The massive recall of Firestone Tires in the autumn of 2000, particularly those on Ford Explorer SUVs, produced a major financial impact on both Firestone (including its parent Bridgestone) and Ford Motor Company. The sweeping recall occurred as tire blowouts and related vehicle rollovers were reported at alarming rates. The tread on certain brands and sizes of Firestone tires had a tendency to separate—particularly if the tires were under-inflated, driven at high speeds in hot climates, or carrying a heavy load. The result has been numerous lawsuits filed against both Bridgestone/Firestone and Ford Motor Co.

The burning question is why didn’t Ford and/or Firestone discover the problem earlier? One reason was that Ford “lack[ed] a database it could use to determine whether incident reports on one type or brand of tire represented a deviation from those of other tires on Ford vehicles.”1 As a result, Ford did not identify the problem until the public relations damage was severe and then only after organizing a team to pore over the documentation on hand in the offices of Firestone. If a database of information related to tire problems had been available, standard data mining techniques likely would have detected the information much earlier. In this chapter, we will explore the advantages of database management systems and related analysis tools that can improve the decision support required for timely decision making.

Synopsis

This chapter introduces approaches used to process data related to major business events that take place in an organization, such as purchasing materials, manufacturing products, and filling customer orders. Data from these events are recorded and processed using differing systems designs. As these business events are processed, data are recorded in files, tables, databases, and so on. The management of data comprises two distinct processes—event data processing and data maintenance. Having laid this foundation, we build your understanding by describing event-driven approaches and the various uses of databases to facilitate data management. Throughout this discus-

sion we consider how various control procedures can enhance accuracy and safeguarding of data. Finally, in the appendices, we discuss the processes used in the design and implementation of entity-relationship (E-R) models and databases.

**LEARNING OBJECTIVES**

- To describe and analyze the major approaches used to process data related to business events
- To describe the major business events in merchandising, service and manufacturing firms
- To explain the complexities and limitations of using traditional data management approaches
- To recognize the advantages of using the database approach to data management
- To be able to perform the basic processes involved in database design and implementation

**Introduction**

An organization engages in various business processes—such as hiring employees, purchasing inventory, and collecting cash from customers—and the activities that occur during execution of these business processes are referred to as events. **Event data processing** is the process whereby event-related data are collected and stored. This chapter describes event data processing, discusses the major approaches employed to capture, process, and store event data, and recounts the types of data collected in event data processing systems. After introducing the major types of events, the text describes a crucial element of the Information System—the data. You need to know how data and databases will become an integral part of your day-to-day work. What data do we collect? How do we collect the data, store the data, and use the data?

Consider the importance of the groundwork laid in this chapter. In Chapters 4 and 5, we focus on advanced techniques for managing data and speeding the delivery of information. In Chapters 6 and 7, we emphasize techniques for developing good Information Systems that capture and deliver the right data. In Chapters 8 and 9, we focus on techniques to assure the reliability and security of data. Finally, in Chapters 10–14, we focus on how data are captured and processed across an organization’s business processes. What’s the moral to the story? If you can’t access good, useful data, you can’t make good business decisions.

**Event Data Processing**

In Chapter 2, we studied several types of diagrams in which something happened (e.g., a customer order) that triggered a series of human and automated business activities. These business activities represent the occurrence of a business event. Every firm has a number of business events that link together to form a business process. The nature of
a firm’s business dictates the range of processes it can adopt to achieve its objectives. The firm’s information system will collect, process, and store business event data in support of its business processes. Each of these business processes may be divided into components, or subprocesses.

Take, for instance, a merchandising firm. A merchandising firm is an organization (e.g., a store) that buys goods from vendors and resells those goods to customers. On the customer side, its business events within the Order-to-Sales process include:

- Capturing and recording of customer orders
- Shipment of goods and recording of sales
- Sending invoices for goods and recording the amount to be received
- Receipt and recording of payments

Figure 3.1 depicts how the physical processing of these business events is also captured in the data recorded and processed by the merchandising firm’s information system. On the supplier end, business events within the Purchase-to-Pay process include:

- Preparation and recording of purchase orders
- Receipt of goods into inventory and recording the receipt of inventory
- Receipt of vendor invoices and recording of the amount owed
- Preparation and recording of payments

There may be other business processes in a typical merchandising business, such as payroll processing, hiring new employees, and many more.

As another example, a service firm is an organization that sells services, rather than merchandise, to its customers. The business events for a service firm that parallel the Order-to-Sales process include:

- Recording of customer services performed
- Billing for services rendered
- Receipt and recording of payments

Review Question
What business events are typically encountered by a merchandising firm?

Figure 3.1 Overview of Order-to-Sales Process for a Merchandising or Manufacturing Firm
Because there is no physical exchange of goods in a service firm, its Order-to-Cash process would have a slightly modified set of events reflected in its corresponding information system, as shown in Figure 3.2. Service firms also must record other business events, including the purchase of materials used in the performance of service engagements and payroll disbursements.

A manufacturing firm acquires raw materials, converts those materials into finished goods, and sells those goods to its customers. Its production process includes recording activities related to the manufacture of goods for sale. A manufacturing firm must also receive customer orders, record sales, send invoices, and receive customer payments. Its Order-to-Cash process is essentially the same as that of the merchandising firm in Figure 3.1. The events that make up these processes will be described in detail in Chapters 10–14.

As each business event occurs, a firm must record at least a minimal set of data about the event so that it can maintain records and produce reports that help assess how well it is meeting its objectives. Today, virtually all of these records are maintained by a computerized Information System.

An organization’s Information System performs event data processing to support an overall business process and its component subprocesses. For example, we describe the Information System employed to prepare and send a bill to a customer as the “billing” portion of the Order-to-Sales process. Similarly, we describe the Information System that prepares and records a purchase order as the “purchasing” portion of the Purchase-to-Pay process.

**Transaction Processing Approach**

Throughout the preceding discussion of event data processing, the focus was on events that take place within various business processes. Once these events have been identified, data that describe the events are collected, organized, manipulated, summarized, stored, and made available for retrieval. Traditionally, computerized Information Systems were designed around particular events called transactions, those business activities that have an economic impact on the firm. These include sales, payroll, accounts payable, and other typical financial transactions. The data that are recorded by a transaction processing system reflect the minimal information needed to represent each

---

**Figure 3.2 Overview of Order-to-Sales Process for a Service Firm**

**PHYSICAL PROCESSING**

- [Receipt]
  - [Bill Enclosed]

**INFORMATION PROCESSING**

- [Record service provided]
  - [Record billing data]
  - [Record receipt of payment]
transaction, and are stored in a file along with the records of all of the other transactions of the same type. This transaction orientation led to the dominance of “file-centric” techniques for systems design, an approach discussed later in this chapter. This traditional approach to transaction processing worked well when technology was expensive and record keeping was not as sophisticated as it is today.

**Event-Driven Approach**

As society progresses in the information age, users’ expectations of the information they need at their fingertips has escalated dramatically. User information demands have highlighted several fundamental weaknesses in the traditional approach of transaction processing. First, in order for data to be in a format that can be easily summarized, only data related to classification (e.g., a customer account number or an inventory part number) and quantitative descriptions can be captured. Thus, only a very narrow view of the event is portrayed by the data we collect—for instance, maybe only a financial accounting assessment or a listing of available stock in the warehouse. Second, once transaction data have been summarized, descriptions of individual transactions may be lost, with only summary information available to users.

*Event-driven systems* capture a complete description of each event, regardless of its economic impact on the firm, and permanently store the individual descriptions of each event. There are many business events that carry no economic impact, which would not be reflected in traditional transaction-oriented systems. Two examples are a sales representative capturing contact information about a potential customer, and a warehouse clerk updating location information when inventory items are moved from long-term storage to a place where “pickers” can get easy access to them. Neither of these events shows up on financial reports, but both are important for running the business.

Of equal importance, however, is the focus on capturing a wider variety of data about each business event to meet the needs of multiple users. *Transaction processing* systems have historically been limited in the diversity of data they capture. This limitation is due to an initial focus on automating paper-driven financial processes. While these traditional systems may play an important role in meeting the financial information needs of an organization, they do not necessarily support the marketing, human resources, and manufacturing aspects of an organization very well. The nonfinancial aspects of business events are of great importance to these varied users. Event-driven systems facilitate use by multiple information users with very different needs for information about the events that have occurred within business processes.

Storing data at the event level makes it much easier to retain data related to other nonfinancial and nonquantitative aspects of an event. Ideally, in an event-driven system, the data captured during business processes will be sufficient for someone who was not a party to the event to reconstruct every important aspect of what happened—whether he or she is in marketing, human resources, financial management, manufacturing, or any other part of the organization. Typically, this mandates that at a minimum data be collected and stored related to the *four Ws*:

- **Who** relates to all individuals and/or organizations that are involved in the event.
- **What** relates to all assets that exchange hands as a result of the event.
- **Where** relates to the locations in which (1) the event takes place, (2) exchanged assets reside before and after the event, and (3) the parties to the event are during the event and for any future correspondence.
When relates to all the time periods involved in completion of the event—including future exchanges of assets (e.g., when will we need to pay a bill?) that result from the event.

