PART TWO

DATABASES

A major task for an accounting information system is to collect, record, store, and manipulate financial data, and to covert these data into meaningful information for management decision making. The chapters in Part Two discuss various techniques for accomplishing these tasks using relational databases.

Chapter 4 discusses database concepts in general, and explains the importance of databases to AISs. The chapter also explains data hierarchy concepts, record structures, and record keys. The chapter then outlines in detail the steps needed to design the database files and relationships needed to support important business functions using the REA framework. The final section of the chapter explains how to use these theoretical tools to create database records and tables in Microsoft Access 2007.

Chapter 5 explains how to organize, validate, and manipulate the data in databases. The first section describes normalization of data in databases—a classical design approach compatible with the REA model. The second section discusses methods for validating the data entered into databases—important controls that safeguard the accuracy, completeness, and integrity of the data stored in them. The third section describes procedures for extracting data from existing database tables. This section also explains how to create ''select'' queries in Access 2007 as well as such other extraction techniques as online analytical processing and data mining. The chapter concludes with a discussion of object-oriented databases, multimedia databases, and data warehouses.

Chapter 6 focuses on the development of database forms and reports. The first section of the chapter explains why database forms are important, describes how to create simple forms as well as forms with subforms, and provides guidelines for creating such database objects. The second section of the chapter explains why database reports are important, describes how to create simple reports as well as complex reports based on multi-table queries using Access 2007, and provides guidelines for creating professional outputs.

Chapter 4

Data Modeling

After reading this chapter, you will:

1. *Appreciate* the importance of databases to AISs.
2. *Be able to describe* the concepts of the data hierarchy, record structures, and record keys.
3. *Understand* the uses of data dictionaries.
4. *Be able to explain* why such design concerns as processing accuracy, concurrency, and security are important to multiuser databases.
5. *Know* how to model a database with REA.
6. *Be able to name* three database structures commonly used to create databases.
7. *Know* how to create tables, records, and relationships using Microsoft Access.

No one likes data integration. It's painstaking to automate and hard to measure in terms of ROI. Yet it's required for making systems work together. . . .

Galen Gruman, ''Whipping Data Into Shape'' *Infoworld* Vol. 28, No. 6 (February 6, 2006), p. 27.

INTRODUCTION

Civilizations have stored accounting data in systematic fashion for at least 6,000 years. The ancient Babylonians, for example, stored clay tablets in their temples that recorded such information as inventory receipts and disbursements, payroll information, and real estate transactions. Modern AISs use computers rather than clay tablets, but much of the same organizing requirements remain—the systematic recording of data, convenient and useful formats, and easy access to required information. This chapter examines how to *design* a database, while the next two chapters look at how to *use* a database effectively. We begin by examining some database concepts and then describe database design and data modeling techniques in depth.

AN OVERVIEW OF DATABASES

In some ways, not much has changed since ancient Babylonian days. For example, even the most basic AIS needs to record accounting data in systematic fashion and to organize accounting records in logical ways. Usually, this is done in a database.

What is a Database?

A **database** is a large collection of related data that are typically stored in computerized, linked files and manipulated by specialized software packages called **database management systems (DBMSs)**. Examples of databases include the repositories of information and related files for inventory systems, general ledger systems, and production scheduling systems. In most applications, these systems are complex combinations of data, processing software, and perhaps separate hardware that interact with one another to support the specific storage and retrieval tasks required of them.

Technically, not every collection of data is a database. For example, the time-card data from a weekly payroll system might be stored in a computer, but a single file containing such data *alone* is generally too simplistic to be called a ''database.'' Similarly, the budget or other financial information typically created in spreadsheet programs such as Excel is not a database. Most commercial databases are very large, invaluable collections of proprietary data that developers carefully design and protect, and that often form the core of efficient accounting information systems.

The Importance of Databases to AISs

It is difficult to overstate the importance of computerized databases to AISs. For example, accounts receivable applications must store information about customers, accounts payable

applications must store information about suppliers, and payroll applications must store information about employees. Here are several other reasons why databases are important.

- *Valuable information.* The information stored in an organization's databases is sometimes its most important asset. Equifax, for example, is one of the nation's largest credit bureaus, maintaining credit information about millions of Americans. Its credit files *are* its business.
- *Volume.* Some of the nation's largest databases are truly spectacular. For example, the U.S. Internal Revenue Service maintains records on over 75 million taxpayers. Ford Motor Company maintains a customer database of 50 million records. Citicorp uses a database of 30 million records. For General Foods, the number is 25 million. Organizing and managing databases of such great size are enormous and often daunting tasks.
- *Complexity.* The databases of some organizations are centralized (i.e., stored in a single location at corporate headquarters or maintained on a single file server in a local area network). Many other databases, however, are distributed (i.e., duplicated in local or regional computers as processing needs dictate). But distributing information makes it harder to (1) ensure data accuracy, consistency, and completeness, (2) secure information from unauthorized access, and (3) recreate files with backups in the event of system failures.
- *Privacy.* Databases often contain sensitive information—for example, employee pay rates or customer credit card numbers. This information must be protected from those unauthorized to have it. Some of the most important control procedures for an AIS are those that protect databases from unwarranted access.

> ***Case-in-Point 4.1*** According to a recent GAO report, over 200 agencies of the federal government now have data mining initiatives designed to collect and analyze personal information about U.S. citizens. Many of these projects use data purchased from commercial database sources. Agencies such as the Department of Homeland Security defend such actions as part of the war against terrorism, but civil rights organizations challenge such activities, partly because the accuracy of the information is not known.[1]

- *Irreplaceable data.* The information of most AISs is necessarily unique to the organization that created it and, therefore, often priceless. This is why a special dimension of database management is again file security.
- *Need for accuracy.* The data stored in commercial databases must be complete, comprehensive, and numbingly accurate. This is especially true for government databases. It is also vital that such systems are easy to use and serve their strategic missions.

> ***Case-in-Point 4.2*** The U.S. Army maintains separate payroll, medical, and personnel databases which investigators say are antiquated, complex, and overtaxed. A recent GAO internal audit of the system estimated that as many as 4,000 U.S. soldiers who were seriously wounded in Iraq were either overpaid or underpaid, and found that even trained finance officers often couldn't unravel the mess.[2]

- *Internet uses.* As you might imagine, databases are critical components for both internal and external corporate web systems. These databases store such things as product information for online catalog sales, emails, product registration data, and current corporate

[1] Caron Carlson, "GAO Reports Rampant Federal Data Mining" *eWeek* Vol. 21, No. 23 (June 7, 2004), p. 31.
[2] Alex Kingsbury, "Of Insult and Injury" *U.S. News and World Report*, Vol. 140, No. 16 (May 1, 2006), p. 31.

data about employment opportunities, stock prices, and executive officers. Internet applications often also store customer-entered data such as online product orders, credit card numbers, subscription information, airline reservations, and university-student registration data.

Storing Data in Databases

To be useful, the data in an organization's databases must be stored efficiently and organized systematically. Three important ideas along these lines are the concepts of (1) the data hierarchy, (2) record structures, and (3) record keys.

The Data Hierarchy. Storing accounting data in computer files means organizing the data into a logical structure. In ascending order, this **data hierarchy** is:

bit → character → data field → record → file → database

To illustrate, imagine a payroll file. The lowest level of information in this file is a binary digit or bit. At the second level, a computer combines eight bits to create a byte of data that can represent a single character—for example, a letter of the alphabet or a special symbol such as a plus sign. The third level combines several characters to form a **data field**—for example, an account balance. Other names for a data field are ''attribute,'' ''column,'' or simply ''field.''

At the fourth level, data fields combine to form a complete record. A database **record** stores all the information about one file entity—for example, one inventory part in an inventory file, one employee in a payroll file, or one customer in a customer file. At this level, it may be helpful to liken the structure of a database to the data in a spreadsheet. Each column defines an individual data field, and each row defines a separate record or *tuple*.

At the fifth level of the data hierarchy, a set of common records forms a file, or in Access parlance, a *table*. Thus, a file or table contains a set of related records—for example, a set of inventory records or customer records. **Master files** typically store permanent information—for example, part number, part description, and location code for the individual records in an inventory parts master file. **Transaction files** typically store transient information—for example, inventory disbursements and replenishments for a specific time period.

Finally, at the highest level, several tables and related files create a complete database (i.e., a collection of files that contain all the information for an accounting application). In an inventory application, for example, this database might contain a part-number master table, a supplier table, a price table, an order transaction table, and so forth, as well as several other files (that we shall identify shortly) that might help end users organize, access, or output inventory information efficiently.

Record Structures. The specific data fields in each record of a database table are part of what is called the **record structure**. In many accounting applications, this structure is fixed, meaning that each record contains the same number, same type, and same-sized data fields as every other record on the file. This would probably be the case for the payroll record illustrated in Figure 4-1. In other applications, either the number of data fields in each record might vary, or the size of a given data field in each record might vary. For example, in a file of customer complaints, the memo field in each record might vary in length to accommodate different-sized descriptions of customer problems.

Social Security number	Last name	First name	Dept. code	Pay rate	Date of hire	Over-time OK?	Other info.
575-64-5589	Smythe	Teri	A	12.85	10-15-2001	yes	....

FIGURE 4-1 Some of the data fields in a computerized payroll record.

Record Keys. The **primary record key**, or just ''primary key'' for short, is the data field in each record that uniquely distinguishes one record from another in a database table. For the payroll record in Figure 4-1, for example, the primary record key might be the employee's Social Security number. End users and computer programs use primary record keys to find a specific record—for example, the record for a particular employee, inventory item, or customer account. Businesses sometimes combine two or more data fields to serve as the record key for a computer record. For example, a bank might combine its branch code with a customer's account number to serve as the record key. Another example would be a ten-digit phone number for a customer, separated into an area code and a local phone number. End users often create a primary key field as the first field in a record, but this is not a requirement.

It is also possible for a computer record to have more than one record key. For the payroll file of Figure 4-1, some examples are the employee's last-name field or the department-code field. These data fields, which are typically not unique across records but can also be used to search records for specific information, are examples of **secondary record keys**.

Finally, some accounting records contain data fields called **foreign keys** that enable them to reference one or more records in other tables. For example, in addition to the payroll table in Figure 4-1, a firm might have a department table with the data fields shown in Figure 4-2. The primary key for the department table is the department code (e.g., ''A,'' ''B,'' and so forth). With this arrangement, the department code field in the payroll record of Figure 4-1 would be a foreign key that the database system could use to reference the appropriate department record from the department table. These foreign keys enable a database system to combine the information from both tables to produce a report such as the one in Figure 4-3.

Note that each line of this report contains information from the records in two tables: the employee records in Figure 4-1 and the department records in Figure 4-2. To create this report, the designers for the entire application must examine the data carefully and organize them efficiently. The following sections of the chapter explain this analysis in detail.

Additional Database Concerns

Small database systems such as the kind used by very small businesses or sole proprietorships tend to be fairly straightforward and manageable. However, large, multiuser databases pose

Department code (primary key)	Manager	Number of employees	Location	Secretary phone	Other info.
A	B. Wright	45	Bldg. 23	x8734	...

FIGURE 4-2 A sample record from a department file.

Employee Roster
Friday, July 28, 20XX

Last Name	First Name	Dept.	Manager	Location	Secretary Phone
Garadis	Sue	B	Garadis	Bldg. 23	ext. 9330
Gold	Karen	A	Wright	Bldg. 23	ext. 8734
Hale	Lois	C	Hale	Bldg. 24	ext. 8655
Smythe	Teri	A	Wright	Bldg. 23	ext. 8734
Wright	Barbara	A	Wright	Bldg. 23	ext. 8734

FIGURE 4-3 A formatted report that uses data from two tables.

special challenges for their designers because of their size and complexity. Here, we describe some database design concerns that are of special importance to accounting applications.

