Part III

Advanced Technologies in Accounting Information

Chapter 9

Database Management Systems

LEARNING OBJECTIVES

After studying this chapter, you should:

- Understand the operational problems inherent in the flat-file approach to data management that gave rise to the database concept.
- Understand the relationships among the defining elements of the database environment.
- Understand the anomalies caused by unnormalized databases and the need for data normalization.
- Be familiar with the stages in database design, including entity identification, data modeling, constructing the physical database, and preparing user views.
- Be familiar with the operational features of distributed databases and recognize the issues that need to be considered in deciding on a particular database configuration.

This chapter deals with the database approach to managing an organization's data resources. The database model is a particular philosophy whose objectives are supported by specific strategies, techniques, hardware, and software that are very different from those associated with flat-file environments.

Chapter 1 drew a distinction between two general data management approaches: the flat-file model and the database model. Because the best way to present the virtues of the database model is by contrast with the flat-file model, the first section of this chapter examines how traditional flat-file problems are resolved under the database approach. Important features of modern relational databases are covered later in the chapter. The second section describes in detail the functions and relationship between four primary elements of the database environment: the users, the database management system (DBMS), the database administrator (DBA), and the physical database. The third section is devoted to an in-depth explanation of the characteristics of the relational model. A number of database design topics are covered, including data modeling, deriving relational tables from entity relationship (ER) diagrams, the creation of user views, and data normalization techniques. The fourth section concludes the chapter with a discussion of distributed database issues. It examines three possible database configurations in a distributed environment: centralized, partitioned, and replicated databases.

Overview of the Flat-File vs. Database Approach

Many so-called legacy systems are characterized by the **flat-file** approach to data management. In this environment, users own their data files. Exclusive ownership of data is a natural consequence of two problems associated with the legacy-system era. The first is a business culture that erects barriers between organizational units that inhibit entity-wide integration of data. The second problem stems from limitations in flat-file management technology that require data files to be structured to the unique needs of the primary user. Thus the same data, used in slightly different ways by different users, may need to be restructured and reproduced in physically different files. Figure 9-1 illustrates this model.

In the figure, the file contents are represented conceptually with letters. Each letter could signify a single **data attribute** (field), a record, or an entire file. Note also that data element B is present in all user files. This is called **data redundancy** and is the cause of three types of data management problems: **data storage, data updating,** and **currency of information**. Each of these, as well as a fourth problem—**task-data dependency,** which is not directly related to data redundancy—will be examined next.

Data Storage

Chapter 1 showed that an efficient information system captures and stores data only once and makes this single source available to all users who need it. This is not possible in the flat-file environment. To meet the private data needs of users, organizations must incur the costs of both multiple collection and multiple storage procedures. Indeed, some commonly used data may be duplicated dozens, hundreds, or even thousands of times, creating excessive storage costs.

Data Updating

Organizations have a great deal of data stored on master files and reference files that require periodic updating to reflect operational and economic changes. For example, a change in a customer's name or address must be reflected in the appropriate master files. This piece of information may be important to several user departments in the organization, such as sales, billing, credit, customer services, sales promotion, and catalog sales. When users maintain separate files, any such change must be made separately for each user. This adds significantly to the cost of data management.

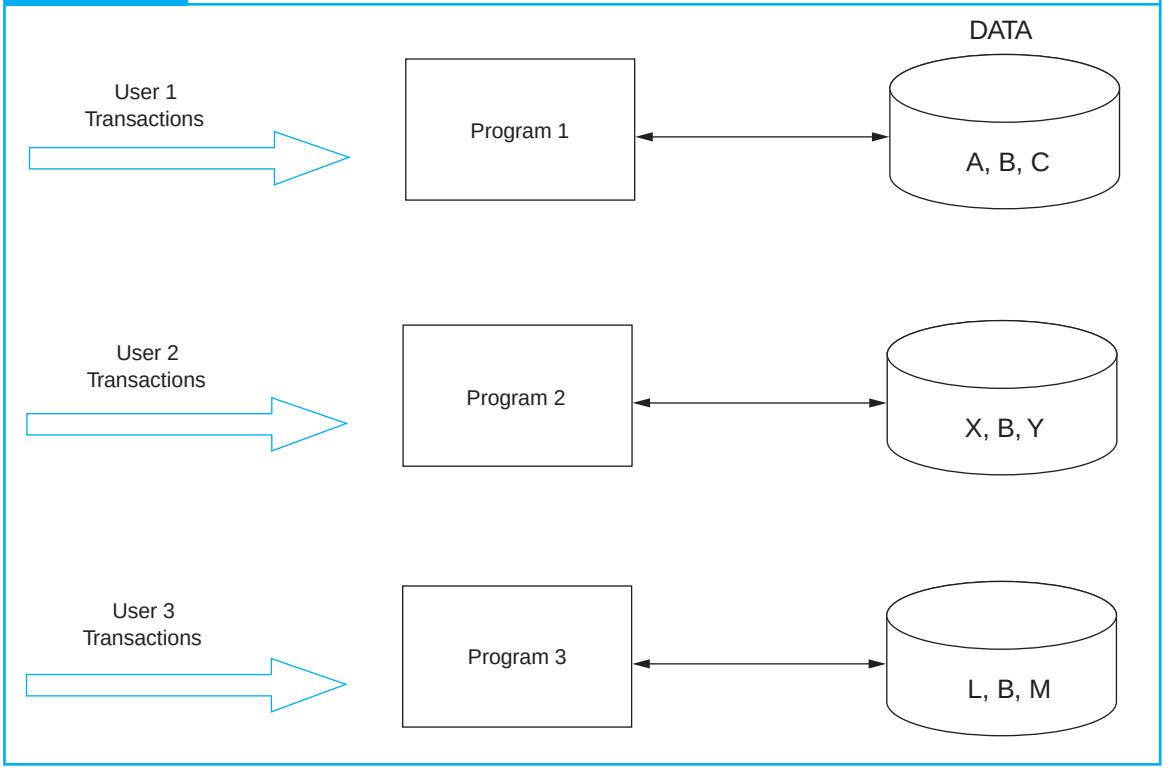
Currency of Information

In contrast to the problem of performing multiple updates is the problem of failing to update the files of all users affected by a change. If update messages are not properly disseminated, then some users may not record the change and will perform their duties and make decisions based on outdated data.

Task-Data Dependency

Another problem with the flat-file approach is the user's inability to obtain additional information as his or her needs change. This problem is called task-data dependency. The user's information set is constrained by the data that he or she possesses and controls. For example, in Figure 9-1, if the information needs of User 1 change to include Data L, User 1's program would not have access to these data. Although Data L exists in the files of another user, keep in mind the culture of this environment. Users do not interact as

FIGURE 9-1 **Flat-File Data Management**



members of a user community. They act independently. As such, User 1 may be unaware of the presence of Data L elsewhere in the organization. In this environment, it is difficult to establish a mechanism for the formal sharing of data. Therefore, Data L would need to be recreated from scratch. This will take time, inhibit User 1's performance, add to data redundancy, and drive data management costs even higher.

The Database Approach

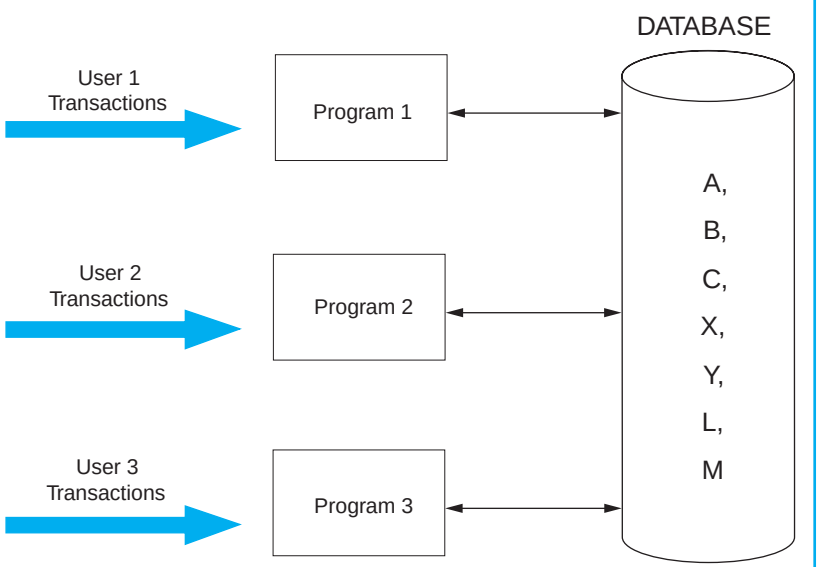
Figure 9-2(a) presents a simple overview of the database approach with the same users and data requirements as in Figure 9-1. The most obvious change from the flat-file model is the pooling of data into a common database that is shared by all the users.

Flat-File Problems Solved

Data sharing (the absence of ownership) is the central concept of the database approach. Let's see how this resolves the problems identified.

- *No data redundancy.* Each data element is stored only once, thereby eliminating data redundancy and reducing storage costs.
- *Single update.* Because each data element exists in only one place, it requires only a single update procedure. This reduces the time and cost of keeping the database current.

FIGURE 9-2(a) **The Database Concept**



- *Current values*. A change any user makes to the database yields current data values for all other users. For example, when User 1 records a customer address change, User 3 has immediate access to this current information.
- *Task-data independence*. Users have access to the full domain of data available to the firm. As users' information needs expand beyond their immediate domain, the new needs can be more easily satisfied than under the flat-file approach. Only the limitations of the data available to the firm (the entire database) and the legitimacy of their need to access it constrains users.

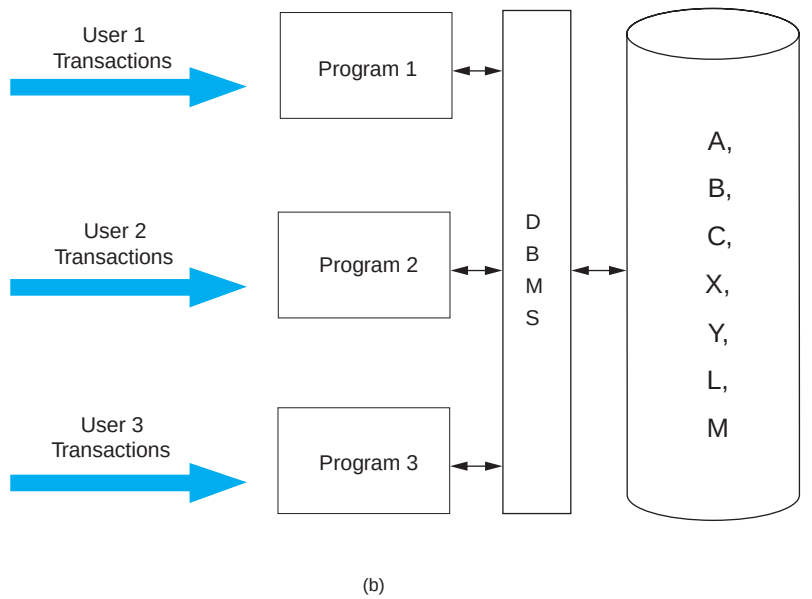
Controlling Access to the Database

The database approach places all the firm's information eggs in one basket. It is essential, therefore, to take very good care of the basket. The example in Figure 9-2(a) has no provision for controlling access to the database. Assume Data X is sensitive, confidential, or secret information that only User 3 is authorized to access. How can the organization prevent others from gaining unauthorized access to it?

The Database Management System

Figure 9-2(b) adds a new element to Figure 9-2(a). Standing between the users' programs and the physical database is the **database management system (DBMS).** The purpose of the DBMS is to provide controlled access to the database. The DBMS is a special software system that is programmed to know which data elements each user is authorized to access. The user's program sends requests for data to the DBMS, which validates and authorizes access to the database in accordance with the user's level of authority. The DBMS will deny requests for data that the user is unauthorized to access. As one might

FIGURE 9-2(b) The Database Concept



(b)

imagine, the organization's criteria, rules, and procedures for assigning user authority are important control issues for accountants to consider.

Three Conceptual Models

Over the years, several different architectures have represented the database approach. Early database models are as different from modern database models as they were from traditional flat files. The most common database approaches used for business information systems are the **hierarchical,** the **network,** and the **relational models.** Because of certain conceptual similarities, the hierarchical and network databases are termed **navigational** or **structured models**. The way that data are organized in these early database systems forces users to navigate between data elements using predefined structured paths. The relational model is far more flexible by allowing users to create new and unique paths through the database to solve a wider range of business problems.

Although their limitations are severe and their ultimate demise is inevitable, hierarchical and network models still exist as legacy systems that support mission-critical functions in some companies. Most modern systems, however, employ relational databases. The main text of the chapter focuses on the relational model. The key features of structured database models are outlined in the chapter appendix.

Elements of the Database Environment

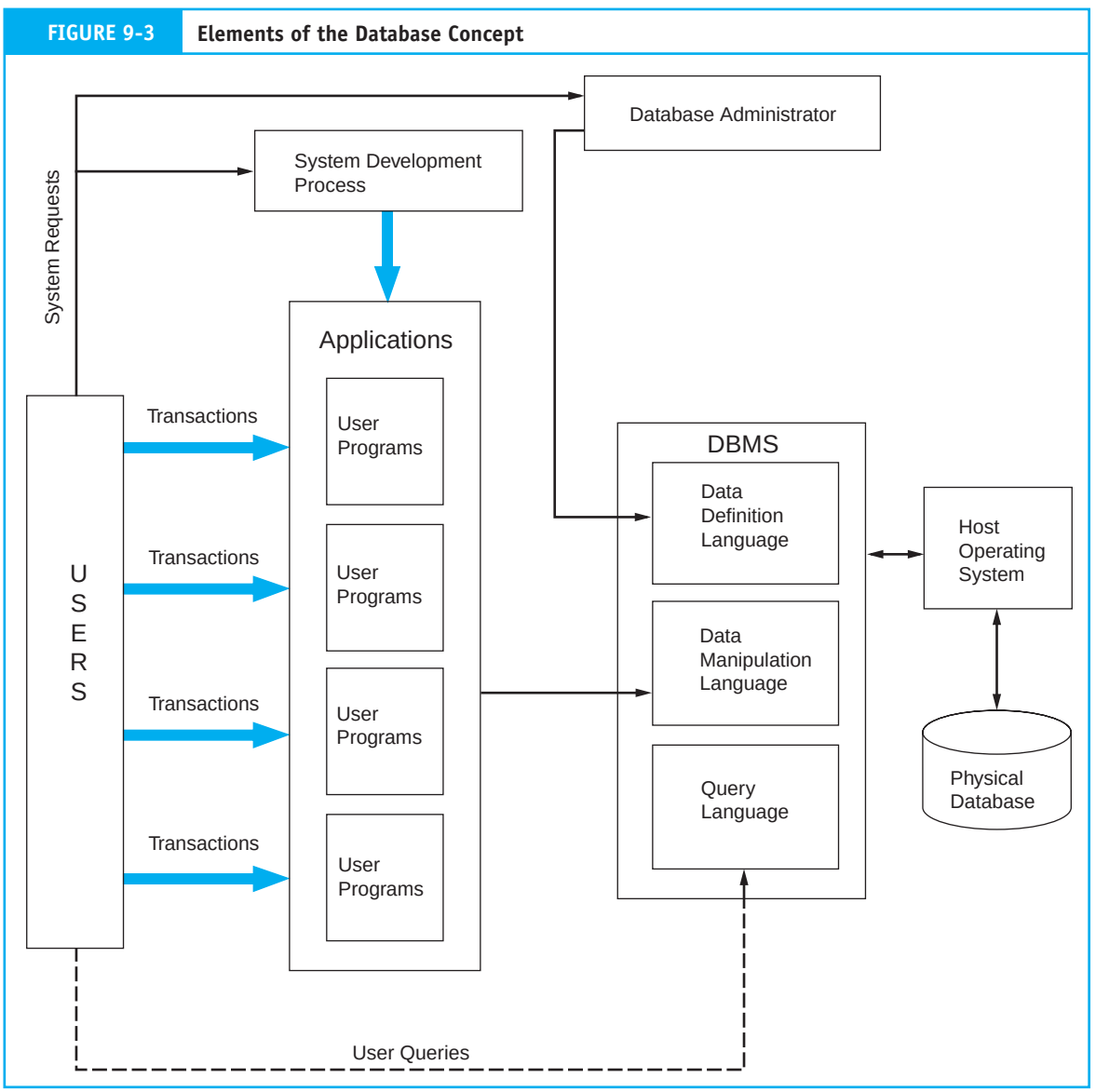
Figure 9-3 presents a breakdown of the database environment into four primary elements: users, the DBMS, the database administrator, and the physical database. In this section we examine each of these elements.

Users

Figure 9-3 shows how **users** access the database in two ways. The first is via user application programs that systems professionals prepare. These programs send data access requests (calls) to the DBMS, which validates the requests and retrieves the data for processing. Under this mode of access, the presence of the DBMS is transparent to the users. Data processing procedures (both batch and real-time) for transactions such as sales, cash receipts, and purchases are essentially the same as they would be in the flat-file environment.

The second method of database access is via direct query, which requires no formal user programs. The DBMS has a built-in query facility that allows authorized users

FIGURE 9-3 **Elements of the Database Concept**



to process data independent of professional programmers. The query facility provides a friendly environment for integrating and retrieving data to produce ad hoc management reports. This feature has been an attractive incentive for users to adopt the database approach.

Database Management System

The second element of the database approach depicted in Figure 9-3 is the database management system. The DBMS provides a controlled environment to assist (or prevent) user access to the database and to efficiently manage the data resource. Each DBMS model accomplishes these objectives differently, but some typical features include

1. *Program development.* The DBMS contains application development software. Both programmers and end users may employ this feature to create applications to access the database.
2. *Backup and recovery.* During processing, the DBMS periodically makes backup copies of the physical database. In the event of a disaster (for example, disk failure, program error, or malicious act) that renders the database unusable, the DBMS can recover an earlier version that is known to be correct. Although some data loss may occur, without the backup and recovery feature, the database would be vulnerable to total destruction.
3. *Database usage reporting.* This feature captures statistics on what data are being used, when they are used, and who uses them. The database administrator (DBA) uses this information to help in assigning user authorization and in maintaining the database. We discuss the role of the DBA later in this section.
4. *Database access.* The most important feature of a DBMS is to permit authorized user access to the database. Figure 9-3 shows the three software modules that facilitate this task. These are the data definition language, data manipulation language, and the query language.

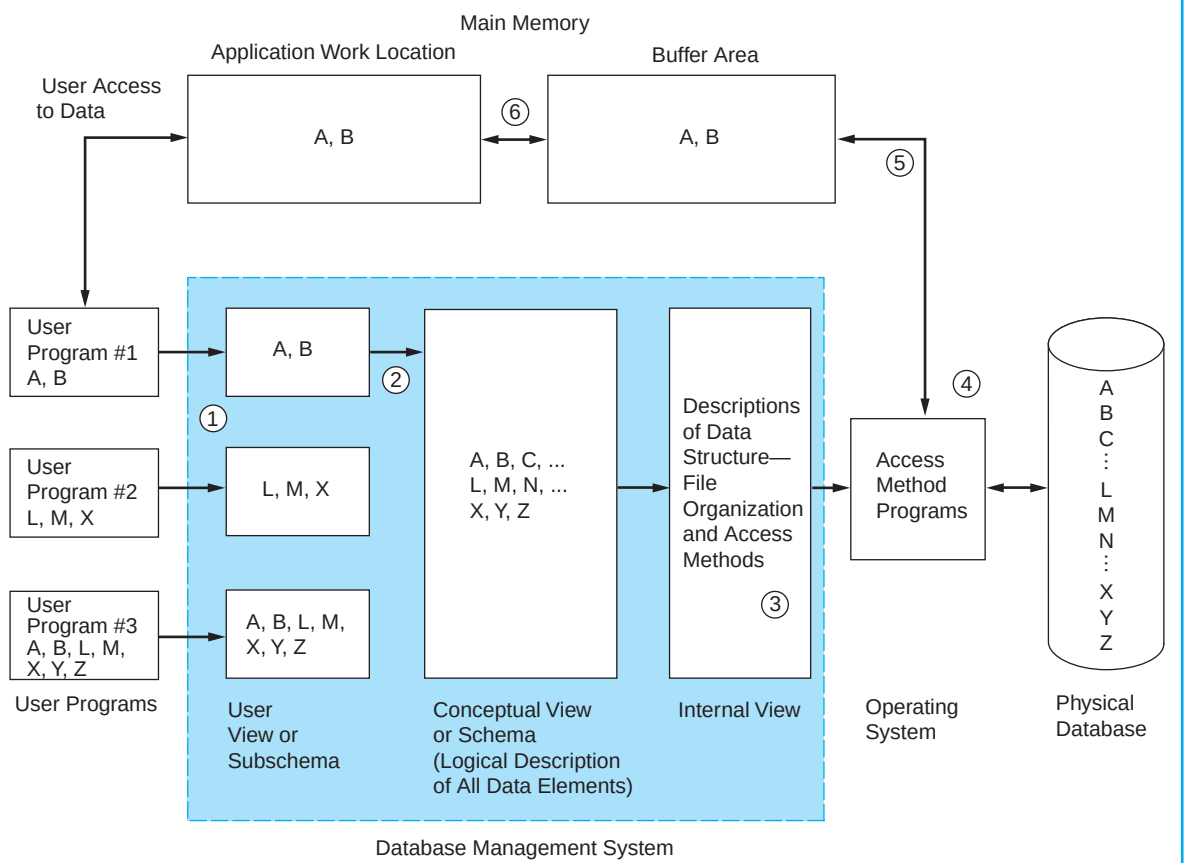
Data Definition Language

Data definition language (DDL) is a programming language used to define the physical database to the DBMS. The definition includes the names and the relationship of all data elements, records, and files that constitute the database. The DDL defines the database on three levels called views: the internal view, the conceptual view (schema), and the user view (subschema). Figure 9-4 shows the relationship between these views.

Internal View. The **internal view** presents the physical arrangement of records in the database. This is the lowest level of representation, which is one step removed from the physical database. The internal view describes the structure of records, the linkages between them, and the physical arrangement and sequence of records in a file. There is only one internal view of the database.

Conceptual View (Schema). The conceptual view or **schema** represents the database logically and abstractly, rather than the way it is physically stored. This view allows users' programs to call for data without knowing or needing to specify how the data are arranged or where the data reside in the physical database. There is only one conceptual view for a database.

FIGURE 9-4 Overview of DBMS Operation



User View (Subschema). The **user view** defines how a particular user sees the portion of the database that he or she is authorized to access. To the user, the user view *is* the database. Unlike the internal and conceptual views, many distinct user views exist. For example, a user in the personnel department may view the database as a collection of employee records and is unaware of the supplier and inventory records seen by the users in the inventory control department.

DBMS Operation. To illustrate the roles of these views, let's look at the typical sequence of events that occurs in accessing data through a DBMS. The following description is hypothetical, and certain technical details are omitted.

1. A user program sends a request (call) for data to the DBMS. The call is written in a special data manipulation language (discussed later) that is embedded in the user program.
2. The DBMS analyzes the request by matching the called data elements against the user view and the conceptual view. If the data request matches, it is authorized and processing proceeds to Step 3. If it does not match the views, access is denied.

3. The DBMS determines the data structure parameters from the internal view and passes them to the operating system, which performs the actual data retrieval. Data structure parameters describe the organization and **access method** (an operating system utility program) for retrieving the requested data.
4. Using the appropriate access method, the operating system interacts with the disk storage device to retrieve the data from the physical database.
5. The operating system then stores the data in a main memory buffer area managed by the DBMS.
6. The DBMS transfers the data to the user's work location in main memory. At this point, the user's program is free to access and manipulate the data.
7. When processing is complete, Steps 4, 5, and 6 are reversed to restore the processed data to the database.

Data Manipulation Language

Data manipulation language (DML) is the proprietary programming language that a particular DBMS uses to retrieve, process, and store data. Entire user programs may be written in the DML or, alternatively, selected DML commands can be inserted into programs that are written in universal languages, such as PL/1, COBOL, and FORTRAN. Inserting DML commands enables legacy application programs, which were originally written for the flat-file environment or earlier types of DBMSs, to be easily converted to work in the current database environment. The use of standard language programs also provides the organization with a degree of independence from the DBMS vendor. If the organization decides to switch its vendors to one that uses a different DML, it will not be necessary to rewrite all the user programs. By replacing the old DML commands with the new commands, user programs can be modified to function in the new environment.

