CHAPTER

2

Modeling Data in the Organization

Visit www.pearsonhighered.com/hoffer to view the accompanying video for this chapter.

Learning Objectives

After studying this chapter, you should be able to:

- Concisely define each of the following key terms: **business rule, term, fact, entity-relationship model (E-R model), entity-relationship diagram (E-R diagram), entity, entity type, entity instance, strong entity type, weak entity type, identifying owner, identifying relationship, attribute, required attribute, optional attribute, composite attribute, simple attribute, multivalued attribute, derived attribute, identifier, composite identifier, relationship type, relationship instance, associative entity, degree, unary relationship, binary relationship, ternary relationship, cardinality constraint, minimum cardinality, maximum cardinality,** and **time stamp**.
- State reasons why many system developers believe that data modeling is the most important part of the systems development process.
- Write good names and definitions for entities, relationships, and attributes.
- Distinguish unary, binary, and ternary relationships and give a common example of each.
- Model each of the following constructs in an E-R diagram: composite attribute, multivalued attribute, derived attribute, associative entity, identifying relationship, and minimum and maximum cardinality constraints.
- Draw an E-R diagram to represent common business situations.
- Convert a many-to-many relationship to an associative entity type.
- Model simple time-dependent data using time stamps and relationships in an E-R diagram.

INTRODUCTION

You have already been introduced to modeling data and the entity-relationship (E-R) data model through simplified examples in Chapter 1. (You may want to review, for example, the E-R models in Figures 1-3 and 1-4.) In this chapter, we formalize data modeling based on the powerful concept of business rules and describe the E-R data model in detail. This chapter begins your journey of learning how to design and use databases. It is exciting to create information systems that run organizations and help people do their jobs well.

Business rules, the foundation of data models, are derived from policies, procedures, events, functions, and other business objects, and they state constraints on the organization. Business rules represent the language and fundamental structure of an organization (Hay, 2003). Business rules formalize the understanding of the organization by organization owners, managers, and leaders with that of information systems architects.

Business rules are important in data modeling because they govern how data are handled and stored. Examples of basic business rules are data names and definitions. This chapter explains guidelines for the clear naming and definition of data objects in a business. In terms of conceptual data modeling, names and definitions must be provided for the main data objects: entity types (e.g., Customer), attributes (Customer Name), and relationships (Customer Places Orders). Other business rules may state constraints on these data objects. These constraints can be captured in a data model, such as an entity-relationship diagram, and associated documentation. Additional business rules govern the people, places, events, processes, networks, and objectives of the organization, which are all linked to the data requirements through other system documentation.

After decades of use, the E-R model remains the mainstream approach for conceptual data modeling. Its popularity stems from factors such as relative ease of use, widespread computer-aided software engineering (CASE) tool support, and the belief that entities and relationships are natural modeling concepts in the real world.

The E-R model is most used as a tool for communications between database designers and end users during the analysis phase of database development (described in Chapter 1). The E-R model is used to construct a conceptual data model, which is a representation of the structure and constraints of a database that is independent of software (such as a database management system).

Some authors introduce terms and concepts peculiar to the relational data model when discussing E-R modeling; the relational data model is the basis for most database management systems in use today. In particular, they recommend that the E-R model be completely normalized, with full resolution of primary and foreign keys. However, we believe that this forces a premature commitment to the relational data model. In today's database environment, the database may be implemented with object-oriented technology or with a mixture of object-oriented and relational technology. Therefore, we defer discussion of normalization concepts to Chapter 4.

The E-R model was introduced in a key article by Chen (1976), in which he described the main constructs of the E-R model—entities and relationships—and their associated attributes. The model has subsequently been extended to include additional constructs by Chen and others; for example, see Teorey et al. (1986) and Storey (1991). The E-R model continues to evolve, but unfortunately there is not yet a standard notation for E-R modeling. Song et al. (1995) present a side-by-side comparison of 10 different E-R modeling notations, explaining the major advantages and disadvantages of each approach. Because data modeling software tools are now commonly used by professional data modelers, we adopt for use in this text a variation of the notation used in professional modeling tools. Appendix A will help you translate between our notation and other popular E-R diagramming notations.

As said in a popular travel service TV commercial, "we are doing important stuff here." Many systems developers believe that data modeling is the most important part of the systems development process for the following reasons (Hoffer et al., 2010):

1. The characteristics of data captured during data modeling are crucial in the design of databases, programs, and other system components. The facts and rules captured during the process of data modeling are essential in assuring data integrity in an information system.

2. Data rather than processes are the most complex aspect of many modern information systems and hence require a central role in structuring system requirements. Often the goal is to provide a rich data resource that might support any type of information inquiry, analysis, and summary.
3. Data tend to be more stable than the business processes that use that data. Thus, an information system design that is based on a data orientation should have a longer useful life than one based on a process orientation.

In an actual work environment, you may not have to develop a data model from scratch. Because of the increased acceptance of packaged software (for example, enterprise resource planning with a predefined data model) and purchased business area or industry data models (which we discuss in Chapter 3), your job of data modeling has a jump start. This is good because such components and patterns give you a starting point based on generally accepted practices. However, your job is not done for several reasons:

1. There are still many times when a new, custom-built application is being developed along with the associated database. The business rules for the business area supported by this application need to be modeled.
2. Purchased applications and data models need to be customized for your particular setting. Predefined data models tend to be very extensive and complex; hence, they require significant data modeling skill to tailor the models to be effective and efficient in a given organization. Although this effort can be much faster, thorough, and accurate than starting from scratch, the ability to understand a particular organization to match the data model to its business rules is an essential task.

In this chapter, we present the main features of E-R modeling, using common notation and conventions. We begin with a sample E-R diagram, including the basic constructs of the E-R model—entities, attributes, and relationships—and then we introduce the concept of business rules, which is the foundation for all the data modeling constructs. We define three types of entities that are common in E-R modeling: strong entities, weak entities, and associative entities; a few more entity types are defined in Chapter 3. We also define several important types of attributes, including required and optional attributes, single- and multivalued attributes, derived attributes, and composite attributes. We then introduce three important concepts associated with relationships: the degree of a relationship, the cardinality of a relationship, and participation constraints in a relationship. We conclude with an extended example of an E-R diagram for Pine Valley Furniture Company.

THE E-R MODEL: AN OVERVIEW

An **entity-relationship model (E-R model)** is a detailed, logical representation of the data for an organization or for a business area. The E-R model is expressed in terms of entities in the business environment, the relationships (or associations) among those entities, and the attributes (or properties) of both the entities and their relationships. An E-R model is normally expressed as an **entity-relationship diagram (E-R diagram, or ERD)**, which is a graphical representation of an E-R model.

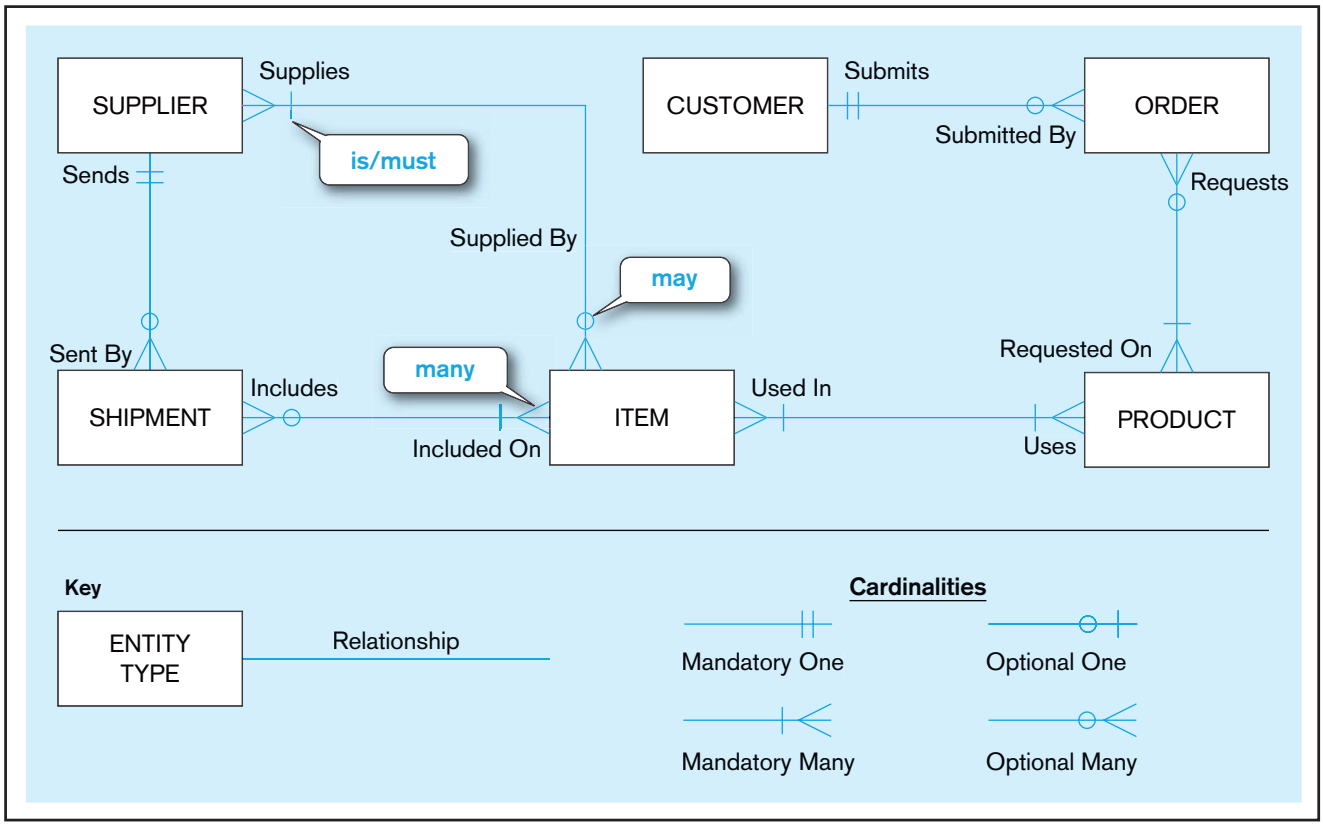
Entity-relationship model (E-R model)
A logical representation of the data for an organization or for a business area, using entities for categories of data and relationships for associations between entities.

Entity-relationship diagram (E-R diagram, or ERD)
A graphical representation of an entity-relationship model.

Sample E-R Diagram

To jump-start your understanding of E-R diagrams, Figure 2-1 presents a simplified E-R diagram for a small furniture manufacturing company, Pine Valley Furniture Company. (This figure, which does not include attributes, is often called an *enterprise data model*, which we introduced Chapter 1.) A number of suppliers supply and ship different items to Pine Valley Furniture. The items are assembled into products that are sold to customers who order the products. Each customer order may include one or more lines corresponding to the products appearing on that order.

FIGURE 2-1 Sample E-R diagram



The diagram in Figure 2-1 shows the entities and relationships for this company. (Attributes are omitted to simplify the diagram for now.) Entities (the objects of the organization) are represented by the rectangle symbol, whereas relationships between entities are represented by lines connecting the related entities. The entities in Figure 2-1 are:

CUSTOMER	A person or an organization that has ordered or might order products. *Example*: L. L. Fish Furniture.
PRODUCT	A type of furniture made by Pine Valley Furniture that may be ordered by customers. Note that a product is not a specific bookcase, because individual bookcases do not need to be tracked. *Example*: A 6-foot, 5-shelf, oak bookcase called O600.
ORDER	The transaction associated with the sale of one or more products to a customer and identified by a transaction number from sales or accounting. *Example*: The event of L. L. Fish buying one product O600 and four products O623 on September 10, 2010.
ITEM	A type of component that goes into making one or more products and can be supplied by one or more suppliers. *Example*: A 4-inch ball-bearing caster called I-27–4375.
SUPPLIER	Another company that may provide items to Pine Valley Furniture. *Example*: Sure Fasteners, Inc.
SHIPMENT	The transaction associated with items received in the same package by Pine Valley Furniture from a supplier. All items in a shipment appear on one bill-of-lading document. *Example*: The receipt of 300 I-27-4375 and 200 I-27-4380 items from Sure Fasteners, Inc., on September 9, 2010.

Note that it is important to clearly define, as metadata, each entity. For example, it is important to know that the CUSTOMER entity includes persons or organizations that have not yet purchased products from Pine Valley Furniture. It is common for different departments in an organization to have different meanings for the same term (homonyms). For example, Accounting may designate as customers only those persons or organizations who have ever made a purchase, thus excluding potential customers, whereas Marketing designates as customers anyone they have contacted or who has purchased from Pine Valley Furniture or any known competitor. An accurate and thorough ERD without clear metadata may be interpreted in different ways by different people. We outline good naming and definition conventions as we formally introduce E-R modeling throughout this chapter.

The symbols at the end of each line on an ERD specify relationship cardinalities, which represent how many entities of one kind relate to how many entities of another kind. On examining Figure 2-1, we can see that these cardinality symbols express the following business rules:

1. A SUPPLIER may supply many ITEMs (by "may supply," we mean the supplier may not supply any items). Each ITEM is supplied by any number of SUPPLIERs (by "is supplied," we mean that the item must be supplied by at least one supplier). See annotations in Figure 2-1 that correspond to underlined words.
2. Each ITEM must be used in the assembly of at least one PRODUCT and may be used in many products. Conversely, each PRODUCT must use one or more ITEMs.
3. A SUPPLIER may send many SHIPMENTs. However, each shipment must be sent by exactly one SUPPLIER. Notice that sends and supplies are separate concepts. A SUPPLIER may be able to supply an item, but may not yet have sent any shipments of that item.
4. A SHIPMENT must include one (or more) ITEMs. An ITEM may be included on several SHIPMENTs.
5. A CUSTOMER may submit any number of ORDERs. However, each ORDER must be submitted by exactly one CUSTOMER. Given that a CUSTOMER may not have submitted any ORDERs, some CUSTOMERs must be potential, inactive, or some other customer possibly without any related ORDERs.
6. An ORDER must request one (or more) PRODUCTs. A given PRODUCT may not be requested on any ORDER, or may be requested on one or more orders.

There are actually two business rules for each relationship, one for each direction from one entity to the other. Note that each of these business rules roughly follows a certain grammar:

```
<entity> <minimum cardinality> <relationship> <maximum cardinality> <entity>
```

For example, rule 5 is:

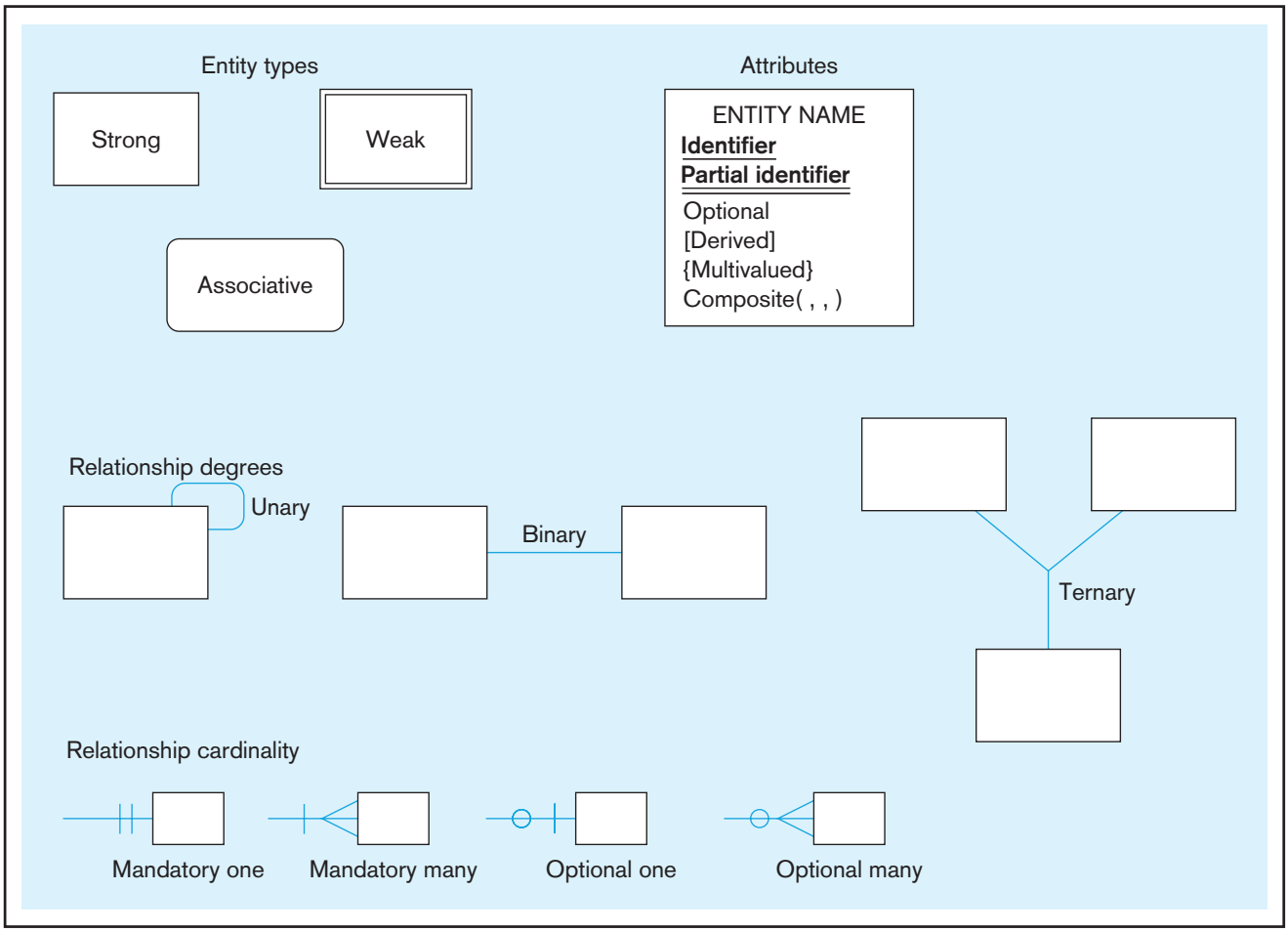
```
<CUSTOMER> <may> <Submit> <any number> <ORDER>
```

This grammar gives you a standard way to put each relationship into a natural English business rule statement.

E-R Model Notation

The notation we use for E-R diagrams is shown in Figure 2-2. As indicated in the previous section, there is no industry-standard notation (in fact, you saw a slightly simpler notation in Chapter 1). The notation in Figure 2-2 combines most of the desirable features of the different notations that are commonly used in E-R drawing tools today and also allows us to model accurately most situations that are encountered in practice.

FIGURE 2-2 **Basic E-R notation**



We introduce additional notation for enhanced entity-relationship models (including class-subclass relationships) in Chapter 3.

In many situations, however, a simpler E-R notation is sufficient. Most drawing tools, either stand-alone ones such as Microsoft Visio or those in CASE tools such as Oracle Designer, CA ERwin, or PowerDesigner, do not show all the entity and attribute types we use. It is important to note that any notation requires special annotations, not always present in a diagramming tool, to show all the business rules of the organizational situation you are modeling. We will use the Visio notation for a few examples throughout the chapter and at the end of the chapter so that you can see the differences. Appendix A llustrates the E-R notation from several commonly used guidelines and diagramming tools. This appendix may help you translate between the notations in the text and the notations you use in classes.

MODELING THE RULES OF THE ORGANIZATION

Now that you have an example of a data model in mind, let's step back and consider more generally what a data model is representing. We will see in this and the next chapter how to use data models, in particular the entity-relationship notation, to document rules and policies of an organization. *In fact, documenting rules and policies of an organization that govern data is exactly what data modeling is all about*. Business rules and policies govern creating, updating, and removing data in an information processing and storage system; thus they must be described along with the data to which they are related. For example, the policy "every student in the university must have a faculty adviser" forces data (in a database) about each student to be associated with data about some student adviser. Also, the statement "a student is any person who has applied for admission or

taken a course or training program from any credit or noncredit unit of the university" not only defines the concept of "student" but also states a policy of the university (e.g., implicitly, alumni are students, and a high school student who attended a college fair but has not applied is not a student, assuming the college fair is not a noncredit training program).

Business rules and policies are not universal; different universities may have different policies for student advising and may include different types of people as students. Also, the rules and policies of an organization may change (usually slowly) over time; a university may decide that a student does not have to be assigned a faculty adviser until the student chooses a major.

Your job as a database analyst is to

- Identify and understand those rules *that govern data*
- Represent those rules so that they can be unambiguously understood by information systems developers and users
- Implement those rules in database technology

Data modeling is an important tool in this process. Because the purpose of data modeling is to document business rules about data, we introduce the discussion of data modeling and the entity-relationship notation with an overview of business rules. Data models cannot represent all business rules (and do not need to, because not all business rules govern data); data models along with associated documentation and other types of information system models (e.g., models that document the processing of data) represent all business rules that must be enforced through information systems.

Overview of Business Rules

A **business rule** is "a statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business . . . rules prevent, cause, or suggest things to happen" (GUIDE Business Rules Project, 1997). For example, the following two statements are common expressions of business rules that affect data processing and storage:

Business rule
A statement that defines or constrains some aspect of the business. It is intended to assert business structure or to control or influence the behavior of the business.

- "A student may register for a section of a course only if he or she has successfully completed the prerequisites for that course."
- "A preferred customer qualifies for a 10 percent discount, unless he has an overdue account balance."

Most organizations (and their employees) today are guided by thousands of combinations of such rules. In the aggregate, these rules influence behavior and determine how the organization responds to its environment (Gottesdiener, 1997; von Halle, 1997). Capturing and documenting business rules is an important, complex task. Thoroughly capturing and structuring business rules, then enforcing them through database technologies, helps to ensure that information systems work right and that users of the information understand what they enter and see.

THE BUSINESS RULES PARADIGM The concept of business rules has been used in information systems for some time. There are many software products that help organizations manage their business rules (for example, JRules from ILOG, an IBM company). In the database world, it has been more common to use the related term *integrity constraint* when referring to such rules. The intent of this term is somewhat more limited in scope, usually referring to maintaining valid data values and relationships in the database.

A business rules approach is based on the following premises:

- Business rules are a core concept in an enterprise because they are an expression of business policy and guide individual and aggregate behavior. Well-structured business rules can be stated in natural language for end users and in a data model for systems developers.

- Business rules can be expressed in terms that are familiar to end users. Thus, users can define and then maintain their own rules.
- Business rules are highly maintainable. They are stored in a central repository, and each rule is expressed only once, then shared throughout the organization. Each rule is discovered and documented only once, to be applied in all systems development projects.
- Enforcement of business rules can be automated through the use of software that can interpret the rules and enforce them using the integrity mechanisms of the database management system (Moriarty, 2000).

Although much progress has been made, the industry has not realized all of these objectives to date (Owen, 2004). Possibly the premise with greatest potential benefit is "Business rules are highly maintainable." The ability to specify and maintain the requirements for information systems as a set of rules has considerable power when coupled with an ability to generate automatically information systems from a repository of rules. Automatic generation and maintenance of systems will not only simplify the systems development process but also will improve the quality of systems.

Scope of Business Rules

In this chapter and the next, we are concerned with business rules that impact only an organization's databases. Most organizations have a host of rules and/or policies that fall outside this definition. For example, the rule "Friday is business casual dress day" may be an important policy statement, but it has no immediate impact on databases. In contrast, the rule "A student may register for a section of a course only if he or she has successfully completed the prerequisites for that course" is within our scope because it constrains the transactions that may be processed against the database. In particular, it causes any transaction to be rejected that attempts to register a student who does not have the necessary prerequisites. Some business rules cannot be represented in common data modeling notation; those rules that cannot be represented in a variation of an entity-relationship diagram are stated in natural language, and some can be represented in the relational data model, which we describe in Chapter 4.

GOOD BUSINESS RULES Whether stated in natural language, a structured data model, or other information systems documentation, a business rule will have certain characteristics if it is to be consistent with the premises outlined previously. These characteristics are summarized in Table 2-1. These characteristics will have a better chance of being satisfied if a business rule is defined, approved, and owned by business, not technical, people. Businesspeople become stewards of the business rules. You, as the database analyst, facilitate the surfacing of the rules and the transformation of ill-stated rules into ones that satisfy the desired characteristics.

GATHERING BUSINESS RULES Business rules appear (possibly implicitly) in descriptions of business functions, events, policies, units, stakeholders, and other objects. These descriptions can be found in interview notes from individual and group information systems requirements collection sessions, organizational documents (e.g., personnel manuals, policies, contracts, marketing brochures, and technical instructions), and other sources. Rules are identified by asking questions about the who, what, when, where, why, and how of the organization. Usually, a data analyst has to be persistent in clarifying initial statements of rules because initial statements may be vague or imprecise (what some people have called "business ramblings"). Thus, precise rules are formulated from an iterative inquiry process. You should be prepared to ask such questions as "Is this always true?" "Are there special circumstances when an alternative occurs?" "Are there distinct kinds of that person?" "Is there only one of those or are there many?" and "Is there a need to keep a history of those, or is the current data all that is useful?" Such questions can be useful for surfacing rules for each type of data modeling construct we introduce in this chapter and the next.

