seven

Structuring System Requirements: Conceptual Data Modeling

© Corbis

Chapter Objectives

After studying this chapter, you should be able to:

- Concisely define each of the following key data-modeling terms: *conceptual data model, entity-relationship diagram, entity type, entity instance, attribute, candidate key, multivalued attribute, relationship, degree, cardinality*, and *associative entity*.
- Ask the right kinds of questions to determine data requirements for an information system.
- Draw an entity-relationship (E-R) diagram to represent common business situations.
- Explain the role of conceptual data modeling in the overall analysis and design of an information system.
- Distinguish between unary, binary, and ternary relationships, and give an example of each.
- Distinguish between a relationship and an associative entity, and use associative entities in a data model when appropriate.
- Relate data modeling to process and logic modeling as different ways of describing an information system.
- Generate at least three alternative design strategies for an information system.
- Select the best design strategy using both qualitative and quantitative methods.

Chapter Preview . . .

In Chapter 6 you learned how to model and analyze the flow of data (data in motion) between manual or automated steps and how to show data stores (data at rest) in a data-flow diagram. Data-flow diagrams show how, where, and when data are used or changed in an information system, but they do not show the definition, structure, and relationships within the data. Data modeling, the subject of this chapter, develops this missing, and crucial, piece of the description of an information system.

Systems analysts perform data modeling during the systems analysis phase, as highlighted in Figure 7-1. Data modeling is typically done at the same time as other requirements structuring steps. Many systems developers believe that a data model is the most important part of the information system requirements statement for four reasons. First, the characteristics of data captured during data modeling are crucial in the design of databases, programs, computer screens, and printed reports. For example, facts such as these—a data element is numeric, a product can be in only one product line at a time, a line item on a customer order can never be moved to another customer order—are all essential in ensuring an information system's data integrity.

Second, data rather than processes are the most complex aspects of many modern information systems. For example, transaction processing systems can have considerable complexity in validating data, reconciling errors, and coordinating the movement of data to various databases. Management information systems (such as sales tracking), decision support systems (such as short-term cash investment), and executive support systems (such as product planning)

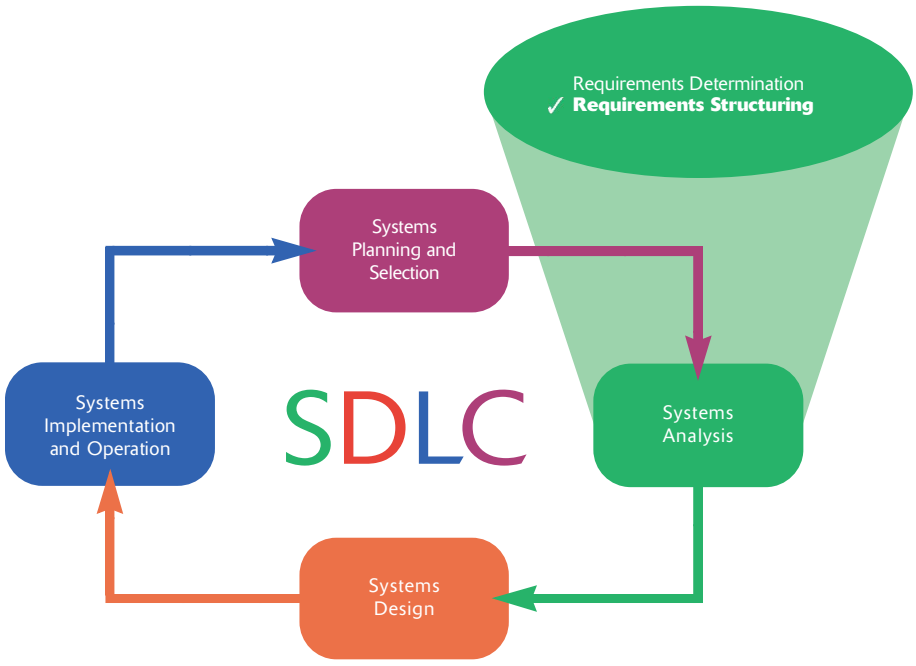


FIGURE 7-1
Systems analysts perform data modeling during the systems analysis phase. Data modeling typically occurs in parallel with other requirements structuring steps.

are data intensive and require extracting data from various data sources.

Third, the characteristics about data (such as format and relationships with other data) are rather permanent. In contrast, who receives which data, the format of reports, and what reports are used change constantly over time. A data model explains the inherent nature of the organization, not its transient form. So, an information system design based on data, rather than processes or logic, should have a longer useful life.

Finally, structural information about data is essential to generate programs automatically. For example, the fact that a customer order has many line items as opposed to just one affects the automatic design of a computer form in Microsoft Access for entry of customer orders.

In this chapter, we discuss the key concepts of data modeling, including the most common format used for data modeling—entity-relationship (E-R) diagramming. During the systems analysis phase of the SDLC, you use data-flow diagrams to show data in motion and E-R diagrams to show the relationships among data objects. We also illustrate E-R diagrams drawn using Microsoft's Visio tool, highlighting this tool's capabilities and limitations.

You have now reached the point in the analysis phase where you are ready to transform all of the information you have gathered and structured into some concrete ideas about the design for the new or replacement information system. This aspect is called the design strategy. From requirements determination, you know what the current system does. You also know what the users would like the replacement system to do. From requirements structuring, you know what forms the replacement system's process flow and data should take, at a logical level independent of any physical implementation. To bring analysis to a conclusion, your job is to take these structured requirements and transform them into several alternative design strategies. One of these strategies will be pursued in the design phase of the life cycle. In this chapter, you learn why you need to come up with alternative design strategies and about guidelines for generating alternatives. You then learn the different issues that must be addressed for each alternative. Once you have generated your alternatives, you will have to choose the best design strategy to pursue. We include a discussion of one technique that analysts and users often use to help them agree on the best approach for the new information system.

Conceptual Data Modeling

Conceptual data model
A detailed model that shows the overall structure of organizational data while being independent of any database management system or other implementation considerations.

A **conceptual data model** is a representation of organizational data. The purpose of a conceptual data model is to show as many rules about the meaning and interrelationships among data as possible, independent of any database management system or other implementation considerations.

Entity-relationship (E-R) data models are commonly used diagrams that show how data are organized in an information system. The main goal of conceptual data modeling is to create accurate E-R diagrams. As a systems analyst, you typically do conceptual data modeling at the same time as other requirements analysis and structuring steps during systems analysis. You can use methods such as interviewing, questionnaires, and JAD sessions to collect information

for conceptual data modeling. On larger systems development teams, a subset of the project team concentrates on data modeling while other team members focus attention on process or logic modeling. You develop (or use from prior systems development) a conceptual data model for the current system and build a conceptual data model that supports the scope and requirements for the proposed or enhanced system.

The work of all team members is coordinated and shared through the project dictionary or repository. As discussed in Chapter 3, this repository and associated diagrams may be maintained by a CASE tool or a specialized tool such as Microsoft's Visio. Whether automated or manual, the process flow, decision logic, and data-model descriptions of a system must be consistent and complete, because each describes different but complementary views of the same information system. For example, the names of data stores on primitive-level DFDs often correspond to the names of data entities in entity-relationship diagrams, and the data elements in data flows on DFDs must be attributes of entities and relationships in entity-relationship diagrams.

The Process of Conceptual Data Modeling

You typically begin conceptual data modeling by developing a data model for the system being replaced, if a system exists. This phase is essential for planning the conversion of the current files or database into the database of the new system. Further, it is a good, but not a perfect, starting point for your understanding of the new system's data requirements. Then, you build a new conceptual data model that includes all of the data requirements for the new system. You discovered these requirements from the fact-finding methods used during requirements determination. Today, given the popularity of prototyping and other rapid development methodologies, these requirements often evolve through various iterations of a prototype, so the data model is constantly changing.

Conceptual data modeling is only one kind of data modeling and database design activity done throughout the systems development process. Figure 7-2 shows the different kinds of data modeling and database design that occur during the systems development life cycle. The conceptual data-modeling methods we discuss in this chapter are suitable for various tasks in the planning and analysis phases. These phases of the SDLC address issues of system scope, general requirements, and content. An E-R data model evolves from project identification and selection through analysis as it becomes more specific and is validated by more detailed analysis of system needs.

In the design phase, the final E-R model developed in analysis is matched with designs for systems inputs and outputs and is translated into a format that enables physical data storage decisions. During physical design, specific data storage architectures are selected, and then, in implementation, files and databases are defined as the system is coded. Through the use of the project repository, a field in a physical data record can, for example, be traced back to the conceptual data attribute that represents it on an E-R diagram. Thus, the data modeling and design steps in each of the SDLC phases are linked through the project repository.

Deliverables and Outcomes

Most organizations today do conceptual data modeling using entity-relationship modeling, which uses a special notation of rectangles, diamonds, and lines to represent as much meaning about data as possible. Thus, the primary deliverable from the conceptual data-modeling step within the analysis phase is an

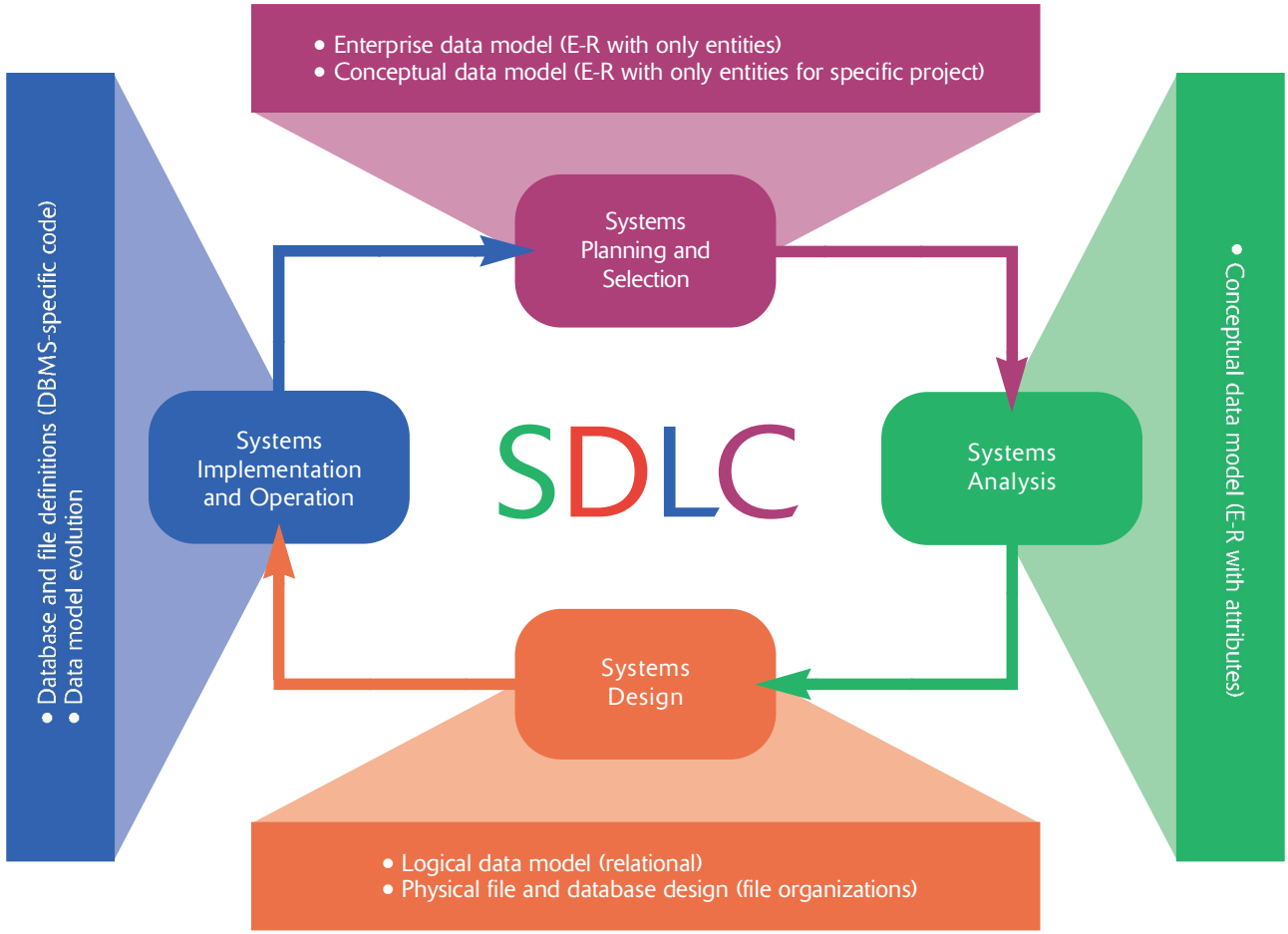


FIGURE 7-2
Relationship between data modeling and the systems development life cycle.

entity-relationship (E-R) diagram. A sample E-R diagram appears in Figure 7-3A. This figure shows the major categories of data (rectangles in the diagram) and the business relationships between them (lines connecting rectangles). For example, Figure 7-3A describes that, for the business represented, a SUPPLIER sometimes supplies ITEMs to the company, and an ITEM is always supplied by one to four SUPPLIERS. The fact that a supplier only sometimes supplies items implies that the business wants to keep track of some suppliers without designating what they can supply. This diagram includes two names on each line, giving you explicit language to read a relationship in each direction. For simplicity, we will not typically include two names on lines in E-R diagrams in this book; however, many organizations use this standard.

It is common that E-R diagrams are developed using CASE tools or other smart drawing packages. These tools provide functions to facilitate consistency of data models across different systems development phases, reverse engineering an existing database definition into an E-R diagram, and provide documentation of objects on a diagram. One popular tool is Microsoft Visio. Figure 7-3B shows the equivalent of Figure 7-3A using Visio. This diagram is developed using the Database Model Diagram tool. The Database|Options|Document settings are specified as relational symbol set, conceptual names on the diagram, optionality is shown, and relationships are shown using the crow's-foot notation with forward and inverse relationship names. These settings cause Visio to draw an E-R diagram that most closely resembles the standards used in this text.

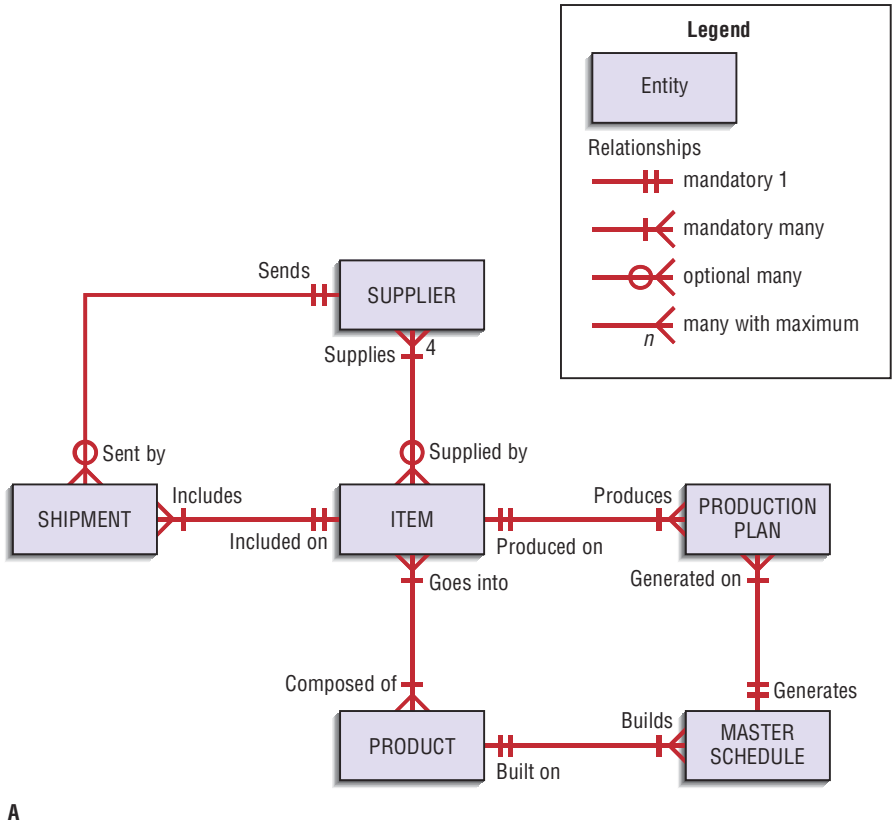


FIGURE 7-3
Sample conceptual data model diagrams: (A) Standard E-R notation.

Some key differences distinguish the standard E-R notation illustrated in Figure 7-3A from the notation used in Visio, including:

- Relationships such as Supplies/Supplied by between SUPPLIER and ITEM in Figure 7-3A require an intermediate category of data (called SUPPLIED ITEM in Figure 7-3B because Visio does not support representing these so-called many-to-many relationships.
- Relationships may be named in both directions, but these names appear near the relationship line, separated by a forward slash.
- Limitations, such as an ITEM is always supplied by at most four SUPPLIERS, are not shown on the diagram but rather are documented in the Miscellaneous set of Database Properties of the relationship, which are part of Visio's version of a CASE repository.
- The symbol for each category of data (e.g., SHIPMENT) includes space for listing other properties of each data category (such as all the attributes or columns of data we know about that data category); we will illustrate these components later in this chapter.

We concentrate on the traditional E-R diagramming notation in this chapter; however, we will include the equivalent Visio version on several occasions so you can see how to show data-modeling concepts in this popular database design tool.

As many as four E-R diagrams may be produced and analyzed during conceptual data modeling:

1. An E-R diagram that covers just the data needed in the project's application. (This first diagram allows you to concentrate on the data requirements without being constrained or confused by unnecessary details.)

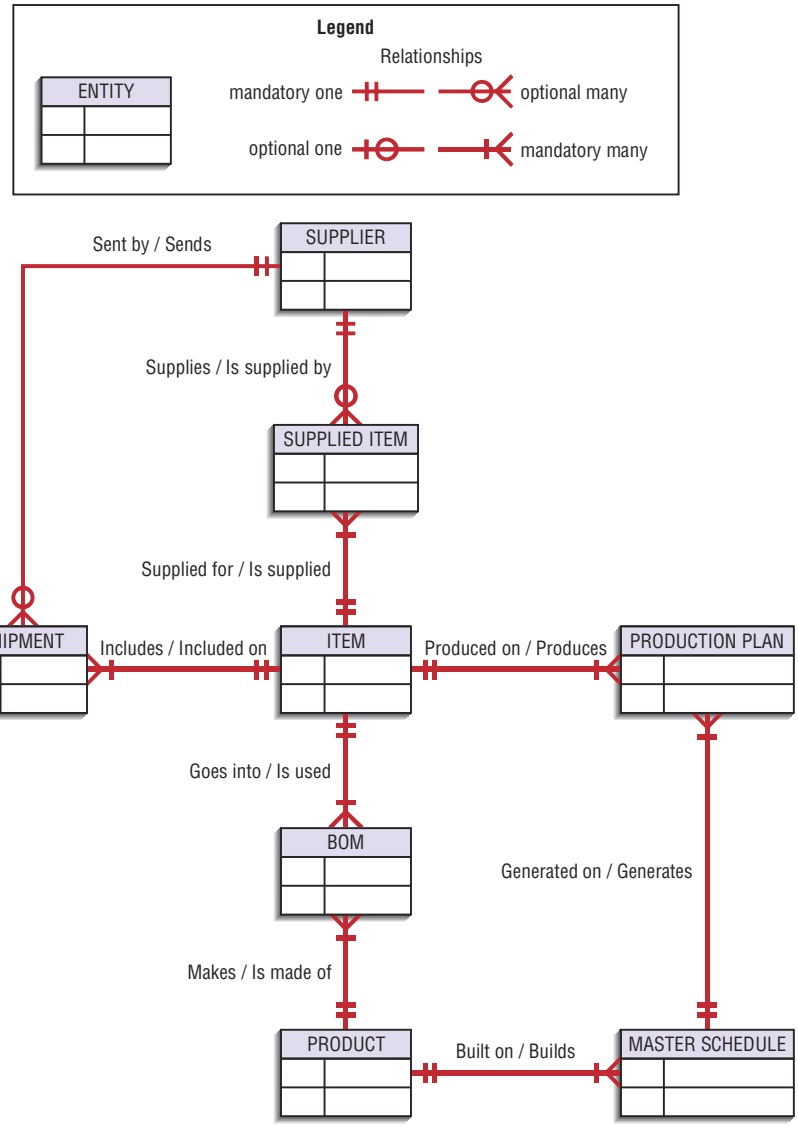


FIGURE 7-3
Sample conceptual data model diagrams: (B) Visio E-R notation.

2. An E-R diagram for the application system being replaced. (Differences between this diagram and the first show what changes you have to make to convert databases to the new application.) This version is, of course, not produced if the proposed system supports a completely new business function.
3. An E-R diagram for the whole database from which the new application's data are extracted. (Because many applications share the same database or even several databases, this and the first diagram show how the new application shares the contents of more widely-used databases.)
4. An E-R diagram for the whole database from which data for the application system being replaced is drawn. (Again, differences between this diagram and the third show what global database changes you have to make to implement the new application.) Even if no system is being replaced, an understanding of the existing data systems is necessary to see where the new data will fit in or if existing data structures must change to accommodate new data.

The other deliverable from conceptual data modeling is a set of entries about data objects to be stored in the project dictionary or repository. The repository

is the mechanism to link data, process, and logic models of an information system. For example, explicit links can be shown between a data model and a data-flow diagram. Some important links are briefly explained here.

- Data elements included in data flows also appear in the data model, and vice versa. You must include in the data model any raw data captured and retained in a data store. The data model can include only data that have been captured or are computed from captured data. Because a data model is a general business picture of data, both manual and automated data stores will be included.
- Each data store in a process model must relate to business objects (what we call *data entities*) represented in the data model. For example, in Figure 6-5, the Inventory File data store must correspond to one or several data objects in a data model.

Gathering Information for Conceptual Data Modeling

Requirements determination methods must include questions and investigations that take a data focus rather than only a process and logic focus. For example, during interviews with potential system users, you must ask specific questions to gain the perspective on data needed to develop a data model. In later sections of this chapter, we introduce some specific terminology and constructs used in data modeling. Even without this specific data-modeling language, you can begin to understand the kinds of questions that must be answered during requirements determination. These questions relate to understanding the rules and policies by which the area supported by the new information system operates. That is, a data model explains what the organization does and what rules govern how work is performed in the organization. You do not, however, need to know how or when data are processed or used to do data modeling.

You typically do data modeling from a combination of perspectives. The first perspective is called the *top-down approach*. It derives the data model from an intimate understanding of the nature of the business, rather than from any specific information requirements in computer displays, reports, or business forms. Table 7-1 summarizes key questions to ask system users and business managers so that you can develop an accurate and complete data model. The questions are purposely posed in business terms. Of course, technical terms do not mean much to a business manager, so you must learn how to frame your questions in business terms.

Alternatively, you can gather the information for data modeling by reviewing specific business documents—computer displays, reports, and business forms—handled within the system. This second perspective of gaining an understanding of data is often called a *bottom-up approach*. These business documents will appear as data flows on DFDs and will show the data processed by the system, which probably are the data that must be maintained in the system's database. Consider, for example, Figure 7-4, which shows a customer order form used at Pine Valley Furniture.

From the form in Figure 7-4, we determine that the following data must be kept in the database:

ORDER NO	CUSTOMER NO
ORDER DATE	NAME
PROMISED DATE	ADDRESS
PRODUCT NO	CITY-STATE-ZIP
DESCRIPTION	
QUANTITY ORDERED	
UNIT PRICE	

TABLE 7-1: Questions to Ask to Develop Accurate and Complete Data Models

Category of Questions	Questions to Ask System Users and Business Managers
1. Data entities and their descriptions	What are the subjects/objects of the business? What types of people, places, things, and materials are used or interact in this business about which data must be maintained? How many instances of each object might exist?
2. Candidate key	What unique characteristics distinguish each object from other objects of the same type? Could any such distinguishing feature change over time or is it permanent? Could this characteristic of an object be missing even though we know the object exists?
3. Attributes and secondary keys	What characteristic describes each object? On what basis are objects referenced, selected, qualified, sorted, and categorized? What must we know about each object in order to run the business?
4. Security controls and understanding who really knows the meaning of data	How do you use these data? That is, are you the source of the data for the organization, do you refer to the data, do you modify them, and do you destroy them? Who is not permitted to use these data? Who is responsible for establishing legitimate values for these data?
5. Cardinality and time dimensions of data	Over what period of time are you interested in these data? Do you need historical trends, current "snapshot" values, and/or estimates or projections? If a characteristic of an object changes over time, must you know the obsolete values?
6. Relationships and their cardinality and degrees	What events occur that imply associations between various objects? What natural activities or transactions of the business involve handling data about several objects of the same or different type?
7. Integrity rules, minimum and maximum cardinality, time dimensions of data	Is each activity or event always handled the same way, or are there special circumstances? Can an event occur with only some of the associated objects, or must all objects be involved? Can the associations between objects change over time (e.g., employees change departments)? Are values for data characteristics limited in any way?

FIGURE 7-4
Customer order form used at Pine Valley Furniture.

PVF CUSTOMER ORDER

ORDER NO: 61384 CUSTOMER NO: 1273

NAME: Contemporary Designs
ADDRESS: 123 Oak St.
CITY-STATE-ZIP: Austin, TX 28384

ORDER DATE: 11/04/2012 PROMISED DATE: 11/21/2012

PRODUCT NO	DESCRIPTION	QUANTITY ORDERED	UNIT PRICE
M128	Bookcase	4	200.00
B381	Cabinet	2	150.00
R210	Table	1	500.00

We also see that each order is from one customer, and an order can have multiple line items, each for one product. We use this kind of understanding of an organization's operation to develop data models.

Introduction to Entity-Relationship Modeling

The basic entity-relationship modeling notation uses three main constructs: data entities, relationships, and their associated attributes. Several different E-R notations exist, and many CASE tools support multiple notations. For simplicity, we have adopted one common notation for this book, the so-called crow's-foot notation. If you use another notation in courses or work, you should be able to easily translate between notations.

An **entity-relationship diagram** (or **E-R diagram**) is a detailed, logical, and graphical representation of the data for an organization or business area. The E-R diagram is a model 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. A rectangle is used to represent an entity, and lines are used to represent the relationship between two or more entities. The notation for E-R diagrams appears in Figure 7-5.

Entity-relationship diagram (E-R diagram)
A graphical representation of the entities, associations, and data for an organization or business area; it is a model of entities, the associations among those entities, and the attributes of both the entities and their associations.

Entities

An **entity** is a person, place, object, event, or concept in the user environment about which the organization wishes to maintain data. As noted in Table 7-1, the first requirements determination question an analyst should ask concerns data

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

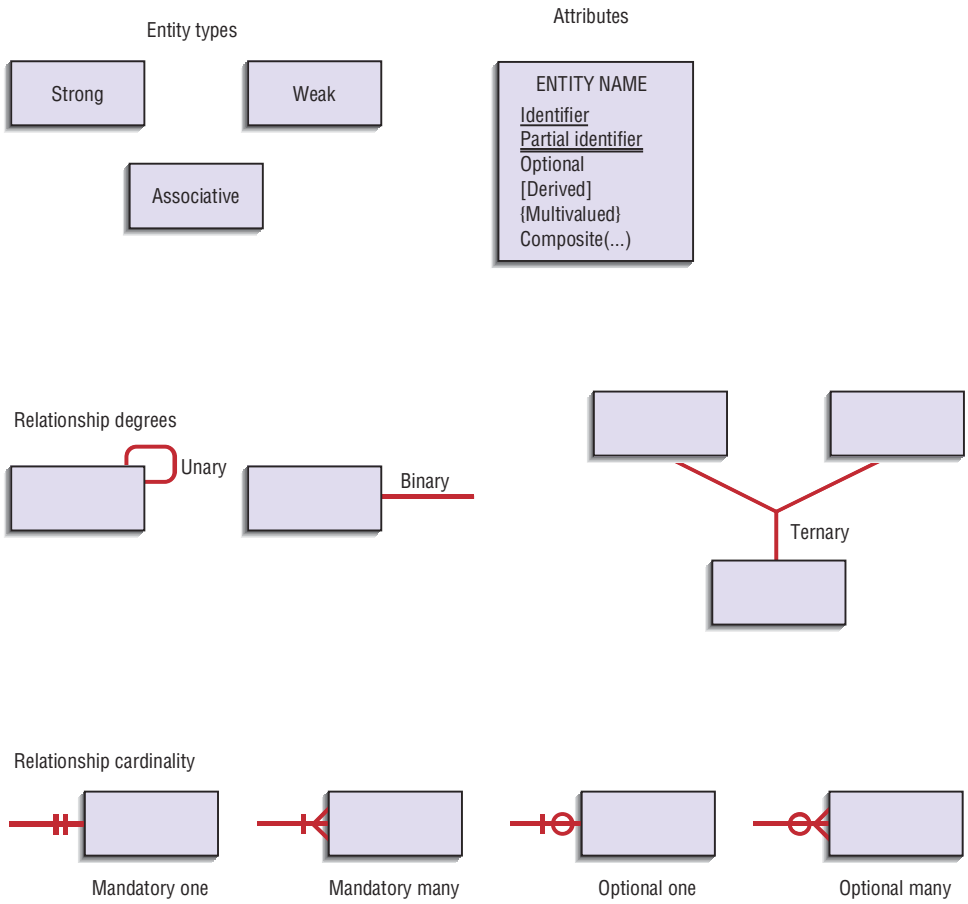


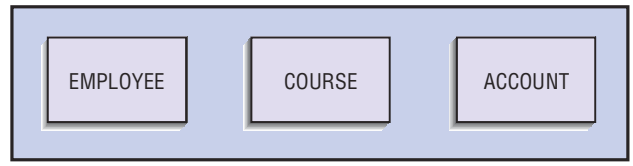
FIGURE 7-5
Entity-relationship diagram notations: basic symbols, relationship degree, and relationship cardinality.

entities. An entity has its own identity, which distinguishes it from every other entity. Some examples of entities follow:

- Person: EMPLOYEE, STUDENT, PATIENT
- Place: STATE, REGION, COUNTRY, BRANCH
- Object: MACHINE, BUILDING, AUTOMOBILE, PRODUCT
- Event: SALE, REGISTRATION, RENEWAL
- Concept: ACCOUNT, COURSE, WORK CENTER

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

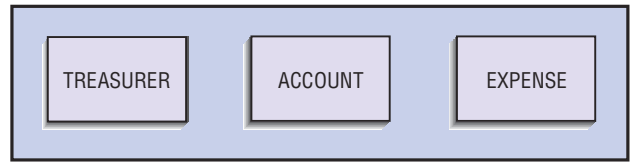
You need to recognize an important distinction between entity *types* and entity *instances*. An **entity type** is a collection of entities that share common properties or characteristics. Each entity type in an E-R model is given a name. Because the name represents a set of entities, it is singular. Also, because an entity is an object, we use a simple noun to name an entity type. We use capital letters in naming an entity type, and in an E-R diagram, the name is placed inside a rectangle representing the entity, for example:



Entity instance (instance)
A single occurrence of an entity type.

An **entity instance** (or **instance**) is a single occurrence of an entity type. An entity type is described just once in a data model, whereas many instances of that entity type may be represented by data stored in the database. For example, most organizations have one EMPLOYEE entity type, but hundreds (or even thousands) of instances of this entity type may be stored in the database.

A common mistake made in learning to draw E-R diagrams, especially if you already know how to do data-flow diagramming, is to confuse data entities with sources/sinks, system outputs, or system users, and to confuse relationships with data flows. 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 the following entity types that might be associated with a church expense system:



In this situation, the church treasurer manages accounts and records expense transactions against each account. However, do we need to keep track of data about the treasurer and her supervision of accounts as part of this accounting system? The treasurer is the person entering data about accounts and expenses and making inquiries about account balances and expense transactions by category. Because the system includes only one treasurer, TREASURER data do not need to be kept. On the other hand, if each account has an account manager (e.g., a church committee chair) who is responsible for assigned accounts, then we may wish to have an ACCOUNT MANAGER entity type, with pertinent attributes as well as relationships to other entity types.

In this same situation, is an expense report an entity type? Because an expense report is computed from expense transactions and account balances, it is a data flow, not an entity type. Even though multiple instances of expense reports will occur over time, the report contents are already represented by the ACCOUNT and EXPENSE entity types.

Often when we refer to entity types in subsequent sections, we simply say *entity*. This shorthand reference is common among data modelers. We will clarify that we mean an entity by using the term *entity instance*.

Attributes

Each entity type has a set of attributes associated with it. An **attribute** is a property or characteristic of an entity that is of interest to the organization (relationships may also have attributes, as we see in the section on relationships). Asking about attributes is the third question noted in Table 7-1 (see page 196). Following are some typical entity types and associated attributes:

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

STUDENT: Student_ID, Student_Name, Address, Phone_Number, Major
AUTOMOBILE: Vehicle_ID, Color, Weight, Horsepower
EMPLOYEE: Employee_ID, Employee_Name, Address, Skill

We use nouns with an initial capital letter followed by lowercase letters in naming an attribute. In E-R diagrams, we represent an attribute by placing its name inside the rectangle that represents the associated entity. In many E-R drawing tools, such as Microsoft Visio, attributes are listed within the entity rectangle under the entity name.

Candidate Keys and Identifiers

Every entity type must have an attribute or set of attributes that distinguishes one instance from other instances of the same type. A **candidate key** is an attribute (or combination of attributes) that uniquely identifies each instance of an entity type. A candidate key for a STUDENT entity type might be Student_ID.

Candidate key
An attribute (or combination of attributes) that uniquely identifies each instance of an entity type.

Sometimes more than one attribute is required to identify a unique entity. For example, consider the entity type GAME for a basketball league. The attribute Team_Name is clearly not a candidate key, because each team plays several games. If each team plays exactly one home game against every other team, then the combination of the attributes Home_Team and Visiting_Team is a candidate key for GAME.

Some entities may have more than one candidate key. One candidate key for EMPLOYEE is Employee_ID; a second is the combination of Employee_Name and Address (assuming that no two employees with the same name live at the same address). If more than one candidate key is involved, the designer must choose one of the candidate keys as the identifier. An **identifier** is a candidate key that has been selected to be used as the unique characteristic for an entity type.

Identifier
A candidate key that has been selected as the unique, identifying characteristic for an entity type.

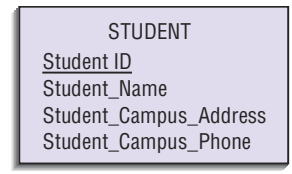
Identifiers should be selected carefully because they are critical for the integrity of data. You should apply the following identifier selection rules:

1. Choose a candidate key that will not change its value over the life of each instance of the entity type. For example, the combination of Employee_Name and Address would probably be a poor choice as a primary key for EMPLOYEE because the values of Employee_Name and Address could easily change during an employee's term of employment.
2. Choose a candidate key such that for each instance of the entity, the attribute is guaranteed to have valid values and not be null. To ensure valid values, you may have to include special controls in data entry and maintenance routines to eliminate the possibility of errors. If the

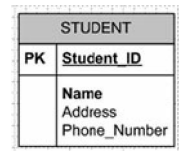
candidate key is a combination of two or more attributes, make sure that all parts of the key have valid values.

3. Avoid the use of so-called intelligent keys, whose structure indicates classifications, locations, and other entity properties. For example, the first two digits of a key for a PART entity may indicate the warehouse location. Such codes are often modified as conditions change, which renders the primary key values invalid.
4. Consider substituting single-attribute surrogate keys for large composite keys. For example, an attribute called Game_ID could be used for the entity GAME instead of the combination of Home_Team and Visiting_Team.

For each entity, the name of the identifier is underlined on an E-R diagram. The following diagram shows the representation for a STUDENT entity type using E-R notation:



The equivalent representation using Microsoft Visio is the following:

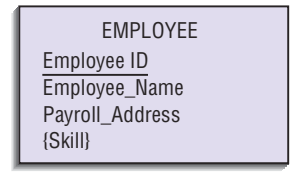


In the Visio notation, the primary key is listed immediately below the entity name with the notation PK, and the primary key is underlined. All required attributes (that is, an instance of STUDENT must have values for Student_ID and Name) are in bold.

Multivalued Attributes

Multivalued attribute
An attribute that may take on more than one value for each entity instance.

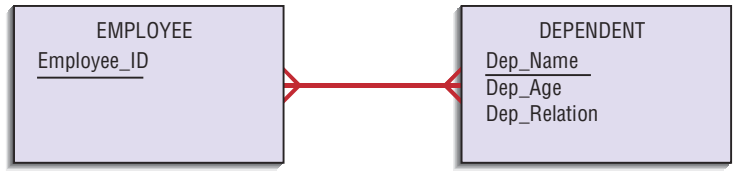
A **multivalued attribute** may take on more than one value for each entity instance. Suppose that, Skill is one of the attributes of EMPLOYEE. If each employee can have more than one Skill, then it is a multivalued attribute. During conceptual design, two common special symbols or notations are used to highlight multivalued attributes. The first is to use curly brackets around the name of the multivalued attribute, so that the EMPLOYEE entity with its attributes is diagrammed as follows:



Many E-R drawing tools, such as Microsoft Visio, do not support multivalued attributes within an entity. Thus, a second approach is to separate the repeating data into another entity, called a *weak* (or *attributive*) entity, and then using a relationship (relationships are discussed in the next section), link the weak entity to its associated regular entity. The approach also easily handles several attributes

that repeat together, called a **repeating group.** Consider an EMPLOYEE and his or her dependents. Dependent name, age, and relation to employee (spouse, child, parent, etc.) are multivalued attributes about an employee, and these attributes repeat together. We can show this repetition using an attributive entity, DEPENDENT, and a relationship, shown here simply by a line between DEPENDENT and EMPLOYEE. The crow's-foot next to DEPENDENT means that many DEPENDENTs may be associated with the same EMPLOYEE. Likewise, a DEPENDENT may be associated with more than one EMPLOYEE (i.e., two different EMPLOYEES are parents to a specific DEPENDENT).

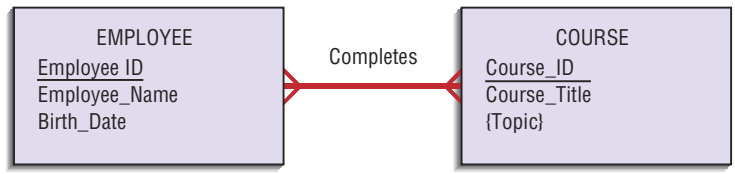
Repeating group
A set of two or more multivalued attributes that are logically related.



Relationships

Relationships are the glue that hold together the various components of an E-R model. In Table 7-1 (see page 196), questions 5, 6, and 7 deal with relationships. A **relationship** is an association between the instances of one or more entity types that are of interest to the organization. An association usually means that an event has occurred or that some natural linkage exists between entity instances. For this reason, relationships are labeled with verb phrases. For example, a training department in a company is interested in tracking which training courses each of its employees has completed. This information leads to a relationship (called Completes) between the EMPLOYEE and COURSE entity types that we diagram as follows:

Relationship
An association between the instances of one or more entity types that is of interest to the organization.



As indicated by the lines, this relationship is considered a many-to-many relationship: Each employee may complete more than one course, and each course may be completed by more than one employee. More significantly, we can use the Completes relationship to determine the specific courses that a given employee has completed. Conversely, we can determine the identity of each employee who has completed a particular course.

Conceptual Data Modeling and the E-R Model

The last section introduced the fundamentals of the E-R data modeling notation—entities, attributes, and relationships. The goal of conceptual data modeling is to capture as much of the meaning of data as possible. The more

details (or what some systems analysts call *business rules*) about data that we can model, the better the system we can design and build. Further, if we can include all these details in an automated repository, such as a CASE tool, and if a CASE tool can generate code for data definitions and programs, then the more we know about data, the more code can be generated automatically, making the system building more accurate and faster. More importantly, if we can keep a thorough repository of data descriptions, we can regenerate the system as needed as the business rules change. Because maintenance is the largest expense with any information system, the efficiencies gained by maintaining systems at the rule, rather than code, level drastically reduce the cost.

In this section, we explore more advanced concepts needed to more thoroughly model data and learn how the E-R notation represents these concepts.

Degree of a Relationship

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

The **degree** of a relationship, question 6 in Table 7-1, is the number of entity types that participate in that relationship. Thus, the relationship Completes, illustrated previously, is of degree two because it involves two entity types: EMPLOYEE and COURSE. The three most common relationships in E-R diagrams are unary (degree one), binary (degree two), and ternary (degree three). 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 7-6.

Unary relationship (recursive relationship)
A relationship between the instances of one entity type.

