**Users of DBMS :**

There are a number of users who can access or retrieve data on demand using the applications and interfaces provided by the DBMS. Each type of user needs different software capabilities. The users of a database system can be classified in the following groups, depending on their degrees of expertise or the mode of their interactions with the DBMS. The users can be:

• Naive Users

• Online Users

• Application Programmers

• Sophisticated Users

• Data Base Administrator (DBA)

**Naive Users:** Naive Users are those users who need not be aware of the presence of the database system or any other system supporting their usage. Naive users are end users of the database who work through a menu driven application program, where the type and range of response is always indicated to the user.

A user of an Automatic Teller Machine (ATM) falls in this category. The user is instructed through each step of a transaction. He or she then responds by pressing a coded key or entering a numeric value. The operations that can be performed by valve users are very limited and affect only a precise portion of the database. For example, in the case of the user of the Automatic Teller Machine, user's action affects only one or more of his/her own accounts.

**Online Users :** Online users are those who may communicate with the database directly via an online terminal or indirectly via a user interface and application program. These users are aware of the presence of the database system and may have acquired a certain amount of expertise with in the limited interaction permitted with a database.

**Sophisticated Users :** Such users interact with the system without ,writing programs.

Instead, they form their requests in database query language. Each such query is submitted to a very processor whose function is to breakdown DML statement into instructions that the storage manager understands.

**Specialized Users :** Such users are those ,who write specialized database application that do not fit into the fractional data-processing framework. For example: Computer-aided design systems, knowledge base and expert system, systems that store data with complex [data type](http://ecomputernotes.com/java/data-type-variable-and-array/explain-data-types-in-java)s (for example, graphics data and audio data).

**Application Programmers** **:** Professional programmers are those who are responsible for developing application programs or user interface. The application programs could be written using general purpose programming language or the commands available to manipulate a database.

**Database Administrator:**The database administrator (DBA) is the person or group in charge for implementing the database system ,within an organization. The "DBA has all the system privileges allowed by the DBMS and can assign (grant) and remove (revoke) levels of access (privileges) to and from other users. DBA is also responsible for the evaluation, selection and implementation of DBMS package.

**What is a Database Management System (DBMS)?**

Database Management System (DBMS) is a collection of programs which enables its users to access database, manipulate data, reporting / representation of  data .

It also helps to control access to the  database.

Database Management Systems are not a new concept and as such had been first implemented in 1960s.

Charles Bachmen's [Integrated Data Store](http://en.wikipedia.org/wiki/Integrated_Data_Store) (IDS) is said to be the first DBMS in history.

With time database technologies evolved a lot while usage and expected functionalities of databases have been increased immensely.

**Types of DBMS**

Let's see how the DBMS family got evolved with the time. Following diagram shows the evolution of DBMS categories.



There are 4 major types of DBMS. Let's look into them in detail.

* **Hierarchical** - this type of DBMS employs the "parent-child" relationship of storing data. This type of DBMS is rarely used nowadays. Its structure is like a tree with nodes representing records and branches representing fields. The windows registry used in Windows XP is an example of a hierarchical database. Configuration settings are stored as tree structures with nodes.
* **Network DBMS** - this type of DBMS supports many-to many relations. This usually results in complex database structures.  RDM Server is an example of a database management system that implements the network model.
* **Relational DBMS** - this type of DBMS defines database relationships in form of tables, also known as relations. Unlike network DBMS, RDBMS does not support many to many relationships.Relational DBMS usually have pre-defined data types that they can support. This is the most popular DBMS type in the market. Examples of relational database management systems include MySQL, Oracle, and Microsoft SQL Server database.
* **Object Oriented Relation DBMS** - this type supports storage of new data types. The data to be stored is in form of objects. The objects to be stored in the database have attributes (i.e. gender, ager) and methods that define what to do with the data. PostgreSQL is an example of an object oriented relational DBMS.

**What is a Database Management System (DBMS)?**

Database Management System (DBMS) is a collection of programs which enables its users to access database, manipulate data, reporting / representation of  data .

It also helps to control access to the  database.

Database Management Systems are not a new concept and as such had been first implemented in 1960s.

Charles Bachmen's [Integrated Data Store](http://en.wikipedia.org/wiki/Integrated_Data_Store) (IDS) is said to be the first DBMS in history.

With time database technologies evolved a lot while usage and expected functionalities of databases have been increased immensely.

