ST. XAVIER'S COLLEGE

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**DATABASE MANAGEMENT SYSTEM**

**LAB ASSIGNMENT # 1**

**SUBMITTED BY:**

AJITA KHATIWADA  
013BSCCSIT004

4rd Semester

**SUBMITTED TO:**

|  |  |
| --- | --- |
| **Er.Sanjya Kr.Yadav**  lecturer,  Department of Computer Science  St. Xavier’s College |  |

**Date of Submission:** 24th July 2015

**Database Management System (DBMS)**

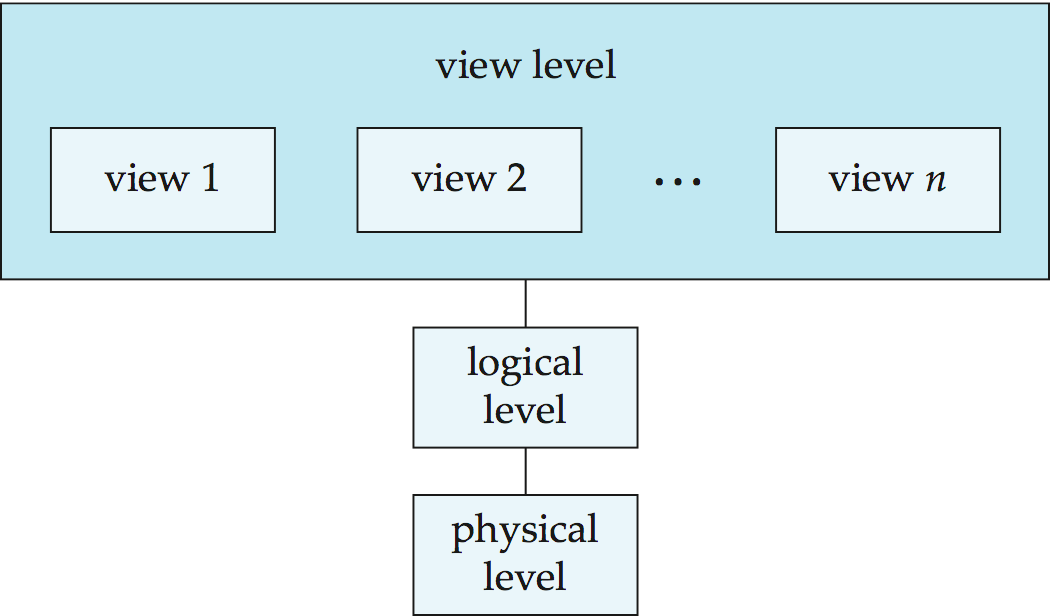
A database management system (DBMS) is a collection of programs that enables you to store, modify, and extract information from a database. There are many different types of database management systems, ranging from small systems that run on personal computers to huge systems that run on mainframes.

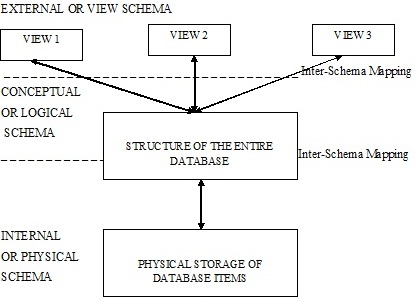
**Views of Data**

* Physical level: describes how a record (e.g., customer) is stored.
* Logical level: describes data stored in database, and the relationships among the data.

* View level: application programs hide details of data types. Views can also hide information (such as an employee’s salary) for security purposes.

An architecture for a database system :





**Functions of a DBMS**

So what does a DBMS really do? It organizes your files to give you more control over your data.

A DBMS makes it possible for users to create, edit and update data in database files. Once created, the DBMS makes it possible to store and retrieve data from those database files.

More specifically, a DBMS provides the following functions:

* Concurrency: concurrent access (meaning 'at the same time') to the same database by multiple users
* Security: security rules to determine access rights of users
* Backup and recovery: processes to back-up the data regularly and recover data if a problem occurs
* Integrity: database structure and rules improve the integrity of the data
* Data descriptions: a data dictionary provides a description of the data

### Types of DBMS languages

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

Statements are used to define the database structure or schema.   
  
