

# Using Data Flow Diagrams

## LEARNING OBJECTIVES

Once you have mastered the material in this chapter you will be able to:

1. Comprehend the importance of using logical and physical data flow diagrams (DFDs) to graphically depict data movement for humans and systems in an organization.
2. Create, use, and explode logical DFDs to capture and analyze the current system through parent and child levels.
3. Develop and explode logical DFDs that illustrate the proposed system.
4. Produce physical DFDs based on logical DFDs you have developed.
5. Understand and apply the concept of partitioning of physical DFDs.



The systems analyst needs to make use of the conceptual freedom afforded by data flow diagrams, which graphically characterize data processes and flows in a business system. In their original state, data flow diagrams depict the broadest possible overview of system inputs, processes, and outputs, which correspond to those of the general systems model discussed in

Chapter 2. A series of layered data flow diagrams may be used to represent and analyze detailed procedures in the larger system.

## THE DATA FLOW APPROACH TO HUMAN REQUIREMENTS DETERMINATION

When systems analysts attempt to understand the information requirements of users, they must be able to conceptualize how data move through the organization, the processes or transformation that the data undergo, and what the outputs are. Although interviews and the investigation of hard data provide a verbal narrative of the system, a visual depiction can crystallize this information for users and analysts in a useful way.

Through a structured analysis technique called data flow diagrams (DFDs), the systems analyst can put together a graphical representation of data processes throughout the organization. By using combinations of only four symbols, the systems analyst can create a pictorial depiction of processes that will eventually provide solid system documentation.

### Advantages of the Data Flow Approach

The data flow approach has four chief advantages over narrative explanations of the way data move through the system:

1. Freedom from committing to the technical implementation of the system too early.
2. Further understanding of the interrelatedness of systems and subsystems.
3. Communicating current system knowledge to users through data flow diagrams.
4. Analysis of a proposed system to determine if the necessary data and processes have been defined.

Perhaps the biggest advantage lies in the conceptual freedom found in the use of the four symbols (covered in the upcoming subsection on DFD conventions). (You will recognize three of the symbols from Chapter 2.) None of the symbols specifies the physical aspects of implementation. DFDs emphasize the processing of data or the transforming of data as they move through a variety of processes. In logical DFDs, there is no distinction between manual or automated processes. Neither are the processes graphically depicted in chronological order. Rather, processes are eventually grouped together if further analysis dictates that it makes sense to do so. Manual processes are put together, and automated processes can also be paired with each other. This concept, called *partitioning*, is taken up in a later section.

Conventions Used in Data Flow Diagrams




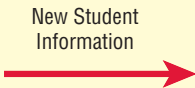
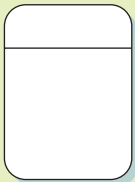
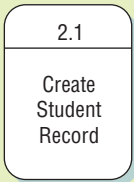
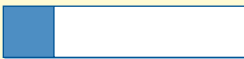
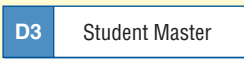
Four basic symbols are used to chart data movement on data flow diagrams: a double square, an arrow, a rectangle with rounded corners, and an open-ended rectangle (closed on the left side and open ended on the right), as shown in Figure 7.1. An entire system and numerous subsystems can be depicted graphically with these four symbols in combination.

The double square is used to depict an external entity (another department, a business, a person, or a machine) that can send data to or receive data from the system. The external entity, or just entity, is also called a source or destination of data, and it is considered to be external to the system being described. Each entity is labeled with an appropriate name. Although it interacts with the system, it is considered as outside the boundaries of the system. Entities should be named with a noun. The same entity may be used more than once on a given data flow diagram to avoid crossing data flow lines.

The arrow shows movement of data from one point to another, with the head of the arrow pointing toward the data’s destination. Data flows occurring simultaneously can be depicted doing just that through the use of parallel arrows. Because an arrow represents data about a person, place, or thing, it too should be described with a noun.

A rectangle with rounded corners is used to show the occurrence of a transforming process. Processes always denote a change in or transformation of data; hence, the data flow leaving a process is *always* labeled differently than the one entering it. Processes represent work being

**FIGURE 7.1**  
The four basic symbols used in data flow diagrams, their meanings, and examples.

Symbol	Meaning	Example
	Entity	
	Data Flow	
	Process	
	Data Store	

performed in the system and should be named using one of the following formats. A clear name makes it easier to understand what the process is accomplishing.

1. When naming a high-level process, assign the process the name of the whole system. An example is INVENTORY CONTROL SYSTEM.
2. When naming a major subsystem, use a name such as INVENTORY REPORTING SUBSYSTEM or INTERNET CUSTOMER FULFILLMENT SYSTEM.
3. When naming detailed processes, use a verb-adjective-noun combination. The verb describes the type of activity, such as COMPUTE, VERIFY, PREPARE, PRINT, or ADD. The noun indicates what the major outcome of the process is, such as REPORT or RECORD. The adjective illustrates which specific output, such as BACKORDERED or INVENTORY, is produced. Examples of complete process names are COMPUTE SALES TAX, VERIFY CUSTOMER ACCOUNT STATUS, PREPARE SHIPPING INVOICE, PRINT BACK-ORDERED REPORT, SEND CUSTOMER EMAIL CONFIRMATION, VERIFY CREDIT CARD BALANCE, and ADD INVENTORY RECORD.

A process must also be given a unique identifying number indicating its level in the diagram. This organization is discussed later in this chapter. Several data flows may go into and out of each process. Examine processes with only a single flow in and out for missing data flows.

The last basic symbol used in data flow diagrams is an open-ended rectangle, which represents a data store. The rectangle is drawn with two parallel lines that are closed by a short line on the left side and are open ended on the right. These symbols are drawn only wide enough to allow identifying lettering between the parallel lines. In logical data flow diagrams, the type of physical storage is not specified. At this point the data store symbol is simply showing a depository for data that allows examination, addition, and retrieval of data.

The data store may represent a manual store, such as a filing cabinet, or a computerized file or database. Because data stores represent a person, place, or thing, they are named with a noun. Temporary data stores, such as scratch paper or a temporary computer file, are not included on the data flow diagram. Give each data store a unique reference number, such as D1, D2, D3, and so on.

## DEVELOPING DATA FLOW DIAGRAMS

Data flow diagrams can and should be drawn systematically. Figure 7.2 summarizes the steps involved in successfully completing data flow diagrams. First, the systems analyst needs to conceptualize data flows from a top-down perspective.

To begin a data flow diagram, collapse the organization's system narrative (or story) into a list with the four categories of external entity, data flow, process, and data store. This list in turn helps determine the boundaries of the system you will be describing. Once a basic list of data elements has been compiled, begin drawing a context diagram.

Here are a few basic rules to follow:

1. The data flow diagram must have at least one process, and must not have any freestanding objects or objects connected to themselves.
2. A process must receive at least one data flow coming into the process and create at least one data flow leaving from the process.
3. A data store should be connected to at least one process.
4. External entities should not be connected to each other. Although they communicate independently, that communication is not part of the system we design using DFDs.

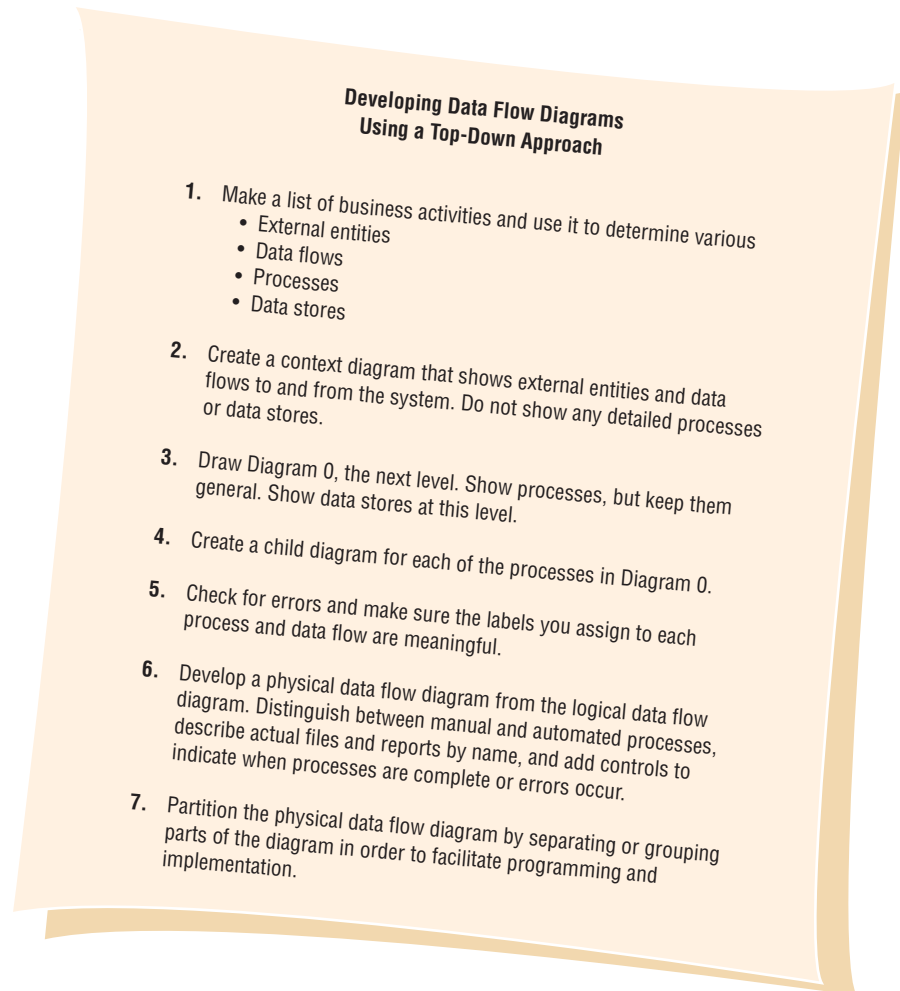
### Creating the Context Diagram

With a top-down approach to diagramming data movement, the diagrams move from general to specific. Although the first diagram helps the systems analyst grasp basic data movement, its general nature limits its usefulness. The initial context diagram should be an overview, one including basic inputs, the general system, and outputs. This diagram will be the most general one, really a bird's-eye view of data movement in the system and the broadest possible conceptualization of the system.

The context diagram is the highest level in a data flow diagram and contains only one process, representing the entire system. The process is given the number zero. All external entities are shown

**FIGURE 7.2**

Steps in developing data flow diagrams.



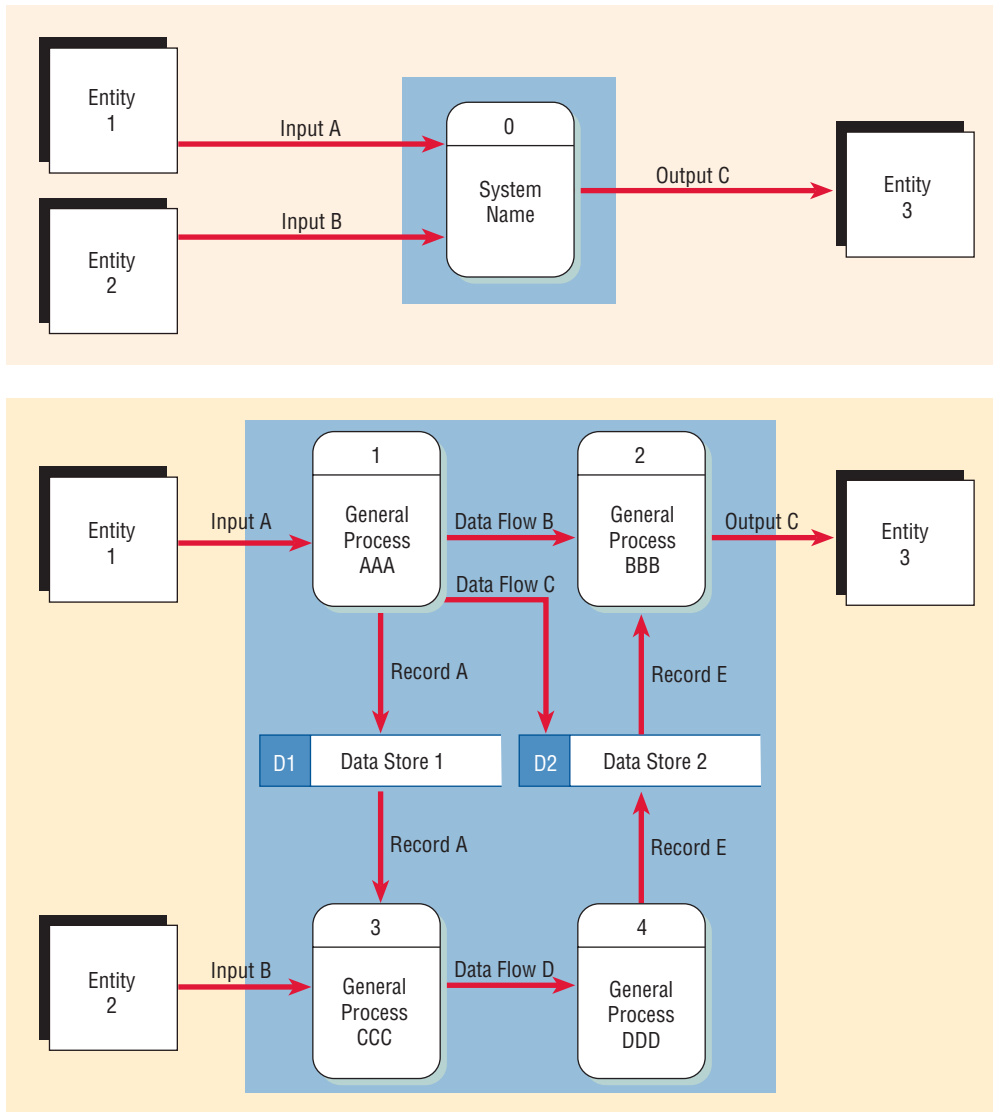
on the context diagram, as well as major data flow to and from them. The diagram does not contain any data stores and is fairly simple to create, once the external entities and the data flow to and from them are known to analysts.

