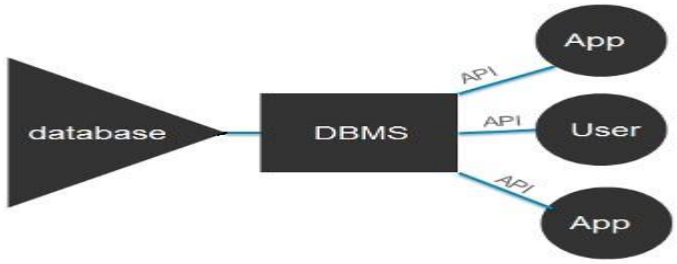
INTRODUCTION TO DATABASE MANAGEMENT SYSTEM

DEFINITON OF TERMS

Database- is a collection of information that is organized so that it can easily be accessed, managed, and updated. A DBMS makes it possible for end users to create, read, update and delete data in a database.

Data management-This involves data collection, management and retrieval and it constitutes a core activity for any organization.

Database management system (DBMS)
- is system software for creating and managing databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data. The DBMS essentially serves as an interface between the database and end users or application programs, ensuring that data is consistently organized and remains easily accessible. The DBMS can offer both logical and physical data independence. That means it can protect users and applications from needing to know where data is stored or having to be concerned about changes to the physical structure of data (storage and hardware). As long as programs use the application programming interface (API).



Well-known DBMSs include MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase and IBM DB2. A database is not generally portable across different DBMSs, but different DBMS can interoperate by using standards such as SQL and ODBC or JDBC to ***Bit*** (Character) - a bit is the smallest unit of data representation (value of a bit may be a 0 or 1). Eight bits make a byte which can represent a character or a special symbol in a character code.
Field - a field consists of a grouping of characters. A data field represents an attribute (a characteristic or quality) of some entity (object, person, place, or event).
Record - a record represents a collection of attributes that describe a real-world entity.
A record consists of fields, with each field describing an attribute of the entity.
Tuple is one record (one row) representing a set of values for an entity in a database
Table -A **table** is a collection of related records held within a database.

It consists of columns (fields) and rows (records).

A table has a specified number of fields, but can have any number of records

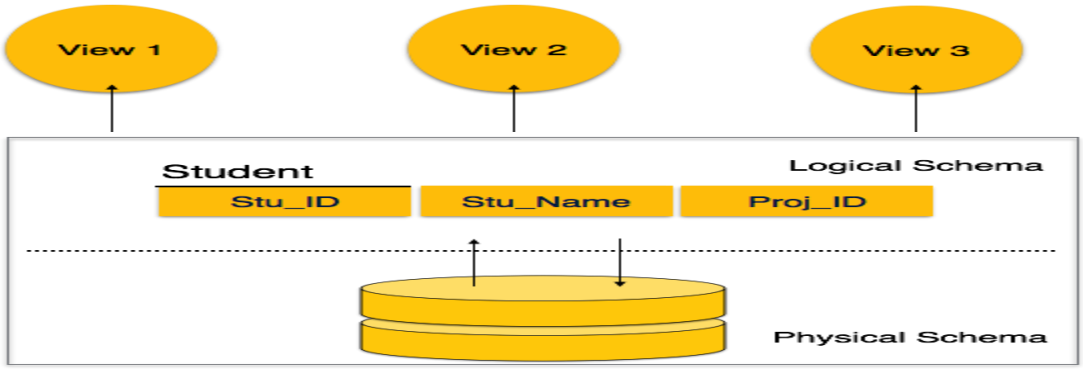
Each record is identified by one or more values appearing in a particular field subset.

A specific choice of a field which uniquely identify a record is called the primary key.

A *database* is a collection of information that is organized so that it can be easily accessed, managed and updated.

They support electronic storage and manipulation of data hence making easy data management allow a single application to work with more than one DBMS

Database Schema: A database schema is the skeleton structure that represents the logical view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data. A database schema defines its entities and the relationship among them. It contains a descriptive detail of the database, which can be depicted by means of schema diagrams. It's the database designers who design the schema to help programmers understand the database and make it useful.



