Database Management System

Introduction

Why any data management system is required? Answer is most simple because we need to manage, manipulate and store data according to our requirements. Perhaps the most important component of the DBMS environment, certainly from the end-users' point of view, is the data. Organizations usually prosper when their managers act on the basis of relevant information. To generate relevant information efficiently, you need quick access to the data from which the required information is produced.

The database is such an integral part of our day to day life that often we are not aware we are using one. Below are some examples of Databases
When you purchase goods from your local supermarket, it is likely that a database will be accessed. The checkout assistant will run a bar code reader over each of your purchases. This will be linked to a database application program, which uses the bar code to find out the price of the item from a product database. The program then reduces the number of such items in stock and rings the price up on the till. If the reorder level falls below a threshold, the system may automatically place an order to obtain more stocks of that item. If a customer telephones the super market, an assistant can check whether an item is in stock by running an application program that determines availability from the database.
When you purchase goods using your credit card, the assistant normally checks that you have sufficient credit left to make the purchase. This check may be carried out by telephone or it may be done automatically by a card reader linked to a computer system. In either case, there is a database somewhere that Contains information about the purchases that you have made using your credit card. To check your credit, there is an application program that uses your credit card number to check that the price of the goods you wish to buy together with the sum of the purchases you have already made this month is within your credit limit. When the purchase is confirmed, the details of the purchase are added to this database. The application program will also access the database to check that the credit card is not on the list of stolen or lost cards before authorizing the purchase. There will be other application programs to send out monthly statements to each cardholder and to credit accounts when payment is received.

When you make enquiries about a holiday, the travel agent may access several databases containing holiday and flight details. When you book your holiday, the database system has to make all the necessary booking arrangements. In this case, the system has to ensure that two different agents do not book the same holiday or overbook the seats on flight. For example, if there is only one seat left on the flight from Karachi to New York and two agents try to reserve the last seat at the same time, the system has to recognize this situation, allow one booking agent to proceed and inform the other agent that there are no seats available. The travel agent may have another, usually separate, database system for invoicing.

Whenever you visit your local library, there is probably a database containing details of the books in the library, details of the users, reservations, and so on. There will be a

computerized index, which allows users to find a book based on its title, or its authors, or its subject area. The database system handles reservations to allow a user to reserve a book and to be informed by post when the book is available. The system also sends out reminders to borrowers who have failed to return books on the due date. Typically, the system will have a bar code reader, similar to , that used by the supermarket described earlier, which is used to keep track of books coming in and going out of the library.

Whenever you wish to take out insurance, for example personal insurance, building, contents insurance for your house, or car insurance, your broker may access several databases containing figures for various insurance organizations. After personal details, such as name, address, age, and whether you drink or smoke, have been supplied, they are used by the database system to determine the cost of insurance. The broker can search several databases to find the organization that gives you the best deal.

Data

An important question is what is meant by data? The word DATA has been derived from the Latin which means "to give" DATA is really given "*raw facts*". Some times 'data' and 'information' are interchangeably used, but the question is that is this interchangeability correct?, some writers use it interchangeably which is not considered as standard.

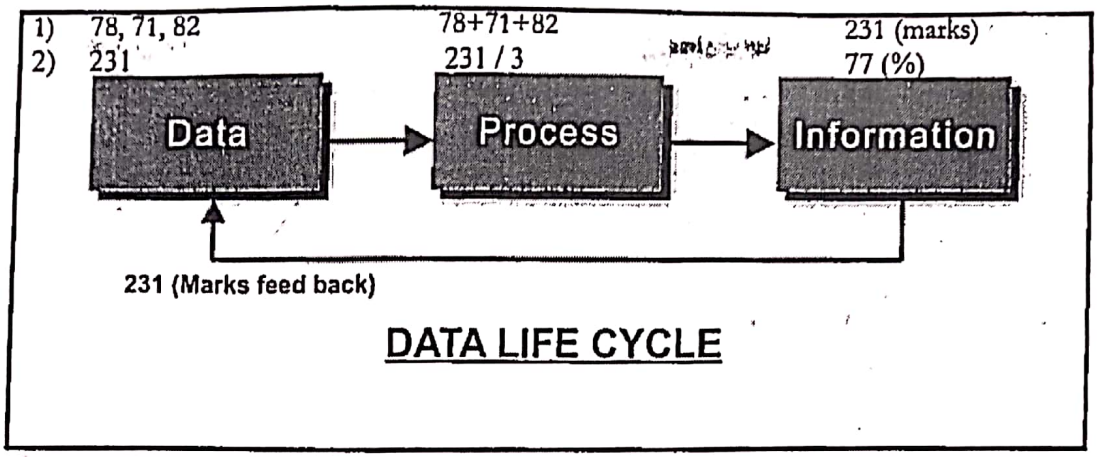
The deviation or difference as explained by some writers is that Data is considered as raw facts where as information is considered as processed data from which we could be able to extract some meaningful figures/facts about facts/record. But this information becomes part of data as it could be reentered/reused for further processing, e.g. Lets suppose that there is a student who has obtained three different marks in three different subjects, like 78 in Math, 71 in Physics and 82 in chemistry, then what these # corresponding to subject shows, actually they are raw facts about specific student who has got these marks in three different subjects. If we sum up these numbers we get the combined marks, as in this case

78+71+82 = 231

But the summed marks i.e. 231 could further be used to calculate the percentage,

$$\frac{231}{3} = 77\ \%$$

but according to our discussion, we use raw facts to get information through some process, in this case we already have processed raw facts to get information (i.e. 231 marks), so how could we use this information to get back further information, in fact this is a cyclic process which is called data life cycle



Historical roots of Database: File Management Systems (FMS)

Historically the first computer applications focused on the clerical tasks order/entry, processing, payroll, work scheduling and so on. Such applications accessed data stored in computer files. Requests for information quickly followed, reports were generated to transform stored data into information useful for management decisions. It may include things like how many widgets were sold, by whom, and to whom?

Although the file system is now largely obsolete, there are several good reasons for studying file system in some detail:

- File system provide a useful historical perspective on how we handle data.
- The design complexity of a database might be easier to understand once the relatively simple file systems' characteristics are understood.
- Understanding the problems inherent in file-based systems may prevent us from repeating these problems in database systems. In other words, we should learn from our earlier mistakes. However, we have learned from this work that there are better ways to handle data.
- If you wish to convert a file based system to a database system, understanding how the file system works will be extremely useful, if not essential.

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

Organization of the data within the file folders was determined by the data's expected use. Ideally, the contents of data within the file folder were logically related. For example, a file folder in a doctor's office may contain patient data – one file folder for each patient. All the data in that file folder described only that patient's medical history. In short, all the data in the file folder were related to only that individual patient. Similarly, a personnel manager might organize personnel data by category of employment (clerical, technical, sales, administrative, etc,). Therefore, a file folder labeled technical would contain data pertaining to only those people whose duties were properly classified as technical.

File based systems were an early attempt to computerize the manual filing system that we are all familiar with. For example, in an organization a manual file is set up to hold all external and internal correspondence relating to a project, product, task, client, or employee. Typically, there are many such files, and for safety they are labeled and stored in one or more cabinets. For security, the cabinets may have locks or may be located in secure areas of the building. In our own home, we probably have some sort of filing system, which contains receipts, guarantees, invoices, bank statements and such like.

When we need to look something up, we go to the filing system and search through the system starting from the first entry until we find what we want. Alternatively, we may have an indexing system that, help locate what we want more quickly. For example, we may have divisions in the filing system or separate folders for different types of item that are in some way logically related.

