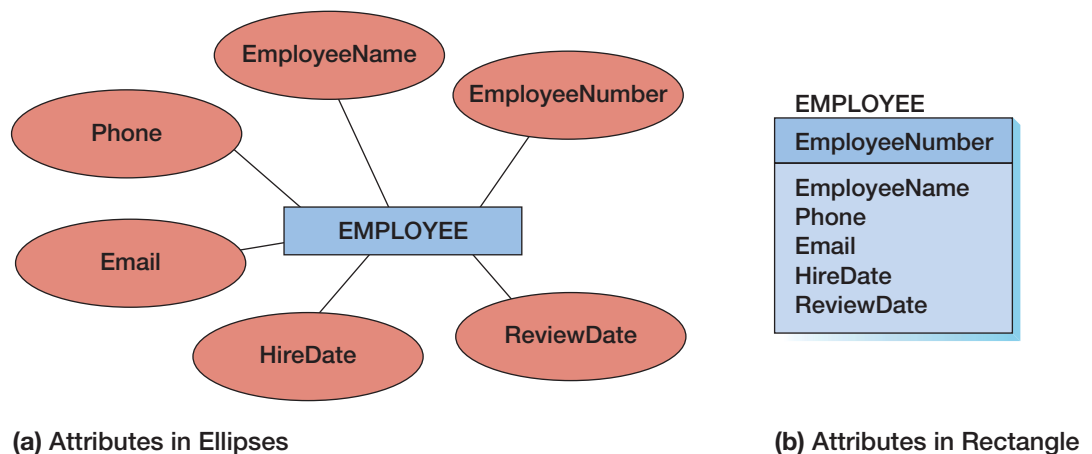
It is important to understand the differences between an entity class and an entity instance. An entity class is a collection of entities and is described by the structure of the entities in that class. An **entity instance** of an entity class is the occurrence of a particular entity, such as CUSTOMER 12345. An entity class usually has many instances of an entity. For example, the entity class CUSTOMER has many instances—one for each customer represented in the database. The CUSTOMER entity class and two of its instances are shown in Figure 5-1.

Attributes

Entities have **attributes** that describe their characteristics. Examples of attributes are EmployeeNumber, EmployeeName, Phone, and Email. In this text, attributes are written in both uppercase and lowercase letters. The E-R model assumes that all instances of a given entity class have the same attributes.

Figure 5-2 shows two different ways of displaying the attributes of an entity. Figure 5-2(a) shows attributes in ellipses that are connected to the entity. This style was used in the original E-R model, prior to the advent of data modeling software products. Figure 5-2(b) shows the rectangle style that is commonly used by data modeling software products today.

Figure 5-2
Variations on Entity Diagram
Attribute Displays



Identifiers

Entity instances have **identifiers**, which are attributes that name, or identify, entity instances. For example, EMPLOYEE instances can be identified by EmployeeNumber, SocialSecurityNumber, or EmployeeName. EMPLOYEE instances are not likely to be identified by attributes such as Salary or HireDate because these attributes are not normally used in a naming role. Similarly, customers can be identified by CustomerNumber or CustomerName, and sales orders can be identified by OrderNumber.

The identifier of an entity instance consists of one or more of the entity’s attributes. Identifiers that consist of two or more attributes are called **composite identifiers**. Examples are (AreaCode, LocalNumber), (ProjectName, TaskName), and (FirstName, LastName, DateOfHire).

BY THE WAY

Notice the correspondence of identifiers and keys. The term *identifier* is used in a data model, and the term *key* (which we have already introduced in our discussion of relational databases in Chapter 3) is used in a database design. Thus, entities have identifiers, and tables (or relations) have keys. Identifiers serve the same role for entities that keys serve for tables.

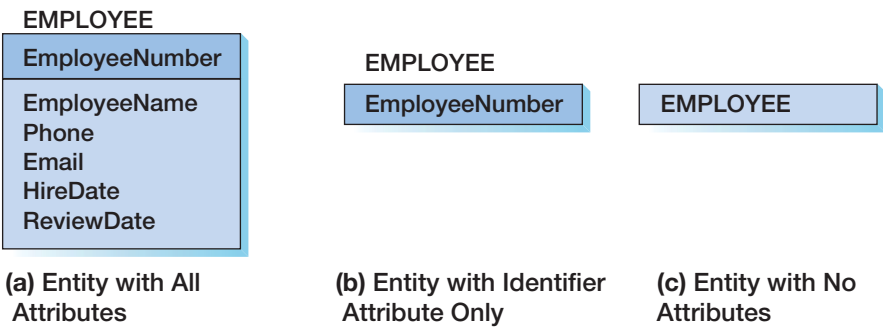
As shown in Figure 5-3, entities are portrayed in three levels of detail in a data model. As shown in Figure 5-3(a), sometimes the entity and all of its attributes are displayed. In such cases, the identifier of the attribute is shown at the top of the entity and a horizontal line is drawn after the identifier. However, in a large data model, so much detail can make the data model diagrams unwieldy. In those cases, the entity diagram is abbreviated by showing just the identifier, as in Figure 5-3(b), or by showing just the name of the entity in a rectangle, as shown in Figure 5-3(c). All three techniques are used in practice; the more abbreviated form in Figure 5-3(c) is used to show the big picture and overall entity relationships. The more detailed view in Figure 5-3(a) is frequently used during database design. Most data modeling software products have the ability to show all three displays.

Relationships

Entities can be associated with one another in **relationships**. The E-R model contains both relationship classes and relationship instances.³ **Relationship classes** are associations among entity classes, and **relationship instances** are associations among entity instances. In the original E-R model, relationships could have attributes. Today, that feature is no longer used.

Relationships are given names that describe the nature of the relationship, as shown in Figure 5-4. In Figure 5-4(a), the Qualification relationship shows which employees have which skills. In Figure 5-4(b), the Assignment relationship shows which combinations of clients, architects, and projects have been created. To avoid unnecessary complexity, in this chapter we will show the names of relationships only if there is a chance of ambiguity.

Figure 5-3
Variations on Level of Entity
Attribute Displays



³ For brevity, we sometimes drop the word *instance* when the context makes it clear that an instance rather than an entity class is involved.

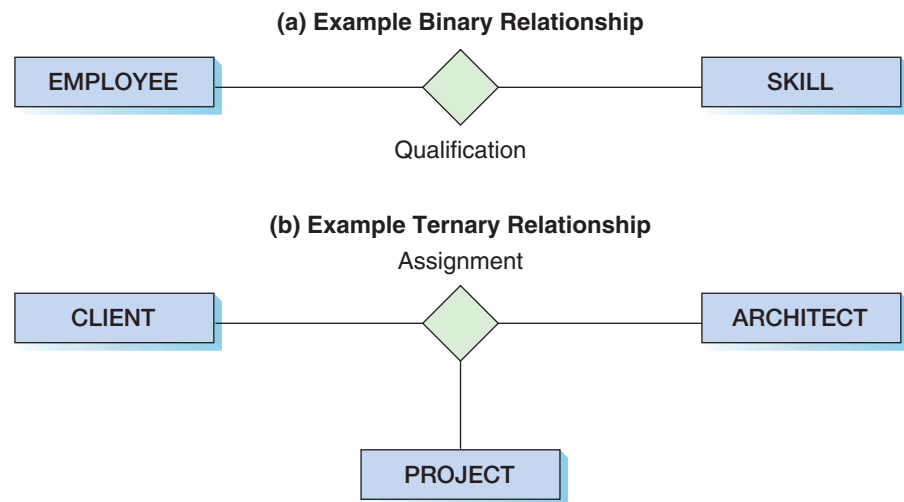


Figure 5-4
Binary Versus Ternary Relationships

BY THE WAY

Your instructor may believe that it is important to always show the name of a relationship. If so, be aware that you can name a relationship from the perspective of either of the entities or both. For example, you can name the relationship between DEPARTMENT and EMPLOYEE as Department Consists Of; or you can name it as Employee Works In; or you can name it both ways, using a slash between the two names, Department Consists Of/Employee Works In. Relationship names are a necessity when there are two different relationships between the same two entities.

A relationship class can involve two or more entity classes. The number of entity classes in the relationship is the **degree** of the relationship. In Figure 5-4(a), the Qualification relationship is of degree two because it involves two entity classes: EMPLOYEE and SKILL. In Figure 5-4(b), the Assignment relationship is of degree three because it involves three entity classes: CLIENT, ARCHITECT, and PROJECT. Relationships of degree two are referred to as **binary relationships**. Similarly, relationships of degree three are called **ternary relationships**.

When transforming a data model into a relational database design, relationships of all degrees are treated as combinations of binary relationships. The Assignment relationship in Figure 5-4(b), for example, is decomposed into three binary relationships (can you spot them?). Most of the time, this strategy is not a problem. However, some nonbinary relationships need additional work, as you will learn in Chapter 6. All data modeling software products require you to express relationships as binary relationships.

BY THE WAY

At this point, you may be wondering, “What’s the difference between an entity and a table?” So far, they seem like different terms for the same thing. *The principle difference between an entity and a table is that you can express a relationship between entities without using foreign keys.* In the E-R model, you can specify a relationship just by drawing a line connecting two entities. Because you are doing logical data modeling and not physical database design, you need not worry about primary and foreign keys, referential integrity constraints, and the like. Most data modeling products will allow you to consider such details if you choose to, but they do not require it.

This characteristic makes entities easier to work with than tables, especially early in a project when entities and relationships are fluid and uncertain. You can show relationships between entities before you even know what the identifiers are. For example, you can say that a DEPARTMENT relates to many EMPLOYEEs before you know any of the attributes of either EMPLOYEE or DEPARTMENT. This characteristic enables you to work from the general to the specific. First identify the entities, then think about relationships, and, finally, determine the attributes.

In the entity-relationship model, relationships are classified by their **cardinality**, a word that means “count.” The **maximum cardinality** is the maximum number of entity instances that can participate in a relationship instance. The **minimum cardinality** is the minimum number of entity instances that must participate in a relationship instance.

Maximum Cardinality

In Figure 5-5, the maximum cardinality is shown inside the diamond that represents the relationship. The three parts of this figure show the three basic maximum cardinalities in the E-R model.

Figure 5-5(a) shows a **one-to-one (abbreviated 1:1) relationship**. In a 1:1 relationship, an entity instance of one type is related to at most one entity instance of the other type. The *Employee_Identity* relationship in Figure 5-5(a) associates one EMPLOYEE instance with one BADGE instance. According to this diagram, no employee has more than one badge, and no badge is assigned to more than one employee.

The *Computer_Assignment* relationship in Figure 5-5(b) illustrates a **one-to-many (abbreviated 1:N) relationship**. Here, a single instance of EMPLOYEE can be associated with many instances of COMPUTER, but a COMPUTER instance is associated with just one instance of EMPLOYEE. According to this diagram, an employee can be associated with several computers, but a computer is assigned to just one employee.

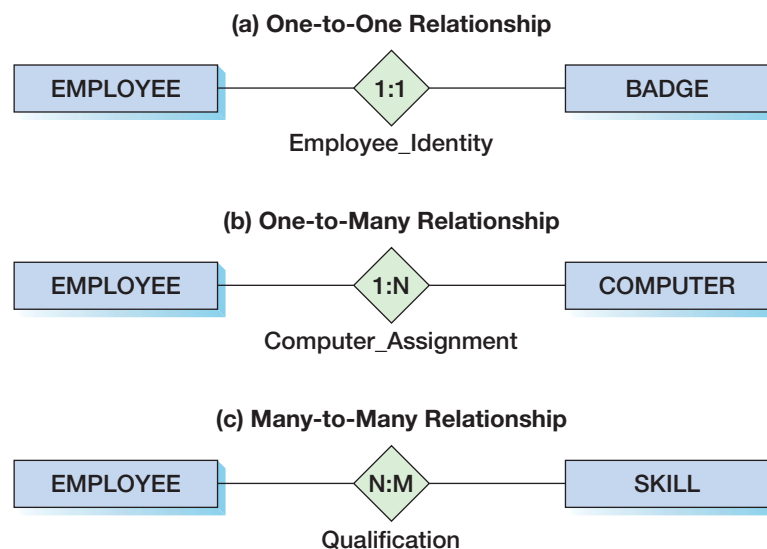
The positions of the 1 and the N are significant. The 1 is close to the line connecting EMPLOYEE, which means that the 1 refers to the EMPLOYEE side of the relationship. The N is close to the line connecting COMPUTER, which means that the N refers to the COMPUTER side of the relationship. If the 1 and the N were reversed and the relationship were written N:1, an EMPLOYEE would have one COMPUTER, and a COMPUTER would be assigned to many EMPLOYEES.

When discussing one-to-many relationships, the terms **parent** and **child** are sometimes used. The *parent* is the entity on the 1 side of the relationship, and the *child* is the entity on the many side of the relationship. Thus, in a 1:N relationship between DEPARTMENT and EMPLOYEE, DEPARTMENT is the parent and EMPLOYEE is the child (one DEPARTMENT has many EMPLOYEES).

Figure 5-5(c) shows a **many-to-many (abbreviated N:M) relationship**. According to the *Qualification* relationship, an EMPLOYEE instance can be associated with many SKILL instances, and a SKILL instance can be associated with many EMPLOYEE instances. This relationship documents that fact that an employee may have many skills, and a skill may be held by many employees.

Sometimes students wonder why we do not write many-to-many relationships as N:N or M:M. The reason is that cardinality in one direction may be different than the cardinality in the other direction. In other words, in an N:M relationship, N need not equal M. An EMPLOYEE

Figure 5-5
Three Types of Maximum
Cardinality



can have five skills, for example, but one of those skills can have three employees. Writing the relationship as N:M highlights the possibility that the cardinalities may be different.

Sometimes the maximum cardinality is an exact number. For example, for a sports team, the number of players on the roster is limited to some fixed number, say, 15. In that case, the maximum cardinality between TEAM and PLAYER would be set to 15 rather than to the more general N.

BY THE WAY

Relationships like those in Figure 5-5 are sometimes called **HAS-A relationships**. This term is used because each entity instance has a relationship to a second entity instance. An employee has a badge, and a badge has an employee. If the maximum cardinality is greater than one, then each entity has a set of other entities. An employee has a set of skills, for example, and a skill has a set of employees who have that skill.

Minimum Cardinality

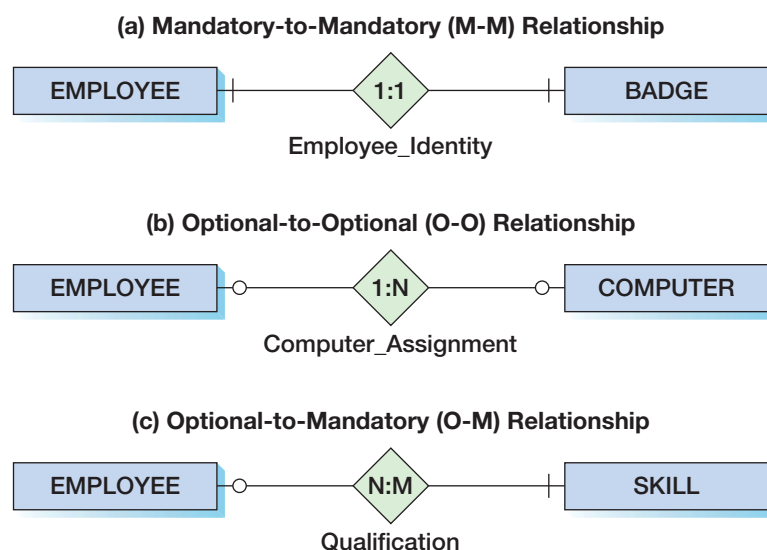
The minimum cardinality is the number of entity instances that *must* participate in a relationship. Generally, minimums are stated as either zero or one. If zero, then participation in the relationship is **optional**. If one, then at least one entity instance must participate in the relationship, which is called **mandatory** participation. In E-R diagrams, an optional relationship is represented by a small circle on the relationship line; a mandatory relationship is represented by a hash mark or line across the relationship line.

To better understand these terms, consider Figure 5-6. In the Employee_Identity relationship in Figure 5-6(a), the hash marks indicate that an EMPLOYEE is required to have a BADGE, and a BADGE must be allocated to an EMPLOYEE. Such a relationship is referred to as a **mandatory-to-mandatory (M-M) relationship**, because entities are required on both sides. The complete specification for the Employee_Identity relationship is that it is a 1:1, M-M relationship.

In Figure 5-6(b), the two small circles indicate that the Computer_Assignment relationship is an **optional-to-optional (O-O) relationship**. This means that an EMPLOYEE need not have a COMPUTER, and a COMPUTER need not be assigned to an EMPLOYEE. The Computer_Assignment relationship is thus a 1:N, O-O relationship.

Finally, in Figure 5-6(c) the combination of a circle and a hash mark indicates an **optional-to-mandatory (O-M) relationship**. Here, an EMPLOYEE must be assigned to at least one SKILL, but a SKILL may not necessarily be related to any EMPLOYEE. The complete specification for the Qualification relationship is thus an N:M, O-M relationship. The position of the circle and the hash mark are important. Because the circle is in front of EMPLOYEE, it means that the employee is optional in the relationship.

Figure 5-6
Minimum Cardinality
Examples



BY THE WAY

Sometimes when interpreting diagrams like Figure 5-6(c) students become confused about which entity is optional and which is required. An easy way to clarify this situation is to imagine that you are standing in the diamond on the relationship line. Imagine looking toward one of the entities. If you see a circle in that direction, then that entity is optional; if you see a hash mark, then that entity is required. Thus, in Figure 5-6(c), if you stand on the diamond and look toward SKILL, you see a hash mark. This means that SKILL is required in the relationship.

A fourth option, a **mandatory-to-optional (M-O) relationship** is not shown in Figure 5-6. But, if we exchange the circle and the hash mark in Figure 5-6(c), then Qualification becomes an M-O relationship. In that case, an EMPLOYEE need not have a SKILL, but a SKILL must have at least one EMPLOYEE.