A database schema can be divided broadly into two categories –

- **Physical Database Schema** – This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.

- **Logical Database Schema** – This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Database Instance: It is important that we distinguish these two terms individually. Database schema is the skeleton of database. It is designed when the database doesn't exist at all. Once the database is operational, it is very difficult to make any changes to it. A database schema does not contain any data or information.

A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

1. The ability to modify a scheme definition in one level without affecting a scheme definition in a higher level is called data independence.
2. There are two kinds:
    - Physical data independence
        - The ability to modify the physical scheme without causing application programs to be rewritten
        - Modifications at this level are usually to improve performance
    - Logical data independence
        - The ability to modify the conceptual scheme without causing application programs to be rewritten
        - Usually done when logical structure of database is altered
3. Logical data independence is harder to achieve as the application programs are usually heavily dependent on the logical structure of the data. An analogy is made to abstract data types in programming languages.

Query Processor

This transforms the user queries into a series of low level instructions. This reads the online user's query and translates it into an efficient series of operations in a form capable of being sent to the run time data manager for execution.

Run Time Database Manager

Sometimes referred to as the database control system, this is the central software component of the DBMS that interfaces with user-submitted application programs and queries, and handles database access at run time. Its function is to convert operations in user's queries. It provides control to maintain the consistency, integrity and security of the data.

Data Manager

Also called the cache manger, this is responsible for handling of data in the database, providing a recovery to the system that allows it to recover the data after a failure.

Database Engine

The core service for storing, processing, and securing data, this provides controlled access and rapid transaction processing to address the requirements of the most demanding data consuming applications. It is often used to create relational databases for online transaction processing or online analytical processing data.

Data Dictionary

This is a reserved space within a database used to store information about the database itself. A data dictionary is a set of read-only table and views, containing the different information about the data used in the enterprise to ensure that database representation of the data follow one standard as defined in the dictionary.

Report Writer

Also referred to as the report generator, it is a program that extracts information from one or more files and presents the information in a specified format. Most report writers allow the user to select records that meet certain conditions and to display selected fields in rows and columns, or also format the data into different charts.

Application of DBMS

Below are the popular applications of DBMS:

Sector	Use of DBMS
Banking	For customer information, account activities, payments, deposits, loans, etc.
Airlines	For reservations and schedule information.
Universities	For student information, course registrations, colleges and grades.
Telecommunication	It helps to keep call records, monthly bills, maintaining balances, etc.

Finance	For storing information about stock, sales, and purchases of financial instruments like stocks and bonds.
Sales	Use for storing customer, product & sales information.
Manufacturing	It is used for the management of supply chain and for tracking production of items. Inventories status in warehouses.
HR Management	For information about employees, salaries, payroll, deduction, generation of paychecks, etc.

Characteristic of a good database

1. Should be able to store all kinds of data that exists in this real world. Since we need to work with all kinds of data and requirements, database should be strong enough to store all kinds of data that is present around us.

2. Should be able to relate the entities / tables in the database by means of a relation. i.e.; any two tables should be related. Let us say, an employee works for a department. This implies that Employee is related to a particular department. We should be able to define such a relationship between any two entities in the database. There should not be any table lying without any mapping.

3. Data and application should be isolated. Because database is a system which gives the platform to store the data, and the data is the one which allows the database to work. Hence there should be clear differentiation between them.

4. There should not be any duplication of data in the database. Data should be stored in such a way that it should not be repeated in multiple tables. If repeated, it would be unnecessary waste of DB space and maintaining such data becomes chaos.

5. DBMS has a strong query language. Once the database is designed, this helps the user to retrieve and manipulate the data. If a particular user wants to see any specific data, he can apply as many filtering conditions that he wants and pull the data that he needs.

6. Multiple users should be able to access the same database, without affecting the other user. i.e.; if teachers want to update a student's marks in Results table at the same time, then they should be allowed to update the marks for their subjects, without modifying other subject marks. A good database should support this feature.
7. It supports multiple views to the user, depending on his role. In a school database, Students will able to see only their reports and their access would be read only. At the same time teachers will have access to all the students with the modification rights. But the database is the same. Hence a single database provides different views to different users.