The manual filing system works well while the number of items to be stored is small. It even works quite adequately when there are large numbers of items and we have, only to store and retrieve them. However, file manual filing system breaks down when we have to cross-reference or process the information in the files. For example, a typical real estate agent's office might have a separate file for each property for sale or rent, each potential buyer and renter and each member of staff. Consider the effort that would be required to answer the following questions:

- What three bedroom properties do you have for sale with a garden and garage?
- What flats do you have for rent within three miles of the city center?
- What is the average house price?
- What is the average rent for a two-bedroom flat!
- What is the total annual salary bill for staff?
- What was last year's monthly turnover derived from property sales?
- How does last month's turnover compare with the Projected figure for this month?
- What is the expected monthly turnover for the next financial year?

Increasingly nowadays, clients, senior managers, and staff want more and more information. In some areas, there is a legal requirement to produce detailed monthly, quarterly and annual reports. Clearly, the manual system is totally inadequate for this type of work. The file-based system was developed in response to the needs of industry

for more efficient data access. However, rather than establish a centralized store for the organization's operational data, a decentralized approach was taken, where each department, with the assistance of Data Processing (DP) staff, stored and controlled its own data.

The Sales Department is responsible for the selling and renting of properties. For example, whenever a client approaches the Sales Department with a view to marketing his or her property for rent, a form is completed. This gives details of the property such as address and number of rooms together with the owner's details. The Sales Department also handles enquiries from potential renters, and a form similar to the one shown in Figure on next page is completed for each one. With the assistance of the DP Department, the Sales Department creates an information system to handle the renting of property. The system consists of three files containing property, owner, and renter details, as illustrated in Figure Lec1_1.2. For simplicity, details relating to members of staff, branch offices, and business owners are omitted.

The Contracts Department is responsible for handling the lease agreements associated with properties for rent. Whenever a client agrees to rent a property, a form is filled in one by one of the Sales staff giving the renter and property details, as shown in Figure Lec1_1.3. This form is passed to the Contracts Department who allocates a lease number and completes the payment and rental period details. Again, with the assistance of the DP Department, the Contracts Department creates all information system to handle lease agreements. The system consists of three files storing lease, property, and renter details, containing similar data to that hold by the Sales Department, as illustrated in Figure Lec1_1.4.

The situation illustrated in Figure Lec1_1.5 shows each department accessing its own files through application programs written specially for them. Each set of departmental application programs handles data entry, file maintenance, and file generation of a fixed set of specific reports. What is more important, the physical structure and storage of the data files and records are defined in the application code.

Fig: Lec1 1.1 Property for Rent Details form Renter Details form

PROPERTY FOR RENT

P. No.	Street	Area	City	P. Code	Typ	Rooms	Rent	One
PA14	16 Holhead	Deer	Aberdeen	AB7 5SU	House	6	650	CO46
PL94	6 Argyll S:	Kilburn	London	NW2	Flat	4	400	GO87
PG4	6 Lawrence St	Partick	Glasgow	G11 9QX	Flat	3	350	CO40
PG36	2 Manor Rd		Glasgow	G32 9QX	Flat	3	375	CO93
PG21	18 Dale Rd	Hyndland	Glasgow	G12	House	5	600	CO87
PG16	5 Novar Dr	Hyndland	Glasgow	G12 9AX	Flat	4	450	CO93

OWNER

One	F.Name	L.Name	Address	Tel No.
CO46	Joe	Keogh	2 Fergus Dr, Banchory, Aberdeen AB2 7SX	01224 861212
CO87	Carol	Furrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St Shawlands, Glasgow G42	0141 943 1728
CO93	Tony	Shaw	12 Park PL Billhead, Glasgow G4 0QR	141-225-7025

RENTER

R.No.	F.Name	L.Name	Address	Tel. No.	Pref. Type	Max Rent
CR76	John	Kay	56 High St, Putney, London SW1 4EH	0171-774-5632	Flat	425
CR56	Aline	Stewart	64 Fern Dr, Pollock, Glasgow G42 0BL	0141-848-1825	Flat	350
CR74	Mike	Ritchie	18 Tain St, Gourock PA1G 1YQ	01475-392178	House	750
CR62	Mary	Tregear	5 Tarbot Rd, Kildary, Aberdeen AB9 3ST	01224-196720	Flat	600

FIG: LEC1_1.2 FILES USED BY THE SALES DEPARTMENT

Fig : Lec1_1.2 files used by the Sales Department

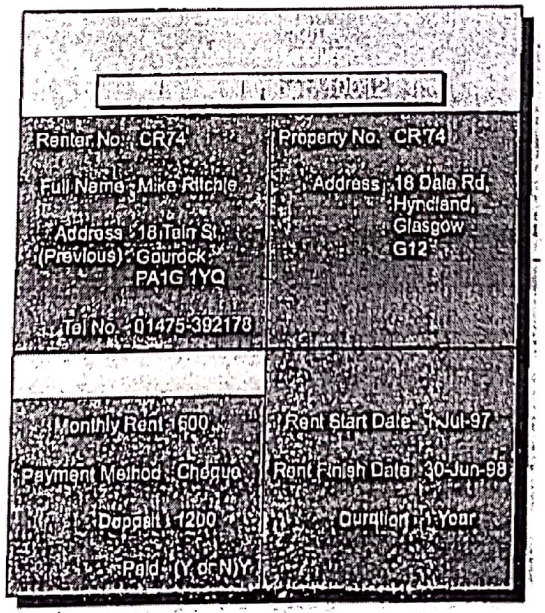


Figure : Lec1_1.3 Lease Details form

LEASE

L.No.	P. No.	R.No.	Rent	Payment	Deposit	Paid	Start	Finish	Duration
10024	PA14	CR62	650	Visa	1300	Y	1-Jun-97	31-May-98	12
10075	Pl94	CR76	400	Cash	800	N	1-Aug-97	31-Jan-98	6
10012	Pg21	CR74	600	Cheque	1200	Y	1-Jul-97	30-Jun-98	12

PROPERTY FOR RENT

P.No.	Street	Area	City	P.Code	Rent
PA14	16 Holhead	Dee	Aberdeen	AB7 5SU	650
PL94	6 Argyll St.	Kilburn	London	NW2	400
PG21	18 Dale Rd	Hyndland	Glasgow	G12	600

OWNER

R.No.	F.Name	L.N	Address	Tel. No.
CR76	John	Kay	56 High St, Putney, London SW1 4EH	0171-774-5632
CR74	Mike	Ritchie	18 Tain St, Gourock PA1G 1YQ	01475-392178
CR62	Mary	Tregear	5 Tarbot Rd, Kildary, Aberdeen AB9 3ST	01224-196720

FIGURE : LEC1_1.4 FILES USED BY CONTRACTS

We can find similar examples in other departments. For example, the Payroll Department store details relating to each employee's salary namely:

Staff-Salary (Staff Number, First Name, Last Name, Address, Sex, Date of Birth.
Salary, National Insurance Number, Branch Number)

The Personnel Department also store staff details, namely:

Staff (Staff Number, Fitst Name, Last Name, Address, Telephone Number, and Position.
Sex, Date of Birth, Salary, National Insurance Number, Branch Number)

It can be seen quite clearly that there is a significant amount of duplication of data in these departments, and this is generally true of file-based systems. Before we discuss the limitations of this approach, it may be useful to understand the terminology used in file based systems. A file is simply a collection of records, which contain logically related data. For example, the Property_for_Rent file in Figure Lec1_1.2 contains six records, one for each property. Each record contains a logically connected set of one or more fields, where each field represents some characteristic of the real-world object that is being modeled in Figure Lec1_1.2, the fields of the Property_for_Rent file represent characteristics of properties, such as address, property type, and number of rooms.