### Drawing Diagram 0 (The Next Level)

More detail than the context diagram permits is achievable by “exploding the diagrams.” Inputs and outputs specified in the first diagram remain constant in all subsequent diagrams. The rest of the original diagram, however, is exploded into close-ups involving three to nine processes and showing data stores and new lower-level data flows. The effect is that of taking a magnifying glass to view the original data flow diagram. Each exploded diagram should use only a single sheet of paper. By exploding DFDs into subprocesses, the systems analyst begins to fill in the details about data movement. The handling of exceptions is ignored for the first two or three levels of data flow diagramming.

Diagram 0 is the explosion of the context diagram and may include up to nine processes. Including more processes at this level will result in a cluttered diagram that is difficult to understand. Each process is numbered with an integer, generally starting from the upper left-hand corner of the diagram and working toward the lower right-hand corner. The major data stores of the system (representing master files) and all external entities are included on Diagram 0. Figure 7.3 schematically illustrates both the context diagram and Diagram 0.

Because a data flow diagram is two-dimensional (rather than linear), you may start at any point and work forward or backward through the diagram. If you are unsure of what you would include at any point, take a different external entity, process, or data store, and then start drawing the flow from it. You may:

**FIGURE 7.3**

Context diagrams (above) can be “exploded” into Diagram 0 (below). Note the greater detail in Diagram 0.

1. Start with the data flow from an entity on the input side. Ask questions such as: “What happens to the data entering the system?” “Is it stored?” “Is it input for several processes?”
2. Work backward from an output data flow. Examine the output fields on a document or screen. (This approach is easier if prototypes have been created.) For each field on the output, ask: “Where does it come from?” or “Is it calculated or stored on a file?” For example, when the output is a PAYCHECK, the EMPLOYEE NAME and ADDRESS would be located on an EMPLOYEE file, the HOURS WORKED would be on a TIME RECORD, and the GROSS PAY and DEDUCTIONS would be calculated. Each file and record would be connected to the process that produces the paycheck.
3. Examine the data flow to or from a data store. Ask: “What processes put data into the store?” or “What processes use the data?” Note that a data store used in the system you are working on may be produced by a different system. Thus, from your vantage point, there may not be any data flow into the data store.
4. Analyze a well-defined process. Look at what input data the process needs and what output it produces. Then connect the input and output to the appropriate data stores and entities.
5. Take note of any fuzzy areas where you are unsure of what should be included or what input or output is required. Awareness of problem areas will help you formulate a list of questions for follow-up interviews with key users.

### Creating Child Diagrams (More Detailed Levels)

Each process on Diagram 0 may in turn be exploded to create a more detailed child diagram. The process on Diagram 0 that is exploded is called the *parent process*, and the diagram that results is called the *child diagram*. The primary rule for creating child diagrams, vertical balancing, dictates that a child diagram cannot produce output or receive input that the parent process does not also produce or receive. All data flow into or out of the parent process must be shown flowing into or out of the child diagram.

The child diagram is given the same number as its parent process in Diagram 0. For example, process 3 would explode to Diagram 3. The processes on the child diagram are numbered using the parent process number, a decimal point, and a unique number for each child process. On Diagram 3, the processes would be numbered 3.1, 3.2, 3.3, and so on. This convention allows the analyst to trace a series of processes through many levels of explosion. If Diagram 0 depicts processes 1, 2, and 3, the child diagrams 1, 2, and 3 are all on the same level.

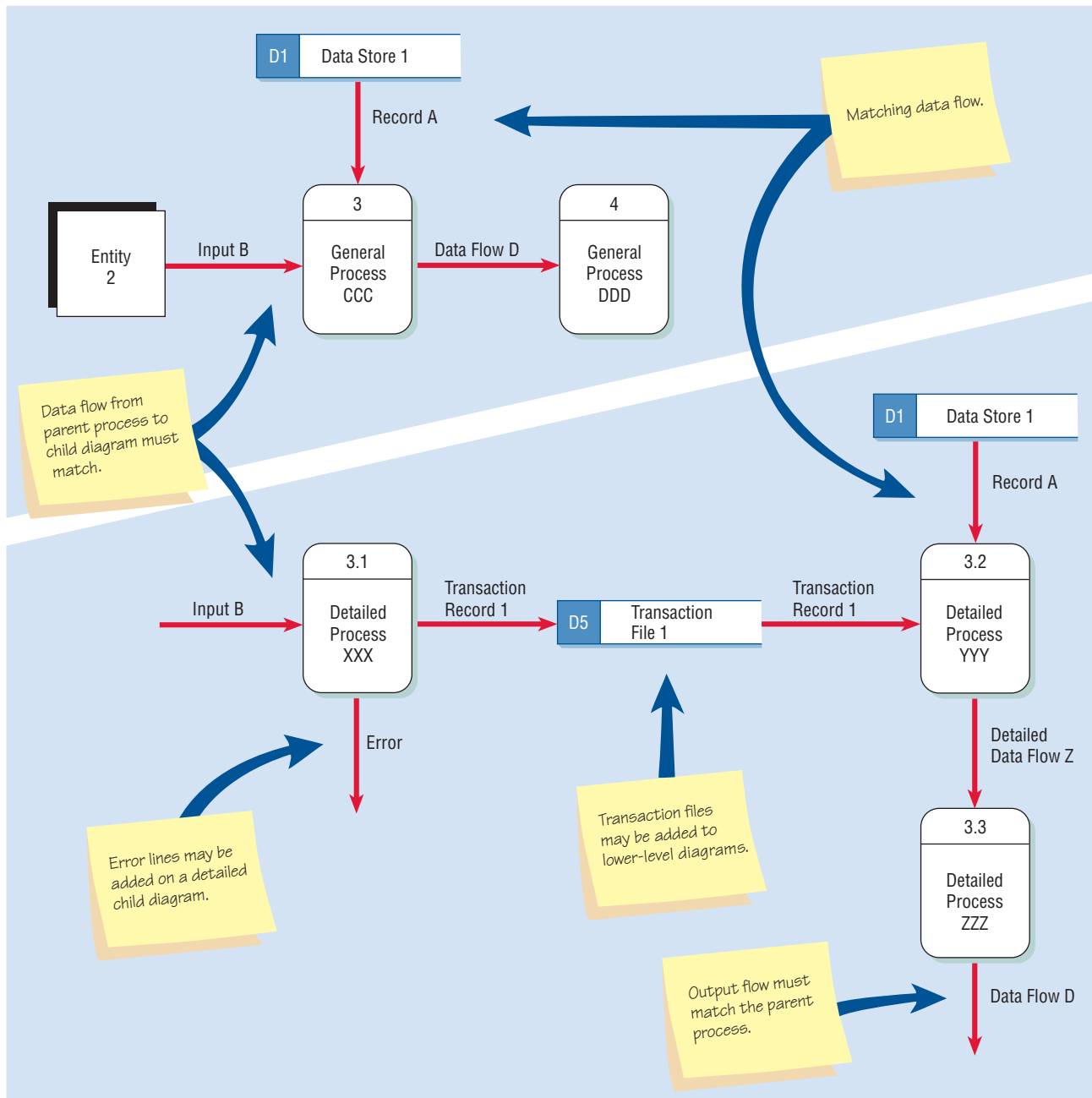
Entities are usually not shown on the child diagrams below Diagram 0. Data flow that matches the parent flow is called an *interface data flow* and is shown as an arrow from or into a blank area of the child diagram. If the parent process has data flow connecting to a data store, the child diagram may include the data store as well. In addition, this lower-level diagram may contain data stores not shown on the parent process. For example, a file containing a table of information, such as a tax table, or a file linking two processes on the child diagram may be included. Minor data flow, such as an error line, may be included on a child diagram but not on the parent.

Processes may or may not be exploded, depending on their level of complexity. When a process is not exploded, it is said to be functionally primitive and is called a *primitive process*. Logic is written to describe these processes and is discussed in detail in Chapter 9. Figure 7.4 illustrates detailed levels in a child data flow diagram.

### Checking the Diagrams for Errors

Several common errors made when drawing data flow diagrams are as follows:

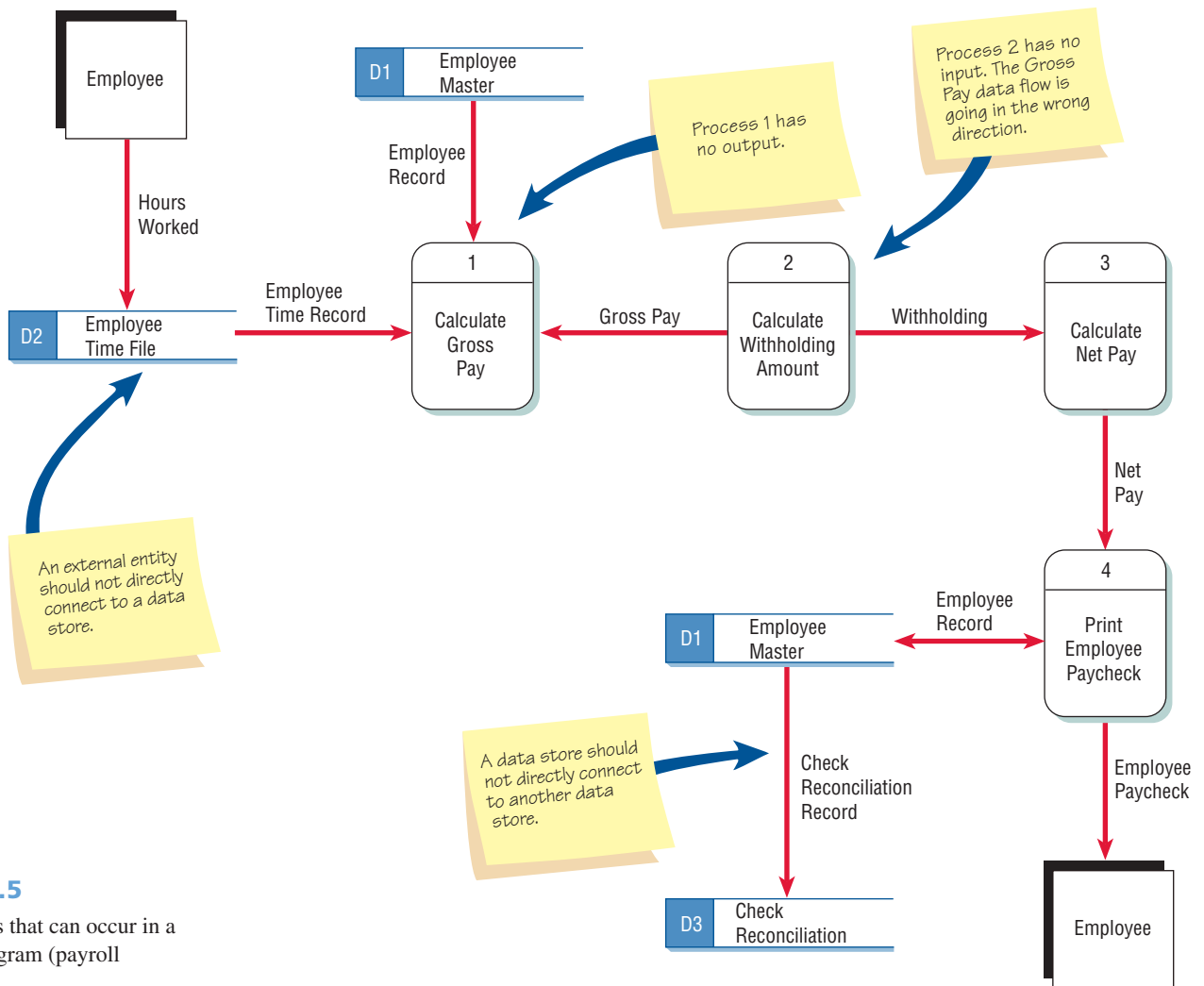
1. Forgetting to include a data flow or pointing an arrow in the wrong direction. An example is a drawn process showing all its data flow as either input or output. Each process transforms data and must receive input and produce output. This type of error usually occurs when the analyst has forgotten to include a data flow or has placed an arrow pointing in the wrong direction. Process 1 in Figure 7.5 contains only input because the GROSS PAY arrow is pointing in the wrong direction. This error also affects process 2, CALCULATE WITHHOLDING AMOUNT, which is in addition missing a data flow representing input for the withholding rates and the number of dependents.
2. Connecting data stores and external entities directly to each other. Data stores and entities may not be connected to each other; data stores and external entities must connect only with a process. A file does not interface with another file without the help of a program or a person moving the data, so EMPLOYEE MASTER cannot directly produce the CHECK RECONCILIATION file. External entities do not directly work with files. For example, you would not want a customer rummaging around in the customer master file. Thus, the EMPLOYEE does not create the EMPLOYEE TIME FILE. Two external entities directly connected indicate that they wish to communicate with each other. This connection is not included on the data flow diagram unless the system is facilitating the communication. Producing a report is an instance of this sort of communication. A process must still be interposed between the entities to produce the report, however.
3. Incorrectly labeling processes or data flow. Inspect the data flow diagram to ensure that each object or data flow is properly labeled. A process should indicate the system name or use the verb-adjective-noun format. Each data flow should be described with a noun.
4. Including more than nine processes on a data flow diagram. Having too many processes creates a cluttered diagram that is confusing to read and hinders rather than enhances communication. If more than nine processes are involved in a system, group some of the processes that work together into a subsystem and place them in a child diagram.