As with maximum cardinalities, in rare cases the minimum cardinality is a specific number. To represent the relationship between PERSON and MARRIAGE, for example, the minimum cardinality would be 2:Optional.

Entity-Relationship Diagrams and Their Versions

The diagrams in Figures 5-5 and 5-6 are sometimes referred to as **entity-relationship (E-R) diagrams**. The original E-R model specified that such diagrams use diamonds for relationships, rectangles for entities, and connected ellipses for attributes, as shown in Figure 5-2. You may still see examples of such E-R diagrams, and it is important for you to be able to interpret them.

For two reasons, however, this original notation is seldom used today. First, there are a number of different versions of the E-R model, and these versions use different symbols. Second, data modeling software products use different techniques. For example, Computer Associates' ERwin product uses one set of symbols, and Microsoft Visio uses a second set.

Variations of the E-R Model

At least three different versions of the E-R model are in use today. One of them, the **Information Engineering (IE) model**, was developed by James Martin in 1990. This model uses crow's feet to show the many side of a relationship, and it is called the **IE Crow's Foot model**. It is easy to understand, and we will use it throughout this text. In 1993, the National Institute of Standards and Technology announced another version of the E-R model as a national standard. This version is called **Integrated Definition 1, Extended (IDEF1X)**.⁴ This standard incorporates the basic ideas of the E-R model, but uses different graphical symbols. Although this model is a national standard, it is difficult to understand and use. As a national standard, however, it is used in government, and thus it may become important to you. Therefore, the fundamentals of the IDEF1X model are described in Appendix C.

Meanwhile, to add further complication, a new object-oriented development methodology called the **Unified Modeling Language (UML)** adopted the E-R model, but introduced its own symbols while putting an object-oriented programming spin on it. UML notation is summarized in Appendix D.

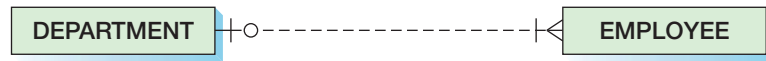
BY THE WAY

In addition to differences due to different versions of the E-R model, there also are differences due to software products. For example, two products that both implement the IE Crow's Foot model may do so in different ways. The result is a mess. When creating a data model diagram, you need to know not just the version of the E-R model you are using, but also the idiosyncrasies of the data modeling product you use.

⁴ *Integrated Definition for Information Modeling (IDEF1X)*, Federal Information Processing Standards Publication 184, 1993.



(a) Original E-R Model Version



(b) Crow's Foot Version

Figure 5-7

Two Versions of a 1:N Relationship

E-R Diagrams Using the IE Crow's Foot Model

Figure 5-7 shows two versions of a one-to-many, optional-to-mandatory relationship. Figure 5-7(a) shows the original E-R model version. Figure 5-7(b) shows the crow's foot model using common crow's foot symbols. Notice that the relationship is drawn as a dashed line. The reason for this will be explained later in this chapter. For now, notice the **crow's foot symbol** used to show the many side of the relationship.

The crow's foot model uses the notation shown in Figure 5-8 to indicate the relationship cardinality. The symbol closest to the entity shows the maximum cardinality, and the other symbol shows the minimum cardinality. A hash mark indicates one (and therefore also mandatory), a circle indicates zero (and thus optional), and the crow's foot symbol indicates many. Thus, the diagram in Figure 5-7(b) means that a DEPARTMENT has one or more EMPLOYEES (the symbol shows many and mandatory), and an EMPLOYEE belongs to zero or one DEPARTMENTS (the symbol shows one and optional).

A 1:1 relationship would be drawn in a similar manner, but the line connecting to each entity should be similar to the connection shown for the one side of the 1:N relationship in Figure 5-7(b).

Figure 5-9 shows two versions of an N:M, optional-to-mandatory relationship. Modeling N:M relationships presents some complications. According to the original E-R model diagram shown in Figure 5-9(a), an EMPLOYEE must have at least one SKILL and may have several. At the same time, although a SKILL may or may not be held by any EMPLOYEE, a SKILL may also be held by several EMPLOYEES. The crow's foot version in Figure 5-9(b) shows the N:M

Figure 5-8

Crow's Foot Notation

Symbol	Meaning
	Mandatory—One
	Mandatory—Many
	Optional—One
	Optional—Many

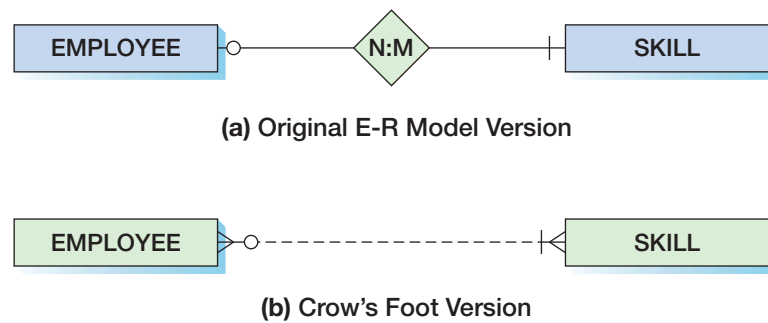


Figure 5-9

Two Versions of a N:M Relationship

maximum cardinalities using the notation in Figure 5-8. The crow's foot symbols again indicate the minimum cardinalities for the N:M relationship.

Except for Appendices C and D, for the rest of this text we will use the IE Crow's Foot model for E-R diagrams. There are no completely standard symbols for the crow's foot notation, and we explain our symbols and notation when we first use it. You can obtain various modeling products that will produce crow's foot models, and they are easily understood and related to the original E-R model. Be aware that those other products may use the oval, hash mark, crow's foot, and other symbols in slightly differently ways. Further, your instructor may have a favorite modeling tool for you to use. If that tool does not support crow's feet, you will have to adapt the data models in this text to your tool. Making these adaptations is a good learning exercise. See, for example, Project Questions 5.57 and 5.58.

BY THE WAY

A number of modeling products are available, and each will have its own idiosyncrasies. Computer Associates produces ERwin, a commercial data modeling product that handles both data modeling and database design tasks. You can download the CA ERwin Data Modeler Community Edition, which is suitable for class use, from www.ca.com/us/software-trials.aspx. You can use ERwin to produce either crow's foot or IDEF1X data models.

Although it is better at creating database designs (discussed in Chapter 6) than data models, Microsoft Visio is a possibility. A trial version is available from the Microsoft Web site at <http://office.microsoft.com/en-us/visio/default.aspx>. See Appendix F for a full discussion of using Microsoft Visio for data modeling and database designs.

Finally, Oracle is continuing development of the MySQL Workbench, as described in this book in Chapters 2 and 10B, and a free (but somewhat limited) version is available at the MySQL Web site at <http://dev.mysql.com/downloads/workbench/5.2.html>. Although, like Microsoft Visio, it is better at database designs than data models, it is a very useful tool, and the database designs it produces can be used with any DBMS, not just MySQL. See Appendix E for a full discussion of using MySQL Workbench for data modeling and database designs.

Strong Entities and Weak Entities

A **strong entity** is an entity that represents something that can exist on its own. For example, PERSON is a strong entity—we consider people to exist as individuals in their own right. Similarly, AUTOMOBILE is a strong entity. In addition to strong entities, the original version of the E-R model included the concept of a **weak entity**, which is defined as any entity whose existence depends on the presence of another entity.

ID-Dependent Entities

The E-R model includes a special type of weak entity called an **ID-dependent entity**. An ID-dependent entity is an entity whose identifier includes the identifier of another entity. Consider, for example, an entity for a student apartment in a building, as shown in Figure 5-10(a).

The identifier of such an entity is a composite (BuildingName, ApartmentNumber), where BuildingName is the identifier of the entity BUILDING. ApartmentNumber by itself is

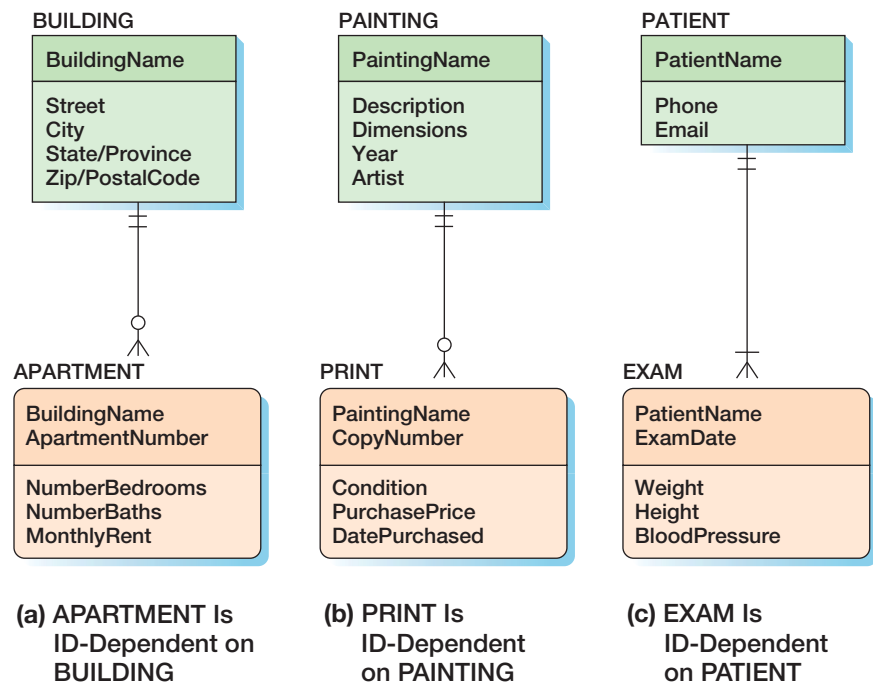


Figure 5-10
Example ID-Dependent
Entities

insufficient to tell someone where you live. If you say you live in apartment number 5, they must ask you, “In what building?”

Figure 5-10 shows three different ID-dependent entities. In addition to APARTMENT, the entity PRINT in Figure 5-10(b) is ID-dependent on PAINTING, and the entity EXAM in Figure 5-10(c) is ID-dependent on PATIENT.

In each of these cases, the ID-dependent entity cannot exist unless the parent (the entity on which it depends) also exists. Thus, the minimum cardinality from the ID-dependent entity to the parent is always one.

However, whether the parent is required to have an ID-dependent entity depends on the application requirements. In Figure 5-10, both APARTMENT and PRINT are optional, but EXAM is required. These restrictions arise from the nature of the application and not from any logical requirement.

As shown in Figure 5-10, in our E-R models we use an entity with rounded corners to represent the ID-dependent entity. We also use a solid line to represent the relationship between the ID-dependent entity and its parent. This type of a relationship is called an **identifying relationship**. A relationship drawn with a dashed line (refer to Figure 5-7) is used between strong entities and is called a **nonidentifying relationship** because there are no ID-dependent entities in the relationship.

ID-dependent entities pose restrictions on the processing of the database that is constructed from them. Namely, the row that represents the parent entity must be created before any ID-dependent child row can be created. Further, when a parent row is deleted, all child rows must be deleted as well.

ID-dependent entities are common. Another example is the entity VERSION in the relationship between PRODUCT and VERSION, where PRODUCT is a software product and VERSION is a release of that software product. The identifier of PRODUCT is ProductName, and the identifier of VERSION is (ProductName, ReleaseNumber). Yet another example is EDITION in the relationship between TEXTBOOK and EDITION. The identifier of TEXTBOOK is Title, and the identifier of EDITION is (Title, EditionNumber).

Non-ID-Dependent Weak Entities

All ID-dependent entities are weak entities. But, according to the original E-R model, some entities that are weak are not ID-dependent. Consider the AUTO_MODEL and VEHICLE entity classes in the database of a car manufacturer, such as Ford or Honda, as shown in Figure 5-11.

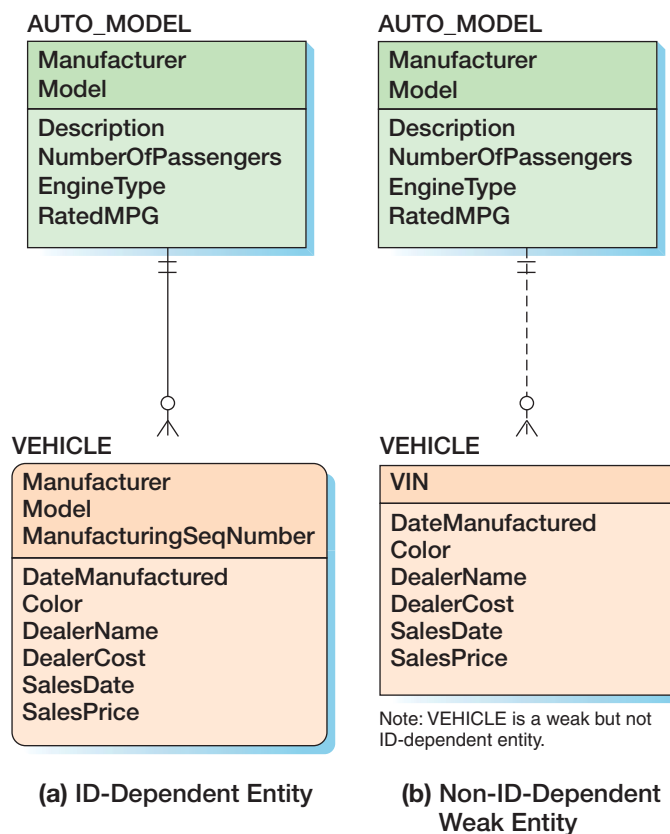


Figure 5-11
Weak Entity Example

In Figure 5-11(a), each VEHICLE is assigned a sequential number as it is manufactured. So, for the manufacturer's "Super SUV" AUTO_MODEL, the first VEHICLE manufactured gets a ManufacturingSeqNumber of 1, the next gets a ManufacturingSeqNumber of 2, and so on. This is clearly an ID-dependent relationship because ManufacturingSeqNumber is based on the Manufacturer and Model.

Now let's assign VEHICLE an identifier that is independent of the Manufacturer and Model. We will use a VIN (vehicle identification number), as shown in Figure 5-11(b). Now, the VEHICLE has a unique identifier of its own and does not need to be identified by its relation to AUTO_MODEL.

This is an interesting situation. VEHICLE has an identity of its own and therefore is not ID-dependent. Yet the VEHICLE is an AUTO_MODEL, and if that particular AUTO_MODEL did not exist, the VEHICLE itself would never have existed. Therefore, VEHICLE is now a *weak but non-ID-dependent entity*.

Consider *your* car—let's say it is a Ford Mustang just for the sake of this discussion. Your individual Mustang is a VEHICLE, and it exists as a physical object and is identified by the VIN that is required for each licensed automobile. It is *not* ID-dependent on AUTO_MODEL, which in this case is Ford Mustang, for its identity. However, if the Ford Mustang had never been created as an AUTO_MODEL—a logical concept that was first designed on paper—your car would never have been built because *no* Ford Mustangs would ever have been built! Therefore, your physical individual VEHICLE would not exist without a logical AUTO_MODEL of Ford Mustang, and in a data model (which *is* what we're talking about) a VEHICLE cannot exist without a related AUTO_MODEL. This makes VEHICLE a weak but non-ID-dependent entity. Most data modeling tools cannot model non-ID-dependent entities. So, to indicate such situations, we will use a nonidentifying relationship with a note added to the data model indicating that the entity is weak, as shown in Figure 5-11(b).

The Ambiguity of the Weak Entity

Unfortunately, an ambiguity is hidden in the definition of *weak entity*, and this ambiguity is interpreted differently by different database designers (as well as different textbook authors). The ambiguity is that in a strict sense, if a weak entity is defined as any entity whose presence

Figure 5-12
Summary of ID-Dependent
and Weak Entities

- A weak entity is an entity whose existence depends on another entity.
- An ID-dependent entity is a weak entity whose identifier includes the identifier of another entity.
- Identifying relationships are used to represent ID-dependent entities.
- Some entities are weak, but not ID-dependent. Using data modeling tools, they are shown with nonidentifying relationships, with separate documentation indicating they are weak.

in the database depends on another entity, then any entity that participates in a relationship having a minimum cardinality of one to a second entity is a weak entity. Thus, in an academic database, if a STUDENT must have an ADVISER, then STUDENT is a weak entity, because a STUDENT entity cannot be stored without an ADVISER.

This interpretation seems too broad to some people. A STUDENT is not physically dependent on an ADVISER (unlike an APARTMENT to a BUILDING), and a STUDENT is not logically dependent on an ADVISER (despite how it might appear to either the student or the adviser), and, therefore, STUDENT should be considered a strong entity.

To avoid such situations, some people interpret the definition of weak entity more narrowly. They say that to be a weak entity an entity must logically depend on another entity. According to this definition, APARTMENT is a weak entity, but STUDENT is not. An APARTMENT cannot exist without a BUILDING in which it is located. However, a STUDENT can logically exist without an ADVISER, even if a business rule requires it.

We agree with the latter approach. Characteristics of ID-dependent and non-ID-dependent weak entities, as used in this book, are summarized in Figure 5-12.

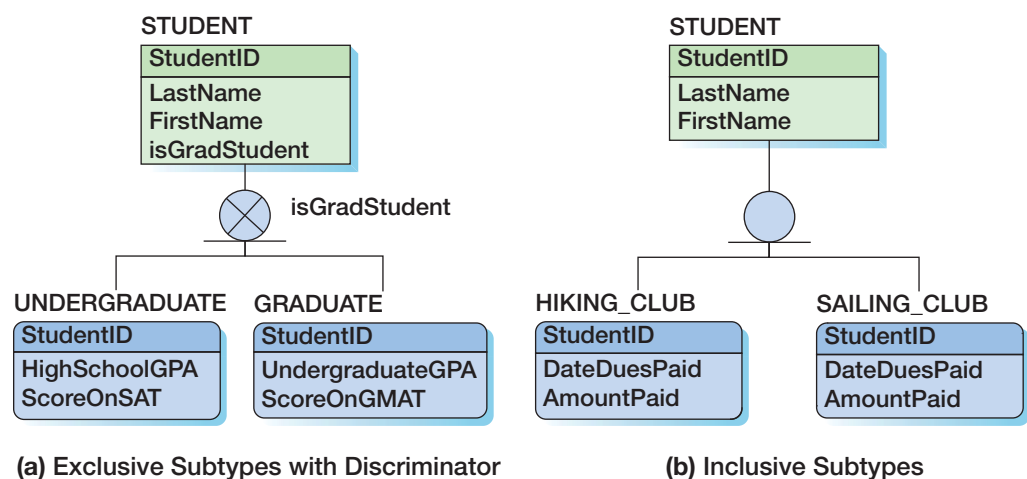
Subtype Entities

The extended E-R model introduced the concept of *subtypes*. A **subtype** entity is a special case of another entity called its **supertype**. Students, for example, may be classified as undergraduate or graduate students. In this case, STUDENT is the supertype, and UNDERGRADUATE and GRADUATE are the subtypes.