Query Language

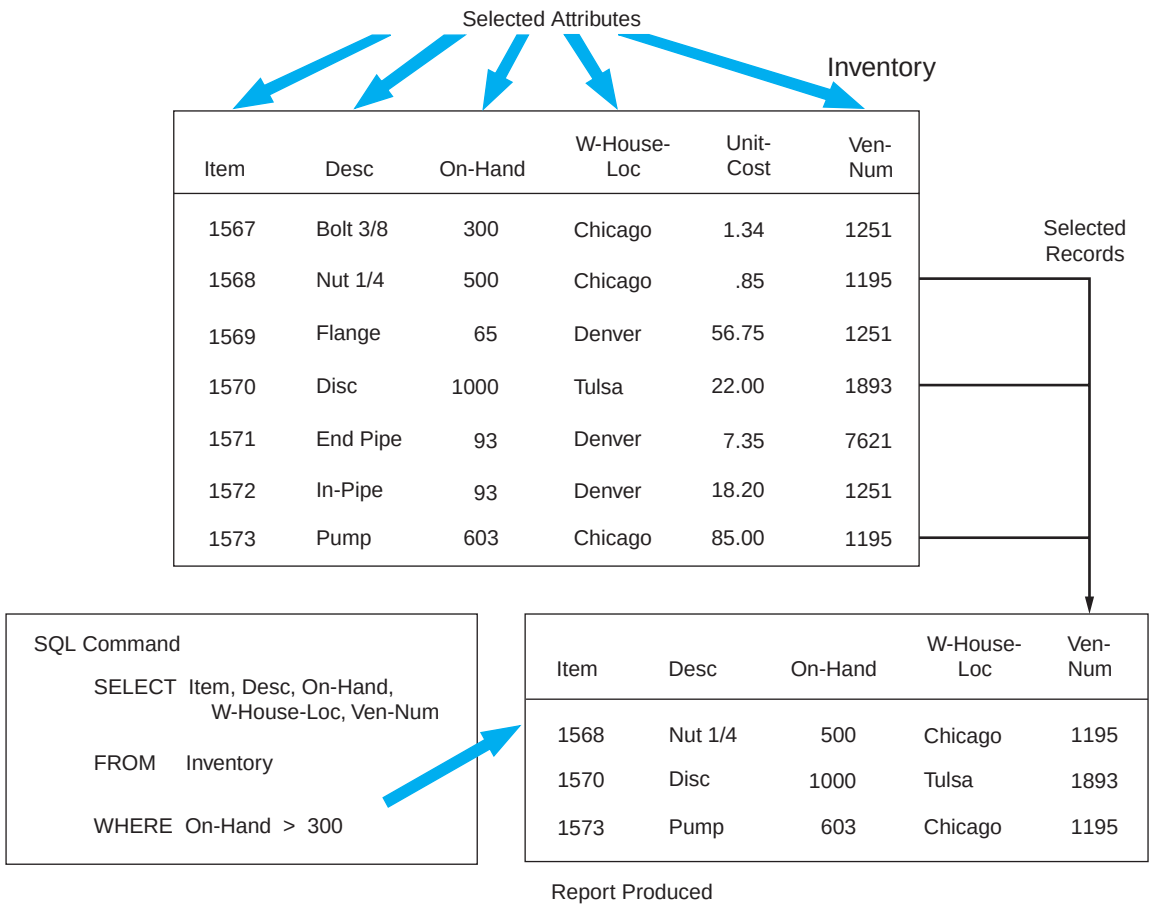
The query capability of the DBMS permits end users and professional programmers to access data in the database directly without the need for conventional programs. IBM's **structured query language** (**SQL,** pronounced *sequel*) has emerged as the standard query language for both mainframe and microcomputer DBMSs. SQL is a fourth-generation, nonprocedural language with many commands that allow users to input, retrieve, and modify data easily. The SELECT command is a powerful tool for retrieving data. The example in Figure 9-5 illustrates the use of the SELECT command to produce a user report from a database called Inventory.

SQL is an efficient data processing tool. Although not a natural English language, SQL requires far less training in computer concepts and fewer programming skills than many languages. In fact, many database query systems require no SQL knowledge at all. Users select data visually by pointing and clicking at the desired attributes. The visual user interface then generates the necessary SQL commands automatically. This feature places ad hoc reporting and data processing capability in the hands of the user/manager. By reducing reliance on professional programmers, managers are better able to deal with problems that pop up.

Database Administrator

Refer to Figure 9-3 and note the administrative position of **database administrator (DBA).** This position does not exist in the flat-file environment. The DBA is responsible for managing the database resource. Multiple users sharing a common database requires organization, coordination, rules, and guidelines to protect the integrity of the database.

FIGURE 9-5 **Example of SELECT Command Used to Query an Inventory Database**



In large organizations the DBA function may consist of an entire department of technical personnel under the database administrator. In smaller organizations someone within the computer services group may assume DBA responsibility. The duties of the DBA fall into the following areas:[1] database planning, database design, database implementation, database operation and maintenance, and database change and growth. Table 9-1 presents a breakdown of specific tasks within these broad areas.

Organizational Interactions of the DBA

Figure 9-6 shows some of the organizational interfaces of the DBA. Of particular importance is the relationship among the DBA, the end users, and the systems professionals of the organization. Refer again to Figure 9-3 during the examination of this relationship.

As information needs arise, users send formal requests for computer applications to the systems professionals (programmers) of the organization. The requests are handled through formal systems development procedures, which produce the programmed

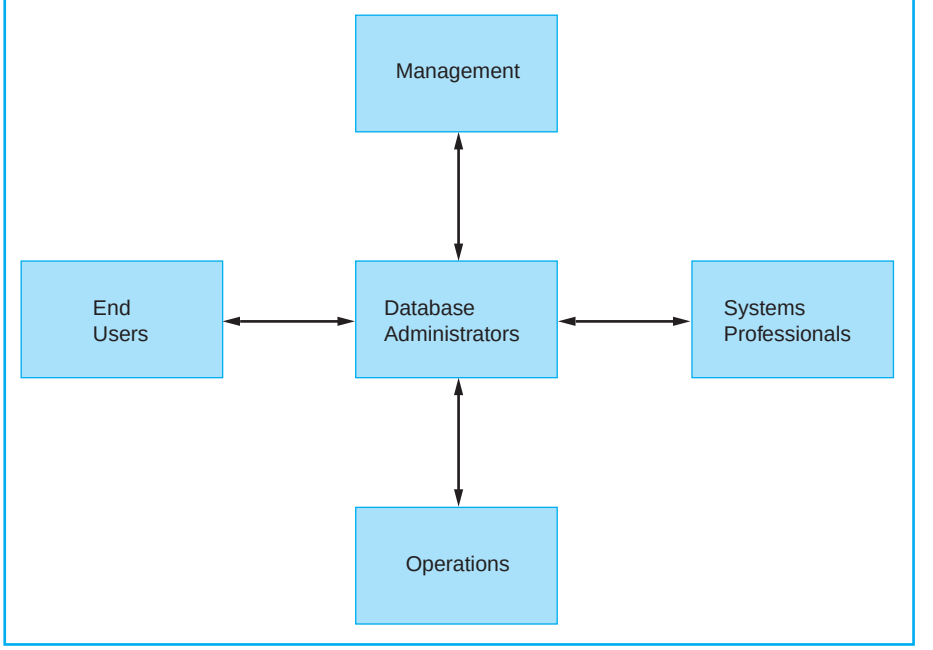
1 Adapted from F. R. McFadden and J. A. Hoffer, *Database Management*, 3rd ed. (Redwood City, CA: Benjamin/Cummings Publishing, 1991): 343.

TABLE 9–1 Functions of the Database Administrator

Database Planning	Implementation
Develop organization's database strategy	Determine access policy
Define database environment	Implement security controls
Define data requirements	Specify test procedures
Develop data dictionary	Establish programming standards
Design	**Operation and Maintenance**
Logical database (schema)	Evaluate database performance
External users' views (subschemas)	Reorganize database as user needs demand
Internal view of database	Review standards and procedures
Database controls	
	Change and Growth
	Plan for change and growth
	Evaluate new technology

applications. Figure 9-3 shows this relationship as the line from the users block to the systems development process block. The user requests also go to the DBA, who evaluates these to determine the user's database needs. Once this is established, the DBA grants the user access authority by programming the user's view (subschema). This relationship is shown as the lines between the user and the DBA and between the DBA and DDL module

FIGURE 9-6 Organizational Interactions of the Database Administrator



in the DBMS. By keeping access authority separate from systems development (application programming), the organization is better able to control and protect the database. Intentional and unintentional attempts at unauthorized access are more likely to be discovered when these two groups work independently. The rationale for this separation of duties is developed in Chapter 15.

The Data Dictionary

Another important function of the DBA is the creation and maintenance of the **data dictionary.** The data dictionary describes every data element in the database. This enables all users (and programmers) to share a common view of the data resource and greatly facilitates the analysis of user needs.

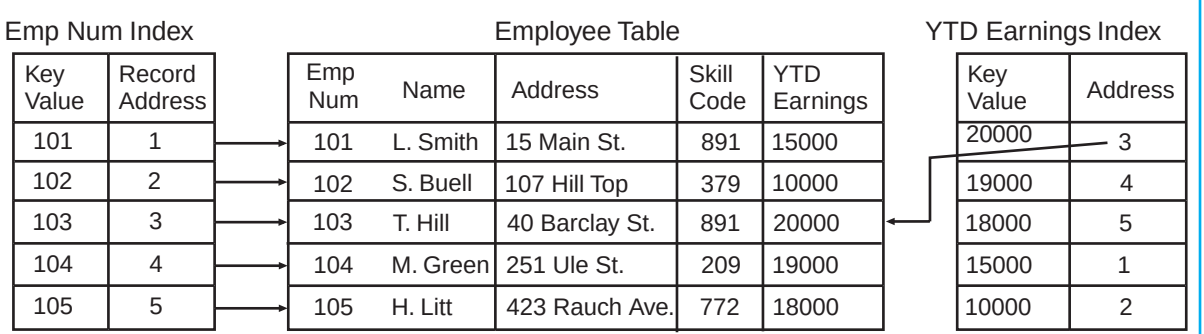
The Physical Database

The fourth major element of the database approach as presented in Figure 9-3 is the **physical database.** This is the lowest level of the database. The physical database consists of magnetic spots on magnetic disks. The other levels of the database (for example, the user view, conceptual view, and internal view) are abstract representations of the physical level.

At the physical level, the database is a collection of records and files. Relational databases are based on the **indexed sequential file** structure. This structure, illustrated in Figure 9-7, uses an index in conjunction with a sequential file organization. It facilitates both direct access to individual records and batch processing of the entire file. Multiple indexes can be used to create a cross-reference, called an **inverted list,** which allows even more flexible access to data. Two indexes are shown in Figure 9-7. One contains the employee number (primary key) for uniquely locating records in the file. The second index contains record addresses arranged by year-to-date earnings. Using this nonunique field as a secondary key permits all employee records to be viewed in ascending or descending order according to earnings. Alternatively, individual records with selected earnings balances can be displayed. Indexes may be created for each attribute in the file, allowing data to be viewed from a multitude of perspectives.

The next section examines the principles that underlie the relational model and the techniques, rules, and procedures for creating relational tables from indexed sequential files. You will also see how tables are linked to other tables to permit complex data representations.

FIGURE 9-7 **Indexed Sequential File**



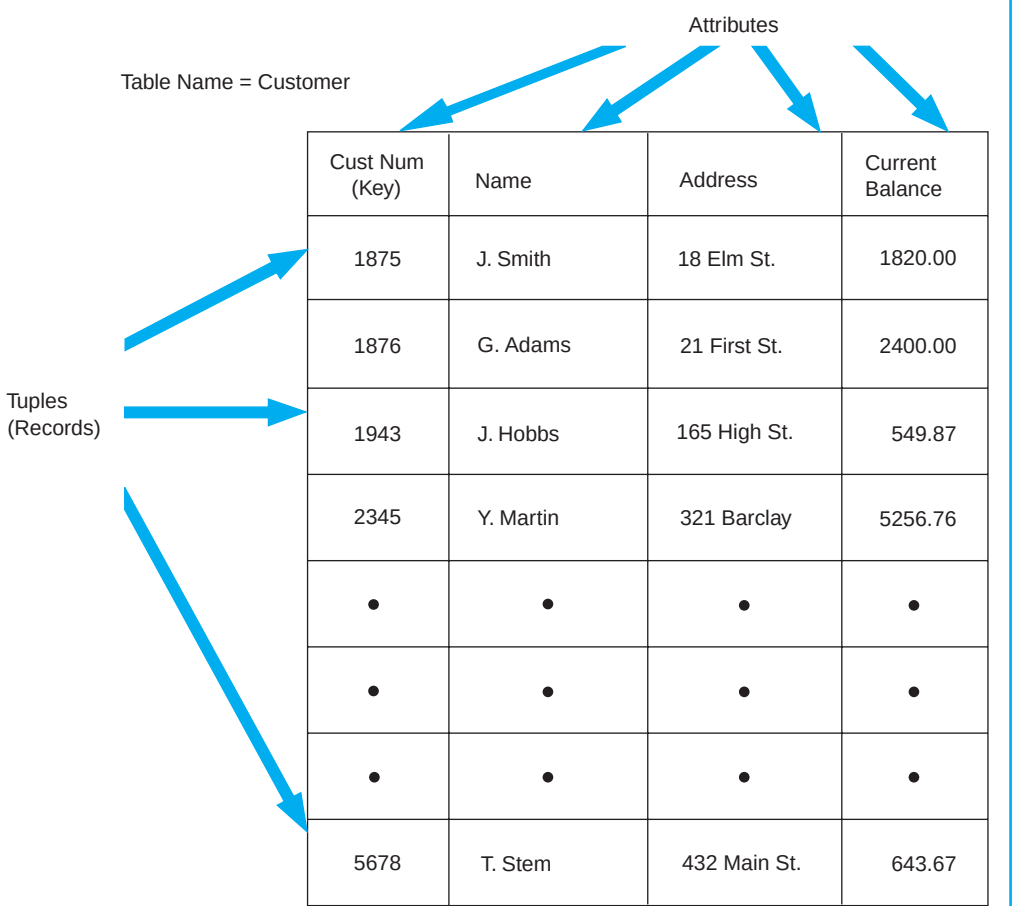
The Relational Database Model

E. F. Codd originally proposed the principles of the relational model in the late 1960s.[2] The formal model has its foundations in relational algebra and set theory, which provide the theoretical basis for most of the data manipulation operations used.

From a purist's point of view, a fully relational system is one that conforms to 12 stringent rules that Codd outlined.[3] As a practical matter, however, not all of Codd's rules are equally important. Some are critical, but some are not. Other theorists have, therefore, proposed less rigid requirements for assessing the relational standing of a system.[4] Accordingly, a system is relational if it:

1. Represents data in the form of two-dimensional tables such as the database table, called Customer, shown in Figure 9-8.
2. Supports the relational algebra functions of restrict, project, and join.

FIGURE 9-8 A Relational Table Called Customer



2 C. J. Date, *An Introduction to Database Systems*, Vol. 1, 4th ed. (Reading, MA: Addison-Wesley, 1986): 99.
3 For a complete discussion of these rules, see McFadden and Hoffer, *Database Management*: 698–703.
4 Date, *An Introduction to Database Systems*: 320–26.

These three algebra functions are examined in the following section.

Restrict: Extracts specified rows from a specified table. This operation, illustrated in Figure 9-9(a), creates a virtual table (one that does not physically exist) that is a subset of the original table.

Project: Extracts specified attributes (columns) from a table to create a virtual table. This is presented in Figure 9-9(b).

Join: Builds a new physical table from two tables consisting of all concatenated pairs of rows, from each table. See Figure 9-9(c).

Although restrict, project, and join is not the complete set of relational functions, it is a useful subset that satisfies most business information needs.

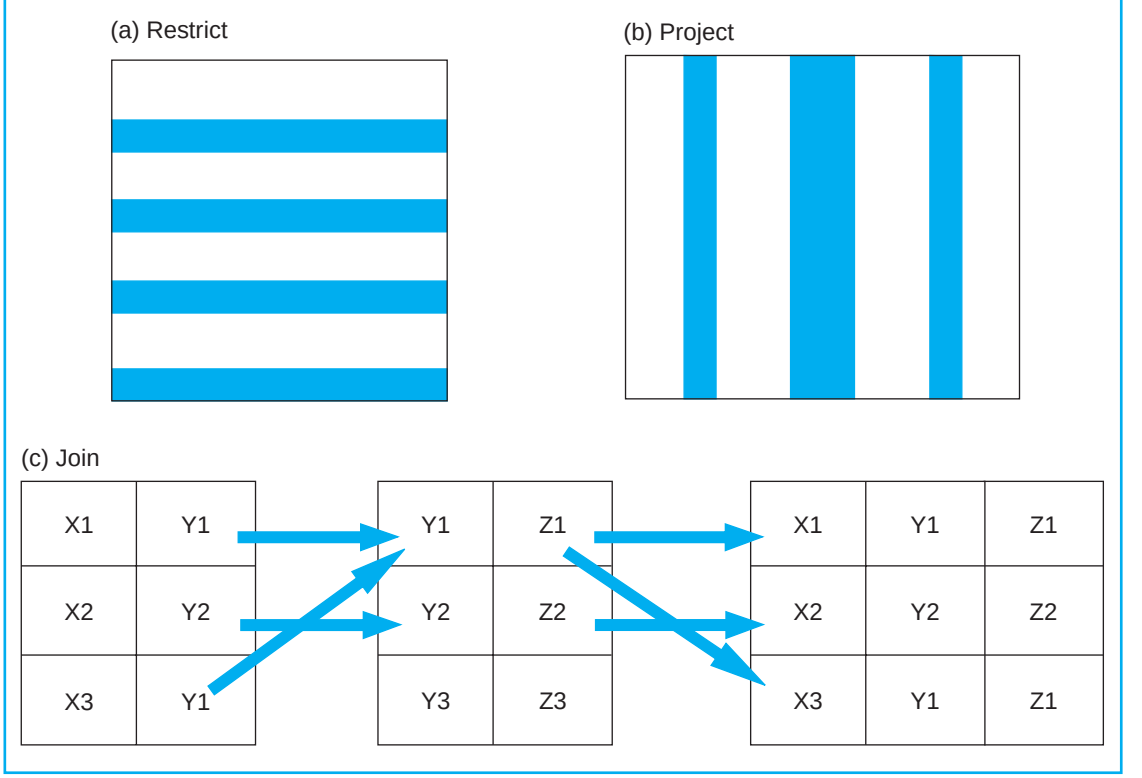
Relational Database Concepts

In this section we review basic concepts, terminology, and techniques common to relational database systems. These building blocks are then used later in the chapter to design a small database from scratch.

Entity, Occurrence, and Attributes

An **entity** is anything about which the organization wishes to capture data. Entities may be physical, such as inventories, customers, or employees. They may also be conceptual,

FIGURE 9-9 **The Relational Algebra Functions Restrict, Project, and Join**



such as sales (to a customer), AR, or AP. Systems designers identify entities and prepare a model of them like the one presented in Figure 9-10. This **data model** is the blueprint for ultimately creating the physical database. The graphical representation used to depict the model is called an **entity relationship (ER) diagram.** As a matter of convention, each entity in a data model is named in the singular noun form, such as Customer rather than Customers. The term **occurrence** is used to describe the number of instances or records that pertain to a specific entity. For example, if an organization has 100 employees, the Employee entity is said to consist of 100 occurrences. **Attributes** are the data elements that define an entity. For example, an Employee entity may be defined by the following partial set of attributes: Name, Address, Job Skill, Years of Service, and Hourly Rate of Pay. Each occurrence in the Employee entity consists of the same types of attributes, but values of each attribute will vary among occurrences. Because attributes are the logical and relevant characteristics of an entity, they are unique to it. In other words, the same attribute should not be used to define two different entities.

Associations and Cardinality

The labeled line connecting two entities in a data model describes the nature of the **association** between them. This association is represented with a verb, such as ships, requests, or receives. **Cardinality** is the degree of association between two entities. Simply stated, cardinality describes the number of possible occurrences in one table that are associated with a single occurrence in a related table. Four basic forms of cardinality are possible: zero or one (0,1), one and only one (1,1), zero or many (0,M), and one or many (1,M). These are combined to represent logical associations between entities. The value of the upper cardinalities at each end of the association line defines the association. For example, a (0,1) cardinality at one end and a (1,M) cardinality at the other is a (1:M) association. Figure 9-11 presents several examples of entity associations, which are discussed next.

Example 1 (1:1). Assume that a company has 1,000 employees but only 100 of them are sales staff. Assume also that each salesperson is assigned a company car, but non sales staff are not. Example 1 in Figure 9-11 shows that for every occurrence (record) in the Employee entity, there is a possibility of zero or one occurrence in the Company Car entity.

FIGURE 9-10 Data Model Using an Entity Relationship (ER) Diagram

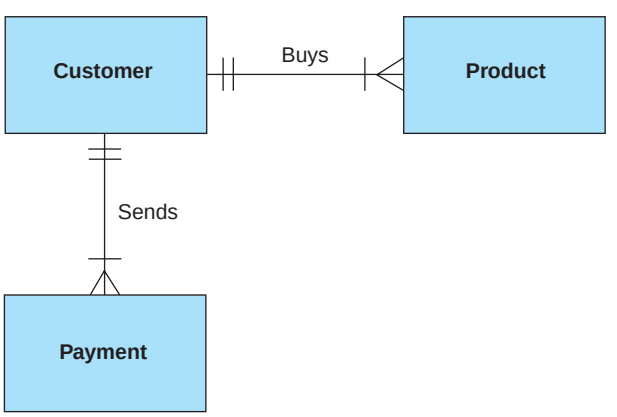
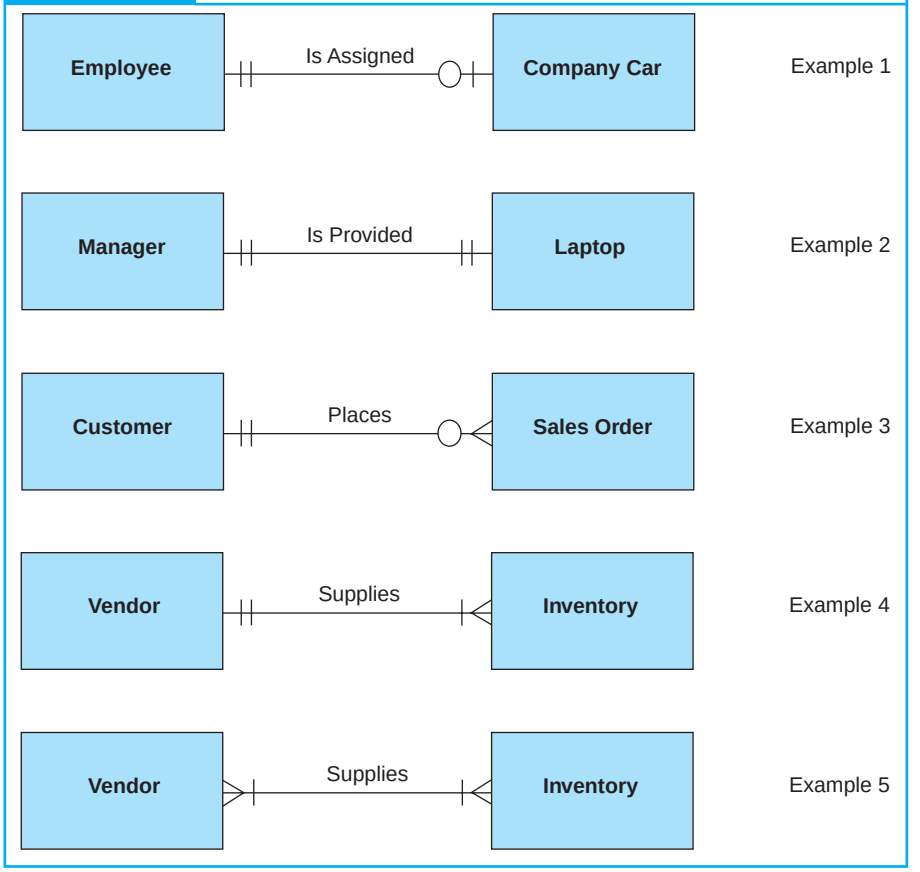


FIGURE 9-11 Examples of Entity Associations



When determining the cardinality values in an entity association, select a single occurrence (record) of one entity and answer the following question: What are the minimum and maximum number of records that may be associated with the single record that has been selected? For example, selecting an Employee entity record and looking toward the Company Car entity, we see two possible values. If the selected Employee record is that of a salesperson, then he or she is assigned one (and only one) company car. This particular Employee record, therefore, is associated with only one record in the Company Car entity. If, however, the selected Employee record is that of a nonsalesperson, then the individual would be assigned no (zero) car. The Employee record in this case is associated with zero Company Car records. Thus, the minimum cardinality is zero, and the maximum cardinality is one. A circle and a short line intersecting the line connecting the two entities depict this degree of cardinality. Notice that from the Employee entity perspective, the cardinality is shown at the Company Car end of the association line. Now select a Company Car record and look back at the Employee entity. Because each company car is assigned to only one employee, both the minimum and maximum number of associated records is one. Two short intersecting lines at the Employee end of the association line signify this cardinality.