**FIGURE 7.4**

Differences between the parent diagram (above) and the child diagram (below).

5. Omitting data flow. Examine your diagram for linear flow, that is, data flow in which each process has only one input and one output. Except in the case of very detailed child data flow diagrams, linear data flow is somewhat rare. Its presence usually indicates that the diagram has missing data flow. For instance, the process CALCULATE WITHHOLDING AMOUNT needs the number of dependents that an employee has and the WITHHOLDING RATES as input. In addition, NET PAY cannot be calculated solely from the WITHHOLDING, and the EMPLOYEE PAYCHECK cannot be created from the NET PAY alone; it also needs to include an EMPLOYEE NAME, as well as the current and year-to-date payroll and WITHHOLDING AMOUNT figures.
6. Creating unbalanced decomposition (or explosion) in child diagrams. Each child diagram should have the same input and output data flow as the parent process. An exception to this rule is minor output, such as error lines, which are included only on the child diagram. The



**FIGURE 7.5**

Typical errors that can occur in a data flow diagram (payroll example).

data flow diagram in Figure 7.6 is correctly drawn. Note that although the data flow is not linear, you can clearly follow a path directly from the source entity to the destination entity.

## LOGICAL AND PHYSICAL DATA FLOW DIAGRAMS

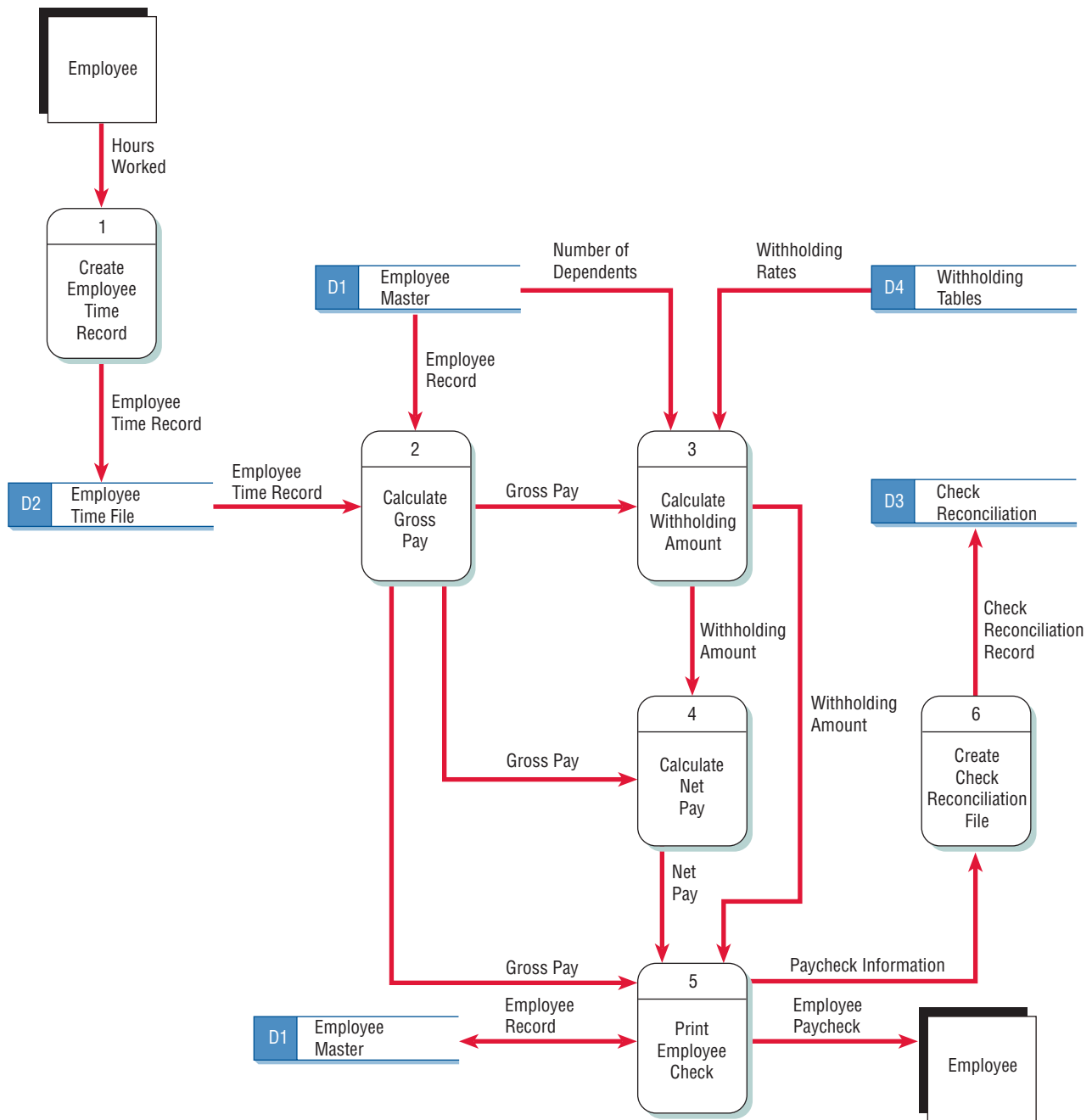
Data flow diagrams are categorized as either logical or physical. A logical data flow diagram focuses on the business and how the business operates. It is not concerned with how the system will be constructed. Instead, it describes the business events that take place and the data required and produced by each event. Conversely, a physical data flow diagram shows how the system will be implemented, including the hardware, software, files, and people involved in the system. The chart shown in Figure 7.7 contrasts the features of logical and physical models. Notice that the logical model reflects the business, whereas the physical model depicts the system.

Ideally, systems are developed by analyzing the current system (the current logical DFD) and then adding features that the new system should include (the proposed logical DFD). Finally, the best methods for implementing the new system should be developed (the physical DFD). This progression is shown in Figure 7.8.

Developing a logical data flow diagram for the current system affords a clear understanding of how the current system operates, and thus a good starting point for developing the logical model of the current system. This time-consuming step is often omitted so as to go straight to the proposed logical DFD.

One argument in favor of taking the time to construct the logical data flow diagram of the current system is that it can be used to create the logical data flow diagram of the new system.



**FIGURE 7.6**

The correct data flow diagram for the payroll example.

Processes that will be unnecessary in the new system may be dropped, and new features, activities, output, input, and stored data may be added. This approach provides a means of ensuring that the essential features of the old system are retained in the new system. In addition, using the logical model for the current system as a basis for the proposed system provides for a gradual transition to the design of the new system. After the logical model for the new system has been developed, it may be used to create a physical data flow diagram for the new system.

Figure 7.9 shows a logical data flow diagram and a physical data flow diagram for a grocery store cashier. The CUSTOMER brings the ITEMS to the register; PRICES for all ITEMS are LOOKED UP and then totaled; next, PAYMENT is given to the cashier; finally, the CUSTOMER is given a RECEIPT. The logical data flow diagram illustrates the processes involved without going into detail about the physical implementation of activities. The physical data flow diagram shows that a bar code—the universal product code (UPC) BAR CODE found on most grocery store items—is used. In addition, the physical data flow diagram mentions manual processes such

**FIGURE 7.7**

Features common to both logical and physical data flow diagrams.

Design Feature	Logical	Physical
What the model depicts	How the business operates.	How the system will be implemented (or how the current system operates).
What the processes represent	Business activities.	Programs, program modules, and manual procedures.
What the data stores represent	Collections of data regardless of how the data are stored.	Physical files and databases, manual files.
Type of data stores	Show data stores representing permanent data collections.	Master files, transition files. Any processes that operate at two different times must be connected by a data store.
System controls	Show business controls.	Show controls for validating input data, for obtaining a record (record found status), for ensuring successful completion of a process, and for system security (example: journal records).

as scanning, explains that a temporary file is used to keep a subtotal of items, and indicates that the PAYMENT could be made by CASH, CHECK, or DEBIT CARD. Finally, it refers to the receipt by its name, CASH REGISTER RECEIPT.

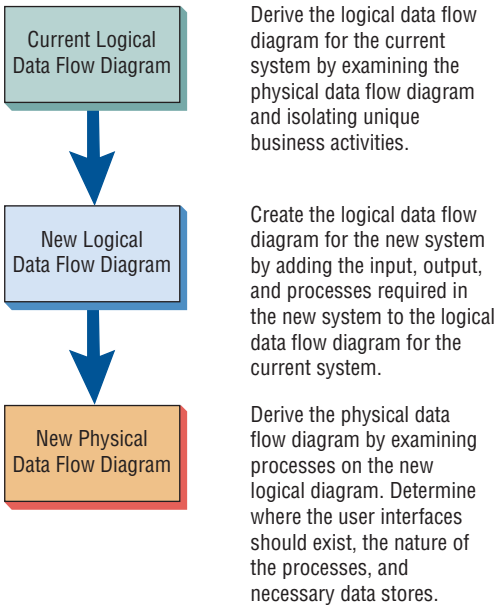
**Developing Logical Data Flow Diagrams**

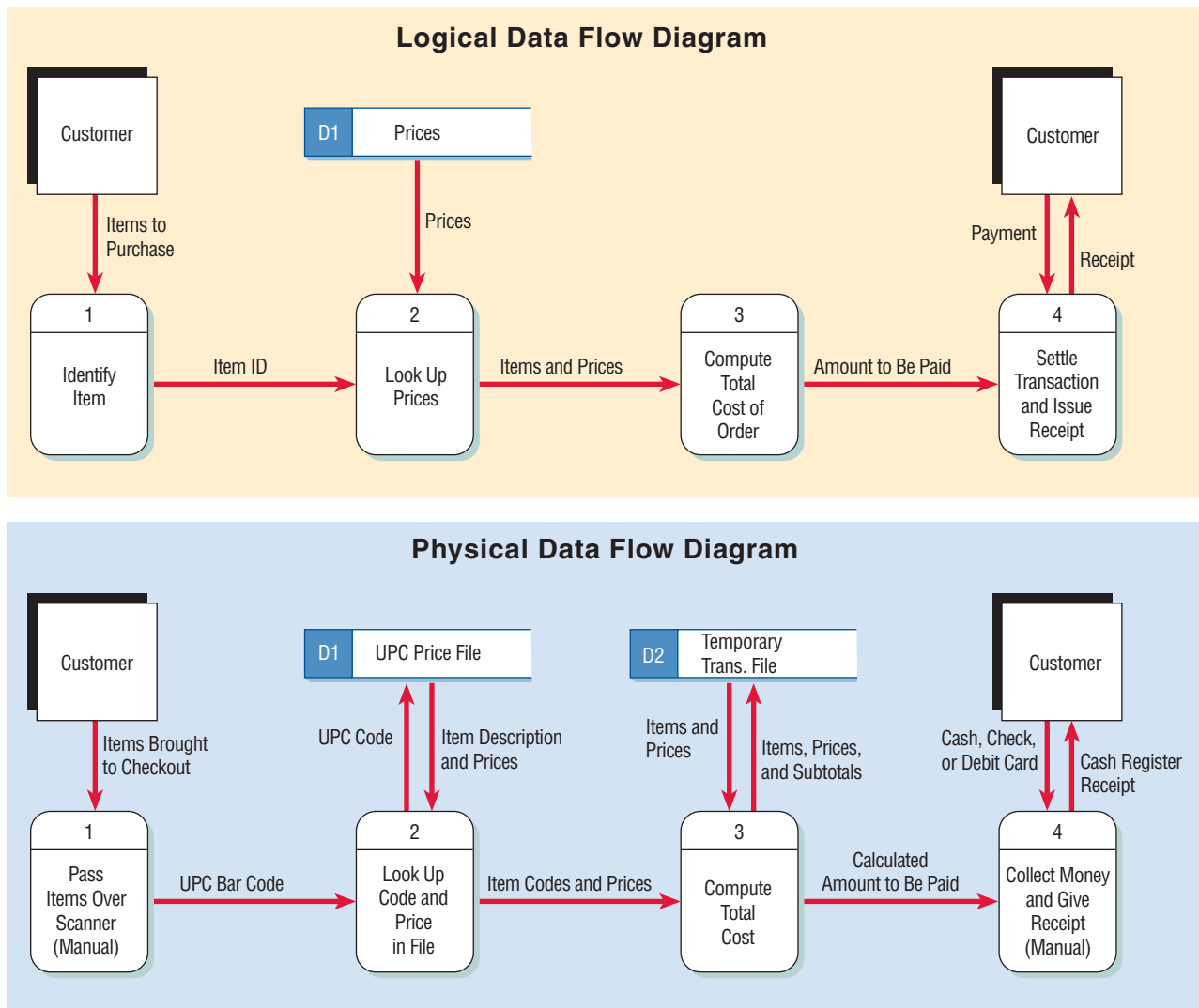
To develop such a diagram, first construct a logical data flow diagram for the current system. There are a number of advantages to using a logical model, including:

- 1. Better communication with users.
- 2. More stable systems.
- 3. Better understanding of the business by analysts.
- 4. Flexibility and maintenance.
- 5. Elimination of redundancies and easier creation of the physical model.

**FIGURE 7.8**

The progression of models from logical to physical.



