Database Systems

1. Database Overview

1.1 Basic Concepts

Database

"An organized collection of logically related data is called Database."

A database has the following implicit properties:

- ✓ A database represents some aspect of the real world, sometimes called the miniworld or the universe of discourse (UoD). Changes to the miniworld are reflected in the database.
- ✓ A database is a logically coherent collection of data with some inherent meaning. A random assortment of data cannot correctly be referred to as a database.
- ✓ A database is designed, built, and populated with data for a specific purpose. It has an intended group of users and some preconceived applications in which these users are interested.

"A shared collection of logically related data, and a description of this data, designed to meet the information needs of an organization. "

Data

The term data referred to facts concerning objects and events that could be recorded and stored on computer media.it is raw facts and figures. By data, we mean known facts that can be recorded and that have implicit meaning.

"Stored representations of objects and events that have meaning and importance in the user's environment is called data."

Information

The terms data and information are closely related and in fact are often used interchangeably. However, it is useful to distinguish between data and information. We define information as data that have been processed in such a way that the knowledge of the person who uses the data is increased.

"Data that have been processed in such a way as to increase the knowledge of the person who uses the data is called information."

Metadata

Metadata are data that describe the properties or characteristics of end-user data and the context of that data. Some of the properties that are typically described include data names, definitions, length (or size), and allowable values. Metadata describing data context include the source of the data, where the data are stored, ownership (or stewardship), and usage. Although it may seem circular, many people think of metadata as "data about data."

"Data that describe the properties or characteristics of end-user data and the context of those data."

File

A file is a collection of related records. Which contains logically related data. File is a collection of records or documents dealing with one organization, person, area or subject.

1.2 File Processing or File-Base Approach

"A collection of application programs that perform services for the end-users such as the production of reports. Each program defines and manages its own data."

When computer-based data processing was first available, there were no databases. To be useful for business applications, computers had to store, manipulate, and retrieve large files of data. Computer file processing systems were developed for this purpose. Although these systems have evolved over time, their basic structure and purpose have changed little over several decades.

"An application program (or set of related programs) that is used to perform a series of database activities (create, read, update, and delete) on behalf of database users is called Database Application."

Disadvantages or Limitations of File Processing System

I. Separation and Isolation of Data

When data is isolated in separate files, it is more difficult to access data that should be available. For example, if we want to produce a list of all houses that match the requirements of clients, we first need to create a temporary file of those clients who have 'house' as the preferred type. We then search the PropertyForRent file for those properties where the property type is 'house' and the rent is less than the client's maximum rent. With file systems, such processing is difficult. The application developer must synchronize the processing of two files to ensure the correct data is extracted. This difficulty is compounded if we require data from more than two files.

II. Duplication of data

Because applications are often developed independently in file processing systems, unplanned duplicate data files are the rule rather than the exception. This duplication is wasteful because it requires additional storage space and increased effort to keep all files up to date. Data formats may be inconsistent or data values may not agree (or both). Reliable metadata are very difficult to establish in file processing systems.

Uncontrolled duplication of data is undesirable for several reasons, including:

- ✓ Duplication is wasteful. It costs time and money to enter the data more than once.
- ✓ It takes up additional storage space, again with associated costs. Often, the duplication of data can be avoided by sharing data files.
- ✓ Perhaps more importantly, duplication can lead to loss of data integrity; in other words, the data is no longer consistent.

III. Data dependence

File descriptions are stored within each database application program that accesses a given file. Because the program contains a detailed file description for these files, any change to a file structure requires changes to the file descriptions for all programs that access the file.

IV. Incompatible file formats

Because the structure of files is embedded in the application programs, the structures are dependent on the application programming language. For example, the structure of a file generated by a COBOL program may be different from the structure of a file generated by a 'C' program. The direct incompatibility of such files makes them difficult to process jointly.

V. Fixed queries/proliferation of application programs

From the end-user's point of view, file-based systems proved to be a great improvement over manual systems. Consequently, the requirement for new or modified queries grew. However, file-based systems are very dependent upon the application developer, who has to write any queries or reports that are required. As a result, two things happened. In some organizations, the type of query or report that could be produced was fixed. There was no facility for asking unplanned (that is, spur-of-the-moment or ad hoc) queries either about the data itself or about which types of data were available.