Some examples:

 CREATE - to create objects in the database

 ALTER - alters the structure of the database

 DROP - delete objects from the database

 TRUNCATE - remove all records from a table, including all spaces allocated for the records are removed

 COMMENT - add comments to the data dictionary

 RENAME - rename an object

**2. Data Manipulation Language (DML)** : Statements are used for managing data within schema objects. Language for accessing and manipulating the data organized by the appropriate data model

DML also known as query language

Two classes of languages

Procedural – user specifies what data is required and how to get those data

Declarative (nonprocedural) – user specifies what data is required without specifying how to get those data

SQL is the most widely used query language   
  
Some examples:

 SELECT - Retrieve data from the a database

 INSERT - Insert data into a table

 UPDATE - Updates existing data within a table

 DELETE - deletes all records from a table, the space for the records remain

 MERGE - UPSERT operation (insert or update)

 CALL - Call a PL/SQL or Java subprogram

 EXPLAIN PLAN - explain access path to data

 LOCK TABLE - control concurrency

**3. Data Control Language (DCL)**

Some examples:

 GRANT - gives user's access privileges to database

 REVOKE - withdraw access privileges given with the GRANT command

**Transaction Control (TCL)** : Statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.   
  
Some examples:

 COMMIT - save work done

 SAVEPOINT - identify a point in a transaction to which you can later roll back

 ROLLBACK - restore database to original since the last COMMIT

 SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use

**Relational database :**

Computer database in which all data is stored in Relations which (to the user) are tables with rows and columns. Each table is composed of records (called Tuples) and each record is identified by a field (attribute) containing a unique value. Every table shares at least one field with another table in 'one to one,' 'one to many,' or 'many to many' relationships. These relationships allow the database user to access the data in almost an unlimited number of ways, and to combine the tables as building blocks to create complex and very large databases.

**Database design**

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database.

**Object-Oriented Model**

Object DBMSs add database functionality to object programming languages. They bring much more than persistent storage of programming language objects. Object DBMSs extend the semantics of the C++, Smalltalk and Java object programming languages to provide full-featured database programming capability, while retaining native language compatibility. A major benefit of this approach is the unification of the application and database development into a seamless data model and language environment. As a result, applications require less code, use more natural data modeling, and code bases are easier to maintain. Object developers can write complete database applications with a modest amount of additional effort.

According to Rao (1994), "The object-oriented database (OODB) paradigm is the combination of object-oriented programming language (OOPL) systems and persistent systems. The power of the OODB comes from the seamless treatment of both persistent data, as found in databases, and transient data, as found in executing programs."

In contrast to a relational DBMS where a complex data structure must be flattened out to fit into tables or joined together from those tables to form the in-memory structure, object DBMSs have no performance overhead to store or retrieve a web or hierarchy of interrelated objects. This one-to-one mapping of object programming language objects to database objects has two benefits over other storage approaches: it provides higher performance management of objects, and it enables better management of the complex interrelationships between objects. This makes object DBMSs better suited to support applications such as financial portfolio risk analysis systems, telecommunications service applications, world wide web document structures, design and manufacturing systems, and hospital patient record systems, which have complex relationships between data.

**Semistructured Model**

In semistructured data model, the information that is normally associated with a schema is contained within the data, which is sometimes called ``self-describing''. In such database there is no clear separation between the data and the schema, and the degree to which it is structured depends on the application. In some forms of semistructured data there is no separate schema, in others it exists but only places loose constraints on the data. Semi-structured data is naturally modelled in terms of graphs which contain labels which give semantics to its underlying structure. Such databases subsume the modelling power of recent extensions of flat relational databases, to nested databases which allow the nesting (or encapsulation) of entities, and to object databases which, in addition, allow cyclic references between objects.