Unary Relationship Also called a **recursive relationship,** a **unary relationship** is a relationship between the instances of one entity type. Two examples are shown in Figure 7-6. In the first example, Is_married_to is shown as a one-to-one relationship between instances of the PERSON entity type. That is, each person may be currently married to one other person. 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 or, reading the Manages relationship in the opposite direction, who the manager is for a given employee.

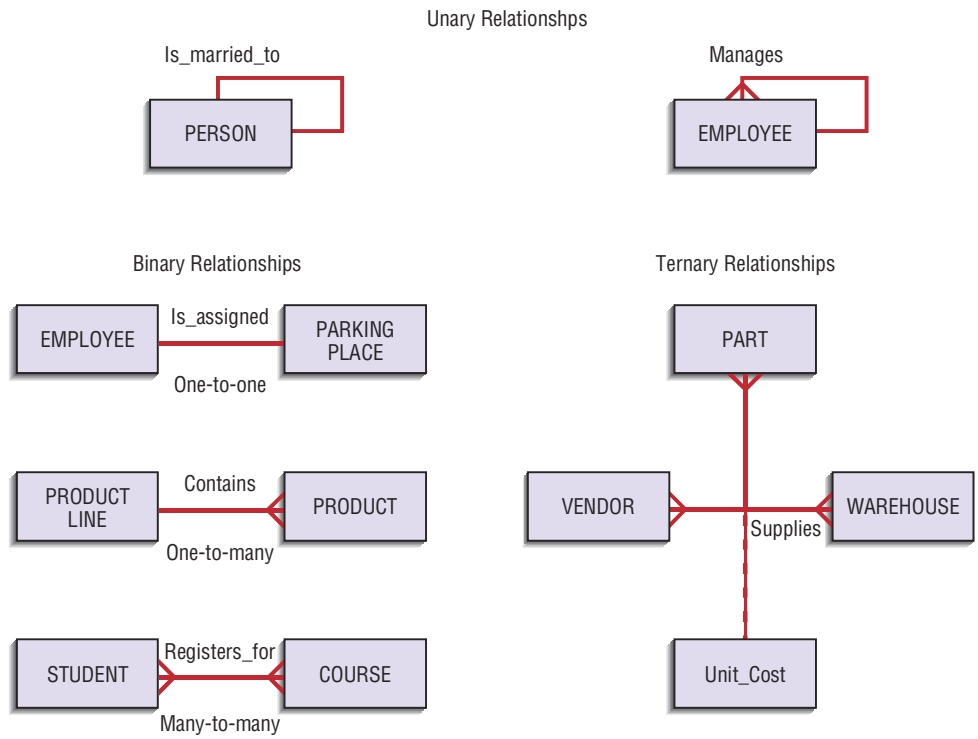


FIGURE 7-6
Examples of the three most common relationships in E-R diagrams: unary, binary, and ternary.

Binary Relationship A **binary relationship** is a relationship between instances of two entity types and is the most common type of relationship encountered in data modeling. Figure 7-6 shows three examples. The first (one-to-one) indicates that an employee is assigned one parking place, and each parking place is assigned to one employee. The second (one-to-many) indicates that a product line may contain several products, and 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.

Binary relationship
A relationship between instances of two entity types.

Ternary Relationship A **ternary relationship** is a simultaneous relationship among instances of three entity types. In the example shown in Figure 7-6, the relationship Supplies tracks the quantity of a given part that is shipped by a particular vendor to a selected warehouse. Each entity may be a one or a many participant in a ternary relationship (in Figure 7-6, all three entities are many participants).

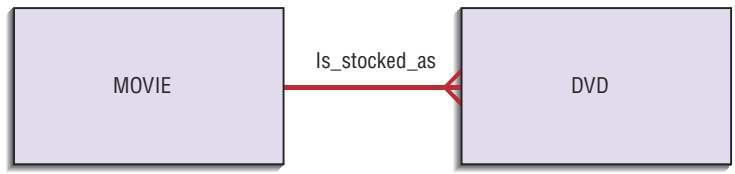
Note that a ternary relationship is not the same as three binary relationships. For example, Unit_Cost is an attribute of the Supplies relationship in Figure 7-6. Unit_Cost cannot be properly associated with any of the three possible binary relationships among the three entity types (such as that between PART and VENDOR) because Unit_Cost is the cost of a particular PART shipped from a particular VENDOR to a particular WAREHOUSE.

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

Cardinalities in Relationships

Suppose that two entity types, A and B, are connected by a relationship. The **cardinality** of a relationship (see the fifth, sixth, and seventh questions in Table 7-1) is the number of instances of entity B that can (or must) be associated with each instance of entity A. For example, consider the following relationship for DVDs and movies:

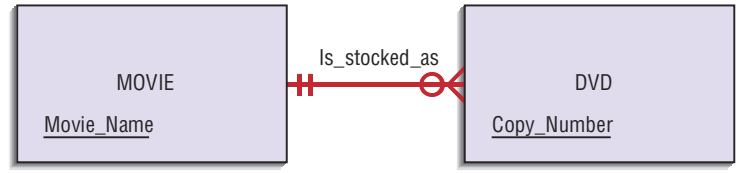
Cardinality
The number of instances of entity B that can (or must) be associated with each instance of entity A.



Clearly, a video store may stock more than one DVD of a given movie. In the terminology we have used so far, this example is intuitively a "many" relationship. Yet, it is also true that the store may not have a single DVD of a particular movie in stock. We need a more precise notation to indicate the range of cardinalities for a relationship. This notation of relationship cardinality was introduced in Figure 7-5, which you may want to review at this point.

Minimum and Maximum Cardinalities 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 the preceding example, the minimum number of DVDs available for a movie is zero, in which case we say that DVD is an optional participant in the Is_stocked_as relationship. When the minimum cardinality of a relationship is one, then we say entity B is a mandatory participant in the relationship. The maximum cardinality is the maximum number of instances. For our example, this maximum is "many" (an unspecified

number greater than one). Using the notation from Figure 7-5, we diagram this relationship as follows:

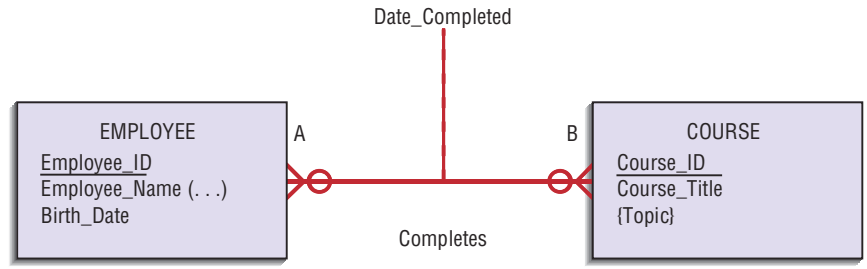


The zero through the line near the DVD entity means a minimum cardinality of zero, whereas the crow's-foot notation means a "many" maximum cardinality. It is possible for the maximum cardinality to be a fixed number, not an arbitrary "many" value. For example, see the Supplies relationship in Figure 7-3A, which indicates that each item involves at most four suppliers.

Associative Entities As seen in the examples of the Supplies ternary relationship in Figure 7-6, attributes may be associated with a many-to-many relationship as well as with an entity. For example, suppose that the organization wishes to record the date (month and year) when an employee completes each course. Some sample data follow:

Employee_ID	Course_Name	Date_Completed
549–23–1948	Basic Algebra	March 2012
629–16–8407	Software Quality	June 2012
816–30–0458	Software Quality	Feb 2012
549–23–1948	C Programming	May 2012

From this limited data, you can conclude that the attribute Date_Completed is not a property of the entity EMPLOYEE (because a given employee, such as 549–23–1948, has completed courses on different dates). Nor is Date_Completed a property of COURSE, because a particular course (such as Software Quality) may be completed on different dates. Instead, Date_Completed is a property of the relationship between EMPLOYEE and COURSE. The attribute is associated with the relationship and diagrammed as follows:



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.

Because many-to-many and one-to-one relationships may have associated attributes, the E-R diagram poses an interesting dilemma: Is a many-to-many relationship actually an entity in disguise? Often the distinction between entity and relationship is simply a matter of how you view the data. An **associative entity** is a relationship that the data modeler chooses to model as an entity type. Figure 7-7 shows the E-R notation for representing the Completes relationship as an associative entity. The lines from CERTIFICATE to the two entities are not two separate binary relationships, so they do not have labels. Note that EMPLOYEE and COURSE have mandatory-one cardinality, because an instance

FIGURE 7-7
Example of an associative entity.

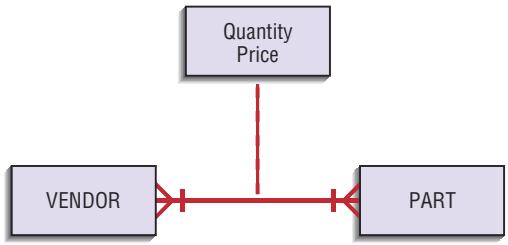
of Completes must have an associated EMPLOYEE and COURSE. The implicit identifier of Completes is the combination of the identifiers of EMPLOYEE and COURSE, Employee_ID, and Course_ID, respectively. The explicit identifier is Certificate_Number, as shown in Figure 7-7.

E-R drawing tools that do not support many-to-many relationships require that any such relationship be converted into an associative entity, whether it has attributes or not. You have already seen an example of this in Figure 7-3 for Microsoft Visio, in which the Supplies/Supplied by relationship from Figure 7-3A was converted in Figure 7-3B into the SUPPLIED ITEM entity (actually, associative entity) and two mandatory one-to-many relationships.

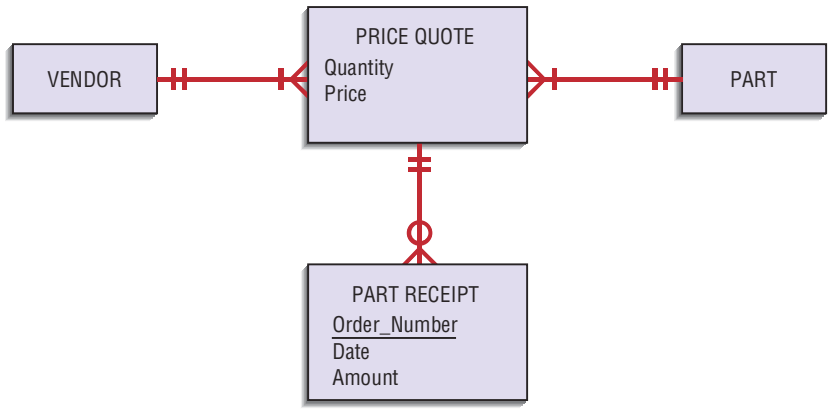
One situation in which a relationship must be turned into an associative entity is when the associative entity has other relationships with entities besides the relationship that caused its creation. For example, consider the E-R model, which represents price quotes from different vendors for purchased parts stocked by Pine Valley Furniture, shown in Figure 7-8A.

PINE VALLEY FURNITURE

Now, suppose that we also need to know which price quote is in effect for each part shipment received. This additional data requirement necessitates that the relationship between VENDOR and PART be transformed into an associative entity. This new relationship is represented in Figure 7-8B.



A Many-to-many relationship with attributes



B Associative entity with separate relationship

FIGURE 7-8
An E-R model that represents each price quote for each part shipment received by Pine Valley Furniture.

In this case, PRICE QUOTE is not a ternary relationship. Rather, PRICE QUOTE is a binary many-to-many relationship (associative entity) between VENDOR and PART. In addition, each PART RECEIPT, based on Amount, has an applicable, negotiated Price. Each PART RECEIPT is for a given PART from a specific VENDOR, and the Amount of the receipt dictates the purchase price in effect by matching with the Quantity attribute. Because the PRICE QUOTE pertains to a given PART and given VENDOR, PART RECEIPT does not need direct relationships with these entities.

An Example of Conceptual Data Modeling at Hoosier Burger

Chapter 6 structured the process and data-flow requirements for a food-ordering system for Hoosier Burger. Figure 7-9 describes requirements for a new system using Microsoft Visio. The purpose of this system is to monitor and report changes in raw material inventory levels and to issue material orders and payments to suppliers. Thus, the central data entity for this system will be an INVENTORY ITEM, shown in Figure 7-10, corresponding to data store D1 in Figure 7-9.

Changes in inventory levels are due to two types of transactions: receipt of new items from suppliers and consumption of items from sales of products. Inventory is added upon receipt of new raw materials, for which Hoosier Burger receives a supplier INVOICE (see Process 1.0 in Figure 7-9). Figure 7-10 shows that each INVOICE indicates that the supplier has sent a specific quantity of one or more INVOICE ITEMs, which correspond to Hoosier's INVENTORY ITEMs. Inventory is used when customers order and pay for PRODUCTs. That is, Hoosier makes a SALE for one or more ITEM SALEs,

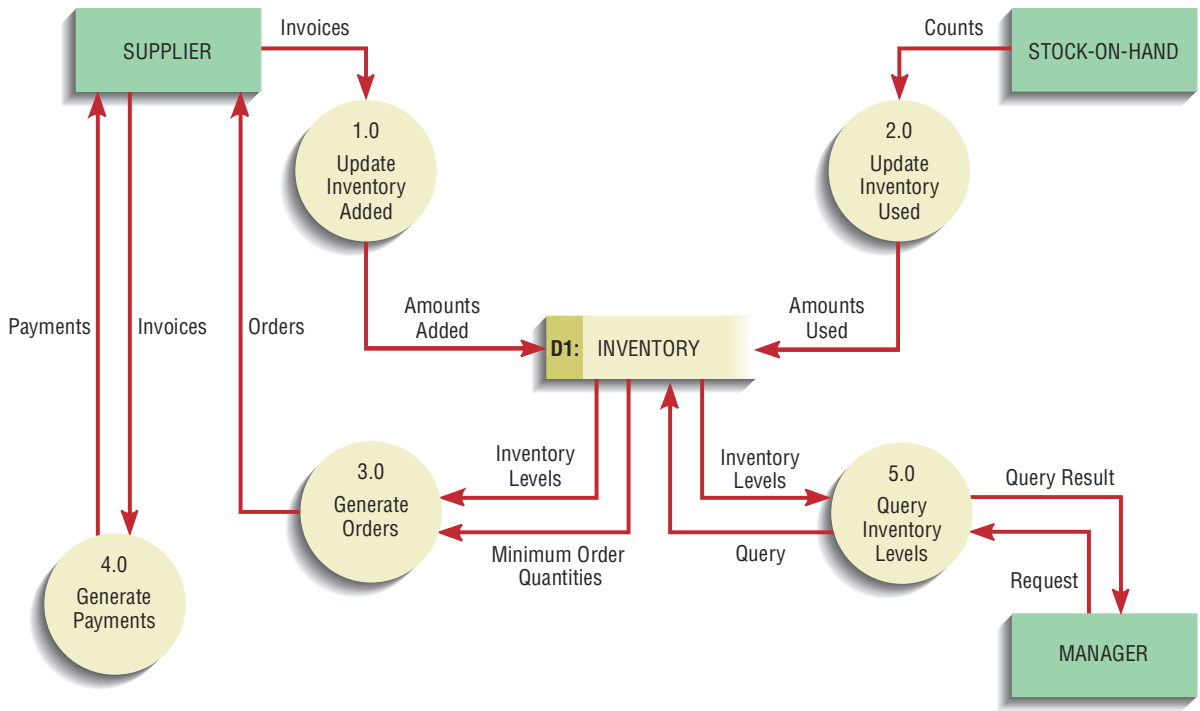


FIGURE 7-9
Level-0 data-flow diagram for Hoosier Burger's new logical inventory control system.

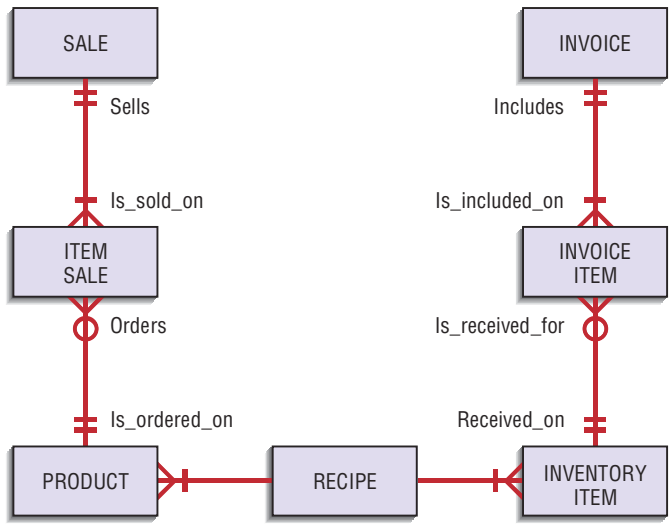


FIGURE 7-10
Preliminary E-R diagram for Hoosier Burger's inventory control system.

each of which corresponds to a food PRODUCT. Because the real-time customer-order processing system is separate from the inventory control system, a source, STOCK-ON-HAND in Figure 7-9, represents how data flow from the order processing to the inventory control system. Finally, because food PRODUCTs are made up of various INVENTORY ITEMs (and vice versa), Hoosier Burger maintains a RECIPE to indicate how much of each INVENTORY ITEM goes into making one PRODUCT. From this discussion, we have identified the data entities required in a data model for the new Hoosier Burger inventory control system: INVENTORY ITEM, INVOICE, INVOICE ITEM, PRODUCT, SALE, ITEM SALE, and RECIPE. To complete the E-R diagram, we must determine necessary relationships among these entities as well as attributes for each entity.

The wording in the previous description tells us much of what we need to know to determine relationships:

- An INVOICE includes one or more INVOICE ITEMs, each of which corresponds to an INVENTORY ITEM. Obviously, an INVOICE ITEM cannot exist without an associated INVOICE, and over time the result will be zero-to-many receipts, or INVOICE ITEMs, for an INVENTORY ITEM.
- Each PRODUCT is associated with INVENTORY ITEMs.
- A SALE indicates that Hoosier Burger sells one or more ITEM SALEs, each of which corresponds to a PRODUCT. An ITEM SALE cannot exist without an associated SALE, and over time the result will be zero-to-many ITEM SALEs for a PRODUCT.

Figure 7-10 shows an E-R diagram with the entities and relationships previously described. We include on this diagram two labels for each relationship, one to be read in either relationship direction (e.g., an INVOICE Includes one-to-many INVOICE ITEMs, and an INVOICE ITEM Is_included_on exactly one INVOICE). Now that we understand the entities and relationships, we must decide which data elements are associated with the entities and associative entities in this diagram.

You may wonder at this point why only the INVENTORY data store is shown in Figure 7-9 when seven entities and associative entities are on the E-R diagram. The INVENTORY data store corresponds to the INVENTORY ITEM

entity in Figure 7-10. The other entities are hidden inside other processes for which we have not shown lower-level diagrams. In actual requirements structuring steps, you would have to match all entities with data stores: Each data store represents some subset of an E-R diagram, and each entity is included in one or more data stores. Ideally, each data store on a primitive DFD will be an individual entity.

To determine data elements for an entity, we investigate data flows in and out of data stores that correspond to the data entity and supplement this information with a study of decision logic that uses or changes data about the entity. Six data flows are associated with the INVENTORY data store in Figure 7-9. The description of each data flow in the project dictionary or repository would include the data flow's composition, which then tells us what data are flowing in or out of the data store. For example, the Amounts Used data flow coming from Process 2.0 indicates how much to decrease an attribute STOCK_ON_HAND due to use of the INVENTORY ITEM to fulfill a customer sale. Thus, the Amounts Used data flow implies that Process 2.0 will first read the relevant INVENTORY ITEM record, then update its STOCK_ON_HAND attribute, and finally store the updated value in the record. Each data flow would be analyzed similarly (space does not permit us to show the analysis for each data flow).

After having considered all data flows in and out of data stores related to data entities, plus all decision logic related to inventory control, we derive the full E-R diagram, with attributes, shown in Figure 7-11. In Visio, the ITEM SALE, RECIPE, and INVOICE ITEM entities participate in what are called *identifying relationships*. Thus, Visio treats them as associative entities, not just the RECIPE entity. Visio automatically includes the primary keys of the identifying entities as primary keys in the identified (associative) entities. Also note that in Visio, because it cannot represent many-to-many relationships, there are two mandatory relationships on either side of RECIPE.

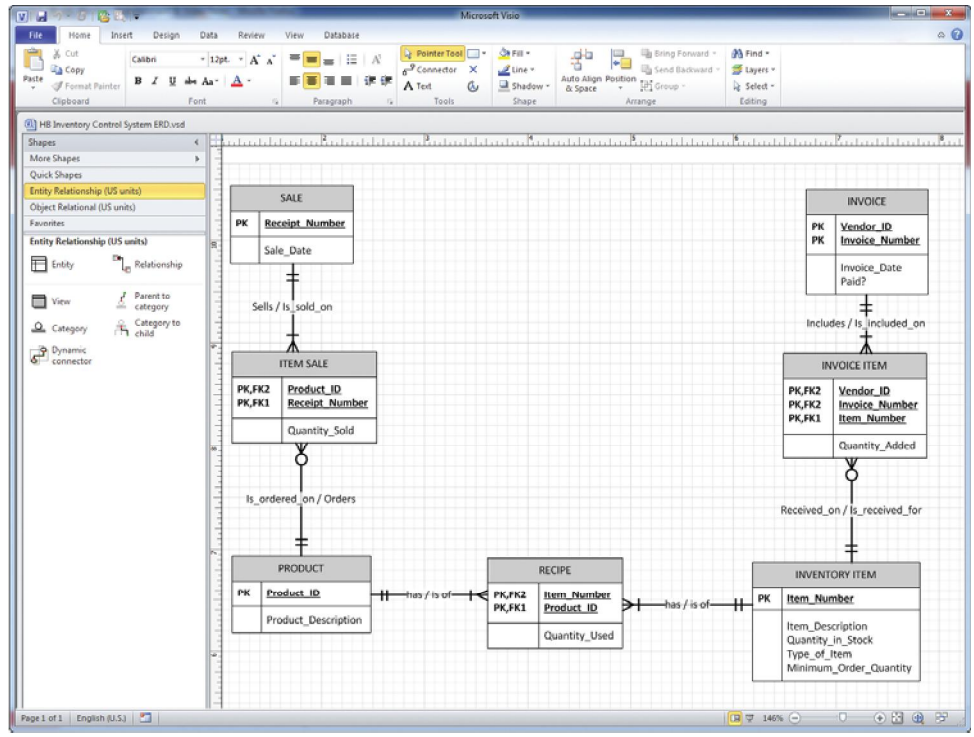


FIGURE 7-11
Final E-R diagram for Hoosier Burger's inventory control system.

PVF WebStore: Conceptual Data Modeling

Conceptual data modeling for an Internet-based electronic commerce application is no different from the process followed when analyzing the data needs for other types of applications. In the last chapter, you read how Jim Woo analyzed the flow of information within the WebStore and developed a data-flow diagram. In this section, we examine the process he followed when developing the WebStore's conceptual data model.

Conceptual Data Modeling for Pine Valley Furniture's WebStore

To better understand what data would be needed within the WebStore, Jim Woo carefully reviewed the information from the JAD session and his previously developed data-flow diagram. Table 7-2 summarizes the customer and inventory information identified during the JAD session. Jim wasn't sure whether this information was complete but knew that it was a good starting place for identifying what information the WebStore needed to capture, store, and process. To identify additional information, he carefully studied the level-0 DFD shown in Figure 7-12. In this diagram, two data stores—Inventory and Shopping Cart—are clearly identified; both were strong candidates to become entities within the conceptual data model. Finally, Jim examined the data flows from the DFD as additional possible sources for entities. Hence, he identified five general categories of information to consider:

- Customer
- Inventory
- Order
- Shopping Cart
- Temporary User/System Messages

After identifying these multiple categories of data, his next step was to define each item carefully. He again examined all data flows within the DFD and recorded each one's source and destination. By carefully listing these flows, he could move more easily through the DFD and understand more thoroughly what information was needed to move from point to point. This activity resulted in the creation of two tables that documented Jim's growing understanding of the WebStore's requirements. The first, Table 7-3, lists each of the data flows within

TABLE 7-2: Customer and Inventory Information for WebStore

Corporate Customer	Home-Office Customer	Student Customer	Inventory Information
Company name	Name	Name	SKU
Company address	Doing business as (company's name)	School	Name
Company phone	Address	Address	Description
Company fax	Phone	Phone	Finished product size
Company preferred shipping method	Fax	E-mail	Finished product weight
Buyer name	E-mail		Available materials
Buyer phone			Available colors
Buyer e-mail			Price
			Lead time

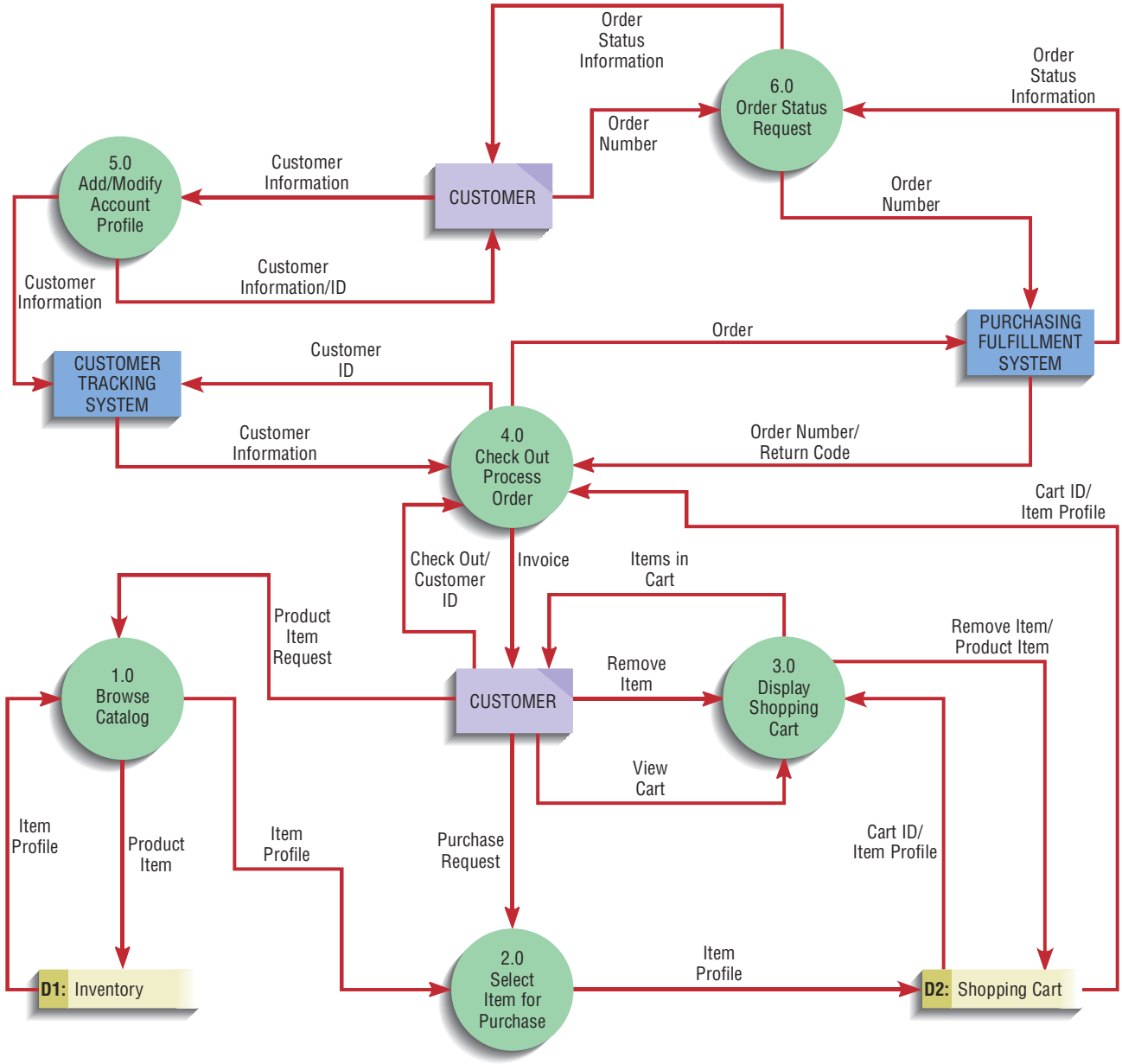


FIGURE 7-12
Level-0 data-flow diagram for the WebStore.

each data category and its corresponding description. The second, Table 7-4, lists each of the unique data flows within each data category. Jim then felt ready to construct an entity-relationship diagram for the WebStore.

He concluded that Customer, Inventory, and Order were all unique entities and would be part of his E-R diagram. Recall that an entity is a person, place, or object; all three of these items meet this criteria. Because the Temporary User/System Messages data were not permanently stored items—nor were they a person, place, or object—he concluded that this should not be an entity in the conceptual data model. Alternatively, although the shopping cart was also a temporarily stored item, its contents needed to be stored for at least the duration of a customer's visit to the WebStore and should be considered an object. As shown in Figure 7-12, Process 4, Check Out/Process Order, moves the Shopping Cart contents to the Purchasing Fulfillment System, where the order details are stored. Thus, he concluded that Shopping Cart—along with Customer, Inventory, and Order—would be entities in his E-R diagram.

TABLE 7-3: Data Category, Data Flow, and Data-Flow Descriptions for the WebStore DFD

Data Category / Data Flow	Description
Customer Related	
Customer ID	Unique identifier for each customer (generated by Customer Tracking System)
Customer Information	Detailed customer information (stored in Customer Tracking System)
Inventory Related	
Product Item	Unique identifier for each product item (stored in Inventory Database)
Item Profile	Detailed product information (stored in Inventory Database)
Order Related	
Order Number	Unique identifier for an order (generated by Purchasing Fulfillment System)
Order	Detailed order information (stored in Purchasing Fulfillment System)
Return Code	Unique code for processing customer returns (generated by/stored in Purchasing Fulfillment System)
Invoice	Detailed order summary statement (generated from order information stored in Purchasing Fulfillment System)
Order Status Information	Detailed summary information on order status (stored/generated by Purchasing Fulfillment System)
Shopping Cart	
Cart ID	Unique identifier for shopping cart
Temporary User/System Messages	
Product Item Request	Request to view information on a catalog item
Purchase Request	Request to move an item into the shopping cart
View Cart	Request to view the contents of the shopping cart
Items in Cart	Summary report of all shopping cart items
Remove Item	Request to remove item from shopping cart
Check Out	Request to check out and process order

The final step was to identify the interrelationships between these four entities. After carefully studying all the related information, he concluded the following:

1. Each Customer owns *zero-to-many* Shopping Cart Instances; each Shopping Cart Instance is-owned-by *one-and-only-one* Customer.
2. Each Shopping Cart Instance contains *one-and-only-one* Inventory item; each Inventory item is-contained-in *zero-to-many* Shopping Cart Instances.
3. Each Customer places *zero-to-many* Orders; each Order is-placed-by *one-and-only-one* Customer.
4. Each Order contains *one-to-many* Shopping Cart Instances; each Shopping Cart Instance is-contained-in *one-and-only-one* Order.

With these relationships defined, Jim drew the E-R diagram shown in Figure 7-13. Through it, he demonstrated his understanding of the requirements, the flow of information within the WebStore, the flow of information between the WebStore and existing PVF systems, and now the conceptual data model.

TABLE 7-4: Data Category, Data Flow, and the Source/Destination of Data Flows within the WebStore DFD

Data Category / Data Flow	From/To
Customer Reated	
Customer ID	From Customer to Process 4.0 From Process 4.0 to Customer Tracking System From Process 5.0 to Customer
Customer Information	From Customer to Process 5.0 From Process 5.0 to Customer From Process 5.0 to Customer Tracking System From Customer Tracking System to Process 4.0
Inventory Related	
Product Item	From Process 1.0 to Data Store D1 From Process 3.0 to Data Store D2
Item Profile	From Data Store D1 to Process 1.0 From Process 1.0 to Process 2.0 From Process 2.0 to Data Store D2 From Data Store D2 to Process 3.0 From Data Store D2 to Process 4.0
Order Related	
Order Number	From Purchasing Fulfillment System to Process 4.0 From Customer to Process 6.0 From Process 6.0 to Purchasing Fulfillment System
Order	From Process 4.0 to Purchasing Fulfillment System
Return Code	From Purchasing Fulfillment System to Process 4.0
Invoice	From Process 4.0 to Customer
Order Status Information	From Process 6.0 to Customer From Purchasing Fulfillment System to Process 6.0
Shopping Cart	
Cart ID	From Data Store D2 to Process 3.0 From Data Store D2 to Process 4.0
Temporary User/System Messages	
Product Item Request	From Customer to Process 1.0
Purchase Request	From Customer to Process 2.0
View Cart	From Customer to Process 3.0
Items in Cart	From Process 3.0 to Customer
Remove Item	From Customer to Process 3.0 From Process 3.0 to Data Store D2
Check Out	From Customer to Process 4.0

Over the next few hours, Jim planned to refine his understanding further by listing the specific attributes for each entity and then comparing these lists with the existing inventory, customer, and order database tables. He had to make sure that all attributes were accounted for before determining a final design strategy.

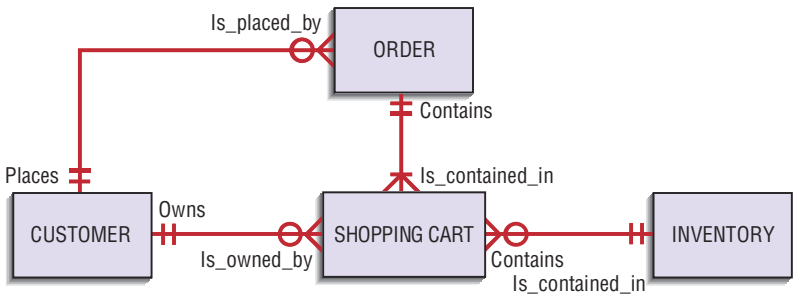


FIGURE 7-13
Entity-relationship diagram for the WebStore system.

Selecting the Best Alternative Design Strategy

Selecting the best alternative system involves at least two basic steps: (1) generating a comprehensive set of alternative design strategies, and (2) selecting the one that is most likely to result in the desired information system, given all of the organizational, economic, and technical constraints that limit what can be done. A system **design strategy** represents a particular approach to developing the system. Selecting a strategy requires you to answer questions about the system's functionality, hardware and system software platform, and method for acquisition. We use the term *design strategy* in this chapter rather than *alternative system* because, at the end of analysis, we are still quite a long way from specifying an actual system. This delay is purposeful because we do not want to invest in design efforts until some agreement is reached on which direction to take the project and the new system. The best we can do at this point is to outline, rather broadly, the approach we can take in moving from logical system specifications to a working physical system. The overall process of selecting the best system strategy and the deliverables from this step in the analysis process are discussed next.

Design strategy
A particular approach to developing an information system. It includes statements on the system's functionality, hardware and system software platform, and method for acquisition.

The Process of Selecting the Best Alternative Design Strategy

Systems analysis involves determining requirements and structuring requirements. After the system requirements have been structured in terms of process flow and data, analysts again work with users to package the requirements into different system configurations. Shaping alternative system design strategies involves the following processes:

- Dividing requirements into different sets of capabilities, ranging from the bare minimum that users would accept (the required features) to the most elaborate and advanced system the company could afford to develop (which includes all the features desired by all users). Alternatively, different sets of capabilities may represent the position of different organizational units with conflicting notions about what the system should do.
- Enumerating different potential implementation environments (hardware, system software, and network platforms) that could be used to deliver the different sets of capabilities. (Choices on the implementation environment may place technical limitations on the subsequent design phase activities.)
- Proposing different ways to source or acquire the various sets of capabilities for the different implementation environments.

In theory, if the system includes three sets of requirements, two implementation environments, and four sources of application software, twenty-four design strategies would be possible. In practice, some combinations are usually infeasible, and only a small number—typically three—can be easily considered.

Selecting the best alternative is usually done with the help of a quantitative procedure, an example of which comes later in the chapter. Analysts will recommend what they believe to be the best alternative, but management (a combination of the steering committee and those who will fund the rest of the project) will make the ultimate decision about which system design strategy to follow. At this point in the life cycle, it is also certainly possible for management to end a project before the more expensive phases of system design or system implementation and operation are begun. Reasons for ending a project might include the costs or risks outweighing the benefits, the needs of the organization having changed since the project began, or other competing projects having become more important while development resources remain limited.

Generating Alternative Design Strategies

The solution to an organizational problem may seem obvious to an analyst. Typically, the analyst is familiar with the problem, having conducted an extensive analysis of it and how it has been solved in the past. On the other hand, the analyst may be more familiar with a particular solution that he or she attempts to apply to all organizational problems encountered. For example, if an analyst is an expert at using advanced database technology to solve problems, then he or she tends to recommend advanced database technology as a solution to every possible problem. Or if the analyst designed a similar system for another customer or business unit, the "natural" design strategy would be the one used before. Given the role of the analysts' experience in the solutions they suggest, analysis teams typically generate at least two alternative solutions for every problem they work on.

A good number of alternatives for analysts to generate is three. Why three? Because three alternatives can neatly represent low, middle, and high ranges of potential solutions. One alternative represents the low end of the range. Low-end alternatives are the most conservative in terms of the effort, cost, and technology involved in developing a new system. Some low-end solutions may not involve computer technology at all, focusing instead on making paper flows more efficient or reducing redundancies in current processes. A low-end strategy provides all the required functionality users demand with a system that is minimally different from the current system.

Another alternative represents the high end of the range. High-end alternatives go beyond simply solving the problem in question and focus instead on systems that contain many extra features users may desire. Functionality, not cost, is the primary focus of high-end alternatives. A high-end alternative will provide all desired features using advanced technologies that often allow the system to expand to meet future requirements. Finally, the third alternative lies between the extremes of the low-end and high-end systems. Such alternatives combine the frugality of low-end alternatives with the focus on functionality of high-end alternatives. Midrange alternatives represent compromise solutions. Other possible solutions exist outside of these three alternatives, of course. Defining the low, middle, and high possibilities allows the analyst to draw bounds around what can be reasonably done.

How do you know where to draw bounds around the potential solution space? The analysis team has already gathered the information it needs to identify the solution, but first that information must be systematically organized. The first of two major considerations in this process is to determine the minimum requirements for the new system. These features are mandatory, and if any of them are missing, the design strategy is useless. Mandatory features are the ones that everyone agrees are necessary to solve the problem or meet the opportunity. Which features are mandatory can be determined from a survey

of users and others who have been involved in requirements determination. You would conduct this survey near the end of the analysis phase after all requirements have been structured and analyzed. In this survey, users rate features discovered during requirements determination or categorize features on some scale, and an arbitrary breakpoint is used to divide mandatory from desired features. Some organizations will break the features into three categories: mandatory, essential, and desired. Whereas mandatory features screen out possible solutions, essential features are the important capabilities of a system that serve as the primary basis for comparison of different design strategies. Desired features are those that users could live without but that are used to select between design strategies that are of almost equal value in terms of essential features. Features can take many different forms, as illustrated in Figure 7-14, and might include:

- *Data kept in system files:* For example, multiple customer addresses so that bills can be sent to addresses different from where we ship goods.
- *System outputs:* Printed reports, online displays, transaction documents (for example, a paycheck or sales summary graph).
- *Analyses to generate the information in system outputs:* For example, a sales forecasting module or an installment billing routine.
- *Expectations on accessibility, response time, or turnaround time for system functions:* For example, online, real-time updating of inventory files.

The second consideration in drawing bounds around alternative design strategies is determining the constraints on system development. Constraints, some of which also appear in Figure 7-14, may include:

- *A date when the replacement system is needed.*
- *Available financial and human resources.*
- *Elements of the current system that cannot change.*
- *Legal and contractual restrictions:* For example, a software package bought off the shelf cannot be legally modified, or a license to use a particular software package may limit the number of concurrent users to twenty-five.
- *The importance or dynamics of the problem that may limit how the system can be acquired:* For example, a strategically important system that uses highly proprietary data probably cannot be outsourced or purchased.

FIGURE 7-14
Essential features to consider during systems development include data (such as customer addresses), output (such as a printed report like a sales summary graph), and analyses (such as a sales forecast). Constraints on systems development may include time, finances, and legal issues.

Remember, be impertinent and question whether stated constraints are firm. You may want to consider some design alternatives that violate constraints you consider to be flexible.

Developing Design Strategies for Hoosier Burger's New Inventory Control System

As an example of alternative generation and selection, let's look at an inventory control system that Hoosier Burger wants developed. Figure 7-15 lists ranked requirements and constraints for the enhanced information system being considered by Hoosier Burger. The requirements represent a sample of those developed from the requirements determination and structuring carried out in prior analysis steps. The system in question is an upgrade to the company's existing inventory system. Before deciding to get a new inventory system, Bob Mellankamp, one of the owners of Hoosier Burger, had to follow several steps in his largely manual inventory control system, as identified in Figure 7-16.