8. Database should also provide security, i.e.; when there are multiple users are accessing the database, each user will have their own levels of rights to see the database. Some of them will be allowed to see whole database, and some will have only partial rights. For example, instructor who is teaching Physics will have access to see and update marks of his subject. He will not have access for other subjects. But the HOD will have full access on all the subjects.

9. Database should also support ACID property. i.e.; while performing any transactions like insert, update and delete, database makes sure that the **real purpose of the data is not lost.** For example, if a student's address is updated, then it should make sure that there is no duplicate data is created nor there is any data mismatch for that student.

DATABASE MANAGEMENT SYSTEM ARCHITECTURE

Database system architecture means design or construction of database system. The database system architecture provides general concept and structure of database system. The architecture of most commercial database management systems is based on the threelevel architecture.

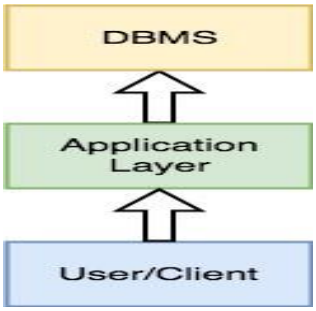
1-tier architecture/single tier.

In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

2-tier architecture

2-tier DBMS architecture includes an **Application layer** between the user and the DBMS, which is responsible to communicate the user's request to the database management system and then send the response from the DBMS to the user.

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.



Such an architecture provides the DBMS extra security as it is not exposed to the End User directly. Also, security can be improved by adding security and authentication checks in the Application layer too.

Advantages of 2-tier Architecture

- Easy to understand as it directly communicates with the database.
- Requested data can be retrieved very quickly, when there is less number of users.
- Easy to modify – any changes required, directly requests can be sent to database
- Easy to maintain – When there are multiple requests, it will be handled in a queue and there will not be any chaos.

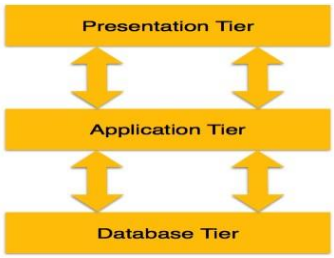
Disadvantages of 2-tier architecture:

- It would be time consuming, when there is huge number of users. All the requests will be queued and handed one after another. Hence it will not respond to multiple users at the same time.
- This architecture would little cost effective.

3-tier DBMS Architecture

3-tier DBMS architecture is the most commonly used architecture for web applications.

- **Database (Data) Tier** – At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier** – At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
- **User (Presentation) Tier** – End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.



Advantages of 3-tier architecture:

- Easy to maintain and modify. Any changes requested will not affect any other data in the database. Application layer will do all the validations.
- Improved security. Since there is no direct access to the database, data security is increased. There is no fear of mishandling the data. Application layer filters out all the malicious actions.
- Good performance. Since this architecture cache the data once retrieved, there is no need to hit the database for each request. This reduces the time consumed for multiple requests and hence enables the system to respond at the same time.

Disadvantage 3-tier Architecture

1. It is little more complex and little more effort is required in terms of hitting the database.

Characteristics of Database Management System

1. **Data stored into Tables:** Data is never directly stored into the database. Data is stored into tables, created inside the database. DBMS also allows to have relationships between tables which makes the data more meaningful and connected. You can easily understand what type of data is stored where by looking at all the tables created in a database.
2. **Reduced Redundancy:** In the modern world hard drives are very cheap, but earlier when hard drives were too expensive, unnecessary repetition of data in database was a big problem. But DBMS follows **Normalization** which divides the data in such a way that repetition is minimum.
3. **Data Consistency:** On Live data, i.e. data that is being continuously updated and added, maintaining the consistency of data can become a challenge. But DBMS handles it all by itself.
4. **Support Multiple user and Concurrent Access:** DBMS allows multiple users to work on it(update, insert, delete data) at the same time and still manages to maintain the data consistency.
5. **Query Language:** DBMS provides users with a simple Query language, using which data can be easily fetched, inserted, deleted and updated in a database.
6. **Security:** The DBMS also takes care of the security of data, protecting the data from unauthorized access. In a typical DBMS, we can create user accounts with different access permissions, using which we can easily secure our data by restricting user access.
7. DBMS supports **transactions**, which allows us to better handle and manage data integrity in real world applications where multi-threading is extensively used.

