

WHY DATABASES?

Imagine trying to operate a business without knowing who your customers are, what products you are selling, who is working for you, who owes you money, and whom you owe money. All businesses have to keep this type of data and much more; and just as importantly, they must have those data available to decision makers when they need them. It can be argued that the ultimate purpose of all business information systems is to help businesses use information as an organizational resource. At the heart of all of these systems are the collection, storage, aggregation, manipulation, dissemination, and management of data.

Consider the case of Internet search staple Google. While Google is reluctant to disclose many details about its data storage specifications, it is estimated that the company responds to over 91 million searches per day across a collection of data that is several terabytes in size. Impressively, the results of these searches are available nearly instantly.

How can these businesses process this much data? How can they store it all, and then quickly retrieve just the facts that decision makers want to know, just when they want to know it? The answer is that they use databases.

DATA VS. INFORMATION

Data are raw facts. The word raw indicates that the facts have not yet been processed to reveal their meaning. For example, suppose that you want to know what the users of a computer lab think of its services. Typically, you would begin by surveying users to assess the computer lab's performance.

Information is the result of processing raw data to reveal its meaning. Data processing can be as simple as organizing data to reveal patterns or as complex as making forecasts or drawing inferences using statistical modeling.

In this "information age," production of accurate, relevant, and timely information is the key to good decision making. In turn, good decision making is the key to business survival in a global market. We are now said to be entering the "knowledge age." Data are the foundation of information, which is the bedrock of **knowledge** - that is, the body of information and facts about a specific subject. Knowledge implies familiarity, awareness, and understanding of information as it applies to an environment. A key characteristic of knowledge is that "new" knowledge can be derived from "old" knowledge.

Here are some key points:

- Data constitute the building blocks of information.
- Information is produced by processing data.
- Information is used to reveal the meaning of data.
- Accurate, relevant, and timely information is the key to good decision making.
- Good decision making is the key to organizational survival in a global environment.

INTRODUCING THE DATABASE

A *database* is a shared, integrated computer structure that stores a collection of:

- End-user data, that is, raw facts of interest to the end user.
- Metadata, or data about data, through which the end-user data are integrated and managed.

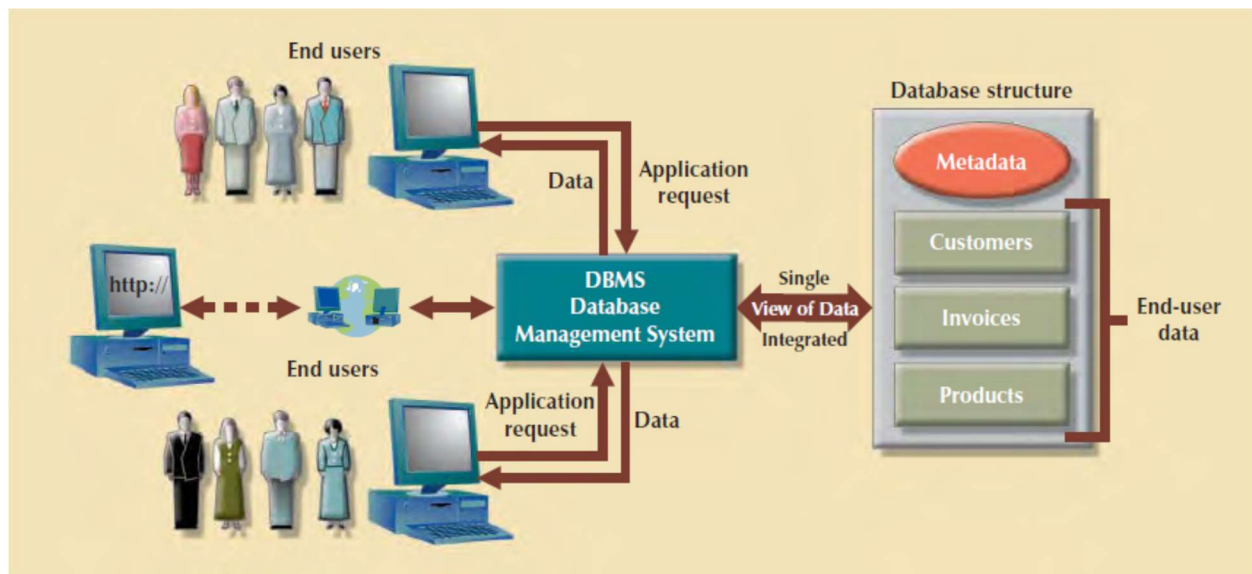
The metadata (data about data) provide a description of the data characteristics and the set of relationships that links the data found within the database. For example, the metadata component stores information such as the name of each data element, the type of values (numeric, dates, or text) stored on each data element, whether or not the data element can be left empty, and so on.

A *database management system* (DBMS) is a collection of programs that manages the database structure and controls access to the data stored in the database. In a sense, a database resembles a very well-

organized electronic filing cabinet in which powerful software, known as a database management system, helps manage the cabinet's contents.

Role and Advantages of the DBMS

The DBMS serves as the intermediary between the user and the database. The database structure itself is stored as a collection of files, and the only way to access the data in those files is through the DBMS. The figure below emphasizes the point that the DBMS presents the end user (or application program) with a single, integrated view of the data in the database.