Administration. Without an overall supervisor, a large commercial database is somewhat akin to a rudderless ship—i.e., an entity without cohesion or direction. Similarly, it does not make sense to permit database designers to work unsupervised, or to develop large databases of critically important information without also creating accountability for subsequent changes. A **database administrator** supervises the design, development, and installation of a large database system, and is also the person responsible for maintaining, securing, and revising the system's data.

> ***Case-in-Point 4.3*** "A qualified database administrator must be a jack of all trades," says Ed Tittel of iLearning.com, a training institute. Skill requirements include designing database systems from scratch, maintaining and updating database information, backing up and restoring database systems, creating user reports, and assisting users with data mining tasks. And as valuable as training is for such jobs, "experience beats certification hands down" he says.[3]

Documentation. Database developers often change database elements during the design phase, as well as later to an operating database system. This makes documentation critical. Descriptions of database structures, contents, security features, E-R diagrams (discussed later in this chapter), and password policies are other examples of important documentation materials. In addition to all these items, it is usually vital to document "what stores what." The **data dictionary** of a database describes the data fields in each database record. In other words, a data dictionary is a data file about data. Although a data dictionary can be manual, it is usually a separate computer file that database administrators create and maintain.

Figure 4-4 identifies some generic information that a data dictionary might contain (listed under the "Entry" column) and an example of such information for a Social Security number (listed under the "Example" column). In this figure, the data dictionary indicates that the Social Security number data field must be nine characters, is a "text" data field (rather than a "number" data field because it is not manipulated mathematically), has no

[3]Source: Ed Tittel, "Certified Expert: Working as a Database Administrator" Vol. 5, No. 1 (January 2003), p. 44

Item	Entry	Example
1	Field name	Social Security number
2	Field size	9 characters
3	Type of data field	text
4	Default value	none
5	Required?	yes
6	Validation rule(s)	all digits must be numeric characters
7	Range	none
8	Source document	employee application form
9	Programs used to modify it	payroll X2.1
10	Individuals allowed access	payroll personnel
11	Individuals not allowed access	non-payroll personnel

FIGURE 4-4 Examples of information that might be stored in a data dictionary for the Social Security number data field of a payroll database.

default value, and so forth. From this illustration, it should be clear that the entries in the data dictionary describe each data field in each record of each table (file) of an AIS database. When developers add a new data field to the record structure of an existing table, they also add the appropriate information about the new field to the data dictionary.

Data dictionaries contain **metadata or** data *about* data, and have a variety of uses. One is as a documentation aid for those who develop, correct, or enhance either the database or the computer programs that access it. As suggested in items 10 and 11 of Figure 4-4, an organization can also use a data dictionary for security purposes—for example, to indicate which users can or cannot access sensitive data fields in a database.

Case-in-Point 4.4 How can you store a database on a mobile device like a cell phone? One way is to compress the data. An alternate approach by the WindSprings corporation is to use a data dictionary with index pointers. Using this technology, the company was able to shrink a 125k byte map of San Diego to 24k—small enough to fit most mobile devices—while still enabling the user to pan and zoom without further calls to an Internet server.[4]

Accountants can also make good use of a data dictionary. For example, a data dictionary can help establish an audit trail because it identifies the input sources of data items, the potential computer programs that use or modify particular data items, and the managerial reports on which the data items are output. When accountants help design a new computer system, a data dictionary can help them trace data paths in the new system. Finally, a data dictionary can serve as a useful aid when investigating or documenting internal control procedures because the basis for data-entry tests, methods of data security, and so forth, can be stored in the data dictionary.

Data Integrity. IT professionals estimate that it costs about ten times as much to correct information that is already in a database as it does to enter it correctly initially. Then, too, even simple errors in databases can lead to costly mistakes, bad decisions, or disasters. (Think about air traffic controllers as an example!) For these reasons, the software used to create databases should also include edit tests that guard databases from erroneous

[4]Source: John R. Quain, ''Sizing Down and Speeding Up'' *PC Magazine* Vol. 24, No. 12 (July 1, 2005), p. 22.

data entries. These **data integrity controls** are designed by the database developers and are customized for the application at hand. Examples include tests for data completeness, conformance to the data type specified for the data field, valid code tests (e.g., a state code such as ''CA''), and reasonableness tests (e.g., regular payroll hours worked must be between ''0'' and ''40''). We shall return to this point in Chapter 5.

Processing Accuracy and Completeness. Within the context of database systems, *transaction processing* refers to the sequence of steps that a database system uses to accomplish a specific processing task. AISs need **transaction controls** to ensure that the database system performs each transaction accurately and completely. To illustrate, imagine an inventory application with two types of inventory records: raw materials records and work-in-process records. An inventory manager wishes to subtract 200 units from a particular raw materials record and add the same number of units to a corresponding work-in-process record.

Now suppose that the database system executes the first part of this transaction (i.e., subtracts 200 units from the raw materials record) and then stops operating for some reason. This is a problem because the transaction has not been executed completely and the balance-on-hand field in the current work-in-process record is wrong. To overcome this problem, databases should either process a transaction entirely or not at all. To achieve this goal, database systems maintain an auditable log of transactions. When a specific transaction only partially executes, the system is now able to recover by verifying that a problem has happened, reversing whatever entries were made, and starting anew. In accounting applications, therefore, the ability to audit any particular transaction to ensure processing accuracy and completeness is critical.

Concurrency. In multiuser systems, it is possible for more than one user to access the same database at the same time. Without **concurrency controls**, it is also possible for two or more users to access the same *record* from the same table at the same time. This creates problems. To illustrate, imagine the same inventory file as the one discussed previously and suppose that ''user A'' and ''user B'' access the same inventory record at the same time. The initial balance-on-hand field for this record is 500 units. When User A accesses this record, the system transfers the entire record to A's work area. User A wants to add 100 units to the balance-on-hand field. The result is a new balance of 600 units. User A completes this transaction, the system writes the new record back on disk, and the new balance on hand in this record is now ''600 units.''

When User B accesses this same record, the system also transfers the same initial record to B's work area. User B wants to decrease the balance on hand by 200 units. This results in a balance of 300 units because this user also starts with an initial balance on hand of 500 units. Because B completes this transaction after A is done, the system replaces the current record in the database with the new one. The end result is an inventory record with a balance on hand of 300 units, not the correct value of 400 ($= 500 + 100 - 200$). To guard against this problem, database systems must prevent multiple-user access to the same file record. Rather, these systems must execute transactions serially (i.e., sequentially).

Backup and Security. As noted earlier, the information in many accounting databases is both invaluable to the day-to-day operations of a company and, because it is unique, irreplaceable. It must be protected. A key security feature of any database, therefore, is backup procedures that enable an organization to recreate its data if the original copies are lost or damaged.

Case-in-Point 4.5 Several companies found out just how complete their disaster recovery and backup procedures were when terrorists attacked the World Trade Center on September 11, 2001. For many, the damage included the loss of (1) data, (2) personnel most knowledgeable about that data, and (3) the building in which the data were stored. Dean Witter and Company, a large brokerage house, had prepared for such a contingency and resumed business within two days in makeshift quarters across the Hudson River from the WTC towers in New Jersey. In contrast, Visa was among the 25 companies in the WTC that had subscribed to an elite, EDS "hot-site service." For these companies the delay was much shorter—in the case of Visa, three minutes![5]

In addition to backup security, an organization must also protect databases from unauthorized access. Another security feature, therefore, is a system's ability to assign, maintain, and require employees to use passwords and guard against unwarranted intrusions. Similarly, database systems can use encryption techniques to scramble data into unintelligible formats, thereby protecting file data even if an unauthorized user obtains access to the company's database.

Case-in-Point 4.6 A recent survey by Cisco Systems, Inc., found that, over 67% of the (more than 2,000) respondents had performed at least one act that threatened their company's database security. The most common breach, at 37%, was failing to log off before stepping away from a personal computer with access to corporate data.[6]

A final database security feature is to use **view controls** that limit each user's access to information on a need-to-know basis. In an inventory application, for example, a defense contractor might limit employee access to its supplier files, inasmuch as information about supplier identities and perhaps part prices might be sensitive information. We cover intrusion detection systems and controls in Chapter 12.

STEPS IN CREATING A DATABASE USING REA

At a state department of social services, the director wants to know how many inquiries were made for a certain type of medical assistance last month. At the headquarters of a department store chain, a vice president wants to know how many credit customers made partial payments to their accounts last month. At a local university bookstore, a manager wants to know how many book orders went unfilled last month.

In each case above, the decision-maker needs information. AISs must gather pertinent data and store the information in formats that enable managers to obtain timely answers to important organizational questions. The challenge of creating large, useful databases is to determine what data to collect, and how to gather, record, organize, and store the data in ways that satisfy a number of objectives. One obvious goal is to satisfy the informational output requirements of the system. A second task is to find hardware and software solutions that can adequately perform the data-gathering, storage, and reporting tasks involved. Another goal is to keep the databases manageable—for example, keep them from becoming too large, complex, and unwieldy. A fourth goal is to protect the privacy of

[5]Source: David O. Stephens, "Protecting Records in the Face of Chaos, Calamity, and Cataclysm" *Information Management Journal* Vol. 37, No. 1 (2003), pp. 33–40.

[6]Source: No author, "Worker's Circumventing IT Security, Putting Company Data at Risk" *Security Director's Report* Vol. 8, No. 12 (December 2008), p. 8.

sensitive information. A fifth goal is to reduce data redundancy, which means storing the same data repeatedly in different tables. These goals make it obvious that databases must be carefully designed to serve their intended uses. The question is, ''How do we do this?''

When a company wants to create a database, it normally hires a database consultant to help it design a new database that meets the organization's needs. Based on the information obtained from managers and end users, the expert then uses a process called **data modeling** to design the database. This is usually the most challenging step in the process of creating a database because the designer must collect a considerable amount of information by investigation and interviews, and then integrate the needs of all stakeholders as accurately and completely as possible.

Although there are a number of different models that may be used to design a database, the one we will describe and use here is the **REA Model**. This model is an acronym for resources (R), events (E), and agents (A). The REA model requires the following steps: (1) identify business and economic events, (2) identify entities, (3) identify relationships among entities, (4) create entity-relationship diagrams, (5) identify the attributes of data entities, and (6) create database tables and records to populate the database. The following discussions describe each of these steps in detail, using the sales process as an example.

Identify Business and Economic Events

Chapters 7 and 8 will discuss business processes and explain that these processes involve a series of events or identifiable activities. There are primarily two types of events: economic events and business events. **Economic events** impact an organization's financial statements, and AISs therefore record data about them in accounting transactions. An example would be a sale on account. This economic event increases an entity's accounts receivable and revenue accounts on its financial statements.

As noted previously, critics sometimes claim that financial accounting systems often ignore organizational activities and events that are important to managers, investors, and creditors. Such **business events** do not affect financial statements but can impact an organization in a value-added way. One example of such an event is a *sales order* from a customer. Because sales orders do not require journal entries, they do not appear anywhere in a company's financial statements. However, suppose that a company received a sales order from a customer that was equal to all its revenues for the previous quarter. This would certainly be important information that many individuals, both inside and outside the firm, would want to know. Another example is when a firm hires a new CEO. Again, this event does not require a journal entry, but is important information for stakeholders.

When creating a database using an REA approach, a systems designer will try to record all events in the database, whether they are business or economic ones. By including both types of events in the database, users can access important information about both business and economic activities.

Identify Entities

Databases contain data about objects of interest called **entities**. Database entities include business and economic events, plus information about ''who'' and ''what'' were involved in those activities—i.e., the system's agents and resources. **Agents** are the ''who'' associated with events. For example, both a salesperson and a customer participate in creating a merchandise sale. We would classify both of them as ''agents.''

