7 Introduction to Database

7.1 INTRODUCTION

The known facts that can be recorded and that have implicit meaning are called data. For example, consider the names, telephone numbers, and addresses of the people you know. These related data items pertaining to each single individual constitute a data record. A database is an organized collection of data records (or related data), that is typically stored on some electronic media (say a disk) so that it is accessible by many concurrent users of such data items. A database is normally application oriented, as it is identified by an application area for which it is designed and populated. For example, one database may be designed and populated to store accounting and financial records, yet another may be meant to store Human Resource data items pertaining to employees and payroll. A database has the following characteristics, technically called, its implicit properties:

(*a*) A database represents some aspect of the real world, sometimes called the miniworld. Changes to the miniworld are reflected in the database.

(*b*) 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.

(*c*) 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.

In other words, a database has some source from which data are derived, some degree of interaction with events in the real world, and an audience that is actively interested in the contents of the database. In order to manage these databases (that is to create, populate and update), now a days a software is used, which is called the Database Management System (DBMS). However, prior to the advent of DBMS, the databases were created and managed using the traditional file systems as discussed below:

Traditional File System (TFS)

It is an electronic mechanism to store and arrange computer files that contain data and information. Primarily, it organizes these files into a database for the organization, storage, manipulation, and retrieval by the computer's operating system. This is supported by a file processing system, which is a collection of files and programs to access and modify these files. Generally, new files and programs are added by different programmers, as and when new data and information needs to be stored and new ways to access information are needed. In the context of databases, TFS is generally used for storing related, structured data, with well-defined data formats, in an efficient manner for insert, update and retrieval operations, depending on application. On the other hand, a simple file system is an unstructured data store for storing arbitrary and normally unrelated data. Some important features of TFS, in the context of databases, are as follows:

- It is characterized by a collection of interrelated ASCII data files;
- The collection of related data constitutes a data record;
- A collection of similar data records is placed together in an ASCII file to become a part of the database;
- A collection of related data files constitutes the backbone of traditional databases.
- The application programs are written using programming languages to operate on such a collection of ASCII files.
- COBOL (common business oriented language), being a programming language, used to have a profound influence on an earlier generation of file processing systems to operate on the collection of these ASCII files, mainly because of its file-oriented processing approach.
- Specific application programs need to be developed to perform a specific task.

A difference between flat file databases must be appreciated as many persons consider both as alike. A flat **file database** is stored on its host computer system as an ordinary unstructured **file** called a "flat **file**". In order to access the

structure of the data and also to manipulate it, the **file** must be read completely and in its entirety into the computer's memory. TFS is always organized as a set of structured file system. However, the TFS emcompassed a number of problems and suffered from various limitations as discussed below:

Problems and limitations of TFS:

1. **Data Security:** The data stored in the TFS is not secure because of being easily accessible by a variety of users. Besides, these electronic files are usually accessible on a network, which allows an unauthorized person to gain access to electronic data over the Internet through hacking methods. Electronic data may also be damaged inadvertently or advertently by the users and are under constant attack by malicious software like computer viruses. In an online banking application where we store the account related information of all customers in flat files, a customer shall have access not only to his account related details but also to the details of others, which are supposed to be private and confidential. It is difficult to restrict the data access to a specific information and this becomes a big security issue.
2. **Data Redundancy and Inconsistency:** In TFS of data storage model, a particular set of data is likely to get duplicated in multiple files, thereby resulting in a higher storage and access cost, besides causing data inconsistency. For example, different departments of an organisation tend to maintain data records of same employees in their respective departments because of the requirement of their specific application programs. This results in data redundancy. If a change is made to data stored in one file of employees in a particular department, a similar change need to be caused accordingly in other departments that maintain the employees' records. A failure to do so because of communication gap and updating coordination shall result in data inconsistency within the organisation in respect of their employees. In a typical TFS, various copies of same data may contain different values. Data is not consistent in this system, it means if a data item needs to be changed then all the files containing that data need to be modified. It may create a risk of out dated values of data. However, it is possible to design file systems with minimal redundancy.
3. **Data Isolation:** It implies the non-availability of data in one file because of being scattered in various files having different formats. As a result, the data stands isolated in File Processing System because of being stored in different files. If you want to extract data from two file then you are required to ascertain as to which part of the file is needed and how they are related to each other. It becomes difficult to retrieve the required data while writing a new application program.
4. **Data Integrity:** It is said to exist when the database system enforces certain consistency constraints. A programmer always puts these constraints

in the programs by adding some checks. In File Processing System, poor data integrity often arises because of the difficulty in adding new constraints in application programs subsequent to their implantation. Consider a situation wherein the date of birth of an employee that exceeds the date of joining the organization is erroneously allowed to be stored in data files because of the absence of an independent mechanism to cross check such errors.