Example 2 (1:1). Example 2 illustrates a situation in which each record in one entity is always associated with one (and only one) record in the associated entity. In this case, each company laptop computer is assigned to only one manager, and every manager is assigned only one computer. Two short lines intersecting the connecting line at both ends depict this cardinality.

Example 3 (1:M). Example 3 presents the relationship between Customer and Sales Order entities. Notice that the minimum number of Sales Order records per Customer record is zero and the maximum is many. This is because in any given period (a year or month) to which the Sales Order entity pertains, a particular customer may have purchased nothing (zero Sales Order records) or purchased several times (many records). From the perspective of the Sales Order entity, however, every record is associated with one and only one customer. The crow's foot symbol (which gives this form of notation its name) depicts the many cardinalities at the Sales Order end of the association line.

Example 4 (1:M). Example 4 represents a situation in which each specific item of inventory is supplied by one and only one vendor, and each Vendor supplies one or many different inventory items to the company. Contrast this (1:M) association with Example 5 next.

Example 5 (M:M). To illustrate the many-to-many association, we again use a Vendor and Inventory relationship in Example 5. This time, however, the company has a policy of purchasing the same types of inventory from multiple suppliers. Management may choose to do this to ensure that they get the best prices or avoid becoming dependent on a single supplier. Under such a policy, each Vendor record is associated with one or many Inventory records and each Inventory record is associated with one or many Vendors.

Examples 4 and 5 demonstrate how cardinality reflects the business rules in place within an organization. The database designer must obtain a thorough understanding of how the client-company and specific users conduct business to properly design the data model. If the data model is wrong, the resulting database tables will also be wrong. Examples 4 and 5 are both valid but different options and, as we shall see, require different database designs.

Alternative Cardinality Notations. The cardinality notation technique shown in Figure 9-11 is called crow's foot. An alternative method is to write the cardinality values on each end of the association line connecting the two entities. Some database designers explicitly show both the upper and lower cardinality values. Some choose a shorthand version that notes only the upper cardinality. For homework assignments, your instructor will advise you of the preferred method to use. Refer to Figure 10-7 in the next chapter for examples of the alternative techniques.

The Physical Database Tables

Physical database tables are constructed from the data model with each entity in the model being transformed into a separate physical table. Across the top of each table are attributes forming columns. Intersecting the columns to form the rows of the table are tuples. A tuple, which Codd gave a precise definition when he first introduced it, corresponds approximately to a record in a flat-file system. In accordance with convention, we will use the term *record* or *occurrence* rather than tuple.

Properly designed tables possess the following four characteristics:

1. The value of at least one attribute in each occurrence (row) must be unique. This attribute is the **primary key**. The values of the other (nonkey) attributes in the row need not be unique.
2. All attribute values in any column must be of the same class.
3. Each column in a given table must be uniquely named. However, different tables may contain columns with the same name.
4. Tables must conform to the rules of normalization. This means they must be free from structural dependencies including repeating groups, partial dependencies, and transitive dependencies (see this chapter's appendix for a complete discussion).

Linkages between Relational Tables

Logically related tables need to be physically connected to achieve the associations described in the data model. Using **foreign keys** accomplishes this, as illustrated in Figure 9-12. In this example the foreign keys are embedded in the related table. For instance, the primary key of the Customer table (CUST NUM) is embedded as a foreign key in both the Sales Invoice and Cash Receipts tables. Similarly, the primary key in the Sales Invoice table (INVOICE NUM) is an embedded foreign key in the Line Item table. Note that the Line Item table uses a composite primary key comprised of INVOICE NUM and ITEM NUM. Both fields are needed to identify each record in the table uniquely, but only the invoice number portion of the key provides the logical link to the Sales Invoice table. Foreign keys are not always embedded like those in Figure 9-12. The nature of the association between the related tables determines the method used for assigning foreign keys. These methods are examined later.

With foreign keys in place, a computer program can be written to navigate among the tables of the database and provide users with the data they need to support their day-to-day tasks and decision-making responsibilities. For example, if a user wants all the invoices for Customer 1875, the program will search the Sales Invoice table for records with a foreign key value of 1875. We see from Figure 9-12 that there is only one such occurrence—invoice number 1921. To obtain the line-item details for this invoice, the program searches the Line Item table for records with a foreign key value of 1921, and two records are retrieved.

User Views

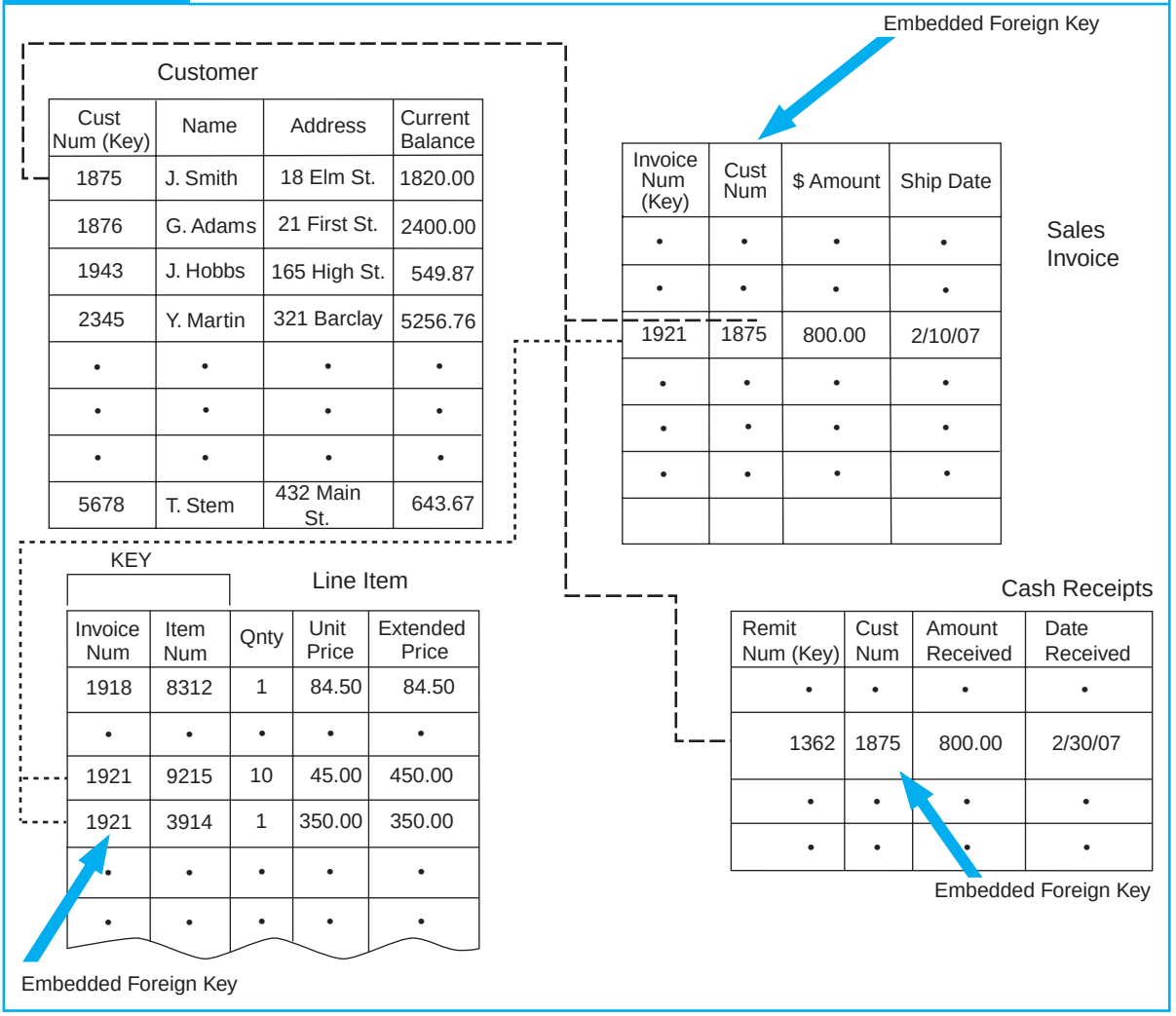
A user view was defined earlier as the set of data that a particular user sees. Examples of user views are computer screens for entering or viewing data, management reports, or source documents such as an invoice. Views may be digital or physical (paper), but in all cases, they derive from underlying database tables. Simple views may be constructed from a single table, while more complex views may require several tables. Furthermore, a single table may contribute data to many different views.

A large organization will have thousands of user views that thousands of individual tables support. The task of identifying all views and translating them into normalized tables is an important responsibility of database designers that has internal control implications. The issues and techniques related to this task are examined next.

Anomalies, Structural Dependencies, and Data Normalization

This section deals with why database tables need to be normalized. In other words, why is it necessary for the organization's database to form an elaborate network of normalized tables linked together like those illustrated in Figure 9-12? Why, instead, can we not

FIGURE 9-12 Linkages between Relational Tables



simply consolidate the views of one user (or several) into a single common table from which all data needs may be met?

Database Anomalies

The answer to the above questions is that improperly normalized tables can cause DBMS processing problems that restrict, or even deny, users access to the information they need. Such tables exhibit negative operational symptoms called **anomalies.** Specifically these are the update anomaly, the insertion anomaly, and the deletion anomaly. One or more of these anomalies will exist in tables that are not normalized or are normalized at a low level, such as **first normal form (1NF)** or **second normal form (2NF).** To be free of anomalies, tables must be normalized to the **third normal form (3NF)** level.

We will demonstrate the negative impact of anomalies with the user view in Figure 9-13. This inventory status report would be used to provide a purchasing agent with information

FIGURE 9-13 **Inventory Status Report**

Ajax Manufacturing Co.
Inventory Status Report

Part Number	Description	Quantity On Hand	Reorder Point	Supplier Number	Name	Address	Telephone
1	Bracket	100	150	22 24 27	Ozment Sup Buell Co. B&R Sup	123 Main St. 2 Broadhead Westgate Mall	555-7895 555-3436 555-7845
2	Gasket	440	450	22 24 28	Ozment Sup Buell Co. Harris Manuf	123 Main St. 2 Broadhead 24 Linden St.	555-7895 555-3436 555-3316
3	Brace	10	10	22 24 28	Ozment Sup Buell Co. Harris Manuf	123 Main St. 2 Broadhead 24 Linden St.	555-7895 555-3436 555-3316
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .

about inventory items to be ordered and the various suppliers (vendors) of those items. If this view were to be produced from a single table, it would look similar to the one presented in Figure 9-14. While such a table could indeed store the data to produce the view, it will exhibit the anomalies mentioned above.

Update Anomaly. The **update anomaly** results from data redundancy in an unnormalized table. To illustrate, notice that Supplier Number 22 provides each of the three inventory items (Part Num 1, 2, and 3) shown in Figure 9-14. The data attributes pertaining to Supplier Number 22 (Name, Address, and Tele Num) are thus repeated in every record of every inventory item that Supplier Number 22 provides. Any change in the

FIGURE 9-14 **Unnormalized Database Table**

Inventory Table

Primary Key — Nonkey Attributes

Part Num	Description	Quantity On Hand	Reorder Point	Supplier Number	Name	Address	Tele Num
1	Bracket	100	150	22	Ozment Sup	123 Main St.	555-7895
1	Bracket	100	150	24	Buell Co.	2 Broadhead	555-3436
1	Bracket	100	150	27	B&R Sup	Westgate Mall	555-7845
2	Gasket	440	450	22	Ozment Sup	123 Main St.	555-7895
2	Gasket	440	450	24	Buell Co.	2 Broadhead	555-3436
2	Gasket	440	450	28	Harris Manuf	24 Linden St.	555-3316
3	Brace	10	10	22	Ozment Sup	123 Main St.	555-7895
3	Brace	10	10	24	Buell Co.	2 Broadhead	555-3436
3	Brace	10	10	28	Harris Manuf	24 Linden St.	555-3316

supplier's name, address, or telephone number must be made to each of these records in the table. In the example, this means three different updates. To better appreciate the implications of the update anomaly, consider a more realistic situation where the vendor supplies 10,000 different items of inventory. Any update to an attribute must then be made 10,000 times.

Insertion Anomaly. To demonstrate the effects of the **insertion anomaly,** assume that a new vendor has entered the marketplace. The organization does not yet purchase from the vendor, but may wish to do so in the future. In the meantime, the organization wants to add the vendor to the database. This is not possible, however, because the primary key for the Inventory table is PART NUM. Because the vendor does not supply the organization with any inventory items, the supplier data cannot be added to the table.

Deletion Anomaly. The **deletion anomaly** involves the unintentional deletion of data from a table. To illustrate, assume that Supplier Number 27 provides the company with only one item: Part Number 1. If the organization discontinues this item of inventory and deletes it from the table, the data pertaining to Supplier Number 27 will also be deleted. Although the company may wish to retain the supplier's information for future use, the current table design prevents it from doing so.

The presence of the deletion anomaly is less conspicuous, but potentially more serious than the update and insertion anomalies. A flawed database design that prevents the insertion of records or requires the user to perform excessive updates attracts attention quickly. The deletion anomaly, however, may go undetected, leaving the user unaware of the loss of important data until it is too late. This can result in the unintentional loss of critical accounting records and the destruction of audit trails. Table design, therefore, is not just an operational efficiency issue; it carries internal control significance that accountants need to recognize.

Normalizing Tables

The database anomalies described above are symptoms of structural problems within tables called dependencies. Specifically, these are known as repeating groups, **partial dependencies,** and transitive dependencies. The normalization process involves identifying and removing structural dependencies from the table(s) under review. (For a full explanation of this process, please review the appendix to this chapter.) The resulting tables will meet the two conditions below:

1. All nonkey (data) attributes in the table are dependent on (defined by) the primary key.
2. All nonkey attributes are independent of the other nonkey attributes.

In other words, a 3NF table is one in which the primary key of a table wholly and uniquely defines each attribute in the table. Furthermore, none of the table attributes are defined by an attribute other than the primary key. If any attributes violate these conditions, they need to be removed and placed in a separate table and assigned an appropriate key.

Upon examination of Figure 9-14, we see that not all table attributes logically relate to the primary key PART NUM. In fact, as it currently stands, two distinct sets of data reside in this table: data about inventory and data about suppliers. The nonkey attributes of Name, Address, and Tele Num are not dependent on PART NUM. Rather, these attributes are dependent on the nonkey attribute Supplier Number. To resolve this **transitive dependency,** the supplier data must be removed from the Inventory table and placed in a separate table, which we will call Supplier. Figure 9-15 shows the two 3NF tables, Inventory

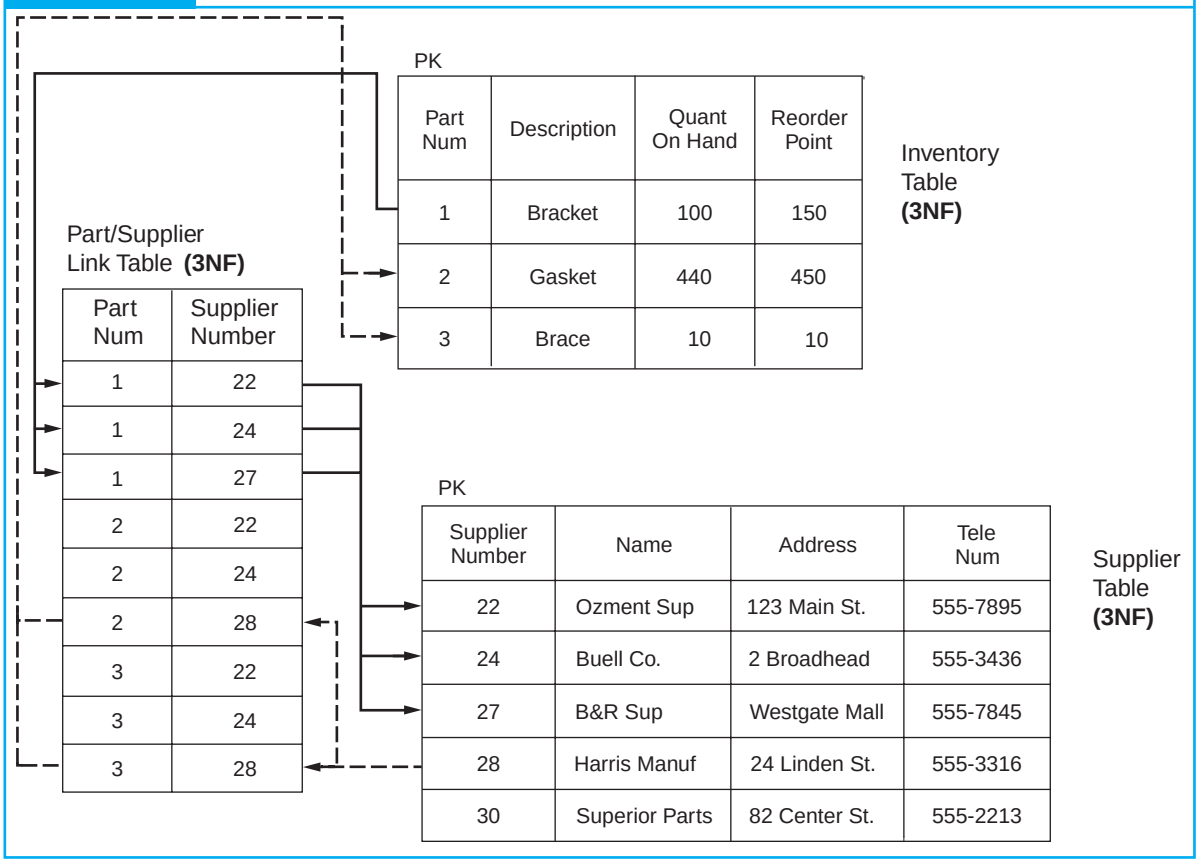
and Supplier, along with a third table called Part/Supplier, which links the two. This linking technique will be explained later.

Normalizing the tables has eliminated the three anomalies. First, the update anomaly is resolved because data about each supplier exist in only one location—the Supplier table. Any change in the data about an individual vendor is made only once, regardless of how many items it supplies. Second, the insert anomaly no longer exists because new vendors can be added to the Supplier table even if they are not currently supplying the organization with inventory. For example, Supplier Number 30 in the table does not supply any inventory items. Finally, the deletion anomaly is eliminated. The decision to delete an inventory item from the database will not result in the unintentional deletion of the supplier data because these data reside independently in different tables.

Linking Normalized Tables

When unnormalized tables are split into multiple 3NF tables, they need to be linked together via foreign keys so the data in them can be related and made accessible to users. The degree of association (joint cardinality) between the resulting tables (that is, 1:1, 1:M, or M:M) determines how the linking occurs. The key-assignment rules for linking tables are discussed next.

FIGURE 9-15 **Normalized Database Tables**



Keys in 1:1 Associations. Where a true 1:1 association exists between tables, either (or both) primary keys may be embedded as foreign keys in the related table. On the other hand, when the lower cardinality value is zero (1:0,1) a more efficient table structure can be achieved by placing the one-side (1:) table's primary key in the zero-or-one (:0,1) table as a foreign key. Using the Employee/Company Car example in Figure 9-11, we see the importance of this key-assignment rule. To illustrate, imagine reversing the rule by placing the Company Car (0 side) table's primary key into the Employee (1 side) table. Because most employees are not assigned a company car, most of the foreign keys in the Employee table will have null (blank) values. While this approach would work, it could cause some technical problems during table searches. Correctly applying the key-assignment rule solves this problem because all Company Car records will have an employee assigned and no null values will occur.

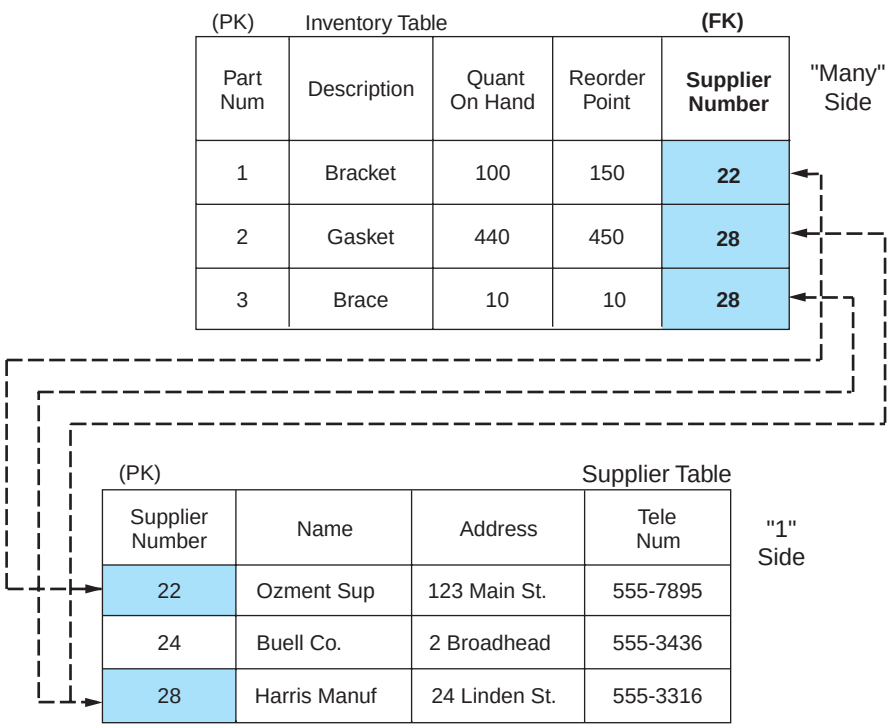
Keys in 1:M Associations. Where a 1:M (or 1:0,M) association exists, the primary key of the 1 side is embedded in the table of the M side. To demonstrate the logic behind this key-assignment rule, consider two alternative business rules for purchasing inventory from suppliers.

> Business Rule 1. Each vendor supplies the firm with three (or fewer) different items of inventory, but each item is supplied by only one vendor.

This unrealistic, but technically possible, business rule describes an upper-bounded 1:M (1:1,3) association between the Supplier and Inventory tables.

