**What is Data?**

Data is a collection of a distinct small unit of information. It can be used in a variety of forms like text, numbers, media, bytes, etc. it can be stored in pieces of paper or electronic memory, etc.

**What is Database?**

A **database** is an organized collection of data, so that it can be easily accessed and managed.

You can organize data into tables, rows, columns, and index it to make it easier to find relevant information.

**Database handlers** create a database in such a way that only one set of software program provides access of data to all the users.

The **main purpose** of the database is to operate a large amount of information by storing, retrieving, and managing data.

There are many **dynamic websites** on the World Wide Web nowadays which are handled through databases. For example, a model that checks the availability of rooms in a hotel. It is an example of a dynamic website that uses a database.

There are many **databases available** like MySQL, Sybase, Oracle, MongoDB, Informix, PostgreSQL, SQL Server, etc.

Modern databases are managed by the database management system (DBMS).

**SQL** or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.

**What is DBMS?**

**Database Management System (DBMS)** is a software for storing and retrieving user’s data while considering appropriate security measures. It consists of a group of programs which manipulate the database. The DBMS accepts the request for data from an application and instructs the operating system to provide the specific data. In large systems, a DBMS helps users and other third-party software to store and retrieve data.

DBMS allows users to create their own databases as per their requirement. The term “DBMS” includes the user of the database and other application programs. It provides an interface between the data and the software application.

## Characteristics of Database Management System

* Provides security and removes redundancy
* Self-describing nature of a database system
* Support of multiple views of the data
* Sharing of data and multiuser transaction processing
* DBMS allows entities and relations among them to form tables.
* It follows the ACID concept ( Atomicity, Consistency, Isolation, and Durability).
* DBMS supports multi-user environment that allows users to access and manipulate data in parallel.

|  |  |
| --- | --- |
| * **DBMS** | **File System** |
| DBMS is a collection of data. In DBMS, the user is not required to write the procedures. | File system is a collection of data. In this system, the user has to write the procedures for managing the database. |
| DBMS gives an abstract view of data that hides the details. | File system provides the detail of the data representation and storage of data. |
| DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from the system failure. | File system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will lost. |
| DBMS provides a good protection mechanism. | It is very difficult to protect a file under the file system. |
| DBMS contains a wide variety of sophisticated techniques to store and retrieve the data. | File system can't efficiently store and retrieve the data. |
| DBMS takes care of Concurrent access of data using some form of locking. | In the File system, concurrent access has many problems like redirecting the file while other deleting some information or updating some information. |

## Advantages of DBMS

* DBMS offers a variety of techniques to store & retrieve data
* DBMS serves as an efficient handler to balance the needs of multiple applications using the same data
* Uniform administration procedures for data
* Application programmers never exposed to details of data representation and storage.
* A DBMS uses various powerful functions to store and retrieve data efficiently.
* Offers Data Integrity and Security
* The DBMS implies integrity constraints to get a high level of protection against prohibited access to data.
* A DBMS schedules concurrent access to the data in such a manner that only one user can access the same data at a time
* Reduced Application Development Time

## Disadvantage of DBMS

DBMS may offer plenty of advantages but, it has certain flaws-

* Cost of Hardware and Software of a DBMS is quite high which increases the budget of your organization.
* Most database management systems are often complex systems, so the training for users to use the DBMS is required.
* In some organizations, all data is integrated into a single database which can be damaged because of electric failure or database is corrupted on the storage media
* Use of the same program at a time by many users sometimes lead to the loss of some data.
* DBMS can't perform sophisticated calculations

|  |  |
| --- | --- |
| Types of DBMS |  |

**Hierarchical DBMS**

In a Hierarchical database model, data is organized in a tree-like structure. Data is Stored Hierarchically (top down or bottom up) format. Data is represented using a parent-child relationship. In Hierarchical DBMS parent may have many children, but children have only one parent. Ex: Window registry in Window-XP

**Network Model**

The network database model allows each child to have multiple parents. It helps you to address the need to model more complex relationships like as the orders/parts many-to-many relationship. In this model, entities are organized in a graph which can be accessed through several paths. Ex: RDM server

**Relational model**

Relational DBMS is the most widely used DBMS model because it is one of the easiest. This model is based on normalizing data in the rows and columns of the tables. Relational model stored in fixed structures and manipulated using SQL. Ex: Oracle, MYSQL, Microsoft access [2 tier], PostgreSQL, DB2 etc.

