Database Systems: Design, Implementation, and Management
9th Edition
Peter Rob & Carlos Coronel

Chapter 1
Introduction

File Systems and Databases

1.1. The historical roots of the database: Files and File systems

In the recent past, a manager of almost any small organization was able to keep track of necessary data by using a manual file system. Such a file system was traditionally composed of a collection of file folders; each properly tagged and kept in a filing cabinet.

Organization of the data within the file folders was determined by the data's expected use. Ideally, the contents of each file folder were logically related. For example, a file folder in a doctor's office might contain patient data, one file folder for each patient. All the data in that file folder described only that particular patient's medical history.

As long as a data collection was relatively small and an organization's managers had few reporting requirements, the manual system served its role well as a data repository. However, as organizations grew and as reporting requirements became more complex, keeping track of data in a manual file system became more difficult. In fact, finding and using data in growing collections of file folders became such a timeconsuming and cumbersome task that it became less and less likely that such data would ever generate useful information.

Unfortunately, report generation from a manual file system can be slow and cumbersome. In fact, some business managers faced government-imposed reporting requirements that required weeks of intensive effort each quarter, even when a well-designed manual system was used. Consequently, the pressure built to design a computer based system that would track data and produce required reports.

1.2. File System Critique (problems):

We can see the problems with the straight file-processing approach:

1.2.1. File System Data Management

- As the number of files expands, system administration becomes difficult, too. Each file must have its own file management system, composed of programs that allow the user to:
    - Create the file structure,
    - Add data to the file.
    - Delete data from the file.
    - Modify the data contained in the file.
    - List the file contents.
- **Security features** - such as effective password protection, locking out parts of files or parts of the system itself. Every user of the system should be able to access only the data they are permitted to see.
- **Multiple users:** Want concurrency for faster response time.
- **Integrity problems:** Data may be required to satisfy consistency constraints.

1.2.2. Structural and Data Dependence

Structural Dependence: A change in any file's structure requires the modification of all programs using that file.
Data Dependence: A change in any file's data characteristics requires changes in all data access programs. Significance of data dependence is that there exists a difference between the data logical format (how the human being views the data) and the data physical format (how the computer "sees" the data).
Data dependence makes file systems extremely cumbersome from a programming and data management point of view.

1.2.3. Field Definitions and Naming Conventions. A good (flexible) record definition anticipates reporting requirements by breaking up fields into their components.

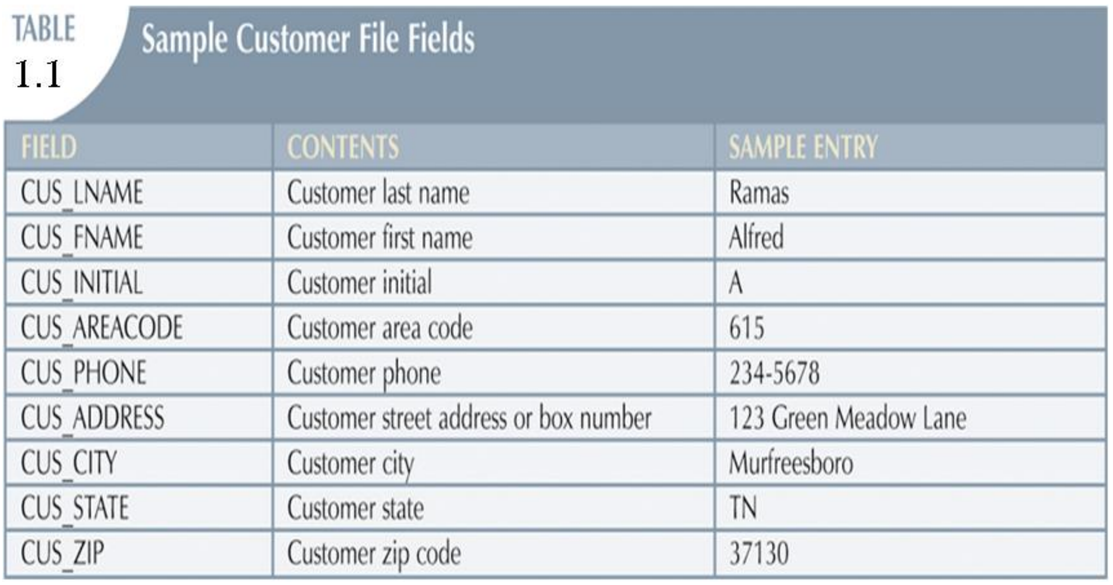


TABLE 1.1 Sample Customer File Fields

FIELD	CONTENTS	SAMPLE ENTRY
CUS_LNAME	Customer last name	Ramas
CUS_FNAME	Customer first name	Alfred
CUS_INITIAL	Customer initial	A
CUS_AREACODE	Customer area code	615
CUS_PHONE	Customer phone	234-5678
CUS_ADDRESS	Customer street address or box number	123 Green Meadow Lane
CUS_CITY	Customer city	Murfreesboro
CUS_STATE	Customer state	TN
CUS_ZIP	Customer zip code	37130

Selecting proper field names is very important. For example, make sure that the field names must be as descriptive as possible within restrictions. It is not obvious that the field name REN represents the customer's insurance renewal date. Using the field name CUSRENEWDATE would be better for two reasons. First, the prefixes CUS can be used as an indicator of the field's origin, which is the CUSTOMER file. Therefore, we know that the field in question yields a CUSTOMER property. Second, the RENEW DATE portion of the field name is more descriptive of the field's contents. By using proper naming conventions, the file structure becomes self-documenting. That is, by simply looking at the field names we are able to determine which files the fields belong to and what information is likely to be contained within those fields.

