# Chapter One

# Introduction to Database system

## 1.1 Database - Definition and Usage

Database systems are designed to manage large data set in an organization. The data management involves both *definition and the manipulation* of the data which ranges from simple representation of the data to considerations of structures for the storage of information. The data management also consider the provision of mechanisms for the manipulation of information.

Today, Databases are essential to every business. They are used to maintain internal records, to present data to customers and clients on the World-Wide-Web, and to support many other commercial processes. Databases are likewise found at the core of many modern organizations.

The power of databases comes from a body of knowledge and technology that has developed over several decades and is embodied in specialized software called a database management system, or DBMS. A DBMS is a powerful tool for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely. These systems are among the most complex types of software available.

Thus, for our question: What is a database? In essence a database is nothing more than a collection of shared information that exists over a long period of time, often many years. In common dialect, the term database refers to a collection of data that is managed by a DBMS. Thus the DB course is about:

- How to organize data
- Supporting multiple users
- Efficient and effective data retrieval
- Secured and reliable storage of data
- Maintaining consistent data
- Making information useful for decision making

## 1.2 Data Management Approaches

Data management passes through the different levels of development along with the development in technology and services. These levels could best be described by categorizing the levels into three levels of development. Even though there is an advantage and a problem overcome at each new level, all methods of data handling are in use to some extent. The major three levels are;

1. Manual Approach
2. Traditional File Based Approach
3. Database Approach

There are two major reason to study the alternative data management approaches

- Understanding the problem in those system or approaches, prevents us from repeating similar problem.
- If you want convert these approaches to a database system, understanding how these system work will be extremely useful.

### *1) Manual Approach*

In the manual approach, data storage and retrieval follows the primitive and traditional way of information handling where cards and paper are used for the purpose. The data storage and retrieval will be performed using human labour.

- Files for as many event and objects as the organization has are used to store information.
- Each of the files containing various kinds of information is labelled and stored in one ore more cabinets.
- The cabinets could be kept in safe places for security purpose based on the sensitivity of the information contained in it.
- Insertion and retrieval is done by searching first for the right cabinet then for the right the file then the information.
- One could have an indexing system to facilitate access to the data

***Limitations of the Manual approach***

- Prone to error
- Difficult to update, retrieve, integrate
- You have the data but it is difficult to compile the information
- Limited to small size information
- Cross referencing is difficult

An alternative approach of data handling is a computerized way of dealing with the information. The computerized approach could also be either decentralized or centralized base on where the data resides in the system.

### *2) Traditional File Based Approach*

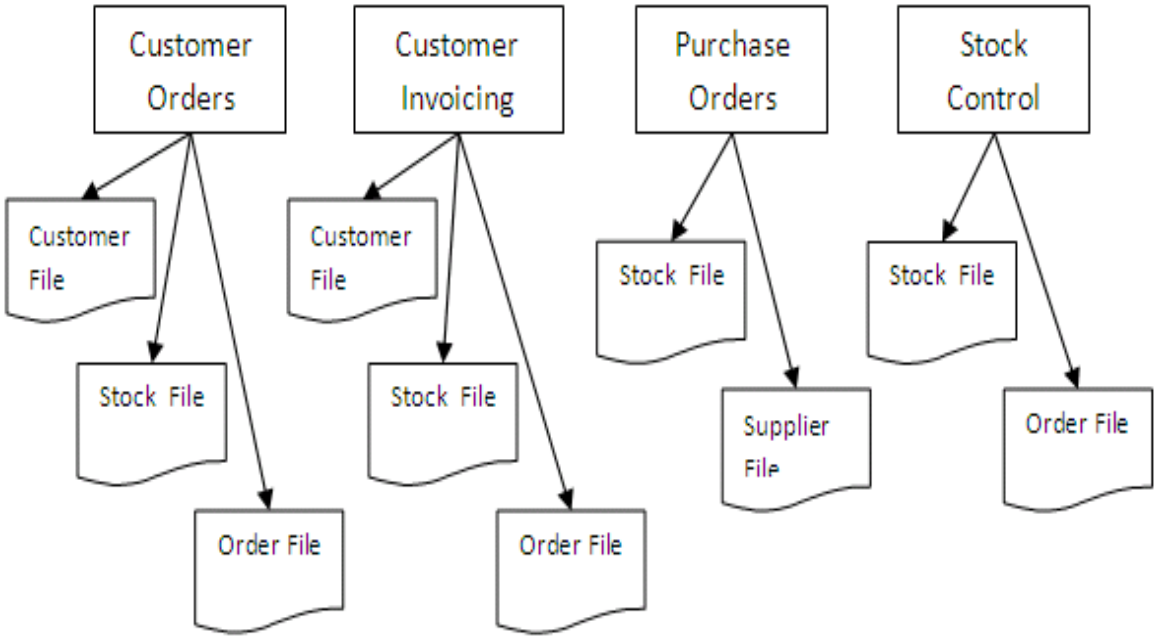
After the introduction of Computer for data processing to the business community, the need to use the device for data storage and processing increase. There were, and still are, several computer applications with file based processing used for the purpose of data handling. Even though the approach evolved over time, the basic structure is still similar if not identical.

The term File-Based approach refers to the situation where data is stored in one or more separate computer files. Typically for example, the details of customers may be stored in one file, orders

in another etc. Computer programs to perform the various tasks required by the business process data that is stored in computer files. Each program, or sometimes a related set of programs, are called computer applications.

For example, all of the programs associated with processing customer's orders are referred to as the order processing application. The file-based approach might have application programs that deal with purchase orders, invoices, sales and marketing, suppliers, customers, employees, and so on. We can imagine that some of these different programs might use the same data. If the data is kept in different files, there could be problems when an item of data needs updating, as it will need to be updated in all the relevant files; if this is not done, the data will be inconsistent, and this could lead to errors. The problem could be made even worse if different items of data are changed in different departments, so that the invoice application uses a different address from the sales mailing list program for the same customer.

The following diagram shows how different applications will each have their own copy of the files they need in order to carry out the activities for which they are responsible.

