Introduction to DBMS

DBMS

DBMS stands for **D**ata**b**ase **M**anagement **S**ystem. We can break it like this DBMS = Database + Management System. Database is a collection of data and Management System is a set of programs to store and retrieve those data. Based on this we can **define DBMS** like this: DBMS is a collection of inter-related data and set of programs to store & access those data in an easy and effective manner.

What is the need of DBMS?
Database systems are basically developed for large amount of data. When dealing with huge amount of data, there are two things that require optimization: **Storage of data** and **retrieval of data**.

Storage: According to the principles of database systems, the data is stored in such a way that it acquires lot less space as the redundant data (duplicate data) has been removed before storage. Let's take a layman example to understand this:
In a banking system, suppose a customer is having two accounts, one is saving account and another is salary account. Let's say bank stores saving account data at one place (these places are called tables we will learn them later) and salary account data at another place, in that case if the customer information such as customer name, address etc. are stored at both places then this is just a wastage of storage (redundancy/ duplication of data), to organize the data in a better way the information should be stored at one place and both the accounts should be linked to that information somehow. The same thing we achieve in DBMS.

Fast Retrieval of data: Along with storing the data in an optimized and systematic manner, it is also important that we retrieve the data

quickly when needed. Database systems ensure that the data is retrieved as quickly as possible.

Purpose of Database Systems

The main purpose of database systems is to manage the data. Consider a university that keeps the data of students, teachers, courses, books etc. To manage this data we need to store this data somewhere where we can add new data, delete unused data, update outdated data, retrieve data, to perform these operations on data we need a Database management system that allows us to store the data in such a way so that all these operations can be performed on the data efficiently.

Database systems are much better than traditional file processing systems which we have discussed in the separate article: DBMS vs File System.

Advantages of DBMS over file system

By Chaitanya Singh | Filed Under: DBMS

In this guide, we will discuss what is a file processing system and how Database management systems are better than file processing systems.

Drawbacks of File system

- **Data redundancy:** Data redundancy refers to the duplication of data, lets say we are managing the data of a college where a student is enrolled for two courses, the same student details in

such case will be stored twice, which will take more storage than needed. Data redundancy often leads to higher storage costs and poor access time.

- **Data inconsistency:** Data redundancy leads to data inconsistency, lets take the same example that we have taken above, a student is enrolled for two courses and we have student address stored twice, now lets say student requests to change his address, if the address is changed at one place and not on all the records then this can lead to data inconsistency.
- **Data Isolation:** Because data are scattered in various files, and files may be in different formats, writing new application programs to retrieve the appropriate data is difficult.
- **Dependency on application programs:** Changing files would lead to change in application programs.
- **Atomicity issues:** Atomicity of a transaction refers to "All or nothing", which means either all the operations in a transaction executes or none.

  For example: Lets say Steve transfers 100$ to Negan's account. This transaction consists multiple operations such as debit 100$ from Steve's account, credit 100$ to Negan's account. Like any other device, a computer system can fail lets say it fails after first operation then in that case Steve's account would have been debited by 100$ but the amount was not credited to Negan's account, in such case the rollback of operation should occur to maintain the atomicity of transaction. It is **difficult to achieve atomicity in file processing systems**.

- **Data Security:** Data should be secured from unauthorised access, for example a student in a college should not be able to see the payroll details of the teachers, such kind of security constraints are difficult to apply in file processing systems.

Advantage of DBMS over file system

There are several advantages of Database management system over file system. Few of them are as follows:

- **No redundant data**: Redundancy removed by data normalization. No data duplication saves storage and improves access time.
- **Data Consistency and Integrity**: As we discussed earlier the root cause of data inconsistency is data redundancy, since data normalization takes care of the data redundancy, data inconsistency also been taken care of as part of it
- **Data Security**: It is easier to apply access constraints in database systems so that only authorized user is able to access the data. Each user has a different set of access thus data is secured from the issues such as identity theft, data leaks and misuse of data.
- **Privacy**: Limited access means privacy of data.
- **Easy access to data** – Database systems manages data in such a way so that the data is easily accessible with fast response times.
- **Easy recovery**: Since database systems keeps the backup of data, it is easier to do a full recovery of data in case of a failure.
- **Flexible**: Database systems are more flexible than file processing systems.