**Object-Oriented Model**

In Object-oriented Model, data is stored in the form of objects. The structure which is called classes which display data within it. It defines a database as a collection of objects which stores both data members values and operations. Ex: PostgreSQL.

## Database Architecture in DBMS: 1-Tier, 2-Tier and 3-Tier

## A Database Architecture is a representation of DBMS design. It helps to design, develop, implement, and maintain the database management system. A DBMS architecture allows dividing the database system into individual components that can be independently modified, changed, replaced, and altered. It also helps to understand the components of a database.

## A [Database](https://www.guru99.com/introduction-to-database-sql.html) stores critical information and helps access data quickly and securely. Therefore, selecting the correct Architecture of DBMS helps in easy and efficient data management.

## 

## Types of DBMS Architecture

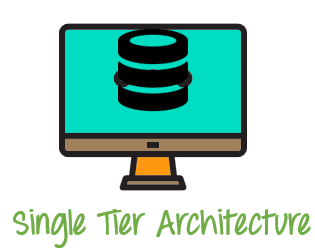
There are mainly three types of DBMS architecture:

* One Tier Architecture (Single Tier Architecture)
* Two Tier Architecture
* Three Tier Architecture

Now, we will learn about different architecture of DBMS with diagram.

## 1-Tier Architecture

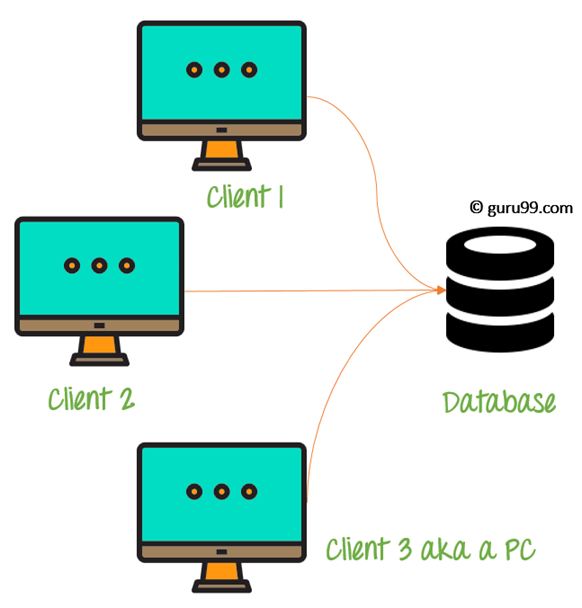
**1 Tier Architecture** in DBMS is the simplest architecture of Database in which the client, server, and Database all reside on the same machine. A simple one tier architecture example would be anytime you install a Database in your system and access it to practice SQL queries. But such architecture is rarely used in production.

[](https://www.guru99.com/images/1/091318_0745_DBMSArchite1.png)

1 Tier Architecture Diagram

## 2-Tier Architecture

A **2 Tier Architecture** in DBMS is a Database architecture where the presentation layer runs on a client (PC, Mobile, Tablet, etc.), and data is stored on a server called the second tier. Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.

[](https://www.guru99.com/images/1/091318_0745_DBMSArchite2.png)

2 Tier Architecture Diagram

In the above 2 Tier client-server architecture of database management system, we can see that one server is connected with clients 1, 2, and 3.

**Two Tier Architecture Example:**

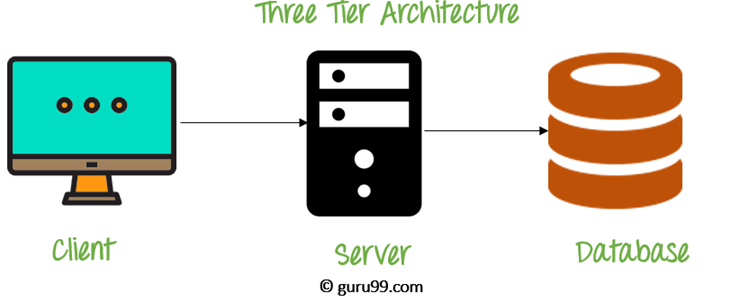
A Contact Management System created using MS- Access.