1.2.4. DATA REDUNDANCY:

Data redundancy: **Same information may be duplicated in several places.**

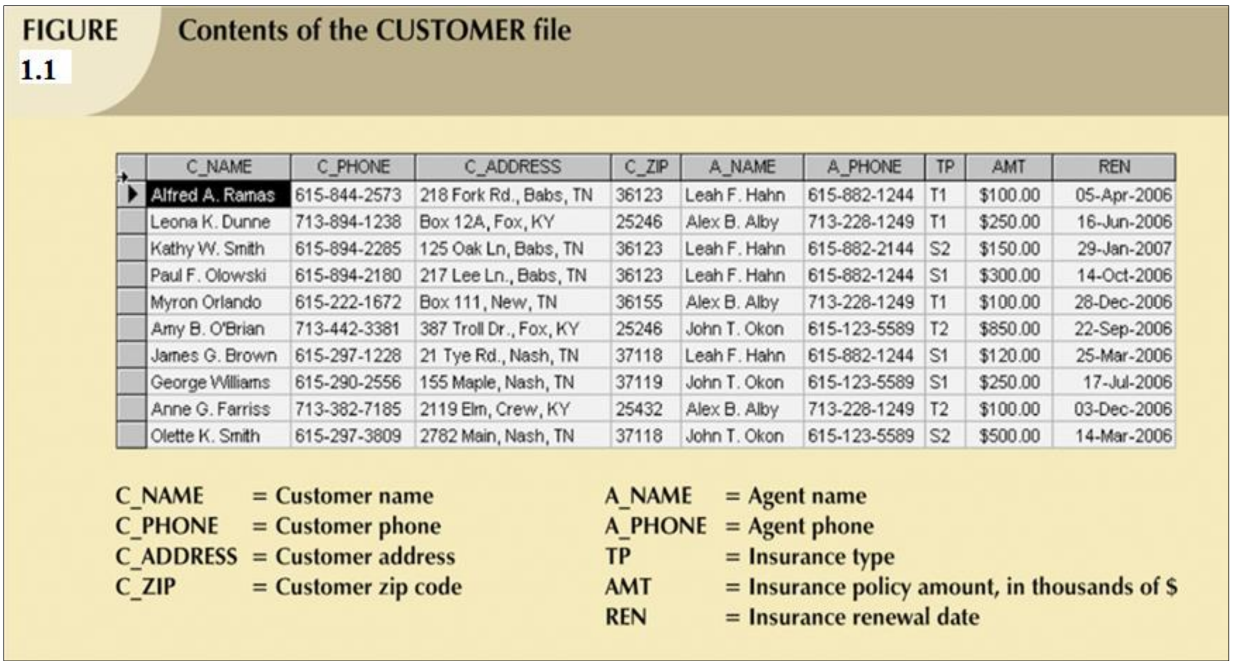
If the file system environment makes it difficult to pool data, it is likely that the same data are stored in many different locations. For example, in Figures 1.1 and 1.2, the agent names and phone numbers occur in both the CUSTOMER and the AGENT files. You need only one correct copy of the agent names and phone numbers. Having them occur in more than one place produces data redundancy. Uncontrolled data redundancy sets the stage for:

1. Data inconsistency. All copies may not be updated properly. Data inconsistency exists when different and conflicting versions of the same data appear in different places. For example, suppose we change an agent's phone number or address in the AGENT file. If we forget to make corresponding changes in the CUSTOMER file, the files contain different data for the same agent. Reports yield inconsistent results, depending on which version of the data is used. Data that display data inconsistency are also referred to as data that lack data integrity.

2. ***Data anomalies.*** The dictionary defines "anomaly" as an abnormality. Ideally, a field value change should be made only in a single place. Data redundancy, however, fosters an abnormal condition by forcing field value changes in many different locations. The data anomalies found in Figure 1.3 are commonly defined as:

- ***Modification anomalies.*** If agent Leah F. Hahn has a new phone number, that new number must be entered in each of the CUSTOMER file records in which Ms. Hahn's phone number is shown. In this case, only three changes must be made. In a large file system, such changes might occur in hundreds or even thousands of records. Clearly, the potential for data inconsistencies is great.
- ***Insertion anomalies.*** To add each new customer in the CUSTOMER file, we must also add the corresponding agent data. If we add several hundred new customers, we must also enter several hundred agent names and telephone numbers. Again, the potential for creating data inconsistencies is great.
- ***Deletion anomalies.*** If agent Alex B. Alby quits and is deleted from the payroll, all the customers in the CUSTOMER file become linked to a nonexistent agent. To resolve this problem, we must modify all records in which Mr. Alby's name and phone number appear.