Events use or generate **resources**. For example, a merchandise sale may require an inventory resource and generate a cash resource. Resources are very similar to accounting assets, but they are more all-inclusive. For instance, we might classify a contract as a resource, but it would not appear as an asset on a financial statement. To determine whether or not something constitutes a resource associated with an event, the resource should pass two tests. First, it should be an object of value associated with an event. Second, it should be an object of sufficient interest that you would want to collect information about it.

The REA model helps identify database entities because each resource, event, and agent is an entity in a relational database. Figure 4-5 provides several additional examples of each type of entity. You may notice that Figure 4-5 does not list accounts receivable as a resource. This is because the REA model does not recognize ''receivables'' or ''payables'' as resources. Rather, receivables and payables are by-products of an information event and only represent *claims* on resources rather than resources themselves. Similarly, the REA model does not treat ''billing'' as a business or economic event because a bill really just conveys information about an economic event such as a sale or purchase.

Identify Relationships Among Entities

A database should contain a table for each entity. The table consists of rows of records, each containing data fields that describe the entity's attributes. Figure 4-6 shows four database tables for our merchandise sale example: (1) an event table (Customer Order), (2) a resource table (Inventory), (3) an agent table (Customer), and (4) another agent table (Salesperson).

Entities are usually related to each other. For instance, a sale may be *of* merchandise inventory and made *to* a customer. The relationship between a sale and inventory or between a sale and a customer is a *direct relationship*. Inventory and customer also share a relationship, but it is an *indirect relationship*. Typically, events have direct relationships with resources and agents, and also with other events. The links between resources and agents are *through* events.

Data modelers need to know about entity relationships so that they can create links between database tables. Without these links, database users could not access data from more than one table at a time. Referring again to the tables in Figure 4-6, in the absence of any database links, users could obtain reports about order data, inventory data, or customer data. But the database system would not be able to show the customer name on a customer invoice because it would require information from more than one table for this task.

Before we can decide on the best way to link database tables, we must first understand the nature of the relationships among entities. We describe these relationships in terms of **cardinalities**. Cardinalities are a notation showing the nature of relationships among

Resources:	Events:	Agents:
Cash	Sales Order	Employee
Contracts	Sales	Customer
Inventory	Purchase Order	Vendor
Equipment	Purchase	Manager
Plant Facilities	Receive Goods	Stockholder
	Hire an Employee	Creditor

FIGURE 4-5 Examples of resource, event, and agent entities.

Customer Order Table (Event)

Order #	Employee #	Customer #	Date	Comments
1003	M24SP	B104	01/03/02	
1004	R63SP	P202	01/03/02	Ship ASAP
1005	M24SP	S200	01/03/02	
1006	W11SP	C100	01/03/02	

Inventory Table (Resource)

Item #	Description	Unit Cost	Sales Price	Beg QOH
1400	Goodie Bar	$0.20	$0.40	13025
1500	Almond Delight	$0.25	$0.45	5010
1600	Gummy Lions	$0.60	$0.95	20109
1700	Pecan Bar	$0.70	$1.09	4508
1800	Milky Bars	$0.18	$0.30	2207

Customer Table (Agent)

Customer #	Name	Address	City	State	Zip Code	Credit Limit
A101	Amanda Wills	22 Yellow Ln.	Charlotte	NC	79803	$20,000.00
B104	Boris Bailey	321 Church St.	Oxford	OH	45056	5,000.00
C100	Carly Riccardi	1899 Green St.	Dayton	OH	43299	10,000.00
P202	Peggy Martin	1260 Main St.	Columbus	OH	43320	10,000.00
S200	Bill Safer	860 Broad St.	Fairfax	VA	22030	5,000.00

Salesperson Table (Agent)

Employee #	Name	Address	City	State	Zip Code	Dept ID	Date Hired
A06SP	Sally Anderson	3026 Skye Ln.	Columbus	OH	43213	247	1/31/1989
M24SP	Randy Merit	262 Main St.	Bexley	OH	43209	182	7/2/1999
R63SP	Barry Rogers	80 N. Long St.	Gahanna	OH	43215	247	1/16/2001
R73SP	Jim Rudolph	64 Lantern Ave.	Columbus	OH	43213	76	8/15/2000
W11SP	John Walker	1028 Fields Ln.	Lancaster	OH	43307	182	9/1/1992

FIGURE 4-6 Four sample tables in a relational database.

entities as *one-to-one, one-to-many, none-to-one, none-to-many*, or *many-to-many*. A one-to-one relationship between two entities, shown as (1,1), means that the entities relate to each other a minimum of one time and a maximum of one time. An example of a one-to-one relationship is the relationship between sales and customers. In a particular organization, the relationship might be that a sale is to a minimum of one customer (a sale cannot exist without a customer) and a maximum of one customer (an individual sale can be to only one customer).

Entity relationships are two-way. Not only does a sale relate to a customer, but customers also have relationships to sales. The relationship between a customer and a sale may be none-to-many (0,N). This would be the case if a customer could exist without a sale (for example, you first research the credit ratings of potential customers before selling them goods). There are also many sales to each customer (the usual case). The two-way relationship between a sale and a customer, then, can be shown as:

(Sale 1,1; Customer 0,N)

We would read this cardinality as: each sale is to a minimum of one customer and a maximum of one customer, and each customer has a minimum of zero sales and a maximum of many sales.

Cardinalities are sometimes difficult to grasp at first but they become easier to understand with practice. So let's try another one. What does the following cardinality tell us?

(Inventory 0,N; Sale 1,N)

Part of the answer is that inventory relates to a sale a minimum of zero times and a maximum of many times. This makes sense in a business organization that keeps inventory on hand to meet future sales. It is also likely that in most businesses, each type of inventory item can be involved in more than one sale. (Think about, for instance, a retail clothing store that stocks several white shirts in a specific size and style.) The rest of the answer is that a sale relates to inventory a minimum of one time and a maximum of many times, or each sale must be for at least one inventory item and may be for many inventory items. (So you would have to buy something in order to have a sale and you could be buying a white shirt plus some jeans and a jacket.)

Cardinal relationships are not fixed across organizations, but vary according to the rules or controls of the specific enterprise. To illustrate using our sales example, recall that a company could have a customer with no sales or a customer with many sales (0, N). This is probably true for some businesses, but not for others. For example, a video rental store will usually collect information about customers before renting movies to them. In contrast, a retail clothing shop may not consider someone a customer until it sells something to this person. Thus, cardinalities can be helpful in describing an organization's rules and thus can also tell us something about the controls for a given business process.

There is just one more point to make about cardinalities. In the case of a sequence of events, you will nearly always have a situation where subsequent events require a minimum cardinality of 1, and earlier events have a minimum of 0. This would be the case between Customer Order (0,N) and Sale (1,N). What these cardinalities mean is that each order relates to a sale (signified by a shipment of goods) a minimum of zero times and a maximum of many times. In plain English, this says that an order may result in no sales or many sales. This makes sense because a customer may place an order that is never shipped or, perhaps due to backorders, requires several shipments.

The other side says that each sale relates to an order a minimum of one time and a maximum of many times. Again, plainly stated, this means that you cannot have a sale without an order, but you could ship several orders at once. Do you see why you could have an earlier event that might not result in a later event but, as a rule, would require an earlier event to take place before a later one was possible? It would be bad business to ship goods without an order.

Create Entity-Relationship Diagrams

Database designers use a graphical documentation technique called the **entity-relationship (E-R) diagram** to depict the entities and their direct relationships. The model consists of four symbols: rectangles, diamonds, ovals, and connecting lines. Rectangles represent entities; diamonds describe the nature of relationships; ovals denote an entity's attributes; and connecting lines depict relationships. Figure 4-7 provides examples of these symbols. For the sake of convenience, we may drop the diamonds and ovals, thus showing only entities and relationships.

Figure 4-8 is an E-R diagram that includes cardinalities for a sample business enterprise. Remember that these cardinalities could change, depending on an organization's rules or

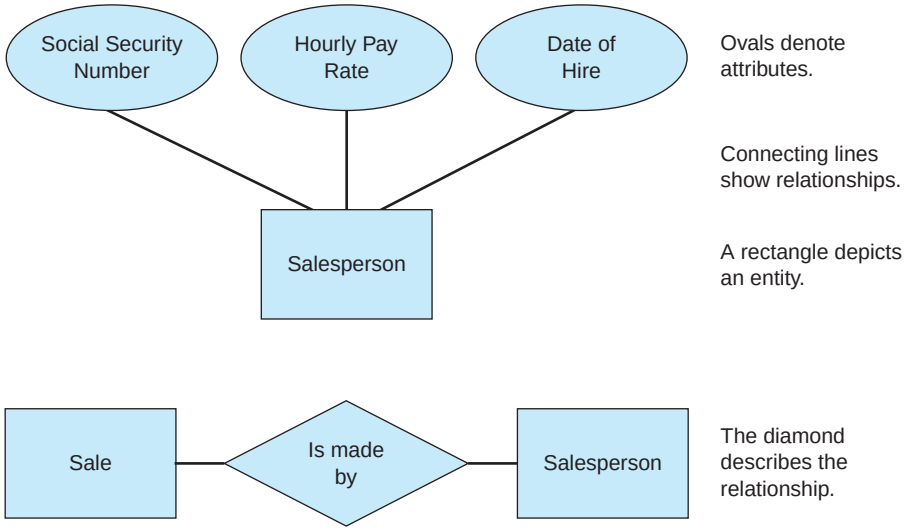


FIGURE 4-7 Examples of entity-relationship (E-R) diagram symbols.

policies. For example, suppose a business starts selling services in addition to products. The cardinality between sale and inventory could change from (Inventory 0,N; Sale 1,N) to (Inventory 0,N; Sale 0,N). Do you see the difference? The cardinality now specifies that each sale can be for no inventory items, in the case of selling a service, or for many inventory items.

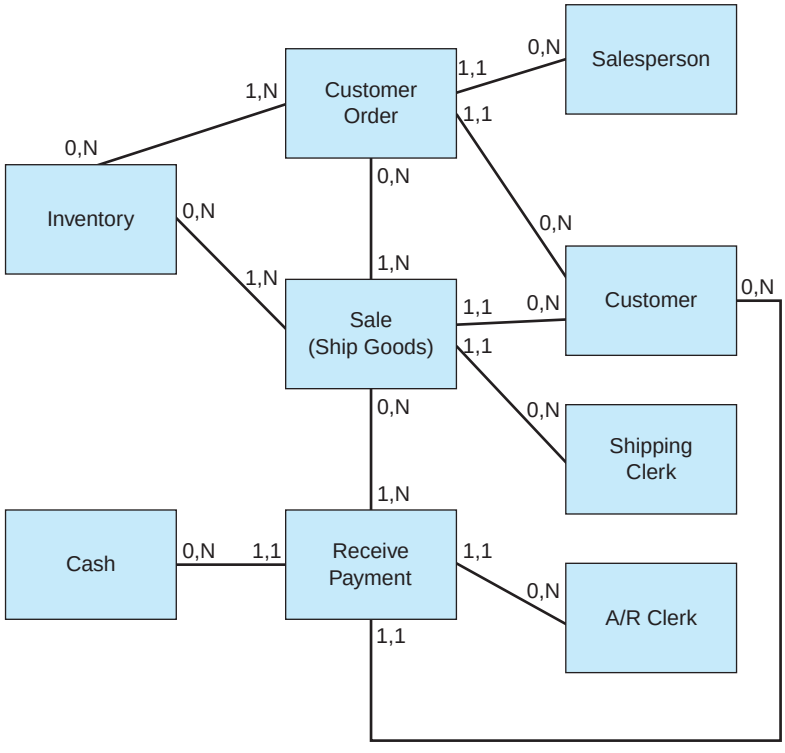


FIGURE 4-8 A sample E-R diagram for the sales process, including cardinalities.