**Types of DBMS**

Let's see how the DBMS family got evolved with the time. Following diagram shows the evolution of DBMS categories.



There are 4 major types of DBMS. Let's look into them in detail.

* **Hierarchical** - this type of DBMS employs the "parent-child" relationship of storing data. This type of DBMS is rarely used nowadays. Its structure is like a tree with nodes representing records and branches representing fields. The windows registry used in Windows XP is an example of a hierarchical database. Configuration settings are stored as tree structures with nodes.
* **Network DBMS** - this type of DBMS supports many-to many relations. This usually results in complex database structures.  RDM Server is an example of a database management system that implements the network model.
* **Relational DBMS** - this type of DBMS defines database relationships in form of tables, also known as relations. Unlike network DBMS, RDBMS does not support many to many relationships.Relational DBMS usually have pre-defined data types that they can support. This is the most popular DBMS type in the market. Examples of relational database management systems include MySQL, Oracle, and Microsoft SQL Server database.
* **Object Oriented Relation DBMS** - this type supports storage of new data types. The data to be stored is in form of objects. The objects to be stored in the database have attributes (i.e. gender, ager) and methods that define what to do with the data. PostgreSQL is an example of an object oriented relational DBMS.

**Tuple (Database)**

**Definition - What does *Tuple (Database)* mean?**

In the context of relational databases, a tuple is one record (one row). The information in a database can be thought of as a spreadsheet, with columns (known as fields or attributes) representing different categories of information, and tuples (rows) representing all the information from each field associated with a single record.

**Techopedia explains *Tuple (Database)***

In a relational database, a tuple contains all the data for an individual record. For example, in a database containing client contact information, the fields may be categories such as name, phone number, email address and mailing address, while a tuple for that database could be:

|  |  |  |  |
| --- | --- | --- | --- |
| Bill Gates | 206-555-1234 | billg@[microsoft.com](https://microsoft.com/) | PO Box 123, Seattle, WA 98100 |

In mathematics, a tuple is an ordered list of elements. Related to this is an n-tuple, which in set theory is a collection (sequence) of "n" elements. Given this, it might be more properly said that tuples are implemented as records even though the terms are commonly used interchangeably.

# What is relation in DBMS?

A general term used in database design is a “relational database"—but a database relation is not the same thing and does not imply, as its name suggests, a relationship between tables. A database relation simply refers to an individual table in a relational database.

In a relational database, the table is a relation because it stores the relation between data in its column-row format. The columns are the table's attributes, while the rows represent the data records. A single row is known as a tuple to database designers.

**The Definition and Properties of a Relation**

A relation, or table, in a relational database has some common properties.

* its name must be unique in the database, for example - a database cannot contain multiple tables of the same name.
* Each relation must have a set of columns or attributes, and it must have a set of rows to contain the data. As with the table names, no attributes can have the same name.
* A tuple (or row) can be a duplicate. In practice, a database might actually contain duplicate rows, but there should be practices in place to avoid this, such as the use of unique primary keys (next up).

Given that a tuple cannot be a duplicate, it follows that a relation must contain at least one attribute (or column) that identifies each tuple (or row) uniquely. This is usually the primary key. This primary key cannot be duplicated.

Further, field must contain a single value. For example - you cannot enter something like "Tom Smith" and expect the database to understand that you have a first and last name; rather, the database will understand that the value of that cell is exactly what has been entered.

**Structure Components, and Functions of DBMS**

**Structure of DBMS:**

* **DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations such as insert, delete, update and retrieval on the database.**
* **The components of DBMS perform these requested operations on the database and provide necessary data to the users.
The various components of DBMS are described below:**
* ****

