[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html" \l "header)

**Chapter**

**9**

**Data Design**

* **9.1**[Data Design Concepts](javascript://)
  + **9.1.1**[Data Structures](javascript://)
  + **9.1.2**[Mario and Danica: A Data Design Example](javascript://)
  + **9.1.3**[Database Management Systems](javascript://)
* **9.2**[DBMS Components](javascript://)
  + **9.2.1**[Interfaces for Users, Database Administrators, and Related Systems](javascript://)
  + **9.2.2**[Schema](javascript://)
  + **9.2.3**[Physical Data Repository](javascript://)
* **9.3**[Web-Based Design](javascript://)
* **9.4**[Data Design Terms](javascript://)
  + **9.4.1**[Definitions](javascript://)
  + **9.4.2**[Key Fields](javascript://)
  + **9.4.3**[Referential Integrity](javascript://)
* **9.5**[Entity-Relationship Diagrams](javascript://)
  + **9.5.1**[Drawing an ERD](javascript://)
  + **9.5.2**[Types of Relationships](javascript://)
  + **9.5.3**[Cardinality](javascript://)
* **9.6**[Data Normalization](javascript://)
  + **9.6.1**[Standard Notation Format](javascript://)
  + **9.6.2**[First Normal Form](javascript://)
  + **9.6.3**[Second Normal Form](javascript://)
  + **9.6.4**[Third Normal Form](javascript://)
  + **9.6.5**[Two Real-World Examples](javascript://)
* **9.7**[Codes](javascript://)
  + **9.7.1**[Overview of Codes](javascript://)
  + **9.7.2**[Types of Codes](javascript://)
  + **9.7.3**[Designing Codes](javascript://)
* **9.8**[Data Storage and Access](javascript://)
  + **9.8.1**[Tools and Techniques](javascript://)
  + **9.8.2**[Logical Versus Physical Storage](javascript://)
  + **9.8.3**[Data Coding](javascript://)
* **9.9**[Data Control](javascript://)
* **9.10**[Summary](javascript://)
* [Chapter Review](javascript://)
  + [Key Terms](javascript://)

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**9.1**Data Design Concepts

Systems analysts must understand basic data design concepts, including data structures and the evolution of the relational database model.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

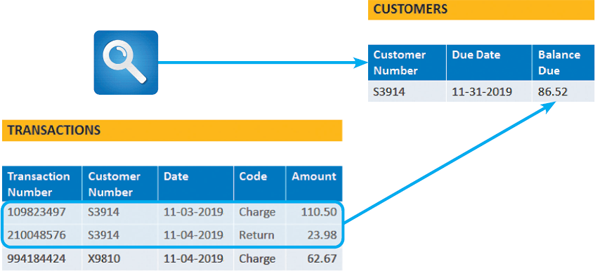
## 9.1.1Data Structures

A **data structure** is a framework for organizing, storing, and managing data. Data structures consist of files or tables that interact in various ways. Each file or table contains data about people, places, things, or events. For example, one file or table might contain data about customers, and other files or tables might store data about products, orders, suppliers, or employees.

Many older legacy systems utilized file processing because it worked well with mainframe hardware and batch input. Some companies still use this method to handle large volumes of structured data on a regular basis because can be cost-effective in certain situations. For example, consider a credit card company that posts thousands of daily transactions from a TRANSACTIONS file to account balances stored in a CUSTOMERS file, as shown in [Figure 9-1](javascript://). For that relatively simple process, file processing might be an option.

**Figure 9-1**

A credit card company that posts thousands of daily transactions might consider a file processing option.



Enlarge Image

Over time, the modern relational database became a standard model for systems developers. The following example of an auto service shop will compare the two concepts.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.1.2Mario and Danica: A Data Design Example

[Figure 9-2](javascript://) shows an auto shop mechanic at work. Imagine two shops that are very similar but use two different information system designs. Let’s call them Mario’s Auto Shop and Danica’s Auto Shop. Mario uses two file-oriented systems, while Danica uses a database management system.

**Figure 9-2**

In the example shown here, data about the mechanic, the customer, and the brake job might be stored in a file-oriented system or in a database system.



Lisa F. Young/ [Shutterstock.com](http://shutterstock.com/" \t "_blank)

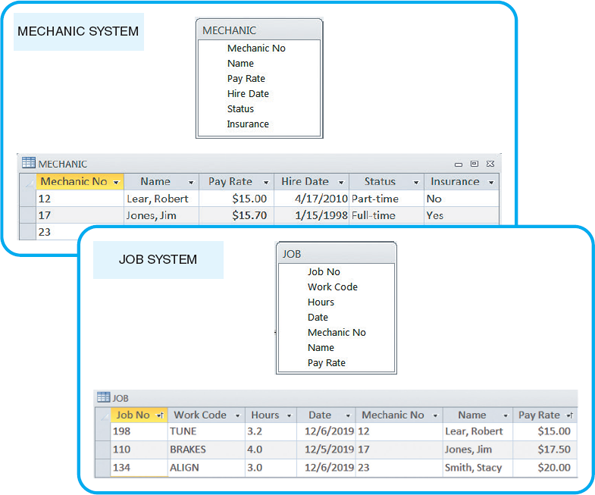
### Mario’s Auto Shop

Mario relies on two [**file-oriented systems**](javascript://), sometimes called file processing systems, to manage his business. The two systems store data in separate files that are not connected or linked. [Figure 9-3](javascript://) shows Mario’s file-oriented systems:

* The MECHANIC SYSTEM uses the MECHANIC file to store data about shop employees
* The JOB SYSTEM uses the JOB file to store data about work performed at the shop.

**Figure 9-3**

Mario’s shop uses two separate systems, so certain data must be entered twice. This redundancy is inefficient and can produce errors.



Enlarge Image

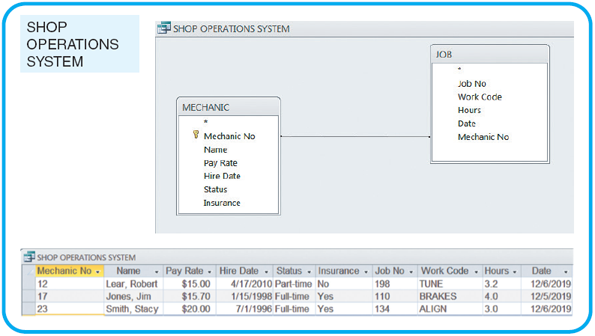
Unfortunately, using two separate systems means that some data is stored in two different places, and the data might or might not be consistent. For example, three data items (Mechanic No, Name, and Pay Rate) are stored in both files. This redundancy is a major disadvantage of file-oriented systems because it threatens data quality and integrity. In fact, [Figure 9-3](javascript://) includes a typical discrepancy: Jim Jones’ pay rate is shown as $18.90 in the MECHANIC SYSTEM file and $19.80 in the JOB SYSTEM file.

### Danica’s Auto Shop

Danica uses a database management system (DBMS) with two separate tables that are joined, so they act like one large table, as shown in [Figure 9-4](javascript://). In Danica’s SHOP OPERATIONS SYSTEM, the tables are linked by the Mechanic No field, which is called a common field because it connects the tables. Note that except for the common field, no other data items are duplicated. The DBMS design, also called a [**relational database**](javascript://) or [**relational model**](javascript://), was introduced in the 1970s and continues to be the dominant approach for organizing, storing, and managing business data.

**Figure 9-4**

Danica’s SHOP OPERATIONS SYSTEM uses a database design, which avoids duplication. The data can be viewed as if it were one large table, regardless of where the data is physically stored.



Enlarge Image

Mario’s file-oriented systems show two different pay rates for Jim Jones, most likely because of a data entry error in one of them. That type of error could not occur in Danica’s relational database, because an employee’s pay rate is stored in only one place. However, DBMSs are not immune to data entry problems, which are discussed in detail later in this chapter.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.1.3Database Management Systems

A database provides an overall framework that avoids data redundancy and supports a real-time, dynamic environment. [Figure 9-5](javascript://) shows a company-wide database that supports four separate information systems.

**Figure 9-5**

In this example, a sales database can support four separate business systems.



A [**database management system (DBMS)**](javascript://) is a collection of tools, features, and interfaces that enables users to add, update, manage, access, and analyze data. From a user’s point of view, the main advantage of a DBMS is that it offers timely, interactive, and flexible data access. Specific DBMS advantages include the following:

* Scalability. **Scalability** means that a system can be expanded, modified, or downsized easily to meet the rapidly changing needs of a business enterprise. For example, if a company decides to add data about secondary suppliers of material it uses, a new table can be added to the relational database and linked with a common field.
* Economy of scale. Database design allows better utilization of hardware. If a company maintains an enterprise-wide database, processing is less expensive using powerful servers and communication networks. The inherent efficiency of high-volume processing on larger computers is called [**economy of scale**](javascript://).
* Enterprise-wide application. A DBMS is typically managed by a person called a [**database administrator (DBA)**](javascript://). The DBA assesses overall requirements and maintains the database for the benefit of the entire organization rather than a single department or user. Database systems can support enterprise-wide applications more effectively than file processing systems.
* Stronger standards. Effective database administration helps ensure that standards for data names, formats, and documentation are followed uniformly throughout the organization.
* Better security. The DBA can define authorization procedures to ensure that only legitimate users can access the database and can allow different users to have different levels of access. Most DBMSs provide sophisticated security support.
* Data independence. Systems that interact with a DBMS are relatively independent of how the physical data is maintained. That design provides the DBA flexibility to alter data structures without modifying information systems that use the data.

Although the trend is toward enterprise-wide database design, many companies still use a combination of centralized DBMSs and smaller, department-level database systems. This is because most large businesses view data as a company-wide resource that must be accessible to users throughout the company. At the same time, other factors encourage a decentralized design, including network expense; a reluctance to move away from smaller, more flexible systems; and a realization that enterprise-wide DBMSs can be highly complex and expensive to maintain. The compromise, in many cases, is a client/server design, where processing is shared among several computers. Client/server systems are described in detail in [Chapter 10](javascript://). As with many design decisions, the best solution depends on the organization’s needs and particular circumstances.

Change font size

[**help**](javascript://)

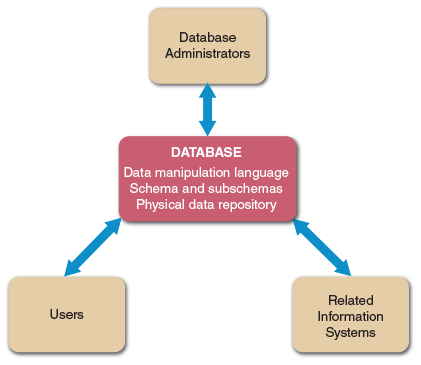
[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**9.2**DBMS Components

A DBMS provides an interface between a database and users who need to access the data. Although users are concerned primarily with an easy-to-use interface and support for their business requirements, a systems analyst must understand all of the components of a DBMS. In addition to interfaces for users, DBAs, and related systems, a DBMS also has a data manipulation language, a schema and subschemas, and a physical data repository, as shown in [Figure 9-6](javascript://).

**Figure 9-6**

In addition to interfaces for users, database administrators, and related information systems, a DBMS also has a data manipulation language, a schema and subschemas, and a physical data repository.



Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.2.1Interfaces for Users, Database Administrators, and Related Systems

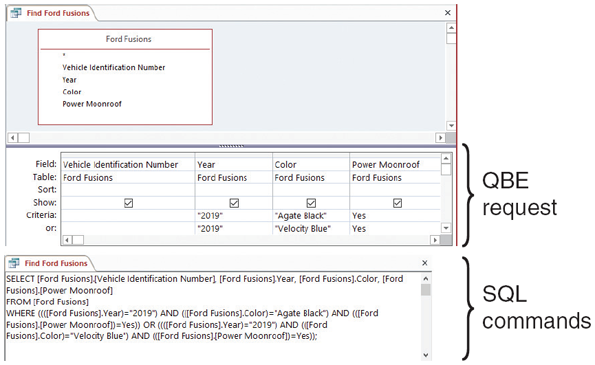
When users, DBAs, and related information systems request data and services, the DBMS processes the request, manipulates the data, and provides a response. A [**data manipulation language (DML)**](javascript://) controls database operations, including storing, retrieving, updating, and deleting data. Most commercial DBMSs, such as Oracle and IBM’s DB2, use a DML. Some database products, such as Microsoft Access, also provide an easy-to-use graphical environment that enables users to control operations with menu-driven commands.

### Users

Users typically work with predefined queries and switchboard commands but also use query languages to access stored data. A [**query language**](javascript://) allows a user to specify a task without specifying how the task will be accomplished. Some query languages use natural language commands that resemble ordinary English sentences. With a [**query by example (QBE)**](javascript://) language, the user provides an example of the data requested. Many database programs also use [**Structured Query Language (SQL)**](javascript://), which is a language that allows client workstations to communicate with servers and mainframe computers. [Figure 9-7](javascript://) shows a QBE request for all Lime Squeeze or Blue Candy 2019 Ford Fusions with a power moonroof. The QBE request generates the SQL commands shown at the bottom of [Figure 9-7](javascript://).