## 3-Tier Architecture

A **3 Tier Architecture** in DBMS is the most popular client server architecture in DBMS in which the development and maintenance of functional processes, logic, data access, data storage, and user interface is done independently as separate modules. Three Tier architecture contains a presentation layer, an application layer, and a database server.

3-Tier database Architecture design is an extension of the 2-tier client-server architecture. A 3-tier architecture has the following layers:

1. Presentation layer (your PC, Tablet, Mobile, etc.)
2. Application layer (server)
3. Database Server

[](https://www.guru99.com/images/1/091318_0745_DBMSArchite3.png)

3 Tier Architecture Diagram

The Application layer resides between the user and the DBMS, which is responsible for communicating the user's request to the DBMS system and send the response from the DBMS to the user. The application layer(business logic layer) also processes functional logic, constraint, and rules before passing data to the user or down to the DBMS.

### The goal of Three Tier client-server architecture is:

* To separate the user applications and physical database
* To support DBMS characteristics
* Program-data independence
* Supporting multiple views of the data

**Three Tier Architecture Example:**

Any dynamic website on the internet.

**Need for DBMS**

A **D**ata **B**ase **M**anagement **S**ystem is a system software for easy, efficient and reliable data processing and management. It can be used for:

* Creation of a database.
* Retrieval of information from the database.
* Updating the database.
* Managing a database.

It provides us with the many functionalities and is more advantageous than the traditional file system in many ways listed below:

**1) Processing Queries and Object Management**:  
In traditional file systems, we cannot store data in the form of objects. In practical-world applications, data is stored in objects and not files. So in a file system, some application software maps the data stored in files to objects so that can be used further.  
We can directly store data in the form of objects in a database management system. Application level code needs to be written to handle, store and scan through the data in a file system whereas a DBMS gives us the ability to query the database.

**2) Controlling redundancy and inconsistency:**  
Redundancy refers to repeated instances of the same data. A database system provides redundancy control whereas in a file system, same data may be stored multiple times. For example, if a student is studying two different educational programs in the same college, say ,Engineering and History, then his information such as the phone number and address may be stored multiple times, once in Engineering dept and the other in History dept. Therefore, it increases time taken to access and store data. This may also lead to inconsistent data states in both places. A DBMS uses **data normalization** to avoid redundancy and duplicates.

**3) Efficient memory management and indexing:**  
DBMS makes complex memory management easy to handle. In file systems, files are indexed in place of objects so query operations require entire file scans whereas in a DBMS , object indexing takes place efficiently through database schema based on any attribute of the data or a data-property. This helps in fast retrieval of data based on the indexed attribute.

**4) Concurrency control and transaction management:**  
Several applications allow user to simultaneously access data. This may lead to inconsistency in data in case files are used. Consider two withdrawal transactions X and Y in which an amount of 100 and 200 is withdrawn from an account A initially containing 1000. Now since these transactions are taking place simultaneously, different transactions may update the account differently. X reads 1000, debits 100, updates the account A to 900, whereas Y also reads 1000, debits 200, updates A to 800. In both cases account A has wrong information. This results in data inconsistency. A DBMS provides mechanisms to deal with this kind of data inconsistency while allowing users to access data concurrently. A DBMS implements [ACID](http://quiz.geeksforgeeks.org/acid-properties-in-dbms/)(atomicity, durability, isolation,consistency) properties to ensure efficient transaction management without data corruption.

**5) Access Control and ease in accessing data:**  
A DBMS can grant access to various users and determine which part and how much of the data can they access from the database thus removing redundancy. Otherwise in file system, separate files have to be created for each user containing the amount of data that they can access. Moreover, if a user has to extract specific data, then he needs a code/application to process that task in case of file system, e.g. Suppose a manager needs a list of all employees having salary greater than X. Then we need to write business logic for the same in case data is stored in files. In case of DBMS, it provides easy access of data through queries, (e.g., **SELECT** queries) and whole logic need not be rewritten. Users can specify exactly what they want to extract out of the data.