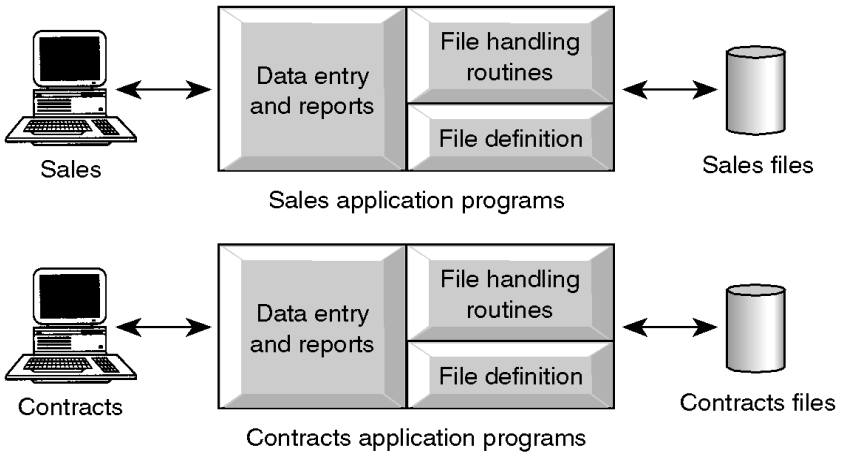



For example, one user, the grade reporting office, may keep a file on students and their grades. Programs to print a transcript and to enter new grades into the file are implemented as part of the application. A second user, the accounting office, may keep track of students fees and their payments. Although both are interested in the data about students, each user maintains separate files and programs to manipulate the files because each requires some data not available from the other users files.

**Summary**

- File based systems were an early attempt to computerize the manual filing system.
- This approach is the decentralized computerized data handling method.

- A collection of application programs that perform services for the end-users. In such systems, every application program that provides service to end users define and manage its own data
- Such systems have number of programs for each of the different applications in the organization.
- Since every application defines and manages its own data, the system is subjected to serious data duplication problem.
- File, in traditional file based approach, is a collection of records which contains logically related data.




*Sales Files*

**Property_for_Rent**(Property Number, Street, Area, City, Post Code, Property Type, Number of Rooms, Monthly Rent, Owner Number)

**Owner**(Owner Number, First Name, Last Name, Address, Telephone Number)

**Renter**(Renter Number, First Name, Last Name, Address, Telephone Number, Preferred Type, Maximum Rent)

*Contracts Files*

**Lease**(Lease Number, Property Number, Renter Number, Monthly Rent, Payment Method, Deposit, Paid, Rent Start Date, Rent Finish Date, Duration)

**Property_for_Rent**(Property Number, Street, Area, City, Post Code, Monthly Rent)

**Renter**(Renter Number, First Name, Last Name, Address, Telephone Number)

## Limitations of the Traditional File Based approach

As business application become more complex demanding more flexible and reliable data handling methods, the shortcomings of the file based system became evident. These shortcomings include, but not limited to:

***1) Separation and isolation of data:-*** When data is isolated in separate files, it is more difficult for us to access data that should be available. Available information in one application may not be known. The application programmer is required to synchronize the processing of two or more files manually to ensure the correct data is extracted.

***2) Duplication of data:-*** When employing the decentralized file-based approach, the uncontrolled duplication of data is occurred. Uncontrolled duplication of data is undesirable because duplication is wasteful (*money and time cost*) and can lead to loss of data integrity

***3) Data dependence on the application:-*** Using file-based system, the physical structure and storage of the data files and records are defined in the application program code. This characteristic is known as program-data dependence. Making changes to an existing data structure are rather difficult and will lead to a modification of program. Such maintenance activities are time-consuming and subject to error.

***4) Incompatible file formats or data structures*** (e.g. "C" and COBOL)**:-** The structures of the file are dependent on the application programming language. However file structure provided in one programming language such as direct file, indexed-sequential file which is available in COBOL programming, may be different from the structure generated by other programming language such as C. The direct incompatibility makes them difficult to process jointly.

***5) Fixed queries / proliferation of application programs:-*** File-based systems are very dependent upon the application programmer. Any required queries or reports have to be written by the application programmer. Normally, a fixed format query or report can only be entertained and no facility for ad-hoc queries if offered. Generally file based approach support fixed query processing which is defined during application development

***6) Update anomalies:-*** The most significant problem experienced by the traditional file based approach of data handling is the "update anomalies". We have three types of update anomalies;

1. **Modification Anomalies**: a problem experienced when one ore more data value is modified on one application program but not on others containing the same data set.
2. **Deletion Anomalies**: a problem encountered where one record set is deleted from one application but remain untouched in other application programs.
3. **Insertion Anomalies**: a problem experienced whenever there is new data item to be recorded, and the recording is not made in all the applications. And when same data item is inserted at different applications, there could be errors in encoding which makes the new data item to be considered as a totally different object.

***7) Limited data sharing:-*** Every application maintains its own data.

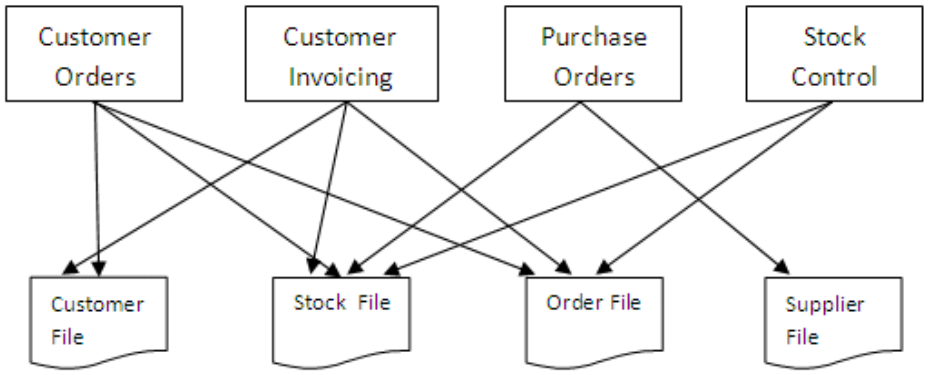
***8) Less security measure:-*** Difficulty of protecting data from unauthorized access and providing different level of access privilege.

The limitations for the traditional file based data handling approach arise from two basic reasons.

i. Definition of the data is embedded in the application program which makes it difficult to modify the database definition easily.
ii. No control over the access and manipulation of the data beyond that imposed by the application programs.

## The Shared File Approach

One approach to solving the problem of each application having its own set of files is to share files between different applications. This will alleviate the problem of inconsistent data between different applications, and is illustrated in the diagram below.