Using the current manual system, Bob first receives invoices from suppliers, and he records their receipt on an invoice log sheet. He puts the actual invoices in his accordion file. Using the invoices, Bob records the amount of stock delivered on the stock logs, paper forms posted near the point of storage for each inventory item. The stock logs include minimum order quantities, as well as spaces for posting the starting amount, amount delivered, and the amount used for each item. Amounts delivered are entered on the sheet when Bob logs stock deliveries; amounts used are entered after Bob has compared

FIGURE 7-15
Ranked system requirements and constraints for Hoosier Burger's inventory system.

SYSTEM REQUIREMENTS (in descending priority)	SYSTEM CONSTRAINTS (in descending order)
1. Must be able to easily enter shipments into system as soon as they are received.	1. System development can cost no more than $50,000.
2. System must automatically determine whether and when a new order should be placed.	2. New hardware can cost no more than $50,000.
3. Management should be able to determine at any time approximately what inventory levels are for any given item in stock.	3. The new system must be operational in no more than six months from the start of the contract.
	4. Training needs must be minimal (i.e., the new system must be easy to use).

FIGURE 7-16
The steps in Hoosier Burger's inventory control system.

1. Meet delivery trucks before opening restaurant.
2. Unload and store deliveries.
3. Log invoices and file in accordion file.
4. Manually add amounts received to stock logs.
5. After closing, print inventory report.
6. Count physical inventory amounts.
7. Compare inventory reports totals to physical count totals.
8. Compare physical count totals to minimum order quantities; if the amount is less, make order; if not, do nothing.
9. Pay bills that are due and record them as paid.

the amounts of stock used, according to a physical count and according to the numbers on the inventory report generated by the food-ordering system. Some Hoosier Burger items, especially perishable goods, have standing orders for daily delivery.

The Mellankamps want to improve their inventory system so that new orders are immediately accounted for, so that the system can determine when new orders should be placed, and so that management can obtain accurate inventory levels at any time of the day. All three of these system requirements have been ranked in order of descending priority in Figure 7-15. A logical data-flow diagram showing the key processes in the desired inventory system is shown in Figure 7-17. The goal of having new orders automatically accounted for is reflected in Process 1.0. The goal of having the system determine when new orders should be placed is realized in Process 3.0. The third goal for the new system, of allowing managers to obtain accurate inventory levels at any time, is captured by Process 5.0. The two other processes in Figure 7-17, generating payments (4.0) and updating inventory levels due to usage (2.0), are part of the existing manual system.

The constraints on developing an enhanced inventory system at Hoosier Burger are also listed in Figure 7-15, again in order of descending priority. The first two constraints cover costs for systems development and for new computer hardware. Development can cost no more than $50,000. New hardware can cost no more than $50,000. The third constraint involves time for development—Hoosier Burger wants the system to be installed and in operation in no more than six months from the beginning of the development project. Finally, Hoosier Burger would prefer that training for the system be simple; the new system must be designed so that it is easy to use. However, because it is the fourth most important constraint, the demands it makes are more flexible than those contained in the other three.

Any set of alternative solutions to Hoosier Burger's inventory system problems must be developed with the company's prioritized requirements and

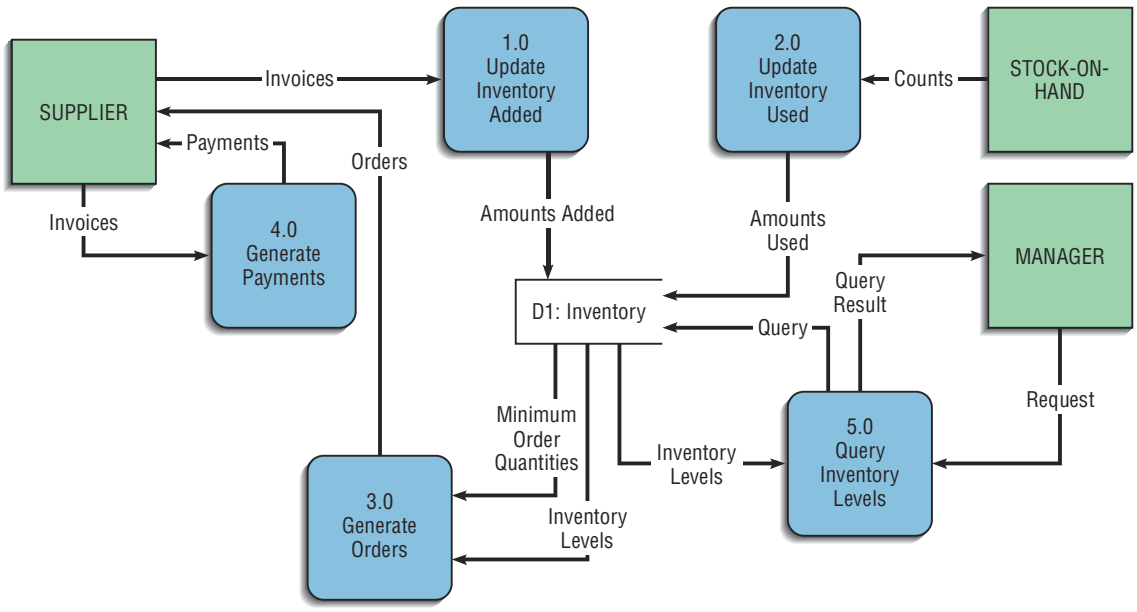


FIGURE 7-17
A logical data-flow diagram showing the key processes in Hoosier Burger's desired inventory system.

CRITERIA	ALTERNATIVE A	ALTERNATIVE B	ALTERNATIVE C
Requirements			
1. Easy real-time entry of new shipment data	Yes	Yes	Yes
2. Automatic reorder decisions	For some items	For all items	For all items
3. Real-time data on inventory levels	Not available	Available for some items only	Fully available
Constraints			
1. Cost to develop	$25,000	$50,000	$65,000
2. Cost of hardware	$25,000	$50,000	$50,000
3. Time to operation	Three months	Six months	Nine months
4. Ease of training	One week of training	Two weeks of training	One week of training

FIGURE 7-18
Description of three alternative systems that could be developed for Hoosier Burger's inventory system.

constraints in mind. Figure 7-18 illustrates how each of three possible alternatives meets (or exceeds) the criteria implied in Hoosier Burger's requirements and constraints. Alternative A is a low-end solution. It meets only the first requirement completely and partially satisfies the second requirement, but it does not meet the final one. However, Alternative A is relatively inexpensive to develop and requires hardware that is much less expensive than the largest amount Hoosier Burger is willing to pay. Alternative A also meets the requirements for the other two constraints: It will take only 3 months to become operational, and users will require only 1 week of training. Alternative C is the high-end solution. Alternative C meets all of the requirements criteria. On the other hand, Alternative C violates two of the four constraints: Development costs are high at $65,000, and time to operation is 9 months. If Hoosier Burger really wants to satisfy all three of its requirements for its new inventory system, the company will have to pay more than it wants and will have to wait longer for development. Once operational, however, Alternative C will take just as much time to train people to use as Alternative A. Alternative B is in the middle. This alternative solution meets the first two requirements, partially satisfies the third, and does not violate any of the constraints.

Now that three plausible alternative solutions have been generated for Hoosier Burger, the analyst hired to study the problem has to decide which one to recommend to management for development. Management will then decide whether to continue with the development project (incremental commitment) and whether the system recommended by the analyst should be developed.

Selecting the Most Likely Alternative

One method we can use to decide among the alternative solutions to Hoosier Burger's inventory system problem is illustrated in Figure 7-19. On the left, you see that we have listed all three system requirements and all four constraints from Figure 7-15. These are our decision criteria. We have weighted requirements as a group and constraints as a group equally; that is, we believe that requirements are just as important as constraints. We do not have to weight requirements and constraints equally; it is certainly possible to make requirements more or less important than constraints. Weights are arrived at

Criteria	Weight	Alternative A		Alternative B		Alternative C	
		Rating	Score	Rating	Score	Rating	Score
Requirements							
Real-time data entry	18	5	90	5	90	5	90
Auto reorder	18	3	54	5	90	5	90
Real-time data query	14	1	14	3	42	5	70
	50		158		222		250
Constraints							
Development costs	20	5	100	4	80	3	60
Hardware costs	15	5	75	4	60	4	60
Time to operation	10	5	50	4	40	3	30
Ease of training	5	5	25	3	15	5	25
	50		250		195		175
Total	100		408		417		425

FIGURE 7-19
Weighted approach for comparing the three alternative systems for Hoosier Burger's inventory system.

in discussions among the analysis team, users, and sometimes managers. Weights tend to be fairly subjective, and for that reason, should be determined through a process of open discussion to reveal underlying assumptions, followed by an attempt to reach consensus among stakeholders. We have also assigned weights to each individual requirement and constraint. Notice that the total of the weights for both requirements and constraints is 50. Our weights correspond with our prioritization of the requirements and constraints.

Our next step is to rate each requirement and constraint for each alternative, on a scale of 1 to 5. A rating of 1 indicates that the alternative does not meet the requirement well or that the alternative violates the constraint. A rating of 5 indicates that the alternative meets or exceeds the requirement or clearly abides by the constraint. Ratings are even more subjective than weights and should also be determined through open discussion among users, analysts, and managers. The next step is to multiply the rating for each requirement and each constraint by its weight and follow this procedure for each alternative. The final step is to add up the weighted scores for each alternative. Notice that we have included three sets of totals: for requirements, for constraints, and for overall totals. If you look at the totals for requirements, Alternative C is the best choice (score of 250), because it meets or exceeds all requirements. However, if you look only at constraints, Alternative A is the best choice (score of 250), as it does not violate any constraints. When we combine the totals for requirements and constraints, we see that the best choice is Alternative C (score of 425), even though it had the lowest score for constraints, as it has the highest overall score.

Alternative C, then, appears to be the best choice for Hoosier Burger. Whether Alternative C is actually chosen for development is another issue. The Mellankamps may be concerned that Alternative C violates two constraints, including the most important one, development costs. On the other hand, the owners (and chief users) at Hoosier Burger may want the full functionality Alternative C offers that they are willing to ignore the constraints violations. Or Hoosier Burger's management may be so interested in cutting costs that it prefers Alternative A, even though its functionality is severely limited. What may appear to be the best choice for a systems development project may not always be the one that ends up being developed.

Key Points Review

1. **Concisely define each of the following key data-modeling terms: *conceptual data model, entity-relationship diagram, entity type, entity instance, attribute, candidate key, multivalued attribute, relationship, degree, cardinality,* and *associative entity*.**

   A conceptual data model represents the overall structure of organizational data, independent of any database technology. An E-R diagram is a detailed representation of the entities, associations, and attributes for an organization or business area. An entity type is a collection of entities that share common properties or characteristics. An attribute is a named property or characteristic of an entity. One or a combination of attributes that uniquely identifies each instance of an entity type is called a candidate key. A multivalued attribute may take on more than one value for an entity instance. A relationship is an association between the instances of one or more entity types, and the number of entity types participating in a relationship is the degree of the relationship. Cardinality is the number of instances of entity B that can (or must) be associated with each instance of entity A. Data that are simultaneously associated with several entity instances are stored in an associative entity.
2. **Ask the right kinds of questions to determine data requirements for an information system.**

   Information is gathered for conceptual data modeling as part of each phase of the systems development life cycle. You must ask questions in business, rather than data modeling, terms so that business managers can explain the nature of the business; the systems analyst represents the objects and events of the business through a data model. Questions include: What are the objects of the business? What uniquely characterizes each object? What characteristics describe each object? How are data used? What history of data must be retained? What events occur that relate different kinds of data, and are there special data-handling procedures? (See Table 7-1 for details.)
3. **Draw an entity-relationship (E-R) diagram to represent common business situations.**

   An E-R diagram uses symbols for entity, relationship, identifier, attribute, multivalued attribute, and associative entity and shows the degree and cardinality of relationships (see Figure 7-5 for all the symbols discussed in this chapter, and see Figures 7-3 and 7-11 for example diagrams). Exercises at the end of this chapter give you practice at drawing E-R diagrams.
4. **Explain the role of conceptual data modeling in the overall analysis and design of an information system.**

   Conceptual data modeling occurs in parallel with other requirements analysis and structuring steps during systems development. Information for conceptual data modeling is collected during interviews, from questionnaires, and in JAD sessions. Conceptual data models may be developed for a new information system and for the system it is replacing, as well as for the whole database for current and new systems. A conceptual data model is useful input to subsequent data-oriented steps in the analysis, design, and implementation phases of systems development where logical data models, physical file designs, and database file coding are done.
5. **Distinguish between unary, binary, and ternary relationships and give an example of each.**

   A unary relationship is between instances of the same entity type (e.g., Is_married_to relates different instances of a PERSON entity type). A binary relationship is between instances of two entity types (e.g., Registers_for relates instances of STUDENT and COURSE entity types). A ternary relationship is a simultaneous association among instances of three entity types (e.g., Supplies relates instances of PART, VENDOR, and WAREHOUSE entity types).
6. **Distinguish between a relationship and an associative entity, and use associative entities in a data model when appropriate.**

   Sometimes many-to-many and one-to-one relationships have associated attributes. When this occurs, it is best to change the relationship into an associative entity. For example, if we needed to know the date an employee completed a course, Date_Completed is neither an attribute of EMPLOYEE nor COURSE but of the relationship between these entities. In this case, we would create a CERTIFICATE associative entity (see Figure 7-7), associate Date_Completed with CERTIFICATE, and draw mandatory one relationships from CERTIFICATE to each of EMPLOYEE and COURSE. An associative entity, like any entity, then may be related to other entities, as shown in Figure 7-8.
7. **Relate data modeling to process and logic modeling as different ways of describing an information system.**

   Process and logic modeling represent the movement and use of data, whereas data modeling represents the meaning and structure of data.

A data model is usually a more permanent representation of the data requirements of an organization than are models of data flow and use. Still, consistency between these models of different views of an information system is required. For example, all the data in an E-R diagram for an information system must be in data stores on associated data-flow diagrams.

8. **Generate at least three alternative design strategies for an information system.**

   Generating different alternatives is something you would do in actual systems analysis or as part of a class project. Three is not a magic number. It represents instead the endpoints and midpoint of a series of alternatives, such as the most expensive, the least expensive, and an alternative somewhere in the middle.

9. **Select the best design strategy using both qualitative and quantitative methods.**

   Once developed, alternatives can be compared to each other through quantitative methods, but the actual decision may depend on other criteria, such as organizational politics. In this chapter, you were introduced to one way to compare alternative design strategies quantitatively.

Key Terms Checkpoint

Here are the key terms from the chapter. The page where each term is first explained is in parentheses after the term.

1. **Associative entity (p. 204)**
2. **Attribute (p. 199)**
3. **Binary relationship (p. 203)**
4. **Candidate key (p. 199)**
5. **Cardinality (p. 203)**
6. **Conceptual data model (p. 190)**
7. **Degree (p. 202)**
8. **Design strategy (p. 213)**
9. **Entity (p. 197)**
10. **Entity instance (instance) (p. 198)**
11. **Entity-relationship diagram (E-R diagram) (p. 197)**
12. **Entity type (p. 198)**
13. **Identifier (p. 199)**
14. **Multivalued attribute (p. 200)**
15. **Relationship (p. 201)**
16. **Repeating group (p. 201)**
17. **Ternary relationship (p. 203)**
18. **Unary relationship (recursive relationship) (p. 202)**

Match each of the key terms above with the definition that best fits it.

____ 1. A graphical representation of the entities, associations, and data for an organization or business area; it is a model of entities, the associations among those entities, and the attributes of both the entities and their associations.
____ 2. A single occurrence of an entity type.
____ 3. An attribute that may take on more than one value for each entity instance.
____ 4. A simultaneous relationship among instances of three entity types.
____ 5. A collection of entities that share common properties or characteristics.
____ 6. A relationship between instances of two entity types.
____ 7. 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.
____ 8. A named property or characteristic of an entity that is of interest to the organization.
____ 9. The number of instances of entity B that can (or must) be associated with each instance of entity A.
____ 10. A candidate key that has been selected as the unique, identifying characteristic for an entity type.
____ 11. An association between the instances of one or more entity types that is of interest to the organization.
____ 12. An attribute (or combination of attributes) that uniquely identifies each instance of an entity type.
____ 13. The number of entity types that participate in a relationship.
____ 14. A relationship between the instances of one entity type.
____ 15. A detailed model that shows the overall structure of organizational data but is independent of any database management system or other implementation considerations.
____ 16. A set of two or more multivalued attributes that are logically related.
____ 17. A person, place, object, event, or concept in the user environment about which the organization wishes to maintain data.
____ 18. A particular approach to developing an information system. It includes statements on the system's functionality, hardware and system software platform, and method for acquisition.

Review Questions

1. What characteristics of data are represented in an E-R diagram?
2. What elements of a data-flow diagram should be analyzed as part of data modeling?
3. Explain why a ternary relationship is not the same as three binary relationships.
4. When must a many-to-many relationship be modeled as an associative entity?
5. Which of the following types of relationships can have attributes associated with them: one-to-one, one-to-many, many-to-many?
6. What is the degree of a relationship? Give an example of each of the relationship degrees illustrated in this chapter.
7. Give an example of a ternary relationship (different from any example in this chapter).
8. List the deliverables from conceptual data modeling.
9. Explain the relationship between minimum cardinality and optional and mandatory participation.
10. List the ideal characteristics of an entity identifier attribute.
11. List the four types of E-R diagrams produced and analyzed during conceptual data modeling.
12. What notation is used on an E-R diagram to show the minimum and maximum cardinalities on a one-to-many relationship?
13. Explain the difference between a candidate key and the identifier of an entity type.
14. What distinguishes a repeating group from a simple multivalued attribute?
15. How do analysts generate alternative solutions to information systems problems?
16. How do managers decide which alternative design strategy to develop?

Problems and Exercises

1. Assume that at Pine Valley Furniture each product (described by Product No., Description, and Cost) consists of at least three components (described by Component No., Description, and Unit of Measure), and components are used to make one or many products (i.e., must be used in at least one product). 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 being used to make products and components being used to make other components, we need to keep track of how many components go into making something else. Draw an E-R diagram for this situation and place minimum and maximum cardinalities on the diagram.
2. A performance venue hosts many concert series a year. Performers have a name and perform several times in a concert series (each constituting a performance with a different date). Concert series have one or more performers and have a name and a specified seating arrangement. A concert series is held in one (and only one) of several concert halls, each of which has a room number. Represent this situation of concerts and performers with an E-R diagram.
3. A restaurant chain has several store locations in a city (with a name and zip code stored for each), and each is managed by one manager. Managers manage only one store. Each restaurant location has its own unique set of menus. Most have more than one menu (e.g., lunch and dinner menus). Each menu has many menu items, and items can appear on multiple menus, and with different prices on different menus. Represent this situation of restaurants with an E-R diagram.
4. Consider the E-R diagram in Figure 7-7.
   a. What is the identifier for the CERTIFICATE associative entity?
   b. Now, assume that the same employee may take the same course multiple times, on different dates. Does this change your answer to part a? Why or why not?
   c. Now, assume we do know the instructor who issues each certificate to each employee for each course. Include this new entity in Figure 7-7 and relate it to the other entities. How did you choose to relate INSTRUCTOR to CERTIFICATE and why?
5. Consider the E-R diagram in Figure 7-20. Based on this E-R diagram, answer the following questions:
   a. How many EMPLOYEES can work on a project?
   b. What is the degree of the Used_on relationship?
   c. Do any associative entities appear in this diagram? If so, name them.
   d. How else could the attribute Skill be modeled?
   e. What attributes might be attached to the Works_on relationship?
   f. Could TOOL be modeled as an associative entity? Why or why not?

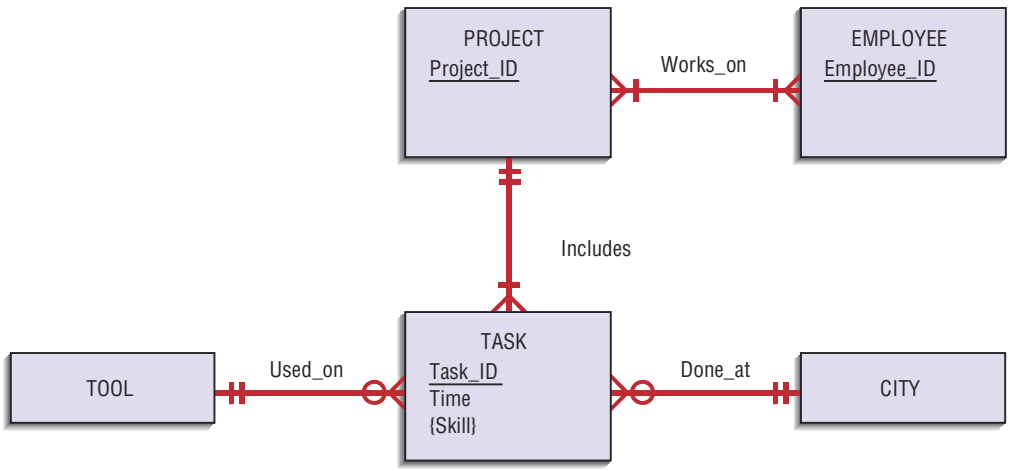


FIGURE 7-20
E-R diagram for Problem and Exercise 5.

6. A car rental is an association between a customer, sales agent, and a car. Select a few pertinent attributes for each of these entity types and represent a rental in an E-R diagram.
7. Consider the E-R diagram in Figure 7-21. Are all three relationships—Holds, Goes_on, and Transports—necessary (i.e., can one of these be deduced from the other two)? What, if any, reasonable assumptions make all three relationships necessary?
8. Draw an E-R diagram to represent the sample customer order in Figure 7-4.
9. A company database contains an entity called EMPLOYEE. Among other information, the company records information about any degrees each employee has earned, along with the graduation date for the degree.
   a. Represent the EMPLOYEE entity and its degree attributes using the notation for multivalued attributes.
   b. Represent the EMPLOYEE entity and its degree attributes using two entity types.
   c. Finally, assume the company decides to also keep data about the institution from which the employees' degrees were earned, including name of the institution, city, and state where the institution is located. Augment your answer to part b to accommodate this new entity type.

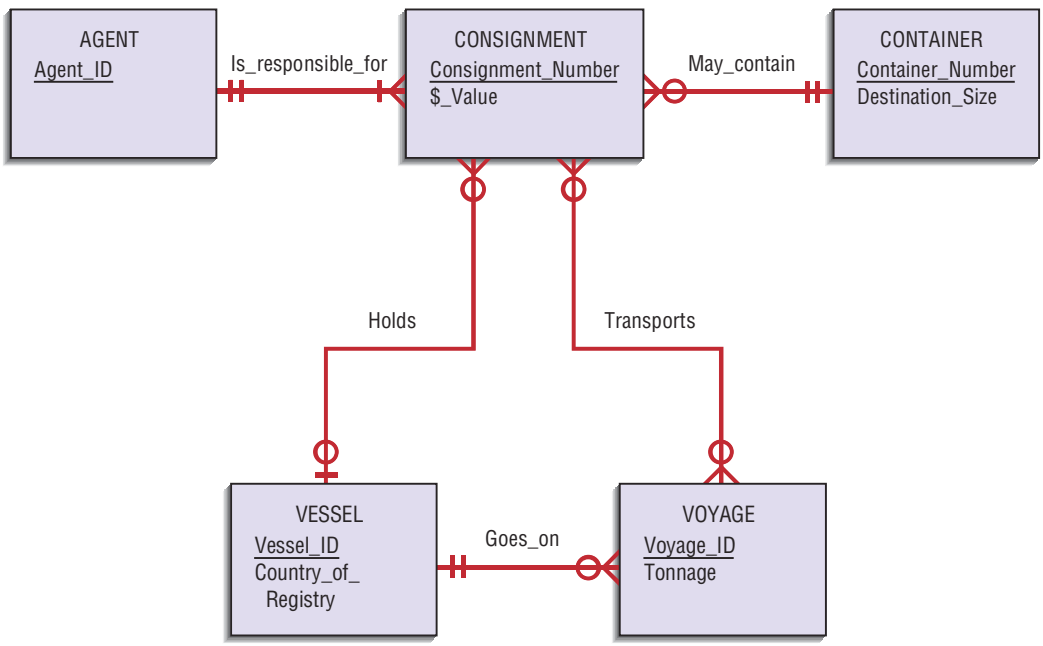


FIGURE 7-21
E-R diagram for Problem and Exercise 7.

10. Consider the Is_married_to unary relationship in Figure 7-6.
    a. Draw minimum and maximum cardinalities for each end of this relationship.
    b. Assume we wanted to know the date on which a marriage occurred. Augment this E-R diagram to include a Date_married attribute.
    c. Because persons sometimes remarry after the death of a spouse or a divorce, redraw this E-R diagram to show the whole history of marriages (not just the current marriage) for PERSONs. Show the Date_married attribute on this diagram.
11. Draw an E-R diagram for each of the following situations:
    a. 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.
    b. 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 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?
12. Recreate the spreadsheet in Figure 7-19 in your spreadsheet package. Change the weights and compare the outcome to Figure 7-19. Change the rankings. Add criteria. What additional information does this "what if" analysis provide for you as a decision maker? What insight do you gain into the decision-making process involved in choosing the best alternative system design?
13. The method for evaluating alternatives used in Figure 7-19 is called weighting and scoring. This method implies that the total utility of an alternative is the sum of the products of the weights and ratings of each criterion for the alternative. What assumptions are characteristic of this method for evaluating alternatives? That is, what conditions must be true for this to be a valid method of evaluating alternatives?
14. Weighting and scoring (see Problem and Exercise 13) is only one method for comparing alternative solutions to a problem. Go to the library, find a book or articles on qualitative and quantitative decision making and voting methods, and outline two other methods for evaluating alternative solutions to a problem. What are the pros and cons of these methods compared to the weighting and scoring method? Under weighting and scoring and the other alternatives you find, how would you incorporate the opinions of multiple decision makers?
15. Prepare an agenda for a meeting at which you would present the findings of the analysis phase of the SDLC to Bob Mellankamp concerning his request for a new inventory control system. Use information provided in Chapters 5 through 7 as background in preparing this agenda. Concentrate on which topics to cover, not the content of each topic.
16. The owner of two pizza parlors located in adjacent towns wants to computerize and integrate sales transactions and inventory management within and between both stores. The point-of-sale component must be easy to use and flexible enough to accommodate a variety of pricing strategies and coupons. The inventory management, which will be linked to the point-of-sale component, must also be easy to use and fast. The systems at each store need to be linked so that sales and inventory levels can be determined instantly for each store and for both stores combined. The owner can allocate $40,000 for hardware and $20,000 for software and must have the new system operational in three months. Training must be short and easy. Briefly describe three alternative systems for this situation and explain how each would meet the requirements and constraints. Are the requirements and constraints realistic? Why or why not?
17. Compare the alternative systems from Problem and Exercise 16 using the weighted approach demonstrated in Figure 7-19. Which system would you recommend? Why? Was the approach taken in this and Problem and Exercise 16 useful even for this relatively small system? Why or why not?

18. Suppose that an analysis team did not generate alternative design strategies for consideration by a project steering committee or client. What might the consequences be of having only one design strategy? What might happen during the oral presentation of project progress if only one design strategy is offered?
19. Assume you are designing a database for a local used car dealership. Attributes for a car include the vehicle identification number, stock number, make, model, year, and trim. What would you use for the primary key in this entity? What attributes are likely to be foreign keys associated with other entities?

Discussion Questions

1. Discuss why some systems developers believe that a data model is one of the most important parts of the statement of information system requirements.
2. Using Table 7-1 as a guide, develop a script of at least ten questions you would ask during an interview of the customer-order processing department manager at Pine Valley Furniture. Assume the focus is on analyzing the requirements for a new order-entry system. The purpose of the interview is to develop a preliminary E-R diagram for this system.
3. If possible, contact a systems analyst in a local organization. Discuss with this systems analyst the role of conceptual data modeling in the overall systems analysis and design of information systems at his or her company. How, and by whom, is conceptual data modeling performed? What training in this technique is given? At what point(s) is this done in the development process? Why?
4. Talk to MIS professionals at a variety of organizations and determine the extent to which CASE tools are used in the creation and editing of entity-relationship diagrams. Try to determine whether they use CASE tools for this purpose; which CASE tools are used; and why, when, and how they are used. In companies that do not use CASE tools for this purpose, determine why not and what would have to change in order to use them.
5. Ask a systems analyst to give you a copy of the standard notation he or she uses to draw E-R diagrams. In what ways is this notation different from notation in this text? Which notation do you prefer and why? What is the meaning of any additional notation?
6. Consider the purchase of a new PC to be used by you at your work (or by you at a job that you would like to have). Describe in detail three alternatives for this new PC that represent the low, middle, and high points of a continuum of potential solutions. Be sure that the low-end PC meets at least your minimum requirements and the high-end PC is at least within a reasonable budget. At this point, without quantitative analysis, which alternative would you choose?
7. For the new PC described in Question 6, develop ranked lists of your requirements and constraints as displayed in Figure 7-19. Display the requirements and constraints, along with the three alternatives, as done in Figure 7-19, and note how each alternative is rated on each requirement and constraint. Calculate scores for each alternative on each criterion and compute total scores. Which alternative has the highest score? Why? Does this choice fit with your selection in the previous question? Why or why not?

Case Problems

1. Pine Valley Furniture

   In order to determine the requirements for the new Customer Tracking System, several JAD sessions, interviews, and observations were conducted. Resulting information from these requirements determination methods was useful in the preparation of the Customer Tracking System's data-flow diagrams.

   One afternoon while you are working on the Customer Tracking System's data-flow diagrams, Jim Woo stops by your desk and assigns you the task of preparing a conceptual entity-relationship diagram for the Customer Tracking System. Later that afternoon, you review the requirements-determination phase deliverables, including the data-flow diagrams you have just finished preparing.

   Your review of these deliverables suggests that the Customer Tracking System's primary objective is to track and forecast customer buying patterns. Additionally, in order to track a customer's buying habits, an order history must be established, satisfaction levels assessed, and a variety of demographic data collected. The demographic data will categorize the customer according to type, geographic location, and type of purchase. Customer Tracking System information will enable Pine Valley Furniture to better forecast its product

demand, control its inventory, and solicit customers. Also, the Customer Tracking System's ability to interface with the WebStore is important to the project.

a. What entities are identified in the previous scenario? Can you think of additional entities? What interrelationships exist between the entities?
b. For each entity, identify its set of associated attributes. Specify identifiers for each entity.
c. Based on the case scenario and your answers to parts a and b, prepare an entity-relationship diagram. Be sure to specify the cardinalities for each relationship.
d. How does this conceptual model differ from the WebStore's conceptual model?

2. Hoosier Burger

Although Hoosier Burger is well recognized for its fast foods, especially the Hoosier Burger Special, plate lunches are also offered. These include such main menu items as barbecue ribs, grilled steak, meat loaf, and grilled chicken breast. The customer can choose from a variety of side items, including roasted garlic mashed potatoes, twice-baked potatoes, coleslaw, corn, baked beans, and Caesar salad.

Many downtown businesses often call and place orders for Hoosier Mighty Meals. These are combination meals consisting of a selection of main menu items and three side orders. The customer can request Hoosier Mighty Meals to feed 5, 10, 15, or 20 individuals. As a convenience to its business customers, Bob and Thelma allow business customers to charge their order. Once each month, a bill is generated and sent to those business customers who have charged their orders. Bob and Thelma have found that many of their business customers are repeat customers and often place orders for the same Hoosier Mighty Meals. Bob asks you if it is possible to track a customer's order history, and you indicate that it is indeed possible.

a. Based on the information provided in the case scenario, what entities will Hoosier Burger need to store information about?
b. For the entities identified in part a, identify a set of attributes for each entity.
c. Specify an identifier for each entity. What rules did you apply when selecting the identifier?
d. Modify Figure 7-10 to reflect the addition of these new entities. Be sure to specify the cardinalities for each relationship.

3. Corporate Technology Center

Five years ago, Megan Thomas was a busy executive seeking to keep herself and her employees current with new technology. She realized that many small companies were facing the same dilemma. Using her life savings and money from investors, Megan founded Corporate Technology Center. Corporate Technology Center's primary objective is to offer technology update seminars to local business executives and their employees. A wide variety of seminars are offered, including ones covering operating systems, spreadsheets, word processing, database management, Internet, Web page design, and telecommunications.

Although Corporate Technology Center offers seminars at its own campus, it also provides on-site training for local companies. One-day, two-day, or four-day seminars are offered. Courses are open to a minimum of twenty students and a maximum of forty students. Although several staff members are capable of teaching any given course, generally only one staff member teaches a given course on a given date.

a. What entities are identified in the previous scenario? Can you identify additional entities?
b. For each entity identified in part a, specify a set of associated attributes.
c. Select an identifier for each entity. What rules did you apply when selecting the identifier?
d. Based on the case scenario and your answers to a, b, and c, prepare an entity-relationship diagram. Be sure to specify the cardinalities for each relationship.

4. Pine Valley Furniture

During your time as a Pine Valley Furniture intern, you have learned much about the systems analysis and design process. You have been able to observe Jim Woo as he serves as the lead analyst on the WebStore project, and you have also received hands-on experience with the Customer Tracking System project. The requirements determination and requirements structuring activities for the Customer Tracking System are now complete, and it is time to begin generating alternative design strategies.

On Monday afternoon, Jim Woo stops by your desk and requests that you attend a meeting scheduled for tomorrow morning. He mentions that during tomorrow's meeting, the Customer Tracking System's requirements and constraints, weighting criteria, and alternative design strategy ratings will be discussed. He also mentions that during the previously conducted systems planning and selection phase, Jackie Judson and he prepared a baseline project plan. At the time the initial baseline project plan was prepared, the in-house development option was the preferred design strategy. The marketing group's unique

information needs seemed to indicate that in-house development was the best option. However, other alternative design strategies have since been investigated.

During Tuesday's meeting, several end users, managers, and systems development team members meet, discuss, and rank the requirements and constraints for the new Customer Tracking System. Also, weights and rankings are assigned to the three alternative design strategies. At the end of the meeting, Jim Woo assigns you the task of arranging this information into a table and calculating the overall scores for each alternative. He would like to review this information later in the afternoon. Tables 7-5 and 7-6 summarize the information obtained from Tuesday's meeting.

a. Generally speaking, what alternative design strategies were available to Pine Valley Furniture?
b. Of the alternative design strategies available to Pine Valley Furniture, which were the most viable? Why?
c. Using the information provided in Table 7-6, calculate the scores for each alternative.
d. Based on the information provided in Tables 7-5 and 7-6, which alternative do you recommend?

5. Hoosier Burger

As the lead analyst on the Hoosier Burger project, you have been busy collecting, structuring, and evaluating the new system's requirements. During a Monday morning meeting with Bob and Thelma, the three of you review the system requirements, system constraints, and alternative design strategies. The proposed alternative design strategies address low-end, midrange, and high-end solutions. Additionally, weights are assigned to the evaluation criteria, and the alternatives are ranked according to the criteria.

Bob has stated repeatedly that his main priority is to implement an inventory control system. However, you are aware that, if at all possible, Bob would like to also implement a delivery system. You inform Bob that two of the alternative design strategies support a delivery system but will increase the system's development cost by at least $20,000 and will add $10,000 in recurring costs to the new system. Bob feels that the addition of the new delivery system will result in $25,000 in yearly benefits over the life of the new system.

The inclusion of a delivery system necessitates the addition of several new requirements and the modification of system constraints. Table 7-7

TABLE 7-5: Pine Valley Furniture Requirements and Constraints

Criteria	Alternative A	Alternative B	Alternative C
New Requirements			
Ease of use	Acceptable	Fair	Good
Easy real-time updating of customer profiles	Yes	Yes	Yes
Tracks customer purchasing activity	No	Yes	Yes
Supports sales forecasting	Some forecasting models are supported	Some forecasting models are supported	Provides support for all necessary forecasting models
Ad hoc report generation	No	Yes	Yes
Constraints			
Must interface with existing systems	Requires significant modifications	Minor modifications	Minor modifications
Costs to develop	$150,000	$200,000	$350,000
Cost of hardware	$80,000	$80,000	$100,000
Time to operation	6 months	7 months	9 months
Must interface with existing systems	Requires significant modifications	Minor modifications	Minor modifications
Ease of training	3 weeks of training	3 weeks of training	2 weeks of training
Legal restrictions	Cannot be modified	Allows for customization	None

TABLE 7-6: Pine Valley Furniture Multi-Criteria Analysis

Criteria	Weight	Alternative A		Alternative B		Alternative C	
		Rating	Score	Rating	Score	Rating	Score
Requirements							
Ease of use	15	2		3		5	
Real-time customer profile updating	12	3		3		4	
Tracks customer purchasing activity	12	1		3		3	
Sales forecasting	8	2		2		3	
Ad hoc report generation	3	1		2		3	
Total	50						
Constraints							
Interfaces with existing systems	15	3		4		2	
Development costs	10	5		4		2	
Hardware costs	10	5		4		2	
Time to operation	5	4		1		2	
Ease of training	5	2		2		4	
Legal restrictions	5	1		2		5	
Total	50						

outlines these changes. The weights, ratings, and scores also require adjustments. Table 7-8 contains information about these adjustments.

a. Generally speaking, what alternative design strategies are available to Hoosier Burger?
b. Is an enterprise resource planning system a viable option for Hoosier Burger? Why or why not?
c. Modify Figure 7-19 to incorporate the criteria mandated by the new delivery system. Which alternative should be chosen?
d. Assuming that Alternative C is still chosen, update Hoosier Burger's economic feasibility analysis to reflect the changes mentioned in this scenario.

TABLE 7-7: Hoosier Burger Requirements and Constraints

Criteria	Alternative A	Alternative B	Alternative C
New Requirements			
Easy real-time entry of new shipment data	Yes	Yes	Yes
Automatic reorder decisions	For some items	For all items	For all items
Real-time data on inventory levels	Not available	Available for some items only	Fully available
Facilitates forecasting	Not available	Available	Available
Track delivery sales	Available	Available	Available
Customer billing	Not available	Not available	Available
Constraints			
Costs to develop	$45,000	$70,000	$85,000
Cost of hardware	$25,000	$50,000	$50,000
Time to operation	4 months	7 months	10 months
Ease of training	1 week of training	3 weeks of training	3 weeks of training

TABLE 7-8: Hoosier Burger Multi-Criteria Analysis

Criteria	Weight	Alternative A		Alternative B		Alternative C	
		Rating	Score	Rating	Score	Rating	Score
Requirements							
Real-time data entry	12	5		5		5	
Auto reorder	12	3		5		5	
Real-time data query	10	1		3		5	
Facilitates forecasting	8	1		2		3	
Track delivery sales	5	3		3		3	
Customer billing	3	1		1		3	
Total	50						
Constraints							
Development costs	20	5		4		2	
Hardware costs	15	5		4		3	
Time to operation	10	5		4		3	
Ease of training	5	2		1		5	
Total	50						

CASE: PETRIE'S ELECTRONICS

Structuring Systems Requirements: Conceptual Data Modeling

Jim Watanabe, manager of the "No Customer Escapes" project, and assistant director of IT for Petrie's Electronics, was sitting in the company cafeteria. He had just finished his house salad and was about to go back to his office when Stephanie Welch sat down at his table. Jim had met Stephanie once, back when he started work at Petrie's. He remembered she worked for the database administrator.

"Hi, Jim, remember me?" she asked.

"Sure, Stephanie, how are you? How are things in database land?"

"Can't complain. Sanjay asked me to talk to you about the database needs for your new customer loyalty system." Stephanie's phone binged. She pulled it out of her oversize bag and looked at it. She started to text as she continued to talk to Jim. "How far along are you on your database requirements?"

That's kinda rude, Jim thought. Oh well. "We are still in the early stages. I can send you a very preliminary E-R diagram we have [PE Figure 7-1], along with a description of the major entities."

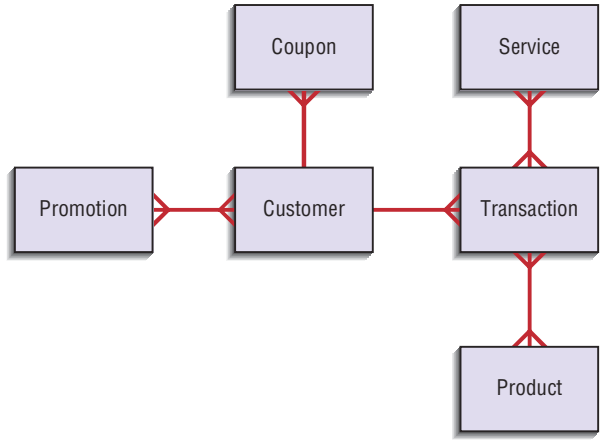
"OK, that will help. I suspect that you won't have too many new entities to add to what's already in the system," Stephanie responded, still looking at her phone and still texting. She briefly looked up at Jim and smiled slightly before going back to texting. "Just send the E-R to me, and I'll let you know if I have any questions." She stood up, still looking at her phone. "Gotta go," she said, and she walked away.