To apply this rule, the designer will need to modify the Inventory table structure to include the Supplier Number as illustrated in Figure 9-16. Under this approach,

FIGURE 9-16 **Applying the 1:M Key-Assignment Rule**



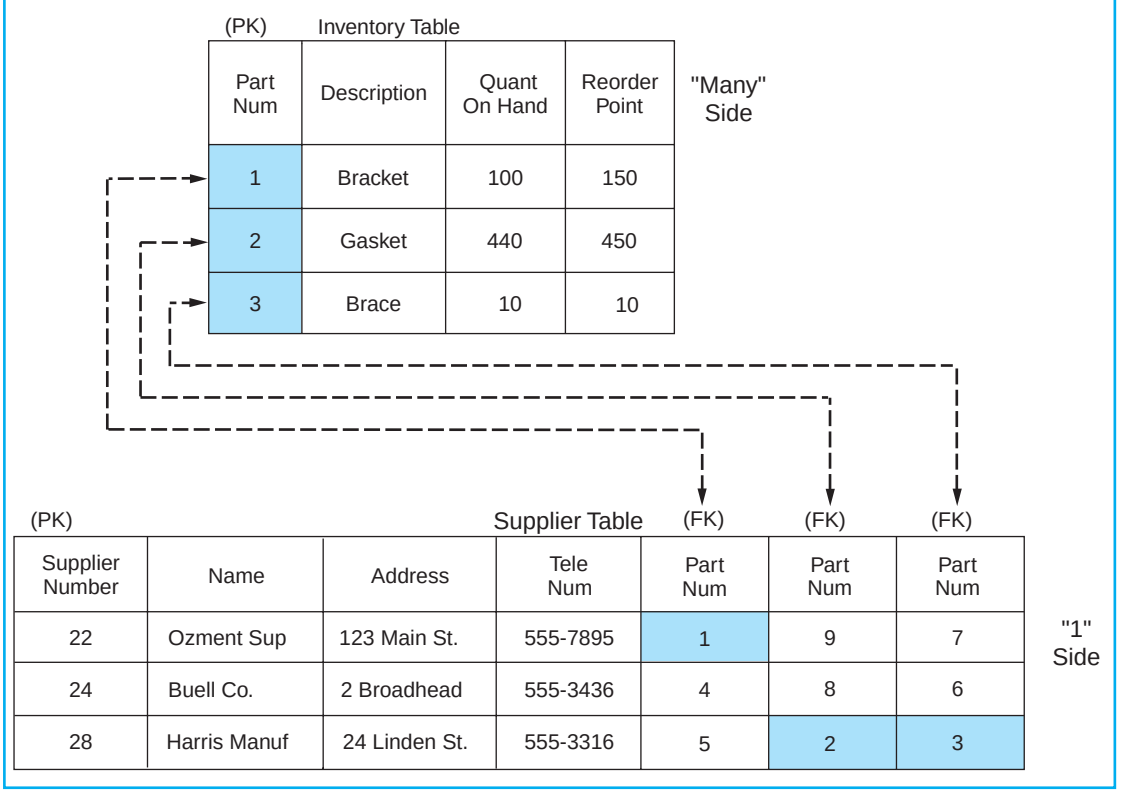
each record in the Inventory table will now contain the value of the key field of the vendor that supplies that item. By contrast, Figure 9-17 shows what the table structure might look like if the designer reversed the key-assignment rule by embedding the PART NUM key in the Supplier table. Notice that the Supplier table now contains three part number fields each linking to an associated record in the Inventory table. Only the links to part numbers 1, 2, and 3 are shown. Although this technique violates the key-assignment rule, it would work. It does so, however, only because the upper limit of the many side of the association is known and is very small (that is, limited to three). How would this table structure look if we assume the following, more realistic business rule?

> Business Rule 2. Each vendor supplies the firm with any number of inventory items, but each item is supplied by only one vendor.

This is a true 1:M association in which the upper limit of the many side of the association is unbounded. For instance, the vendor may supply one item of inventory or 10,000 items. How many fields must we add to the Supplier table structure to accommodate all possible links to the Inventory table? Here we can see the logic behind the 1:M key-assignment rule. The structure in Figure 9-16 still works under this business rule, whereas the technique illustrated in Figure 9-17 does not.

Keys in M:M Associations. To represent the M:M association between tables, a link table needs to be created. The link table has a combined (composite) key consisting of the primary

FIGURE 9-17 **Reversing the 1:M Key-Assignment Rule**



keys of two related tables. Let's now return to the table relationship depicted in Figure 9-15. These tables illustrate an M:M association that the following business rule describes:

> Business Rule 3. Each vendor supplies the firm with any number of inventory items, and each item may be supplied by any number of vendors.

This business rule is evident by examining the contents of the Inventory Status Report (the user view) in Figure 9-13. Each part number shown has multiple suppliers, and each supplier may supply multiple items. For example, Ozment Supply provides items 1, 2, and 3. Harris Manufacturing also provides items 2 and 3.

An M:M association between tables requires the creation of a separate link table because embedding a foreign key within either table is not possible. The logic in the previous 1:M example that prevented us from embedding the primary key from the many side table of an unbounded association into the table of the one side applies here also. Neither table can donate an embedded foreign key to the other because both are on the many side of the association. The solution, therefore, is to create a new link table containing the key fields of the other two tables.

The link table called Part/Supplier in Figure 9-15 is a table of foreign keys. It contains the primary keys for the records in the Inventory table (PART NUM) and the related Supplier table (SUPPLIER NUMBER). Via the link table, each inventory item can be linked to the corresponding supplier of the item, and each supplier can be linked to the inventory items that it supplies. For example, by searching the Inventory table for PART NUM 1, we see that suppliers 22, 24, and 27 provide this item. Searching in the opposite direction, SUPPLIER NUMBER 28 provides parts 2 and 3. A separate record in the link table represents each unique occurrence of a supplier/inventory association. For example, if supplier 24 provides 500 different inventory items, then the link table will contain 500 records to depict these associations.

Accountants and Data Normalization

Database normalization is a technical matter that is usually the responsibility of systems professionals. The subject, however, has implications for internal control that make it the concern of accountants also. For example, the update anomaly can generate conflicting and obsolete database values, the insertion anomaly can result in unrecorded transactions and incomplete audit trails, and the deletion anomaly can cause the loss of accounting records and the destruction of audit trails. Although most accountants will not be responsible for normalizing an organization's databases, they should have an understanding of the process and be able to determine whether a table is properly normalized.

Designing Relational Databases

This section examines the steps involved in creating a relational database. Keep in mind that database design is a portion of a much larger systems development process that involves extensive analysis of user needs, which are not covered at this time. That body of material is the subject of Chapters 13 and 14. Thus, our starting point is one that normally follows considerable preliminary work that has identified in detail the key elements of the system under development. With this backdrop, the focus will be on the following six phases of database design, which are collectively known as **view modeling:**

1. Identify entities.
2. Construct a data model showing entity associations.

3. Add primary keys and attributes to the model.
4. Normalize the data model and add foreign keys.
5. Construct the physical database.
6. Prepare the user views.

Identify Entities

View modeling begins by identifying the primary entities of the business function in question. Recall that entities are things about which the organization wishes to capture data. To demonstrate entity identification, we will analyze the following key features of a simplified purchasing system:

1. The purchasing agent reviews the inventory status report (Figure 9-13) for items that need to be reordered.
2. The agent selects a supplier and prepares an online purchase order.
3. The agent prints a copy of the purchase order (Figure 9-18a) and sends it to the supplier.
4. The supplier ships inventory to the company. Upon its arrival, the receiving clerk inspects the inventory and prepares an online receiving report (Figure 9-18b). The computer system automatically updates the inventory records.

FIGURE 9-18 **Purchase Order and Receiving Report for Purchases System**

a)

Purchase Order			PO Number	
Supplier Name				
Supplier Address				
			Tel No.	
Order Date / /	Date Required / /	Supplier Number	Terms	
Part Number	Description		Order Quantity	Unit Cost	Extended Cost
				Total Cost	

b)

Receiving Report			Receiving Report Number
Date Received	PO Number	Carrier Code	BOL Number
Freight: Prepaid / Collect	Supplier Name		
	Tel No.		
Supplier Address			
Part Number	Description	Quantity Received	Condition Code

Entities are represented as nouns in a system description. A number of candidate entities can be identified in the previous description: Purchasing Agent, Receiving Clerk, Inventory, Supplier, Inventory Status Report, Purchase Order, and Receiving Report. Not all of these candidates are true entities that need to be modeled in the database. To pass as valid entities, two conditions need to be met:

Condition 1. An entity must consist of two or more occurrences.

Condition 2. An entity must contribute at least one attribute that is not provided through other entities.

We need to test these conditions for each candidate to eliminate any false entities.

Purchasing Agent. Assuming that the organization has only one purchasing agent, then the Purchasing Agent candidate fails Condition 1. If, however, more than one agent exists, Condition 1 is met but Condition 2 may be a problem. If we assume that an Employee table already exists as part of a human resources or payroll system, then basic data about the agent as an employee is captured in that table. We need to determine what data about the agent that is unique to his or her role of order placing needs to be captured. Note that we are not referring to data about the order, but data about the agent. Because we have no information on this point in our brief description of the system, we will assume no agent-specific data are captured. Hence, the Purchasing Agent candidate is not an entity to be modeled.

Receiving Clerk. The previous argument applies also to the Receiving Clerk entity. We will assume that no clerk-specific data need to be captured that require a dedicated table.

Inventory. The Inventory entity meets both conditions. The description suggests that the organization holds many items of inventory; thus this entity would contain multiple occurrences. Also, we can logically assume that the attributes that define the Inventory entity are not provided through other tables. The Inventory entity is, therefore, a true entity that will need to be modeled.

Supplier. The description states that multiple vendors supply inventory; hence the Supplier entity meets the first condition. We can also assume that it meets the second condition since no other entity would logically provide supplier data. The Supplier entity, therefore, will be included in the data model.

Inventory Status Report. The Inventory Status Report is a user view derived from the Inventory and Supplier entities (see Figure 9-15). While it contains multiple occurrences, it is not an entity because it does not satisfy Condition 2. The view is derived entirely from existing entities and provides no additional data that requires a separate entity. The view will be carefully analyzed, however, to ensure that all the attributes needed for it are included in the existing entities.

Purchase Order. The Purchase Order entity meets both conditions. Many purchase orders will be processed in a period; thus the entity will have many occurrences. While some purchase order data can be derived from other entities (Inventory and Supplier) in the model, some attributes unique to the purchase event such as order date and order quantity will require a separate entity that needs to be modeled.

Receiving Report. The Receiving Report meets both conditions. Many receipt events will take place in the period; thus a Receiving Report entity will have multiple occurrences. A Receiving Report entity will contain attributes such as date received and quantity received that are unique to this entity and thus not provided by other entities in the model.

At this point our search has revealed four entities: Inventory, Supplier, Purchase Order, and Receiving Report. These will be used to construct a data model and, ultimately, the physical database tables.

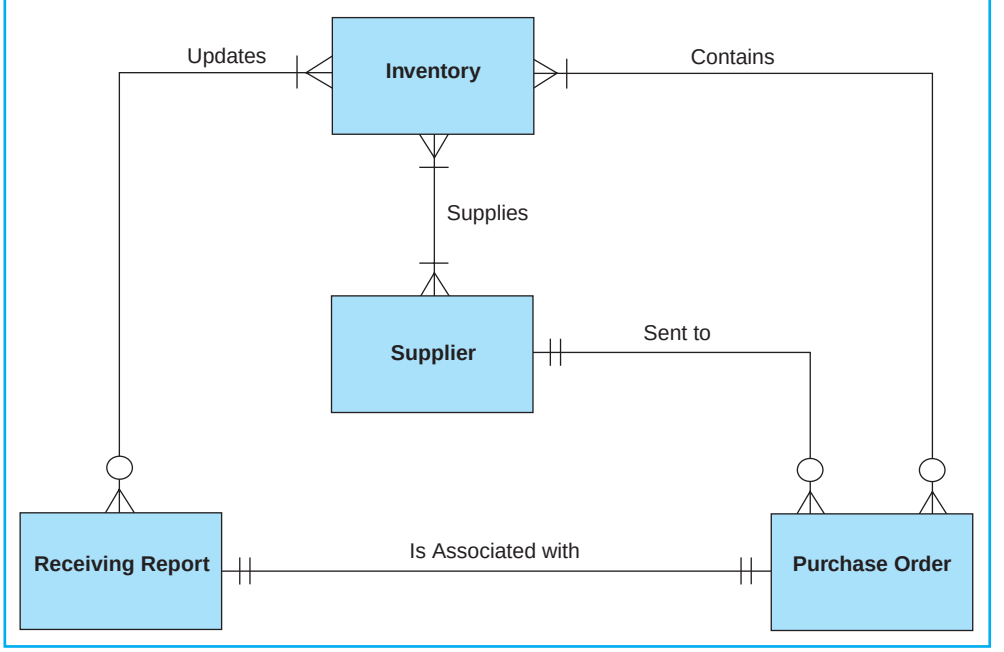
Construct a Data Model Showing Entity Associations

The next step in view modeling is to determine the associations between entities and document them with an ER diagram. Recall that associations represent business rules. Sometimes the rules are obvious and are the same for all organizations. For example, the normal association between a Customer entity and a Sales Order entity is 1:M (or 1:0,M). This signifies that one customer may place many orders during a sales period. The association would never be 1:1. This would mean that the organization restricts each customer to a single sale, which is illogical.

Sometimes the association between entities is not apparent because different rules may apply in different organizations. To reiterate an important point made earlier, the organization's business rules directly impact the structure of the database tables. If the database is to function properly, its designers need to understand the organization's business rules as well as the specific needs of individual users. Figure 9-19 illustrates the entity associations in our example. The underlying business rules are explained next.

1. There is a 0,M:M association between the Purchase Order and Inventory entities. This means that each inventory item may have been ordered many times or never

FIGURE 9-19 Data Model Showing Entity Associations



ordered in the current business period. Obviously, every inventory item must have been purchased at least once in the past, so why do we show a 0,M cardinality for the Purchase Order entity? We must keep in mind that transaction entities, such as sales and purchases, are associated with a particular time frame. We will assume that the Purchase Order table for this system will contain records for purchases made in the current period only. Closed purchase orders of past periods will have been removed to an archive table, which is not shown in our example.

2. There is an M:M association between the Inventory and Supplier entities. This means that one or more vendors supply each inventory item, and each of them supplies one or more items of inventory.
3. There is a 1:0,M association between the Supplier and the Purchase Order entities. This means that in the current period, each supplier may have received zero or many purchase orders, but each order goes to only one supplier.
4. There is a 1:1 association between the Purchase Order and Receiving Report entities. A single receiving report record reflects the receipt of goods that are specified on a single purchase order record. Multiple purchase orders are not combined on a single receiving report.
5. The association between the Receiving Report and Inventory entities is 0,M:M. This signifies that within the period, each item of inventory may have been received many times or never. Also, each receiving report is associated with at least one and possibly many inventory items.

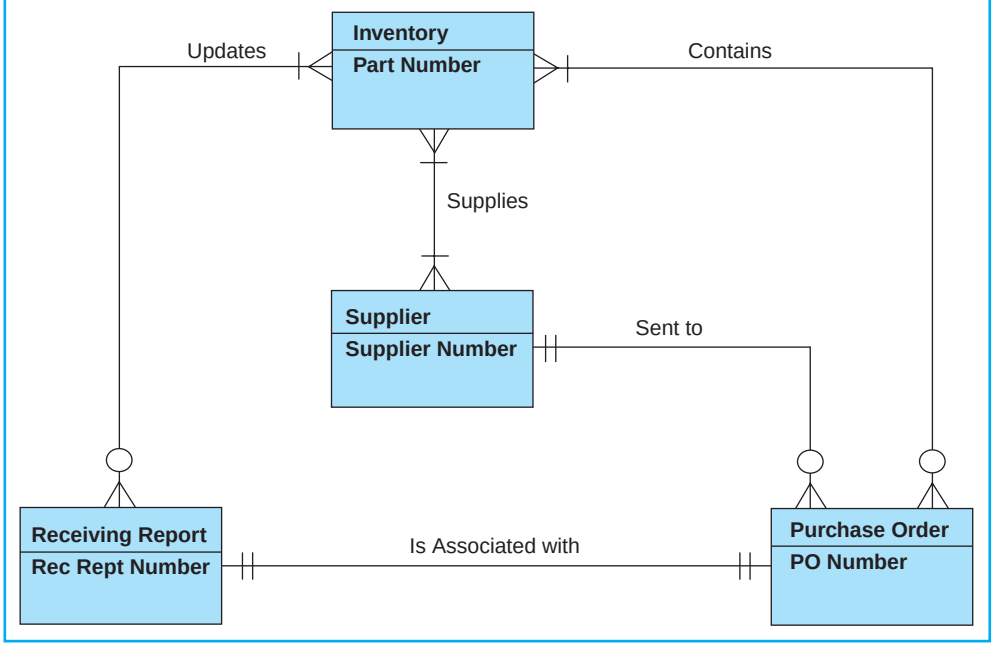
The many-to-many (M:M and 0,M:M) associations in the data model need to be resolved before the physical databases can be created. We know from previous discussion that these associations signify a missing entity that is needed to link them. We will resolve these problems during the normalization process.

Add Primary Keys and Attributes to the Model

Add Primary Keys. The next step in the process is to assign primary keys to the entities in the model. The analyst should select a primary key that logically defines the nonkey attributes and uniquely identifies each occurrence in the entity. Sometimes this can be accomplished using a simple sequential code such as an Invoice Number, Check Number, or Purchase Order number. Sequential codes, however, are not always efficient or effective keys. Through careful design of block codes, group codes, alphabetic codes, and mnemonic codes, primary keys can also impart useful information about the nature of the entity. These techniques are discussed in detail in Chapter 8. Figure 9-20 presents the four entities in the model with primary keys assigned.

Add Attributes. Every attribute in an entity should appear directly or indirectly (a calculated value) in one or more user views. Entity attributes are, therefore, originally derived and modeled from user views. In other words, if stored data are not used in a document, report, or a calculation that is reported in some way, then it serves no purpose and should not be part of the database. The attributes assigned to each entity in Figure 9-21 are derived from the user views of the Purchase Order and Receiving Report illustrated in Figure 9-18 and from the Inventory Status Report that we previously normalized.

FIGURE 9-20 Data Model Showing Primary Keys

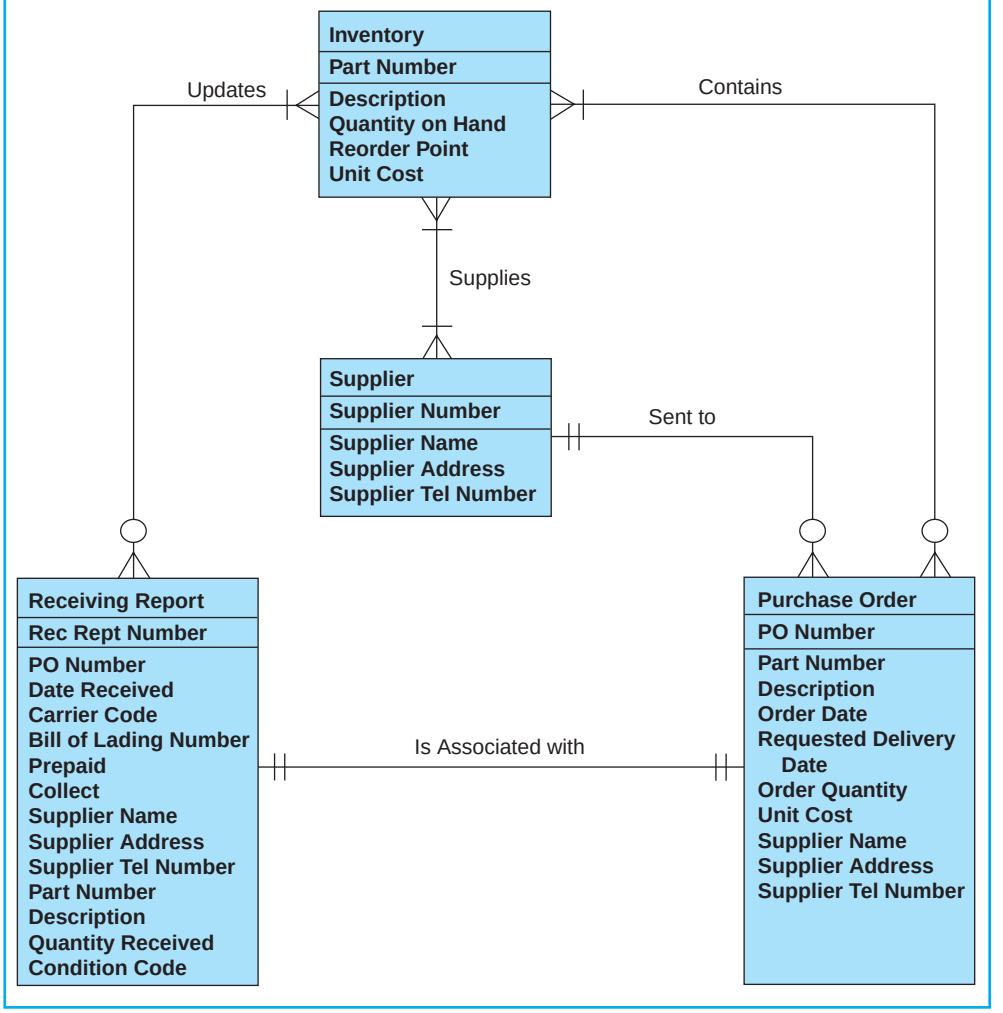


Normalize Data Model and Add Foreign Keys

Figure 9-22 presents a normalized data model. The normalization issues that needed resolution are outlined in the following section:

1. **Repeating Group Data in Purchase Order.** The attributes Part Number, Description, Order Quantity, and Unit Cost are **repeating group** data. This means that when a particular purchase order contains more than one item (most of the time), then multiple values will need to be captured for these attributes. To resolve this, these repeating group data were removed to a new PO Item Detail entity. The new entity was assigned a primary key that is a composite of Part Number and PO Number. The creation of the new entity also resolved the M:M association between the Purchase Order and Inventory entities by providing a link.
2. **Repeating Group Data in Receiving Report.** The attributes Part Number, Quantity Received, and Condition Code are repeating groups in the Receiving Report entity and were removed to a new entity called Rec Report Item Detail. A COMPOSITE KEY composed of PART NUMBER and REC REPT NUMBER was assigned. As in the previous example, creating this new entity also resolved the M:M association between Receiving Report and Inventory.
3. **Transitive Dependencies.** The Purchase Order and Receiving Report entities contain attributes that are redundant with data in the Inventory and Supplier entities. These redundancies occur because of transitive dependencies (see the appendix of this chapter) in the Purchase Order and Receiving Report entities and are dropped.

FIGURE 9-21 Data Model Showing Keys and Attributes



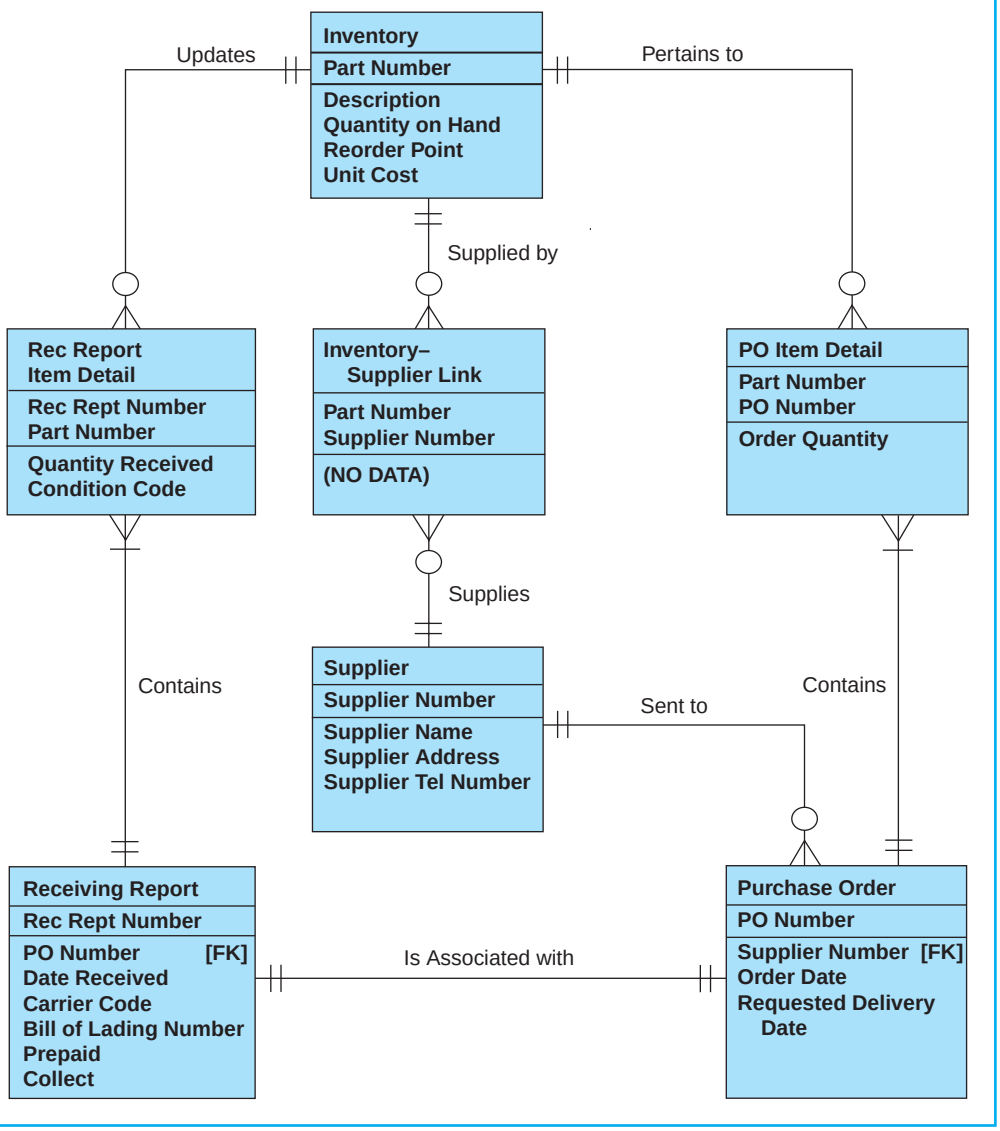
Construct the Physical Database

Figure 9-23 illustrates the 3NF table structures for the database. The primary and foreign keys linking the tables are represented by dotted lines. The following points are worth elaboration.