Once the event data are collected and recorded, the data can be aggregated and summarized in any manner that a given user chooses. The key is that any aggregations and summaries are temporary and only for the user’s application, but the event data remain available to other users in its original form. For applications such as the generation of financial and inventory reports that are frequently required in the same format, programmed procedures can be developed within computerized systems to generate such reports automatically. Thus, the same needs for financial information fulfilled by traditional transaction processing systems are fulfilled by event-driven systems, but with the latter systems a host of other users’ needs can also be met more efficiently and effectively.

Let us take, for example, a series of events that might take place during the course of capturing a customer’s order, putting through a job order to produce the ordered goods, and delivering the goods to the customer. When setting up our event-based system, we will want to capture multifaceted data to track the progression of the process. To capture the sales order event, we need to record data related to the salesperson and customer (the who), the goods ordered (the what), the delivery location (the where), and the date of sale and promised delivery (the when). This information would then be linked with information already stored that relates to a selected supplier for goods. Based on the combined information, an order would be placed with the supplier. A purchase order becomes a link between the purchaser and the supplier (the who) already in the system, the goods (the what) that have already been entered, the location to which the goods will be delivered (the where), and the delivery date from the supplier to our company (the when).

Notice in our sales example that all of the traditional systems data are readily available. The data required for the Order-to-Sales, Purchase-to-Pay, and Business Reporting processes are all captured and available for processing. But, now if the supplier changes the delivery date, the salesperson can also have immediate access to the change and notify the customer. The salesperson can pull together the necessary data by using links between the changed order information, the sales order, and the customer, and narrow the search down to only the sales that he or she is handling. Very quickly, the salesperson can have the information needed to notify the customer immediately of any delay in shipment.

It is important to note that event-driven systems may appear no different to the average user than more traditional transaction processing systems for collecting business event data. Rather, the underlying data storage and management (that is unobservable to most users) differ, while at the same time new sets of users have access to more relevant information for business decision making. In subsequent sections of this chapter, we will revisit these two approaches to Information Systems and discuss the underlying information technologies that enable their existence.

**File Management Processes**

File management comprises the functions that collect, organize, store, retrieve, and manipulate data maintained in traditional file-oriented data processing environments. We have already noted that business event data processing systems collect, process, and store data. So, admittedly, there is an overlap between these two environments,
for data used by the system must physically reside somewhere! This section concentrates on file management. We see how data are managed—particularly how data are stored and retrieved, knowing that part of file management is undertaken by Information Systems underlying Order-to-Cash, Purchase-to-Pay, and other business processes. Thus, file management supports the generation of reports associated with traditional transaction processing systems.

Managing Data Files

Let’s quickly review the hierarchy of data that may already be familiar to you. A character is a basic unit of data such as a letter, number, or special character. A field is a collection of related characters, such as a customer number or a customer name. A record is a collection of related data fields pertaining to a particular entity (person, place, or thing, such as a customer record) or event (sale, hiring of a new employee, and so on). A file is a collection of related records, such as a customer file or a payroll file. A record layout describes the fields making up a record. These relationships are depicted in part (a) of Figure 3.3.

Chapter 1 introduced you to two types of data, master data (entity-type files) and business event data (event-type files). A business event data processing system may operate on one or more files. Some of these files are used to obtain reference information, such as the warehouse location of an item of merchandise. Other files are used to organize and store the data being collected, such as sales orders or inventory data. Some companies still rely on older, legacy systems that use file structures for data storage.

A database approach is a superior data storage method. In a database approach, tables, not files, are used to organize and store data. For now, we want to discuss data management using well-known terms and concepts associated with files; we’ll get to tables later.

Let’s examine two flowcharts. Figure 3.3, part (b), depicts a typical data maintenance activity—the addition of a new customer record to the customer master data. Figure 3.4 (page 74) depicts a typical business event data processing activity—entering a customer’s order. In this text, we use the more generic term, data store to distinguish the conceptual file from its physical implementation as a file or database table.

What is important about these two figures? First, the data maintenance activity (Figure 3.3, part b) does not involve a business transaction as such; however, the business event data processing activity (Figure 3.4) does.

Second, the existence of the customer record—including the credit limit [see Figure 3.3, part (a)]—provides the basic authorization required to enter the customer order. Without the customer record, the computer would reject the customer order in Figure 3.4. It is important to separate authorizations for data maintenance activities from authorizations for business event data processing activities. This separation provides an important control, a topic explored in greater detail in Chapters 8 and 9.

Limitations of File Processing

In Figure 3.5 (page 75) we compare and contrast the applications-based file approach found in a transaction processing system (discussed in this section) with the database approach to data management (discussed in the following section). Figure 3.6 (page 76) contains the record layouts for the files in Figure 3.5, part (a).

Prior to the development of database concepts, companies tended to view data as a necessary adjunct of the program or process that used the data. As shown in part (a) of Figure 3.5, this view of data, based on the transaction processing approach to file
management, concentrates on the process being performed; therefore, the data play a secondary or supportive role in each application system. Under this approach, each application collects and manages its own data, generally in dedicated, separate, physically distinguishable files for each application. For example, Figures 3.3, part (b) and 3.4 assumed a “transaction-centric” approach to file management. One outgrowth of this approach is the data redundancy that occurs among various files. For example, notice the redundancies (indicated by double-ended arrows) depicted in the record
Data redundancy often leads to inconsistencies among the same data in different files and increases the storage cost associated with multiple versions of the same data. In addition, data residing in separate files are not shareable among applications. Now let’s examine how some of these redundancies might come about.

The data represented in Figure 3.6 have two purposes. The data (1) mirror and monitor the business operations (the *horizontal information flows*) and (2) provide the basis for managerial decisions (the *vertical information flows*). In addition to data derived from the horizontal flows, managers use information unrelated to event data processing. These data would be collected and stored with the business event related data. Let’s look at a few examples to tie this discussion together.

Suppose that the sales application wished to perform sales analysis, such as product sales by territory, by customer, or by salesperson. To do so, the sales application

---

2 The horizontal and vertical information flows are depicted in Figure 1.5 (page 14).
would store data for sales territory and salesperson in the sales order record shown in part (a) of Figure 3.6. But, what if the inventory application wanted to perform similar analyses? To do so, the inventory application would store similar, redundant data about territory and salesperson as depicted in part (b) of Figure 3.6. As implied by Figure 3.5, part (a), the sales data in the inventory data file—including customer territory and salesperson—could be updated by the sales application or updated separately by the inventory application. As a second example, what if the sales application wanted to know very quickly the amount of sales for a customer? Then, the summary data on the customer master data file (year-to-date sales) could be stored as shown in part (c) of Figure 3.6. Alternatively, the information could be obtained as needed by summarizing data on the sales order or in the inventory data.

As a final sales example, let’s assume we would like to know all the products that a particular customer buys (perhaps so we can promote those products the customer is not buying). Given the record layouts depicted in Figure 3.6, we could obtain that information by sorting the inventory or sales order data by customer number. Al-
Alternatively, we could have collected these data in the customer master data store! In either case, the data are difficult and expensive to obtain. And, if the applications were not originally designed to give us these data, this approach to file management makes it quite difficult to add this access after the fact.

We could provide several more examples from inventory, but by now we trust that we have made our point. All of these examples consist of business event data related to the selling of merchandise. The transaction processing approach leaves us with similar problems for standing data. Note in Figure 3.6 the redundancies among the three data files with respect to standing data such as customer number, territory, and salesperson. Could these redundant fields become inconsistent? Again, we would have to say yes. The database approach to data management solves many of these problems. We will return to Figure 3.6 and describe what these data might look like with a database, rather than separate application files.

**Database Management Systems**

A database management system is a set of integrated programs designed to simplify the tasks of creating, accessing, and managing data. Database management systems integrate
Database Management Systems  Chapter 3

a collection of files or data tables that are independent of application programs and are available to satisfy a number of different processing needs. A database management system is really the means by which an organization coordinates the disparate activities of its many functional areas. The database management system, containing data related to all of an organization’s applications, supports normal event data processing needs and enhances the organization’s management activities by providing data useful to managers. While in its strictest sense a database is a collection of files, we will use the term database synonymously with database management system since this has evolved as the normal meaning intended by the vast majority of computer users and developers.

Logical vs. Physical Database Models

The concept underlying the database approach to data management is to decouple the data from the system applications (i.e., to make the data independent of the application or other users). Therefore, as reflected in part (b) of Figure 3.5 (page 75), the data become the focus of attention. Several other aspects of part (b) are noteworthy:

- The database is now shared by multiple system applications that support related business processes, as shown at the left of Figure 3.5, part (b).
- In addition to being used by application programs, the data can also be accessed through two other user interfaces: (1) report generation, as shown in the upper-right portion of part (b), and (2) ad hoc user inquiries, i.e., queries handled through query language software, depicted in the lower-right portion of part (b).³
- A “layer” of software called the database management system (DBMS) is needed to translate a user’s logical view of the data into instructions for retrieving the data from physical storage. Some of the more technical design issues of database management systems are described in Technology Insight 3.1 (page 78).