In other organizations, there was a proliferation of files and application programs. Eventually, this reached a point where the DP Department, with its current resources, could not handle all the work. This put tremendous pressure on the DP staff, resulting in programs that were inadequate or inefficient in meeting the demands of the users, documentation that was limited, and maintenance that was difficult. Often, certain types of functionality were omitted including:

- ✓ There was no provision for security or integrity
- ✓ Recovery, in the event of a hardware or software failure, was limited or non-existent
- ✓ Access to the files was restricted to one user at a time – there was no provision for shared access by staff in the same department.

VI. Limited Data Sharing

With the traditional file processing approach, each application has its own private files, and users have little opportunity to share data outside their own applications.

VII. Lengthy Development Times

With traditional file processing systems, each new application requires that the developer essentially start from scratch by designing new file formats and descriptions and then writing the file access logic for each new program. The lengthy development times required are inconsistent with today's fast-paced business environment, in which time to market (or time to production for an information system) is a key business success factor.

VIII. Excessive Program Maintenance

The preceding factors all combined to create a heavy program maintenance load in organizations that relied on traditional file processing systems. In fact, as much as 80 percent of the total information system's development budget might be devoted to program maintenance in such organizations. This in turn means that resources (time, people, and money) are not being spent on developing new applications.

1.3 Database Approach

To overcome limitations or flaws of file processing system we follow Database systems. We first begin by defining some core concepts that are fundamental in understanding the database approach to managing data.

A. Data Models

Designing a database properly is fundamental to establishing a database that meets the needs of the users. Data models capture the nature of and relationships among data and are used at different levels of abstraction as a database is conceptualized and designed. The effectiveness and efficiency of a database is directly associated

with the structure of the database. Various graphical systems exist that convey this structure and are used to produce data models that can be understood by end users, systems analysts, and database designers.

"Graphical systems used to capture the nature and relationships among data."

A typical data model is made up entities, attributes, and relationships and the most common data modeling representation is the entity relationship model. Informally, a data model is a type of data abstraction that is used to provide this conceptual representation. The data model uses logical concepts, such as objects, their properties, and their interrelationships, that may be easier for most users to understand than computer storage concepts. Hence, the data model *hides* storage and implementation details that are not of interest to most database users.

The data models can be classified into four different categories:

i. Relational Model

The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name. Tables are also known as relations. The relational model is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes.

ii. Entity-Relationship Model

The entity-relationship (E-R) data model uses a collection of basic objects, called entities, and relationships among these objects. An entity is a "thing" or "object" in the real world that is distinguishable from other objects.

iii. Object-Based Data Model

Object-oriented programming (especially in Java, C++, or C#) has become the dominant software-development methodology. This led to the development of an object-oriented data model that can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity. The object-relational data model combines features of the object-oriented data model and relational data model.

iv. Semi structured Data Model

The semi structured data model permits the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast to the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. The Extensible Markup Language (XML) is widely used to represent semi structured data.

B. Entities

"An entity is a distinct object (a person, place, thing, concept, or event) in the organization that is to be represented in the database."

Customers and orders are objects about which a business maintains information. They are referred to as "entities." An entity is like a noun in that it describes a person, a place, an object, an event, or a concept in the business environment for which information must be recorded and retained.

"A person, a place, an object, an event, or a concept in the user environment about which the organization wishes to maintain data."

C. Attribute

An attribute is a property that describes some aspect of the object that we wish to record. The data you are interested in capturing about the entity (e.g., Customer Name) is called an attribute.

D. Relationship

A well-structured database establishes the relationships between entities that exist in organizational data so that desired information can be retrieved.

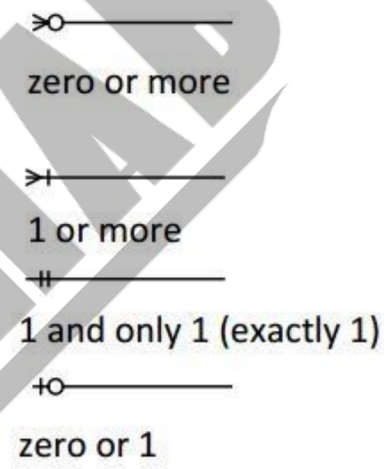
"A relationship is an association between entities."