Alternatively, a student could be classified as a freshman, sophomore, junior, or senior. In that case, STUDENT is the supertype, and FRESHMAN, SOPHOMORE, JUNIOR, and SENIOR are the subtypes.

As illustrated in Figure 5-13, in our E-R models we use a circle with a line under it as a subtype symbol to indicate a supertype-subtype relationship. Think of this as a symbol for an optional (the circle), 1:1 (the line) relationship. In addition, we use a solid line to represent an ID-dependent subtype entity because each subtype is ID-dependent on the supertype. Also note that none of the line end symbols shown in Figure 5-8 are used on the connecting lines.

Figure 5-13
Example Subtype Entities



In some cases, an attribute of the supertype indicates which of the subtypes is appropriate for a given instance. An attribute that determines which subtype is appropriate is called a **discriminator**. In Figure 5-13(a), the attribute named `isGradStudent` (which has only the values Yes and No) is the discriminator. In our E-R diagrams, the discriminator is shown next to the subtype symbol, as illustrated in Figure 5-13(a). Not all supertypes have a discriminator. Where a supertype does not have a discriminator, application code must be written to create the appropriate subtype.

Subtypes can be exclusive or inclusive. With **exclusive subtypes**, a supertype instance is related to at most one subtype. With **inclusive subtypes**, a supertype instance can relate to one or more subtypes. In Figure 5-13(a), the *X* in the circle means that the `UNDERGRADUATE` and `GRADUATE` subtypes are exclusive. Thus, a `STUDENT` can be either an `UNDERGRADUATE` or a `GRADUATE`, but not both. Figure 5-13(b) shows that a `STUDENT` can join either the `HIKING_CLUB` or the `SAILING_CLUB`, or both. These subtypes are inclusive (note there is no *X* in the circle). Because a supertype may relate to more than one subtype, inclusive subtypes do not have a discriminator.

The most important (some would say the only) reason for creating subtypes in a data model is to avoid value-inappropriate nulls. Undergraduate students take the SAT exam and report that score, whereas graduate students take the GMAT and report their score on that exam. Thus, the SAT score would be NULL in all `STUDENT` entities for graduates, and the GMAT score would be NULL for all undergraduates. Such null values can be avoided by creating subtypes.

BY THE WAY

The relationships that connect supertypes and subtypes are called **IS-A relationships** because a subtype is the same entity as the supertype. Because this is so, the identifier of a supertype and all its subtypes must be the same; they all represent different aspects of the same entity. Contrast this with HAS-A relationships, in which an entity has a relationship to another entity, but the identifiers of the two entities are different.

The elements of the entity-relationship model and their IE Crow's Foot representation are summarized in Figure 5-14. The identifier and attributes are shown only in the first example. Note that for 1:1 and 1:N nonidentifying relationships a relationship to a parent entity may be optional. For identifying relationships, the parent is always required.

Patterns in Forms, Reports, and E-R Models

A data model is a representation of how users view their world. Unfortunately, you cannot walk up to most computer users and ask questions like, "What is the maximum cardinality between the `EMPLOYEE` and `SKILL` entities?" Few users would have any idea of what you mean. Instead, you must infer the data model indirectly from user documents and from users' conversations and behavior.

One of the best ways to infer a data model is to study the users' forms and reports. From such documents, you can learn about entities and their relationships. In fact, the structure of forms and reports determines the structure of the data model, and the structure of the data model determines the structure of forms and reports. This means that you can examine a form or report and determine the entities and relationships that underlie it.

You can also use forms and reports to validate the data model. Rather than showing the data model to the users for feedback, an alternative is to construct a form or report that reflects the structure of the data model and obtain user feedback on that form or report. For example, if you want to know if an `ORDER` has one or many `SALESPERSON`s, you can show the users a form that has a space for entering just one salesperson's name. If the user asks, "Where do I put the name of the second salesperson?" then you know that orders have at least two and possibly many salespeople. Sometimes, when no appropriate form or report exists, teams create a prototype form or report for the users to evaluate.

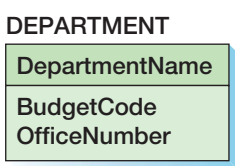
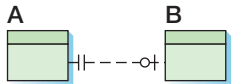
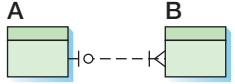
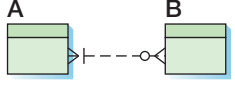
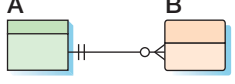
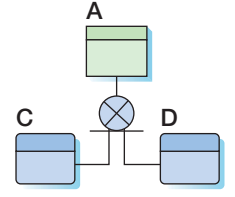
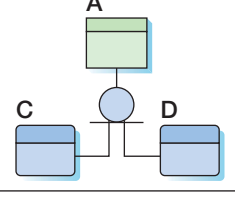
	DEPARTMENT entity; DepartmentName is identifier; BudgetCode and OfficeNumber are attributes.
	1:1, nonidentifying relationship. A relates to zero or one B; B relates to exactly one A. Identifier and attributes not shown.
	1:N, nonidentifying relationship. A relates to one or many Bs; B relates to zero or one A. Identifier and attributes not shown.
	Many-to-many, nonidentifying relationship. A relates to zero or more Bs; B relates to one or more As.
	1:N identifying relationship. A relates to zero, one, or many Bs. B relates to exactly one A. Identifier and attributes not shown. For identifying relationships, the child must always relate to exactly one parent. The parent may relate to zero, one, many, or a combination of these minimum cardinalities.
	A is supertype, C and D are exclusive subtypes. Discriminator not shown. Identifier and attributes not shown.
	A is supertype, C and D are inclusive subtypes. Identifier and attributes not shown.

Figure 5-14
IE Crow's Foot Symbol
Summary

All of this means that you must understand how the structure of forms and reports determines the structure of the data model, and the reverse. Fortunately, many forms and reports fall into common patterns. If you learn how to analyze these patterns, you will be well on your way to understanding the logical relationship between forms and reports and the data model. Accordingly, in the next sections we will discuss the most common patterns in detail.

Strong Entity Patterns

Three relationships are possible between two strong entities: 1:1, 1:N, and N:M. When modeling such relationships, you must determine both the maximum and minimum cardinality. The maximum cardinality often can be determined from forms and reports. In most cases, to determine the minimum cardinality you will have to ask the users.

1.1 Strong Entity Relationships

Figure 5-15 shows a data entry form and a report that indicate a one-to-one relationship between the entities CLUB_MEMBER and LOCKER. The MEMBER_LOCKER form in Figure 5-15(a) shows data for an athletic club member, and it lists just one locker for that member. This form indicates that a CLUB_MEMBER has at most one locker. The report in Figure 5-15(b) shows the lockers in the club and indicates the member who has been allocated that locker. Each locker is assigned to one club member.

The form and report in Figure 5-15 thus suggest that a CLUB_MEMBER has one LOCKER, and a LOCKER is assigned to one CLUB_MEMBER. Hence, the relationship

(a) Club Membership Data Entry Form

Member Number	1000
Member Name	Jones
Phone	123-456-7777
Email	Jones@somewhere.com
Locker Number	2100
Locker Room	Mens

Record: 1 of 4 No Filter Search

(b) Club Locker Report

CLUB LOCKERS				
Member Number	Member Name	Locker Number	Locker Room	Locker Size
1000	Jones	2100	Mens	Med
2000	Abernathy	2200	Womens	Large
3000	Wu	2115	Mens	Large
4000	Lai	2217	Womens	Small

Figure 5-15

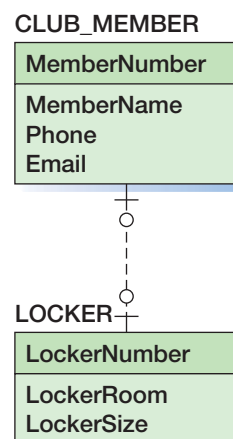
Form and Report Indicating
a 1:1 Relationship

between them is 1:1. To model that relationship, we draw a nonidentifying relationship (meaning the relationship is strong and not ID-dependent) between the two entities, as shown in Figure 5-16. We then set the maximum cardinality to 1:1. You can tell that this is a nonidentifying relationship, because the relationship line is dashed. Also, the absence of a crow's foot indicates that the relationship is 1:1.

Regarding minimum cardinality, every club member shown in the form has a locker, and every locker shown in the report is assigned to a club member, so it appears that the relationship is mandatory to mandatory. However, this form and report are just instances; they may not show every possibility. If the club allows social, nonathletic memberships, then not every club member will have a locker. Furthermore, it is unlikely that every locker is occupied; there must be some lockers that are unused and nonallocated. Accordingly, Figure 5-16 shows this relationship as optional to optional, as indicated by the small circles on the relationship lines.

Figure 5-16

Data Model for the 1:1
Relationship in Figure 5-15



BY THE WAY

How do you recognize strong entities? You can use two major tests. First, does the entity have an identifier of its own? If it shares a part of its identifier with another entity, then it is an ID-dependent entity, and is therefore weak. Second, does the entity seem to be logically different from and separate from the other entities? Does it stand alone, or is it part of something else? In this case, a CLUB_MEMBER and a LOCKER are two very different, separate things; they are not part of each other or of something else. Hence, they are strong.

Note also that a form or report shows only one side of a relationship. Given entities A and B, a form can show the relationship from A to B, but it cannot show the relationship from B to A at the same time. To learn the cardinality from B to A, you must examine a second form or report, ask the users, or take some other action.

Finally, it is seldom possible to infer minimum cardinality from a form or report. Generally, you must ask the users.

1:N Strong Entity Relationships

Figure 5-17 shows a form that lists the departments within a company. The company has many departments, so the maximum cardinality from COMPANY to DEPARTMENT is N. But what about the opposite direction? To determine if a department relates to one or N companies, we need to examine a form or report that shows the relationship from a department to a company. However, assume that no such form or report exists. Also assume that the users never view company data from the perspective of a department. We cannot ignore the issue, because we need to know whether the relationship is 1:N or N:M.

In such a case, we must ask the users or at least make a determination by thinking about the nature of the business setting. Can a department belong to more than one company? Is a department shared among companies? Because this seems unlikely, we can reasonably assume that DEPARTMENT relates to just one COMPANY. Thus, we conclude the relationship is 1:N. Figure 5-18 shows the resulting data model. Note that the many side of the relationship is indicated by the crow's foot next to DEPARTMENT.

Considering minimum cardinality, we do not know if a COMPANY must have a DEPARTMENT or if a DEPARTMENT must have a COMPANY. We will definitely need to ask the users. Figure 5-18 depicts the situation in which a DEPARTMENT must have a COMPANY, but a COMPANY need not have any DEPARTMENTS.

N:M Strong Entity Relationships

Figure 5-19(a) shows a form with data about a supplier and the parts it is prepared to supply. Figure 5-19(b) shows a report that summarizes parts and lists the companies that can supply those parts. In both cases, the relationship is many: A SUPPLIER supplies many PARTs, and a PART is supplied by many SUPPLIERS. Thus, the relationship is N:M.

Figure 5-17

Form Indicating a 1:N Relationship

The screenshot shows a form titled 'Company Departments'. It has two main sections: 'Company Information' and 'Departments'. The 'Company Information' section contains fields for 'Company Name' (Ajax Manufacturing) and 'City' (Sydney). The 'Departments' section contains a table with columns 'Department Name', 'Budget Code', and 'Mail Stop'. The table lists four departments: Accounting, Production, Information Systems, and Sales. The 'Record' indicator at the bottom shows '1 of 4', indicating that there are four records in the table, which represents the 'N' side of the 1:N relationship.

Department Name	Budget Code	Mail Stop
Accounting	A-100	MS-100
Production	P-100	MS-400
Information Systems	IS-200	MS-417
Sales	S-1400	MS-500

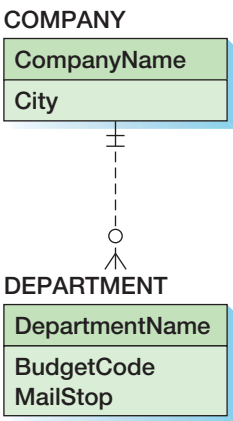


Figure 5-18

Data Model for the 1:N Relationship in Figure 5-17

Figure 5-20 shows a data model that extends the data model in Figure 5-18 to include this new relationship. A supplier is a company, so we show the supplier entity as a COMPANY.

Because not all companies are suppliers, the relationship from COMPANY to PART must be optional. However, every part must be supplied from somewhere, so the relationship from PART to COMPANY is mandatory.

In summary, the three types of strong entity relationships are 1:1, 1:N, and N:M. You can infer the maximum cardinality in one direction from a form or report. You must examine a second form or report to determine the maximum cardinality in the other direction. If no form

Figure 5-19

Form and Report Indicating an N:M Relationship

(a) SUPPLIERS Form

Suppliers

Company Name: Forrest Supplies

City: Denver

Country: US

Volume (USD): \$177,990.00

PartNumber	PartName	SalesPrice	ReOrderQuantity	QuantityOnHand
1000	Cedar Shakes	\$22.00	100	200
2000	Garage Heater	\$1,750.00	3	4
3000	Utility Cabinet	\$55.00	7	3

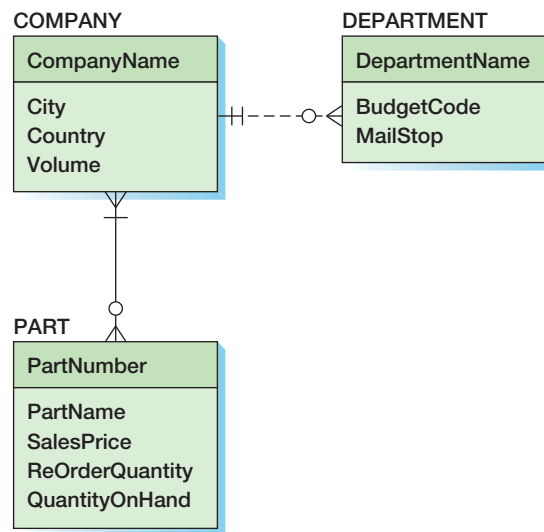
Record: 14 1 of 3 No Filter Search

(b) PART Report

PART							
Part Number	Part Name	Sales Price	ROQ	QOH	Company Name	City	Country
1000	Cedar Shakes	\$22.00	100	200	Bristol Systems	Manchester	England
					ERS Systems	Vancouver	Canada
					Forrest Supplies	Denver	US
2000	Garage Heater	\$1,750.00	3	4	Bristol Systems	Manchester	England
					ERS Systems	Vancouver	Canada
					Forrest Supplies	Denver	US
3000	Utility Cabinet	\$55.00	7	3	Kyoto Importers	Kyoto	Japan
					Ajax Manufacturing	Sydney	Australia
					Forrest Supplies	Denver	US

Figure 5-20

Data Model for the N:M Relationship in Figure 5-19



or report that shows the relationship is available, you must ask the users. Generally, it is not possible to determine minimum cardinality from forms and reports.

ID-Dependent Relationships

Three principle patterns use ID-dependent entities: multivalued attribute, archetype/instance (also called *version/instance*), and association. Because the association pattern is often confused with the N:M strong entity relationships just discussed, we will look at that pattern first.

The Association Pattern

An **association pattern** is subtly and confusingly similar to an N:M strong relationship. To see why, examine the report in Figure 5-21 and compare it with the report in Figure 5-19(b).

What is the difference? If you look closely, you'll see that the only difference is that the report in Figure 5-21 contains Price, which is the price quotation for a part from a particular supplier. The first line of this report indicates that the part Cedar Shakes is supplied by Bristol Systems for \$14.00.

Price is neither an attribute of COMPANY nor is it an attribute of PART. It is an attribute of the combination of a COMPANY with a PART. Figure 5-22 shows the appropriate data model for such a case.

Here, a third entity, QUOTATION, has been created to hold the Price attribute. The identifier of QUOTATION is the combination of PartNumber and CompanyName. Note that PartNumber is the identifier of PART, and CompanyName is the identifier of COMPANY. Hence, QUOTATION is ID-dependent on *both* PART and COMPANY.

Figure 5-21

Report Indicating an Association Pattern

PART QUOTATIONS								
PartNumber	PartName	SalesPrice	ROQ	QOH	CompanyName	City	Country	Price
1000	Cedar Shakes	\$22.00	100	200	Bristol Systems	Manchester	England	\$14.00
					ERS Systems	Vancouver	Canada	\$12.50
					Forrest Supplies	Denver	US	\$15.50
2000	Garage Heater	\$1,750.00	3	4	Bristol Systems	Manchester	England	\$950.00
					ERS Systems	Vancouver	Canada	\$875.00
					Forrest Supplies	Denver	US	\$915.00
					Kyoto Importers	Kyoto	Japan	\$1,100.00
3000	Utility Cabinet	\$55.00	7	3	Ajax Manufacturing	Sydney	Australia	\$37.50
					Forrest Supplies	Denver	US	\$42.50

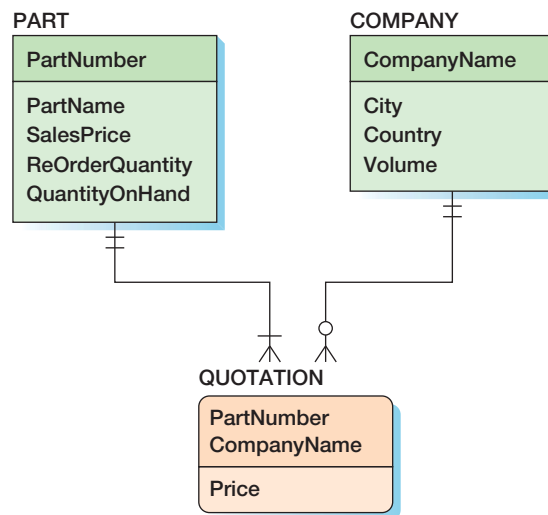


Figure 5-22
Association Pattern Data
Model for the Report in
Figure 5-21

In Figure 5-22, then, the relationships between PART and QUOTATION and between COMPANY and QUOTATION are both identifying. This fact is shown in Figure 5-22 by the solid, nondashed line that represents these relationships.