The introduction of shared files solves the problem of inconsistent data across different versions of the same file held by different departments, but other problems may emerge, including:

- When each department had its own version of a file for processing, each department could ensure that the structure of the file suited their specific application. If departments have to share files, the file structure that suits one department might not suit another, for example, data might need to be sorted in a different sequence for different applications (for instance, customer details could be stored in alphabetical order, or numerical order, or ascending or descending order of customer number).
- Some applications may require access to more data than others, for instance a credit control application will need access to customer credit limit information, whereas an delivery note printing application will only need access to customer name and address details. The file will still need to contain the additional information to support the application that requires it.
- If the structure of the data file needs to be changed in some way (for example, to reflect a change in currency), this alteration will need to be reflected in all application programs that use that data file. This problem is known as physical data dependence, and will be examined in more detail later in the unit.
- While a data file is being processed by one application, the file will not be available for other applications or for ad hoc queries. This is because, if more than one application is allowed to alter data in a file at one time, serious problems can arise in ensuring that the updates made by each application do not clash with one another. This issue of ensuring consistent, concurrent updating of information is an extremely important one, and is dealt with in detail for database systems in the unit on concurrency control. File-Based systems avoid these problems by not allowing more than one application to access a file at one time.

## *3) Database Approach*

A database is a collection of information that is organized so that it can easily be accessed, managed, and updated. The database approach emphasizes the integration and sharing of data throughout the organization. Thus in Database Approach logically related collection of data are stored in database.

Database is just a computerized record keeping system or a kind of electronic filing cabinet. It can viewed as a *repository* for collection of computerized data files.

Database is a *shared collection of logically related data designed to meet the information needs of an organization.* Since it is a shared corporate resource, the database is integrated with minimum amount of or no duplication. These logically related data *comprises entities, attributes, relationships, and business rules* of an organization's information. In addition to containing data required by an organization, database also contains a description of the data which called as "Metadata" or "Data Dictionary" or "Systems Catalogue" or "Data about Data". Since a database contains information about the data (metadata), it is called a self *descriptive collection* on integrated records.

The purpose of a database is to store information and to allow users to retrieve and update that information on demand. Database is designed once and used simultaneously by many users.

Unlike the traditional file based approach in database approach there is program data independence. That is the separation of the data definition from the application. Thus the application is not affected by changes made in the data structure and file organization. Each database application will perform the combination of: Creating database, Reading, Updating and Deleting data.

### Characteristics of the Database Approach

With the database approach, a single repository of data is maintained that is defined once and is then accesses by various users. The main characteristics of the database approach are as follows:

- Self describing nature of the database system
- Insulation between programs and data and data abstraction
- Support of multiple views of the data
- Sharing of data and multi-user transaction processing.

#### (a) Self-Describing Nature of a Database System

A fundamental characteristic of the database approach is that the database system contains not only the database itself, but a complete definition or description of the database structure and constraints. The definition is stored in the DBMS catalog which contains information such as the structure of each file, the type and storage format of each data item, and various constraints on the data. The information stored in the catalog is called *metadata*, and it describes the structure of the primary database. The catalog is used by the DBMS software and also by database users who need information about the database structure.

A general purpose database application is not written for a specific database application, so it must refer to the catalog to know the structure of the files in a specific database, such as the type and format of data it will access. *The DBMS software must work equally well with any number of database applications.*

In traditional file processing, the data definition is part of the application programs themselves, hence they are constrained to work with only a specific database (i.e. a banking database, or a university database). For example, an application written in C++ may have a struct or class declaration. File processing software can access only specific databases; DBMS software can access diverse databases by extracting the database definitions from the catalog, and using these definitions.

**Example:** In the example database, the catalog will store the definitions of all files shown. The database designer prior to creating the actual database specifies the definitions. Whenever a request is made to access for example the name of a student record, the DBMS software refers to the catalog to determine the structure of the student file and the position and size of the name data item within a student record. In traditional file processing, the size and file structure are coded within the application that accesses the data item.

**(b) Insulation between Programs and Data, and Data Abstraction**

In traditional file processing, the structure of the data files is embedded in the application programs so any changes to the structure of a file may require changing all programs that access the file. For example, a file access program may be written so that it can only access student records with the exact structure specified by the starting position and length. If we add another piece of data to the student record, such as BirthDate, the program must then be changed.

By contrast, DBMS access programs in most cases do not require such changes. The structure of data files is stored in the catalog separately from the access programs. This is called program-data independence. So that on previous example, in a DBMS environment, we just need to change the description of a student record in the catalog to reflect the inclusion of the new data item BirthDate, and no programs need to be changed. The next time a program refers to the catalog, the new structure of Student records will be accessed and used.

In some types of database systems, such as Object Oriented database systems users can define operations on data as part of the database. An operation is specified in two parts. The interface of an operation includes the operation name and the data types of its arguments. The implementation of the operation is specified separately and can be changed without affecting the interface.

User application programs can operate on the data by invoking the operations through their names and arguments regardless of how the operations are implemented. This can be called program-operation independence.

This is called data abstraction. A DBMS provides users with the conceptual representation of data that does not include details of how the data is stored or how the operations are

implemented. A data model is a type of data abstraction that is used to provide the conceptual representation.

Example: The internal implementation of a file may be defined by its record length (the number of characters in each record), and each data item may be specified by its starting byte within a record and its length in bytes. But a typical database user is not concerned with the location of each data item within a record or its length. The DBMS hides details of the file storage organization from the user.

**(c) Support of Multiple Views of the Data**

A database typically has many users, each of whom may require a different perspective or view of the database. A view may be a subset of the database, or it may contain virtual data that is derived from the database files, but not explicitly stored. (Give examples of these types of data). Users generally do not need to know if the data is stored or derived.

**(d) Sharing of Data and Multi-User Transaction Processing**

A multi-user DBMS must allow multiple users to access the database at the same time. This is important if data for multiple applications is to be integrated (brought together) and maintained in a single database.

The DBMS must include concurrency control software to ensure that several users trying to update the same data do so in a controlled manor. For example, when several reservation clerks try to assign a seat on a flight, the DBMS should ensure that each seat can be accesses by only one clerk.

An important concept in database applications is a transaction which an execution program or process that includes one or more database accesses, such as reading or updating of database records. Each transaction is supposed to execute a logically correct database access if executed in its entirety without interference from other transactions. The isolation property ensures that each transaction appears to execute in isolation from other transactions, even though hundreds of transactions may be executing concurrently.