BRIEF HISTORY OF DATABASE AND DBMS History of DBMS

Here, are the important landmarks from the history:

- 1960 - Charles Bachman designed first DBMS system
- 1970 - Codd introduced IBM'S Information Management System (IMS)
- 1976- Peter Chen coined and defined the Entity-relationship model also know as the ER model
- 1980 - Relational Model becomes a widely accepted database component 1985- Object-oriented DBMS develops.
- 1990s- Incorporation of object-orientation in relational DBMS.

Following the technology progress in the areas of processors, computer memory, computer storage and computer networks, the sizes, capabilities, and performance of databases and their respective DBMSs have grown in orders of magnitude. The development of database technology can be divided into three eras based on data model or structure: navigational,SQL/relational, and post-relational.

The two main early navigational data models were the hierarchical model, epitomized by IBM's IMS system, and the CODASYL model (network model), implemented in a number of products such as IDMS.

The relational model, first proposed in 1970 by Edgar F. Codd, departed from this tradition by insisting that applications should search for data by content, rather than by following links. The relational model employs sets of ledger-style tables, each used for a different type of entity. Only in the mid-1980s did computing hardware become powerful enough to allow the wide deployment of relational systems (DBMSs plus applications). By the early 1990s, however, relational systems dominated in all large-scale data processing applications, and as of 2015 they remain dominant : IBM DB2, Oracle, MySQL and Microsoft SQL Server are the top DBMS. The dominant database language, standardised SQL for the relational model, has influenced database languages for other data model

Object databases were developed in the 1980s to overcome the inconvenience of objectrelational impedance mismatch, which led to the coining of the term "post-relational" and also the development of hybrid object-relational databases.
The next generation of post-relational databases in the late 2000s became known as NoSQL databases, introducing fast key-value stores and document-oriented databases. A competing "next generation" known as NewSQL databases attempted new implementations that retained the relational/SQL model while aiming to match the high performance of NoSQL compared to commercially available relational DBMSs.

As computers grew in speed and capability, a number of general-purpose database systems emerged; by the mid-1960s a number of such systems had come into commercial use. Interest in a standard began to grow, and Charles Bachman, author of one such product, the Integrated Data Store (IDS), founded the "Database Task Group" within CODASYL, the group responsible for the creation and standardization of COBOL. In 1971 the Database Task Group delivered their standard, which generally became known as the "CODASYL approach", and soon a number of commercial products based on this approach entered the market.

The CODASYL approach relied on the "manual" navigation of a linked data set which was formed into a large network. Applications could find records by one of three methods:

1. Use of a primary key (known as a CALC key, typically implemented by hashing)
2. Navigating relationships (called sets) from one record to another

3. Scanning all the records in a sequential order

1970s, relational DBMS

Edgar Codd worked at IBM in San Jose, California, in one of their offshoot offices that was primarily involved in the development of hard disk systems. He was unhappy with the navigational model of the CODASYL approach, notably the lack of a "search" facility. In 1970, he wrote a number of papers that outlined a new approach to database construction that eventually culminated in the groundbreaking *A Relational Model of Data for Large Shared Data Banks*.

In this paper, he described a new system for storing and working with large databases. Instead of records being stored in some sort of linked list of free-form records as in CODASYL, Codd's idea was to use a "table" of fixed-length records, with each table used for a different type of entity. A linked-list system would be very inefficient when storing "sparse" databases where some of the data for any one record could be left empty. The relational model solved this by splitting the data into a series of normalized tables (or *relations*), with optional elements being moved out of the main table to where they would take up room only if needed. Data may be freely inserted, deleted and edited in these tables, with the DBMS doing whatever maintenance needed to present a table view to the application/user.