As with all identifying relationships, the parent entities are required. Thus, the minimum cardinality from QUOTATION to PART is one, and the minimum cardinality from QUOTATION to COMPANY also is one. The minimum cardinality in the opposite direction is determined by business requirements. Here, a PART must have a QUOTATION, but a COMPANY need not have a QUOTATION.

BY THE WAY

Consider the differences between the data models in Figure 5-20 and Figure 5-22. The only difference between the two is that in the latter the relationship between COMPANY and PART has an attribute, Price. Remember this example whenever you model an N:M relationship. Is there a missing attribute that pertains to the combination and not just to one of the entities? If so, you are dealing with an association, ID-dependent pattern and not an N:M, strong entity pattern.

Associations can occur among more than two entity types. Figure 5-23, for example, shows a data model for the assignment of a particular client to a particular architect for a particular

Figure 5-23
Association Pattern Data
Model for the Ternary
Relationship in Figure 5-4

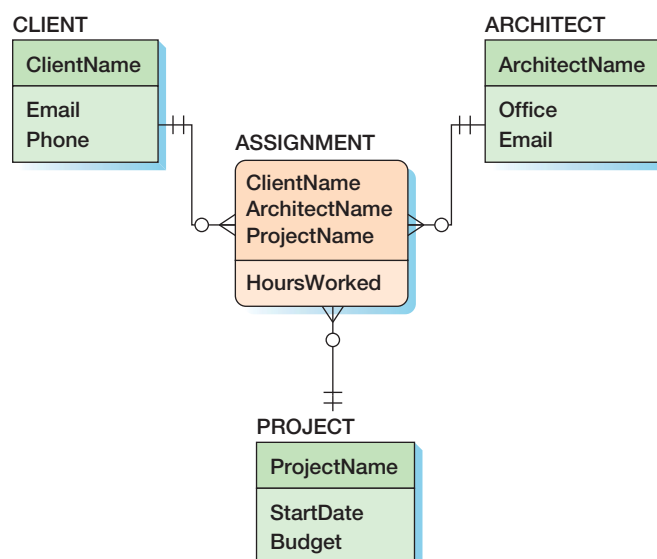


Figure 5-24

Data Entry Form with a Multivalued Attribute

project. The attribute of the assignment is HoursWorked. This data model shows how the ternary relationship in Figure 5-4(b) can be modeled as a combination of three binary relationships.

The Multivalued Attribute Pattern

In the E-R model as used today,⁵ attributes must have a single value. If the COMPANY entity has PhoneNumber and Contact attributes, then a company can have at most one value for phone number and at most one value for contact.

In practice, however, companies can have more than one phone number and one contact. Consider, for example, the data entry form in Figure 5-24. This particular company has three phone numbers; other companies might have one or two or four, or whatever. We need to create a data model that allows companies to have multiple phones, and placing the attribute PhoneNumber in COMPANY will not do it.

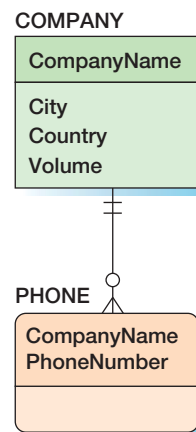
Figure 5-25 shows the solution. Instead of including PhoneNumber as an attribute of COMPANY, we create an ID-dependent entity, PHONE, that contains the attribute PhoneNumber. The relationship from COMPANY to PHONE is 1:N, so a company can have multiple phone numbers. Because PHONE is an ID-dependent entity, its identifier includes both CompanyName and PhoneNumber.

We can extend this strategy for as many multivalued attributes as necessary. The COMPANY data entry form in Figure 5-26 has multivalued Phone and multivalued Contact attributes. In this case, we just create a separate ID-dependent entity for each multivalued attribute, as shown in Figure 5-27.

In Figure 5-27, PhoneNumber and Contact are independent. PhoneNumber is the phone number of the company and not necessarily the phone number of a contact. If PhoneNumber

Figure 5-25

Data Model for the Form in Figure 5-24



⁵ The original E-R model allowed for multivalued attributes. Over time, that feature has been ignored, and today most people assume that the E-R model requires single-valued attributes. We will do so in this text.

Figure 5-26

Data Entry Form with
Separate Multivalued
Attributes

is not a general company phone number, but rather the phone number of a particular person at that company, then the data entry form would appear as in Figure 5-28. Here, for example, Alfred has one phone number and Jackson has another.

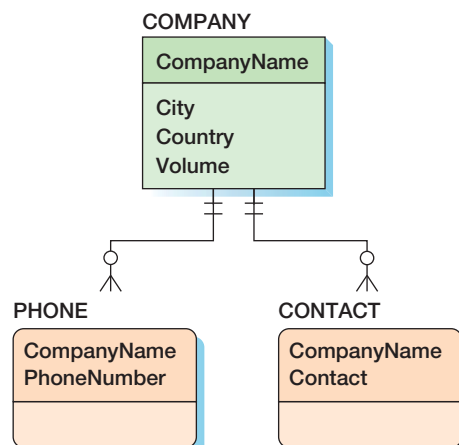
In this case, the attributes `PhoneNumber` and `Contact` belong together. Accordingly, we place them into a single ID-dependent entity, as shown in Figure 5-29. Notice that the identifier of `PHONE_CONTACT` is `Contact` and `CompanyName`. This arrangement means that a given `Contact` name can appear only once per company. Contacts can share phone numbers, however, as shown for employees Lynda and Swee. If the identifier of `PHONE_CONTACT` was `PhoneNumber` and `CompanyName`, then a phone number could occur only once per company, but contacts could have multiple numbers. Work through these examples to ensure that you understand them.

In all of these examples, the child requires a parent, which is always the case for ID-dependent entities. The parent may or may not require a child, depending on the application. A `COMPANY` may or may not require a `PHONE` or a `CONTACT`. You must ask the users to determine whether the ID-dependent entity is required.

Multivalued attributes are common, and you need to be able to model them effectively. Review the models in Figures 5-25, 5-27, and 5-29 and be certain that you understand their differences and what those differences imply.

Figure 5-27

Data Model for the Form
in Figure 5-26



PhoneNumber	Contact
1.100.334.8000	Alfred
1.100.444.9988	Jackson
800-123-4455	Lynda
800-123-4455	Swee
*	

Figure 5-28
Data Entry Form with
Composite Multivalued
Attributes

The Archetype/Instance Pattern

The archetype/instance pattern (also called *version/instance*) occurs when one entity represents a manifestation or an instance of another entity. You have already seen one example of archetype/instance in the example of PAINTING and PRINT in Figure 5-10. The painting is the archetype, and the prints made from the painting are the instances of that archetype.

Other examples of archetype/instances are shown in Figure 5-30. One familiar example concerns classes and sections of classes. The class is the archetype, and the sections of the class are instances of that archetype. Other examples involve designs and instances of designs. A yacht manufacturer has various yacht designs, and each yacht is an instance of a particular design archetype. In a housing development, a contractor offers several different house models, and a particular house is an instance of that house model archetype.

As with all ID-dependent entities, the parent entity is required. The child entities (here SECTION, YACHT, and HOUSE) may or may not be required, depending on application requirements.

Logically, the child entity of every archetype/instance pattern is an ID-dependent entity. All three of the examples in Figure 5-30 are accurate representations of the logical structure of the underlying data. However, sometimes users will add additional identifiers to the instance entity and in the process change the ID-dependent entity to a weak entity that is not ID-dependent.

For example, although you can identify a SECTION by class name and section, colleges and universities often will add a unique identifier to SECTION, such as ReferenceNumber.

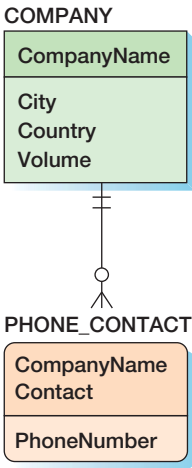


Figure 5-29
Data Model for the Form
in Figure 5-28

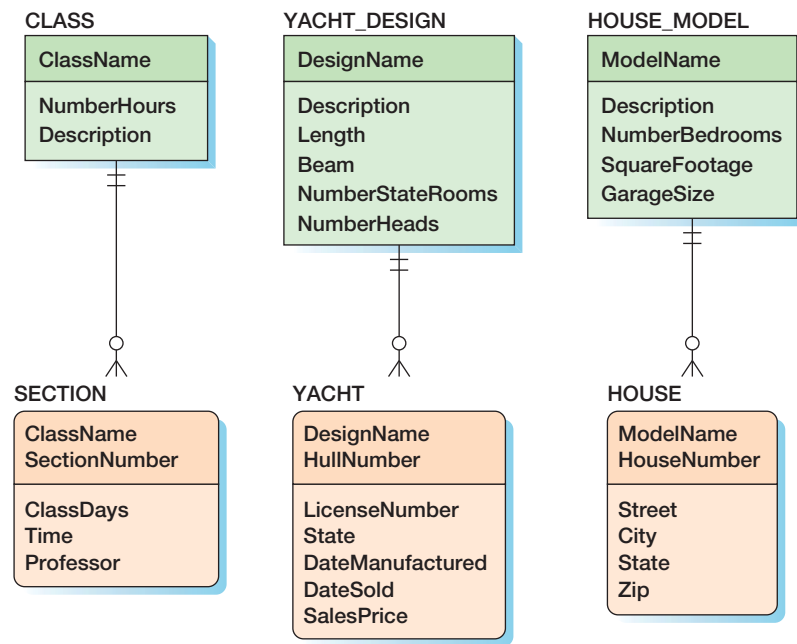


Figure 5-30

Three Archetype/Instance
Pattern Examples

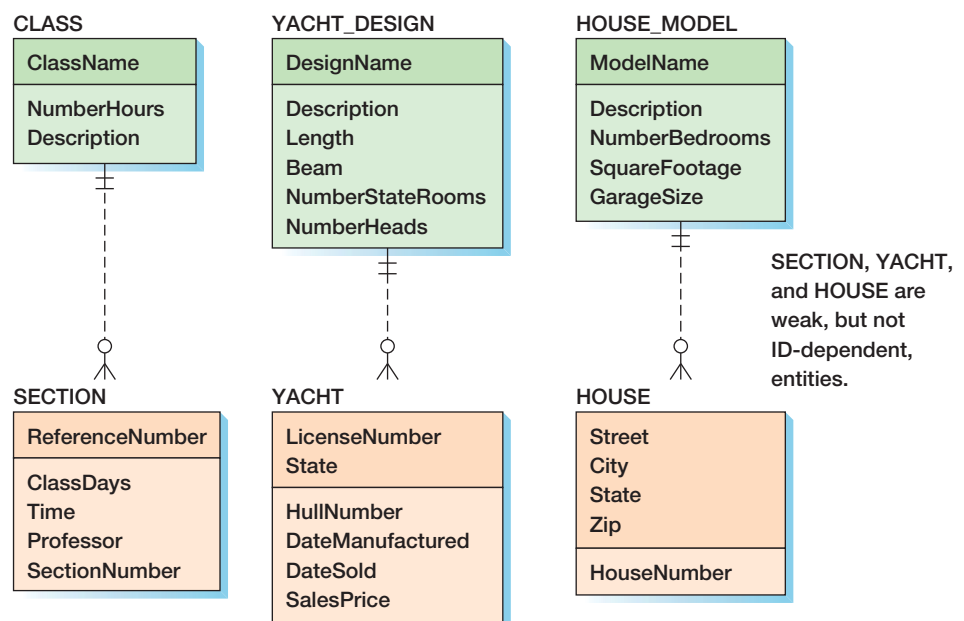
In that case, SECTION is no longer an ID-dependent entity, but it is still existence dependent on CLASS. Hence, as shown in Figure 5-31, SECTION is weak, but not ID-dependent.

A similar change may occur to the YACHT entity. Although the manufacturer of a yacht may refer to it by specifying the hull number of a given design, the local tax authority may refer to it by State and LicenseNumber. If we change the identifier of YACHT from (HullNumber, DesignName) to (LicenseNumber, State), then YACHT is no longer ID-dependent; it becomes a weak, non-ID-dependent entity.

Similarly, although the home builder may think of a home as the third house constructed according to the Cape Codd design, everyone else will refer to it by its address. When we change the identifier of HOUSE from (HouseNumber, ModelName) to (Street, City, State, Zip), then HOUSE becomes a weak, non-ID-dependent entity. All of these changes are shown in Figure 5-31.

Figure 5-31

Three Weak But Not
ID-Dependent Relationships



BY THE WAY

Data modelers continue to debate the importance of weak, non-ID-dependent entities. Everyone agrees that they exist, but not everyone agrees that they are important.

First, understand that existence dependence influences the way we write database applications. For the CLASS/SECTION example in Figure 5-31, we must insert a new CLASS before we can add a SECTION for that class. Additionally, when we delete a CLASS, we must delete all of the SECTIONs for that CLASS as well. This is one reason that some data modelers believe that weak, non-ID-dependent entities are important.

Skeptics say that although weak, non-ID-dependent entities may exist, they are not necessary. They say that we can obtain the same result by calling SECTION strong and making CLASS required. Because CLASS is required, the application will need to insert a CLASS before a SECTION is created and delete dependent SECTIONs when deleting a CLASS. So, according to that viewpoint, there is no practical difference between a weak, non-ID-dependent entity and a strong entity with a required relationship.

Others disagree. Their argument goes something like this: The requirement that a SECTION must have a CLASS comes from a logical necessity. It has to be that way—it comes from the nature of reality. The requirement that a strong entity must have a relationship to another strong entity arises from a business rule. Initially, we say that an ORDER must have a CUSTOMER (both strong entities), and then the application requirements change and we say that we can have cash sales, meaning that an ORDER no longer has to have a CUSTOMER. Business rules frequently change, but logical necessity never changes. We need to model weak, non-ID-dependent entities so that we know the strength of the required parent rule.

And so it goes. You, with the assistance of your instructor, can make up your own mind. Is there a difference between a weak, non-ID-dependent entity and a strong entity with a required relationship? In Figure 5-31, should we call the entities SECTION, YACHT, and HOUSE strong, as long as their relationships are required? We think not—we think there is a difference. Others think differently, however.

Mixed Identifying and Nonidentifying Patterns

Some patterns involve both identifying and nonidentifying relationships. The classic example is the line-item pattern, but there are other instances of mixed patterns as well. We begin with line items.

The Line-Item Pattern

Figure 5-32 shows a typical sales order, or invoice. Such forms usually have data about the order itself, such as the order number and order date, data about the customer, data about the salesperson, and then data about the items on the order. A data model for a typical SALES_ORDER is shown in Figure 5-33.

In Figure 5-33, CUSTOMER, SALESPERSON, and SALES_ORDER are all strong entities, and they have the nonidentifying relationships you would expect. The relationship from CUSTOMER to SALES_ORDER is 1:N, and the relationship from SALESPERSON to SALES_ORDER also is 1:N. According to this model, a SALES_ORDER must have a CUSTOMER and may or may not have a SALESPERSON. All of this is readily understood.

The interesting relationships concern the line items on the order. Examine the data grid in the form in Figure 5-32. Some of the data values belong to the order itself, but other data values belong to items in general. In particular, Quantity and ExtendedPrice belong to the SALES_ORDER, but ItemNumber, Description, and UnitPrice belong to ITEM. The lines on an order do not have their own identifier. No one ever says, “Give me the data for line 12.” Instead, they say, “Give me the data for line 12 of order 12345.” Hence, the identifier of a line is a composite of the identifier of a particular line and the identifier of a particular order. Thus, entries for line items are always ID-dependent on the order in which they appear. In Figure 5-33, ORDER_LINE_ITEM is ID-dependent on SALES_ORDER. The identifier of the ORDER_LINE_ITEM entity is (SalesOrderNumber, LineNumber).

Now, and this is the part that is sometimes confusing for some students, ORDER_LINE_ITEM is not existence-dependent on ITEM. It can exist even if no item has yet been

The screenshot shows a web-based sales order form titled "Carbon River Furniture Sales Order Form". It contains several input fields for customer and salesperson information, a table of line items, and a summary section.

Sales Order Number: 2011003845
Sales Order Date: 9/25/2011
Salesperson: Anne Dodsworth
Salesperson Code: EN-01

Customer Name: Kelly Welsch
Address: 1145 Elm Street
City: Carbon River
State: IL
Zip: 60662
Phone: 733-357-8462

Sales Order Line Items:

ItemNumber	Description	Quantity	UnitPrice	ExtendedPrice
92	Desk Chair	1	\$650.00	\$650.00
81	Conference Table	1	\$7,750.00	\$7,750.00
91	Side Chair	8	\$485.00	\$3,880.00
78	Executive Desk	1	\$3,500.00	\$3,500.00

Summary:
Subtotal: \$15,780.00
Tax: \$1,388.64
Total: \$17,168.64

Figure 5-32
Data Entry Form for a Sales Order

assigned to it. Further, if an ITEM is deleted, we do not want the line item to be deleted with it. The deletion of an ITEM may make the value of ItemNumber and other data invalid, but it should not cause the line item itself to disappear.