FIGURE 1.1 Contents of the CUSTOMER file



C_NAME	C_PHONE	C_ADDRESS	C_ZIP	A_NAME	A_PHONE	TP	AMT	REN
Alfred A. Ramas	615-844-2573	218 Fork Rd., Babs, TN	36123	Leah F. Hahn	615-882-1244	T1	$100.00	05-Apr-2006
Leona K. Dunne	713-894-1238	Box 12A, Fox, KY	25246	Alex B. Alby	713-228-1249	T1	$250.00	16-Jun-2006
Kathy W. Smith	615-894-2285	125 Oak Ln, Babs, TN	36123	Leah F. Hahn	615-882-2144	S2	$150.00	29-Jan-2007
Paul F. Olowski	615-894-2180	217 Lee Ln., Babs, TN	36123	Leah F. Hahn	615-882-1244	S1	$300.00	14-Oct-2006
Myron Orlando	615-222-1672	Box 111, New, TN	36155	Alex B. Alby	713-228-1249	T1	$100.00	28-Dec-2006
Amy B. O'Brian	713-442-3381	387 Troll Dr., Fox, KY	25246	John T. Okon	615-123-5589	T2	$850.00	22-Sep-2006
James G. Brown	615-297-1228	21 Tye Rd., Nash, TN	37118	Leah F. Hahn	615-882-1244	S1	$120.00	25-Mar-2006
George Williams	615-290-2556	155 Maple, Nash, TN	37119	John T. Okon	615-123-5589	S1	$250.00	17-Jul-2006
Anne G. Farriss	713-382-7185	2119 Elm, Crew, KY	25432	Alex B. Alby	713-228-1249	T2	$100.00	03-Dec-2006
Olette K. Smith	615-297-3809	2782 Main, Nash, TN	37118	John T. Okon	615-123-5589	S2	$500.00	14-Mar-2006

C_NAME	= Customer name	A_NAME	= Agent name
C_PHONE	= Customer phone	A_PHONE	= Agent phone
C_ADDRESS	= Customer address	TP	= Insurance type
C_ZIP	= Customer zip code	AMT	= Insurance policy amount, in thousands of $
		REN	= Insurance renewal date

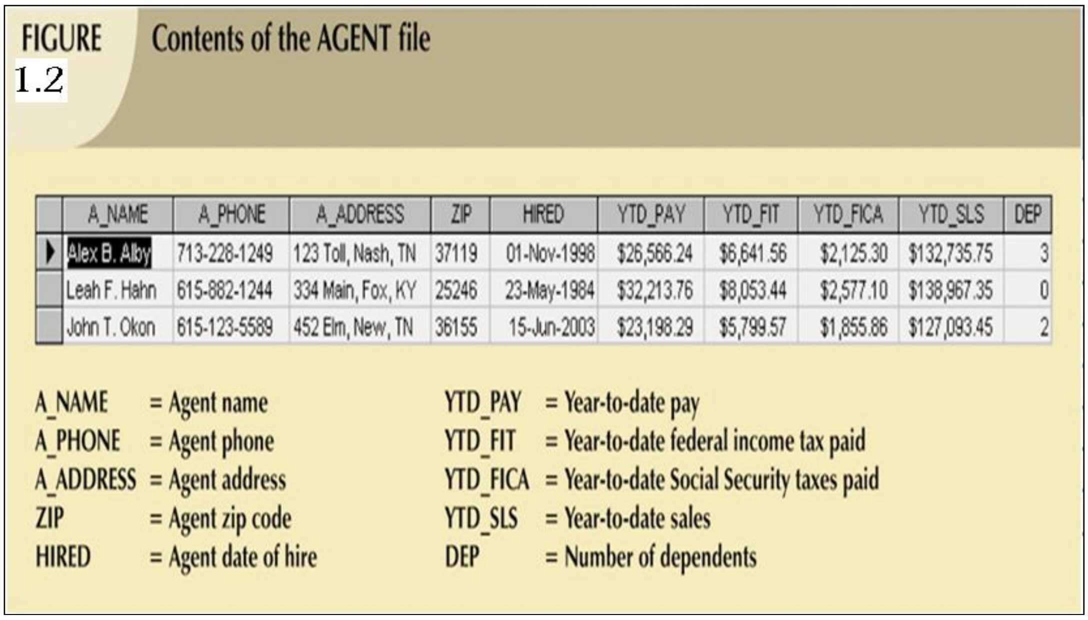


FIGURE 1.2 Contents of the AGENT file

A_NAME	A_PHONE	A_ADDRESS	ZIP	HIRED	YTD_PAY	YTD_FIT	YTD_FICA	YTD_SLS	DEP
Alex B. Alby	713-228-1249	123 Toll, Nash, TN	37119	01-Nov-1998	$26,566.24	$6,641.56	$2,125.30	$132,735.75	3
Leah F. Hahn	615-882-1244	334 Main, Fox, KY	25246	23-May-1984	$32,213.76	$8,053.44	$2,577.10	$138,967.35	0
John T. Okon	615-123-5589	452 Elm, New, TN	36155	15-Jun-2003	$23,198.29	$5,799.57	$1,855.86	$127,093.45	2

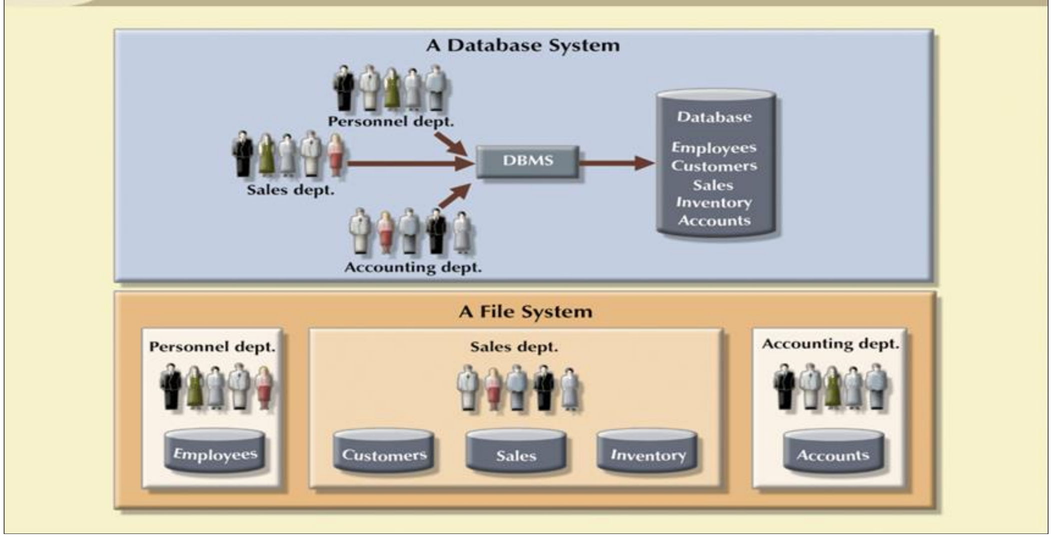
A_NAME	= Agent name	YTD_PAY	= Year-to-date pay
A_PHONE	= Agent phone	YTD_FIT	= Year-to-date federal income tax paid
A_ADDRESS	= Agent address	YTD_FICA	= Year-to-date Social Security taxes paid
ZIP	= Agent zip code	YTD_SLS	= Year-to-date sales
HIRED	= Agent date of hire	DEP	= Number of dependents