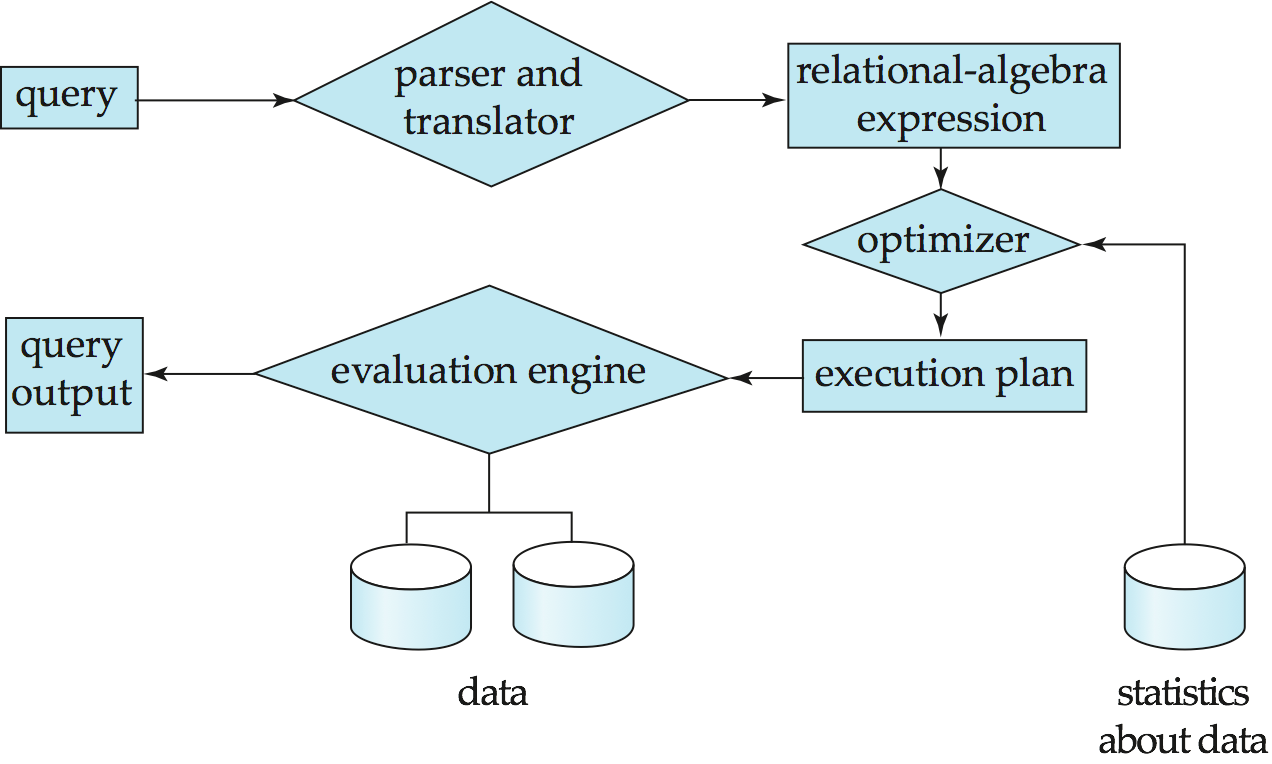
Semistructured data has recently emerged as an important topic of study for a variety of reasons. First, there are data sources such as the Web, which we would like to treat as databases but which cannot be constrained by a schema. Second, it may be desirable to have an extremely flexible format for data exchange between disparate databases. Third, even when dealing with structured data, it may be helpful to view it as semistructured for the purposes of browsing.

**Query Processing**

1. Parsing and translation

2. Optimization

3. Evaluation



* Alternative ways of evaluating a given query

Equivalent expressions

Different algorithms for each operation

* Cost difference between a good and a bad way of evaluating a query can be enormous
* Need to estimate the cost of operations

Depends critically on statistical information about relations which the database must maintain

* Need to estimate statistics for intermediate results to compute cost of complex expressions

**Transaction Management**

* Advant What if the system fails?
* What if more than one user is concurrently updating the same data?
* A **transaction** is a collection of operations that performs a single logical function in a database application
* **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
* **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database ages

**Example of Transactions and Concurrent Access**

* Transaction to transfer $50 from account *A* to account *B*:

1. **read**(*A*)