TABLE 2-1 Characteristics of a Good Business Rule

Characteristic	Explanation
Declarative	A business rule is a statement of policy, not how policy is enforced or conducted; the rule does not describe a process or implementation, but rather describes what a process validates.
Precise	With the related organization, the rule must have only one interpretation among all interested people, and its meaning must be clear.
Atomic	A business rule marks one statement, not several; no part of the rule can stand on its own as a rule (that is, the rule is indivisible, yet sufficient).
Consistent	A business rule must be internally consistent (that is, not contain conflicting statements) and must be consistent with (and not contradict) other rules.
Expressible	A business rule must be able to be stated in natural language, but it will be stated in a structured natural language so that there is no misinterpretation.
Distinct	Business rules are not redundant, but a business rule may refer to other rules (especially to definitions).
Business-oriented	A business rule is stated in terms businesspeople can understand, and because it is a statement of business policy, only businesspeople can modify or invalidate a rule; thus, a business rule is owned by the business.

Source: Based on Gottesdiener (1999) and Plotkin (1999).

Data Names and Definitions

Fundamental to understanding and modeling data are naming and defining data objects. Data objects must be named and defined before they can be used unambiguously in a model of organizational data. In the entity-relationship notation you will learn in this chapter, you have to give entities, relationships, and attributes clear and distinct names and definitions.

DATA NAMES We will provide specific guidelines for naming entities, relationships, and attributes as we develop the entity-relationship data model, but there are some general guidelines about naming any data object. Data names should (Salin, 1990; ISO/IEC, 2005)

- ***Relate to business, not technical (hardware or software), characteristics;*** so, Customer is a good name, but File10, Bit7, and Payroll Report Sort Key are not good names.
- ***Be meaningful,*** almost to the point of being self-documenting (i.e., the definition will refine and explain the name without having to state the essence of the object's meaning); you should avoid using generic words such as *has*, *is*, *person*, or *it*.
- ***Be unique*** from the name used for every other distinct data object; words should be included in a data name if they distinguish the data object from other similar data objects (e.g., Home Address versus Campus Address).
- ***Be readable,*** so that the name is structured as the concept would most naturally be said (e.g., Grade Point Average is a good name, whereas Average Grade Relative To A, although possibly accurate, is an awkward name).
- ***Be composed of words taken from an approved list;*** each organization often chooses a vocabulary from which significant words in data names must be chosen (e.g., maximum is preferred, never upper limit, ceiling, or highest); alternative, or alias names, also can be used as can approved abbreviations (e.g., CUST for CUSTOMER), and you may be encouraged to use the abbreviations so that data names are short enough to meet maximum length limits of database technology.
- ***Be repeatable,*** meaning that different people or the same person at different times should develop exactly or almost the same name; this often means that there is a standard hierarchy or pattern for names (e.g., the birth date of a student

would be Student Birth Date and the birth date of an employee would be Employee Birth Date).
- ***Follow a standard syntax,*** meaning that the parts of the name should follow a standard arrangement adopted by the organization.

Salin (1990) suggests that you develop data names by

1. Preparing a definition of the data. (We talk about definitions next.)
2. Removing insignificant or illegal words (words not on the approved list for names); note that the presence of AND and OR in the definition may imply that two or more data objects are combined, and you may want to separate the objects and assign different names.
3. Arranging the words in a meaningful, repeatable way.
4. Assigning a standard abbreviation for each word.
5. Determining whether the name already exists, and if so, adding other qualifiers that make the name unique.

We will see examples of good data names as we develop a data modeling notation in this chapter.

DATA DEFINITIONS A definition (sometimes called a *structural assertion*) is considered a type of business rule (GUIDE Business Rules Project, 1997). A definition is an explanation of a term or a fact. A **term** is a word or phrase that has a specific meaning for the business. Examples of terms are *course*, *section*, *rental car*, *flight*, *reservation*, and *passenger*. Terms are often the key words used to form data names. Terms must be defined carefully and concisely. However, there is no need to define common terms such as *day*, *month*, *person*, or *television*, because these terms are understood without ambiguity by most persons.

Term
A word or phrase that has a specific meaning for the business.

A **fact** is an association between two or more terms. A fact is documented as a simple declarative statement that relates terms. Examples of facts that are definitions are the following (the defined terms are underlined):

Fact
An association between two or more terms.

- "A course is a module of instruction in a particular subject area." This definition associates two terms: *module of instruction* and *subject area*. We assume that these are common terms that do not need to be further defined.
- "A customer may request a model of car from a rental branch on a particular date." This fact, which is a definition of *model rental request*, associates the four underlined terms (GUIDE Business Rules Project, 1997). Three of these terms are business- specific terms that would need to be defined individually (date is a common term).

A fact statement places no constraints on instances of the fact. For example, it is inappropriate in the second fact statement to add that a customer may not request two different car models on the same date. Such constraints are separate business rules.

GOOD DATA DEFINITIONS We will illustrate good definitions for entities, relationships, and attributes as we develop the entity-relationship notation in this and the next chapters. There are, however, some general guidelines to follow (Aranow, 1989; ISO/IEC, 2004):

- Definitions (and all other types of business rules) are gathered from the same sources as all requirements for information systems. Thus, systems and data analysts should be looking for data objects and their definitions as these sources of information systems requirements are studied.
- Definitions will usually be accompanied by diagrams, such as entity-relationship diagrams. The definition does not need to repeat what is shown on the diagram but rather supplement the diagram.
- Definitions will be stated in the singular and explain what the data is, not what it is not. A definition will use commonly understood terms and abbreviations and stand alone in its meaning and not embed other definitions within it. It should be concise and concentrate on the essential meaning of the data, but it may also state such characteristics of a data object as
  - Subtleties
  - Special or exceptional conditions

  - Examples
  - Where, when, and how the data are created or calculated in the organization
  - Whether the data are static or changes over time
  - Whether the data are singular or plural in its atomic form
  - Who determines the value for the data
  - Who owns the data (i.e., who controls the definition and usage)
  - Whether the data are optional or whether empty (what we will call null) values are allowed
  - Whether the data can be broken down into more atomic parts or are often combined with other data into some more composite or aggregate form

  If not included in a data definition, these characteristics need to be documented elsewhere, where other metadata are stored.
- A data object should not be added to a data model, such as an entity-relationship diagram, until after it has been carefully defined (and named) and there is agreement on this definition. But expect the definition of the data to change once you place the object on the diagram because the process of developing a data model tests your understanding of the meaning of data. (In other words, *modeling data is an iterative process*.)

There is an unattributed phrase in data modeling that highlights the importance of good data definitions: "He who controls the meaning of data controls the data." It might seem that obtaining concurrence in an organization on the definitions to be used for the various terms and facts should be relatively easy. However, this is usually far from the case. In fact, it is likely to be one of the most difficult challenges you will face in data modeling or, for that matter, in any other endeavor. It is not unusual for an organization to have multiple definitions (perhaps a dozen or more) for common terms such as *customer* or *order*.

To illustrate the problems inherent in developing definitions, consider a data object of Student found in a typical university. A sample definition for Student is "a person who has been admitted to the school and who has registered for at least one course during the past year." This definition is certain to be challenged, because it is probably too narrow. A person who is a student typically proceeds through several stages in relationship with the school, such as the following:

1. Prospect—some formal contact, indicating an interest in the school
2. Applicant—applies for admission
3. Admitted applicant—admitted to the school and perhaps to a degree program
4. Matriculated student—registers for at least one course
5. Continuing student—registers for courses on an ongoing basis (no substantial gaps)
6. Former student—fails to register for courses during some stipulated period (now may reapply)
7. Graduate—satisfactorily completes some degree program (now may apply for another program)

Imagine the difficulty of obtaining consensus on a single definition in this situation! It would seem you might consider three alternatives:

1. ***Use multiple definitions to cover the various situations.*** This is likely to be highly confusing if there is only one entity type, so this approach is not recommended (multiple definitions are not good definitions). It might be possible to create multiple entity types, one for each student situation. However, because there is likely considerable similarity across the entity types, the fine distinctions between the entity types may be confusing, and the data model will show many constructs.
2. ***Use a very general definition that will cover most situations.*** This approach may necessitate adding additional data about students to record a given student's actual status. For example, data for a student's status, with values of prospect, applicant, and so forth might be sufficient. On the other hand, if the same student could hold multiple statuses (e.g., prospect for one degree and matriculated for another degree), this might not work.

3. ***Consider using multiple, related, data objects for Student.*** For example, we could create a general entity type for Student and then other specific entity types for kinds of students with unique characteristics. We describe the conditions that suggest this approach in Chapter 3.

MODELING ENTITIES AND ATTRIBUTES

The basic constructs of the E-R model are entities, relationships, and attributes. As shown in Figure 2-2, the model allows numerous variations for each of these constructs. The richness of the E-R model allows designers to model real-world situations accurately and expressively, which helps account for the popularity of the model.

Entities

Entity
A person, a place, an object, an event, or a concept in the user environment about which the organization wishes to maintain data.

An **entity** is a person, a place, an object, an event, or a concept in the user environment about which the organization wishes to maintain data. Thus, an entity has a noun name. Some examples of each of these *kinds* of entities follow:

Person:	EMPLOYEE, STUDENT, PATIENT
Place:	STORE, WAREHOUSE, STATE
Object:	MACHINE, BUILDING, AUTOMOBILE
Event:	SALE, REGISTRATION, RENEWAL
Concept:	ACCOUNT, COURSE, WORK CENTER

ENTITY TYPE VERSUS ENTITY INSTANCE There is an important distinction between entity types and entity instances. An **entity type** is a collection of entities that share common properties or characteristics. Each entity type in an E-R model is given a name. Because the name represents a collection (or set) of items, it is always singular. We use capital letters for names of entity type(s). In an E-R diagram, the entity name is placed inside the box representing the entity type (see Figure 2-1).

Entity type
A collection of entities that share common properties or characteristics.

Entity instance
A single occurrence of an entity type.

An **entity instance** is a single occurrence of an entity type. Figure 2-3 illustrates the distinction between an entity type and two of its instances. An entity type is described just once (using metadata) in a database, whereas many instances of that entity type may be represented by data stored in the database. For example, there is one EMPLOYEE entity type in most organizations, but there may be hundreds (or even thousands) of instances of this entity type stored in the database. We often use the single term *entity* rather than *entity instance* when the meaning is clear from the context of our discussion.

Entity type: EMPLOYEE

Attributes	Attribute Data Type	Example Instance	Example Instance
Employee Number	CHAR (10)	642-17-8360	534-10-1971
Name	CHAR (25)	Michelle Brady	David Johnson
Address	CHAR (30)	100 Pacific Avenue	450 Redwood Drive
City	CHAR (20)	San Francisco	Redwood City
State	CHAR (2)	CA	CA
Zip Code	CHAR (9)	98173	97142
Date Hired	DATE	03-21-1992	08-16-1994
Birth Date	DATE	06-19-1968	09-04-1975

FIGURE 2-3 **Entity type EMPLOYEE with two instances**

ENTITY TYPE VERSUS SYSTEM INPUT, OUTPUT, OR USER A common mistake people make when they are learning to draw E-R diagrams, especially if they are already familiar with data process modeling (such as data flow diagramming), is to confuse data entities with other elements of an overall information systems model. A simple rule to avoid such confusion is that a true data entity will have many possible instances, each with a distinguishing characteristic, as well as one or more other descriptive pieces of data.

Consider Figure 2-4a, which might be drawn to represent a database needed for a college sorority's expense system. (For simplicity in this and some other figures, we show only one name for a relationship.) In this situation, the sorority treasurer manages accounts, receives expense reports, and records expense transactions against each account. However, do we need to keep track of data about the Treasurer (the TREASURER entity type) and her supervision of accounts (the Manages relationship) and receipt of reports (the Receives relationship)? The Treasurer is the person entering data about accounts and expenses and receiving expense reports. That is, she is a user of the database. Because there is only one Treasurer, TREASURER data do not need to be kept. Further, is the EXPENSE REPORT entity necessary? Because an expense report is computed from expense transactions and account balances, it is the result of extracting data from the database and received by the Treasurer. Even though there will be multiple instances of expense reports given to the Treasurer over time, data needed to compute the report contents each time are already represented by the ACCOUNT and EXPENSE entity types.

Another key to understanding why the ERD in Figure 2-4a might be in error is the nature of the *relationship names*, Receives and Summarizes. These relationship names refer to business activities that transfer or translate data, not to simply the association of one kind of data with another kind of data. The simple E-R diagram in Figure 2-4b shows entities and a relationship that would be sufficient to handle the sorority expense system as described here. See Problem and Exercise 19 for a variation on this situation.

STRONG VERSUS WEAK ENTITY TYPES Most of the basic entity types to identify in an organization are classified as strong entity types. A **strong entity type** is one that exists independently of other entity types. (Some data modeling software, in fact, use the term *independent entity*.) Examples include STUDENT, EMPLOYEE, AUTOMOBILE, and COURSE.

Strong entity type
An entity that exists independently of other entity types.

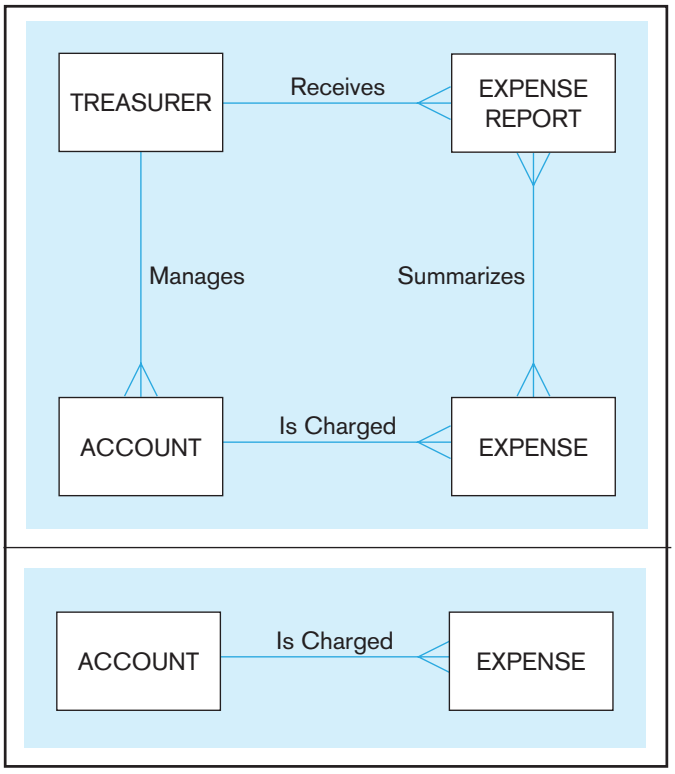


FIGURE 2-4 Example of inappropriate entities (a) System user (Treasurer) and output (Expense Report) shown as entities

(b) E-R diagram with only the necessary entities

Instances of a strong entity type always have a unique characteristic (called an *identifier*)—that is, an attribute or a combination of attributes that uniquely distinguish each occurrence of that entity.

Weak entity type
An entity type whose existence depends on some other entity type.

In contrast, a **weak entity type** is an entity type whose existence depends on some other entity type. (Some data modeling software, in fact, use the term *dependent entity*.) A weak entity type has no business meaning in an E-R diagram without the entity on which it depends. The entity type on which the weak entity type depends is called the **identifying owner** (or simply *owner* for short). A weak entity type does not typically have its own identifier. Generally, on an E-R diagram, a weak entity type has an attribute that serves as a *partial* identifier. During a later design stage (described in Chapter 4), a full identifier will be formed for the weak entity by combining the partial identifier with the identifier of its owner or by creating a surrogate identifier attribute.

Identifying owner
The entity type on which the weak entity type depends.

An example of a weak entity type with an identifying relationship is shown in Figure 2-5. EMPLOYEE is a strong entity type with identifier Employee ID (we note the identifier attribute by underlining it). DEPENDENT is a weak entity type, as indicated by the double-lined rectangle. The relationship between a weak entity type and its owner is called an **identifying relationship.** In Figure 2-5, Carries is the identifying relationship (indicated by the double line). The attribute Dependent Name serves as a *partial* identifier. (Dependent Name is a composite attribute that can be broken into component parts, as we describe later.) We use a double underline to indicate a partial identifier. During a later design stage, Dependent Name will be combined with Employee ID (the identifier of the owner) to form a full identifier for DEPENDENT.

Identifying relationship
The relationship between a weak entity type and its owner.

NAMING AND DEFINING ENTITY TYPES In addition to the general guidelines for naming and defining data objects, there are a few special guidelines for *naming* entity types, which follow:

- An entity type name is a *singular noun* (such as CUSTOMER, STUDENT, or AUTOMOBILE); an entity is a person, a place, an object, an event, or a concept, and the name is for the entity type, which represents a set of entity instances (i.e., STUDENT represents students Hank Finley, Jean Krebs, and so forth). It is common to also specify the plural form (possibly in a CASE tool repository accompanying the E-R diagram), because sometimes the E-R diagram is read best by using plurals. For example, in Figure 2-1, we would say that a SUPPLIER may supply ITEMs. Because plurals are not always formed by adding an *s* to the singular noun, it is best to document the exact plural form.
- An entity type name should be *specific to the organization*. Thus, one organization may use the entity type name CUSTOMER and another organization may use the entity type name CLIENT (this is one task, for example, done to customize a purchased data model). The name should be descriptive for everyone in the organization and distinct from all other entity type names within that organization. For example, a PURCHASE ORDER for orders placed with suppliers is distinct from CUSTOMER ORDER for orders placed with us by our customers. Both of these entity types cannot be named ORDER.

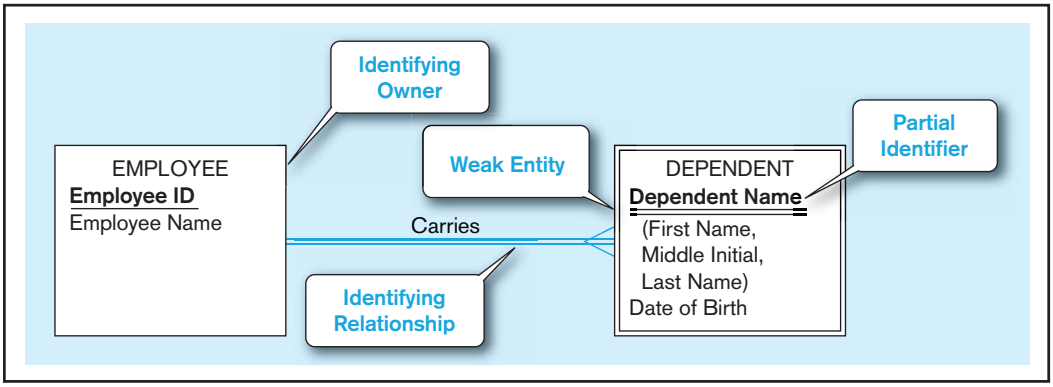


FIGURE 2-5 Example of a weak entity and its identifying relationship

- An entity type name should be *concise*, using as few words as possible. For example, in a university database, an entity type REGISTRATION for the event of a student registering for a class is probably a sufficient name for this entity type; STUDENT REGISTRATION FOR CLASS, although precise, is probably too wordy because the reader will understand REGISTRATION from its use with other entity types.
- An *abbreviation*, or a *short name*, should be specified for each entity type name, and the abbreviation may be sufficient to use in the E-R diagram; abbreviations must follow all of the same rules as do the full entity names.
- *Event entity types* should be named for the result of the event, not the activity or process of the event. For example, the event of a project manager assigning an employee to work on a project results in an ASSIGNMENT, and the event of a student contacting his or her faculty adviser seeking some information is a CONTACT.
- The *name* used for the same entity type *should be the same* on all E-R diagrams on which the entity type appears. Thus, as well as being specific to the organization, the name used for an entity type should be a standard, adopted by the organization for all references to the same kind of data. However, some entity types will have aliases, or alternative names, which are synonyms used in different parts of the organization. For example, the entity type ITEM may have aliases of MATERIAL (for production) and DRAWING (for engineering). Aliases are specified in documentation about the database, such as the repository of a CASE tool.

There are also some specific guidelines for *defining* entity types, which follow:

- *An entity type definition usually starts with* "An X is. . . ." This is the most direct and clear way to state the meaning of an entity type.
- An entity type definition should *include a statement of what the unique characteristic is for each instance of the entity type*. In many cases, stating the identifier for an entity type helps to convey the meaning of the entity. An example for Figure 2-4b is "An expense is a payment of the purchase of some good or service. An expense is identified by a journal entry number."
- An entity type definition should make it clear what *entity instances are included and not included* in the entity type; often, it is necessary to list the kinds of entities that are excluded. For example, "A customer is a person or organization that has placed an order for a product from us or one that we have contacted to advertise or promote our products. A customer does not include persons or organizations that buy our products only through our customers, distributors, or agents."
- An entity type definition often includes a description of *when an instance of the entity type is created and deleted*. For example, in the previous bullet point, a customer instance is implicitly created when the person or organization places its first order; because this definition does not specify otherwise, implicitly a customer instance is never deleted, or it is deleted based on general rules that are specified about the purging of data from the database. A statement about when to delete an entity instance is sometimes referred to as the retention of the entity type. A possible deletion statement for a customer entity type definition might be "A customer ceases to be a customer if it has not placed an order for more than three years."
- For some entity types, the definition must specify *when an instance might change into an instance of another entity type*. For example, consider the situation of a construction company for which bids accepted by potential customers become contracts. In this case, a bid might be defined by "A bid is a legal offer by our organization to do work for a customer. A bid is created when an officer of our company signs the bid document; a bid becomes an instance of contract when we receive a copy of the bid signed by an officer of the customer." This definition is also a good example to note how one definition can use other entity type names (in this case, the definition of bid uses the entity type name CUSTOMER).
- For some entity types, the definition must specify *what history is to be kept about instances of the entity type*. For example, the characteristics of an ITEM in Figure 2-1

may change over time, and we may need to keep a complete history of the individual values and when they were in effect. As we will see in some examples later, such statements about keeping history may have ramifications about how we represent the entity type on an E-R diagram and eventually how we store data for the entity instances.

Attributes

Attribute
A property or characteristic of an entity or relationship type that is of interest to the organization.

Each entity type has a set of attributes associated with it. An **attribute** is a property or characteristic of an entity type that is of interest to the organization. (Later we will see that some types of relationships may also have attributes.) Thus, an attribute has a noun name. Following are some typical entity types and their associated attributes:

STUDENT	Student ID, Student Name, Home Address, Phone Number, Major
AUTOMOBILE	Vehicle ID, Color, Weight, Horsepower
EMPLOYEE	Employee ID, Employee Name, Payroll Address, Skill

In naming attributes, we use an initial capital letter followed by lowercase letters. If an attribute name consists of more than one words, we use a space between the words and we start each word with a capital letter; for example Employee Name or Student Home Address. In E-R diagrams, we represent an attribute by placing its name in the entity it describes. Attributes may also be associated with relationships, as described later. Note that an attribute is associated with exactly one entity or relationship.

Notice in Figure 2-5 that all of the attributes of DEPENDENT are characteristics only of an employee's dependent, not characteristics of an employee. In traditional E-R notation, an entity type (not just weak entities but any entity) does not include attributes of entities to which it is related (what might be called foreign attributes). For example, DEPENDENT does not include any attribute that indicates to which employee this dependent is associated. This nonredundant feature of the E-R data model is consistent with the shared data property of databases. Because of relationships, which we discuss shortly, someone accessing data from a database will be able to associate attributes from related entities (e.g., show on a display screen a Dependent Name and the associated Employee Name).

Required attribute
An attribute that must have a value for every entity (or relationship) instance with which it is associated.

Optional attribute
An attribute that may not have a value for every entity (or relationship) instance with which it is associated.

REQUIRED VERSUS OPTIONAL ATTRIBUTES Each entity (or instance of an entity type) potentially has a value associated with each of the attributes of that entity type. An attribute that must be present for each entity instance is called a **required attribute**, whereas an attribute that may not have a value is called an **optional attribute**. For example, Figure 2-6 shows two STUDENT entities (instances) with their respective

FIGURE 2-6 Entity type STUDENT with required and optional attributes

Entity type: STUDENT

Attributes	Attribute Data Type	Required or Optional	Example Instance	Example Instance
Student ID	CHAR (10)	Required	876-24-8217	822-24-4456
Student Name	CHAR (40)	Required	Michael Grant	Melissa Kraft
Home Address	CHAR (30)	Required	314 Baker St.	1422 Heft Ave
Home City	CHAR (20)	Required	Centerville	Miami
Home State	CHAR (2)	Required	OH	FL
Home Zip Code	CHAR (9)	Required	45459	33321
Major	CHAR (3)	Optional	MIS	

attribute values. The only optional attribute for STUDENT is Major. (Some students, specifically Melissa Kraft in this example, have not chosen a major yet; MIS would, of course, be a great career choice!) However, every student must, by the rules of the organization, have values for all the other attributes; *that is, we cannot store any data about a student in a STUDENT entity instance unless there are values for all the required attributes*. In various E-R diagramming notations, a symbol might appear in front of each attribute to indicate whether it is required (e.g., *) or optional (e.g., o), or required attributes will be in **boldface**, whereas optional attributes will be in normal font (the format we use in this text); in many cases, required or optional is indicated within supplemental documentation. In Chapter 3, when we consider entity supertypes and subtypes, we will see how sometimes optional attributes imply that there are different types of entities. (For example, we may want to consider students who have not declared a major as a subtype of the student entity type.) An attribute without a value is said to be null. Thus, each entity has an identifying attribute, which we discuss in a subsequent section, plus one or more other attributes. If you try to create an entity that has only an identifier, that entity is likely not legitimate. Such a data structure may simply hold a list of legal values for some attribute, which is better kept outside the database.