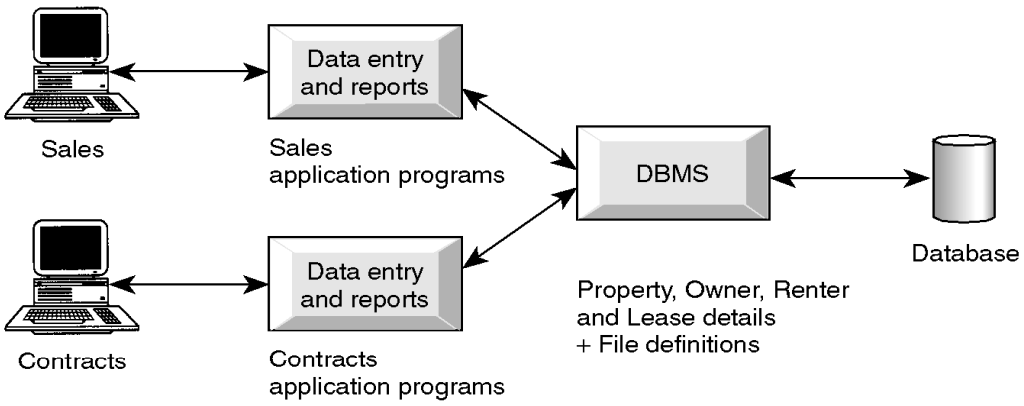
## Benefits of the database approach

- *Data can be shared*: two or more users can access and use the same data instead of storing data in redundant manner for each user. i.e. Database belongs to the entire organization and can be shared by all authorized users.
- *Integrity can be maintained*: data at different applications will be integrated together with additional constraints to facilitate shared data resource. Database integrity provides 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.
- *Improved data accessibility and responsiveness*: By having an integration in the database approach, data accessing can be crossed departmental boundaries. This feature provides

more functionality and better services to the users. So that by using structured query languages, the users can easily access data without programming experience.

- *Control of data redundancy*: isolated data is integrated in database to decrease the redundant data stored at different applications. Although the database approach does not eliminate redundancy entirely, it controls the amount of redundancy inherent in the database.
- *Inconsistency can be avoided*: by eliminating or controlling data redundancy, the database approach reduces the risk of inconsistencies occurring of the data in the database to some extent. It will avoid inconsistency and ensures all copies of the data are kept consistent.
- *Security majors can be enforced*: the shared data can be secured by having different levels of clearance and other data security mechanisms. Database approach provides a protection of the data from the unauthorized users. It may take the term of user names and passwords to identify user type and their access right in the operation including retrieval, insertion, updating and deletion
- *Standards can be enforced*: The integration of the database enforces the necessary standards including data formats, naming conventions, documentation standards, update procedures and access rules. Therefore the different ways of using and dealing with data by different unite of an organization can be balanced and standardized by using database approach.
- *Improved backing and recovery services:-* Modern database management system provides facilities to minimize the amount of processing that can be lost following a failure by using the transaction approach.
- *Improved maintenance:-* Database approach provides a data independence. As a change of data structure in the database will be affect the application program, it simplifies database application maintenance.
- *Physical data dependence is resolved;* this means that the underlying structure of a data file can be changed without the application programs needing amendment. This is achieved by a hierarchy of levels of data specification. Each such specification of data in a database system is called a schema. The different levels of schema provided in database systems are described and further details of what is included within each specific schema are discussed later on.
- *Transaction support can be provided*: basic demands of any transaction support systems are implanted in a full scale DBMS.
- *Quality data can be maintained to Improved decision support*: the different integrity constraints in the database approach will maintain the quality of data leading to better and improved decision making.

- *Increased productivity:-* The database approach provides all the low-level file-handling routines. The provision of these functions allows the programmer to concentrate more on the specific functionality required by the users. The fourth-generation environment provided by the database can simplify the database application development.
- *Compactness*: since it is an electronic data handling method, the data is stored compactly (no voluminous papers).
- *Speed*: data storage and retrieval is fast as it will be using the modern fast computer systems.
- *Less labour*: unlike the other data handling methods, data maintenance will not demand much resource.
- *Centralized information control:* since relevant data in the organization will be stored at one repository, it can be controlled and managed at the central level.
- *Increased concurrency:-* Database can manage concurrent data access effectively. It ensures no interference between users that would not result any loss of information nor loss of integrity.
- *Balance of conflicting requirements:-* By having a structural design in the database, the conflicts between users or departments can be resolved. Decisions will be based on the base use of resources for the organization as a whole rather that for an individual entity.
- *More information from the same amount of data:-* With the integration of the operated data in the database approach, it may be possible to derive additional information for the same data.




**Property_for_Rent**(Property Number, Street, Area, City, Post Code, Property Type, Number of Rooms, Monthly Rent, Owner Number)

**Owner**(Owner Number, First Name, Last Name, Address, Telephone Number)

**Renter**(Renter Number, First Name, Last Name, Address, Telephone Number), Preferred Type, Maximum Rent)

**Lease**(Lease Number, Property Number, Renter Number, Payment Method, Deposit, Paid, Rent Start Date, Rent Finish Date)

**Limitations and risk of Database Approach**

In split of a large number of advantages can be found in the database approach, it is not without any challenge. The following disadvantages can be found including:

- ***Complexity in designing and managing data:-*** Database management system is an extremely complex piece of software. All parties must be familiar with its functionality and take full advantage of it. The *changes introduced by the adoption of a database* system must be properly managed to *ensure that they help advance the company's objectives.* Given the fact that database systems hold crucial company data that are accessed from multiple sources, security issues must be assessed constantly. Therefore, training for the administrators, designers and users is required.
- **Size:-** The database management system consumes a substantial amount of main memory as well as a large number amount of disk space in order to make it run efficiently.
- **Cost of DBMS:-** A multi-user database management system require sophisticated *hardware and software and highly skilled personnel.* Therefore it may need higher cost to be incurred to develop and maintain the system. Even after the installation, there is a high recurrent annual maintenance cost on the software. As a general the *cost of maintaining* the *hardware, software, and personnel required to operate and manage a database system can be substantial*. *Training, licensing, and regulation compliance costs* are often overlooked when database systems are implemented.
- **Risk and Cost of conversion:-** When moving from a file-base system to a database system, the company is required to have additional expenses on hardware acquisition and training cost.
- **Reduced Performance (**due to centralization and data independency**):-** As the database approach is to cater for many applications rather than exclusively for a particular one, some applications may not run as fast as before.
- **Higher impact of a failure:-** The database approach increases the vulnerability of the system due to the centralization. As all users and applications reply on the database availability, the failure of any component can bring operations to a halt and affect the services to the customer seriously. This shows that high impact on the system when failure occurs to the central system.
- **Vendor Dependence:-** Given the heavy *investment in technology and personnel training,* companies might be reluctant to change database vendors.
- **Frequent Upgrade/Replacement Cycles:-** DBMS vendors frequently *upgrade their products by adding new functionality.* Such new features often come bundled in new upgrade versions of the software. Some of these versions require *hardware upgrades*. Not only do the *upgrades themselves cost money*, but it also *costs money to train database users and administrators* to properly use and manage the new features.
- Complex backup and recovery services from the users perspective

## 1.3 Database Management System (DBMS)

Database Management System (DBMS) is a general-purpose Software packages (a collection of programs) used for providing EFFICIENT, CONVENIENT and SAFE MULTI-USER (many people/programs accessing same database, or even same data, simultaneously) storage of and access to MASSIVE amounts of PERSISTENT (data outlives programs that operate on it) data.