Each record in the Rec Report Item Detail table represents an individual item on the receiving report. The table has a combined key comprising REC REPT NUMBER and PART NUMBER. This composite key is needed to uniquely identify the Quantity Received and Condition attributes of each item-detail record. The REC REPT NUMBER portion of the key provides the link to the Receiving Report table that contains general data about the receiving event. The PART NUMBER portion of the key is used to access the Inventory table to facilitate updating the Quantity on Hand field from the Quantity Received field of the Item-Detail record.

The PO Item Detail table uses a composite primary key of PO NUMBER and PART NUMBER to uniquely identify the Order Quantity attribute. The PO NUMBER

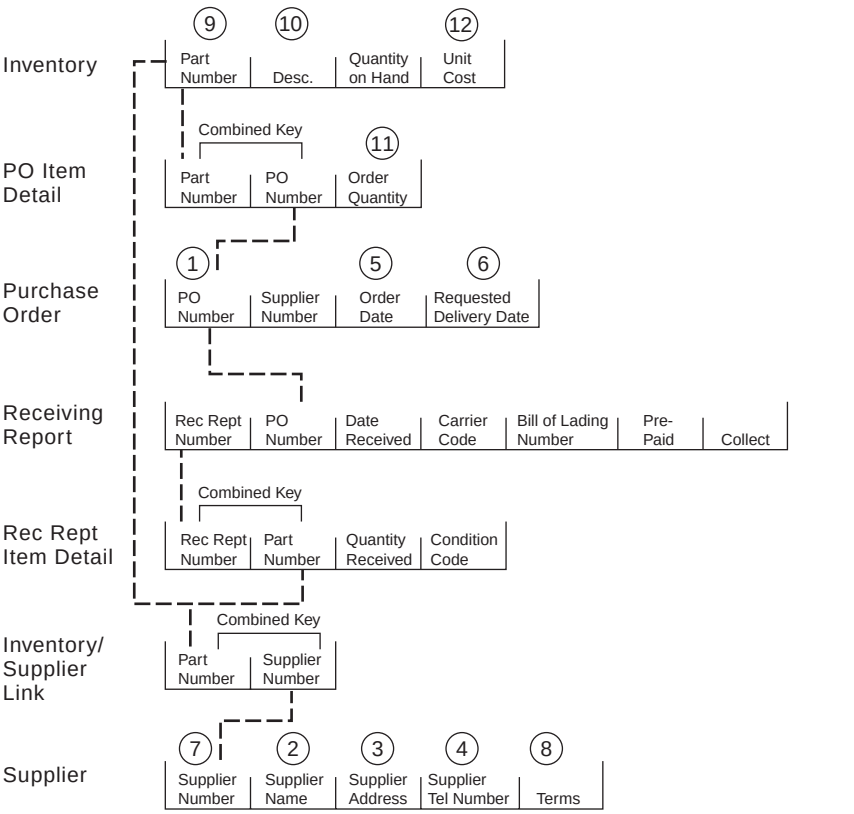
FIGURE 9-22 Normalized Data Model



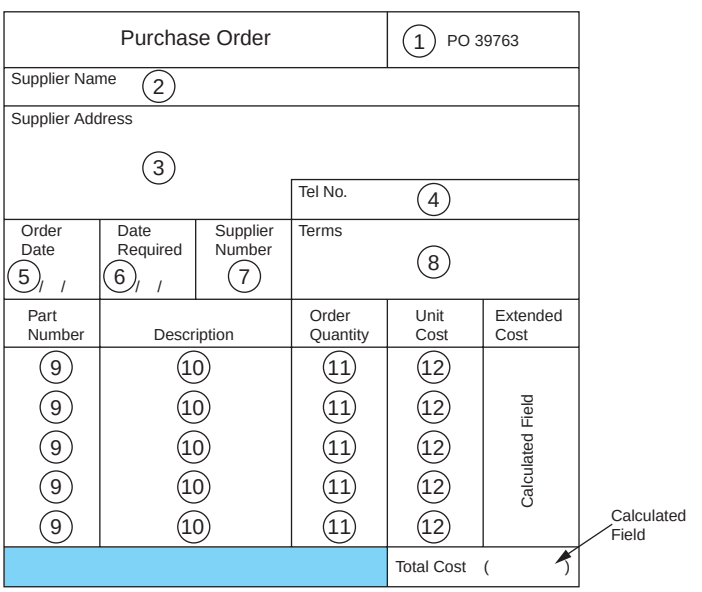
component of the composite key provides a link to the Purchase Order table. The PART NUMBER element of the key is a link to the Inventory table where Description and Unit Cost data reside.

The next step is to create the physical tables and populate them with data. This is an involved step that must be carefully planned and executed and may take many months in a large installation. Programs will need to be written to transfer organization data currently stored in flat files or legacy databases to the new relational tables. Data currently stored on paper documents may need to be entered into the database tables manually. Once this is done, the physical user views can be produced.

FIGURE 9-23 Normalized Tables



User's View—Purchase Order



Prepare the User Views

The normalized tables should be rich enough to support the views of all users of the system being modeled. For example, the purchase order in Figure 9-23, which could be the data entry screen for a purchasing clerk, has been constructed from attributes located in several tables. To illustrate the relationship, the fields in the user view are cross-referenced via circled numbers to the attributes in the supporting tables. Keep in mind that these tables may also provide data for many other views not shown here, such as the receiving report, purchase requisition listing, inventory status report, and vendor purchases activity report.

The query function of a relational DBMS allows the system designer to easily create user views from tables. The designer simply tells the DBMS which tables to use, their primary and foreign keys, and the attributes to select from each table. Older DBMSs require the designer to specify view parameters directly in SQL. Newer systems do this visually. The designer simply points and clicks at the tables and the attributes. From this visual representation, the DBMS generates the SQL commands for the query to produce the view.

The Receiving Report, Purchase Order, and Inventory Status Report views would all be created in this way. To illustrate, the SQL commands needed to produce the inventory status report illustrated in Figure 9-13 are given in the following section.

```
SELECT inventory.part-num, description, quant-on-hand, reorder-point, EOQ, part-supplier.
part-num, part-supplier.supplier-number, supplier.supplier-number, name, address, tele-num,

FROM inventory, part-supplier, supplier

WHERE inventory.part-num=part-supplier.part-num AND part-supplier. supplier-number=supplier.
supplier-number AND quant-on hand ≤ reorder-point.
```

- The SELECT command identifies all of the attributes to be contained in the view. When the same attribute appears in more than one table (for example, PART-NUM), the source table name must also be specified.
- The FROM command identifies the tables used in creating the view.
- The WHERE command specifies how rows in the Inventory, Part-Supplier, and Supplier tables are to be matched to create the view. In this case, the three tables are algebraically joined on the primary keys PART-NUM and SUPPLIER-NUMBER.
- Multiple expressions may be linked with the AND, OR, and NOT operators. In this example, the last expression uses AND to restrict the records to be selected with the logical expression quant-on-hand ≤ reorder-point. Only records whose quantities on hand have fallen to or below their reorder points will be selected for the view. The user will not see the many thousands of other inventory items that have adequate quantities available.

These SQL commands will be saved in a user program called a query. To view the Inventory Status report, the purchasing agent executes the query program. Each time this is done, the query builds a new view with current data from the Inventory and Vendor tables. By providing the user with his or her personal query, rather than permitting access to the underlying base tables, the user is limited to authorized data only.

A report program is used to make the view visually attractive and easy to use. Column headings can be added, fields summed, and averages calculated to produce a hard-copy or computer screen report that resembles the original user report in Figure 9-13. The report program can suppress unnecessary data from the view, such as duplicated fields and the key values in the Inventory/Vendor link table. These keys are necessary to build the view, but are not needed in the actual report.

Global View Integration

The view modeling process described above pertained to only one business function—the purchases system—and the resulting tables and views constitute only a subschema of the overall database schema. A modern company, however, would need hundreds or thousands of views and associated tables. Combining the data needs of all users into a single schema or enterprise-wide view is called **view integration.** This is a daunting undertaking when creating the entire database from scratch. To facilitate this task, modern Enterprise Resource Planning (ERP) systems (discussed in Chapter 11) come equipped with a core schema, normalized tables, and view templates. These best practices databases are derived from economic models that identify commonalities among the data needs of different organizations. For example, all organizations that sell products to customers will need an Inventory table, a Customer table, a Supplier table, and so forth. Many of the attributes and keys in these tables are also common to all organizations. Working from a core ERP database, the view modeling process thus becomes one of configuring or tailoring predefined views to accommodate specific user needs. ERP vendors cannot, however, anticipate the information needs of all users in advance. Therefore, new tables and new attributes may need to be added to the core schema. Although configuring the core database in this fashion is far more efficient than working from scratch, the objective is the same. The database designer must produce a set of integrated tables that are free of the update, insert, and deletion anomalies and sufficiently rich to serve the needs of all users.

In Chapter 10 we will examine in detail the resources events agents (REA) model that underlies many ERP database designs. Applying REA analysis to accounting information system solutions promotes an understanding of the relevant economic events and exchanges between trading partners. ERP vendors have thus been able to anticipate the information needs of both accounting and nonaccounting users and produce core databases that support multiple views for most organization types.

Databases in a Distributed Environment

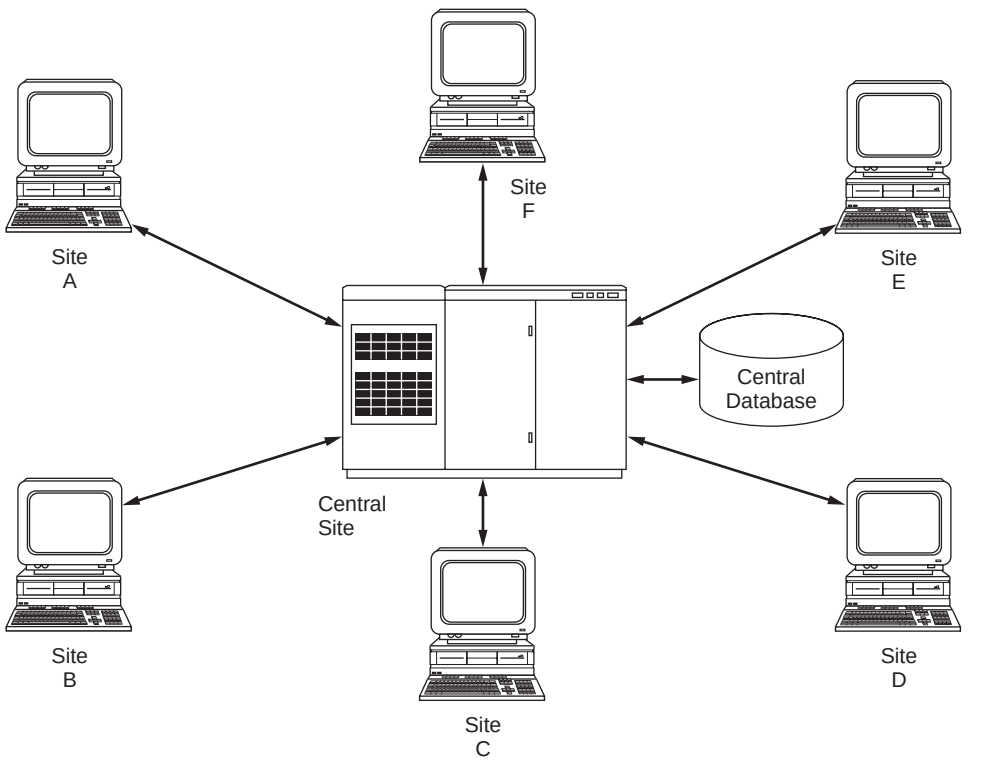
Chapter 1 introduced the concept of **distributed data processing (DDP)** as an alternative to the centralized approach. Most modern organizations use some form of distributed processing and networking to process their transactions. Some companies process all of their transactions in this way. An important consideration in planning a distributed system is the location of the organization's database. In addressing this issue, the planner has two basic options: databases can be centralized, or they can be distributed. Distributed databases fall into two categories: partitioned and replicated databases. This section examines issues, features, and trade-offs that should be carefully evaluated in deciding how databases should be distributed.

Centralized Databases

Under the **centralized database** approach, remote users send requests via terminals for data to the central site, which processes the requests and transmits the data back to the user. The central site performs the functions of a file manager that services the data needs of the remote users. The centralized database approach is illustrated in Figure 9-24.

Earlier in the chapter, three primary advantages of the database approach were presented: the reduction of data storage costs, the elimination of multiple update procedures, and the establishment of **data currency** (that is, the firm's data files reflect accurately the effects of its transactions). Achieving data currency is critical to database integrity and reliability. However, in the DDP environment, this can be a challenging task.

FIGURE 9-24 Centralized Database



Data Currency in a DDP Environment

During data processing, account balances pass through a state of **temporary inconsistency,** in which their values are incorrectly stated. This occurs during the execution of any accounting transaction. To illustrate, consider the computer logic for recording the credit sale of $2,000 to customer Jones.

	INSTRUCTION	DATABASE VALUES AR-Jones	AR-Control
	START		
1	Read AR-Sub account (Jones)	1500	
2	Read AR-Control account		10000
3	Write AR-Sub account (Jones) + $2000	3500	
4	Write AR-Control account + $2000		12000
	END		

Immediately after the execution of Instruction Number 3, and before the execution of Instruction Number 4, the AR-Control account value is temporarily inconsistent by the sum of $2,000. Only after the completion of the entire transaction is this inconsistency resolved. In a DDP environment, such temporary inconsistencies can result in the permanent corruption of the database. To illustrate the potential for damage, look at a slightly more complicated example. Using the same computer logic as before, consider the processing of two separate transactions from two remote sites: Transaction 1 (T1) is the sale of $2,000 on account to customer Jones from Site A; Transaction 2 (T2) is the sale of

$1,000 on account to customer Smith from Site B. The following logic shows the possible interweaving of the two processing tasks and the effect on data currency.

INSTRUCTION			DATABASE VALUES Central Site		
SITE A	SITE B		AR-Jones	AR-Smith	AR-Control
T1	T2	START			
1		Read AR-Sub account (Jones)	1500		
	1	Read AR-Sub account (Smith)		3000	
2		Read AR-Control account			10000
3		Write AR-Sub account (Jones) + $2000		3500	
	2	**Read AR-Control account**			10000
4		Write AR-Control account + $2000			12000
	3	Write AR-Sub account (Smith) + $1000		4000	
	4	Write AR-Control account + $1000			11000
		END			

Notice that Site B seized the AR-Control data value of $10,000 when it was in an inconsistent state. By using this value to process its transaction, Site B effectively destroyed the record of Transaction T1 that had been processed by Site A. Therefore, instead of $13,000, the new AR-Control balance is misstated at $11,000.

Database Lockout

To achieve data currency, simultaneous access to individual data elements by multiple sites needs to be prevented. The solution to this problem is to use a **database lockout,** which is a software control (usually a function of the DBMS) that prevents multiple simultaneous accesses to data. The previous example can be used to illustrate this technique: immediately upon receiving the access request from Site A for AR-Control (T1, Instruction Number 2), the central site DBMS places a lock on AR-Control to prevent access from other sites until Transaction T1 is complete. Thus, when Site B requests AR-Control (T2, Instruction Number 2), it is placed on wait status until the lock is removed. Only then can Site B access AR-Control and complete Transaction T2.

Distributed Databases

Distributed databases can be distributed using either the partitioned or replicated technique.

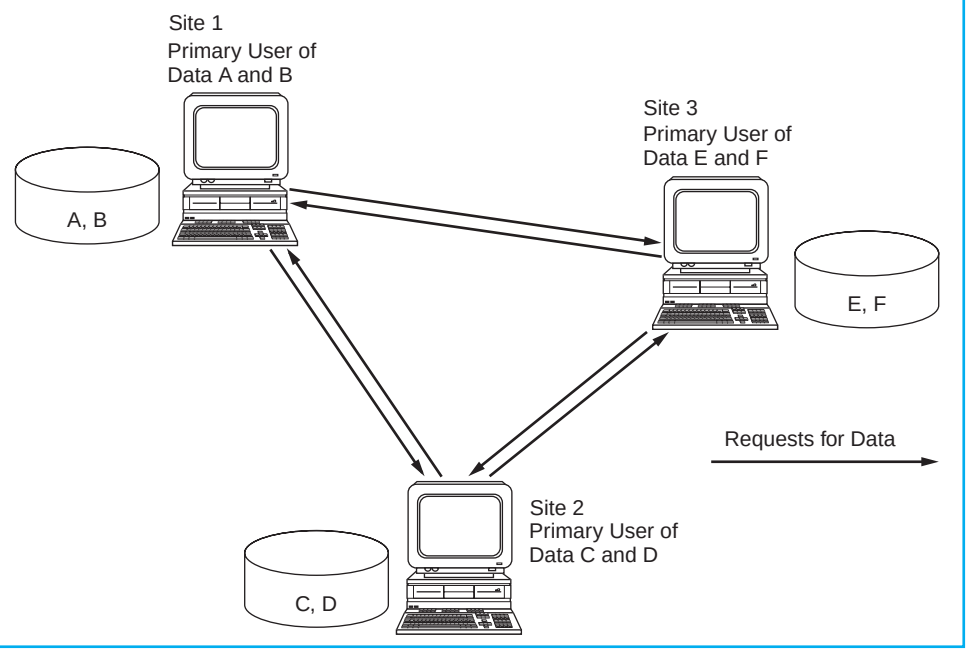
Partitioned Databases

The **partitioned database** approach splits the central database into segments or partitions that are distributed to their primary users. The advantages of this approach are:

- Storing data at local sites increases users' control.
- Permitting local access to data and reducing the volume of data that must be transmitted between sites improves transaction processing response time.
- Partitioned databases can reduce the potential for disaster. By having data located at several sites, the loss of a single site cannot terminate all data processing by the organization.

The partitioned approach, which is illustrated in Figure 9-25, works best for organizations that require minimal data sharing among users at remote sites. To the extent that remote users share common data, the problems associated with the centralized approach still apply. The primary user must now manage requests for data from other sites. Selecting

FIGURE 9-25 The Partitioned Database Approach



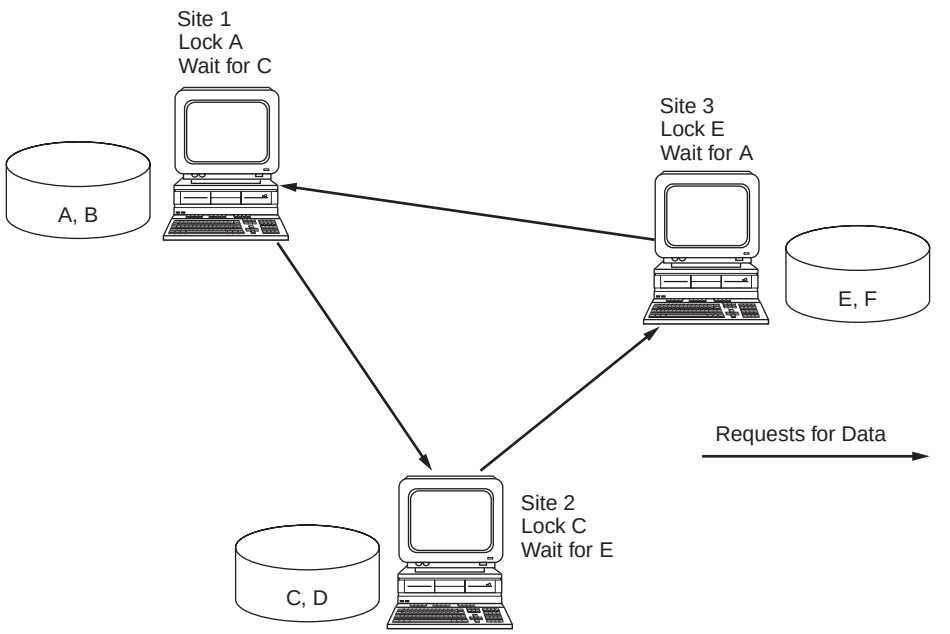
the optimum host location for the partitions will minimize data access problems. This requires an in-depth analysis of end-user data needs.

The Deadlock Phenomenon. In a distributed environment, it is possible that multiple sites will lock out each other, thus preventing each from processing its transactions. For example, Figure 9-26 illustrates three sites and their mutual data needs. Notice that Site 1 has requested (and locked) Data A and is waiting for the removal of the lock on Data C to complete its transaction. Site 2 has a lock on C and is waiting for E. Finally, Site 3 has a lock on E and is waiting for A. A **deadlock** occurs here because there is mutual exclusion to data, and the transactions are in a wait state until the locks are removed. This can result in transactions being incompletely processed and corruption of the database. A deadlock is a permanent condition that must be resolved by special software that analyzes each deadlock condition to determine the best solution. Because of the implications for transaction processing, accountants should be aware of the issues pertaining to deadlock resolutions.

Deadlock Resolution. Resolving a deadlock usually involves sacrificing one or more transactions. These must be terminated to complete the processing of the other transactions in the deadlock. The preempted transactions must then be reinitiated. In preempting transactions, the deadlock resolution software attempts to minimize the total cost of breaking the deadlock. Although not an easy task to automate, some of the factors that influence this decision are:

1. The resources currently invested in the transaction. This may be measured by the number of updates that the transaction has already performed and that must be repeated if the transaction is terminated.
2. The transaction's stage of completion. In general, deadlock resolution software will avoid terminating transactions that are close to completion.

FIGURE 9-26 The Deadlock Condition



3. The number of deadlocks associated with the transaction. Because terminating the transaction breaks all deadlock involvement, the software should attempt to terminate transactions that are part of more than one deadlock.

Replicated Databases

In some organizations, the entire database is replicated at each site. **Replicated databases** are effective in companies where there exists a high degree of data sharing but no primary user. Since common data are replicated at each site, the data traffic between sites is reduced considerably. Figure 9-27 illustrates the replicated database model.

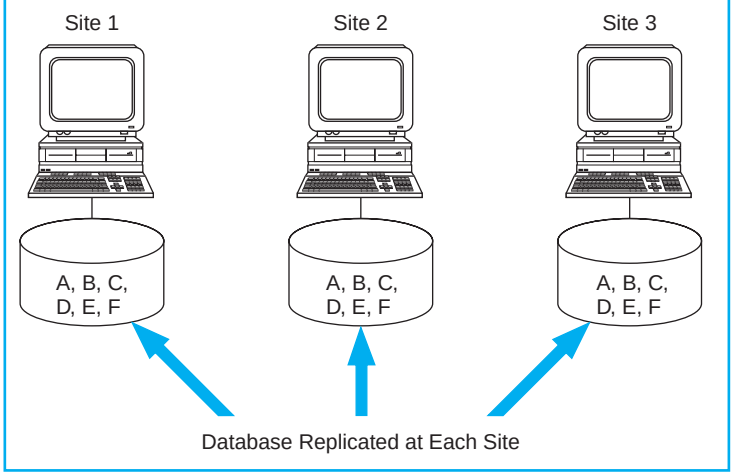
The primary justification for a replicated database is to support read-only queries. With data replicated at every site, data access for query purposes is ensured, and lockouts and delays because of network traffic are minimized. A problem arises, however, when local sites also need to update the replicated database with transactions.