Now consider what happens to a line item when an order is deleted. Unlike with the deletion of an item, which only causes data items to become invalid, the deletion of the order removes the existence of the line item. Logically, a line item cannot exist if its order is deleted. Hence, line items are existence-dependent on orders.

Work through each of the relationships in Figure 5-33 and ensure that you understand their type and their maximum and minimum cardinalities. Also understand the implications

Figure 5-33
Data Model for the Sales Order in Figure 5-32

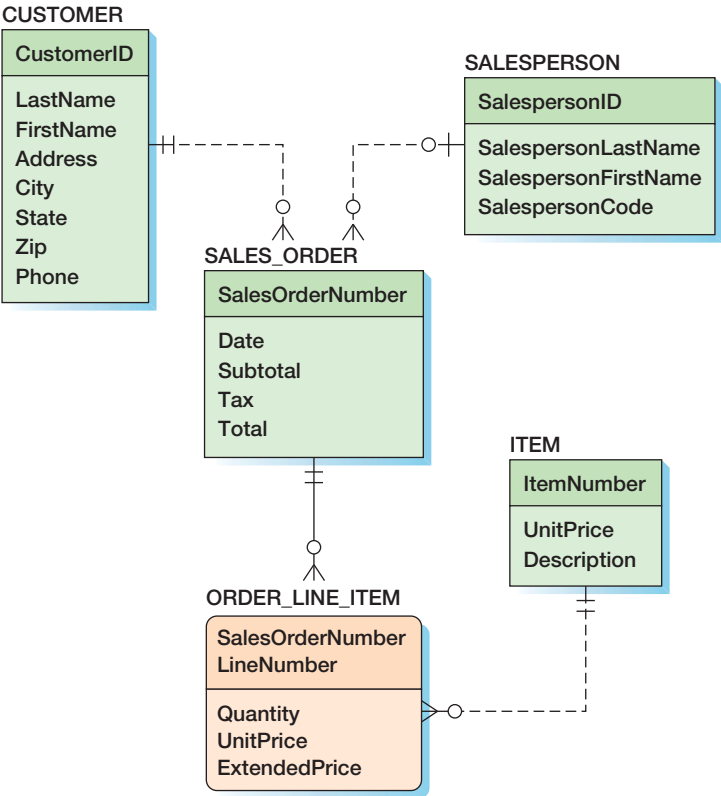
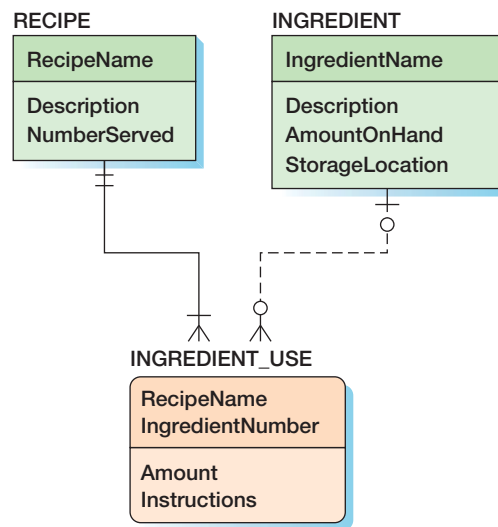


Figure 5-34
Mixed Relationship Pattern
for Restaurant Recipe



of this data model. For example, do you see why this sales order data model is unlikely to be used by a company in which salespeople are on commission?

Other Mixed Patterns

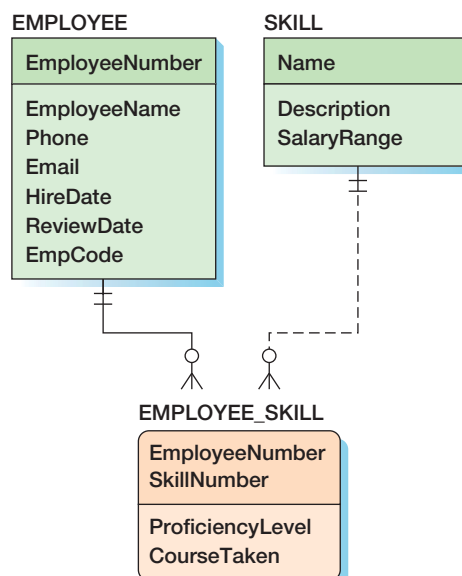
Mixed identifying and nonidentifying relationships occur frequently. Learn to look for a mixed pattern when a strong entity has a multivalued composite group and when one of the elements in the composite group is an identifier of a second strong entity.

Consider, for example, baking recipes. Each recipe calls for a certain amount of a specific ingredient, such as flour, sugar, or butter. The ingredient list is a multivalued composite group, but one of the elements of that group, the name of the ingredient, is the identifier of a strong entity. As shown in Figure 5-34, the recipe and the ingredients are strong entities, but the amount and instructions for using each ingredient are ID-dependent on **RECIPE**.

Or, consider employees' skill proficiencies. The name of the skill and the proficiency level the employee has are a multivalued group, but the skill itself is a strong entity, as shown in Figure 5-35. Dozens of other examples are possible.

Before continuing, compare the models in Figures 5-33, 5-34, and 5-35 with the association pattern in Figure 5-22. Make sure that you understand the differences and why the model in Figure 5-22 has two identifying relationships and the models in Figures 5-33, 5-34, and 5-35 have just one.

Figure 5-35
Mixed Relationship Pattern
for Employee Skills



Resident Fishing License 2011 Season <i>State of Washington</i>					License No: 03-1123432	
Name:						
Street:						
City:		State:		Zip:		
For Use by Commercial Fishers Only			For Use by Sport Fishers Only			
Vessel Number:			Number Years at This Address:			
Vessel Name:			Prior Year License Number:			
Vessel Type:						
Tax ID:						

Figure 5-36

Data Entry Form Suggesting the Need for Subtypes

The For-Use-By Pattern

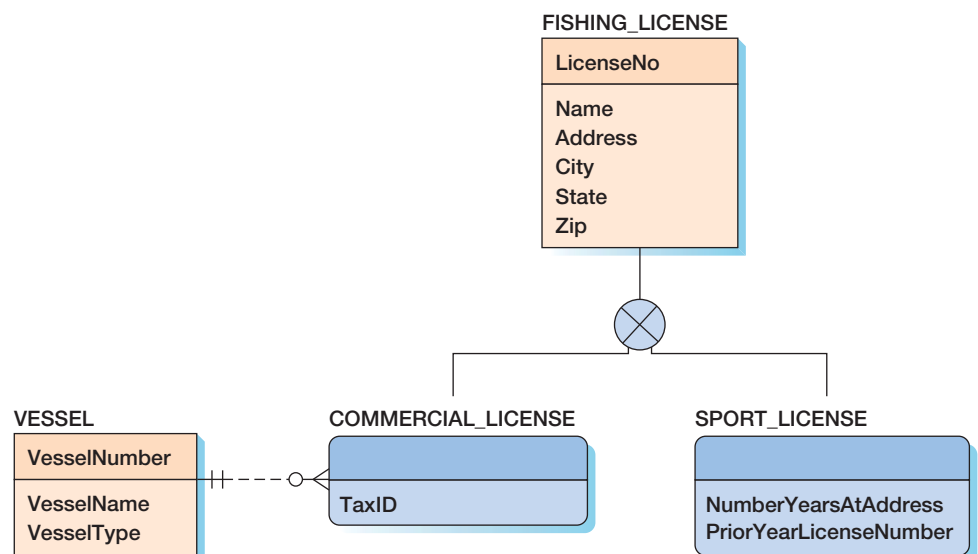
As stated earlier in this chapter, the major reason for using subtypes in a database design is to avoid value-inappropriate nulls. Some forms suggest the possibility of such nulls when they show blocks of data fields that are colored and labeled “For Use by *someone/something* Only.” For example, Figure 5-36 shows two tan sections, one for commercial fishers and another for sport fishers. The presence of these grayed-out sections indicates the need for subtype entities.

The data model for this form is shown in Figure 5-37. Observe that each tan section has a subtype. Notice that the subtypes differ not only in their attributes, but that one has a relationship that the other does not have. Sometimes, the only differences between subtypes are differences in the relationships they have.

The nonidentifying relationship from VESSEL to COMMERCIAL_LICENSE is shown as 1:N, mandatory to mandatory. In fact, this form does not have sufficient data for us to conclude that the maximum cardinality from VESSEL to COMMERCIAL_LICENSE is N. This fact was determined by interviewing users and learning that one boat is sometimes used by more than one commercial fisher. The minimum cardinalities indicate a commercial fisher must have a vessel, and that only vessels that are used for licenses are to be stored in this database.

Figure 5-37

Data Model for Form in Figure 5-36



The point of this example is to illustrate how forms often suggest the need for subtypes. Whenever you see a grayed out or otherwise distinguished section of a form with the words “For use by . . .,” think “subtype.”

Recursive Patterns

A recursive relationship occurs when an entity type has a relationship to itself. The classic examples of recursive relationships occur in manufacturing applications, but there are many other examples as well. As with strong entities, three types of recursive relationships are possible: 1:1, 1:N, and N:M. Let’s consider each.

1:1 Recursive Relationships

Suppose you are asked to construct a database for a railroad, and you need to make a model of a freight train. You know that one of the entities is BOXCAR, but how are BOXCARs related? To answer that question, envision a train. Except for the first boxcar, each has one boxcar in front, and, except for the last boxcar, each boxcar has one boxcar in back. Thus, the relationship is 1:1 between boxcars, with an optional relationship for the first and last cars.

Figure 5-38 shows a data model in which each BOXCAR has a 1:1 relationship to the BOXCAR ahead. The BOXCAR entity at the head of the train has a 1:1 relationship to ENGINE. (This model assumes a train has just one engine. To model trains with multiple engines, create a second recursive relationship among engines. Construct that relationship just like the Boxcar Ahead relationship.)

Figure 5-39 shows example entity instances that illustrate this data model. Not surprisingly, this set of entity instances looks just like a train.

An alternative model is to use the relationship to represent the BOXCAR behind. Either model works. Other examples of 1:1 recursive relationships are the succession of U.S. presidents, the succession of deans in a college of business, and the order of passengers on a waiting list.

Figure 5-38
Data Model for a 1:1
Recursive Relationship

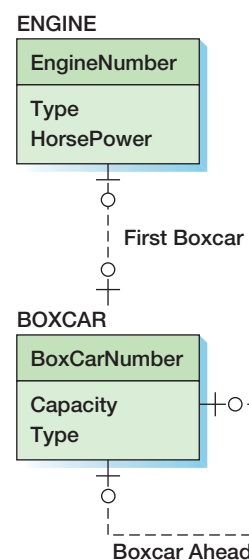
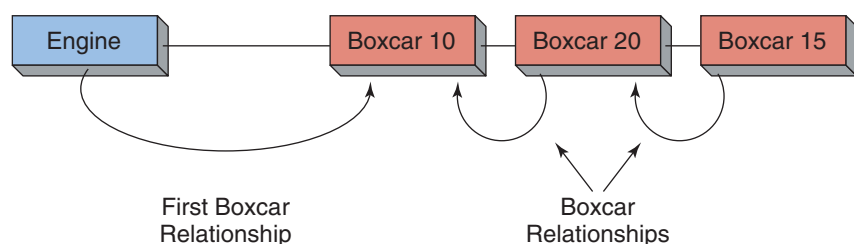


Figure 5-39
Sample Entities for the Data
Model in Figure 5-38



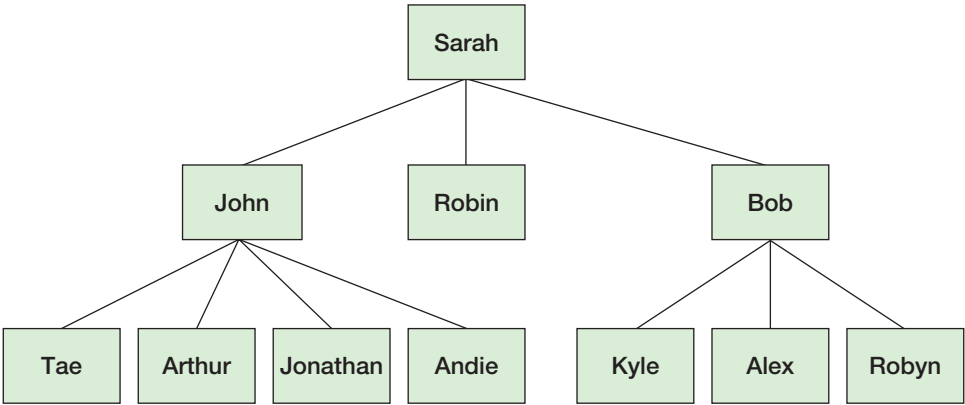


Figure 5-40
Managerial Relationships

1:N Recursive Relationships

The classic example of a 1:N recursive relationship occurs in organizational charts, in which an employee has a manager who may, in turn, manage several other employees. Figure 5-40 shows an example managerial chart. Note that the relationship between employees is 1:N.

Figure 5-41 shows a data model for the managerial relationship. The crow's foot indicates that a manager may manage more than one employee. The relationship is optional to optional because one manager (the president) has no manager and because some employees manage no one.

Another example of a 1:N recursive relationship concerns maps. For example, a world map has a relationship to many continent maps, each continent map has a relationship to many nation maps, and so forth. A third example concerns biological parents where the relationship from PERSON to PERSON is shown by tracing either mother or father (but not both).

N:M Recursive Relationships

N:M recursive relationships occur frequently in manufacturing applications, where they are used to represent bills of materials. Figure 5-42 shows an example.

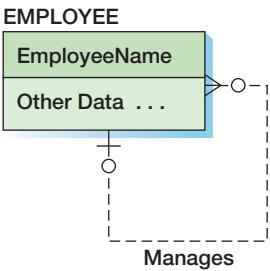


Figure 5-41
Data Model for the
Management Structure in
Figure 5-40 as a 1:N
Recursive Relationship

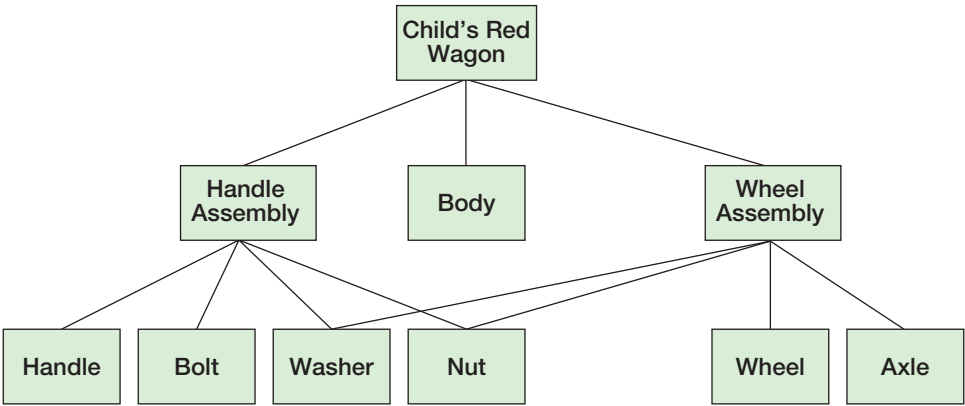
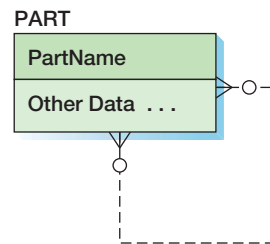


Figure 5-42
Bill of Materials

Figure 5-43

Data Model for the Bill of Materials in Figure 5-42 as an N:M Recursive Relationship



The key idea of a bill of materials is that one part is composed of other parts. A child's red wagon, for example, consists of a handle assembly, a body, and a wheel assembly, each of which is a part. The handle assembly, in turn, consists of a handle, a bolt, a washer, and a nut. The wheel assembly consists of wheels, axles, washers, and nuts. The relationship among the parts is N:M, because a part can be made up of many parts and because a part (such as washers and nuts) can be used in many parts.

The data model for a bill of materials is shown in Figure 5-43. Notice that each part has an N:M relationship to other parts. Because a part need not have any component parts, and because a part need not have any parts that contain it, the minimum cardinality is optional to optional.

BY THE WAY

What would happen to the data model if the diagram showed how many of each part are used? Suppose, for example, that the wheel assembly requires four washers and the handle assembly requires just one. The data model in Figure 5-43 will not be correct for this circumstance. In fact, adding Quantity to this N:M relationship is analogous to adding Price to the N:M relationship in Figure 5-22. See Project Question 5.67.

N:M recursive relationships can be used to model directed networks, such as the flow of documents through organizational departments or the flow of gas through a pipeline. They also can be used to model the succession of parents, in which mothers, fathers, and stepparents are included.

If recursive structures seem hard to comprehend, don't fret. They may seem strange at first, but they are not difficult. Work through some data examples to gain confidence. Make up a train and see how the model in Figure 5-38 applies or change the example in Figure 5-40 from employees to departments and see how the model in Figure 5-41 needs to be adjusted. Once you have learned to identify recursive patterns, you'll find it easy to create models for them.

The Data Modeling Process

During the data modeling process, the development team analyzes user requirements and constructs a data model from forms, reports, data sources, and user interviews. The process is always iterative; a model is constructed from one form or report and then supplemented and adjusted as more forms and reports are analyzed. Periodically, users are asked for additional information, such as that needed to assess minimum cardinality. Users also review and validate the data model. During that review, prototypes evidencing data model constructs may need to be constructed, as explained earlier.

To give you an idea of the iterative nature of data modeling, we will consider the development of a simple data model for a university. As you read this example, strive to appreciate how the model evolves as more and more requirements are analyzed.

For a more detailed version of this data modeling exercise, combined with an overview of the systems analysis and design process, see Appendix B.

BY THE WAY

One of the authors worked on a large data model for the U.S. Army’s logistical system. The model contained over 500 different entity types, and it took a team of seven people more than a year to develop, document, and validate it. On some occasions, the analysis of a new requirement indicated that the model had been conceived incorrectly, and days of work had to be redone. The most difficult aspect of the project was managing complexity. Knowing which entities related to which; whether an entity had already been defined; and whether a new entity was strong, weak, a supertype, or a subtype required a global understanding of the model. Memory was of poor help because an entity created in July could be a subtype of an entity created hundreds of entities earlier in February. To manage the model, the team used many different administrative tools. Keep this example in mind as you read through the development of the Highline University data model.