These problems and others led to the development of **database management systems.**

1.3. Database System

The problems inherent in file systems make using a database system very desirable. Unlike the file system, with its many separate and unrelated files, the database consists of logically related data stored in a single data repository. Therefore, the database represents a change in the way end user data are stored, accessed, and managed. The database's DBMS, shown in Figure 1.6, provides numerous advantages over file system management, by making it possible to eliminate most of the file system's data inconsistency, data anomalies, data dependency, and structural dependency problems.

FIGURE 1.3 Contrasting database and file systems



Remember that the DBMS is just one of several crucial components of a database system. Perhaps it is even appropriate to refer to the DBMS as the database system's heart. However, just as it takes more than a heart alone to make a human being function, it takes more than a DBMS to make a database system function.

Database is a collection of data or information that can be stored, sorted, organized and retrieved. Your local telephone book, your Rolodex file, and the card catalog at your local library are all examples of a database.

Traditional databases are organized by ***fields, records,*** and ***files (tables).*** A *field* is single piece of information; a *record* is one complete set of fields; and a *file* is a collection of records. For example, a telephone book is analogous to a file. It contains a list of records, each of which consists of three fields: name, address, and telephone number.

To access information from a database, you need a database management system (DBMS). This is a collection of programs that enables you to enter, organize, and select data in a database.

1.3.1. A database management system (DBMS)

DBMS is a collection of programs that manages the database structure and controls access to I the data stored in the database. The DBMS makes it possible to share the data in the database among multiple applications or users. Because data are the crucial (H^) raw material from which information is derived, there are many good reasons why DBMSs are important in our information-based

The goal of a **DBMS** is to provide an environment that is both **convenient** and **efficient** to use in:

- Retrieving information from the database.
- Storing information into the database.

FIGURE 1.4 The DBMS manages the interaction between the end user and the database

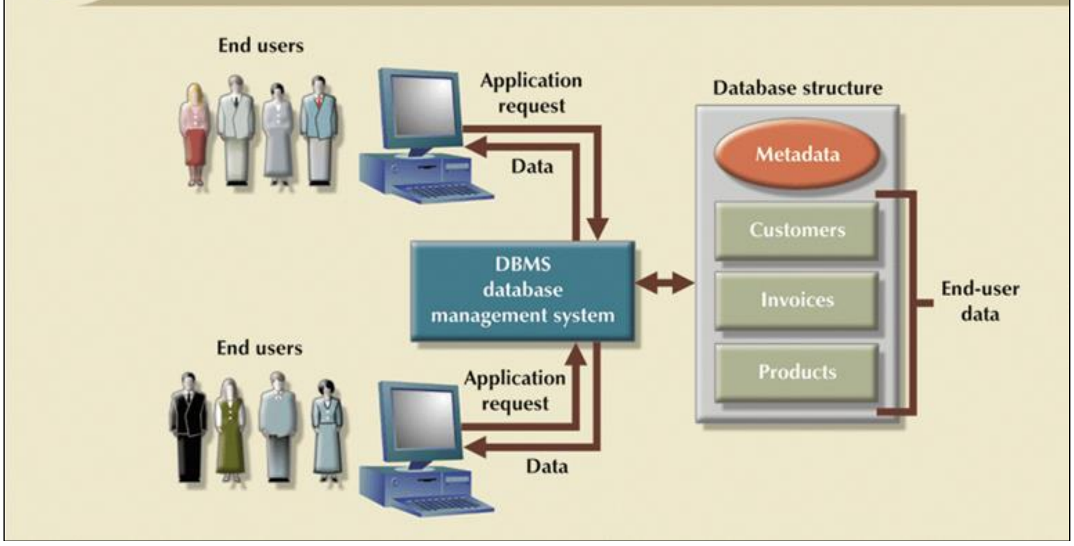


Figure 1.4 illustrates the concept that the DBMS stands between the database and the user(s). In effect, the DBMS serves as the intermediary between the user and the database by translating user requests into the complex code required to fulfill those requests. The DBMS hides much of the database's internal complexity from the application programs that use the database. The application program might be written by a programmer, using a programming language such as COBOL, Microsoft Access, and Oracle or it might be created through a DBMS utility program.

1.3.2. DBMS Functions

A DBMS perform several important functions that guarantee the integrity and consistency of the data in the database. Most of these functions are transparent to end users. These functions include:

1- **Data Dictionary Management:** The DBMS requires that definitions of the data elements and their relationships (metadata: data about data) be stored in a data dictionary.
2- **Data Storage Management:** The DBMS creates the complex structures required for data storage.data structures that are required to store the data.
3- **Data Transformation and Presentation:** The DBMS transforms entered data to conform to the data structures that are required to store the data.
4- **Security Management:** The **DBMS** creates a security system that enforces user security and data privacy within the database. Security rules determine which users can access the database, which data item each user may access, and which data operations the user may perform.
5- **Multi-User Access Control:** The DBMS creates the complex structures that allow multi-user access to the data. In order to provide data integrity and data consistency,

the DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently and still guarantee the integrity of the database.