The 1:1 Relationship: One entity in a 1:1 relationship can only be related to one other entity and vice versa. For example, every department has only one Chair and each chair can only head one department. 1:1 relationships should be rare.

The 1: M Relationship: The 1: M relationship is the norm for relational databases. For example, one painter usually has many paintings. Each painting was painted by only one painter, but a painter could have many paintings.

The M:N Relationship: A many to many (M:N) relationship is not supported directly in the relational environment. They are usually implemented by creating a new entity in 1:M relationships with the original entities. Example: Each CLASS is taken by many students, and each STUDENT can take many CLASSes.

Maximal cardinality and minimal cardinality are the indicators of the associations around a relationship. Maximal Cardinality refers to the maximum number of times an instance in one entity can be associated with instances in the related entity. Maximal Cardinality can be 1 or many and the symbol is placed on the outside ends of the relationship line, closest to the entity,



E. Relational Databases

Relational databases establish the relationships between entities by means of common fields included in a file, called a relation.

"A database that represents data as a collection of tables in which all data relationships are represented by common values in related tables."

F. Database Management Systems

"A software system that is used to create, maintain, and provide controlled access to user databases."

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

"A software system that is used to create, maintain, and provide controlled access to user databases."

G. Instances and Schemas

Database change over time as information is inserted or deleted. The collection of information s tore d in the database at a particular moment is called an **instance** of the database. The overall design of the database is called the database **schema**. Schemas are changed infrequently, if at all.

Database systems have several schemas, partitioned according to the levels of abstraction. The physical schema describes the database design at the physical level, while the logical schema describes the database design at the logical level. A database may also have several schemas at the view level, sometimes called subschemas that describe different views of the database.

H. Data Dictionary (Repository)

A repository is a centralized knowledge base for all data definitions, data relationships, screen and report formats, and other system components. A repository contains an extended set of metadata important for managing databases as well as other components of an information system.

"The place where all metadata for a particular database is stored."

I. Database Languages

There are three categories of database languages as follows:

- **Data-Definition Language**

It allows users to define the database, usually through a Data Definition Language (DDL). The DDL allows users to specify the data types and structures and the constraints on the data to be stored in the database. We specify a database schema by a set of definitions expressed by a special language called a data-definition language (DDL). The DDL is also used to specify additional properties of the data.

- **Data Manipulation Language**

A data-manipulation language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model. The types of access are:

- ✓ Retrieval of information stored in the database
- ✓ Insertion of new information into the database
- ✓ Deletion of information from the database
- ✓ Modification of information stored in the database

- **Query Language**

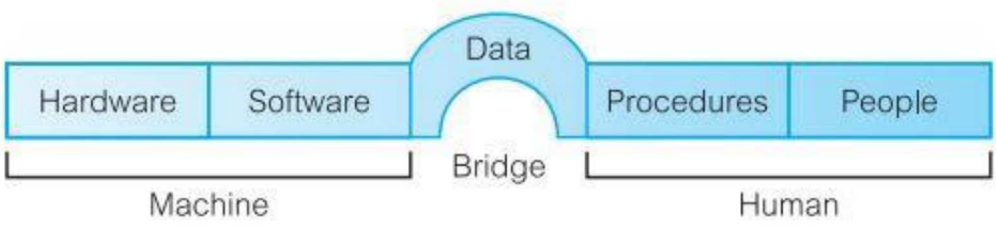
A query is a statement requesting the retrieval of information. The portion of a DML that involves information retrieval is called a query language. Although technically incorrect, it is common practice to use the terms query language and data-manipulation language synonymously.

J. (Database) Application Programs

Application programs are programs that are used to interact with the database. Users interact with the database through a number of application programs that are used to create and maintain the database and to generate information. These programs can be conventional batch applications or, more typically nowadays, they will be online applications. The application programs may be written in some programming language or in some higher-level fourth-generation language.

"A computer program that interacts with the database by issuing an appropriate request (typically an SQL statement) to the DBMS."