**Figure 9-7**

Using QBE, a user can display all 2019 Ford Fusions that have a power moonroof and are either Lime Squeeze or Blue Candy color.



Enlarge Image

### Database Administrators

A DBA is responsible for DBMS management and support. DBAs are concerned with data security and integrity, preventing unauthorized access, providing backup and recovery, audit trails, maintaining the database, and supporting user needs. Most DBMSs provide utility programs to assist the DBA in creating and updating data structures, collecting and reporting patterns of database usage, and detecting and reporting database irregularities.

### Related Information Systems

A DBMS can support several related information systems that provide input to, and require specific data from, the DBMS. Unlike a user interface, no human intervention is required for two-way communication between the DBMS and the related systems.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.2.2Schema

The complete definition of a database, including descriptions of all fields, tables, and relationships, is called a [**schema**](javascript://). One or more subschemas can also be defined. A [**subschema**](javascript://) is a view of the database used by one or more systems or users. A subschema defines only those portions of the database that a particular system or user needs or is allowed to access. For example, to protect individual privacy, the project management system should not be permitted to retrieve employee pay rates. In that case, the project management system subschema would not include the pay rate field. Database designers also use subschemas to restrict the level of access permitted. For example, specific users, systems, or locations might be permitted to create, retrieve, update, or delete data, depending on their needs and the company’s security policies.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.2.3Physical Data Repository

[Chapter 5](javascript://) discussed a data dictionary, which describes all data elements included in the logical design. At this stage of the systems development process, the data dictionary is transformed into a physical data repository, which also contains the schema and subschemas. The physical repository might be centralized, or it might be distributed at several locations. In addition, the stored data might be managed by a single DBMS, or several systems. To resolve potential database connectivity and access problems, companies use ODBC-compliant software that enables communication among various systems and DBMSs. [**Open database connectivity (ODBC)**](javascript://) is an industry-standard protocol that makes it possible for software from different vendors to interact and exchange data. ODBC uses SQL statements that the DBMS understands and can execute, similar to the ones shown in [Figure 9-7](javascript://). Another common standard is called [**java database connectivity (JDBC)**](javascript://). JDBC enables Java applications to exchange data with any database that uses SQL statements and is JDBC-compliant.

Physical design issues are described in [Chapter 10](javascript://), which discusses system architecture, and in [Chapter 11](javascript://), which discusses system implementation and data conversion.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**9.3**Web-Based Design

[Figure 9-8](javascript://) lists some major characteristics of web-based design. In a web-based design, the Internet serves as the front end, or interface, for the DBMS. Internet technology provides enormous power and flexibility because the related information system is not tied to any specific combination of hardware and software. Access to the database requires only a web browser and an Internet connection. Web-based systems are popular because they offer ease of access, cost-effectiveness, and worldwide connectivity—all of which are vital to companies that must compete in a global economy.

**Figure 9-8**

Web-based design characteristics include global access, ease of use, multiple platforms, cost-effectiveness, security issues, and adaptability issues. In a web-based design, the Internet serves as the front end, or interface, to the database management system. Access to the database requires only a web browser and an Internet connection.

| **CHARACTERISTIC** | **EXPLANATION** |
| --- | --- |
| Global access | The Internet enables worldwide access, using existing infrastructure and standard telecommunications protocols. |
| Ease of use | Web browsers provide a familiar interface that is user-friendly and easily learned. |
| Multiple platforms | Web-based design is not dependent on a specific combination of hardware or software. All that is required is a browser and an Internet connection. |
| Cost effectiveness | Initial investment is relatively low because the Internet serves as the communication network. Users require only a browser, and web-based systems do not require powerful workstations. Flexibility is high because numerous outsourcing options exist for development, hosting, maintenance, and system support. |
| Security issues | Security is a universal issue, but Internet connectivity raises special concerns. These can be addressed with a combination of good design, software that can protect the system and detect intrusion, stringent rules for passwords and user identification, and vigilant users and managers. |
| Adaptability issues | The Internet offers many advantages in terms of access, connectivity, and flexibility. Migrating a traditional database design to the web, however, can require design modification, additional software, and some added expense. |

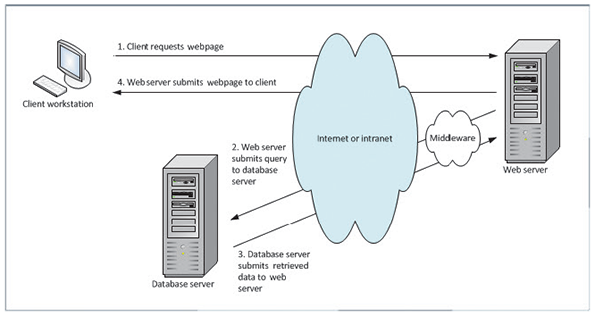
Enlarge Table

To access data in a web-based system, the database must be connected to the Internet or intranet. The database and the Internet speak two different languages, however. Databases are created and managed by using various languages and commands that have nothing to do with HTML, which is the language of the web. The objective is to connect the database to the web and enable data to be viewed and updated.

To bridge the gap, it is necessary to use **middleware**, which is a software that integrates different applications and allows them to exchange data. Middleware can interpret client requests in HTML form and translate the requests into commands that the database can execute. When the database responds to the commands, middleware translates the results into HTML pages that can be displayed by the user’s browser, as shown in [Figure 9-9](javascript://). Note that the four steps in the process can take place using the Internet or a company intranet as the communications channel. Middleware is discussed in more detail in [Chapter 10](javascript://).

**Figure 9-9**

When a client workstation requests a web page (1), the web server uses middleware to generate a data query to the database server (2). The database server responds (3), and the middleware translates the retrieved data into an HTML page that can be sent by the web server and displayed by the user’s browser (4).



Enlarge Image

Web-based data must be secure, yet easily accessible to authorized users. To achieve this goal, well-designed systems provide security at three levels: the database itself, the web server, and the telecommunication links that connect the components of the system. Data security is discussed in more detail in [Section 9.9](javascript://) and in [Chapter 12](javascript://).

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**9.4**Data Design Terms

Using the concepts discussed in the previous sections, a systems analyst can select a design approach and begin to construct the system. The first step is to understand data design terminology.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.4.1Definitions

Data design terms include entity, table, file, field, record, tuple, and key field. These terms are explained in the following sections.

### Entity

An **entity** is a person, a place, a thing, or an event for which data is collected and maintained. For example, an online sales system may include entities named CUSTOMER, ORDER, PRODUCT, and SUPPLIER. When DFDs were prepared during the systems analysis phase, various entities and data stores were identified. Now the relationships among the entities will be considered.

### Table or File

Data is organized into tables or files. A [**table**](javascript://), or [**file**](javascript://), contains a set of related records that store data about a specific entity. Tables and files are shown as two-dimensional structures that consist of vertical columns and horizontal rows. Each column represents a field, or characteristic of the entity, and each row represents a record, which is an individual instance, or occurrence of the entity. For example, if a company has 10,000 customers, the CUSTOMER table will include 10,000 records, each representing a specific customer.

Although they can have different meanings in a specific context, the terms table and file often can be used interchangeably.

### Field

A **field**, also called an **attribute**, is a single characteristic or fact about an entity. For example, a CUSTOMER entity might include the Customer ID, First Name, Last Name, Address, City, State, Postal Code, and Email Address.

A [**common field**](javascript://) is an attribute that appears in more than one entity. Common fields can be used to link entities in various types of relationships.

### Record

A [**record**](javascript://), also called a [**tuple**](javascript://) (rhymes with couple), is a set of related fields that describes one instance, or occurrence, of an entity, such as one customer, one order, or one product. A record might have one or dozens of fields, depending on what information is needed.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.4.2Key Fields

During the systems design phase, [**key fields**](javascript://) are used to organize, access, and maintain data structures. The four types of keys are primary keys, candidate keys, foreign keys, and secondary keys.

### Primary Key

A [**primary key**](javascript://) is a field or combination of fields that uniquely and minimally identifies a particular member of an entity. For example, in a customer table, the customer number is a unique primary key because no two customers can have the same customer number. That key also is minimal because it contains no information beyond what is needed to identify the customer. In a CUSTOMER table, a Customer ID might be used as a unique primary key. Customer ID is an example of a primary key based on a single field.

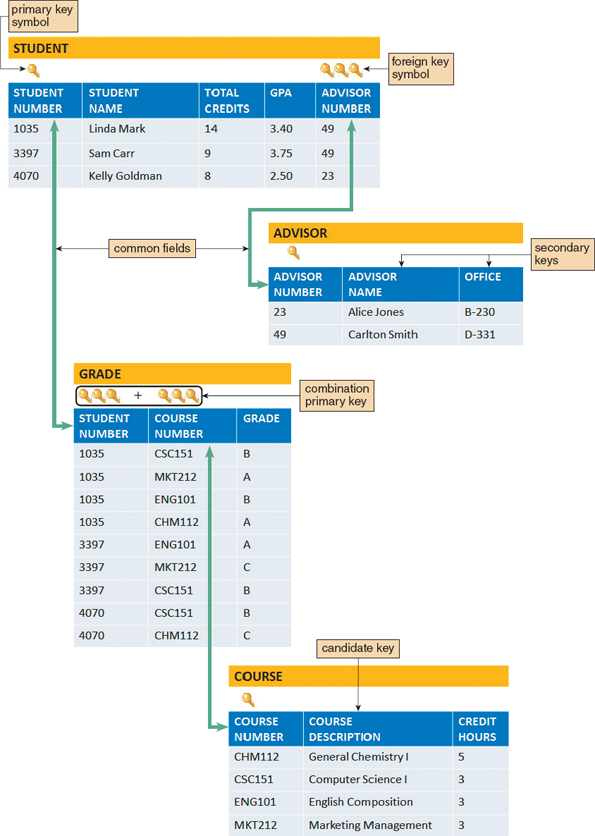
A primary key also can be composed of two or more fields. For example, if a student registers for three courses, his or her student number will appear in three records in the registration system. If one of those courses has 20 students, 20 separate records will exist for that course number—one record for each student who registered.

In the registration file, neither the student number nor the course ID is unique, so neither field can be a primary key. To identify a specific student in a specific course, the primary key must be a combination of student number and course ID. In that case, the primary key is called a [**combination key**](javascript://). A combination key also can be called a [**composite key**](javascript://), [**concatenated key**](javascript://), or [**multivalued key**](javascript://).

[Figure 9-10](javascript://) shows four different tables: STUDENT, ADVISOR, COURSE, and GRADE. Three of these tables have single-field primary keys. Note that in the GRADE table, however, the primary key is a combination of two fields: STUDENT NUMBER and COURSE NUMBER.

**Figure 9-10**

Examples of common fields, primary keys, candidate keys, foreign keys, and secondary keys.



Enlarge Image

### Candidate Key

Sometimes there is a choice of fields or field combinations to use as the primary key. Any field that can serve as a primary key is called a [**candidate key**](javascript://). For example, if every employee has a unique employee number, then it could be used as a primary key. Note that an employee’s Social Security number would not be a good choice for a candidate key: Contrary to popular belief, Social Security numbers are not unique. Because only one field can be designated as a primary key, the field that contains the least amount of data and is the easiest to use should be selected. Any field that is not a primary key or a candidate key is called a **[nonkey field](javascript://)**.

The primary keys shown in [Figure 9-10](javascript://) also are candidate keys. Another candidate key is the COURSE DESCRIPTION field in the COURSE table. What about the OFFICE field in the ADVISOR table? It could not be a candidate key because more than one advisor might share the same office.

### Foreign Key

Recall that a common field exists in more than one table and can be used to form a relationship, or link, between the tables. For example, in [Figure 9-10](javascript://), the ADVISOR NUMBER field appears in both the STUDENT table and the ADVISOR table and joins the tables together. Note that ADVISOR NUMBER is a primary key in the ADVISOR table, where it uniquely identifies each advisor, and is a foreign key in the STUDENT table. A [**foreign key**](javascript://) is a field in one table that must match a primary key value in another table in order to establish the relationship between the two tables.

Unlike a primary key, a foreign key need not be unique. For example, Carlton Smith has advisor number 49. The value 49 must be a unique value in the ADVISOR table because it is the primary key, but 49 can appear any number of times in the STUDENT table, where the advisor number serves as a foreign key.

[Figure 9-10](javascript://) also shows how two foreign keys can serve as a composite primary key in another table. Consider the GRADE table at the bottom of the figure. The two fields that form the primary key for the GRADE table are both foreign keys: the STUDENT NUMBER field, which must match a student number in the STUDENT table, and the COURSE NUMBER field, which must match one of the course IDs in the COURSE table.

How can these two foreign keys serve as a primary key in the GRADE table? Note that student numbers and course IDs can appear any number of times in the table, but the combination of a specific student and a specific course occurs only once. For example, student 1035 appears four times and course CSC151 appears three times—but there is only one combined instance of student 1035 and course CSC151. Because the combination of the specific student (1035) and the specific course (CSC151) is unique, it ensures that the grade (B) will be assigned to the proper student in the proper course.