5. **Lack of data Catalogue:** The data files used in TFS does not contain any information as to the structure and layout of the stored data. This is because no data catalogue maintained by the TFS. As a result, the users of data files, the application programmers, have to struggle a lot in enquiring and understanding the implied data layout used while storing the data.
6. **Program and Data Dependence:** In traditional file approach, application programs are closely dependent on the files in which data is stored. If we make any changes in the physical format of the file(s), say addition of a data field, all application programs need to be modified to ensure perform correct read and write operation on data files. Consequently, for each of the application programs that a programmer writes or maintains, the programmer must be concerned with data management. There is no centralized execution of the data management functions. Data management is scattered among all the application programs.
7. **Lack of Flexibility:** The TFS are good enough to allow the retrieval of information for predetermined and planned requests for data. If any unanticipated data and information is required at any stage, huge programming effort is needed to make such an information available, subject to the availability of the information in the files. Quite often, the information becomes available, after a decision opportunity is lost and it may no longer be required or useful. Consider a traditional file based software application, which generates employees' monthly salary report. Suppose it is desired to retrieve all the employee details whose monthly salary exceeds ₹ 40,000. It is not easy to generate such on-demand reports simply because is not planned and therefore not programmed. A lot of time shall be needed for application developers to modify the application to meet such requirements.
8. **Concurrent Access Anomalies:** Many TFS, through their application programs allow multiple users to access and update the same set of data items simultaneously. However, these concurrent updates often may result in inconsistent data. In order to prevent this possibility, the TFS needs to maintain some form of supervision. But supervision is difficult because data may be accessed by many different application programs and it is difficult to have coordination among these application programs. Consider a Human Resource information system, which has the data of

all employees. While an employee is updating his address details in the system and at the same time, an administrator may be taking a report containing the data of all employees. There a possibility of the administrator reading an incorrect address during this concurrent access.

9. **Atomicity Problem:** Atomicity, after updating data values, means that data are either completely saved or abandoned. Any system may fail at any time and at that time it is desired that data should be in a consistent state. Unfortunately, it is difficult to roll back an incomplete transaction with TFS. Consider, if you are buying an air ticket and you are in the process of money transaction. Suddenly, the internet gets disconnected then you may or may not have paid for the ticket. If you have paid then your ticket will be booked and if not then you will not be charged anything. That is called consistent state to imply that either you have succeeded or failed to buy the ticket. In TFS, it is likely that the payment has been made while the ticket has not been allotted to the buyer.

These difficulties led to the emergence and development of modern database systems, duly supported by DBMS. A **Database Management System (DBMS)** is a collection of related software (or programs) that are meant to enable the users of database to create and maintain a database. The DBMS, therefore, is a *general-purpose software system* that is used to facilitate the processes of *defining, constructing, updating and querying* databases for various applications. Let us understand each of these processes in brief:

- **Defining:** The act of specifying the data types, structures, and constraints for the data to be stored in the database is to define a database.
- **Constructing:** The process of storing the data itself on some storage medium that is controlled by the DBMS, is called constructing or populating a database.
- **Updating:** It includes editing and inserting data records. The existing data records may be edited to cause necessary corrections in stored data values. The changes in the mini world keep occurring. As result, new data emerges that is required to be inserted in the database;
- **Querying:** The purpose of querying the database is to retrieve specific data, with or without processing for the purpose of preparing reports and decision-making.

Besides, the DBMS software is also expected to perform such functions as:

- security of data
- ensuring integrity of data by locking, logging and defining application oriented rules (including triggers)
- batch and on-line programs
- backups and recoveries

- optimal performance
- maximum availability
- catalog and directory of database objects
- management of the buffer pools
- interface to other systems programs
- support to user interface packages, such as the popular SQL interface for relational database systems;

It is not necessary to use general-purpose DBMS software to implement a computerized database. Alternatively, we may write our own tailored made set of programs to create and maintain the database. Such an effort, in effect leads to creating our own *special-purpose* DBMS software. Irrespective of whether we use a general-purpose DBMS or rely upon tailored made programs, we usually have to employ a considerable amount of software to manage and maintain the database. The database and DBMS software together constitute a **database system.**

On the basis of design, there are three traditional types of database management systems: hierarchical, relational, and network. Currently popular general-purpose database management systems are Oracle; MS SQL Server, Sybase (same as Microsoft's SQL Server but on a different platform); IBM's DB2, IMS, and SQL/DS; Ingres; Informix; and smaller, but reasonably powerful off-the-shelf products such as dBase, Access, FoxPro, Paradox, and dozens of others.