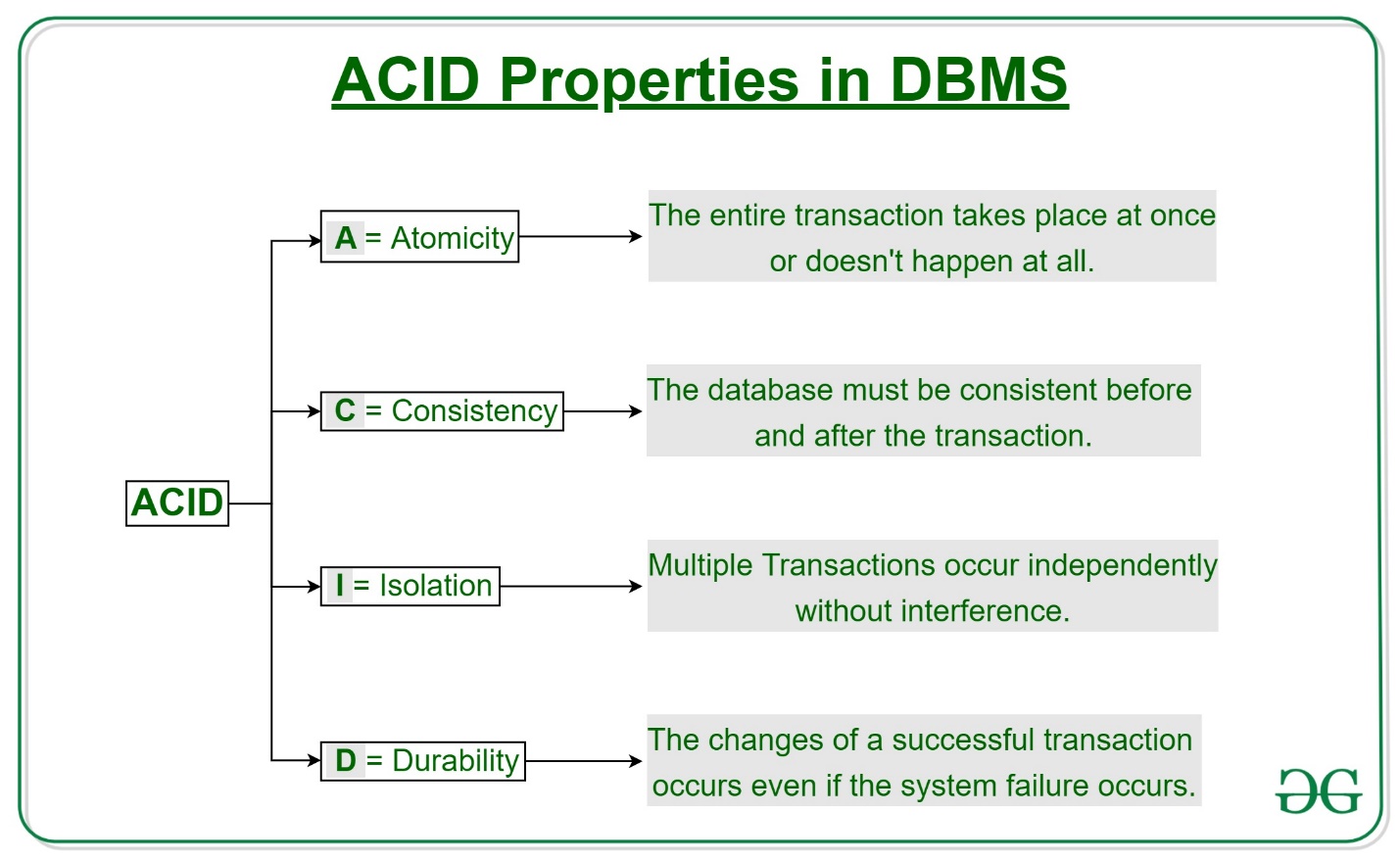
**6) Integrity constraints:** Data stored in databases must satisfy integrity constraints. For example, Consider a database schema consisting of the various educational programs offered by a university such as(B.Tech/M.Tech/B.Sc/M.Sc/BCA/MCA) etc. Then we have a schema of students enrolled in these programs. A DBMS ensures that it is only out of one of the programs offered schema , that the student is enrolled in, i.e. Not anything out of the blue. Hence, database integrity is preserved.

Apart from the above mentioned features a database management also provides the following:

* + **Multiple User Interface**
  + **Data scalability, expandability and flexibility**: We can change schema of the database, all schema will be updated according to it.
  + Overall the time for developing an application is reduced.
  + **Security:** Simplifies data storage as it is possible to assign security permissions allowing restricted access to data.