1.3.1 Components of the DBMS Environment



i. Hardware

The DBMS and the applications require hardware to run. The hardware can range from a single personal computer, to a single mainframe, to a network of computers. The particular hardware depends on the organization's requirements and the DBMS used. Some DBMSs run only on particular hardware or operating systems, while others run on a wide variety of hardware and operating systems. A DBMS requires a minimum amount of main memory and disk space to run, but this minimum configuration may not necessarily give acceptable performance.

ii. Software

The software component comprises the DBMS software itself and the application programs, together with the operating system, including network software if the DBMS is being used over a network.

iii. Data

Perhaps the most important component of the DBMS environment, certainly from the end-users' point of view, is the data. The database contains both the operational data and the metadata, the 'data about data'.

iv. Procedures

Procedures refer to the instructions and rules that govern the design and use of the database. The users of the system and the staff that manage the database require documented procedures on how to use or run the system. These may consist of instructions on how to:

- ✓ log on to the DBMS;
- ✓ use a particular DBMS facility or application program;
- ✓ start and stop the DBMS;
- ✓ make backup copies of the database;
- ✓ Handle hardware or software failures.
- ✓ Change the structure of a table, reorganize the database across multiple disks, improve performance, or archive data to secondary storage.

v. People

The final component is the people involved with the system. This includes data and database administrators, database designers, application developers, and the end-users.

- **Data and Database Administrators**

Data and database administration are the roles generally associated with the management and control of a DBMS and its data. The **Data Administrator** (DA) is responsible for the management of the data resource including database planning, development and maintenance of standards, policies and procedures, and conceptual/logical database design.

The **Database Administrator** (DBA) is responsible for the physical realization of the database, including physical database design and implementation, security and integrity control, maintenance of the operational system, and ensuring satisfactory performance of the applications for users. The role of the DBA is more technically oriented than the role of the DA, requiring detailed knowledge of the target DBMS and the system environment.

- **Database Designers**

In large database design projects, we can distinguish between two types of designer: Logical database designers and physical database designers.

The logical database designer is concerned with identifying the data (that is, the entities and attributes), the relationships between the data, and the constraints on the data that is to be stored in the database. The logical database designer must have a thorough and complete understanding of the organization's data and any constraints on this data

The physical database designer decides how the logical database design is to be physically realized. This involves:

- ✓ mapping the logical database design into a set of tables and integrity constraints;
- ✓ selecting specific storage structures and access methods for the data to achieve good performance;
- ✓ Designing any security measures required on the data.

- **Application Developers**

Once the database has been implemented, the application programs that provide the required functionality for the end-users must be implemented. This is the responsibility of the application developers. Typically, the application developers work from a specification produced by systems analysts. Each program contains statements that request the DBMS to perform some operation on the database. This includes retrieving data, inserting, updating, and deleting data.

- **End-Users**

The end-users are the 'clients' for the database, which has been designed and implemented, and is being maintained to serve their information needs. End-users can be classified according to the way they use the system:

Naive users
These users are typically unaware of the DBMS. They access the database through specially written application programs that attempt to make the operations as simple as possible. They invoke database operations by entering simple commands or choosing options from a menu. This means that they do not need to know anything about the database or the DBMS.

Sophisticated users
The sophisticated end-user is familiar with the structure of the database and the facilities offered by the DBMS. Sophisticated end-users may use a high-level query language such as SQL to perform the required operations. Some sophisticated end-users may even write application programs for their own use.

1.3.2 Advantages of DBMS

The primary advantages of a database approach, enabled by DBMS are:

i. Control of data redundancy

Good database design attempts to integrate previously separate (and redundant) data files into a single, logical structure. Ideally, each primary fact is recorded in only one place in the database.

The database approach attempts to eliminate the redundancy by integrating the files so that **multiple copies** of the same data are not stored. However, the database approach does not eliminate redundancy entirely, but controls the amount of redundancy inherent in the database. Sometimes, it is necessary to duplicate key data items to model relationships. At other times, it is desirable to duplicate some data items to improve performance.

ii. Data consistency

By eliminating or controlling data redundancy, we greatly reduce the opportunities for inconsistency. If a data item is stored only once in the database, any update to its value has to be performed only once and the new value is available immediately to all users. If a data item is stored more than once and the system is aware of this, the system can ensure that all copies of the item are kept consistent.

iii. Sharing of data

A database is designed as a shared corporate resource. Authorized internal and external users are granted permission to use the database, and each user (or group of users) is provided one or more user views into the database to facilitate this use.