A DBMS also provides a systematic method for define/create, construct, manipulate, maintain, storing, retrieving data in a database and share databases among various users and applications. It also provides the service of controlling data access, enforcing data integrity, managing concurrency control, and recovery. Having this in mind, a full scale DBMS performs several important functions that guarantee the ***integrity and consistency*** of the data in the database. Most of those functions are ***transparent to end users***, and most can be achieved only through the use of a DBMS. This includes:

- ***Data dictionary management:-*** The DBMS stores definitions of the data elements and their relationships (metadata) in a data dictionary. So that it define and specify the *data types, structures and constraints* for the data to be stored in the database. 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. Additionally, any changes made in a database structure are automatically recorded in the data dictionary. In other words, the DBMS provides data abstraction, and it removes structural and data dependence from the system.
- ***Data storage management:-*** DBMS handles Construction of database, the process of storing the data itself on some storage medium. The DBMS *creates and manages the complex structures required for data storage.* A modern DBMS provides storage not only for the data, but also for related data entry forms or screen definitions, report definitions, data validation rules, procedural code, structures to handle video and picture formats, and so on. *Data storage management* is also important for database performance tuning. *Performance tuning* relates to the activities that make the database perform more efficiently in terms of storage and access speed. Although the user sees the database as a single data storage unit, the DBMS actually stores the data files may even be stored on different storage media. Therefore, the DBMS doesn't have to wait for one disk request to finish before the next one starts. In other words, the DBMS can fulfill database requests concurrently.
- ***Data transformation and presentation (****Manipulating):-* includes such functions as querying the database to retrieve specific data, updating the database to reflect changes in the mini world, and generating reports from the data.
- ***Security management:-*** DBMS support the implementation of access and authorization service to database administrator and users. Also provide *protection services which* includes system protection against software/hardware malfunction and security protection against unauthorized/malicious access.

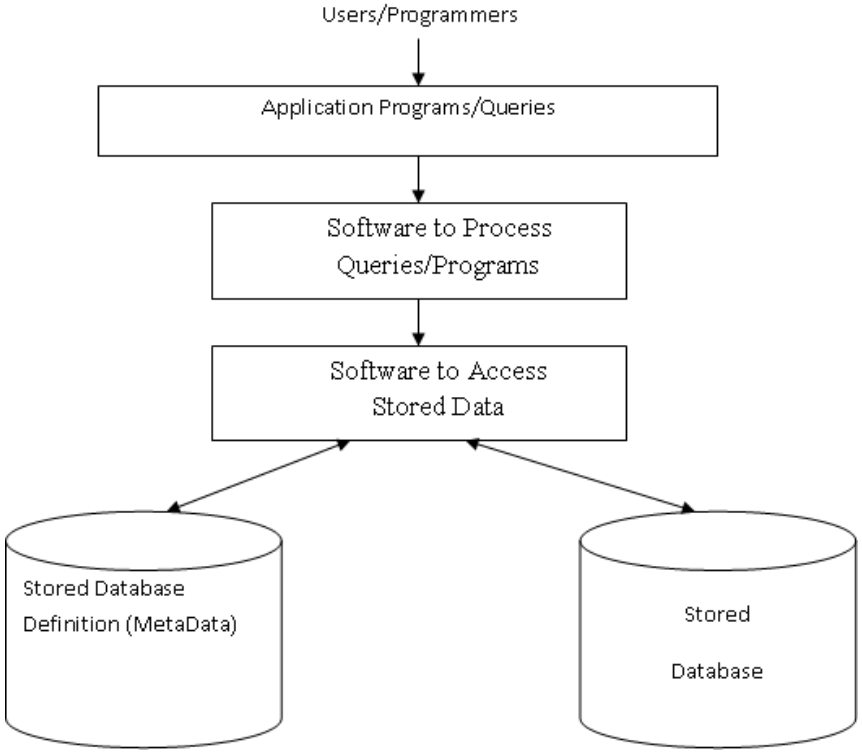
- ***Multi-user access control (Sharing:-*** allows multiple users and programs to access the database concurrently. To provide *data integrity and data consistency*, the DBMS uses sophisticated algorithms to ensure that *multiple users* can *access the database concurrently* without compromising the *integrity of the database.*
- ***Backup and recovery management:-*** The DBMS provides backup and data recovery to ensure data safety and integrity. Current DBMS systems provide special utilities that allow the DBA to perform routine and special backup and restore procedures. Recovery Management deals with the recovery of the database after a failure, such as bad sector in the disk or power failure. Such capability is critical to preserving the database's integrity.
- ***Data integrity management:-*** Integrity Services rules about data and the change that took place on the data, correctness and consistency of stored data, and quality of data based on business constraints. The DBMS promotes and enforces integrity rules, thus minimizing data redundancy and maximizing data consistency. The data relationships stored in the data dictionary are used to enforce data integrity. Ensuring data integrity is especially important in transaction-oriented database systems.
- ***Database access language:-*** The DBMS provides data access through a *query language.* A *query language* is a nonprocedural language, one that lets the *user specify what must be done without having to specify how it is to be done.* Structured Query Language (SQL) is one of the data access standard supported by the majority of DBMS vendors.
- ***Application program interface management*:-** DBMS provides application programming interfaces to procedural languages such as COBOL, C, Java, Visual Basic.NET, and C#.
- ***Utility services*:** The DBMS provides *administrative utilities* used by the DBA and the database designer to create, implement, monitor, and maintain the database. These utilities include like *importing data, statistical analysis support, index reorganization, garbage collection*
- ***Database Communication Interfaces:-*** Current-generation DBMSs accept end-user requests via multiple, different network environments. For example, the DBMS might provide access to the database via the Internet through the use of Web browsers such as Mozilla Firefox or Microsoft Internet Explorer. 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 Web browser. The DBMS can automatically publish predefined reports on a Website. The DBMS can connect to third-party systems to distribute information via e-mail or other productivity applications.
- ***Concurrency Control Services***: access and update on the database by different users simultaneously should be implemented correctly.
- Services to promote *data independency* between the data and the application

Databases can be implemented by general purpose DMBSs, or special purpose custom software can be written to create and maintain a database.