**ACID Properties in DBMS**

A [**transaction**](https://www.geeksforgeeks.org/sql-transactions/) is a single logical unit of work which accesses and possibly modifies the contents of a database. Transactions access data using read and write operations.   
In order to maintain consistency in a database, before and after the transaction, certain properties are followed. These are called **ACID** properties.



**Atomicity**   
By this, we mean that either the entire transaction takes place at once or doesn’t happen at all. There is no midway i.e. transactions do not occur partially. Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves the following two operations.   
—**Abort**: If a transaction aborts, changes made to database are not visible.   
—**Commit**: If a transaction commits, changes made are visible.   
Atomicity is also known as the ‘All or nothing rule’. 

Consider the following transaction **T** consisting of **T1** and **T2**: Transfer of 100 from account **X** to account **Y**.



If the transaction fails after completion of **T1** but before completion of **T2**.( say, after **write(X)** but before **write(Y)**), then amount has been deducted from **X** but not added to **Y**. This results in an inconsistent database state. Therefore, the transaction must be executed in entirely in order to ensure correctness of database state.

**Consistency**   
This means that integrity constraints must be maintained so that the database is consistent before and after the transaction. It refers to the correctness of a database. Referring to the example above,   
The total amount before and after the transaction must be maintained.   
Total **before T** occurs = **500 + 200 = 700**.   
Total **after T occurs** = **400 + 300 = 700**.   
Therefore, database is **consistent**. Inconsistency occurs in case **T1** completes but **T2** fails. As a result T is incomplete.

**Isolation**   
This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of database state. Transactions occur independently without interference. Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed. This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.   
Let **X**= 500, **Y** = 500.   
Consider two transactions **T** and **T”.**



Suppose **T** has been executed till **Read (Y)** and then **T’’** starts. As a result , interleaving of operations takes place due to which **T’’** reads correct value of **X** but incorrect value of **Y** and sum computed by   
**T’’: (X+Y = 50, 000+500=50, 500)**   
is thus not consistent with the sum at end of transaction:   
**T: (X+Y = 50, 000 + 450 = 50, 450)**.   
This results in database inconsistency, due to a loss of 50 units. Hence, transactions must take place in isolation and changes should be visible only after they have been made to the main memory.

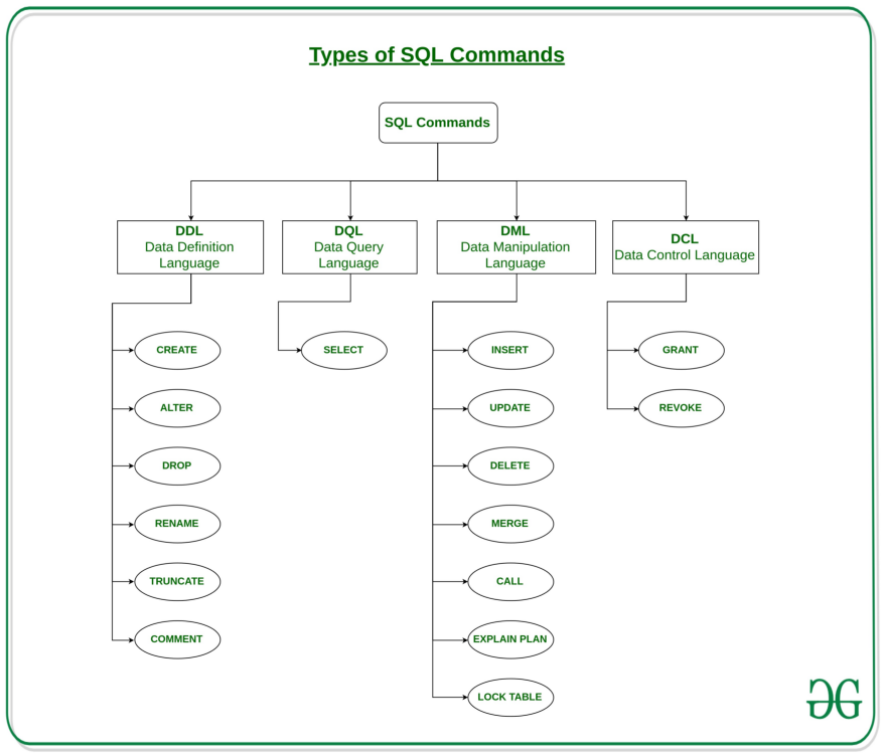
**Durability:**   
This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile memory. The effects of the transaction, thus, are never lost.