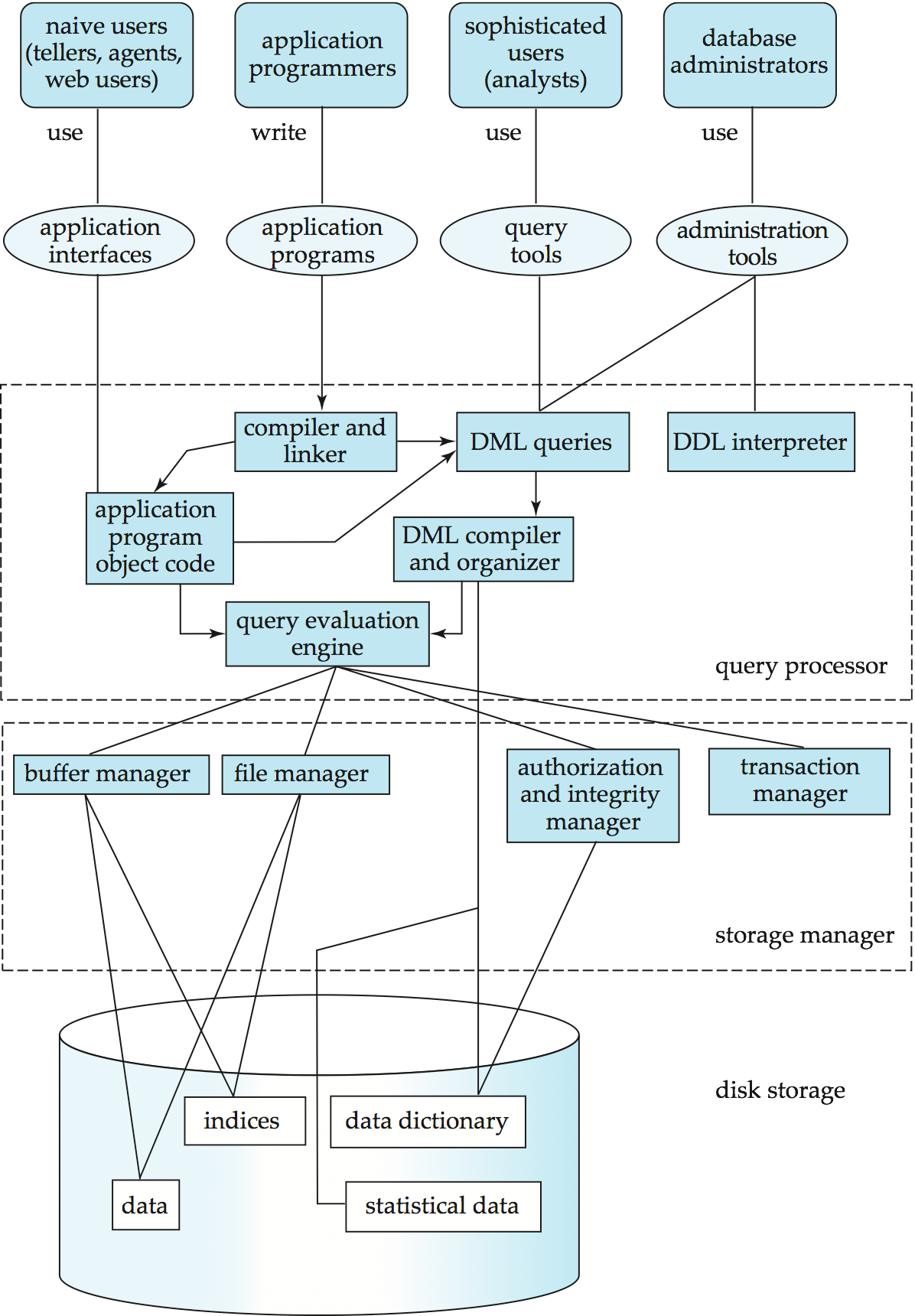
2. *A* := *A –* 50

3. **write**(*A*)

4. **read**(*B*)