***Database System =====> Database** and **DBMS software** together.*

## Components of DBMS Environment

As discussed above, DBMS is software package used to design, manage, and maintain databases. Each DBMS should have facilities to define the database, manipulate the content of the database and control the database. These facilities will help the designer, the user as well as the database administrator to discharge their responsibility in designing, using and managing the database. So that DBMS provides the following facilities and referred as components of a DBMS.




### DBMS Engine

The engine is the central component of a DBMS. This component provides access to the database and coordinates all of the functional elements of the DBMS. An important source of data for the DBMS engine, and the database system as a whole, is known as metadata. Metadata means data about data. Metadata is contained in a part of the DBMS called the data dictionary (described below), and is a key source of information to guide the processes of the DBMS engine. The DBMS engine receives logical requests for data (and metadata) from human users and from applications, determines the secondary storage location (i.e. the disk address of the requested data), and issues physical input/output requests to the computer operating system. The data requested is fetched from physical storage into computer main memory; it is contained therein special data structures provided by the DBMS. Whilst the data remains in memory, it is managed by the DBMS engine. Additional data structures are created by the database system itself, or by users of the system, in order to provide rapid access to data being processed by the system. These data structures include indexes to speed up access to the data, buffer areas into which particular types of data are retrieved, lists of free space etc. The management of these additional data structures is also carried out by the DBMS engine.

### User Interface Subsystem

The interface subsystem provides facilities for users and applications to access the various components of the DBMS. Most DBMS products provide a range of languages and other interfaces, since the system will be used both by programmers (or other technical persons) and by users with little or no programming experience. Some of the typical interfaces to a DBMS are the following:

***1. Data Definition Language (DDL):***

- Language used to define each data element required by the organization. It used by the DBA and database designers to specify the *conceptual schema* of a database. In many DBMSs, it also used to define internal and external schemas (views).
- In some DBMSs, separate storage definition language (SDL) and view definition language (VDL) are used to define internal and external schemas.
- DDL allows DBA or user to describe and name entitles, attributes and relationships required for the application. Specification notation for defining the database schema
- Generally it is a commands for setting up schema or the intension of database. These commands are used to setup a database, create, delete and alter table with the facility of handling constraints. Also used to define, modify or remove database structures such as records, tables, files, and views

***2. Data Manipulation Language (DML):***

- Language for accessing and manipulating the data organized by appropriate data model.
- Is a core command used by end-users and programmers to store, retrieve, and access the data in the database.
- Used to specify database retrievals and updates.
- Since the required data or Query by the user will be extracted using this type of language, it is also called "Query Language"
- DML commands (data sublanguage) can be *embedded* in a general-purpose programming language (host language), such as COBOL, C or an Assembly Language. Alternatively, *stand-alone* DML commands can be applied directly (query language).
- There are two type of DML, namely *procedural DML (*user specifies what data is required and how to get the data*)* and *non procedural DML (*user specifies what data is required but not how it is to be retrieved). SQL is the most widely used non-procedural language query language

***3. Data Dictionary:***

- Due to the fact that a database is a self describing system, this tool, Data Dictionary, is used to store and organize information about the data stored in the database.

***4. Data Control Language:***

- Database is shared resource that demands control of data access and usage. The database administrator should have the facility to control the overall operation of the system.
- Data Control Languages are commands that will help the Database Administrator to control access to the database. It allows a Database administrator to have overall control of the system, often including the administration of security, so that access to both the data and processes of the database system can be controlled.
- It used to define the security on the data in the database. The commands include grant or revoke privileges to access the database or particular object within the database and to store or remove database transactions

***5. A graphical user interface***:

- Provide a visual means of browsing or querying the data, including a range of different display options, such as bar charts, pie charts etc. One particular example of such a system is Query-by-Example, in which the system displays a skeleton table (or tables), and users pose requests by suitable entry in the table.
- A forms user interface in which a screen-oriented form is presented to the user who responds by filling in blanks in the form. Such forms based systems are a popular means of providing a visual front-end both to developers and to users of a database system. Typically developers use the Forms-based system in "developer mode", where they design the forms or screens that will make up an application, and attach fragments of code which will be triggered by the actions of users as they use the forms-based user interface.
- User-friendly interfaces:
    - *Menu-based, popular for browsing on the web*
    - *Forms-based, designed for native users*
    - *Graphics-based (Point and Click, Drag and Drop etc.)*
    - *Natural language: requests in written English*
    - *Speech as Input (?) and Output*
    - *Setting system parameters*
    - *Changing schemas or access path*
    - *Creating accounts, granting authorizations*
    - *Parametric interfaces (e.g., bank tellers) using function keys.*
    - *Combinations of the above*
- A natural language user interface that allows users to present requests in free form English statements.

- A DBMS procedural programming language, often based on standard third-generation programming languages such as C and COBOL which allows programmers to develop sophisticated applications.
- 4th Generation Languages, such as Smalltalk, JavaScript, etc. permit applications to be developed relatively quickly compared to the procedural languages mentioned above.
- Fourth Generation Language (4GL)
    - *Query Languages*
    - *Forms Generators*
    - *Report Generators*
    - *Graphics Generator*
    - *Application Generator*

6. ***Database System Utilities:-*** Used to perform certain functions such as:

- *Loading* data stored in files into a database. Includes data conversion tools.
- *Backing up* the database periodically on tape.
- *Reorganizing* database file structures.
- *Report generation* utilities.
- *Performance monitoring* utilities.
- Other functions, such as *sorting*, *user monitoring*, *data compression*, etc.

### Data Dictionary Subsystem

The data dictionary subsystem is used to store data about many aspects of how the DBMS works. The data contained in the Dictionary subsystem varies from DBMS to DBMS, but in all systems it is a key component of the database. Typical data to be contained in the Dictionary includes: definitions of the users of the system and the access rights they have, usage standards, details of the data structures used to contain data in the DBMS, design decisions, application program descriptions, descriptions of business rules that are stored and enforced within the DBMS, definitions of the additional data structures used to improve systems performance.

It is important to understand that because of the importance and sensitive nature of the data contained in the Dictionary subsystem, most users will have none or little direct access to this information. However the Database Administrator will need to have regular access to much of the dictionary system, and should have a detailed knowledge of the way in which the Dictionary is organized.

- ***Active data dictionary*** is accessed by DBMS software and users/DBA.
- ***Passive data dictionary*** is accessed by users/DBA only.

### Performance Management Subsystem

The performance management subsystem provides facilities to optimize (or at least improve) DBMS performance. This is necessary because the large and complex software in a DBMS requires attention to ensure it performs efficiently, i.e. it allows retrieval and changes to data to be made without requiring users to wait for significant periods of time for the DBMS to carry out the requested action.