The **ACID** properties, in totality, provide a mechanism to ensure correctness and consistency of a database in a way such that each transaction is a group of operations that acts a single unit, produces consistent results, acts in isolation from other operations and updates that it makes are durably stored.

**SQL**

**Tuple** − A single row of a table, which contains a single record for that relation is called a **tuple**

**DDL, DQL, DML, DCL and TCL Commands**



1. **DDL(Data Definition Language) : No Rollback possible**

DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in the database.

**Examples of DDL commands:**

* + [**CREATE**](https://www.geeksforgeeks.org/sql-create/) – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
  + [**DROP**](https://www.geeksforgeeks.org/sql-drop-truncate/) – is used to delete objects from the database.
  + [**ALTER**](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/)-is used to alter the structure of the database.
  + [**TRUNCATE**](https://www.geeksforgeeks.org/sql-drop-truncate/)–is used to remove all records from a table, including all spaces allocated for the records are removed.
  + [**COMMENT**](https://www.geeksforgeeks.org/sql-comments/) –is used to add comments to the data dictionary.
  + [**RENAME**](https://www.geeksforgeeks.org/sql-alter-rename/)–is used to rename an object existing in the database.

1. **DQL (Data Query Language) :** DQL statements are used for performing queries on the data within schema objects. The purpose of the DQL Command is to get some schema relation based on the query passed to it.

**Example of DQL:**

[**SELECT**](https://www.geeksforgeeks.org/sql-select-clause/) – is used to retrieve data from the database.

1. **DML(Data Manipulation Language): Rollback possible**

The SQL commands that deals with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

**Examples of DML:**

* + [**INSERT**](https://www.geeksforgeeks.org/sql-insert-statement/) – is used to insert data into a table.
  + [**UPDATE**](https://www.geeksforgeeks.org/sql-update-statement/) – is used to update existing data within a table.
  + [**DELETE**](https://www.geeksforgeeks.org/sql-delete-statement/) – is used to delete records from a database table.

1. **DCL(Data Control Language):**DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions and other controls of the database system.

**Examples of DCL commands:**

* + **GRANT**-gives user’s access privileges to the database.
  + **REVOKE**-withdraw user’s access privileges given by using the GRANT command.

1. **TCL(transaction Control Language):**TCL commands deal with the [transaction within the database](https://www.geeksforgeeks.org/sql-transactions/).

**Examples of TCL commands:**

* + COMMIT– commits a Transaction.
  + [ROLLBACK](https://www.geeksforgeeks.org/sql-transactions/)– rollbacks a transaction in case of any error occurs.
  + SAVEPOINT–sets a savepoint within a transaction.
  + SET TRANSACTION–specify characteristics for the transaction.

Delete VS Truncate VS Delete

<https://www.c-sharpcorner.com/blogs/difference-between-truncate-delete-and-drop-in-sql-server1>

**Instances, Schema and Sub Schema In DBMS**

### **Instances in DBMS**

In simple words, it is the snapshot of the database taken at a particular moment. It can also be described in more significant way as the collection of the information stored in the database at that particular moment. Instance can also be called as the database state or current set of occurrence due the fact that it is information that is present at the current state.

Every time we update the state say we insert, delete or modify the value of the data item in the record, it changes from one state to other. At the given time, each schema has its own set of instances.

Lets take an**example** to understand in a much better way,

An organization with an employees database will have three different instances such as production that is used to monitor the data right at that moment, per-production that is used to test new functionality prior to release of production and the development that is used by database developers to create new functionality.

### **Schema in DBMS**

It is the overall description or the overall design of the database specified during the database design. Important thing to be remembered here is it should not be changed frequently. Basically, it displays the record types(entity),names of data items(attribute) but not the relation among the files.

Interesting point is the values in schema might change but not the structure of schema.

To understand it well, Schema can be assumed as a framework where in the values of data items are to be fitted, these values can be changed but not frame/format of the schema.