Disadvantages of DBMS:

- DBMS implementation cost is high compared to the file system
- Complexity: Database systems are complex to understand
- Performance: Database systems are generic, making them suitable for various applications. However this feature affect their performance for some applications.

DBMS Architecture

In the previous tutorials, we learned basics of DBMS. In this guide, we will see the **DBMS architecture**. Database management systems architecture will help us understand the components of database system and the relation among them.

The architecture of DBMS depends on the computer system on which it runs. For example, in a client-server DBMS architecture, the database systems at server machine can run several requests made by client machine. We will understand this communication with the help of diagrams.

Types of DBMS Architecture

There are three types of DBMS architecture:

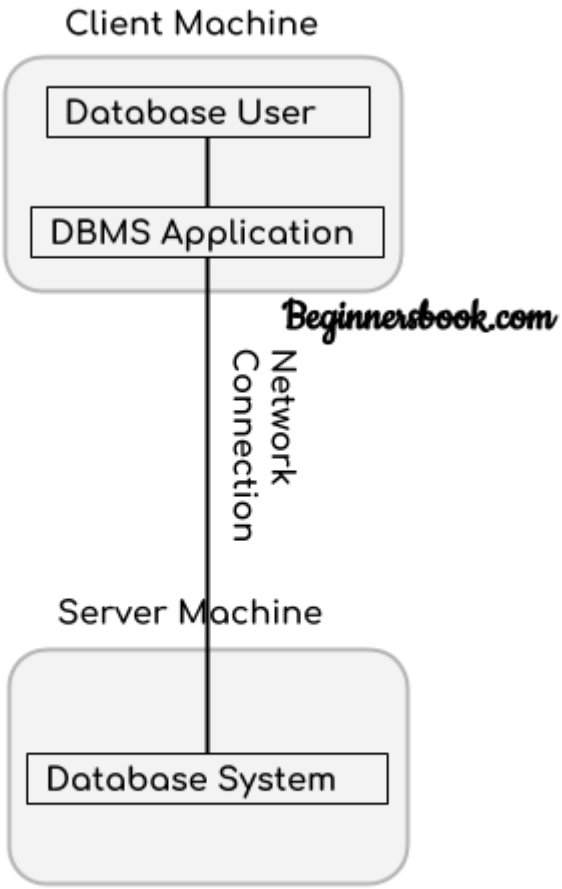
1. Single tier architecture
2. Two tier architecture
3. Three tier architecture

1. Single tier architecture

In this type of architecture, the database is readily available on the client machine, any request made by client doesn't require a network connection to perform the action on the database.

For example, lets say you want to fetch the records of employee from the database and the database is available on your computer system, so the request to fetch employee details will be done by your computer and the records will be fetched from the database by your computer as well. This type of system is generally referred as local database system.