The choice of a DBMS is often influenced by such factors as:

- the computing platform: hardware and operating system
- the volume of data to be managed
- the number of transactions required per second
- existing applications (or interfaces) that an organization may have
- support for heterogeneous and/or distributed computing
- cost of acquisition and maintenance, and
- vendor support

Database Technology

It refers to a set of techniques that are used to design a database. These techniques use certain concepts, which are crucial to the creation of structure and development of the design. These concepts are: reality, data, database, information, DBMS and database system. A brief description of these concepts is as given below:

(*a*) **Reality:** It implies some aspect of the real world. It consists of an organization, its different components and the environment in which the organization exists and operates. Any organization includes people, fa-

cilities and other resources that are organized to achieve certain goals. Each organization operates within an environment. While operating, the organization interacts, influences and gets influenced by the environment.

> An organization may be viewed as a system consisting of several components called its sub-systems. Each of these sub-systems follows certain procedures and continuously interacts with each other and their external environment to accomplish the goals of organization. During the course of their interaction, events take place, which take the shape of data items. These sub-systems communicate continuously with AIS to provide data and seek information. A part of AIS is Financial Accounting System, which is designed for processing accounting transactions. For example: A firm uses a voucher to document an accounting transaction. The contents of voucher consist of accounting data, which need be stored in an organized manner.

This continuous interaction results in real world transactions. These transactions are analyzed with a view to identify the components called data items. A data item is the smallest named unit of data in an information system. In a transaction, the names of accounts (or their accounting codes), date of transaction, amount etc is all data items.

(*b*) **Data:** are known facts that can be recorded and which have implicit meaning. Data represent facts concerning people, places, objects, entities, events or even concepts. Data can be quantitative and qualitative or they can be financial and non-financial in character. Consider the following transaction:

April 1, 2015 Commenced business with Cash ₹ 5,00,000.

that has been recorded below using a transaction voucher as shown below, using either simple transaction voucher, Figure 7.1 or Debit and Credit voucher shown in, Figure 7.2(*a*) and Figure 7.2(*b*).

M/s Soumya Computers and Services Pvt Ltd.

Transaction Voucher

Voucher No: 01 **Date:** 01-Apr-15

Debit Account: 631001 Cash Account

Credit Account: 110001 Capital Account

Amount in ₹ : ₹ 5,00,000

Narration: Commenced business with Cash.

Authorized By: Aditya **Prepared By :** Smith

FIGURE 7.1: A SAMPLE TRANSACTION VOUCHER TO DOCUMENT SIMPLE TRANSACTIONS WITH ONE DEBIT AND ONE CREDIT

Debit Voucher

Voucher No: 05 **Date:** 03-Apr-15

Credit Account: 631001 Cash Account **M/s Soumya Computers and Services**

Debit Accounts

S. No.	Code	Name of Account	Amount	Narration
1	711001	Purchases	50,000	Purchases from R.S & Sons
2	711003	Carriage Inwards	2,000	Paid to M/s Saini Transports
		Total Amount	52,000	

Authorized By: Aditya **Prepared By :** Smith

FIGURE 7.2(A): SAMPLE VOUCHERS FOR MULTIPLE DEBITS AGAINST ONE CREDIT

Credit Voucher

Voucher No: 01 **Date:** 01-Apr-15

Debit Account: 631001 Cash Account **M/s Soumya Computers and Services**

Credit Accounts

S.No.	Code	Name of Account	Amount	Narration
1	110001	Capital Account	5,00,000	Commenced Business with cash
		Total Amount	5,00,000	

Authorized By: Aditya **Prepared By :** Bimal

FIGURE 7.2(B): SAMPLE VOUCHERS FOR MULTIPLE CREDITS AGAINST ONE DEBIT

This transaction, before being recorded through a Transaction Voucher, as shown in **Figure: 7.1**, need be decomposed into its data contents as: "01", 01-Apr-15, 642001, Bank Account, 110001, Capital Account, ₹ 5,00,000. Data are not useful for decision-making unless they are processed to suit to the requirements of decision-making situation. However, credit voucher [**Figure 7.2(b)**] could be used to document a simple transaction with one debit and one credit. A cash purchase transaction using debit voucher has been recorded as shown in **Figure 7.2(a)**.

(*c*) **Database:** The data, after being collected, must be stored so that different people may use them. This requires the creation of a database. A database is a shared collection of interrelated data tables, files or structures, which are designed to meet the varied informational needs of an organization (*See* Example database in **Figure 7.20.** It has two important properties (or characteristics): one it is integrated and second it is shared. *Integrated property* implies that distinct data tables have been logically organized. The purpose is to reduce or eliminate redundancy (or duplicity) and also to facilitate better data access. The *shared property* means that all those who are authorized to use data/information have access to relevant data. Thus, a database is a collection of related data that represents some aspect of the real world (called *mini-world*