Consider the below **two examples of schema for database** stores and discounts

STORES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| store\_name | store\_id | store\_add | city | state | zip\_code |

DISCOUNTS

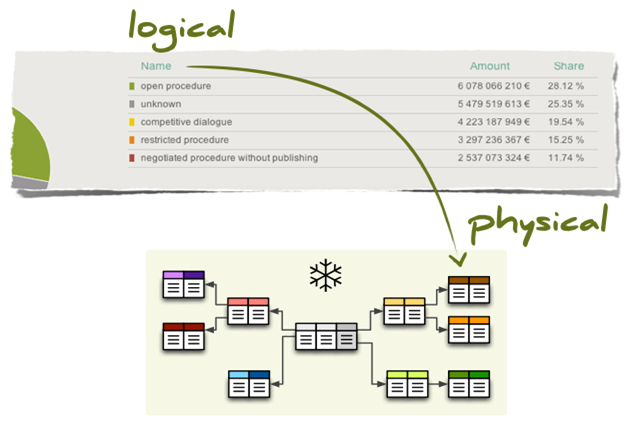
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| discount\_type | store\_id | lowqty | highqty | discount |

The former example shows the schema for stores displaying the name of the store, store id,address,city and state in which it is located and the zip code of respective location.

The latter example is all about schema of discounts that clearly shows the type,id and quality,thus we can now relate to the fact that schema only displays the record types (entities) and  names of data items(attributes) but does not  show the relation among  the files.

Schema can be partitioned as logical schema and physical schema.

Look at the below diagram

[](https://whatisdbms.com/wp-content/uploads/2017/03/Schema.jpg)

Here,former part shows the logical schema which is concerned with the data structure with exploring data structure offered to DBMS so that schema is very easy for the computer to understand.

The latter part that is the physical schema is concerned with the way or the manner in which conceptual database gets represented in the computer as it is stored in the database.Physical schema is hidden behind the logical schema and thus can be be modified without affecting the application programs

Database management system provides data definition language(DDL) and document schema definition language(DSDL) to specify both logical and physical schema.

### **Sub schema in DBMS**

It can be defined as the subset or sub-level of schema that has the same properties as the schema. In simple words it is just a effective plan or the schema for the view. Well, it is interesting to note that it provides the users a window through which the user can view only that part of database which is of matter of interest to him. It Identifies subset of areas, sets, records, data names defined in database that is of interest to him. Thus a portion of database can be seen by application programs and different application programs has different view of data.

Quickly we can summarize the above things, information/data in database at particular moment is known as instance, physical arrangement of data as it appears in database can be defined as schema, and the logical view of data as it appears to the application can be called as sub schema.

**Data models in DBMS:**

<https://afteracademy.com/blog/what-is-data-model-in-dbms-and-what-are-its-types>

**What is Data Abstraction in DBMS and what are its three levels?**

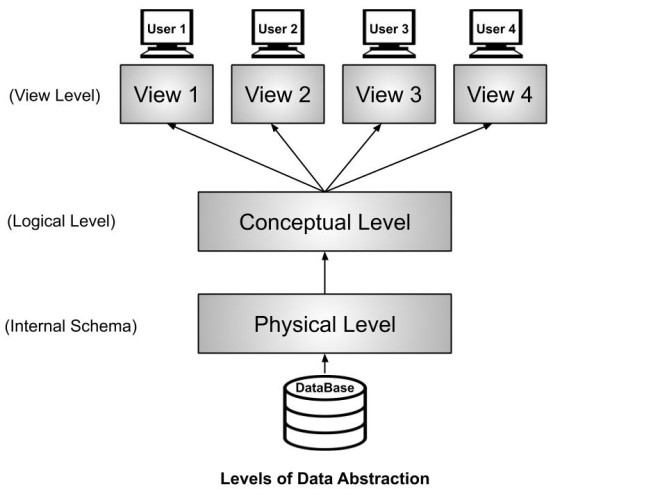
Have you ever wondered how the same website has different views for different users? For example, a college website has a different view for student, faculty and the dean. A student will see the details of his/her attendance, homework, etc. While a faculty will see his/her class time-table and all information that is related to a faculty. We see only that much amount of data which is necessary and other data is hidden from us. So, what is this phenomenon called? Yes, you got it right. This phenomenon is called data abstraction. In this blog, we will learn about data abstraction and we will also see the three levels of abstraction in DBMS. So, let's get started.