2. Two tier architecture

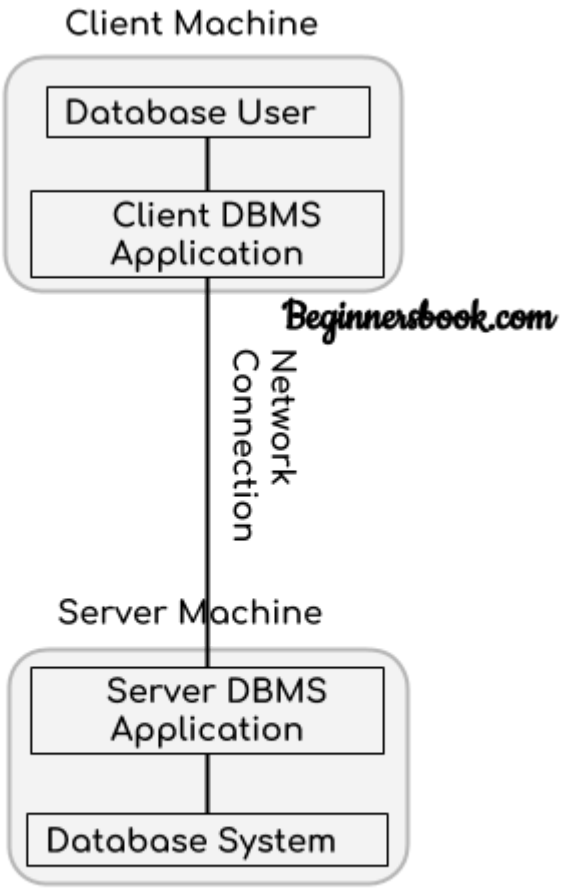


Two-Tier architecture

In two-tier architecture, the Database system is present at the server machine and the DBMS application is present at the client machine, these two machines are connected with each other through a reliable network as shown in the above diagram.

Whenever client machine makes a request to access the database present at server using a query language like sql, the server perform the request on the database and returns the result back to the client. The application connection interface such as JDBC, ODBC are used for the interaction between server and client.

3. Three tier architecture



Three-Tier architecture

In three-tier architecture, another layer is present between the client machine and server machine. In this architecture, the client application doesn't communicate directly with the database systems present at the server machine, rather the client application communicates with server application and the server application internally communicates with the database system present at the server.

View of Data in DBMS

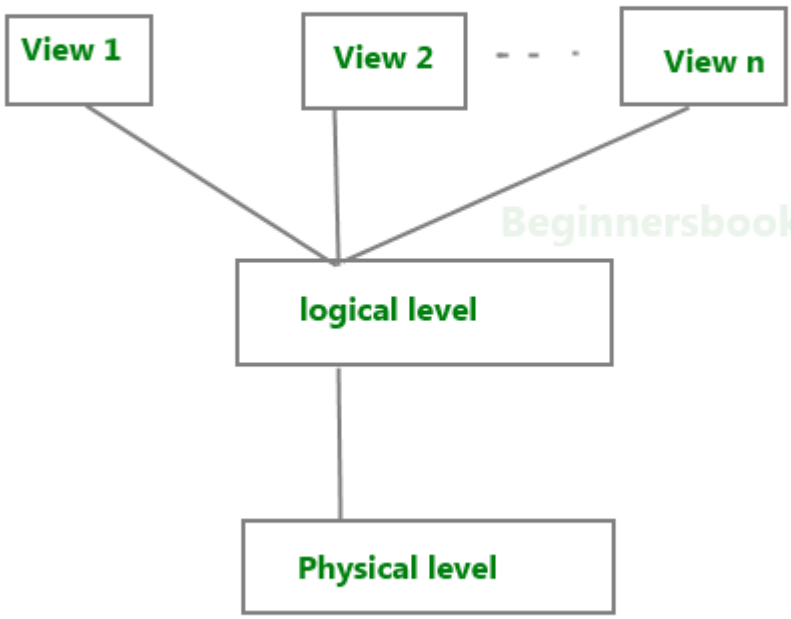
Abstraction is one of the main features of database systems. Hiding irrelevant details from user and providing abstract view of data to users, helps in easy and efficient **user-database** interaction. In the previous tutorial, we discussed the three level of DBMS architecture, The top level of that architecture is "view level". The view level provides the "**view of data**" to the users and hides the irrelevant details such as data relationship, database schema, constraints, security etc from the user.

To fully understand the view of data, you must have a basic knowledge of data abstraction and instance & schema. Refer these two tutorials to learn them in detail.

1. Data abstraction
2. Instance and schema

Data Abstraction in DBMS

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.



Three Levels of data abstraction

We have three levels of abstraction:
Physical level: This is the lowest level of data abstraction. It describes how data is actually stored in database. You can get the complex data structure details at this level.

Logical level: This is the middle level of 3-level data abstraction architecture. It describes what data is stored in database.