SIMPLE VERSUS COMPOSITE ATTRIBUTES Some attributes can be broken down into meaningful component parts (detailed attributes). A common example is Name, which we saw in Figure 2-5; another is Address, which can usually be broken down into the following component attributes: Street Address, City, State, and Postal Code. A **composite attribute** is an attribute, such as Address, that has meaningful component parts, which are more detailed attributes. Figure 2-7 shows the notation that we use for composite attributes applied to this example. Most drawing tools do not have a notation for composite attributes, so you simply list all the component parts.

Composite attribute
An attribute that has meaningful component parts (attributes).

Composite attributes provide considerable flexibility to users, who can either refer to the composite attribute as a single unit or else refer to individual components of that attribute. Thus, for example, a user can either refer to Address or refer to one of its components, such as Street Address. The decision about whether to subdivide an attribute into its component parts depends on whether users will need to refer to those individual components, and hence, they have organizational meaning. Of course, the designer must always attempt to anticipate possible future usage patterns for the database.

A **simple (or atomic) attribute** is an attribute that cannot be broken down into smaller components that are meaningful for the organization. For example, all the attributes associated with AUTOMOBILE are simple: Vehicle ID, Color, Weight, and Horsepower.

Simple (or atomic) attribute
An attribute that cannot be broken down into smaller components that are meaningful to the organization.

SINGLE-VALUED VERSUS MULTIVALUED ATTRIBUTES Figure 2-6 shows two entity instances with their respective attribute values. For each entity instance, each of the attributes in the figure has one value. It frequently happens that there is an attribute that may have more than one value for a given instance. For example, the EMPLOYEE entity type in Figure 2-8 has an attribute named Skill, whose values record the skill (or skills) for that employee. Of course, some employees may have more than one skill,

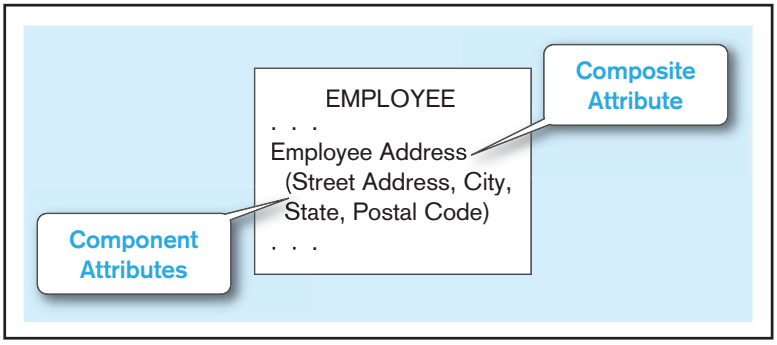


FIGURE 2-7 A composite attribute

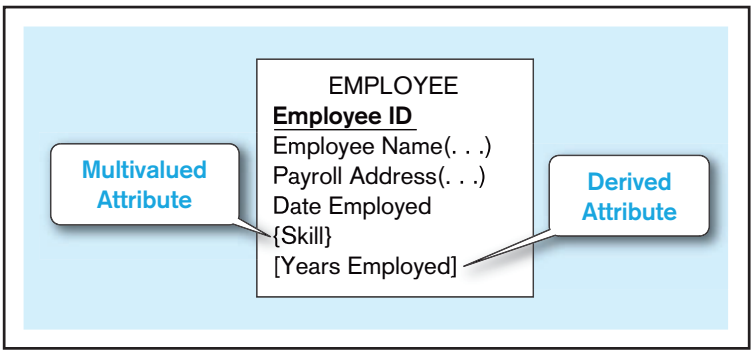


FIGURE 2-8 Entity with multivalued attribute (Skill) and derived attribute (Years Employed)

Multivalued attribute
An attribute that may take on more than one value for a given entity (or relationship) instance.

such as PHP Programmer and C++ Programmer. A **multivalued attribute** is an attribute that may take on more than one value for a given entity (or relationship) instance. In this text we indicate a multivalued attribute with curly brackets around the attribute name, as shown for the Skill attribute in the EMPLOYEE example in Figure 2-8. In Microsoft Visio, once an attribute is placed in an entity, you can edit that attribute (column), select the Collection tab, and choose one of the options. (Typically, MultiSet will be your choice, but one of the other options may be more appropriate for a given situation.) Other E-R diagramming tools may use an asterisk (*) after the attribute name, or you may have to use supplemental documentation to specify a multivalued attribute.

Multivalued and composite are different concepts, although beginner data modelers often confuse these terms. Skill, a multivalued attribute, may occur multiple times for each employee; Employee Name and Payroll Address are both likely composite attributes, each of which occurs once for each employee, but which have component, more atomic attributes, which are not shown in Figure 2-8 for simplicity. See Problem and Exercise 14 to review the concepts of composite and multivalued attributes.

STORED VERSUS DERIVED ATTRIBUTES Some attribute values that are of interest to users can be calculated or derived from other related attribute values that are stored in the database. For example, suppose that for an organization, the EMPLOYEE entity type has a Date Employed attribute. If users need to know how many years a person has been employed, that value can be calculated using Date Employed and today's date. A **derived attribute** is an attribute whose values can be calculated from related attribute values (plus possibly data not in the database, such as today's date, the current time, or a security code provided by a system user). We indicate a derived attribute in an E-R diagram by using square brackets around the attribute name, as shown in Figure 2-8 for the Years Employed attribute. Some E-R diagramming tools use a notation of a forward slash (/) in front of the attribute name to indicate that it is derived. (This notation is borrowed from UML for a virtual attribute.)

Derived attribute
An attribute whose values can be calculated from related attribute values.

In some situations, the value of an attribute can be derived from attributes in related entities. For example, consider an invoice created for each customer at Pine Valley Furniture Company. Order Total would be an attribute of the INVOICE entity, which indicates the total dollar amount that is billed to the customer. The value of Order Total can be computed by summing the Extended Price values (unit price times quantity sold) for the various line items that are billed on the invoice. Formulas for computing values such as this are one type of business rule.

IDENTIFIER ATTRIBUTE An **identifier** is an attribute (or combination of attributes) whose value distinguishes individual instances of an entity type. That is, no two instances of the entity type may have the same value for the identifier attribute. The identifier for the STUDENT entity type introduced earlier is Student ID, whereas the identifier for AUTOMOBILE is Vehicle ID. Notice that an attribute such as Student Name is not a candidate identifier, because many students may potentially have the same name, and students, like all people, can change their names. To be a candidate identifier, each entity instance must have a single value for the attribute and the attribute must be associated with the entity. We underline identifier names on the E-R diagram, as

Identifier
An attribute (or combination of attributes) whose value distinguishes instances of an entity type.

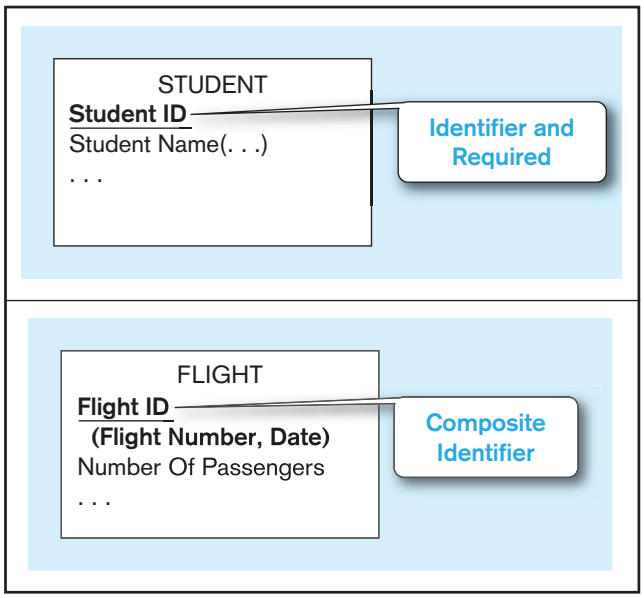


FIGURE 2-9 Simple and composite identifier attributes
(a) Simple identifier attribute

(b) Composite identifier attribute

shown in the STUDENT entity type example in Figure 2-9a. To be an identifier, the attributed is also required (so the distinguishing value must exist), so an identifier is also in bold. Some E-R drawing software will place a symbol, called a stereotype, in front of the identifier (e.g., <<ID>> or <<PK>>).

For some entity types, there is no single (or atomic) attribute that can serve as the identifier (i.e., that will ensure uniqueness). However, two (or more) attributes used in combination may serve as the identifier. A **composite identifier** is an identifier that consists of a composite attribute. Figure 2-9b shows the entity FLIGHT with the composite identifier Flight ID. Flight ID in turn has component attributes Flight Number and Date. This combination is required to identify uniquely individual occurrences of FLIGHT. We use the convention that the composite attribute (Flight ID) is underlined to indicate it is the identifier, while the component attributes are not underlined. Some data modelers think of a composite identifier as "breaking a tie" created by a simple identifier. Even with Flight ID, a data modeler would ask a question, such as "Can two flights with the same number occur on the same date?" If so, yet another attribute is needed to form the composite identifier and to break the tie.

Composite identifier
An identifier that consists of a composite attribute.

Some entities may have more than one candidate identifier. If there is more than one candidate identifier, the designer must choose one of them as the identifier. Bruce (1992) suggests the following criteria for selecting identifiers:

1. Choose an identifier that will not change its value over the life of each instance of the entity type. For example, the combination of Employee Name and Payroll Address (even if unique) would be a poor choice as an identifier for EMPLOYEE because the values of Employee Name and Payroll Address could easily change during an employee's term of employment.
2. Choose an identifier such that for each instance of the entity, the attribute is guaranteed to have valid values and not be null (or unknown). If the identifier is a composite attribute, such as Flight ID in Figure 2-9b, make sure that all parts of the identifier will have valid values.
3. Avoid the use of so-called intelligent identifiers (or keys), whose structure indicates classifications, locations, and so on. For example, the first two digits of an identifier value may indicate the warehouse location. Such codes are often changed as conditions change, which renders the identifier values invalid.
4. Consider substituting single-attribute surrogate identifiers for large composite identifiers. For example, an attribute called Game Number could be used for the entity type GAME instead of the combination of Home Team and Visiting Team.

NAMING AND DEFINING ATTRIBUTES In addition to the general guidelines for naming data objects, there are a few special guidelines for naming attributes, which follow:

- An attribute name is a *singular noun or noun phrase* (such as Customer ID, Age, Product Minimum Price, or Major). Attributes, which materialize as data values, are concepts or physical characteristics of entities. Concepts and physical characteristics are described by nouns.
- An attribute name should be *unique*. No two attributes of the same entity type may have the same name, and it is desirable, for clarity purposes, that no two attributes across all entity types have the same name.
- To make an attribute name unique and for clarity purposes, *each attribute name should follow a standard format*. For example, your university may establish Student GPA, as opposed to GPA of Student, as an example of the standard format for attribute naming. The format to be used will be established by each organization. A common format is [Entity type name { [Qualifier] }] Class, where [. . .] is an optional clause, and { . . . } indicates that the clause may repeat. *Entity type name* is the name of the entity with which the attribute is associated. The entity type name may be used to make the attribute name explicit. It is almost always used for the identifier attribute (e.g., Customer ID) of each entity type. *Class* is a phrase from a list of phrases defined by the organization that are the permissible characteristics or properties of entities (or abbreviations of these characteristics). For example, permissible values (and associated approved abbreviations) for Class might be Name (Nm), Identifier (ID), Date (Dt), or Amount (Amt). Class is, obviously, required. *Qualifier* is a phrase from a list of phrases defined by the organization that are used to place constraints on classes. One or more qualifiers may be needed to make each attribute of an entity type unique. For example, a qualifier might be Maximum (Max), Hourly (Hrly), or State (St). A qualifier may not be necessary: Employee Age and Student Major are both fully explicit attribute names. Sometimes a qualifier is necessary. For example, Employee Birth Date and Employee Hire Date are two attributes of Employee that require one qualifier. More than one qualifier may be necessary. For example, Employee Residence City Name (or Emp Res Cty Nm) is the name of an employee's city of residence, and Employee Tax City Name (or Emp Tax Cty Nm) is the name of the city in which an employee pays city taxes.
- *Similar attributes* of different entity types *should use the same qualifiers and classes*, as long as those are the names used in the organization. For example, the city of residence for faculty and students should be, respectively, Faculty Residence City Name and Student Residence City Name. Using similar names makes it easier for users to understand that values for these attributes come from the same possible set of values, what we will call *domains*. Users may want to take advantage of common domains in queries (e.g., find students who live in the same city as their adviser), and it will be easier for users to recognize that such a matching may be possible if the same qualifier and class phrases are used.

There are also some specific guidelines for defining attributes, which follow:

- An attribute definition states *what the attribute is and possibly why it is important*. The definition will often parallel the attribute's name; for example, Student Residence City Name could be defined as "The name of the city in which a student maintains his or her permanent residence."
- An attribute definition should make it clear *what is included and not included* in the attribute's value; for example, "Employee Monthly Salary Amount is the amount of money paid each month in the currency of the country of residence of the employee exclusive of any benefits, bonuses, reimbursements, or special payments."
- Any *aliases*, or alternative names, for the attribute can be specified in the definition, or may be included elsewhere in documentation about the attribute, possibly stored in the repository of a CASE tool used to maintain data definitions.
- It may also be desirable to state in the definition *the source of values for the attribute*. Stating the source may make the meaning of the data clearer. For example, "Customer Standard Industrial Code is an indication of the type of business for

the customer. Values for this code come from a standard set of values provided by the Federal Trade Commission and are found on a CD we purchase named SIC provided annually by the FTC."

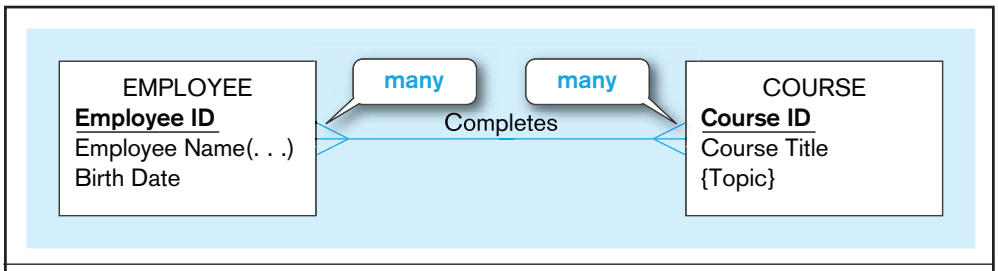
- An attribute definition (or other specification in a CASE tool repository) also should indicate *if a value for the attribute is required or optional*. This business rule about an attribute is important for maintaining data integrity. The identifier attribute of an entity type is, by definition, required. If an attribute value is required, then to create an instance of the entity type, a value of this attribute must be provided. Required means that an entity instance must always have a value for this attribute, not just when an instance is created. Optional means that a value may not exist for an instance of an entity instance to be stored. Optional can be further qualified by stating whether once a value is entered, a value must always exist. For example, "Employee Department ID is the identifier of the department to which the employee is assigned. An employee may not be assigned to a department when hired (so this attribute is initially optional), but once an employee is assigned to a department, the employee must always be assigned to some department."
- An attribute definition (or other specification in a CASE tool repository) may also indicate *whether a value for the attribute may change* once a value is provided and before the entity instance is deleted. This business rule also controls data integrity. Nonintelligent identifiers may not change values over time. To assign a new nonintelligent identifier to an entity instance, that instance must first be deleted and then re-created.
- For a multivalued attribute, the attribute definition should indicate *the maximum and minimum number of occurrences of an attribute value for an entity instance*. For example, "Employee Skill Name is the name of a skill an employee possesses. Each employee must possess at least one skill, and an employee can choose to list at most 10 skills." The reason for a multivalued attribute may be that a history of the attribute needs to be kept. For example, "Employee Yearly Absent Days Number is the number of days in a calendar year the employee has been absent from work. An employee is considered absent if he or she works less than 50 percent of the scheduled hours in the day. A value for this attribute should be kept for each year in which the employee works for our company."
- An attribute definition may also indicate *any relationships that attribute has with other attributes*. For example, "Employee Vacation Days Number is the number of days of paid vacation for the employee. If the employee has a value of 'Exempt' for Employee Type, then the maximum value for Employee Vacation Days Number is determined by a formula involving the number of years of service for the employee."

MODELING RELATIONSHIPS

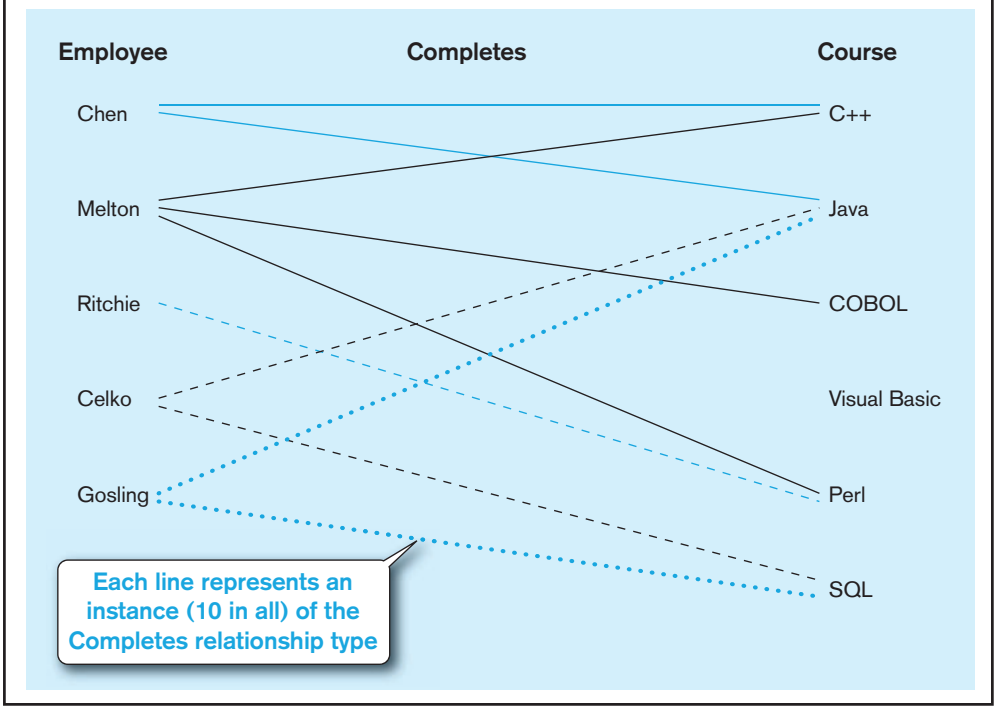
Relationships are the glue that holds together the various components of an E-R model. Intuitively, a *relationship* is an association representing an interaction among the instances of one or more entity types that is of interest to the organization. Thus, a relationship has a verb phrase name. Relationships and their characteristics (degree and cardinality) represent business rules, and usually relationships represent the most complex business rules shown in an ERD. In other words, this is where data modeling gets really interesting and fun, as well as crucial for controlling the integrity of a database.

To understand relationships more clearly, we must distinguish between relationship types and relationship instances. To illustrate, consider the entity types EMPLOYEE and COURSE, where COURSE represents training courses that may be taken by employees. To track courses that have been completed by particular employees, we define a relationship called Completes between the two entity types (see Figure 2-10a). This is a many-to-many relationship, because each employee may complete any number of courses (zero, one, or many courses), whereas a given course may be completed by any number of employees (nobody, one employee, many employees). For example, in Figure 2-10b, the employee Melton has completed three courses (C++, COBOL, and Perl). The SQL course has been completed by two employees (Celko and Gosling), and the Visual Basic course has not been completed by anyone.

FIGURE 2-10 Relationship type and instances
(a) Relationship type (Complete)



(b) Relationship instances



In this example, there are two entity types (EMPLOYEE and COURSE) that participate in the relationship named Completes. In general, any number of entity types (from one to many) may participate in a relationship.

We frequently use in this and subsequent chapters the convention of a single verb phrase label to represent a relationship. Because relationships often occur due to an organizational event, entity instances are related because an action was taken; thus a verb phrase is appropriate for the label. This verb phrase should be in the present tense and descriptive. There are, however, many ways to represent a relationship. Some data modelers prefer the format with two relationship names, one to name the relationship in each direction. One or two verb phrases have the same structural meaning, so you may use either format as long as the meaning of the relationship in each direction is clear.

Basic Concepts and Definitions in Relationships

Relationship type
A meaningful association between (or among) entity types.

A **relationship type** is a meaningful association between (or among) entity types. The phrase *meaningful association* implies that the relationship allows us to answer questions that could not be answered given only the entity types. A relationship type is denoted by a line labeled with the name of the relationship, as in the example shown in Figure 2-10a, or with two names, as in Figure 2-1. We suggest you use a short, descriptive verb phrase that is meaningful to the user in naming the relationship. (We say more about naming and defining relationships later in this section.)

Relationship instance
An association between (or among) entity instances where each relationship instance associates exactly one entity instance from each participating entity type.

A **relationship instance** is an association between (or among) entity instances, where each relationship instance associates exactly one entity instance from each participating entity type (Elmasri and Navathe, 1994). For example, in Figure 2-10b, each of the 10 lines in the figure represents a relationship instance between one employee and one course, indicating that the employee has completed that course. For example, the line between Employee Ritchie to Course Perl is one relationship instance.

TABLE 2-2 Instances Showing Date Completed

Employee Name	Course Title	Date Completed
Chen	C++	06/2009
Chen	Java	09/2009
Melton	C++	06/2009
Melton	COBOL	02/2010
Melton	SQL	03/2009
Ritchie	Perl	11/2009
Celko	Java	03/2009
Celko	SQL	03/2010
Gosling	Java	09/2009
Gosling	Perl	06/2009

ATTRIBUTES ON RELATIONSHIPS It is probably obvious to you that entities have attributes, but attributes may be associated with a many-to-many (or one-to-one) relationship, too. For example, suppose the organization wishes to record the date (month and year) when an employee completes each course. This attribute is named Date Completed. For some sample data, see Table 2-2.

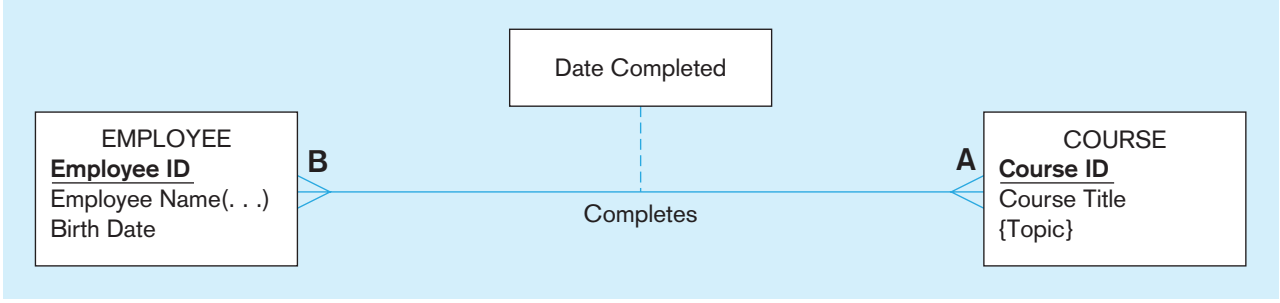
Where should the attribute Date Completed be placed on the E-R diagram? Referring to Figure 2-10a, you will notice that Date Completed has not been associated with either the EMPLOYEE or COURSE entity. That is because Date Completed is a property of the relationship Completes, rather than a property of either entity. In other words, for each instance of the relationship Completes, there is a value for Date Completed. One such instance (for example) shows that the employee named Melton completed the course titled C++ in 06/2009.

A revised version of the ERD for this example is shown in Figure 2-11a. In this diagram, the attribute Date Completed is in a rectangle connected to the Completes relationship line. Other attributes might be added to this relationship if appropriate, such as Course Grade, Instructor, and Room Location.

It is interesting to note that an attribute cannot be associated with a one-to-many relationship, such as Carries in Figure 2-5. For example, consider Dependent Date, similar to Date Completed above, for when the DEPENDENT begins to be carried by the EMPLOYEE. Because each DEPENDENT is associated with only one EMPLOYEE, such a date is unambiguously a characteristic of the DEPENDENT (i.e., for a given DEPENDENT, Dependent Date cannot vary by EMPLOYEE). So, if you ever have the urge to associate an attribute with a one-to-many relationship, "step away from the relationship!"

