LET'S START WITH DBMS :)

ACID Properties

ACID properties are the properties which ensures that transactions are processed reliably and accurately, even in complex situations(sytem failures/network issues)

- **A–> Atomicity (Either execute all operations or none)**
- **C–> Consistency (Read should fetch upto date data and write shouldn't voilate integrity constraints)**
- **I–> Isolation (One transaction should be independent of other transaction)**
- **D–> Durability (The commited transaction should remain even after a failure/crash)**

LET'S START WITH DBMS :)

ACID Properties

A-> Atomicity: It ensures that a transaction is treated as a single unit of work
Either all operations are completed successfully (commit) or none of them are applied (rollback).

This guarantees that the database remains in a consistent state despite any failures or interruptions during the transaction.

Ex :Consider Ram is transferring money to Shyam. The transaction must deduct the amount from the Ram's account and add it to the Shyam's account as a single operation.
If at any moment or at any part, this transaction fails (e.g., due to insufficient funds/system error/network error), the entire transaction is rolled back, ensuring that none of the accounts is affected partially.

LET'S START WITH DBMS :)

ACID Properties

C-> Consistency: It ensures that
1.**Read operations** retrieve consistent and up-to-date data from the database, and
2.**Write operations** ensure that data modifications maintain database constraints(such as foreign key relationships or unique constraints so that that data remains accurate)

It guarantees that the database remains in a consistent state before and after the execution of each transaction.

Ex: Consider you had 100rs in your account but you want 50rs cash, so you transferred 50rs to a person X and he gave you 50rs cash.
Before transaction- 100rs(in acc)
After transaction- 100rs(50rs in acc+ 50 rs cash)

LET'S START WITH DBMS :)

ACID Properties

I-> Isolation : It ensures that if there are two transactions 1 and 2, then the changes made by Transaction 1 are not visible to Transaction 2 until Transaction 1 commits.

While the transaction is reading data, the dbms ensures that the data is consistent and isolated from other transactions. This means that other transactions cannot modify the data being read by the current transaction until it is committed or rolled back.

Ex : A= 40rs (in DB)
Transaction 1 : Update value of A to 50rs
Transaction 2 : Read/get/Fetch value of A

If Transaction 1 is committed acc to Transaction 2 the value of A=50rs
If Transaction 1 is PENDING/RUNNING acc to Transaction 2 the value of A=40rs

LET'S START WITH DBMS :)

ACID Properties

D-> Durability : It ensures that once you save your data (commit a transaction), it stays saved, even if the system crashes or there is a power failure. Your data is always safe and won't disappear after you save as committed transactions are not lost.

Most DBMS use a technique called Write-Ahead Logging (WAL) to ensure durability. Before modifying data in the database, the DBMS writes the changes to a transaction log (often stored on disk) in a sequential manner. This ensures that if there is a failure event, the database can recover to a consistent state.

Ex : Consider if your are transferring 100rs to your friend and there is a sudden power outage or the system crashes right after the transaction is committed, the changes (the transfer of 100) will still be saved in the database. When the system is back up, both your account and your friend's account will reflect the updated balances.