In the relational model, records are "linked" using virtual keys not stored in the database but defined as needed between the data contained in the records.
The relational model also allowed the content of the database to evolve without constant rewriting of links and pointers. The relational part comes from entities referencing other entities in what is known as one-to-many relationship, like a traditional hierarchical model, and many-to-many relationship, like a navigational (network) model. Thus, a relational model can express both hierarchical and navigational models, as well as its native tabular model, allowing for pure or combined modeling in terms of these three models, as the application requires.

For instance, a common use of a database system is to track information about users, their name, login information, various addresses and phone numbers. In the navigational approach all of these data would be placed in a single record, and unused items would simply not be placed in the database. In the relational approach, the data would be *normalized* into a user table, an address table and a phone number table (for instance). Records would be created in these optional tables only if the address or phone numbers were actually provided.

Linking the information back together is the key to this system. In the relational model, some bit of information was used as a "key", uniquely defining a particular record. When

information was being collected about a user, information stored in the optional tables would be found by searching for this key. For instance, if the login name of a user is unique, addresses and phone numbers for that user would be recorded with the login name as its key. This simple "re-linking" of related data back into a single collection is something that traditional computer languages are not designed for.

Integrated approach

Database machine

In the 1970s and 1980s attempts were made to build database systems with integrated hardware and software. The underlying philosophy was that such integration would provide higher performance at lower cost. Examples were IBM System/38, the early offering of Teradata, and the Britton Lee, Inc.

Late 1970s, SQL DBMS

IBM started working on a prototype system loosely based on Codd's concepts as *System R* in the early 1970s. The first version was ready in 1974/5, and work then started on multitable systems in which the data could be split so that all of the data for a record (some of which is optional) did not have to be stored in a single large "chunk". Subsequent multiuser versions were tested by customers in 1978 and 1979, by which time a standardized query language – SQL[*citation needed*] – had been added. Codd's ideas were establishing themselves as both workable and superior to CODASYL, pushing IBM to develop a true production version of System R, known as *SQL/DS*, and, later, *Database 2* (DB2).

1980s, on the desktop

The 1980s ushered in the age of desktop computing. The new computers empowered their users with spreadsheets like Lotus 1-2-3 and database software like dBASE. The dBASE product was lightweight and easy for any computer user to understand out of the box. C. Wayne Ratliff the creator of dBASE stated: "dBASE was different from programs like BASIC, C, FORTRAN, and COBOL in that a lot of the dirty work had already been done. The data manipulation is done by dBASE instead of by the user, so the user can concentrate on what he is doing, rather than having to mess with the dirty details of opening, reading, and closing files, and managing space allocation- dBASE was one of the top selling software titles in the 1980s and early 1990s.

1990s, object-oriented

The 1990s, along with a rise in object-oriented programming, saw a growth in how data in various databases were handled. Programmers and designers began to treat the data in their databases as objects. That is to say that if a person's data were in a database, that

person's attributes, such as their address, phone number, and age, were now considered to belong to that person instead of being extraneous data. This allows for relations between data to be relations to objects and their attributes and not to individual fields.[20] The term "object-relational impedance mismatch" described the inconvenience of translating between programmed objects and database tables. Object databases and object-relational databases attempt to solve this problem by providing an object-oriented language (sometimes as extensions to SQL) that programmers can use as alternative to purely relational SQL. On the programming side, libraries known as object-relational mappings (ORMs) attempt to solve the same problem.

2000s, NoSQL and NewSQL

The next generation of post-relational databases in the 2000s became known as NoSQL databases, including fast key-value stores and document-oriented databases.

NoSQL databases are often very fast, do not require fixed table schemas, avoid join operations by storing denormalized data, and are designed to scale horizontally. The most popular NoSQL systems include MongoDB, Couchbase, Riak, Memcached, Redis, CouchDB, Hazelcast, Apache Cassandra and HBase,which are all open-source software products.