A user view is a logical description of some portion of the database that is required by a user to perform some task. A user view is often developed by identifying a form or report that the user needs on a regular basis.

iv. Improved data integrity

Database integrity refers to the validity and consistency of stored data. Integrity is usually expressed in terms of constraints, which are consistency rules that the database is not permitted to violate. Constraints may apply to data items within a single record or they may apply to relationships between records.

v. Improved security

Database security is the protection of the database from unauthorized users. Without suitable security measures, integration makes the data more vulnerable than file-based systems. However, integration allows the DBA to define, and the DBMS to enforce, database security. This may take the form of user names and passwords to identify people authorized to use the database. The access that an authorized user is allowed on the data may be restricted by the operation type (retrieval, insert, update, delete).

vi. Enforcement of standards

When the database approach is implemented with full management support, the database administration function should be granted single point authority and responsibility for establishing and enforcing data standards.

These standards will include naming conventions, data quality standards, and uniform procedures for accessing, updating, and protecting data. The data repository provides database administrators with a powerful set of tools for developing and enforcing these standards.

vii. Economy of scale

Combining all the organization's operational data into one database, and creating a set of applications that work on this one source of data, can result in cost savings. In this case, the budget that would normally be allocated to each department for the development and maintenance of its file-based system can be combined, possibly resulting in a lower total cost, leading to an economy of scale. The combined budget can be used to buy a system configuration that is more suited to the organization's needs.

viii. Balance of conflicting requirements

Each user or department has needs that may be in conflict with the needs of other users. Since the database is under the control of the DBA, the DBA can make decisions about the design and operational use of the database that provide the best use of resources for the organization as a whole. These decisions will provide optimal performance for important applications, possibly at the expense of less critical ones.

ix. Improved data accessibility and responsiveness

With a relational database, end users without programming experience can often retrieve and display data, even when they cross traditional departmental boundaries. Data that crosses departmental boundaries is

directly accessible to the end-users. This provides a system with potentially much more functionality that can be used to provide better services to the end-user or the organization's clients.

Many DBMSs provide query languages or report writers that allow users to ask ad hoc questions and to obtain the required information almost immediately at their terminal, without requiring a programmer to write some software to extract this information from the database.

x. Increased productivity

A major advantage of the database approach is that it greatly reduces the cost and time for developing new business applications. There are three important reasons that database applications can often be developed much more rapidly than conventional file applications:

- ✓ Assuming that the database and the related data capture and maintenance applications have already been designed and implemented, the application developer can concentrate on the specific functions required for the new application, without having to worry about file design or low-level implementation details.
- ✓ The database management system provides a number of high-level productivity tools, such as forms and report generators, and high-level languages that automate some of the activities of database design and implementation. We describe many of these tools in subsequent chapters.
- ✓ Significant improvement in application developer productivity, estimated to be as high as 60 percent (Long, 2005), is currently being realized through the use of Web services, based on the use of standard Internet protocols and a universally accepted data format (XML).

xi. Program-Data Independence

The separation of data descriptions (metadata) from the application programs that use the data is called data independence. With the database approach, data descriptions are stored in a central location called the repository. This property of database systems allows an organization's data to change and evolve (within limits) without changing the application programs that process the data.

xii. Increased concurrency

In some file-based systems, if two or more users are allowed to access the same file simultaneously, it is possible that the accesses will interfere with each other, resulting in loss of information or even loss of integrity. Many DBMSs manage concurrent database access and ensure such problems cannot occur.

xiii. Improved backup and recovery services

Many file-based systems place the responsibility on the user to provide measures to protect the data from failures to the computer system or application program. This may involve taking a nightly backup of the data. In the event of a failure during the next day, the backup is restored and the work that has taken place since this backup is lost and has to be re-entered. In contrast, modern DBMSs provide facilities to minimize the amount of processing that is lost following a failure.