OK, Jim thought, I need to remember to send Stephanie the preliminary E-R we have. I should probably send her the entity descriptions too (PE Table 7-1), just in case. Jim stood up, carried his tray over to the recycling area of the cafeteria, and went back to his office.

When Jim got back to his office, Sanjay was waiting for him.

"I've got more information on those alternatives we talked about earlier," Sanjay said. "I had one of my employees gather some data on how the alternatives might satisfy our needs." (See the descriptions of the alternatives in PE Table 5-2.) Sanjay handed Jim a short report. "The matrix shows the requirements and constraints for each alternative and makes it relatively easy to compare them." (See PE Figure 7-2.)

PE FIGURE 7-1
Initial E-R for Petrie's customer loyalty program.



PE TABLE 7-1: Entity Descriptions for the Preliminary E-R Diagram for Petrie's Customer Loyalty System

Entity	Description
Coupon	A coupon is a special promotion created specifically for an individual customer. A coupon is for a set dollar amount, for example, $10. The customer may use it like cash or like a dollars-off promotion when purchasing products or services. Coupons can only be created for an individual customer based on the points in his or her customer loyalty account. For each dollar value of a coupon, a certain number of points must be redeemed. Coupons must be accounted for when created and when redeemed.
Customer	A customer is someone who buys products and/or services from Petrie's Electronics. Customers include both online customers and those who shop in Petrie's Brick-and-Mortar stores.
Product	An item made available for sale to a Petrie's customer. For example, a product is a 40" Sony LCD HD television. Products can be purchased online or in Brick-and-Mortar stores.
Promotion	A promotion is a special incentive provided to a customer to entice the customer into buying a specific product or service. For example, a promotion intended to sell Blu-ray disks may involve 2-for-1 coupons. Promotions are targeted to all customers, or to subsets of customers, not just to individual customers.
Service	A job performed by one of Petrie's associates for a customer. For example, upgrading the memory in a computer by installing new memory cards is a service that Petrie's provides for a fee. Services may only be ordered and performed in Brick-and-Mortar stores, not online.
Transaction	A record that a particular product or service was sold to a specified customer on a particular date. A transaction may involve more than one product or service, and it may involve more than one of a particular kind of product or service. For example, one transaction may involve blank DVDs and prerecorded DVDs, and the prerecorded DVDs may all be of the same movie. For members of the loyalty program, each transaction is worth a number of points, depending on the dollar value of the transaction.

"The matrix favors the XRA CRM system," Jim said, after looking over the report. "It looks like their proposal meets our requirements the best, but the Nova group's proposal does the best job with the constraints."

"Yes, but just barely," Sanjay said. "There is only a five-point difference between XRA and Nova, so they are pretty comparable when it comes to constraints. But I think the XRA system has a pretty clear advantage in meeting our requirements."

"XRA seems to be pretty highly rated in your matrix in terms of all of the requirements. You have them ranked better than the other two proposals for implementation, scalability, and vendor support," Jim said. "The '5' you gave them for proven performance is one of the few '5s' you have in your whole matrix."

"That's because they are one of the best companies in the industry to work with," Sanjay responded, "Their reputation is stellar."

"This looks really promising," Jim said. "Let's see if reality matches what we have here. It's time to put together the formal request for proposal. I'll get that work started today. I hope that all three of these companies decide to bid."

Criteria	Weight	Alternative A	SBSI	Alternative B	XRA	Alternative C	Nova
		Rating	Score	Rating	Score	Rating	Score
Requirements							
Effective customer incentives	15	5	75	4	60	4	60
Easy for customers to use	10	3	30	4	40	5	50
Proven performance	10	4	40	5	50	3	30
Easy to implement	5	3	15	4	20	3	15
Scalable	10	3	30	4	40	3	30
Vendor support	10	3	30	4	40	3	30
	60		220		250		215
Constraints							
Cost to buy	15	3	45	4	60	5	75
Cost to operate	10	3	30	4	40	4	40
Time to implement	5	3	15	3	15	3	15
Staff to implement	10	3	30	4	40	3	30
	40		120		155		160
Total	100		340		405		375

PE FIGURE 7-2
Evaluation matrix for customer loyalty proposals.

Case Questions

1. Review the data-flow diagrams you developed for questions in the Petrie's Electronics case at the end of Chapter 6 (or diagrams given to you by your instructor). Study the data flows and data stored on these diagrams and decide whether you agree with the team's conclusion that the only six entity types needed are listed in the case and in PE Figure 7-1. If you disagree, define additional entity types, explain why they are necessary, and modify PE Figure 7-1 accordingly.
2. Again, review the DFDs you developed for the Petrie's Electronics case (or those given to you by your instructor). Use these DFDs to identify the attributes of each of the six entities listed in this case plus any additional entities identified in your answer to Question 1. Write an unambiguous definition for each attribute. Then, redraw PE Figure 7-1 by placing the six (and additional) entities in this case on the diagram along with their associated attributes.
3. Using your answer to Question 2, designate which attribute or attributes form the identifier for each entity type. Explain why you chose each identifier.
4. Using your answer to Question 3, draw the relationships between entity types needed by the system. Remember, a relationship is needed only if the system wants data about associated entity instances. Give a meaningful name to each relationship. Specify cardinalities for each relationship and explain how you decided on each minimum and maximum cardinality at each end of each relationship. State any assumptions you made if the Petrie's Electronics cases you have read so far and the answers to questions in these cases do not provide the evidence to justify the cardinalities you choose. Redraw your final E-R diagram in Microsoft Visio.
5. Now that you have developed in your answer to Question 4 a complete E-R diagram for the Petrie's Electronics database, what are the consequences of not having an employee entity type in this diagram? Assuming only the attributes you show on the E-R diagram, would any attribute be moved from the entity it is currently associated with to an employee entity type if it were in the diagram? Why or why not?
6. Write project dictionary entries (using standards given to you by your instructor) for all the entities, attributes, and relationships shown in the E-R diagram in your answer to Question 4. How detailed are these entries at this point? What other details still must be filled in? Are any of the entities on the E-R diagram in your answer to Question 4 weak entities? Why? In particular, is the SERVICE entity type a weak entity? If so, why? If not, why not?
7. What date-related attributes did you identify in each of the entity types in your answer to Question 4? Why are each of these needed? Can you make some observations about why date attributes must be kept in a database, based on your analysis of this database?

eight

Designing the Human Interface

Blend Images/SuperStock

Chapter Objectives

After studying this chapter, you should be able to:

- Explain the process of designing forms and reports, and the deliverables for their creation.
- Apply the general guidelines for formatting forms and reports.
- Format text, tables, and lists effectively.
- Explain the process of designing interfaces and dialogues, and the deliverables for their creation.
- Describe and apply the general guidelines for interface design, including guidelines for layout design, structuring data-entry fields, providing feedback, and system help.
- Design human-computer dialogues, including the use of dialogue diagramming.
- Discuss interface design guidelines unique to the design of Internet-based electronic commerce systems.

Chapter Preview . . .

Analysts must complete two important activities in the systems design phase, as illustrated in Figure 8-1: designing the human interface and designing databases. In this chapter, you learn guidelines to follow when designing the human-computer interface. In the first section, we describe the process of designing forms and reports and provide guidance on the deliverables produced during this process. Properly formatted segments of information are the building blocks for designing all forms and reports. We present guidelines for formatting information and for designing interfaces and dialogues. Next, we show you a method for representing human-computer dialogues called *dialogue diagramming*. Finally, we close the chapter by examining various human-computer interface design issues for Internet-based applications, specifically as they apply to Pine Valley Furniture's WebStore.

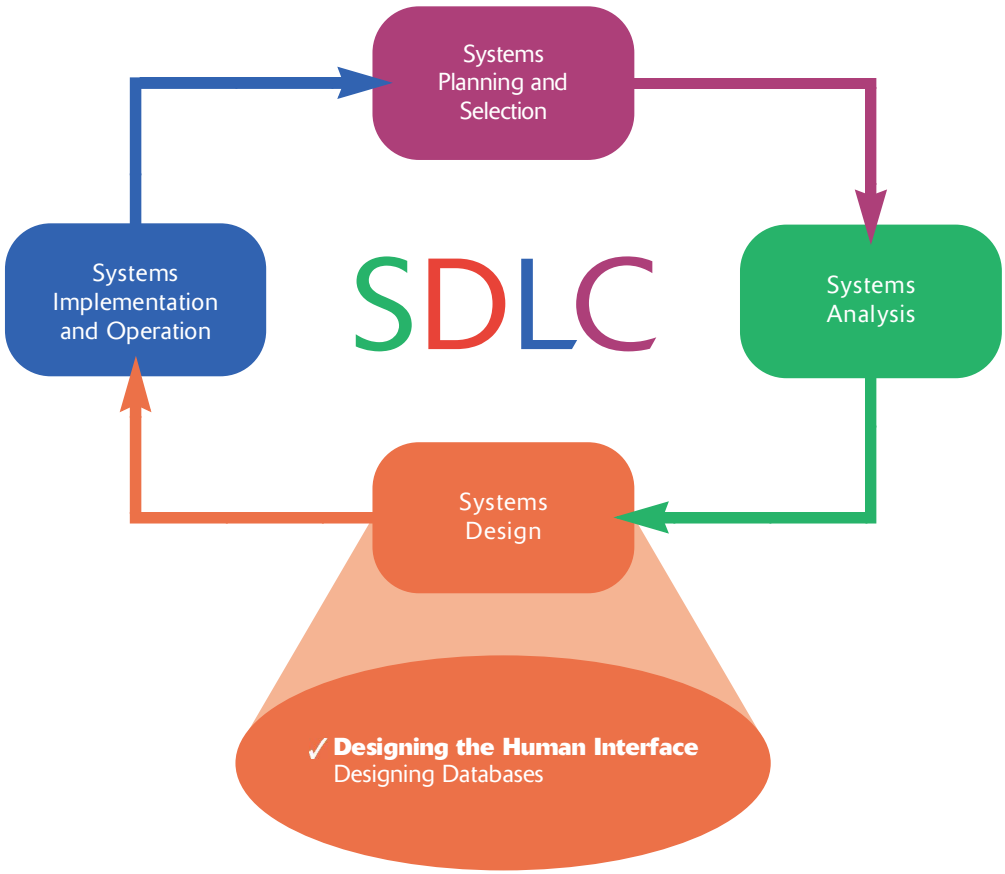


FIGURE 8-1
The systems design phase consists of two important activities: designing the human interface and designing databases.

Designing Forms and Reports

System inputs and outputs—forms and reports—are produced at the end of the systems analysis phase of the SDLC. During systems analysis, however, you may not have been concerned with the precise appearance of forms and reports. Instead, you focused on which forms and reports needed to exist and the content they needed to contain. You may have distributed to users the prototypes of forms and reports that emerged during analysis as a way to confirm requirements. Forms and reports are integrally related to the DFD and E-R diagrams developed during requirements structuring. For example, every input form is associated with a data flow entering a process on a DFD, and every output form or report is a data flow produced by a process on a DFD. Therefore, the contents of a form or report correspond to the data elements contained in the associated data flow. Further, the data on all forms and reports must consist of data elements in data stores and on the E-R data model for the application or else be computed from these data elements. (In rare instances, data simply go from system input to system output without being stored within the system.) It is common to discover flaws in DFDs and E-R diagrams as you design forms and reports; these diagrams should be updated as designs evolve.

Form
A business document that contains some predefined data and may include some areas where additional data are to be filled in; typically based on one database record.

If you are unfamiliar with computer-based information systems, it will be helpful to clarify exactly what we mean by a form or report. A **form** is a business document containing some predefined data and often includes some areas where additional data are to be filled in. Most forms have a stylized format and are usually not in simple rows and columns. Examples of business forms are product order forms, employment applications, and class registration sheets. Traditionally, forms have been displayed on a paper medium, but today, video display technology allows us to duplicate the layout of almost any printed form, including an organizational logo or any graphic, on a video display terminal. Forms on a video display may be used for data display or data entry. Additional examples of forms are an electronic spreadsheet, computer sign-on or menu, and an automated teller machine (ATM) transaction layout. On the Internet, form interaction is the standard method of gathering and displaying information when consumers order products, request product information, or query account status.

Report
A business document that contains only predefined data; it is a passive document used only for reading or viewing; typically contains data from many unrelated records or transactions.

A **report** is a business document containing only predefined data; it is a passive document used solely for reading or viewing. Examples of reports are invoices, weekly sales summaries by region and salesperson, and a pie chart of population by age categories. We usually think of a report as printed on paper, but it may be printed to a computer file, a visual display screen, or some other medium such as microfilm. Often a report has rows and columns of data, but a report may consist of any format—for example, mailing labels. Frequently, the differences between a form and a report are subtle. A report is only for reading and often contains data about multiple unrelated records in a computer file. On the other hand, a form typically contains data from only one record or is, at least, based on one record, such as data about one customer, one order, or one student. The guidelines for the design of forms and reports are similar.

The Process of Designing Forms and Reports

Designing forms and reports is a user-focused activity that typically follows a prototyping approach (see Figure 1-12 to review the prototyping method). First, you must gain an understanding of the intended user and task objectives during the requirements determination process. During this process, the intended user must answer several questions that attempt to answer the who, what, when, where, and how related to the creation of all forms or reports, as listed in Table 8-1. Gaining an understanding of these questions is a required first step in the creation of any form or report.

TABLE 8-1: Fundamental Questions When Designing Forms and Reports

1. Who will use the form or report?
2. What is the purpose of the form or report?
3. When is the form or report needed and used?
4. Where does the form or report need to be delivered and used?
5. How many people need to use or view the form or report?

Understanding the skills and abilities of the users helps you create an effective design. Are your users experienced computer users or novices? What is their educational level, business background, and task-relevant knowledge? Answers to these questions provide guidance for both the format and the content of your designs. Also, what is the purpose of the form or report? What task will users be performing, and what information is needed to complete this task? Other questions are also important to consider. Where will the users be when performing this task? Will users have access to online systems or will they be in the field? How many people will need to use this form or report? If, for example, a report is being produced for a single user, the design requirements and usability assessment will be relatively simple. A design for a larger audience, however, may need to go through a more extensive requirements collection and usability assessment process.

After collecting the initial requirements, you structure and refine this information into an initial prototype. Structuring and refining the requirements are completed without assistance from the users, although you may occasionally need to contact users to clarify some issue overlooked during analysis. Finally, you ask users to review and evaluate the prototype; then they may accept the design or request that changes be made. If changes are needed, repeat the construction-evaluate-refinement cycle until the design is accepted. Usually, several repetitions of this cycle occur during the design of a single form or report. As with any prototyping process, you should make sure that these iterations occur rapidly in order to gain the greatest benefit from this design approach.

The initial prototype may be constructed in numerous environments, including Visual Basic, Java, or HTML. The obvious choice is to employ standard development tools used within your organization. Often, initial prototypes are simply mock screens that are not working modules or systems. Mock screens can also be produced from a word processor, computer graphics design package, or presentation, software. It is important to remember that the focus of this phase within the SDLC is on the *design*—content and layout. How specific forms or reports are implemented (e.g., the programming language or screen painter code) is left for a later stage. Nonetheless, tools for designing forms and reports are rapidly evolving. In the past, inputs and outputs of all types were typically designed by hand on a coding or layout sheet. For example, Figure 8-2 shows the layout of a data input form using a coding sheet.

Although coding sheets are still used, their importance has diminished because of significant changes in system operating environments and the evolution of automated design tools. Prior to the creation of graphical operating environments, for example, analysts designed many inputs and outputs that were 80 columns (characters) by 25 rows, the standard dimensions for most video displays. These limits in screen dimensions are radically different in graphical operating environments such as Mac OS or Windows where font sizes and screen dimensions can often be changed from user to user. Consequently, the creation of new tools and development environments was needed to help analysts and programmers develop these graphical and flexible designs.

FIGURE 8-2
The layout of a data input form using a coding sheet.

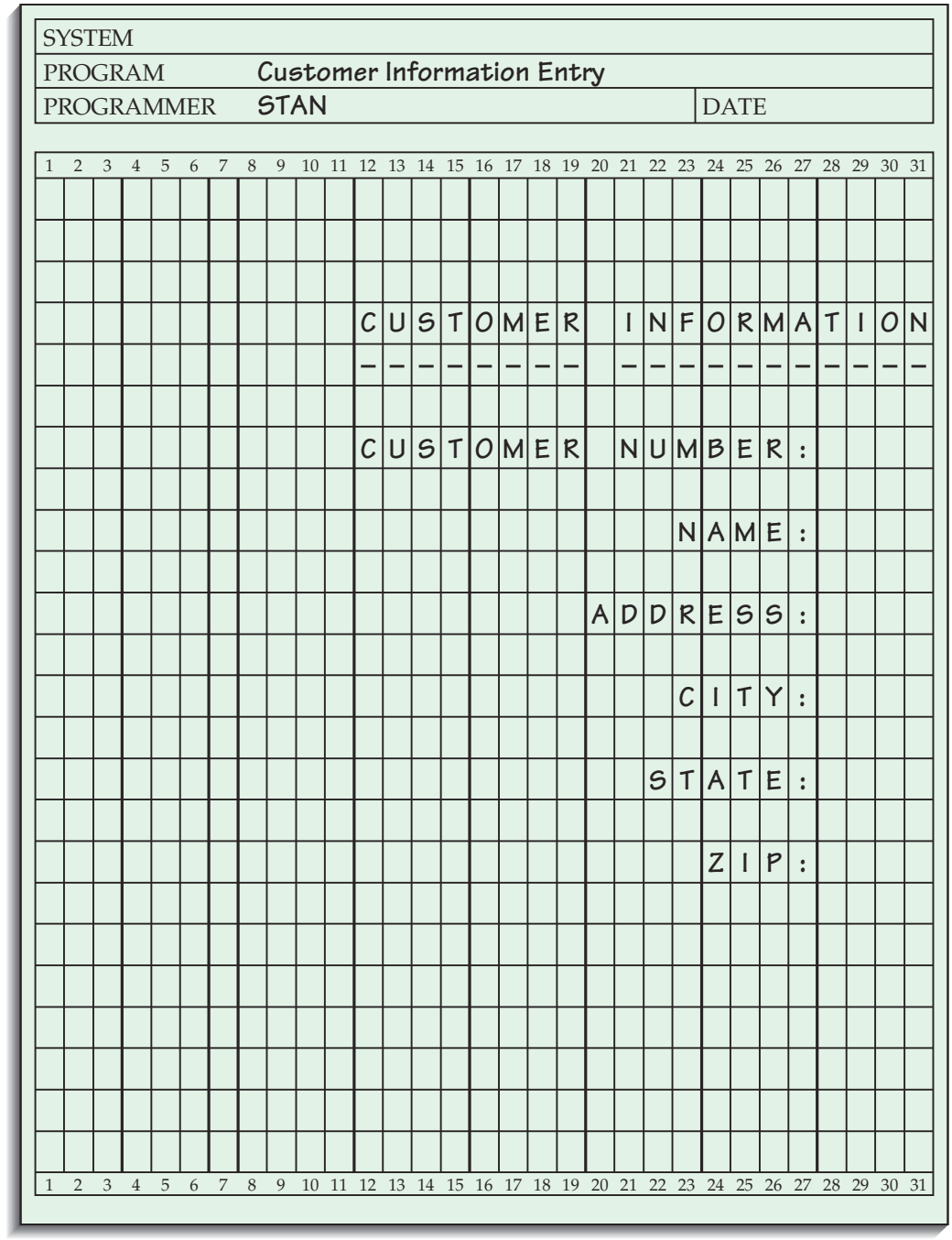


Figure 8-3 shows an example of the same data input form as designed in Microsoft's Visual Basic.Net. Note the variety of fonts, sizes, and highlighting that was used. Online graphical tools for designing forms and reports are rapidly becoming the standard in most professional development organizations.

Deliverables and Outcomes

Each SDLC activity helps you to construct a system. In order to move from phase to phase, each activity produces some type of deliverable that is used in a later activity. For example, within the systems planning and selection phase of the SDLC, the baseline project plan serves as input to many subsequent SDLC activities. In the case of designing forms and reports, design specifications are

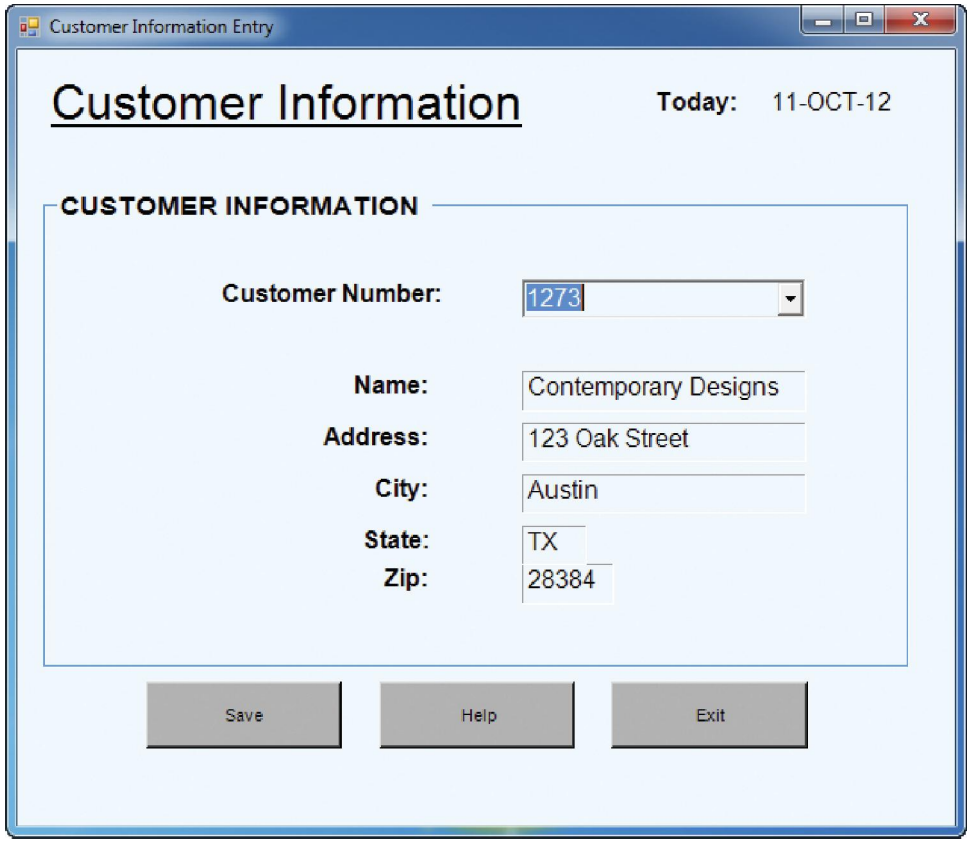


FIGURE 8-3
A data input screen designed in Microsoft's Visual Basic.Net.

the major deliverables and are inputs to the system implementation and operation phase. Design specifications have three sections:

1. Narrative overview
2. Sample design
3. Testing and usability assessment

The narrative overview provides a general overview of the characteristics of the target users, tasks, system, and environmental factors in which the form or report will be used. Its purpose is to explain to those who will actually develop the final form, why this form exists, and how it will be used so that they can make the appropriate implementation decisions. In this section, you list general information and the assumptions that helped shape the design. For example, Figure 8-4 shows an excerpt of a design specification for a Customer Account Status form for Pine Valley Furniture. The first section of the specification, Figure 8-4A, provides a narrative overview containing the information relevant to developing and using the form within PVF. The overview explains the tasks supported by the form, where and when the form is used, characteristics of the people using the form, the technology delivering the form, and other pertinent information. For example, if the form is delivered on a visual display terminal, this section would describe the capabilities of this device, such as navigation and whether it has a touch screen and whether color and a mouse are available.

PINE VALLEY FURNITURE

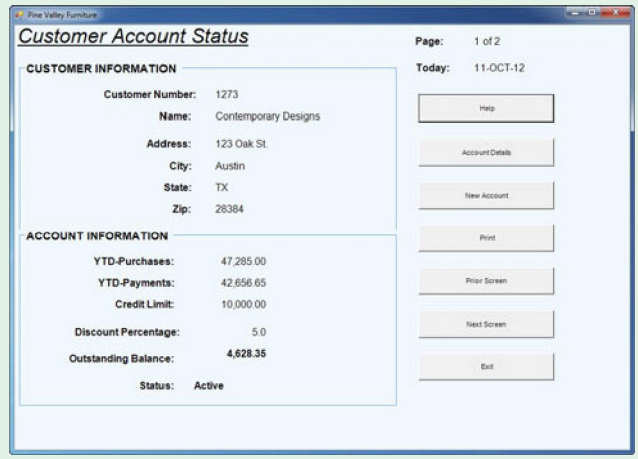
In the second section of the specification, Figure 8-4B, a sample design of the form is shown. This design may be hand-drawn using a coding sheet, although, in most instances, it is developed using standard development tools. Using actual development tools allows the design to be more thoroughly tested and assessed. The final section of the specification, Figure 8-4C, provides all testing and usability assessment information. Some specification information may be irrelevant when designing certain forms and reports. For example, the design of a simple yes/no

FIGURE 8-4
A design specification for a Customer Account Status form for Pine Valley Furniture: (A) The narrative overview containing the information relevant to developing and using the form within PVF, (B) A sample design of the PVF form, (C) Testing and usability assessment information.

(A) Narrative overview

Form:	Customer Account Status
Users:	Customer account representatives within corporate offices
Task:	Assess customer account information: address, account balance, year-to-date purchases and payments, credit limit, discount percentage, and account status.
System:	Microsoft Windows
Environment:	Standard office environment

(B) Sample design



(C) Testing and usability assessment

User-Rated Perceptions (average 14 users):

consistency [1 = consistent to 7 = inconsistent]:	1.52
sufficiency [1 = sufficient to 7 = insufficient]:	1.43
accuracy [1 = accurate to 7 = inaccurate]:	1.67
…	

selection form may be so straightforward that no usability assessment is needed. Also, much of the narrative overview may be unnecessary unless intended to highlight some exception that must be considered during implementation.

Formatting Forms and Reports

A wide variety of information can be provided to users of information systems, ranging from text to video to audio. As technology continues to evolve, a greater variety of data types will be used. A definitive set of rules for delivering every type of information to users has yet to be defined because these rules are continuously evolving along with the rapid changes in technology. Research conducted by computer scientists on human-computer interaction has provided numerous general guidelines for formatting information. Many of these guidelines undoubtedly will apply to the formatting of all evolving information types on yet-to-be-determined devices. Keep in mind that designing usable forms and reports requires your active interaction with users. If this single and fundamental activity occurs, you will likely create effective designs.

For example, the human-computer interface is one of the greatest challenges for designing mobile applications that run on devices such as the iPhone. In

particular, the small video display of these devices presents significant challenges for application designers. Nevertheless, as these and other computing devices evolve and gain popularity, standard guidelines will emerge to make the process of designing interfaces much less challenging.

General Formatting Guidelines Over the past several years, industry and academic researchers have investigated how information formatting influences individual task performance and perceptions of usability. Through this work, several guidelines for formatting information have emerged, as highlighted in Table 8-2. These guidelines reflect some of the general truths of formatting most types of information. The differences between a well-designed form or report and a poorly designed one often will be obvious. For example, Figure 8-5A shows a poorly designed form for viewing a current account balance for a PVF customer. Figure 8-5B is a better design, incorporating several general guidelines from Table 8-2.

The first major difference between the two forms has to do with the title. The title in Figure 8-5A (Customer Information) is ambiguous, whereas the title in Figure 8-5B (Detail Customer Account Information) clearly and specifically describes the contents of the form. The form in Figure 8-5B also includes the date (October 11, 2012) the form was generated so that, if printed, it will be clear to the reader when this occurred. Figure 8-5A displays the account status and customer address, information that is extraneous to viewing the current account balance, which is the intent of the form and provides information that is not in the most useful format for the user. For example, Figure 8-5A provides all customer data, as well as account transactions and a summary of year-to-date purchases and payments. The form does not, however, provide the current outstanding balance of the account, leaving the reader to perform a manual calculation. The layout of information between the two forms also varies in balance and information density. Gaining an understanding of the skills of the

TABLE 8-2: Guidelines for Designing Forms and Reports

Guideline	Description
Use meaningful titles	Clear and specific titles describing content and use of form or report
	Revision date or code to distinguish a form or report from prior versions
	Current date that identifies when the form or report was generated
	Valid date that identifies on what date (or time) the data in the form or report were accurate
Include meaningful information	Only needed information displayed
	Information provided in a usable manner without modification
Balance the layout	Information balanced on the screen or page
	Adequate spacing and margins used
	All data and entry fields clearly labeled
Design an easy navigation system	Clearly show how to move forward and backward
	Clearly show where you are (e.g., page 1 of 3)
	Notify user of the last page of a multipage sequence

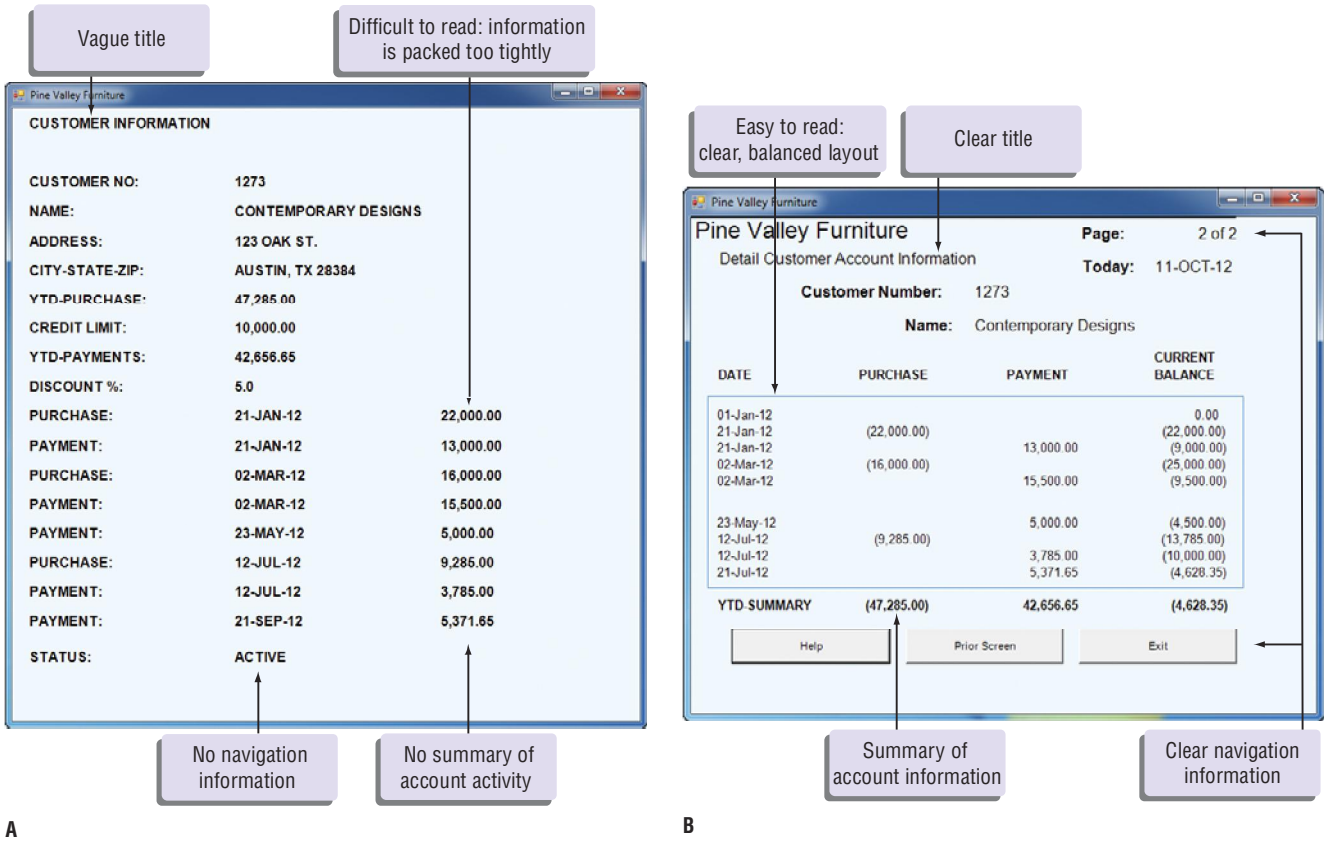


FIGURE 8-5
Contrast of a poorly designed and a well-designed form: (A) A poorly designed form for viewing a current account balance for a PVF customer, (B) A better design that incorporates several general guidelines from Table 8-2.

intended system users and the tasks they will be performing is invaluable when constructing a form or report. By following these general guidelines, your chances of creating effective forms and reports will be enhanced. In the next sections, we discuss specific guidelines for highlighting information, displaying text, and presenting numeric tables and lists.

Highlighting Information As display technologies continue to improve, a greater variety of methods will be available to highlight information. Table 8-3 lists the most commonly used methods for highlighting information. Given this vast array of options, it is important to consider how highlighting can be used to enhance an output without being a distraction. In general, highlighting should be used sparingly to draw the user to or away from certain information and to group together related information. In several situations, highlighting can be a valuable technique for conveying special information:

- Notifying users of errors in data entry or processing
- Providing warnings to users regarding possible problems, such as unusual data values or an unavailable device
- Drawing attention to keywords, commands, high-priority messages, and data that have changed or gone outside normal operating ranges

Highlighting techniques can be used singularly or in tandem, depending upon the level of emphasis desired by the designer. Figure 8-6 shows a form where several types of highlighting are used. In this example, columns clarify different categories of data; capital letters and different fonts distinguish labels from actual data; and bolding is used to draw attention to important data.

TABLE 8-3: Methods of Highlighting

Blinking and audible tones
Color differences
Intensity differences
Size differences
Font differences
Reverse video
Boxing
Underlining
All capital letters
Offsetting the position of nonstandard information

Highlighting should be used conservatively. For example, blinking and audible tones should be used only to highlight critical information requiring the user's immediate response. Once a response is made, these highlights should be turned off. Additionally, highlighting methods should be consistently selected and used based upon the level of importance of the emphasized information. It is also important to examine how a particular highlighting method appears on

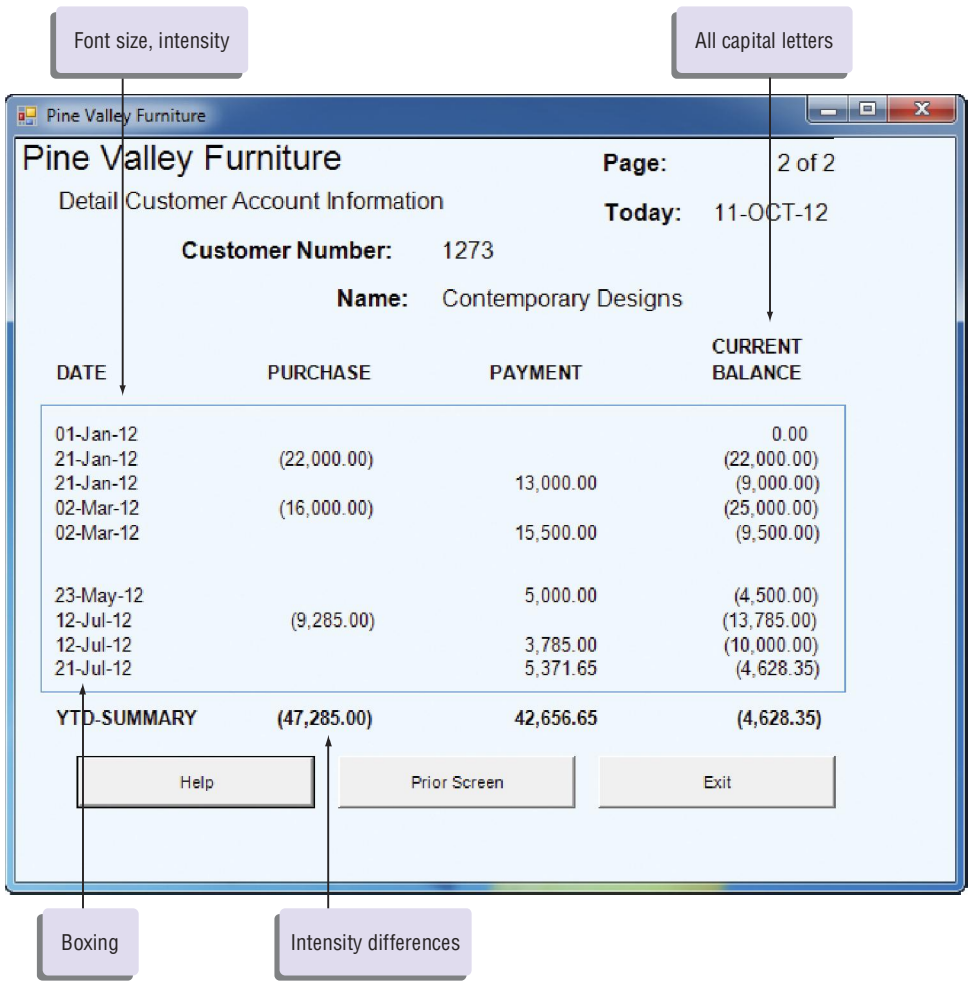


FIGURE 8-6
A form in which several types of highlighting are used.

all possible output devices that could be used with the system. For example, some color combinations may convey appropriate information on one display configuration but wash out and reduce legibility on another.

Recent advances in the development of graphical operating environments such as Windows, Mac OS, or Linux provide designers with some standard highlighting guidelines. However, because these guidelines are continuously evolving, they are often quite vague and leave a great deal of control in the hands of the systems developer. To realize the benefits of using standard graphical operating environments—such as reduced user training time and interoperability among systems—you must be disciplined in how you use highlighting.

Displaying Text In business-related systems, textual output is becoming increasingly important as text-based applications, such as electronic mail, blogs, and information services (e.g., Dow Jones Industrial Average stock index), are more widely used. The display and formatting of system help screens, which often contain lengthy textual descriptions and examples, is one example of textual data that can benefit from following the simple guidelines that have emerged from systems design research. These guidelines appear in Table 8-4. The first one is simple: You should display text using common writing conventions such as mixed upper- and lowercase and appropriate punctuation. For large blocks of text, and if space permits, text should be double spaced. However, if the text is short, or rarely used, it may make sense to use single spacing and place a blank line between each paragraph. You should also left-justify text with a ragged right margin—research shows that a ragged right margin makes it easier to find the next line of text when reading than when text is both left- and right-justified.

When displaying textual information, you should also be careful not to hyphenate words between lines or use obscure abbreviations and acronyms. Users may not know whether the hyphen is a significant character if it is used to continue words across lines. Information and terminology that are not widely understood by the intended users may significantly influence the usability of the system. Thus, you should use abbreviations and acronyms only if they are significantly shorter than the full text and are commonly known by the intended system users. Figure 8-7 shows two versions of a help screen from an application system at PVF. Figure 8-7A shows many violations of the general guidelines for displaying text, whereas Figure 8-7B shows the same information following the general guidelines. Formatting guidelines for the entry of text and alphanumeric data are also very important and will be discussed later in the chapter.

Designing Tables and Lists Unlike textual information, where context and meaning are derived through reading, the context and meaning of tables and lists are derived from the format of the information. Consequently, the usability of information displayed in tables and alphanumeric lists is likely to be much more influenced by effective layout than most other types of information display. As with

TABLE 8-4: Guidelines for Displaying Text

Case	Display text in mixed upper- and lowercase and use conventional punctuation.
Spacing	Use double spacing if space permits. If not, place a blank line between paragraphs.
Justification	Left-justify text and leave a ragged right margin.
Hyphenation	Do not hyphenate words between lines.
Abbreviations	Use abbreviations and acronyms only when they are widely understood by users and are significantly shorter than the full text.