File System Data Management; a look in depth

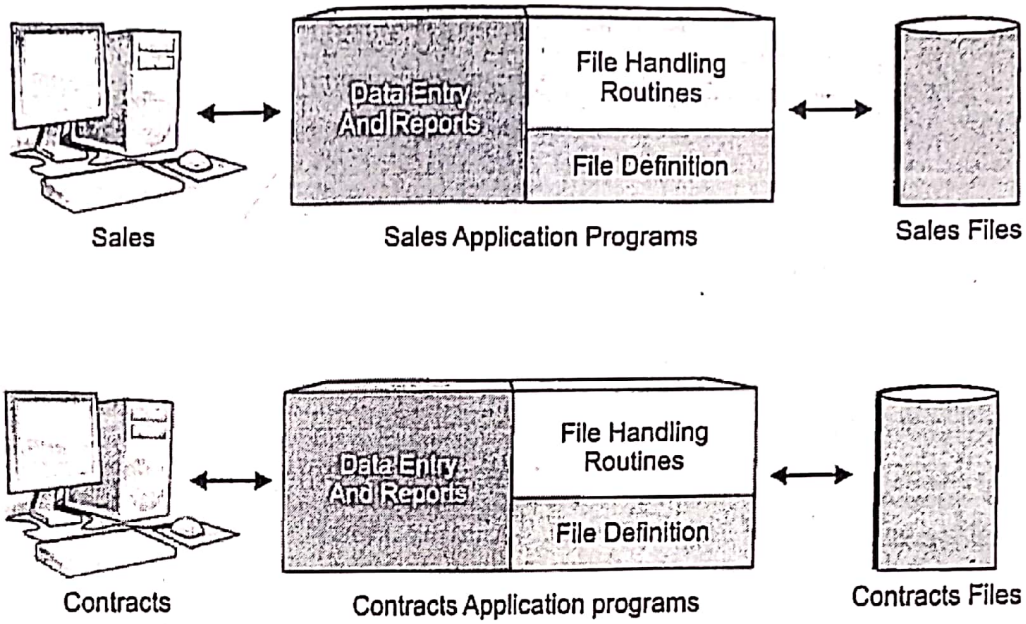


Figure : Lec1_1.5 File based processing.

Even the simplest data-retrieval task required extensive programming in some third Generation programming language (3GL). A 3GL requires the programmer to specify both what must be done and how it is to be done. Examples of 3GL include COBOL, BASIC, FORTRAN, C etc.

Programming in 3GL can be a time consuming, high skill activity. Because the simple as seen or depicted by a person is quite different from the way the computer physically stores the data on disk, the programmer must be familiar with the physical file structures. There fore, every file reference in a program requires the programmer to use complex coding to match the data characteristics and to define the precise access paths to the various file and system components. As file systems become more complex, the access paths become difficult to manage and tend to produce system malfunctions.

The need to write 3GL programs to produce even the simplest reports makes the ad_hoc query impossible. Harried DP specialists or DP managers 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 either serve the purpose or your information needs?

As the number of files in the system expands, the systems' administration becomes difficult too. Each file must have its own file management system, composed of programs that allow the user to

- Create the file structure.
- Add data to the file.
- Delete data from the file.
- Modify the data contained in the file.
- List the file contents.

Even a simple file system of only twenty files requires 5 x 20 = 100 file management programs. If each of the files is accessed by ten different reporting programs, an additional 20 x 10 = 200 programs must be written. Because ad hoc queries are possible, the file reporting programs can multiply quickly. And, because each department in the organization owns its data by creating its own files, the number of files can multiply rapidly.

Planning the file structures carefully is especially important to the DP manager because making changes in an existing structure can be difficult in a file system environment. For example, changing just one field in the original file (lets suppose a CUSTOMER file) requires program that

1. Puts the new file structure into a special portion of memory called the **buffer.**
2. Opens the original file, using a different buffer.
3. Reads a record from the original file.
4. Transform the original data to conform to the new structure, using complex string manipulation functions.
5. Writes the transformed data into the new file structure.

Next, the original file is deleted. Finally, all the programs that use the CUSTOMER file must be modified to fit the revised file structure. In fact, any file structure change, no matter how minor, forces modifications in all programs that use the data as in that file. Modifications are likely to produce errors (bugs), and additional time and effort has to be spent finding the errors in a debugging process. The advice "Think before you do" is especially valid in the file system environment. Because all data access programs are subjected to change when the file structure changes, the file system is said to exhibit **structural dependence.**

As discussed above a change in the files structure, such as the addition or deletion of a field, requires the modification of all programs using that file. Such modifications are required because the file system exhibits the **structural dependence** i.e. access to a file is dependent on its structure. Even the changes in the file data characteristics, such as changing the field from integer to decimal, require changes in all programs that access that file. Because all the data access programs are subjected to change when any of the files' data characteristics change, the file systems is said to exhibit **data dependence.**

The practical significance of the data dependence is dependence is that there exists a difference between the **data logical format** (how the human being views the data) and the **data physical format** (how the computer "sees" the data). Therefore, any program that access a file systems' file must not only tell the computer *what to do?*, but also *how to do?*. Consequently, each program, must contain lines that specify the opening of a specific file type, its' record specification, and its field definitions. Data dependence thus helps to make the file system extremely cumbersome from a programming and data management point of view. *Any change in the data characteristics or the file structures, no matter how minor, requires change s in all the programs that use a modified file.*

Limitations of the File-Based Approach

This brief description of traditional file based system should be sufficient to discuss the limitations of this approach. Five problems are stated below

Limitations of file based systems

- Separation and isolation of data
- Duplication of data
- Data and structural dependence
- Incompatibility of files
- Fixed queries/proliferation of applications

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 potential renters, we first need to create a temporary file of those renters who have 'house' as the preferred type. We then search the Property_for_Rent file for those properties where the property type is 'house' and the rent is less than the renter's maximum rent. With file systems, such processing is difficult. The application programmer 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: .

Duplication of data

Due to the decentralized approach taken by each department, the file-based approach encouraged, if not necessitated, the uncontrolled duplication of data. For example, we can clearly see that there is duplication of both property and renter details in the Sales and Contracts Departments. Uncontrolled duplication of data is undesirable for several reasons:

(a) Duplication is wasteful. It costs time and money to enter the data more than Once. Furthermore; it takes up additional storage space, again with associated costs. Often, the duplication of data can be avoided by sharing data files.

(b) Perhaps more important, duplication can lead to loss of data integrity; in other words, the data is no longer consistent. For example, consider the duplication of data between the Payroll and Personnel Departments listed above. If an employee moves house and the change of address is communicated only to Personnel and not to Payroll, the person's pay slip will be sent to the wrong address. A more serious problem occurs if an employee is promoted to a more senior position with an associated increase in salary. Again, the change is notified to Personnel but the change does not filter through to Payroll. Now, the employee is receiving the wrong salary. When this error is detected, it will take time and effort to resolve it. Both the examples illustrate inconsistencies that may result from the duplication of data. As there is no automatic way for Personnel to update the data in the Payroll files, it is not difficult to foresee such inconsistencies arising. Even if Payroll is notified of the changes, it is possible that the data will be entered incorrectly.