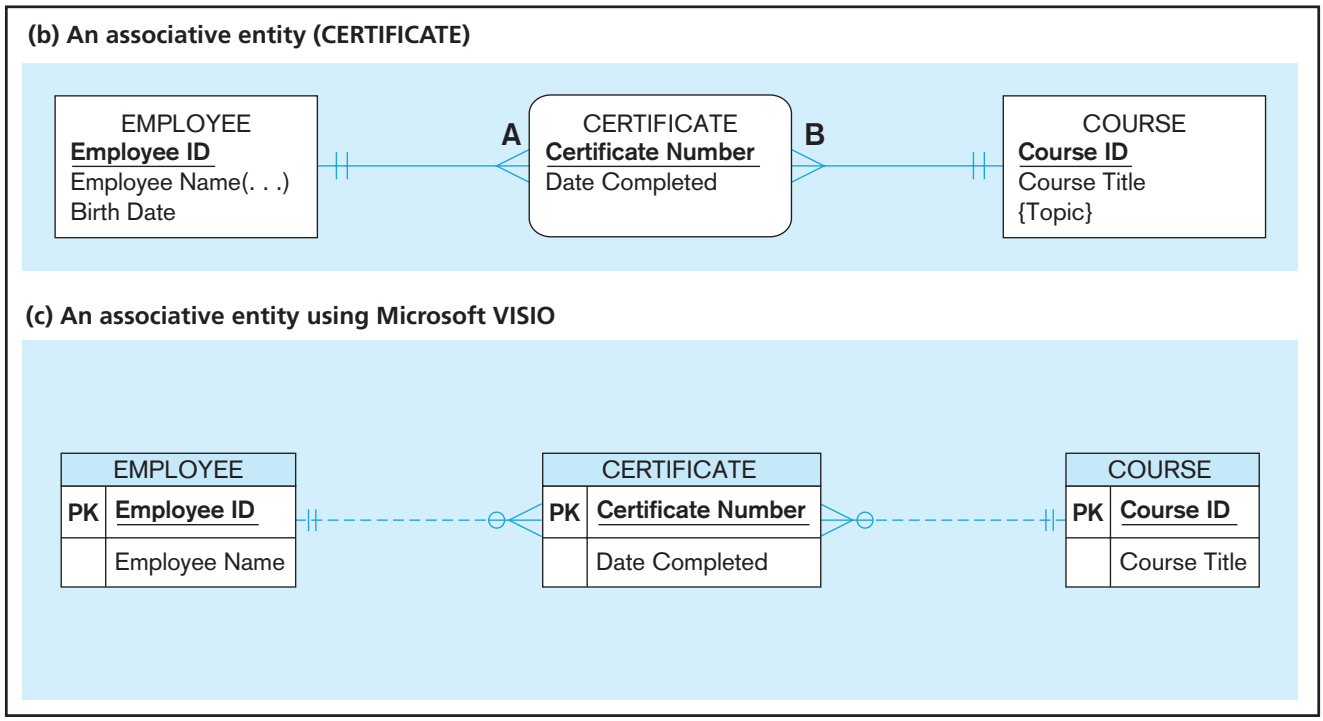
FIGURE 2-11 An associative entity

(a) Attribute on a relationship



(continued)

FIGURE 2-11 (*continued*)



ASSOCIATIVE ENTITIES The presence of one or more attributes on a relationship suggests to the designer that the relationship should perhaps instead be represented as an entity type. To emphasize this point, most E-R drawing tools require that such attributes be placed in an entity type. An **associative entity** is an entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances. The associative entity CERTIFICATE is represented with the rectangle with rounded corners, as shown in Figure 2-11b. Most E-R drawing tools do not have a special symbol for an associative entity. Associative entities are sometimes referred to as gerunds, because the relationship name (a verb) is usually converted to an entity name that is a noun. Note in Figure 2-11b that there are no relationship names on the lines between an associative entity and a strong entity. This is because the associative entity represents the relationship. Figure 2-11c shows how associative entities are drawn using Microsoft Visio, which is representative of how you would draw an associative entity with most E-R diagramming tools. In Visio, the relationship lines are dashed because CERTIFICATE does not include the identifiers of the related entities in its identifier. (Certificate Number is sufficient.)

Associative entity
An entity type that associates the instances of one or more entity types and contains attributes that are peculiar to the relationship between those entity instances.

How do you know whether to convert a relationship to an associative entity type? Following are four conditions that should exist:

1. All the relationships for the participating entity types are "many" relationships.
2. The resulting associative entity type has independent meaning to end users and, preferably, can be identified with a single-attribute identifier.
3. The associative entity has one or more attributes in addition to the identifier.
4. The associative entity participates in one or more relationships independent of the entities related in the associated relationship.

Figure 2-11b shows the relationship Completes converted to an associative entity type. In this case, the training department for the company has decided to award a certificate to each employee who completes a course. Thus, the entity is named CERTIFICATE, which certainly has independent meaning to end users. Also, each certificate has a number (Certificate Number) that serves as the identifier. The attribute Date Completed is also included. Note also in Figure 2-11b and the Visio version of Figure 2-11c that both EMPLOYEE and COURSE are mandatory participants in the two relationships with CERTIFICATE. This is exactly what occurs when you have to represent a many-to-many

relationship (Completes in Figure 2-11a) as two one-to-many relationships (the ones associated with CERTIFICATE in Figures 2-11b and 2-11c).

Notice that converting a relationship to an associative entity has caused the relationship notation to move. That is, the "many" cardinality now terminates at the associative entity, rather than at each participating entity type. In Figure 2-11, this shows that an employee, who may complete one or more courses (notation A in Figure 2-11a), may be awarded more than one certificate (notation A in Figure 2-11b); and that a course, which may have one or more employees complete it (notation B in Figure 2-11a), may have many certificates awarded (notation B in Figure 2-11b). See Problem and Exercise 18 for an interesting variation on Figure 2-11a, which emphasizes the rules for when to convert a many-to-many relationship, such as Completes, into an associative entity.

Degree of a Relationship

Degree
The number of entity types that participate in a relationship.

The **degree** of a relationship is the number of entity types that participate in that relationship. Thus, the relationship Completes in Figure 2-11 is of degree 2, because there are two entity types: EMPLOYEE and COURSE. The three most common relationship degrees in E-R models are unary (degree 1), binary (degree 2), and ternary (degree 3). Higher-degree relationships are possible, but they are rarely encountered in practice, so we restrict our discussion to these three cases. Examples of unary, binary, and ternary relationships appear in Figure 2-12. (Attributes are not shown in some figures for simplicity.)

As you look at Figure 2-12, understand that any particular data model represents a specific situation, not a generalization. For example, consider the Manages relationship in Figure 2-12a. In some organizations, it may be possible for one employee to be managed by many other employees (e.g., in a matrix organization). It is important when you develop an E-R model that you understand the business rules of the particular organization you are modeling.

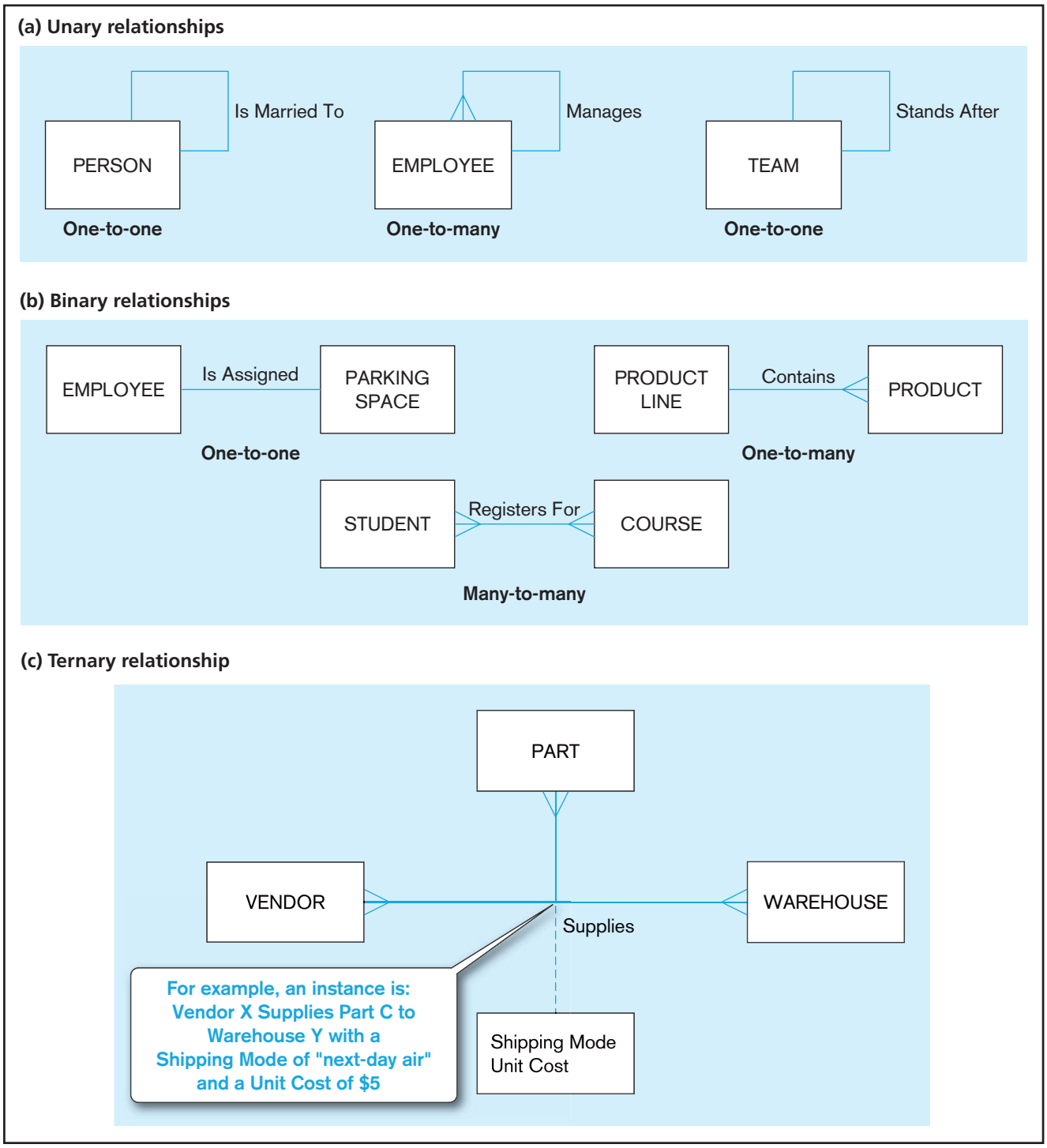
Unary relationship
A relationship between instances of a single entity type.

UNARY RELATIONSHIP A **unary relationship** is a relationship between the instances of a *single* entity type. (Unary relationships are also called *recursive relationships*.) Three examples are shown in Figure 2-12a. In the first example, Is Married To is shown as a one-to-one relationship between instances of the PERSON entity type. Because this is a one-to-one relationship, this notation indicates that only the current marriage, if one exists, needs to be kept about a person. What would change if we needed to retain the history of marriages for each person? See Review Question 20 and Problem and Exercise 10 for other business rules and their effect on the Is Married To relationship representation. In the second example, Manages is shown as a one-to-many relationship between instances of the EMPLOYEE entity type. Using this relationship, we could identify, for example, the employees who report to a particular manager. The third example is one case of using a unary relationship to represent a sequence, cycle, or priority list. In this example, sports teams are related by their standing in their league (the Stands After relationship). (Note: In these examples, we ignore whether these are mandatory- or optional-cardinality relationships or whether the same entity instance can repeat in the same relationship instance; we will introduce mandatory and optional cardinality in a later section of this chapter.)

Figure 2-13 shows an example of another unary relationship, called a *bill-of-materials structure*. Many manufactured products are made of assemblies, which in turn are composed of subassemblies and parts, and so on. As shown in Figure 2-13a, we can represent this structure as a many-to-many unary relationship. In this figure, the entity type ITEM is used to represent all types of components, and we use Has Components for the name of the relationship type that associates lower-level items with higher-level items.

Two occurrences of this bill-of-materials structure are shown in Figure 2-13b. Each of these diagrams shows the immediate components of each item as well as the quantities of that component. For example, item TX100 consists of item BR450 (quantity 2) and item DX500 (quantity 1). You can easily verify that the associations are in fact many-to-many. Several of the items have more than one component type (e.g., item MX300 has three immediate component types: HX100, TX100, and WX240). Also, some of the components are used in several higher-level assemblies. For example, item WX240 is used in both item MX300 and item WX340, even at different levels of the bill-of-materials. The many-to-many relationship guarantees that, for example, the same subassembly structure of WX240 (not shown) is used each time item WX240 goes into making some other item.

FIGURE 2-12 Examples of relationships of different degrees

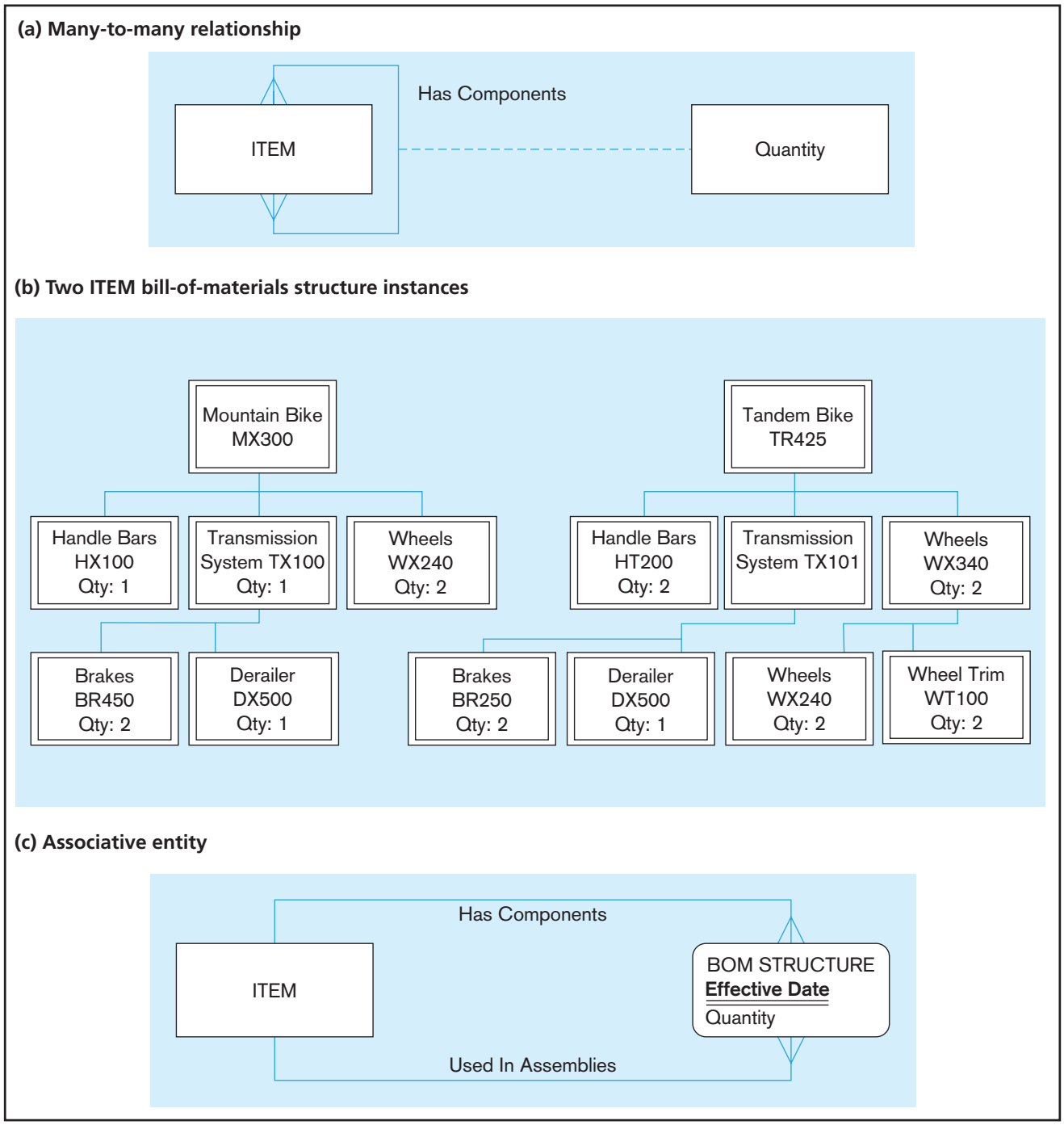


The presence of the attribute Quantity on the relationship suggests that the analyst consider converting the relationship Has Components to an associative entity. Figure 2-13c shows the entity type BOM STRUCTURE, which forms an association between instances of the ITEM entity type. A second attribute (named Effective Date) has been added to BOM STRUCTURE to record the date when this component was first used in the related assembly. Effective dates are often needed when a history of values is required. Other data model structures can be used for unary relationships involving such hierarchies; we show some of these other structures in Chapter 9.

Binary relationship
A relationship between the instances of two entity types.

BINARY RELATIONSHIP A **binary relationship** is a relationship between the instances of two entity types and is the most common type of relationship encountered in data

FIGURE 2-13 **Representing a bill-of-materials structure**



modeling. Figure 2-12b shows three examples. The first (one-to-one) indicates that an employee is assigned one parking place, and that each parking place is assigned to one employee. The second (one-to-many) indicates that a product line may contain several products, and that each product belongs to only one product line. The third (many-to-many) shows that a student may register for more than one course, and that each course may have many student registrants.

TERNARY RELATIONSHIP A **ternary relationship** is a *simultaneous* relationship among the instances of three entity types. A typical business situation that leads to a ternary relationship is shown in Figure 2-12c. In this example, vendors can supply various parts to warehouses. The relationship Supplies is used to record the specific parts that are supplied by a given vendor to a particular warehouse. Thus there are three entity

Ternary relationship
A simultaneous relationship among the instances of three entity types.

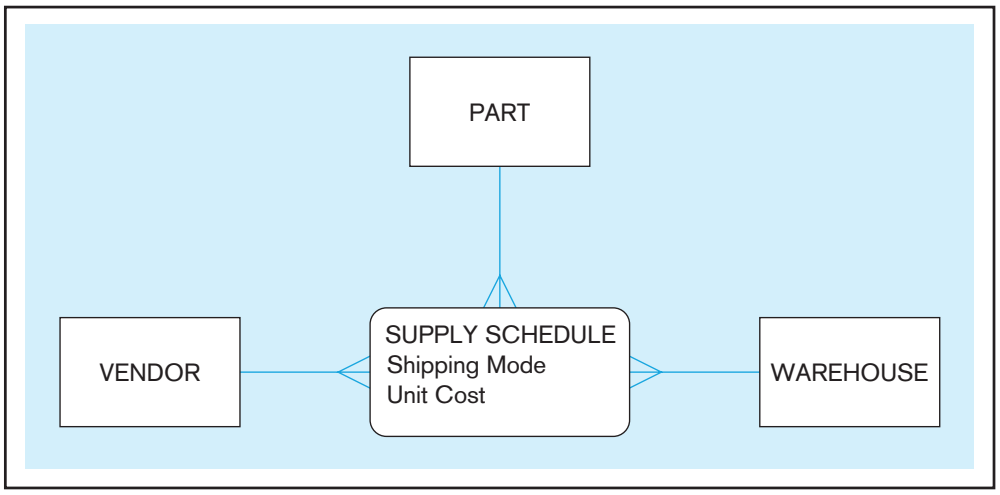


FIGURE 2-14 Ternary relationship as an associative entity

types: VENDOR, PART, and WAREHOUSE. There are two attributes on the relationship Supplies: Shipping Mode and Unit Cost. For example, one instance of Supplies might record the fact that vendor X can ship part C to warehouse Y, that the shipping mode is next-day air, and that the cost is $5 per unit.

Don't be confused: A ternary relationship is not the same as three binary relationships. For example, Unit Cost is an attribute of the Supplies relationship in Figure 2-12c. Unit Cost cannot be properly associated with any one of the three possible binary relationships among the three entity types, such as that between PART and WAREHOUSE. Thus, for example, if we were told that vendor X can ship part C for a unit cost of $8, those data would be incomplete because they would not indicate to which warehouse the parts would be shipped.

As usual, the presence of an attribute on the relationship Supplies in Figure 2-12c suggests converting the relationship to an associative entity type. Figure 2-14 shows an alternative (and preferable) representation of the ternary relationship shown in Figure 2-12c. In Figure 2-14, the (associative) entity type SUPPLY SCHEDULE is used to replace the Supplies relationship from Figure 2-12c. Clearly the entity type SUPPLY SCHEDULE is of independent interest to users. However, notice that an identifier has not yet been assigned to SUPPLY SCHEDULE. This is acceptable. If no identifier is assigned to an associative entity during E-R modeling, an identifier (or key) will be assigned during logical modeling (discussed in Chapter 4). This will be a composite identifier whose components will consist of the identifier for each of the participating entity types (in this example, PART, VENDOR, and WAREHOUSE). Can you think of other attributes that might be associated with SUPPLY SCHEDULE?

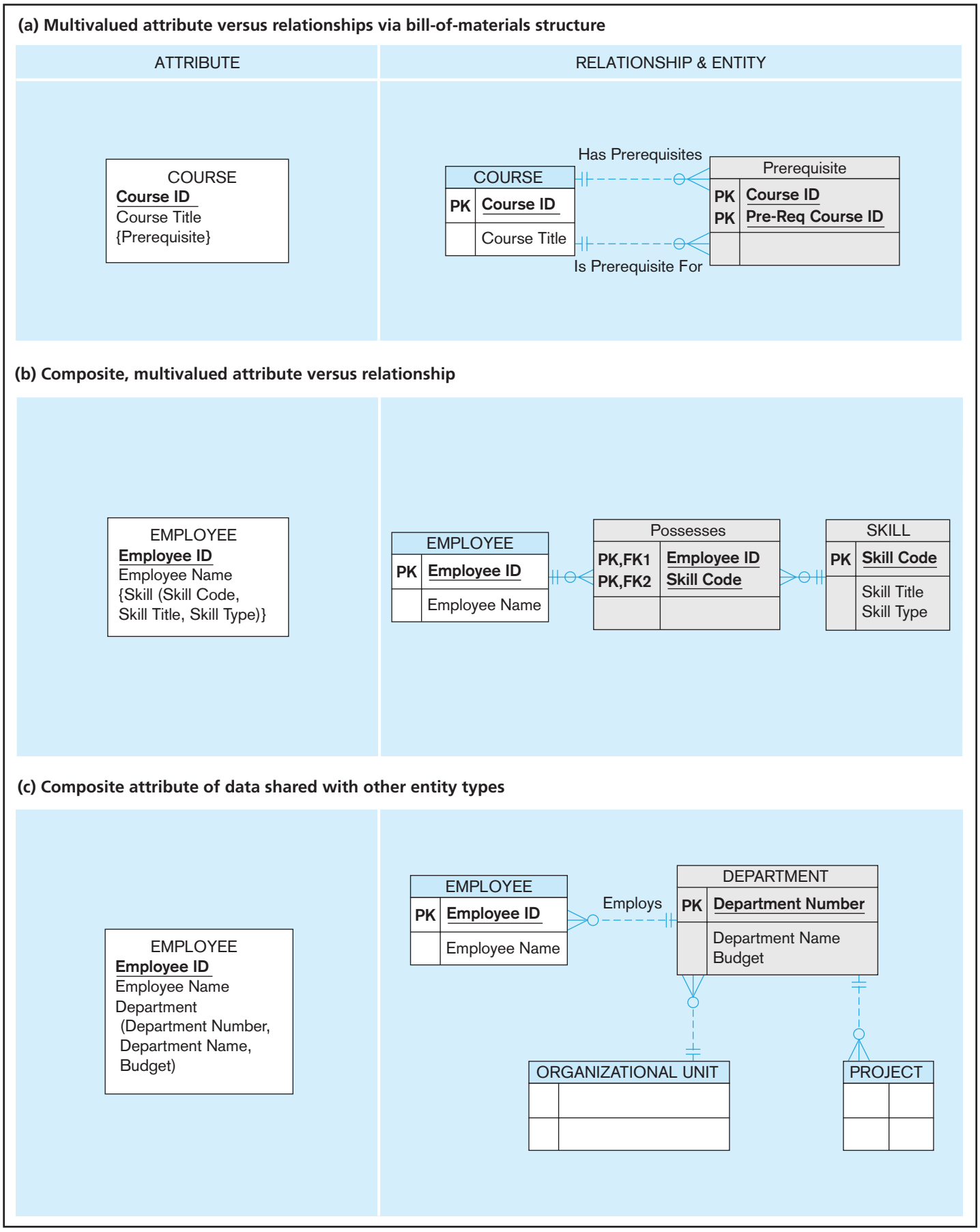
As noted earlier, we do not label the lines from SUPPLY SCHEDULE to the three entities. This is because these lines do not represent binary relationships. To keep the same meaning as the ternary relationship of Figure 2-12c, we cannot break the Supplies relationship into three binary relationships, as we have already mentioned.

So, here is a guideline to follow: Convert all ternary (or higher) relationships to associative entities, as in this example. Song et al. (1995) show that participation constraints (described in a following section on cardinality constraints) cannot be accurately represented for a ternary relationship, given the notation with attributes on the relationship line. However, by converting to an associative entity, the constraints can be accurately represented. Also, many E-R diagram drawing tools, including most CASE tools, cannot represent ternary relationships. So, although not semantically accurate, you must use these tools to represent the ternary relationship with an associative entity and three binary relationships, which have a mandatory association with each of the three related entity types.

Attributes or Entity?

Sometimes you will wonder if you should represent data as an attribute or an entity; this is a common dilemma. Figure 2-15 includes three examples of situations when an attribute could be represented via an entity type. We use this textbook's E-R notation in

FIGURE 2-15 **Using relationships and entities to link related attributes**



the left column and the notation from Microsoft Visio in the right column; it is important that you learn how to read ERDs in several notations because you will encounter various styles in different publications and organizations. In Figure 2-15a, the potentially multiple prerequisites of a course (shown as a multivalued attribute in the Attribute cell) are also courses (and a course may be a prerequisite for many other courses). Thus, prerequisite could be viewed as a bill-of-materials structure (shown in the Relationship & Entity cell) between courses, not a multivalued attribute of COURSE. Representing prerequisites via a bill-of-materials structure also means that finding the prerequisites of a course and finding the courses for which a course is prerequisite both deal with relationships between entity types. When a prerequisite is a multivalued attribute of COURSE, finding the courses for which a course is a prerequisite means looking for a specific value for a prerequisite across all COURSE instances. As was shown in Figure 2-13a, such a situation could also be modeled as a unary relationship among instances of the COURSE entity type. In Visio, this specific situation requires creating the equivalent of an associative entity (see the Relationship & Entity cell in Figure 2-15a; Visio does not use the rectangle with rounded corners symbol). By creating the associative entity, it is now easy to add characteristics to the relationship, such as a minimum grade required. Also note that Visio shows the identifier (in this case compound) with a PK stereotype symbol and boldface on the component attribute names, signifying these are required attributes.