6- **Backup and Recovery Management:** The DBMS provides backup and data recovery procedures to ensure data safety and integrity.

7- **Data Integrity Management:** The DBMS promotes and enforces integrity rules to eliminate data integrity problems, thus minimizing data redundancy and maximizing data consistency.

8- **Database Access Languages (DDL and DML) and Application Programming Interfaces:** The DBMS provides data access via a query language. A query language is a nonprocedural language; which contains two components: a Data Definition language (DDL) and a Data Manipulation Language (DML).

9- **Database Communication Interfaces:** Current-generation DBMSs provide special communications routines designed to allow the database to accept end user requests within a computer network environment.

1.3.3. The Database System Environment:

- **Hardware:** Computer, Peripherals.
- **Software:** refers to the collection of programs that are used by the computers within the database system. The three types of software are:
  - **Operating systems software:** manages all hardware components and makes it possible for all other software to run on the computers. Such as DOS, OS/2, UNIX, XP windows.
  - **DBMS software:** manages the database within the database system. Such as Oracle, Access.
  - **Applications programs and utilities software:** are used to access and manipulate the data in the DBMS and to manage the computer environment in which the data access and manipulation take place.
- **People (User): Systems administrators:** oversee the database system's general operations.

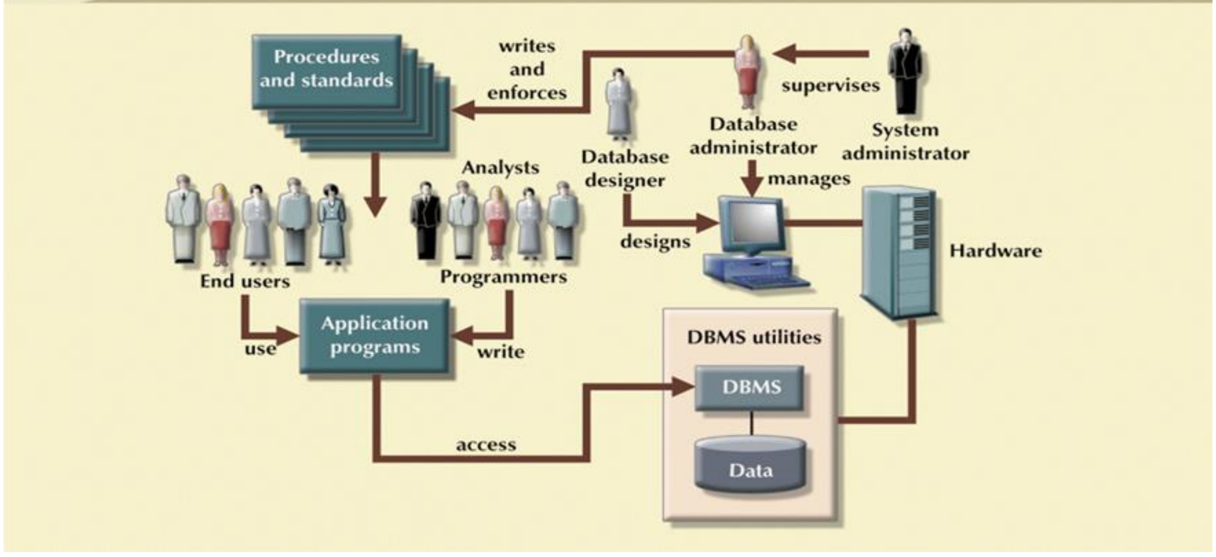
  Database administrators (DBAs): is a person having central control over data and programs accessing that data.

  Database designers: design the database structure
  Systems analysts and programmers: design and implement the application programs.
  End users: are the people who use the applications programs to run the organization's daily operations

- **Procedures:** Instructions and rules that govern the design and use of the database system.

- **Data:** Collection of facts stored in the database.

FIGURE 1.5 The database system environment



- **1.3.4. Types of Database Systems:** Can be classified by user:
  - **Single-user:** Supports only one user at a time
  - **Desktop:** Single-user database running on a personal computer
  - **Multi-user:** Supports multiple users at the same time
  - **Workgroup:** Multi-user database that supports a small group of users or a single department
  - **Enterprise:** Multi-user database that supports a large group of users or an entire organization

  Can be classified by location:
  - **Centralized:** Supports data located at a single site
  - **Distributed:** Supports data distributed across several sites

The DBMS, on which the database system is based, can be classified according on the number of users, the database site location(s)

1- The number of users determines whether the DBMS is classified as single-user or multi-user. A single-user DBMS supports only one user at a time. In other words, if user A is using the database, users B and C must wait until user A has completed database work. In contrast, a multi-user DBMS supports multiple users at the same time. If the multi-user database supports a relatively small number of users or a specific department within an organization, it is called a workgroup database.

The DBMS, on which the database system is based, can be classified according on the number of users, the database site location(s).

1. The number of users determines whether the DBMS is classified as single-user or multi-user. A single-user DBMS supports only one user at a time. In other words, if user A is using the database, users B and C must wait until user A has completed database work. In contrast, a multi-user DBMS supports multiple users at the same time. If the multi-user database supports a relatively small number of users or a specific department within an organization, it is called a workgroup database.