Suppose the administration at a hypothetical university named Highline University wants to create a database to track colleges, departments, faculty, and students. To do this, a data modeling team has collected a series of reports as part of its requirements determination. In the next sections, we will analyze these reports to produce a data model.

The College Report

Figure 5-44 shows an example report from Highline University about one college within the university, specifically, the College of Business. This example is one instance of this report; Highline University has similar reports about other colleges, such as the College of Engineering and the College of Social Sciences. The data modeling team needs to gather enough examples to form a representative sample of all the college reports. Here, assume that the report in Figure 5-44 is representative.

Examining the report, we find data specific to the college—such as the name, dean, telephone number, and campus address—and also facts about each department within the college. These data suggest that the data model should have COLLEGE and DEPARTMENT entities with a relationship between them, as shown in Figure 5-45.

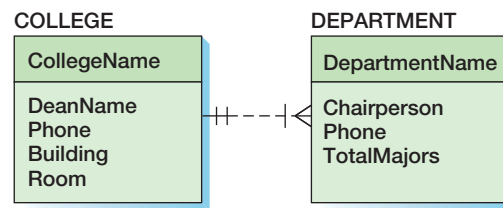
The relationship in Figure 5-45 is nonidentifying. This relationship is used because DEPARTMENT is not ID-dependent, and, logically, a DEPARTMENT is independent of a COLLEGE. We cannot tell from the report in Figure 5-44 whether a department can belong to many colleges. To answer this question, we need to ask the users or look at other forms and reports.

Assume that we know from the users that a department belongs to just one college, and the relationship is thus 1:N from COLLEGE to DEPARTMENT. The report in Figure 5-44 does not show us the minimum cardinalities. Again, we must ask the users. Assume we learn from the users that a college must have at least one department, and a department must be assigned to exactly one college.

Figure 5-44
Highline University Sample
College Report

College of Business Mary B. Jefferson, Dean			
Phone: 232-1187		Campus Address: Business Building, Room 100	
Department	Chairperson	Phone	Total Majors
Accounting	Jackson, Seymour P.	232-1841	318
Finance	HeuTeng, Susan	232-1414	211
Info Systems	Brammer, Nathaniel D.	236-0011	247
Management	Tuttle, Christine A.	236-9988	184
Production	Barnes, Jack T.	236-1184	212

Figure 5-45
Data Model for the College
Report in Figure 5-44



The Department Report

The Department Report shown in Figure 5-46 contains departmental data along with a list of the professors who are assigned to that department. This report contains data concerning the department's campus address. Because these data do not appear in the DEPARTMENT entity in Figure 5-45, we need to add them, as shown in Figure 5-47(a). This is typical of the data modeling process. That is, entities and relationships are adjusted as additional forms, reports, and other requirements are analyzed.

Figure 5-47(a) also adds the relationship between DEPARTMENT and PROFESSOR. We initially model this as an N:M relationship, because a professor might have a joint appointment. The data modeling team must further investigate the requirements to determine whether joint appointments are allowed. If not, the relationship can be redefined as a nonidentifying 1:N, as shown in Figure 5-47(b).

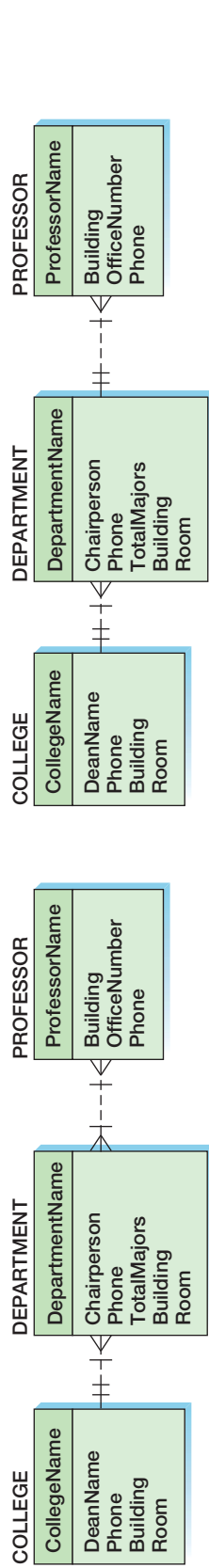
Another possibility regarding the N:M relationship is that some attribute about the combination of a professor and a department is missing. If so, then an association pattern is more appropriate. At Highline, suppose the team finds a report that describes the title and employment terms for each professor in each department. Figure 5-47(c) shows an entity for such a report, named APPOINTMENT. As you would expect from the association pattern, APPOINTMENT is ID-dependent on both DEPARTMENT and PROFESSOR.

A chairperson is a professor, so another improvement on the model is to remove the Chairperson data from DEPARTMENT and replace it with a chairperson relationship. This has been done in Figure 5-47(d). In the Chairs/Chaired By relationship, the PROFESSOR is the parent entity. A professor can be a chair of zero or one departments, and a department must have exactly one professor as chair.

With the Chairs/Chaired By relationship, the attribute Chairperson is no longer needed in DEPARTMENT, so it is removed. Normally, a chairperson has his or her office in the department office; if this is the case, Phone, Building, and Room in DEPARTMENT duplicate Phone, Building, and OfficeNumber in PROFESSOR. Consequently, it might be possible to remove Phone, Building, and Room from DEPARTMENT. However, a professor may have a different phone from the official department phone, and the professor may also have an office outside of the department's office. Because of this possibility, we will leave Phone, Building, and Room in DEPARTMENT.

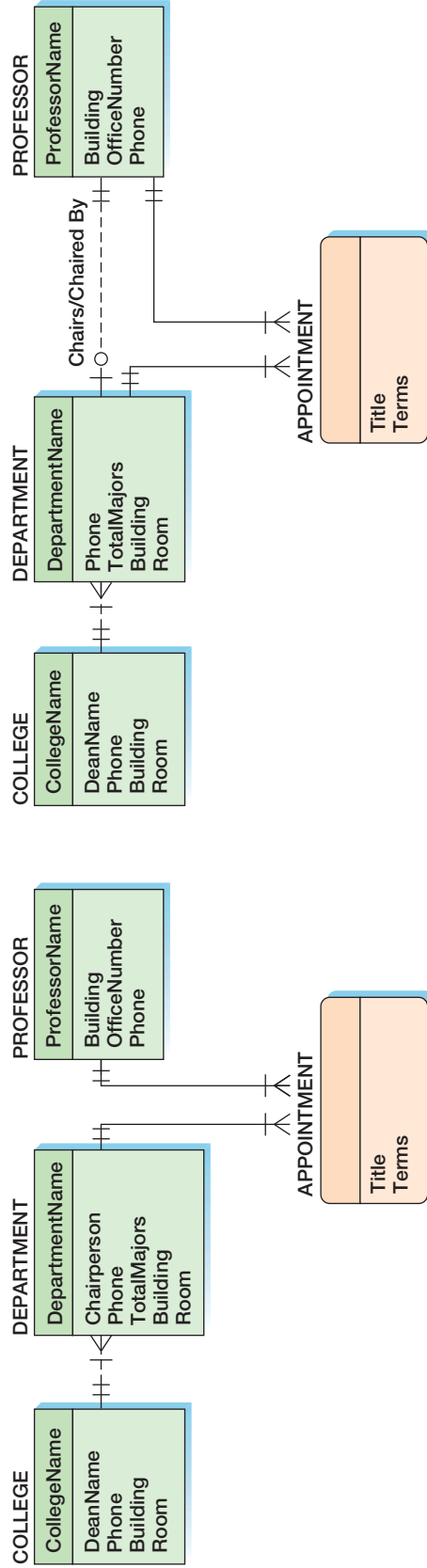
Figure 5-46
Highline University Sample
Department Report

Information Systems Department College of Business		
Chairperson:	Brammer, Nathaniel D	
Phone:	236-0011	
Campus Address:	Social Science Building, Room 213	
Professor	Office	Phone
Jones, Paul D.	Social Science, 219	232-7713
Parks, Mary B	Social Science, 308	232-5791
Wu, Elizabeth	Social Science, 207	232-9112



(a) Data Model Using an N:M Relationship

(b) Data Model Using a 1:N Relationship



(c) Data Model Using an Association Pattern

(d) Data Model Using an Association Pattern and 1:1 Relationship

Figure 5-47

Alternate Data Models for the DEPARTMENT-to-PROFESSOR Relationship

The Department/Major Report

Figure 5-48 shows a report of a department and the students who major in that department. This report indicates the need for a new entity called STUDENT. Because students are not ID-dependent on departments, the relationship between DEPARTMENT and STUDENT is non-identifying, as shown in Figure 5-49. We cannot determine the minimum cardinality from Figure 5-48, but assume that interviews with users indicate that a STUDENT must have MAJOR, but no MAJOR need have any students. Also, using the contents of this report as a guide, attributes StudentNumber, StudentName, and Phone are placed in STUDENT.

There are two subtleties in this interpretation of the report in Figure 5-48. First, observe that Major's Name was changed to StudentName when the attribute was placed in STUDENT. This was done because StudentName is more generic. Major's Name has no meaning outside the context of the Major relationship. Additionally, the report heading in Figure 5-48 has an ambiguity. Is the phone number for the department a value of DEPARTMENT.Phone or a value of PROFESSOR.Phone? The team needs to investigate this further with the users. Most likely, it is a value of DEPARTMENT.Phone.

The Student Acceptance Letter

Figure 5-50 shows the acceptance letter that Highline sends to its incoming students. The data items in this letter that need to be represented in the data model are shown in boldface. In addition to data concerning the student, this letter also contains data regarding the student's major department as well as data about the student's adviser.

We can use this letter to add an Advises/Advised By relationship to the data model. However, which entity should be the parent of this relationship? Because an adviser is a professor, it is tempting to make PROFESSOR the parent. However, a professor acts as an

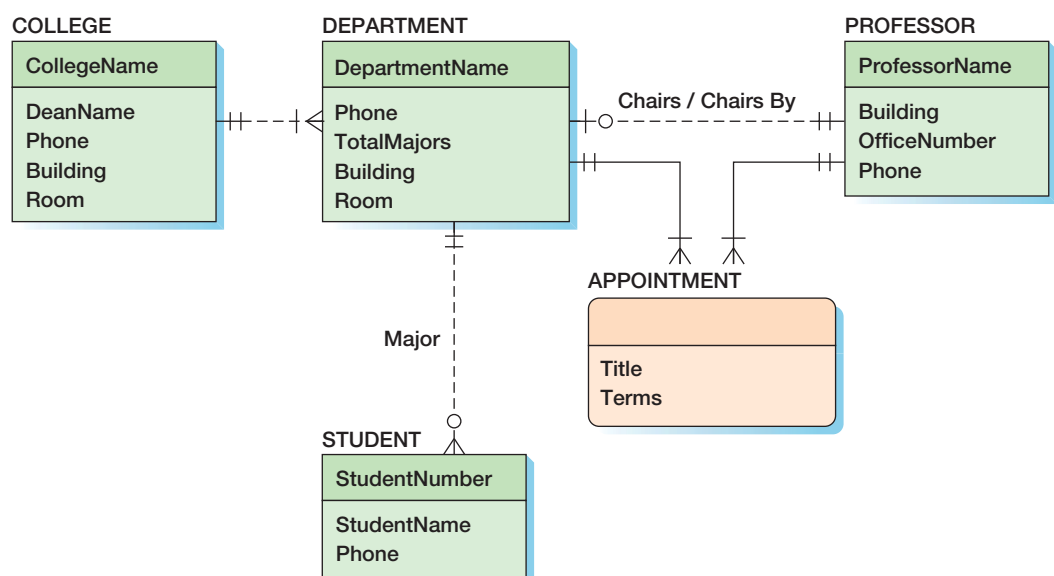
Figure 5-48

Highline University Sample
Department Student Report

Student Major List Information Systems Department		
Chairperson: Brammer, Nathaniel D Phone: 236-0011		
Major's Name	Student Number	Phone
Jackson, Robin R.	12345	237-8713
Lincoln, Fred J.	48127	237-8713
Madison, Janice A.	37512	237-8713

Figure 5-49

Data Model with STUDENT
Entity



Mr. Fred Parks
123 Elm Street
Los Angeles, CA 98002

Dear Mr. Parks:

You have been admitted as a major in the **Accounting** Department at Highline University, starting in the Fall Semester, 2011. The office of the Accounting Department is located in the **Business** Building, Room 210.

Your adviser is professor **Elizabeth Johnson**, whose telephone number is **232-8740** and whose office is located in the **Business** Building, Room 227. Please schedule an appointment with your adviser as soon as you arrive on campus.

Congratulations and welcome to Highline University!

Sincerely,

Jan P. Smathers
President

JPS/rkp

Figure 5-50

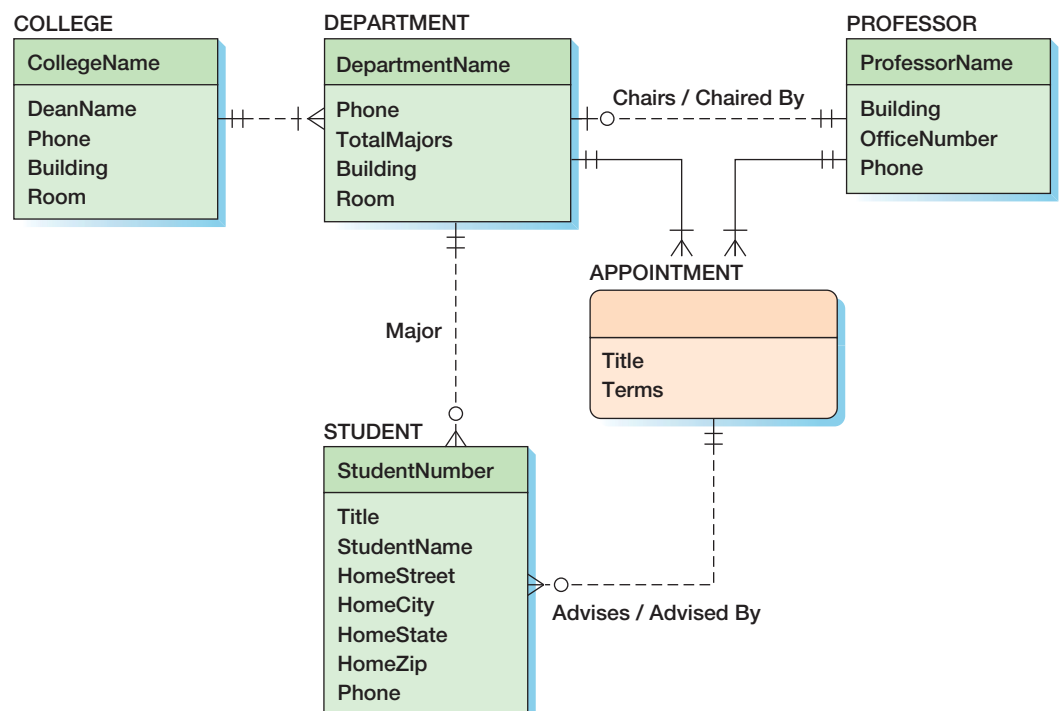
Highline University Sample
Student Acceptance Letter

adviser within the context of a particular department. Therefore, Figure 5-51 shows APPOINTMENT as the parent of STUDENT. To produce the report in Figure 5-50, the professor's data can be retrieved by accessing the related APPOINTMENT entity and then accessing that entity's PROFESSOR parent. This decision is not cut-and-dried, however. One can make a strong argument that the parent of the relationship should be PROFESSOR.

According to this data model, a student has at most one adviser. Also, a student must have an adviser, but no professor (via APPOINTMENT) need advise any students. These constraints cannot be determined from any of the reports shown and will need to be verified with the users.

Figure 5-51

Data Model with Advises
Relationship



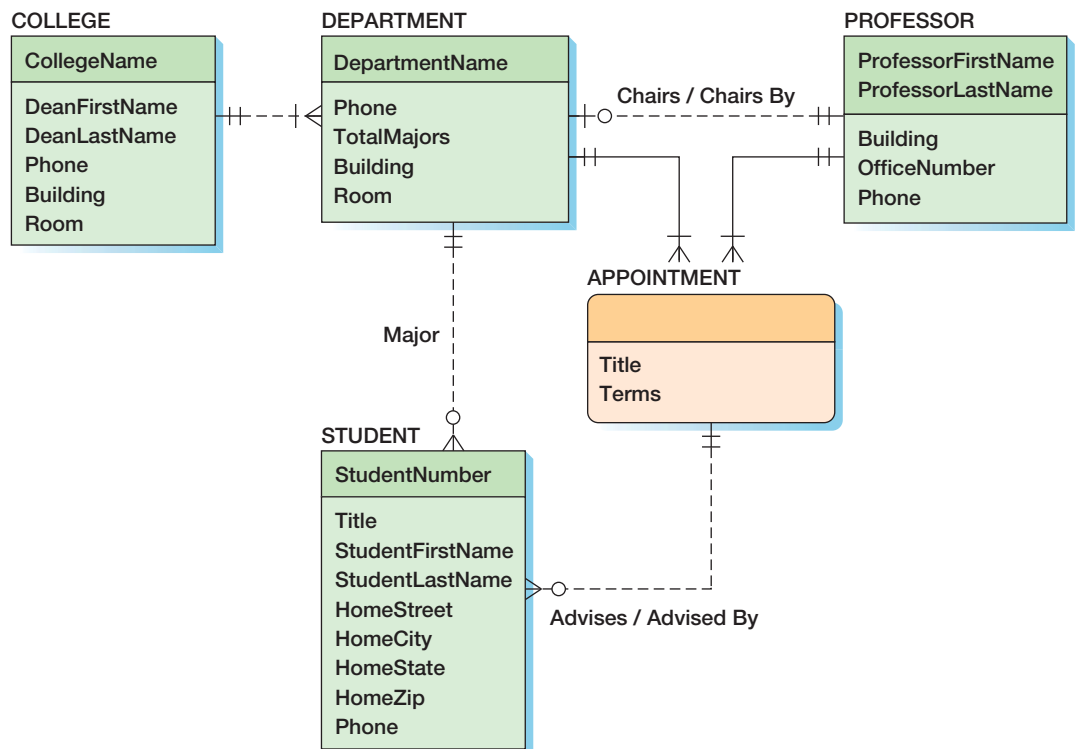


Figure 5-52
Final Data Model

The acceptance letter uses the title *Mr.* in the salutation. Therefore, a new attribute called Title is added to STUDENT. Observe that this Title is different from the one in APPOINTMENT. This difference will need to be documented in the data model to avoid confusion. The acceptance letter also shows the need to add new home address attributes to STUDENT.