Identify Attributes of Entities

Entities have characteristics or **attributes** that describe them. We know that in a database model, a database table represents each entity. But what data appears in the table? The data within a table will be based on the attributes. For example, a salesperson is an agent, which is an entity. The attributes are the data fields *describing* each salesperson. What data should you collect about a salesperson? Because each salesperson within the Salesperson database table is a unique record, one attribute should be unique to that record. This is the database primary key that we discussed earlier. The salesperson's identification number, which could be the employee's social security number, would be a likely attribute of a salesperson entity. Other attributes might be last name, middle name, first name, phone, address, email, date of birth, date hired, department assignment, salary, and so on.

It is not always easy to decide what to include as attributes of an entity. There are, however, two guidelines you can use. First, the attributes should describe one entity and that entity only. For example, if you have an inventory table, you would not include information about the vendor in this table. You can reference the vendor, but the name, address, and other information about the vendor belongs in a separate Vendor table. A second guideline for determining entity attributes is to keep in mind that the attributes included in the tables will *determine* the outputs of the database system. What you fail to include as an attribute is data that you will not collect and cannot report. For instance, have you ever been asked for your Zip code while shopping in a retail store? If so, the store is collecting an attribute of a sales transaction that can also be of value—for example, data that can help the store determine where to advertise or perhaps where to build another store.

Organizing Database Records

There are several ways to organize the individual records in a database. The particular method used is called the **database structure**. As with other design elements, the objective is to develop an efficient structure that enables users to access data quickly and store data efficiently. Three types of database structures are (1) hierarchical, (2) network, and (3) relational.

Hierarchical Structures. Accounting data are often organized in a hierarchy. For example, a sales office will have several salespersons, each salesperson will have several customers, each customer can make several purchases, and each customer invoice can have several line items. The result is a natural **hierarchical structure**, with successive levels of data in an inverted, tree-like pattern. For this reason, hierarchical database structures are also known as **tree structures**.

Typically, hierarchical data structures have a genealogy that naturally organizes the data into a series of one-to-many relationships. For any two adjacent records, the ''elder'' or higher-level record is called the **parent record**, while the ''younger'' or lower-level record is called the **child record**. Two records on the same level (e.g., two line items on the same purchase invoice) are called **sibling records**.

Network Structures. Often, the data stored in an AIS are interrelated in several ways (i.e., in many-to-many relationships), and thus a single hierarchical structure cannot capture their relationships adequately. At a university, for example, students each take

several classes, and each class has many students. In such instances, AIS databases can use a **network structure** to link related records together and capture these relationships. This linking is usually accomplished with pointer fields embedded in each record that contain the disk addresses of related records. For example, the payroll record of Figure 4-1 could contain a pointer field for another employee working in Department A. The pointers maintain the data relationships, thereby enabling an AIS to prepare familiar reports—for example, a list of all employees working in Department A.

Relational Structures. Hierarchical and network database structures require advanced planning. This means that, if accounting data of one type (e.g., employee information) must be used with accounting data of another type (e.g., payroll information), the database must be planned to create these linkages. But many relationships can exist among data items, and it is difficult to anticipate all of them at the time designers first create a database. Thus, hierarchical and network data structures afford little additional flexibility once further data processing needs are discovered.

This problem is overcome with a **relational database structure**, which enables designers to identify relationships at the time the database is first created, or later, as users discover new informational requirements in the future. Each entity in the E-R diagram will be a table in the database. However, a database is likely to contain more tables than those representing entities. This is because we must provide links among the database tables to represent relationships among tables. As noted earlier, without these links a user would be unable to produce any database outputs that use information contained in more than one table.

The rows of a database table are individual records for database entities and the columns are entity attributes. Two important features of records are: (1) within a record, there should be no attributes that are a result of a mathematical computation, and (2) there should be no repeating attributes. The reason that attributes should not be mathematical computations is because the system itself can recalculate them as needed. For example, in a student-records database, there is no reason to store a student's grade point average (GPA)—this can be calculated from other information in the database about that student.

Repeating attributes typically occur when you attempt to store too much data in the same table, or too much data in the data field of the same record of one table. For example, suppose a charity creates a ''Contributors'' table in a database and uses a single data field in each record in that table for an individual's donation. How would you store the information for a donor who made two donations? Creating duplicate records for the *same* contributor table doesn't make sense (because it duplicates the contributor information), and neither does storing the two donation values in the single donation field of the *same* record (because this isn't possible).

The solution to this problem is to create two tables: one table for contributors and a separate table for donations. Then, the problem is how to link the two tables together. There are two ways to do this within a relational database. The first uses foreign keys as described earlier. For example, in Figure 4-6, the Customer Number in the Customer Order table is a foreign key that references the primary key of a particular customer in the Customer table. As noted earlier, therefore, this value enables database software to link the two tables together—for example, to create a customer orders report that shows the *name of the customer* associated with each order.

Linking tables with foreign keys is only appropriate when you do not have a many-to-many relationship between two entities. Looking at the sample E-R diagram for a sales process in Figure 4-8, for example, we see that the cardinality between Customer Order and Salesperson is (Customer Order 1,1; Salesperson 0,N). This is not a many-to-many

relationship, so we can use a foreign key to link the tables to one another. In deciding which key to use as a foreign key, the general rule is to use the primary key from the table closest to a relationship (nearest the cardinality in the E-R diagram) containing a ''many'' or N, as the foreign key in the other table. In our example, this means that we would use the primary key from the Salesperson table as the foreign key in the customer order table. Looking at Figure 4-6, this is the case. The primary key for the Salesperson table, Employee #, appears in the Customer Order table.

A second way to represent relationships between two database tables is by creating a separate **relationship table**. Relationship tables are necessary when you have many-to-many relationships between database entities. The reason for this is that, without them, you would need to have repeating fields in a database table. For example, there is a many-to-many relationship between Sale and Inventory in Figure 4-8: (Inventory 0,N; Sale 1,N). Because a sale can be for multiple inventory items, if we posted inventory items in the Sale table, we would have to leave many fields available for the primary key for inventory.

Alternatively, because a company can sell each inventory item many times, there would have to be repeating fields for the sales number field in the Inventory table to allow for this. To avoid these repeat fields, data modelers use relationship tables. A simple relationship table just lists the primary keys of the two tables that it joins. More complex relationship tables may include other data, such as quantity. Figure 4-9 shows a relationship table joining the Customer Order and Inventory tables. Notice that some of the orders are for more than one type of inventory item.

How many tables, including join tables, will we have for a complete database and the sales process described in Figure 4-8? Looking at the diagram, we see that there are nine entities. If we have a table for each entity, the database would require nine tables. There are also three many-to-many relationships: (1) Inventory and Customer Orders, (2) Inventory and Sales, and (3) Sales and Receive Payment. Therefore, we might have as many as twelve tables in the finished database: nine tables for entities and three additional ''joining'' tables. There is another possibility, though: three of the entities are employees. It might be possible therefore to use just one database table for employees if we include a field or identifier that specifies the employee type. For instance, we could have a column or attribute for employee classification, and within that you would specify salesperson, cashier, and so on. This reduces the table count to ten.

Figure 4-10 lists all the database tables and their attributes for our sales-process example. Because data modeling is a creative effort, there are many other possible sets of database tables and other attributes that you might include in a database for a sales process. Figure 4-10 is only an example.

Sale #	Item #	Quantity
1003	1400	230
1004	1400	430
1005	1600	180
1005	1800	200
1005	1900	360
1006	1400	80
1006	1800	100

FIGURE 4-9 A relationship table joining the Customer Order and Inventory tables.

Inventory Table
Item#, Description, Unit Cost, Sales Price, Beginning Quantity on Hand, Beginning Quantity on Hand Date

Cash Table
Account#, Account Type, Bank, Beginning Balance, Beginning Balance Date

Customer Order Table
Order#, [Employee#], [Customer#], Date, Comments

Sales Table
Sale#, [Employee#], [Customer#], Ship Date, [Order#]

Receive Payment Table
Cash Receipt#, Amount Received, Date, [Employee#], [Account#]

Employee Table
Employee#, First Name, Middle Name, Last Name, Address, City, State, Zip Code, [Department#], [Job Classification Code], Date of Birth, Date Hired, Last Date of Review

Customer Table
Customer#, Company Name, Address[1], City, State, Zip Code, Contact Person, Credit Limit

Inventory/Order Relationship Table
Order#, Item#[2], Quantity

Inventory/Sale Relationship Table
Sale#, Item#, Quantity

Sale/Receive Payment Relationship Table
Sale#, Cash Receipt#

Order/Sale Relationship Table
Order#, Sale#

[1]May use multiple addresses for different departments or for shipping versus billing.
[2]Relationship tables require two fields together to represent a primary key. Either field alone would not be unique to a record.

FIGURE 4-10 A schematic of database tables for the Sales Process. (*Note:* Underlining signifies a primary key and brackets denote foreign keys.)

CREATING DATABASE TABLES AND RECORDS

It is only after system designers have gone through the steps outlined above that they can begin to create database tables with records. This section of the chapter explains how to perform these tasks using Microsoft Access. The procedures for creating tables and records in alternate database systems are similar.

An Introduction to Microsoft Access

Microsoft Access is a popular relational database that many businesses and individuals use for small database applications. Although this software has many of the same tools, dialog boxes, and menu options as Microsoft Word or Excel, there are some important differences. In Access 2007, tabs and buttons have replaced many of the menus. One capability of

FIGURE 4-11 Opening screen for Access Database Program.

Access 2007 is the ability to use predefined templates. Some of them are Assets, Contacts, Issues, and Events (Figure 4-11). If you click on one of these, you can name your database and download the template from the web. These templates are ready to use and allow you to add fields, delete fields, or use pre-made queries or reports.

Figure 4-12 shows the starting screen for Access 2007. If you click on the ''Blank Database'' icon in the top portion of the screen in Figure 4-11, you will launch the option to open a new blank database (Figure 4-12). A panel will appear on the right side of your screen asking you to name your database. The default name is ''Database1.accdb.'' The ''accdb'' suffix stands for ''Access database'' and Access will attach it automatically if you rename your database (which you should!)—you don't need to add it.

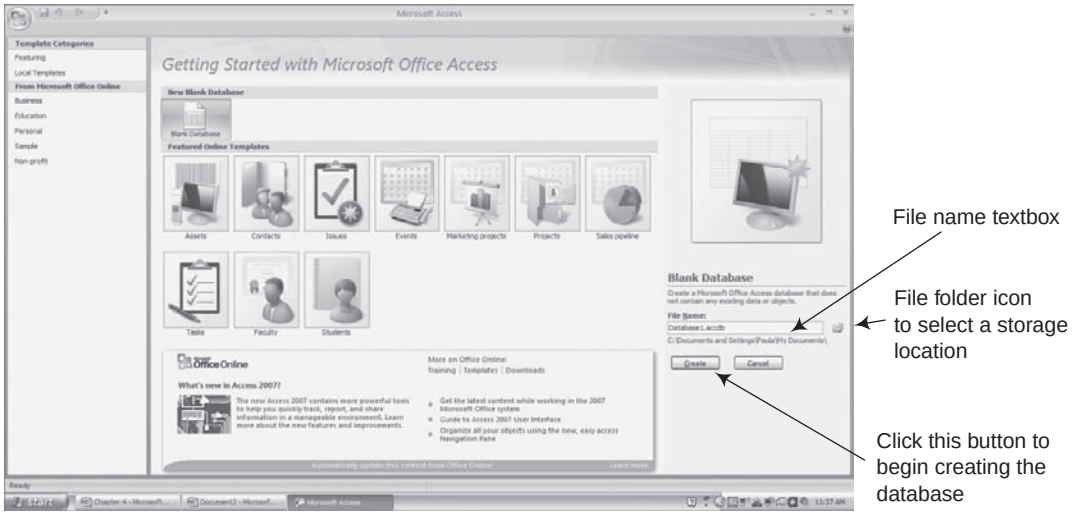


FIGURE 4-12 The screen for getting started with Microsoft Access.