Data dependence

As it has already been mentioned that, the physical structure and storage of the data files and records are defined in the application code. This means that changes to an existing structure are difficult. For example, increasing the size of the Property_for_Rent address field from 40 to 41 characters sounds like a simple change, but it requires the creation of a one-off program (that is, a program that is run only once and can then be discarded) that converts the Property_for_Rent file to the new format. This program has to:

- Open the original, Property_for_Rent file for reading.
- Open a temporary file with the new structure.
- Read a record from the original, convert the data to conform to the new structure, and write it to the temporary file. Repeat u1is step for all records in tile original file.
- Delete the original Property_for_Rent file.
- Rename the temporary file as Property_for_Rent.

In addition, all programs that access the Property_for_Rent file must be modified to conform to the new file structure. There might be many such programs that access the Property_for_Rent file. Thus, the programmer need to identify all the affected programs, modify them, and then retest them. Note that a program does not even have to use the address field to be affected; it has only to use the Property_for_Rent file. Clearly, this could be very time-consuming and subject to error. This characteristic of file-based systems is known as program-data dependence.

Incompatible file formats/platforms

As the structure of files is embedded in the application program, 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 tile Structure of a file

generated by a 'C' program. The direct incompatibility of such files makes them difficult to process jointly.

For example, suppose that the Contracts Department want to find file names and addresses of all owners whose property is currently under lease. Unfortunately, Contracts do not,hold the details of property owners; only the Sales Department hold these. However, Contracts have the Property Number, which can be used to find the corresponding Property Number in the Sales Department's Property_for_Rent file. This file holds, the Owner Number, which can be used to find the owner details, in the Owner file. The Contracts Department program in COBOL and the Sales Department program in 'C'. Therefore, to match Property Numbers in the two Property_for_Rent files requires an application programmer to write software to convert the files, to some common format to facilitate processing; Again, this can be time-consuming and expensive.

To worsen the situations there is also platform dependence, i.e. if one department is running its system on intel based machines with windows as operating system and the other department is using Apple Macintosh with its own Operating system, then the files on one platform will not be used by other platform/department.

Fixed queries/proliferation of application programs

From the end-user's point of view, file-based systems proved to be a great improvement cover manual systems. Consequently, the requirement for new or modified queries grew. However, file-based systems are very dependent upon the application programmer. Any queries or reports that are required have to be written by the application programmer. 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 program that were inadequate or inefficient in meeting the demands of the users, documentation that was limited, and maintenance that was difficult. Often functionality was omitted: 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. In either case, the outcome was not acceptable. Another solution was required.

The History of Database Management Systems

We have already seen that the predecessor to the DBMS was the file-based system. However, there was never a time when the database approach began and the file based system ceased. In fact, the file-based system is still in existence today in specific areas. It has been suggested that the DBMS has its roots in the 1960s Apollo moon-landing project, which was initiated in response to American President J.F. Kennedy's objective of landing a man on the moon by, the end of the decade. At that time, there was no system available that would be able to handle and manage the vast amounts of information that the project would require. As a result, North American Aviation (NAA -now Rockwell International), the prime contractor for the project, developed software known as GUAM (Generalized Update Access Method). GUAM was based on the concept that smaller components come together as parts of larger components, and so on, until the final product is assembled. This structure, which conforms to an upside-down tree, is also known as a hierarchical structure. In the mid-1960s, IBM joined NAA to develop GUAM into what is now known as IMS (Information Management System). The reason why IBM restricted IMS to the management of hierarchies of records was to allow the use of serial storage devices, most notably magnetic tape, which was a market requirement at that time. This restriction was subsequently dropped. Although one of the earliest commercial DBMSs, IMS is still the main hierarchical DBMS used by most large mainframe installations.

In the mid-1960s, another significant development was the emergence of IDS (Integrated Data Store) from General Electric. This work was headed by one of the early pioneers of database systems, Charles Bachmann. This development led to the new type of database system known as the network DBMS which had a profound effect on the information systems of that generation. The network database was developed partly to address the need to represent more complex data, relationships than could be modeled with hierarchical structures, and partly to impose a database standard. To help establish such standards, the Conference on DAta SYstems Languages (CODASYL), comprising representatives of the US Government and the world of business and commerce, formed a List Processing task Force in 1969, subsequently renamed the Data Base Task Group (DBTG) in 1967. The terms of reference for the DBTG were to define standard specifications for an environment that would allow database creation and data manipulation. A draft report was issued in 1969 and the first definitive report in 1971. The DBTG proposal identified three components:

- The network schema -the logical organization of the entire database as seen by the DBA -which includes a definition of the database name, the type of each record, and the components of each record type.
- The subschema -the part of the database as seen by the user or application program.

- A data management language to define the data characteristics and the data structure, and to manipulate the data.

for standardization, the DBTG specified three distinct languages:

- A schema Data Definition Language (DDL), which enables the DBA to define the schema.
- A subschema DDL, which allows the application programs to define the parts of the database they require.
- A Data Manipulation Language (DML), to manipulate the data.

Although the report was not formally adopted by the American National Standards Institute (ANSI), a number of systems were subsequently developed following the DBTG proposal. These systems are now known as CODASYL, or DBTG systems. The CODASYL and hierarchical approaches represented the first generation of DBMSs. However, these two models have some fundamental disadvantages:

- Complex programs have to be written to answer even simple queries based on navigational record-oriented access.
- There is minimal data independence.
- There is no widely accepted theoretical foundation.

In 1970, E. F. Codd of the IBM 'Research Laboratory produced his highly influential paper on the relational data model. This paper was, very timely and addressed the disadvantages of the former approaches. Many experimental relational DBMSs were implemented thereafter with the first commercial products appearing in the late 1970s and early 1980s. Of particular note is the System R project at IBM's San Jose Research Laboratory in California; which was developed during the late 1970s (Astrahan et al., 1976), This project was designed to prove the practicality of the relational model by providing an implementation of its data structures and operations, and led to two major developments:

- The development of a structured query language called SQL, which has since become the standard language for relational DBMSs.
- The production of various commercial relational DBMS products during the 1980s; for example, DB2 and SQL/DS from IBM and ORACLE from ORACLE Corporation.

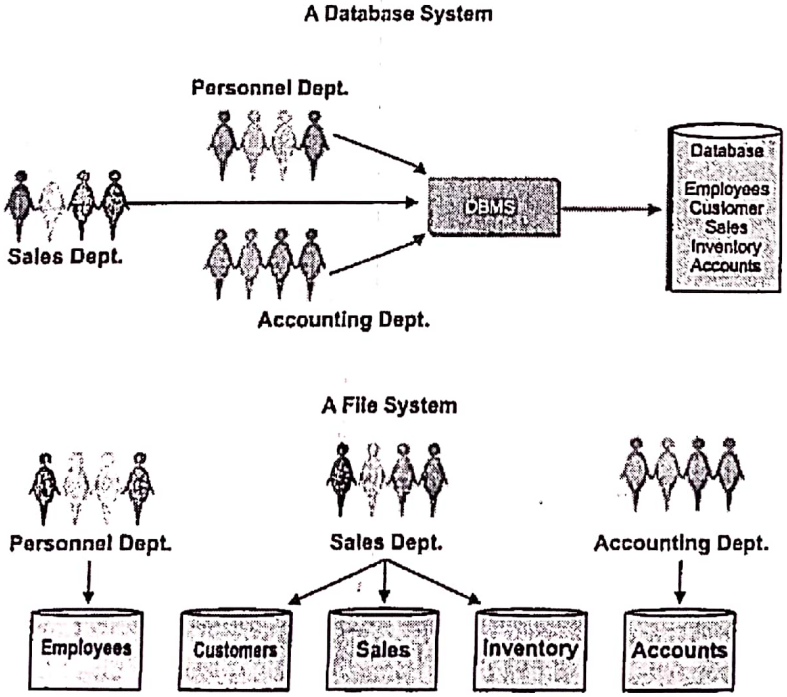
Now there are several hundred relational DBMSs for both mainframe and microcomputer environment, though many are stretching the definition of the relational model. Other examples of multi-user relational DBMSs are CA-OpenIngres from Computer Associates, and Informix from Informix Software Inc. Examples of microcomputer based relational DBMSs are Access and FoxPro from