Because each site processes only its local transactions, different transactions will update the common data attributes that are replicated at each site and, thus, each site will possess uniquely different values after the respective updates. Using the data from the earlier example, Figure 9-28 illustrates the effect of processing credit sales for Jones at Site A and Smith at Site B. After the transactions are processed, the value shown for the common AR-Control account is inconsistent ($12,000 at Site A and $11,000 at Site B) and incorrect at both sites.

Concurrency Control

Database concurrency is the presence of complete and accurate data at all remote sites. System designers need to employ methods to ensure that transactions processed at each site are accurately reflected in the databases at all other sites. This task, while problematic, has implications for accounting records and is a matter of concern for accountants.

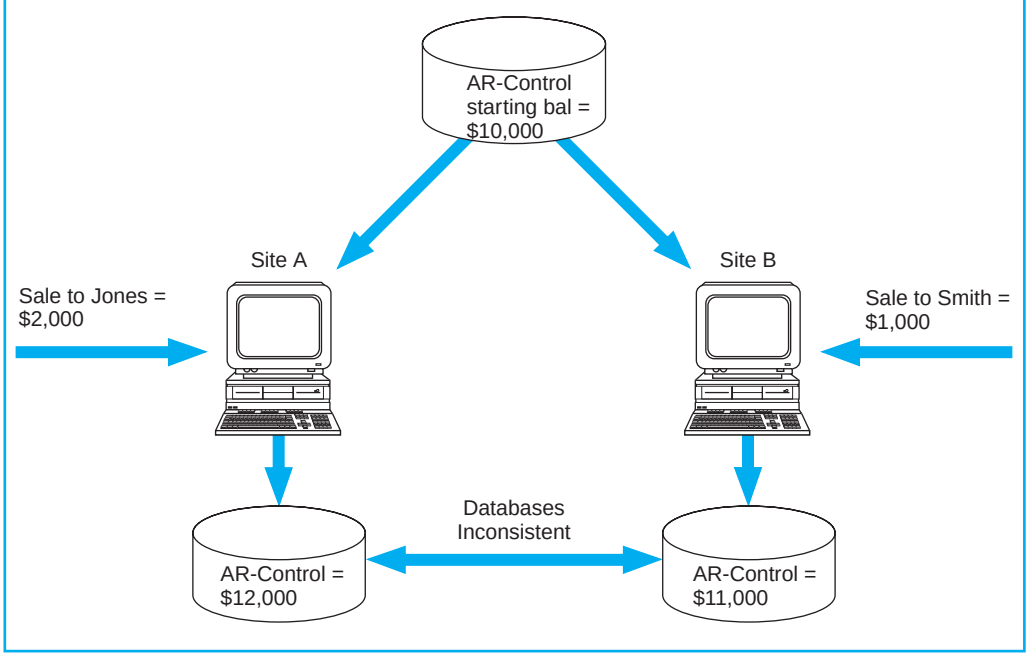
FIGURE 9-27 Replicated Database Approach



A commonly used method for **concurrency control** is to serialize transactions. This involves labeling each transaction by two criteria. First, special software groups transactions into classes to identify potential conflicts. For example, read-only (query) transactions do not conflict with other classes of transactions. Similarly, AP and AR transactions are not likely to use the same data and do not conflict. However, multiple sales order transactions involving both read and write operations will potentially conflict.

The second part of the control process is to time stamp each transaction. A system-wide clock is used to keep all sites, some of which may be in different time zones, on the

FIGURE 9-28 Replicated Databases Updated Independently



same logical time. Each time stamp is made unique by incorporating the site's ID number. When transactions are received at each site, they are examined first for potential conflicts. If conflicts exist, the transactions are entered into a serialization schedule. An algorithm is used to schedule updates to the database based on the transaction time stamp and class. This method permits multiple interleaved transactions to be processed at each site as if they were serial events.

Distributed Databases and the Accountant

The decision to distribute databases is one that should be entered into thoughtfully. There are many issues and trade-offs to consider. Some of the most basic questions to be addressed are:

- Should the organization's data be centralized or distributed?
- If data distribution is desirable, should the databases be replicated or partitioned?
- If replicated, should the databases be totally replicated or partially replicated?
- If the database is to be partitioned, how should the data segments be allocated among the sites?

The choices involved in each of these questions impact the organization's ability to maintain database integrity. The preservation of audit trails and the accuracy of accounting records are key concerns. Clearly, these are decisions that the modern accountant should understand and influence intelligently.

Summary

This chapter examined the database approach to data management and showed how this approach enables business organizations to overcome data redundancy and the associated problems that plague the flat-file approach to data management. It showed that the database concept is composed of four dynamically interrelated components: users, the database management system, the database administrator, and the physical database. The DBMS stands between the physical database and the user community. Its principal function is to provide a controlled and secure environment for the database. This is achieved through software modules, such as a query language, a data definition language, and a data manipulation language. The DBMS also provides security against human error and natural disaster through various backup and recovery modules.

Database models are abstract representations of data about entities, events, and activities, and their relationships within an organization. The focus of attention was on the relational model. A number of database design topics were covered, including data modeling, the creation of user views from ER diagrams, and data normalization techniques. Finally, the chapter presented a number of issues associated with distributed databases. It examined three possible database configurations in a distributed environment: centralized, partitioned, and the replicated databases.

Appendix

The Hierarchical Database Model

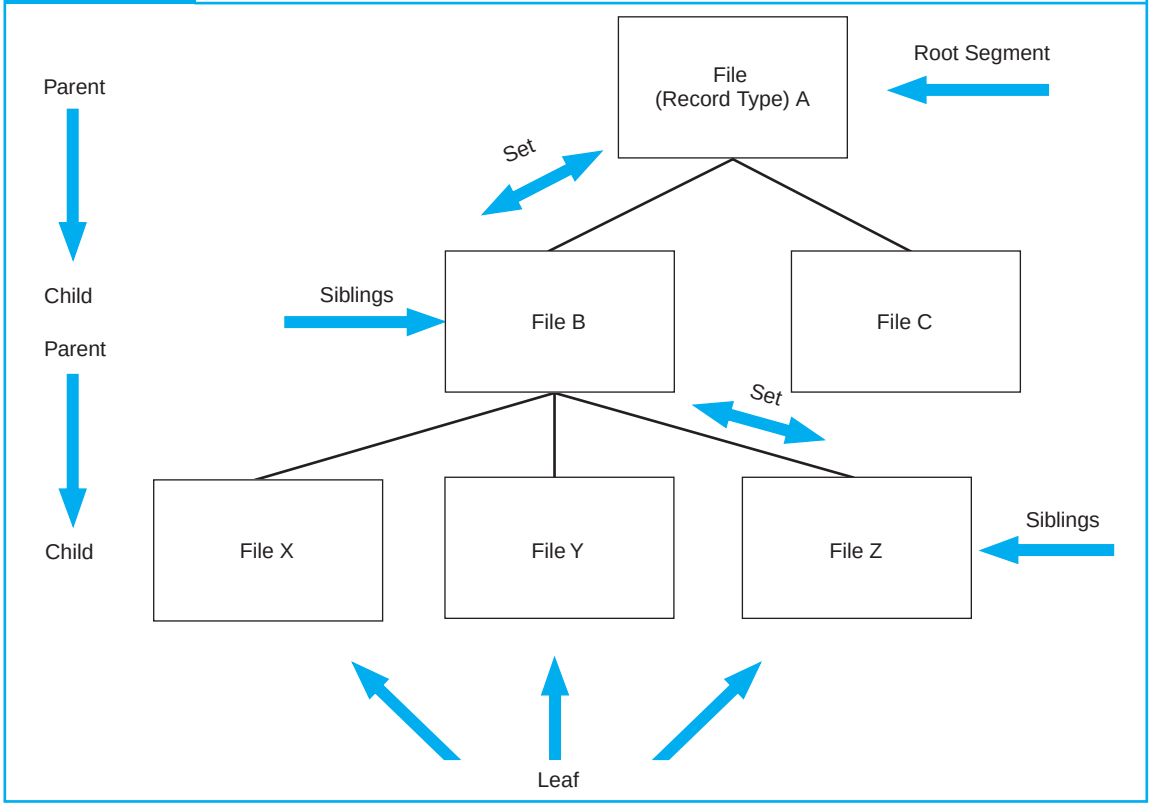
The earliest database management systems were based on the hierarchical data model. This was a popular approach to data representation because it reflected, more or less faithfully, many aspects of an organization that are hierarchical in relationship. Also, it was an efficient data processing tool for highly structured problems. Figure 9-29 presents a data structure diagram showing a portion of a hierarchical database.

The hierarchical model is constructed of sets of files. Each set contains a parent and a child. Notice that File B, at the second level, is both the child in one set and the parent in another set. Files at the same level with the same parent are called siblings. This structure is also called a tree structure. The file at the most aggregated level in the tree is the root segment, and the file at the most detailed level in a particular branch is called a leaf.

A Navigational Database

The hierarchical data model is called a navigational database because traversing it requires following a predefined path. This is established through pointers (discussed in Chapter 2) that create explicit linkages between related records. The only way to access data at lower levels in the tree is from the root and via the

FIGURE 9-29 Hierarchical Data Model



pointers down the navigational path to the desired records. For example, consider the partial database in Figure 9-30. To retrieve an invoice line-item record, the DBMS must first access the customer record (the root). That record contains a pointer to the sales invoice record, which points to the invoice line-item record.

Limitations of the Hierarchical Model

The hierarchical model presents a limited view of data relationships. Based on the proposition that all business relationships are hierarchical (or can be represented as such), this model does not always reflect reality. The following rules, which govern the hierarchical model, reveal its operating constraints:

1. A parent record may have one or more child records. For example, in Figure 9-30, the customer is the parent of both sales invoice and cash receipts.
2. No child record can have more than one parent.

The second rule is often restrictive and limits the usefulness of the hierarchical model. Many firms need a view of data associations that permit multiple parents like that represented by Figure 9-31(a). In this example, the sales invoice file has two natural parents: the customer file and the salesperson file. A specific sales order is the product of both the customer's purchase activity and a salesperson's selling efforts. Management, wishing to keep track of sales orders by customer and by salesperson, will want to view sales order records as the logical child of both parents. This relationship, although logical, violates the single parent rule of the hierarchical model. Figure 9-31(b) shows the most common way of resolving this problem. By duplicating the sales invoice file, two separate hierarchical representations are created. Unfortunately, this improved functionality is achieved at a cost—increased data redundancy. The network model, examined next, deals with this problem more efficiently.

The Network Database Model

The network model is a variation of the hierarchical model. The principal distinguishing feature between the two is that the network model allows a child record to have multiple parents. The multiple ownership rule is flexible in allowing complex relationships to be represented. Figure 9-31(a) illustrates a simple network model in which the sales invoice file has two parent files—Customer and Salesperson.

FIGURE 9-30 Portion of a Hierarchical Database

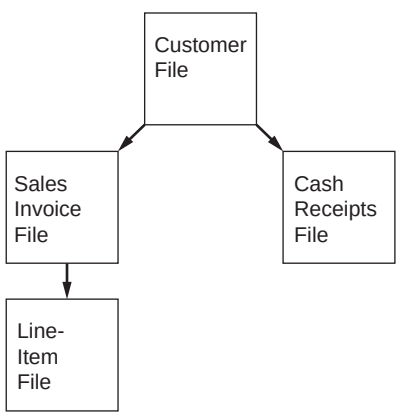
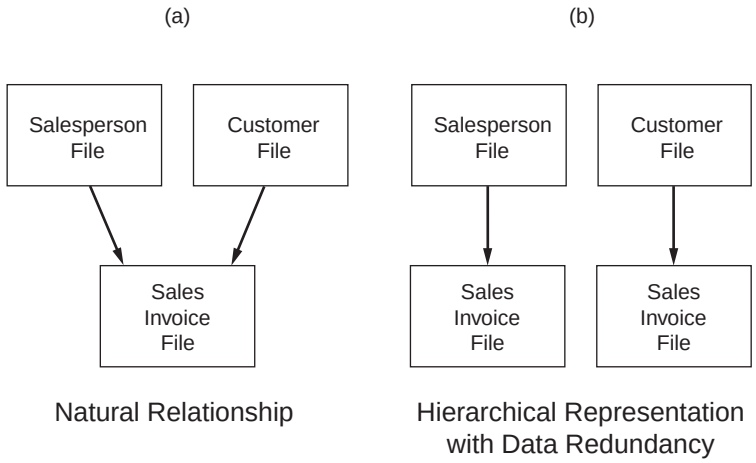


FIGURE 9-31 Multiple Parent Association



The navigational DBMS dominated the data processing industry for many years, although the current trend is overwhelmingly toward the relational model. Many hierarchical and network systems still exist and their manufacturers continue to maintain and upgraded them.

Data Structures

Data structures are the bricks and mortar of the database. The data structure allows records to be located, stored, and retrieved and enables movement from one record to another. Chapter 2 examined data structures commonly used by flat-file systems. These include sequential, indexed, hashing, and pointer structures. These basic structures form the foundation for the more complex hierarchical and network database structures discussed next.

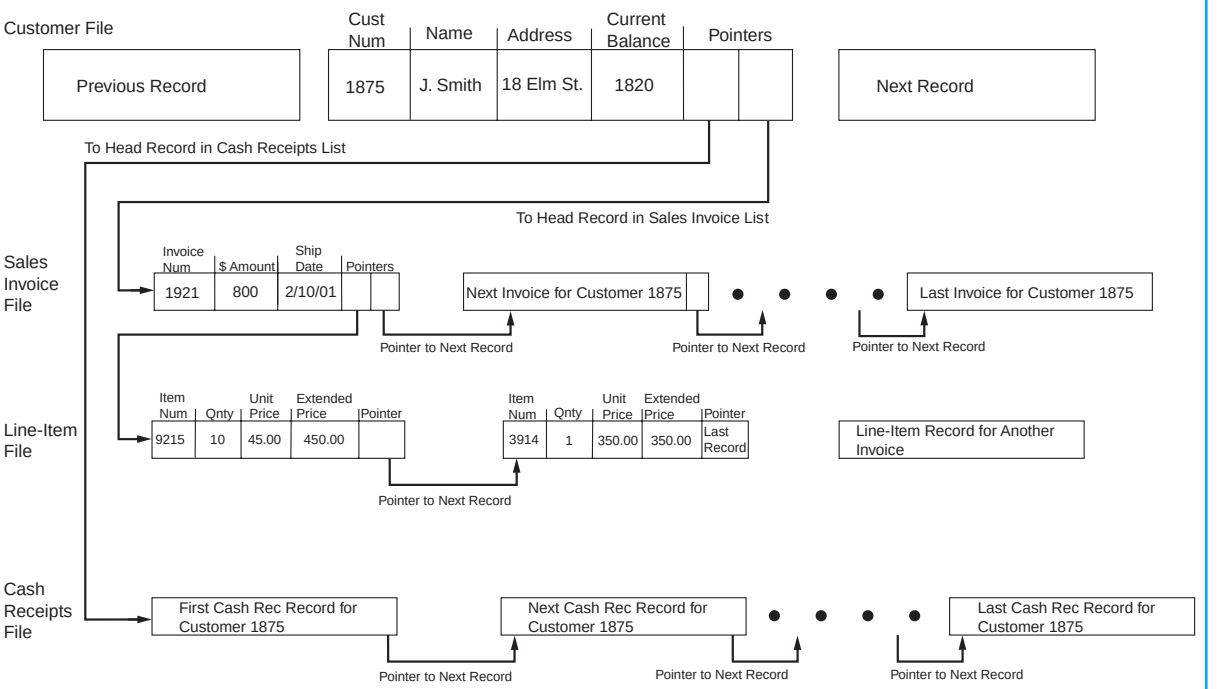
Hierarchical Model Data Structure

Figure 9-32 shows the data structures and linkages between files for the partial database in Figure 9-30. Because the purpose is to illustrate the navigational nature of the data structure, the content of the records has been simplified.

Assume that a user of this system is using a query program to retrieve all the data pertinent to a particular sales invoice (Invoice Number 1921) for a customer John Smith (Account Number 1875). The access method used for this situation is the **hierarchical indexed direct access method (HIDAM).** Under this method, the root segment (customer file) of the database is organized as an indexed file. Lower-level records (for example, sales invoice, invoice line item, and cash receipts records) use pointers in a linked-list arrangement. This allows both efficient processing of the root records for tasks, such as updating AR and billing customers, and direct access of detail records for inquiries.

The access method retrieves the primary key—CUST NUM 1875—entered by the user via the query program and compares this against the index for the root segment. Upon matching the key with the index, it directly accesses John Smith's customer record. Notice the customer record contains only summary information. The current balance figure represents the total dollar amount John Smith owes ($1,820). This is the difference between the sum of all sales to this customer and all cash received in payment on the

FIGURE 9-32 **Linkages between Files in a Hierarchical Database**



account. The supporting details about these transactions are contained in the lower-level sales invoice and cash receipts records. In response to the user's inquiry for data pertaining to Invoice Number 1921, the access method follows the invoice record pointer to the sales invoice file.

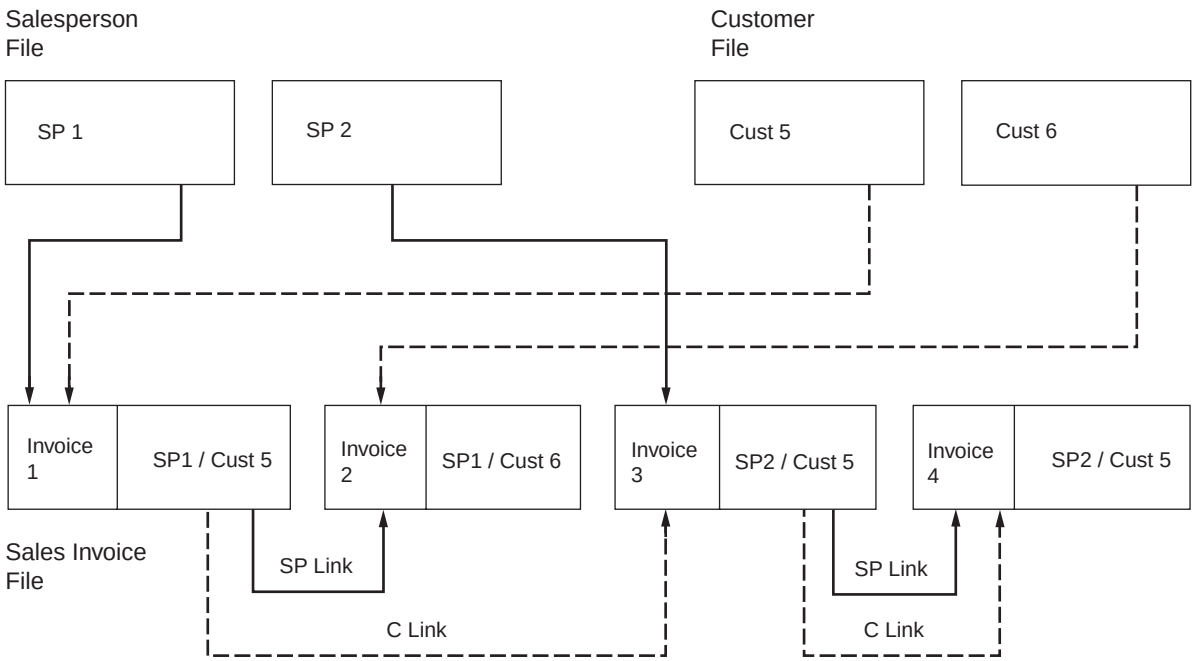
The sales invoice file is a randomly organized file of invoices for all customers arranged in a series of linked lists. The records in each linked list have a common property—they all relate to a specific customer. The first record in the list—the head record—has a pointer to the next in the list (if one exists), which points to the next, and so on. The pointer in the customer record (the level above) directs the access method to the head record in the appropriate linked list. The access method then compares the key value sought (Invoice Number 1921) against each record in the list until it finds a match. The records in the sales invoice file contain only summary information about sales transactions. Pointers in these records identify the supporting detail records (the specific items sold) in the invoice line-item file. The structure of the line-item file is also a linked-list arrangement. Starting with the head record, the access method retrieves the entire list of line items for Invoice Number 1921. In this list there are two records. The sales invoice and line-item records are returned to the user's application for processing.

Network Model Data Structure

As with the hierarchical model, the network model is a navigational database with pointers creating explicit linkages between records. Whereas the hierarchical model allows only one path, the network model supports multiple paths to a particular record. Figure 9-33 shows the linkages for the data structure in Figure 9-31(a).

The structure can be accessed at either of the root-level records (salesperson or customer) by hashing their respective primary keys (SP NUM or CUST NUM) or by reading their addresses from an index. A pointer field in the parent record explicitly defines the path to the child record, as discussed earlier. Notice

FIGURE 9-33 Linkages in a Network Database



the structure of the sales invoice file. In this example, each child now has two parents and contains explicit links to other records that form linked lists related to each parent. For example, Invoice Number 1 is the child of Salesperson Number 1 and Customer Number 5. This record structure has two links to related records. One of these is a salesperson (SP) link to Invoice Number 2. This represents a sale by Salesperson Number 1 to Customer Number 6. The second pointer is the customer (C) link to Invoice Number 3. This represents the second sale to Customer Number 5, which was processed this time by Salesperson Number 2. Under this data structure, management can track and report sales information pertaining to both customers and sales staff.

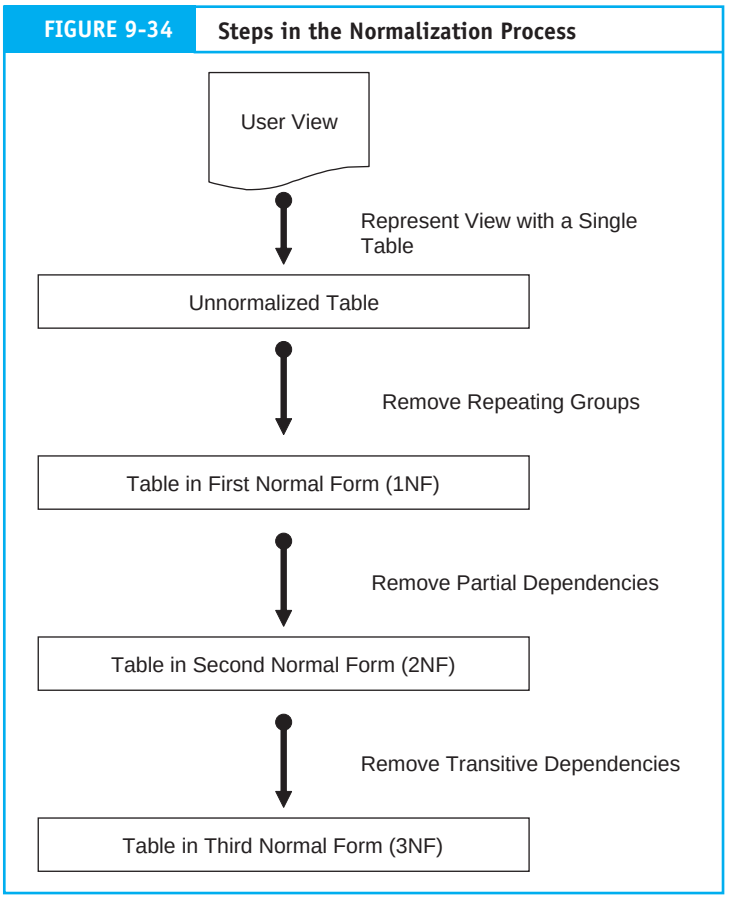
The Data Normalization Process

The database anomalies (described in the chapter) are symptoms of structural problems within tables called dependencies. Specifically, these are known as repeating groups, partial dependencies, and transitive dependencies. The normalization process involves systematically identifying and removing these dependencies from the table(s) under review. Figure 9-34 graphically illustrates the unnormalized table's progression toward 3NF as each type of dependency is resolved. Tables in 3NF will be free of anomalies and will meet two conditions:

1. All nonkey attributes will be wholly and uniquely dependent on (defined by) the primary key.
2. None of the nonkey attributes will be dependent on (defined by) other nonkey attributes.