Creating Database Tables

As you already know, database tables store data about specific table entities—e.g., customers, vendors, or employees. To illustrate how to create tables in Access, let's create a table of payroll records similar to the one in Figure 4-1.

Getting Started. Your first task is to rename your database something more meaningful—for example, ''My First Database.'' (Blanks are permitted in Access database names.) Type your new name in the filename box. Your next task is to decide where to store it. To do this, note the file folder icon with an arrow to the right of the filename textbox. Clicking on this icon will display a Microsoft ''Save As'' dialog box (not shown) that enables you to select where to store your database. After you have done this, click on the ''Create'' button in the lower-right portion of Figure 4-12. A larger version of the screen shown in Figure 4-13 will appear.

Defining a Record Format. The Ribbon across the top of the screen in Figure 4-13 shows five tabs: Home, Create, External Data, Database Tools, and Datasheet. The figure also shows two important components. First, Access assumes that your next job is to create a table of records, and accordingly supplies the default name ''Table1'' in the left portion of the screen in Figure 4-13. Second, the system assumes that each record will have at least one data field with default name ''ID'' as shown in the right side of the screen.

Before you enter data in your new database, you must first define the record structure for your table. *It is much easier to spend time developing this format prior to entering data than to spend hours changing it later*. Figure 4-14 displays the form for developing your database. To get to this screen, right click on the table name ''Table1: Table'' and select ''Design View'' from the set of choices in the drop-down menu that subsequently appears.

The screen in Figure 4-14 is a template for creating the record format (i.e., the data fields) of your records. To define a record format, begin typing the name of the first data field you wish to create—e.g., the term ''SocSecNum''—in the upper right portion of the screen in Figure 4-14. When you do, the following three columns will appear in that area of the screen: (1) Field Name (which is required), (2) Data Type (also required), and (3) Description (optional). Let's look at each of these items separately.

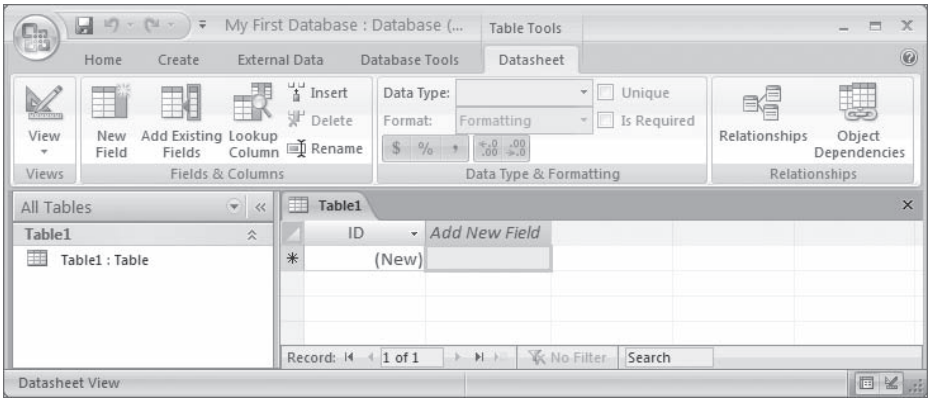


FIGURE 4-13 The opening screen for creating a table in Access.

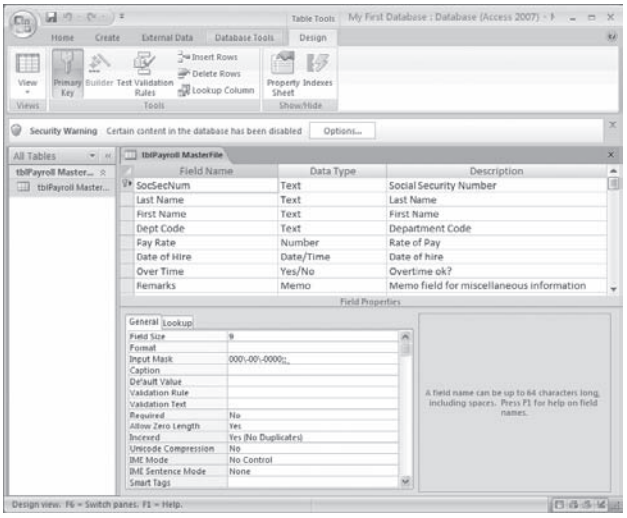


FIGURE 4-14 Payroll Master File table displaying field name, data type, description, and field properties for SocSecNum.

Field Name. Field names are the names you assign to the data fields in your record. As illustrated in Figure 4-14, you can embed blanks in field names and capitalize selected letters in names as desired. Two general rules to follow when naming data fields are (1) use mnemonic names (that help you remember their use such as ''Zip code''), and (2) do not use long names (which are cumbersome to use).

Although it isn't obvious from Figure 4-14, you can use the same field name in each of two tables—the field names in tables are completely independent of one another. In fact, using the same field names for the same data—for example, ''VendorNumber''—in both a Vendor table and a Vendor Invoices table often makes sense because this makes it easy to identify the data field (foreign key) that can link the tables together. We'll look at this shortly.

Data Type. For each data field you create in a table, you must also specify a **data type**. This tells Access how to store the data—for example, as text, a number, or a date. Several examples of such data are, ''text'' data types for an employee's First Name and Last Name, a ''currency'' data type for the employee's pay rate, a ''date'' data type for the employee's date of hire, a ''Yes/No'' data type for the employee's qualifications to earn overtime pay, and a ''Memo'' data type (that stores variable-length text) for the Remarks data field.

Each data field you specify in a table also includes a set of **field properties**, whose values show in the lower portion of the screen in Figure 4-14. These include such settings as ''field size'' (e.g., a length of 9 bytes), ''format'' (e.g., a number with a percent sign), and ''input mask'' (e.g., a template for entering a phone number). Figure 4-14 shows the field properties for the SocSecNum field in our table. Note the Input Mask entry, which you can select from a drop down set of items if you click on this property. You might also be curious why we defined this as a ''text'' field rather than a ''number'' field. The reason is because this data value is not really a number that we will mathematically manipulate, but rather a code. Thus, we create it as a text field and limit its Field Size property to 9 characters (see the bottom portion of Figure 4-14).

Finally, if you use a ''number'' data type, you must also select the type of number you wish to use—for example, Integer, Long Integer, Single (a small decimal value), or Double (a large decimal value). These choices are important when using numeric data fields to link tables together—the field types must match exactly for the join to work.

Description. The last item that you can create for each data field in a table is its description. This is an optional field that you can ignore when defining record structures. However, as you can see from the figure, data field descriptors help document the table itself, and can also describe exception conditions or contain special notes.

Identifying a Primary Key. Recall that a primary key is the data field in each record that uniquely identifies the record. After you have defined the data fields in your table, you can also designate a primary key. This is optional but usually a good idea. For our payroll file example, we will use the employee's Social Security number (SocSecNum) for the primary key. One way to designate this field as the primary key is to click on the name of this field and then select ''Primary Key'' icon () from banner at the top of the screen. An alternate way is to right click on the field with your mouse and select ''primary key'' from the set of choices in the drop down list. The end result in either case will be the same—a little key icon appearing in the first column opposite the data field you selected, as illustrated in Figure 4-14.

Finally, some tables such as join tables do not have an obvious primary key. In such instances, you can ask Access to assign an artificial one by creating a Transaction Number or similar data field name and use an *AutoNumber* data type for it (see the first column in Figure 4-9). In so doing, Access will automatically assign sequential numbers to each record you create, which can also act as a primary key.

Saving a Table. Because you named your database when setting it up, it already has a name. However, if you look at the left of your screen (Figure 4-13) you will see that the table is still named Table1: Table. If you attempt to close your table at this point, Access will prompt you for a name. You can of course use the default name ''Table1,'' but it is better to create your own name for it—for example, ''Payroll Master File.'' You should also include the conventional *tbl* prefix in any name you create for a table. Thus, for example, we used the name ''tblPayroll Master File'' for our table name in Figure 4-14.

Creating Records

After specifying the names, data types, sizes, descriptions, and perhaps primary key for the data fields in your table, you can create individual records for it. To do so, you must switch to ''datasheet'' (or run) view. An easy way to do this is to close the design view of this table and then select the ''Datasheet'' view from the View menu in the upper-left portion of the Access screen in Figure 4-14.

After making these choices, you should see a screen similar to Figure 4-15. This is a table in *datasheet view*, and you are now free to input the data for individual records. Begin by entering data in the row with the asterisk (*) and use the tab key to transition from data field to data field. Every time you complete the data entry for a new record, Access will save the record in the appropriate table automatically.

If you make a mistake while entering data, you can use your backspace key or delete key to correct it just as you would when correcting text in a word processor. Also, if you wish, you can delete an entire record by clicking on the first column to select an entire row (record) and then hitting the delete key. Because Access saves changes immediately, it

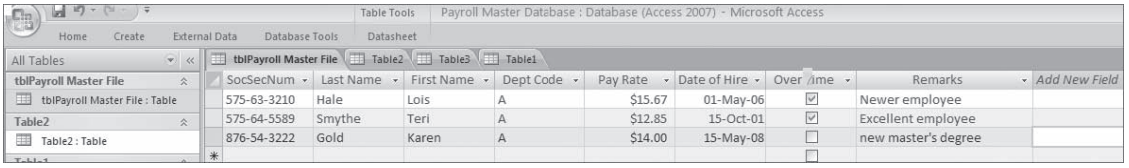


FIGURE 4-15 Dataview sheet for Payroll Master File table.

will first remind you (via a small dialog box) that such a change will be permanent. If you indicate that this is your intent, Access will proceed to delete the record.

Creating Database Relationships

Lastly, it is important to know how to create relationships between database tables. As you've seen from earlier discussions, these relationships link tables together. They also enable users to create multi-table reports, such as the one in Figure 4-3. To illustrate how to create relationships in Access, assume that you have created a department table with records similar to the one in Figure 4-2. Figure 4-16 illustrates the record structure for this table, which you name ''tblDepartments.'' The department code is the primary key for this table.

You now have two tables—''tblDepartments'' and ''tblPayroll Master File.'' They are related in a one-to-many relationship because each department has many employees, but each employee belongs to only one department. The department code is common to both sets of records, although its name differs slightly from one table to another. (We purposely used different names to demonstrate the fact that the names do not have to match exactly to link tables.) This field will act as the foreign key in the Payroll Master File table. To create a relationship between the two tables, follow these steps:

Step 1: Select Tables. First, select the choices Relationships/Show Table from the main menu in Figure 4-13. From the tables listed on the left of your screen, right click on the table you wish to link (tblPayroll Master File: Table) and drag it into the Relationships

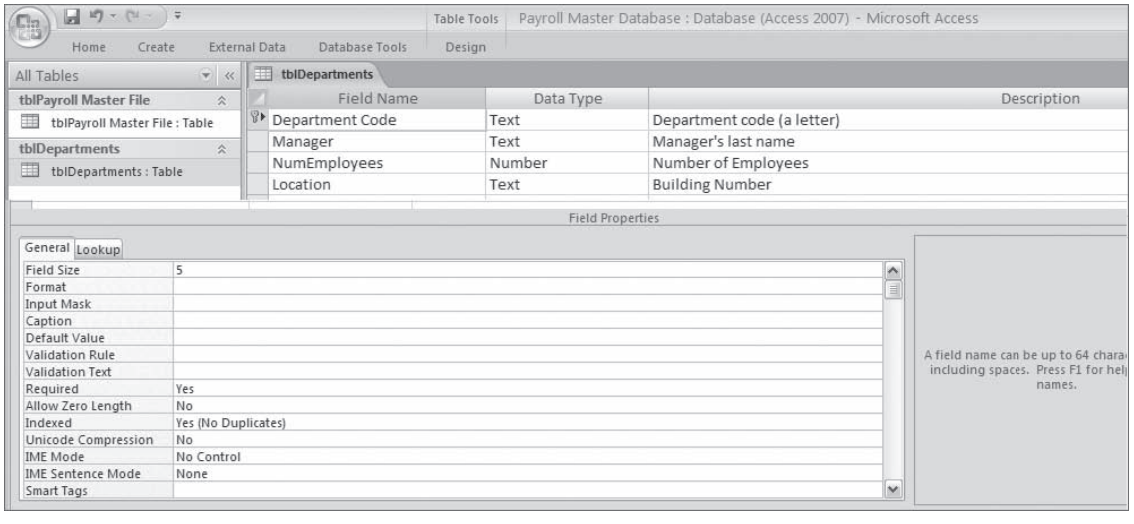


FIGURE 4-16 Departments table with properties for the department code.