Figure 3.7 (page 79) depicts how a database might look to us if the data were stored using a relational data structure. The data from our three files are now stored in four relational tables: CUSTOMERS (instead of customer master data), INVENTORY_ITEMS (inventory master data), SALES_ORDERS, and SALES_LINES (i.e., the last two tables store the data from the sales order master data). These tables are logical views of data that are physically stored in a database. The logical database view is how the data appear to the user to be stored. This view represents the structure that the user must interface with in order to extract data from the database. The physical database storage is how the data are actually physically stored on the storage medium used in the database management system. It has little relationship to how the data appear to be stored (e.g., the logical view). The user can access the data in the tables (e.g., the logical view in a relational database) by:

1. Formulating a query, or
2. Preparing a report using a report writer, or
3. Including a request for data within an application program.

These three methods are depicted in the flowchart in Figure 3.5, part (b) (page 75).

³ In many database management systems, report generation and queries may not be distinct functions.
Database Management Systems (DBMS)

A database management system (DBMS) is a set of integrated programs designed to simplify the tasks of creating, accessing, and managing a database. The DBMS performs several functions, such as:

- defining the data.
- defining the relationships among data (e.g., whether the data structure is relational or object-oriented).
- mapping each user’s view of the data (through subschema → schema).

In the language of DBMS, a schema is a complete description of the configuration of record types and data items and the relationships among them. The schema defines the logical structure of the database. The schema, therefore, defines the organizational view of the data.

A subschema is a description of a portion of a schema. The DBMS maps each user’s view of the data from subschemas to the schema. In this way the DBMS provides flexibility in identifying and selecting records. Each of the many database users may want to access records in his or her own way. For example, the accounts receivable manager may want to access customer records by invoice number, whereas a marketing manager may want to access the customer records by geographic location. The figure below portrays the schema-subschema relationship.

<table>
<thead>
<tr>
<th>Customer number</th>
<th>Customer name</th>
<th>Customer address</th>
<th>Credit limit</th>
<th>Sales person</th>
<th>Sales territory</th>
<th>Year-to-date sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) Schema</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer number</th>
<th>Customer name</th>
<th>Credit limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b) Credit department subschema</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer number</th>
<th>Customer name</th>
<th>Sales person</th>
<th>Sales territory</th>
<th>Year-to-date sales</th>
</tr>
</thead>
<tbody>
<tr>
<td>(c) Sales manager subschema</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A chief advantage of a DBMS is that it contains a query language, which is a language much like ordinary language. A query language is used to access a database and to produce inquiry reports. These languages allow a nontechnical user to bypass the programmer and to access the database directly. Deriving data from the database using a query does not replace applications programs, which are still required to perform routine data processing tasks. However, when information is needed quickly, or when a manager wishes to “browse” through the database, combining data in unique ways, the query facility of a DBMS is a vast improvement over the traditional method of requesting that a program be written to generate a report.

A DBMS normally contains a number of security controls to protect the data from access by unauthorized users as well as from accidental or deliberate alteration or destruction. A DBMS also contains routines for ensuring that the data can be simultaneously shared by multiple users.
### CUSTOMERS

<table>
<thead>
<tr>
<th>Cust_Code</th>
<th>Cust_Name</th>
<th>Cust_City</th>
<th>Salesperson</th>
<th>Sales_Territory</th>
<th>Credit_limit</th>
<th>Sales_YTD</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTC</td>
<td>Bikes Et Cetera</td>
<td>Elgin</td>
<td>JOHNSON</td>
<td>CENT</td>
<td>10000.00</td>
<td>9561.55</td>
</tr>
<tr>
<td>IBS</td>
<td>Inter. Bicycle Sales</td>
<td>New York</td>
<td>FONG</td>
<td>EAST</td>
<td>5000.00</td>
<td>4191.18</td>
</tr>
<tr>
<td>ROSEBYKE</td>
<td>Rodebyke Bie. &amp; Mopeds</td>
<td>San Jose</td>
<td>JOHNSON</td>
<td>WEST</td>
<td>2000.00</td>
<td>1142.50</td>
</tr>
<tr>
<td>STANS</td>
<td>Stan's Cyclery</td>
<td>Hawthorne</td>
<td>GARCIA</td>
<td>WEST</td>
<td>10000.00</td>
<td>8330.00</td>
</tr>
<tr>
<td>WHEEL</td>
<td>Wheelway Cycle Center</td>
<td>Campbell</td>
<td>GARCIA</td>
<td>WEST</td>
<td>10000.00</td>
<td>6454.00</td>
</tr>
</tbody>
</table>

### INVENTORY ITEMS

<table>
<thead>
<tr>
<th>Item_Number</th>
<th>Item_Name</th>
<th>Qty_On_Hand</th>
<th>Unit_Cost</th>
<th>Unit_Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000-1</td>
<td>20 in. Bicycle</td>
<td>247</td>
<td>55.00</td>
<td>137.50</td>
</tr>
<tr>
<td>1001-1</td>
<td>26 in. Bicycle</td>
<td>103</td>
<td>60.00</td>
<td>150.00</td>
</tr>
<tr>
<td>1002-1</td>
<td>24 in. Bicycle</td>
<td>484</td>
<td>60.00</td>
<td>150.00</td>
</tr>
<tr>
<td>1003-1</td>
<td>20 in. Bicycle</td>
<td>4</td>
<td>24.37</td>
<td>60.99</td>
</tr>
<tr>
<td>1240-056</td>
<td>Kickstand</td>
<td>72</td>
<td>6.50</td>
<td>16.25</td>
</tr>
<tr>
<td>2010-0050</td>
<td>Formed Handlebar</td>
<td>90</td>
<td>4.47</td>
<td>11.25</td>
</tr>
<tr>
<td>3050-2197</td>
<td>Pedal</td>
<td>23</td>
<td>0.75</td>
<td>1.88</td>
</tr>
<tr>
<td>3961-1010</td>
<td>Tire. 26 in.</td>
<td>42</td>
<td>1.45</td>
<td>3.13</td>
</tr>
<tr>
<td>3961-1041</td>
<td>Tire Tube. 26 in.</td>
<td>19</td>
<td>1.25</td>
<td>3.13</td>
</tr>
<tr>
<td>3965-1050</td>
<td>Spoke Reflector</td>
<td>232</td>
<td>0.29</td>
<td>0.63</td>
</tr>
<tr>
<td>3970-1011</td>
<td>Wheel. 26 in.</td>
<td>211</td>
<td>10.50</td>
<td>25.00</td>
</tr>
</tbody>
</table>

### SALES ORDERS

<table>
<thead>
<tr>
<th>SO_Number</th>
<th>Cust_Code</th>
<th>Cust_Order_Number</th>
<th>SO_Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>WHEEL</td>
<td>453</td>
<td>021205</td>
</tr>
<tr>
<td>1011</td>
<td>ETC</td>
<td>347</td>
<td>021205</td>
</tr>
<tr>
<td>1012</td>
<td>WHEEL</td>
<td>546</td>
<td>021205</td>
</tr>
<tr>
<td>1013</td>
<td>IBS</td>
<td>3422</td>
<td>021205</td>
</tr>
<tr>
<td>1014</td>
<td>ETC</td>
<td>778</td>
<td>021205</td>
</tr>
<tr>
<td>1015</td>
<td>WHEEL</td>
<td>5673</td>
<td>021206</td>
</tr>
<tr>
<td>1016</td>
<td>ETC</td>
<td>3345</td>
<td>021206</td>
</tr>
</tbody>
</table>

### SALES LINES

<table>
<thead>
<tr>
<th>SO_Number</th>
<th>Item_Number</th>
<th>Qty_Ordered</th>
<th>Sales_Price</th>
<th>Qty_Shipped</th>
</tr>
</thead>
<tbody>
<tr>
<td>1010</td>
<td>1000-1</td>
<td>5</td>
<td>137.50</td>
<td>0</td>
</tr>
<tr>
<td>1010</td>
<td>2010-0050</td>
<td>2</td>
<td>11.25</td>
<td>0</td>
</tr>
<tr>
<td>1011</td>
<td>1001-1</td>
<td>10</td>
<td>127.50</td>
<td>8</td>
</tr>
<tr>
<td>1011</td>
<td>1002-1</td>
<td>5</td>
<td>150.00</td>
<td>4</td>
</tr>
<tr>
<td>1012</td>
<td>1003-1</td>
<td>5</td>
<td>60.93</td>
<td>0</td>
</tr>
<tr>
<td>1012</td>
<td>1001-1</td>
<td>10</td>
<td>127.50</td>
<td>5</td>
</tr>
<tr>
<td>1013</td>
<td>1001-1</td>
<td>50</td>
<td>78.30</td>
<td>0</td>
</tr>
<tr>
<td>1014</td>
<td>1003-1</td>
<td>25</td>
<td>37.42</td>
<td>0</td>
</tr>
<tr>
<td>1015</td>
<td>1003-1</td>
<td>25</td>
<td>37.42</td>
<td>0</td>
</tr>
<tr>
<td>1016</td>
<td>1003-1</td>
<td>5</td>
<td>60.93</td>
<td>0</td>
</tr>
<tr>
<td>1016</td>
<td>3965-1050</td>
<td>50</td>
<td>33.00</td>
<td>0</td>
</tr>
<tr>
<td>1016</td>
<td>3961-1041</td>
<td>5</td>
<td>3.13</td>
<td>0</td>
</tr>
<tr>
<td>1016</td>
<td>1000-1</td>
<td>4</td>
<td>137.50</td>
<td>0</td>
</tr>
</tbody>
</table>
Now let’s see how easily data can be obtained from the relational tables in Figure 3.7 using the database query language SQL (structured query language).