A DBMS provides advantages such as:

- *Improved data sharing.* The DBMS helps create an environment in which end users have better access to more and better-managed data. Such access makes it possible for end users to respond quickly to changes in their environment.
- *Improved data security.* The more users access the data, the greater the risks of data security breaches. Corporations invest considerable amounts of time, effort, and money to ensure that corporate data are used properly. A DBMS provides a framework for better enforcement of data privacy and security policies.
- *Better data integration.* Wider access to well-managed data promotes an integrated view of the organization's operations and a clearer view of the big picture. It becomes much easier to see how actions in one segment of the company affect other segments.
- *Minimized data inconsistency.* Data inconsistency exists when different versions of the same data appear in different places. For example, data inconsistency exists when a company's sales department stores a sales representative's name as "Bill Brown" and the company's personnel department stores that same person's name as "William G. Brown," or when the company's regional sales office shows the price of a product as \$45.95 and its national sales office shows the same product's price as \$43.95. The probability of data inconsistency is greatly reduced in a properly designed database.
- *Improved data access.* The DBMS makes it possible to produce quick answers to ad hoc queries. From a database perspective, a query is a specific request issued to the DBMS for data manipulation—for example, to read or update the data. Simply put, a query is a question, and an ad hoc query is a spur-of-the-moment question. The DBMS sends back an answer (called the query result set) to the application.

- *Improved decision making.* Better-managed data and improved data access make it possible to generate better-quality information, on which better decisions are based. The quality of the information generated depends on the quality of the underlying data. **Data quality** is a comprehensive approach to promoting the accuracy, validity, and timeliness of the data. While the DBMS does not guarantee data quality, it provides a framework to facilitate data quality initiatives.
- *Increased end-user productivity.* The availability of data, combined with the tools that transform data into usable information, empowers end users to make quick, informed decisions that can make the difference between success and failure in the global economy.

Types of Databases

Databases can be classified according to the number of users, the database location(s), and the expected type and extent of use.

The number of users determines whether the database is classified as single-user or multiuser. A **single-user database** supports only one user at a time.

A single-user database that runs on a personal computer is called a **desktop database**. In contrast, a **multiuser database** supports multiple users at the same time. When the multiuser database supports a relatively small number of users (usually fewer than 50) or a specific department within an organization, it is called a **workgroup database**. When the database is used by the entire organization and supports many users (more than 50, usually hundreds) across many departments, the database is known as an **enterprise database**.

Location might also be used to classify the database. For example, a database that supports data located at a single site is called a **centralized database**. A database that supports data distributed across several different sites is called a **distributed database**.

A database that is designed primarily to support a company's day-to-day operations is classified as an **operational database** (sometimes referred to as a **transactional** or **production database**). In contrast, a **data warehouse** focuses primarily on storing data used to generate information required to make tactical or strategic decisions. Such decisions typically require extensive "data massaging" (data manipulation) to extract information to formulate pricing decisions, sales forecasts, market positioning, and so on. Most decision support data are based on data obtained from operational databases over time and stored in data warehouses.

Databases can also be classified to reflect the degree to which the data are structured. **Unstructured data** are data that exist in their original (raw) state, that is, in the format in which they were collected.

Therefore, unstructured data exist in a format that does not lend itself to the processing that yields information. **Structured data** are the result of taking unstructured data and formatting (structuring) such data to facilitate storage, use, and the generation of information. You apply structure (format) based on the type of processing that you intend to perform on the data.

To illustrate the structure concept, imagine a stack of printed paper invoices. If you want to merely store these invoices as images for future retrieval and display, you can scan them and save them in a graphic format. On the other hand, if you want to derive information such as monthly totals and average sales, such graphic storage would not be useful. Instead, you could store the invoice data in a (structured) spreadsheet format so that you can perform the requisite computations. Actually, most data you encounter are best classified as semistructured. **Semistructured data** are data that have already been processed to some extent. For example, if you look at a typical Web page, the data are presented to you in a prearranged format to convey some information.

However, corporations are not limited to the use of structured data. They also use semistructured and unstructured data. Just think of the very valuable information that can be found on company e-mails,

memos, documents such as procedures and rules, Web pages, and so on. Unstructured and semistructured data storage and management needs are being addressed through a new generation of databases known as XML databases. **Extensible Markup Language (XML)** is a special language used to represent and manipulate data elements in a textual format. An **XML database** supports the storage and management of semistructured XML data.