### Secondary Key

A [**secondary key**](javascript://) is a field or combination of fields that can be used to access or retrieve records. Secondary key values are not unique. For example, to access records for only those customers in a specific postal code, the postal code field would be used as a secondary key. Secondary keys also can be used to sort or display records in a certain order. For example, the GPA field in a STUDENT file could be used to display records for all students in grade point order.

The need for a secondary key arises because a table can have only one primary key. In a CUSTOMER file, the CUSTOMER NUMBER is the primary key, so it must be unique. The customer’s name might be known, but not the customer’s number. For example, to access a customer named James Morgan without knowing his customer number, the table is searched using the CUSTOMER NAME field as a secondary key. The records for all customers named James Morgan are retrieved and then the correct record is selected.

In [Figure 9-10](javascript://), student name and advisor names are identified as secondary keys, but other fields also could be used. For example, to find all students who have a particular advisor, the ADVISOR NUMBER field in the STUDENT table could be used as a secondary key.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.4.3Referential Integrity

Validity checks can help avoid data input errors. One type of validity check, called [**referential integrity**](javascript://), is a set of rules that avoids data inconsistency and quality problems. In a relational database, referential integrity means that a foreign key value cannot be entered in one table unless it matches an existing primary key in another table. For example, referential integrity would prevent a customer order from being entered in an order table unless that customer already exists in the customer table. Without referential integrity, there might be an order called an [**orphan**](javascript://), because it had no related customer.

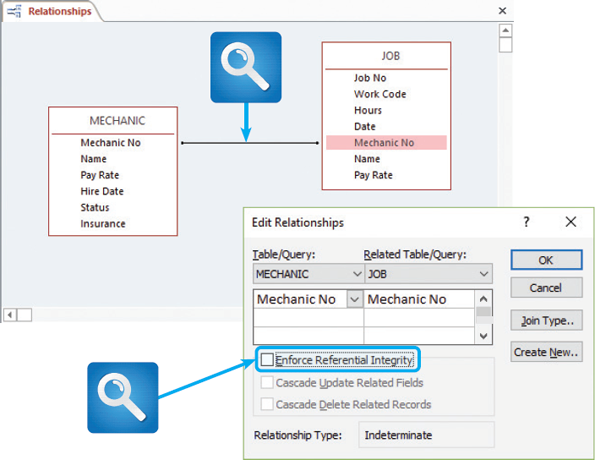
In the example shown in [Figure 9-10](javascript://), referential integrity will not allow a user to enter an advisor number (foreign key value) in the STUDENT table unless a valid advisor number (primary key value) already exists in the ADVISOR table.

Referential integrity can also prevent the deletion of a record if the record has a primary key that matches foreign keys in another table. For example, suppose that an advisor resigns to accept a position at another school. The advisor cannot be deleted from the ADVISOR table while records in the STUDENT table still refer to that advisor number. Otherwise, the STUDENT records would be orphans. To avoid the problem, students must be reassigned to other advisors by changing the value in the ADVISOR NUMBER field; then the advisor record can be deleted.

When creating a relational database, referential integrity can be built into the design. [Figure 9-11](javascript://) shows a Microsoft Access screen that identifies a common field and allows the user to enforce referential integrity rules.

**Figure 9-11**

Microsoft Access allows a user to specify that referential integrity rules will be enforced in a relational database design.



Enlarge Image

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**9.5**Entity-Relationship Diagrams

Recall that an entity is a person, a place, a thing, or an event for which data is collected and maintained. For example, entities might be customers, sales regions, products, or orders. An information system must recognize the relationships among entities. For example, a CUSTOMER entity can have several instances of an ORDER entity, and an EMPLOYEE entity can have one instance, or none, of a SPOUSE entity.

An [**entity-relationship diagram (ERD)**](javascript://) is a model that shows the logical relationships and interaction among system entities. An ERD provides an overall view of the system and a blueprint for creating the physical data structures.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

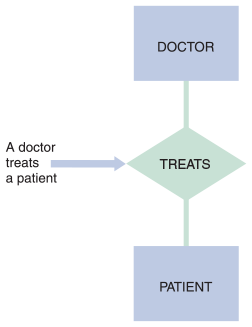
## 9.5.1Drawing an ERD

The first step is to list the entities that were identified during the systems analysis phase and to consider the nature of the relationships that link them. At this stage, a simplified method can be used to show the relationships between entities.

Although there are different ways to draw ERDs, a popular method is to represent entities as rectangles and relationships as diamond shapes. The entity rectangles are labeled with singular nouns, and the relationship diamonds are labeled with verbs, usually in a top-to-bottom and left-to-right fashion. For example, in [Figure 9-12](javascript://), a DOCTOR entity treats a PATIENT entity. Unlike data flow diagrams, ERDs depict relationships, not data or information flows.

**Figure 9-12**

In an entity-relationship diagram, entities are labeled with singular nouns and relationships are labeled with verbs. The relationship is interpreted as a simple English sentence.



Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

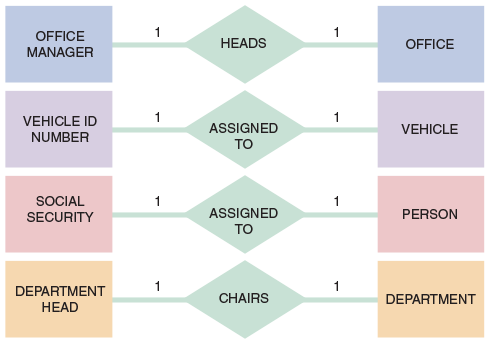
## 9.5.2Types of Relationships

Three types of relationships can exist between entities: one-to-one, one-to-many, and many-to-many.

A [**one-to-one relationship**](javascript://), abbreviated [**1:1**](javascript://), exists when exactly one of the second entity occurs for each instance of the first entity. [Figure 9-13](javascript://) shows examples of several 1:1 relationships. A number 1 is placed alongside each of the two connecting lines to indicate the 1:1 relationship.

**Figure 9-13**

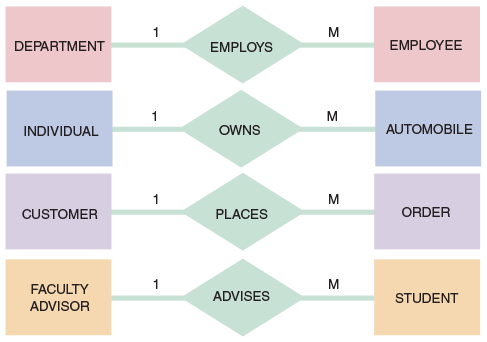
Examples of one-to-one (1:1) relationships.



A [**one-to-many relationship**](javascript://), abbreviated [**1:M**](javascript://), exists when one occurrence of the first entity can relate to many instances of the second entity, but each instance of the second entity can associate with only one instance of the first entity. For example, the relationship between DEPARTMENT and EMPLOYEE is one-to-many: One department can have many employees, but each employee works in only one department at a time. [Figure 9-14](javascript://) shows several 1:M relationships. The line connecting the many entity is labeled with the letter M, and the number 1 labels the other connecting line. How many is many? The first 1:M relationship shown in [Figure 9-14](javascript://) shows the entities INDIVIDUAL and AUTOMOBILE. One individual might own five automobiles, or one, or none. Thus, many can mean any number, including zero.

**Figure 9-14**

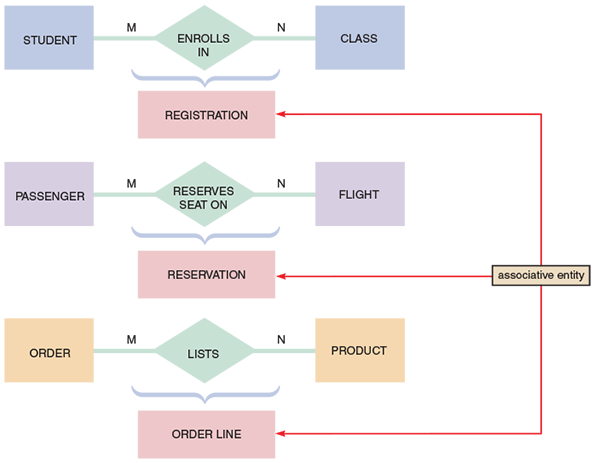
Examples of one-to-many (1:M) relationships.



A [**many-to-many relationship**](javascript://), abbreviated [**M:N**](javascript://), exists when one instance of the first entity can relate to many instances of the second entity, and one instance of the second entity can relate to many instances of the first entity. The relationship between STUDENT and CLASS, for example, is many-to-many—one student can take many classes, and one class can have many students enrolled. [Figure 9-15](javascript://) shows several M:N entity relationships. One of the connecting lines is labeled with the letter M, and the letter N labels the other connection.

**Figure 9-15**

Examples of many-to-many (M:N) relationships. Notice that the event or transaction that links the two entities is an associative entry with its own set of attributes and characteristics.



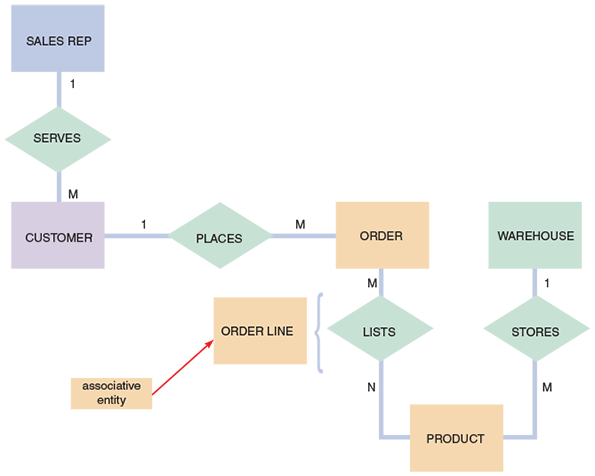
Enlarge Image

Note that an M:N relationship is different from 1:1 or 1:M relationships because the event or transaction that links the two entities is actually a third entity, called an [**associative entity**](javascript://), that has its own characteristics. In the first example in [Figure 9-15](javascript://), the ENROLLS IN symbol represents a REGISTRATION entity that records each instance of a specific student enrolling in a specific course. Similarly, the RESERVES SEAT ON symbol represents a RESERVATION entity that records each instance of a specific passenger reserving a seat on a specific flight. In the third example, the LISTS symbol represents an ORDER LINE entity that records each instance of a specific product listed in a specific customer order.

[Figure 9-16](javascript://) shows an ERD for a sales system. Note the various entities and relationships shown in the figure, including the associative entity named ORDER LINE. The detailed nature of these relationships is called cardinality. An analyst must understand cardinality in order to create a data design that accurately reflects all relationships among system entities.

**Figure 9-16**

An entity-relationship diagram for SALES REP, CUSTOMER, ORDER, PRODUCT, and WAREHOUSE. Notice that the ORDER and PRODUCT entities are joined by an associative entity named ORDER LINE.



Enlarge Image

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

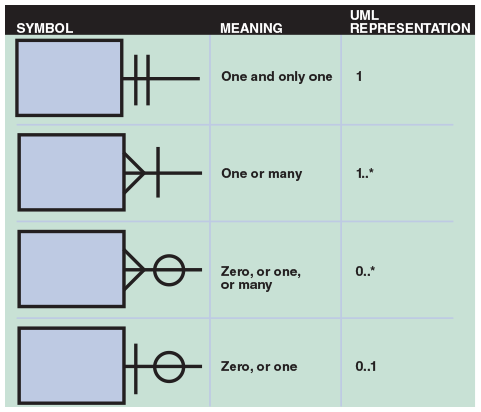
## 9.5.3Cardinality

After an analyst draws an initial ERD, he or she must define the relationships in more detail by using a technique called cardinality. **Cardinality** describes the numeric relationship between two entities and shows how instances of one entity relate to instances of another entity. For example, consider the relationship between two entities: CUSTOMER and ORDER. One customer can have one order, many orders, or none, but each order must have one and only one customer. An analyst can model this interaction by adding [**cardinality notation**](javascript://), which uses special symbols to represent the relationship.

A common method of cardinality notation is called [**crow’s foot notation**](javascript://) because of the shapes, which include circles, bars, and symbols that indicate various possibilities. A single bar indicates one, a double bar indicates one and only one, a circle indicates zero, and a crow’s foot indicates many. [Figure 9-17](javascript://) shows various cardinality symbols, their meanings, and the UML representations of the relationships. As described in [Chapter 4](javascript://), the **Unified Modeling Language (UML)** is a widely used method of visualizing and documenting software systems design.