1.3.3 Disadvantages of DBMS

i. Complexity

The provision of the functionality we expect of a good DBMS makes the DBMS an extremely complex piece of software. Database designers and developers, the data and database administrators, and end-users must understand this functionality to take full advantage of it. Failure to understand the system can lead to bad design decisions, which can have serious consequences for an organization.

ii. Size

The complexity and breadth of functionality makes the DBMS an extremely large piece of software, occupying many megabytes of disk space and requiring substantial amounts of memory to run efficiently.

iii. Cost of DBMSs

The cost of DBMSs varies significantly, depending on the environment and functionality provided. There is also the recurrent annual maintenance cost, which is typically a percentage of the list price.

iv. Additional hardware costs

The disk storage requirements for the DBMS and the database may necessitate the purchase of additional storage space. Furthermore, to achieve the required performance, it may be necessary to purchase a larger machine, perhaps even a machine dedicated to running the DBMS. The procurement of additional hardware results in further expenditure.

v. Cost of conversion

The term legacy system is widely used to refer to older applications in an organization that are based on file processing and/or older database technology. The cost of converting these older systems to modern database technology—measured in terms of dollars, time, and organizational commitment—may often seem prohibitive to an organization. This cost also includes the cost of training staff to use these new systems, and possibly the employment of specialist staff to help with the conversion and running of the system. This cost is one of the main reasons why some organizations feel tied to their current systems and cannot switch to more modern database technology.

vi. Need for Explicit Backup and Recovery

A shared corporate database must be accurate and available at all times. This requires that comprehensive procedures be developed and used for providing backup copies of data and for restoring a database when damage occurs. These considerations have acquired increased urgency in today's security-conscious environment. A modern database management system normally automates many more of the backup and recovery tasks than a file system.

vii. Performance

Typically, a file-based system is written for a specific application, such as invoicing. As a result, performance is generally very good. However, the DBMS is written to be more general, to cater for many applications rather than just one. The effect is that some applications may not run as fast as they used to.

viii. Higher impact of a failure

The centralization of resources increases the vulnerability of the system. Since all users and applications rely on the availability of the DBMS, the failure of certain components can bring operations to a halt.

1.3.4 Database-System Applications

Databases are widely used. Here are some representative applications:

- **Enterprise Information**
  - ✓ Sales: For customer, product, and purchase information.
  - ✓ Accounting: For payments, receipts, account balances, assets and other accounting information.
  - ✓ Human Resources: For information about employees, salaries, payroll taxes, and benefits, and for generation of paychecks.
  - ✓ Manufacturing: For management of the supply chain and for tracking production of items in factories, inventories of items in warehouses and stores, and orders for items.

- ✓ Online retailers: For sales data noted above plus online order tracking, generation of recommendation lists, and maintenance of online product evaluations.

- **Banking and Finance**
  - ✓ Banking: For customer information, accounts, loans, and banking transactions.
  - ✓ Credit card transactions: For purchases on credit cards and generation of monthly statements.
  - ✓ Finance: For storing information about holdings, sales, and purchases of financial instruments such as stocks and bonds; also for storing real-time market data to enable online trading by customers and automated trading by the firm.
- **Universities:** For student information, course registrations, and grades (in addition to standard enterprise information such as human resources and accounting).
- **Airlines:** For reservations and schedule information. Airlines were among the first to use databases in a geographically distributed manner.
- **Telecommunication:** For keeping records of calls made, generating monthly bills, maintaining balances on prepaid calling cards, and storing information about the communication networks.

Over the course of the last four decades of the twentieth century, use of databases grew in all enterprises. In the early days, very few people interacted directly with database systems, although without realizing it, they interacted with databases indirectly—through printed reports such as credit card statements, or through agents such as bank tellers and airline reservation agents. Then automated teller machines came along and let users interact directly with databases. Phone interfaces to computers (interactive voice-response systems) also allowed users to deal directly with databases—a caller could dial a number, and press phone keys to enter information or to select alternative options, to find flight arrival/departure times, for example, or to register for courses in a university.

The importance of database systems can be judged in another way — today, database system vendors like Oracle are among the largest software companies in the world, and database systems form an important part of the product line of Microsoft and IBM.

❖ Personal Databases

Personal databases are designed to support one user. Personal databases have long resided on personal computers (PCs), including laptops, and increasingly on smartphones and PDAs. The purpose of these databases is to provide the user with the ability to manage (store, update, delete, and retrieve) small amounts of data in an efficient manner. Simple database applications that store customer information and the details of contacts with each customer can be used from a PC and easily transferred from one device to the other for backup and work purposes.

Personal databases are widely used because they can often improve personal productivity. However, they entail a risk: The data cannot easily be shared with other users.