In Figure 2-15b, employees potentially have multiple skills (shown in the Attribute cell), but skill could be viewed instead as an entity type (shown in the Relationship & Entity cell as the equivalent of an associative entity) about which the organization wants to maintain data (the unique code to identify each skill, a descriptive title, and the type of skill, for example technical or managerial). An employee has skills, which are not viewed as attributes, but rather as instances of a related entity type. In the cases of Figures 2-15a and 2-15b, representing the data as a multivalued attribute rather than via a relationship with another entity type may, in the view of some people, simplify the diagram. On the other hand, the right-hand drawings in these figures are closer to the way the database would be represented in a standard relational database management system, the most popular type of DBMS in use today. Although we are not concerned with implementation during conceptual data modeling, there is some logic for keeping the conceptual and logical data models similar. Further, as we will see in the next example, there are times when an attribute, whether simple, composite, or multivalued, should be in a separate entity.

So, when *should* an attribute be linked to an entity type via a relationship? The answer is: when the attribute is the identifier or some other characteristic of an entity type in the data model and multiple entity instances need to share these same attributes. Figure 2-15c represents an example of this rule. In this example, EMPLOYEE has a composite attribute of Department. Because Department is a concept of the business, and multiple employees will share the same department data, department data could be represented (nonredundantly) in a DEPARTMENT entity type, with attributes for the data about departments that all other related entity instances need to know. With this approach, not only can different employees share the storage of the same department data, but projects (which are assigned to a department) and organizational units (which are composed of departments) also can share the storage of this same department data.

Cardinality Constraints

Cardinality constraint
A rule that specifies the number of instances of one entity that can (or must) be associated with each instance of another entity.

There is one more important data modeling notation for representing common and important business rules. Suppose there are two entity types, A and B, that are connected by a relationship. A **cardinality constraint** specifies the number of instances of entity B that can (or must) be associated with each instance of entity A. For example, consider a video store that rents DVDs of movies. Because the store may stock more than one DVD for each movie, this is intuitively a one-to-many relationship, as shown in Figure 2-16a. Yet it is also true that the store may not have any DVDs of a given movie in stock at a particular time (e.g., all copies may be checked out). We need a

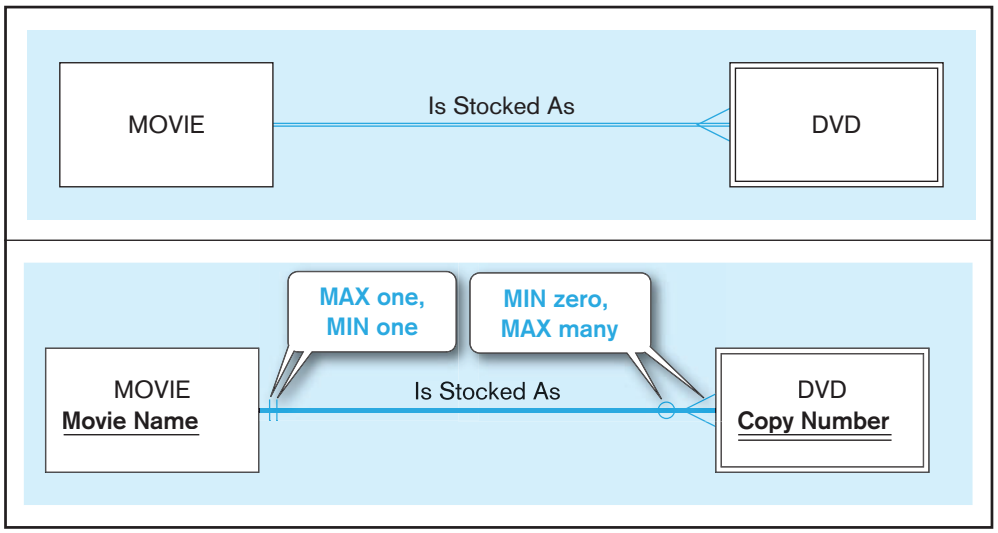


FIGURE 2-16 Introducing cardinality constraints
(a) Basic relationship

(b) Relationship with cardinality constraints

more precise notation to indicate the range of cardinalities for a relationship. This notation was introduced in Figure 2-2, which you may want to review at this time.

MINIMUM CARDINALITY The **minimum cardinality** of a relationship is the minimum number of instances of entity B that may be associated with each instance of entity A. In our DVD example, the minimum number of DVDs for a movie is zero. When the minimum number of participants is zero, we say that entity type B is an optional participant in the relationship. In this example, DVD (a weak entity type) is an optional participant in the Is Stocked As relationship. This fact is indicated by the symbol zero through the line near the DVD entity in Figure 2-16b.

Minimum cardinality
The minimum number of instances of one entity that may be associated with each instance of another entity.

MAXIMUM CARDINALITY The **maximum cardinality** of a relationship is the maximum number of instances of entity B that may be associated with each instance of entity A. In the video example, the maximum cardinality for the DVD entity type is "many"—that is, an unspecified number greater than one. This is indicated by the "crow's foot" symbol on the line next to the DVD entity symbol in Figure 2-16b. (You might find interesting the explanation of the origin of the crow's foot notation found in the Wikipedia entry about the entity-relationship model; this entry also shows the wide variety of notation used to represent cardinality; see **http:/en.wikipedia.org/wiki/Entity-relationship_model**.)

Maximum cardinality
The maximum number of instances of one entity that may be associated with each instance of another entity.

A relationship is, of course, bidirectional, so there is also cardinality notation next to the MOVIE entity. Notice that the minimum and maximum are both one (see Figure 2-16b). This is called a *mandatory one* cardinality. In other words, each DVD of a movie must be a copy of exactly one movie. In general, participation in a relationship may be optional or mandatory for the entities involved. If the minimum cardinality is zero, participation is optional; if the minimum cardinality is one, participation is mandatory.

In Figure 2-16b, some attributes have been added to each of the entity types. Notice that DVD is represented as a weak entity. This is because a DVD cannot exist unless the owner movie also exists. The identifier of MOVIE is Movie Name. DVD does not have a unique identifier. However, Copy Number is a *partial* identifier, which, together with Movie Name, would uniquely identify an instance of DVD.

Some Examples of Relationships and Their Cardinalities

Examples of three relationships that show all possible combinations of minimum and maximum cardinalities appear in Figure 2-17. Each example states the business rule for each cardinality constraint and shows the associated E-R notation. Each example also shows some relationship instances to clarify the nature of the relationship. You should

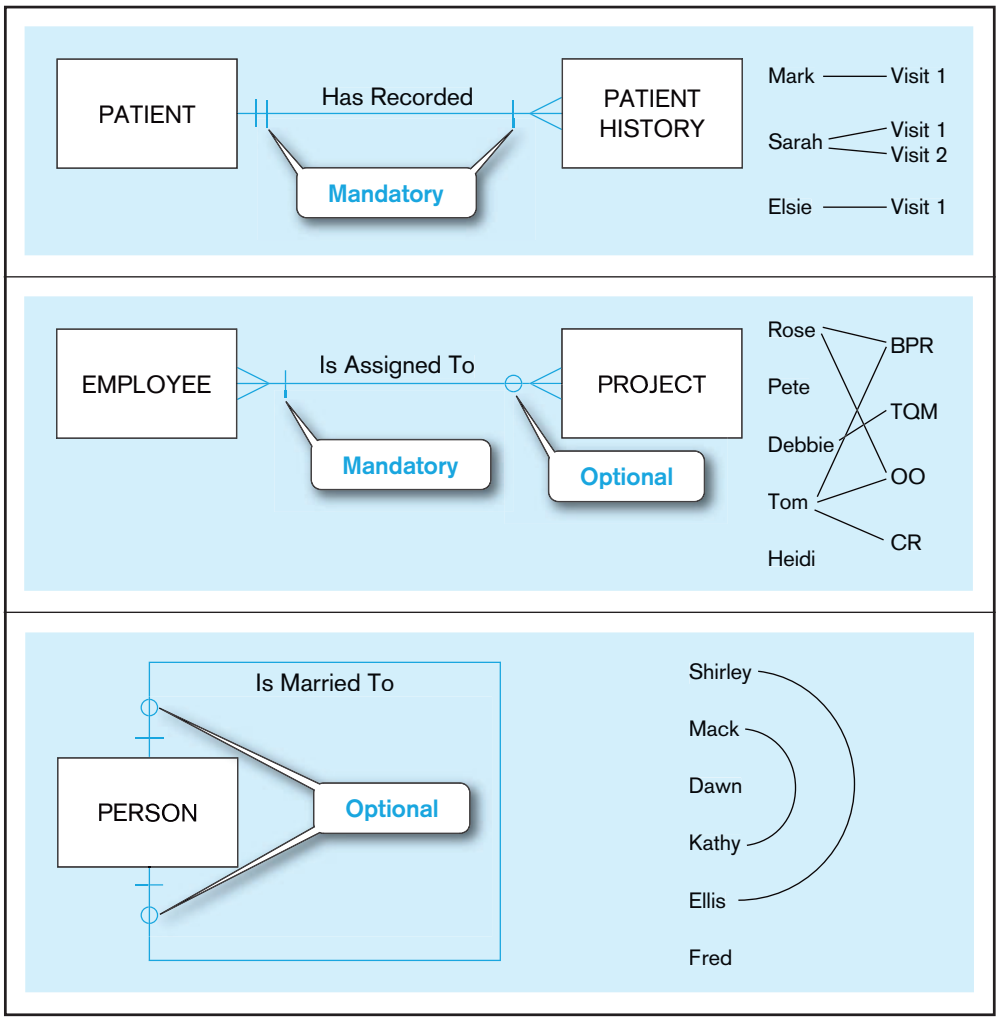


FIGURE 2-17 Examples of cardinality constraints
(a) Mandatory cardinalities

(b) One optional, one mandatory cardinality

(c) Optional cardinalities

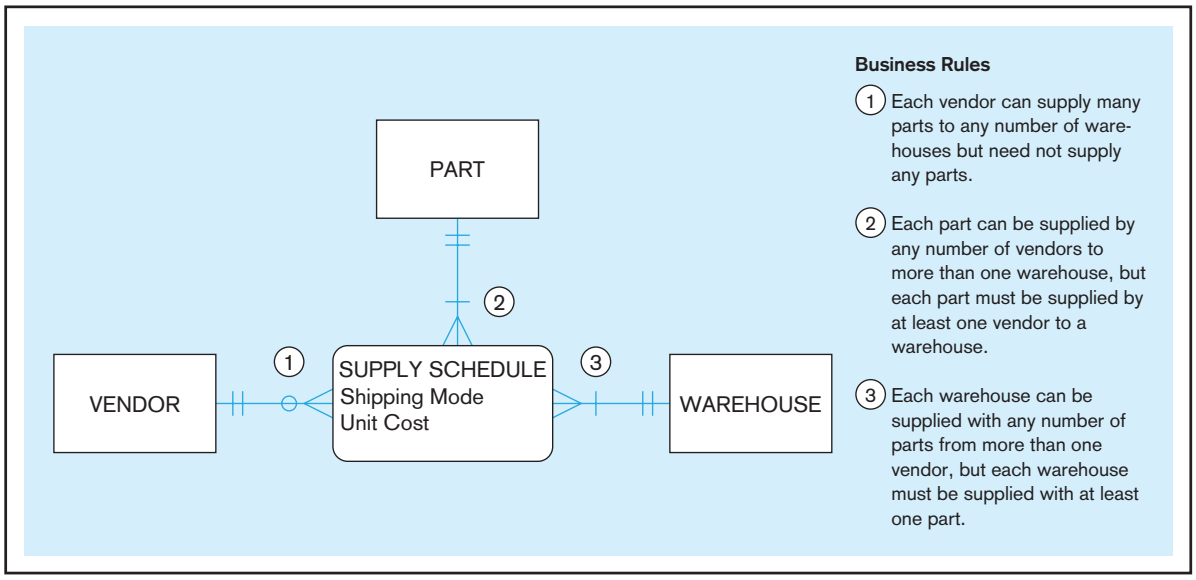
study each of these examples carefully. Following are the business rules for each of the examples in Figure 2-17:

1. ***PATIENT Has Recorded PATIENT HISTORY (Figure 2-17a)*** Each patient has one or more patient histories. (The initial patient visit is always recorded as an instance of PATIENT HISTORY.) Each instance of PATIENT HISTORY "belongs to" exactly one PATIENT.
2. ***EMPLOYEE Is Assigned To PROJECT (Figure 2-17b)*** Each PROJECT has at least one EMPLOYEE assigned to it. (Some projects have more than one.) Each EMPLOYEE may or (optionally) may not be assigned to any existing PROJECT (e.g., employee Pete), or may be assigned to one or more PROJECTs.
3. ***PERSON Is Married To PERSON (Figure 2-17c)*** This is an optional zero or one cardinality in both directions, because a person may or may not be married at a given point in time.

It is possible for the maximum cardinality to be a fixed number, not an arbitrary "many" value. For example, suppose corporate policy states that an employee may work on at most five projects at the same time. We could show this business rule by placing a 5 above or below the crow's foot next to the PROJECT entity in Figure 2-17b.

A TERNARY RELATIONSHIP We showed the ternary relationship with the associative entity type SUPPLY SCHEDULE in Figure 2-14. Now let's add cardinality constraints to this diagram, based on the business rules for this situation. The E-R diagram, with the relevant business rules, is shown in Figure 2-18. Notice that PART and

FIGURE 2-18 **Cardinality constraints in a ternary relationship**



WAREHOUSE must relate to some SUPPLY SCHEDULE instance, and a VENDOR optionally may not participate. The cardinality at each of the participating entities is a mandatory one, because each SUPPLY SCHEDULE instance must be related to exactly one instance of each of these participating entity types. (Remember, SUPPLY SCHEDULE is an associative entity.)

As noted earlier, a ternary relationship is not equivalent to three binary relationships. Unfortunately, you are not able to draw ternary relationships with many CASE tools; instead, you are forced to represent ternary relationships as three binaries (i.e., an associative entity with three binary relationships). If you are forced to draw three binary relationships, then do not draw the binary relationships with names, and be sure that the cardinality next to the three strong entities is a mandatory one.

Modeling Time-Dependent Data

Database contents vary over time. With renewed interest today in traceability and reconstruction of a historical picture of the organization for various regulatory requirements, such as HIPAA and Sarbanes-Oxley, the need to include a time series of data has become essential. For example, in a database that contains product information, the unit price for each product may be changed as material and labor costs and market conditions change. If only the current price is required, Price can be modeled as a single-valued attribute. However, for accounting, billing, financial reporting, and other purposes, we are likely to need to preserve a history of the prices and the time period during which each was in effect. As Figure 2-19 shows, we can conceptualize this requirement as a series of prices and the effective date for each price. This results in the (composite) multivalued attribute named Price History, with components Price and Effective Date. An important characteristic of such a composite, multivalued attribute is that the component attributes go together. Thus, in Figure 2-19, each Price is paired with the corresponding Effective Date.

In Figure 2-19, each value of the attribute Price is time stamped with its effective date. A **time stamp** is simply a time value, such as date and time, that is associated with a data value. A time stamp may be associated with any data value that changes over time when we need to maintain a history of those data values. Time stamps may be recorded to indicate the time the value was entered (transaction time), the time the value becomes valid or stops being valid, or the time when critical actions were performed, such as updates, corrections, or audits. This situation is similar to the employee skill diagrams in Figure 2-15b; thus, an alternative, not shown in Figure 2-19, is to make Price History a separate entity type, as was done with Skill using Microsoft Visio.

Time stamp
A time value that is associated with a data value, often indicating when some event occurred that affected the data value.

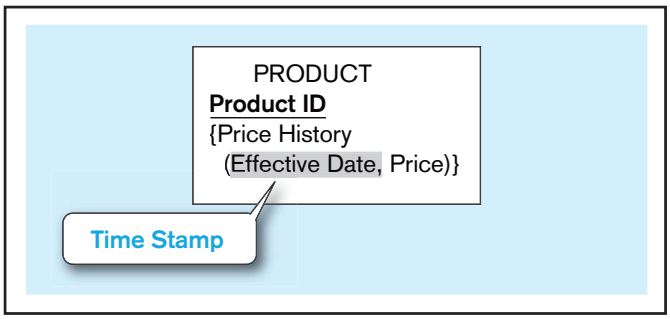
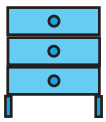


FIGURE 2-19 Simple example of time stamping

The use of simple time stamping (as in the preceding example) is often adequate for modeling time-dependent data. However, time can introduce subtler complexities to data modeling. For example, consider again Figure 2-17c. This figure is drawn for a given point in time, not to show history. If, on the other hand, we needed to record the full history of marriages for individuals, the Is Married To relationship would be an optional many-to-many relationship. Further, we might want to know the beginning and ending date (optional) of each marriage; these dates would be, similar to the bill-of-materials structure in Figure 2-13c, attributes of the relationship or associative entity.

Financial and other compliance regulations, such as Sarbanes-Oxley and Basel II, require that a database maintain history rather than just current status of critical data. In addition, some data modelers will argue that a data model should always be able to represent history, even if today users say they need only current values. These factors suggest that all relationships should be modeled as many-to-many (which is often done in purchased data model). Thus, for most databases, this will necessitate forming an associative entity along every relationship. There are two obvious negatives to this approach. First, many additional (associative) entities are created, thus cluttering ERDs. Second, a many-to-many (*M:N*) relationship is less restrictive than a one-to-many (1:*M*). So, if initially you want to enforce only one associated entity instance for some entity (i.e., the "one" side of the relationships), this cannot be enforced by the data model with an *M:N* relationship. It would seem likely that some relationships would never be *M:N*; for example, would a 1:*M* relationship between customer and order ever become *M:N* (but, of course, maybe someday our organization would sell items that would allow and often have joint purchasing, like vehicles or houses)? The conclusion is that if history or a time series of values might ever be desired or required by regulation, you should consider using an *M:N* relationship.



PINE VALLEY FURNITURE

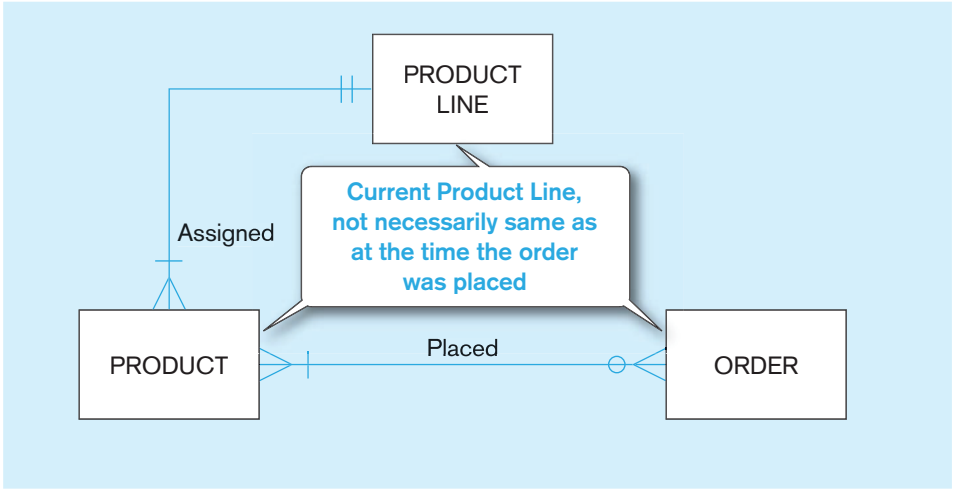
An even more subtle situation of the effect of time on data modeling is illustrated in Figure 2-20a, which represents a portion of an ERD for Pine Valley Furniture Company. Each product is assigned (i.e., current assignment) to a product line (or related group of products). Customer orders are processed throughout the year, and monthly summaries are reported by product line and by product within product line.

Suppose that in the middle of the year, due to a reorganization of the sales function, some products are reassigned to different product lines. The model shown in Figure 2-20a is not designed to track the reassignment of a product to a new product line. Thus, all sales reports will show cumulative sales for a product based on its current product line rather than the one at the time of the sale. For example, a product may have total year-to-date sales of $50,000 and be associated with product line B, yet $40,000 of those sales may have occurred while the product was assigned to product line A. This fact will be lost using the model in Figure 2-20a. The simple design change shown in Figure 2-20b will correctly recognize product reassignments. A new relationship, called Sales For Product Line, has been added between ORDER and PRODUCT LINE. As customer orders are processed, they are credited to both the correct product (via Sales For Product) and the correct product line (via Sales For Product Line) as of the time of the sale. The approach of Figure 2-20b is similar to what is done in a data warehouse to retain historical records of the precise situation at any point in time. (We will return to dealing with the time dimension in Chapter 9.)

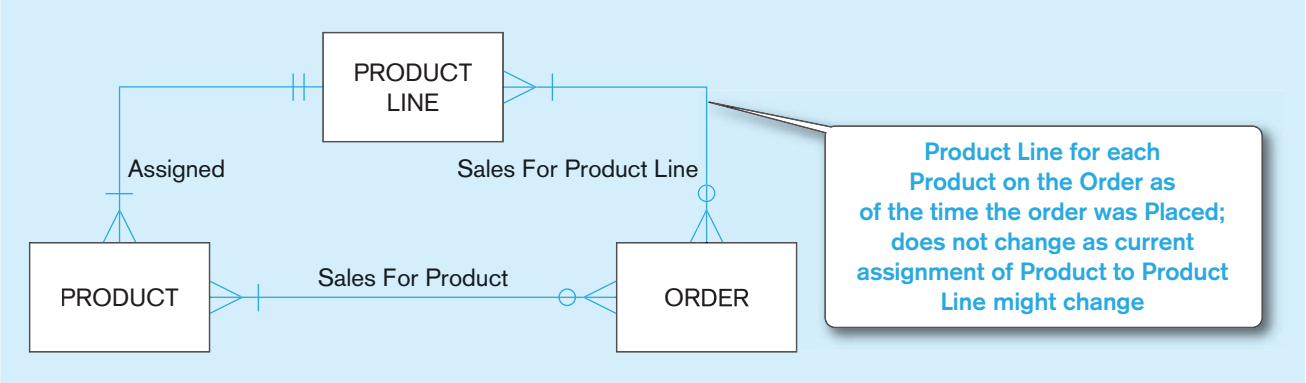
Another aspect of modeling time is recognizing that although the requirements of the organization today may be to record only the current situation, the design of

FIGURE 2-20 Example of time in Pine Valley Furniture product database

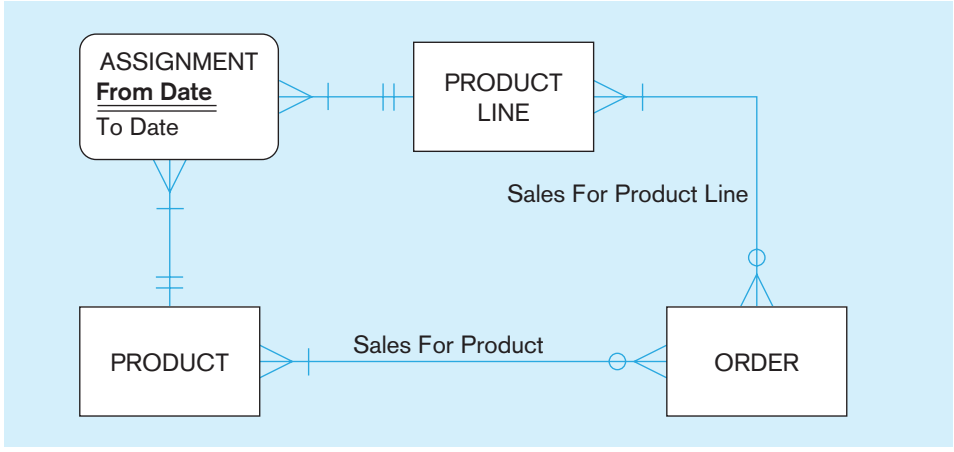
(a) E-R diagram not recognizing product reassignment



(b) E-R diagram recognizing product reassignment



(c) E-R diagram with associative entity for product assignment to product line over time



the database may need to change if the organization ever decides to keep history. In Figure 2-20b, we know the current product line for a product and the product line for the product each time it is ordered. But what if the product were ever reassigned to a product line during a period of zero sales for the product? Based on this data model in Figure 2-20b, we would not know of these other product line assignments. A common solution to this need for greater flexibility in the data model is to consider whether

a one-to-many relationship, such as Assigned, should become a many-to-many relationship. Further, to allow for attributes on this new relationship, this relationship should actually be an associative entity. Figure 2-20c shows this alternative data model with the ASSIGNMENT associative entity for the Assigned relationship. The advantage of the alternative is that we now will not miss recording any product line assignment, and we can record information about the assignment (such as the from and to effective dates of the assignment); the disadvantage is that the data model no longer has the restriction that a product may be assigned to only one product line at a time.