**Figure 9-17**

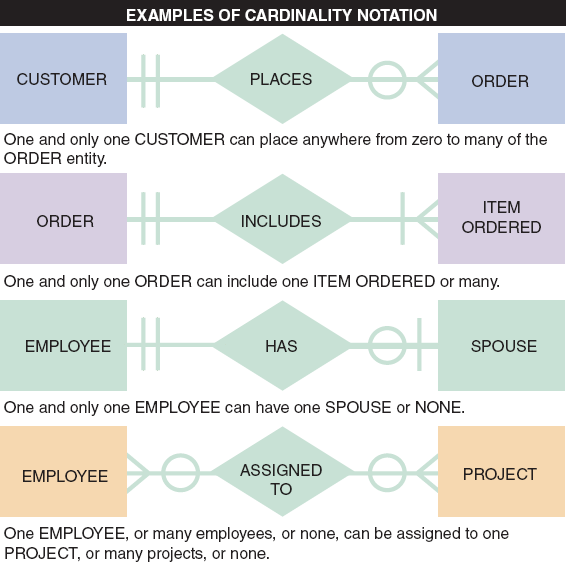
Crow’s foot notation is a common method of indicating cardinality. The four examples show how various symbols can be used to describe the relationships between entities.



In [Figure 9-18](javascript://), four examples of cardinality notation are shown. In the first example, one and only one CUSTOMER can place anywhere from zero to many of the ORDER entity. In the second example, one and only one ORDER can include one ITEM ORDERED or many. In the third example, one and only one EMPLOYEE can have one SPOUSE or none. In the fourth example, one EMPLOYEE, or many employees, or none, can be assigned to one PROJECT, or many projects, or none.

**Figure 9-18**

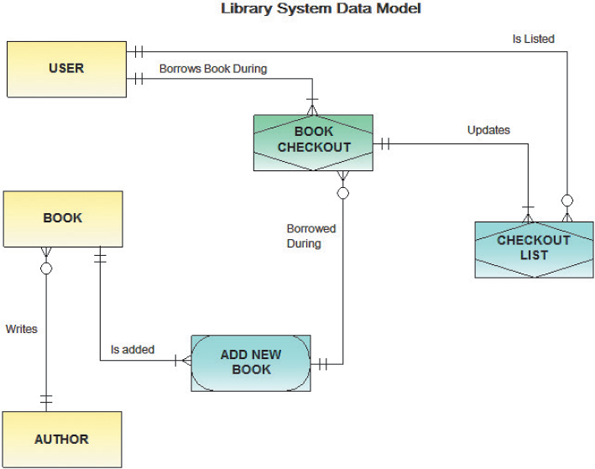
In the first example of cardinality notation, one and only one CUSTOMER can place anywhere from zero to many of the ORDER entity. In the second example, one and only one ORDER can include one ITEM ORDERED or many. In the third example, one and only one EMPLOYEE can have one SPOUSE or none. In the fourth example, one EMPLOYEE, or many employees, or none, can be assigned to one PROJECT, or many projects, or none.



Most CASE products support the drawing of ERDs from entities in the data repository. [Figure 9-19](javascript://) shows part of a library system ERD drawn using the Visible Analyst CASE tool. Note that crow’s foot notation is used to show the nature of the relationships, which are described in both directions.

**Figure 9-19**

An ERD for a library system drawn with Visible Analyst. Notice that crow’s foot notation has been used and relationships are described in both directions.



Enlarge Image

**Case in Point 9.1**

### TopText Publishing

TopText Publishing is a textbook publishing company with a headquarters location, a warehouse, and three sales offices that each has a sales manager and sales reps. TopText sells to schools, colleges, and individual customers. Many authors write more than one book for TopText, and more than one author writes some books. TopText maintains an active list of more than 100 books, each identified by a universal code called an ISBN. How would you draw an ERD for the TopText information system, including cardinality notation?

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**9.6**Data Normalization

[**Normalization**](javascript://) is the process of creating table designs by assigning specific fields or attributes to each table in the database. A [**table design**](javascript://) specifies the fields and identifies the primary key in a particular table or file. Working with a set of initial table designs, normalization is used to develop an overall database design that is simple, flexible, and free of data redundancy. Normalization involves applying a set of rules that can help identify and correct inherent problems and complexities in table designs. The concept of normalization is based on the work of Edgar Codd, a British computer scientist, who formulated the basic principles of relational database design.

The normalization process typically involves four stages: unnormalized design, first normal form, second normal form, and third normal form. The three normal forms constitute a progression in which third normal form represents the best design. Most business-related databases must be designed in third normal form. Note that normal forms beyond 3NF exist, but they rarely are used in business-oriented systems.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.6.1Standard Notation Format

Designing tables is easier if a [**standard notation format**](javascript://) is used to show a table’s structure, fields, and primary key. The standard notation format in the following examples of an ORDER system starts with the name of the table, followed by a parenthetical expression that contains the field names separated by commas. The primary key field(s) is/are underlined, like this:

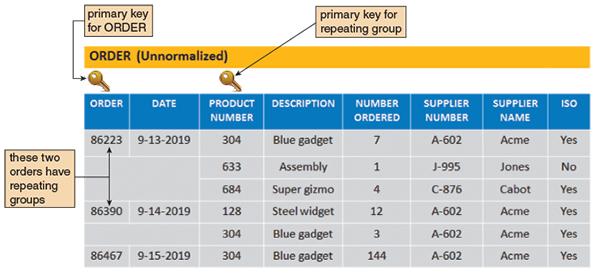
* NAME (FIELD 1, FIELD 2, FIELD 3)

During data design, the analyst must be able to recognize a repeating group of fields. A [**repeating group**](javascript://) is a set of one or more fields that can occur any number of times in a single record, with each occurrence having different values.

A typical example of a repeating group is shown in [Figure 9-20](javascript://). If a company used written source documents to record orders, they might look like this. As [Figure 9-20](javascript://) shows, two orders contain multiple items, which constitute repeating groups within the same order number. Note that in addition to the order number and date, the records with multiple products contain repetitions of the product number, description, number ordered, supplier number, supplier name, and ISO status. A repeating group can be thought of as a set of child (subsidiary) records contained within the parent (main) record.

**Figure 9-20**

In the ORDER table design, two orders have repeating groups that contain several products. ORDER is the primary key for the ORDER table, and PRODUCT NUMBER serves as a primary key for the repeating group. Because it contains repeating groups, the ORDER table is unnormalized.



Enlarge Image

A table design that contains a repeating group is called [**unnormalized**](javascript://). The standard notation method for representing an unnormalized design is to enclose the repeating group of fields within a second set of parentheses. An example of an unnormalized table looks like this:

* NAME (FIELD 1, FIELD 2, FIELD 3, (REPEATING FIELD 1, REPEATING FIELD 2))

Now review the unnormalized ORDER table design shown in [Figure 9-20](javascript://). Following the notation guidelines, the design can be described as follows:

* ORDER (ORDER, DATE, (PRODUCT NUMBER, DESCRIPTION, NUMBER ORDERED, SUPPLIER NUMBER, SUPPLIER NAME, ISO))

The notation indicates that the ORDER table design contains eight fields, which are listed within the outer parentheses. The ORDER field is underlined to show that it is the primary key. The PRODUCT NUMBER, DESCRIPTION, NUMBER ORDERED, SUPPLIER NUMBER, SUPPLIER NAME, and ISO and NUMBER ORDERED fields are enclosed within an inner set of parentheses to indicate that they are fields within a repeating group. Note that PRODUCT NUMBER also is underlined because it acts as the primary key of the repeating group. If a customer orders three different products in one order, then six fields must be repeated for each product, as shown in [Figure 9-20](javascript://).

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.6.2First Normal Form

A table is in [**first normal form (1NF)**](javascript://) if it does not contain a repeating group. To convert an unnormalized design to 1NF, the table’s primary key must be expanded to include the primary key of the repeating group.

For example, in the ORDER table shown in [Figure 9-20](javascript://), the repeating group consists of six fields: PRODUCT NUMBER, DESCRIPTION, NUMBER ORDERED, SUPPLIER NUMBER, SUPPLIER NAME, and ISO. Of the three fields, only PRODUCT NUMBER can be a primary key because it uniquely identifies each instance of the repeating group. The DESCRIPTION cannot be a primary key because it might or might not be unique. For example, a company might sell a large number of parts with the same descriptive name, such as washer, relying on a coded part number to identify uniquely each washer size.

When the primary key of the ORDER table is expanded to include PRODUCT NUMBER, the repeating group is eliminated, and the ORDER table is now in 1NF, as shown:

* ORDER (ORDER, DATE, PRODUCT NUMBER, DESCRIPTION, NUMBER ORDERED, SUPPLIER NUMBER, SUPPLIER NAME, ISO)

[Figure 9-21](javascript://) shows the ORDER table in 1NF. Note that when the repeating group is eliminated, additional records emerge—one for each combination of a specific order and a specific product. The result is more records but a greatly simplified design. In the new version, the repeating group for order number 86223 has become three separate records, and the repeating group for order number 86390 has become two separate records. Therefore, when a table is in 1NF, each record stores data about a single instance of a specific order and a specific product.

**Figure 9-21**

The ORDER table as it appears in 1NF. The repeating groups have been eliminated. Notice that the repeating group for order 86223 has become three separate records, and the repeating group for order 86390 has become two separate records. The 1NF primary key is a combination of ORDER and PRODUCT NUMBER, which uniquely identifies each record.



Enlarge Image

Also note that the 1NF design shown in [Figure 9-21](javascript://) has a combination primary key. The primary key of the 1NF design cannot be the ORDER field alone, because the order number does not uniquely identify each product in a multiple-item order. Similarly, PRODUCT NUMBER cannot be the primary key, because it appears more than once if several orders include the same product. Because each record must reflect a specific product in a specific order, both fields are needed, ORDER and PRODUCT NUMBER, to identify a single record uniquely. Therefore, the primary key is the combination of two fields: ORDER and PRODUCT NUMBER.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.6.3Second Normal Form

To understand second normal form (2NF), the concept of functional dependence must be understood. For example, Field A is [**functionally dependent**](javascript://) on Field B if the value of Field A depends on Field B. For example, in [Figure 9-21](javascript://), the DATE value is functionally dependent on the ORDER, because for a specific order number, there can be only one date. In contrast, a product description is not dependent on the order number. For a particular order number, there might be several product descriptions—one for each item ordered.

A table design is in [**second normal form (2NF)**](javascript://) if it is in 1NF and if all fields that are not part of the primary key are functionally dependent on the entire primary key. If any field in a 1NF table depends on only one of the fields in a combination primary key, then the table is not in 2NF.

Note that if a 1NF design has a primary key that consists of only one field, the problem of partial dependence does not arise—because the entire primary key is a single field. Therefore, a 1NF table with a single-field primary key is automatically in 2NF.

Now reexamine the 1NF design for the ORDER table shown in [Figure 9-21](javascript://):

* ORDER (ORDER, DATE, PRODUCT NUMBER, DESCRIPTION, NUMBER ORDERED, SUPPLIER NUMBER, SUPPLIER NAME, ISO)

Recall that the primary key is the combination of the order number and the product number. The NUMBER ORDERED field depends on the entire primary key because NUMBER ORDERED refers to a specific product number and a specific order number. In contrast, the DATE field depends on the order number, which is only a part of the primary key. Similarly, the DESCRIPTION field depends on the product number, which also is only a part of the primary key. Because some fields are not dependent on the entire primary key, the design is not in 2NF.

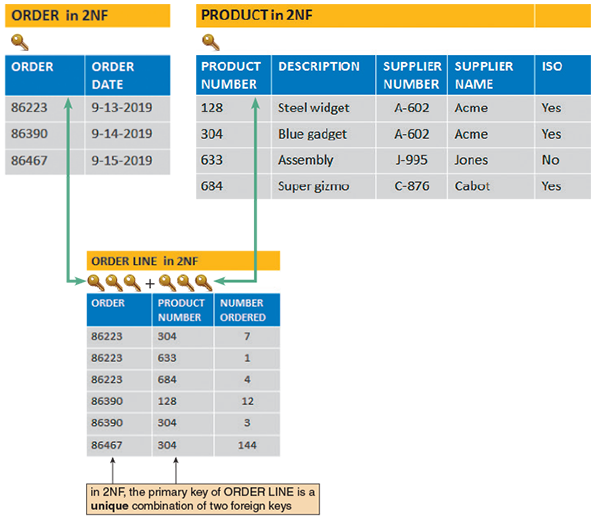
A standard process exists for converting a table from 1NF to 2NF. The objective is to break the original table into two or more new tables and reassign the fields so that each nonkey field will depend on the entire primary key in its table. To accomplish this, the following steps should be followed:

1. Create and name a separate table for each field in the existing primary key. For example, in [Figure 9-21](javascript://), the ORDER table’s primary key has two fields, ORDER and PRODUCT NUMBER, so two tables must be created. The ellipsis (…) indicates that fields will be assigned later. The result is:
   * ORDER (ORDER, …)
   * PRODUCT (PRODUCT NUMBER, …)
2. Create a new table for each possible combination of the original primary key fields. In the [Figure 9-21](javascript://) example, a new table would be created with a combination primary key of ORDER and PRODUCT NUMBER. This table describes individual lines in an order, so it is named ORDER LINE, as shown:
   * ORDER LINE (ORDER, PRODUCT NUMBER)
3. Study the three tables and place each field with its appropriate primary key, which is the minimal key on which it functionally depends. When all the fields have been placed, remove any table that did not have any additional fields assigned to it. The remaining tables are the 2NF version of the original table. The three tables can be shown as:
   * ORDER (ORDER, DATE)
   * PRODUCT (PRODUCT NUMBER, DESCRIPTION, SUPPLIER NUMBER, SUPPLIER NAME, ISO)
   * ORDER LINE (ORDER, PRODUCT NUMBER)

[Figure 9-22](javascript://) shows the 2NF table designs. By following the steps, the original 1NF table has been converted into three 2NF tables.