WHY DATABASE DESIGN IS IMPORTANT

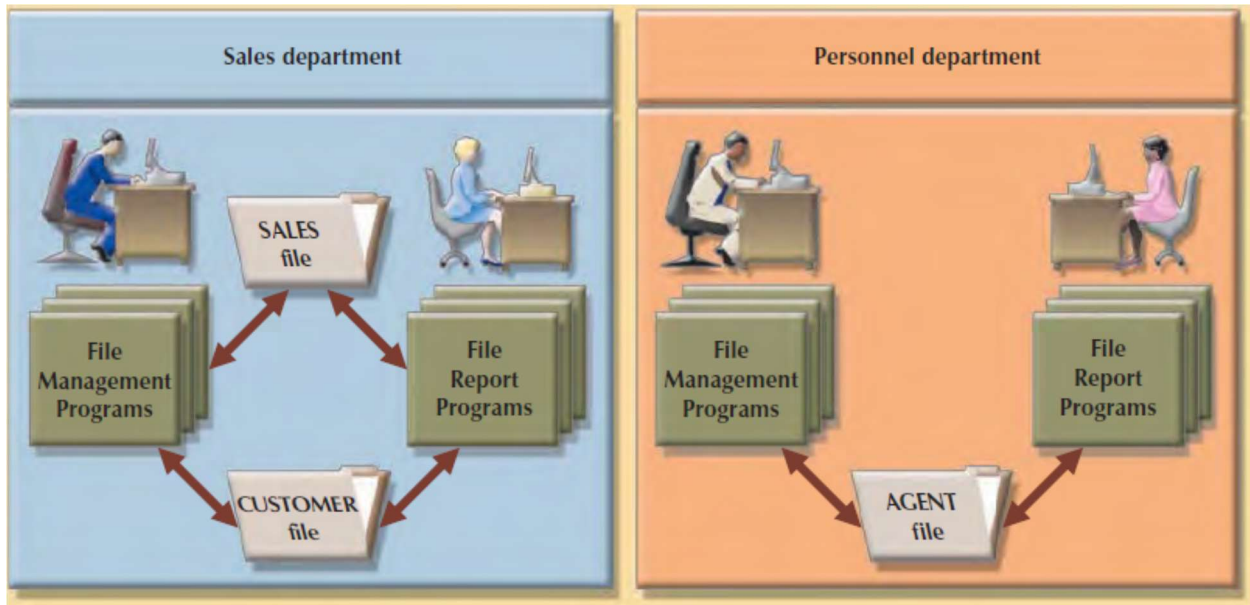
Database design refers to the activities that focus on the design of the database structure that will be used to store and manage end-user data. Database design is such a crucial aspect of working with databases since good DBMS will perform poorly with a badly designed database.

EVOLUTION OF FILE SYSTEM DATA PROCESSING

(Manual) Historically, such systems were often manual, paper-and-pencil systems. The papers within these systems were organized in order to facilitate the expected use of the data. Typically, this was accomplished through a system of file folders and filing cabinets. As long as a data collection was relatively small and an organization's business users had few reporting requirements, the manual system served its role well as a data repository.

(Computerized) Generating reports from manual file systems was 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. Therefore, a **data processing (DP) specialist** was hired to create a computer-based system that would track data and produce required reports.

TERM	DEFINITION
Data	"Raw" facts, such as a telephone number, a birth date, a customer name, and a year-to-date (YTD) sales value. Data have little meaning unless they have been organized in some logical manner.
Field	A character or group of characters (alphabetic or numeric) that has a specific meaning. A field is used to define and store data.
Record	A logically connected set of one or more fields that describes a person, place, or thing. For example, the fields that constitute a record for a customer might consist of the customer's name, address, phone number, date of birth, credit limit, and unpaid balance.
File	A collection of related records. For example, a file might contain data about the students currently enrolled at Gigantic University.



A simple file system

The advent of computer files to store company data was significant; it not only established a landmark in the use of computer technologies but also represented a huge step forward in a business's ability to process data. Previously, users had direct, hands-on access to all of the business data. But they didn't have the tools to convert those data into the information that they needed. The creation of computerized file systems gave them improved tools for manipulating the company data that allowed them to create new information.

However, such developments also created a split between the ways DP specialists and end users viewed the data.

- From the DP specialist's perspective, the computer files within the file system were created to be similar to the manual files. Data management programs were created to add to, update, and delete data from the file.
- From the end user's perspective, the systems separated the users from the data. As the users' competitive environment pushed them to make more and more decisions in less and less time, the delay from when the users conceived of a new way to create information from the data to when the DP specialist could create the programs to generate that information was a source of great frustration.

PROBLEMS WITH FILE SYSTEM DATA PROCESSING

A critique of the file system method serves two major purposes:

- Understanding the shortcomings of the file system enables you to understand the development of modern databases.
- Many of the problems are not unique to file systems. Failure to understand such problems is likely to lead to their duplication in a database environment, even though database technology makes it easy to avoid them.

The following problems associated with file systems, whether created by DP specialists or through a series of spreadsheets, severely challenge the types of information that can be created from the data as well as the accuracy of the information:

- *Lengthy development times.* The first and most glaring problem with the file system approach is that even the simplest data-retrieval task requires extensive programming. With the older file systems, programmers had to specify what must be done and how it was to be done.
- *Difficulty of getting quick answers.* The need to write programs to produce even the simplest reports makes ad hoc queries impossible. Harried DP specialists who work with mature file systems often receive numerous requests for new reports. They are often forced to say that the report will be ready “next week” or even “next month.” If you need the information now, getting it next week or next month will not serve your information needs.
- *Complex system administration.* System administration becomes more difficult as the number of files in the system expands. Even a simple file system with a few files requires creating and maintaining several file management programs (each file must have its own file management programs that allow the user to add, modify, and delete records; to list the file contents; and to generate reports). Because ad hoc queries are not possible, the file reporting programs can multiply quickly. The problem is compounded by the fact that each department in the organization “owns” its data by creating its own files.
- *Lack of security and limited data sharing.* Another fault of a file system data repository is a lack of security and limited data sharing. Data sharing and security are closely related. Sharing data among multiple geographically dispersed users introduces a lot of security risks. In terms of spreadsheet data, while many spreadsheet programs provide rudimentary security options, they are not always used, and even when they are used, they are insufficient for robust data sharing among users.
- *Extensive programming.* Making changes to an existing file structure can be difficult in a file system environment.