**FIGURE 7.9**

The physical data flow diagram (below) shows certain details not found on the logical data flow diagram (above).

A logical model is easier to use when communicating with users of the system because it is centered on business activities. Users will thus be familiar with the essential activities and many of the human information requirements of each activity.

Systems formed using a logical data flow diagram are often more stable because they are based on business events and not on a particular technology or method of implementation. Logical data flow diagrams represent features of a system that would exist no matter what the physical means of doing business are. For example, activities such as applying for a video store membership card, checking out a DVD, and returning the DVD, would all occur whether the store had an automated, manual, or hybrid system.

### Developing Physical Data Flow Diagrams

After you develop the logical model of the new system, you may use it to create a physical data flow diagram. The physical data flow diagram shows how the system will be constructed, and usually contains most, if not all, of the elements found in Figure 7.10. Just as logical data flow diagrams have certain advantages, physical data flow diagrams have others, including:

1. Clarifying which processes are performed by humans (manual) and which are automated.
2. Describing processes in more detail than logical DFDs.
3. Sequencing processes that have to be done in a particular order.
4. Identifying temporary data stores.
5. Specifying actual names of files, database tables, and printouts.
6. Adding controls to ensure the processes are done properly.

**FIGURE 7.10**

Physical data flow diagrams contain many items not found in logical data flow diagrams.

#### Contents of Physical Data Flow Diagrams

- Manual processes
- Processes for adding, deleting, changing, and updating records
- Data entry and verifying processes
- Validation processes for ensuring accurate data input
- Sequencing processes to rearrange the order of records
- Processes to produce every unique system output
- Intermediate data stores
- Actual file names used to store data
- Controls to signify completion of tasks or error conditions

Physical data flow diagrams are often more complex than logical data flow diagrams simply because of the many data stores present in a system. The acronym CRUD is often used for Create, Read, Update, and Delete, the activities that must be present in a system for each master file. A CRUD matrix is a tool to represent where each of these processes occurs in a system. Figure 7.11 is a CRUD matrix for an Internet storefront. Notice that some of the processes include more than one activity. Data entry processes such as keying and verifying are also part of physical data flow diagrams.

Physical data flow diagrams also have intermediate data stores, often a transaction file or a temporary database table. Intermediate data stores often consist of transaction files used to store data between processes. Because most processes that require access to a given set of data are unlikely to execute at the same instant in time, transaction files must hold the data from one process to the next. An easily understood example of this concept is found in the everyday experiences of grocery shopping, meal preparation, and eating. The activities are:

1. Selecting items from shelves.
2. Checking out and paying the bill.
3. Transporting the groceries home.
4. Preparing a meal.
5. Eating the meal.

**FIGURE 7.11**

A CRUD matrix for an Internet storefront. This tool can be used to represent where each of four processes (Create, Read, Update, and Delete) occurs within a system.

Activity	Customer	Item	Order	Order Detail
Customer Logon	R			
Item Inquiry		R		
Item Selection		R	C	C
Order Checkout	U	U	U	R
Add Account	C			
Add Item		C		
Close Customer Account	D			
Remove Obsolete Item		D		
Change Customer Demographics	RU			
Change Customer Order	RU	RU	RU	CRUD
Order Inquiry	R	R	R	R

Each of these five activities would be represented by a separate process on a physical data flow diagram, and each one occurs at a different time. For example, you would not typically transport the groceries home and eat them at the same time. Therefore, a “transaction data store” is required to link each task. When you are selecting items, the transaction data store is the shopping cart. After the next process (checking out), the cart is unnecessary. The transaction data store linking checking out and transporting the groceries home is the shopping bag (cheaper than letting you take the cart home!). Bags are an inefficient way of storing the groceries once they are home, so cupboards and a refrigerator are used as a transaction data store between the activity of transporting the goods home and preparing the meal. Finally, a plate, bowl, and cup constitute the link between preparing and eating the meal.

Timing information may also be included. For example, a physical DFD may indicate that an edit program must be run before an update program. Updates must be performed before producing a summary report, or an order must be entered on a Web site before the amount charged to a credit card may be verified with the financial institution. Note that because of such considerations, a physical data flow diagram may appear more linear than a logical model.

Create the physical data flow diagram for a system by analyzing its output and input. When creating a physical data flow diagram, input data flow from an external entity is sometimes called a *trigger* because it starts the activities of a process, and output data flow to an external entity is sometimes called a *response* because it is sent as the result of some activity. Determine which data fields or elements need to be keyed. These fields are called *base elements* and must be stored in a file. Elements that are not keyed but are rather the result of a calculation or logical operation are called *derived elements*.

Sometimes it is not clear how many processes to place in one diagram and when to create a child diagram. One suggestion is to examine each process and count the number of data flows entering and leaving it. If the total is greater than four, the process is a good candidate for a child diagram. Physical data flow diagrams are illustrated later in this chapter.

**EVENT MODELING AND DATA FLOW DIAGRAMS.** A practical approach to creating physical data flow diagrams is to create a simple data flow diagram fragment for each unique system event. Events cause the system to do something and act as a trigger to the system. Triggers start activities and processes, which in turn use data or produce output. An example of an event is a customer reserving a flight on the Web. As each Web form is submitted, processes are activated, such as validating and storing the data and formatting and displaying the next Web page.

Events are usually summarized in an event response table. An example of an event response table for an Internet storefront business is illustrated in Figure 7.12. A data flow diagram fragment is represented by a row in the table. Each DFD fragment is a single process on a data flow diagram. All the fragments are then combined to form Diagram 0. The trigger and response columns become the input and output data flows, and the activity becomes the process. The analyst must determine the data stores required for the process by examining the input and output data flows. Figure 7.13 illustrates a portion of the data flow diagram for the first three rows of the event response table.

The advantage of building data flow diagrams based on events is that the users are familiar with the events that take place in their business area and know how the events drive other activities.

**USE CASES AND DATA FLOW DIAGRAMS.** In Chapter 2, we introduced the concept of a *use case*. We use this notion of a use case in creating data flow diagrams. A use case summarizes an event and has a similar format to process specifications (described in Chapter 9). Each use case defines one activity and its trigger, input, and output. Figure 7.14 illustrates a use case for Process 3, Add Customer Item.

This approach allows the analyst to work with users to understand the nature of the processes and activities and then create a single data flow diagram fragment. When creating use cases, first make an initial attempt to define the use cases without going into detail. This step provides an overview of the system and leads to the creation of Diagram 0. Decide what the names should be and provide a brief description of the activity. List the activities, inputs, and outputs for each one.

Make sure you document the steps used in each use case. These should be in the form of business rules that list or explain the human and system activities completed for each use case. If at all

Event	Source	Trigger	Activity	Response	Destination
Customer logs on	Customer	Customer number and password	Find customer record and verify password. Send Welcome Web page.	Welcome Web page	Customer
Customer browses items at Web storefront	Customer	Item information	Find item price and quantity available. Send Item Response Web page.	Item Response Web page	Customer
Customer places item into shopping basket at Web storefront	Customer	Item purchase (item number and quantity)	Store data on Order Detail Record. Calculate shipping cost using shipping tables. Update customer total. Update item quantity on hand.	Items Purchased Web page	Customer
Customer checks out	Customer	Clicks “Check Out” button on Web page	Display Customer Order Web page.	Verification Web page	
Obtain customer payment	Customer	Credit card information	Verify credit card amount with credit card company. Send.	Credit card data Customer feedback	Credit card company Customer
Send customer email		Temporal, hourly	Send customer an email confirming shipment.		Customer

**FIGURE 7.12**

An event response table for an Internet storefront.

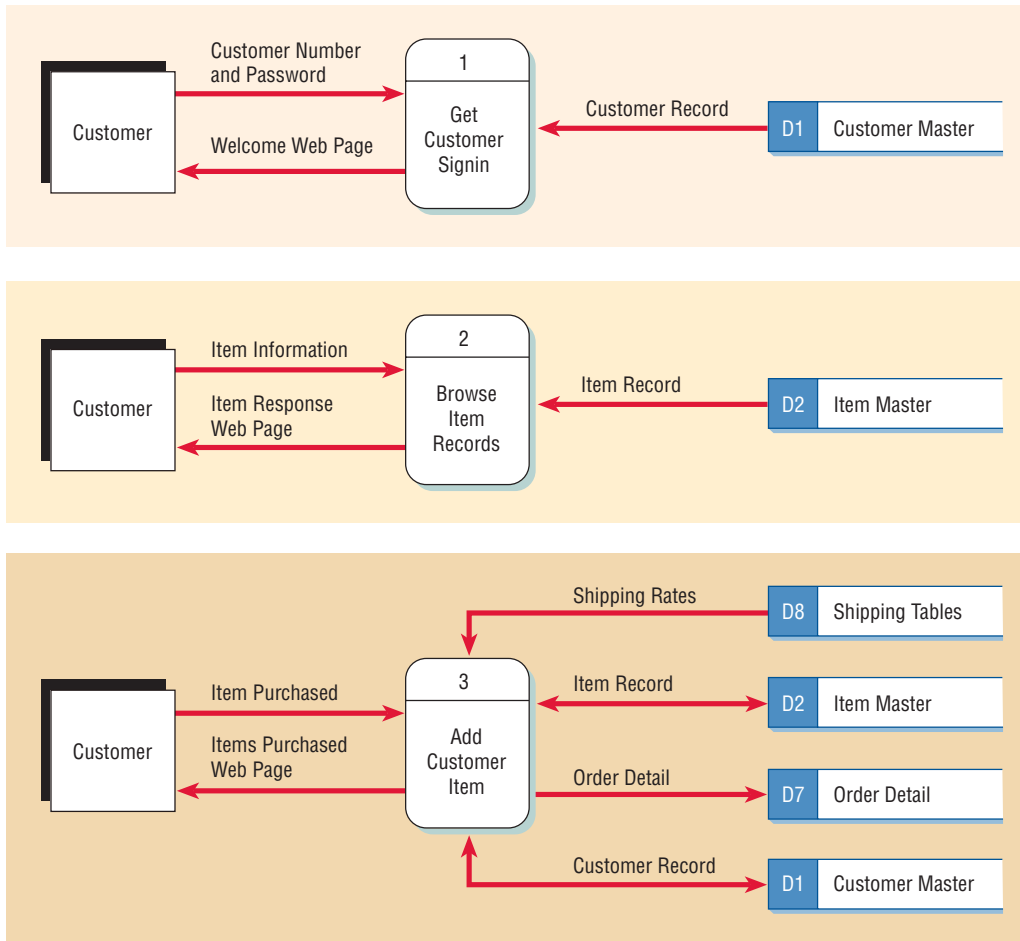
possible, list them in the sequence that they would normally be executed. Next, determine the data used by each step. This step is easier if a data dictionary has been completed. Finally, ask the users to review and suggest modifications of the use cases. It is important that the use cases are written clearly. (See Chapter 10 for a further discussion of UML, use cases, and use case diagrams.)

### Partitioning Data Flow Diagrams

Partitioning is the process of examining a data flow diagram and determining how it should be divided into collections of manual procedures and collections of computer programs. Analyze each process to determine whether it should be a manual or automated procedure. Group automated procedures into a series of computer programs. A dashed line is often drawn around a process or group of processes that should be placed into a single computer program.

There are six reasons for partitioning data flow diagrams:

1. **Different user groups.** Are the processes performed by several different user groups, often at different physical locations in the company? If so, they should be partitioned into different computer programs. An example is the need to process customer returns and customer payments in a department store. Both processes involve obtaining financial information that is used to adjust customer accounts (subtracting from the amount the customer owes), but they are performed by different people at different locations. Each group needs a different screen for recording the particulars of the transaction, either a credit screen or a payment screen.
2. **Timing.** Examine the timing of the processes. If two processes execute at different times, they cannot be grouped into one program. Timing issues may also involve how much data is presented at one time on a Web page. If an ecommerce site has rather lengthy Web pages for ordering items or making an airline reservation, the Web pages may be partitioned into separate programs that format and present the data.

**FIGURE 7.13**

Data flow diagrams for the first three rows of the Internet storefront event response table.

3. **Similar tasks.** If two processes perform similar tasks, they may be grouped into one computer program.
4. **Efficiency.** Several processes may be combined into one program for efficient processing. For example, if a series of reports needs to use the same large input files, producing them together may save considerable computer run time.
5. **Consistency of data.** Processes may be combined into one program for consistency of data. For example, a credit card company may take a “snapshot” and produce a variety of reports at the same time just so figures are consistent.
6. **Security.** Processes may be partitioned into different programs for security reasons. A dashed line may be placed around Web pages that are on a secure server to separate them from those Web pages on a server that is not secured. A Web page that is used for obtaining the user’s identification and password is usually partitioned from order entry or other business pages.

## A DATA FLOW DIAGRAM EXAMPLE

The following example is intended to illustrate the development of a data flow diagram by selectively looking at each of the components explored earlier in this chapter. This example, called “World’s Trend Catalog Division,” will also be used to illustrate concepts covered in Chapters 8 and 9.