2. The database site location might also be used to classify the DBMS, for example, a DBMS system that supports a database located at a single site is called a centralized DBMS. A DBMS that supports a database distributed across several different sites is called a distributed DBMS.

1.4. Data Independence

1. The ability to modify a scheme definition in one level without affecting a scheme definition in a higher level is called **data independence.**
2. There are two kinds:
   - **Physical data independence:**
     1- The ability to modify the physical scheme without causing application programs to be rewritten.
     2- Modifications at this level are usually to improve performance.
   - **Logical data independence:**
     1-The ability to modify the conceptual scheme without causing application programs to be rewritten
     2- Usually done when logical structure of database is altered.

3- Logical data independence is more difficult to achieve than physical data independency since application programs are usually heavily dependent on the logical structure of the data access. The concept of data independence is similar in many respects to the concept of *abstract data types* in modern programming languages.

1.5. Database Models

A **database model** is a collection of logical constructs used to represent the data structure and the data relationships found within the database. Database models can be grouped into two categories: ***conceptual models*** and ***implementation models.***

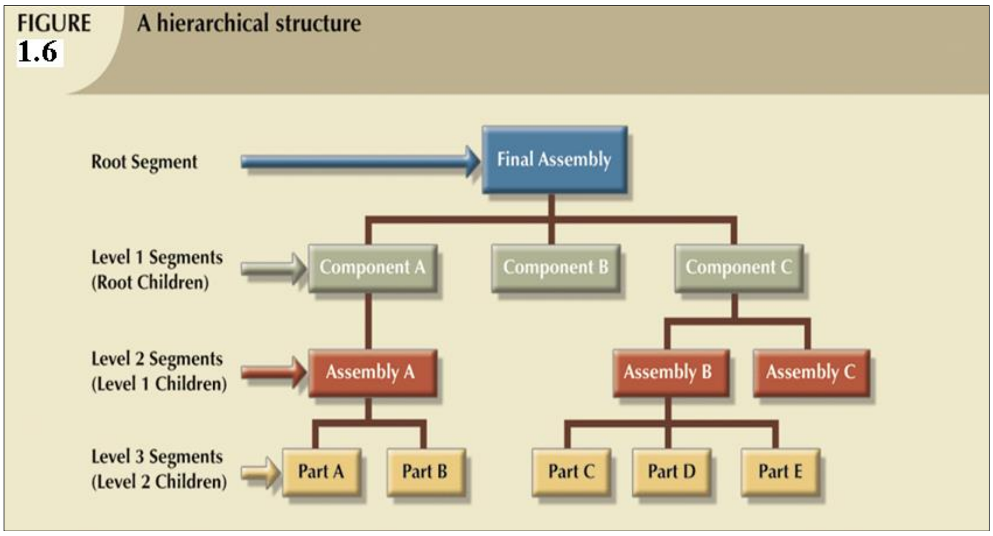
- The **conceptual model** focuses on the logical nature of the data representation. Therefore, the conceptual model is concerned with *what* is represented in the database, rather than with *how* it is represented. Conceptual models include the entity relationship (E-R) model.
- The **implementation model** places the emphasis on how the data are represented in the database or on how the data structures are implemented to represent what is modeled. Implementation models include the hierarchical database model, the network database model, the relational database model, and the object-oriented database model. Conceptual models use three types of relationships to describe associations among data: one-to-many, many-to many, and one-to-one. Database designers usually use the shorthand notations 1:M, M:N, and 1:1 for them, respectively. (Although the M:N notation is a standard label for the many-to-many relationship, the label M:M may also be used.) The following examples illustrate the distinctions among the three:

1. ***One-to-many relationship.*** Each record in a table is related to many records in another table for example, a painter paints many different paintings, but each one of them is painted by only that painter. Thus the painter (the "one") is related to the paintings (the "many"). Therefore, database designers label the relationship "PAINTER paints PAINTING" as 1:M. Similarly, a customer account (the "one") might contain many invoices, but those invoices (the "many") are related to only a single customer account. The "CUSTOMER generates INVOICE" relationship would also be labeled 1:M.
2. ***Many-to-many relationship.*** Many records in a table are related to many records in another table for example, Student can take many courses, and each course can be taken by many students, thus yielding the M:N relationship label for the relationship expressed by "STUDENT takes COURSE."

3. ***One-to-one relationship.*** Each record in a table is related to one record in another table. Dean of the college is related with the college in 1:1 relationship each college has one Dean and each Dean is managed one college.

1.5.1. The Hierarchical Database Model

FIGURE 1.6 A hierarchical structure



Basic Structure

The user perceives the hierarchical database as a hierarchy of segments. A segment is the equivalent of a file system's record type. In other words, the hierarchical database is a collection of records that is logically organized to conform to the upside-down tree (hierarchical) structure shown in Figure 1.8.

Within the hierarchy, the top layer (the root) is perceived as the parent of the segment directly beneath it. For example, in Figure 1.8, the root segment is the parent of the level 1 segment, which, in turn, is the parent of the level 2 segments, and so on. In turn, the segments below other segments are the children of the segment above them. In short,

- Each parent can have many children.
- Each child has only one parent.

Given this hierarchical structure, it is easy to trace both the database's components and the 1 :M relationships among them.

Keep in mind that the tree structure shown in Figure 1.8 cannot be duplicated on the computer's storage media. The computer does not the logical tree structure as human beings do. Instead, the tree is defined by a path that traces the parent segments to the child segments, beginning from the left.

This ordered sequencing of segments tracing the hierarchical structure is called the hierarchical path.