**Figure 9-22**

ORDER, PRODUCT, and ORDER LINE tables in 2NF. All fields are functionally dependent on the primary key.



Enlarge Image

Why is it important to move from 1NF to 2NF? Four kinds of problems are found with 1NF designs that do not exist in 2NF:

1. Consider the work necessary to change a particular product’s description. Suppose 500 current orders exist for product number 304. Changing the product description involves modifying 500 records for product number 304. Updating all 500 records would be cumbersome and expensive.
2. 1NF tables can contain inconsistent data. Because someone must enter the product description in each record, nothing prevents product number 304 from having different product descriptions in different records. In fact, if product number 304 appears in a large number of order records, some of the matching product descriptions might be inaccurate or improperly spelled. Even the presence or absence of a hyphen in the orders for all-purpose gadget would create consistency problems. If a data entry person must enter a term such as IO1Queue Controller numerous times, it certainly is possible that some inconsistency will result.
3. Adding a new product is a problem. Because the primary key must include an order number and a product number, values are needed for both fields in order to add a record. What value should be used for the order number when no customer has ordered the product? A dummy order number could be used and later replaced with a real order number when the product is ordered to solve the problem, but that solution also creates difficulties.
4. Deleting a product is a problem. If all the related records are deleted once an order is filled and paid for, what happens if the only record that contains product number 633 is deleted? The information about that product number and its description is lost.

Has the 2NF design eliminated all potential problems? To change a product description, now just one PRODUCT record needs to be changed. Multiple, inconsistent values for the product description are impossible because the description appears in only one location. To add a new product, a new PRODUCT record is created, instead of creating a dummy order record. When the last ORDER LINE record for a particular product number is removed, that product number and its description is not lost because the PRODUCT record still exists. The four potential problems are eliminated, and the three 2NF designs are superior to both the original unnormalized table and the 1NF design.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.6.4Third Normal Form

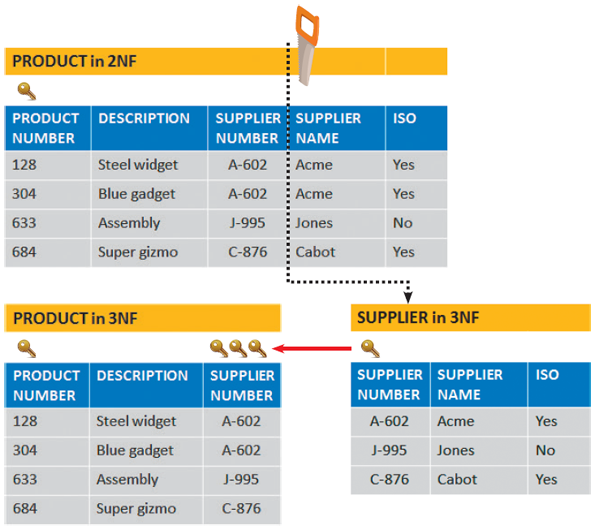
A popular rule of thumb is that a design is in 3NF if every nonkey field depends on the key, the whole key, and nothing but the key. A 3NF design avoids redundancy and data integrity problems that still can exist in 2NF designs.

Continuing the ORDER example, now review the PRODUCT table design in [Figure 9-23](javascript://):

* PRODUCT (PRODUCT NUMBER, DESCRIPTION, SUPPLIER NUMBER, SUPPLIER NAME, ISO)

**Figure 9-23**

When the PRODUCT table is transformed from 2NF to 3NF, the result is two separate tables: PRODUCT and SUPPLIER. Note that in 3NF, all fields depend on the key alone.



Enlarge Image

The PRODUCT table is in 1NF because it has no repeating groups. The table also is in 2NF because the primary key is a single field. But the table still has four potential problems:

1. To change a supplier name, every record in which that name appears must be changed. With hundreds, or even thousands of records, the process would be slow, expensive, and subject to input errors.
2. The 2NF design allows a supplier to have a different name or ISO status in different records.
3. Because the supplier name is included in the ORDER table, a dummy ORDER record must be created to add a new supplier who has not yet been received any orders.
4. If all the orders for a supplier are deleted, that supplier’s number and name will be lost.

Those potential problems are caused because the design is not in 3NF. A table design is in [**third normal form (3NF)**](javascript://) if it is in 2NF and if no nonkey field is dependent on another nonkey field. Remember that a nonkey field is a field that is not a candidate key for the primary key.

The PRODUCT table at the top of [Figure 9-23](javascript://) is not in 3NF because two nonkey fields, SUPPLIER NAME and ISO, both depend on another nonkey field, SUPPLIER NUMBER.

To convert the table to 3NF, all fields from the 2NF table that depend on another nonkey field must be removed and placed in a new table that uses the nonkey field as a primary key. In the PRODUCT example, SUPPLIER NAME and ISO must be removed and placed into a new table that uses SUPPLIER NUMBER as the primary key. As shown in [Figure 9-23](javascript://), 3NF divides the 2NF version into two separate 3NF tables:

* PRODUCT (PRODUCT NUMBER, DESCRIPTION, SUPPLIER NUMBER)
* SUPPLIER (SUPPLIER NUMBER, SUPPLIER NAME, ISO)

**Case in Point 9.2**

### CyberToys

You handle administrative support for CyberToys, a small chain that sells computer hardware and software and specializes in personal service. The company has four stores located at malls and is planning more. Each store has a manager, a technician, and between one and four sales reps.

The owners want to create a personnel records database, and they asked you to review a table that they had designed. They suggested fields for store number, location, store telephone, manager name, and manager home telephone. They also want fields for technician name and technician home telephone and fields for up to four sales rep names and sales rep home telephones.

Draw their suggested design and analyze it using the normalization concepts you learned in the chapter. What do you think of their design and why? What would you propose?

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.6.5Two Real-World Examples

A good way to learn about normalization is to apply the rules to a representative situation. This section presents two different scenarios: first a school and then a technical service company. If a step-by-step process is followed, data designs can be created that are efficient, maintainable, and error-resistant.

### Example 1: Crossroads College

Consider the familiar situation in [Figure 9-24](javascript://), which depicts several entities in the Crossroads College advising system: ADVISOR, COURSE, and STUDENT. The relationships among the three entities are shown in the ERD in [Figure 9-25](javascript://). The following sections discuss normalization rules for these three entities.

**Figure 9-24**

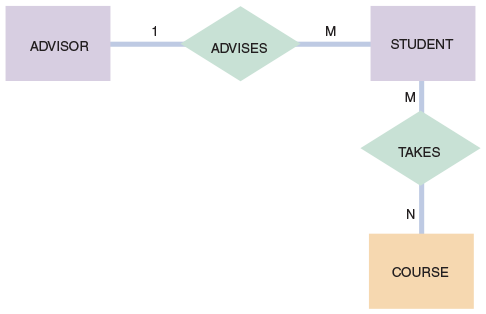
A faculty advisor, who represents an entity, can advise many students, each of whom can register for one or many courses.



Monkey Business Images/ [Shutterstock.com](http://shutterstock.com/" \t "_blank)

**Figure 9-25**

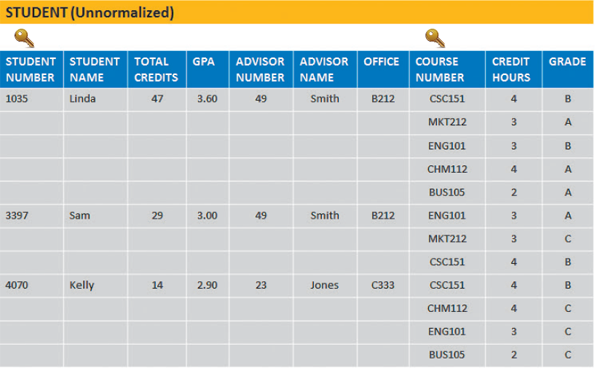
An initial entity-relationship diagram for ADVISOR, STUDENT, and COURSE.



Before the normalization process is started, it is noted that the STUDENT table contains fields that relate to the ADVISOR and COURSE entities, so a decision is made to begin with the initial design for the STUDENT table, which is shown in [Figure 9-26](javascript://). Note that the table design includes the student number, student name, total credits taken, grade point average (GPA), advisor number, advisor name, and, for every course the student has taken, the course number, course description, number of credits, and grade received.

**Figure 9-26**

The STUDENT table is unnormalized because it contains a repeating group that represents the course each student has taken.



Enlarge Image

The STUDENT table in [Figure 9-26](javascript://) is unnormalized because it has a repeating group. The STUDENT table design can be written as:

* STUDENT (STUDENT NUMBER, STUDENT NAME, TOTAL CREDITS, GPA, ADVISOR NUMBER, ADVISOR NAME, OFFICE, (COURSE NUMBER, CREDIT HOURS, GRADE))

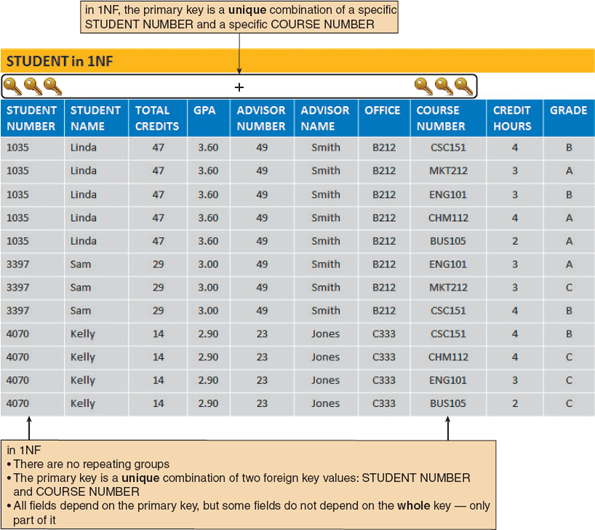
To convert the STUDENT record to 1NF, the primary key must be expanded to include the key of the repeating group, producing:

* STUDENT (STUDENT NUMBER, STUDENT NAME, TOTAL CREDITS, GPA, ADVISOR NUMBER, ADVISOR NAME, OFFICE, COURSE NUMBER, CREDIT HOURS, GRADE)

[Figure 9-27](javascript://) shows the 1NF version of the sample STUDENT data. Do any of the fields in the 1NF STUDENT table depend on only a portion of the primary key? The student name, total credits, GPA, advisor number, and advisor name all relate only to the student number and have no relationship to the course number. The course description depends on the course number but not on the student number. Only the GRADE field depends on the entire primary key.

**Figure 9-27**

The student table in 1NF. Notice that the primary key has been expanded to include STUDENT NUMBER and COURSE NUMBER.



Enlarge Image

Following the 1NF-2NF conversion process described earlier, a new table would be created for each field and combination of fields in the primary key, and the other fields would be placed with their appropriate key. The result is:

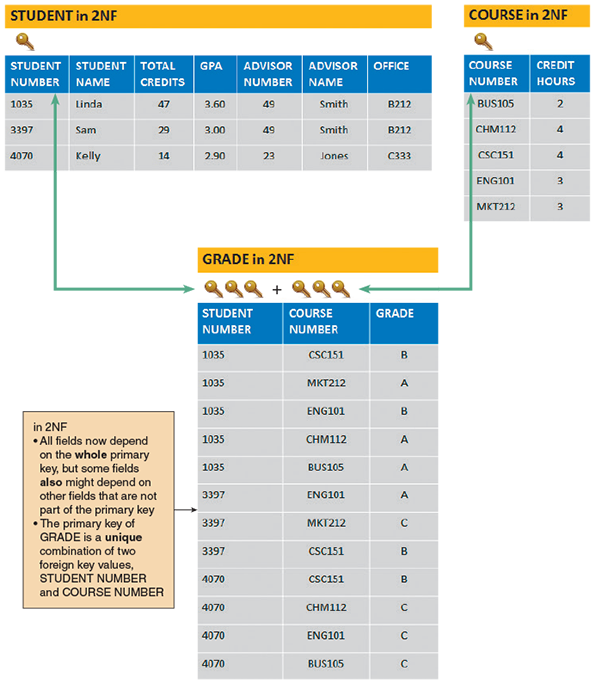
* STUDENT (STUDENT NUMBER, STUDENT NAME, TOTAL CREDITS, GPA, ADVISOR NUMBER, ADVISOR NAME, OFFICE)
* COURSE (COURSE NUMBER, CREDIT HOURS)
* GRADE (STUDENT NUMBER, COURSE NUMBER, GRADE)

The original 1NF STUDENT table has now been converted into three tables, all in 2NF. In each table, every nonkey field depends on the entire primary key.