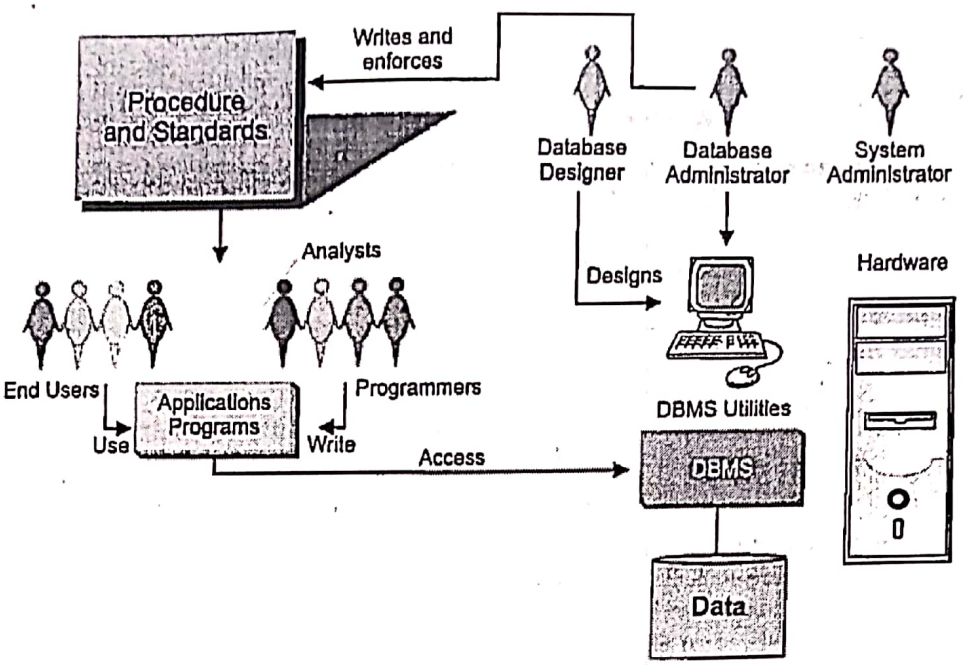
Microsoft, Paradox and Visual dBase from Borland, and R:Base from Microrim. Relational DBMS are referred to as second generation DBMSs.

However, the relational model is not without its failing, and in particular its limited modeling capabilities. There has been much research since then attempting to address this problem. In 1976, Chen presented the Entity-Relationship model, which is now a widely accepted technique for database design. In 1979 Codd himself attempted to address some of the failings in his original work with an extended version of the relational model called RM/T (1979) and more recently RM/V2 (1990). The attempts to provide a data model that represents the "real world" more closely have been loosely classified as semantic data modeling.

In response to the increasing, complexity of database applications, two 'new' systems have emerged: the Object-Oriented DBMS (OODBMS) and the Object Relational DBMS (ORDBMS). However, unlike previous models, the actual composition of these models is not clear. This evolution represents third generation DBMSs.

Database Approach





1.3fg_18Coron,1.4_19

All the above limitations of the file-based approach can be attributed to two factors:

(I) The definition of the data is embedded in the application programs, rather than being stored separately and independently.
(2) There is no control over the access and manipulation of data beyond that imposed by the application programs.

To become more effective, a new approach was required. What emerged were the database and the Database Management System (DBMS).

What Is A Database?

It can be referred to as "persistent data" although it might not actually persist for very long!). By persistent it is meant intuitively, that database data differs in kind from other more ephemeral data, such as input data, output data, control statements, work queues, software control blocks, intermediate results, and more generally any data that is transient in nature. More precisely, we say, that data in the database persists because, once It has been accepted by the DBMS for entry into the database in the first place, it can subsequently be removed from the database only by some explicit request to the DBMS not a mere side effect of (eg.) some program completing execution. This notion of persistence thus allows us to give a slightly more precise definition of the term "database":

Database

A database is a collection of persistent data that is used by the application systems of some given enterprise.

Or

Database is a shared collection of logically related data (and a description of this data referred to as metadata), designed to meet the information needs of an organization.

The term "enterprise" here is simply a convenient generic term for any reasonably self contained commercial, scientific, technical or other organization. An enterprise might be a single individual (with a small personal database), or a complete corporation or similar large body (with a large shared database), or anything in between. Here are some examples:

1. A manufacturing company
2. A bank
3. A hospital
4. A university
5. A government department

Any enterprise must necessarily maintain a lot of data about its operation. Such data is the "persistent data" as referred above. The enterprises just mentioned would typically include the following among their persistent data:

1. Product data
2. Account data
3. Patient data
4. Student data
5. Planning data

Some time the word "operational data" is used in place of "persistent data. But actually that earlier term (operational data) reflected the original emphasis in database systems on operational or production applications i.e., routine, highly repetitive applications that were executed over and over again to support the day-to-day operation of the enterprise (for example, an application to support the deposit or withdrawal of cash in a banking system). The term (OLTP) "online transaction processing" has come to be used to refer to this kind of environment. However, databases are now increasingly being used for other kinds of applications as well i.e., decision support applications and the term "operational data" is thus no longer entirely used to refer to this type of (so called) persistent data. Indeed, enterprises nowadays often maintain two separate databases, one containing operational data and one, often called the data warehouse containing, decision support data. The data warehouse often includes summary- information, (e.g. totals, averages), where that summary information in turn is extracted from the operational database on a periodic basis-say once a day or once a week.

To understand the concept of Database fully, it can be defined as a single, large repository of data, which is defined once and used simultaneously by many departments and users. Instead of disconnected files with redundant data, all data is integrated with a minimum amount of duplication. The database is no longer owned by one department but is now a shared corporate resource. The database holds not only the organization's operational data but, in addition, it holds a description of this data, called Metadata. For this reason, a database is also defined as a *self describing collection of integrated records.* The description of the data is known as the **system catalog** (or data dictionary or meta-data the 'data about data'). It is the self-describing nature of a database that provides **program data independence.**

The approach taken with database systems, whereby we separate the definition of data from the application programs is very similar to the approach taken in modern software development, whereby we provide an internal definition of an object and a separate external definition. The users of an object see only the external definition and are unaware of how the object is defined and how it functions. One advantage of this approach, known as data abstraction, is that we can change the internal definition of an object without affecting the users of the object, provided the external definition remains the same. In the same way, the database approach separates the structure of the data from the application programs and stores it in the database. If new data structures are added or existing structures are modified then the application programs are unaffected, provided they do not directly depend upon what has been modified. For example, if we add a new field to a record or create a new file existing applications are unaffected. However, if we remove a field from a file that an application program uses, then that application program is affected by this change and must be modified accordingly.