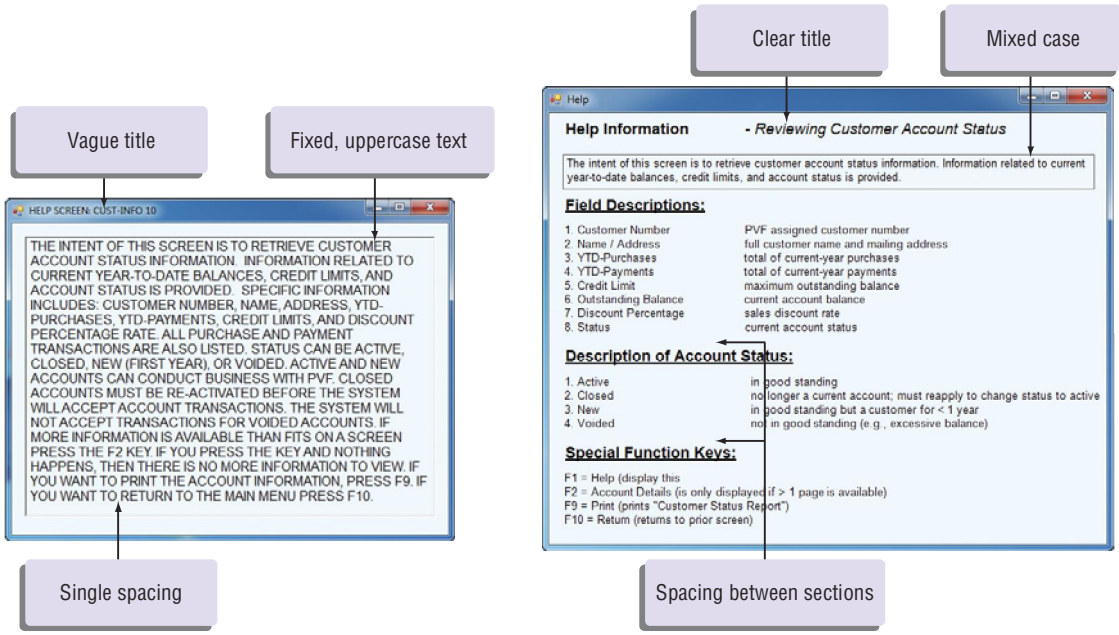


FIGURE 8-7
Contrasting two help screens from an application system at PVF: (A) A poorly designed help screen with many violations of the general guidelines for displaying text, (B) An improved design for a help screen.

the display of textual information, tables and lists can also be greatly enhanced by following a few simple guidelines. These are summarized in Table 8-5.

Figure 8-8 displays two versions of a form design from a Pine Valley Furniture application system that displays customer year-to-date transaction information in a table format. Figure 8-8A displays the information without consideration of the guidelines presented in Table 8-5, and Figure 8-8B (only page 2 of 2 is shown) displays this information after consideration of these guidelines.

One key distinction between these two display forms relates to labeling. The information reported in Figure 8-8B has meaningful labels that stand out more clearly compared to the display in Figure 8-8A. Transactions are sorted by date and transaction type, and numeric data are right-justified and aligned by decimal point in Figure 8-8B, which helps to facilitate scanning. Adequate space is left between columns, and blank lines are inserted after every five rows in Figure 8-8B to help ease the finding and reading of information. Such spacing also provides room for users to annotate data that catch their attention. Using the guidelines presented in Table 8-5 helped create an easy-to-read layout of the information for the user.

Most of the guidelines in Table 8-5 are rather obvious, but this and other tables serve as a quick reference to validate that your form and report designs will be usable. It is beyond our scope here to discuss each of these guidelines, but you should read each carefully and think about why it is appropriate. For example, why should labels be repeated on subsequent screens and pages (the first guideline in Table 8-5)? One explanation is that pages may be separated or copied, and the original labels will no longer be readily accessible to the reader of the data. Why should long alphanumeric data (see the last guideline) be broken into small groups? (If you have a credit card or bank check, look at how your account number is displayed.) Two reasons are that the characters will be easier to remember as you read and type them, and this approach provides a natural and consistent place to pause when you speak them over the phone (e.g., when you are placing a phone order for products in a catalog).

TABLE 8-5: General Guidelines for Displaying Tables and Lists

Guideline	Description
Use meaningful labels	All columns and rows should have meaningful labels.
	Labels should be separated from other information by using highlighting.
	Redisplay labels when the data extend beyond a single screen or page.
Format columns, rows, and text	Sort in a meaningful order (e.g., ascending, descending, or alphabetical).
	Place a blank line between every five rows in long columns.
	Similar information displayed in multiple columns should be sorted vertically (i.e., read from top to bottom, not left to right).
	Columns should have at least two spaces between them.
	Allow white space on printed reports for user to write notes.
	Use a single typeface, except for emphasis.
	Use same family of typefaces within and across displays and reports.
	Avoid overly fancy fonts.
Format numeric, textual, and alphanumeric data	Right-justify *numeric data* and align columns by decimal points or other delimiter.
	Left-justify *textual data*. Use short line length, usually 30 to 40 characters per line (this guideline is what newspapers use, and it is easy to speed-read).
	Break long sequences of *alphanumeric data* into small groups of three to four characters each.

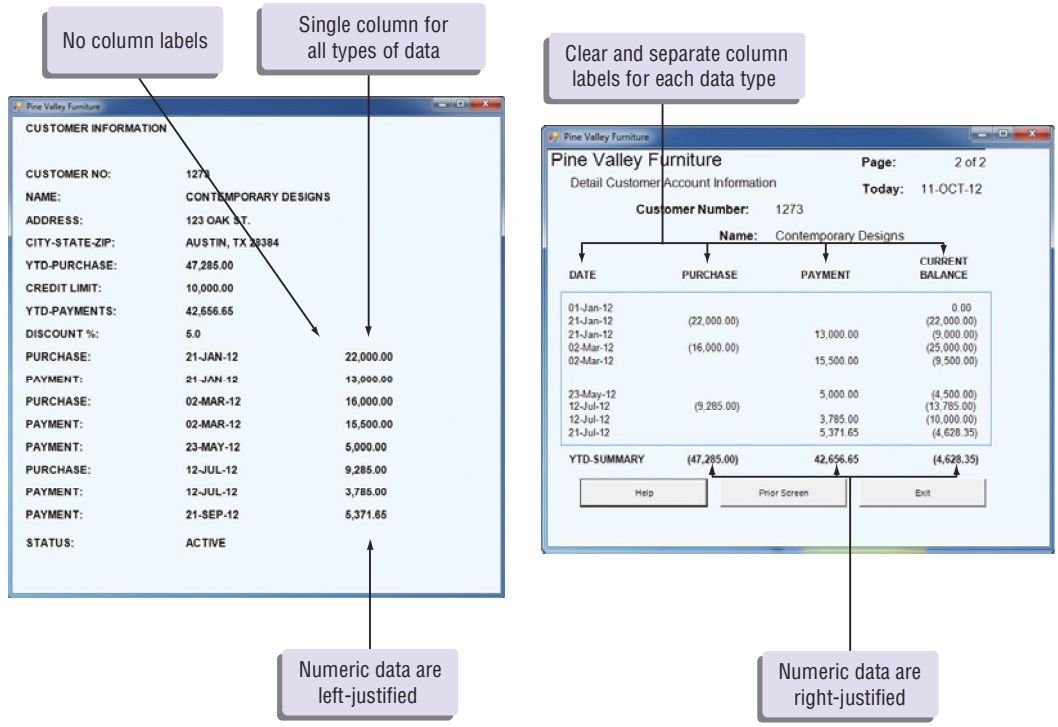


FIGURE 8-8
Contrasting two Pine Valley Furniture forms: (A) A poorly designed form, (B) An improved design form.

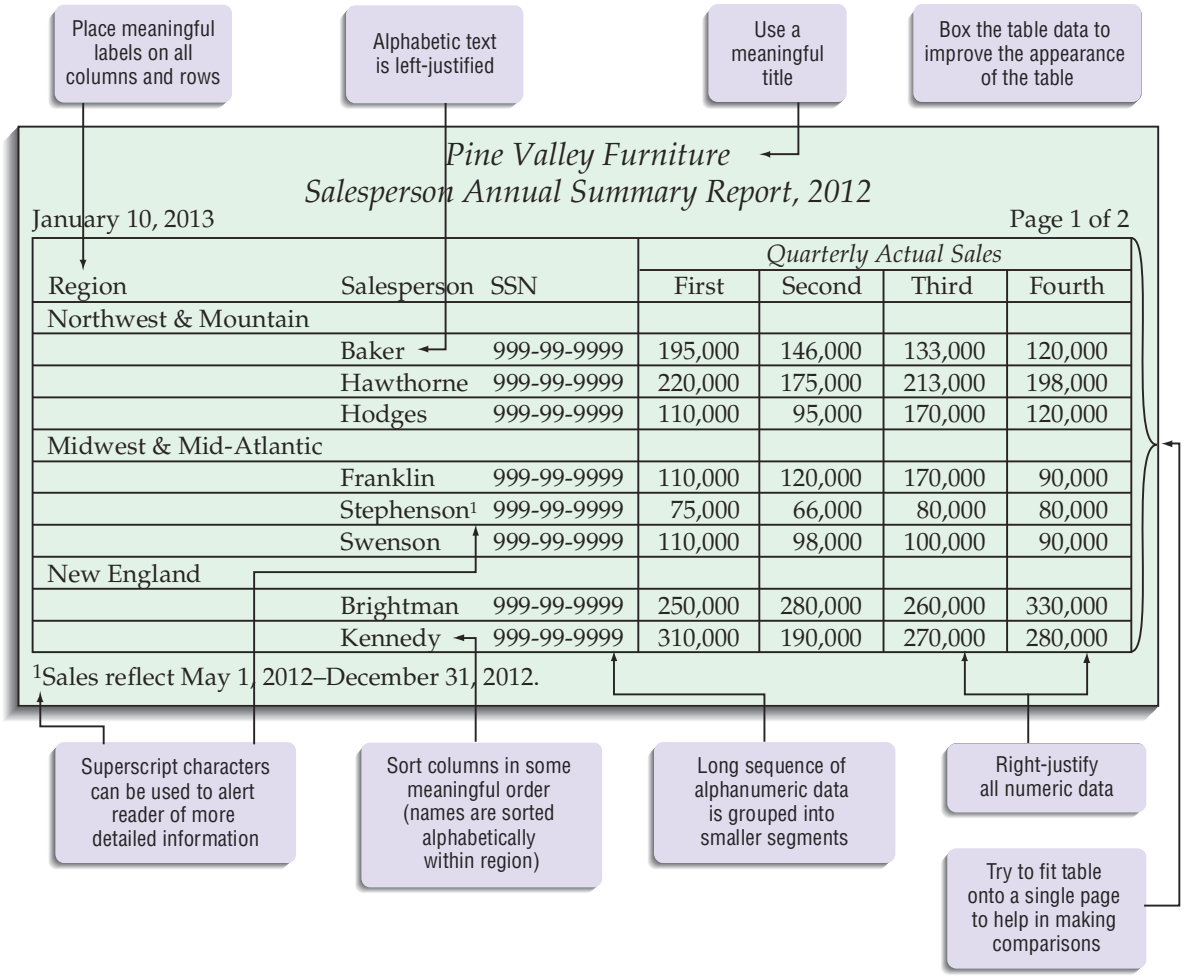


FIGURE 8-9
Tabular report illustrating good report design guidelines.

When you design the display of numeric information, you must determine whether a table or a graph should be used. In general, tables are best when the user's task involves finding an individual data value from a larger data set, whereas line and bar graphs are more appropriate for analyzing data changes over time. For example, if the marketing manager for Pine Valley Furniture needed to review the actual sales of a particular salesperson for a particular quarter, a tabular report such as the one shown in Figure 8-9 would be most useful. This report has been annotated to emphasize good report design practices. The report has both a printed date as well as a clear indication, as part of the report title, of the period over which the data apply. Sufficient white space also provides some room for users to add personal comments and observations. Often, to provide such white space, a report must be printed in landscape, rather than portrait, orientation. Alternatively, if the marketing manager wished to compare the overall sales performance of each sales region, a line or bar graph would be more appropriate, as illustrated in Figure 8-10.

Paper versus Electronic Reports When a report is produced on paper rather than on a computer display, you need to consider some additional things. For example, laser printers (especially color laser printers) and ink-jet printers allow you to produce a report that looks exactly as it does on the display screen. Thus, when using these types of printers, you can follow our general design guidelines to create a report with high usability. However, other types of

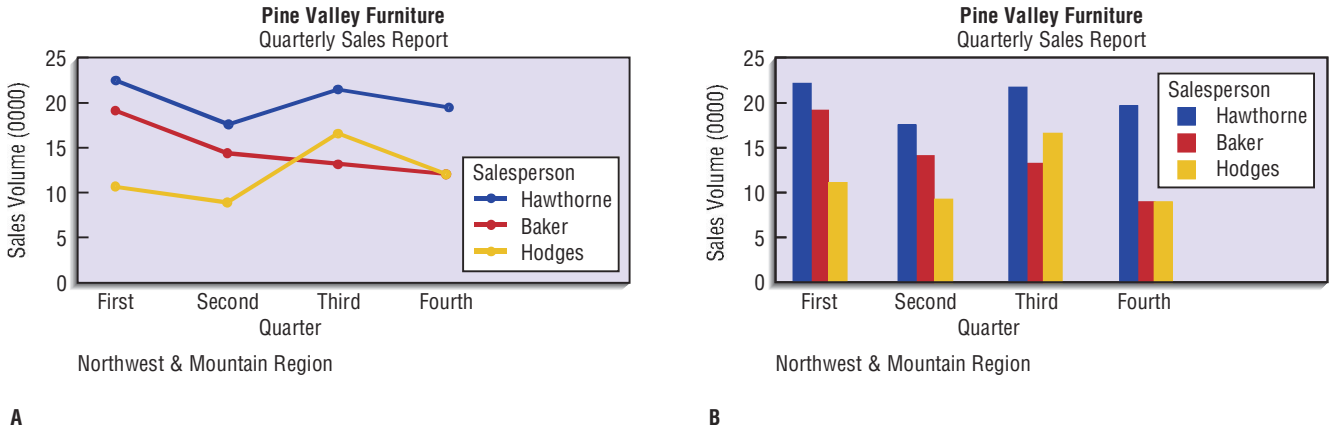


FIGURE 8-10
Graphs showing quarterly sales at Pine Valley Furniture: (A) Line graph, (B) Bar graph.

printers cannot closely reproduce the display screen image onto paper. For example, many business reports are produced using high-speed impact printers that produce characters and a limited range of graphics by printing a fine pattern of dots. The advantages of impact printers are that they are fast, reliable, and relatively inexpensive. Their drawbacks are that they have a limited ability to produce graphics and have a somewhat lower print quality. In other words, they are good at rapidly producing reports that contain primarily alphanumeric information but cannot exactly replicate a screen report onto paper. For this reason, impact printers are mostly used for producing large batches of reports, such as a batch of phone bills for your telephone company, on a wide range of paper widths and types. When designing reports for impact printers, you use a coding sheet similar to the one displayed in Figure 8-2, although coding sheets for designing printer reports typically can have up to 132 columns. Like the process for designing all forms and reports, you follow a prototyping process and carefully control the spacing of characters in order to produce a high-quality report. However, unlike other form and report designs, you may be limited in the range of formatting, text types, and highlighting options. Nonetheless, you can easily produce a highly usable report of any type if you carefully and creatively use the available formatting options.

Designing Interfaces and Dialogues

Interface and dialogue design focuses on how information is provided to and captured from users. Dialogues are analogous to a conversation between two people. The grammatical rules followed by each person during a conversation are analogous to the human-computer interface. The design of interfaces and dialogues involves defining the manner in which humans and computers exchange information. A good human-computer interface provides a uniform structure for finding, viewing, and invoking the different components of a system. In this section we describe how to design interfaces and dialogues.

The Process of Designing Interfaces and Dialogues

Similar to designing forms and reports, the process of designing interfaces and dialogues is a user-focused activity. You follow a prototyping methodology of iteratively collecting information, constructing a prototype, assessing usability, and making refinements. To design usable interfaces and dialogues, you must answer the same who, what, when, where, and how questions used to guide the

Design Specification

1. Narrative Overview
   a. Interface/Dialogue Name
   b. User Characteristics
   c. Task Characteristics
   d. System Characteristics
   e. Environmental Characteristics

2. Interface/Dialogue Designs
   a. Form/Report Designs
   b. Dialogue Sequence Diagram(s) and Narrative Description

3. Testing and Usability Assessment
   a. Testing Objectives
   b. Testing Procedures
   c. Testing Results
      i) Time to Learn
      ii) Speed of Performance
      iii) Rate of Errors
      iv) Retention over Time
      v) User Satisfaction and Other Perceptions

FIGURE 8-11
An outline for a design specification for interfaces and dialogues.

design of forms and reports (see Table 8-1). Thus, this process parallels that of designing forms and reports.

Deliverables and Outcomes

The deliverable and outcome from system interface and dialogue design is the creation of a design specification. This specification is similar to the specification produced for form and report designs—with one exception. Recall that the design specification for forms and reports had three sections (see Figure 8-4):

1. Narrative overview
2. Sample design
3. Testing and usability assessment

For interface and dialogue designs, one additional subsection is included: a section outlining the dialogue sequence—the ways a user can move from one display to another. Later in the chapter you will learn how to design a dialogue sequence by using dialogue diagramming. An outline for a design specification for interfaces and dialogues is shown in Figure 8-11.

Designing Interfaces

In this section we discuss the design of interface layouts. This discussion provides guidelines for structuring and controlling data-entry fields, providing feedback, and designing online help. Effective interface design requires you to gain a thorough understanding of each of these concepts.

Designing Layouts To ease user training and data recording, use standard formats for computer-based forms and reports similar to paper-based forms and reports for recording or reporting information. A typical paper-based form for

PINE VALLEY FURNITURE

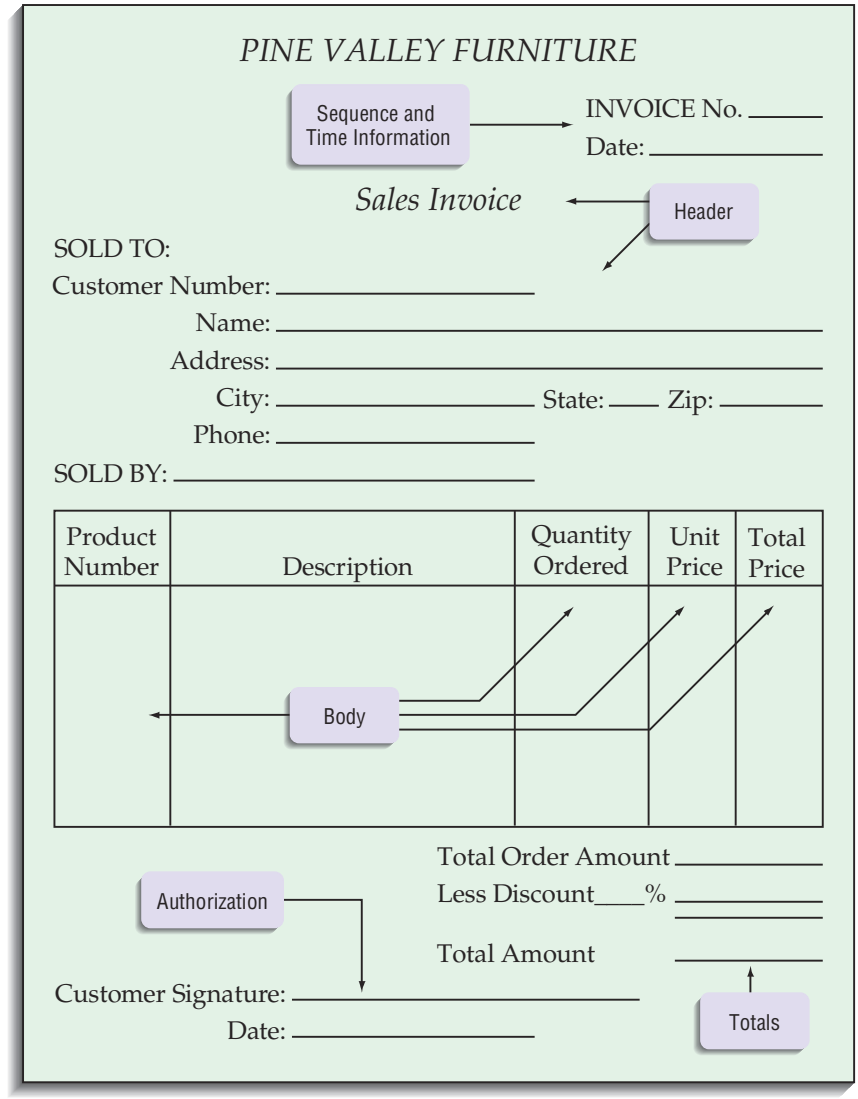


FIGURE 8-12
Paper-based form for reporting customer sales activity at Pine Valley Furniture.

reporting customer sales activity is shown in Figure 8-12. This form has several general areas common to most forms:

- Header information
- Sequence and time-related information
- Instruction or formatting information
- Body or data details
- Totals or data summary
- Authorization or signatures
- Comments

In many organizations, data are often first recorded on paper-based forms and then later recorded within application systems. When designing layouts to record or display information on paper-based forms, try to make both as similar as possible. Additionally, data-entry displays should be consistently formatted across applications to speed data entry and reduce errors. Figure 8-13 shows an equivalent computer-based form to the paper-based form shown in Figure 8-12.

The design of between-field navigation is another item to consider when designing the layout of computer-based forms. Because you can control the

FIGURE 8-13
Computer-based form for reporting customer sales activity at Pine Valley Furniture.

sequence for users to move between fields, standard screen navigation should flow from left-to-right and top-to-bottom just as when you work on paper-based forms. For example, Figure 8-14 contrasts the flow between fields on a form used to record business contacts. Figure 8-14A uses a consistent left-to-right, top-to-bottom flow. Figure 8-14B uses a flow that is nonintuitive. When appropriate, you should also group data fields into logical categories with labels describing the contents of the category. Areas of the screen not used for data entry or commands should be inaccessible to the user.

When designing the navigation procedures within your system, flexibility and consistency are primary concerns. Users should be able to move freely forward and backward or to any desired data-entry fields. Users should be able to navigate each form in the same way or in as similar a manner as possible. Additionally, data should not usually be permanently saved by the system until the user makes an explicit request to do so. This option allows the user to abandon a data-entry screen, back up, or move forward without adversely impacting the contents of the permanent data.

Consistency extends to the selection of keys and commands. Assign each key or command only one function. This assignment should be consistent throughout the entire system and across systems, if possible. Depending upon the application, various types of functional capabilities will be required to provide smooth navigation and data entry. Table 8-6 provides a checklist for testing

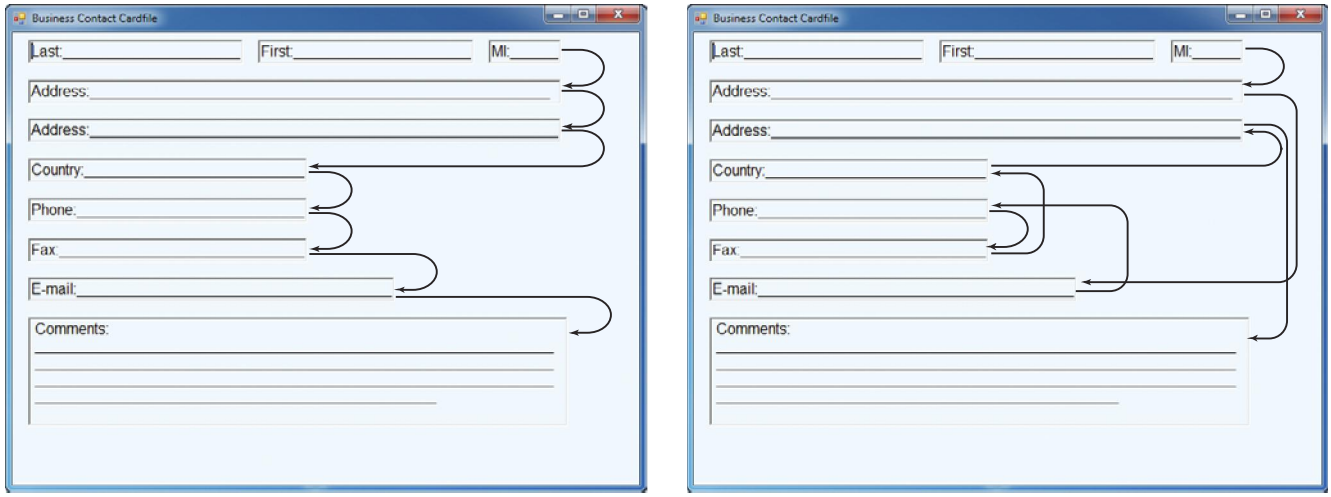


FIGURE 8-14
Contrasting the navigation flow within a data-entry form: (A) Proper flow between data-entry fields with a consistent left-to-right, top-to-bottom flow, (B) Poor flow between data-entry fields with inconsistent flow.

the functional capabilities for providing smooth and easy navigation within a form. For example, a good interface design provides a consistent way for moving the cursor to different places on the form, editing characters and fields, moving among form displays, and obtaining help. These functions may be provided by keystrokes, mouse, menu, or function keys. It is possible that, for a single application, not all capabilities listed in Table 8-6 may be needed in order

TABLE 8-6: Checklist for Validating the Usability of User Interface

Cursor-Control Capabilities
Move the cursor forward to the next data field.
Move the cursor backward to the previous data field.
Move the cursor to the first, last, or some other designated data field.
Move the cursor forward one character in a field.
Move the cursor backward one character in a field.
Editing Capabilities
Delete the character to the left of the cursor.
Delete the character under the cursor.
Delete the whole field.
Delete data from the whole form (empty the form).
Exit Capabilities
Transmit the screen to the application program.
Move to another screen/form.
Confirm the saving of edits or go to another screen/form.
Help Capabilities
Get help on a data field.
Get help on a full screen/form.

Source: Based on J. S. Dumas (1988). *Designing User Interfaces for Software*. Upper Saddle River, NJ: Prentice Hall.

to create a good user interface. Yet, the capabilities that are used should be consistently applied to provide an optimal user environment. Table 8-6 provides you with a checklist for validating the usability of user interface designs.

Structuring Data Entry You should consider several guidelines when structuring data-entry fields on a form. These guidelines are listed in Table 8-7. The first is simple, yet, is often violated by designers. To minimize data-entry errors and user frustration, never require the user to enter information that is already available within the system or information that can be easily computed by the system. For example, never require the user to enter the current date and time, because each of these values can be easily retrieved from the computer system's internal calendar and clock. By allowing the system to do these tasks, the user simply confirms that the calendar and clock are working properly.

Other guidelines are equally important. For example, suppose that a bank customer is repaying a loan on a fixed schedule with equal monthly payments. Each month when a payment is sent to the bank, a clerk needs to record that the payment has been received into a loan-processing system. Within such a system, default values for fields should be provided whenever appropriate, which allows the clerk to enter specific data into the system only when the customer pays more or less than the scheduled amount. In all other cases, the clerk simply verifies that the check is for the default amount provided by the system and presses a single key to confirm the receipt of payment.

When entering data, do not require the user to specify the dimensional units of a particular value, for example, whether an amount is in dollars or a weight is in tons. Use field formatting and the data-entry prompt to make clear the type of data being requested. In other words, place a caption describing the data to be entered adjacent to each data field so that the user knows what type of data is being requested. As with the display of information, all data entered onto a form should automatically justify in a standard format (e.g., date, time, money).

TABLE 8-7: Guidelines for Structuring Data-Entry Fields

Entry	Never request data that are already online or that can be computed, for example, do not request customer data on an order form if that data can be retrieved from the database, and do not request extended prices that can be computed from quantity sold and unit prices.
Defaults	Always provide default values when appropriate, for example, assume today's date for a new sales invoice, or use the standard product price unless overridden.
Units	Make clear the type of data units requested for entry, for example, indicate quantity in tons, dozens, pounds, etc.
Replacement	Use character replacement when appropriate, for example, allow the user to look up the value in a table or automatically fill in the value once the user enters enough significant characters.
Captioning	Always place a caption adjacent to fields; see Table 8-8 for caption options.
Format	Provide formatting examples when appropriate, for example, automatically show standard embedded symbols, decimal points, credit symbols, or dollar signs.
Justify	Automatically justify data entries; numbers should be right-justified and aligned on decimal points, and text should be left-justified.
Help	Provide context-sensitive help when appropriate, for example, provide a hot key, such as the F1 key, that opens the help system on an entry that is most closely related to where the cursor is on the display.

TABLE 8-8: Display Design Options for Entering Text

Options	Example
Line caption	Phone Number ( ) -
Drop caption	( ) - Phone Number
Boxed caption	Phone Number
Delimited characters	|(| | |)| | | -| | | | | Phone Number
Check-off boxes	Method of payment (check one) ❑ Check ❑ Cash ❑ Credit card: Type

Table 8-8 illustrates display design options for printed forms. For data entry on video display terminals, highlight the area in which text is entered so that the exact number of characters per line and number of lines are clearly shown. You can also use check-off boxes or radio buttons to allow users to choose standard textual responses. Use data-entry controls to ensure that the proper type of data (alphabetic or numeric, as required) is entered. Data-entry controls are discussed next.

Controlling Data Input One objective of interface design is to reduce data-entry errors. As data are entered into an information system, steps must be taken to ensure that the input is valid. As a systems analyst, you must anticipate the types of errors users may make and design features into the system's interfaces to avoid, detect, and correct data-entry mistakes. Several types of data errors are summarized in Table 8-9. Data errors can occur from appending extra data onto a field, truncating characters off a field, transcripting the wrong characters into a field, or transposing one or more characters within a field. Systems designers have developed numerous tests and techniques for detecting invalid data before saving or transmission, thus improving the likelihood that data will be valid. Table 8-10 summarizes these techniques. These tests and techniques are often incorporated into both data-entry screens and when data are transferred from one computer to another.

Correcting erroneous data is much easier to accomplish before it is permanently stored in a system. Online systems can notify a user of input problems as data are being entered. When data are processed online as events occur, it is much less likely that data-validity errors will occur and not be caught. In an online system, most problems can be easily identified and resolved before permanently saving data to a storage device using many of the techniques

TABLE 8-9: Types of Data Errors

Data Error	Description
Appending	Adding additional characters to a field
Truncating	Losing characters from a field
Transcripting	Entering invalid data into a field
Transposing	Reversing the sequence of one or more characters in a field

TABLE 8-10: Techniques Used by Systems Designers to Detect Data Errors before Saving or Transmission

Validation Test	Description
Class or composition	Test to ensure that data are of proper type (e.g., all numeric, all alphabetic, alphanumeric)
Combinations	Test to see that value combinations of two or more data fields are appropriate or make sense (e.g., does the quantity sold make sense given the type of product?)
Expected values	Test to see whether data are what is expected (e.g., match with existing customer names, payment amount, etc.)
Missing data	Test for existence of data items in all fields of a record (e.g., is there a quantity field on each line item of a customer order?)
Pictures/templates	Test to ensure that data conform to a standard format (e.g., are hyphens in the right places for a student ID number?)
Range	Test to ensure data are within a proper range of values (e.g., is a student's grade-point average between 0 and 4.0?)
Reasonableness	Test to ensure data are reasonable for situation (e.g., pay rate for a specific type of employee)
Self-checking digits	Technique by which extra digits, derived using a standard formula (see Figure 8-15), are added to a numeric field before transmission and checked after transmission
Size	Test for too few or too many characters (e.g., is social security number exactly nine digits?)
Values	Test to make sure values come from a set of standard values (e.g., two-letter state codes)

described in Table 8-10. However, in systems where data inputs are stored and entered (or transferred) in batches, the identification and notification of errors are more difficult. Batch processing systems can, however, reject invalid inputs and store them in a log file for later resolution.

Most of the straightforward tests and techniques shown in Table 8-10 are widely used. Some can be handled by data-management technologies, such as a database management system (DBMS), to ensure that they are applied for all data-maintenance operations. If a DBMS cannot perform these tests, then you must design the tests into program modules. Self-checking digits, shown in Figure 8-15, is an example of a sophisticated program. The figure provides a description and an outline of how to apply the technique. A short example then shows how a check digit is added to a field before data entry or transfer. Once entered or transferred, the check digit algorithm is again applied to the field to "check" whether the check digit received obeys the calculation. If it does, it is likely (but not guaranteed, because two different values could yield the same check digit) that no data transmission or entry error occurred. If not equal, then some type of error occurred.

In addition to validating the data values entered into a system, controls must be established to verify that all input records are correctly entered and processed only once. A common method used to enhance the validity of entering batches of data records is to create an **audit trail** of the entire sequence of data entry, processing, and storage. In such an audit trail, the actual sequence, count, time, source location, and human operator are recorded in a separate transaction log in the event of a data input or processing error. If an error occurs, corrections can be made by reviewing the contents of the log. Detailed

Audit trail
A record of the sequence of data entries and the date of those entries.

FIGURE 8-15
How check digits are calculated.

Description	Techniques where extra digits are added to a field to assist in verifying its accuracy
Method	1. Multiply each digit of a numeric field by weighting factor (e.g., 1,2,1,2, . . .). 2. Sum the results of weighted digits. 3. Divide sum by modulus number (e.g., 10). 4. Subtract remainder of division from modulus number to determine check digit. 5. Append check digits to field.
Example	Assume a numeric part number of: 12473 1–2. Multiply each digit of part number by weighting factor from right to left and sum the results of weighted digits: 1 2 4 7 3 ×1 ×2 ×1 ×2 ×1 1 + 4 + 4 + 14 + 3 = 26 3. Divide sum by modulus number. 26/10 = 2 remainder 6 4. Subtract remainder from modulus number to determine check digit. check digit = 10 – 6 = 4 5. Append check digits to field. Field value with appended check digit = 124734

logs of data inputs not only are useful for resolving batch data-entry errors and system audits, but also serve as a powerful method for performing backup and recovery operations in the case of a catastrophic system failure.

Providing Feedback When you talk with friends, you expect them to give you feedback by nodding and replying to your questions and comments. Without feedback, you would be concerned that they were not listening. Similarly, when designing system interfaces, providing appropriate feedback is an easy way to make a user's interaction more enjoyable; not providing feedback is a sure way to frustrate and confuse. System feedback can consist of three types:

1. Status information
2. Prompting cues
3. Error and warning messages

1. *Status Information.* Providing status information is a simple technique for keeping users informed of what is going on within a system. For example, relevant status information, such as displaying the current customer name or time, placing appropriate titles on a menu or screen, and identifying the number of screens following the current one (e.g., Screen 1 of 3), all provide needed feedback to the user. Providing status information during processing operations is especially important if the operation takes longer than a second or two. For example, when opening a file, you might display, "Please wait while I open the file," or when performing a large calculation, flash the message "Working . . ." to the user. Further, it is important to tell the user that besides working, the system has accepted the user's input and the input was in the correct form. Sometimes it is important to give the user a chance to obtain more feedback. For example, a function key could toggle between showing

a "Working . . ." message and giving more specific information as each intermediate step is accomplished. Providing status information reassures users that nothing is wrong and makes them feel in command of the system, not vice versa.

2. *Prompting Cues.* A second feedback method is to display prompting cues. When prompting the user for information or action, it is useful to be specific in your request. For example, suppose a system prompted users with the following request:

   READY FOR INPUT: _____

   With such a prompt, the designer assumes that the user knows exactly what to enter. A better design would be specific in its request, possibly providing an example, default values, or formatting information. An improved prompting request might be as follows:

   Enter the customer account number (123-456-7):____-___-_

3. *Error and Warning Messages.* A final method available to you for providing system feedback is using error and warning messages. Following a few simple guidelines can greatly improve the usefulness of these messages. First, make messages specific and free of error codes and jargon. Additionally, messages should never scold the user but attempt to guide the user toward a resolution. For example, a message might say, "No customer record found for that customer ID. Please verify that digits were not transposed." Messages should be in user, not computer, terms. Terms such as *end of file*, *disk I/O error*, or *write protected* may be too technical and not helpful for many users. Multiple messages can be useful so that a user can get more detailed explanations if wanted or needed. Also, make sure error messages appear in roughly the same format and placement each time so that they are recognized as error messages and not as some other information. Examples of bad and good messages are provided in Table 8-11. Use these guidelines to provide useful feedback in your designs. A special type of feedback is answering help requests from users. This important topic is described next.

Providing Help Designing a help system is one of the most important interface design issues you will face. When designing help, you need to put yourself in the user's place. When accessing help, the user likely does not know what to do next, does not understand what is being requested, or does not know how the requested information needs to be formatted. A user requesting help is much like a ship in distress, sending an SOS. In Table 8-12, we provide our SOS guidelines for the design of system help: Simplify, Organize, and Show. Our first

TABLE 8-11: Examples of Poor and Improved Error Messages

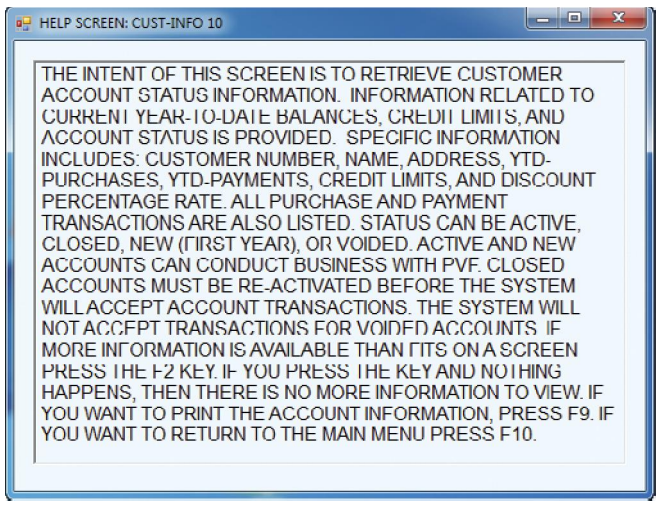
Poor Error Messages	Improved Error Messages
ERROR 56 OPENING FILE	The file name you typed was not found. Press F2 to list valid file names.
WRONG CHOICE	Please enter an option from the menu.
DATA ENTRY ERROR	The prior entry contains a value outside the range of acceptable values. Press F9 for list of acceptable values.
FILE CREATION ERROR	The file name you entered already exists. Press F10 if you want to overwrite it. Press F2 if you want to save it with a new name.

TABLE 8-12: Guidelines for Designing System Help

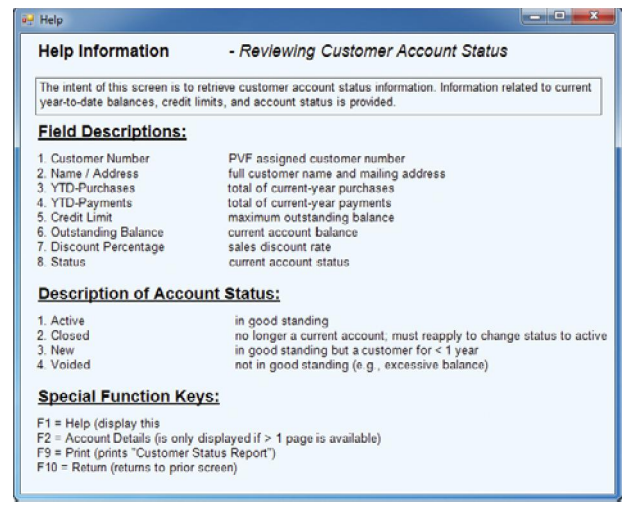
Guideline	Explanation
Simplify	Use short, simple wording, common spelling, and complete sentences. Give users only what they need to know, with ability to find additional information.
Organize	Use lists to break information into manageable pieces.
Show	Provide examples of proper use and the outcomes of such use.

guideline, *simplify*, suggests that help messages should be short, to the point, and use words that users can understand. The second guideline, *organize*, means that the information in help messages should be easy for users to absorb. Long paragraphs of text are often difficult for people to understand. A better design organizes lengthy information in a manner easier for users to digest through the use of bulleted and ordered lists. Finally, it is often useful to explicitly *show* users how to perform an operation and the outcomes of procedural steps. Figure 8-16 contrasts the designs of two help screens, one that employs our guidelines and one that does not.

Many commercially available systems provide extensive system help. For example, Table 8-13 lists the range of help available in a popular electronic spreadsheet. Many systems are also designed so that users can vary the level of detail provided. Help may be provided at the system level, screen or form level, and individual field level. The ability to provide field-level help is often referred to as *context-sensitive* help. For some applications, providing context-sensitive help for all system options is a tremendous undertaking that is virtually a project in itself. If you do decide to design an extensive help system with many levels of detail, you must be sure that you know exactly what the user needs help with, or your efforts may confuse users more than help them. After leaving a help screen, users should always return back to where they were prior to



A



B

FIGURE 8-16
Contrasting help screens: (A) A poorly designed help screen, (B) An improved design for a help screen.

TABLE 8-13: Types of Help

Type of Help	Example of Question
Help on help	How do I get help?
Help on concepts	What is a customer record?
Help on procedures	How do I update a record?
Help on messages	What does "Invalid File Name" mean?
Help on menus	What does "Graphics" mean?
Help on function keys	What does each function key do?
Help on commands	How do I use the "Cut" and "Paste" commands?
Help on words	What do "merge" and "sort" mean?

requesting help. If you follow these simple guidelines, you will likely design a highly usable help system.