[Figure 9-28](javascript://) shows the 2NF STUDENT, COURSE, and GRADE designs and sample data. Are all three tables STUDENT in 3NF? The COURSE and GRADE tables are in 3NF. STUDENT is not in 3NF, however, because the ADVISOR NAME and OFFICE fields depend on the ADVISOR NUMBER field, which is not part of the STUDENT primary key. To convert STUDENT to 3NF, the ADVISOR NAME and OFFICE fields are removed from the STUDENT table and placed into a table with ADVISOR NUMBER as the primary key.

**Figure 9-28**

The STUDENT, COURSE, and GRADE tables in 2NF. Notice that all fields are functionally dependent on the entire primary key of their respective tables.



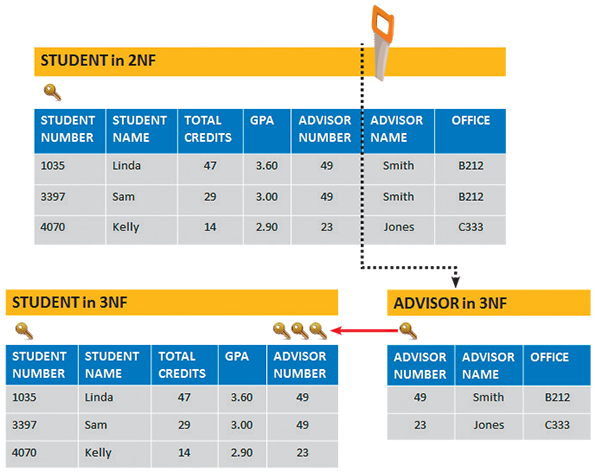
Enlarge Image

[Figure 9-29](javascript://) shows the 3NF versions of the sample data for STUDENT, ADVISOR, COURSE, and GRADE. The final 3NF design is:

* STUDENT (STUDENT NUMBER, STUDENT NAME, TOTAL CREDITS, GPA, ADVISOR NUMBER)
* ADVISOR (ADVISOR NUMBER, ADVISOR NAME, OFFICE)
* COURSE (COURSE NUMBER, CREDIT HOURS)
* GRADE (STUDENT NUMBER, COURSE NUMBER, GRADE)

**Figure 9-29**

STUDENT, ADVISOR, COURSE, and GRADE tables in 3NF. When the STUDENT table is transformed from 2NF to 3NF, the result is two tables: STUDENT and ADVISOR.

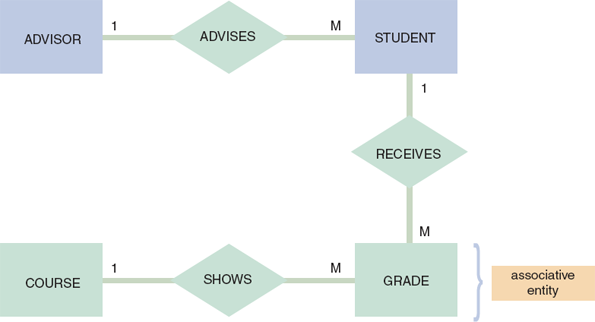


Enlarge Image

[Figure 9-30](javascript://) shows the complete ERD after normalization. Now there are four entities: STUDENT, ADVISOR, COURSE, and GRADE (which is an associative entity). Note how [Figure 9-25](javascript://), which was drawn before GRADE was identified as an entity, shows that the M:N relationship between STUDENT and COURSE has been converted into two 1:M relationships: one relationship between STUDENT and GRADE and the other relationship between COURSE and GRADE.

**Figure 9-30**

The entity-relationship diagram for STUDENT, ADVISOR, and COURSE after normalization. The GRADE entry was identified during the normalization process. GRADE is an associative entity that links the STUDENT and COURSE tables.



Enlarge Image

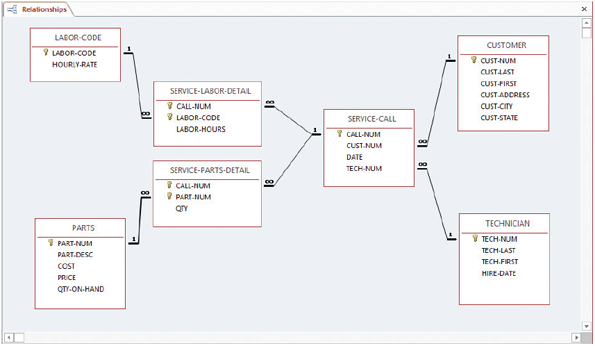
To create 3NF designs, the nature of first, second, and third normal forms must be understood. A systems analyst will encounter designs that are much more complex than the examples in this chapter.

### Example 2: Magic Maintenance

Magic Maintenance provides on-site service for electronic equipment. [Figure 9-31](javascript://) shows the overall database design that such a firm might use. The figure contains examples of many concepts described earlier. The database consists of seven separate tables, all joined by common fields, so they form an integral data structure.

**Figure 9-31**

A relational database design for a computer service company uses common fields to link the tables and form an overall data structure. Notice the one-to-many notation symbols and the primary keys, which are indicated with gold-colored key symbols.

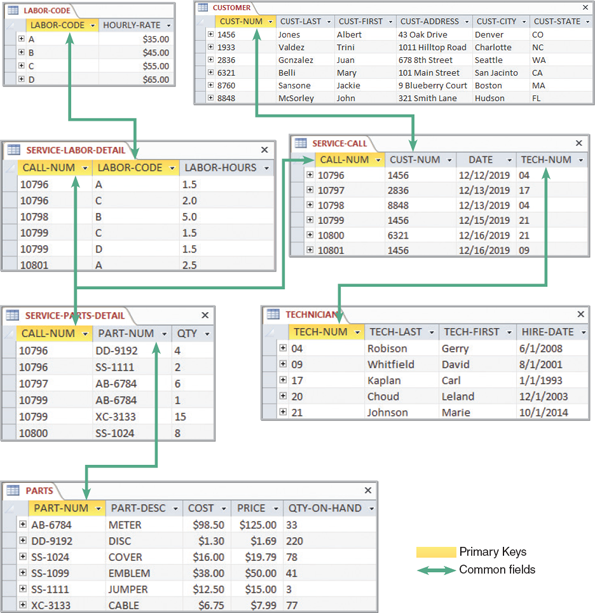


Enlarge Image

[Figure 9-32](javascript://) shows even more detail, including sample data, primary keys, and common fields. Note that the entities include customers, technicians, service calls, and parts. Other tables store data about labor and parts that are used on specific service calls. Also note that all tables use a single field as a primary key, except the SERVICE-LABOR-DETAIL and SERVICE-PARTS-DETAIL tables, where the primary key requires a combination of two fields to uniquely identify each record.

**Figure 9-32**

Sample data, primary keys, and common fields for the database shown in [Figure 9-31](javascript://). The design is in 3NF. Notice that all nonkey fields functionally depend on a primary key alone.



Enlarge Image

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**9.7**Codes

A [**code**](javascript://) is a set of letters or numbers that represents a data item. Codes can be used to simplify output, input, and data formats.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.7.1Overview of Codes

Because codes can represent data, they are encountered constantly in everyday life. Student numbers, for example, are unique codes to identify students in a school registration system. Three students with the name John Turner might be enrolled at the same school, but only one is student number 268960.

A postal code is another common example. A nine-digit postal code contains a lot of information. For example, the first digit identifies 1 of 10 main geographical areas in the United States. The combination of the next three digits identifies a major city or major distribution point. The fifth digit identifies an individual post office, an area within a city, or a specific delivery unit. The last four digits identify a post office box or a specific street address.

For example, consider the zip code 32901-6975 shown in [Figure 9-33](javascript://). This is called the “5+4” zip code format. The first digit, 3, indicates a broad geographical area in the southeastern United States. The next two digits, 29, indicate the area east of Orlando in Florida. The next two digits, 01, represent the city of Melbourne, Florida. The last four digits represent the specific location of the Florida institute of Technology: 150 W. University Blvd.

**Figure 9-33**

A zip code is an example of a significant digit code that uses groups and subgroups to store data. This example is for the zip code 32901-6975, which is the location of the Florida Institute of Technology in Melbourne, Florida.



Codes can be used in many ways. Because codes are shorter than the data they represent, they save storage space and costs, reduce data transmission time, and decrease data entry time. Codes also can be used to reveal or conceal information. The last two digits of a seven-digit part number, for example, might represent the supplier number or the maximum discount that a salesperson can offer.

Finally, codes can reduce data input errors in situations when the coded data is easier to remember and enter than the original source data, when only certain valid codes are allowed, and when something within the code itself can provide immediate verification that the entry is correct.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.7.2Types of Codes

Companies use many different coding methods. Because information system users must work with coded data, the codes should be easy to learn and apply. If it is planned to create new codes or change existing ones, comments and feedback from users should be obtained. The following section describes seven common coding methods.

1. [**Sequence codes**](javascript://) are numbers or letters assigned in a specific order. Sequence codes contain no additional information other than an indication of order of entry into the system. For example, a human resource system issues consecutive employee numbers to identify employees. Because the codes are assigned in the order in which employees are hired, the code can be used to see that employee number 584 was hired after employee number 433. The code, however, does not indicate the starting date of either person’s employment.
2. [**Block sequence codes**](javascript://) use blocks of numbers for different classifications. College course numbers usually are assigned using a block sequence code. 100-level courses, such as Chemistry 110 and Mathematics 125, are freshman-level courses, whereas course numbers in the 200s indicate sophomore-level courses. Within a particular block, the sequence of numbers can have some additional meaning, such as when English 151 is the prerequisite for English 152.
3. [**Alphabetic codes**](javascript://) use alphabet letters to distinguish one item from another based on a category, an abbreviation, or an easy-to-remember value, called a mnemonic code. Many classification codes fit more than one of the following definitions:
   1. [**Category codes**](javascript://) identify a group of related items. For example, a local department store uses a two-character category code to identify the department in which a product is sold: GN for gardening supplies, HW for hardware, and EL for electronics.
   2. [**Abbreviation codes**](javascript://) are alphabetic abbreviations. For example, standard state codes include NY for New York, ME for Maine, and MN for Minnesota. Some abbreviation codes are called [**mnemonic codes**](javascript://) because they use a specific combination of letters that are easy to remember. Many three-character airport codes such as those pictured in [Figure 9-34](javascript://) are mnemonic codes, such as ATL for Atlanta and MIA for Miami. However, some airport codes are not mnemonic, such as ORD (Chicago O’Hare) or MCO (Orlando).

**Figure 9-34**

Abbreviations for some of the world’s busiest airports.



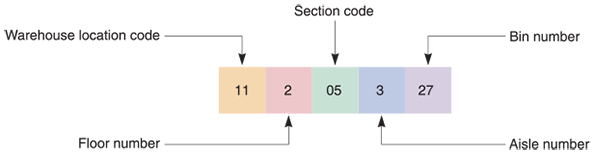
Enlarge Image

BLANKartist/ [Shutterstock.com](http://shutterstock.com/" \t "_blank)

1. [**Significant digit codes**](javascript://) distinguish items by using a series of subgroups of digits. Postal codes, for example, are significant digit codes. Other such codes include inventory location codes that consist of a two-digit warehouse code, followed by a one-digit floor number code, a two-digit section code, a one-digit aisle number, and a two-digit bin number code. [Figure 9-35](javascript://) illustrates the inventory location code 11205327. What looks like a large eight-digit number is actually five separate numbers, each of which has significance.

**Figure 9-35**

Sample of a code that uses significant digits to pinpoint the location of an inventory item.

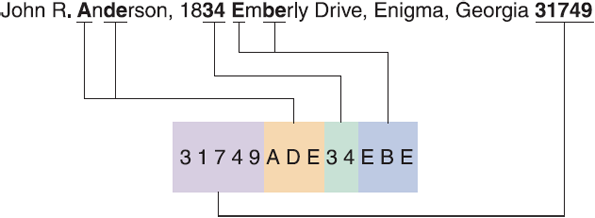


Enlarge Image

1. [**Derivation codes**](javascript://) combine data from different item attributes, or characteristics. Most magazine subscription codes are derivation codes. For example, one popular magazine uses a subscriber’s five-digit postal code, followed by the first, third, and fourth letters of the subscriber’s last name, the last two digits of the subscriber’s house number, and the first, third, and fourth letters of the subscriber’s street name. A sample is shown in [Figure 9-36](javascript://).

**Figure 9-36**

A magazine subscriber code is derived from various parts of the name and address.