1. We can perform a query (the SELECT command) to find the customers assigned to salesperson Garcia.

   ```sql
   SELECT CUST_CODE, CUST_NAME, CUST_CITY
   FROM CUSTOMERS
   WHERE SALESPERSON = 'Garcia'
   ```

   We see that there are two customers, STANS and WHEEL.

2. We can also create a listing using the SELECT command.

   ```sql
   SELECT SO_Number, INVENTORY_ITEMS.Item_Number, Sales_Price, Unit_Price
   FROM SALES_LINES, INVENTORY_ITEMS
   WHERE Sales_Price != Unit_Price AND INVENTORY_ITEMS.Item_Number = SALES_LINES.Item_Number
   ```

   The SELECT command combines the SALES_LINES and INVENTORY_ITEMS tables over the list of items (Item_Number) and finds those items in the combined table that were sold at a price (Sales_Price) other than the price contained on the INVENTORY_ITEMS table (Unit_Price). We see that there are six instances.

   Neither the user nor the application programs have any idea that these mappings are taking place. In preparing the query, a user might formulate the SELECT command (i.e., choose the selection criteria). Alternatively, rather than being visible to the users, the SQL commands might be part of an underlying application or database structure. Finally, the user might execute these commands using drop-down menus and a mouse. In these latter two alternatives, the commands would be embedded within the application programs. Relational data structures are discussed in greater detail later in this chapter.

### Overcoming the Limitations of File Processing

We discussed earlier some of the limitations of applications that rely on traditional file management. What are the advantages of the database approach?

- **Eliminating data redundancy.** With the database approach to data management, data need only be stored once. Applications that need data can access the data from the central database. For example, in Figure 3.5 (page 75), part (a), there are multiple versions of the inventory master data, while in part (b) of that
figure there is but one. Further, Figure 3.6 (page 76) depicts the same data elements on more than one file, whereas Figure 3.7 (page 79) shows each data element only once. An organization using the applications-based file approach to data management must incur the costs and risks of storing and maintaining these duplicate files and data elements.

- **Ease of maintenance.** Because each data element is stored only once, any additions, deletions, or changes to the database are accomplished easily. Contrast this to the illustration in Figure 3.6, where a change in a salesperson, territory, or customer combination, for instance, would require a change in three different files.

- **Reduced storage costs.** By eliminating redundant data, storage space is reduced, resulting in associated cost savings. However, in most database installations, this savings is *more than offset* by the additional costs of DBMS software.

- **Data integrity.** This advantage, like several others, results from eliminating data redundancy. As mentioned earlier, storing multiple versions of the same data element is bound to produce inconsistencies among the versions. For instance, the salesperson and sales territory data might differ among their many versions, not only because of clerical errors but because of timing differences in making data maintenance changes. We could make similar comments about inconsistent data resulting from the timing differences that might occur during *event data processing* of the inventory master data by the sales and inventory applications. With only one version of each data element stored in the database, such inconsistencies disappear.

- **Data independence.** As illustrated in part (b) of Figure 3.5, the database approach allows multiple application programs to use the data concurrently. In addition, the data can be accessed in several different ways (e.g., through applications processing, online query, and report writing programs). And, the access can be quickly changed by modifying the definition of the tables or views. With the traditional *applications-based file approach*, the programs would have to be revised to provide access to more or less data.

- **Privacy.** The security modules available through DBMS software can contain powerful features to protect the database against unauthorized disclosure, alteration, or destruction. Control over data access can typically be exercised down to the data element level. Users can be granted access to data for reading or updating (add, revise, delete) data. Other ways to implement security include *data classification* (i.e., data objects are given classification levels and users are assigned clearance levels) and *data encryption* (discussed in Chapter 9).

These advantages add greatly to the incentive for firms relying on legacy systems to move to database-supported applications.

**Enabling Event-Driven Systems**

Earlier we noted that file management approaches are often sufficient to support traditional transaction processing. Without question, database management systems can improve the efficiency of processing data by eliminating data redundancies, improving data integrity, and so forth. However, the big change that database management systems have enabled is the realization of event-driven data processing systems. As noted earlier, event-driven systems are oriented toward the concept that complete data de-
scribing business events should be kept in its original form, where multiple users from throughout the organization can view and aggregate event data according to their needs.

At the heart of this movement toward event-driven systems is a fundamental shift in the view of information processing in business organizations. Traditionally, organizational Information Systems have been focused first on capturing data for the purpose of generating reports, and using the reporting function to support decision making. Increasingly, management is shifting to viewing the primary purpose of organizational Information Systems as decision support while reporting is secondary. This perspective leads to a focus on aggregating and maintaining data in an original form from which reports can be derived, but users can also access and manipulate data using their own models and their own data aggregations. In Chapter 5, we will discuss Information Systems such as business intelligence and expert systems that are designed to improve decision making. If you look ahead to the figures in Technology Insights 5.2 (page 149) and 5.3 (page 152) you will notice that both types of support systems generally require access to detailed data stored in databases.

The strategic shift toward event-driven systems is further embodied in two contemporary concepts that are driving new database management systems implementations: data warehousing and data mining. Data warehousing is the use of Information Systems facilities to focus on the collection, organization, integration, and long-term storage of entity-wide data. Data warehousing’s purpose is to provide users with easy access to large quantities of varied data from across the organization for the sole purpose of improving decision-making capabilities. Data mining is the complementary action to data warehousing. Data mining refers to the exploration, aggregation, and analysis of large quantities of varied data from across the organization to better understand an organization’s business processes, trends within these processes, and potential opportunities to improve the effectiveness and/or efficiency of the organization. The “warehouses of data” analogy makes sense as the software to support data storage is akin to physical warehousing approaches used to store and retrieve inventory—when an item needs to be restocked on the store shelf, there must be some system whereby the item can be located in the warehouse and retrieved.

Data warehousing and data mining opportunities are enabled and enriched through the use of event-driven systems focused on capturing data that provide comprehensive views of business events. However, neither effective event-driven systems nor data warehouses are possible without effective implementation of database management systems. Both objectives are dependent on the massive data integration and data independence made possible through database technology. Both warehousing and data mining may also be limited if well-designed database models that provide for future information needs are not effectively implemented. This process starts with the information requirements analysis and successful attainment of an understanding of all users’ potential data and information needs.

**Review Question**

What do the concepts of data warehousing and data mining mean?

**Entity-Relationship (E-R) Modeling**

Chapter 2 described E-R diagrams and showed you how to read them. Before moving on to developing E-R diagrams (in Appendix A), you should expand upon your knowledge of E-R diagrams to ensure a solid understanding of entities and attributes. This knowledge aids in the development of solid data models that lead to effective database structures. Although the diagrams can appear complex at first, they provide
a very useful high-level tool for understanding the complicated relationships that exist among data in typical firms. E-R diagrams permit users and/or developers to communicate with a common understanding of how different types of data relate, including data about entities and business events. This understanding also permits a decision maker to interpret reports and analyses correctly based on extracts from a database.

Entities and Attributes

An entity is an object, event, or agent about which data are collected. As examples, objects could include such things as orders, sales, and purchases. Agents include people such as customers, employees, and vendors. Basically, an entity is anything that independently exists.

In order to understand which entity we are capturing in our database and, likewise, to be able to identify that unique entity when we retrieve the data, we need to describe the entity in detail. In a data model, we describe entities by recording the essential characteristics of that entity that fully describe it. In other words, we record its attributes. An attribute is an item of data that characterizes an entity or relationship. Figure 3.8 displays an attribute hierarchy for an entity CLIENT. Notice that to describe fully a CLIENT we need to record several attributes such as Name, Address, Contact_Person, and Phone_Number. Sometimes, attributes are a combination of parts that have unique meanings of their own. For instance, in Figure 3.8, Address might consist of several independent subattributes such as the Street_Address, City, State, and Zip_Code. Attributes that consist of multiple sub-attributes are referred to as composite attributes.

Note that an inherent assumption we have made in specifying the attributes for the entity, CLIENT, is that there is a common set of attributes for each entity of interest in our database. That is, for every client we need to know the client’s name, address, contact person, and phone number. To design an effective data model, you must learn to identify the complete set of entities and the common attributes that fully describe each entity. It is very important that the attributes are also sufficient to allow the user of a database to identify uniquely each entity in the database.

Review Question

What is an entity?