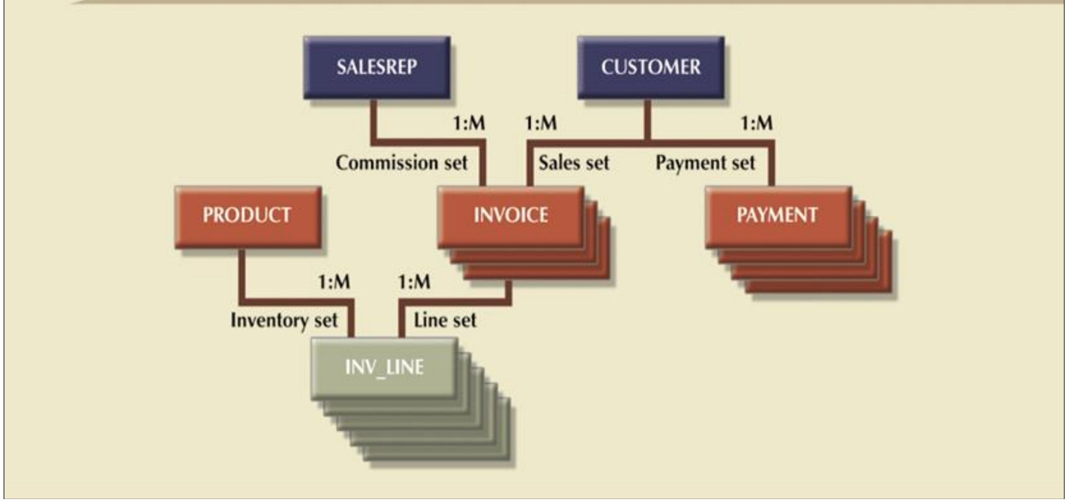
For example, the hierarchical path to the segment labeled "Part D" in Figure 1.8 can be traced this way: Final assembly —► Component A —* Assembly A—► Part A —> Part B —> Component B —> Component C —> Assembly B —> Part C —> Part D

Note:

1. Organization of records is as a collection of trees, rather than arbitrary graphs.
2. Each segment is represented record.

1.5.2. The Network Database Model

FIGURE 1.7 A network data model



The network database model was created to represent complex data relationships more effectively than the hierarchical model could, to improve database performance, and to impose database standard specifications for three crucial database components:

1. The network **schema,** the conceptual organization of the entire database as viewed by the database administrator. The schema includes a definition of the database name, the record type for each record, and the components that make up those records. "<u>The overall design of the database</u>”.

The **subschema,** which defines the portion of the database "seen" by the application programs that actually produce the desired information from the data contained within the database. The existence of the network database model was created to represent complex data relationships more effectively than the hierarchical model could, to improve database performance, and to impose database standard specifications for three crucial database components:

2. The network **schema,** the conceptual organization of the entire database as the database administrator. The schema includes a definition of the database name, the record type for each record, and the components that make up those records. "The overall design of the database-

Basic Structure

As in the hierarchical model, the user perceives the network database as a collection of records in 1:M relationships. However, quite unlike the hierarchical model, the network model allows a record to have more than one parent. Therefore, the commonly encountered relationships depicted in Figure 1.9 can be handled easily by the network database model.

Using network database terminology, a relationship is called a **set.** Each set is composed of at least two record types: an owner record that is equivalent to the hierarchical model's parent, and a **member** record that is equivalent to the hierarchical model's child. The difference between the hierarchical model and the network model is that the latter might include a condition in which a record can appear (as a member) in more than one set. In other words, a member may have several owners. *A **set represents** a 1:M relationship between the owner and the member.* An example of such a relationship is depicted in Figure 1.10.

The network database model in Figure 1.10 illustrates transactions that are based on a series of one-to- many relationships:

1. A SALESREP might have written many INVOICE tickets, but each INVOICE ticket was written by a single SALESREP. Therefore, there is a 1:M relationship between SALESREP and INVOICE.
2. A CUSTOMER might have made purchases on different occasions. Each time a shopping trip is made, a new INVOICE ticket is written; that is, a CUSTOMER may have generated many INVOICE tickets over time, but each INVOICE belongs to only a single CUSTOMER. Therefore, there is a 1:M relationship between CUSTOMER and INVOICE.

3. Each product purchased is listed on one of the invoice's lines. Because a customer may buy more than one product at a time—for example, you can buy a screwdriver, a box of screws, and a can of paint during your visit to a store each invoice can have one *or more* lines on it. But each INVLINE occurring on a *particular* INVOICE "belongs" to that invoice. Therefore, there is a 1:M relationship between INVOICE and INV LINE.
Each invoice line references a single product. But, because a store can sell many screwdrivers during any period of time, a product can appear on many different invoice lines. Therefore, there is a 1:M relationship between PRODUCT and INV_LINE.

A customer may make many payments over a period of time. But only one customer makes each payment. Therefore, there is a 1:M relationship between CUSTOMER and PAYMENT.