We have discussed the problem of time-dependent data with managers in several organizations who are considered leaders in the use of data modeling and database management. Before the recent wave of financial reporting disclosure regulations, these discussions revealed that data models for operational databases were generally inadequate for handing time-dependent data, and that organizations often ignored this problem and hoped that the resulting inaccuracies balanced out. However, with these new regulations, you need to be alert to the complexities posed by time-dependent data as you develop data models in your organization. For a thorough explanation of time as a dimension of data modeling, see a series of articles by T. Johnson and R. Weis beginning in May 2007 in *DM Review* (now *Information Management*) and accessible from the Magazine Archives section of the Information Center at of **www.information-management.com**.

Modeling Multiple Relationships Between Entity Types

There may be more than one relationship between the same entity types in a given organization. Two examples are shown in Figure 2-21. Figure 2-21a shows two relationships between the entity types EMPLOYEE and DEPARTMENT. In this figure we use the notation with names for the relationship in each direction; this notation makes explicit what the cardinality is for each direction of the relationship (which becomes important for clarifying the meaning of the unary relationship on EMPLOYEE). One relationship associates employees with the department in which they work. This relationship is one-to-many in the Has Workers direction and is mandatory in both directions. That is, a department must have at least one employee who works there (perhaps the department manager), and each employee must be assigned to exactly one department. (Note: These

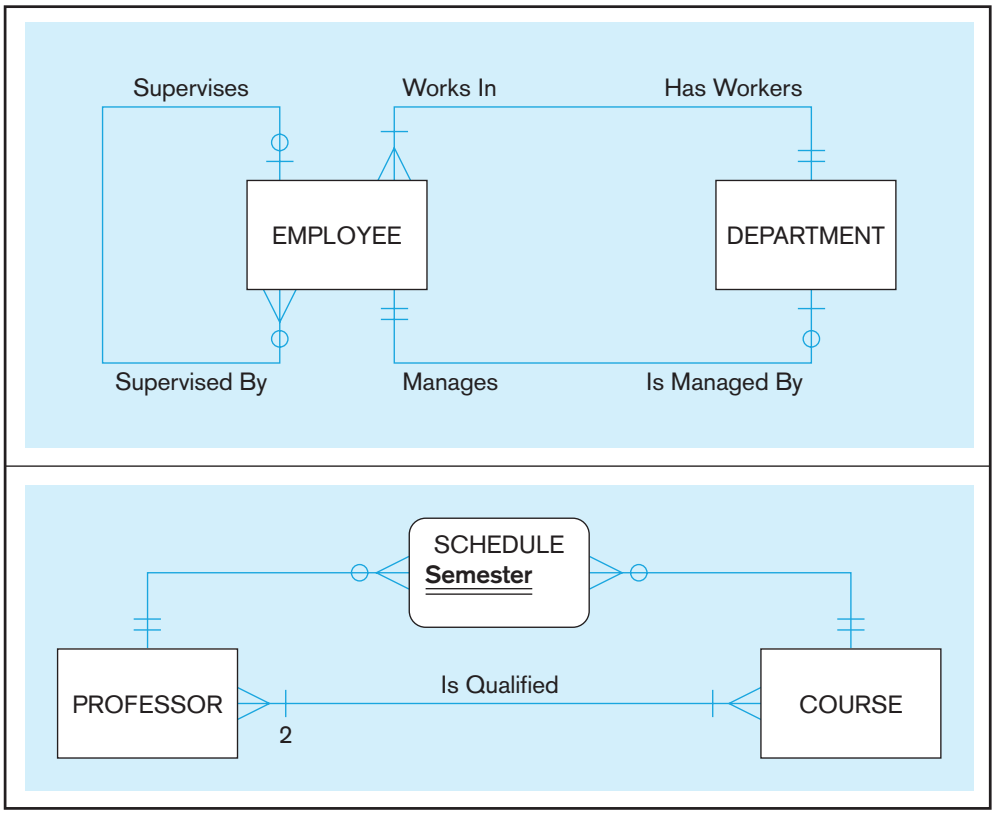


FIGURE 2-21 Examples of multiple relationships (a) Employees and departments

(b) Professors and courses (fixed lower limit constraint)

are specific business rules we assume for this illustration. It is crucial when you develop an E-R diagram for a particular situation that you understand the business rules that apply for that setting. For example, if EMPLOYEE were to include retirees, then each employee may not be currently assigned to exactly one department; further, the E-R model in Figure 2-21a assumes that the organization needs to remember in which DEPARTMENT each EMPLOYEE currently works, rather than remembering the history of department assignments. Again, the structure of the data model reflects the information the organization needs to remember.)

The second relationship between EMPLOYEE and DEPARTMENT associates each department with the employee who manages that department. The relationship from DEPARTMENT to EMPLOYEE (called Is Managed By in that direction) is a mandatory one, indicating that a department must have exactly one manager. From EMPLOYEE to DEPARTMENT, the relationship (Manages) is optional because a given employee either is or is not a department manager.

Figure 2-21a also shows the unary relationship that associates each employee with his or her supervisor, and vice versa. This relationship records the business rule that each employee may have exactly one supervisor (Supervised By). Conversely, each employee may supervise any number of employees, or may not be a supervisor.

The example in Figure 2-21b shows two relationships between the entity types PROFESSOR and COURSE. The relationship Is Qualified associates professors with the courses they are qualified to teach. A given course must have at a minimum two qualified instructors (an example of how to use a fixed value for a minimum or maximum cardinality). This might happen, for example, so that a course is never the "property" of one instructor. Conversely, each instructor must be qualified to teach at least one course (a reasonable expectation).

The second relationship in this figure associates professors with the courses they are actually scheduled to teach during a given semester. Because Semester is a characteristic of the relationship, we place an associative entity, SCHEDULE, between PROFESSOR and COURSE.

One final point about Figure 2-21b: Have you figured out what the identifier is for the SCHEDULE associative entity? Notice that Semester is a partial identifier; thus, the full identifier will be the identifier of PROFESSOR along with the identifier of COURSE as well as Semester. Because such full identifiers for associative entities can become long and complex, it is often recommended that surrogate identifiers be created for each associative entity; so, Schedule ID would be created as the identifier of SCHEDULE, and Semester would be an attribute. What is lost in this case is the explicit business rule that the combination of the PROFESSOR identifier, COURSE identifier, and Semester must be unique for each SCHEDULE instance (because this combination is the identifier of SCHEDULE). Of course, this can be added as another business rule.

Naming and Defining Relationships

In addition to the general guidelines for naming data objects, there are a few special guidelines for naming relationships, which follow:

- A relationship name is a *verb phrase* (such as Assigned To, Supplies, or Teaches). Relationships represent actions being taken, usually in the present tense, so transitive verbs (an action on something) are the most appropriate. A relationship name states the action taken, not the result of the action (e.g., use Assigned To, not Assignment). The name states the essence of the interaction between the participating entity types, not the process involved (e.g., use an Employee is *Assigned To* a project, not an Employee is *Assigning* a project).
- You should *avoid vague names*, such as Has or Is Related To. Use descriptive, powerful verb phrases, often taken from the action verbs found in the definition of the relationship.

There are also some specific guidelines for defining relationships, which follow:

- A relationship definition *explains what action is being taken and possibly why it is important*. It may be important to state who or what does the action, but it is not

important to explain how the action is taken. Stating the business objects involved in the relationship is natural, but because the E-R diagram shows what entity types are involved in the relationship and other definitions explain the entity types, you do not have to describe the business objects.

- It may also be important to *give examples to clarify the action*. For example, for a relationship of Registered For between student and course, it may be useful to explain that this covers both on-site and online registration and includes registrations made during the drop/add period.
- The definition should explain any *optional participation*. You should explain what conditions lead to zero associated instances, whether this can happen only when an entity instance is first created, or whether this can happen at any time. For example, "Registered For links a course with the students who have signed up to take the course, and the courses a student has signed up to take. A course will have no students registered for it before the registration period begins and may never have any registered students. A student will not be registered for any courses before the registration period begins and may not register for any classes (or may register for classes and then drop any or all classes)."
- A relationship definition should also *explain the reason for any explicit maximum cardinality* other than many. For example, "Assigned To links an employee with the projects to which that employee is assigned and the employees assigned to a project. Due to our labor union agreement, an employee may not be assigned to more than four projects at a given time." This example, typical of many upper-bound business rules, suggests that maximum cardinalities tend not to be permanent. In this example, the next labor union agreement could increase or decrease this limit. Thus, the implementation of maximum cardinalities must be done to allow changes.
- A relationship definition should *explain any mutually exclusive relationships*. Mutually exclusive relationships are ones for which an entity instance can participate in only one of several alternative relationships. We will show examples of this situation in Chapter 3. For now, consider the following example: "Plays On links an intercollegiate sports team with its student players and indicates on which teams a student plays. Students who play on intercollegiate sports teams cannot also work in a campus job (i.e., a student cannot be linked to both an intercollegiate sports team via Plays On and a campus job via the Works On relationship)." Another example of a mutually exclusive restriction is when an employee cannot both be Supervised By and be Married To the same employee.
- A relationship definition should *explain any restrictions on participation in the relationship*. Mutual exclusivity is one restriction, but there can be others. For example, "Supervised By links an employee with the other employees he or she supervises and links an employee with the other employee who supervises him or her. An employee cannot supervise him- or herself, and an employee cannot supervise other employees if his or her job classification level is below 4."
- A relationship definition should *explain the extent of history that is kept in the relationship*. For example, "Assigned To links a hospital bed with a patient. Only the current bed assignment is stored. When a patient is not admitted, that patient is not assigned to a bed, and a bed may be vacant at any given point in time." Another example of describing history for a relationship is "Places links a customer with the orders they have placed with our company and links an order with the associated customer. Only two years of orders are maintained in the database, so not all orders can participate in this relationship."
- A relationship definition should *explain whether an entity instance involved in a relationship instance can transfer participation to another relationship instance.* For example, "Places links a customer with the orders they have placed with our company and links an order with the associated customer. An order is not transferable to another customer." Another example is "Categorized As links a product line with the products sold under that heading and links a product to its associated product line. Due to changes in organization structure and product design features, products may be recategorized to a different product line. Categorized As keeps track of only the current product line to which a product is linked."

E-R MODELING EXAMPLE: PINE VALLEY FURNITURE COMPANY

Developing an E-R diagram can proceed from one (or both) of two perspectives. With a top-down perspective, the designer proceeds from basic descriptions of the business, including its policies, processes, and environment. This approach is most appropriate for developing a high-level E-R diagram with only the major entities and relationships and with a limited set of attributes (such as just the entity identifiers). With a bottom-up approach, the designer proceeds from detailed discussions with users, and from a detailed study of documents, screens, and other data sources. This approach is necessary for developing a detailed, "fully attributed" E-R diagram.

In this section, we develop a high-level ERD for Pine Valley Furniture Company, based largely on the first of these approaches (see Figure 2-22 for a Microsoft Visio version). For simplicity, we do not show any composite or multivalued attributes (e.g., skill is shown as a separate entity type associated with EMPLOYEE via an associative entity, which allows an employee to have many skills and a skill to be held by many employees).

Figure 2-22 provides many examples of common E-R modeling notations, and hence, it can be used as an excellent review of what you have learned in this chapter. In a moment, we will explain the business rules that are represented in this figure. However, before you read that explanation, one way to use Figure 2-22 is to search for typical E-R model constructs in it, such as one-to-many, binary, or unary relationships. Then, ask yourself why the business data was modeled this way. For example, ask yourself

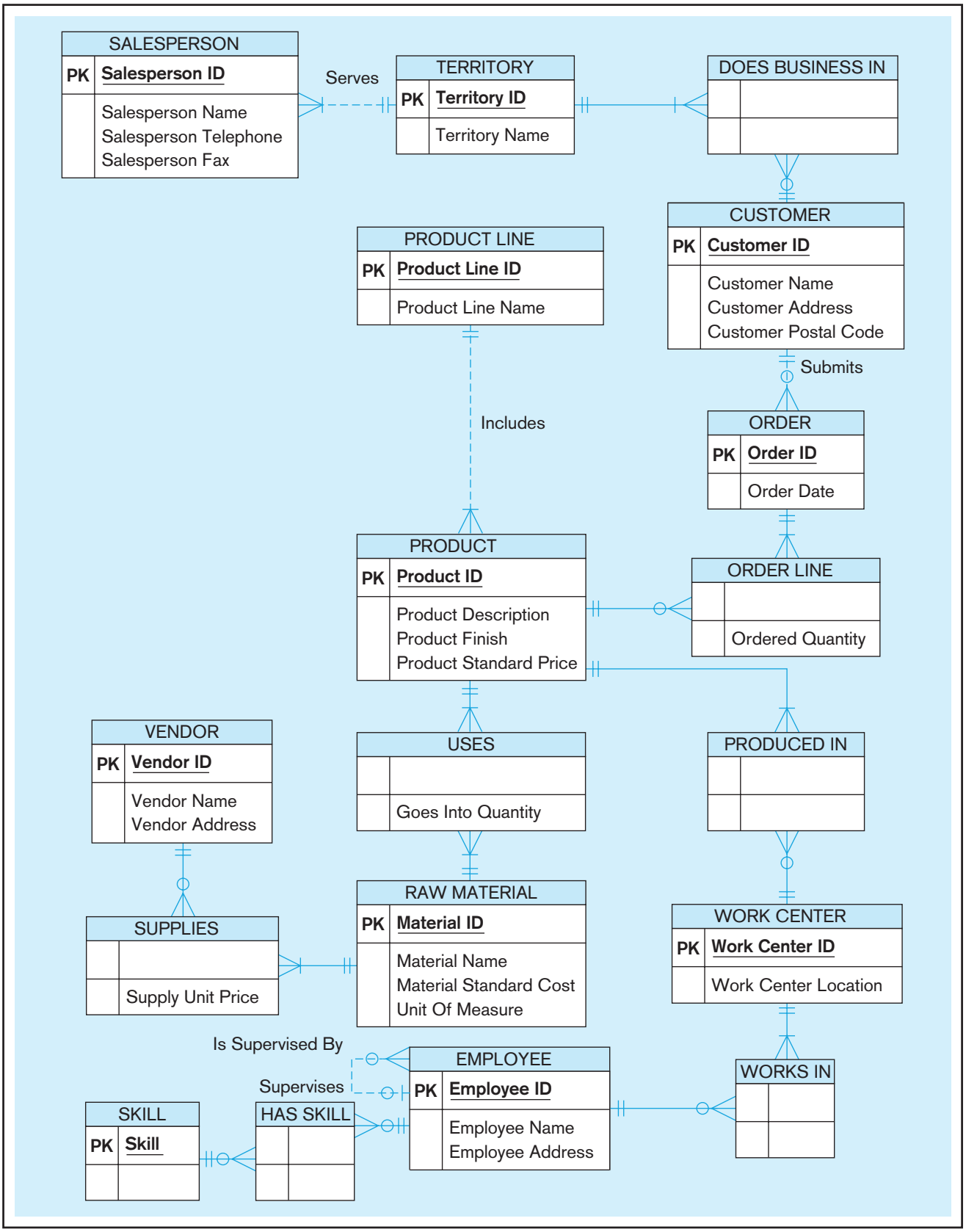
- Where is a unary relationship, what does it mean, and for what reasons might the cardinalities on it be different in other organizations?
- Why is Includes a one-to many relationship, and why might this ever be different in some other organization?
- Does Includes allow for a product to be represented in the database before it is assigned to a product line (e.g., while the product is in research and development)?
- If there were a different customer contact person for each sales territory in which a customer did business, where in the data model would we place this person's name?
- What is the meaning of the Does Business In associative entity, and why does each Does Business In instance have to be associated with exactly one SALES TERRITORY and one CUSTOMER?
- In what way might Pine Valley change the way it does business that would cause the Supplies associative entity to be eliminated and the relationships around it change?

Each of these questions is included in Problem and Exercise 1 at the end of the chapter, but we suggest you use these now as a way to review your understanding of E-R diagramming.

From a study of the business processes at Pine Valley Furniture Company, we have identified the following entity types. An identifier is also suggested for each entity, together with selected important attributes:

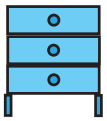
- The company sells a number of different furniture products. These products are grouped into several product lines. The identifier for a product is Product ID, whereas the identifier for a product line is Product Line ID. We identify the following additional attributes for product: Product Description, Product Finish, and Product Standard Price. Another attribute for product line is Product Line Name. A product line may group any number of products but must group at least one product. Each product must belong to exactly one product line.
- Customers submit orders for products. The identifier for an order is Order ID, and another attribute is Order Date. A customer may submit any number of orders, but need not submit any orders. Each order is submitted by exactly one customer. The identifier for a customer is Customer ID. Other attributes include Customer Name, Customer Address, and Customer Postal Code.
- A given customer order must request at least one product and only one product per order line item. Any product sold by Pine Valley Furniture may not appear on any order line item or may appear on one or more order line items. An attribute associated with each order line item is Ordered Quantity.

FIGURE 2-22 **Data model for Pine Valley Furniture Company in Microsoft Visio notation**



- Pine Valley Furniture has established sales territories for its customers. Each customer may do business in any number of these sales territories or may not do business in any territory. A sales territory has one to many customers. The identifier for a sales territory is Territory ID and an attribute of a Territory Name.
- Pine Valley Furniture Company has several salespersons. The identifier for a salesperson is Salesperson ID. Other attributes include Salesperson Name, Salesperson Telephone, and Salesperson Fax. A salesperson serves exactly one sales territory. Each sales territory is served by one or more salespersons.
- Each product is assembled from a specified quantity of one or more raw materials. The identifier for the raw material entity is Material ID. Other attributes include Unit Of Measure, Material Name, and Material Standard Cost. Each raw material is assembled into one or more products, using a specified quantity of the raw material for each product.
- Raw materials are supplied by vendors. The identifier for a vendor is Vendor ID. Other attributes include Vendor Name and Vendor Address. Each raw material can be supplied by one or more vendors. A vendor may supply any number of raw materials or may not supply any raw materials to Pine Valley Furniture. Supply Unit Price is the unit price a particular vendor supplies a particular raw material.
- Pine Valley Furniture has established a number of work centers. The identifier for a work center is Work Center ID. Another attribute is Work Center Location. Each product is produced in one or more work centers. A work center may be used to produce any number of products or may not be used to produce any products.
- The company has more than 100 employees. The identifier for employee is Employee ID. Other attributes include Employee Name, Employee Address, and Skill. An employee may have more than one skill. Each employee may work in one or more work centers. A work center must have at least one employee working in that center, but may have any number of employees. A skill may be possessed by more than one employee or possibly no employees.
- Each employee has exactly one supervisor; however, a manager has no supervisor. An employee who is a supervisor may supervise any number of employees, but not all employees are supervisors.

DATABASE PROCESSING AT PINE VALLEY FURNITURE



The purpose of the data model diagram in Figure 2-22 is to provide a conceptual design for the Pine Valley Furniture Company database. It is important to check the quality of such a design through frequent interaction with the persons who will use the database after it is implemented. An important and often performed type of quality check is to determine whether the E-R model can easily satisfy user requests for data and/or information. Employees at Pine Valley Furniture have many data retrieval and reporting requirements. In this section, we show how a few of these information requirements can be satisfied by database processing against the database shown in Figure 2-22.

We use the SQL database processing language (explained in Chapters 6 and 7) to state these queries. To fully understand these queries, you will need to understand concepts introduced in Chapter 4. However, a few simple queries in this chapter should help you to understand the capabilities of a database to answer important organizational questions and give you a jump-start toward understanding SQL queries in Chapter 6 as well as in later chapters.

Showing Product Information

Many different users have a need to see data about the products Pine Valley Furniture produces (e.g., salespersons, inventory managers, and product managers). One specific need is for a salesperson who wants to respond to a request from a customer for a list of products of a certain type. An example of this query is

> List all details for the various computer desks that are stocked by the company.

The data for this query are maintained in the PRODUCT entity (see Figure 2-22). The query scans this entity and displays all the attributes for products that contain the description Computer Desk.

The SQL code for this query is

```
SELECT *
FROM Product
WHERE ProductDescription LIKE "Computer Desk%";
```

Typical output for this query is

PRODUCTID	PRODUCTDESCRIPTION	PRODUCTFINISH	PRODUCTSTANDARDPRICE
3	Computer Desk 48"	Oak	375.00
8	Computer Desk 64"	Pine	450.00

SELECT * FROM Product says display all attributes of PRODUCT entities. The WHERE clause says to limit the display to only products whose description begins with the phrase Computer Desk.

Showing Product Line Information

Another common information need is to show data about Pine Valley Furniture product lines. One specific type of person who needs this information is a product manager. The following is a typical query from a territory sales manager:

```
List the details of products in product line 4.
```

The data for this query are maintained in the PRODUCT entity. As we explain in Chapter 4, the attribute Product Line ID will be added to the PRODUCT entity when a data model in Figure 2-22 is translated into a database that can be accessed via SQL. The query scans the PRODUCT entity and displays all attributes for products that are in the selected product line.

The SQL code for this query is

```
SELECT *
FROM Product
WHERE ProductLineID = 4;
```

Typical output for this query is

PRODUCTID	PRODUCTDESCRIPTION	PRODUCTFINISH	PRODUCTSTANDARDPRICE	PRODUCTONHAND	PRODUCTLINEID
18	Grandfather Clock	Oak	890.0000	0	4
19	Grandfather Clock	Oak	1100.0000	0	4

The explanation of this SQL query is similar to the explanation of the previous one.

Showing Customer Order Status

The previous two queries are relatively simple, involving data from only one table in each case. Often, data from multiple tables are needed in one information request. Although the previous query is simple, we did have to look through the whole database to find the entity and attributes needed to satisfy the request.

To simplify query writing and for other reasons, many database management systems support creating restricted views of a database suitable for the information needs

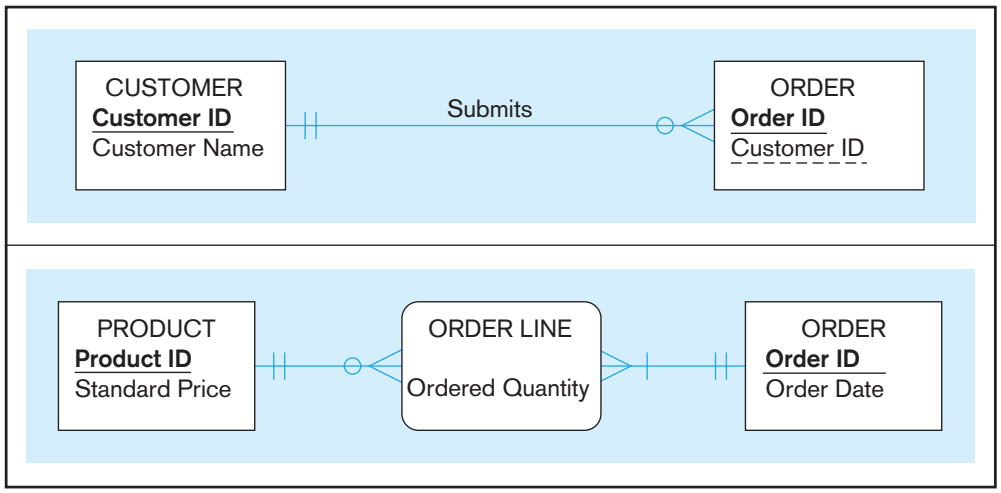


FIGURE 2-23 Two user views for Pine Valley Furniture
(a) User View 1: Orders for customers

(b) User View 2: Orders for products

of a particular user. For queries related to customer order status, Pine Valley utilizes such a user view called "Orders for customers," which is created from the segment of an E-R diagram for PVFC shown in Figure 2-23a. This user view allows users to see only CUSTOMER and ORDER entities in the database, and only the attributes of these entities shown in the figure. For the user, there is only one (virtual) table, ORDERS FOR CUSTOMERS, with the listed attributes. As we explain in Chapter 4, the attribute Customer ID will be added to the ORDER entity (as shown in Figure 2-23a). A typical order status query is

```
How many orders have we received from Value Furniture?
```

Assuming that all the data we need are pulled together into this one user view, or virtual entity, called OrdersForCustomers, we can simply write the query as follows:

```
SELECT COUNT(Order ID)
FROM OrdersForCustomers
WHERE CustomerName = "Value Furniture";
```

Without the user view, we can write the SQL code for this query in several ways. The way we have chosen is to compose a query within a query, called a *subquery*. (We will explain subqueries in Chapter 7, with some diagramming techniques to assist you in composting the query.) The query is performed in two steps. First, the subquery (or inner query) scans the CUSTOMER entity to determine the Customer ID for the customer named Value Furniture. (The ID for this customer is 5, as shown in the output for the previous query.) Then the query (or outer query) scans the ORDER entity and counts the order instances for this customer.

The SQL code for this query without the "Orders for customer" user view is as follows:

```
SELECT COUNT (OrderID)
FROM Order
WHERE CustomerID =
    (SELECT CustomerID
    FROM Customer
    WHERE CustomerName = "Value Furniture");
```

For this example query, using a subquery rather than a view did not make writing the query much more complex.

Typical output for this query using either of the query approaches above is

```
COUNT(ORDERID)
        4
```

Showing Product Sales

Salespersons, territory managers, product managers, production managers, and others have a need to know the status of product sales. One kind of sales question is what products are having an exceptionally strong sales month. Typical of this question is the following query:

> What products have had total sales exceeding $25,000 during the past month (June, 2009)?