Review Question

What is an attribute?

Figure 3.8 Attribute Hierarchy for the Entity CLIENT

4 Technically, CLIENT would be an “entity type” in that it describes a collective group of entities (e.g., different clients). However, most database developers use the term entity rather than “entity type,” as it is understood that all entities of interest will fall into some category of similar-type entities. We will use this common terminology throughout the remainder of our discussion.
To achieve the objective of uniquely identifying each entity to be stored in our database, it is necessary that one or more attributes be identified that will always allow the user to access the entity that he or she is seeking. A key attribute is the attribute whose value is unique (i.e., different) for every entity that will ever appear in the database and is the most meaningful way of identifying each entity. This key attribute becomes the primary key. For our CLIENT entity, we might be able to use the Name for the key attribute; but alphabetic-based attributes like names are tricky because computers are sometimes sensitive to the use (or non-use) of capital letters. Further, spellings and full names can be tricky in that one user might view the company name as “Arnold Consultants” while another user might use the full name, “Arnold Consultants, LLP.” If possible, it is preferable to use a numeric-valued or a non-naming alphabetic attribute. For instance, we could use a client number that would typically be assigned to each CLIENT. A numeric form using a sequential coding scheme might assign a number such as “12345.” A non-naming alphabetic form using block coding to categorize companies by the first letter of a company’s name might assign an alphanumeric such as “A1234” for the client number.

Figure 3.9 part (a), displays the symbols that are used to represent entities and attributes, as seen earlier in the E-R diagrams in Chapter 2. In Figure 3.9, part (b), the rectangle is accordingly used to represent the CLIENT entity. In order to map the attributes of an entity, we add oval connectors [as shown in part (a)] for each attribute. Notice in part (b) that we have added an oval for each of the attributes shown.

**Figure 3.9** Developing Model Representations for Entities and Attributes

(a) Entity and attribute symbols

(b) Entity and attribute model for CLIENT with Client_Number added
in Figure 3.8. For the composite attribute Address, we use the same oval connectors for each of the subattributes of the main attribute. Note that we have added an attribute beyond those shown in Figure 3.8—Client_Number. We have added this attribute as a key attribute, and have used an underline on the attribute name to document its selection as the key attribute.

**Relationships**

Relationships are associations between entities. As we have discussed in the previous section, a database consists of several (or many) different types of entities. However, in order to make the data stored in these entities effective for users to reconstruct descriptions of various business events, the various entities must be logically linked to represent the relationships that exist during such business events. The ease with which a user can ultimately extract related data from a database is heavily dependent on the quality of the database’s logical design—that is, effective identification of the relationships between different entities. These relationships map and define the way in which data can be extracted from the database in the future. The mapping of the relationships between entities (i.e., development of the E-R diagram) provides a roadmap for getting from one piece of data in the database to another related piece of data.

A three-step strategy is generally most effective in identifying all of the relationships that should be included in a model. First, consider the existing and desired information requirements of users to determine if relationships can be established within the data model to fulfill these requirements. Second, evaluate each of the entities in pairs to determine if any entity provides an improvement in the describing of an attribute contained in the other entity. Third, evaluate each entity to determine if there would be any need for two occurrences of the same entity type to be linked—e.g., identify recursive relationships. Appendix A describes the development of an E-R model in greater detail.

A major thrust in many organizations has been a move toward completely integrating all data across an organization. These completely integrated enterprise models are the foundations for implementing enterprise systems, which are discussed in Technology Insight 3.2. Integration allows many users to share entity-level data by linking business events within related business processes.

**TECHNOLOGY INSIGHT 3.2**

**Enterprise Systems**

*Enterprise systems* are integrated software packages designed to provide complete integration of an organization’s business information processing systems and all related data. These systems are based on event-driven systems concepts, which include the capturing of business data for supporting decision making, as well as integration of the underlying data to facilitate access and ad hoc analysis.

A number of enterprise systems are commercially available. The dominant player is System Application Products (SAP) R/3, which commands the largest percentage of the Fortune 500 market. Several other products are available and have established large customer bases—often through establishing excellence in certain market niches. These other vendors include JD Edwards,
Part II  Technology for Business Processes and Information Systems

PeopleSoft, and Oracle. While these products are designed to offer integration of everything from accounting and human resources to manufacturing and sales staff logistics, products designed to focus on specific industries are also appearing in the marketplace. These systems are capable of extracting data from both enterprise systems’ data sources and legacy systems that may still exist within an organization (or subsidiary of the organization). They can also support a Web interface to allow business partners to initiate business events directly.

Originally, the implementation of enterprise systems was predominantly targeted at large multinational manufacturers such as General Motors, IBM, and General Mills. This strategy aimed where benefits would be expected to be the greatest, in that large multilocation and multidivision companies often present the greatest challenges to managers who mine data from corporate databases to improve overall organizational decision making. Enterprise systems allow companies to standardize systems across multiple locations and multiple divisions in order to link data in a consistent fashion and provide organization-wide accessibility.

Large enterprises were the predominant implementers of enterprise systems, but largely due to the costs of implementation. These systems typically took a year or more to implement at a cost of up to hundreds of millions of dollars, necessitating a similarly significant return in benefits. As advances in technology underlying these systems has evolved, small and medium sized enterprises (SMEs) have driven the new implementation base. This shift has happened primarily due to two drivers: (1) the move towards web-browser driven systems that reduce the expense of both the technology and training; and (2) the emergence of application service providers (ASPs) that implement enterprise systems and then lease out use of the enterprise system to several other companies. In other words, the ASP runs the hardware and software for the company that wants its data integrated via an enterprise system, and the company saves money by essentially sharing the costs of the enterprise system implementation and maintenance with several other companies that also use the same ASP. ASPs are discussed in greater detail in Chapter 7 of the text.

Relational Databases

Review Question
What are the relative advantages of the relational and object-oriented database models?
Encapsulation is the biggest difference in object-oriented database models. Encapsulation refers to the ability to build into the database model, as part of an object’s definition, programmed procedures that change that object’s value (i.e., any of the attribute values). At the same time, no other object can change the value of a given object, as this is controlled within each object. On the other hand, part of the encapsulation may be the querying of information from another object in order to have sufficient data by which to perform encapsulated operations.

Users usually don’t see much difference. Other than encapsulation, other characteristics of object-oriented models could be integrated into relational data models and are increasingly being integrated in commercial packages. The most successful integration tends to occur within relational models where attributes can be redefined to handle more complex object data.
database vendors provide modified versions of their software that support objects within the relational structure. We anticipate that relational-based systems will remain dominant for the foreseeable future.

**Basic Relational Concepts**

Relational databases are often perceived to be a collection of tables. This is a reasonable perception in that the logical view of the data is a tabular type format referred to as a *relation*. A *relation* is defined as a collection of data representing multiple occurrences of an object, event, or agent. Similar to an *entity*, objects include such things as inventory, equipment, and cash. Events may include orders, sales, and purchases. Agents could include customers, employees, and vendors.

Figure 3.10 displays an example relation along with labels for each of its components. Consistent with a tabular representation, a *relation* consists of rows and columns. Rows are referred to as *tuples* and columns are referred to as *attributes*. Tuples are sets of data that describe an instance of the entity represented by a relation (e.g., one employee in the EMPLOYEE relation). We may think of a tuple as being akin to a record in a traditional file structure. While technically they are different, logically they are similar. Attributes, as they do in an E-R diagram, represent an item of data that characterizes an object, event, or agent. In terms of traditional file structures, we would parallel attributes (i.e., the columns in a *relation*) to fields.

In viewing the relation in Figure 3.10, note that the data contained in the table do not appear to be in any particular order. In a relational database model there is no ordering of tuples contained within a relation. This is different from the traditional file structures you studied earlier in this chapter, where sequence or keyed location was usually critical. Rather, ordering of the tuples is unimportant since the tuples are recalled by the database through matching an attribute’s value with some prescribed value, or through a query by which ordering could be established if desired (e.g., by sorting on one of the attributes—such as by Pay_Rate or Billing_Rate).

In order to uniquely identify a tuple, it is critical that each be distinct. This means that each tuple in a relation can be uniquely identified by a single attribute or some combination of multiple attributes. Similar to the rules used for constructing an E-R diagram, a primary key (which is equivalent to a key attribute in an E-R diagram) is specified to uniquely identify each tuple in the relation. Notice in Figure 3.10 that Employee_Number is the *primary key* (the attribute name is underlined) and
that it is unique for every tuple. There may be other attributes in the relation that also have the ability to serve as a key attribute, and in a relation these additional attributes can form secondary keys referred to as candidate keys. For any attribute specified as a key attribute, that attribute must have a unique and “non-null” value (i.e., there has to be some value assigned to the attribute for each tuple). Notice that Soc_Sec_No would also be unique and could possibly be used as a candidate key, but constraints would have to be put in place to ensure every tuple has a value since it is possible that an employee could, at least temporarily, not have a social security number.