Two important functions of the Performance management subsystem are:

- **Query optimization**: Structuring SQL queries (or other forms of user queries) to minimize response times
- **DBMS reorganization**: Maintaining statistics on database usage and taking (or recommending) actions such as database reorganization, creating indexes, and so on to improve DBMS performance

### Data Integrity Management Subsystem

The data integrity management subsystem provides facilities for managing the integrity of data in the database and the integrity of metadata in the Dictionary. This subsystem is concerned with ensuring that data is, as far as software can ensure, correct and consistent. There are three important functions:

- *Intra-record integrity:* Enforcing constraints on data item values and types within each record in the database.
- *Referential integrity:* Enforcing the validity of references between records in the database.
- *Concurrency control:* Assuring the validity of database updates when multiple users access the database.

### Backup and Recovery Subsystem

The backup and recovery subsystem provides facilities for logging transactions and database changes, periodically making backup copies of the database, and recovering the database in the event of some type of failure. (backup and recovery will explained in greater detail in a advanced database). A good DBMS will provide comprehensive and flexible mechanisms for backing up and restoring copies of data, and it will be up to the Database Administrator, in consultation with users of the system, to decide precisely how these features should be used.

### Application Development Subsystem

The application development sub-system is for programmers to develop complete database applications. It includes CASE tools (software to enable the modeling of applications), as well as facilities such as screen generators (for automatically creating the screens of an application given details about the data to be input and/or output) and report generators.

In most commercial situations there will in fact be a number of different database systems, operating within a number of different computer environments. By computer environment we mean a set of programs and data made available usually on a particular computer. One such set of database systems, used in a number of medium to large companies, involves the establishment of 3 different computer environments.

The first of these is the Development environment, where new applications are developed and new applications, whether written within the company or bought in from outside, are tested. The Development environment usually contains relatively little data, just enough in fact to adequately test the logic of the applications being developed and tested. Security within the Development environment is usually not an important issue, unless the actual logic of the applications being developed is, in its own right, of a sensitive nature.

The second of the three environments is often called pre-production. Applications that have been tested in the development environment will be moved into pre-production for volume testing, that is testing with quantities of data that are typical of the application when it is in live operation.

The final environment is known as the production or live environment. Applications should only be moved into this environment when they have been fully tested in Pre-production. Security is nearly always a very important issue in the production environment, as the data being used reflects important information in current use by the Organization.

Each of these separate environments will have at least one database system, and because of the widely varying activities and security measures required in each environment, the volumes of data and degree of administration required will itself vary considerably between environments, with the production database (s) requiring by far the most support.

Given the need for the Database Administrator to migrate both programs and data between these environments, an important tool in performing this process will be a set of utilities or programs for migrating applications and their associated data both forwards and backwards between the environments in use.

### Security Management Subsystem

The security management subsystem provides facilities to protect and control access to the database and data dictionary.

## 1.4 Components of Database System

Taking a DBMS as a system, one can describe it with respect to its environment or other systems interacting with the DBMS. To design and use a database, there should be the interaction or integration of Hardware, Software, Data, Procedure and People.

- ***Hardware*:**

Hardware are components that one can touch and feel. These components are comprised of various types of personal computers, mainframe or any server computers to be used in multi-user system, network infrastructure, and other peripherals required in the system.

- ***Software*:**

Software are collection of commands and programs used to manipulate the hardware to perform a function. These include components like the DBMS software, application programs, operating systems, network software, language software and other relevant software.

- ***Data*:**

Since the goal of any database system is to have better control of the data and making data useful, Data is the most important component to the user of the database. There are two categories of data in any database system: that is *Operational* and *Metadata*. Operational data is the data actually stored in the system to be used by the user. Metadata is the data that is used to store information about the database itself. The structure of the data in the database is called the *schema*, which is composed of the *Entities*, *Properties of entities*, and *relationship between entities*.

- ***Procedure:***

Procedure is the rules and regulations on *how to design and use* a database. It includes procedures like how to log on to the DBMS, how to use facilities, how to start and stop transaction, how to make backup, how to treat hardware and software failure, how to change the structure of the database.

- ***People*:**

This component is composed of the people in the organization that are responsible or play a role in designing, implementing, managing, administering and using the resources in the database. This component includes group of people with high level of knowledge about the database and the design technology to other with no knowledge of the system except using the data in the database. In general database users are include the following people:

- *Database Administrator*
- *Application programmer and System analysts*
- *Database Designer*
- *End Users*

## Roles in Database Design and Use

As people are one of the components in DBMS environment, there are group of roles played by different stakeholders of the designing and operation of a database system.

**1. Database Administrator (DBA)**

- Responsible to oversee, control and manage the database resources (the database itself, the DBMS and other related software)
- Authorizing access to the database
- Coordinating and monitoring the use of the database
- Responsible for determining and acquiring hardware and software resources
- Accountable for problems like poor security, poor performance of the system
- Involves in all steps of database development

We can have further classifications of this role in big organizations having huge amount of data and user requirement.

A. *Data Administrator (DA):* is responsible on management of data resources. Involved in database planning, development, maintenance of standards, policies and procedures at the conceptual and logical design phases.

B. *Database Administrator (DBA):* is a more technical role. Is responsible for the physical realization of the database. Involves in physical design, implementation, security and integrity control of the database.

**2. Database Designer (DBD)**

- Identifies the data to be stored and choose the appropriate structures to represent and store the data. Most of these functions are done before the database is implemented and populated with the data.
- Should understand the user requirement and should choose how the user views the database. So that it is responsibility to communicate with all prospective users to understand their requirements and come up with a design that meets these requirements.
- Involve on the design phase before the implementation of the database system.
- Database designers interact with all potential users and develop views of the database that meet the data and processing requirements of these groups.

We have two distinctions of database designers, one involving in the logical and conceptual design and another involving in physical design.

***a. Logical and Conceptual DBD***

- ✓ Identifies data (entity, attributes and relationship) relevant to the organization
- ✓ Identifies constraints on each data
- ✓ Understand data and business rules in the organization
- ✓ Sees the database independent of any data model at conceptual level and consider one specific data model at logical design phase.

***b. Physical DBD***

- ✓ Take logical design specification as input and decide how it should be physically realized.
- ✓ Map the logical data model on the specified DBMS with respect to tables and integrity constraints. (DBMS dependent designing)
- ✓ Select specific storage structure and access path to the database
- ✓ Design security measures required on the database

**3. Application Programmer and Systems Analyst**

- × System analyst determines the user requirement and how the user wants to view the database.
- × The application programmer implements these specifications as programs; code, test, debug, document and maintain the application program.
- × Determines the interface on how to retrieve, insert, update and delete data in the database.
- × The application could use any high level programming language according to the availability, the facility and the required service.