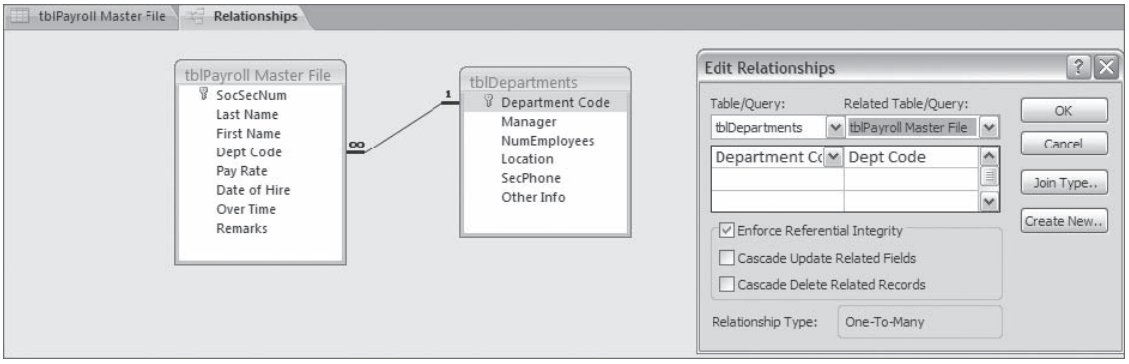


FIGURE 4-17 Linking tables and enforcing referential integrity of table relationships.

window and release the mouse. Now do the same with the tblDepartments: Table. As a result, you should see boxes for the two tables in the Relationships window of Figure 4-17, but there will not be a line drawn between the two tables. That's our next task.

Step 2: Link the Tables. To link the two tables together, drag and drop the department code name from either table to the similar name in the other table. When you do, you should also see the Edit Relationships dialog window of Figure 4-17. This window enables you to enforce **referential integrity**. Check this box. In the context of this example, referential integrity is a control that prohibits users from creating employee records with references to non-existent departments. (It does not affect your ability to create a department with no employees, however.)

If you follow these steps successfully, you should end up with a Relationships window with linked tables as shown in Figure 4-17. What you've done is link the tables together, using the department code as a foreign key. One dramatic way to see this linkage is to open the Departments table in run view (Figure 4-18). Note that there are now plus marks to the left of each department, indicating linked records. If you click on one of these plus marks with your mouse, you'll be able to see them, as illustrated for department A in Figure 4-18. Although it isn't obvious, the relationship you've created for your two tables will also enable you to create the multi-table report illustrated in Figure 4-3. We'll explain how to do that in Chapter 6.

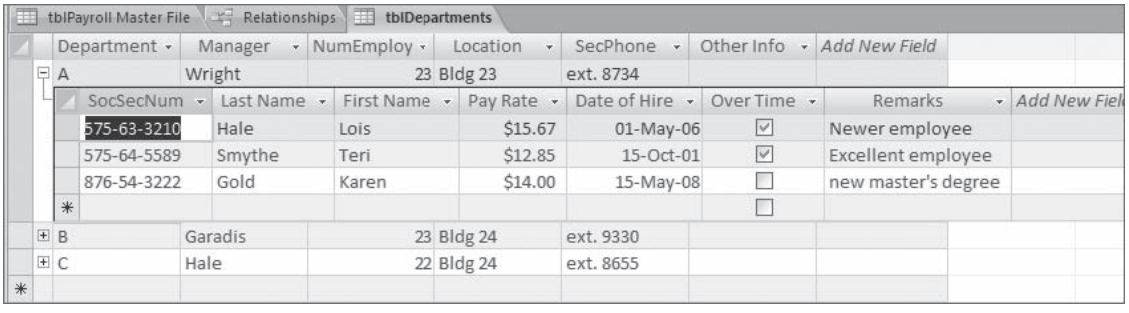


FIGURE 4-18 Showing subordinate data for multi-table relationships.

Guidelines for Creating Database Tables and Records

The preceding discussions both described how to design databases and how to create individual tables and records *within* a database. There are many things that can go wrong when performing these tasks. Here are some guidelines to help you avoid them.

1. Design first; create tables and records last. Some people don't have time to do things right—only time to do things over. Don't be one of them. A careful definition of database entities and their relationships can avoid many problems later.

2. Name tables systematically and use conventional *tbl* prefixes. Even small databases contain many tables, queries, forms, and reports. Using conventional prefixes such as ''tbl'' for tables and ''qry'' for queries enables database designers to distinguish among them. You may also find it useful to name related tables systematically—e.g., use names like ''tblCustomer_MasterFile'' or ''tblCustomer_Returns'' for different types of customer files.

3. Use mnemonic names for data fields. Each data field within a record must have a name, and mnemonic names help you remember what each field means. For example, the name ''State'' is better than ''Address Line 3'' to represent the data field for the customer's state. Similarly, the names ''State Abbreviation'' or ''State Code'' may even be better if you allocate just 2 digits for this field.

4. Assign correct data types to data fields. If you plan to manipulate a data field mathematically, you must define this field as a number—not a text field. Alternately, you should use text data types for such fields as Social Security, credit card, or phone numbers. These numbers are really codes that are too long to store as numbers, but ones that Access can store easily as text values.

5. Data fields that link tables together must be the same data type. If you use the data fields from separate tables to link two tables together, these fields *must* be of the exact same data type. Thus, you cannot link tables together if the foreign key in one table is a text field and the other is a date field. As noted earlier, when using ''number'' data fields, the *type* of number must also match—e.g., each data field must be a Long integer. Violating this rule is one of the most common errors novices make when creating database tables and relationships in Access.

6. Limit the size of data fields to reasonable lengths. Access assigns a default size of ''255'' characters to text fields. If, for example, you designate a state code of only two digits, you should change the default size to two digits. This will limit users to entering no more than two digits. A similar guideline applies to Social Security numbers, telephone numbers, product numbers, and similar values of predetermined, fixed length.

7. Use input masks. An **input mask** is a template that outlines the expected values for a data field. An example of a phone number input mask is (999) 000-0000, which limits the values in a phone number field to 10 numeric digits. Input masks help ensure accurate data input and help reduce mistakes.

AIS AT WORK
Retailers Now Save Questions as well as Answers to Improve Customer Service

Retailers throughout the world know that fast responses to customer questions help them provide better customer service—a hallmark of profitable retailing. For some time, therefore, large organizations such as department stores and airlines have maintained large banks of computer-enabled agents and more recently, sophisticated websites, to answer customer questions quickly, and, hopefully, sell these same customers goods and services on the spot.

These retailers are now also learning that saving the *questions* these customers ask—for example, using the search engines these organizations provide on their websites—can also help sell merchandise. Thus, several companies such as *Ask Jeeves, Vality Technology*, and *SAS* are developing ''natural language software tools'' that can detect patterns in customer inquiries and alert such users as Dell, E-Trade, Nike, and Williams Sonoma to customer search patterns or difficulties using websites. ''If lots of people are asking questions on something and they're not finding information, the search engine will tell us'' says Joan Broughton, director of web publishing at Office Depot.

One retailer that is benefiting from such analyses is Amazon.com. By examining what products a website user requests, the e-commerce retailer's website can match that user to a specific ''customer profile'' and therefore suggest similar products that ''others like you have bought.'' Similarly, Nordstroms (a department store chain) now uses website monitoring software from DigiMine to analyze customer ''clickstream data'' and detect patterns. The company was surprised to learn that one of the top-ten search phrases entered by customers was ''Kate Spade,'' a shoe and handbag maker. The company responded to this discovery by redirecting these customers to offline phone personnel to provide more personal service and sell these products.

Etown.com sells electronic products on the web. The company's *Ask Ida* software asks consumers questions to determine desired features and price-feature tradeoffs. In one analysis, the company learned that buyers of upgraded, feature-rich HD-TVs preferred smaller TV screens to save money. Finally, when Office Depot web designers examined web customer inquiries, they found that many asked about ''next-day delivery''—information that was already on their website, but not easily found. The discovery helped this company redesign its website.

''You get in one question an entire snapshot of what's going on in that person's mind,'' says Michael Callahan, director of advance development at *Ask Jeeves*. And the better a retailer understands its customers, the better it can make a sale.

Source: L. Scott Tillett, ''A 24-Hour Focus Group—Sites Dig Into Search Queries to Learn Customer Preferences'' *Internetweek* (April 10, 2000).

SUMMARY

- Almost every AIS uses databases to store accounting data. The hierarchy of data in such databases is ''bit, character, data field, record, file, and database.''
- Primary, secondary, and foreign record keys enable database systems to identify database records uniquely as well as link records to one another.

- Large, multiuser accounting databases pose additional design concerns. These include the administration and supervision of database development and maintenance, the need for documentation, the importance of data integrity, data processing accuracy and data completeness, database security and backup, and the usefulness of concurrency controls to safeguard data when two users wish to access the same record.
- Databases must be designed carefully. The REA model is a methodology that encourages designers to think of database components in terms of resources, events, and agents.
- Using E-R diagrams, the REA model graphically depicts the entities involved in a database application and the types of relationships between them. The ultimate goal is to determine what to store in sets of records, and how to organize these records efficiently.
- Three database structures are hierarchical, network, and relational. The relational model is most commonly used today.
- Microsoft Access is a popular database management system that small businesses can use to create complete accounting systems. The final section of the chapter illustrated the techniques you can use to create database tables, records, and relationships with this software.

KEY TERMS YOU SHOULD KNOW

agent (REA model)
business events (REA model)
cardinalities
child record
concurrency controls
data dictionary
data field
data hierarchy
data integrity controls
data modeling
data redundancy
data type (Access data field)
database
database administrator
database management system
database transaction
economic events (REA model)
entity (REA model)
entity-relationship (E-R) diagram
foreign key
hierarchical data structure
input mask
master file
metadata
network data structure
parent record
primary record key
REA model
record
record structure
referential integrity
relational data structure
relationship table
resources (REA model)
secondary record key
sibling record
table
transaction controls
transaction file
view controls

TEST YOURSELF

Q4-1. Which of these does *not* characterize a typical database?

a. Large number of records

b. Irreplaceable data

c. High need for accuracy

d. Simple systems

Q4-2. Which of these is *not* part of the ''data hierarchy'' (within the context of databases)?

a. Record **b**. Bit **c**. Character **d**. Data type

Q4-3. Which of these would *not* be a good primary key for a file of employee records?

a. Social security number

b. Last name

c. Company employee number

d. All of these would make equally good primary keys

Q4-4. In the REA model, the ''A'' stands for:

a. Agents **b**. Additions **c**. Accounts **d**. Associations

Q4-5. In the REA model, which of these would *not* be classified as an event?

a. Cash sale **b**. Credit sale

c. Hiring a new chief executive **d**. Date of the office picnic

Q4-6. Which of these is *not* a cardinality between two database entities?

a. One-to-one **b**. None-to-none **c**. One-to-many **d**. Many-to-many

Q4-7. E-R diagrams use all the following symbols *except:*

a. Ovals **b**. Rectangles **c**. Circles **d**. Diamonds

Q4-8. A parent-child relationship between two records is characteristic of:

a. Pyramid databases **b**. Network databases

c. Hierarchical databases **d**. Family databases

Q4-9. To link the records in a many-to-many relationship within a relational database:

a. You must create an intermediate ''relationships'' table

b. You must instead use a network database

c. You must use foreign keys and a spreadsheet system

d. You cannot link records together under these circumstances

Q4-10. Within the context of databases, the term ''concurrency'' refers to the possibility that:

a. A customer of one store might also be a customer of another store

b. Two database users might want to access the same record at the same time

c. A credit entry for a customer requires a debit entry for a matching account

d. None of these

DISCUSSION QUESTIONS

4-1. Why is the storage of accounting data important to an accounting information system? Describe some important concerns, and explain why each one is important.