**Components of a DBMS**

**The components of DBMS can be divided into two parts:**

***Function and Services of DBMS***

1. **DDL Compiler:**
	* **Data Description Language compiler processes schema definitions specified in the DDL.**
	* **It includes metadata information such as the name of the files, data items, storage details of each file, mapping information and constraints etc.**
2. **DML Compiler and Query optimizer:**
	* **The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access.**
	* **The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.**
3. **Data Manager:**
	* **The Data Manager is the central software component of the DBMS also knows as Database Control System.**
	* **The Main Functions Of Data Manager Are:**
		1. **Convert operations in user's Queries coming from the application programs or combination of DML Compiler and Query optimizer which is known as Query Processor from user's logical view to physical file system.**
		2. **Controls DBMS information access that is stored on disk.**
		3. **It also controls handling buffers in main memory.**
		4. **It also enforces constraints to maintain consistency and integrity of the data.**
		5. **It also synchronizes the simultaneous operations performed by the concurrent users.**
		6. **It also controls the backup and recovery operations.**
4. **Data Dictionary:**
	* **Data Dictionary, which stores metadata about the database, in particular the schema of the database.**
	* **names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.**
	* **Detailed information on physical database design such as storage structure, access paths, files and record sizes.**
	* **Usage statistics such as frequency of query and transactions.**
	* **Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as a important part of the DBMS**
5. **Data Files:**
	* **Which store the database itself.**
6. **Compiled DML:**
	* **The DML complier converts the high level Queries into low level file access commands known as compiled DML.**
7. **End Users:**
	* **The second class of users then is end user, who interacts with system from online workstation or terminals.**
	* **Use the interface provided as an integral part of the database system software.**
	* **User can request, in form of query, to access database either directly by using particular language, such as SQL, or by using some pre-developed application interface.**
	* **Such request are sent to query evaluation engine via DML pre-compiler and DML compiler**
	* **The query evaluation engine accepts the query and analyses it.**
	* **It finds the suitable way to execute the compiled SQL statements of the query.**
	* **Finally, the compiled SQL statements are executed to perform the specified operation**
	* **Query Processor Units:**

**Interprets DDL statements into a set of tables containing metadata.
Translates DML statements into low level instructions that the query evaluation engine understands.
Converts DML statements embedded in an application program into procedure calls int he host language.
Executes low level instructions generated by DML compiler.**

* + 1. **DDL Interpreter**
		2. **DML Compiler**
		3. **Embedded DML Pre-compiler**
		4. **Query Evalution Engine**
	+ **Storage Manager Units**

**Checks the authority of users to access data.**

**Checks for the satisfaction of the integrity constraints.**

**Preserves atomicity and controls concurrency.**

**Manages allocation of splace on disk.**

**Fetches data from disk storage to memory for being used.**

* + 1. **Authorization Manager**
		2. **Integrity Manager**
		3. **Transaction Manager**
		4. **File manager**
		5. **Buffer Manager**
	+ **Functions of DBMS:**
		1. **DBMS free the programmers from the need to worry about the organization and location of the data i.e. it shields the users from complex hardware level details.**
		2. **DBMS can organize process and present data elements from the database. This capability enables decision makers to search and query database contents in order to extract answers that are not available in regular Reports.**
		3. **Programming is speeded up because programmer can concentrate on logic of the application.**
		4. **It includes special user friendly query languages which are easy to understand by non programming users of the system.**
	+ **The service provided by the DBMS includes :-**
		1. **Authorization services like log on to the DBMS start the database stop the Database etc.**
		2. **Transaction supports like Recovery, Rollback etc,**
		3. **Import and Export of Data.**
		4. **Maintaining data dictionary**
		5. **User's Monitoring**

**Entities**

An entity is an object that exists. It doesn't have to do anything; it just has to exist. In database administration, an **entity** can be a single thing, person, place, or object. Data can be stored about such entities. A design tool that allows database administrators to view the relationships between several entities is called the **entity relationship diagram (ERD)**.

In database administration, only those things about which data will be captured or stored is considered an entity. If you aren't going to capture data about something, there's no point in creating an entity in a database.

If you're creating a database of your employees, examples of entities you may have include employees and health plan enrollment.

**Entity Attributes**

An **attribute** defines the information about the entity that needs to be stored. If the entity is an employee, attributes could include name, employee ID, health plan enrollment, and work location. An entity will have zero or more attributes, and each of those attributes apply only to that entity. For example, the employee ID of 123456 belongs to that employee entity alone.

Attributes also have further refinements, such as domain and key. The **domain** of an entity describes the possible values of attributes. In the entity, each attribute will have only one value, which could be blank or it could be a number, text, a date, or a time. Here are examples of entity types and domains:

* Name: Jane Doe
* Employee ID: 123456
* Health Plan Enrollment: Premium Plan
* Work Location: RO, ME, Floor 2

Here's an example figure of an entity.

|  |
| --- |
| entity type |

**Domains:**

In data management and database analysis, a Data Domain refers to all the valid values which a data element (column) may contain. The rule for determining the domain boundary may be as simple as a data type with a list of possible values.