Enlarge Image

1. [**Cipher codes**](javascript://) use a keyword to encode a number. A retail store, for example, might use a 10-letter word, such as CAMPGROUND, to code wholesale prices, where the letter C represents 1, A represents 2, and so on. Thus, the code, GRAND, indicates that the store paid $562.90 for the item.
2. [**Action codes**](javascript://) indicate what action is to be taken with an associated item. For example, a student records program might prompt a user to enter or click an action code such as D (to display a record), A (to add a record), and X (to exit the program).

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.7.3Designing Codes

Here are some code design suggestions:

* Keep codes concise. Do not create codes that are longer than necessary. For example, if a code is needed to identify each of 250 customers, a six-digit code is not needed.
* Allow for expansion. A coding scheme must allow for reasonable growth in the number of assigned codes. For example, if the company currently has eight warehouses, do not use a one-digit code for the warehouse number. If two more warehouses are added, the code must be increased to two digits or changed to a character code in order to identify each location. The rule also applies to using a single letter as a character code because more than 26 data items might be needed in the future. Of course, more characters can be added, which is just what the airline industry has done. Most airlines now use six-character codes that allow millions of combinations.
* Keep codes stable. Changes in codes can cause consistency problems and require data updates. During the changeover period, all the stored occurrences of a particular code and all documents containing the old code will have to change as users switch to the new code. Usually, both the old and new codes are used for an interim period, and special procedures are required to handle the two codes. For example, when telephone area codes change, either area code (old or new) can be used for a certain time period.
* Make codes unique. Codes used for identification purposes must be unique to have meaning. If the code HW can indicate hardware or houseware, the code is not very useful.
* Use sortable codes. If products with three-digit codes in the 100s or the 300s are of one type, while products with codes in the 200s are a different type, a simple sort will not group all the products of one type together. In addition, be careful that single-digit character codes will sort properly with double-digit codes—in some cases a leading zero must be added (01, 02, 03, etc.) to ensure that codes sort correctly.
* Use a simple structure. Do not code some part numbers with two letters, a hyphen, and one digit, and others with one letter, a hyphen, and two digits. Avoid allowing both letters and numbers to occupy the same positions within a code because some of those are easily confused. This situation might be a good place to use an input mask to assure that the correct data type is entered.
* Avoid confusion. It is easy to confuse the number zero (0) and the uppercase letter O, or the number one (1) with the lowercase letter L (l) or uppercase letter I. For example, the five-character code 5Z081 easily can be misread as 5ZO8I, or 52081.
* Make codes meaningful. Codes should be easy to remember, user-friendly, convenient, and easy to interpret. Using SW as a code for the southwest sales region, for example, has far more meaning than the code 14. Using ENG as the code for the English department is easier to interpret and remember than either XVA or 132.
* Use a code for a single purpose. Do not use a code to classify unrelated attributes. For example, if a single code is used to identify the combination of an employee’s department and the employee’s insurance plan type, users will have difficulty identifying all the subscribers of a particular plan, or all the workers in a particular department, or both. A separate code for each separate characteristic makes much more sense.
* Keep codes consistent. For example, if the payroll system already is using two-digit codes for departments, do not create a new, different coding scheme for the personnel system. If the two systems already are using different coding schemes, try to establish a consistent coding scheme.

**Case in Point 9.3**

### Madera Tools

Madera Tools operates a small business that specializes in hard-to-find woodworking tools. The firm advertises in various woodworking magazines and currently accepts mail and telephone orders. Madera is planning a website that will be the firm’s primary sales channel. The site will feature an online catalog, powerful search capabilities, and links to woodworking information and resources.

Madera has asked you, an IT consultant, whether a set of codes would be advantageous. What codes would you suggest? Provide at least two choices for a customer code and at least two choices for a product code. Be sure to describe your choices and provide some specific examples. Also include an explanation of why you selected these particular codes and what advantages they might offer.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**9.8**Data Storage and Access

Data storage and access involve strategic business tools, such as data warehousing and data mining software, as well as logical and physical storage issues, selection of data storage formats, and special considerations regarding storage of date fields.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.8.1Tools and Techniques

Companies use data warehousing and data mining as strategic tools to help manage the huge quantities of data they need for business operations and decisions. A large number of software vendors compete for business in this fast-growing IT sector.

### Data Warehousing

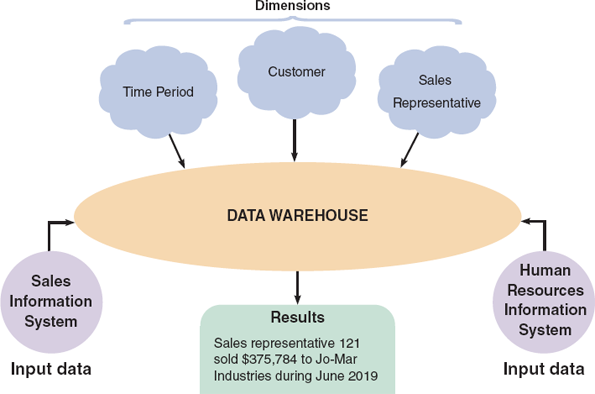
Large firms maintain many databases, which might or might not be linked together into an overall structure. To provide rapid access to this information, companies use software packages that organize and store data in special configurations called data warehouses. A [**data warehouse**](javascript://) is an integrated collection of data that can include seemingly unrelated information, no matter where it is stored in the company. Because it can link various information systems and databases, a data warehouse provides an enterprise-wide view to support management analysis and decision making.

A data warehouse allows users to specify certain dimensions, or characteristics. By selecting values for each characteristic, a user can obtain multidimensional information from the stored data. For example, in a typical company, most data is generated by transaction-based systems, such as order processing systems, inventory systems, and payroll systems. If a user wants to identify the customer on sales order 34071, he or she can retrieve the data easily from the order processing system by entering an order number.

On the other hand, suppose that a user wants to see June 2019 sales results for the sales rep assigned to Jo-Mar Industries. The data is stored in two different systems with different databases: the sales information system and the human resources information system, as shown in [Figure 9-37](javascript://). Without a data warehouse, it would be difficult for a user to extract data that spans several information systems and time frames. Rather than accessing separate systems, a data warehouse stores transaction data in a format that allows users to retrieve and analyze the data easily.

**Figure 9-37**

A data warehouse stores data from several systems. By selecting data dimensions, a user can retrieve specific information without having to know how or where the data is stored.



Enlarge Image

While a data warehouse typically spans the entire enterprise, many firms prefer to use a [**data mart**](javascript://), which is designed to serve the needs of a specific department, such as sales, marketing, or finance. Each data mart includes only the data that users in that department require to perform their jobs. There are pros and cons to both approaches, and the best solution usually depends on the specific situation.

Regardless of the overall approach, storing large quantities of data is like building a house—it doesn’t just happen. A well-constructed data warehouse needs an architecture that includes detailed planning and specifications.

### Data Mining

[**Data mining**](javascript://) software looks for meaningful data patterns and relationships. For example, data mining software could help a consumer products firm identify potential customers based on their prior purchases. Information about customer behavior is valuable, but data mining also raises serious ethical and privacy issues, such as the example in the Question of Ethics feature in this chapter.

The enormous growth in e-commerce has focused attention on data mining as a marketing tool. In an article called “Data Mining on the Web” that appeared in the January 2000 issue of New Architect, a web-based magazine, Dan R. Greening noted that web hosts typically possess a lot of information about visitors, but most of it is of little value. His article mentions that smart marketers and business analysts are using data mining techniques, which he describes as “machine learning algorithms that find buried patterns in databases, and report or act on those findings.” He concludes by saying that “the great advantage of web marketing is that you can measure visitor interactions more effectively than in brick-and-mortar stores or direct mail. Data mining works best when you have clear, measurable goals.” Some of the goals he suggests are as follows:

* Increase the number of pages viewed per session.
* Increase the number of referred customers.
* Reduce [**clicks to close**](javascript://), which means average page views to accomplish a purchase or obtain desired information.
* Increase checkouts per visit.
* Increase average profit per checkout.

This type of data gathering is sometimes called [**clickstream storage**](javascript://). Armed with this information, a skillful web designer could build a profile of typical new customers, returning customers, and customers who browse but do not buy. Although this information would be very valuable to the retailer, clickstream storage could raise serious legal and privacy issues if an unscrupulous firm sought to link a customer’s web behavior to a specific name or email address and then sell or otherwise misuse the information.

Because it can detect patterns and trends in large amounts of data, data mining is a valuable tool for managers. There is a well-known story about a chain of supermarkets that performed a detailed affinity analysis of purchases and found that beer and diapers were often purchased together. It is unclear whether or not this story is true, but without attempting to explain this correlation, the obvious tactic for a retailer would be to display these items in the same area of the store. This data mining technique relies on association rule learning and is often called [**market basket analysis**](javascript://).

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.8.2Logical Versus Physical Storage

It is important to understand the difference between logical storage and physical storage. [**Logical storage**](javascript://) refers to data that a user can view, understand, and access, regardless of how or where that information actually is organized or stored. In contrast, [**physical storage**](javascript://) is strictly hardware related because it involves the process of reading and writing binary data to physical media such as a hard drive, CD/DVD, or network-based storage device. For example, portions of a document might be stored in different physical locations on a hard drive, but the user sees the document as a single logical entity on the computer screen.

Logical storage consists of alphabetic and numeric [**characters**](javascript://), such as the letter A or the number 9. As described earlier in this chapter, a set of related characters forms a field, which describes a single characteristic, or attribute, of a person, a place, a thing, or an event. A field also is called a **data element** or a **data item**.

When designing fields, space should be provided for the largest values that can be anticipated, without allocating unnecessarily large storage capacities that will not be used. For example, suppose a customer order entry system is being designed for a firm with 800 customers. It would be a mistake to limit the customer number field to three or even four characters. Instead, a five-character field with leading zeros that could store customer numbers from 00001 to 99999 should be considered.

A mix of alphabetic and numeric characters can also be considered, which many people find easier to view and use. Alphabetic characters expand the storage capacity because there are 26 possible values for each character position. Most airlines now use six alphabetic characters as a record locator, which has over 300 million possible values.

A [**logical record**](javascript://) is a set of field values that describes a single person, place, thing, or event. For example, a logical customer record contains specific field values for a single customer, including the customer number, name, address, telephone number, credit limit, and so on. Application programs see a logical record as a group of related fields, regardless of how or where the data is stored physically.

The term record usually refers to a logical record. Whenever an application program issues a read or write command, the operating system supplies one logical record to the program or accepts one logical record from the program. The physical data might be stored on one or more servers, in the same building or thousands of miles away, but all the application program sees is the logical record—the physical storage location is irrelevant.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

## 9.8.3Data Coding

Computers represent data as [**bits**](javascript://) (short for binary digits) that have only two possible values: 1 and 0. A computer understands a group of bits as a digital code that can be transmitted, received, and stored. Computers use various data coding and storage schemes, such as EBCDIC, ASCII, and binary. A more recent coding standard called Unicode also is popular. Also, the storage of dates raises some design issues that must be considered.

### EBCDIC, ASCII, and Binary

[**EBCDIC**](javascript://) (pronounced EB-see-dik), which stands for Extended Binary Coded Decimal Interchange Code, is a coding method used on mainframe computers and high-capacity servers. [**ASCII**](javascript://) (pronounced ASK-ee), which stands for American Standard Code for Information Interchange, is a coding method used on most personal computers. EBCDIC and ASCII both require eight bits, or one [**byte**](javascript://), for each character. For example, the name Ann requires 3 bytes of storage, the number 12,345 requires 5 bytes of storage, and the number 1,234,567,890 requires 10 bytes of storage.

Compared with character-based formats, a [**binary storage format**](javascript://) offers a more efficient storage method because it represents numbers as actual binary values, rather than as coded numeric digits. For example, an integer format uses only 16 bits, or two bytes, to represent the number 12,345 in binary form. A long integer format uses 32 bits, or four bytes, to represent the number 1,234,567,890 in binary form.

### Unicode

[**Unicode**](javascript://) is a more recent coding standard that uses two bytes per character, rather than one. This expanded scheme enables Unicode to represent more than 65,000 unique, multilingual characters. Why is this important? Consider the challenge of running a multinational information system or developing a program that will be sold in Asia, Europe, and North America. Because it supports virtually all languages, Unicode has become a global standard.

Traditionally, domestic software firms developed a product in English and then translated the program into one or more languages. This process was expensive, slow, and error-prone. In contrast, Unicode creates translatable content right from the start. Today, most popular operating systems support Unicode, and the Unicode Consortium maintains standards and support, as shown in [Figure 9-38](javascript://).

**Figure 9-38**

Unicode is an international coding format that represents characters as integers, using 16 bits (two bytes) per character. The Unicode Consortium maintains standards and support for Unicode.



Enlarge Image

**Source:** Unicode Consortium

### Storing Dates

What is the best way to store dates? The answer depends on how the dates will be displayed and whether they will be used in calculations.