4-2. What is the hierarchy of data in databases? Provide an example for a particular accounting application.

4-3. Describe some generic types of record keys in typical accounting databases. Are such keys simple or complicated?

4-4. Name some specific accounting files and a potential primary key for each one.

4-5. Describe each of the following database concerns, and give an example of each: (1) data integrity, (2) transaction accuracy and completeness, (3) concurrency processing, and (4) security.

4-6. What is the REA model of database design? How does REA differ from more traditional accounting views of data collection and storage? Hint: would a traditional accounting database store data about personnel matters?

4-7. What are database cardinalities? Give some examples of such cardinalities for an accounting application other than sales.

4-8. What is an entity-relationship diagram? Describe some symbols used in ER modeling, and explain the function of each one.

4-9. Suppose that a data modeler creates a database that includes a Sales table and a Salesperson table. Would you be likely to need a relationship table to link these two entities? Why or why not?

4-10. Why is it important to store primary key values consistently within different tables of the same database?

4-11. Access has five choices on the Menu bar. One of them is Create. What are the other four?

PROBLEMS

4-12. An internal auditor should have a sound understanding of basic data processing concepts such as data organization and storage in order to adequately evaluate systems and make use of retrieval software.

a. Define the following terms as used in a data processing environment (all are nouns): (1) field, (2) record, (3) file.

b. (1) Define a database. (2) List two advantages and two disadvantages of a database system. (CIA adapted)

4-13. What attributes (database table columns) would you be likely to include in a Cash table? In a Cash Receipts table?

4-14. Describe the meaning of each of the entity-relationship diagrams shown in Figure 4-19.

4-15. Draw entity-relationship diagrams for each of the following:

a. The attributes of a customer in an accounts receivable database include name, address, and charge card number.

b. The attributes of a student in a student database include student number (primary key), name, and class rank.

c. The attributes of an asset in a general ledger database include inventory number (primary key), description, and date of purchase.

d. The relationship between an employee and ''is assigned parking'' is one-to-many.

e. The relationship between an employee and ''completes training program'' is many-to-many.

f. The relationship between ''employee'' and ''health plan'' is many-to-one.

g. A customer can be a cash customer or a credit customer. If the customer is a credit customer, an attribute is his or her credit card number.

h. A patient is either an outpatient or an inpatient. If the patient is an inpatient, he or she is assigned a bed (one-to-one).

i. An investment asset could be cash, a stock, a bond, or a certificate of deposit (CD).

j. An account at a bank could be a checking account, a savings account, or a loan account. Each type of account requires an account or loan number. If it is a loan account, another attribute is the monthly payment amount.

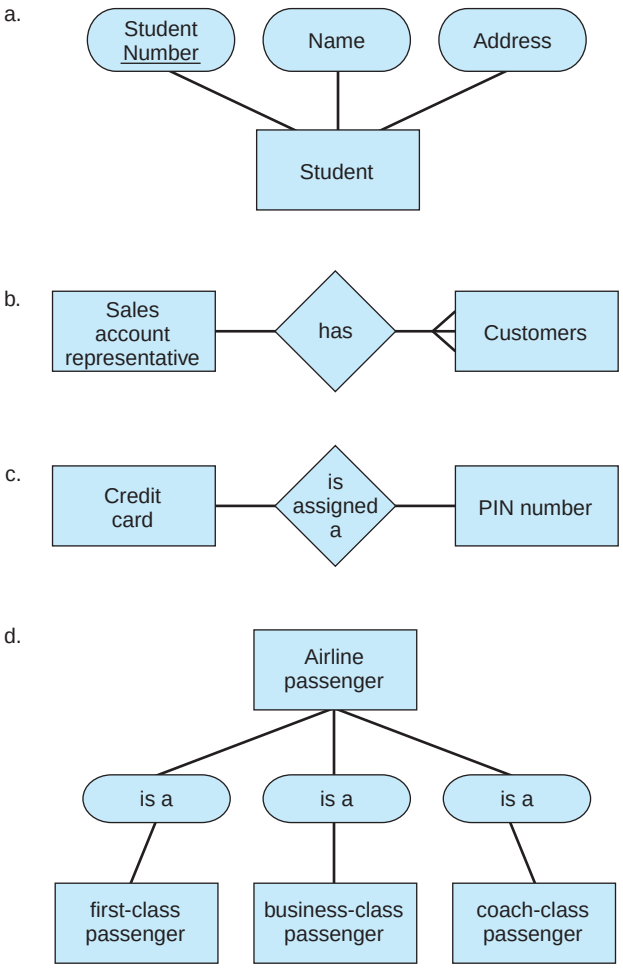


FIGURE 4-19 Entity-relationship diagrams for Problem 4-14.

4-16. Guido Geerts at the University of Delaware has created a website tool for learning more about database cardinalities. You can access the site, *Stevie*, at www.aisvillage.com/stevie. There are two sets of exercises. Use the entries basicpatterns/public as the username/password for one set, and the entries intro/public for the other set.[7]

4-17. This chapter described how to create tables and records in Microsoft Access. What other database management systems are available? Use the Internet to determine their current retail prices.

4-18. Give some examples of field names that you would use for the Customer table in Figure 4-6. What data type would you assign to each field using Microsoft Access, and how large would you make each text field? Would you use one data field or two for the customer name? Why?

4-19. Identify the different data types available for creating data fields in Microsoft Access. Similarly, identify the different types of numbers (e.g., Long integer) you can use if you define a field as a ''number'' data type. (Hint: create a data field in a throwaway database table, assign it a ''number'' data type, and examine the possibilities for the Field Size property as shown in Figure 4-14).

[7]Instructors who are interested in creating their own problems and customized assignments can send an email to Professor Geerts at **geertsg@lerner.udel.edu** to obtain a User ID and Password to enter the system.

4-20. Create a Salesperson table and a Customer Order table using the data in Figure 4-6. Create records for each table using the data provided. Add one more Salesperson record with your own name and an employee number of your choosing. Also add at least one customer order with your number as the salesperson. Finally, create a relationship for the two tables. Create hard-copy documentation of your work.

CASE ANALYSES

4-21. Furry Friends Foundation I (Creating a New Database from Scratch)

The Furry Friends Foundation is a non-profit organization that finds homes for abandoned animals that are suitable for adoption. FFF began operations with a bequest from a wealthy gentleman who lived his life taking care of stray animals and wanted to be sure that such animals were looked after once he was gone. Although the amount the foundation started with was sufficient to set up an office and begin operations, it depends upon continuing donations to run daily operations.

FFF has been keeping its records on 4 × 6 cards. Over the years, the foundation has had requests from contributors for year-end statements that document their donations to the Foundation for tax purposes. (Usually, donations are given with a particular type of animal in mind—for example, ''for dogs.'') Now that the number of contributors exceeds 500, the president has decided to develop a database to handle the foundation's accounting and reporting needs. The following is a sample of some of the records at FFF.

FFF Contributor File

Contributor ID	Last Name	First Name	Street Address	City	State	Zip	Phone Number
13456	Smythe	Jonathan	1845 Backpack Lane	Franktown	NV	55655	501 666-1234
13480	Lawrence	Marie	9190 Teepee Road	Doolittle	NV	54984	501 767-1114
13484	Funky	Robert	5815 Pearly Gate Lane	Happiness	NV	53887	502 995-7654

FFF Donation File

Donation Date	Animal Code	Amount	Contributor ID
September 30, 2009	C	25	13456
September 20, 2009	D	125	13456
October 15, 2009	C	25	13456
October 15, 2009	D	10	13456
October 31, 2009	C	20	13456
October 31, 2009	D	20	13456
November 30, 2009	D	250	13456
November 15, 2009	C	25	13456
December 1, 2009	O	70	13456
December 10, 2009	C	100	13480
September 10, 2009	C	250	13480
October 10, 2009	C	500	13480
November 11, 2009	C	150	13480
December 14, 2009	D	100	13484
September 5, 2009	C	100	13484
October 10, 2009	O	100	13484
November 8, 2009	O	100	13484
December 15, 2009	D	50	13484

FFF Animal Code Table

Contribution for	Code
Dogs	D
Cats	C
Hamster	H
Guinea Pig	G
Rabbit	R
Other	O

Requirements

1. Using Access or a similar relational database, create the tables needed to set up a database for contributors, contributions, and animals.
2. What data field did you use for the primary record key of the FFF contributor table? Why did you use it?
3. Using *Access* or similar software as required by your instructor, add yourself as a contributor.
4. Create relationships for the tables.
5. Document your work by printing hard copies of each table in datasheet view and the relationships report that shows how they are related.

4-22. Carl Beers Enterprises (Using a Relational Database)

Carl Beers Enterprises manufactures and sells specialized electronic components to customers across the country. The tables in Figure 4-20 illustrate some of the records in its accounting databases. Thus, for example, the ''Sales by Inventory Number'' records show detailed sales data for each of the company's inventory items, and the ''Customer Payments'' records indicate customer cash payments, listed by invoice number. Use the information in these tables to answer the following questions.

Requirements

1. The ''Sales by Inventory Number'' records are listed by inventory item number. How is this useful? Why might this information also be useful if it were listed by invoice number instead of inventory number?
2. In the ''Sales by Invoice Number,'' invoice V-3 shows a sales amount of $16,000. What was the name of the customer that made this purchase? What specific inventory items did this customer purchase? How much did this customer pay for each item?
3. Customers can choose among one of three payment options: (1) 5% discount if immediate cash payment, (2) 2% discount off list amount if total invoice paid by the fifteenth day of the month following purchase, or (3) deferred payment plan, using six monthly payments. Which option does J. P. Carpenter appear to be using for invoice V-2?
4. Using just the information provided, what are the quarterly sales amounts for salespeople S-10, S-11, and S-12?
5. Assume that customers C-1 through C-5 began this quarter with net accounts receivable balances of zero. What are their balances now?

4-23. Martin Shoes, Inc. (Planning a Database Using REA and E-R Methodology)

Martin Shoes, Inc. manufactures and distributes orthopedic footwear. To sell its products, the marketing department requires sales personnel to call on the shoe retailers within their assigned geographic territories. Each salesperson has a laptop computer, which he

Sales by Inventory Number

Item Number	Invoice Number	Quantity	Price Each
I-1	V-1	1	2,000
	V-3	1	2,000
	V-6	3	1,575
I-2	V-5	2	3,000
	V-6	10	3,500
I-3	V-3	6	1,000
I-4	V-1	2	600
	V-5	2	300
I-5	V-3	2	4,000
	V-7	3	3,000
I-6	V-2	2	5,000
	V-4	2	5,000
	V-5	2	5,000
	V-7	2	7,000

Sales by Invoice Number

Invoice Number	Amount	Customer Number	Date	Salesperson Number
V-1	7,200	C-1	July 1	S-12
V-2	10,000	C-2	July 12	S-10
V-3	16,000	C-5	July 22	S-10
V-4	10,000	C-2	July 26	S-10
V-5	16,600	C-5	July 31	S-10
V-6	35,000	C-3	Aug 1	S-10
V-7	23,000	C-4	Aug 2	S-11

Sales by Salesperson

Salesperson Number	Quarterly Sales	Commission Rate
S-10	?	.10
S-11	?	.10
S-12	?	.12
S-78	0	.08

Customer Payments

Invoice Number	Remittance Advice Number	Amount
V-1	R-3	7,200
V-2	R-1	1,666
V-2	R-5	1,666
V-3	R-4	16,000
V-4	R-2	10,000
V-5	R-4	16,600

Customer Data

Customer Number	Customer Name	Accounts Receivable Amount	Sales-person
C-1	Dunn, Inc.	?	S-12
C-2	J. P. Carpenter	?	S-10
C-3	Mabadera Corp.	?	S-10
C-4	Ghymn and Sons	?	S-99
C-5	D. Lund, Inc.	?	S-10

FIGURE 4-20 Sample of some of the records in the Beers Enterprise Accounting databases.

or she uses to record sales orders during the day and to send these sales orders to Martin's network nightly for updating the company's sales order file.