### Developing the List of Business Activities

A list of business activities for World’s Trend can be found in Figure 7.15. You could develop this list using information obtained through interacting with people in interviews, through investigation, and through observation. The list can be used to identify external entities such as CUSTOMER, ACCOUNTING, and WAREHOUSE as well as data flows such as ACCOUNTS



Use case name: <i>Add Customer Item</i>		Process ID: 3	
Description: <i>Adds an item for a customer Internet order.</i>			
Trigger: <i>Customer places an order item in the shopping basket.</i>			
Trigger type: External <input checked="" type="checkbox"/> Temporal <input type="checkbox"/>			
Input Name	Source	Output Name	Destination
<i>Item Purchased (Item Number and Quantity)</i>	<i>Customer</i>	<i>Items Purchased Confirmation Web Page</i>	<i>Customer</i>

Steps Performed	Information for Steps
1. Find Item Record using the Item Number. If the item is not found, place a message on the Items Purchased Web page.	Item Number, Item Record
2. Store item data on Order Detail Record.	Order Detail Record
3. Use the Customer Number to find the Customer Record.	Customer Number, Customer Record
4. Calculate Shipping Cost using shipping tables. Using the Item Weight from the Item Record and the Zip Code from the Customer Record, look up the Shipping Cost in the Shipping Tables.	Zip Code, Item Weight, Shipping Table
5. Modify the Customer Total using the Quantity Purchased and the Item Price. Add the Shipping Cost. Update the Customer Record.	Item Record, Quantity Purchased, Shipping Cost, Customer Record
6. Modify the Item Quantity on Hand and update the Item Record.	Quantity Ordered, Item Record

**FIGURE 7.14**

A use case form for the Internet storefront describes the Add Customer Item activity and its triggers, input, and output.

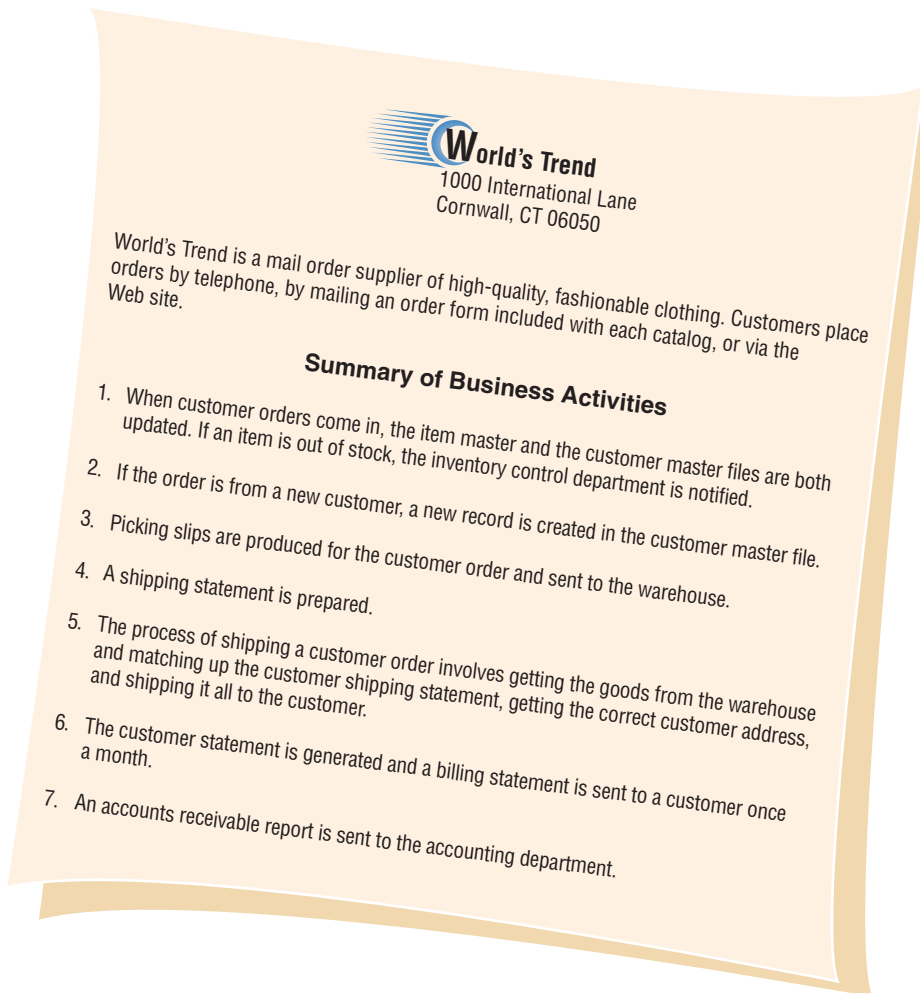
RECEIVABLE REPORT and CUSTOMER BILLING STATEMENT. Later (when developing level 0 and child diagrams), the list can be used to define processes, data flows, and data stores.

### Creating a Context-level Data Flow Diagram

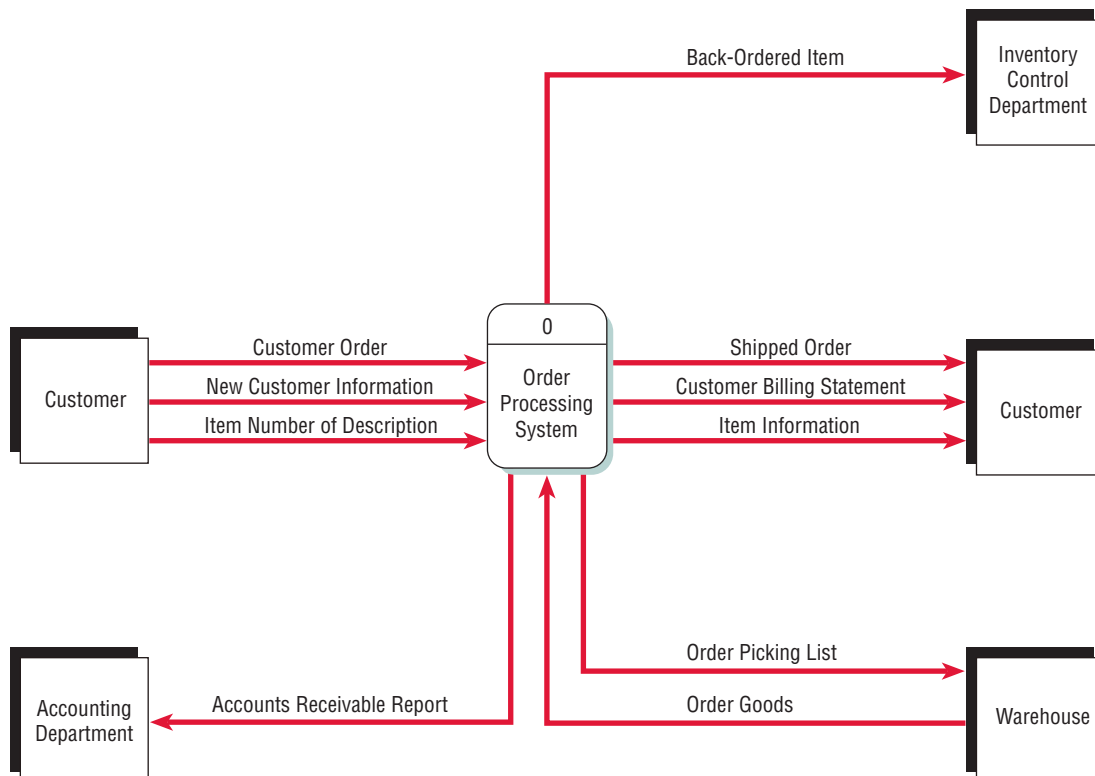
Once this list of activities is developed, create a context-level data flow diagram as shown in Figure 7.16. This diagram shows the ORDER PROCESSING SYSTEM in the middle (no processes are described in detail in the context-level diagram) and five external entities (the two

**FIGURE 7.15**

A summary of business activities for World's Trend Catalog Division.

**FIGURE 7.16**

A context-level data flow diagram for the order processing system at World's Trend.



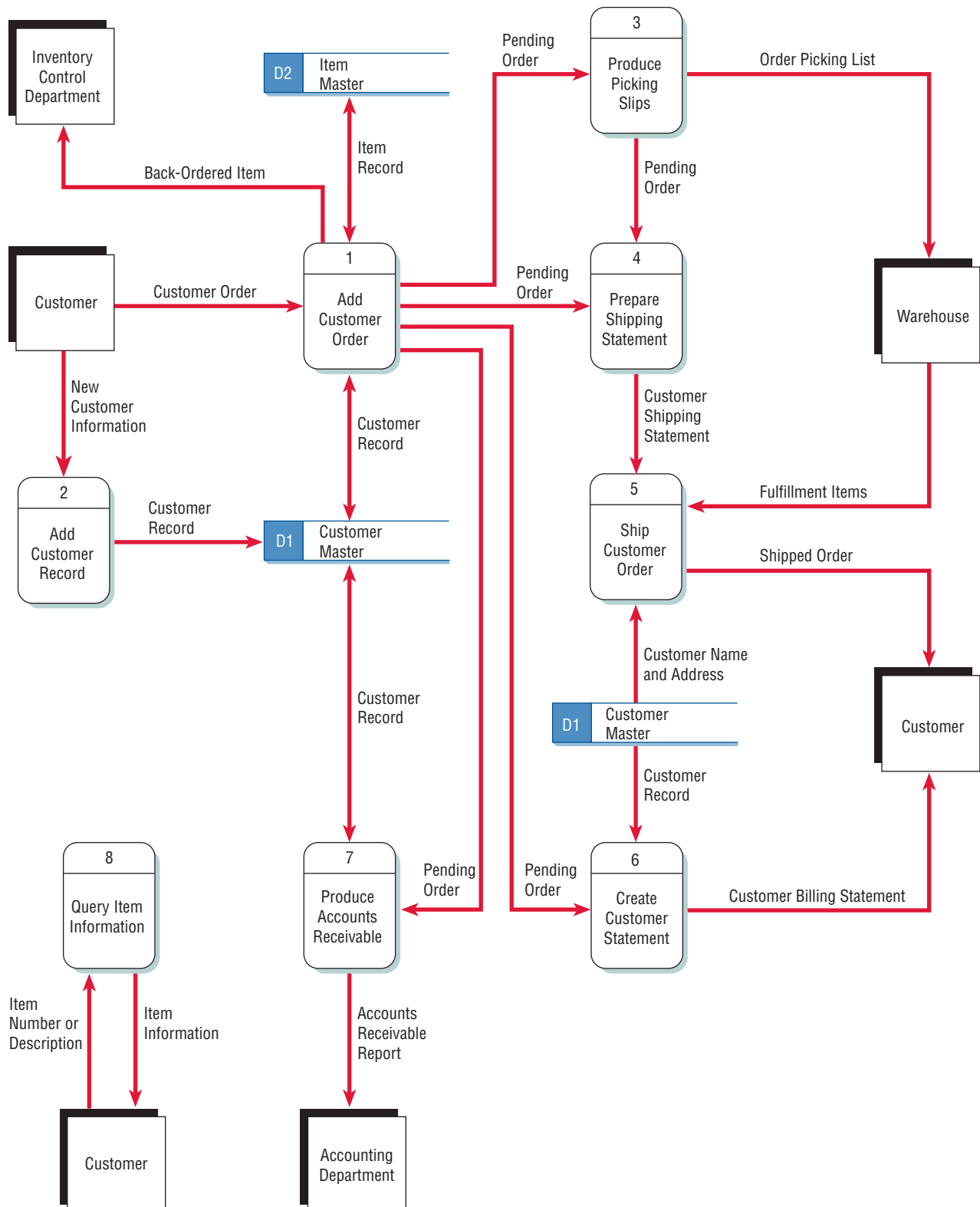
separate entities both called CUSTOMER are really one and the same). The data flows that come from and go to the external entities are shown as well (for example, CUSTOMER ORDER and ORDER PICKING LIST).

**FIGURE 7.17**

Diagram 0, of the order processing system for World's Trend Catalog Division.

### Drawing Diagram 0

Next, go back to the activity list and make a new list of as many processes and data stores as you can find. You can add more later, but start making the list now. If you think you have enough information, draw a level 0 diagram such as the one found in Figure 7.17. Call this Diagram 0 and keep



the processes general so as not to overcomplicate the diagram. Later, you can add detail. When you are finished drawing the seven processes, draw data flows between them and to the external entities (the same external entities shown in the context-level diagram). If you think there need to be data stores such as ITEM MASTER or CUSTOMER MASTER, draw those in and connect them to processes using data flows. Now take the time to number the processes and data stores. Pay particular attention to making the labels meaningful. Check for errors and correct them before moving on.