As with the construction of menus, many programming environments provide powerful tools for designing system help. For example, Microsoft's HTML Help SDK allows you to construct hypertext-based help systems quickly. In this environment, you use a text editor to construct help pages that can be easily linked to other pages containing related or more specific information. Linkages are created by embedding special characters into the text document that make words hypertext buttons—that is, direct linkages—to additional information. The HTML Help SDK transforms the text document into a hypertext document. For example, Figure 8-17 shows a hypertext-based help screen from Microsoft.

FIGURE 8-17
Hypertext-based help system from Microsoft.

Hypertext-based help systems have become the standard environment for most commercial operating environments for two primary reasons. First, standardizing system help across applications eases user training. Second, hypertext allows users to selectively access the level of help they need, making it easier to provide effective help for both novice and experienced users within the same system.

Designing Dialogues

Dialogue
The sequence of interaction between a user and a system.

The process of designing the overall sequences that users follow to interact with an information system is called *dialogue design.* A **dialogue** is the sequence in which information is displayed to and obtained from a user. As with other design processes, designing dialogues is a three-step process:

1. Designing the dialogue sequence
2. Building a prototype
3. Assessing usability

The primary design guideline for designing dialogues is consistency; dialogues need to be consistent in sequence of actions, keystrokes, and terminology. In other words, use the same labels for the same operations on all screens and the same location of the same information on all displays.

One example of these guidelines concerns removing data from a database or file (see the Reversal entry in Table 8-14). It is good practice to display the information that will be deleted before making a permanent change to the file. For

TABLE 8-14: Guidelines for the Design of Human-Computer Dialogues

Guideline	Explanation
Consistency	Dialogues should be consistent in sequence of actions, keystrokes, and terminology (e.g., use the same labels for the same operations on all screens and the same location of the same information on all displays).
Shortcuts and sequence	Allow advanced users to take shortcuts using special keys (e.g., CTRL-C to copy highlighted text). A natural sequence of steps should be followed (e.g., enter first name before last name, if appropriate).
Feedback	Feedback should be provided for every user action (e.g., confirm that a record has been added, rather than simply putting another blank form on the screen).
Closure	Dialogues should be logically grouped and have a beginning, middle, and end (e.g., the last in the sequence of screens should indicate that there are no more screens).
Error handling	All errors should be detected and reported; suggestions on how to proceed should be made (e.g., suggest why such errors occur and what the user can do to correct the error). Synonyms for certain responses should be accepted (e.g., accept either "t," "T," or "TRUE").
Reversal	Dialogues should, when possible, allow the user to reverse actions (e.g., undo a deletion); data should not be deleted without confirmation (e.g., display all the data for a record the user has indicated is to be deleted).
Control	Dialogues should make the user (especially an experienced user) feel in control of the system (e.g., provide a consistent response time at a pace acceptable to the user).
Ease	Dialogues should provide simple means for users to enter information and navigate between screens (e.g., provide means to move forward, backward, and to specific screens, such as first and last).

Source: Based on B. Shneiderman, C. Plaisant, M. Cohen, and S. Jacobs (2009). *Designing the User Interface: Strategies for Effective Human-Computer Interaction, 5th Edition.* Reading, MA: Addison-Wesley.

example, if the customer service representative wanted to remove a customer from the database, the system should ask only for the customer ID in order to retrieve the correct customer account. Once found, and before allowing the confirmation of the deletion, the system should display the account information. For actions making permanent changes to system data files and when the action is not commonly performed, many system designers use the double-confirmation technique by which the users must confirm their intention twice before being allowed to proceed.

Designing the Dialogue Sequence Your first step in dialogue design is to define the sequence. In other words, you must have a clear understanding of the user, task, technological, and environmental characteristics when designing dialogues. Suppose that the marketing manager at Pine Valley Furniture (PVF) wants sales and marketing personnel to be able to review the year-to-date transaction activity for any PVF customer. After talking with the manager, you both agree that a typical dialogue between a user and the Customer Information System for obtaining this information might proceed as follows:

PINE VALLEY FURNITURE

1. Request to view individual customer information
2. Specify the customer of interest
3. Select the year-to-date transaction summary display
4. Review customer information
5. Leave system

As a designer, once you understand how a user wishes to use a system, you can then transform these activities into a formal dialogue specification.

A method for designing and representing dialogues is **dialogue diagramming.** Dialogue diagrams, illustrated in Figure 8-18, have only one symbol, a box with three sections; each box represents one display (which might be a full screen or a specific form or window) within a dialogue. The three sections of the box are used as follows:

Dialogue diagramming
A formal method for designing and representing human-computer dialogues using box-and-line diagrams.

1. Top: Contains a unique display reference number used by other displays for referencing it
2. Middle: Contains the name or description of the display
3. Bottom: Contains display reference numbers that can be accessed from the current display

All lines connecting the boxes within dialogue diagrams are assumed to be bidirectional and, thus, do not need arrowheads to indicate direction. With this capability, users are allowed to always move forward and backward between adjacent displays. If you desire only unidirectional flows within a dialogue, arrowheads should be placed at one end of the line. Within a dialogue diagram, you can easily represent the sequencing of displays, the selection of one display over another, or the repeated use of a single display (e.g., a data-entry display). These three concepts—sequence, selection, and iteration—are illustrated in Figure 8-19.

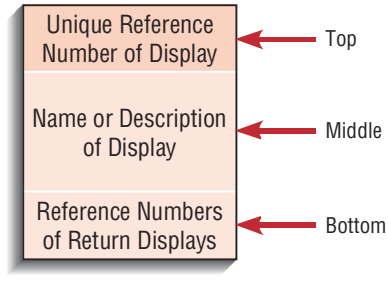
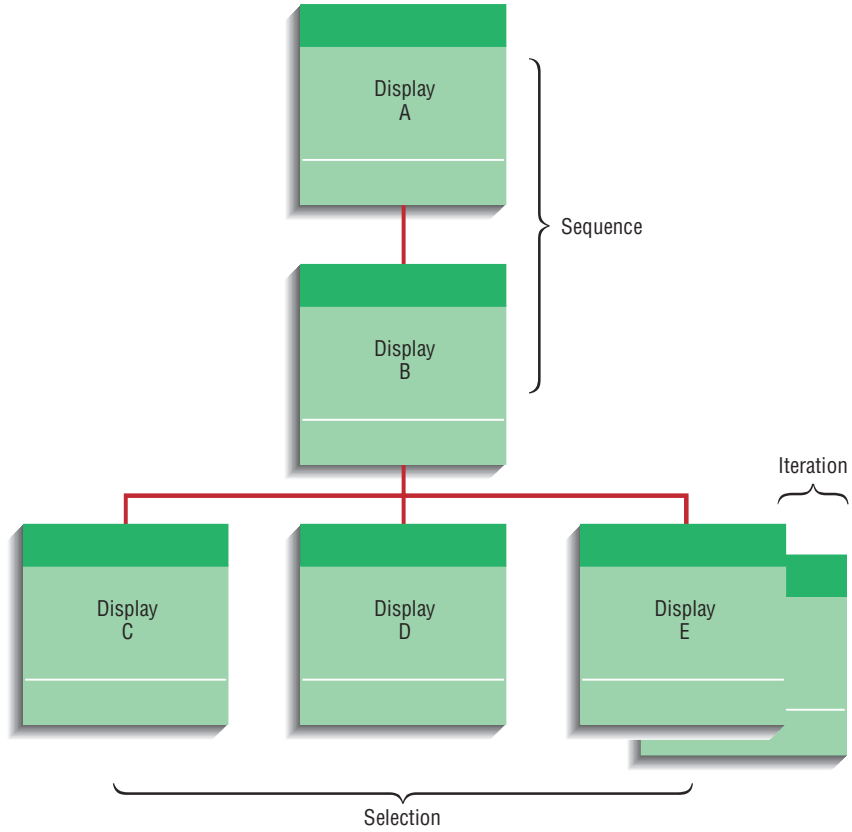


FIGURE 8-18
A dialogue-diagramming box has three sections.

FIGURE 8-19
Dialogue diagram illustrating sequence, selection, and iteration.



Continuing with our PVF example, Figure 8-20 shows a partial dialogue diagram for processing the marketing manager's request. In this diagram, the analyst placed the request to view year-to-date customer information within the context of the overall Customer Information System. The user must first gain access to the system through a log-on procedure (item 0). If log-on is successful, a main menu is displayed that has four items (item 1). Once the user selects the Individual Customer Information display (item 2), control is transferred to the Select Customer display (item 2.1). After a customer is selected, the user is presented with an option to view customer information four different ways (item 2.1.1). Once the user views the customer's year-to-date transaction activity (item 2.1.1.2), the system will allow the user to back up to select a different customer or back up to the main menu (see bottom of item 2.1.1.2).

Building Prototypes and Assessing Usability Building dialogue prototypes and assessing usability are often optional activities. Some systems may be simple and straightforward. Others may be more complex but are extensions to existing systems where dialogue and display standards have already been established. In either case, you may not be required to build prototypes and do a formal assessment. However, for many other systems, it is critical that you build prototype displays and then assess the dialogue; developing a prototype can pay numerous dividends later in the systems development life cycle (e.g., it may be easier to implement a system or train users on a system they have already seen and used).

Building prototype displays is often a relatively easy activity if you use graphical development environments such as Microsoft's Visual Basic.Net. Some systems development environments include easy-to-use input and output (form, report, or window) design utilities. Also several tools called "Prototypers" or "Demo Builders" allow you to design displays quickly and show how an

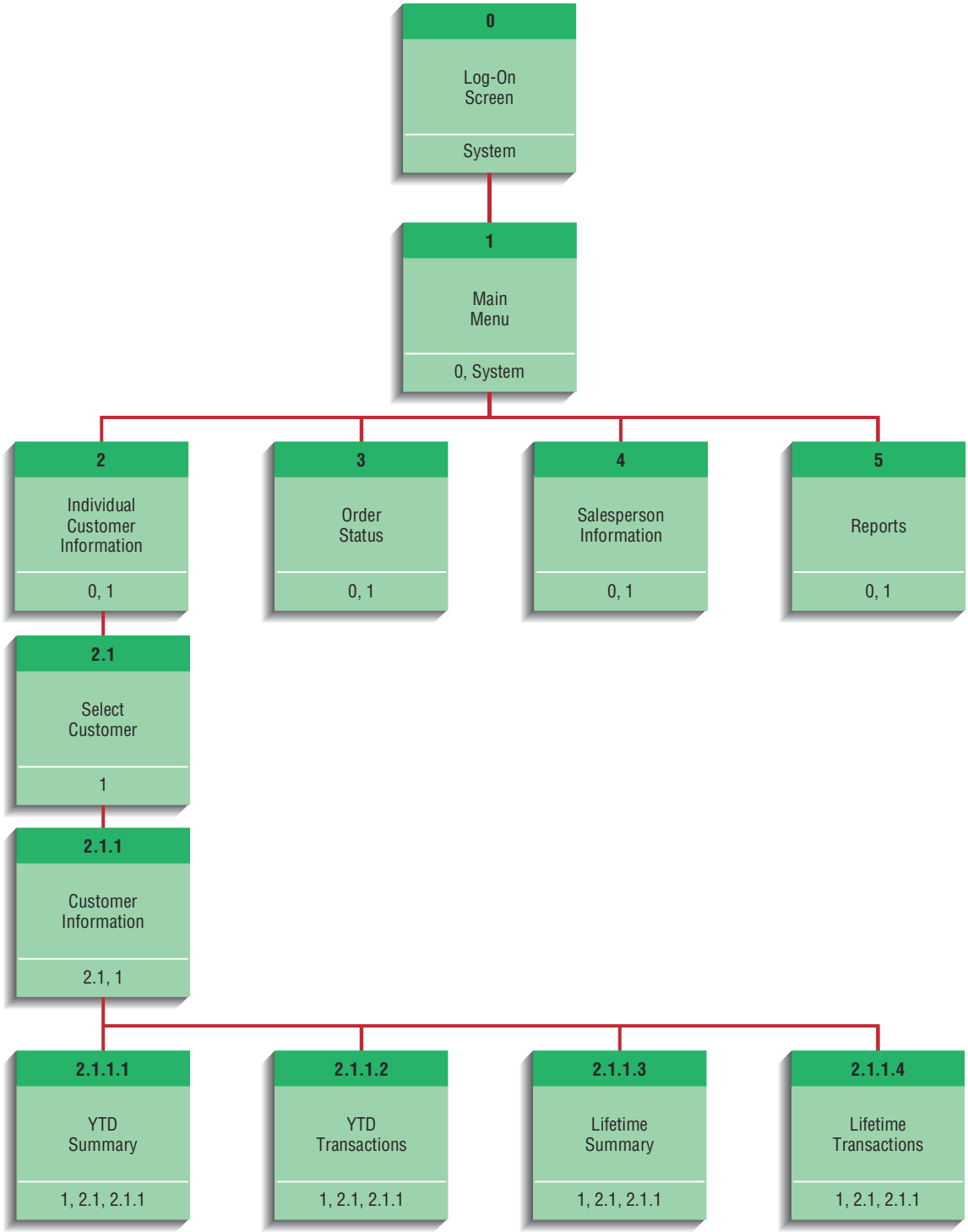


FIGURE 8-20
Dialogue diagram for the Customer Information System at Pine Valley Furniture.

interface will work within a full system. These demo systems allow users to enter data and move through displays as if they were using the actual system. Such activities are useful not only for showing how an interface will look and feel but also for assessing usability and performing user training long before actual systems are completed.

Pine Valley Furniture WebStore: Designing the Human Interface

Designing the human interface for an Internet-based electronic commerce application is a central and critical design activity. Because customers will interact with a company at this point, much care must be put into its design. Like the process followed when designing the interface for other types of systems, a prototyping design process is most appropriate when designing the human interface for an Internet electronic commerce system. Although the techniques and technology for building the human interface for Internet sites are rapidly evolving, several general design guidelines have emerged. In this section, we examine some of these as they apply to the design of Pine Valley Furniture's WebStore.

General Guidelines for Designing Web Interfaces

Over the years, interaction standards have emerged for virtually all of the commonly used desktop computing environments such as Windows or Mac OS. However, some interface design experts believe that the growth of the Web has resulted in a big step backward for interface design. One problem is that countless nonprofessional developers are designing commercial Web applications. In addition, four other important factors contribute to a lack of standards (Johnson, 2007):

- Web's single "click-to-act" method of loading static hypertext documents (i.e., most buttons on the Web do not provide click feedback)
- Limited capabilities of most Web browsers to support finely grained user interactivity
- Limited agreed-upon standards for encoding Web content and control mechanisms
- Lack of maturity of Web scripting and programming languages as well as limitations in commonly used Web GUI component libraries

In addition to these contributing factors, designers of Web interfaces and dialogues are often guilty of many design errors. Although not inclusive of all possible errors, Table 8-15 summarizes those errors that are particularly troublesome.

General Guidelines for Web Layouts

As previously mentioned, the rapid deployment of Internet Web sites has resulted in having countless people design sites who, arguably, have limited ability to do so. To put this into perspective, consider the following quote from Web design guru, Jakob Nielsen (1999a, pp. 65–66):

> If the [Web's] growth rate does not slow down, the Web will reach 200 million sites sometime during 2003. . . . The world has about 20,000 user interface [UI] professionals. If all sites were to be professionally designed by a single UI professional, we can conclude that every UI professional in the world would need to design one Web site every working hour from now on to meet demand. This is obviously not going to happen. . . .

Continued growth in the number of unique Web sites, estimated to exceed 250 million in early 2011, makes this problem increasingly dire. Three possible solutions to the problem include the following:

- Make it possible to design reasonably usable sites without having UI expertise
- Train more people in good Web design
- Live with poorly designed sites that are hard to use

TABLE 8-15: Common Errors When Designing the Interface and Dialogues of Web Sites

Error	Description
Opening new browser window	Avoid opening a new browser window when a user clicks on a link unless it is clearly marked that a new window will be opened; users may not see that a new window has been opened, which will complicate navigation, especially when moving backward.
Breaking or slowing down the back button	Make sure users can use the back button to return to prior pages. Avoid opening new browser window; using immediate redirect where and when a user clicks the back button, they are pushed forward to an undesired location; or prevent caching such that each click of the back button requires a new trip to the server.
Complex URLs	Avoid overly long and complex URLs that make it more difficult for users to understand where they are and can cause problems if users want to e-mail page locations to others.
Orphan pages	Avoid having pages with no "parent" that can be reached by using a back button; requires users to "hack" the end of the URL to get back to a prior page.
Scrolling navigation pages	Avoid placing navigational links below where a page opens, because many users may miss these important options that are not immediately visible.
Lack of navigation support	Make sure your pages conform to user expectations by providing commonly used icon links, such as a site logo at the top of major elements. Also place these elements on pages in a consistent manner.
Hidden links	Make sure you leave a border around images that are links, don't change link colors from normal defaults, and avoid embedding links within long blocks of text.
Links that don't provide enough information	Avoid not turning off link-marking borders so that links clearly show which links users have clicked and which they have not. Make sure users know which links are internal anchor points versus external links, and indicate if a link brings up a separate browser window from those that do not. Finally, make sure link images and text provide enough information to the user so that they understand the meaning of the link.
Buttons that provide no click feedback	Avoid using image buttons that don't clearly change when being clicked; use Web GUI toolkit button, HTML from-submit buttons, or simple textual links.

Designing forms and reports may lead to errors that are specific to Web site design. It is unfortunately beyond the scope of this book to critically examine all possible design problems with contemporary Web sites. Here, we will simply summarize those errors that commonly occur and are particularly detrimental to the user's experience (see Table 8-16). Fortunately, numerous excellent sources are available for learning more about designing useful Web sites (Ash, 2008; Loveday and Niehaus, 2007; Nielsen and Loranger, 2006; Veeny, 2008; www.useit.com; www.webpagesthatsuck.com).

Designing the Human Interface at Pine Valley Furniture

The first design activity that Jim Woo and the PVF development team focused on was the human-computer interface. To begin, they reviewed many popular electronic commerce Web sites and established the following design guidelines:

- Menu-driven navigation with cookie crumbs
- Lightweight graphics
- Forms and data integrity rules
- Template-based HTML

In order to ensure that all team members understood what was meant by each guideline, Jim organized a design briefing to explain how each would be incorporated into the WebStore interface design.

TABLE 8-16: Common Errors When Designing the Layout of Web Pages

Error	Recommendation
Nonstandard use of GUI widgets	Make sure that when using standard design items, that they behave in accordance to major interface design standards. For example, the rules for radio buttons state that they are used to select one item among a set of items that is, not confirmed until "OK'ed" by a user. In many Web sites, selecting radio buttons are used as both *selection* and *action*.
Anything that looks like advertising	Because research on Web traffic has shown that many users have learned to stop paying attention to Web advertisement, make sure that you avoid designing any legitimate information in a manner that resembles advertising (e.g., banners, animations, pop-ups).
Bleeding-edge technology	Make sure that users don't need the latest browsers or plug-ins to view your site.
Scrolling text and looping animators	Avoid scrolling text and animations because they are both hard to read and often equated by users with advertising.
Nonstandard link colors	Avoid using nonstandard colors to show links and for showing links that users have already used; nonstandard colors will confuse the user and reduce ease of use.
Outdated information	Make sure that your site is continuously updated so that users "feel" that the site is regularly maintained and updated. Outdated content is a sure way to lose credibility.
Slow download times	Avoid using large images, lots of images, unnecessary animations, or other time-consuming content that will slow the downloading time of a page.
Fixed-formatted text	Avoid fixed-formatted text that requires users to scroll horizontally to view contents or links.
Displaying long lists as long pages	Avoid requiring users to scroll down a page to view information, especially navigational controls. Manage information by showing only *N* items at a time, using multiple pages, or by using a scrolling container within the window.

Menu-Driven Navigation with Cookie Crumbs

After reviewing several sites, the team concluded that menus should stay in the exact same place throughout the entire site. They concluded that placing a menu in the same location on every page will help customers to become familiar with the site more quickly and therefore to navigate through the site more rapidly. Experienced Web developers know that the quicker customers can reach a specific destination at a site, the quicker they can purchase the product they are looking for or get the information they set out to find. Jim emphasized this point by stating, "These details may seem silly, but the second users find themselves 'lost' in our site, they're gone. One mouse click and they're no longer shopping at Pine Valley Furniture but at one of our competitor's sites."

Cookie crumbs
A technique for showing users where they are in a Web site by placing a series of "tabs" on a Web page that shows users where they are and where they have been.

A second design feature, and one that is being used on many electronic commerce sites, is cookie crumbs. **Cookie crumbs** are a technique for showing users where they are in the site by placing "tabs" on a Web page that remind users where they are and where they have been. These tabs are hypertext links that can allow users to move backward quickly in the site. For example, suppose that a site is four levels deep, with the top level called "Entrance," the second "Products," the third "Options," and the fourth "Order." As the user moves deeper into the site, a tab is displayed across the top of the page showing the user where she is and giving her the ability to jump backward quickly one or more levels. In other words, when first entering the store, a tab is displayed at the top (or some other standard place) of the screen with the word "Entrance." After moving down a level, two tabs are displayed, "Entrance" and "Products." After selecting a product on the second level, a third level is displayed where a user can choose product options. When this level is displayed, a third tab is

produced with the label "Options." Finally, if the customer decides to place an order and selects this option, a fourth-level screen is displayed and a fourth tab displayed with the label "Order." In summary:

Level 1: Entrance
Level 2: Entrance → Products
Level 3: Entrance → Products → Options
Level 4: Entrance → Products → Options → Order

By using cookie crumbs, users know exactly how far they have wandered from "home." If each tab is a link, users can quickly jump back to a broader part of the store should they not find exactly what they are looking for. Cookie crumbs serve two important purposes. First, they allow users to navigate to a point previously visited and will ensure that they are not lost. Second, it clearly shows users where they have been and how far they have gone from home.

Lightweight Graphics

In addition to easy menu and page navigation, the PVF development team wants a system where Web pages load quickly. A technique to assist in making pages load quickly is **lightweight graphics.** Lightweight graphics are the use of small simple images that allow a page to load as quickly as possible. "Using lightweight graphics allows pages to load quickly and helps users to reach their final location in the site—hopefully the point-of-purchase area—as quickly as possible. Large color images will only be used for displaying detailed product pictures that customers explicitly request to view," Jim explained. Experienced Web designers have found that customers are not willing to wait at each hop of navigation for a page to load, just so they have to click and wait again. The quick feedback that a Web site with lightweight graphics can provide will help to keep customers at the WebStore longer.

Lightweight graphics
The use of small simple images to allow a Web page to be displayed more quickly.

Forms and Data Integrity

Because the goal of the WebStore is to have users place orders for products, all forms that request information should be clearly labeled and provide adequate room for input. If a specific field requires a specific input format such as a date of birth or phone number, it must provide a clear example for the user so that data errors can be reduced. Additionally, the site must clearly designate which fields are optional, which are required, and which have a range of values.

Jim emphasized, "All of this to me seems a bit like overkill, but it makes processing the data much simpler. Our site checks all data before submitting it to the server for processing. This allows us to provide quicker feedback to the user on any data-entry error and eliminate the possibility of writing erroneous data into the permanent database. Additionally, we want to provide a disclaimer to reassure our customers that the data will be used only for processing orders, will never be sold to marketers, and will be kept strictly confidential."

Template-Based HTML

When Jim talked with the consultants about the WebStore during the analysis phase, they emphasized the advantages of using **template-based HTML.** He was told that when displaying individual products, it would be advantageous to try to have a few "templates" that could be used to display the entire product line. In other words, not every product needs its own page; the development time for that would be far too great. Jim explained, "We need to look for ways to write a module once and reuse it. This way, a change requires modifying one page, not seven hundred. Using HTML templates will help us create an interface that is easy to maintain. For example, a desk and a filing cabinet are two

Template-based HTML
Templates to display and process common attributes of higher-level, more abstract items.

completely different products. Yet, both have an array of finishes to choose from. Logically, each item requires the same function—namely: 'display all finishes.' If designed correctly, this function can be applied to all products in the store. On the other hand, if we write a separate module for each product, it would require us to change each and every module every time we make a product change, like adding a new finish. But a function such as 'display all finishes,' written once and associated with all appropriate products, will require the modification of one generic or 'abstract' function, not hundreds."

Key Points Review

1. **Explain the process of designing forms and reports, and the deliverables for their creation.**

   Forms and reports are created through a prototyping process. Once created, designs may be stand-alone or integrated into actual working systems. The purpose of the prototyping process, however, is to show users what a form or report will look like when the system is implemented. The outcome of this activity is the creation of a specification document where characteristics of the users, tasks, system, and environment are outlined along with each form and report design. Performance testing and usability assessments may also be included in the design specification.
2. **Apply the general guidelines for formatting forms and reports.**

   Guidelines should be followed when designing forms and reports. These guidelines, proven over years of experience with human-computer interaction, help to create professional, usable systems. Guidelines are available for the use of titles, layout of fields, navigation between pages or screens, highlighting information, format of text, and the appropriate use and layout of tables and lists.
3. **Format text, tables, and lists effectively.**

   Textual output is becoming increasingly important as text-based applications such as electronic mail, bulletin boards, and information services become more popular. Text should be displayed using common writing conventions such as mixed uppercase and lowercase, appropriate punctuation, left-justified, and a minimal amount of obscure abbreviations. Words should not be hyphenated between lines, and blocks of text should be double-spaced or, minimally, a blank line should be placed between each paragraph. Tables and lists should have meaningful labels that clearly stand out. Information should be sorted and arranged in a meaningful way. Numeric data should be right-justified.
4. **Explain the process of designing interfaces and dialogues, and the deliverables for their creation.**

   Designing interfaces and dialogues is a user-focused activity that follows a prototyping methodology of iteratively collecting information, constructing a prototype, assessing usability, and making refinements. The deliverable and outcome from interface and dialogue design is the creation of a specification that can be used to implement the design.
5. **Describe and apply the general guidelines for interface design, including guidelines for layout design, structuring data-entry-fields, providing feedback, and system help.**

   To have a usable interface, users must be able to move the cursor position, edit data, exit with different consequences, and obtain help. Numerous techniques for structuring and controlling data entry, as well as providing feedback, prompting, error messages, and a well-organized help function can be used to enhance usability.
6. **Design human-computer dialogues, including the use of dialogue diagramming.**

   Human-computer dialogues should be consistent in design, allowing for shortcuts, providing feedback and closure on tasks, handling errors, allowing for action reversal, and giving the user a sense of control and ease of navigation. Dialogue diagramming is a technique for representing human-computer dialogues. The technique uses boxes to represent screens, forms, or reports and lines to show the flow between each.
7. **Discuss interface design guidelines unique to the design of Internet-based electronic commerce systems.**

   The human-computer interface is a central and critical aspect of any Internet-based electronic commerce system. Using menu-driven navigation with cookie crumbs ensures that users can easily understand and navigate a system. Using lightweight graphics ensures that Web pages load quickly. Ensuring data integrity means that customer information is processed quickly, accurately, and securely. Using common templates ensures a consistent interface that is easy to maintain.

Key Terms Checkpoint

Here are the key terms from the chapter. The page where each term is first explained is in parentheses after the term.

1. **Audit trail (p. 253)**
2. **Cookie crumbs (p. 264)**
3. **Dialogue (p. 258)**
4. **Dialogue diagramming (p. 259)**
5. **Form (p. 234)**
6. **Lightweight graphics (p. 265)**
7. **Report (p. 234)**
8. **Template-based HTML (p. 265)**

Match each of the key terms above with the definition that best fits it.

____ 1. Templates to display and process common attributes of higher-level, more abstract items.

____ 2. A formal method for designing and representing human-computer dialogues using box and line diagrams.

____ 3. A business document that contains only predefined data; it is a passive document used only for reading or viewing; typically contains data from many unrelated records or transactions.

____ 4. A technique for showing users where they are in a Web site by placing a series of "tabs" on a Web page that shows users where they are and where they have been.

____ 5. The sequence of interaction between a user and a system.

____ 6. A business document that contains some predefined data and may include some areas where additional data are to be filled in; typically based on one database record.

____ 7. A record of the sequence of data entries and the date of those entries.

____ 8. The use of small simple images to allow a Web page to be displayed more quickly.

Review Questions

1. Describe the prototyping process of designing forms and reports. What deliverables are produced from this process? Are these deliverables the same for all types of system projects? Why or why not?
2. To which initial questions must the analyst gain answers in order to build an initial prototype of a system output?
3. How should textual information be formatted on a help screen?
4. What type of labeling can you use in a table or list to improve its usability?
5. What column, row, and text formatting issues are important when designing tables and lists?
6. Describe how numeric, textual, and alphanumeric data should be formatted in a table or list.
7. Provide some examples where variations in user, task, system, and environmental characteristics might impact the design of system forms and reports.
8. Describe the process of designing interfaces and dialogues. What deliverables are produced from this process? Are these deliverables the same for all types of system projects? Why or why not?
9. List and describe the functional capabilities needed in an interface for effective entry and navigation. Which capabilities are most important? Why? Will this be the same for all systems? Why or why not?
10. Describe the general guidelines for structuring data-entry fields. Can you think of any instances when it would be appropriate to violate these guidelines?
11. Describe four types of data errors.
12. Describe the types of system feedback. Is any form of feedback more important than the others? Why or why not?
13. Describe the general guidelines for designing usable help. Can you think of any instances when it would be appropriate to violate these guidelines?
14. What steps do you need to follow when designing a dialogue? Of the guidelines for designing a dialogue, which is most important? Why?
15. Describe what is meant by a cookie crumb. How do these help prevent users from getting lost?
16. Describe why you might want to use lightweight graphics on some Web pages and large detailed graphics on others.
17. Why is it especially important to eliminate data-entry errors on an electronic commerce Web site?
18. How can template-based HTML help to make a large electronic commerce site more maintainable?

Problems and Exercises

1. Imagine that you are to design a budget report for a colleague at work using a spreadsheet package. Following the prototyping discussed in the chapter (see also Figure 1-12), describe the steps you would take to design a prototype of this report.
2. Consider a system that produces inventory reports at a local retailer. Alternatively, consider a system that produces student academic records for the records office at a university. For whichever system you choose, answer the following design questions: Who will use the output? What is the purpose of the output? When is the output needed, and when is the information that will be used within the output available? Where does the output need to be delivered? How many people need to view the output?
3. Imagine the worst possible reports from a system. What is wrong with them? List as many problems as you can. What are the consequences of such reports? What could go wrong as a result? How does the prototyping process help guard against each problem?
4. Given the guidelines presented in this chapter, identify flaws in the design of the Report of Employees shown below. What assumptions about users and tasks did you make in order to assess this design? Redesign this report to correct these flaws.

Report of Employees-1-2-08

Em_ID	Name, Title
0124543	John Smith, VP Marketing
2345645	Jared Wright, Project Manager
2342456	Jennifer Chang, Systems Analyst
4564234	Mark Walters, Software Engineer
7875468	Nick Shelley, BI Analyst
4446789	Kim Eagar, HR Manager
4678899	Emily Graham, Receptionist
4452378	Matt Hoffman, Network Operations Specialist

5. Consider the design of a registration system for a hotel. Following design specification items in Figure 8-11, briefly describe the relevant users, tasks, and displays involved in such a system.
6. Obtain a report of some information, either from your employer (e.g., a budget or project report) or from your school (e.g., your student academic record). Evaluate the design of the report using the general guidelines in Table 8-2.
7. Design one sample data-entry screen for a hotel registration system using the data-entry guidelines provided in this chapter (see Table 8-7). Support your design with arguments for each of the design choices you made.
8. Describe some typical dialogue scenarios between users and a hotel registration system. For hints, reread the section in this chapter that provides sample dialogue between users and the Customer Information System at Pine Valley Furniture.
9. Represent the dialogues from the previous question through the use of dialogue diagrams.
10. Think of an online retailer you've recently used or considered using for a purchase. Why is good design of that retailer's interface important for the retailer? Visit the online retailer and evaluate the interface, highlighting several good things and several bad things.

Discussion Questions

1. Discuss the differences between a form and a report. What characteristics make a form or report good (bad) and effective (ineffective)?
2. Discuss the various ways that information can be highlighted on a computer display. Which methods are most effective? Are some methods better than others? If so, why and when?
3. What problems can occur if a system fails to provide clear feedback and error messages to users?
4. Use a search engine to find recommendations for good design of Web interfaces. How are these recommendations similar to those discussed in this chapter? How do they differ?

Case Problems

1. Pine Valley Furniture

Pine Valley Furniture's Customer Tracking System project is now ready to move into the systems design phase. You are excited because this phase involves designing the new system's forms, reports, and databases. During this morning's meeting with Jim Woo, he asked you to design several forms and reports for the new Customer Tracking System.

During the requirements determination phase, Jackie Judson requested that a customer profile be created for each customer. The customer profile is established when new customers place their first order. Customers will have the option of not completing a profile; however, to encourage customer participation, a 10 percent discount on the customer's total order will be given to each customer who completes a profile. In the beginning, existing customers will also be given the opportunity to participate in the customer profiling process. Customer profile information will be collected via a Customer Profile Form.

Gracie Breshers, a marketing executive, has requested that the Customer Tracking System generate a Products by Demographics Summary Report. This summary report should identify Pine Valley Furniture's major furniture categories, such as business furniture, living room, dining room, home office, and kitchen. Within each furniture category, she would like the total sales by region and customer age reported. She has also requested that several detailed reports be prepared; these reports will associate customer demographics with specific furniture category items.

Thi Hwang, a Pine Valley Furniture sales executive, would like to know, in a Customer Purchasing Frequency Report, how many of Pine Valley Furniture's customers are repeat customers, in terms of percentages, and how often they make purchases. Additionally, he would like to have this information categorized by customer type. For each customer type, he would like to know the frequency of the purchases. For instance, does this type of customer place an order at least once a month, at least every six months, at least once a year, or longer than one year? To be considered a repeat customer, the customer must have made two separate purchases within a two-year period.

a. What data will the Customer Profile Form need to collect? Using the guidelines presented in the chapter, design the Customer Profile Form.
b. Using the guidelines presented in the chapter, design the Products by Demographics Summary Report.
c. Using the guidelines presented in the chapter, design the Customer Purchasing Frequency Report.
d. Modify the dialogue diagram presented in Figure 8-20 to reflect the addition of the Customer Profile Form, Products by Demographics Summary Report, and the Customer Purchasing Frequency Report.

2. Hoosier Burger

As the lead analyst for the Hoosier Burger project, you have worked closely with Bob and Thelma Mellankamp. Having completed the systems analysis phase, you are now ready to begin designing the new Hoosier Burger information system. As the lead analyst on this project, you are responsible for overseeing the development of the forms, reports, and databases required by the new system. Because the inventory system is being automated and a new delivery system is being implemented, the Hoosier Burger system requires the development of several forms and reports.

Using your data-flow diagrams and entity-relationship diagrams, you begin the task of identifying all the necessary forms and reports. You readily identify the need for a Delivery Customer Order Form, a Customer Account Balance Form, a Low-in-Stock Report, and a Daily Delivery Summary Report. The Delivery Customer Order Form will capture order details for those customers placing delivery orders. Bob will use the Customer Account Balance Form to look up a customer's current account balance. The Low-in-Stock Report will be generated daily to identify all food items or supplies that are low in stock. The Daily Delivery Summary Report will summarize each day's delivery sales by menu item sold.

a. What data will the Delivery Customer Order Form need to collect? Using the design guidelines presented in the chapter, design the Delivery Customer Order Form.
b. What data will the Customer Account Balance Form need to show? Using the design guidelines presented in the chapter, design the Customer Account Balance Form.
c. Using the design guidelines presented in the chapter, design the Daily Delivery Summary Report.
d. Using the design guidelines presented in the chapter, design the Low-in-Stock Report.

3. Pet Nanny

Pet owners often have difficulty locating pet-sitters for their pets, boarding their pets, or just

getting the pets to the veterinarian. Recognizing these needs, Gladys Murphy decided to open Pet Nanny, a business providing specialized pet-care services to busy pet owners. The company provides a multitude of services, including pet grooming, massage, day care, home care, aromatherapy, boarding, and pickup and delivery. The company has been experiencing a steady increase in demand for its services.

Initially, when the company was founded, all pet-care records were kept manually. However, Gladys recognized the need to update Pet Nanny's existing systems and hired your consulting firm to design the system changes. Your analyst team has just completed the requirements structuring phase and has selected an alternative design strategy. You are now ready to begin the systems design phase.

During the analysis phase, you determined that several forms and reports were necessary, including a Pet Enrollment Form, Pet Service Form, Pickup and Delivery Schedule Report, and Daily Boarding Report. When a customer wishes to use Pet Nanny's services for a new pet, the customer must provide basic information about the pet. For instance, the customer is asked to provide his or her name, address, phone number, the pet's name, birth date (if known), and special care instructions. When a customer requests a special service for the pet, such as grooming or a massage, a service record is created. Because the pickup and delivery service is one of the most popular services offered by Pet Nanny, Gladys wants to make sure that no pets are forgotten. Each morning a report listing the pet pickups and deliveries is created. She also needs a report listing the pets being boarded, their special needs, and their length of stay.

a. What data should the Pet Enrollment Form collect? Using the guidelines provided in the chapter, design the Pet Enrollment Form.
b. What data should the Pet Service Form collect? Using the guidelines provided in the chapter, design the Pet Service Form.
c. Using the guidelines provided in the chapter, design the Pickup and Delivery Schedule Report.
d. Using the guidelines provided in the chapter, design the Daily Boarding Report.

CASE: PETRIE'S ELECTRONICS

Designing the Human Interface

Jim Watanabe, project director for the "No Customer Escapes" customer loyalty system for Petrie's Electronics, walked into the conference room. Sally Fukuyama, from marketing, and Sanjay Agarwal, from IT, were already there. Also at the meeting was Sam Waterston, one of Petrie's key interface designers.

"Good morning," Jim said. "I'm glad everyone could be here today. I know you are all busy, but we need to make some real progress on the customer account area for 'No Customer Escapes.' We have just awarded the development of the system to XRA, and once all the documents are signed, they will be coming over to brief us on the implementation process and our role in it."

"I'm sorry," Sally said, "I don't understand. If we are licensing their system, what's left for us to do? Don't we just install the system and we're done?" Sally took a big gulp of coffee from her cup.

"I wish it was that easy," Jim said. "While it is true that we are licensing their system, there are many parts of it that we need to customize for our own particular needs. One obvious area where we need to customize is all of the human interfaces. We don't want the system to look generic to our loyal customers—we need to make it unique to Petrie's."

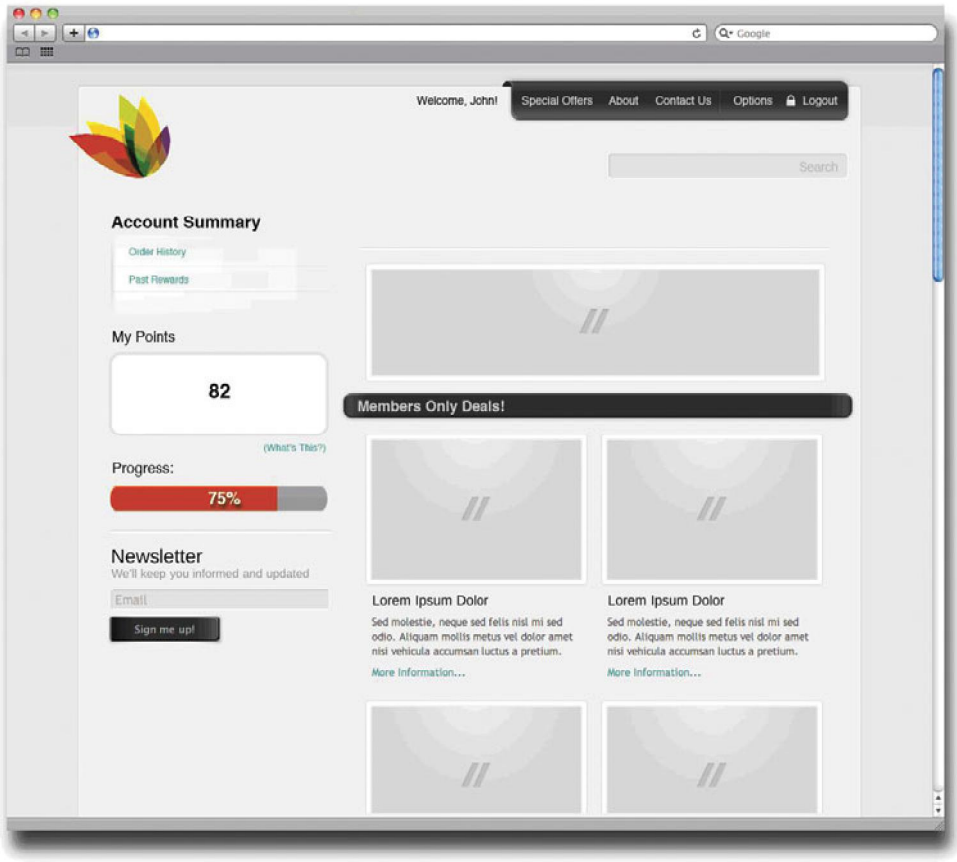
"And we have to integrate the XRA system with our own operations," added Sanjay. "For example, we have to integrate our existing marketing and product databases with the XRA CRM (see PE Figure 6-2). That's just one piece of all the technical work we have to do."