For example, changing just one field in the original CUSTOMER file would require a program that:

1. Reads a record from the original file.
2. Transforms the original data to conform to the new structure’s storage requirements.
3. Writes the transformed data into the new file structure.
4. Repeats steps 2 to 4 for each record in the original file.

Structural and Data Dependence

A file system exhibits **structural dependence**, which means that access to a file is dependent on its structure. Conversely, **structural independence** exists when it is possible to make changes in the file structure without affecting the application program’s ability to access the data.

Even changes in the characteristics of data, such as changing a field from integer to decimal, require changes in all the programs that access the file. Because all data access programs are subject to change when any of the file’s data storage characteristics change (that is, changing the data type), the file system is said to exhibit **data dependence**.

Conversely, **data independence** exists when it is possible to make changes in the data storage characteristics without affecting the application program’s ability to access the data.

Data Redundancy

The file system’s structure makes it difficult to combine data from multiple sources, and its lack of security renders the file system vulnerable to security breaches. The organizational structure promotes the storage of the same basic data in different locations. (Database professionals use the term **islands of information** for such scattered data locations.)

Data redundancy exists when the same data are stored unnecessarily at different places.

Uncontrolled data redundancy sets the stage for:

- *Poor data security.* Having multiple copies of data increases the chances for a copy of the data to be susceptible to unauthorized access.

- *Data inconsistency.* Data inconsistency exists when different and conflicting versions of the same data appear in different places. For example, suppose you change an agent's phone number or address in the AGENT file. If you forget to make corresponding changes in the CUSTOMER file, the files contain different data for the same agent. Reports will yield inconsistent results that depend on which version of the data is used.

Data that display data inconsistency are also referred to as data that lack data integrity. **Data integrity** is defined as the condition in which all of the data in the database are consistent with the real-world events and conditions.

In other words, data integrity means that:

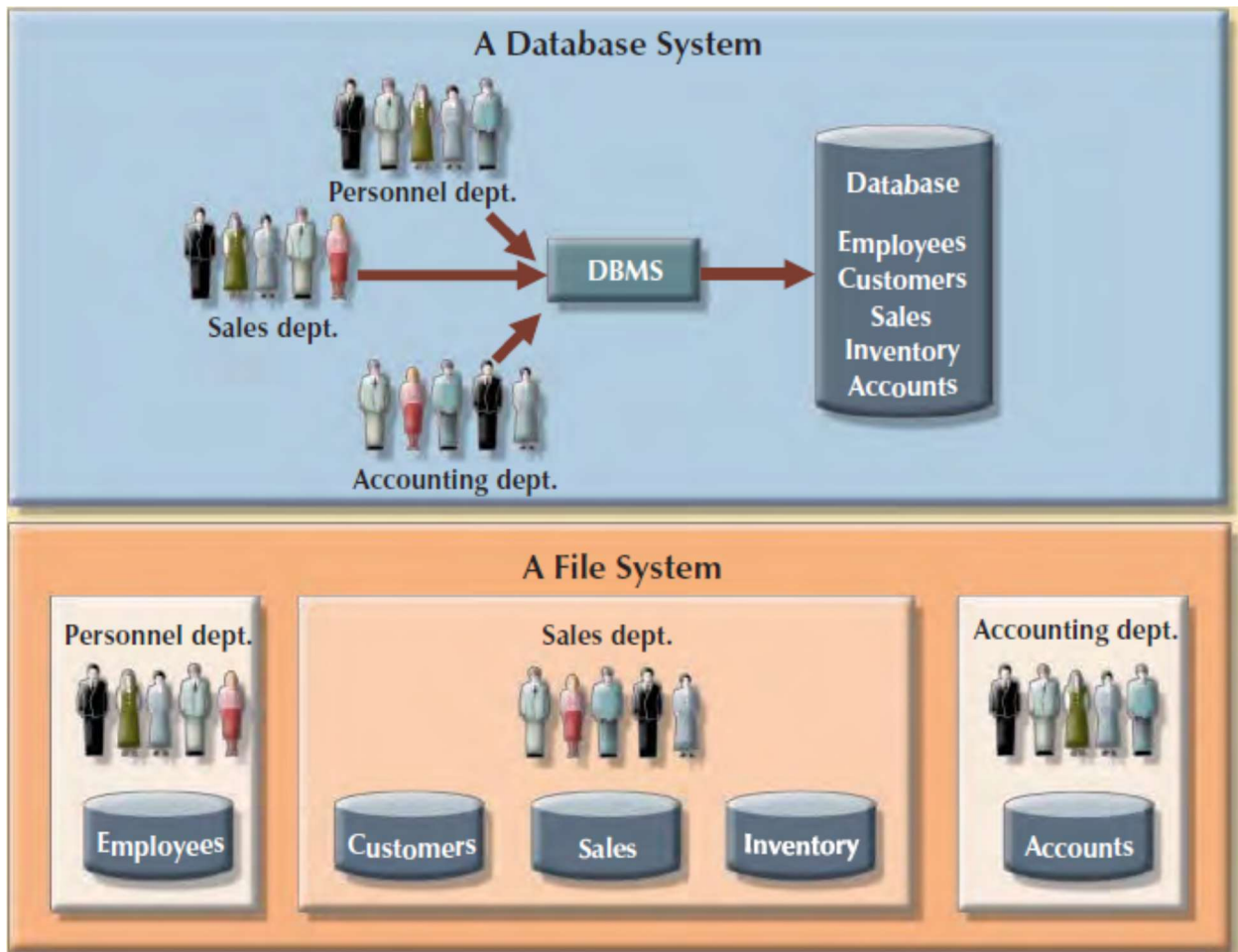
- Data are *accurate*—there are no data inconsistencies.
- Data are *verifiable*—the data will always yield consistent results.

