**1. Database**: Database is a collection of inter-related data which helps in efficient retrieval, insertion and deletion of data from database and organizes the data in the form of tables, views, schemas, reports etc.

**2. Database Management System**: The software which is used to manage database is called Database Management System (DBMS). For Example, MySQL, Oracle etc. are popular commercial DBMS used in different applications. DBMS allows users the following tasks:

**Data Definition**: It helps in creation, modification and removal of definitions that define the organization of data in database.

**Data Updating**: It helps in insertion, modification and deletion of the actual data in the database.

**Data Retrieval:** It helps in retrieval of data from the database which can be used by applications for various purposes.

**User Administration:** It helps in registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity (completeness or wholeness of data), dealing with concurrency control and recovering information corrupted by unexpected failure.

**3. Shift from File System to DBMS**

File System manages data using files in hard disk. Users are allowed to create, delete, and update the files according to their requirement. Let us consider the example of file based University Management System. Data of students is available to their respective Departments, Academics Section, Result Section, Accounts Section, Hostel Office etc. Some of the data is common for all sections like Roll No, Name, Father Name, Address and Phone number of students but some data is available to a particular section only like Hostel allotment number which is a part of hostel office.

Let us discuss **the issues with this system**:

**Redundancy of data:**

**Inconsistency of Data**: Data is said to be inconsistent if multiple copies of same data does not match with each other. If Phone number is different in Accounts Section and Academics Section, it will be inconsistent. Inconsistency may be because of typing errors or not updating all copies of same data.

**Difficult Data Access:** A user should know the exact location of file to access data.

**Unauthorized Access**

**No Concurrent Access**: The access of same data by multiple users at same time is known as concurrency.

**No Backup and Recovery:**

**4. Data organization in DBMS (3 tire model of DBMS):**

the data organization in a DBMS may be viewed as an aggregation of following levels of organization:

**Physical Level of data organization**: It defines the location where the data base is stored and maintained.

**Conceptual Level of Data organization**: organization of data in the form of tables, records etc.

**External level of data organization**: determines the view of data. Different views may be generated for different users depending upon their need.

**data independence** is required to be maintained at physical and conceptual level of organization. i.e. change in location of data base or the conceptual schema of data organization should not alter/ change the external level of data organization.

**DATA ABSTRACTION IN DBMS:**

ABSTRACTION means hiding the complexities of the database in order to make it easily accessible to the user.

data abstraction is needed at the three levels. they are:

**data abstraction at physical level**: defines how the data is stored and how can it be accessed (randomly/sequentially).

**data abstraction at conceptual level**: it defines that the complex organizational schema of the database should be hidden from the user.

**data abstraction at view/ external level**: complex view of data should be hidden from the user. It should appear to user in a way that it is easy to read, retrieve and understand.

**5. Advantages of DBMS:**

**2-TIRE ARCHITECTURE OF DBMS:**

consist of:

**Client application**

**Server application**

(Interaction b/w the two by means of API)

**7. E-R model:**

The pictorial representation of the interconnected data and relationship between them is called E-R model. Based on real life data.

Entity (any object of importance): represented by rectangle. Entity may have attributes of following types:

1 attribute: represented by ellipse.

2. multi-valued attributes: represented by double elipse.

3. primary key: underlined or bold.

8**. DDL:** (data definition language/ data description language);

The programming language that is used to define the data format and structure using which the database is maintained is called DDL.

Commonly used keywords are**: Create, Truncate**(Removes the content of all rows), **Alter, Rename, Drop**, and **Describe**. **Delete** (removes all the rows).

9. **DML**(Data Manipulation Language): Language used to insert delete or update the data in data base.

10. **Normalization** is the process of making the data perfect, by removing all kinds of redundancy from it.

First normal form: to eliminate the duplicate columns.

If A is functionally dependent on B and B is functionally dependent on C. Then in-directly, A is functionally dependent on C. This is called transitivity. Used in third normal form.

11. Query: It is a command that directs the database to perform some function.

12. create table TableName(

column\_name data\_type Primary key(if)

……………

);

15. insert into Table\_name values(‘value1’,’val2’,….);

16. update TableName set Column1=’value’, Column2=’value’; ///where clause optional.

17. delete from TableName where CoulmnName=’this’;///////////where clause.

18. alter TableName add/remove ColumnName.