This query can be written using the user view "Orders for products," which is created from the segment of an E-R diagram for PVFC shown in Figure 2-23b. Data to respond to the query are obtained from the following sources:

- Order Date from the ORDER entity (to find only orders in the desired month)
- Ordered Quantity for each product on each order from the associative entity ORDER LINE for an ORDER entity in the desired month
- Standard Price for the product ordered from the PRODUCT entity associated with the ORDER LINE entity

For each item ordered during the month of June 2009, the query needs to multiply Ordered Quantity by Product Standard Price to get the dollar value of a sale. For the user, there is only one (virtual) table, ORDERS FOR PRODUCTS, with the listed attributes. The total amount is then obtained for that item by summing all orders. Data are displayed only if the total exceeds $25,000.

The SQL code for this query is beyond the scope of this chapter, because it requires techniques introduced in Chapter 7. We introduce this query now only to suggest the power that a database such as the one shown in Figure 2-22 has to find information for management from detailed data. In many organizations today, users can use a Web browser to obtain the information described here. The programming code associated with a Web page then invokes the required SQL commands to obtain the requested information.

Summary

This chapter has described the fundamentals of modeling data in the organization. Business rules, derived from policies, procedures, events, functions, and other business objects, state constraints that govern the organization and, hence, how data are handled and stored. Using business rules is a powerful way to describe the requirements for an information system, especially a database. The power of business rules results from business rules being core concepts of the business, being able to be expressed in terms familiar to end users, being highly maintainable, and being able to be enforced through automated means, mainly through a database. Good business rules are ones that are declarative, precise, atomic, consistent, expressible, distinct, and business oriented.

Examples of basic business rules are data names and definitions. This chapter explained guidelines for the clear naming and definition of data objects in a business. In terms of conceptual data modeling, names and definitions must be provided for entity types, attributes, and relationships. Other business rules may state constraints on these data objects. These constraints can be captured in a data model and associated documentation.

The data modeling notation most frequently used today is the entity-relationship data model. An E-R model is a detailed, logical representation of the data for an organization. An E-R model is usually expressed in the form of an E-R diagram, which is a graphical representation of an E-R model. The E-R model was introduced by Chen in 1976. However, at the present time there is no standard notation for E-R modeling. Notations such as those found in Microsoft Visio are used in many CASE tools.

The basic constructs of an E-R model are entity types, relationships, and related attributes. An entity is a person, a place, an object, an event, or a concept in the user environment about which the organization wishes to maintain data. An entity type is a collection of entities that share common properties, whereas an entity instance is a single occurrence of an entity type. A strong entity type is an entity that has its own identifier and can exist without other entities. A weak entity type is an entity whose existence depends on the existence of a strong entity type. Weak entities do not have their own identifier, although they normally have a partial identifier. Weak entities are identified through an identifying relationship with their owner entity type.

An attribute is a property or characteristic of an entity or relationship that is of interest to the organization. There are several types of attributes. A required attribute must have a value for an entity instance, whereas an optional attribute value may be null. A simple attribute is one that has no component parts. A composite attribute is

an attribute that can be broken down into component parts. For example, Person Name can be broken down into the parts First Name, Middle Initial, and Last Name.

A multivalued attribute is one that can have multiple values for a single instance of an entity. For example, the attribute College Degree might have multiple values for an individual. A derived attribute is one whose values can be calculated from other attribute values. For example, Average Salary can be calculated from values of Salary for all employees.

An identifier is an attribute that uniquely identifies individual instances of an entity type. Identifiers should be chosen carefully to ensure stability and ease of use. Identifiers may be simple attributes, or they may be composite attributes with component parts.

A relationship type is a meaningful association between (or among) entity types. A relationship instance is an association between (or among) entity instances. The degree of a relationship is the number of entity types that participate in the relationship. The most common relationship types are unary (degree 1), binary (degree 2), and ternary (degree 3).

In developing E-R diagrams, we sometimes encounter many-to-many (and one-to-one) relationships that have one or more attributes associated with the relationship, rather than with one of the participating entity types. In such cases, we might consider converting the relationship to an associative entity. This type of entity associates the instances of one or more entity types and contains attributes that are peculiar to the relationship. Associative entity types may have their own simple identifier, or they may be assigned a composite identifier during logical design.

A cardinality constraint is a constraint that specifies the number of instances of entity B that may (or must) be associated with each instance of entity A. Cardinality constraints normally specify the minimum and maximum number of instances. The possible constraints are mandatory one, mandatory many, optional one, optional many, and a specific number. The minimum cardinality constraint is also referred to as the participation constraint. A minimum cardinality of zero specifies optional participation, whereas a minimum cardinality of one specifies mandatory participation.

Because many databases need to store the value of data over time, modeling time-dependent data is an important part of data modeling. Data that repeat over time may be modeled as multivalued attributes or as separate entity instances; in each case, a time stamp is necessary to identify the relevant date and time for the data value. Sometimes separate relationships need to be included in the data model to represent associations at different points in time. The recent wave of financial reporting disclosure regulations have made it more important to include time-sensitive and historical data in databases.

Chapter Review

Key Terms

Associative entity *80*
Attribute *72*
Binary relationship *82*
Business rule *63*
Cardinality constraint *86*
Composite attribute *73*
Composite identifier *75*
Degree *81*
Derived attribute *74*
Entity *68*
Entity instance *68*
Entity-relationship diagram (E-R diagram) *59*
Entity-relationship model (E-R model) *59*
Entity type *68*
Fact *66*
Identifier *74*
Identifying owner *70*
Identifying relationship *70*
Maximum cardinality *87*
Minimum cardinality *87*
Multivalued attribute *74*
Optional attribute *72*
Relationship instance *78*
Relationship type *78*
Required attribute *72*
Simple (or atomic) attribute *73*
Strong entity type *69*
Term *66*
Ternary relationship *83*
Time stamp *89*
Unary relationship *81*
Weak entity type *70*

Review Questions

1. Define each of the following terms:
a. entity type
b. entity-relationship model
c. entity instance
d. attribute
e. relationship type
f. identifier
g. multivalued attribute
h. associative entity
i. cardinality constraint
j. weak entity
k. identifying relationship
l. derived attribute
m. business rule

2. Match the following terms and definitions.

____composite attribute	a. uniquely identifies entity instances
____associative entity	b. relates instances of a single entity type
____unary relationship	c. specifies maximum and minimum number of instances
____weak entity	d. relationship modeled as an entity type
____attribute	e. association between entity types
____entity	f. collection of similar entities
____relationship type	g. number of participating entity types in relationship
____cardinality constraint	h. property of an entity
____degree	i. can be broken into component parts
____identifier	j. depends on the existence of another entity type
____entity type	k. relationship of degree 3
____ternary	l. many-to-many unary relationship
____bill-of-materials	m. person, place, object, concept, event

3. Contrast the following terms:
   a. stored attribute; derived attribute
   b. simple attribute; composite attribute
   c. entity type; relationship type
   d. strong entity type; weak entity type
   e. degree; cardinality
   f. required attribute; optional attribute
   g. composite attribute; multivalued attribute
   h. ternary relationship; three binary relationships
4. Give three reasons why many system designers believe that data modeling is the most important part of the systems development process.
5. Give four reasons why a business rules approach is advocated as a new paradigm for specifying information systems requirements.
6. Explain where you can find business rules in an organization.
7. State six general guidelines for naming data objects in a data model.
8. State four criteria for selecting identifiers for entities.
9. Why must some identifiers be composite rather than simple?
10. State three conditions that suggest the designer should model a relationship as an associative entity type.
11. List the four types of cardinality constraints, and draw an example of each.
12. Give an example, other than those described in this chapter, of a weak entity type. Why is it necessary to indicate an identifying relationship?
13. What is the degree of a relationship? List the three types of relationship degrees described in the chapter and give an example of each.
14. Give an example (other than those described in this chapter) for each of the following, and justify your answer:
    a. derived attribute
    b. multivalued attribute
    c. atomic attribute
    d. composite attribute
    e. required attribute
    f. optional attribute
15. Give an example of each of the following, other than those described in this chapter, and clearly explain why your example is this type of relationship and not of some other degree.
    a. ternary relationship
    b. unary relationship
16. Give an example of the use of effective (or effectivity) dates as attributes of an entity.
17. State a rule that says when to extract an attribute from one entity type and place it in a linked entity type.
18. What are the special guidelines for naming relationships?
19. In addition to explaining what action is being taken, what else should a relationship definition explain?
20. For the Manages relationship in Figure 2-12a, describe one or more situations that would result in different cardinalities on the two ends of this unary relationship. Based on your description for this example, do you think it is always clear simply from an E-R diagram what the business rule is that results in certain cardinalities? Justify your answer.
21. Explain the distinction between entity type and entity instance.
22. Why is it recommended that all ternary relationships be converted into an associative entity?

Problems and Exercises

1. Answer the following questions concerning Figure 2-22:
   a. Where is a unary relationship, what does it mean, and for what reasons might the cardinalities on it be different in other organizations?
   b. Why is Includes a one-to many relationship, and why might this ever be different in some other organization?
   c. Does Includes allow for a product to be represented in the database before it is assigned to a product line (e.g., while the product is in research and development)?
   d. If there is a rating of the competency for each skill an employee possesses, where in the data model would we place this rating?
   e. What is the meaning of the DOES BUSINESS IN associative entity, and why does each DOES BUSINESS IN instance have to be associated with exactly one TERRITORY and one CUSTOMER?
   f. In what way might Pine Valley change the way it does business that would cause the Supplies associative entity to be eliminated and the relationships around it to change?
2. There is a bulleted list associated with Figure 2-22 that describes the entities and their relationships in Pine Valley Furniture. For each of the 10 points in the list, identify the subset of Figure 2-22 described by that point.
3. You may have been assigned a CASE or a drawing tool to develop conceptual data models. Using this tool, attempt to redraw all the E-R diagrams in this chapter. What difficulties did you encounter? What E-R notations did not translate well to your tool? How did you incorporate the E-R notation that did not directly translate into the tool's notation?
4. Consider the two E-R diagrams in Figure 2-24, which represent a database of community service agencies and volunteers in two different cities (A and B). For each of the following three questions, place a check mark under City A, City B, or Can't Tell for the choice that is the best answer.

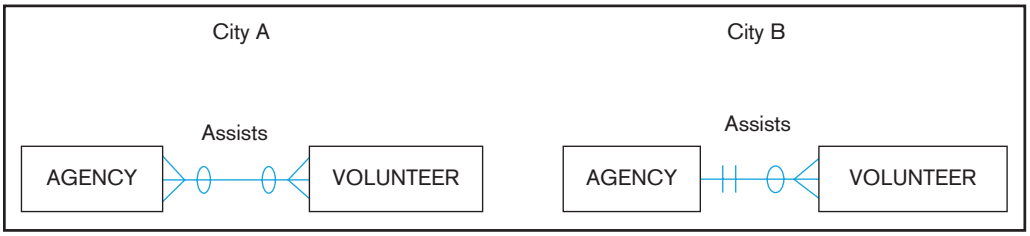


FIGURE 2-24 Diagram for Problem and Exercise 4

	City A	City B	Can't Tell
a. Which city maintains data about only those volunteers who currently assist agencies?			
b. In which city would it be possible for a volunteer to assist more than one agency?			
c. In which city would it be possible for a volunteer to change which agency or agencies he or she assists?			

5. The entity type STUDENT has the following attributes: Student Name, Address, Phone, Age, Activity, and No of Years. Activity represents some campus-based student activity, and No of Years represents the number of years the student has engaged in this activity. A given student may engage in more than one activity. Draw an ERD for this situation. What attribute or attributes did you designate as the identifier for the STUDENT entity? Why?
6. Are associative entities also weak entities? Why or why not? If yes, is there anything special about their "weakness"?
7. Because Visio does not explicitly show associative entities, it is not clear in Figure 2-22 which entity types are associative. List the associative entities in this figure. Why are there so many associative entities in Figure 2-22?
8. Figure 2-25 shows a grade report that is mailed to students at the end of each semester. Prepare an ERD reflecting the data contained in the grade report. Assume that each course is taught by one instructor. Also, draw this data model using the tool you have been told to use in the course. Explain what you chose for the identifier of each entity type on your ERD.
9. Add minimum and maximum cardinality notation to each of the following figures, as appropriate:
   a. Figure 2-5
   b. Figure 2-10a
   c. Figure 2-11b
   d. Figure 2-12 (all parts)
   e. Figure 2-13c
   f. Figure 2-14
10. The Is Married To relationship in Figure 2-12a would seem to have an obvious answer in Problem and Exercise 9d—that is, until time plays a role in modeling data. Draw a data model for the PERSON entity type and the Is Married To relationship for each of the following variations by showing the appropriate cardinalities and including, if necessary, any attributes:
    a. All we need to know is who a person is currently married to, if anyone. (This is likely what you represented in your answer to Problem and Exercise 9d.)
    b. We need to know who a person has ever been married to, if anyone.
    c. We need to know who a person has ever been married to, if anyone, as well as the date of their marriage and the date, if any, of the dissolution of their marriage.
    d. The same situation as in c, but now assume (which you likely did not do in c) that the same two people can remarry each other after a dissolution of a prior marriage to each other.
    e. In history, and even in some cultures today, there may be no legal restriction on the number of people to whom one can be currently married. Does your answer to part c of this Problem and Exercise handle this situation or must you make some changes (if so, draw a new ERD).
11. Figure 2-26 represents a situation of students who attend and work in schools and who also belong to certain clubs that are located in different schools. Study this diagram carefully to try to discern what business rules are represented.
    a. You will notice that cardinalities are not included on the Works For relationship. State a business rule for this relationship and then represent this rule with the cardinalities that match your rule.
    b. State a business rule that would make the Located In relationship redundant (i.e., where the school in which a club is located can be surmised or derived in some way from other relationships).
    c. Suppose a student could work for only a school that student attends but might not work. Would the Works For relationship still be necessary, or could you represent whether a student works for the school she attends in some other way (if so, how)?
12. Figure 2-27 shows two diagrams (A and B), both of which are legitimate ways to represent that a stock has a history of

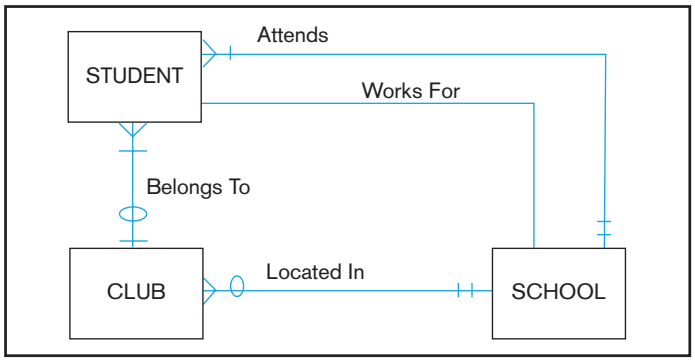
MILLENNIUM COLLEGE
GRADE REPORT
FALL SEMESTER 200X

NAME: Emily Williams ID: 268300458
CAMPUS ADDRESS: 208 Brooks Hall
MAJOR: Information Systems

COURSE ID	TITLE	INSTRUCTOR NAME	INSTRUCTOR LOCATION	GRADE
IS 350	Database Mgt.	Codd	B104	A
IS 465	System Analysis	Parsons	B317	B

FIGURE 2-25 **Grade report**

FIGURE 2-26 E-R diagram for Problem and Exercise 11



many prices. Which of the two diagrams do you consider a better way to model this situation and why?

13. Modify Figure 2-11a to model the following additional information requirements: The training director decides, for each employee who completes each class, who (what employees) should be notified of the course completion. The training director needs to keep track of which employees are notified about each course completion by a student. The date of notification is the only attribute recorded about this notification.

14. Review Figure 2-8 and Figure 2-22.

a. Identify any attributes in Figure 2-22 that might be composite attributes but are not shown that way. Justify your suggestions. Redraw the ERD to reflect any changes you suggest.

b. Identify any attributes in Figure 2-22 that might be multivalued attributes but are not shown that way. Justify your suggestions. Redraw the ERD to reflect any changes you suggest.

c. Is it possible for the same attribute to be both composite and multivalued? If no, justify your answer; if yes, give an example (Hint: Consider the CUSTOMER attributes in Figure 2-22).

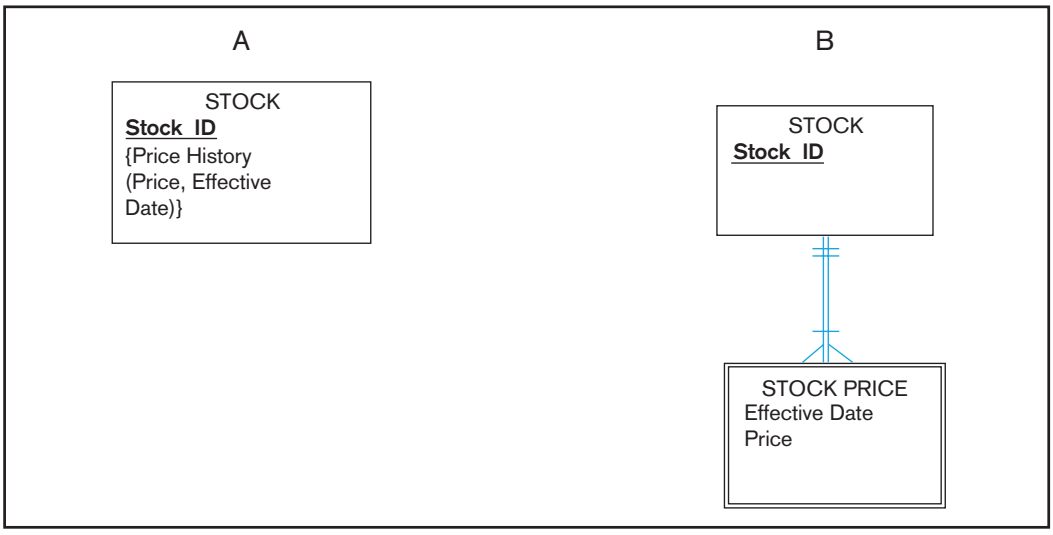
15. Draw an ERD for each of the following situations. (If you believe that you need to make additional assumptions, clearly state them for each situation.) Draw the same situation using the tool you have been told to use in the course.

a. A company has a number of employees. The attributes of EMPLOYEE include Employee ID (identifier), Name, Address, and Birthdate. The company also has several projects. Attributes of PROJECT include Project ID (identifier), Project Name, and Start Date. Each employee may be assigned to one or more projects, or may not be assigned to a project. A project must have at least one employee assigned and may have any number of employees assigned. An employee's billing rate may vary by project, and the company wishes to record the applicable billing rate (Billing Rate) for each employee when assigned to a particular project. Do the attribute names in this description follow the guidelines for naming attributes? If not, suggest better names. Do you have any associative entities on your ERD? If so, what are the identifiers for those associative entities? Does your ERD allow a project to be created before it has any employees assigned to it? Explain. How would you change your ERD if the Billing Rate could change in the middle of a project?

b. A laboratory has several chemists who work on one or more projects. Chemists also may use certain kinds of equipment on each project. Attributes of CHEMIST include Employee ID (identifier), Name, and Phone No. Attributes of PROJECT include Project ID (identifier) and Start Date. Attributes of EQUIPMENT include Serial No and Cost. The organization wishes to record Assign Date—that is, the date when a given equipment item was assigned to a particular chemist working on a specified project. A chemist must be assigned to at least one project and one equipment item. A given equipment item need not be assigned, and a given project need not be assigned either a chemist or an equipment item. Provide good definitions for all of the relationships in this situation.

c. A college course may have one or more scheduled sections, or may not have a scheduled section. Attributes of COURSE include Course ID, Course Name, and Units. Attributes of SECTION include Section Number

FIGURE 2-27 E-R diagram for Problem and Exercise 12



and Semester ID. Semester ID is composed of two parts: Semester and Year. Section Number is an integer (such as 1 or 2) that distinguishes one section from another for the same course but does not uniquely identify a section. How did you model SECTION? Why did you choose this way versus alternative ways to model SECTION?

d. A hospital has a large number of registered physicians. Attributes of PHYSICIAN include Physician ID (the identifier) and Specialty. Patients are admitted to the hospital by physicians. Attributes of PATIENT include Patient ID (the identifier) and Patient Name. Any patient who is admitted must have exactly one admitting physician. A physician may optionally admit any number of patients. Once admitted, a given patient must be treated by at least one physician. A particular physician may treat any number of patients, or may not treat any patients. Whenever a patient is treated by a physician, the hospital wishes to record the details of the treatment (Treatment Detail). Components of Treatment Detail include Date, Time, and Results. Did you draw more than one relationship between physician and patient? Why or why not? Did you include hospital as an entity type? Why or why not? Does your ERD allow for the same patient to be admitted by different physicians over time? How would you include on the ERD the need to represent the date on which a patient is admitted for each time they are admitted?

e. The loan office in a bank receives from various parties requests to investigate the credit status of a customer. Each credit request is identified by a Request ID and is described by a Request Date and Requesting Party Name. The loan office also received results of credit checks. A credit check is identified by a Credit Check ID and is described by the Credit Check Date and the Credit Rating. The loan office matches credit requests with credit check results. A credit request may be recorded before its result arrives; a particular credit result may be used in support of several credit requests. Draw an ERD for this situation. Now, assume that credit results may not be reused for multiple credit requests. Redraw the ERD for this new situation using two entity types, and then redraw it again using one entity type. Which of these two versions do you prefer, and why?

f. Companies, identified by Company ID and described by Company Name and Industry Type, hire consultants, identified by Consultant ID and described by Consultant Name, Consultant Specialty, which is multivalued. Assume that a consultant can work for only one company at a time, and we need to track only current consulting engagements. Draw an ERD for this situation. Now, consider a new attribute, Hourly Rate, which is the rate a consultant charges a company for each hour of his or her services. Redraw the ERD to include this new attribute. Now, consider that each time a consultant works for a company, a contract is written describing the terms for this consulting engagement. Contract is identified by a composite identifier of Company ID, Consultant ID, and Contract Date. Assuming that a consultant can still work for only one company at a time, redraw the ERD for this new situation. Did you move any attributes to different entity types in this latest situation? As a final situation, now consider that although a consultant can work for only one company at a time, we now need to keep the complete history of all consulting engagements for each consultant and company. Draw an ERD for this final situation. Explain why these different changes to the situation led to different data models, if they did.

g. An art museum owns a large volume of works of art. Each work of art is described by an item code (identifier), title, type, and size; size is further composed of height, width, and weight. A work of art is developed by an artist, but the artist for some works is unknown. An artist is described by an artist ID (identifier), name, date of birth, and date of death (which is null for still living artists). Only data about artists for works currently owned by the museum are kept in the database. At any point in time, a work of art is either on display at the museum, held in storage, away from the museum as part of a traveling show, or on loan to another gallery. If on display at the museum, a work of art is also described by its location within the museum. A traveling show is described by a show ID (identifier), the city in which the show is currently appearing, and the start and end dates of the show. Many of the museum works may be part of a given show, and only active shows with at least one museum work of art need be represented in the database. Finally, another gallery is described by a gallery ID (identifier), name, and city. The museum wants to retain a complete history of loaning a work of art to other galleries, and each time a work is loaned, the museum wants to know the date the work was loaned and the date it was returned. As you develop the ERD for this problem, follow good data naming guidelines.

h. Each case handled by the law firm of Dewey, Cheetim, and Howe has a unique case number; a date opened, date closed, and judgment description are also kept on each case. A case is brought by one or more plaintiffs, and the same plaintiff may be involved in many cases. A plaintiff has a requested judgment characteristic. A case is against one or more defendants, and the same defendant may be involved in many cases. A plaintiff or defendant may be a person or an organization. Over time, the same person or organization may be a defendant or a plaintiff in cases. In either situation, such legal entities are identified by an entity number, and other attributes are name and net worth. As you develop the ERD for this problem, follow good data naming guidelines.

i. Each publisher has a unique name; a mailing address and telephone number are also kept on each publisher. A publisher publishes one or more books; a book is published by exactly one publisher. A book is identified by its ISBN, and other attributes are title, price, and number of pages. Each book is written by one or more authors; an author writes one or more books, potentially for different publishers. Each author is uniquely described by an author ID, and we know each author's name and address. Each author is paid a certain royalty rate on each book he or she

authors, which potentially varies for each book and for each author. An author receives a separate royalty check for each book he or she writes. Each check is identified by its check number, and we also keep track of the date and amount of each check. As you develop the ERD for this problem, follow good data naming guidelines.

16. Assume that at Pine Valley Furniture each product (described by product number, description, and cost) comprises at least three components (described by component number, description, and unit of measure), and components are used to make one or many products. In addition, assume that components are used to make other components and that raw materials are also considered to be components. In both cases of components, we need to keep track of how many components go into making something else. Draw an ERD for this situation, and place minimum and maximum cardinalities on the diagram. Also, draw a data model for this situation using the tool you have been told to use in your course.
17. Emerging Electric wishes to create a database with the following entities and attributes:
    - Customer, with attributes Customer ID, Name, Address (Street, City, State, Zip Code), and Telephone
    - Location, with attributes Location ID, Address (Street, City, State, Zip Code), and Type (values of Business or Residential)
    - Rate, with attributes Rate Class and RatePerKWH

    After interviews with the owners, you have come up with the following business rules:
    - Customers can have one or more locations.
    - Each location can have one or more rates, depending on the time of day.