Data anomalies. The dictionary defines *anomaly* as “an abnormality.” Ideally, a field value change should be made in only a single place. Data redundancy, however, fosters an abnormal condition by forcing field value changes in many different locations. A **data anomaly** develops when not all of the required changes in the redundant data are made successfully.

- *Update anomalies.* If agent Leah F. Hahn has a new phone number, that 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 a change might occur in hundreds or even thousands of records. Clearly, the potential for data inconsistencies is great.
- *Insertion anomalies.* If only the CUSTOMER file existed, to add a new agent, you would also add a dummy customer data entry to reflect the new agent's addition. Again, the potential for creating data inconsistencies would be great.
- *Deletion anomalies.* If you delete the customers Amy B. O'Brian, George Williams, and Olette K. Smith, you will also delete John T. Okon's agent data. Clearly, this is not desirable.

DATABASE SYSTEMS

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 system consists of logically related data stored in a single logical data repository. (The “logical” label reflects the fact that, although the data repository appears to be a single unit to the end user, its contents may actually be physically distributed among multiple data storage facilities and/or locations.) Because the database's data repository is a single logical unit, the database represents a major change in the way end-user data are stored, accessed, and managed.



The Database System Environment

The term **database system** refers to an organization of components that define and regulate the collection, storage, management, and use of data within a database environment. From a general management point of view, the database system is composed of the five major parts shown in the Figure below: hardware, software, people, procedures, and data.

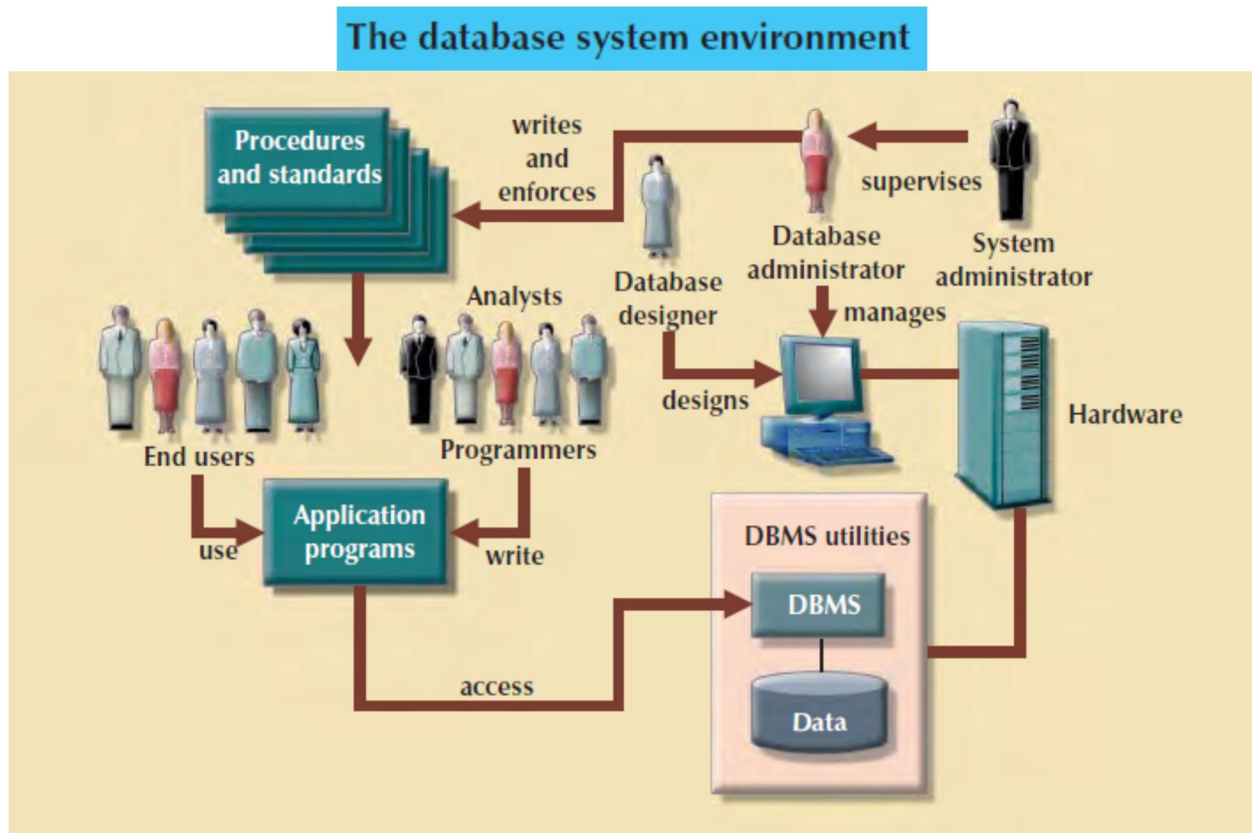
Hardware. Hardware refers to all of the system's physical devices; for example, computers (PCs, workstations, servers, and supercomputers), storage devices, printers, network devices (hubs, switches, routers, fiber optics), and other devices (automated teller machines, ID readers, and so on).