View level: Highest level of data abstraction. This level describes the user interaction with database system.

Example: Let's say we are storing customer information in a customer table. At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory. These details are often hidden from the programmers.

At the **logical level** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented. The programmers generally work at this level because they are aware of such things about database systems.

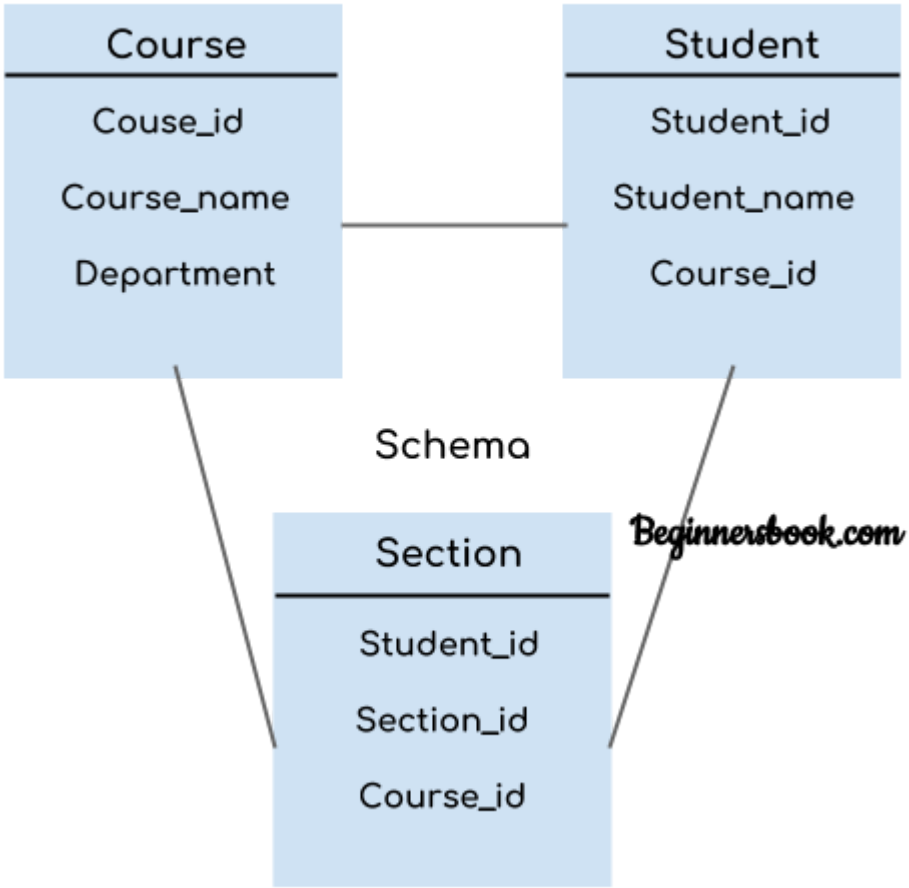
At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.

Instance and schema in DBMS

DBMS Schema

Definition of schema: Design of a database is called the schema. Schema is of three types: Physical schema, logical schema and view schema.

For example: In the following diagram, we have a schema that shows the relationship between three tables: Course, Student and Section. The diagram only shows the design of the database, it doesn't show the data present in those tables. Schema is only a structural view(design) of a database as shown in the diagram below.



The design of a database at physical level is called **physical schema**, how the data stored in blocks of storage is described at this level.

Design of database at logical level is called **logical schema**, programmers and database administrators work at this level, at this level data can be described as certain types of data records gets stored in data structures, however the internal details such as implementation of data structure is hidden at this level (available at physical level).

Design of database at view level is called **view schema**. This generally describes end user interaction with database systems.

DBMS Instance

Definition of instance: The data stored in database at a particular moment of time is called instance of database. Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

For example, lets say we have a single table student in the database, today the table has 100 records, so today the instance of the database has 100 records. Lets say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table. In short, at a particular moment the data stored in database is called the instance, that changes over time when we add or delete data from the database.