    Draw an ERD for this situation and place minimum and maximum cardinalities on the diagram. Also, draw a data model for this situation using the tool you have been told to use in your course. State any assumptions that you have made.
18. Each semester, each student must be assigned an adviser who counsels students about degree requirements and helps students register for classes. Each student must register for classes with the help of an adviser, but if the student's assigned adviser is not available, the student may register with any adviser. We must keep track of students, the assigned adviser for each, and the name of the adviser with whom the student registered for the current term. Represent this situation of students and advisers with an E-R diagram. Also, draw a data model for this situation using the tool you have been told to use in your course.
19. In the chapter, when describing Figure 2-4a, it was argued that the Received and Summarizes relationships and TREASURER entity were not necessary. Within the context of this explanation, this is true. Now, consider a slightly different situation. Suppose it is necessary, for compliance purposes (e.g., Sarbanes-Oxley compliance), to know when each expense report was produced and which officers (not just the treasurer) received each expense report and when they each signed off on that report. Redraw Figure 2-4a, now including any attributes and relationships required for this revised situation.
20. Prepare an ERD for a real estate firm that lists property for sale. Also prepare a definition for each entity type, attribute, and relationship on your diagram. In addition, draw a data model for this situation using the tool you have been told to use in your course. The following describes this organization:
    - The firm has a number of sales offices in several states. Attributes of sales office include Office Number (identifier) and Location.
    - Each sales office is assigned one or more employees. Attributes of employee include Employee ID (identifier) and Employee Name. An employee must be assigned to only one sales office.
    - For each sales office, there is always one employee assigned to manage that office. An employee may manage only the sales office to which he or she is assigned.
    - The firm lists property for sale. Attributes of property include Property ID (identifier) and Location. Components of Location include Address, City, State, and Zip Code.
    - Each unit of property must be listed with one (and only one) of the sales offices. A sales office may have any number of properties listed or may have no properties listed.
    - Each unit of property has one or more owners. Attributes of owners are Owner ID (identifier) and Owner Name. An owner may own one or more units of property. An attribute of the relationship between property and owner is Percent Owned.
21. After completing a course in database management, you are asked to develop a preliminary ERD for a symphony orchestra. You discover the following entity types that should be included:

CONCERT SEASON	The season during which a series of concerts will be performed. Identifier is Opening Date, which includes Month, Day, and Year.
CONCERT	A given performance of one or more compositions. Identifier is Concert Number. Another important attribute is Concert Date, which consists of the following: Month, Day, Year, and Time. Each concert typically has more than one concert date.
COMPOSITION	Compositions to be performed at each concert. Identifier is Composition ID, which consists of the following: Composer Name and Composition Name. Another attribute is Movement ID, which consists of two parts: Movement Number and Movement Name. Many, but not all, compositions have multiple movements.
CONDUCTOR	Person who will conduct the concert. Identifier is Conductor ID. Another attribute is Conductor Name.
SOLOIST	Solo artist who performs a given composition on a particular concert. Identifier is Soloist ID. Another attribute is Soloist Name.

During further discussions you discover the following:

- A concert season schedules one or more concerts. A particular concert is scheduled for only one concert season.
- A concert includes the performance of one or more compositions. A composition may be performed at one or more concerts or may not be performed.
- For each concert there is one conductor. A conductor may conduct any number of concerts or may not conduct any concerts.
- Each composition may require one or more soloists or may not require a soloist. A soloist may perform one or more compositions at a given concert or may not perform any composition. The symphony orchestra wishes to record the date when a soloist last performed a given composition (Date Last Performed).

Draw an ERD to represent what you have discovered. Identify a business rule in this description and explain how this business rule is modeled on the E-R diagram. Also draw a data model for this situation using the tool you have been told to use in your course.

22. Obtain several common user views such as a credit card receipt, credit card statement, and annual summary, or some other common document from one organization with which you interact.

a. Prepare an ERD for one of these documents. Also prepare a data model for this document, using the tool you have been told to use in your course.

b. Prepare an ERD for another of these documents. Also prepare a data model for this document, using the tool you have been told to use in your course.

c. Do you find the same entities, attributes, and relationships in the two ERDs you developed for parts a and b? What differences do you find in modeling the same data entities, attributes, and relationships between the two ERDs? Can you combine the two ERDs into one ERD for which the original two are subsets? Do you encounter any issues in trying to combine the ERDs? Suggest some issues that might arise if two different data modelers had independently developed the two data models.

d. How might you use data naming and definition standards to overcome the issues you identified in part c?

23. Draw an ERD for the following situation (Batra et al., 1988). Also, develop the list of words for qualifiers and classes that you use to form attribute names. Explain why you chose the words on your list. Also, draw a data model for this situation using the tool you have been told to use in your course.

> Projects, Inc., is an engineering firm with approximately 500 employees. A database is required to keep track of all employees, their skills, projects assigned, and departments worked in. Every employee has a unique number assigned by the firm and is required to store his or her name and date of birth. If an employee is currently married to another employee of Projects, Inc., the date of marriage and who is married to whom must be stored; however, no record of marriage is required if an employee's spouse is not also an employee. Each employee is given a job title (e.g., engineer, secretary, and so on). An employee does only one type of job at any given time, and we only need to retain information for an employee's current job.
>
> There are 11 different departments, each with a unique name. An employee can report to only 1 department. Each department has a phone number.
>
> To procure various kinds of equipment, each department deals with many vendors. A vendor typically supplies equipment to many departments. We are required to store the name and address of each vendor and the date of the last meeting between a department and a vendor.
>
> Many employees can work on a project. An employee can work on many projects (e.g., Southwest Refinery, California Petrochemicals, and so on) but can only be assigned to at most one project in a given city. For each city, we are interested in its state and population. An employee can have many skills (preparing material requisitions, checking drawings, and so on), but she or he may use only a given set of skills on a particular project. (For example, an employee MURPHY may prepare requisitions for the Southwest Refinery project and prepare requisitions as well as check drawings for California Petrochemicals.) Employees use each skill that they possess in at least one project. Each skill is assigned a number, and we must store a short description of each skill. Projects are distinguished by project numbers, and we must store the estimated cost of each project.

24. Draw an ERD for the following situation. (State any assumptions you believe you have to make in order to develop a complete diagram.) Also, draw a data model for this situation using the tool you have been told to use in your course: Stillwater Antiques buys and sells one-of-a-kind antiques of all kinds (e.g., furniture, jewelry, china, and clothing). Each item is uniquely identified by an item number and is also characterized by a description, asking price, condition, and open-ended comments. Stillwater works with many different individuals, called clients, who sell items to and buy items from the store. Some clients only sell items to Stillwater, some only buy items, and some others both sell and buy. A client is identified by a client number and is also described by a client name and client address. When Stillwater sells an item in stock to a client, the owners want to record the commission paid, the actual selling price, sales tax (tax of zero indicates a tax exempt sale), and date sold. When Stillwater buys an item from a client, the owners want to record the purchase cost, date purchased, and condition at time of purchase.

25. Draw an ERD for the following situation. (State any assumptions you believe you have to make in order to develop a complete diagram.) Also, draw a data model for this situation using the tool you have been told to use in your course: The H. I. Topi School of Business operates international business programs in 10 locations throughout Europe. The school had its first class of 9,000 graduates in 1965. The school keeps track of each graduate's student number, name when a student, country of birth, current country of citizenship, current name, and current address, as well as the name of each major the student completed. (Each student has one or two majors.) To maintain strong ties to its alumni, the school holds various events around the world. Events have a title, date, location, and type (e.g., reception, dinner, or seminar). The school needs to keep track of which graduates have attended which events. For an attendance by a graduate at an event, a comment is recorded about information school officials learned

from that graduate at that event. The school also keeps in contact with graduates by mail, e-mail, telephone, and fax interactions. As with events, the school records information learned from the graduate from each of these contacts. When a school official knows that he or she will be meeting or talking to a graduate, a report is produced showing the latest information about that graduate and the information learned during the past two years from that graduate from all contacts and events the graduate attended.

26. Wally Los Gatos, owner of Wally's Wonderful World of Wallcoverings, has hired you as a consultant to design a database management system for his chain of three stores that sell wallpaper and accessories. He would like to track sales, customers, and employees. After an initial meeting with Wally, you have developed a list of business rules and specifications to begin the design of an E-R model:

- Customers place orders through a branch.
- Wally would like to track the following about customers: Name, Address, City, State, Zip Code, Telephone, Date of Birth, and Primary Language.
- A customer may place many orders.
- A customer does not always have to order through the same branch all the time.
- Customers may have one or more accounts, although they may also have no accounts.
- The following information needs to be recorded about accounts: Balance, Last payment date, Last payment amount, and Type.
- A branch may have many customers.
- The following information about each branch needs to be recorded: Branch Number, Location (Address, City, State, Zip Code), and Square Footage.
- A branch may sell all items or may only sell certain items.
- Orders are composed of one or more items.
- The following information about each order needs to be recorded: Order Date and Credit Authorization Status.
- Items may be sold by one or more branches.
- We wish to record the following about each item: Description, Color, Size, Pattern, and Type.
- An item can be composed of multiple items; for example, a dining room wallcovering set (item 20) may consist of wallpaper (item 22) and borders (item 23).
- Wally employs 56 employees.
- He would like to track the following information about employees: Name, Address (Street, City, State, Zip Code), Telephone, Date of Hire, Title, Salary, Skill, and Age.
- Each employee works in one and only one branch.
- Each employee may have one or more dependents. We wish to record the name of the dependent as well as the age and relationship.
- Employees can have one or more skills.

Based upon this information, draw an E-R model. Please indicate any assumptions that you have made. Also, draw a data model for this situation using the tool you have been told to use in your course.

27. Our friend Wally Los Gatos (see Problem and Exercise 26), realizing that his wallcovering business had a few wrinkles in it, decided to pursue a law degree at night. After graduating, he has teamed up with Lyla El Pàjaro to form Peck and Paw, Attorneys at Law. Wally and Lyla have hired you to design a database system based upon the following set of business rules. It is in your best interest to perform a thorough analysis, to avoid needless litigation. Please create an ERD based upon the following set of rules:

- An ATTORNEY is retained by one or more CLIENTS for each CASE.
- Attributes of ATTORNEY are Attorney ID, Name, Address, City, State, Zip Code, Specialty (may be more than one), and Bar (may be more than one).
- A CLIENT may have more than one ATTORNEY for each CASE.
- Attributes of CLIENT are Client ID, Name, Address, City, State, Zip Code, Telephone, and Date of Birth.
- A CLIENT may have more than one CASE.
- Attributes of CASE are Case ID, Case Description, and Case Type.
- An ATTORNEY may have more than one CASE.
- Each CASE is assigned to one and only one COURT.
- Attributes of COURT are Court ID, Court Name, City, State, and Zip Code.
- Each COURT has one or more JUDGES assigned to it.
- Attributes of JUDGE are Judge ID, Name, and Years In Practice.
- Each JUDGE is assigned to exactly one court.

Please state any assumptions that you have made. Also, draw a data model for this situation using the tool you have been told to use in your course.

28. Review your answer to Problem and Exercise 25; if necessary, change the names of the entities, attributes, and relationships to conform to the naming guidelines presented in this chapter. Then, using the definition guidelines, write a definition for each entity, attribute, and relationship. If necessary, state assumptions so that each definition is as complete as possible.

Field Exercises

1. Interview a database analyst or systems analyst and document how he or she decides on names for data objects in data models. Does the organization in which this person works have naming guidelines? If so, describe the pattern used. If there are no guidelines, ask whether your contact has ever had any problems because guidelines did not exist. Does the organization use any tool to help management metadata, including data names?

2. Visit two local small businesses, one in the service sector (e.g., dry cleaner, auto repair shop, veterinarian, or bookstore) and one that manufactures tangible goods. Interview employees from these organizations to elicit from them the entities, attributes, and relationships that are commonly encountered in these organizations. Use this information to construct E-R diagrams. What differences and similarities are there between the diagrams for the service- and the product-oriented companies? Does the E-R diagramming technique handle both situations equally well? Why or why not?

3. Ask a database or systems analyst to give you examples of unary, binary, and ternary relationships that the analyst has

dealt with personally at his or her company. Ask which is most common and why.

4. Ask a database or systems analyst in a local company to show you an E-R diagram for one of the organization's primary databases. Ask questions to be sure you understand what each entity, attribute, and relationship means. Does this organization use the same E-R notation used in this text? If not, what other or alternative symbols are used and what do these symbols mean? Does this organization model associative entities on the E-R diagram? If not, how are associative entities modeled? What metadata are kept about the objects on the E-R diagram?
5. For the same E-R diagram used in Field Exercise 4 or for a different database in the same or a different organization, identify any uses of time stamping or other means to model time- dependent data. Why are time-dependent data necessary for those who use this database? Would the E-R diagram be much simpler if it were not necessary to represent the history of attribute values?
6. Search on the Internet for products that help document and manage business rules, standards, and procedures. One such site is **www.axisboulder.com**. Choose a couple tools and summarize their capabilities and discuss how they would be useful in managing business rules.

References

Aranow, E. B. 1989. "Developing Good Data Definitions." *Database Programming & Design* 2,8 (August): 36–39.

Batra, D., J. A. Hoffer, and R. B. Bostrom. 1988. "A Comparison of User Performance Between the Relational and Extended Entity Relationship Model in the Discovery Phase of Database Design." *Proceedings of the Ninth International Conference on Information Systems*. Minneapolis, November 30–December 3: 295–306.

Bruce, T. A. 1992. *Designing Quality Databases with IDEF1X Information Models*. New York: Dorset House.

Chen, P. P.-S. 1976. "The Entity-Relationship Model—Toward a Unified View of Data." *ACM Transactions on Database Systems* 1,1(March): 9–36.

Elmasri, R., and S. B. Navathe. 1994. *Fundamentals of Database Systems*. 2d ed. Menlo Park, CA: Benjamin/Cummings.

Gottesdiener, E. 1997. "Business Rules Show Power, Promise." *Application Development Trends* 4,3 (March): 36–54.

Gottesdiener, E. 1999. "Turning Rules into Requirements." *Application Development Trends* 6,7 (July): 37–50.

Hay, D. C. 2003. "What Exactly IS a Data Model?" Parts 1, 2, and 3. *DM Review* Vol 13, Issues 2 (February: 24–26), 3 (March: 48–50), and 4 (April: 20–22, 46).

GUIDE. 1997 (October)."GUIDE Business Rules Project." Final Report, revision 1.2.

Hoffer, J. A., J. F. George, and J. S. Valacich. 2010. *Modern Systems Analysis and Design*. 6th ed. Upper Saddle River, NJ: Prentice Hall.

ISO/IEC. 2004. "Information Technology—Metadata Registries (MDR)—Part 4: Formulation of Data Definitions." July. Switzerland. Available at **http:/metadata-standards.org/11179**.

ISO/IEC. 2005. "Information Technology—Metadata Registries (MDR)—Part 5: Naming and Identification Principles." September. Switzerland. Available at **http:/metadata-standards.org/11179**.

Johnson, T. and R. Weis. 2007. "Time and Time Again: Managing Time in Relational Databases, Part 1." May. *DM Review*. Available from Magazine Archives section in the Information Center of **www.information-management.com**. See whole series of articles called "Time and Time Again" in subsequent issues.

Moriarty, T. 2000. "The Right Tool for the Job." *Intelligent Enterprise* 3,9 (June 5): 68, 70–71.

Owen, J. 2004. "Putting Rules Engines to Work." *InfoWorld* (June 28): 35–41.

Plotkin, D. 1999. "Business Rules Everywhere." *Intelligent Enterprise* 2,4 (March 30): 37–44.

Salin, T. 1990. "What's in a Name?" *Database Programming & Design* 3,3 (March): 55–58.

Song, I.-Y., M. Evans, and E. K. Park. 1995. "A Comparative Analysis of Entity-Relationship Diagrams." *Journal of Computer & Software Engineering* 3,4: 427–59.

Storey, V. C. 1991. "Relational Database Design Based on the Entity-Relationship Model." *Data and Knowledge Engineering* 7: 47–83.

Teorey, T. J., D. Yang, and J. P. Fry. 1986. "A Logical Design Methodology for Relational Databases Using the Extended Entity- Relationship Model." *Computing Surveys* 18, 2 (June): 197–221.

von Halle, B. 1997. "Digging for Business Rules." *Database Programming & Design* 8,11: 11–13.

Further Reading

Batini, C., S. Ceri, and S. B. Navathe. 1992. *Conceptual Database Design: An Entity-Relationship Approach*. Menlo Park, CA: Benjamin/Cummings.

Bodart, F., A. Patel, M. Sim, and R. Weber. 2001. "Should Optional Properties Be Used in Conceptual Modelling? A Theory and Three Empirical Tests." *Information Systems Research* 12,4 (December): 384–405.

Carlis, J., and J. Maguire. 2001. *Mastering Data Modeling: A User-Driven Approach*. Upper Saddle River, NJ: Prentice Hall.

Keuffel, W. 1996. "Battle of the Modeling Techniques." *DBMS* 9,8 (August): 83, 84, 86, 97.

Moody, D. 1996. "The Seven Habits of Highly Effective Data Modelers." *Database Programming & Design* 9,10 (October): 57, 58, 60–62, 64.

Teorey, T. 1999. *Database Modeling & Design*. 3d ed. San Francisco, CA: Morgan Kaufman.

Tillman, G. 1994. "Should You Model Derived Data?" *DBMS* 7,11 (November): 88, 90.

Tillman, G. 1995. "Data Modeling Rules of Thumb." *DBMS* 8,8 (August): 70, 72, 74, 76, 80–82, 87.

Web Resources

http://dwr.ais.columbia.edu/info/Data%20Naming%20Standards.html Web site that provides guidelines for naming entities, attributes, and relationships similar to those suggested in this chapter.

www.adtmag.com Web site of *Application Development Trends*, a leading publication on the practice of information systems development.

www.axisboulder.com Web site for one vendor of business rules software.

www.businessrulesgroup.org Web site of the Business Rules Group, formerly part of GUIDE International, which formulates and supports standards about business rules.

http://en.wikipedia.org/wiki/Entity-relationship_model The Wikipedia entry for entity-relationship model, with an explanation of the origins of the crow's foot notation, which is used in this book.

www.intelligententerprise.com Web site of *Intelligent Enterprise*, a leading publication on database management and related areas. This magazine is the result of combining two previous publications, *Database Programming & Design* and *DBMS*.

http://ss64.com/ora/syntax-naming.html Web site that suggests naming conventions for entities, attributes, and relationships within an Oracle database environment.

www.tdan.com Web site of *The Data Administration Newsletter*, is an online journal that includes articles on a wide variety of data management topics. This Web site is considered a "must follow" Web site for data management professionals.

CASE

Mountain View Community Hospital

Case Description

After completing a course in database management, you have been hired as a summer intern by Mountain View Community Hospital. Your first assignment is to work as part of a team of three people to develop a high-level E-R diagram for the hospital. You conduct interviews with a number of hospital administrators and staff to identify the key entity types for the hospital. You have also seen the preliminary enterprise-level diagram shown in MVCH Figure 1-3 and subsequent revisions. As a result, your team has identified the following entity types:

- Care Center—a treatment center within the hospital. Examples of care centers are maternity, emergency care, or multiple sclerosis center. Each care center has a care center ID (identifier) and a care center name.
- Patient—a person who is either admitted to the hospital or is registered as an outpatient. Each patient has an identifier, the medical record number (MRN), and a name.
- Physician—a member of the hospital medical staff who may admit patients to the hospital and who may administer medical treatments. Each physician has a physician ID (identifier) and name.
- Bed—a hospital bed that may be assigned to a patient who is admitted to the hospital. Each bed has a bed number (identifier), a room number, and a care center ID.
- Item—any medical or surgical item that may be used in treating a patient. Each item has an item number (identifier), description, and unit cost.
- Employee—any person employed as part of the hospital staff. Each employee has an employee number (identifier) and name.
- Diagnosis—a patient's medical condition diagnosed by a physician. Each diagnosis has a diagnosis ID/code and diagnosis name. Mountain View Community Hospital is using the HIPAA-mandated ICD-9-CM Volume 1 diagnosis codes[1] for patient conditions (e.g., 00.50, STAPH FOOD POISONING, 173.3, BASAL CELL CARCINOMA, 200.2, MALIGNANT MELANOMA, BURKITT'S TYPE, or 776.5. CONGENITAL ANEMIA).
- Treatment—any test or procedure ordered by and/or performed by a physician for a patient. Each treatment has a treatment ID/treatment code and treatment name using standard codes. HIPAA-mandated ICD-9-CM Volume 3 Procedure Codes are used for diagnostic and therapeutic procedures (e.g., 03.31, SPINAL TAP, 14.3, REPAIR OF RETINAL TEAR, 87.44, ROUTINE CHEST X-RAY, or 90.5, MICROSCOPIC EXAMINATION OF BLOOD).
- Order—any order issued by a physician for treatment and/or services such as diagnostic tests (radiology, laboratory) and therapeutic procedures (physical therapy, diet orders), or drugs and devices (prescriptions). Each order has an order ID, order date, and order time.

The team next recorded the following information concerning relationships:

- Each hospital employee is assigned to work in one or more care centers. Each care center has at least one employee and may have any number of employees. The hospital records the number of hours per week that a given employee works in a particular care center.
- Each care center has exactly one employee who is designated nurse-in-charge for that care center.
- A given patient may or may not be assigned to a bed (since some patients are outpatients). Occupancy rates are seldom at 100 percent, so a bed may or may not be assigned to a patient.
- A patient may be referred to the hospital by exactly one physician. A physician may refer any number of patients or may not refer any patients.
- A patient must be admitted to the hospital by exactly one physician. A physician may admit any number of patients or may not admit any patients.
- Prior to a patient being seen by a physician, a nurse typically obtains and records relevant information about the patient. This includes the patient's weight, blood pressure, pulse, and temperature. The nurse who assesses the vital signs also records the date and time. Finally, the reasons for the visit and any symptoms the patient describes are recorded.
- Physicians diagnose any number of conditions affecting a patient, and a diagnosis may apply to many patients. The hospital records the following information: date and time of diagnosis, diagnosis code, and description.
- Physicians may order and perform any number of services/treatments for a patient or may not perform any treatment. A treatment or service may be performed on any number of patients, and a patient may have treatments performed or ordered by any number of physicians. For each treatment or service rendered, the hospital records the following information: physician ordering the treatment, treatment date, treatment time, and results.
- A patient may also consume any number of items. A given item may be consumed by one or more patients, or may not be consumed. For each item consumed by a patient, the hospital records the following: date, time, quantity, and total cost (which can be computed by multiplying quantity times unit cost).

Case Questions

1. Why would Mountain View Community Hospital want to use E-R modeling to understand its data requirements? What other ways might the hospital want to model its information requirements?
2. Is Mountain View Community Hospital itself an entity type in the data model? Why or why not?

[1] Note: ICD refers to the International Classification of Diseases, which, in the United States, is the HIPAA-mandated coding system used in medical billing. More information can be found at **www.cms.hhs.gov/medlearn/icd9code.asp**.

3. Do there appear to be any of the following in the description of the Mountain View Community Hospital data requirements? If so, what are they?
   a. weak entities
   b. multivalued attributes
   c. multiple relationships
4. When developing an E-R diagram for Mountain View Community Hospital, what is the significance of the business rule that states that some patients are assigned to a bed, but outpatients are not assigned to a bed?
5. Do you think that *Items* should be split into two separate entities, one for nonreusable and one for reusable items? Why or why not?
6. What quality check(s) would you perform to determine whether the E-R model you developed can easily satisfy user requests for data and/or information?

Case Exercises

1. Study the case description very closely. What other questions would you like to ask to understand the data requirements at Mountain View Community Hospital?
2. Develop an E-R diagram for Mountain View Community Hospital. State any assumptions you made in developing the diagram. If you have been assigned a particular data modeling tool, redraw your E-R diagram using this tool.
3. The case describes an entity type called *Item.* Given your answer to Case Exercise 2, will this entity type also be able to represent in-room TVs as a billable item to patients? Why or why not?
4. Suppose the attribute bed number were a composite attribute, composed of care center ID, room number, and individual bed number. Redraw any parts of your answer to Case Exercise 2 that would have to change to handle this composite attribute.
5. Consider your new E-R diagram for Case Exercise 4. Now, additionally assume that a care center contains many rooms, and each room may contain items that are billed to patients assigned to that room. Redraw your E-R diagram to accommodate this new assumption.
6. Does your answer to Case Exercise 2 allow more than one physician to perform a treatment on a patient at the same time? If not, redraw your answer to Case Exercise 2 to accommodate this situation. Make any additional assumptions you consider necessary to represent this situation.
7. Does your answer to Case Exercise 2 allow the same treatment to be performed more than once on the same patient by the same physician? If not, redraw your answer to Case Exercise 2 to accommodate this situation. Make any additional assumptions you consider necessary in order to represent this situation.

Project Assignments

P1. Develop an E-R diagram for Mountain View Community Hospital, based on the enterprise data model you developed in Chapter 1 and the case description, questions, and exercises presented previously. Using the notation described in this chapter, clearly indicate the different types of entities, attributes (identifiers, multivalued attributes, composite attributes, derived attributes) and relationships that apply in this case.

P2. Develop a list of well-stated business rules for your E-R diagram.

P3. Prepare a list of questions that have arisen as a result of your E-R modeling efforts, and that need to be answered to clarify your understanding of Mountain View Community Hospital's business rules and data requirements.