The Database Management System (DBMS)

DBMS: A software system that enables users to define, create, and maintain the database and provides controlled access to this database.

The DBMS is the software that interacts with the users' application programs and the database. Typically, a DBMS provides the following facilities:

- 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.

- It allows users to insert, update, delete and retrieve data from the database, usually through a Data Manipulation language (DML). Having a central repository for all data and data descriptions allows the DML to provide a general enquiry facility to this data, called a **query language.** The provision of a query language alleviates the problem with file-based systems where the user has to work with a fixed set of queries or there is a proliferation of programs; giving major, software management problems. There are two types of DML,

procedural and non-procedural, which we can distinguish according to the retrieval operations. The main difference between them is that procedure languages typically manipulate the database record by record, while non-procedural languages operate on sets of record. Consequently, procedural languages specify how the output of a DML statement is to be obtained, while non-procedural, DMLs describe only what data is to be obtained. The most common type of non-procedural language is the Structured Query Language (SQL -pronounced 'S-Q-L' or sometimes 'SeeQuel'), which is now both the standard and the de facto language for relational DBMSs.

- It provides controlled access to the database. For example, it may provide:
  - A security system, which prevents unauthorized users from accessing the database;
  - An integrity system, which maintains the consistency of stored data;
  - A concurrency control system which allows shared access of the database;
  - A recovery control system, which restores the database to a previous consistent state following 3 hardware or software failure.
  - A user accessible catalog, which contains descriptions of the data in the database.

The database approach is based on the file approach. It shows the Sales and Contracts Departments using their application programs to access the database through the DBMS. Each set of departmental application programs handles data entry, data maintenance, and the generation of reports. However, compared with the file-based approach, the physical structure and storage of the data are now managed by the DBMS.

With this functionality, the DBMS is an extremely useful tool. However, as the end-users are not too interested in how complex or easy a task is for the system, it could be argued that the DBMS has made things more complex because they now see more data than they actually need or want. For example, the details that the Contracts Department wants to see for a rental property have changed in the database approach, shown ill. Now the database also holds the property type, the number of rooms, and the owner details. In recognition of this problem, a DBMS provides another facility known as a view mechanism, which allows each user to have his or her own view of the database. The DDL allows views to be defined, where a view is a subset of the database. For example, we could set up a view that allows the Contracts Department to see only the data that they want to see for rental properties.

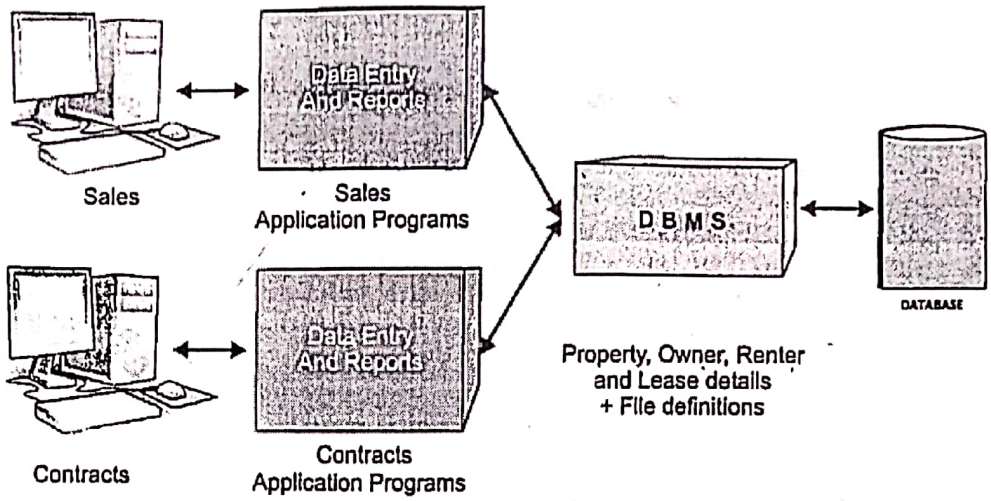


Figure : Lec1_1.6 Database processing.

As well as reducing complexity by letting users see the data in the way they want to see it, views have, several other benefits:

- Views provide a level of security. Views can be set up to exclude data that some user should not see for example, we could create a view that allows a branch manager including the Payroll Department to see all staff data, including) salary details. However, we could create a second view that other staff would use, which includes salary details.

- Views provide a mechanism to customize the appearance of database For example, the Contracts Department may wish to call the Monthly Rent field by the simpler name, Rent.

- A view can present a consistent, unchanging picture of the structure of the database, even if the underlying database is changed (for example, fields added or removed, relationships changed, files split, restructured, or renamed). If fields are added or removed from a file, and these fields are not required by the view, the view is not affected by this change. Thus, a view helps provide the program-data independence

The above discussion is general. The actual level of functionality offered by a, DBMS differs from product to product. For example, a DBMS for a personal computer may not support concurrent shared access, and it may only provide limited security, integrity, and recovery control. However, modern, large multi-user DBMS products offer all the above functions and much more. Modem systems are extremely complex pieces of software consisting of millions of lines of code, with documentation comprising many volumes. This is a result, of having to provide software that handles requirements of a

more general nature. Furthermore, the use of DBMSs nowadays requires a system that provides almost 100% reliability and availability, even in the presence of hardware or software failures. The DBMS is continually evolving and has to be expanded to cope with new user requirements. For example, some applications now require the storage of graphic images, video, sound, and so on. To reach this market, the DBMS must change. It is likely that new functionality will always be required, so that the functionality of the DBMS will never become static.

Components of the DBMS Environment

We can identify five major components in the DBMS environment: hardware, software, data, procedures, and people.

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. A simplified hardware configuration for a Database .

It consists of a network of minicomputers, with a central computer located at Karachi running at the backend of the DBMS that is, the part of the DBMS that manages and controls access to the database. It also shows several computers at various locations running the **front end** of tile DBMS: that is, the part of tile DBMS that interfaces with the user. This is called client-server architecture: the backend is the server and the front end is clients.

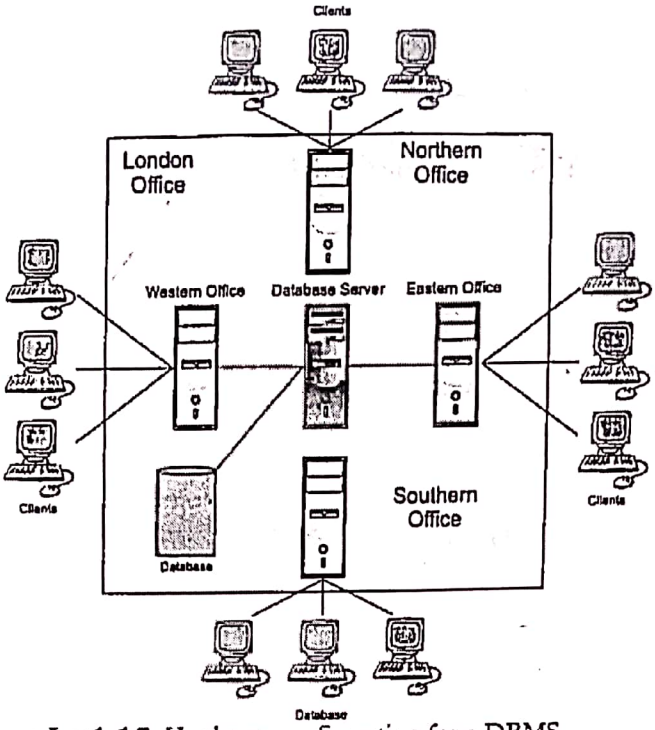


Figure : Lec1_1.7 Hardware configuration for a DBMS.