19. **Relational Database Management System: (RDBMS)** The DBMS that maintains database records in a table maintained along with indices.

20. **Trigger** arises when Insert, Delete or Update query are called.

21. **SQL:** structured Query language.

22. **Distributed database**: Central server contains all the data that can be accessed from more than one node.

23. Three types of command in SQL:

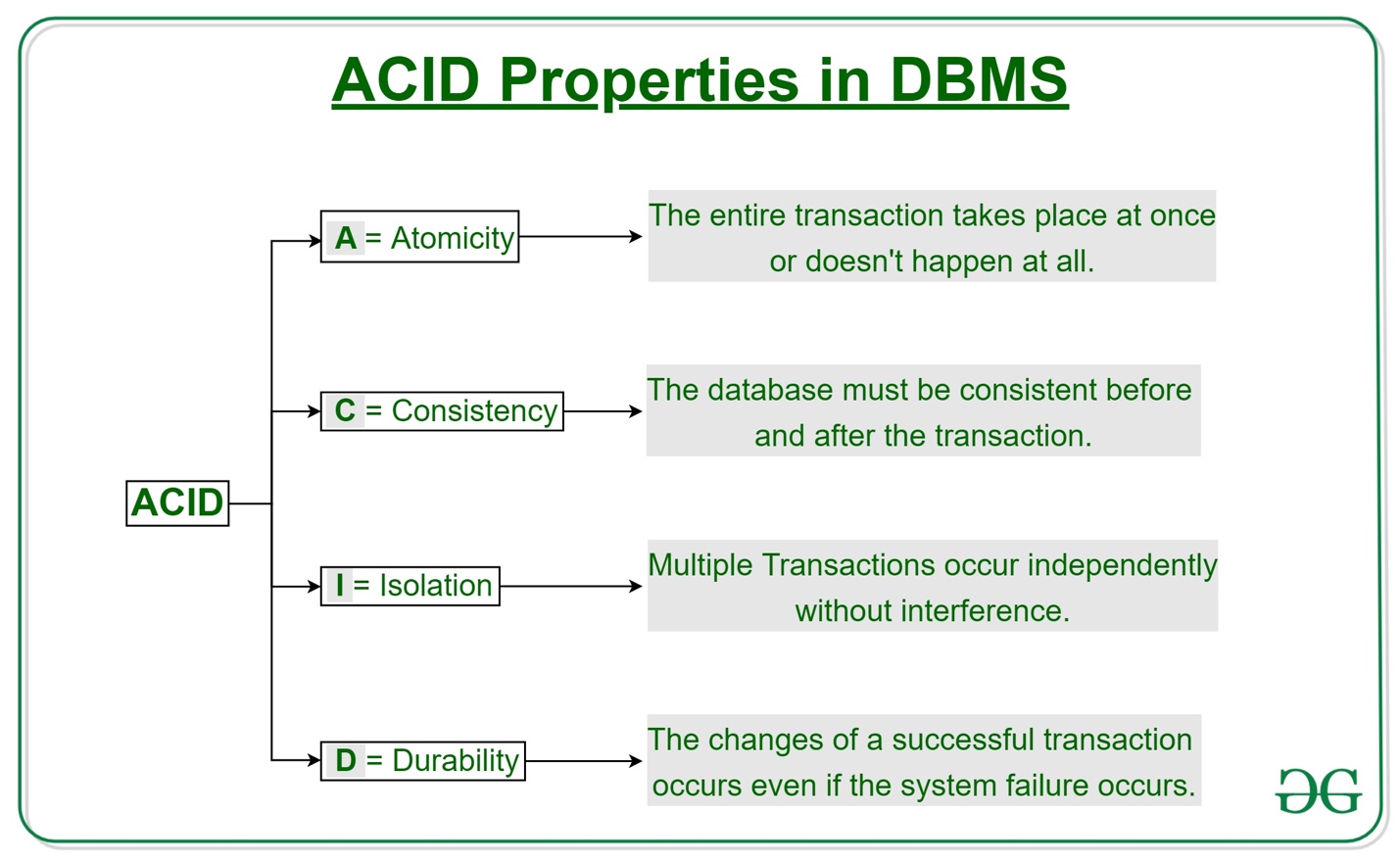
Data Definition commands

Data manipulation commands and

Data control commands (administrator use).

**ACID properties of DB:**

A transaction 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 entirety 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.