Note

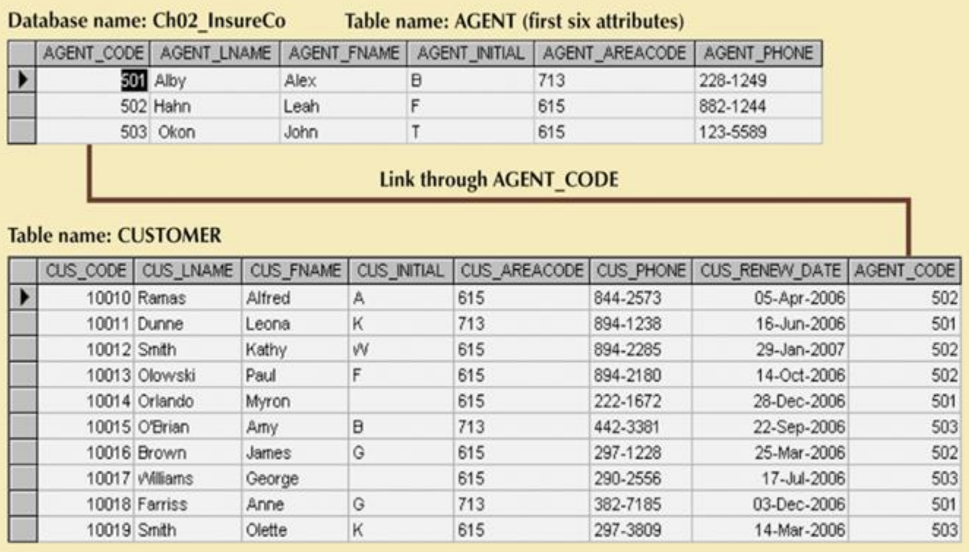
1. Data are repeated by collections of records.
2. Relationships among data are represented by links "pointer".
3. Organization is an arbitrary graph.

1.5.3 The Relational Database Model Basic Structure

-The relational database model is implemented through a very sophisticated relational database management system (RDBMS). The RDBMS performs the same basic functions provided by the hierarchical and network DB MS systems plus a host of other functions that make the relational database model easier to understand and to implement.

-The data and relationships are represented by a collection of tables.
-The relational model does not use pointers or links, but relates records by the values they contain (P.K., F.K)
-Each table is a matrix consisting of a series of row/column intersections. Tables, also called relations, are related to each other by sharing a common entity characteristic. For example, the CUSTOMER table in Figure 1.11 might contain a sales agent's number that is also contained in the AGENT table.

FIGURE 1.8 Linking relational tables



Database name: Ch02_InsureCo Table name: AGENT (first six attributes)

AGENT_CODE	AGENT_LNAME	AGENT_FNAME	AGENT_INITIAL	AGENT_AREACODE	AGENT_PHONE
501	Alby	Alex	B	713	228-1249
502	Hahn	Leah	F	615	882-1244
503	Okon	John	T	615	123-5589

Link through AGENT_CODE

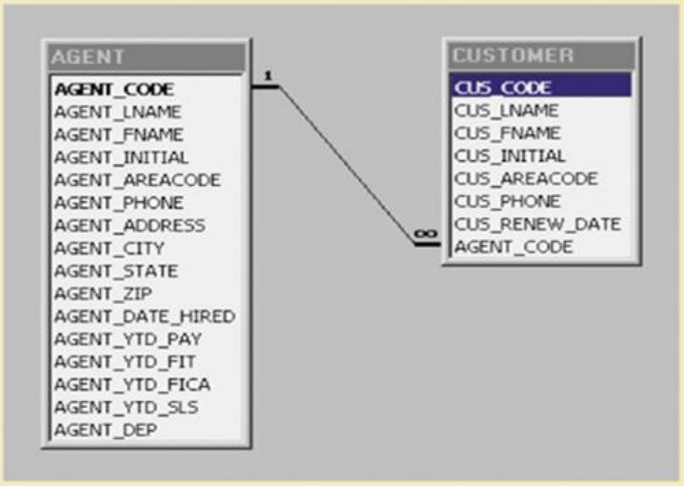
Table name: CUSTOMER

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_INITIAL	CUS_AREACODE	CUS_PHONE	CUS_RENEW_DATE	AGENT_CODE
10010	Ramas	Alfred	A	615	844-2573	05-Apr-2006	502
10011	Dunne	Leona	K	713	894-1238	16-Jun-2006	501
10012	Smith	Kathy	W	615	894-2285	29-Jan-2007	502
10013	Olowski	Paul	F	615	894-2180	14-Oct-2006	502
10014	Orlando	Myron		615	222-1672	28-Dec-2006	501
10015	O'Brian	Amy	B	713	442-3381	22-Sep-2006	503
10016	Brown	James	G	615	297-1228	25-Mar-2006	502
10017	Williams	George		615	290-2556	17-Jul-2006	503
10018	Farriss	Anne	G	713	382-7185	03-Dec-2006	501
10019	Smith	Olette	K	615	297-3809	14-Mar-2006	503

-Although the tables are completely independent of one another, we can e asily connect the data between tables. The relational model thus provides a minimum level of controlled redundancy to eliminate most of the redundancies commonly found in file systems.

-The relationship type (1:1, 1:M, or M:N) is often shown in a **relational schema,** an example of which is depicted in Figure 1.12. The relational schema shows the connecting fields (in this case, AGENTCODE) and the relationship type. Microsoft Access, the database software application used to generate Figure 1.12, employs the ∞ symbol to indicate the "many" side. In this example, the CUSTOMER represents the "many" side, because an AGENT can have many CUSTOMERS. The AGENT represents the "1" side, because each CUSTOMER has only one AGENT.

FIGURE 1.9 A relational diagram


AGENT
AGENT_CODE
AGENT_LNAME
AGENT_FNAME
AGENT_INITIAL
AGENT_AREACODE
AGENT_PHONE
AGENT_ADDRESS
AGENT_CITY
AGENT_STATE
AGENT_ZIP
AGENT_DATE_HIRED
AGENT_YTD_PAY
AGENT_YTD_FIT
AGENT_YTD_FICA
AGENT_YTD_SLS
AGENT_DEP
1
∞
CUSTOMER
CUS_CODE
CUS_LNAME
CUS_FNAME
CUS_INITIAL
CUS_AREACODE
CUS_PHONE
CUS_RENEW_DATE
AGENT_CODE