Design the User View

As Illustrated in Figure 9-34, the process begins with a user view such as an output report, a source document, or an input screen. Images of user views may be prepared using a word processor, a graphics package,



or simply pencil and paper. At this point, the view is merely a pictorial representation of a set of data the user will eventually have when the project is completed. To demonstrate the normalization process, we will use the customer sales invoice and sample data presented in Figure 9-35. The basic issue here is, can we store all the data needed for this view in a single table that meets the two conditions above?

Represent the View as a Single Table

The next step is to represent the view as a single table that contains all of the view attributes. Figure 9-36 presents a single-table structure containing the sample data from Figure 9-35. Since the table contains customer invoices, invoice number (INVOICE NUM) will serve as a logical primary key. Notice the attributes Ex Price and Total Due have been grayed out in Figure 9-36. The values for these attributes may be either stored or calculated. Since Ex Price is the product of two other attributes (Quantity × Unit Price) and Total Due is the sum of all Ex Price values, they both can be calculated from existing stored attributes rather than storing them directly in the database table. To simplify this example, therefore, we will assume that the system will calculate these attributes and they will be ignored from further analysis.

Now that we have a base table to work from, the next few steps in the normalization process involve identifying and, if necessary, eliminating structural dependencies that exist. If dependencies exist, correcting them will involve splitting the original single-table structure into two or more smaller and independent 3NF tables. Each of the structural dependencies and the techniques for identifying and removing them is outlined in the following sections.

FIGURE 9-35 **A User View**

SALES INVOICE

Invoice Number: 16459
Order Date: 09/22/2007
Shipped Date: 09/27/2007
Shipped Via: UPS

Customer Number: **1765**
Customer Name: **ABC Associates**
Street Address: **132 Elm St.**
City: **Bethlehem**
State: **PA**
Telephone Number: **610-555-6721**

Prod Num	Description	Quantity	Unit Price	Ex. Price
r234	Bolt cutter	2	$42.50	$85.00
m456	Gear puller	1	$16.50	$16.50
W62	Electric welder	1	$485.00	$485.00
			Total Due	**$586.50**

Remove Repeating Group Data

The first step in correcting structural dependencies is to determine if the table under review contains repeating groups. **Repeating group** data is the existence of multiple values for a particular attribute in a specific record. For example, the sales invoice in Figure 9-35 contains multiple values for the attributes Prod Num, Description, Quantity, and Unit Price (we ignore Ex Price). These repeating groups represent the transaction details of the invoice. We see repeating group data in many business user views, such as purchase orders, receiving reports, bills of lading, and so on. Relational database theory prohibits the construction of a table in which a single record (a row in the table) represents multiple values for an attribute (a column in the table). To represent repeating group values in a single table, therefore, will require multiple rows as illustrated Figure 9-36. Notice that the invoice attributes, which are common to each occurrence of the repeating group data, will also be represented multiple times. For example, Order Date, Shipped Date, Customer Name, Customer Address, etc., are recorded along with each unique occurrence of Prod Num, Description, Quantity, and Unit Price. To avoid such data redundancy, the repeating group data need to be removed from the table and placed in a separate table. Figure 9-37 shows the resulting tables. One is called Sales Invoice Table, with INVOICE NUM as the primary key. The second table contains the transaction details for the invoice and is called Line Item Table.

Notice that the primary key of the Line Item Table is a **composite key** comprised of two attributes: INVOICE NUM and PROD NUM. Keep in mind that this table will contain the transaction details for our example invoice as well as the transaction details for the invoices for all customers. Relational database

FIGURE 9-36 **Unnormalized Table Supporting User View**

PK

Single-Table Structure for Sales Invoice

Invoice Num	Order Date	Shpd Date	Shpd Via	Total Due	Cust Num	Cust Name	Street Address	City	St	Tele Number	Prod Num	Description	Qunty	Unit Price	Ex Price
16459	09/22/07	09/27/07	UPS	586.50	1765	ABC Assoc	132 Elm St	Bethlehem	PA	**610-555-6721**	r234	Bolt cutter	2	42.50	85.00
16459	09/22/07	09/27/07	UPS	586.50	1765	ABC Assoc	132 Elm St	Bethlehem	PA	**610-555-6721**	m456	Gear puller	1	16.50	16.50
16459	09/22/07	09/27/07	UPS	586.50	1765	ABC Assoc	132 Elm St	Bethlehem	PA	**610-555-6721**	W62	Elec welder	1	485.00	485.00

Redundant Data

Repeating Group Data

theory requires that a table's primary key uniquely identify each record stored in the table. PROD NUM alone cannot do this since a particular product, such as r234 (bolt cutter), may well have been sold to many other customers whose transactions are also in the table. By combining PROD NUM with the INVOICE NUM, however, we can uniquely define each transaction since the table will never contain two occurrences of the same invoice number and product number together.

Remove Partial Dependencies

Next we check to see if the resulting tables contain partial dependencies. A **partial dependency** occurs when one or more nonkey attributes are dependent on (defined by) only part of the primary key, rather than the whole key. This can occur only in tables that have composite (two or more attribute) primary keys. Since the Sales Invoice Table has a single attribute primary key, we can ignore it in this step of the analysis. This table is already in 2NF. The Line Item Table however, needs to be examined further. Figure 9-38 illustrates the partial dependencies in it.

FIGURE 9-37 **Resulting Tables after Removing Repeating Group Data and Calculated Fields**

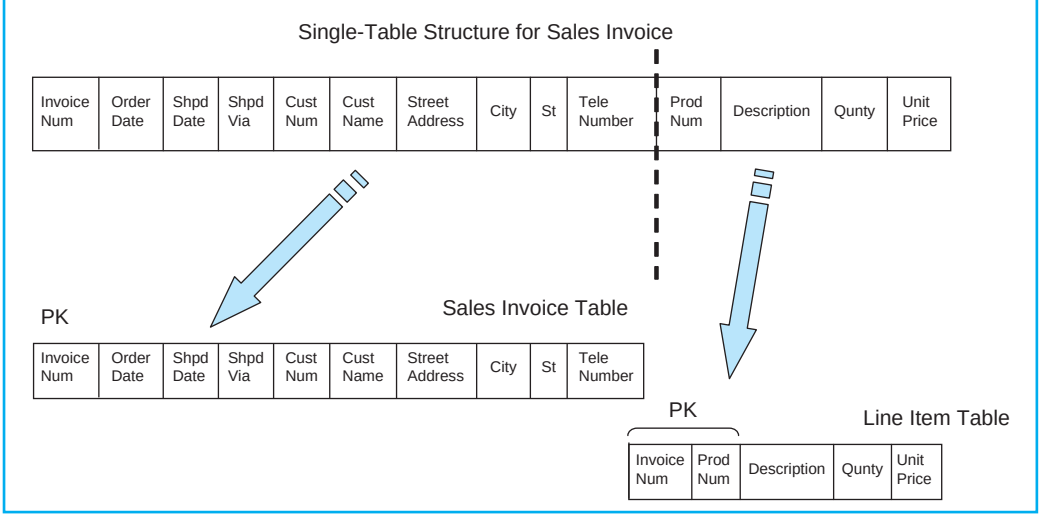
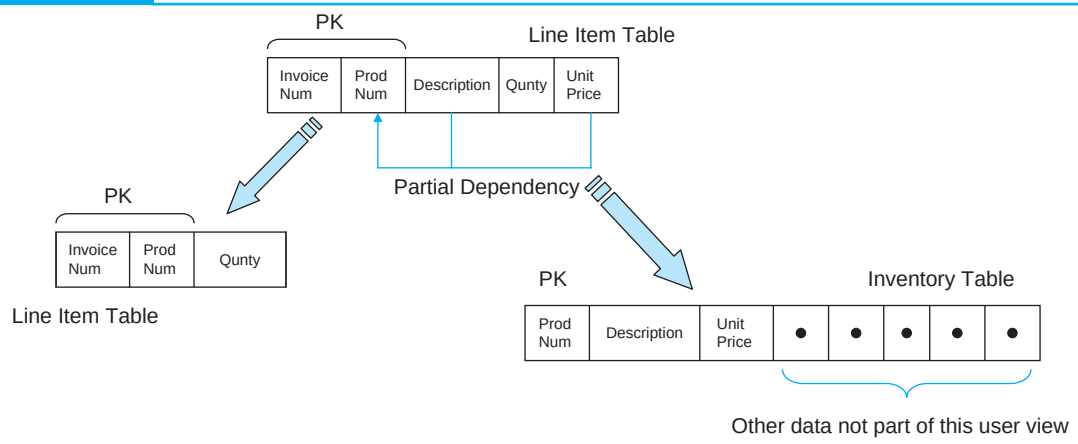


FIGURE 9-38 Resulting Tables after Removing Partial Dependency



In the Line Item Table, INVOICE NUM and PROD NUM together define the quantity sold attribute (Qunty). If we assume, however, that the price charged for r234 is the same for all customers, then the Unit Price attribute is common to all transactions involving product r234. Similarly, the attribute Description is common to all such transactions. These two attributes are not dependent on the Invoice Num component of the composite key. Instead, they are defined by Prod Num and, therefore, only partially rather than wholly dependent on the primary key.

We resolve this by splitting the table into two, as illustrated in Figure 9-38. The resulting Line Item Table is now left with the single nonkey attribute Qunty. Product description and unit price data are placed in a new table called Inventory. Notice that the Inventory table contains additional attributes that do not pertain to this user view. A typical inventory table may contain attributes such as reorder point, quantity on hand, supplier code, warehouse location, and more. This demonstrates how a single table may be used to support many different user views and reminds us that this normalization example pertains to only a small portion of the entire database. We will return to this issue later.

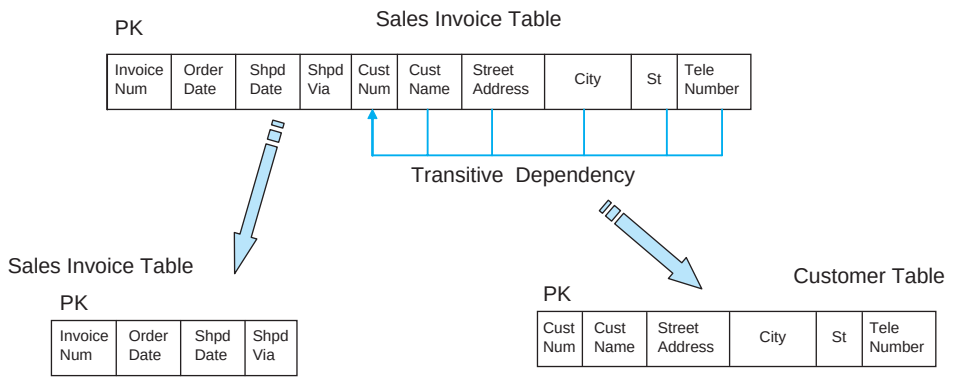
At this point, both of the tables in Figure 9-38 are in 3NF. The Line Item Table's primary key (INVOICE NUM PROD NUM) wholly defines the attribute Qunty. Similarly, in the Inventory Table, the attributes Description and Unit Price are wholly defined by the primary key PROD NUM.

Remove Transitive Dependencies

The final step in resolving structural dependencies is to remove transitive dependencies. A **transitive dependency** occurs in a table where nonkey attributes are dependent on another nonkey attribute and independent of the table's primary key. An example of this is illustrated by the Sales Invoice Table in Figure 9-39. The primary key INVOICE NUM uniquely and wholly defines the economic event that the attributes Order Date, Shpd Date, and Shpd Via represent. The key does not, however, uniquely define the customer attributes. The attributes Cust Name, Street Address, and so on, define an entity (Customer) that is independent of the specific transaction captured by a particular invoice record. For example, assume that during the period the firm had sold to a particular customer on 10 different occasions. This would result in 10 different invoice records stored in the table. Using the current table structure, each of these invoice records would capture the data uniquely related to the respective transaction along with customer data that are common to all 10 transactions. Therefore, the primary key does not uniquely define customer attributes in the table. Indeed, they are independent of it.

We resolve this transitive dependency by splitting out the customer data and placing them in a new table called Customer. The logical key for this table is CUST NUM, which was the nonkey attribute in

FIGURE 9-39 **Resulting Tables after Removing Transitive Dependency**



the former table on which the other nonkey customer attributes were dependent. With this dependency resolved, both the revised Sales Invoice Table and the new Customer Table are in 3NF.

Linking the Normalized Tables

At this point the original single-table structure has been reduced to the four normalized but independent tables presented in Figure 9-40. The tables contain the sample data used in the original single-table structure presented in Figure 9-36. Notice how data redundancy in the original single-table structure has been eliminated from the more efficient structure represented here. To work together, however, these tables need to be linked via foreign keys. This requires first determining the cardinality (degree of association) between the tables and then assigning foreign keys.

Determine Cardinality. In our example the cardinality between the four tables is one-to-many (1:M) as explained below.

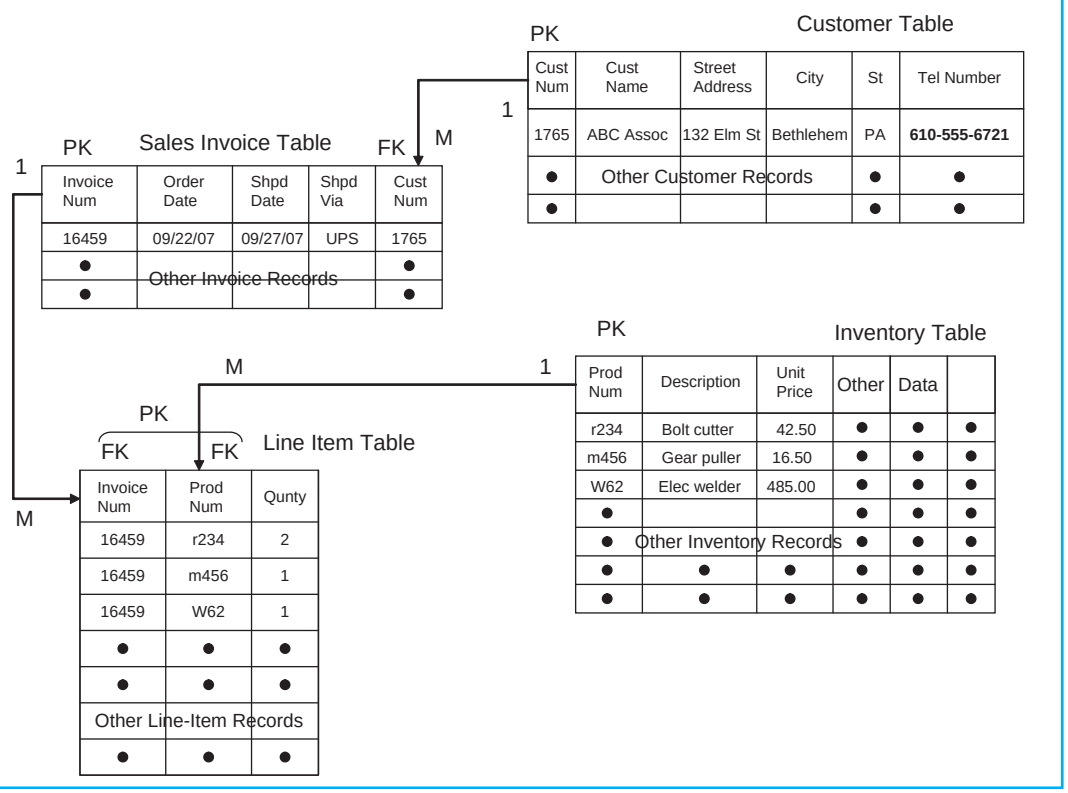
1. Each customer (Customer Table) may be associated with one or many sales events (Sales Invoice Table), but each invoice is for a single customer.
2. Each Sales Invoice record is associated with one or more Line-Item records, but each Line Item is associated with only one Sales Invoice.
3. Each Inventory record is associated with one or more Line Items (a particular product has been sold many times to many customers) but each Line-Item record represents only one inventory item.

Assign Foreign Keys. Rules assigning foreign keys are explained in detail in the chapter. When linking in a 1:M relation as depicted in Figure 9-40, the rule is to take the primary key from the table on the 1 side of the relation and embed it as a foreign key in the table of the M side. Notice that in the relations between the Line Item Table, the Invoice Table, and the Inventory Table, this is already the case because of the Line Item Table's composite key. The Sales Invoice Table, however, needs to be modified to include CUST NUM as the FOREIGN KEY, which links it to the Customer Table.

Producing the User View from the Normalized Tables

After these tables have been created within the DBMS, they will be populated with data from several sources. For example, customer services will add customers' data to the Customer table, inventory control will enter product values into the Inventory table, and the Sales Invoice and Line Item tables will be populated by sales transactions data from the Sales Order process. The following steps describe how the batch process might produce the actual invoices.

FIGURE 9-40 Linkages between Normalized Tables



1. A computer program reads the Sales Invoice Table. We will assume that the first record read is invoice number 16459 (our sample record). The record attributes are stored in memory.
2. The program then reads the foreign key Cust Num and searches the Customer Table for a record with the primary key value 1765. The customer record attributes are then stored in memory.
3. The computer program then reads the primary key Invoice Num and searches the Line Item Table for all occurrences of records whose Invoice Num component of the primary key has a value of 16459. It locates three such records and stores them in memory.
4. The program next reads the Prod Num component of the three Line-Item records and one by one searches the Inventory File for each occurrence. The Description and Unit Price attributes of each located record is stored in memory.
5. The program then calculates the Ex Price attribute for each item and sums these to obtain the Total Due attribute.
6. At this point all the attributes needed to produce the original user view are in memory. They are formatted and sent to printing.
7. The computer then clears memory, reads the next record from the Invoice table, and repeats the steps above until all invoices have been processed.

Key Terms

access method (438)
anomalies (448)
association (444)
attributes (444)
cardinality (444)
centralized database (464)
composite key (477)
concurrency control (469)
currency of information (431)
data attribute (431)
data currency (464)
data definition language (DDL) (436)
data dictionary (441)
data manipulation language (DML) (438)
data model (444)
data redundancy (431)
data storage (431)
data structures (473)
data updating (431)
database administrator (DBA) (438)
database lockout (466)
database management system (DBMS) (433)
deadlock (467)
deletion anomaly (450)
distributed data processing (DDP) (464)
distributed databases (466)
entity (443)
entity relationship (ER) diagram (444)
first normal form (1NF) (448)
flat file (431)
foreign keys (447)
hierarchical indexed direct access method (HIDAM) (473)
hierarchical model (434)
indexed sequential file (441)
insertion anomaly (450)
internal view (436)
inverted list (441)
join (443)
navigational model (434)
network model (434)
occurrence (444)
partial dependency (450)
partitioned database (466)
physical database (441)
primary key (447)
project (443)
relational model (434)
repeating group (459)
replicated databases (468)
restrict (443)
schema (conceptual view) (436)
second normal form (2NF) (448)
structured model (434)
structured query language (SQL) (438)
task-data dependency (431)
temporary inconsistency (465)
third normal form (3NF) (448)
transitive dependency (479)
update anomaly (449)
user view (subschema) (437)
users (435)
view integration (464)
view modeling (454)

Review Questions

1. Give five general duties of the database administrator.
2. What are the four primary elements of the database environment?
3. How are the network and hierarchical models different?
4. What flat-file data management problems are solved as a result of using the database concept?
5. What are four ways in which database management systems provide a controlled environment to manage user access and the data resources?
6. Explain the relationship between the three levels of the data definition language. As a user, which level would you be most interested in?
7. What is a primary key?

8. What is a foreign key?
9. What is a data dictionary and what purpose does it serve?
10. Give an application for a partitioned database.
11. What is an entity?
12. Give an application for a replicated database.
13. Discuss and give an example of the following types of associations: (1:0,1), (1:1), (1:M), and (M:M).
14. Distinguish between association and cardinality.
15. Explain how a separate linking table works in a many-to-many association.
16. What are the four characteristics of properly designed relational database tables?
17. What do the relational features restrict, project, and join mean?
18. What are the conditions for third normal form (3NF)?
19. Explain how the SELECT and WHERE commands help a user to view the necessary data from multiple database files (tables).
20. What is a data model?
21. How can a poorly designed database result in unintentional loss of critical records?
22. What is a user view?
23. Does a user view always require multiple tables to support it? Explain.
24. What two conditions must valid entities meet?
25. Can two different entities have the same defining attributes? Explain.

Discussion Questions

1. In the flat-file data management environment, users are said to own their data files. What is meant by the ownership concept?
2. Discuss the potential aggravations you might face as a student as a result of your university using a flat-file data management environment, that is, different files for the registrar, library, parking, and so on.
3. Discuss why control procedures over access to the data resource become more crucial under the database approach than in the flat-file environment. What role does the DBMS play in helping to control the database environment?
4. What is the relationship between a database table and a user view?
5. Explain how linkages between relational tables are accomplished.
6. Explain the purpose of an ER diagram in database design.
7. SQL has been said to place power in the hands of the user. What does this statement mean?
8. Discuss the importance of the role of the database administrator. In the flat-file environment, why is such a role not necessary? What tasks does the DBA perform?
9. As users determine new computer application needs, requests must be sent to both the system programmers and the DBA. Why is it important that these two groups perform separate functions, and what are these functions?
10. Why is a separate link table required when an M:M association exits between related tables?
11. As an accountant, why would you need to be familiar with data normalization techniques?
12. How does a database lockout contribute to financial data integrity?
13. How does concurrency control contribute to financial data integrity?
14. In a relational database environment, certain accounting records (for example, journals, subsidiary ledgers, and event general ledger accounts) may not exist. How is this possible?
15. Explain how to link tables in a 1:1 association. Why may this be different in a 1:0,1 association?
16. Discuss the accounting implications of the update, insertion, and deletion anomalies associated with improperly normalized tables.
17. Give three examples that illustrate how cardinality reflects an organization's underlying business rules.

18. Discuss the key factors to consider in determining how to partition a corporate database.
19. Distinguish between a database lockout and a deadlock.
20. Replicated databases create considerable data redundancy, which is in conflict with the database concept. Explain the justification of this approach.

Multiple-Choice Questions

1. The data attributes that a particular user has permission to access are defined by the
   a. operating system view.
   b. systems design view.
   c. database schema.
   d. user view.
   e. application program.
2. The database approach has several unique characteristics not found in traditional (flat-file) systems. Which of the following statements does not apply to the database model?
   a. Database systems have data independence; that is, the data and the programs are maintained separately except during processing.
   b. Database systems contain a data-definition language that helps describe each schema and subschema.
   c. The database administrator is the part of the software package that instructs the operating aspects of the program when data are retrieved.
   d. A primary goal of database systems is to minimize data redundancy.
   e. Database systems provide increased accessibility to data and flexibility in its usage.
3. One of the first steps in the creation of a relational database is to
   a. integrate accounting and nonfinancial data.
   b. plan for increased secondary storage capacity.
   c. order data-mining software that will facilitate data retrieval.
   d. create a data model of the key entities in the system.
   e. construct the physical user view using SQL.
4. Database currency is achieved by
   a. implementing partitioned databases at remote sites.
   b. employing data-cleansing techniques.
   c. ensuring that the database is secure from accidental entry.
   d. an external auditor's reconciliation of reports from multiple sites.
   e. a database lockout that prevents multiple simultaneous access.
5. The installation of a database management system is likely to have the least impact on
   a. data redundancy.
   b. entity-wide sharing of common data.
   c. exclusive ownership of data.
   d. the logic needed to solve a problem in an application program.
   e. the internal controls over data access.
6. The functions of a database administrator are
   a. database planning, data input preparation, and database design.
   b. data input preparation, database design, and database operation.
   c. database design, database operation, and equipment operations.
   d. database design, database implementation, and database planning.
   e. database operations, database maintenance, and data input preparation.
7. A relational database system contains the following inventory data: part number, description, quantity-on-hand, and reorder point. These individual items are called
   a. attributes.
   b. relations.

c. associations.
d. occurrences.