Each day, warehouse personnel review the current sales orders in its file, and where possible, pick the goods and ready them for shipment. (Martin ships goods via common carrier, and shipping terms are generally FOB from the shipping point.) When the shipping department completes a shipment, it also notifies the billing department, which then prepares an invoice for the customer. Payment terms vary by customer, but most are ''net 30.'' When the billing department receives a payment, the billing clerk credits the customer's account and records the cash received.

Requirements

1. Identify the resources, events, and agents within Martin's revenue process.
2. Develop an E-R diagram for this process.
3. With a particular DBMS in mind, design the tables for this revenue process. Note that you will need tables for each resource, event, and agent, as well as tables for each many-to-many relationship.

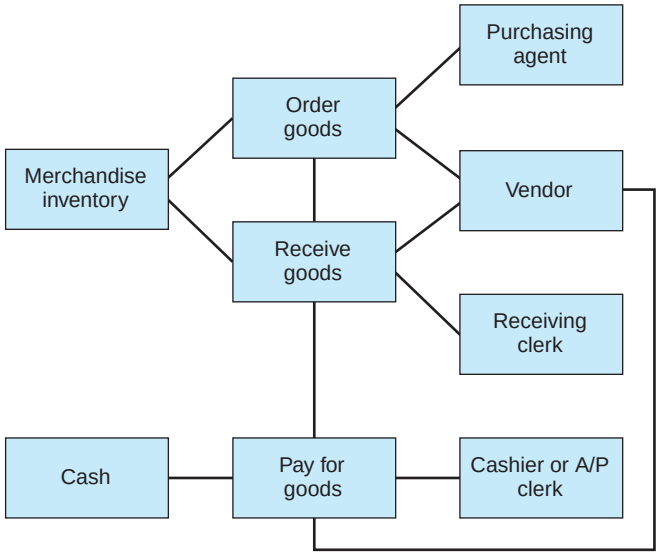


FIGURE 4-21 An E-R diagram for the purchasing system of Souder, Oles, and Franek LLP.

4-24. Souder, Oles, and Franek LLP (Data Modeling with REA)

Souder, Oles, and Franek is an international consulting firm headquartered in Chicago, Illinois. The Entity-Relationship diagram in Figure 4-21 shows a simplified version of the company's process for purchasing and paying for equipment and supplies.

Requirements

1. Insert appropriate pairs of cardinalities for the relationships in the Entity-Relationship model developed with the REA data modeling approach.
2. Describe the database table attributes for this model. You will need a table for each entity, as well as one or more relationship tables. First identify the table name and then indicate the primary key by underlining it. Show any foreign keys by framing them in brackets (e.g., [Vendor#]). Include at least three fields in each table. Below is an example for the Vendor table and the Order Goods table:

   Vendor#, Name, Street Address 1, Street Address 2, City, State, Zip Code, Phone, Email, Fax, Contact, Comments.
   Order#, Date, [Vendor#], [Employee#], Shipping Instructions, Comments.

4-25. BSN Bicycles (Creating a Database from Scratch with Microsoft Access)

Bill Barnes and Tom Freeman opened their BSN bicycle shop in 2005. Not counting Jake—a friend who helps out occasionally at the store—Bill and Tom are the only employees. The shop occupies a small commercial space that was once a restaurant. The former kitchen now stores spare parts and provides space for bicycles repairs, while the former dining

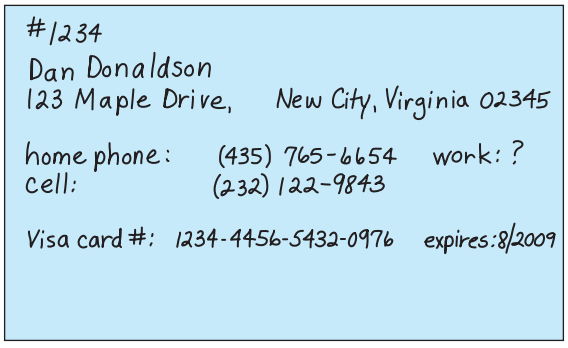
area in the front is now the retail sales area. The ''corporate office'' is just a desk and file cabinet in the back corner of the retail area.

Bill and Tom are more friends and bicycling enthusiasts than businessmen. They've pretty much sunk their life savings into the shop and are anxious that it succeed. In the first year of operations, they worked hard to convert the space into its present condition, which includes an old-timey sign above the door with their name ''BSN Bicycles.''

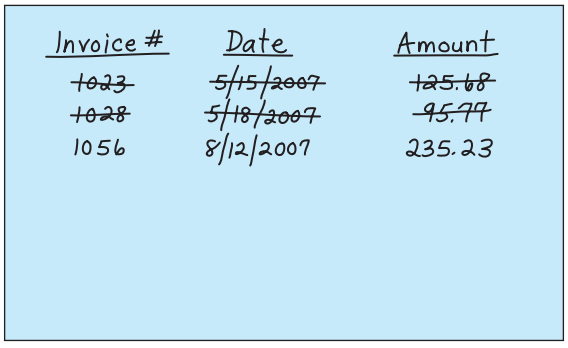
With all the other work that had to be done the first year, marketing efforts have been limited to chatting with friends, distributing flyers at bicycle races and similar sporting events, and placing a few ads in the local newspaper. Similarly, the owners haven't paid much attention to accounting tasks. Who has time with all the other things that had to get done? But at least two things are now clear to the owners: (1) some of their loyal customers prefer to buy items on credit, and (2) all of their suppliers want to be paid on time.

Right now, BSN's ''customer credit system'' is a box of 3x5 cards. Each hand-written card contains customer information on the front and invoice information on the back (Figure 4-22). When a customer pays an invoice, one of the owners simply crosses off the invoice information on the card. The ''supplier accounts system'' is similar, except that the vendor box of 3x5 cards is green whereas the customer box is grey.

Jake is a part-time student at the local community college. He recently completed a course on microcomputer applications that included a segment on Microsoft Access. He doesn't know very much about database theory, but thinks that converting the shop's current ''accounting systems'' to a DBMS might be a good idea. He thinks, for example,



(*a*) The front of a 3x5 BSN customer card.



(*b*) The back of a 3x5 BSN customer card.

FIGURE 4-22 A customer record for the BSN company.

that BSN needs a customer table and a vendor (supplier) table. He also thinks that BSN will need an inventory table to keep track of inventory, but that even more tables might be required. Can you help them?

Requirements

1. Identify the resources, events, and agents for BSN's accounting systems. Draw one or more E-R diagrams that illustrate the relationships between these items.
2. Identify the tables that you would need to create a working database for the company's receivables, payables, and inventory.
3. Using Access or similar software as required by your instructor, create at least three records for each of the tables you identified in part 2. Hints: (1) Use the information on the front of the 3x5 card in Figure 4-22 for the customer record structure. (2) The data fields for the Vendors table should include the vendor ID, vendor name and address information, phone number, fax number, and contact person. (3) The data fields for the Inventory table should include item number, item description, units (e.g., dozen, each, etc.), unit cost, unit retail sales price, and quantity on hand.
4. Create relationships for your various tables.
5. Document your work by printing hard copies of each table in data sheet view and each relationship.

REFERENCES AND RECOMMENDED READINGS

Ames, Anton & Ben Scaff. ''Preparing for a Data Disaster'' *Benefits & Compensation Digest* Vol 45, No. 11 (November 2008), pp. 40–45.

Barbellotto, Gianluca. ''How Good Are Your Standards?'' *Strategic Finance* Vol. 89, No. 12 (June 2008), pp. 67–68.

Britt, Phil. ''Records Management: Beware, Prepare'' *KM World* Vol. 17, No. 10 (November 2008), pp. 12–26.

Carlson, Caron. ''GAO Reports Rampant Federal Data Mining'' *eWeek* Vol. 21, No. 23 (June 7, 2004), p. 31.

Codd, E. F., *The Relational Model for Database Management Version 2* (Reading, MA: Addison-Wesley, 1990).

Curtin, Matthew. ''Database Security: Solve the Right Problem Now for Fewer Headaches Tomorrow'' *Accounting Today* Vol. 21, No. 4 (February 26, 2007), pp. 18–26.

Egana, Nilva & Fiona, Bruinsma. ''Data Dictionaries—What do They Have to do with Public Health?'' *Australian & New Zealand Journal of Public Health* Vol. 32, No. 3 (June 2008), pp. 286–287.

Farrell, Mike, ''Linking Databases, Easing Headaches'' *Multichannel News* Vol. 26, No. 30 (July 25, 2005), pp. 58–59.

Garbellotto, Glanluca. ''How Good are Your (XBRL) Standards?'' *Strategic Finance* Vol. 89, No. 12 (June 2008), pp. 67–68.

Gerard, Gregory J. ''The REA Pattern, Knowledge Structures, and Conceptual Modeling Performance'' *Journal of Information Systems* Vol. 19, No. 2 (Fall 2005), pp. 57–77.

Gruman, Galen. ''Whipping Data Into Shape'' *InfoWorld* Vol. 28, No. 6 (February 6, 2006), pp. 26–32.

Hoberman, Steve ''Data Dictionary, Are You Out There?'' *DM Review* Vol. 17, No. 8 (August 2007), p. 30.

Joseph, George & George Asha. ''Merging Management Accounting with Database Design'' *Management Accounting Quarterly* Vol. 6, No. 2 (Winter, 2005), pp. 34–43.

McCarthy, William E., ''An Entity-Relationship View of Accounting Models,'' *The Accounting Review* (October 1979), pp. 667–686.

McCarthy, William E., ''The REA Accounting Model: A Generalized Framework for Accounting Systems in a Shared Data Environment,'' *Accounting Review* (July 1982), pp. 554–578.

Mutel, Glen. ''Industry Talks to Put Value on Data'' *Precision Marketing* Vol. 18, No. 34 (June 23, 2006), p. 1.

Nemes, Judith. ''Keep on Target as Database Grows'' *B to B* Vol. 93, No. 13 (September 29, 2008), p. 16.

Quain, John R., ''Sizing Down and Speeding Up'' *PC Magazine* Vol. 24, No. 12 (July 1, 2005), p. 22.

Samuels, Janet A. & Robert E. Wood ''Want More Effective and Efficient Data Analysis? Use Access!'' *Strategic Finance* Vol. 89, No. 4 (October 2007), pp. 47–51.

Solomon, Martin D. ''It's all About the Data'' *Information Systems Management* Vol. 22, No. 3 (Summer, 2005), pp. 75–80.

Spinelli, Lisa. ''Data Mining Tools Drilling into Small Business Market'' *Accounting Today* Vol. 19, No. 16 (September 9, 2005), pp. 5 and 23.

Syracuse, Amy. ''Data Integration Helps Pitt-Ohio Express Get to Know its Customers'' *B to B* Vol. 93, No. 8 (June 9, 2008), pp. 39–46.

ANSWERS TO TEST YOURSELF

1. **d** 2. **d** 3. **b** 4. **a** 5. **d** 6. **b** 7. **c** 8. **c** 9. **a** 10. **b**