"We've already done some preliminary work on system functionality and the conceptual database," Jim said. "I want to start working on interface issues now. That's why Sam is here. What we want to do today is start work on how the customer account area should look and operate. And Sally, the customer loyalty site is a great opportunity for marketing. We can advertise specials and other promotions to our best customers on this site. Maybe we could use it to show offers that are only good for members of our loyalty program."

"Oh yeah," Sally replied, "that's a great idea. How would that look?"

"I have ideas," said Sam. Using a drawing program on a tablet PC, he started to draw different zones that would be part of the interface. "Here at the top we would have a simple banner that says 'Petrie's' and the name of the program."

"It's not really going to be called 'No Customer Escapes,' is it?" asked Sally.



PE FIGURE 8-1
Preliminary design for the customer account area.

"No, that's an internal name," replied Jim, "but I don't know what the real name will be yet."

"OK, so the real name of the program will go in the banner, after 'Petrie's.' Then on the left side, we'll have a sidebar that has overview information about the customer account, things like name and points balance," said Sam, drawing in a sidebar on the left of the screen. "There will also be links to more detailed information about the account, so the customer can see more details on past transactions and on his or her profile."

"So the rest of the screen is open. That would be a perfect place for marketing information," suggested Sally. "Would we want just one big window for marketing? Maybe we could divide it up into additional windows, so we could use one to focus on general promotions and one to advertise 'member only' promotions?"

"Yeah, we can do that," said Sam.

Just then Jim's phone beeped. Jim looked at it. Uh-oh, it was an urgent message from his boss, the director of IT. "Sorry, I need to take care of this immediately," he told the group. "Can you guys work on this some more and then send me some of the screen designs you come up with?"

Later that afternoon, after the crisis was over, Jim sat back down at his desk for the first time in what seemed like a very long time. He glanced over his e-mail and noticed there was a message from Sam. Attached was a preliminary design for the customer account area. Jim opened it and looked it over (PE Figure 8-1). Hmmm, not bad, he thought. This is a good place for us to start.

Case Questions

1. Using the guidelines from this chapter and other sources, evaluate the usability of the page design depicted in PE Figure 8-1.
2. Chapter 8 encourages the design of a help system early in the design of the human interface. How would you incorporate help into the interface as shown in PE Figure 8-1?
3. Describe how cookie crumbs could be used in this system. Are cookie crumbs a desirable navigation aid for this system? Why or why not?
4. The page design depicted in PE Figure 8-1 links to an Order History page. Sketch a similar layout for the Order History page, following guidelines from Chapter 8.
5. Describe how the use of template-based HTML might be leveraged in the design of the "No Customer Escapes" system.

nine

Designing Databases

Monkey Business Images/Shutterstock

Chapter Objectives

After studying this chapter, you should be able to:

- Concisely define each of the following key database design terms: *relation, primary key, functional dependency, foreign key, referential integrity, field, data type, null value, denormalization, file organization, index,* and *secondary key.*
- Explain the role of designing databases in the analysis and design of an information system.
- Transform an entity-relationship (E-R) diagram into an equivalent set of well-structured (normalized) relations.
- Merge normalized relations from separate user views into a consolidated set of well-structured relations.
- Choose storage formats for fields in database tables.
- Translate well-structured relations into efficient database tables.
- Explain when to use different types of file organizations to store computer files.
- Describe the purpose of indexes and the important considerations in selecting attributes to be indexed.

Chapter Preview . . .

In Chapter 7 you learned how to represent an organization's data graphically using an entity-relationship (E-R) diagram and Microsoft Visio. In this chapter, you learn guidelines for clear and efficient data files and about logical and physical database design. It is likely that the human interface and database design steps will happen in parallel, as illustrated in the SDLC in Figure 9-1.

Logical and physical database design has five purposes:

1. Structure the data in stable structures that are not likely to change over time and that have minimal redundancy.
2. Develop a logical database design that reflects the actual data requirements that exist in the forms (hard copy and computer displays) and reports of an information system. For this reason, database design is often done in parallel with the design of the human interface of an information system.
3. Develop a logical database design from which we can do physical database design. Because most information systems today use relational database management systems, logical database design usually uses a relational database model, which represents data in simple tables with common columns to link related tables.
4. Translate a relational database model into a technical file and database design.
5. Choose data-storage technologies (such as hard disk, CD-ROM, or flash disk) that will efficiently, accurately, and securely process database activities.

The implementation of a database (i.e., creating and loading data into files and databases) is done during the next phase of the systems development life cycle. Because implementation is technology specific, we address implementation issues only at a general level in Chapter 10.

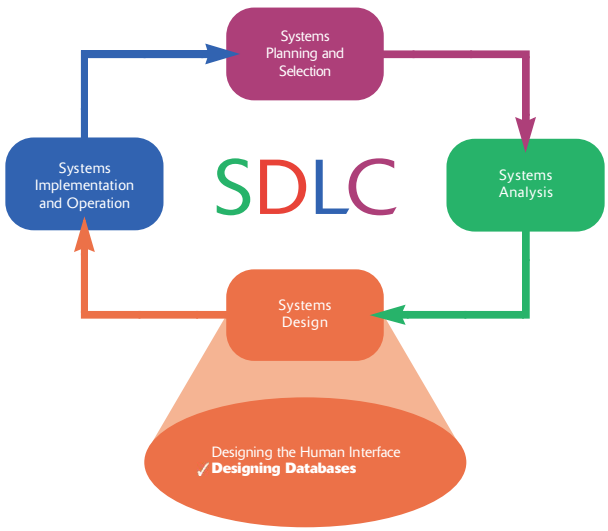


FIGURE 9-1
Systems development life cycle. Systems analysts design databases during the systems design phase. Database design typically occurs in parallel with other design steps.

Database Design

File and database design occurs in two steps. You begin by developing a logical database model, which describes data using a notation that corresponds to a data organization used by a database management system. This system software is responsible for storing, retrieving, and protecting data (such as Microsoft Access, Oracle, or SQL Server). The most common style for a logical database model is the relational database model. Once you develop a clear and precise logical database model, you are ready to prescribe the technical specifications for computer files and databases in which to store the data ultimately. A physical database design provides these specifications.

You typically do logical and physical database design in parallel with other systems design steps. Thus, you collect the detailed specifications of data necessary for logical database design as you design system inputs and outputs. Logical database design is driven not only from the previously developed E-R data model for the application but also from form and report layouts. You study data elements on these system inputs and outputs and identify interrelationships among the data. As with conceptual data modeling, the work of all systems development team members is coordinated and shared through the project dictionary or repository. The designs for logical databases and system inputs and outputs are then used in physical design activities to specify to computer programmers, database administrators, network managers, and others how to implement the new information system. We assume for this text that the design of computer programs and distributed information processing and data networks are topics of other courses, so we concentrate on the aspect of physical design most often undertaken by a systems analyst—physical file and database design.

The Process of Database Design

Figure 9-2 shows that database modeling and design activities occur in all phases of the systems development process. In this chapter we discuss methods that help you finalize logical and physical database designs during the design phase. In logical database design you use a process called *normalization*, which is a way to build a data model that has the properties of simplicity, nonredundancy, and minimal maintenance.

In most situations, many physical database design decisions are implicit or eliminated when you choose the data-management technologies to use with the application. We concentrate on those decisions you will make most frequently and use Microsoft Access to illustrate the range of physical database design parameters you must manage. The interested reader is referred to Hoffer, Ramesh, and Topi (2011) for a more thorough treatment of techniques for logical and physical database design.

Four steps are key to logical database modeling and design:

1. Develop a logical data model for each known user interface (form and report) for the application, using normalization principles.
2. Combine normalized data requirements from all user interfaces into one consolidated logical database model; this step is called *view integration*.
3. Translate the conceptual E-R data model for the application, developed without explicit consideration of specific user interfaces, into normalized data requirements.
4. Compare the consolidated logical database design with the translated E-R model and produce, through view integration, one final logical database model for the application.

During physical database design, you use the results of these four key logical database design steps. You also consider definitions of each attribute; descriptions

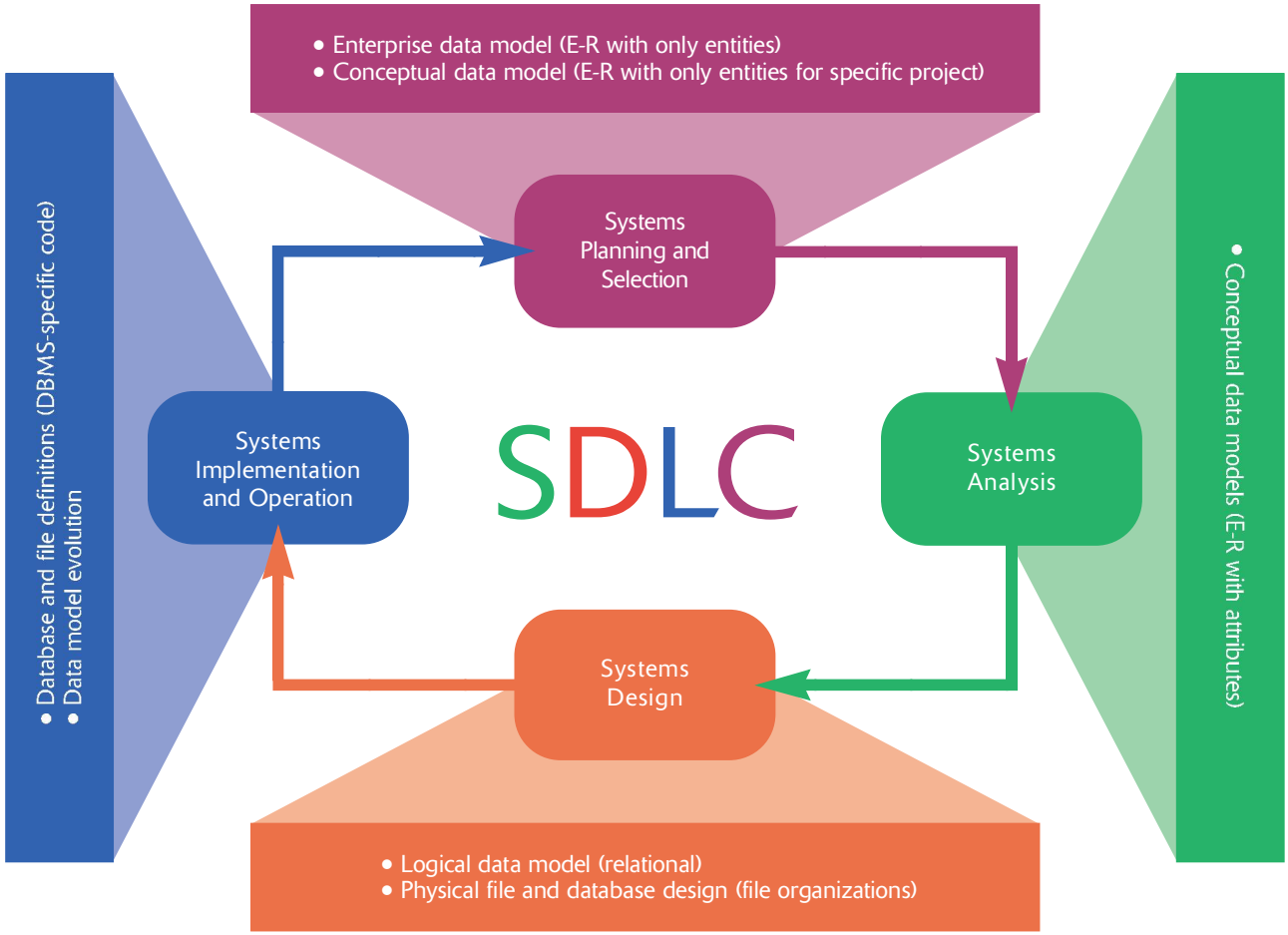


FIGURE 9-2
Relationship between data modeling and the systems development life cycle.

of where and when data are entered, retrieved, deleted, and updated; expectations for response time and data integrity; and descriptions of the file and database technologies to be used. These inputs allow you to make key physical database design decisions, including the following:

1. Choosing the storage format (called *data type*) for each attribute from the logical database model; the format is chosen to minimize storage space and to maximize data quality. Data type involves choosing length, coding scheme, number of decimal places, minimum and maximum values, and potentially many other parameters for each attribute.
2. Grouping attributes from the logical database model into physical records (in general, this is called *selecting a stored record*, or *data structure*).
3. Arranging related records in secondary memory (hard disks and magnetic tapes) so that individual and groups of records can be stored, retrieved, and updated rapidly (called *file organizations*). You should also consider protecting data and recovering data after errors are found.
4. Selecting media and structures for storing data to make access more efficient. The choice of media affects the utility of different file organizations. The primary structure used today to make access to data more rapid is key indexes, on unique and nonunique keys.

In this chapter we show how to do each of the logical database design steps and discuss factors to consider in making each physical file and database design decision.

Deliverables and Outcomes

During logical database design, you must account for every data element on a system input or output—form or report—and on the E-R model. Each data element (like customer name, product description, or purchase price) must be a piece of raw data kept in the system's database, or in the case of a data element on a system output, the element can be derived from data in the database. Figure 9-3 illustrates the outcomes from the four-step logical database design process. Figures 9-3A and 9-3B (step 1) contain two sample system outputs for a customer order processing system at Pine Valley Furniture. A description of the associated database requirements, in the form of what we call *normalized relations*, is listed below each output diagram. Each relation (think of a relation as a table with rows and columns) is named, and its attributes (columns) are listed within parentheses. The **primary key** attribute—that attribute whose value is unique across all occurrences of the relation—is indicated by an underline, and an attribute of a relation that is the primary key of another relation is indicated by a dashed underline.

Primary key
An attribute whose value is unique across all occurrences of a relation.

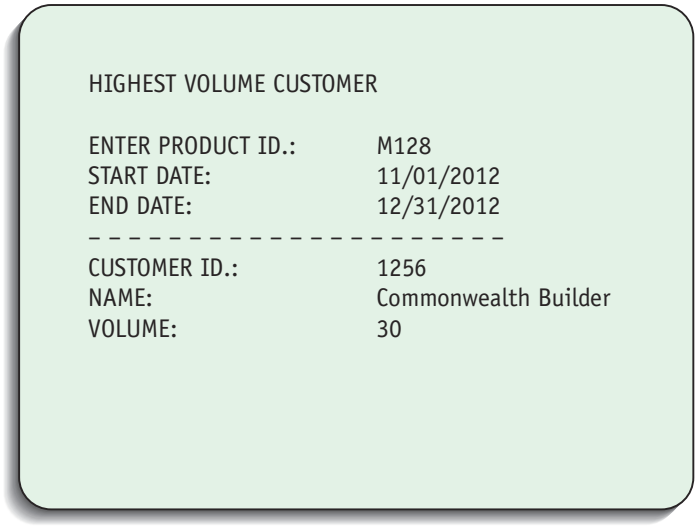
In Figure 9-3A data are shown about customers, products, and the customer orders and associated line items for products. Each of the attributes of each relation either appears in the display or is needed to link related relations. For example, because an order is for some customer, an attribute of ORDER is the associated Customer_ID. The data for the display in Figure 9-3B are more complex. A backlogged product on an order occurs when the amount ordered (Order_Quantity) is less than the amount shipped (Ship_Quantity) for invoices associated with an order. The query refers to only a specified time period, so the Order_Date is needed. The INVOICE Order_Number links invoices with the associated order.

Figure 9-3C (step 2) shows the result of integrating these two separate sets of normalized relations. Figure 9-3D (step 3) shows an E-R diagram for a customer order processing application that might be developed during conceptual data modeling along with equivalent normalized relations. Figure 9-3E (step 4) shows a set of normalized relations that would result from reconciling the logical database designs of Figures 9-3C and 9-3D. Normalized relations like those in Figure 9-3E are the primary deliverable from logical database design.

Finally, Figure 9-3F shows the E-R diagram drawn in Microsoft Visio. Visio actually shows the tables and relationships between the tables from the normalized relations. Thus, the associative entities, LINE ITEM and SHIPMENT, are shown as entities on the Visio diagram; we do not place relationship names on either side of these entities on the Visio diagram because these represent associative entities. Visio also shows for these entities the primary keys of the associated ORDER, INVOICE, and PRODUCT entities. Also, note that the lines for the Places and Bills relationships are dashed. This Visio notation indicates that ORDER and INVOICE have their own primary keys that do not include the primary keys of CUSTOMER and ORDER, respectively (what Visio calls non-identifying relationships). Because LINE ITEM and SHIPMENT both include in their primary keys the primary keys of other entities (which is common for associative entities), the relationships around LINE ITEM and SHIPMENT are identifying, and hence the relationship lines are solid.

It is important to remember that relations do not correspond to computer files. In physical database design, you translate the relations from logical database design into specifications for computer files. For most information

A

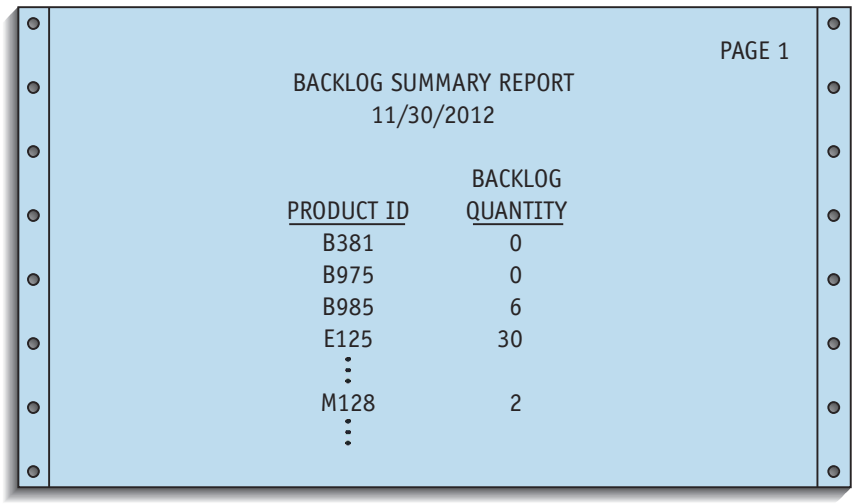


This inquiry screen shows the customer with the largest volume total sales of a specified product during an indicated time period.

Relations:

CUSTOMER(Customer_ID,Name)
ORDER(Order_Number,Customer_ID,Order_Date)
PRODUCT(Product_ID)
LINE ITEM(Order_Number,Product_ID,Order_Quantity)

B



This report shows the unit volume of each product that has been ordered less than amount shipped through the specified date.

Relations:

PRODUCT(Product_ID)
LINE ITEM(Product_ID,Order_Number,Order_Quantity)
ORDER(Order_Number,Order_Date)
SHIPMENT(Product_ID,Invoice_Number,Ship_Quantity)
INVOICE(Invoice_Number,Invoice_Date,Order_Number)

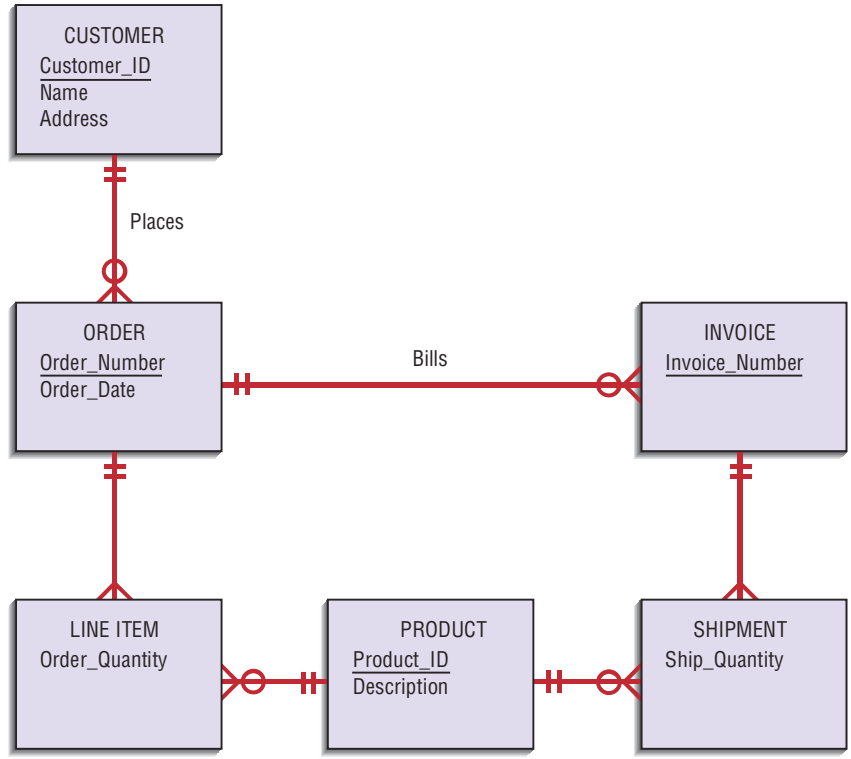
C

CUSTOMER(Customer_ID,Name)
PRODUCT(Product_ID)
INVOICE(Invoice_Number,Invoice_Date,Order_Number)
ORDER(Order_Number,Customer_ID,Order_Date)
LINE ITEM(Order_Number,Product_ID,Order_Quantity)
SHIPMENT(Product_ID,Invoice_Number,Ship_Quantity)

FIGURE 9-3
Simple example of logical data modeling: (A) Highest-volume customer query screen, (B) Backlog summary report, (C) Integrated set of relations, (D) Conceptual data model and transformed relations, (E) Final set of normalized relations, (F) Microsoft Visio E-R diagram.

FIGURE 9-3
(*continued*)

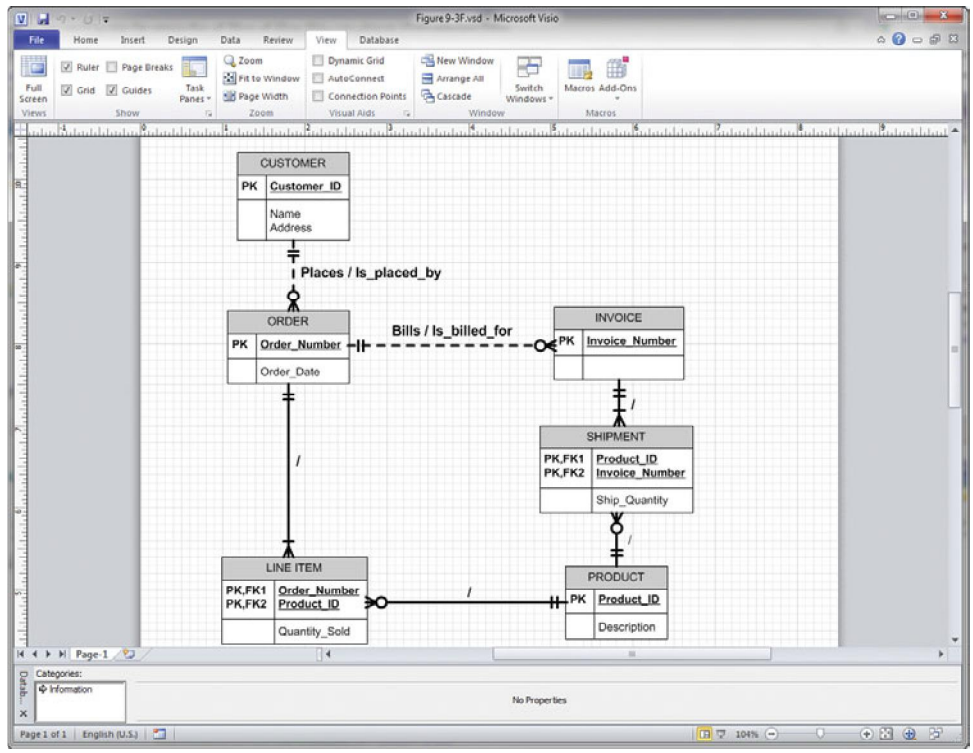
D



E

CUSTOMER(Customer_ID,Name,Address)
PRODUCT(Product_ID,Description)
ORDER(Order_Number,Customer_ID,Order_Date)
LINE ITEM(Order_Number,Product_ID,Order_Quantity)
INVOICE(Invoice_Number,Order_Number,Invoice_Date)
SHIPMENT(Invoice_Number,Product_ID,Ship_Quantity)

F



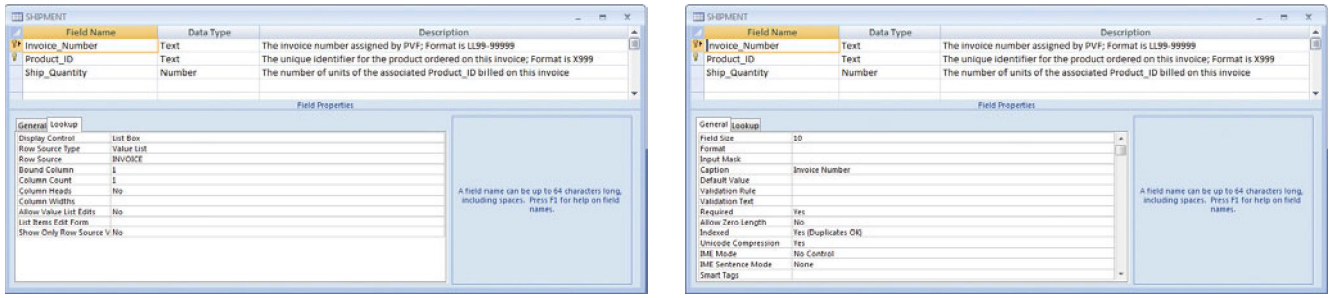


FIGURE 9-4
Definition of SHIPMENT table in Microsoft Access: (A) Table with invoice_number properties, (B) Invoice_number lookup properties.

systems, these files will be tables in a relational database. These specifications are sufficient for programmers and database analysts to code the definitions of the database. The coding, done during systems implementation, is written in special database definition and processing languages, such as Structured Query Language (SQL), or by filling in table definition forms, such as with Microsoft Access. Figure 9-4 shows a possible definition for the SHIPMENT relation from Figure 9-3E using Microsoft Access. This display of the SHIPMENT table definition illustrates choices made for several physical database design decisions.

- All three attributes from the SHIPMENT relation, and no attributes from other relations, have been grouped together to form the fields of the SHIPMENT table.
- The Invoice_Number field has been given a data type of Text, with a maximum length of 10 characters.
- The Invoice_Number field is required because it is part of the primary key for the SHIPMENT table (the value that makes every row of the SHIPMENT table unique is a combination of Invoice_Number and Product_ID).
- An index is defined for the Invoice_Number field, but because there may be several rows in the SHIPMENT table for the same invoice (different products on the same invoice), duplicate index values are allowed (so Invoice_Number is what we will call a *secondary key*).
- The Invoice_Number, because it references the Invoice_Number from the INVOICE table, is defined as a Lookup to the first column (Invoice_Number) of the INVOICE table; in this way, all values that are placed in the Invoice_Number field of the SHIPMENT table must correspond to a previously entered invoice.

Many other physical database design decisions were made for the SHIPMENT table, but they are not apparent on the display in Figure 9-4. Further, this table is only one table in the PVF Order Entry database, and other tables and structures for this database are not illustrated in this figure.

Relational Database Model

Many different database models are in use and are the basis for database technologies. Although hierarchical and network models have been popular in the past, they are not often used today for new information systems. Object-oriented database models are emerging but are still not common. The vast majority of information systems today use the relational database model.

FIGURE 9-5
EMPLOYEE1 relation with sample data.

EMPLOYEE1

Emp_ID	Name	Dept	Salary
100	Margaret Simpson	Marketing	42,000
140	Allen Beeton	Accounting	39,000
110	Chris Lucero	Info Systems	41,500
190	Lorenzo Davis	Finance	38,000
150	Susan Martin	Marketing	38,500

Relational database model
Data represented as a set of related tables or relations.

Relation
A named, two-dimensional table of data. Each relation consists of a set of named columns and an arbitrary number of unnamed rows.

The **relational database model** represents data in the form of related tables or relations. A **relation** is a named, two-dimensional table of data. Each relation (or table) consists of a set of named columns and an arbitrary number of unnamed rows. Each column in a relation corresponds to an attribute of that relation. Each row of a relation corresponds to a record that contains data values for an entity.

Figure 9-5 shows an example of a relation named EMPLOYEE1. This relation contains the following attributes describing employees: Emp_ID, Name, Dept, and Salary. The table contains five sample rows, corresponding to five employees.

You can express the structure of a relation by a shorthand notation in which the name of the relation is followed (in parentheses) by the names of the attributes in the relation. The identifier attribute (called the *primary key* of the relation) is underlined. For example, you would express EMPLOYEE1 as follows:

Employee (Emp_ID, Name, Dept, Salary)

Not all tables are relations. Relations have several properties that distinguish them from nonrelational tables:

1. Entries in cells are simple. An entry at the intersection of each row and column has a single value.
2. Entries in columns are from the same set of values.
3. Each row is unique. Uniqueness is guaranteed because the relation has a nonempty primary key value.
4. The sequence of columns can be interchanged without changing the meaning or use of the relation.
5. The rows may be interchanged or stored in any sequence.

Well-Structured Relations

Well-structured relation (or table)
A relation that contains a minimum amount of redundancy and allows users to insert, modify, and delete the rows without errors or inconsistencies.

What constitutes a **well-structured relation** (or **table**)? Intuitively, a well-structured relation contains a minimum amount of redundancy and allows users to insert, modify, and delete the rows in a table without errors or inconsistencies. EMPLOYEE1 (Figure 9-5) is such a relation. Each row of the table contains data describing one employee, and any modification to an employee's data (such as a change in salary) is confined to one row of the table.

In contrast, EMPLOYEE2 (Figure 9-6) contains data about employees and the courses they have completed. Each row in this table is unique for the combination of Emp_ID and Course, which becomes the primary key for the table. It is not a well-structured relation, however. If you examine the sample data in the table, you notice a considerable amount of redundancy. For example, the Emp_ID, Name, Dept, and Salary values appear in two separate rows for employees 100, 110, and 150. Consequently, if the salary for employee 100 changes, we must record this fact in two rows (or more, for some employees).

The problem with this relation is that it contains data about two entities: EMPLOYEE and COURSE. You will learn to use principles of normalization to divide EMPLOYEE2 into two relations. One of the resulting relations is

EMPLOYEE2

Emp_ID	Name	Dept	Salary	Course	Date_Completed
100	Margaret Simpson	Marketing	42,000	SPSS	6/19/2012
100	Margaret Simpson	Marketing	42,000	Surveys	10/7/2012
140	Alan Beeton	Accounting	39,000	Tax Acc	12/8/2012
110	Chris Lucero	Info Systems	41,500	SPSS	1/12/2012
110	Chris Lucero	Info Systems	41,500	C++	4/22/2012
190	Lorenzo Davis	Finance	38,000	Investments	5/7/2012
150	Susan Martin	Marketing	38,500	SPSS	6/19/2012
150	Susan Martin	Marketing	38,500	TQM	8/12/2012

FIGURE 9-6
Relation with redundancy.

EMPLOYEE1 (Figure 9-5). The other we will call EMP COURSE, which appears with sample data in Figure 9-7. The primary key of this relation is the combination of Emp_ID and Course (we emphasize this by underlining the column names for these attributes).

Normalization

We have presented an intuitive discussion of well-structured relations, however, we need rules and a process for designing them. **Normalization** is a process for converting complex data structures into simple, stable data structures. For example, we used the principles of normalization to convert the EMPLOYEE2 table with its redundancy to EMPLOYEE1 (Figure 9-5) and EMP COURSE (Figure 9-7).

Normalization
The process of converting complex data structures into simple, stable data structures.

Rules of Normalization

Normalization is based on well-accepted principles and rules. The many normalization rules, are too numerous to cover in this text (see Hoffer, Ramesh, and Topi [2011] for more complete coverage). Besides the five properties of relations outlined previously, two other rules are frequently used.

1. *Second normal form (2NF).* Each nonprimary key attribute is identified by the whole key (what we call *full functional dependency*).
2. *Third normal form (3NF).* Nonprimary key attributes do not depend on each other (what we call *no transitive dependencies*).

The result of normalization is that every nonprimary key attribute depends upon the whole primary key and nothing but the primary key. We discuss second and third normal form in more detail next.

EMP COURSE

Emp_ID	Course	Date_ Completed
100	SPSS	6/19/2012
100	Surveys	10/7/2012
140	Tax Acc	12/8/2012
110	SPSS	1/22/2012
110	C++	4/22/2012
190	Investments	5/7/2012
150	SPSS	6/19/2012
150	TQM	8/12/2012

FIGURE 9-7
EMP COURSE relation.

Functional Dependence and Primary Keys

Functional dependency
A particular relationship between two attributes. For a given relation, attribute B is functionally dependent on attribute A if, for every valid value of A, that value of A uniquely determines the value of B. The functional dependence of B on A is represented by A→B.

Normalization is based on the analysis of functional dependence. A **functional dependency** is a particular relationship between two attributes. In a given relation, attribute B is functionally dependent on attribute A if, for every valid value of A, that value of A uniquely determines the value of B. The functional dependence of B on A is represented by an arrow, as follows: A→B (e.g., Emp_ID→Name in the relation of Figure 9-5). Functional dependence does not imply mathematical dependence—that the value of one attribute may be computed from the value of another attribute; rather, functional dependence of B on A means that there can be only one value of B for each value of A. Thus, for a given Emp_ID value, only one Name value can be associated with it; the value of Name, however, cannot be derived from the value of Emp_ID. Other examples of functional dependencies from Figure 9-3B are in ORDER, Order_Number→Order_Date, and in INVOICE, Invoice_Number→Invoice_Date and Order_Number.

An attribute may be functionally dependent on two (or more) attributes, rather than on a single attribute. For example, consider the relation EMP COURSE (Emp_ID, Course, Date_Completed) shown in Figure 9-7. We represent the functional dependency in this relation as follows: Emp_ID, Course Date_Completed. In this case, Date_Completed cannot be determined by either Emp_ID or Course alone, because Date_Completed is a characteristic of an employee taking a course.

You should be aware that the instances (or sample data) in a relation do not prove that a functional dependency exists. Only knowledge of the problem domain, obtained from a thorough requirements analysis, is a reliable method for identifying a functional dependency. However, you can use sample data to demonstrate that a functional dependency does not exist between two or more attributes. For example, consider the sample data in the relation EXAMPLE (A, B, C, D) shown in Figure 9-8. The sample data in this relation prove that attribute B is not functionally dependent on attribute A, because A does not uniquely determine B (two rows with the same value of A have different values of B).

Second Normal Form

Second normal form (2NF)
A relation for which every nonprimary key attribute is functionally dependent on the whole primary key.

A relation is in **second normal form (2NF)** if every nonprimary key attribute is functionally dependent on the whole primary key. Thus, no nonprimary key attribute is functionally dependent on a part, but not all, of the primary key. Second normal form is satisfied if any one of the following conditions apply:

1. The primary key consists of only one attribute (such as the attribute Emp_ID in relation EMPLOYEE1).
2. No nonprimary key attributes exist in the relation.
3. Every nonprimary key attribute is functionally dependent on the full set of primary key attributes.

FIGURE 9-8
EXAMPLE relation.

EXAMPLE

A	B	C	D
X	U	X	Y
Y	X	Z	X
Z	Y	Y	Y
Y	Z	W	Z

EMPLOYEE2 (Figure 9-6) is an example of a relation that is not in second normal form. The shorthand notation for this relation is:

EMPLOYEE2(Emp_ID, Name, Dept, Salary, Course, Date_Completed)

The functional dependencies in this relation are the following:

Emp_ID→Name, Dept, Salary
Emp_ID, Course→Date_Completed

The primary key for this relation is the composite key Emp_ID, Course. Therefore, the nonprimary key attributes Name, Dept, and Salary are functionally dependent on only Emp_ID but not on Course. EMPLOYEE2 has redundancy, which results in problems when the table is updated.

To convert a relation to second normal form, you decompose the relation into new relations using the attributes, called *determinants*, that determine other attributes; the determinants are the primary keys of these relations. EMPLOYEE2 is decomposed into the following two relations:

1. EMPLOYEE1(Emp_ID, Name, Dept, Salary): This relation satisfies the first second normal form condition (sample data shown in Figure 9-5).
2. EMP COURSE(Emp_ID, Course, Date_Completed): This relation satisfies second normal form condition three (sample data appear in Figure 9-7).

Third Normal Form

A relation is in **third normal form (3NF)** if it is in second normal form with no functional dependencies between two (or more) nonprimary key attributes (a functional dependency between nonprimary key attributes is also called a *transitive dependency*). For example, consider the relation SALES(Customer_ID, Customer_Name, Salesperson, Region) (sample data shown in Figure 9-9A).

Third normal form (3NF)
A relation that is in second normal form and has no functional (transitive) dependencies between two (or more) nonprimary key attributes.

The following functional dependencies exist in the SALES relation:

1. Customer_ID→Customer_Name, Salesperson, Region (Customer_ID is the primary key.)
2. Salesperson→Region (Each salesperson is assigned to a unique region.)

Notice that SALES is in second normal form because the primary key consists of a single attribute (Customer_ID). However, Region is functionally dependent on Salesperson, and Salesperson is functionally dependent on Customer_ID. As a result, data maintenance problems arise in SALES.

1. A new salesperson (Robinson) assigned to the North region cannot be entered until a customer has been assigned to that salesperson (because a value for Customer_ID must be provided to insert a row in the table).
2. If customer number 6837 is deleted from the table, we lose the information that salesperson Hernandez is assigned to the East region.
3. If salesperson Smith is reassigned to the East region, several rows must be changed to reflect that fact (two rows are shown in Figure 9-9A).

These problems can be avoided by decomposing SALES into the two relations, based on the two determinants, shown in Figure 9-9(B). These relations are the following:

SALES1(Customer_ID, Customer_Name, Salesperson)
SPERSON(Salesperson, Region)

Foreign key
An attribute that appears as a nonprimary key attribute in one relation and as a primary key attribute (or part of a primary key) in another relation.

Note that Salesperson is the primary key in SPERSON. Salesperson is also a foreign key in SALES1. A **foreign key** is an attribute that appears as a nonprimary key attribute in one relation (such as SALES1) and as a primary key

FIGURE 9-9
Removing transitive dependencies: (A) Relation with transitive dependency, (B) Relations in 3NF.

A

SALES

Customer_ID	Customer_Name	Salesperson	Region
8023	Anderson	Smith	South
9167	Bancroft	Hicks	West
7924	Hobbs	Smith	South
6837	Tucker	Hernandez	East
8596	Eckersley	Hicks	West
7018	Arnold	Faulb	North

B

SALES1

Customer_ID	Customer_Name	Salesperson
8023	Anderson	Smith
9167	Bancroft	Hicks
7924	Hobbs	Smith
6837	Tucker	Hernandez
8596	Eckersley	Hicks
7018	Arnold	Faulb

SPERSON

Salesperson	Region
Smith	South
Hicks	West
Hernandez	East
Faulb	North

attribute (or part of a primary key) in another relation. You designate a foreign key by using a dashed underline.

Referential integrity
An integrity constraint specifying that the value (or existence) of an attribute in one relation depends on the value (or existence) of the same attribute in another relation.

A foreign key must satisfy **referential integrity,** which specifies that the value of an attribute in one relation depends on the value of the same attribute in another relation. Thus, in Figure 9-9B, the value of Salesperson in each row of table SALES1 is limited to only the current values of Salesperson in the SPERSON table. Referential integrity is one of the most important principles of the relational model.

Transforming E-R Diagrams into Relations

Normalization produces a set of well-structured relations that contains all of the data mentioned in system inputs and outputs developed in human interface design. Because these specific information requirements may not represent all future information needs, the E-R diagram you developed in conceptual data modeling is another source of insight into possible data requirements for a new application system. To compare the conceptual data model and the normalized relations developed so far, your E-R diagram must be transformed into relational notation, normalized, and then merged with the existing normalized relations.

Transforming an E-R diagram into normalized relations and then merging all the relations into one final, consolidated set of relations can be accomplished in four steps. These steps are summarized briefly here, and then steps 1, 2, and 4 are discussed in detail in subsequent sections of this chapter.