Software. Although the most readily identified software is the DBMS itself, to make the database system function fully, three types of software are needed: operating system software, DBMS software, and application programs and utilities.

People. This component includes all users of the database system. On the basis of primary job functions, five types of users can be identified in a database system: system administrators, database administrators, database designers, system analysts and programmers, and end users.

Procedures. Procedures are the instructions and rules that govern the design and use of the database system. Procedures are a critical, although occasionally forgotten, component of the system. Procedures play an important role in a company because they enforce the standards by which business is conducted within the organization and with customers. Procedures are also used to ensure that there is an organized way to monitor and audit both the data that enter the database and the information that is generated through the use of those data.

Data. The word *data* covers the collection of facts stored in the database. Because data are the raw material from which information is generated, the determination of what data are to be entered into the database and how those data are to be organized is a vital part of the database designer's job.



DBMS Functions

A DBMS performs several important functions that guarantee the integrity and consistency of the data in the database. Most of those functions are transparent to end users, and most can be achieved only through the use of a DBMS.

- *Data dictionary management.* The DBMS stores definitions of the data elements and their relationships (metadata) in a **data dictionary**. In turn, all programs that access the data in the database work through the DBMS. The DBMS uses the data dictionary to look up the required data component structures and relationships, thus relieving you from having to code such complex relationships in each program. Additionally, any changes made in a database structure are automatically recorded in the data dictionary, thereby freeing you from having to modify all of the programs that access the changed structure. In other words, the DBMS provides data abstraction, and it removes structural and data dependence from the system.
- *Data storage management.* The DBMS creates and manages the complex structures required for data storage, thus relieving you from the difficult task of defining and programming the physical data characteristics.
- *Data transformation and presentation.* The DBMS transforms entered data to conform to required data structures. The DBMS relieves you of the chore of making a distinction between the logical data format and the physical data format. That is, the DBMS formats the physically retrieved data to make it conform to the user's logical expectations. For example, imagine an enterprise database used by a multinational company. An end user in England would expect to enter data such as July 11, 2010, as "11/07/2010." In contrast, the same date would be entered in

the United States as “07/11/2010.” Regardless of the data presentation format, the DBMS must manage the date in the proper format for each country.

- *Security management.* The DBMS creates a security system that enforces user security and data privacy. Security rules determine which users can access the database, which data items each user can access, and which data operations (read, add, delete, or modify) the user can perform. This is especially important in multiuser database systems.
- *Multiuser access control.* To provide data integrity and data consistency, the DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently without compromising the integrity of the database.
- *Backup and recovery management.* The DBMS provides backup and data recovery to ensure data safety and integrity. Current DBMS systems provide special utilities that allow the DBA to perform routine and special backup and restore procedures. Recovery management deals with the recovery of the database after a failure, such as a bad sector in the disk or a power failure. Such capability is critical to preserving the database’s integrity.
- *Data integrity management.* The DBMS promotes and enforces integrity rules, thus minimizing data redundancy and maximizing data consistency. The data relationships stored in the data dictionary are used to enforce data integrity.
- *Database access languages and application programming interfaces.* The DBMS provides data access through a query language. A **query language** is a nonprocedural language—one that lets the user specify what must be done without having to specify how it is to be done. **Structured Query Language (SQL)** is the de facto query language and data access standard supported by the majority of DBMS vendors.
- *Database communication interfaces.* Current-generation DBMSs accept end-user requests via multiple, different network environments. For example, the DBMS might provide access to the database via the Internet through the use of Web browsers such as Mozilla Firefox or Microsoft Internet Explorer.

Disadvantages of Database Systems

Although the database system yields considerable advantages over previous data management approaches, database systems do carry significant disadvantages. For example:

- *Increased costs.* Database systems require sophisticated hardware and software and highly skilled personnel. The cost of maintaining the hardware, software, and personnel required to operate and manage a database system can be substantial. Training, licensing, and regulation compliance costs are often overlooked when database systems are implemented.
- *Management complexity.* Database systems interface with many different technologies and have a significant impact on a company’s resources and culture. The changes introduced by the adoption of a database system must be properly managed to ensure that they help advance the company’s objectives. Given the fact that database systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly.
- *Maintaining currency.* To maximize the efficiency of the database system, you must keep your system current. Therefore, you must perform frequent updates and apply the latest patches and security measures to all components. Because database technology advances rapidly, personnel training costs tend to be significant.
- *Vendor dependence.* Given the heavy investment in technology and personnel training, companies might be reluctant to change database vendors. As a consequence, vendors are less likely to offer pricing point advantages to existing customers, and those customers might be limited in their choice of database system components.
- *Frequent upgrade/replacement cycles.* DBMS vendors frequently upgrade their products by adding new functionality. Such new features often come bundled in new upgrade versions of the software. Some of these versions require hardware upgrades. Not only do the upgrades

themselves cost money, but it also costs money to train database users and administrators to properly use and manage the new features.