NewSQL is a class of modern relational databases that aims to provide the same scalable performance of NoSQL systems for online transaction processing (read-write) workloads while still using SQL and maintaining the ACID guarantees of a traditional database system. Such databases include ScaleBase, Clustrix, EnterpriseDB, MemSQL, NuoDB[25] and VoltDB.

Due to the advancement in the electronic industry, the increased processing and storage capacity of computer has opened the doors for computer scientists to develop various techniques to store large amount of related data in an efficient and compact manner. The concept of database was introduced by IBM in 1960s. A brief description about the development of DBMS and database models is given below:

DATABASE VIEWS/LEVELS OF DATA ABSTRACTION

Database systems are made-up of complex data structures. To ease the user interaction with database, the developers hide internal irrelevant details from users. This process of hiding irrelevant details from user is called data abstraction.

1. **External level -** This is the highest level in data abstraction. At this level users see the data in the form of rows and columns. This level illustrates the users how the data is stored in terms of tables and relations. Users view full or partial data based on the business requirement. The users will have different views here, based on their levels of access rights. For example, student will not have access to see Lecturers salary details, one employee will not have access to see other employees details, unless he is a manager. At this level, one can access the data from database and perform some calculations based on

the data. For example calculate the tax from the salary of employee, calculate CGPA of a Student, Calculate age of a person from his Date of Birth etc. These users can be real users or any programs.

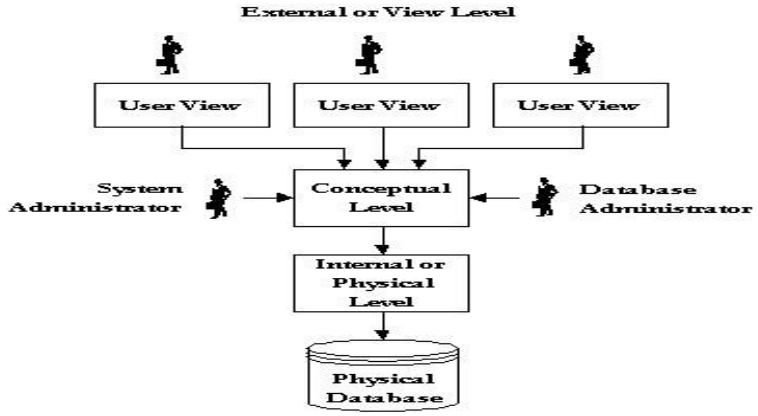
Any changes/ computations done at this level will not affect other levels of data. That means, if we retrieve the few columns of the STUDENT table, it will not change the whole table, or if we calculate the CGPA of a Student, it will not change/update the table. This level of data is based on the below levels, but it will not alter the data at below levels.

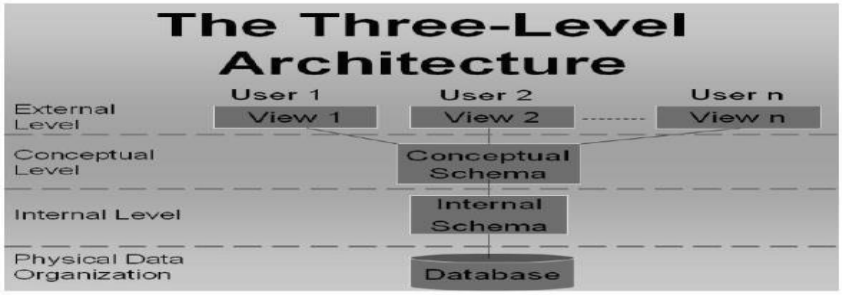
2. **Logical/ Conceptual level -** This is the next level of abstraction. It describes the actual data stored in the database in the form of tables and relates them by means of mapping. This level will not have any information on what a user views at external level. This level will have all the data in the database.

Any changes done in this level will not affect the external or physical levels of data. That is any changes to the table structure or the relation will not modify the data that the user is viewing at the external view or the storage at the physical level. For example, suppose we have added a new column 'skills' which will not modify the external view data on which the user was viewing Ages of the students. Similarly, it will have space allocated for 'Skills' in the physical memory, but it will not modify the space or address of Date of Birth (using which Age will be derived) in the memory. Hence external and physical independence is achieved.