The acceptance letter reveals a problem. The name of the student is Fred Parks, but we have allocated only one attribute, StudentName, in STUDENT. It is difficult to reliably disentangle first and last names from a single attribute, so a better model is to have two attributes: StudentFirstName and StudentLastName. Similarly, note that the adviser in this letter is Elizabeth Johnson. So far, all professor names have been in the format Johnson, Elizabeth. To accommodate both forms of name, ProfessorName in PROFESSOR must be changed to the two attributes ProfessorFirstName and ProfessorLastName. A similar change is necessary for DeanName. These changes are shown in Figure 5-52, which is the final form of this data model.

This section should give you a feel for the nature of a data modeling project. Forms and reports are examined in sequence, and the data model is adjusted as necessary to accommodate the knowledge gained from each new form or report. It is very typical to revise the data model many, many times throughout the data modeling process. See Project Question 5.67 for yet another possible revision.

Summary

When databases are developed as part of a new information systems project, the database design is accomplished in two phases. First, a data model is constructed from forms, reports, data sources, and other requirements. The data model is then transformed into a database design. A data model is a blueprint for a database design. Like blueprints for buildings, data models can be altered as necessary, with little effort. Once the database is constructed, however, such alterations are time consuming and very expensive.

The most prominent data model in use today is the entity-relationship, or E-R, data model. It was invented by Peter Chen and extended by others to include subtypes. An entity is something that users want to track. An entity class is a collection of entities of the same type and is described by the structure of the entities in the class. An entity instance is one entity of a given class. Entities have attributes that describe their characteristics. Identifiers are attributes that name entity instances. Composite identifiers consist of two or more attributes.

The E-R model includes relationships, which are associations among entities. Relationship classes are associations among entity classes, and relationship instances are associations among entity instances. Today, relationships are not allowed to have attributes. Relationships can be given names so that they can be identified.

The degree of a relationship is the number of entity types that participate in the relationship. Binary relationships have only two entity types. In practice, relationships of degrees greater than two are decomposed into multiple binary relationships.

The difference between an entity and a table is that you can express an entity relationship without specifying foreign keys. Working with entities reduces complexity and makes it easier to revise the data model as work progresses.

Relationships are classified according to their cardinality. Maximum cardinality is the maximum number of instances that can participate in a relationship instance. Minimum cardinality is the least number of entities that must participate in a relationship.

Relationships commonly have one of three maximum cardinalities: 1:1, 1:N, or N:M. In rare instances, a maximum cardinality might be a specific number, such as 1:15. Relationships commonly have one of four basic minimum cardinalities: optional to optional, mandatory to optional, optional to mandatory, or mandatory to mandatory. In rare cases, the minimum cardinality is a specific number.

Unfortunately, many variations of the E-R model are in use. The original version represented relationships with diamonds. The Information Engineering version uses a line with a crow's foot, the IDEF1X version uses another set of symbols, and UML uses yet another set. To add further complication, many data modeling products have added their own symbols. In this text, we will use the IE Crow's Foot model with symbols, as summarized in Figure 5-14. Other models and techniques are summarized in Appendices B, C, D, and H.

An ID-dependent entity is an entity whose identifier includes the identifier of another entity. Such entities use an

identifying relationship. In such relationships, the parent is always required, but the child (the ID-dependent entity) may or may not be required, depending on application requirements. Identifying relationships are shown with solid lines in E-R diagrams.

A weak entity is an entity whose existence depends on the presence of another entity. All ID-dependent entities are weak. Additionally, some entities are weak, but not ID-dependent. Some people believe such entities are not important; others believe they are.

A subtype entity is a special case of another entity called its supertype. Subtypes may be exclusive or inclusive. Exclusive subtypes sometimes have discriminators, which are attributes that specify a supertype's subtype. The most important (and perhaps only) reason for creating subtypes in a data model is to avoid value-inappropriate nulls.

Relationships among nonsubtype entities are called HAS-A relationships. Relationships among supertype/subtype entities are called IS-A relationships.

The elements of a data model are constructed by analyzing forms, reports, and data sources. Many forms and reports fall into common patterns. In this text, we discussed the 1:1, 1:N, and N:M strong entity patterns. We also discussed three patterns that use ID-dependent relationships: association, multivalue attribute, and version/instance. Some forms involve mixed identifying and nonidentifying patterns. Line items are the classic example of mixed forms, but there are other examples as well.

The for-use-by pattern indicates the need for subtypes. In some cases, subtypes differ because they have different attributes, but they also can differ because they have different relationships. A recursive relationship occurs when an entity has a relationship to itself. The three types of recursive relationship are 1:1, 1:N, and N:M.

The data modeling process is iterative. Forms and reports are analyzed, and the data model is created, modified, and adjusted as necessary. Sometimes, the analysis of a form or report will require that earlier work be redone. *C'est la vie!*

Key Terms

association pattern
attribute
binary relationship
cardinality
child
composite identifier
crow's foot symbol
data model
degree
discriminator
entity
entity class
entity instance

entity-relationship (E-R) diagrams
entity-relationship (E-R) model
exclusive subtype
extended E-R model
HAS-A relationship
ID-dependent entity
identifier
identifying relationship
IE Crow's Foot model
inclusive subtype
Information Engineering (IE) model
Integrated Definition 1, Extended (IDEF1X)
IS-A relationship

mandatory	optional-to-optional (O-O) relationship
mandatory-to-mandatory (M-M) relationship	parent
mandatory-to-optional (M-O) relationship	relationship
many-to-many (N:M) relationship	relationship class
maximum cardinality	relationship instance
minimum cardinality	strong entity
nonidentifying relationship	subtype
one-to-many (1:N) relationship	supertype
one-to-one (1:1) relationship	ternary relationship
optional	Unified Modeling Language (UML)
optional-to-mandatory (O-M) relationship	weak entity

Review Questions

- 5.1 Describe the two phases in designing databases that arise from the development of new information systems.
- 5.2 In general terms, explain how a data model could be used to design a database for a small video rental store.
- 5.3 Explain how a data model is like a building blueprint. What is the advantage of making changes during the data modeling stage?
- 5.4 Who is the author of the entity-relationship data model?
- 5.5 Define *entity*. Give an example of an entity (other than one presented in this chapter).
- 5.6 Explain the difference between an entity class and an entity instance.
- 5.7 Define *attribute*. Give an example attribute for the entity in your answer to Review Question 5.5.
- 5.8 Define *identifier*. Give an example identifier for the entity in your answer to Review Question 5.5.
- 5.9 Give an example of a composite identifier.
- 5.10 Define *relationship*. Give an example of a relationship (other than one presented in this chapter). Name your relationship.
- 5.11 Explain the difference between a relationship class and a relationship instance.
- 5.12 What is the degree of relationship? Give an example of a relationship of degree three (other than one presented in this chapter).
- 5.13 What is a binary relationship?
- 5.14 Explain the difference between an entity and a table. Why is this difference important?
- 5.15 What does cardinality mean?
- 5.16 Define the terms *maximum cardinality* and *minimum cardinality*.
- 5.17 Give examples of 1:1, 1:N, and N:M relationships (other than those presented in this chapter). Use the traditional diamond notation to diagram your examples.
- 5.18 Give an example for which the maximum cardinality must be an exact number.
- 5.19 Give examples of M-M, M-O, O-M, and O-O relationships (other than those presented in this chapter). Use the circle and hash mark notation on the diamond portrayal of relationships.
- 5.20 Explain, in general terms, how the traditional E-R model, the IE Crow's Foot version, the IDEF1X version, and the UML version differ. Which version is used primarily in this text?

- 5.21 Explain how the notations shown in Figure 5-7 differ.
- 5.22 Explain how the notations shown in Figure 5-9 differ.
- 5.23 What is an ID-dependent entity? Give an example of an ID-dependent entity (other than one presented in this chapter).
- 5.24 Explain how to determine the minimum cardinality of both sides of an ID-dependent relationship.
- 5.25 What rules exist when creating an instance of an ID-dependent entity? What rules exist when deleting the parent of an ID-dependent entity?
- 5.26 What is an identifying relationship? How is it used?
- 5.27 Explain why the relationship between PRODUCT and VERSION discussed on page 165 is an identifying relationship.
- 5.28 What is a weak entity? How do weak entities relate to ID-dependent entities?
- 5.29 What distinguishes a weak entity from a strong entity that has a required relationship to another entity?
- 5.30 Define *subtype* and *supertype*. Give an example of a subtype–supertype relationship (other than one presented in this chapter).
- 5.31 Explain the difference between exclusive subtypes and inclusive subtypes. Give an example of each.
- 5.32 What is a discriminator?
- 5.33 Explain the difference between IS-A and HAS-A relationships.
- 5.34 What is the most important reason for using subtypes in a data model?
- 5.35 Describe the relationship between the structure of forms and reports and the data model.
- 5.36 Explain two ways forms and reports are used for data modeling.
- 5.37 Explain why the form and report in Figure 5-15 indicate that the underlying relationship is 1:1.
- 5.38 Why is it not possible to infer minimum cardinality from the form and report in Figure 5-15?
- 5.39 Describe two tests for determining if an entity is a strong entity.
- 5.40 Why does the form in Figure 5-17 not indicate that the underlying relationship is 1:N? What additional information is required to make that assertion?
- 5.41 Explain why two forms or reports are usually needed to infer maximum cardinality.
- 5.42 How can you assess minimum cardinality for the entities in the form in Figure 5-17?
- 5.43 Explain why the form and report in Figure 5-19 indicate that the underlying relationship is N:M.
- 5.44 Name three patterns that use ID-dependent relationships.
- 5.45 Explain how the association pattern differs from the N:M strong entity pattern. What characteristic of the report in Figure 5-21 indicates that an association pattern is needed?
- 5.46 In general terms, explain how to differentiate an N:M strong entity pattern from an association pattern.
- 5.47 Explain why two entities are needed to model multivalued attributes.
- 5.48 How do the forms in Figures 5-26 and 5-28 differ? How does this difference affect the data model?

- 5.49 Describe, in general terms, the archetype/instance pattern. Why is an ID-dependent relationship needed for this pattern? Use the CLASS/SECTION example shown in Figure 5-30 in your answer.
- 5.50 Explain what caused the entities in Figure 5-31 to change from ID-dependent entities.
- 5.51 Summarize the two sides in the argument about the importance of weak, but not ID-dependent, entities.
- 5.52 Give an example of the line-item pattern as it could be used to describe the contents of a shipment. Assume that the shipment includes the names and quantities of various items as well as each item's insured value. Place the insurance value per item in an ITEM entity.
- 5.53 What entity type should come to mind when you see the words "For use by" in a form?
- 5.54 Give examples of 1:1, 1:N, and N:M recursive relationships (other than those presented in this chapter).
- 5.55 Explain why the data modeling process must be iterative. Use the Highline University example.

Project Questions

- 5.56 This question is for Microsoft Visio users. Convert the data models in Figures 5-16, 5-20, 5-22, 5-23, 5-30, 5-33, 5-37, and 5-52 into Visio format. Use the Visio arrow notation. (Hint: See Appendix F, "Getting Started with Microsoft Visio 2010.")
- 5.57 This question is for Visio users. Convert the data models in Figures 5-16, 5-20, 5-22, 5-23, 5-30, 5-33, 5-37, and 5-52 into Visio format. Use the Visio version of IE Crow's Foot notation. (Hint: See Appendix F.)

Answer the following questions using IE Crow's Foot notation.

- 5.58 Examine the subscription form shown in Figure 5-53. Using the structure of this form, do the following:
 - A. Create a model with one entity. Specify the identifier and attributes.
 - B. Create a model with two entities, one for customer and a second for subscription. Specify identifiers, attributes, relationship name, type, and cardinalities.
 - C. Under what conditions do you prefer the model in A to that in B?
 - D. Under what conditions do you prefer the model in B to that in A?

Figure 5-53
Subscription Form

Fine Wood ▲▲▲▲▲Working	To subscribe
<p> <input type="checkbox"/> 1 year (6 issues) for just \$18—20% off the newsstand price. (Outside the U.S. \$21/year—U.S. funds, please) </p> <p> <input type="checkbox"/> 2 years (12 issues) for just \$34—save 24% (Outside the U.S. \$40/2 years—U.S. funds, please) </p> <p> Name _____ </p> <p> Address _____ </p> <p> City _____ State _____ Zip _____ </p> <p> <input type="checkbox"/> My payment is enclosed. <input type="checkbox"/> Please bill me. </p> <p> Please start my subscription with <input type="checkbox"/> current issue <input type="checkbox"/> next issue. </p>	

WASHINGTON STATE PATROL CORRECTION NOTICE

NAME		Kroenke		David M	
LAST		FIRST			
ADDRESS 5053 88 Ave SE					
CITY		STATE		ZIP CODE	
Mercer Island		Wa		98040	
DRIVERS LICENSE		STATE		BIRTH DATE	
00000		Wa		2/27/46	
VEHICLES LICENSE		STATE		COLOR	
AAA000		Wa			
YEAR		MAKE		TYPE	
90		Saab		900	
VIN					
REGISTERED					
OWNER					
ADDRESS					
VIOLATION DATE		TIME		DIST	
MO 11 DAY 7 YEAR 2011		935		2	
LOCATION		MILES		OF	
17		E		Enumckum	
ON		SR410			
VIOLATIONS					
Writing text while driving					
OFFICERS SIGNATURE		PERSONNEL NUMBER			
S Scott		850			
<input checked="" type="checkbox"/> This is a warning, no further action is required.					
<input type="checkbox"/> You are released to take this vehicle to a place of repair. Continued operation on the roadway is not authorized.					
<input type="checkbox"/> CORRECT VIOLATION(S) IMMEDIATELY. Return this signed card for proof of compliance within 15/30 days. (if this box checked)					
<input checked="" type="checkbox"/> DRIVERS SIGNATURE					

Figure 5-54
Traffic Citation

- 5.59 Consider the traffic citation shown in Figure 5-54. The rounded corners on this form provide graphical hints about the boundaries of the entities represented.
- A. Create a data model with five entities. Use the data items on the form to specify identifiers and attributes for those entities.
 - B. Specify relationships among the entities. Name the relationship and give its type and cardinalities. Indicate which cardinalities can be inferred from data on the form and which need to be checked out with systems users.
- 5.60 Examine the list of e-mail messages in Figure 5-55. Using the structure and example data items in this list, do the following:
- A. Create a single-entity data model for this list. Specify the identifier and all entities.
 - B. Modify your answer to A to include entities SENDER and SUBJECT. Specify the identifiers and attributes of entities and the type and cardinalities of the relationships. Explain which cardinalities can be inferred from Figure 5-55 and which need to be checked out with users.
 - C. The e-mail address in the From column in Figure 5-55 is in two different styles. One style has the true e-mail address; the second style (e.g., Tom Cooper) is the name of an entry in the user's e-mail directory. Create two categories of SENDER based on these two styles. Specify identifiers and attributes.
- 5.61 Examine the list of stock quotes in Figure 5-56. Using the structure and example data items in this list, do the following:
- A. Create a single-entity data model for this list. Specify the identifier and attributes.
 - B. Modify your answer to A to include the entities COMPANY and INDEX. Specify the identifier and attributes of the entities and the type and cardinalities of the relationships. Explain which cardinalities can be inferred from Figure 5-56 and which need to be checked out with users.

Figure 5-55
Email List

	From	Subject	Date ↓	Size
<input type="checkbox"/>	WDA2259@sailmail.com	Big Wind	5/13/2011	3 KB
<input type="checkbox"/>	WDA2259@sailmail.com	Update	5/12/2011	4 KB
<input type="checkbox"/>	WDA2259@sailmail.com	Re: Saturday Am	5/11/2011	4 KB
<input type="checkbox"/>	WDA2259@sailmail.com	Re: Weather window!	5/10/2011	4 KB
<input type="checkbox"/>	WDA2259@sailmail.com	Re: Howdy!	5/10/2011	3 KB
<input type="checkbox"/>	WDA2259@sailmail.com	Still here	5/9/2011	3 KB
<input type="checkbox"/>	WDA2259@sailmail.com	Re: Turtle Bay	5/8/2011	4 KB
<input type="checkbox"/>	WDA2259@sailmail.com	Turtle Bay	5/8/2011	4 KB
<input type="checkbox"/>	WDA2259@sailmail.com	Re: Hi	5/6/2011	3 KB
<input type="checkbox"/>	WDA2259@sailmail.com	Sunday, Santa Maria	5/5/2011	3 KB
<input type="checkbox"/>	Ki6yu@aol.com	Cabo, Thurs. Noon	5/2/2011	2 KB
<input type="checkbox"/>	WDA2259@sailmail.com	turbo	5/1/2011	3 KB
<input type="checkbox"/>	WDA2259@sailmail.com	on our way	4/28/2011	3 KB
<input type="checkbox"/>	Tom Cooper	RE: Hola!	4/26/2011	3 KB
<input type="checkbox"/>	Tom Cooper	RE: Hola!	4/24/2011	2 KB
<input type="checkbox"/>	Tom Cooper	RE: Hola!	4/23/2011	3 KB

- C.** The list in Figure 5-56 is for a quote on a particular day at a particular time of day. Suppose that the list were changed to show closing daily prices for each of these stocks and that it includes a new column: QuoteDate. Modify your model in B to reflect this change.
- D.** Change your model in C to include the tracking of a portfolio. Assume the portfolio has an owner name, a phone number, an e-mail address, and a list of stocks held. The list includes the identity of the stock and the number of shares held. Specify all additional entities, their identifiers and attributes, and the type and cardinality of all relationships.
- E.** Change your answer to part D to keep track of portfolio stock purchases and sales in a portfolio. Specify entities, their identifiers and attributes, and the type and cardinality of all relationships.