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, Typically, application programs are written in a third generation programming language, such as 'C', COBOL, Fortran, Ada, or Pascal, or using' a Fourth-generation language, such as SQL embedded in a third-generation language. The target DBMS may have its own fourth-generation tools that allow rapid development of applications through the provision of non-procedural query languages, reports generators, forms generators graphics generators, and application generators. The use of fourth-generation tools can improve productivity significantly and produce programs that are easier to maintain.

Data

Perhaps the most important component of the DBMS environment, certainly from the end-users' point of view, is the data. We observe that the data acts as a bridge between the machine components and the human components. The database contains the operational data and the metadata, the "data about data". The structure of the database is called the schema. The schema consists of four files, or tables, namely: Property_for_Rent, Owner, Renter, and Lease.

The Property_for_Rent table has nine fields, or attributes, namely: Property Number, Street, Area, City, Post Code, Property Type, Number of Rooms, Monthly Rent, and Owner Number. The Owner Number attribute models the relationship between Property_for_Rent and Owner: that is, an owner owns a property for rent. For example, we observe that Owner C046, Joe Keogh, owns property PA14.
The system catalog contains data such as:

- Names, types, and sizes of data items.
- Names of relationships.
- Integrity constraint on the data.
- Names of authorized users who have access to the data.
- What indexes and what storage structures are being used, such as hashing, inverted tiles, or B+ -Tree.

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. This may include procedures on how to identify the failed component, how to fix the failed component (for example, telephone the appropriate hardware engineer) and following the repair of the fault how to recover the database.
- Change the structure of a table, reorganize the database, across multiple disks, improve performance, or archive data to secondary storage.

People
The final component is the people involved with the system, this topic will be discussed in detail in the roles of DB environment.

Database Design -The Paradigm Shift

Until now, we have taken it for granted that there is a structure to the data in the database for example Propelty_for_Rent, Renter, Owner, and Lease. But how did we get this structure? The answer is quite simple: the structure of the database is determined during database **design.** However, carrying out database design can be extremely complex. To produce a system that will satisfy the organizations information needs requires a different approach to that of file-based systems, where the work was driven by the application needs of individual departments. For the database approach to succeed, the organization now has to think of the data first and the application second. This change in approach is referred to as *paradigm shift*. For

the system to be acceptable to the end-users the database design activity is crucial. A poorly designed database will generate errors that may lead to bad decisions being made, which may have serious repercussions for the organization. On the other hand, a well-designed database produces a system that provides the correct information for the decision making process to succeed, in all efficient way.

Unfortunately, database design methodologies are not very popular; most organizations and individual designers rely very little on methodologies for conducting the design of databases, and this is commonly considered a major cause of failure in the development of information systems. Due to the lack of structured approaches to database design, the time or resources required for a database project are typically under estimated, the databases developed are inadequate or inefficient, in meeting the demands of applications, documentation is limited, and maintenance is difficult.

Roles in the Database Environment

In this section, there will detailed description of the fifth component of the DBMS environment i.e. the PEOPLE. We can identify four distinct types of people that participate in the DBMS environment: data and database administrator, database designers, application programmers, and the end-users.

Data and Database Administrators

The database and the DBMS are corporate resources that must be, managed like any other resource. 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 DA consults with and advises senior managers, ensuring that the direction of database development will ultimately support corporate objectives. 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 for the applications and 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. In some organizations there is no distinction between these two roles. In. others, the importance of the corporate resources is, reflected in the allocation of teams of staff dedicated to each of these roles.

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 its business rules. Business rules describe the main characteristics pf the data as viewed by the organization. Examples of business rules are:

- A member of staff cannot handle the sale or rent of more than ten properties at the same time.
- A member of staff cannot handle the sale or rent of his or her own property.
- A solicitor cannot act for both the buyer and seller of a property.

To be effective, the logical database designer must involve all prospective database users in the development of the data model, and the involvement should begin as early in the process as possible. Generally the work of logical Database design is split into two stages:

- Conceptual database design, which is independent of implementation details such as the target DBMS, application programs, programming languages, or any other physical considerations.
- Logical database design, which is targeted at a specific data model, such as relational, network, hierarchical, or object-oriented.

The **physical database** designer takes the logical data model and decides how it is to be physically realized. This involves:

- Mapping the logical data model into a set of tables and integrity constraints.
- Selecting specific storage structures and access methods for the data to achieve good performance for the database activities.
- Debugging any security measures required on the data.

Many parts of physical database design are highly dependent on the target DBMS, and there may be more than one way of implementing a mechanism. Consequently, the physical database designer must be fully aware of the functionality of the target DBMS and must understand the advantages and disadvantages of each alternative for a particular implementation. The physical database designer must be capable of selecting a suitable storage strategy that takes account of usage. Whereas conceptual and logical database design are concerned with the *what* physical database design is Concerned. It requires different skills, which are often found in different people.

Application Programmers

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 programmers. Typically, the application programmers work from a specification produced by systems analysts. Each program contains statements that requests the DBMS to perfom some operation on the database. This includes retrieving data, inserting, updating, and deleting data. The

programs may be written in a third generation programming language or a fourth generation language, as discussed In the previous section.

End-Users

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

- **Naive users** are typically unaware of the DBMS. They access the database through specially written application programs which 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. For example, tile checkout assistant at the local supermarket uses a bar code reader to find out the price of the item. However there is an application program present that reads the bar code, looks up the price of the item in the database, reduces the database field containing the number of such items in stock, and ring up the price on the till.

- **Sophisticated users.** At tile other end of the spectrum 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 users may even write application program for their own use.

Advantages and Disadvantages of DatabaseManagement Systems

The database management system has promising potential advantages. Unfortunately, there are also disadvantages

Advantages of DBMSs

List of Advantages of database management systems

Control of data redundancy
Economy of scale
Data consistency
Balance of conflicting requirements
More information from the same amount of data
Improved data accessibility and responsiveness
Increased productivity
Sharing of data
Improved maintenance through data independence

Improved data integrity
Increased concurrency
Improved security
Improved backup and recovery services
Enforcement of standards

Control of data redundancy
As discussed previously, the traditional file-based systems waste space by storing the same information in more than one file; we stored similar data for properties for rent and renters in both the Sales and Contracts Departments. In contrast, the database approach attempts to eliminate the redundancy by integrated the files so that several 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, when it receives an update to an item that appears more than once it can often do **cascading updates**, automatically updating every occurrence of that item, keeping the data consistent. At other times, it is desirable to duplicate some data items to improve performance.

Data consistency
By eliminating or controlling redundancy, we are reducing the risk of inconsistencies occurring. 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 immediately available 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. Unfortunately, many of today's DBMSs do not automatically ensure this type of consistency.

More information from the same amount of data
With the integration of the operational data, it may be possible for the organization to derive additional information from the same data. For example, in the file-based system, the Contracts Department does not know who owns a leased property. Similarly, the Sales Department has no knowledge of lease details. When we integrate these files together, the Contracts Department has access to owner details and the Sales Department has access to lease details. We may now be able to derive more information from the same amount of data.

Sharing of data
Typically, files are owned by the people or departments that use them. On the other hand, the database belongs to the entire organization and can be shared by all authorized users. In this way, more users share more of the data. Furthermore, new applications can build on the existing data in the database and add only additional data that is not currently stored, rather than having to define, all data requirements again. The new applications can also rely on the functions provided by the DBMS,

such as data definition and manipulation, and concurrency and recovery control, rather than having to provide these functions themselves.