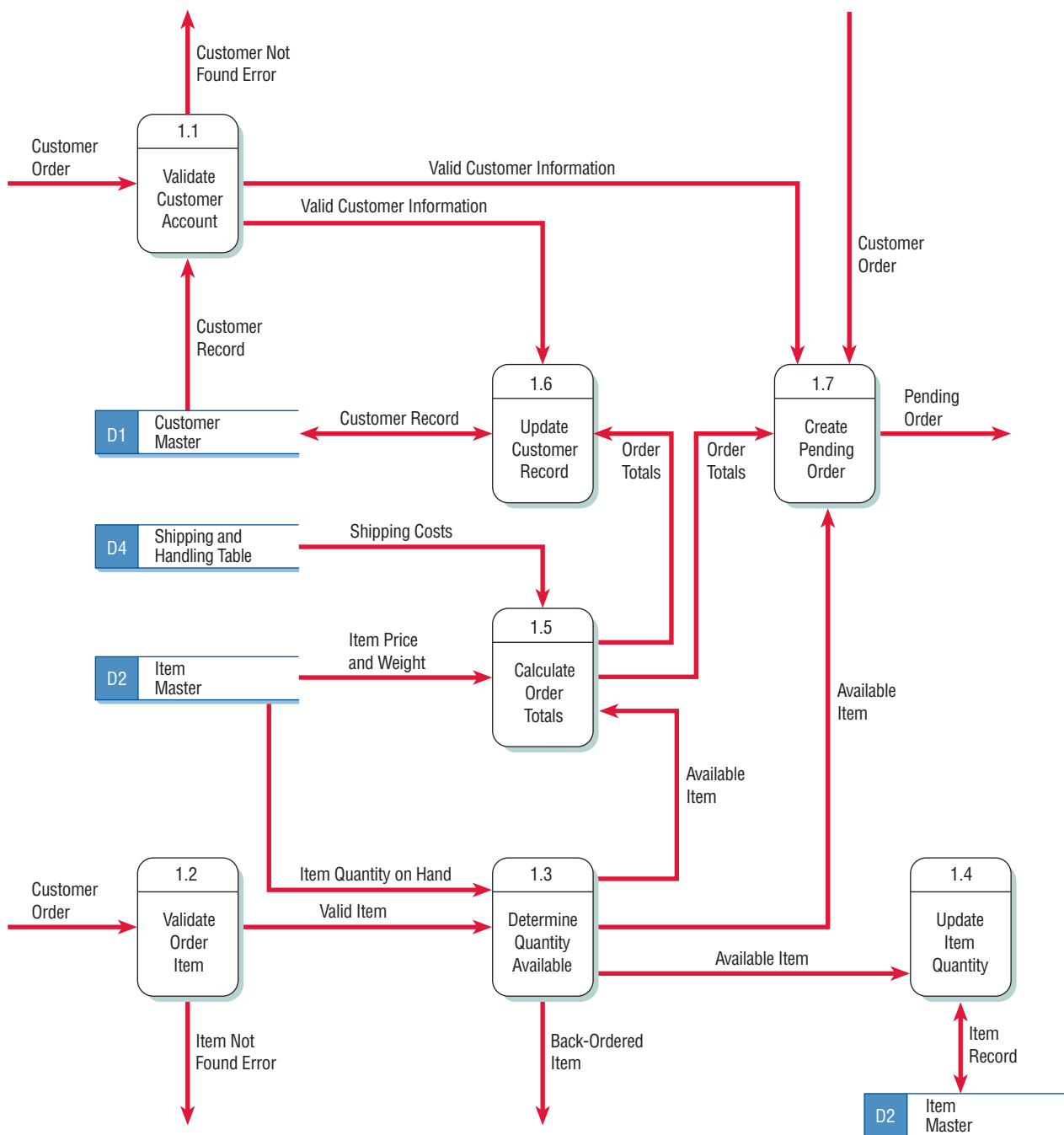
### Creating a Child Diagram

At this point try to draw a child diagram (sometimes also called a level 1 diagram) such as the one in Figure 7.18. Child diagram processes are more detailed, illustrating the logic required to produce the output. Number your child diagrams Diagram 1, Diagram 2, and so on, in accordance with the number you assigned to each process in the level 0 diagram.

When you draw a child diagram, make a list of subprocesses first. A process such as ADD CUSTOMER ORDER can have subprocesses (in this case, there are seven). Connect these subprocesses

**FIGURE 7.18**

Diagram 1, of the order processing system for World's Trend Catalog Division.



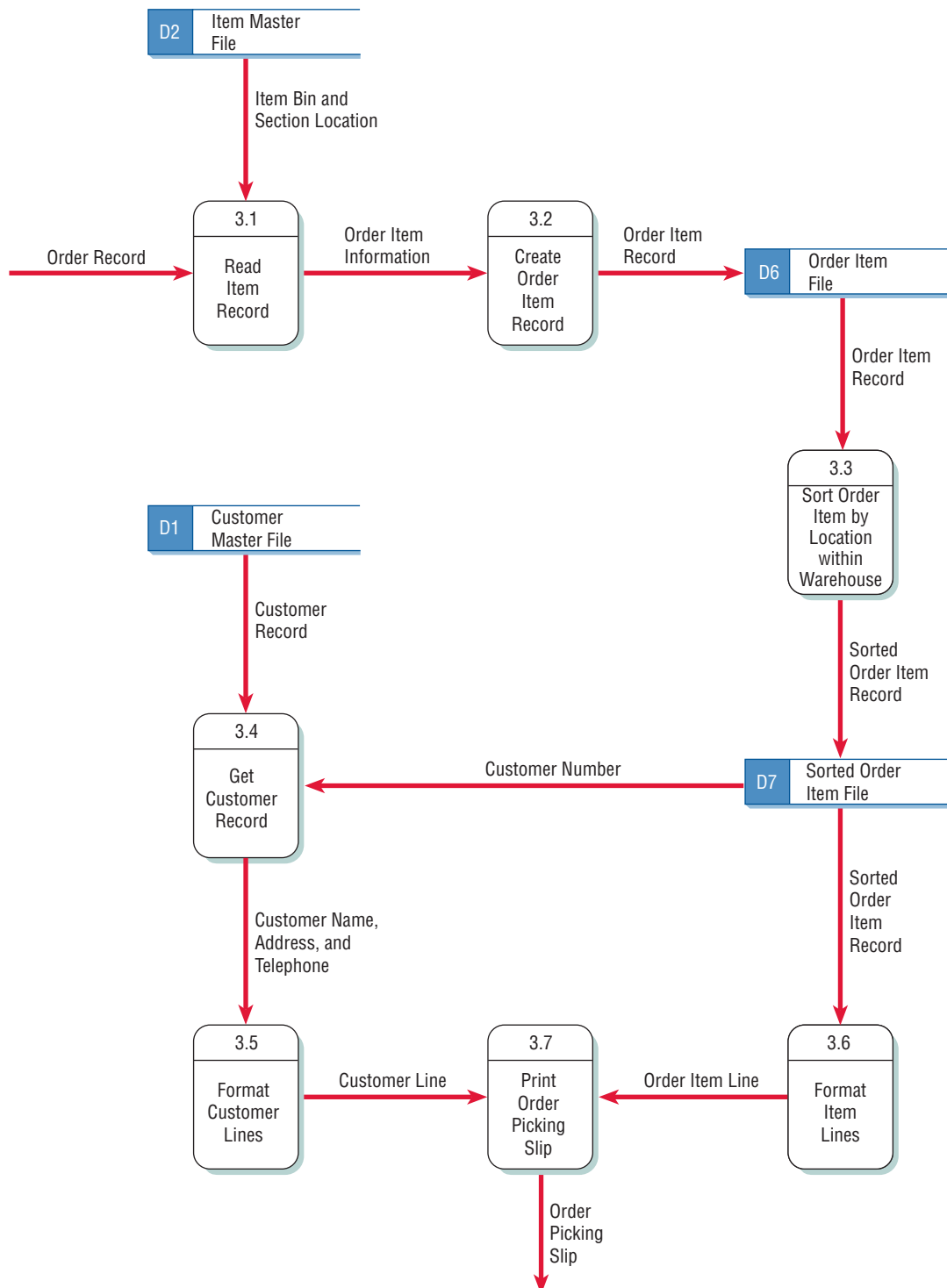
to one another and also to data stores when appropriate. Subprocesses do not have to be connected to external entities, because we can always refer to the parent (or level 0) data flow diagram to identify these entities. Label the subprocesses 1.1, 1.2, 1.3, and so on. Take the time to check for errors and make sure the labels make sense.

### Creating a Physical Data Flow Diagram from the Logical DFD

If you want to go beyond the logical model and draw a physical model as well, look at Figure 7.19, which is an example of a physical data flow child diagram of process 3, PRODUCE PICKING SLIPS. Physical DFDs give you the opportunity to identify processes for scanning bar codes, dis-

**FIGURE 7.19**

A physical data flow child diagram for World's Trend Catalog Division.



playing screens, locating records, and creating and updating files. The sequence of activities is important in physical DFDs, because the emphasis is on how the system will work and in what order events happen.

When you label a physical model, take care to describe the process in great detail. For example, subprocess 3.3 in a logical model could simply be SORT ORDER ITEM, but in the physical model, a better label is SORT ORDER ITEM BY LOCATION WITHIN CUSTOMER. When you write a label for a data store, refer to the actual file or database, such as CUSTOMER MASTER FILE or SORTED ORDER ITEM FILE. When you describe data flows, describe the actual form, report, or screen. For example, when you print a slip for order picking, call the data flow ORDER PICKING SLIP.

### Partitioning the Physical DFD

Finally, take the physical data flow diagram and suggest partitioning through combining or separating the processes. As stated earlier, there are many reasons for partitioning: identifying distinct processes for different user groups, separating processes that need to be performed at different times, grouping similar tasks, grouping processes for efficiency, combining processes for consistency, or separating them for security. Figure 7.20 shows that partitioning is useful in the case of World's Trend Catalog Division. You would first group processes 1 and 2 because it would make sense to add new customers at the same time their first order was placed. You would then put processes 3 and 4 in two separate partitions because these must be done at different times from each other and thus cannot be grouped into a single program.

The process of developing a data flow diagram is now completed from the top down, first drawing a companion physical data flow diagram to accompany the logical data flow diagram, then partitioning the data flow diagram by grouping or separating the processes. The World's Trend example is used again in Chapters 8 and 9.

## PARTITIONING WEB SITES

Partitioning is a very useful principle when designing a Web site. Web site designers who use forms to collect data may find it more appropriate to divide a Web site into a series of Web pages, which will improve the way humans use the site, the speed of processing, and the ease of maintaining the site. Each time data must be obtained from a data store or an external partner, a Web site designer might consider creating a unique Web form and DFD process to validate and process the data.

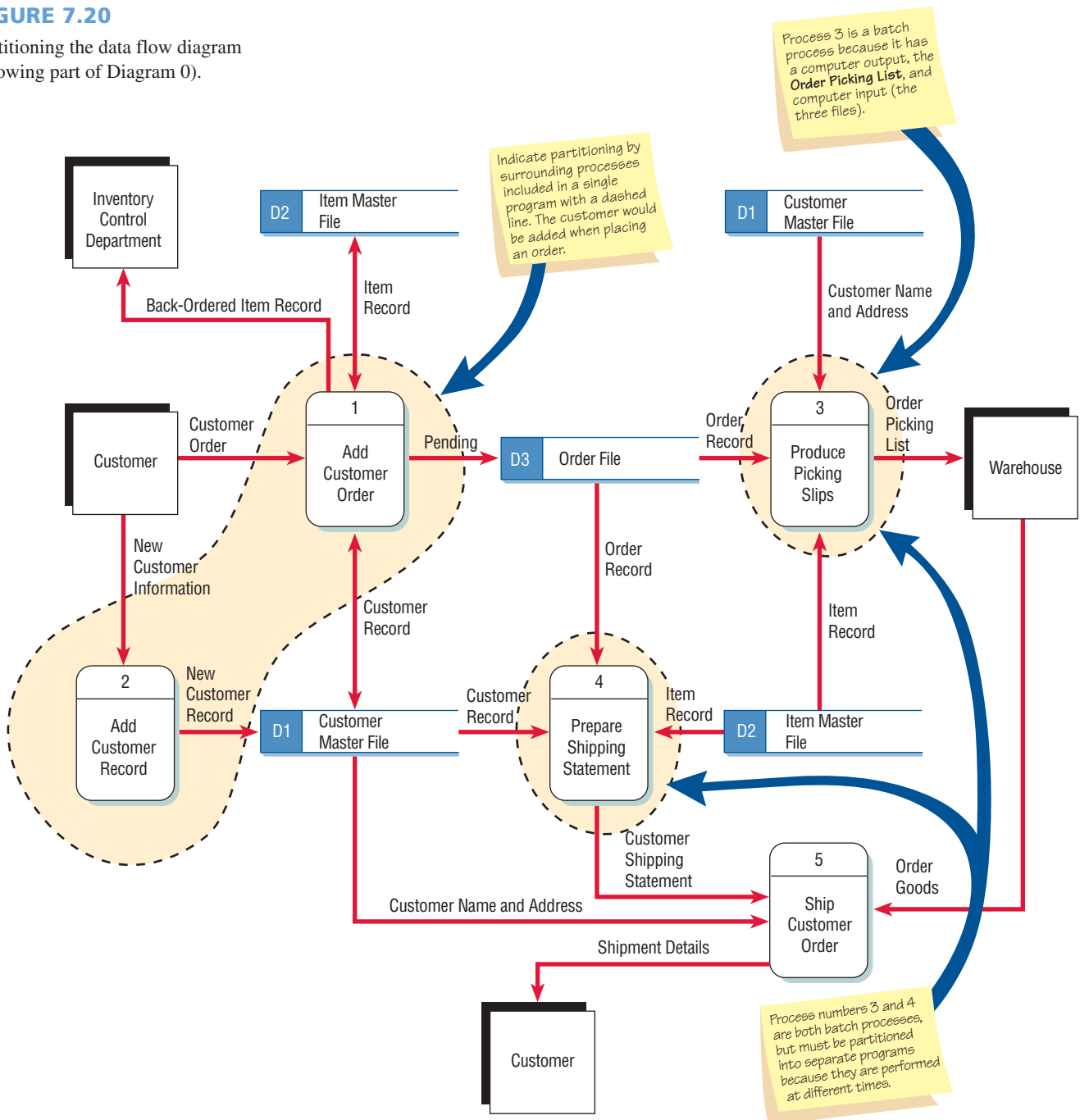
The Web developer may also use Ajax, sending a request to the server and obtaining a small amount of data or an XML document returned to the same page. Ajax may be used to avoid creating too many small pages containing only a few extra or changed Web form elements. However, the analyst should create several pages when needed. One consideration is when a large amount of data needs to be obtained from the server, such as a list of all the flights that match starting and destination airports for specific travel days. When accessing different database tables on the same database, the data may be obtained containing fields from different database tables and passed to one process. However, if different databases are involved, the analyst may decide to use separate Web pages. When user input is required, the analyst may either use separate Web pages or use Ajax to facilitate a change in a drop-down list or to change a small amount of data.