DATA MODELS

A **data model** is a relatively simple representation, usually graphical, of more complex real-world data structures. Within the database environment, a data model represents data structures and their characteristics, relations, constraints, transformations, and other constructs with the purpose of supporting a specific problem domain.

Data modeling is an iterative, progressive process. You start with a simple understanding of the problem domain, and as your understanding of the problem domain increases, so does the level of detail of the data model. Done properly, the final data model is in effect a “blueprint” containing all the instructions to build a database that will meet all end-user requirements. This blueprint is narrative and graphical in nature, meaning that it contains both text descriptions in plain, unambiguous language and clear, useful diagrams depicting the main data elements.

THE IMPORTANCE OF DATA MODELS

Data models can facilitate interaction among the designer, the applications programmer, and the end user. A well-developed data model can even foster improved understanding of the organization for which the database design is developed. In short, data models are a communication tool.

The importance of data modeling cannot be overstated. Data constitute the most basic information units employed by a system. Applications are created to manage data and to help transform data into information. But data are viewed in different ways by different people. For example, contrast the (data) view of a company manager with that of a company clerk. Although the manager and the clerk both work for the same company, the manager is more likely to have an enterprise-wide view of company data than the clerk.

The different users and producers of data and information often reflect the “blind people and the elephant” analogy: the blind person who felt the elephant’s trunk had quite a different view of the elephant from the one who felt the elephant’s leg or tail. What is needed is a view of the whole elephant. Similarly, a house is not a random collection of rooms; if someone is going to build a house, he or she should first have the overall view that is provided by blueprints. Likewise, a sound data environment requires an overall database blueprint based on an appropriate data model.

When a good database blueprint is available, it does not matter that an applications programmer’s view of the data is different from that of the manager and/or the end user. Conversely, when a good database blueprint is not available, problems are likely to ensue. For instance, an inventory management program and an order entry system may use conflicting product-numbering schemes, thereby costing the company thousands (or even millions) of dollars.

DATA MODEL BASIC BUILDING BLOCKS

The basic building blocks of all data models are entities, attributes, relationships, and constraints. An **entity** is anything (a person, a place, a thing, or an event) about which data are to be collected and stored. An entity represents a particular type of object in the real world. Because an entity represents a particular type of object, entities are “distinguishable”—that is, each entity occurrence is unique and distinct. For example, a CUSTOMER entity would have many distinguishable customer occurrences, such as John Smith, Pedro Dinamita, Tom Strickland, etc. Entities may be physical objects, such as customers or products, but entities may also be abstractions, such as flight routes or musical concerts.

An **attribute** is a characteristic of an entity. For example, a CUSTOMER entity would be described by attributes such as customer last name, customer first name, customer phone, customer address, and customer credit limit. Attributes are the equivalent of fields in file systems.

A **relationship** describes an association among entities. For example, a relationship exists between customers and agents that can be described as follows: an agent can serve many customers, and each customer may be served by one agent. Data models use three types of relationships: one-to-many, many-to-many, and one-to-one. Database designers usually use the shorthand notations 1:M or 1..*, M:N or *.* , and 1:1 or 1..1, respectively. (Although the M:N notation is a standard label for the many-to-many relationship, the label M:M may also be used.)

- **One-to-many (1:M or 1..*) relationship.** A painter paints many different paintings, but each one of them is painted by only one 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. (Note that entity names are often capitalized as a convention, so they are easily identified.) Similarly, a customer (the “one”) may generate many invoices, but each invoice (the “many”) is generated by only a single customer. The “CUSTOMER generates INVOICE” relationship would also be labeled 1:M.
- **Many-to-many (M:N or *.*) relationship.** An employee may learn many job skills, and each job skill may be learned by many employees. Database designers label the relationship “EMPLOYEE learns SKILL” as M:N. Similarly, a student can take many classes and each class can be taken by many students, thus yielding the M:N relationship label for the relationship expressed by “STUDENT takes CLASS.”
- **One-to-one (1:1 or 1..1) relationship.** A retail company’s management structure may require that each of its stores be managed by a single employee. In turn, each store manager, who is an employee, manages only a single store. Therefore, the relationship “EMPLOYEE manages STORE” is labeled 1:1.

The preceding discussion identified each relationship in both directions; that is, relationships are bidirectional:

- *One* CUSTOMER can generate *many* INVOICES.
- Each of the *many* INVOICES is generated by only *one* CUSTOMER.

A **constraint** is a restriction placed on the data. Constraints are important because they help to ensure data integrity. Constraints are normally expressed in the form of rules. For example:

- An employee’s salary must have values that are between 6,000 and 350,000.
- A student’s GPA must be between 0.00 and 4.00.
- Each class must have one and only one teacher.

How do you properly identify entities, attributes, relationships, and constraints? The first step is to clearly identify the business rules for the problem domain you are modeling.

BUSINESS RULES

From a database point of view, the collection of data becomes meaningful only when it reflects properly defined *business rules*. A **business rule** is a brief, precise, and unambiguous description of a policy, procedure, or principle within a specific organization. In a sense, business rules are misnamed: they apply to *any* organization, large or small—a business, a government unit, a religious group, or a research laboratory—that stores and uses data to generate information.

Business rules, derived from a detailed description of an organization’s operations, help to create and enforce actions within that organization’s environment. Business rules must be rendered in writing and updated to reflect any change in the organization’s operational environment.