Additionally, constraints should be put in place to assure that the referential integrity of the database is maintained. Referential integrity requires that for every attribute value in one relation that has been specified in order to allow reference to another relation, the tuple being referenced must remain intact. In other words, as you look at the relation, EMPLOYEE, in Figure 3.10, notice that EMPLOYEE is party to a recursive relation [also modeled in Appendix A in Figure 3.11, part (b), page 90]. In this recursive relation, Supervisor_No is used to reference the Employee_Number of the supervising employee. If the tuple for Greg Kinman were deleted from the database, note that four other employees would no longer have a valid Supervisor_No (e.g., the Supervisor_No would be referencing a tuple [A632] that no longer exists). Hence, a referential integrity constraint would require the user to reassign the four employees to a new supervisor before the tuple for Greg Kinman could be deleted.

**Conclusions**

As information needs and wants of users escalate, integrated databases have become the norm rather than the exception. The focus is no longer on the question, “Where can we implement databases?” But rather the focus has shifted to, “How do we integrate as much of our data as possible into a single logical database?” To retain flexibility, some organizations pursue self-developed and self-designed integrated database systems. For many organizations, the packaged solution of an enterprise system is the desired approach for data integration. Either way, the expertise of information specialists is key to successfully overcoming the challenges of implementing such integrated database systems.

With these opportunities and challenges also come huge responsibilities. The very lifeblood of an organization becomes wrapped up in a database that contains all of the organization’s information. If the database is destroyed and cannot be recovered, the organization will probably not survive in today’s business environment. Likewise, if competitors or others gain access to the data, the organization’s ability to compete can also be seriously jeopardized.

Safeguarding data, while at the same time getting the information to users who need the information, is not a simple task. In Chapters 8 and 9, our discussion will shift to the issues surrounding data reliability, access, and security. You will learn about procedures that organizations implement to assure the reliability of information that is updated or added to the database. You will also learn about safeguarding data and maintaining backups of data so that if something should happen to the database, it can be recovered in a timely manner. These are truly challenging but exciting times for managers and other professionals who are prepared to operate in an information systems environment.
E-R Model Development

As we mentioned earlier, there is a three-step strategy to identify the relationships that should be included in a data model. First, it is very important that you study business events, and understand users’ information requirements, in order to identify all of the ways in which different entities are related. This information will provide the foundation level of relationships required in the database model. The remaining two steps (i.e., evaluating each of the entities in pairs to determine if any entity provides an improvement in describing an attribute contained in the other entity, and evaluating each entity to determine if there would be any need for two occurrences of the same entity to be linked) enable you to refine and improve this foundation-level model.

The focus for our E-R model development will be on the client billing process generally used by service firms such as architecture, consulting, and legal firms. The nature of the process is that each employee in the firm keeps track of time spent working on each client’s service, generally filling out a time sheet each week. The hours spent on a client are then multiplied by that employee’s billable rate for each hour worked. The cumulative fees for all employees’ work are used to generate the bill for the client. This way, the client only pays for the services it actually receives. The challenge here is capturing all of the information necessary to track employees’ work hours and client billing information.

Examine Figure 3.11, part (a). Desirable linkages between entities will often be fairly easy to recognize when the relationship appears to define an attribute more clearly. If our billing system requires that we know for which client an employee has worked, the entity representing work completed needs to include a client number. This client number would link the WORK_COMPLETED entity to the CLIENT en-

---

**Figure 3.11** Modeling Relationship Types

(a) Relationship for expanding attribute description

(b) Recursive relationship
entity that provides us with a full description of the attribute denoted by client number in WORK_COMPLETED. Obviously, as shown in Figure 3.11, part (a), CLIENT is a separate entity and not an attribute. At the same time, CLIENT does improve the description of an attribute for the work completed—the client for whom the work was performed. This descriptive value makes it apparent there should be a relationship between the CLIENT entity and the entity capturing the completed work. Hence, we can often identify the need for defining relationships (such as Works_For) by also looking at the prescribed entities as pairs (in this case, we jointly examined the pair CLIENT and WORK_COMPLETED) to identify logical linkages that would improve the description of an entity’s attributes.

Let’s look at another type of relationship that is displayed in Figure 3.11, part (b). The relationship Supervises is referred to as a recursive relationship. A recursive relationship is a relationship between two different entities of the same entity type. For instance, there usually are relationships between two employees, such as one employee who supervises another employee. This relationship may be important in some decision-making contexts and, therefore, should be represented in our database. We represent this relationship using the technique demonstrated in Figure 3.11, part (b). Consider the alternative: If we try to represent supervisors and their supervised employees as separate entities in our model, we end up with data redundancies when the supervisor is in fact supervised by a third employee. It is easier simply to create a recursive relationship to the entity, EMPLOYEE, whereby a link is created between one employee who is being supervised and another employee who is the supervisor. As shown in our sample diagram, the diamond is still used to represent the recursive relationship, Supervises, just as would be used to show any relationship (e.g., Works_For in part a).

Model Constraints

In this section we explore the various types of relationships that can occur and discuss the constraints that are used to specify such relationships. In Chapter 2, we briefly explored three different relationship types: 1:N (one-to-many), M:N (many-to-many), and 1:1 (one-to-one). The connotations of these three relationships are what we refer to as cardinality. The cardinality constraint of a relationship relates to the specification of how many occurrences of an entity can participate in the given relationship with any one occurrence of the other entity in the relationship.

In Figure 3.12 (page 92), part (a), we demonstrate the specification of cardinality constraints for the one-to-many relationship Works. Note that the specification is done by placing the “1” above the left line of the relationship, specifying one employee performs an employee work day; and the “N” above the right line, specifying that many client work days may be performed by an employee. To determine the cardinality of a relationship, you have to ask yourself the question, “How many items (records) in this entity could be related to any one item (record) in the other entity—one or multiple?” The answer determines that half of the cardinality ratio, and then the same question is asked in the reverse direction of the relationship in order to determine the other half of the cardinality ratio. In our example, we take the relationship in Figure 3.12, part (a) and ask the question, “How many work days can an employee have on a client service engagement?” The answer is many (based on the attributes specified for WORK_COMPLETED in Figure 3.12, part (a), which indicates that a given occurrence in the WORK_COMPLETED entity relates to one employee’s time spent on a given client in a single day—based on time being captured.
by the Date attribute). The question is then reversed and we ask, “How many employees can provide a specific employee work day?” The maximum number will be one. Hence, the cardinality of the relationship is specified as one-to-many and noted on the diagram with the “1” and “N.” For the database to maintain this relationship, a constraint must be enforced to ensure that data are never entered indicating that more than one employee is responsible for a given client work day.

Cardinality is the most common constraint specified in E-R diagrams. The other meaningful constraint that may be specified is participation. The participation constraint is used to specify both the minimum and maximum participation of one entity in the relationship with the other entity. In Figure 3.12, part (b), the participation constraints are reflected in the partial E-R diagram. In our Works relationship we just discussed, not every employee will have worked on specific client service projects, but rather may have non-client service responsibilities, such as training, that he or she spends time on. The “many” in the cardinality ratio only specifies the maximum participation in the relationship, not the minimum. In specifying the participation in the relationship, the maximum is still many, but the minimum may be zero. The line on the right reflects the range of zero to many occurrences of work being completed on client projects with the notation (0,N), where the numbers reflect (minimum, maximum). On the other hand, for any given occurrence of a client workday, the maximum of one employee providing the specific service still holds. At the same time, the minimum is also one as there must be an employee who performs a particular occurrence of the completed work. Note the required participation of one, and only one, employee is shown on the left line of the relationship as (1,1).

While the participation constraint may provide more information, it is still used much less frequently than the cardinality constraint. As such, we will tend to present the diagrams in this text using cardinality constraints. It is important, however, that you are familiar with both types of constraints and the notation applied, since as a member of a team developing or using an E-R diagram, you need to be able to communicate using the methods selected by a given organization.

**Entity Relationship (E-R) Diagrams**

We have now worked our way through all the pieces necessary to develop effective E-R diagrams. If you have successfully gotten a handle on each of the concepts explored so far in this chapter, you should be ready to start developing an integrated
database model. Each of the data model segments that has been displayed in Figures 3.8 through 3.12 represents part of the evolution toward our diagram.

At this point, it may be worth recalling our discussion earlier in this chapter on event-driven systems. One of the fundamental requirements for moving toward an event-driven model was the complete integration of data related to an organization’s various business events. We will use our data modeling techniques to demonstrate the integration of just two business processes: client billing and human resources.

The objective in the development of an E-R diagram is to integrate the data in a manner that allows business processes access to the data necessary for effective performance. Figure 3.13 presents the integrated data model for two business processes (i.e., the billing and human resources functions).

In a service organization such as a consultancy firm, billing of clients is heavily dependent on tracking the actual person-hours put into providing service to a client. To effectively execute the client billing process, our database needs to capture data related to all employees who contributed time to client service and, at the same time, be able to tie these employees’ efforts to a specific client. To further complicate mat-
ters, each employee may have a different billing rate for his or her time. To meet the needs of the billing process, we must be able to aggregate specific employees’ time put into providing service to a client, each employee’s billing rate, and sufficient information about the client to deliver the billing statement. Three entities are involved in the billing process: (1) EMPLOYEE, (2) CLIENT, and (3) WORK_COMPLETED. Note in Figure 3.13 that the three entities for the billing process are linked together on the right half of the diagram. The linkages allow us to pull together information related to the employees’ hours worked on a specific client, their billing rates, and the contact address for sending the billing statement.