**4. End Users**

Workers, whose job requires accessing the database frequently for various purposes, there are different group of users in this category.

***i. Naïve Users:***

- ✓ Sizable proportion of users
- ✓ Unaware of the DBMS
- ✓ Only access the database based on their access level and demand
- ✓ Use standard and pre-specified types of queries.

***ii. Sophisticated Users***

- ✓ Are users familiar with the structure of the Database and facilities of the DBMS.
- ✓ Have complex requirements
- ✓ Have higher level queries
- ✓ Are most of the time engineers, scientists, business analysts, etc

***iii. Casual Users***

- ✓ Users who access the database occasionally.
- ✓ Need different information from the database each time.
- ✓ Use sophisticated database queries to satisfy their needs.
- ✓ Are most of the time middle to high level managers.

These users can be again classified as "Actors on the Scene" and "Workers Behind the Scene".

**Actors on the Scene:**

- ➢ Data Administrator
- ➢ Database Administrator
- ➢ Database Designer
- ➢ End Users

**Workers behind the Scene**

- ➢ *DBMS designers and implementers*: who design and implement different DBMS software.
- ➢ *Tool Developers:* experts who develop software packages that facilitates database system designing and use. Prototype, simulation, code generator developers could be an example. Independent software vendors could also be categorized in this group.
- ➢ *Operators and Maintenance Personnel:* system administrators who are responsible for actually running and maintaining the hardware and software of the database system and the information technology facilities.

## 1.5 Database Development Life Cycle

As it is one component in most information system development tasks, there are several steps in designing a database system. Here more emphasis is given to the design phases of the system development life cycle. The major steps in database design are;

1. **Planning**: that is identifying information gap in an organization and propose a database solution to solve the problem.

2. **Data analysis and requirements**: that concentrates more on fact finding about the problem or the opportunity. Feasibility analysis, requirement determination and structuring, and selection of best design method are also performed at this phase.

    i. Designer's efforts are focused on

        a) Information needs of Information users.

        b) Information sources. Information constitution.

    ii. Sources of information for the designer

        a) Developing and gathering end user data views

        b) Direct observation of the current system: existing and desired output

        c) Interface with the systems design group

    iii. The designer must identify the company's business rules and analyze their impacts.

3. **Define Problems and Constraints**

    - How does the existing system function?
    - What input does the system require?
    - What documents does the system generate?
    - How is the system output used? By Whom?
    - What are the operational relationships among business units?
    - What are the limits and constraints imposed on the system?

4. **Design**: in database designing more emphasis is given to this phase. The phase is further divided into three sub-phases.

    **a)** **<u>Conceptual Design</u>**: concise description of the data, data type, relationship between data and constraints on the data. There is no implementation or physical detail consideration. Used to elicit and structure all information requirements

**b) Logical Design**: a higher level conceptual abstraction with selected *specific data model* to implement the data structure. It is particular DBMS independent and with no other physical considerations.

**c) Physical Design**: physical implementation of the upper level design of the database with respect to internal storage and file structure of the database for the selected DBMS. To develop all technology and organizational specification.

**5. DBMS Selection**

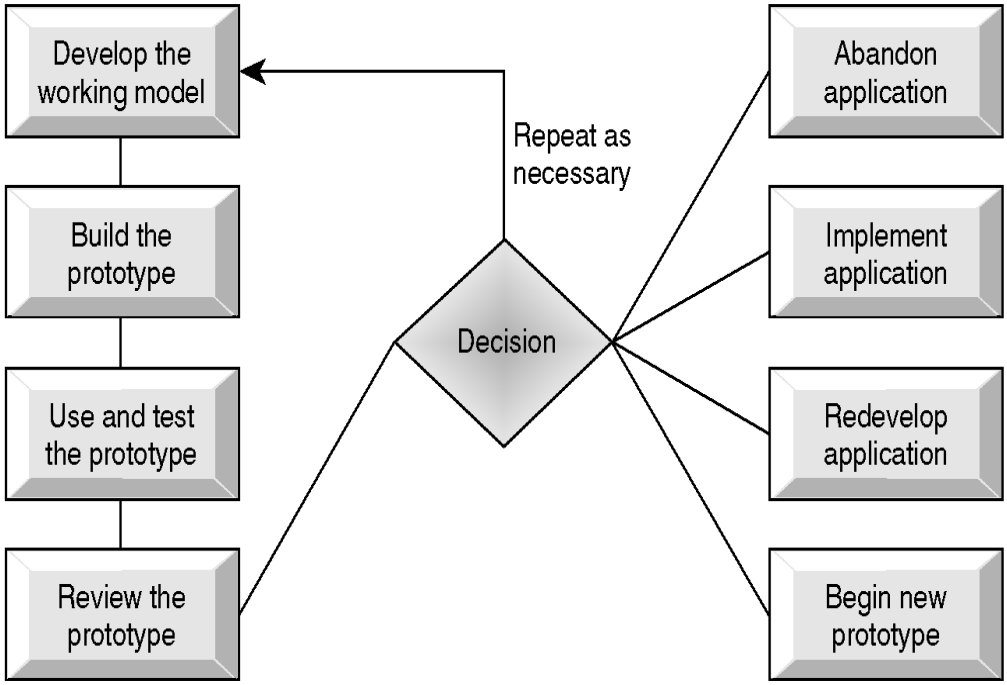
- The selection of an appropriate DBMS to support the database application.
- Undertaken at any time prior to logical design provided sufficient information is available regarding system requirements.
- Also design the user interface and the application programs using the selected DBMS

**6. Prototyping:** Building a working model of a database application.

Purpose

- To identify features of a system that work well, or are inadequate
- To suggest improvements or even new features
- To clarify the users' requirements

**Prototype Development Method Stages**



7. **Implementation**: the deployment and testing of the designed database for use.
8. **Data conversion and loading**
9. **Testing**
    - ✓ The process of executing the application programs with the intent of finding errors.
    - ✓ Use carefully planned test strategies and realistic data.
    - ✓ Testing cannot show the absence of faults; it can show only that software faults are present.
    - ✓ Demonstrates that database and application programs *appear* to be working according to requirements.
10. **Operation and Support**: administering and maintaining the operation of the database system and providing support to users.
    - ✓ The process of monitoring and maintaining the system following installation.
    - ✓ Monitoring the performance of the system.
    - ✓ If performance falls, may require reorganization of the database.
    - ✓ Maintaining and upgrading the database application (when required).
    - ✓ Incorporating new requirements into the database application.