Properly written business rules are used to define entities, attributes, relationships, and constraints. Any time you see relationship statements such as “an agent can serve many customers, and each customer can be served by only one agent,” you are seeing business rules at work.

Business rules describe, in simple language, the main and distinguishing characteristics of the data *as viewed by the company*. Examples of business rules are as follows:

- A customer may generate many invoices.
- An invoice is generated by only one customer.
- A training session cannot be scheduled for fewer than 10 employees or for more than 30 employees.

Note that those business rules establish entities, relationships, and constraints. For example, the first two business rules establish two entities (CUSTOMER and INVOICE) and a 1:M relationship between those two entities. The third business rule establishes a constraint (no fewer than 10 people and no more than 30 people), two entities (EMPLOYEE and TRAINING), and a relationship between EMPLOYEE and TRAINING.

Discovering Business Rules

A faster and more direct source of business rules is direct interviews with end users. Unfortunately, because perceptions differ, end users are sometimes a less reliable source when it comes to specifying business rules. For example, a maintenance department mechanic might believe that any mechanic can initiate a maintenance procedure, when actually only mechanics with inspection authorization can perform such a task. Such a distinction might seem trivial, but it can have major legal consequences. The database designer's job is to reconcile any differences and verify the results of the reconciliation to ensure that the business rules are appropriate and accurate.

The process of identifying and documenting business rules is essential to database design for several reasons:

- They help to standardize the company's view of data.
- They can be a communications tool between users and designers.
- They allow the designer to understand the nature, role, and scope of the data.
- They allow the designer to understand business processes.
- They allow the designer to develop appropriate relationship participation rules and constraints and to create an accurate data model.

Not all business rules can be modeled. For example, a business rule that specifies that "no pilot can fly more than 10 hours within any 24-hour period" cannot be modeled. However, such a business rule can be enforced by application software.

As a general rule, a noun in a business rule will translate into an entity in the model, and a verb (active or passive) associating nouns will translate into a relationship among the entities. For example, the business rule "a customer may generate many invoices" contains two nouns (*customer* and *invoices*) and a verb (*generate*) that associates the nouns. From this business rule, you could deduce that:

- Customer and invoice are objects of interest for the environment and should be represented by their respective entities.
- There is a "generate" relationship between customer and invoice.

To properly identify the type of relationship, you should consider that relationships are bidirectional; that is, they go both ways. For example, the business rule "a customer may generate many invoices" is complemented by the business rule "an invoice is generated by only one customer." In that case, the relationship is one-to-many (1:M). Customer is the "1" side, and invoice is the "many" side.

As a general rule, to properly identify the relationship type, you should ask two questions:

- How many instances of B are related to one instance of A?
- How many instances of A are related to one instance of B?

For example, you can assess the relationship between student and class by asking two questions:

- In how many classes can one student enroll? Answer: many classes.
- How many students can enroll in one class? Answer: many students.

Therefore, the relationship between student and class is many-to-many (M:N).

NAMING CONVENTIONS

Entity names should be descriptive of the objects in the business environment, and use terminology that is familiar to the users. An attribute name should also be descriptive of the data represented by that attribute. It is also a good practice to prefix the name of an attribute with the name of the entity (or an abbreviation of the entity name) in which it occurs. For example, in the CUSTOMER entity, the customer's credit limit may be called CUS_CREDIT_LIMIT. The CUS indicates that the attribute is descriptive of the CUSTOMER entity, while CREDIT_LIMIT makes it easy to recognize the data that will be contained in the attribute.

THE EVOLUTION OF DATA MODELS

The quest for better data management has led to several models that attempt to resolve the file system's critical shortcomings. These models represent schools of thought as to what a database is, what it should do, the types of structures that it should employ, and the technology that would be used to implement these structures.

Hierarchical and Network Models

The **hierarchical model** was developed in the 1960s to manage large amounts of data for complex manufacturing projects such as the Apollo rocket that landed on the moon in 1969. Its basic logical structure is represented by an upside-down tree. The hierarchical structure contains levels, or segments. A **segment** is the equivalent of a file system's record type. Within the hierarchy, a higher layer is perceived as the parent of the segment directly beneath it, which is called the child. The hierarchical model depicts a set of one-to-many (1:M) relationships between a parent and its children segments. (Each parent can have many children, but each child has only one parent.)

The **network model** was created to represent complex data relationships more effectively than the hierarchical model, to improve database performance, and to impose a database standard. In the network model, the user perceives the network database as a collection of records in 1:M relationships. However, unlike the hierarchical model, the network model allows a record to have more than one parent. While the network database model is generally not used today, the definitions of standard database *concepts* that emerged with the network model are still used by modern data models. Some important concepts that were defined at this time are:

- The **schema**, which is the conceptual organization of the entire database as viewed by the database administrator.
- 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.
- A **data management language (DML)**, which defines the environment in which data can be managed and to work with the data in the database.
- A schema **data definition language (DDL)**, which enables the database administrator to define the schema components.

As information needs grew and as more sophisticated databases and applications were required, the network model became too cumbersome. The lack of ad hoc query capability put heavy pressure on programmers to generate the code required to produce even the simplest reports. And although the existing databases provided limited data independence, any structural change in the database could still produce havoc in all application programs that drew data from the database. Because of the disadvantages of the hierarchical and network models, they were largely replaced by the relational data model in the 1980s.