1. *Represent entities.* Each entity type in the E-R diagram becomes a relation. The identifier of the entity type becomes the primary key of the relation, and other attributes of the entity type become nonprimary key attributes of the relation.
2. *Represent relationships.* Each relationship in an E-R diagram must be represented in the relational database design. How we represent a relationship depends on its nature. For example, in some cases we represent a relationship by making the primary key of one relation a foreign key of another relation. In other cases, we create a separate relation to represent a relationship.

3. *Normalize the relations.* The relations created in steps 1 and 2 may have unnecessary redundancy. So, we need to normalize these relations to make them well structured.
4. *Merge the relations.* So far in database design we have created various relations from both a bottom-up normalization of user views and from transforming one or more E-R diagrams into sets of relations. Across these different sets of relations, redundant relations (two or more relations that describe the same entity type) may need to be merged and renormalized to remove the redundancy.

Represent Entities

Each regular entity type in an E-R diagram is transformed into a relation. The identifier of the entity type becomes the primary key of the corresponding relation. Each nonkey attribute of the entity type becomes a nonkey attribute of the relation. You should check to make sure that the primary key satisfies the following two properties:

1. The value of the key must uniquely identify every row in the relation.
2. The key should be nonredundant; that is, no attribute in the key can be deleted without destroying its unique identification.

Some entities may have keys that include the primary keys of other entities. For example, an EMPLOYEE DEPENDENT may have a Name for each dependent, but, to form the primary key for this entity, you must include the Employee_ID attribute from the associated EMPLOYEE entity. Such an entity whose primary key depends upon the primary key of another entity is called a *weak entity.*

Representation of an entity as a relation is straightforward. Figure 9-10A shows the CUSTOMER entity type for Pine Valley Furniture. The corresponding CUSTOMER relation is represented as follows:

CUSTOMER(Customer_ID, Name, Address, City_State_ZIP, Discount)

In this notation, the entity type label is translated into a relation name. The identifier of the entity type is listed first and underlined. All nonkey attributes are listed after the primary key. This relation is shown as a table with sample data in Figure 9-10B.

A

FIGURE 9-10
Transforming an entity type to a relation: (A) E-R diagram, (B) Relation.

B

CUSTOMER

Customer_ID	Name	Address	City_State_Zip	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

Represent Relationships

The procedure for representing relationships depends on both the degree of the relationship—unary, binary, ternary—and the cardinalities of the relationship.

Binary 1:N and 1:1 Relationships A binary one-to-many (1:*N*) relationship in an E-R diagram is represented by adding the primary key attribute (or attributes) of the entity on the one side of the relationship as a foreign key in the relation that is on the many side of the relationship.

Figure 9-11A, an example of this rule, shows the Places relationship (1:*N*) linking CUSTOMER and ORDER at Pine Valley Furniture. Two relations, CUSTOMER and ORDER, were formed from the respective entity types (see Figure 9-11B). Customer_ID, which is the primary key of CUSTOMER (on the one side of the relationship) is added as a foreign key to ORDER (on the many side of the relationship).

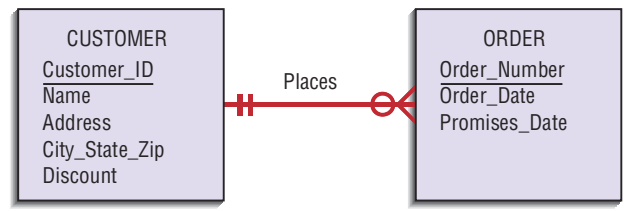
One special case under this rule was mentioned in the previous section. If the entity on the many side needs the key of the entity on the one side as part of its primary key (this is a so-called weak entity), then this attribute is added not as a nonkey but as part of the primary key.

For a binary or unary one-to-one (1:1) relationship between the two entities A and B (for a unary relationship, A and B would be the same entity type), the relationship can be represented by any of the following choices:

1. Adding the primary key of A as a foreign key of B
2. Adding the primary key of B as a foreign key of A
3. Both of the above

Binary and Higher-Degree M:N Relationships Suppose that a binary many-to-many (*M:N*) relationship (or associative entity) exists between two entity types A and B. For such a relationship, we create a separate relation C. The primary key of this relation is a composite key consisting of the primary key for each of the two entities in the relationship.

FIGURE 9-11
Representing a 1:*N* relationship: (A) E-R diagram, (B) Relations.



A

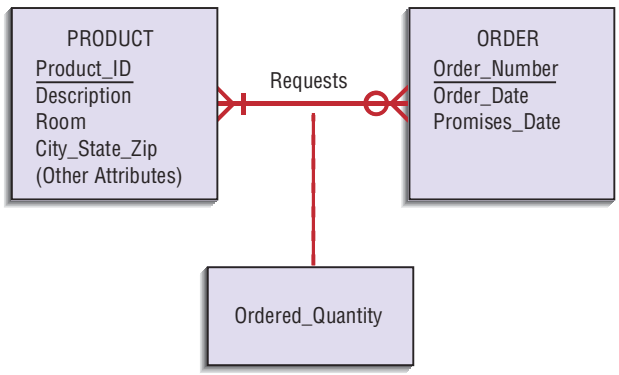
CUSTOMER

Customer_ID	Name	Address	City_State_ZIP	Discount
1273	Contemporary Designs	123 Oak St.	Austin, TX 28384	5%
6390	Casual Corner	18 Hoosier Dr.	Bloomington, IN 45821	3%

ORDER

Order_Number	Order_Date	Promised_Date	Customer_ID
57194	3/15/12	3/28/12	6390
63725	3/17/12	4/01/12	1273
80149	3/14/12	3/24/12	6390

B



A

ORDER

Order_Number	Order_Date	Promised_Date
61384	2/17/2012	3/01/2012
62009	2/13/2012	2/27/2012
62807	2/15/2012	3/01/2012

ORDER LINE

Order_Number	Product_ID	Quantity_ Ordered
61384	M128	2
61384	A261	1

PRODUCT

Product_ID	Description	(Other Attributes)
M128	Bookcase	—
A261	Wall unit	—
R149	Cabinet	—

B

FIGURE 9-12
Representing an *M:N* relationship: (A) E-R diagram, (B) Relations.

Any nonkey attributes that are associated with the *M:N* relationship are included with the relation C.

Figure 9-12A, an example of this rule, shows the Requests relationship (*M:N*) between the entity types ORDER and PRODUCT for Pine Valley Furniture. Figure 9-12B shows the three relations (ORDER, ORDER LINE, and PRODUCT) that are formed from the entity types and the Requests relationship. A relation (called ORDER LINE in Figure 9-12B) is created for the Requests relationship. The primary key of ORDER LINE is the combination (Order_Number, Product_ID), which consists of the respective primary keys of ORDER and PRODUCT. The nonkey attribute Quantity_Ordered also appears in ORDER LINE.

Occasionally, the relation created from an *M:N* relationship requires a primary key that includes more than just the primary keys from the two related relations. Consider, for example, the following situation:

In this case, Date must be part of the key for the SHIPMENT relation to uniquely distinguish each row of the SHIPMENT table, as follows:

SHIPMENT(Customer_ID, Vendor_ID, Date, Amount)

If each shipment has a separate nonintelligent key (a system-assigned unique value that has no business meaning; e.g., order number, customer number), say a shipment number, then Date becomes a nonkey and Customer_ID and Vendor_ID become foreign keys, as follows:

SHIPMENT(Shipment_Number, Customer_ID, Vendor_ID, Date, Amount)

In some cases, a relationship may be found among three or more entities. In such cases, we create a separate relation that has as a primary key the composite of the primary keys of each of the participating entities (plus any necessary additional key elements). This rule is a simple generalization of the rule for a binary *M:N* relationship.

Unary Relationships To review, a unary relationship is a relationship between the instances of a single entity type, which are also called *recursive relationships*. Figure 9-13 shows two common examples. Figure 9-13A shows a one-to-many relationship named Manages that associates employees with another employee who is their manager. Figure 9-13B shows a many-to-many relationship that associates certain items with their component items. This relationship is called a *bill-of-materials structure*.

For a unary 1:*N* relationship, the entity type (such as EMPLOYEE) is modeled as a relation. The primary key of that relation is the same as for the entity type. Then a foreign key is added to the relation that references the primary key values. A **recursive foreign key** is a foreign key in a relation that references the primary key values of that same relation. We can represent the relationship in Figure 9-13A as follows:

Recursive foreign key
A foreign key in a relation that references the primary key values of that same relation.

EMPLOYEE(Emp_ID, Name, Birthdate, Manager_ID)

In this relation, Manager_ID is a recursive foreign key that takes its values from the same set of worker identification numbers as Emp_ID.

For a unary *M:N* relationship, we model the entity type as one relation. Then we create a separate relation to represent the *M:N* relationship. The primary key of this new relation is a composite key that consists of two attributes (which need not have the same name) that both take their values from the same primary key. Any attribute associated with the relationship (such as Quantity in Figure 9-13B) is included as a nonkey attribute in this new relation. We can express the result for Figure 9-13B as follows:

ITEM(Item_Number, Name, Cost)
ITEM-BILL(Item_Number, Component_Number, Quantity)

Summary of Transforming E-R Diagrams to Relations

We have now described how to transform E-R diagrams to relations. Table 9-1 lists the rules discussed in this section for transforming entity-relationship diagrams into equivalent relations. After this transformation, you should check the resulting relations to determine whether they are in third normal form and, if necessary, perform normalization as described earlier in the chapter.

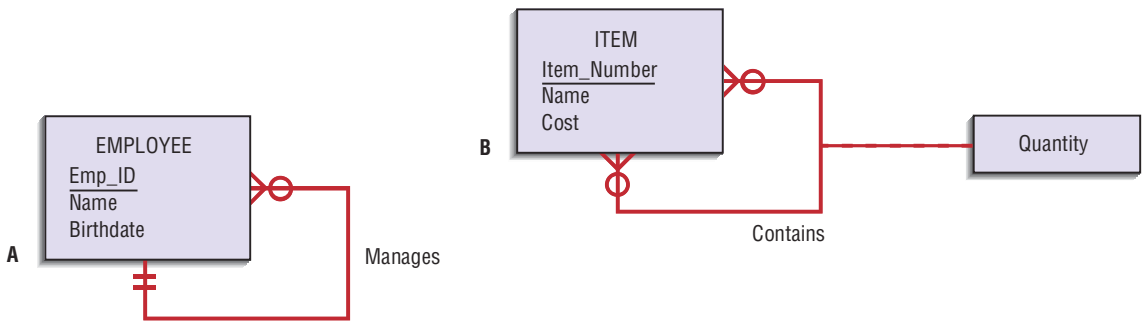


FIGURE 9-13
Two unary relations: (A) EMPLOYEE with manages relationship (1:*N*), (B) Bill-of-materials structure (*M:N*).

TABLE 9-1: E-R to Relational Transformation

E-R Structure	Relational Representation
Regular entity	Create a relation with primary key and nonkey attributes.
Weak entity	Create a relation with a composite primary key (which includes the primary key of the entity on which this weak entity depends) and nonkey attributes.
Binary or unary 1:1 relationship	Place the primary key of either entity in the relation for the other entity or do it for both entities.
Binary 1:*N* relationship	Place the primary key of the entity on the one side of the relationship as a foreign key in the relation for the entity on the many side.
Binary or unary *M*:*N* relationship or associative entity	Create a relation with a composite primary key using the primary keys of the related entities, plus any nonkey attributes of the relationship or associative entity.
Binary or unary *M*:*N* relationship or associative entity with additional key(s)	Create a relation with a composite primary key using the primary keys of the related entities and additional primary key attributes associated with the relationship or associative entity, plus any nonkey attributes of the relationship or associative entity.
Binary or unary *M*:*N* relationship or associative entity with its own key	Create a relation with the primary key associated with the relationship or associative entity, plus any nonkey attributes of the relationship or associative entity and the primary keys of the related entities (as nonkey attributes).

Merging Relations

As part of the logical database design, normalized relations likely have been created from a number of separate E-R diagrams and various user interfaces. Some of the relations may be redundant—they may refer to the same entities. If so, you should merge those relations to remove the redundancy. This section describes merging relations, or *view integration*, which is the last step in logical database design and prior to physical file and database design.

An Example of Merging Relations

Suppose that modeling a user interface or transforming an E-R diagram results in the following 3NF relation:

EMPLOYEE1(Emp_ID, Name, Address, Phone)

Modeling a second user interface might result in the following relation:

EMPLOYEE2(Emp_ID, Name, Address, Jobcode, Number_of_Years)

Because these two relations have the same primary key (Emp_ID) and describe the same entity, they should be merged into one relation. The result of merging the relations is the following relation:

EMPLOYEE(Emp_ID, Name, Address, Phone, Jobcode, Number_of_Years)

Notice that an attribute that appears in both relations (such as Name in this example) appears only once in the merged relation.

View Integration Problems

When integrating relations, you must understand the meaning of the data and must be prepared to resolve any problems that may arise in that process. In this section, we describe and illustrate three problems that arise in view integration: synonyms, homonyms, and dependencies between nonkeys.

Synonyms
Two different names that are used for the same attribute.

Synonyms In some situations, two or more attributes may have different names but the same meaning, as when they describe the same characteristic of an entity. Such attributes are called **synonyms.** For example, Emp_ID and Employee_Number may be synonyms.

When merging the relations that contain synonyms, you should obtain, if possible, agreement from users on a single standardized name for the attribute and eliminate the other synonym. Another alternative is to choose a third name to replace the synonyms. For example, consider the following relations:

STUDENT1(Student_ID, Name)
STUDENT2(Matriculation_Number, Name, Address)

In this case, the analyst recognizes that both the Student_ID and the Matriculation_Number are synonyms for a person's social security number and are identical attributes. One possible resolution would be to standardize on one of the two attribute names, such as Student_ID. Another option is to use a new attribute name, such as SSN, to replace both synonyms. Assuming the latter approach, merging the two relations would produce the following result:

STUDENT(SSN, Name, Address)

Homonym
A single attribute name that is used for two or more different attributes.

Homonyms In other situations, a single attribute name, called a **homonym,** may have more than one meaning or describe more than one characteristic. For example, the term *account* might refer to a bank's checking account, savings account, loan account, or other type of account; therefore, *account* refers to different data, depending on how it is used.

You should be on the lookout for homonyms when merging relations. Consider the following example:

STUDENT1(Student_ID, Name, Address)
STUDENT2(Student_ID, Name, Phone_Number, Address)

In discussions with users, the systems analyst may discover that the attribute Address in STUDENT1 refers to a student's campus address, whereas in STUDENT2 the same attribute refers to a student's home address. To resolve this conflict, we would probably need to create new attribute names and the merged relation would become:

STUDENT(Student_ID, Name, Phone_Number, Campus_Address,
Permanent_Address)

Dependencies between Nonkeys When two 3NF relations are merged to form a single relation, dependencies between nonkeys may result. For example, consider the following two relations:

STUDENT1(Student_ID, Major)
STUDENT2(Student_ID, Adviser)

Because STUDENT1 and STUDENT2 have the same primary key, the two relations may be merged:

STUDENT(Student_ID, Major, Adviser)

However, suppose that each major has exactly one adviser. In this case, Adviser is functionally dependent on Major:

Major→Adviser

If the previous dependency exists, then STUDENT is in 2NF but not 3NF, because it contains a functional dependency between nonkeys. The analyst can create 3NF relations by creating two relations with Major as a foreign key in STUDENT:

STUDENT(Student_ID, Major)
MAJOR ADVISER(Major, Adviser)

Logical Database Design for Hoosier Burger

In Chapter 7 we developed an E-R diagram for a new inventory control system at Hoosier Burger (Figure 9-14 repeats the diagram from Chapter 7). In this section we show how this E-R model is translated into normalized relations and how to normalize and then merge the relations for a new report with the relations from the E-R model.

In this E-R model, four entities exist independently of other entities: SALE, PRODUCT, INVOICE, and INVENTORY ITEM. Given the attributes shown in Figure 9-14, we can represent these entities in the following four relations:

SALE(Receipt_Number, Sale_Date)
PRODUCT(Product_ID, Product_Description)
INVOICE(Vendor_ID, Invoice_Number, Invoice_Date, Paid?)
INVENTORY ITEM(Item_Number, Item_Description, Quantity_in_Stock, Minimum_Order_Quantity, Type_of_Item)

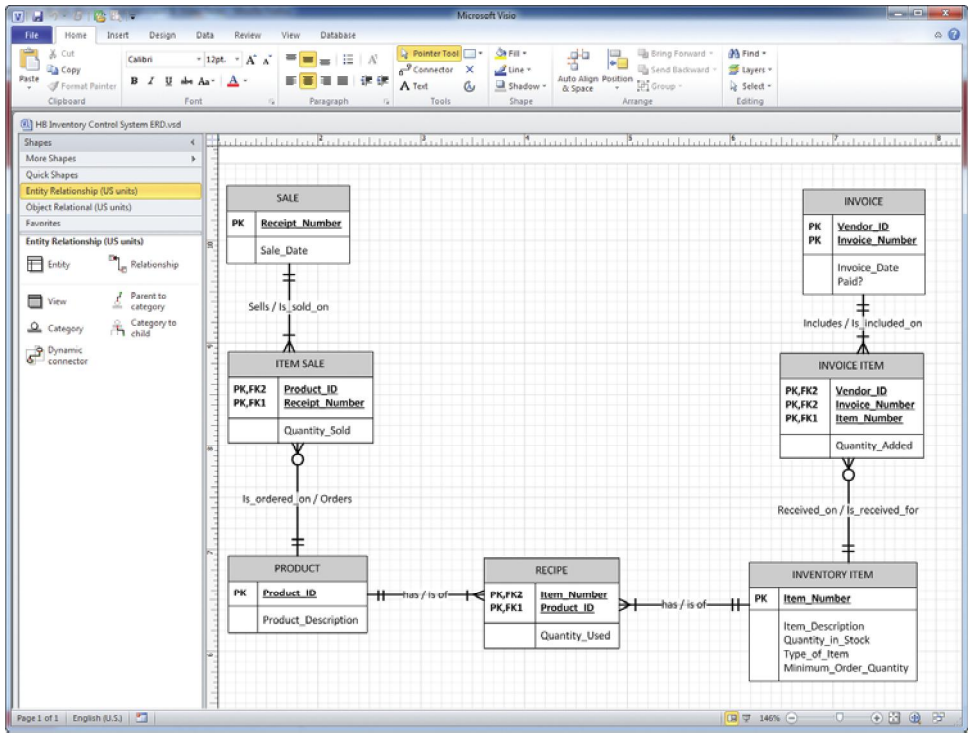


FIGURE 9-14
Final E-R diagram for Hoosier Burger's inventory control system.

The entities ITEM SALE and INVOICE ITEM as well as the associative entity RECIPE each have composite primary keys taken from the entities to which they relate, so we can represent these three entities in the following three relations:

ITEM SALE(Receipt_Number, Product_ID, Quantity_Sold)
INVOICE ITEM(Vendor_ID, Invoice_Number, Item_Number, Quantity_ Added)
RECIPE(Product_ID, Item_Number, Quantity_Used)

Because no many-to-many, one-to-one, or unary relationships are involved, we have now represented all the entities and relationships from the E-R model. Also, each of the previous relations is in 3NF because all attributes are simple, all nonkeys are fully dependent on the whole key, and there are no dependencies between nonkeys in the INVOICE and INVENTORY ITEM relations.

Now suppose that Bob Mellankamp wanted an additional report that was not previously known by the analyst who designed the inventory control system for Hoosier Burger. A rough sketch of this new report, listing volume of purchases from each vendor by type of item in a given month, appears in Figure 9-15. In this report, the same type of item may appear many times if multiple vendors supply the same type of item.

This report contains data about several relations already known to the analyst, including:

INVOICE(Vendor_ID, Invoice_Number, Invoice_Date): primary keys and the date are needed to select invoices in the specified month of the report
INVENTORY ITEM(Item_Number, Type_of_Item): primary key and a nonkey in the report
INVOICE ITEM(Vendor_ID, Invoice_Number, Item_Number, Quantity_Added): primary keys and the raw quantity of items invoiced that are subtotaled by vendor and type of item in the report

In addition, the report includes a new attribute: Vendor_Name. After some investigation, an analyst determines that Vendor_ID→Vendor_Name. Because the whole primary key of the INVOICE relation is Vendor_ID and Invoice_Number, if Vendor_Name were part of the INVOICE relation, this relation would violate the 3NF rule. So, a new VENDOR relation must be created as follows:

VENDOR(Vendor_ID, Vendor_Name)

Now, Vendor_ID not only is part of the primary key of INVOICE but also is a foreign key referencing the VENDOR relation. Hence, a one-to-many relationship from VENDOR to INVOICE is necessary. The systems analyst determines that an

FIGURE 9-15
Hoosier Burger monthly vendor load report.

Monthly Vendor Load Report
for Month: xxxxx

Page *x* of *n*

Vendor			
ID	Name	Type of Item	Total Quantity Added
V1	V1name	aaa	nnn1
		bbb	nnn2
		ccc	nnn3
V2	V2name	bbb	nnn4
		mmm	nnn5

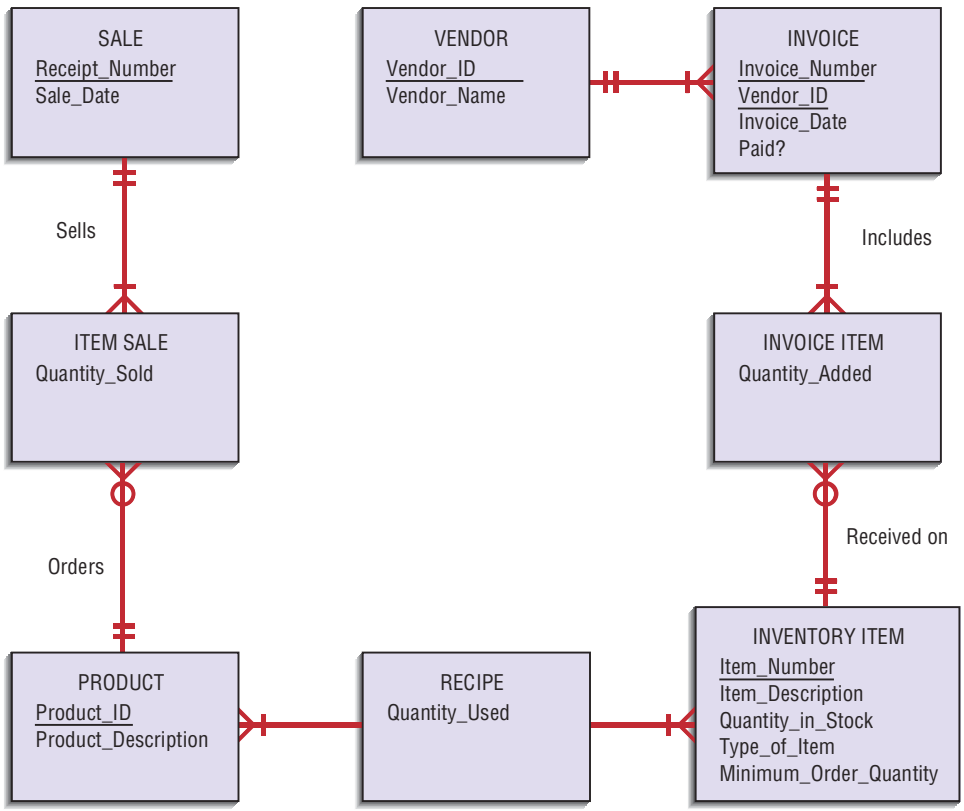


FIGURE 9-16
E-R diagram corresponding to normalized relations of Hoosier Burger's inventory control system.

invoice must come from a vendor, and keeping data about a vendor is not necessary unless the vendor invoices Hoosier Burger. An updated E-R diagram, reflecting these enhancements for new data needed in the monthly vendor load report, appears in Figure 9-16. The normalized relations for this database are:

SALE(Receipt_Number, Sale_Date)
PRODUCT(Product_ID, Product_Description)
INVOICE(Vendor_ID, Invoice_Number, Invoice_Date, Paid?)
INVENTORY ITEM(Item_Number, Item_Description, Quantity_in_Stock, Type_of_Item, Minimum_Order_Quantity)
ITEM SALE(Receipt_Number, Product_ID, Quantity_Sold)
INVOICE ITEM(Vendor_ID, Invoice_Number, Item_Number, Quantity_Added)
RECIPE(Product_ID, Item_Number, Quantity_Used)
VENDOR(Vendor_ID, Vendor_Name)

Physical File and Database Design

Designing physical files and databases requires certain information that should have been collected and produced during prior SDLC phases. This information includes:

- Normalized relations, including volume estimates
- Definitions of each attribute
- Descriptions of where and when data are used: entered, retrieved, deleted, and updated (including frequencies)
- Expectations or requirements for response time and data integrity
- Descriptions of the technologies used for implementing the files and database so that the range of required decisions and choices for each is known

Normalized relations are, of course, the result of logical database design. Statistics on the number of rows in each table, as well as the other information listed may have been collected during requirements determination in systems analysis. If not, these items need to be discovered to proceed with database design.

We take a bottom-up approach to reviewing physical file and database design. Thus, we begin the physical design phase by addressing the design of physical fields for each attribute in a logical data model.

Designing Fields

Field
The smallest unit of named application data recognized by system software.

A **field** is the smallest unit of application data recognized by system software, such as a programming language or database management system. An attribute from a logical database model may be represented by several fields. For example, a student name attribute in a normalized student relation might be represented as three fields: last name, first name, and middle initial. Each field requires a separate definition when the application system is implemented.

In general, you will represent each attribute from each normalized relation as one or more fields. The basic decisions you must make in specifying each field concern the type of data (or storage type) used to represent the field and data integrity controls for the field.

Choosing Data Types

Data type
A coding scheme recognized by system software for representing organizational data.

A **data type** is a coding scheme recognized by system software for representing organizational data. The bit pattern of the coding scheme is usually immaterial to you, but the space to store data and the speed required to access data are of consequence in the physical file and database design. The specific file or database management software you use with your system will dictate which choices are available to you. For example, Table 9-2 lists the data types available in Microsoft Access.

Selecting a data type balances four objectives that will vary in degree of importance for different applications:

1. Minimize storage space
2. Represent all possible values of the field
3. Improve data integrity for the field
4. Support all data manipulations desired on the field

You want to choose a data type for a field that minimizes space, represents every possible legitimate value for the associated attribute, and allows the data to be manipulated as needed. For example, suppose a "quantity sold" field can be represented by a Number data type. You would select a length for this field that would handle the maximum value, plus some room for growth of the business. Further, the Number data type will restrict users from entering inappropriate values (text), but it does allow negative numbers (if this is a problem, application code or form design may be required to restrict the values to positive).

Be careful—the data type must be suitable for the life of the application; otherwise, maintenance will be required. Choose data types for future needs by anticipating growth. Also, be careful that date arithmetic can be done so that dates can be subtracted or time periods can be added to or subtracted from a date.

Several other capabilities of data types may be available with some database technologies. We discuss a few of the most common of these features next: calculated fields and coding and compression techniques.

TABLE 9-2: Microsoft Access Data Types

Data Type	Description
Text	Text or combinations of text and numbers, as well as numbers that don't require calculations, such as phone numbers. A specific length is indicated, with a maximum number of characters of 255. One byte of storage is required for each character used.
Memo	Lengthy (up to 65,535 characters) text or combinations of text and numbers. One byte of storage is required for each character used.
Number	Numeric data used in mathematical calculations. Either 1, 2, 4, or 8 bytes of storage space is required, depending on the specified length of the number.
Date/Time	Date and time values for the years 100 through 9999. Eight bytes of storage space is required.
Currency	Currency values and numeric data used in mathematical calculations involving data with one to four decimal places. Accurate to 15 digits on the left side of the decimal separator and to 4 digits on the right side. Eight bytes of storage space is required.
Autonumber	A unique sequential (incremented by 1) number or random number assigned by Microsoft Access whenever a new record is added to a table. Typically, 4 bytes of storage is required.
Yes/No	Yes and No values and fields that contain only one of two values (Yes/No, True/False, or On/Off). One bit of storage is required.
OLE Object	An object (such as a Microsoft Excel spreadsheet, a Microsoft Word document, graphics, sounds, or other binary data) linked to or embedded in a Microsoft Access table. Up to 1 gigabyte of storage possible.
Hyperlink	Text or combinations of text and numbers stored as text and used as a hyperlink address (typical URL form).
Lookup Wizard	Creates a field that allows you to choose a value from another table (the table's primary key) or from a list of values by using a list box or combo box. Clicking this option starts the Lookup Wizard, which creates a Lookup field. After you complete the wizard, Microsoft Access sets the data type based on the values selected in the wizard. Used for foreign keys to enforce referential integrity. Space requirement depends on length of foreign key or lookup value.

Calculated Fields It is common that an attribute is mathematically related to other data. For example, an invoice may include a "total due" field, which represents the sum of the amount due on each item on the invoice. A field that can be derived from other database fields is called a **calculated** (or **computed** or **derived**) **field** (recall that a functional dependency between attributes does not imply a calculated field). Some database technologies allow you to explicitly define calculated fields along with other raw data fields. If you specify a field as calculated, you would then usually be prompted to enter the formula for the calculation; the formula can involve other fields from the same record and possibly fields from records in related files. The database technology will either store the calculated value or compute it when requested.

Calculated (or computed or derived) field
A field that can be derived from other database fields.

Coding and Compression Techniques Some attributes have few values from a large range of possible values. For example, although a six-digit field (five numbers plus a value sign) can represent numbers −99999 to 99999, maybe only 100 positive values within this range will ever exist. Thus, a Number data type does not adequately restrict the permissible values for data integrity, and storage space for five digits plus a value sign is wasteful. To use space more efficiently (and less space may mean faster access because the data you need are closer together), you can define a field for an attribute so that the possible attribute values are not represented literally but rather are abbreviated. For example, suppose in Pine Valley Furniture each product has a finish attribute, with possible values Birch, Walnut, Oak, and so forth. To store this attribute as Text might require 12, 15, or even 20 bytes to represent the longest finish value.

Suppose that even a liberal estimate is that Pine Valley Furniture will never have more than twenty-five finishes. Thus, a single alphabetic or alphanumeric character would be more than sufficient. We not only reduce storage space but also increase integrity (by restricting input to only a few values), which helps to achieve two of the physical file and database design goals. Codes also have disadvantages. If used in system inputs and outputs, they can be more difficult for users to remember, and programs must be written to decode fields if codes will not be displayed.

Controlling Data Integrity

We have already explained that data typing helps control data integrity by limiting the possible range of values for a field. You can use additional physical file and database design options to ensure higher-quality data. Although these controls can be imposed within application programs, it is better to include these as part of the file and database definitions so that the controls are guaranteed to be applied all the time, as well as uniformly for all programs. The five popular data integrity control methods are default value, input mask, range control, referential integrity, and null value control.

Default value
The value a field will assume unless an explicit value is entered for that field.

- *Default value:* A **default value** is the value a field will assume unless an explicit value is entered for the field. For example, the city and state of most customers for a particular retail store will likely be the same as the store's city and state. Assigning a default value to a field can reduce data-entry time (the field can simply be skipped during data entry) and data-entry errors, such as typing *IM* instead of *IN* for *Indiana*.

Input mask
A pattern of codes that restricts the width and possible values for each position of a field.

- *Input mask:* Some data must follow a specified pattern. An **input mask** (or field template) is a pattern of codes that restricts the width and possible values for each position within a field. For example, a product number at Pine Valley Furniture is four alphanumeric characters—the first is alphabetic and the next three are numeric—defined by an input mask of L999, where L means that only alphabetic characters are accepted, and 9 means that only numeric digits are accepted. M128 is an acceptable value, but 3128 or M12H would be unacceptable. Other types of input masks can be used to convert all characters to uppercase, indicate how to show negative numbers, suppress showing leading zeros, or indicate whether entry of a letter or digit is optional.
- *Range control:* Both numeric and alphabetic data may have a limited set of permissible values. For example, a field for the number of product units sold may have a lower bound of 0, and a field that represents the month of a product sale may be limited to the values JAN, FEB, and so forth.
- *Referential integrity:* As noted earlier in this chapter, the most common example of referential integrity is cross-referencing between relations. For example, consider the pair of relations in Figure 9-17A. In this case, the values for the foreign key Customer_ID field within a customer order must be limited to the set of Customer_ID values from the customer relation; we would not want to accept an order for a nonexisting or unknown customer. Referential integrity may be useful in other instances. Consider the employee relation example in Figure 9-17B. In this example, the employee relation has a field of Supervisor_ID. This field refers to the Employee_ID of the employee's supervisor and should have referential integrity on the Employee_ID field within the same relation. Note in this case that because some employees do not

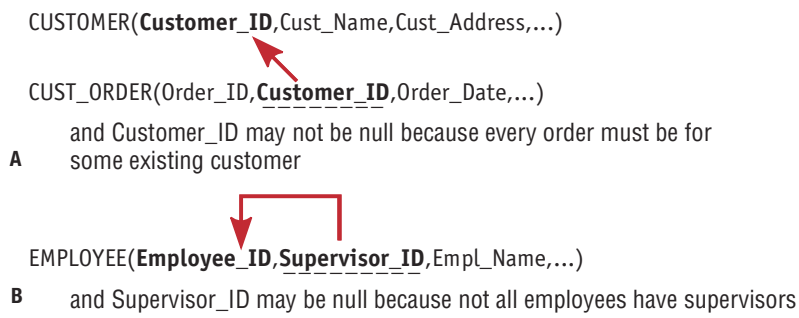


FIGURE 9-17
Examples of referential integrity field controls: (A) Referential integrity between relations, (B) Referential integrity within a relation.

have supervisors, this referential integrity constraint is weak because the value of a Supervisor_ID field may be empty.

- *Null value control:* A **null value** is a special field value, distinct from 0, blank, or any other value, that indicates that the value for the field is missing or otherwise unknown. It is not uncommon that when it is time to enter data—for example, a new customer—you might not know the customer's phone number. The question is whether a customer, to be valid, must have a value for this field. The answer for this field is probably initially no, because most data processing can continue without knowing the customer's phone number. Later, a null value may not be allowed when you are ready to ship product to the customer. On the other hand, you must always know a value for the Customer_ID field. Because of referential integrity, you cannot enter any customer orders for this new customer without knowing an existing Customer_ID value, and the customer's name is essential for visual verification of correct data entry. Besides using a special null value when a field is missing its value, you can also estimate the value, produce a report indicating rows of tables with critical missing values, or determine whether the missing value matters in computing needed information.

Null value
A special field value, distinct from 0, blank, or any other value, that indicates that the value for the field is missing or otherwise unknown.

Designing Physical Tables

A relational database is a set of related tables (tables are related by foreign keys referencing primary keys). In logical database design, you grouped into a relation those attributes that concern some unifying, normalized business concept, such as a customer, product, or employee. In contrast, a **physical table** is a named set of rows and columns that specifies the fields in each row of the table. A physical table may or may not correspond to one relation. Whereas normalized relations possess properties of well-structured relations, the design of a physical table has two goals different from those of normalization: efficient use of secondary storage and data-processing speed.

Physical table
A named set of rows and columns that specifies the fields in each row of the table.

The efficient use of secondary storage (disk space) relates to how data are loaded on disks. Disks are physically divided into units (called *pages*) that can be read or written in one machine operation. Space is used efficiently when the physical length of a table row divides close to evenly into the length of the storage unit. For many information systems, this even division is difficult to achieve because it depends on factors, such as operating system parameters, outside the control of each database. Consequently, we do not discuss this factor of physical table design in this text.

A second and often more important consideration when selecting a physical table design is efficient data processing. Data are most efficiently processed

Denormalization
The process of splitting or combining normalized relations into physical tables based on affinity of use of rows and fields.

when they are stored close to one another in secondary memory, thus minimizing the number of input/output (I/O) operations that must be performed. Typically, the data in one physical table (all the rows and fields in those rows) are stored close together on disk. **Denormalization** is the process of splitting or combining normalized relations into physical tables based on affinity of use of rows and fields. Consider Figure 9-18. In Figure 9-18A, a normalized product relation is split into separate physical tables with each containing only engineering, accounting, or marketing product data; the primary key must be included in each table. Note, the Description and Color attributes are repeated in both the engineering and marketing tables because these attributes relate to both kinds of data. In Figure 9-18B, a customer relation is denormalized by putting rows from different geographic regions into separate tables. In both cases, the goal is to create tables that contain only the data used together in programs. By placing data used together close to one another on disk, the number of disk I/O operations needed to retrieve all the data needed in a program is minimized.

Denormalization can increase the chance of errors and inconsistencies that normalization avoided. Further, denormalization optimizes certain data processing at the expense of others, so if the frequencies of different processing activities change, the benefits of denormalization may no longer exist.

FIGURE 9-18
Examples of denormalization: (A) Denormalization by columns, (B) Denormalization by rows.

A

Normalized Product Relation

PRODUCT(Product_ID,Description,Drawing_Number,Weight,Color,Unit_Cost,Burden_Rate,Price,Product_Manager)

Denormalized Functional Area Product Relations for Tables

Engineering: E_PRODUCT(Product_ID,Description,Drawing_Number,Weight,Color)
Accounting: A_PRODUCT(Product_ID,Unit_Cost,Burden_Rate)
Marketing: M_PRODUCT(Product_ID,Description,Color,Price,Product_Manager)

B

Normalized Customer Table

CUSTOMER

Customer_ID	Name	Region	Annual_Sales
1256	Rogers	Atlantic	10,000
1323	Temple	Pacific	20,000
1455	Gates	South	15,000
1626	Hope	Pacific	22,000
2433	Bates	South	14,000
2566	Bailey	Atlantic	12,000

Denormalized Regional Customer Tables

A_CUSTOMER

Customer_ID	Name	Region	Annual_Sales
1256	Rogers	Atlantic	10,000
2566	Bailey	Atlantic	12,000

P_CUSTOMER

Customer_ID	Name	Region	Annual_Sales
1323	Temple	Pacific	20,000
1626	Hope	Pacific	22,000

S_CUSTOMER

Customer_ID	Name	Region	Annual_Sales
1455	Gates	South	15,000
2433	Bates	South	14,000

Various forms of denormalization can be done, but no hard-and-fast rules will help you decide when to denormalize data. Here are three common situations in which denormalization often makes sense (see Figure 9-19 for illustrations):

1. *Two entities with a one-to-one relationship.* Figure 9-19A shows student data with optional data from a standard scholarship application a student may complete. In this case, one record could be formed with four fields from the STUDENT and SCHOLARSHIP APPLICATION FORM normalized relations. (*Note:* In this case, fields from the optional entity must have null values allowed.)
2. *A many-to-many relationship (associative entity) with nonkey attributes.* Figure 9-19B shows price quotes for different items from different vendors. In this case, fields from ITEM and PRICE QUOTE relations might be combined into one physical table to avoid having to combine all three tables together. (*Note:* It may create considerable duplication of data—in the example, the ITEM fields, such as Description, would repeat for each price quote—and excessive updating if duplicated data changes.)
3. *Reference data.* Figure 9-19C shows that several ITEMs have the same STORAGE INSTRUCTIONS, and STORAGE INSTRUCTIONS relate only to ITEMs. In this case, the storage instruction data could be stored in the ITEM table, thus reducing the number of tables to access but also creating redundancy and the potential for extra data maintenance.

Arranging Table Rows

The result of denormalization is the definition of one or more physical files. A computer operating system stores data in a **physical file,** which is a named set of table rows stored in a contiguous section of secondary memory. A file contains rows and columns from one or more tables, as produced from denormalization. To the operating system—like Windows, Linux, or Mac OS—each table may be one file or the whole database may be in one file, depending on how the database technology and database designer organize data. The way the operating system arranges table rows in a file is called a **file organization.** With some database technologies, the systems designer can choose among several organizations for a file.

Physical file
A named set of table rows stored in a contiguous section of secondary memory.

File organization
A technique for physically arranging the records of a file.

If the database designer has a choice, he or she chooses a file organization for a specific file to provide:

1. Fast data retrieval
2. High throughput for processing transactions
3. Efficient use of storage space
4. Protection from failures or data loss
5. Minimal need for reorganization
6. Accommodation of growth
7. Security from unauthorized use

Often these objectives conflict, and you must select an organization for each file that provides a reasonable balance among the criteria within the resources available.

To achieve these objectives, many file organizations utilize the concept of a pointer. A **pointer** is a field of data that can be used to locate a related field or row of data. In most cases, a pointer contains the address of the associated data, which has no business meaning. Pointers are used in file organizations when it is not possible to store related data next to each other. Because such situations are often the case, pointers are common. In most cases, fortunately, pointers are hidden from a programmer. Yet, because a database designer may need to decide whether and how to use pointers, we introduce the concept here.

Pointer
A field of data that can be used to locate a related field or row of data.