Service organizations are also interested in tracking employee work activities through the human resources process. The human resources process includes (among other activities) both payroll activities and employee education and development. To complete the payroll process, information is needed regarding work hours completed, pay rate, vacation time, sick days, and training time. Payroll activity information can be drawn from four entities (i.e., RELEASE_TIME, TRAINING_COMPLETED, EMPLOYEE, and WORK_COMPLETED) in order to aggregate the information necessary to determine the employee’s pay rate, hours worked, hours in training, and hours used of allocated sick and vacation time.

Regarding employee education and development, the human resources department monitors training activities to assure the employee is receiving enough continuing education. At the same time, human resources also monitors the percentage of billable hours the employee has accumulated as a measure of job performance. To handle all of these activities, human resources needs to be able to link data related to completed work activities and training programs. This information can be drawn from three entities (i.e., EMPLOYEE, TRAINING_COMPLETED, and WORK_COMPLETED) to determine a given employee’s training coverage and percentage of billable hours.

Again, it is important to recognize that Figure 3.13 demonstrates only a small part of the overall enterprise model that would be required to integrate all information across an organization. The E-R diagram does effectively integrate the data required for the prescribed business processes. As other business processes are selected and integrated, the model will continue to expand through an explosion of entities and relationships.
Mapping an E-R Diagram to a Relational DBMS

In this chapter, we have discussed the development of E-R diagrams and the foundations for implementing well-constrained relational database models. It is now time to put these two concepts together. This process is referred to as mapping an E-R diagram into a logical database model—in this case a relational data model.

We introduce here a five-step process for specifying relations based on an E-R diagram. Based on the constraints we have discussed in this chapter, we will use this five-step process to develop a well-constrained relational database implementation. Follow along as we map the E-R diagram in Figure 3.13 (page 93) to the relational database schema in Figure 3.14.

1. **Create a separate relational table for each entity.**

   This a logical starting point when mapping an E-R diagram into a relational database model. It is generally useful first to specify the database schema before proceeding to expansion of the relations to account for specific tuples. Notice that each of the entities in Figure 3.13 has become a relation in Figure 3.14. To complete the schema, however, steps 2 and 3 must also be completed.

2. **Determine the primary key for each of the relations. The primary key must uniquely identify any row within the table.**

3. **Determine the attributes for each of the entities.**

---

**Figure 3.14** Schema for the Billing and Human Resources Portion of the Database

<table>
<thead>
<tr>
<th>CLIENT</th>
<th>WORK_COMPLETED</th>
<th>EMPLOYEE</th>
<th>TRAINING_COMPLETED</th>
<th>RELEASE_TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client_No</td>
<td>Employee_No</td>
<td>Employee_Number</td>
<td>Employee_No</td>
<td>Employee_No</td>
</tr>
<tr>
<td>Name</td>
<td>Date</td>
<td>Soc_Sec_No</td>
<td>Date</td>
<td>Date</td>
</tr>
<tr>
<td>Street_Address</td>
<td>Date</td>
<td>Name</td>
<td>Hours</td>
<td>Hours</td>
</tr>
<tr>
<td>City</td>
<td></td>
<td>Supervisor_No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>State</td>
<td></td>
<td>Billing_Rate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zip_Code</td>
<td></td>
<td>Pay_Rate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contact</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phone_Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Client_No</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hours</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hours</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vacation_Sick</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Note in Figure 3.13 that a complete E-R diagram includes specification of all attributes, including the key attribute. This eliminates the need to expend energy on this function during development of the relations. Rather, the focus is on step 2 and now becomes simply a manner of determining how to implement the prescribed key attribute within a relation. With a single attribute specified as the key, this is a very straightforward matching between the key attribute specified in the E-R diagram and the corresponding attribute in the relation (e.g., Employee_Number in the EMPLOYEE entity of Figure 3.13 and the EMPLOYEE relation in Figure 3.14). For a composite key, this is a little trickier—but not much. For a composite key, we can simply break it down into its component subattributes. For instance, in the implementation of the WORK_COMPLETED relation, Employee_No, Date, and Client_No would be three distinct attributes in the relation, but would also be defined as the key via a combination of the three. The completed schema is presented in Figure 3.14. Note the direct mapping between the entities and attributes in the E-R diagram and the relations and attributes respectively in the relational schema.

4. Implement the relationships among the entities. This is accomplished by ensuring that the primary key in one table also exists as an attribute in every table (entity) for which there is a relationship specified in the entity-relationship diagram.

With the availability of the full E-R diagram, the mapping of the relationships in the diagram with the relationships embedded in the relational schema is again fairly straightforward. References to the key attributes in one entity are captured through the inclusion of a corresponding attribute in the other entity participating in the relationship. However, the dominance of 1:N relationships in our model simplifies this process. Let’s take a quick look at how the different categories of relationships (i.e., cardinality constraints) affect the mapping to a relational schema.

- One-to-many (1:N or N:1) relationships are implemented by including the primary key of the “one” relationship as an attribute in the “many” relationship. This is the situation we have for all of the relationships in Figure 3.13. The linking between these relations in the schema are drawn in Figure 3.15. Note that Client_No in CLIENT and Employee_Number in EMPLOYEE provide the links to WORK_COMPLETED. Similarly, Employee_Number in EMPLOYEE provides links to TRAINING_COMPLETED and RELEASE_TIME. The recursive relationship with EMPLOYEE is linked using Supervisor_No to identify the correct EMPLOYEE as the supervisor.

- One-to-one (1:1) relationships are treated as a one-to-many relationship. But, to implement the one-to-one relationship, we must decide which of the entities is to be the “many” and which is to be the “one.” To do this we might predict which of the “ones” might become a “many” in the future and make that the “many.” If we can’t decide, then either will do. For example, if at present one employee workday was sufficient to complete any client project, then a 1:1 relationship would exist between WORK_COMPLETED and CLIENT. Even in this situation we would still select the Client_No in CLIENT to establish the primary key (see Figure 3.15) in anticipation that in the future a client engagement might require more than one day to complete and more than one employee to complete (i.e., the formation of the many dimension shown in the relationship of Figure 3.13, page 93).

- Many-to-many (M:N) relationships are implemented by creating a new relation whose primary key is a composite of the primary keys of the relations to be linked. In our model we do not have any M:N relationships, but if we had
not needed to record the Date and Hours in the WORK_COMPLETED entity, that entity would not have existed. Still, we would then need a relationship between the EMPLOYEE and CLIENT entities which would then be a M:N relationship. This creates problems because our relations cannot store multiple client numbers in a single EMPLOYEE tuple for all clients in which an employee provides services. Similarly, a single CLIENT tuple cannot store multiple employee numbers to record all employees working on an engagement. In that situation, we would have needed to develop a relation to link the EMPLOYEE and CLIENT relations (see Figure 3.16). This new relation would have a composite key consisting of Employee_Number from EMPLOYEE and Client_No from CLIENT—essentially the same as what we currently have with the composite key in the existing relation, WORK_COMPLETED (see Figure 3.15). Note that
we wouldn’t combine the columns, but rather just as we have done in the WORK_COMPLETED, TRAINING_COMPLETED, and RELEASE_TIME relations, the individual attributes making up the composite key remain independent in the corresponding relation.
Beyond concerns over meeting the constraint requirements for primary keys, we must also assure adherence to the referential integrity constraints. We identify the referential integrity constraints by locating the corresponding attribute in each relation that is linked via a relationship. We then determine which of the relations contain the tuple that if the reference attribute were deleted or changed would jeopardize the integrity of the database. In Figure 3.15 the referential integrity constraints are represented by arrows, with the destination of the arrow being the attribute requiring control for referential integrity. In other words, the attribute that is pointed to, if changed or deleted, could cause an attribute to have a nonmatching value at the source of the arrow. To ensure referential integrity, constraints should be put in place to assure Employee_Number is not altered or deleted for any EMPLOYEE until the referencing attribute values for the Employee_No attributes in WORK_COMPLETED, TRAINING_COMPLETED, and RELEASE_TIME have first been corrected. Likewise, a similar constraint should be placed on Client_No in CLIENT until Client_No has been corrected in WORK_COMPLETED.

5. Determine the attributes, if any, for each of the relationship tables.

Again, in the extended version of the E-R diagram, the attributes map directly over to the relations. The implementation of the schema is shown in Figure 3.17.

**REVIEW QUESTIONS**

**RQ3-1** What business events are typically encountered by a merchandising firm?

**RQ3-2** How do the business events for a service firm differ from those of a merchandising firm? How are they similar for the two firms?

**RQ3-3** How are event-driven systems different from traditional transaction processing systems?

**RQ3-4** What is meant by the idea of “storing data at the event level”?

**RQ3-5** Why is it important to capture the who, what, where, and when in describing business events?

**RQ3-6**

a. What is file management?

b. How are the applications-based file and database approaches to data management the same? How are they different?

c. What are the relative advantages of the database approach?