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. For example, an integrity constraint could state that an employee's salary cannot be greater than Rs.40,000 or that the branch number contained in the, employee's record, representing the branch that the employee works at, must correspond to an existing branch office. Again, integration allows the DBA to define, and the DBMS to enforce, integrity constraints.

Improved security

Database security is the protection of the database from unauthorized users. With out 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). For example, the DBA has access to all the data in the database, a branch manager may have access to all data that relates to his or her branch office and a sales assistant may have access to all data relating to properties but no access to sensitive data, such as staff salary details.

Enforcement of standards

Again, integration allows the DBA to define and enforce the necessary standards. These may include departmental, organizational, national, or international standards for such things as data formats to facilitate exchange of data between systems, naming conventions, documentation standards, update procedures, and access rules.

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 their file-based systems can be combined, possibly resulting, in a lower total cost, leading to an economy of scale. The combined budget can he used, to buy a system configuration that is more suited to the organization's needs: This may consist of one large, powerful computer or a network of smaller computers.

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.

Improved data accessibility and responsiveness
Again, as a result of integration, data that crosses departmental boundaries is directly accessible to the end-users. This provides a system with potentially much more functionality that, for example, can be used to provide better services to the end-users or the organization's clients. Many DBMSs provide query languages or report writers that allow users to ask ad hoc questions and obtain the required information almost immediately at their terminals, without requiring a programmer to write some software to extract this information from the database. For example, a branch manager could list its all flats with a monthly rent greater than Rs.400 by entering the following SQL command at a terminal:

```
SELECT * ,
FROM property_for_rent
WHERE type = 'Flat' AND rent> 400;
```

Increased productivity
As mentioned previously, the DBMS provides many of the standard functions that the programmer would normally have to write in a file-based application. At a basic level, the DBMS provides all the low level file-handling routines that are typical in application programs. The provision of these functions allows the programmer to concentrate more on the specific functionality required by the users without having to worry about low-level implementation details. Many DBMSs also provide a forth-generation environment consisting of tools to simplify the development of database applications. This results in increased programmer productivity and reduced development time (with associated cost savings).

Improved maintenance through data independence
In file-based systems, the descriptions of the data and the logic for accessing the data are built into each application program, making the programs dependent on the data. A change to the structure of' the data, for example making an address 41 characters instead of 40 characters, or a change to the way the data is stored on disk, Call require substantial alterations to the programs that are affected by the change, In contrast, a DBMS separates' the data descriptions from the applications, thereby making applications immune to the changes in the data descriptions. This is known

as data independence. The provision of data independence simplifies database application maintenance.

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.

Improved backup and recovery services
Many file-based systems place the responsibility on the user lo 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 lo be reentered. In contrast, modern DBMSs provide, facilities to minimize the amount of processing that can be lost following a failure.

Disadvantages of DBMSs

List of Disadvantage

Complexity
Size
Cost of DBMSs
Additional hardware costs
Cost of conversion
Performance
Higher impact of a failure

Complexity
The provision of the functionality we expect of a good DBMS makes tile DBMS all 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 tile system can lead to bad design decisions, which can have serious consequences for all organization,

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

Cost of DBMSs
The cost of DBMSs varies significantly, depending on the environment and functionality provided. for example, a single-user DBMS for a personal computer

may only cost Rs.10000. However, a large mainframe multi-user DBMS servicing hundreds of users can he extremely expensive, perhaps Rs.1, 00,000 to Rs.500,000. There is also the recurrent annual maintenance cost, which is typically a percentage of the list price.

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.

Cost of conversion
In some situations, the cost of the DBMS and extra hardware may be insignificant compared with the cost of converting existing applications to run on the new DBMS and hardware. 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. The term legacy system is sometimes used to refer to an older, and usually inferior system

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 any more.

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 any component can bring operations to a halt.

DBMS Functions

A DBMS performs several important functions that guarantee the integrity and consistency of the data in the Database. Most of these functions are transparency to end users. And most of these functions can be achieved only through the use of a DBMS. These functions include data dictionary management, data storage management, data transformation and presentation, security management, multi-user access control, backup and recovery management, data integrity management, database access languages and application programming, interfaces, and database communication interfaces.

Data Dictionary Management

The DBMS requires that definitions of the data elements and their relationships (meta data) be stored 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 us from having to code such complex relationships in each program. Additionally, any changes made in a database structures are automatically recorded in the data dictionary, thereby freeing us from having to modify all the programs that access the changed structure. In other words, the DBMS provides data abstraction and it removes structural and data dependency from the system.

Data Storage Management

The DBMS creates the complex structures required for data storage, thus relieving us from the difficult task of defining and programming the physical data characteristics. A modern DBMS system provides storage not only for the data, but also for related data entry forms or screen definitions, reports definitions, data validation rules, procedural code, structures to handle video and picture formats, and so on.

Data Transformation and Presentation

The DBMS transforms entered data to conform to the data structures that are required to store the data. Therefore, the DBMS relieves us of the core of making a distinction between the data logical format and the data physical format. By maintaining data independence, the DBMS translates logical requests into commands that physically retrieved data to make it conform to the users' logical expectations. In other words, a DBMS provides applications programs with software independence and data abstraction.

Security Management

The DBMS creates a security system that enforces user security and data privacy within the database. Security rules determine which users can access the database, which data items each user may access, and which data operations (read, add, delete, or modify) the user may perform. This is specially important in multi-user database systems where many users can access the database simultaneously.

Multi-user Access Control

The DBMS creates the complex structures that allow multi-user access to the data. In order to provide data integrity and data consistency, the DBMS uses sophisticated algorithms to ensure that multiple users can access the database concurrently and still guarantee the integrity of the database.

Backup and Recovery Management

The DBMS provides backup and data recovery procedures to ensure data safety and integrity. Current DBMS system 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 bad sector in the disk, a power failure, and so on. Such capability is critical to the preservation of the databases' integrity.

Data integrity Management

The DBMS promotes and enforces integrity rules to eliminate data integrity problems, thus minimizing data redundancy and maximizing data consistency. The data relationship stored in the data dictionary is used to enforce data integrity. Ensuring data integrity is especially important in transaction-oriented database systems.

Database Access Languages and Application Programming Interfaces

The DBMS provides data access via a query language. A query language is a nonprocedural language; that is one that lets the user specify what must be done without having to specify how it is to be done. The DBMSs' query language contains two components a **Data Definition Language** (DDL) and a **Data Manipulation Language** (DML). The DDL defines the structures in which the data are housed and the DML allows end users to extract the data from the database. The DBMS also provides data access to programmers via procedural (3GL) languages such as COBOL, C, PASCAL etc. the DHBMS also provides administrative utilities used by the DBA and the database designer to create, implement, monitor, and maintain the database.

Database Communication Interfaces

Current generation DBMSs provide special communications routines designed to allow the database to accept end user requests within a computer network environment. In fact, database communications capabilities are an essential feature of the modern DBMS. For example, the DBMS might provide communications functions to access the database through the internet, using internet browsers such as Netscape or Internet Explorer as the front end. In this environment, communications can be accomplished in several ways

- End users can generate answers to queries by filling in screen forms through their preferred World Wide Web browser.
- The DBMS can automatically publish predefined reports on the internet, using a Web format that enables any Web user to browse it.
- The DBMS can connect to third-party systems to distribute information via Email or other productivity applications such as Lotus notes