A good example of partitioning can be seen in the development of a Web-based travel booking site. To simplify, we will only look at the airline booking portion of the Web site, shown in the data flow diagram in Figure 7.21. Notice that the Web designer has chosen to create several processes and unique partitions in making a flight reservation. Process 1 receives and validates the dates and airports entered by the customer (or travel agent acting for a customer). The selection data is used to obtain flight details and create a transaction data store of flight details that match the flight request.

It is advisable to partition the process of finding the flight information as a separate process, because a data store must be searched and the flight details are used to display a series of successive Web pages with matching flights. Then, once a customer chooses a flight, the information must be sent to a selected airline. It is important to have the FLIGHT DETAILS transaction file available to display each Web page of new flights, because redoing the search may take a lengthy amount of time that is unacceptable to a human user trying to complete a transaction.

**FIGURE 7.20**

Partitioning the data flow diagram  
(showing part of Diagram 0).



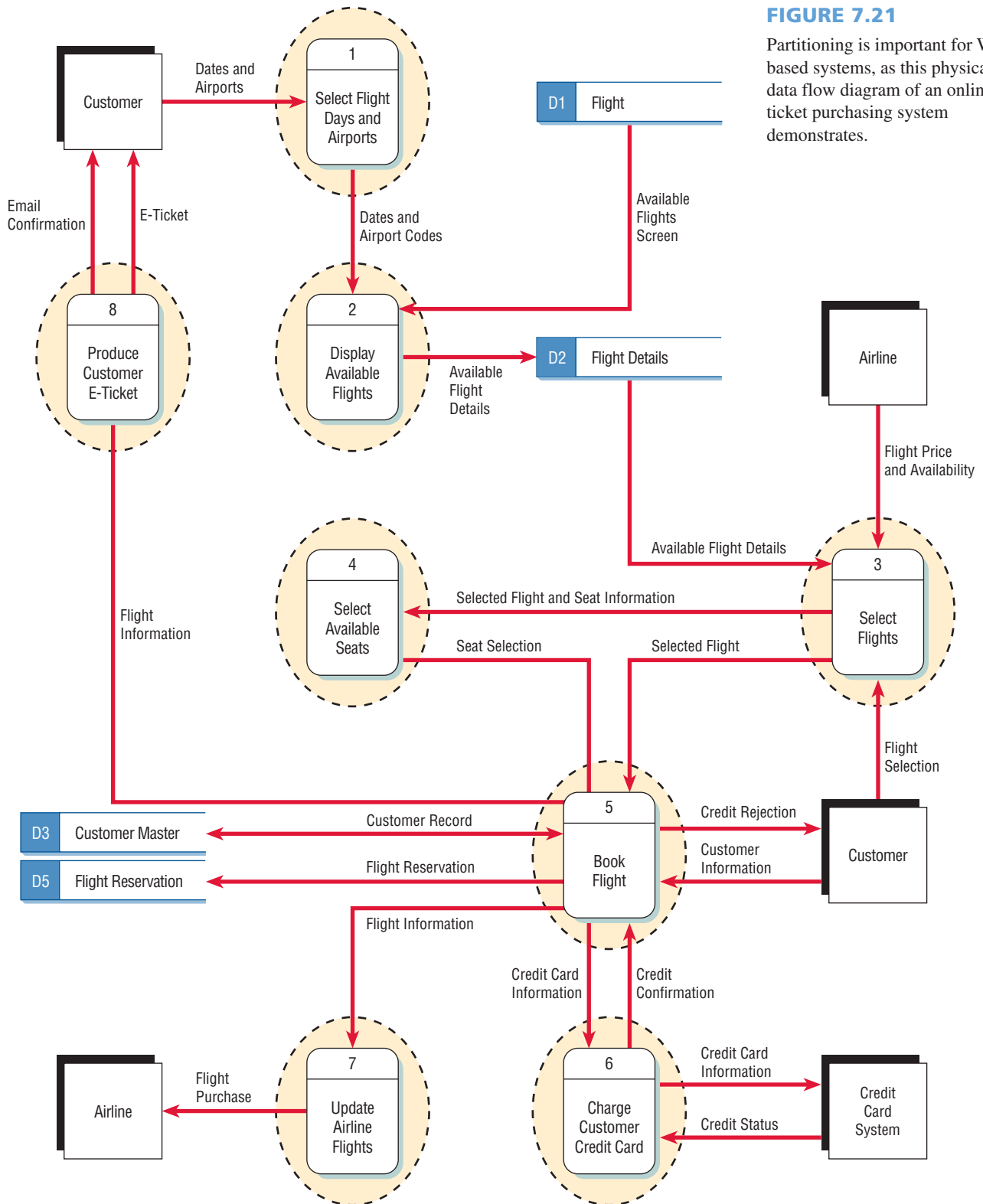
The selection of available flights (process 2) uses an internal database, but this database does not have information about availability of seats, because the airlines are receiving reservations from many travel service organizations. This means that there must be a separate process and small program partitioned for determining if seats are available and for reserving specific seats.

Because there is a lot of user input, forms are designed to handle all the user requests. Having separate forms means that the forms are less complex, and therefore users will find them more attractive and easier to fill out. This design meets both the usability and usefulness criteria important when designing Web sites for human–computer interaction. It also means that processing will take place more quickly, because once the flight is chosen, the next step involving the choice of seats should not require the customer to input or even see the flight details again at this time. Most airline Web sites now use pop-up windows in which customers point to their seat selection.



**FIGURE 7.21**

Partitioning is important for Web-based systems, as this physical data flow diagram of an online ticket purchasing system demonstrates.



Another reason for partitioning is to keep the transaction secure. Once the seat has been selected, the customer must confirm the reservation and supply credit card information. This is done using a secure connection, and the credit card company is involved in validating the amount of purchase. The secure connection means a separate process must be used. Once the credit card has been confirmed, two additional processes must be included, one to format and send an email confirmation and an e-ticket to the customer, and another to send notification of the flight purchase to the airline.



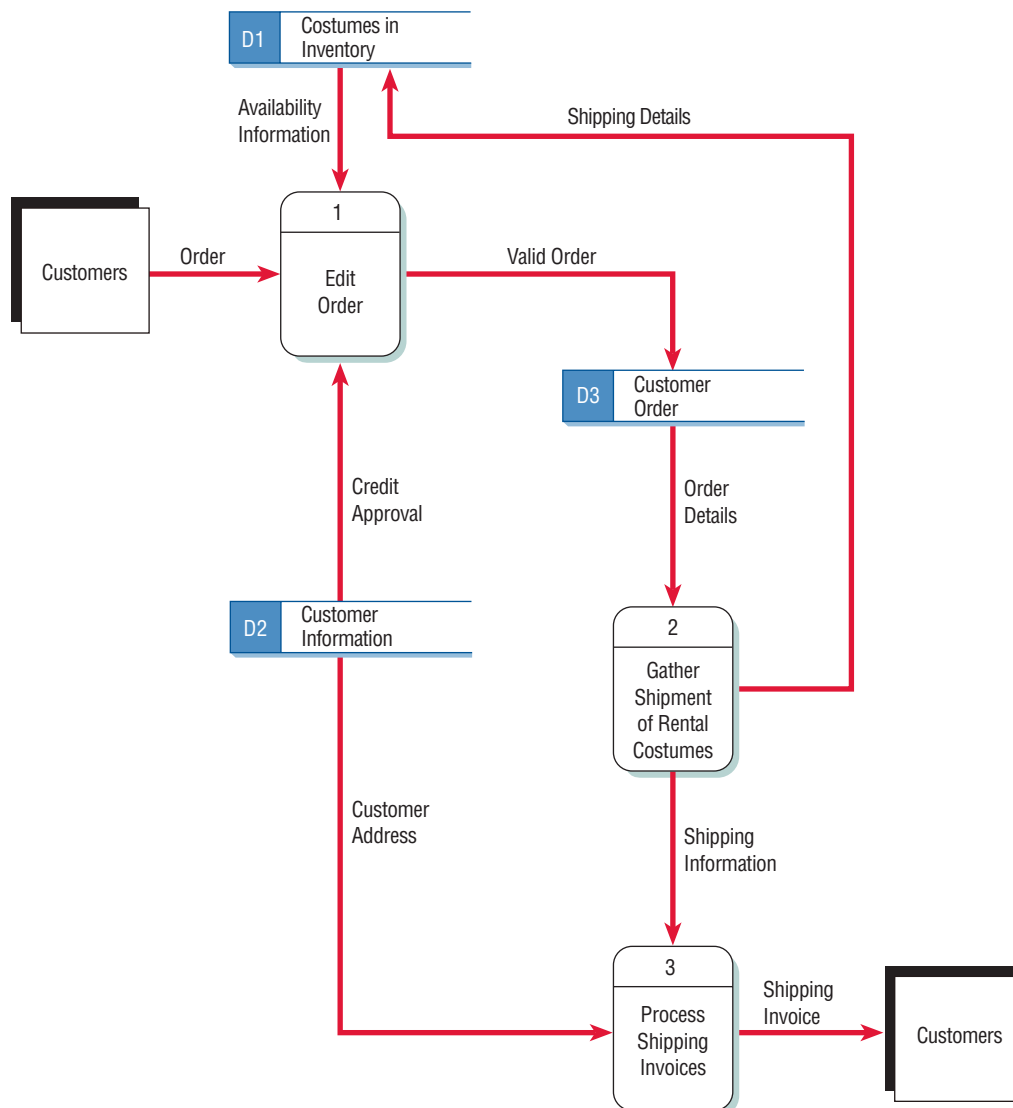
## CONSULTING OPPORTUNITY 7.1

## There's No Business Like Flow Business

The phone at Merman's Costume Rentals rings, and Annie Oaklea, head of costume inventory, picks it up and answers a query by saying, "Let me take a look at my inventory cards. Sorry, it looks as if there are only two male bear suits in inventory, with extra growly expressions at that. We've had a great run on bear. When do you need them? Perhaps one will be returned. No, can't do it, sorry. Would you like these two sent, regardless? The name of your establishment? Manhattan Theatre Company? London branch? Right. Delightful company! I see by our account card that you've rented from us before. And how long will you be needing the costumes?"

Figure 7.C1 is a data flow diagram that sets the stage for processing of costume rentals from Merman's. It shows rentals such as the one Annie is doing for Manhattan Theatre Company.

After conversing for another few moments about shop policy on alterations, Annie concludes her conversation by saying, "You are very lucky to get the bears on such short notice. I've got another company reserving them for the first week in July. I'll put you down for the bear suits, and they'll be taken to you directly by our courier. As always, prompt return will save enormous trouble for us all."



**FIGURE 7.C1**

A data flow diagram for Merman's Costume Rentals.

Merman's costume rental enterprise is located in London's world-famous West End theatre district. When a theatre or television production company lacks the resources (either time or expertise) to construct a costume in its own shop, the cry goes up to "Ring up Merman's!" and it proceeds to rent what it needs with a minimum of fuss.

The shop (more aptly visualized as a warehouse) goes on for three floors full of costume racks, holding thousands of costumes hung together by historical period, then grouped as to whether they

are for men or women, and then by costume size.<sup>1</sup> Most theatre companies are able to locate precisely what they need through Annie's capable assistance.

Now tailor-make the *rental return* portion of the data flow diagram given earlier. Remember that timely returns are critical for keeping the spotlight on costumes rented from Merman's.

<sup>1</sup>Western Costume Company in Hollywood, California, is said to have more than 1 million costumes worth about \$40 million.

The entire procedure must be partitioned into a series of interacting processes, each with a corresponding Web page or interaction with an external system. Each time a new data store is used to obtain additional data, a process must be included to format or obtain the data. Each time an external company or system is involved, a process needs to be partitioned into a separate program. When processes or forms need to be revised, it is not a major task. The small size of the programs makes them easy to change. In this way, the Web site is secure, efficient, and more easily maintained.

## COMMUNICATING USING DATA FLOW DIAGRAMS

Data flow diagrams are useful throughout the analysis and design process. Use original, unexploded data flow diagrams early when ascertaining information requirements. At this stage they can help provide an overview of data movement through the system, lending a visual perspective unavailable in narrative data.

A systems analyst might be quite competent at sketching through the logic of the data stream for data flow diagrams, but to make the diagrams truly communicative to users and other members of the project team, meaningful labels for all data components are also required. Labels should not be generic, because then they do not tell enough about the situation at hand. All general systems models bear the configuration of input, process, and output, so labels for a data flow diagram need to be more specific than that.

Finally, remember that data flow diagrams are used to document the system. Assume that data flow diagrams will be around longer than the people who drew them, which is, of course, always true if an external consultant is drawing them. Data flow diagrams can be used for documenting high or low levels of analysis and helping to substantiate the logic underlying the data flows of the organizations.

## SUMMARY

To better understand the logical movement of data throughout a business, the systems analyst draws data flow diagrams (DFDs). Data flow diagrams are structured analysis and design tools that allow the analyst to comprehend the system and subsystems visually as a set of interrelated data flows.

Graphical representations of data movement storage and transformation are drawn with the use of four symbols: a rounded rectangle to depict data processing or transformations, a double square to show an outside data entity (source or receiver of data), an arrow to depict data flow, and an open-ended rectangle to show a data store.