❖ Two-Tier Client/Server Databases

As noted earlier, the utility of a personal (single-user) database is quite limited. Often, what starts off as a single-user database evolves into something that needs to be shared among several users. A workgroup is a relatively small team of people (typically fewer than 25 persons) who collaborate on the same project or application or on a group of similar projects or applications. These persons might be engaged (for example) with a construction project or with developing a new computer application and need to share data among the group.

The most common method of sharing data for this type of need is based on creating a two-tier client/server application as shown in Figure 1-11. Each member of the workgroup has a computer, and the computers are linked by means of a network (wired or wireless local area network [LAN]). In most cases, each computer has a

copy of a specialized application (client) that provides the user interface as well as the business logic through which the data are manipulated. The database itself and the DBMS are stored on a central device called the database server, which is also connected to the network. Thus, each member of the workgroup has access to the shared data.

- **Multitier Client/Server Databases**

One of the drawbacks of the two-tier database architecture is that the amount of functionality that needs to be programmed into the application on the users' computer can be pretty significant because it needs to contain both the user interface logic as well as the business logic. This, of course, means that the client computers need to be powerful enough to handle the programmed application. Another drawback is that each time there is a change to either the business logic or user interface, each client computer that has the application needs to be updated.

To overcome these limitations, most modern applications that need to support a large number of users are built using the concept of multi-tiered architecture. In most organizations, these applications are intended to support a department (such as marketing or accounting) or a division (such as a line of business), which is generally larger than a workgroup (typically between 25 and 100 persons).

In a three-tiered architecture, the user interface is accessible on the individual users' computers. This user interface may be either Web browser based or written using programming languages such as Visual Basic.NET, Visual C#, or Java.

- **Enterprise Applications**

An enterprise (that's small "e," not capital "E," as in Starship) application/database is one whose scope is the entire organization or enterprise (or, at least, many different departments). Such databases are intended to support organization-wide operations and decision making. Note that an organization may have several enterprise databases, so such a database is not inclusive of all organizational data.

The evolution of enterprise databases has resulted in two major developments:

I. Enterprise resource planning (ERP) systems
II. Data warehousing implementations

Enterprise resource planning (ERP) systems: A business management system that integrates all functions of the enterprise, such as manufacturing, sales, finance, marketing, inventory, accounting, and human resources. ERP systems are software applications that provide the data necessary for the enterprise to examine and manage its activities.

Data warehouse: An integrated decision support database whose content is derived from the various operational databases.

1.3.5 When Not to Use a DBMS

In spite of the advantages of using a DBMS, there are a few situations in which a DBMS may involve unnecessary overhead costs that would not be incurred in traditional file processing. The overhead costs of using a DBMS are due to the following:

- ✓ High initial investment in hardware, software, and training
- ✓ The generality that a DBMS provides for defining and processing data
- ✓ Overhead for providing security, concurrency control, recovery, and integrity functions

Therefore, it may be more desirable to use regular files under the following circumstances:

- ✓ Simple, well-defined database applications that are not expected to change at all
- ✓ Stringent, real-time requirements for some application programs that may not be met because of DBMS overhead
- ✓ Embedded systems with limited storage capacity, where a general-purpose DBMS would not fit
- ✓ No multiple-user access to data