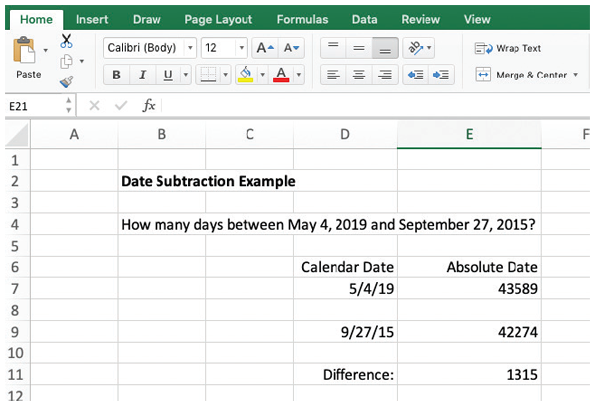
At the beginning of the twenty-first century, many firms that used only two digits to represent the year were faced with a major problem called the [**Y2K issue**](javascript://). Based on that experience, most date formats now are based on the model established by the [**International Organization for Standardization (ISO)**](javascript://), which requires a format of four digits for the year, two for the month, and two for the day (YYYYMMDD). A date stored in that format can be sorted easily and used in comparisons. If a date in ISO form is larger than another date in the same form, then the first date is later. For example, 20150504 (May 4, 2015) is later than 20130927 (September 27, 2013).

But what if dates must be used in calculations? For example, if a manufacturing order placed on June 23 takes three weeks to complete, when will the order be ready? If a payment due on August 13 is not paid until April 27 of the following year, exactly how late is the payment and how much interest is owed? In these situations, it is easier to use absolute dates.

An [**absolute date**](javascript://) is the total number of days from some specific base date. To calculate the number of days between two absolute dates, one date is subtracted from the other. For example, if the base date is January 1, 1900, then May 4, 2015, has an absolute date of 42128. Similarly, September 27, 2013, has an absolute date value of 41544. If the earlier date value is subtracted from the later one, the result is 584 days. A spreadsheet can be used to determine and display absolute dates easily, as shown in [Figure 9-39](javascript://).

**Figure 9-39**

Microsoft Excel uses absolute dates in calculations. In this example, May 4, 2019, is displayed as 43589, and September 27, 2015, is displayed as 42274. The difference between the dates is 1315 days.



Enlarge Image

Scott Tilley

Change font size

[**help**](javascript://)

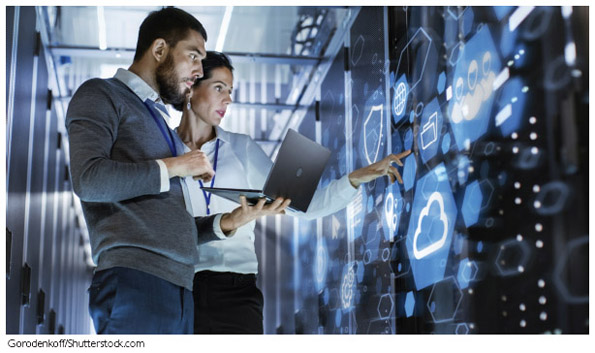
[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

# 9.9Data Control

Just as it is important to secure the physical part of the system, as shown in [Figure 9-40](javascript://), file and database control must include all measures necessary to ensure that data storage is correct, complete, and secure. File and database control is also related to input and output techniques discussed earlier.

**Figure 9-40**

In addition to network monitoring, system security includes access codes, data encryption, passwords, and audit trails.



Enlarge Image

Gorodenkoff/ [Shutterstock.com](http://shutterstock.com/" \t "_blank)

A well-designed DBMS must provide built-in control and security features, including subschemas, passwords, encryption, audit trail files, and backup and recovery procedures to maintain data. The analyst’s main responsibility is to ensure that the DBMS features are used properly.

Earlier in this chapter, it was explained that a subschema can be used to provide a limited view of the database to a specific user or level of users. Limiting access to files and databases is the most common way of protecting stored data. Users must furnish a proper user ID and password to access a file or database. Different privileges, also called [**permissions**](javascript://), can be associated with different users, so some employees can be limited to read-only access, while other users might be allowed to update or delete data. For highly sensitive data, additional access codes can be established that restrict specific records or fields within records. Stored data also can be encrypted to prevent unauthorized access. **Encryption** is the process of converting readable data into unreadable characters to prevent unauthorized access to the data.

All system files and databases must be backed up regularly and a series of [**backup**](javascript://) copies must be retained for a specified period of time. In the event of a file catastrophe, [**recovery procedures**](javascript://) can be used to restore the file or database to its current state at the time of the last backup. [**Audit log files**](javascript://), which record details of all accesses and changes to the file or database, can be used to recover changes made since the last backup. [**Audit fields**](javascript://), which are special fields within data records to provide additional control or security information, can also be included. Typical audit fields include the date the record was created or modified, the name of the user who performed the action, and the number of times the record has been accessed.

### A Question of Ethics

* [iStock.com](https://istock.com/" \t "_blank)/faberfoto\_itTip Top Toys is a relatively small division of Worldwide Enterprises. Worldwide has nine other divisions, which include insurance, health-care products, and financial planning services, to name a few.

The corporate marketing director for Worldwide has requested Tip Top’s customer shopping data to target people who might be likely to purchase items or services from other Worldwide divisions. The database manager is not totally comfortable with this and pointed out Tip Top’s web privacy policy, which states “Tip Top Toys, a division of Worldwide Enterprises, will not share personal data with other companies without a customer’s consent.”

The marketing director replied that the statement only applies to outside companies—not other Worldwide divisions. He said he checked with the corporate legal department, and they agreed. The database manager replied, “Even if it is legally OK, it’s not the right thing to do. Many people take our statement to mean that their data does not leave Tip Top. At the very least, we should give customers a choice, and share the data only with their consent.”

Do you agree with the marketing director? Why or why not?

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

**9.10**Summary

This chapter continued the study of the systems design phase of the SDLC. It was explained that files and tables contain data about people, places, things, or events that affect the information system. File-oriented systems, also called file processing systems, manage data stored in separate.

A database consists of linked tables that form an overall data structure. A DBMS is a collection of tools, features, and interfaces that enable users to add, update, manage, access, and analyze data in a database.

DBMS designs are more powerful and flexible than traditional file-oriented systems. A database environment offers scalability, support for organization-wide access, economy of scale, data sharing among user groups, balancing of conflicting user requirements, enforcement of standards, controlled redundancy, effective security, flexibility, better programmer productivity, and data independence. Large-scale databases are complex and require extensive security and backup/recovery features.

DBMS components include interfaces for users, DBAs, and related systems; a DML; a schema; and a physical data repository. Other data management techniques include data warehousing, which stores data in an easily accessible form for user access, and data mining, which looks for meaningful patterns and relationships among data. Data mining also includes clickstream storage, which records how users interact with a site, and market basket analysis, which can identify product relationships and consumer buying patterns.

In a web-based design, the Internet serves as the front end, or interface, for the DBMS. Access to the database requires only a web browser client and an Internet connection. Middleware can interpret client requests in HTML form and translate the requests into commands that the database can execute. Web-based data must be secure, yet easily accessible to authorized users. To achieve this goal, well-designed systems provide security at three levels: the database itself, the web server, and the telecommunication links that connect the components of the system

In an information system, an entity is a person, a place, a thing, or an event for which data is collected and maintained. A field, or attribute, is a single characteristic of an entity. A record, or tuple, is a set of related fields that describes one instance of an entity. Data is stored in files (in a file-oriented system) and tables (in a database environment).

A primary key is the field or field combination that uniquely and minimally identifies a specific record; a candidate key is any field that could serve as a primary key. A foreign key is a field or field combination that must match the primary key of another file or table. A secondary key is a field or field combination used as the basis for sorting or retrieving records.

An ERD is a graphic representation of all system entities and the relationships among them. The ERD is based on entities and data stores in DFDs prepared during the systems analysis phase. The three basic relationships represented in an ERD are one-to-one (1:1), one-to-many (1:M), and many-to-many (M:N). In a M:N relationship, the two entities are linked by an associative entity.

The relationship between two entities also is referred to as cardinality. A common form of cardinality notation is called crow’s foot notation, which uses various symbols to describe the characteristics of the relationship.

Normalization is a process for avoiding problems in data design. A 1NF record has no repeating groups. A record is in 2NF if it is in 1NF and all nonkey fields depend on the entire primary key. A record is in 3NF if it is in 2NF and if no field depends on a nonkey field.

Data design tasks include creating an initial ERD; assigning data elements to an entity; normalizing all table designs; and completing the data dictionary entries for files, records, and data elements.

A code is a set of letters or numbers used to represent data in a system. Using codes can speed up data entry, reduce data storage space, and reduce transmission time. Codes also can be used to reveal or to conceal information. The main types of codes are sequence codes, block sequence codes, classification codes, alphabetic codes (e.g., category codes, abbreviation codes, and mnemonic codes), significant digit codes, derivation codes, cipher codes, and action codes.

Logical storage is information seen through a user’s eyes, regardless of how or where that information actually is organized or stored. Physical storage is hardware related and involves reading and writing binary data to physical media. A logical record is a related set of field values that describes a single person, place, thing, or event. Data storage formats include EBCDIC, ASCII, binary, and Unicode. Dates can be stored in several formats, including ISO and absolute format.

File and database control measures include limiting access to the data, data encryption, backup/recovery procedures, audit trail files, and internal audit fields.

Change font size

[**help**](javascript://)

[Main content](https://ng.cengage.com/static/nbreader/ui/apps/nbreader/nbreader.html#header)

# Chapter Review

## **Key Terms**

* [**1:1**](javascript://)
* [**1:M**](javascript://)
* [**Abbreviation codes**](javascript://)
* [**absolute date**](javascript://)
* [**Action codes**](javascript://)
* [**Alphabetic codes**](javascript://)
* [**ASCII**](javascript://)
* [**associative entity**](javascript://)
* **attribute**
* [**audit fields**](javascript://)
* [**audit log files**](javascript://)
* [**backup**](javascript://)
* [**binary storage format**](javascript://)
* [**bits**](javascript://)
* [**Block sequence codes**](javascript://)
* [**byte**](javascript://)
* [**candidate key**](javascript://)
* **cardinality**
* [**cardinality notation**](javascript://)
* [**category codes**](javascript://)
* [**characters**](javascript://)
* [**cipher codes**](javascript://)
* [**clicks to close**](javascript://)
* [**clickstream storage**](javascript://)
* [**code**](javascript://)
* [**combination key**](javascript://)
* [**common field**](javascript://)
* [**composite key**](javascript://)
* [**concatenated key**](javascript://)
* [**crow’s foot notation**](javascript://)
* **data element**
* **data item**
* [**data manipulation language (DML)**](javascript://)
* [**data mart**](javascript://)
* [**data mining**](javascript://)
* **data structure**
* [**data warehouse**](javascript://)
* [**database administrator (DBA)**](javascript://)
* [**database management system (DBMS)**](javascript://)
* [**Derivation codes**](javascript://)
* [**EBCDIC**](javascript://)
* [**economy of scale**](javascript://)
* **encryption**
* **entity**
* [**entity-relationship diagram (ERD)**](javascript://)
* **field**
* [**file**](javascript://)
* [**file-oriented systems**](javascript://)
* [**first normal form (1NF)**](javascript://)
* [**foreign key**](javascript://)
* [**functionally dependent**](javascript://)
* [**International Organization for Standardization (ISO)**](javascript://)
* [**java database connectivity (JDBC)**](javascript://)
* [**key fields**](javascript://)
* [**logical record**](javascript://)
* [**logical storage**](javascript://)
* [**M:N**](javascript://)
* [**many-to-many relationship**](javascript://)
* [**market basket analysis**](javascript://)
* **middleware**
* [**mnemonic codes**](javascript://)
* [**multivalued key**](javascript://)
* [**nonkey field**](javascript://)
* [**normalization**](javascript://)
* [**one-to-many relationship**](javascript://)
* [**one-to-one relationship**](javascript://)
* [**Open database connectivity (ODBC)**](javascript://)
* [**orphan**](javascript://)
* [**permissions**](javascript://)
* [**physical storage**](javascript://)
* [**primary key**](javascript://)
* [**query by example (QBE)**](javascript://)
* [**query language**](javascript://)
* [**record**](javascript://)
* [**recovery procedures**](javascript://)
* [**referential integrity**](javascript://)
* [**relational database**](javascript://)
* [**relational model**](javascript://)
* [**repeating group**](javascript://)
* **scalability**
* [**schema**](javascript://)
* [**second normal form (2NF)**](javascript://)
* [**secondary key**](javascript://)
* [**Sequence codes**](javascript://)
* [**Significant digit codes**](javascript://)
* [**standard notation format**](javascript://)
* [**Structured Query Language (SQL)**](javascript://)
* [**subschema**](javascript://)
* [**table**](javascript://)
* [**table design**](javascript://)
* [**third normal form (3NF)**](javascript://)
* [**tuple**](javascript://)
* [**Unicode**](javascript://)
* **Unified Modeling Language (UML)**
* [**unnormalized**](javascript://)
* [**Y2K issue**](javascript://)

Change font size

[**help**](javascript://)