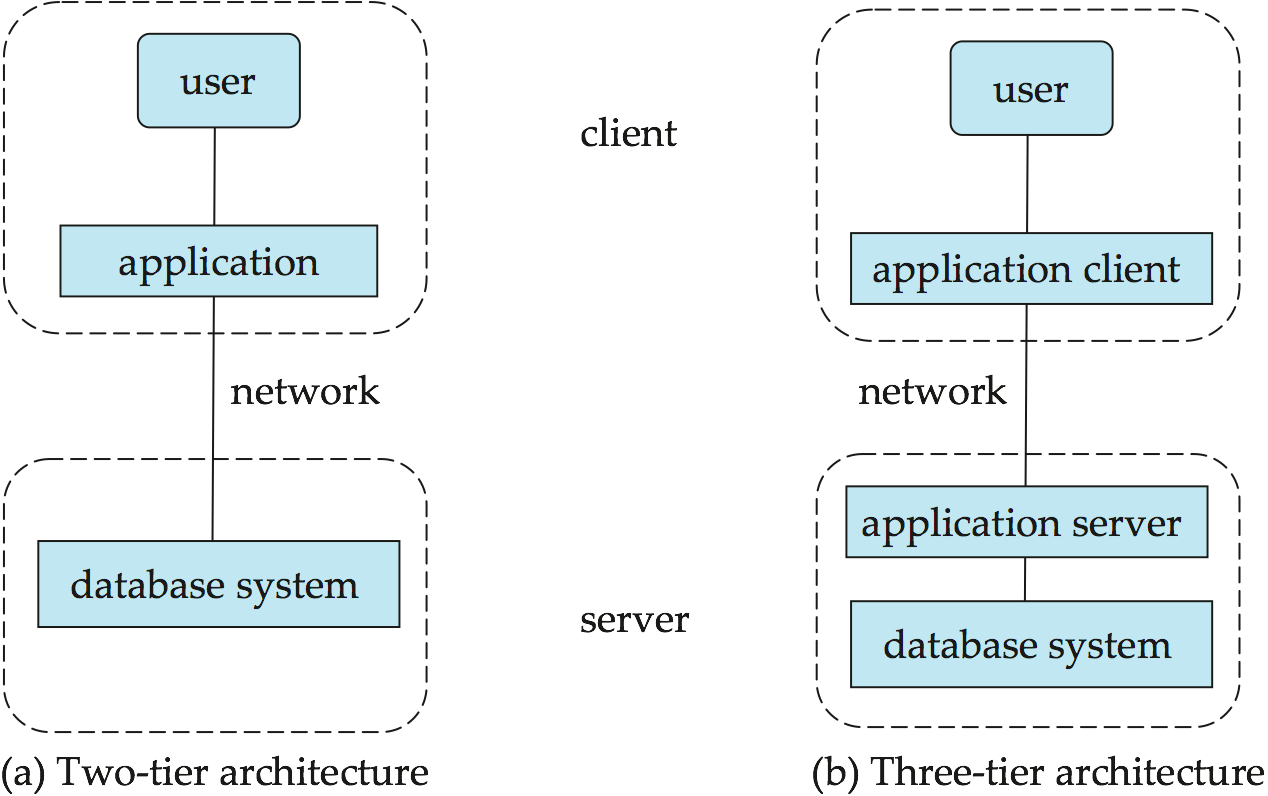
5. *B* := *B +* 50

**Overall Database System Structure**



The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:

* Centralized
* Client-server
* Parallel (multi-processor)
* Distributed



**Database Users**

Usersare differentiated by the way they expect to interact with the system

* **Application programmers** – interact with system through DML calls
* **Sophisticated users** – form requests in a database query language
* **Specialized users** – write specialized database applications that do not fit into the traditional data processing framework
* **Naïve users** – invoke one of the permanent application programs that have been written previously
* Examples, people accessing database over the web, bank tellers, clerical staff

**Database Administrator**

* Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise’s information resources and needs.
* Database administrator's duties include:
* Schema definition
* Storage structure and access method definition
* Schema and physical organization modification
* Granting user authority to access the database
* Specifying integrity constraints
* Acting as liaison with users
* Monitoring performance and responding to changes in requirements

**History of Database Systems**

* 1950s and early 1960s:
  + Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  + Punched cards for input
* Late 1960s and 1970s:
  + Hard disks allowed direct access to data
  + Network and hierarchical data models in widespread use
  + Ted Codd defines the relational data model
    - Would win the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley begins Ingres prototype
* High-performance (for the era) transaction processing
* **1980s:**
* Research relational prototypes evolve into commercial systems
* SQL becomes industrial standard
* Parallel and distributed database systems
* Object-oriented database systems
* **1990s:**
* Large decision support and data-mining applications
* Large multi-terabyte data warehouses
* Emergence of Web commerce
* **Early 2000s:**
* XML and XQuery standards
* Automated database administration
* Later 2000s:
* Giant data storage systems
* Google BigTable, Yahoo PNuts, Amazon, ..

**Advantages and**

**Disadvantages of Database Management System**

We must evaluate whether there is any gain in using a DBMS over a situation where we do not use it. Let us summarize the advantages.

* **Reduction of Redundancy**: This is perhaps the most significant advantage of using DBMS. Redundancy is the problem of storing the same data item in more one place. Redundancy creates several problems like requiring extra storage space, entering same data more than once during data insertion, and deleting data from more than one place during deletion. Anomalies may occur in the database if insertion, deletion etc are not done properly.
* **Sharing of Data**: In a paper-based record keeping, data cannot be shared among many users. But in computerized DBMS, many users can share the same database if they are connected via a network.
* **Data Integrity**: We can maintain data integrity by specifying integrity constrains, which are rules and restrictions about what kind of data may be entered or manipulated within the database. This increases the reliability of the database as it can be guaranteed that no wrong data can exist within the database at any point of time.
* **Data security**: We can restrict certain people from accessing the database or allow them to see certain portion of the database while blocking sensitive information. This is not possible very easily in a paper-based record keeping.

However, there could be a few disadvantages of using DBMS. They can be as following:

* As DBMS needs computers, we have to invest a good amount in acquiring the hardware, software, installation facilities and training of users.
* We have to keep regular backups because a failure can occur any time. Taking backup is a lengthy process and the computer system cannot perform any other job at this time.
* While data security system is a boon for using DBMS, it must be very robust. If someone can bypass the security system then the database would become open to any kind of mishandling.