3. **Internal level -** This is one of the intermediary levels. In most of the cases this level is not mentioned and usually it is said that we have 3 levels of data abstraction. This level depends on the DBMS software. This level is how the database is seen from DBMS. We can even combine logical level and this level.

-Physical level - This is the lowest level in data abstraction. This level describes how the data is actually stored in the physical memory like magnetic tapes, hard disks etc. In this level the file organization methods like hashing, sequential, B+ tree comes into picture. At this level, developer would know the requirement, size and accessing frequency of the records clearly. So designing this level will not be much complex for him.





Components of DBMS

A database management system (DBMS) consists of several components. Each component plays very important role in the database management system environment. The major components of database management system are:

- Software

- Hardware
- Data
- Procedures
- People

1. **Software**: The main component of a DBMS is the software. It is the set of programs used to handle the database and to control and manage the overall computerized database

   -DBMS software itself, is the most important software component in the overall system

   -Operating system including network software being used in network, to share the data of database among multiple users.

   -Application programs developed in programming languages such as C++, Visual Basic that are used to access database in database management system. Each program contains statements that request the DBMS to perform operation on database. The operations may include retrieving, updating, deleting data etc . The application program may be conventional or online workstations or terminals.

2. **Hardware:** consists of a set of physical electronic devices such as computers (together with associated I/O devices like disk drives), storage devices, I/O channels, electromechanical devices that make interface between computers and the real world systems etc, and so on. It is impossible to implement the DBMS without the hardware devices, In a network, a powerful computer with high data processing speed and a storage device with large storage capacity is required as database server.
3. **Data**: Data is the most important component of the DBMS. The main purpose of DBMS is to process the data. In DBMS, databases are defined, constructed and then data is stored, updated and retrieved to and from the databases. The database contains both the actual (or operational) data and the metadata (data about data or description about data.

4. **Procedures:** Procedures refer to the instructions and rules that help to design the database and to use the DBMS. The users that operate and manage the DBMS require documented procedures on hot use or run the database management system. These may include.

   1. Procedure to install the new DBMS.
   2. To log on to the DBMS.
   3. To use the DBMS or application program.
   4. To make backup copies of database.
   5. To change the structure of database.
   6. To generate the reports of data retrieved from database.

2. Users: The users are the people who manage the databases and perform different operations on the databases in the database system. There are three kinds of people who play different roles in database system

1. Application Programmers
2. Database Administrators
3. End-Users

Application Programmers

The people who write application programs in programming languages (such as Visual Basic, Java, or C++) to interact with databases are called Application Programmer.

Database Administrators

4. A person who is responsible for managing the overall database management system is called database administrator or simply DBA. : Other primary roles will include:

- Implementation of data models
- Database design
- Database accessibility
- Performance issues
- Capacity issues
- Data replication
- Table Maintenance

End Users

The end-users are the people who interact with database management system to perform different operations on database such as retrieving, updating, inserting, deleting data etc.

There are several categories of end users:

• **Casual end users -**occasionally access the database, but they may need different information each time.

Application Programmers - They are the developers who interact with the database by means of DML queries. These DML queries are written in the application programs like C, C++, JAVA, Pascal etc. These queries are converted into object code to communicate with the database. For example, writing a C program to generate the report of employees who are working in particular department will involve a query to fetch the data from database. It will include a embedded SQL query in the C Program.

Sophisticated Users - They are database developers, who write SQL queries to select/insert/delete/update data. They do not use any application or programs to request

the database. They directly interact with the database by means of query language like SQL. These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the concepts in their requirement. In short, we can say this category includes designers and developers of DBMS and SQL.

Specialized Users - These are also sophisticated users, but they write special database application programs. They are the developers who develop the complex programs to the requirement.

Stand-alone Users - These users will have stand –alone database for their personal use. These kinds of database will have readymade database packages which will have menus and graphical interfaces.

Native Users - these are the users who use the existing application to interact with the database. For example, online library system, ticket booking systems, ATMs etc which has existing application and users use them to interact with the database to fulfill their requests