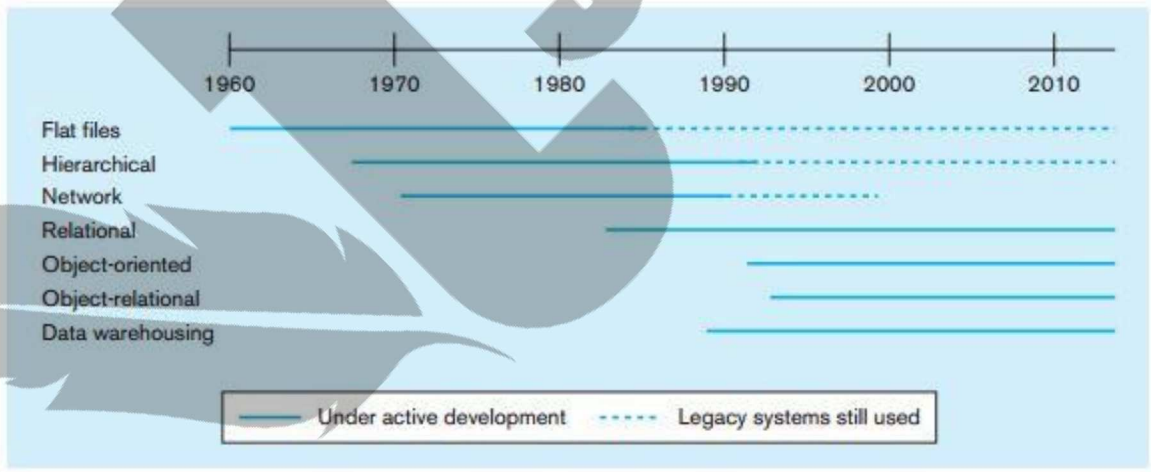
1.3.6 Evolution of Database Systems

Database management systems were first introduced during the 1960s and have continued to evolve during subsequent decades. Database management systems were developed to overcome the limitations of file processing systems, described in a previous section. To summarize, some of the following four objectives generally drove the development and evolution of database technology:

- ✓ The need to provide greater independence between programs and data, thereby reducing maintenance costs
- ✓ The desire to manage increasingly complex data types and structures
- ✓ The desire to provide easier and faster access to data for users who have neither a background in programming languages nor a detailed understanding of how data are stored in databases
- ✓ The need to provide ever more powerful platforms for decision support applications

i. 1960s

File processing systems were still dominant during the 1960s. However, the first database management systems were introduced during this decade and were used primarily for large and complex ventures such as the Apollo moon-landing project. We can regard this as an experimental "proof-of-concept" period in which the feasibility of managing vast amounts of data with a DBMS was demonstrated. Also, the first efforts at standardization were taken with the formation of the Data Base Task Group in the late 1960s.



ii. 1970s

During this decade, the use of database management systems became a commercial reality. The hierarchical and network database management systems were developed, largely to cope with increasingly complex data structures such as manufacturing bills of materials that were extremely difficult to manage with conventional file processing methods. The hierarchical and network models are generally regarded as first-generation DBMS. Both approaches were widely used, and in fact many of these systems continue to be used today. However, they suffered from the same key disadvantages as file processing systems: limited data independence and lengthy development times for application development.

iii. 1980s

To overcome these limitations, E. F. Codd and others developed the relational data model during the 1970s. This model, considered second-generation DBMS, received widespread commercial acceptance and diffused throughout the business world during the 1980s. With the relational model, all data are represented in the form of tables. Typically, SQL is used for data retrieval. Thus, the relational model provides ease of access for nonprogrammers, overcoming one of the major objections to first-generation systems. The relational model has also proven well suited to client/server computing, parallel processing, and graphical user interfaces (Gray, 1996).

iv. 1990s

The 1990s ushered in a new era of computing, first with client/server computing, and then with data warehousing and Internet applications becoming increasingly important. Whereas the data managed by a DBMS during the 1980s were largely structured (such as accounting data), multimedia data (including graphics, sound, images, and video) became increasingly common during the 1990s. To cope with these increasingly complex data, object-oriented databases (considered third generation) were introduced during the late 1980s (Grimes, 1998).

Because organizations must manage a vast amount of structured and unstructured data, both relational and object-oriented databases are still of great importance today. In fact, some vendors are developing combined object-relational DBMSs that can manage both types of data.

v. 2000 and Beyond

Currently, the major type of database that is still most widely used is the relational database. However, object-oriented and object-relational databases continue to garner attention, especially as the growth in unstructured content continues. Another recent trend is the emergence of NoSQL (Not Only SQL) databases. NoSQL is an umbrella term that refers to a set of database technologies that is specifically designed to address large (structured and unstructured) data that are potentially stored across various locations. Popular examples of NoSQL databases are Apache Cassandra and Google's BigTable.

This search for non-relational database technologies is fueled by the needs of Web 2.0 applications such as blogs, wikis, and social networking sites (Facebook, MySpace, Twitter, LinkedIn, etc.) and partially by how easy it has become to create unstructured data such as pictures and images. As larger computer memory chips become less expensive, new database technologies to manage in-memory databases are emerging. This trend opens up new possibilities for even faster database processing.

An emerging trend that is making it more convenient to use database technologies is that of cloud computing. One popular technology available in the cloud is databases. Databases, relational and non-relational, can now be created, deployed, and managed through the use of technologies provided by a service provider.