The systems analyst extracts data processes, sources, stores, and flows from early organizational narratives or stories told by users or revealed by data and uses a top-down approach to first draw a context-level data flow diagram of the system within the larger picture. Then a level 0 logical data flow diagram is drawn. Processes are shown and data stores are added. Next, the analyst creates a child diagram for each of the processes in Diagram 0. Inputs and outputs remain constant, but the data stores and sources change. Exploding the original data flow diagram allows the systems analyst to focus on ever more detailed depictions of data movement in the system. The analyst then develops a physical data flow diagram from the logical data flow diagram, partitioning it to facilitate programming. Each process is analyzed to determine whether it should be a manual or automated procedure.

Six considerations for partitioning data flow diagrams include whether processes are performed by different user groups, processes execute at the same times, processes perform similar tasks, batch processes can be combined for efficient processing, processes may be combined into one program for consistency of data, or processes may be partitioned into different programs for security reasons.



## HYPERCASE® EXPERIENCE 7

“You take a very interesting approach to the problems we have here at MRE. I’ve seen you sketching diagrams of our operation almost since the day you walked in the door. I’m actually getting used to seeing you doodling away now. What did you call those? Oh, yes. Context-level diagrams. And flow networks? Oh, no. Data flow diagrams. That’s it, isn’t it?”

### HYPERCASE Questions

1. Find the data flow diagrams already drawn in MRE. Make a list of those you found and add a column to show where in the organization you found them.
2. Draw a context-level diagram modeling the Training Unit Project Development process, one that is based on interviews with relevant Training Unit staff. Then draw a level 0 diagram detailing the process.

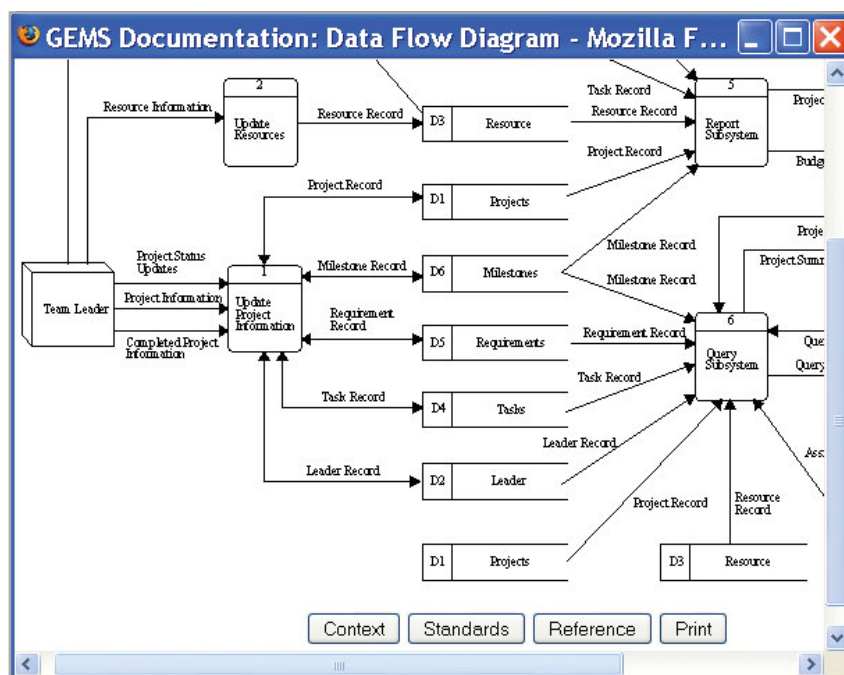


FIGURE 7.HC1

In HyperCase you can click on elements in a data flow diagram.

## KEYWORDS AND PHRASES

Ajax  
base element  
child diagram  
context-level data flow diagram  
data flow diagram  
data flow diagram fragment  
data-oriented system  
data store  
derived elements  
event modeling  
event response table  
event trigger

exploding  
external entity (source or destination)  
functionally primitive  
interface data flow  
level 0 diagram  
logical model  
online process  
parent process  
partitioning  
physical data store  
physical model  
primitive process

top-down approach  
transaction data store  
transforming process

unified modeling language (UML)  
use case  
vertical balancing

## REVIEW QUESTIONS

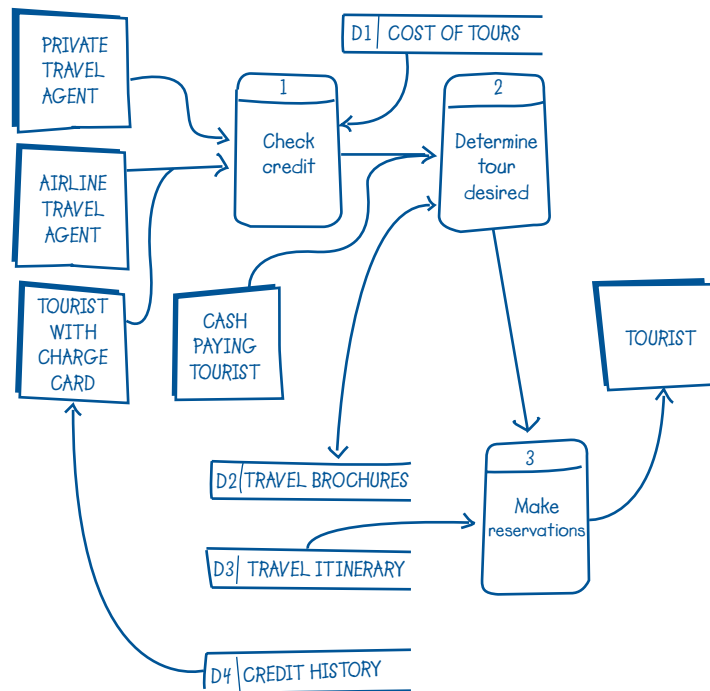
1. What is one of the main methods available for the analyst to use when analyzing data-oriented systems?
2. What are the four advantages of using a data flow approach over narrative explanations of data movement?
3. What are the four data items that can be symbolized on a data flow diagram?
4. What is a context-level data flow diagram? Contrast it to a level 0 DFD.
5. Define the top-down approach as it relates to drawing data flow diagrams.
6. Describe what “exploding” data flow diagrams means.
7. What are the trade-offs involved in deciding how far data streams should be exploded?
8. Why is labeling data flow diagrams so important? What can effective labels on data flow diagrams accomplish for those unfamiliar with the system?
9. What is the difference between a logical and physical data flow diagram?
10. List three reasons for creating a logical data flow diagram.
11. List five characteristics found on a physical data flow diagram that are not on a logical data flow diagram.
12. When are transaction files required in the system design?
13. How can an event table be used to create a data flow diagram?
14. List the major sections of a use case.
15. How can a use case be used to create a data flow diagram?
16. What is partitioning, and how is it used?
17. How can an analyst determine when a user interface is required?
18. List three ways of determining partitioning in a data flow diagram.
19. List three ways to use completed data flow diagrams.

## PROBLEMS

1. Up to this point you seem to have had excellent rapport with Kevin Cahoon, the owner of a musical instrument manufacturing company. When you showed him a set of data flow diagrams you drew, he wasn’t able to see how the system you were proposing was described in the diagrams.
  - a. In a paragraph, write down in general terms how to explain a data flow diagram to a user. Be sure to include a list of symbols and what they mean.
  - b. It takes some effort to educate users about data flow diagrams. Is it worthwhile to share them with users? Why or why not? Defend your response in a paragraph.
  - c. Compare data flow diagrams to use cases and use case scenarios. What do data flow diagrams show that use case diagrams have a difficult time trying to explain?
2. Your latest project is to combine two systems used by Producers Financial. Angie Schworer’s loan application system is fairly new, but has no documentation. Scott Wittman’s loan management system is older, needs much revision, and the records are coded and kept independently of the other system. The loan application system accepts applications, processes them, and recommends loans for approval. The loan management system takes loans that have been approved and follows them through their final disposition (paid, sold, or defaulted). Draw a context diagram and a level 1 data flow diagram that shows what an idealized combined system would look like.
3. One common experience that students in every college and university share is enrolling in a college course.
  - a. Draw a level 1 data flow diagram of data movement for enrollment in a college course. Use a single sheet and label each data item clearly.
  - b. Explode one of the processes in your original data flow diagram into subprocesses, adding data flows and data stores.
  - c. List the parts of the enrollment process that are “hidden” to the outside observer and about which you have had to make assumptions to complete a second-level diagram.
4. Figure 7.EX1 is a level 1 data flow diagram of data movement in a Niagara Falls tour agency called Marilyn’s Tours. Read it over, checking for any inaccuracies.
  - a. List and number the errors that you have found in the diagram.
  - b. Redraw and label the data flow diagram of Marilyn’s so that it is correct. Be sure that your new diagram employs symbols properly so as to cut down on repetitions and duplications where possible.

**FIGURE 7.EX1**

A hand-sketched data flow diagram for Marilyn's Tours.



5. Perfect Pizza wants to install a system to record orders for pizza and chicken wings. When regular customers call Perfect Pizza on the phone, they are asked their phone number. When the number is typed into a computer, the name, address, and last order date is automatically brought up on the screen. Once the order is taken, the total, including tax and delivery, is calculated. Then the order is given to the cook. A receipt is printed. Occasionally, special offers (coupons) are printed so the customer can get a discount. Drivers who make deliveries give customers a copy of the receipt and a coupon (if any). Weekly totals are kept for comparison with last year's performance. Write a summary of business activities for taking an order at Perfect Pizza.
6. Draw a context-level data flow diagram for Perfect Pizza (Problem 5).
7. Explode the context-level diagram in Problem 6 showing all the major processes. Call this Diagram 0. It should be a logical data flow diagram.
8. Draw a logical child diagram for Diagram 0 in Problem 7 for the process that adds a new customer if he or she is not currently in the database (has never ordered from Perfect Pizza before).
9. Draw a physical data flow diagram for Problem 7.
10. Draw a physical data flow diagram for Problem 8.
11. Partition the physical data flow diagram in Problem 7, grouping and separating processes as you deem appropriate. Explain why you partitioned the data flow diagram in this manner. (Remember that you do not have to partition the entire diagram, only the parts that make sense to partition.)
12.
  - a. Draw a logical child diagram for process 6 in Figure 7.17.
  - b. Draw a physical child diagram for process 6 in Figure 7.17.
13. Draw a physical data flow diagram for process 1.1 in Figure 7.18.
14. Create a context diagram for a real estate agent trying to create a system that matches buyers with potential houses.
15. Draw a logical data flow diagram showing general processes for Problem 14. Call it Diagram 0.
16. Create a context-level diagram for billing in a dental office. External entities include the patients and insurance companies.
17. Draw a logical data flow diagram showing general processes for Problem 16. Call it Diagram 0.
18. Create an event response table for the activities listed for World's Trend order processing system.
19. Create a use case for the list of seven processes for the World's Trend order processing system.
20. Create a CRUD matrix for the files of World's Trend.
21. Use the principles of partitioning to determine which of the processes in Problem 18 should be included in separate programs.



22. Create a physical data flow child diagram for the following situation: The local PC Users Group holds meetings once a month with informative speakers, door prizes, and sessions for special interest groups. A laptop computer is taken to the meetings, and is used to add the names of new members to the group. The diagram represents an online process and is the child of process 1, ADD NEW MEMBERS. The following tasks are included:
  - a. Key the new member information.
  - b. Validate the information. Errors are displayed on the screen.
  - c. When all the information is valid, a confirmation screen is displayed. The operator visually confirms that the data are correct and either accepts the transaction or cancels it.
  - d. Accepted transactions add new members to the MEMBERSHIP MASTER file, which is stored on the laptop hard drive.
  - e. Accepted transactions are written to a MEMBERSHIP JOURNAL file, which is stored on a second hard drive.

## GROUP PROJECTS

1. Meet with your group to develop a context-level data flow diagram for Maverick Transport (first introduced in Chapter 4). Use any data you have subsequently generated with your group about Maverick Transport. (*Hint:* Concentrate on one of the company's functional areas rather than try to model the entire organization.)
2. Using the context-level diagram developed in Problem 1, develop with your group a level 0 logical data flow diagram for Maverick Transport. Make any assumptions necessary to draw it. List them.
3. With your group, choose one key process and explode it into a logical child diagram. Make any assumptions necessary to draw it. List follow-up questions and suggest other methods to get more information about processes that are still unclear to you.
4. Use the work your group has done to date to create a physical data flow diagram of a portion of the new system you are proposing for Maverick Transport.

## SELECTED BIBLIOGRAPHY

- Ambler, S. W., and L. L. Constantine (Eds.). *The Unified Process Inception Phase: Best Practices for Implementing the UP*. Lawrence, KS: CMP Books, 2000.
- Gane, C., and T. Sarson. *Structured Systems Analysis and Design Tools and Techniques*. Englewood Cliffs, NJ: Prentice Hall, 1979.
- Hoffer, J. A., M. Prescott, and H. Topi. *Modern Database Management*, 9th ed. Upper Saddle River: Prentice Hall, 2009.
- Kotonya, G., and I. Sommerville. *Requirements Engineering: Processes and Techniques*. New York: John Wiley & Sons, 1999.
- Lucas, H. *Information Systems Concepts for Management*, 3d ed. New York: McGraw-Hill, 1986.
- Martin, J. *Strategic Data-Planning Methodologies*. Englewood Cliffs, NJ: Prentice Hall, 1982.
- Thayer, R. H., M. Dorfman, and D. Garr. *Software Engineering: Vol. 1: The Development Process*, 2d ed. New York: Wiley-IEEE Computer Society Press, 2002.