8. Which of the following is a characteristic of a relational database system?
   a. All data within the system are shared by all users to facilitate integration.
   b. Database processing follows explicit links that are contained within the records.
   c. User views limit access to the database.
   d. Transaction processing and data warehousing systems share a common database.
9. Partitioned databases are most effective when
   a. users in the system need to share common data.
   b. primary users of the data are clearly identifiable.
   c. read-only access is needed at each site.
   d. all of the above.
10. Database entities
   a. may contain zero or many occurrences.
   b. are represented as verbs in an ER diagram.
   c. may represent both physical assets and intangible phenomena.
   d. are often defined by common attributes that also define other entities.
   e. are unique to a specific user view.
11. A transitive dependency
   a. is a database condition that is resolved through special monitoring software.
   b. is a name given to one of the three anomalies that result from unnormalized database tables.
   c. can exist only in a table with a composite primary key.
   d. cannot exist in tables that are normalized at the 2NF level.
   e. is none of the above.
12. A partial dependency
   a. is the result of simultaneous user requests for the same data in a partitioned database environment.
   b. is a name given to one of the three anomalies that result from unnormalized database tables.
   c. can exist only in a table with a composite primary key.
   d. may exist in tables that are normalized at the 2NF level.
   e. is none of the above.
13. Repeating group data
   a. is a form of data redundancy common to replicated databases in a distributed database environment.
   b. is a name given to one of the three anomalies that result from unnormalized database tables.
   c. can exist only in a table with a composite primary key.
   d. cannot exist in tables that are normalized at the 2NF level.
   e. is none of the above.
14. The database model most likely to be used in the development of a modern (not legacy) system is
   a. hierarchical
   b. structured
   c. relational
   d. network
   e. navigational
15. Typical DBMS features include all of the following EXCEPT
   a. database interface design and development.
   b. backup and recovery.
   c. program development.
   d. database access.
   e. all of these are typical DBMS features.
16. Which of the following is least likely to be an attribute of an employee table in a normalized database?
   a. employee name
   b. employee address
   c. employee number
   d. employee supervisor's name
   e. all of these would be attributes of the employee table in a normalized database.
17. The advantages to using a partitioned database approach include all of the following EXCEPT:
   a. the possibility for the deadlock phenomenon is reduced.
   b. user control is increased.

c. transaction processing time is decreased.
d. the potential for wide-scale disaster is reduced.
e. these are all advantages of partitioned databases.

18. Of the following, select the attribute that would be the best primary key in an inventory table.
a. ITEM NAME
b. ITEM LOCATION
c. ITEM COST
d. ITEM NUMBER
e. ITEM SUPPLIER

Problems

1. DBMS versus Flat-File Processing

Werner Manufacturing Corporation has a flat-file processing system. The information-processing facility is very large. Different applications, such as order processing, production planning, inventory management, accounting systems, payroll, and marketing systems, use separate tape and disk files. The corporation has recently hired a consulting firm to investigate the possibility of switching to a database management system. Prepare a memo to the top management team at Werner explaining the advantages of a DBMS. Also, discuss the necessity of a database administrator and the job functions this person would perform.

2. Access Methods

For each of the following file processing operations, indicate whether a sequential file, indexed random file, indexed sequential access method (ISAM), hashing, or pointer structure works the best. You may choose as many as you wish for each step. Also, indicate which would perform the least optimally.

a. Retrieve a record from the file based upon its primary key value.
b. Update a record in the file.
c. Read a complete file of records.
d. Find the next record in a file.
e. Insert a record into a file.
f. Delete a record from a file.
g. Scan a file for records with secondary keys.

3. Database Design

Design a relational database for a video rental store. The store, which rents only DVDs and has no sales other than DVD rentals, has approximately 5,000 customers and approximately 1,200 DVD titles. There are 25 suppliers of DVDs. Business rules include: (1) each DVD title may have many copies; (2) customers use their telephone numbers to uniquely identify themselves for video rentals; (3) a telephone number may apply to anyone in the household; (4) customers may rent more than one copy of any title, but, obviously, a copy of a title can only be rented by one customer at a time; and (5) suppliers may provide many titles and a DVD may be acquired from several different suppliers. Design the necessary database tables. State the primary key for each table as well as all other necessary attributes. Make certain the tables are in third normal form. Mark the attribute you have selected as the primary key (PK) as well as the attributes that serve as foreign keys (FK).

4. Database Design

Sears Roebuck, the most well-known and oldest mail-order retailer in the country, discontinued its mail-order operations in 1993. Other mail-order marketers are using information systems to trim printing and

postage costs of their catalogs. They also want to more effectively target their customers. Explain how an appropriately designed coding system for inventory items (see Chapter 8), incorporated in a relational database system with SQL capabilities, could allow more cost-efficient and effective mail-order operations. Sketch the necessary database table structure.

5. Database Deadlock

How is a lockout different from a deadlock? Give an accounting example to illustrate why a database lockout is necessary and how a deadlock can occur. Provide meaningful table names in your example.

6. Structured Query Language

A vehicle rental company has a database that includes the following tables and attributes within those tables:

Customer Table

Customer number (PK); customer name, customer address, customer phone number, customer credit card number

Inventory Table

Inventory number (PK); item description; item cost; item rental price; number of days for which an item can be rented

Rental Form Table

Rental Form number (PK); rental date; customer number (FK)

Rental Line Item Table

Rental Form number (PK); inventory number (PK); due date

Required:

Using the SQL commands given in this chapter, write the code that would be necessary to generate a report of each customer with rented items that are past due. Customers can rent more than one item at a time, and items may have different rental periods, so this report should only include overdue items. (You may use the word *today* to signify the current date.)

7. Distributed Databases

The XYZ Company is a geographically distributed organization with several sites around the country. Users at these sites need rapid access to common data for read-only purposes. Which distributed database method is best under these circumstances? Explain your reasoning.

8. Distributed Databases

The ABC Company is a geographically distributed organization with several sites around the country. Users at these sites need rapid access to data for transaction processing purposes. The sites are autonomous; they do not share the same customers, products, or suppliers. Which distributed database method is best under these circumstances? Explain your reasoning.

9. Normalization of Data

A table of data for a library is found on the following page. Normalize this data into the third normal form, preparing it for use in a relational database environment. The library's computer is programmed to compute the due date to be 14 days after the checkout date. Document the steps necessary to normalize the data similar to the procedures found in the chapter. Index any fields necessary and show how the databases are related.

10. Normalization of Data

A college bookstore maintains all of its records on index cards. This has proven to be a very tedious chore, and the manager has asked you to design a database that will simplify this task. The index cards are preprinted and contain the following labels:

Title:
Author(s):
Edition:
Publisher:
Publisher Address:
Publisher Phone #:
Cost of Text:
Selling Price:

Problem 9: Normalization of Data

STUDENT ID NUMBER	STUDENT FIRST NAME	STUDENT LAST NAME	NUMBER OF BOOKS OUT	BOOK CALL NO	BOOK TITLE	DATE OUT	DUE DATE
678-98-4567	Amy	Baker	4	hf351.j6	Avalanches	09-02-07	09-16-07
678-98-4567	Amy	Baker	4	hf878.k3	Tornadoes	09-02-07	09-16-07
244-23-2348	Ramesh	Sunder	1	i835.123	Politics	09-02-07	09-16-07
398-34-8793	James	Talley	3	k987.d98	Sports	09-02-07	09-16-07
398-34-8793	James	Talley	3	d879.39	Legal Rights	09-02-07	09-16-07
678-98-4567	Amy	Baker	4	p987.t87	Earthquakes	09-03-07	09-17-07
244-23-2348	Ramesh	Sunder	1	q875.i76	Past Heroes	09-03-07	09-17-07

Number on Hand:
Professor:
Class:
Semester:
Year:
Professor Phone #:

Using your understanding of the business rules that exist in most colleges and for most college bookstores, prepare the base tables (in third normal form) that will be required to translate the index card information into a working database for the bookstore.

11. Normalization of Data

Prepare the base tables, in third normal form, needed to produce the user view below.

12. Normalization of Data

Prepare the base tables, in third normal form, needed to produce the user view on the following page.

13. Normalization of Data

Prepare the 3NF base tables needed to produce the sales report view on page 489.

14. Normalization of Data—Purchase Order

Acme Plywood Company uses the purchase order on page 490.

Acme business rules:

1. Each vendor may supply many items; an item is supplied by only one vendor.
2. A purchase order may list many items; an item may be listed on many purchase orders.
3. An employee may complete several purchase orders, but only one employee may fill out an individual PO.

Prepare the 3FN base tables needed to produce this purchase order.

Problem 11: Normalization of Data

USER VIEW

Part Num	Description	QOH	Reorder Point	EOQ	Unit Cost	Ven Num	Ven Name	Ven Address	Tel
132	Bolt	100	50	1000	1.50	987	ABC Co.	654 Elm St	555 5498
143	Screw	59	10	100	1.75	987	ABC Co.	654 Elm St	555 5498
760	Nut	80	20	500	2.00	742	XYZ Co.	510 Smit	555 8921
982	Nail	100	50	800	1.00	987	ABC Co.	654 Elm St	555 5498

Problem 12:
Normalization of Data

USER VIEW

Part Num	Description	QOH	Reorder Point	EOQ	Unit Cost	Ven Num	Ven Name	Ven Address	Tel
132	BOLT	100	50	1000	1.50	987	ABC Co.	654 Elm St	555 5498
					1.55	750	RST Co.	3415 8th St	555 3421
					1.45	742	XYZ Co.	510 Smit	555 8921
982	NAIL	100	50	800	1.00	987	ABC Co.	654 Elm St	555 5498
					1.10	742	XYZ Co.	510 Smit	555 8921
					1.00	549	LMN Co.	18 Oak St	555 9987

Problem 13:
Normalization of Data

Sales Report

Customer Number: 19321
Customer Name : Jon Smith
Address : 520 Main St.,City

Invoice Num	Date	Invoice Total	Part Num	Quantity	Unit Price	Ext'd Price
12390	11/11/07	$850	2	5	$20	$100
			1	10	50	500
			3	25	10	250
12912	11/21/07	$300	4	10	$30	$300

Customer Total: $1,150

** * *** * *** * *** * *** * *** * ** *

Customer Number: 19322
Customer Name : Mary Smith
Address : 2289 Elm St., City

Invoice Num	Date	Invoice Total	Part Num	Quantity	Unit Price	Ext'd Price
12421	11/13/07	$1,000	6	10	$20	$200
			1	2	50	100
			5	7	100	700
12901	11/20/07	$500	4	10	$30	$300
			2	10	20	200

Customer Total: $1,500

** * *** * *** * *** * *** * *** * ** *

Next Customer

•
•
•

Next Customer

Problem 14:
Normalization of Data

Purchase Order

Acme Plywood Co.
1234 West Ave.
Somewhere, OH 000000

P.O. #
Date: __/__/__

Vendor: _____
_____
_____

Ship Via: _____ Please refer this P.O. number on all correspondence.

Prepared by: _____

Item #	Description	Quantity	Cost	Extension

15. Table Linking

Refer to text in Problem 15 on page 492.

16. Defining Entities and Data Modeling—Payroll

Employees at the Sagerod manufacturing company record their hours worked on paper time cards that are inserted into a time clock machine at the beginning and end of each shift. On Fridays, the supervisor collects the time cards, reviews and signs them, and sends them to the payroll clerk. The clerk calculates the pay for each employee and updates the individual earnings records. The clerk then prepares a paycheck for each employee and records it in the check register. Based on these records, the clerk prepares a payroll register, which is sent with the paychecks to the cash disbursements clerk. The clerk reviews the payroll register, updates the cash disbursements journal to record the total payroll, and prepares a check for the total payroll, which is deposited into the imprest account. The clerk then signs the paychecks and distributes them to the employees.

Required:

Assume that this manual system is to be automated using a relational database system. Perform the following tasks. You may need to make assumptions about how certain automated activities will be performed.

a. List all candidate entities in the procedures described.
b. Identify the valid entities and explain why the rejected entities should not be modeled.
c. Create a data model of the process showing entity associations.

17. Defining Entities and Data Modeling—Purchases Procedures

The business rules that constitute the purchases system for the Safe Buy Grocery Stores chain are similar at all the store locations.

The purchase manager at each location is responsible for selecting his or her local suppliers. If the manager needs a product, he or she chooses a supplier. Each store follows the steps described below.

1. The purchasing function begins with sales representatives from suppliers periodically observing the shelves and displays at each location and recognizing the need to restock inventory. Inventory declines by direct sales to the customers or by spoilage of perishable goods. In addition, the supplier's sales representatives review obsolescence reports that the purchase manager prepares. These reports identify slow-moving and dated products that are deemed unsalable at a particular location. These products are returned to the supplier and replaced with more successful products. The sales representatives prepare a hard-copy purchase requisition and meet with the purchase managers of the individual store locations. Together the sales representative and the purchase manager create a purchase order defining the products, the quantity, and the delivery date.
2. At the intended delivery date, Safe Buy Grocery Stores receive the goods from the suppliers. Goods received are unloaded from the delivery trucks and stocked on the shelves and displays by part-time employees.
3. The unloading personnel create a receiving report. Each day a receiving report summary is prepared and sent to the purchase managers for review.
4. The supplier subsequently submits an invoice to the AP department clerk, who creates an invoice record. The clerk reconciles the invoice against the receiving report and purchase order and then creates a payment obligation to be paid at a future date, depending on the terms of trade.
5. On the due date, a check is automatically prepared and sent to the supplier, and the payment is recorded in the check register. At the end of each day, a payment summary is sent to the purchase managers for review.

Required:

Assume that the manual system above is to be automated using a relational database system. Perform the following tasks. You may need to make assumptions about how certain automated activities will be performed.

a. List all candidate entities in the procedures described.
b. Identify the valid entities and explain why the rejected entities should not be modeled.
c. Create a data model of the process showing entity associations.
d. Create a fully attributed model by adding primary keys, foreign keys, and data attributes. Normalize the model.

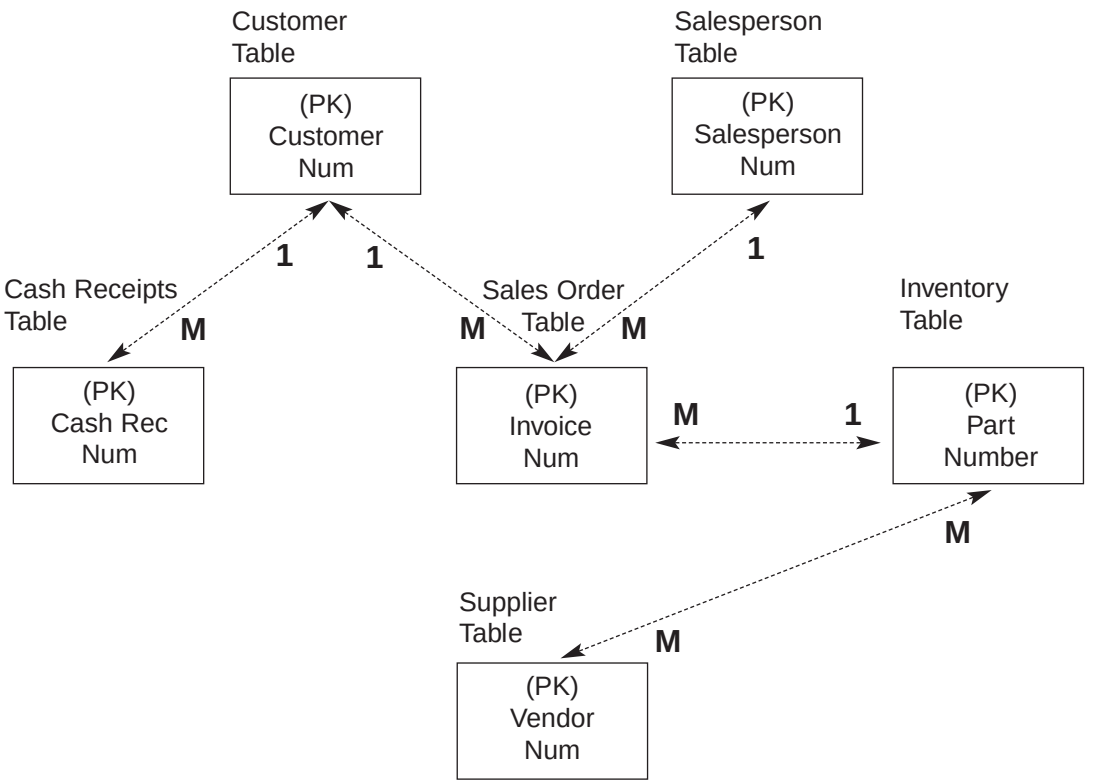
18. Defining Entities and Data Modeling—Fixed Asset Procedures

The business rules that constitute the fixed asset procedures for the Safe Buy Grocery Stores chain are similar at all the store locations. The store manager at each location is responsible for identifying needed fixed assets and for selecting the vendor. Freezers, refrigerators, delivery vans, and store shelving are examples of fixed asset purchases. Once the need has been identified, each store follows the procedures described next.

The manager creates a purchase order, which is sent to the supplier. The supplier delivers the asset to the receiving clerk, who prepares a receiving report. Each week the fixed asset department clerk reviews the fixed asset receiving report summary and creates a fixed asset inventory record for each receipt. The fixed asset clerk maintains the inventory records and depreciation schedules. The vendor subsequently submits an invoice to the AP department clerk, who creates an invoice record. The clerk reconciles the invoice

Problem 15:
Table Linking

Several related tables with their primary keys (PK) are shown below. Place the foreign key(s) in the tables to link them according to the associations shown (e.g., 1:M and M:M). Create any new table(s) that may be needed.



against the receiving report and purchase order and then creates a payment obligation to be paid at a future date, depending on the terms of trade. On the due date, a check is automatically prepared and sent to the vendor, and the payment is recorded in the check register. At the end of each day, a payment summary is sent to the AP manager for review.

Required:
Assume that the manual system above is to be automated using a relational database system. Perform the following tasks. You may need to make assumptions about how certain automated activities will be performed.

a. List all candidate entities in the procedures described.
b. Identify the valid entities and explain why the rejected entities should not be modeled.
c. Create a data model of the process showing entity associations.
d. Create a fully attributed model by adding primary keys, foreign keys, and data attributes. Normalize the model.

19. Defining Entities and Data Modeling—Business Rules

Given the following business rules, construct an ER diagram so each rule is captured for the database. Presume each rule is to be treated individually. Construct an ER diagram for each rule.

a. A retail sales company prepares sales orders for its customers' purchases. A customer can make many purchases, but a sales order is written for a single customer.

b. A retail sales company orders inventory using a purchase order. An inventory item may be ordered many times, and a purchase order may be created for more than one inventory item.

c. A company that sells antique cars prepares a sales order for each car sold. The inventory for this company consists of unique automobiles, and only one of these automobiles may be listed on a sales order.

d. A grocery store identifies returning customers via a plastic card that the clerk scans at the time of each purchase. The purpose of this card is to track inventory and to maintain a database of customers and their purchases. Obviously, a customer may purchase an unlimited number of items from the grocery store. Items are unique only by a UPC code, and each UPC code may be associated with many different customers.

e. A video rental store uniquely identifies each of its inventory items so customers can rent a movie and return the movie via a drop box and the store can identify which copy of the movie was rented and returned. A customer is allowed to rent up to six movies at a time, but a copy of a movie can only be rented by one customer at a time.

20. Comprehensive Case

(Prepared by Katie Daley and Gail Freeston, Lehigh University)

D&F is a distributor of CDs and cassettes that offers benefits such as discount prices and an introductory offer of 10 CDs or cassettes for a penny (not including the shipping and handling costs). Its primary target customers are college students; its main marketing strategy is constant deals to club members. The company's main competitors in the industry are BMG and Columbia House; both offer similar promotions.

D&F started in 1993 with an office in Harrisburg, Pennsylvania, initially targeting college students in the surrounding area. The company realized there was a high demand for discounted music merchandise and the convenience of mail delivery within universities. After its second year, with a constant increase in customer orders, D&F relocated to Philadelphia, where it was located near more colleges and universities. The move has had a positive effect on net profits and demand, supporting the decision to continue the growth of the company. D&F recently expanded its facility to be able to fulfill a higher demand for its services. Its customer base ranges from areas as close as Villanova University to as far as Boston College. As of 2007, there were 103 employees. Their prior year's gross sales were $125 million.

D&F's market share is on the rise, but is not yet comparable to the magnitude of BMG and Columbia House. However, the corporation's goals for the upcoming years include establishing itself as an industry player through increased customer satisfaction and loyalty. D&F is also considering the installation of a new information processing system. This system will reengineer their current business functions by reducing loopholes in their internal control problems.

D&F receives CDs and cassettes from various wholesale suppliers and music store chains, totaling 32 suppliers nationwide. The office has its own warehouse, stores its own merchandise, and is responsible for replenishing the inventory. D&F has had no substantial problems in the past with their suppliers. On the other hand, it has

encountered problems with excess inventory, stock-outs, and discrepancies with inventory records.

Revenue Cycle

Becoming a member of D&F Music Club involves calling the toll-free number and speaking with a sales representative, who establishes a new customer account. A customer's account record contains his or her name, address, phone number, previous orders he or she made with the company, and a sequentially assigned unique customer account number.

Customers place orders by phone with a sales representative, who prepares a sales order record. John, in the billing department, reviews the sales orders, adds prices and shipping charges, and prints a copy (invoice) that is sent to the customer. John then adds a record to the sales journal to record the sale.

Chris, a warehouse employee, verifies the information on the sales order, picks the goods, prints the packing slip, and updates the inventory subsidiary ledger. Chris then prepares the bill of lading for the carrier. The goods are then shipped.

Sandy in AR updates the customer accounts and general ledger control accounts. When customers make a payment on account, they send both the remittance advice (that was attached to the invoice) and a check with their account number on it. Scott, a mail room clerk, opens all the cash receipts. He separates the check and remittance advice and prepares a remittance list, which, along with the checks, is sent to the cash receipts department.

Laura, the cash receipts clerk, reconciles the checks with the remittance, updates the customer's account and the general ledger, and then deposits the checks in the bank. She sends the deposit slip to Sandy in the accounting department.

Upon receiving the bank receipt, Sandy files it and updates the cash receipts journal to record the amount deposited. Upon the receipt of the CDs or cassettes ordered, the customer has a 15-day trial period. If, at the end of that period, he or she sends a payment, it is understood that the goods have been accepted. If, on the other hand, the customer is dissatisfied with the product for any reason, he or she can return it to D&F Music Club at no charge. However, to return the CD or cassette, the customer must call the company to obtain an authorization number. When the goods arrive, Chris prepares the return record and updates the inventory subsidiary ledger. Printed copies of the return record are sent to John and Sandy. John reviews the return record and updates the sales journal. Sandy credits the customer's account and updates the general ledger to reverse the transaction.

Expenditure Cycle

The purchases system and the cash disbursements system comprise D&F Music Club's expenditure cycle. The three departments within the purchasing system are the warehouse, purchasing, and accounting. The purchasing function begins in the warehouse, which stores the inventory of CDs and cassettes. Jim, the warehouse manager, compares inventory records with the various demand forecasts of each week, which the market research analyst teams provide, to determine the necessary orders to make. At the end of the week, Jim prepares the purchase requisition record.

Sara, the purchasing clerk, reviews the purchase requisitions, selects the suppliers, and prepares the purchase orders. Copies of the purchase orders are sent to the supplier and accounting.

When the shipment arrives, Chris, the warehouse clerk, working from a blind copy of the purchase order, counts and inspects the goods for damage. He then prepares a receiving report and updates the inventory records.

Upon receipt of the supplier's invoice, Diana, the accounting clerk, compares it to

the respective purchase order and receiving report. If the invoice is accurate, Diana creates an AP record, sets a due date to be paid, and updates general ledger accounts.

On the due date, Evan, the cash disbursements clerk, closes the AP record, cuts a check, and sends it sent to the supplier. He then updates the check register and the general ledger.

Required:

Assume that the manual system above is to be automated using a relational database system. Perform the following tasks. You may need to make assumptions about how certain automated activities will be performed.

a. List all candidate entities in the procedures described.
b. Identify the valid entities and explain why the rejected entities should not be modeled.
c. Create a data model of the processes showing entity associations.
d. Create a fully attributed model by adding primary keys, foreign keys, and data attributes. Normalize the model.
e. Prepare a data flow diagram of the system showing the data stores.