**RQ3-7** What do the concepts of data warehousing and data mining mean?

**RQ3-8**

a. What is an entity?

b. What is an attribute?

c. What is a relationship?

d. What is a key attribute?

**RQ3-9** Why is it preferable to use a numeric-based attribute as the key attribute?
Part II  Technology for Business Processes and Information Systems

RQ3-10  Why is it important that you identify all of the important relationships when developing an entity-relationship (E-R) diagram?

RQ3-11  What are the relative advantages of the relational and object-oriented database models (Technical Insight 3.3)?

RQ3-12  a. What is a relation?
        b. What is a tuple?
        c. What is an attribute in a relational data model?
        d. What is referential integrity?

RQ3-13  Compare and contrast the data structures for data stored in a file and data stored in a relational table.

RQ3-14  (Appendix A) What is different about a recursive relationship in comparison to other relationships in a data model?

RQ3-15  (Appendix A) What is different about the information provided through cardinality constraints versus participation constraints?

RQ3-16  What is the importance of an E-R diagram in facilitating event-driven systems and the integration of data between business processes?

RQ3-17  (Appendix B) How is a composite key implemented in a relational database model?

RQ3-18  (Appendix B) What is the difference in implementation of a one-to-many and a one-to-one relationship in a relational database model?

DISCUSSION QUESTIONS

DQ3-1  Using the descriptions provided in Figure 3.4 (page 74), identify the key event data you would want to capture based on the 4 W’s (who, what, where, and when).

DQ3-2  “The database approach to data management is the only approach that makes sense for most organizations in today’s economic and technical environment.” Do you agree? Discuss fully.

DQ3-3  “Our company’s database contains some very sensitive information about our customers. Shouldn’t we keep that in a separate file so that those who shouldn’t have access to it (like the payroll supervisor and the receptionist) can’t look through it?” Discuss.

DQ3-4  Examine Figures 3.18 (page 101) and 3.19 (page 102). Using the combination of the E-R diagram and the schema, identify each of the referential integrity constraints that should be considered. Explain why each is necessary.

DQ3-5  (Appendix A) Examine Figure 3.18 and the five relationships. Determine the cardinality constraints for each of the relationships. Be prepared to defend the rationale for your selection.
DQ3-6  (Appendix A) Examine Figure 3.18 and the five relationships. Determine the participation constraints for each of the relationships. Be prepared to defend the rationale for your selection.

Notes regarding Problems 1 through 5: These problems should be completed with a database software package, such as Microsoft Access. For Problems 1 through 3, you may use data that you (or your instructor) downloaded from a database. Problem 4 provides an alternative to Problems 1 through 3 by using the database structure and sample data from Figure 3.19 (page 102). This problem may also be completed using the software of your choosing.

P3-1  Before starting this problem, you should consult the customer master file record layout in Figure 3.6 (page 76).

REQUIRED: Using the database software indicated by your instructor:

a. Create the “structure” for the records in the customer file. Use Figure 3.6 as a general guide to the data elements to be included in the customer records. However, observe the following specific requirements:

   (1) Devise your own coding scheme for the “customer number.”
   (2) For the customer address, provide three separate fields, one each for street address, state, and ZIP code.
(3) Provide for two additional data elements that are not shown in Figure 3.6 (because they normally would be accessed from other files)—open sales orders and accounts receivable balance.

b. If the software package supports a function to design input screens, create the screen format to be used for entering customer data.

c. Create hypothetical customer records and key the data into the database. The only design constraint is to use a variety of names, street addresses, states/ZIP codes, etc.
codes, open sales order amounts, accounts receivable balances, and credit limits. (The number of records will be indicated by your instructor.)

d. Obtain a printout of the database records.

P3-2 NOTE: This problem is a continuation of Problem 1.

REQUIRED:

a. “Search” the database for all customers with a ZIP code of ZZZZZ (choose a code that is common to at least two, but not to all, of your customers). Obtain a printout of your search algorithm and a list of customers whose records met the search parameter.
b. “Sort” the database in the *descending* order of credit limit amounts. Obtain both a printout of your sort algorithm and the sorted list of customers.
c. Create a “Customer Status Report” (the report title). Observe the following specific requirements:
   (1) Provide column headings, in left-to-right order, for customer name, credit limit, accounts receivable balance, and open orders.
   (2) For each state, print subtotals of the accounts receivable balance and open orders columns.

P3-3 NOTE: This problem is a continuation of Problem 1.

REQUIRED:

a. Write a “program” to enter customer order amounts into the system and to have the system either warn the user if the new order places the customer over his or her credit limit or advise the user if the credit limit is not exceeded. Store the program in the system, and obtain a hardcopy printout of the program.
b. Test the program developed in (a) by entering the amounts of customer order transactions (use a variety of order amounts and different customers, such that you test all possible combinations of variables involved in the credit-checking algorithm). (The number of order transactions will be indicated by your instructor.) Obtain hardcopy evidence of the results of your testing.

P3-4 Using the database structure and sample data in Figure 3.19 as a starting point (rather than Figure 3.6), complete the requirements of Problems 1 through 3 (or whatever portions of those problems your instructor may indicate).

P3-5 Use the database structure and sample data in Figure 3.19 to:

a. Combine the tables to obtain a complete record of the order and shipment. Obtain both a printout of the algorithm(s) used to combine the tables and the list of these records.
b. Select the inventory items for which there is no order. Obtain both a printout of the algorithm(s) used to select the items and the list of the selected records.
c. Select those orders that have not yet been shipped (i.e., open orders). Obtain both a printout of the algorithm(s) used to select the open orders and the list of the selected records.
d. Calculate the total value (price) of the inventory items that are on hand. Sort the items in descending order of value. Obtain both a printout of the algorithm(s) used to perform the calculations and to sort the records, and a list of the sorted records.

P3-6 This problem asks you to research the literature for controls that apply to database management software.

REQUIRED: Develop a paper that discusses control plans for single user PC database management systems (Microsoft Access or another of your choosing). Your paper should explain how each plan operates (with illustrations where appropriate) and how the plan helps to achieve the information process control goals discussed in Chapter 8 (see Table 8.1 on page 250). (The number of pages will be indicated by your instructor.) Note: Limit yourself to controls that apply only to database application software. Do not discuss PC pervasive control plans.

P3-7 Write a short paper describing the database underlying a small company enterprise system (e.g., Quickbooks®, Peachtree®, MYOB®, or another you have access to).

a. Does it appear to be integrated? When you change an item of data in one application, does it carry through to others? (For example, if a customer’s billing address is changed, do all existing invoices reflect the change?)

b. How easy is it to set up the data relationships for the database?

c. Can you view the schema? Are subschemas supported?

d. Are there any controls in place to ensure that data relationships make sense? (For example, is referential integrity supported?)

P3-8 REQUIRED: Develop an entity-relationship diagram of the Information System that supports the purchasing process of Proware Company described below. Include the cardinality constraints.

The individual departments of Proware Company refer to various vendor catalogs when they need to purchase items. They then complete a purchase request form over the company intranet for each vendor. Once completed, the form is forwarded to the Purchasing department, where it is validated and checked against the department's budget to ensure there are still funds available in its supply budget. Each form is assigned a unique serial number, referred to as the PO number. Purchasing transmits the validated form to the vendor, or faxes it if the vendor is not online.

When the vendor fills the order, it sends an invoice to purchasing and enclose a copy of the invoice with the shipment when it is delivered to the department. The department verifies that the contents match the PO, attaches a note approving payment, and forwards the approved invoice to purchasing. An invoice may only be partially approved, if an item was missing or incorrect. A second payment authorization may be submitted at a later time when the correct item is received.

Each week purchasing generates vendor checks based on the approved authorizations from departments, and maintains a copy of the check in the computer file. Vendors may be paid for multiple orders (e.g. for several departments) in any given week.

Each month, purchasing generates a budget report for each department, which itemizes the amount paid for each invoice, the amount allocated for outstanding purchase orders, and the remaining available funds for the department.
REQUIRED: Develop an entity-relationship diagram of the information system that supports the hiring process of Proware Company described below. Include cardinality and participation constraints.

When a manager needs to hire an employee, he or she first completes an Employee Requisition form over the company intranet, which indicates the position open, the rate of pay, hours, skills needed, and whether the requisition is for a replacement or additional employee.

Once submitted, Human Resources recruits as needed to fill the position. When applications arrive, Human Resources is responsible for prescreening the applicants. Anyone who appears suitable is scheduled for an interview with the hiring manager. The interview date and time is noted on the application, and forwarded to the hiring manager.

The hiring manager completes an interview form after each interview and attaches it to the application. When all of the interviews are complete, the hiring manager gives the name of the top candidate to Human Resources, who prepares an offer letter to be sent to the applicant. The applicant signs the letter to indicate acceptance and returns it to Human Resources. An employee file for the newest member of Proware is created, and the original application, the interview form, and the offer letter are included in it.