#### Data Abstraction

Data Abstraction refers to the process of hiding irrelevant details from the user. So, what is the meaning of irrelevant details? Let's understand this with one example. ***Example:*** If we want to access any mail from our Gmail then we don't know where that data is physically stored i.e is the data present in India or USA or what data model has been used to store that data? We are not concerned about these things. We are only concerned with our email. So, information like these i.e. location of data and data models are irrelevant to us and in data abstraction, we do this only. Apart from the location of data and data models, there are other factors that we don't care of. We hide the unnecessary data from the user and this process of hiding unwanted data is called Data Abstraction.

There are mainly three levels of data abstraction and we divide it into three levels in order to achieve *Data Independence*. Data Independence means users and data should not directly interact with each other. The user should be at a different level and the data should be present at some other level. By doing so, Data Independence can be achieved. So, let's see in details what are these three levels of data abstraction:

1. View Level
2. Conceptual Level
3. Physical Level



#### View Level or External Schema

This level tells the application about how the data should be shown to the user. ***Example:***If we have a login-id and password in a university system, then as a student, we can view our marks, attendance, fee structure, etc. But the faculty of the university will have a different view. He will have options like salary, edit marks of a student, enter attendance of the students, etc. So, both the student and the faculty have a different view. By doing so, the security of the system also increases. In this example, the student can't edit his marks but the faculty who is authorized to edit the marks can edit the student's marks. Similarly, the dean of the college or university will have some more authorization and accordingly, he will has his view. So, different users will have a different view according to the authorization they have.

#### Conceptual Level or Logical Level

This level tells how the data is actually stored and structured. We have different data models by which we can store the data(You can read more about the different types of data model from [here](https://afteracademy.com/blog/what-is-data-model-in-dbms-and-what-are-its-types)). ***Example***: Let us take an example where we use the relational model for storing the data. We have to store the data of a student, the columns in the student table will be student\_name, age, mail\_id, roll\_no etc. We have to define all these at this level while we are creating the database. Though the data is stored in the database but the structure of the tables like the student table, teacher table, books table, etc are defined here in the conceptual level or logical level. Also, how the tables are related to each other are defined here. Overall, we can say that we are creating a blueprint of the data at the conceptual level.

#### Physical Level or Internal Schema

As the name suggests, the Physical level tells us that where the data is actually stored i.e. it tells the actual location of the data that is being stored by the user. The Database Administrators(DBA) decide that which data should be kept at which particular disk drive, how the data has to be fragmented, where it has to be stored etc. They decide if the data has to be centralized or distributed. Though we see the data in the form of tables at view level the data here is actually stored in the form of files only. It totally depends on the DBA, how he/she manages the database at the physical level.

**Referential Integrity Rule in RDBMS**

Referential Integrity Rule in DBMS is based on Primary and Foreign Key. The Rule defines that a foreign key have a matching primary key. Reference from a table to another table should be valid.

**Referential Integrity Rule example** −

**<Employee>**

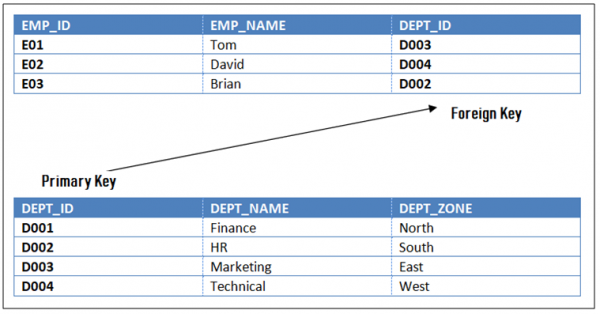
|  |  |  |
| --- | --- | --- |
| **EMP\_ID** | EMP\_NAME | **DEPT\_ID** |

**<Department>**

|  |  |  |
| --- | --- | --- |
| **DEPT\_ID** | DEPT\_NAME | DEPT\_ZONE |

The rule states that the **DEPT\_ID** in the Employee table has a matching valid **DEPT\_ID** in the **Department**table.

To allow join, the referential integrity rule states that the Primary Key and Foreign Key have same data types.



PostgreSQL : <https://www.guru99.com/postgresql-tutorial.html>