For example, a database table that has information about people, with one record per person, might have an "age" column. This gender column might be declared as a SMALLINT data type, and allowed to have a value between 0 and 120. The data domain for the age column is hence 0 - 120.

In a normalized data model, the reference domain is typically specified in a reference table. Following the previous example, the age reference table *could have*exactly 120 records, one per allowed value. Reference tables are formally related to other tables in a database by the use of foreign keys.

A better way would be to enforce the data domain through a check constraint. For example, the age column would require positive numeric values between 0 and 120.

 [**Advantages and Disadvantages of DBMS.**](http://ecomputernotes.com/fundamental/what-is-a-database/advantages-and-disadvantages-of-dbms)

## Advantages of DBMS

The database management system has promising potential advantages, which are explained below:

**1. Controlling Redundancy:** In file system, each application has its own private files, which cannot be shared between multiple applications. 1:his can often lead to considerable redundancy in the stored data, which results in wastage of storage space. By having centralized database most of this can be avoided. It is not possible that all redundancy should be eliminated. Sometimes there are sound business and technical reasons for· maintaining multiple copies of the same data. In a database system, however this redundancy can be controlled.

**For example:** In case of college database, there may be the number of applications like General Office, Library, Account Office, Hostel etc. Each of these applications may maintain the following information into own private file applications:

             

It is clear from the above file systems, that there is some common data of the student which has to be mentioned in each application, like Rollno, Name, Class, Phone\_No~ Address etc. This will cause the problem of redundancy which results in wastage of storage space and difficult to maintain, but in case of centralized database, data can be shared by number of applications and the whole college can maintain its computerized data with the following database:

              

It is clear in the above database that Rollno, Name, Class, Father\_Name, Address,

Phone\_No, Date\_of\_birth which are stored repeatedly in file system in each application, need not be stored repeatedly in case of database, because every other application can access this information by joining of relations on the basis of common column i.e. Rollno. Suppose any user of Library system need the Name, Address of any particular student and by joining of Library and General Office relations on the basis of column Rollno he/she can easily retrieve this information.

Thus, we can say that centralized system of DBMS reduces the redundancy of data to great extent but cannot eliminate the redundancy because RollNo is still repeated in all the relations.

**2. Integrity can be enforced:** Integrity of data means that data in database is always accurate, such that incorrect information cannot be stored in database. In order to maintain the integrity of data, some integrity constraints are enforced on the database. A DBMS should provide capabilities for defining and enforcing the constraints.

For Example: Let us consider the case of college database and suppose that college having only BTech, MTech, MSc, BCA, BBA and BCOM classes. But if a \.,ser enters the class MCA, then this incorrect information must not be stored in database and must be prompted that this is an invalid data entry. In order to enforce this, the integrity constraint must be applied to the class attribute of the student entity. But, in case of file system tins constraint must be enforced on all the application separately (because all applications have a class field).

In case of DBMS, this integrity constraint is applied only once on the class field of the

General Office (because class field appears only once in the whole database), and all other applications will get the class information about the student from the General Office table so the integrity constraint is applied to the whole database. So, we can conclude that integrity constraint can be easily enforced in centralized DBMS system as compared to file system.

**3. Inconsistency can be avoided**: When the same data is duplicated and changes are made at one site, which is not propagated to the other site, it gives rise to inconsistency and the two entries regarding the same data will not agree. At such times the data is said to be inconsistent. So, if the redundancy is removed chances of having inconsistent data is also removed.

Let us again, consider the college system and suppose that in case of General\_Office file

it is indicated that Roll\_Number 5 lives in Amritsar but in library file it is indicated that

Roll\_Number 5 lives in Jalandhar. Then, this is a state at which tIle two entries of the same object do not agree with each other (that is one is updated and other is not). At such time the database is said to be inconsistent.

An inconsistent database is capable of supplying incorrect or conflicting information. So there should be no inconsistency in database. It can be clearly shown that inconsistency can be avoided in centralized system very well as compared to file system ..

Let us consider again, the example of college system and suppose that RollNo 5 is .shifted from Amritsar to Jalandhar, then address information of Roll Number 5 must be updated, whenever Roll number and address occurs in the system. In case of file system, the information must be updated separately in each application, but if we make updation only at three places and forget to make updation at fourth application, then the whole system show the inconsistent results about Roll Number 5.