5.62 Figure 5-57 shows the specifications for single-stage air compressor products. Note that there are two product categories that are based on Air Performance: The A models are at 125 pounds per square inch of pressure, and the E models are at 150 pounds per square inch of pressure. Using the structure and example data items in this list, do the following:

- A.** Create a set of exclusive subtypes to represent these compressors. The supertype will have attributes for all single-stage compressors, and the subtypes will have

Figure 5-56
Stock Quotations

Symbol	Name	Last	Change	% Chg
\$COMPX	Nasdaq Combined Composite Index	1,400.74 ▼	-4.87	-0.35%
\$INDU	Dow Jones Industrial Average Index	9,255.10 ▼	-19.80	-0.21%
\$INX	S&P 500 INDEX	971.14 ▼	-5.84	-0.60%
ALTR	Altera Corporation	13.45 ▼	-0.450	-3.24%
AMZN	Amazon.com, Inc.	15.62 ▲	+0.680	+4.55%
CSCO	Cisco Systems, Inc.	13.39 ▼	-0.280	-2.05%
DELL	Dell Computer Corporation	24.58 ▼	-0.170	-0.69%
ENG CX	Enterprise Growth C	14.60 ▼	-0.210	-1.42%
INTC	Intel Corporation	18.12 ▼	-0.380	-2.05%
JNJ	Johnson & Johnson	53.29 ▼	-0.290	-0.54%
KO	Coca-Cola Company	56.70 ▼	-0.580	-1.01%
MSFT	Microsoft Corporation	53.96 ▲	+1.040	+1.97%
NKE	NIKE, Inc.	57.34 ▲	+0.580	+1.02%

Single Stage												
Set 95 to 150 PSI also available, substitute "E" for "A" in model number. i.e., K15A-30 make K15E-30												
HP	Model	Tank Gal	Air Performance						Approx Ship Weight	Dimensions		
			A @ 125			E@ 150				L	W	H
			Pump RPM	CFM Disp	DEL'D Air	Pump RPM	CFM Disp	DEL'D Air				
1/2	F12A-17	17	680	3.4	2.2	590	2.9	1.6	135	37	14	25
3/4	F34A-17	17	1080	5.3	3.1	950	4.7	2.3	140	37	14	25
3/4	F34A-30	30	1080	5.3	3.1	950	4.7	2.3	160	38	16	31
1	K1A-30	30	560	6.2	4.0	500	5.7	3.1	190	38	16	34
1 1/2	K15A-30	30	870	9.8	6.2	860	9.7	5.8	205	49	20	34
1 1/2	K15A-60	60	870	9.8	6.2	860	9.7	5.8	315	38	16	34
2	K2A-30	30	1140	13.1	8.0	1060	12.0	7.0	205	49	20	39
2	K2A-60	60	1140	13.1	8.0	1060	12.0	7.0	315	48	20	34
2	GC2A-30	30	480	13.1	9.1	460	12.4	7.9	270	38	16	36
2	GC2A-60	60	480	13.1	9.1	460	12.4	7.9	370	49	20	41
3	GC3A-60	60	770	21.0	14.0	740	19.9	12.3	288	38	16	36
5	GC5A-80	60	770	21.0	14.0	740	19.9	12.3	388	49	20	41
5	GC5A-60	60	1020	27.8	17.8	910	24.6	15.0	410	49	20	41
5	GC5A-80	80	1020	27.8	17.8	910	24.6	15.0	450	62	20	41
5	J5A-80	60	780	28.7	19.0	770	28.6	18.0	570	49	23	43
5	J5A-80	80	780	28.7	19.0	770	28.6	18.0	610	63	23	43

Figure 5-57

Air Compressor Specifications

attributes for products having the two different types of Air Performance. Assume that there might be additional products with different types of Air Performance. Specify the entities, identifiers, attributes, relationships, type of category cluster, and possible determinant.

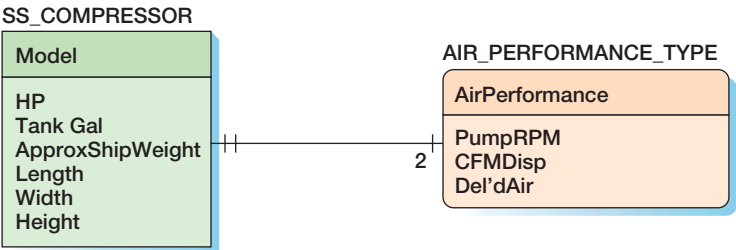
- B. Figure 5-58 shows a different model for the compressor data. Explain the entities, their type, the relationship, its type, and its cardinality. How well do you think this model fits the data shown in Figure 5-57?
- C. Compare your answer in part A with the model in Figure 5-58. What are the essential differences between the two models? Which do you think is better?
- D. Suppose you had the job of explaining the differences in these two models to a highly motivated, intelligent end user. How would you accomplish this?

5.63 Figure 5-59 shows a listing of movie times at theaters in Seattle. Using the data in this figure as an example, do the following:

- A. Create a model to represent this report using the entities MOVIE, THEATER, and SHOW_TIME. Assume that theaters may show multiple movies. Although this report is for a particular day, your data model should allow for movie times on different days as well. Specify the identifier of the entities and their attributes. Name the relationships and the type and cardinality of all relationships. Explain which cardinalities you can logically deduce from Figure 5-59 and which need to be checked out with users. Assume that distance is an attribute of THEATER.

Figure 5-58

Alternative Model for Compressor Data



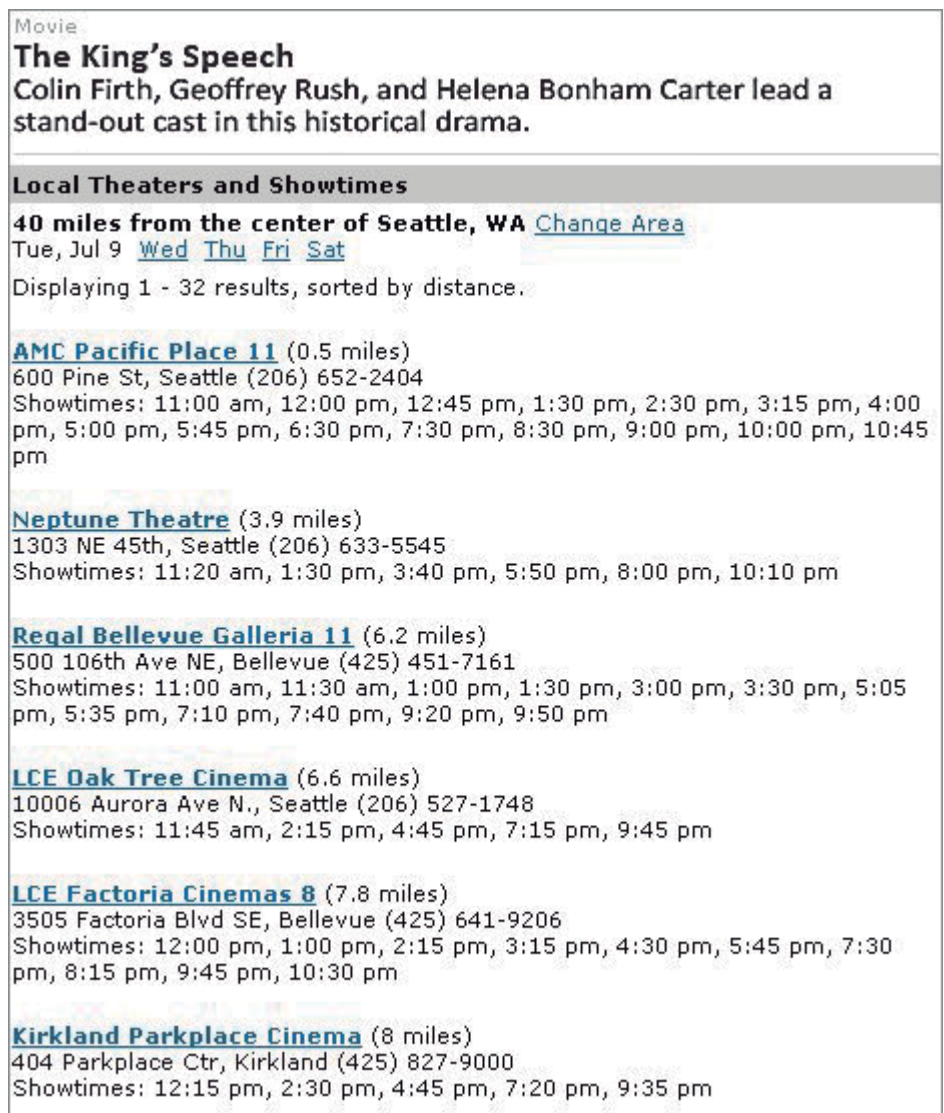


Figure 5-59
 Movie Time Listing

- B.** This report was prepared for a user who is located near downtown Seattle. Suppose that it is necessary to produce this same report for these theaters, but for a user located in a Seattle suburb, such as Bellevue, Renton, Redmond, or Tacoma. In this case, distance cannot be an attribute of THEATER. Change your answer in A for this situation. Specify the entity identifiers and attributes. Name the relationships and identify the type and cardinality of all relationships.
- C.** Suppose that you want to make this data model national. Change your answer to B so that it can be used for other metropolitan areas. Change your answer in A for this situation. Specify the entity identifiers and attributes. Name the relationships and identify the type and cardinality of all relationships.
- D.** Modify your answer to C to include the leading cast members. Assume that the role of a cast member is not to be modeled. Specify the identifier of new entities and their attributes. Name the relationships and identify the type and cardinality of all relationships.
- E.** Modify your answer to C to include the leading cast members. Assume that the role of a cast member is specified. Specify the identifier of new entities and their attributes. Name the relationships and identify the type and cardinality of all relationships.
- 5.64** Consider the three reports in Figure 5-60. The data are samples of data that would appear in the reports like these.

KELLY'S RICE Cereal Nutrition Information		
SERVING SIZE: 1 OZ. (28.4 g, ABOUT 1 CUP)		
SERVINGS PER PACKAGE: 13		
	CEREAL	WITH 1/2 CUP VITAMINS A & D SKIM MILK
CALORIES	110	150*
PROTEIN	2 g	6g
CARBOHYDRATE	25 g	31g
FAT	0 g	0g*
CHOLESTEROL	0 mg	0mg*
SODIUM	290 mg	350mg
POTASSIUM	35 mg	240mg
PERCENTAGE OF U.S. RECOMMENDED DAILY ALLOWANCES (U.S. RDA)		
PROTEIN	2	10
VITAMIN A	25	30
VITAMIN C	25	25
THIAMIN	35	40
RIBOFLAVIN	35	45
NIACIN	35	35
CALCIUM	**	15
IRON	10	10
VITAMIN D	10	25
VITAMIN B ₆	35	35
FOLIC ACID	35	35
PHOSPHORUS	4	15
MAGNESIUM	2	6
ZINC	2	6
COPPER	2	4
*WHOLE MILK SUPPLIES AN ADDITIONAL 30 CALORIES, 4 g FAT, AND 15 mg CHOLESTEROL.		
**CONTAINS LESS THAN 2% OF THE U.S. RDA OF THIS NUTRIENT.		
INGREDIENTS: RICE, SUGAR, SALT, CORN SYRUP		
VITAMINS AND IRON: VITAMIN C (SODIUM ASCORBATE AND ASCORBIC ACID), NIACINAMIDE, IRON, VITAMIN B ₆ (PY- RIDOXINE HYDROCHLORIDE), VITAMIN A (PALMITATE), VITAMIN B ₂ (RIBOFLAVIN), VITAMIN B ₁ (THIAMIN HYDROCHLORIDE), FOLIC ACID, AND VITAMIN D.		

FDA REPORT #6272	
Date: June 30, 2011	
Issuer: Kelly's Corporation	
Report Title: Product Summary by Ingredient	
Corn	Kelly's Corn Cereal Kelly's Multigrain Cereal Kelly's Crunchy Cereal
Corn syrup	Kelly's Corn Cereal Kelly's Rice Cereal Kelly's Crunchy Cereal
Malt	Kelly's Corn Cereal Kelly's Crunchy Cereal
Wheat	Kelly's Multigrain Cereal Kelly's Crunchy Cereal

(a)

SUPPLIERS LIST		
Date: June 30, 2011		
Ingredient	Supplier	Price
Corn	Wilson	2.80
	J. Perkins	2.72
	Pollack	2.83
	McKay	2.80
Wheat	Adams	1.19
	Kroner	1.19
	Schmidt	1.22
Barley	Wilson	0.85
	Pollack	0.84

(b)

Figure 5-60

Cereal Product Reports

- Make a list of as many potential entities as these reports suggest.
- Examine your list to determine whether any entities are synonyms. If so, consolidate your list.
- Construct an IE Crow's Foot model showing relationships among your entities. Name each relationship and specify cardinalities. Indicate which cardinalities you can justify on the basis of these reports and which you will need to check out with the users.

5.65 Consider the CD cover in Figure 5-61.

- Specify identifiers and attributes for the entities CD, ARTIST, ROLE, and SONG.
- Construct a crow's foot model showing relationships among these four entities. Name each relationship and specify cardinalities. Indicate which cardinalities you can justify on the basis of the CD cover and which you will need to check out with the users.

<p>West Side Story Based on a conception of Jerome Robbins</p> <p>Book by ARTHUR LAURENTS Music by LEONARD BERNSTEIN Lyrics by STEPHEN SONDHEIM</p> <p>Entire Original Production Directed and Choreographed by JEROME ROBBINS</p> <p>Originally produced on Broadway by Robert E. Griffith and Harold S. Prince by arrangement with Roger L. Stevens Orchestration by Leonard Bernstein with Sid Ramin and Irwin Kostal</p> <p>HIGHLIGHTS FROM THE COMPLETE RECORDING</p> <p>Maria KIRI TE KANAWA Tony JOSE CARRERAS Anita TATIANA TROYANOS Riff KURT OLLMAN and MARILYN HORNE singing “Somewhere”</p> <p>Rosalia Louise Edeiken Diesel Marty Nelson Consuela Stella Zambalis Baby John Stephen Bogardus Fancisca Angelina Reaux A-rab Peter Thom Action David Livingston Snowboy Todd Lester Bernardo Richard Harrell</p>	
1	Jet Song [3'13] (Riff, Action, Baby John, A-rab, Chorus)
2	Something's Coming [2'33] (Tony)
3	Maria [2'56] (Tony)
4	Tonight [5'27] (Maria, Tony)
5	America [4'47] (Anita, Rosalia, Chorus)
6	Cool [4'37] (Riff, Chorus)
7	One Hand, One Heart [5'38] (Tony, Maria)
8	Tonight (Ensemble) [3'40] (Entire Cast)
9	I Feel Pretty [3'22] (Maria, Chorus)
10	Somewhere [2'34] (A Girl)
11	Gee Officer Krupke [4'18] (Action, Snowboy, Diesel, A-rab, Baby John, Chorus)
12	A Boy Like That [2'05] (Anita, Maria)
13	I Have a Love [3'30] (Maria, Anita)
14	Taunting Scene [1'21] (Orchestra)
15	Finale [2'40] (Maria, Tony)

Figure 5-61

CD Cover

- C. Consider a CD that does not involve a musical, so there is no need for ROLE. However, the entity SONG_WRITER is needed. Create a crow's foot model for CD, ARTIST, SONG, and SONG_WRITER. Assume that an ARTIST can either be a group or an individual. Assume that some artists record individually and as part of a group.
- D. Combine the models you developed in your answers to B and C. Create new entities if necessary, but strive to keep your model as simple as possible. Specify identifiers and attributes of new entities, name new relationships, and indicate their cardinalities.

5.66 Consider the data model in Figure 5-43. How should this model be altered if the users want to keep track of how many of each part are used? Suppose, for example, that the wheel assembly requires four washers and the handle assembly requires just one, and the database must store these quantities. (Hint: Adding Quantity to this N:M relationship is analogous to adding Price to the N:M relationship in Figure 5-22.)

5.67 The data model in Figure 5-52 uses the attribute Room in COLLEGE and DEPARTMENT, but uses OfficeNumber in PROFESSOR. These attributes have the same kind of data, even though they have different names. Examine Figure 5-46 and explain how this situation came to be. Do you think having different names for the same attribute types is rare? Do you think it is a problem? Why or why not?

**Marcia's
Dry
Cleaning**



Suppose that you have been hired by Marcia's Dry Cleaning to create a database application to track customers, orders, and items. Marcia also wants to start a Frequent Cleaner's Club, whereby she will offer a 50 percent discount on every 10th customer order.

- A. Using your knowledge, create a data model for Marcia's business. Name each entity, describe its type, and indicate all attributes and identifiers. Name each relationship, describe its type, and specify minimum and maximum cardinalities.
- B. List any item in your answer to A that you believe should be checked out with Marcia and/or her employees.

**Morgan
Importing**



Suppose that you have been hired by Morgan Importing to create a database application to track stores, purchases, shipments, and shippers. Sometimes several items are purchased from a store on a single visit, but do not assume that all of the items are placed on the same shipment. You want to track each item in a shipment and assign an insurance value to each item.

- A. Using your knowledge, create a data model for Morgan Importing. Name each entity, describe its type, and indicate all attributes and identifiers. Name each relationship, describe its type, and specify minimum and maximum cardinalities.
- B. List any item in your answer to A that you believe should be checked out with Phillip Morgan and/or his employees.