In case of DBMS, Roll number and address occurs together only single time in General\_Office table. So, it needs single updation and then an other application retrieve the address information from General\_Office which is updated so, all application will get the current and latest information by providing single update operation and this single update operation is propagated to the whole database or all other application automatically, this property is called as Propagation of Update.

We can say the redundancy of data greatly affect the consistency of data. If redundancy is less, it is easy to implement consistency of data. Thus, DBMS system can avoid inconsistency to great extent.

**4. Data can be shared:** As explained earlier, the data about Name, Class, Father \_\_name etc. of General\_Office is shared by multiple applications in centralized DBMS as compared to file system so now applications can be developed to operate against the same stored data. The applications may be developed without having to create any new stored files.

**5. Standards can be enforced** : Since DBMS is a central system, so standard can be enforced easily may be at Company level, Department level, National level or International level. The standardized data is very helpful during migration or interchanging of data. The file system is an independent system so standard cannot be easily enforced on multiple independent applications.

**6. Restricting unauthorized access:** When multiple users share a database, it is likely that some users will not be authorized to access all information in the database. For example, account office data is often considered confidential, and hence only authorized persons are allowed to access such data. In addition, some users may be permitted only to retrieve data, whereas other are allowed both to retrieve and to update. Hence, the type of access operation retrieval or update must also be controlled. Typically, users or user groups are given account numbers protected by passwords, which they can use to gain access to the database. A DBMS should provide a security and authorization subsystem, which the DBA uses to create accounts and to specify account restrictions. The DBMS should then enforce these restrictions automatically.

**7. Solving Enterprise Requirement than Individual Requirement:** Since many types of users with varying level of technical knowledge use a database, a DBMS should provide a variety of user interface. The overall requirements of the enterprise are more important than the individual user requirements. So, the DBA can structure the database system to provide an overall service that is "best for the enterprise".

For example: A representation can be chosen for the data in storage that gives fast access for the most important application at the cost of poor performance in some other application. But, the file system favors the individual requirements than the enterprise requirements

**8.** **Providing Backup and Recovery:**A DBMS must provide facilities for recovering from hardware or software failures. The backup and recovery subsystem of the DBMS is responsible for recovery. For example, if the computer system fails in the middle of a complex update program, the recovery subsystem is responsible for making sure that the .database is restored to the state it was in before the program started executing.

**9.** **Cost of developing and maintaining system is lower:**It is much easier to respond to unanticipated requests when data is centralized in a database than when it is stored in a conventional file system. Although the initial cost of setting up of a database can be large, but the cost of developing and maintaining application programs to be far lower than for similar service using conventional systems. The productivity of programmers can be higher in using non-procedural languages that have been developed with DBMS than using procedural languages.

**10. Data** **Model can be developed :**The centralized system is able to represent the complex data and interfile relationships, which results better data modeling properties. The data madding properties of relational model is based on Entity and their Relationship, which is discussed in detail in chapter 4 of the book.

11. **Concurrency Control :**DBMS systems provide mechanisms to provide concurrent access of data to multiple users.

## Disadvantages of DBMS

The disadvantages of the database approach are summarized as follows:

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

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

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

**4.** **Higher impact of a failure:**The centralization of resources increases the vulnerability of the system. Since all users and applications rely on the ~vailabi1ity of the DBMS, the failure of any component can bring operations to a halt.

**5.** **Cost of DBMS:**The cost of DBMS varies significantly, depending on the environment and functionality provided. There is also the recurrent annual maintenance cost.

**6. 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.

**7. Cost of Conversion:** In some situations, the cost oftlle 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 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 modern database technology.

## When not to Use a DBMS

In spite of the advantages of using a DBMS, there are a few situations in which such a system may involve unnecessary overhead costs, as that would not be incurred in traditional file processing.

The overhead costs of using a DBMS are due to the following:

•  High initial investment in hardware, software, and training.

• Generality that a DBMS provides for defining and processing data.

• Overhead for providing security, concurrency control, recovery, and integrity functions.

Additional problems may arise, if the database designers and DBA do not properly design the database or if the database systems applications are not implemented properly.

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

• The database and applications are simple, well defined and not expected to change.

• There are tight real-time requirements for some programs that may not be met because of DBMS overhead.

• Multiple user access to data is not required.

• An application may need to manipulate the data in a way not supported by the query language.