ACID Properties

1. Transaction

A **transaction** is a logical unit of work performed on a database.

- It executes a set of SQL statements in a proper sequence.
- Sequence matters — because order of operations can change results.
- A transaction either **completes fully** (all changes saved permanently) or **fails completely** (all changes rolled back).

Example
Suppose you transfer ₹1000 from Account A to Account B.

Steps involved:

1. Debit ₹1000 from A
2. Credit ₹1000 to B

Both steps must execute **successfully**, then only transaction is complete.
If electricity goes off after step 1, money must not disappear — entire operation should rollback.

2. ACID Properties

To maintain data correctness and reliability, a transaction must follow **ACID** properties.

A → Atomicity

- All steps of a transaction must execute completely or not at all.
- No partial completion is allowed.

Example:
If ₹1000 is deducted from A but not added to B due to failure, rollback happens and money returns to A.
Partial success is not allowed — either **full success or complete undo.**

C → Consistency

- Before and after a transaction, database must remain valid.
- All integrity rules should remain intact.

Example:
Total money in the bank before and after transfer should remain same.
System should not create or lose money magically — rules must be preserved.

I → Isolation

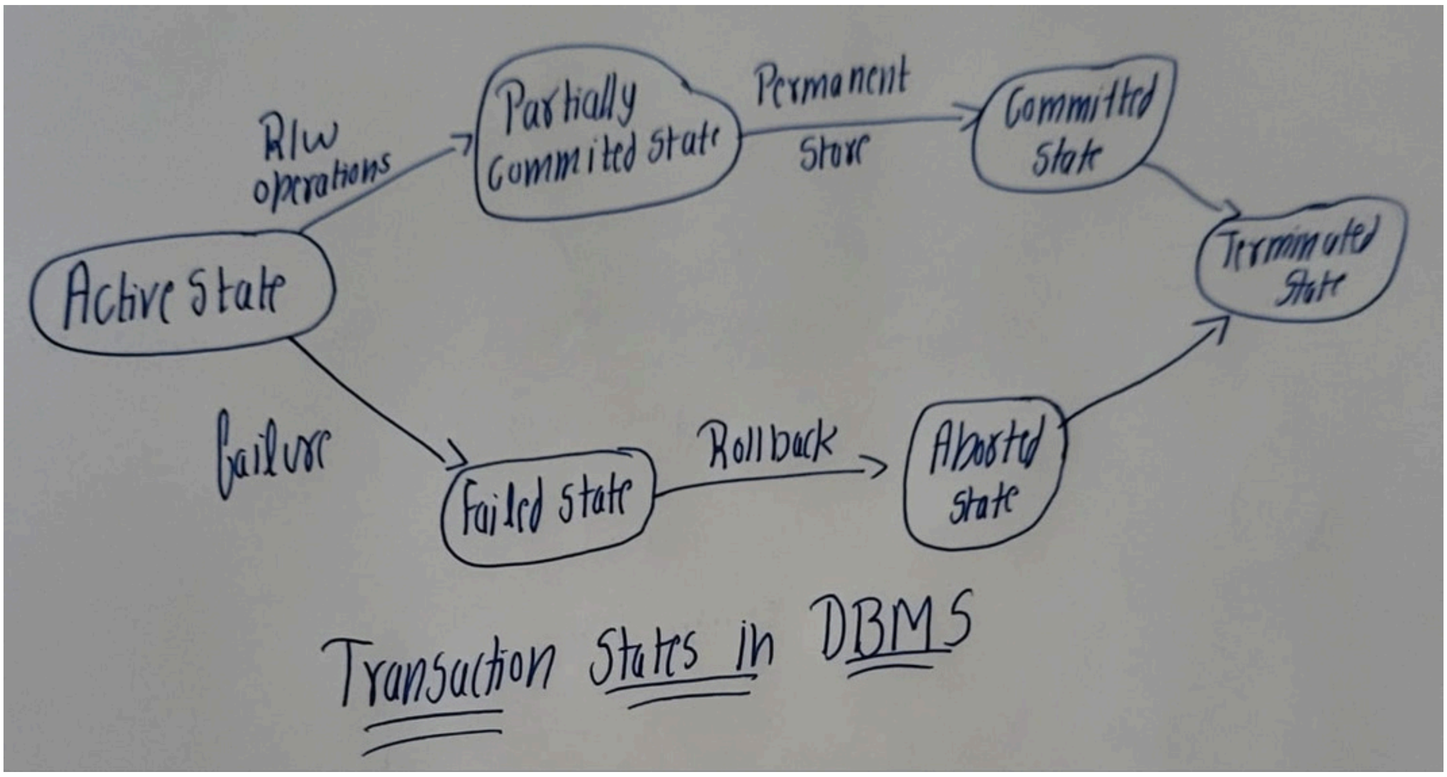
- Even if many transactions run at the same time, each transaction should feel like it is running alone.
- Parallel transactions should not interfere with each other.

Example:
Two people booking the last train seat together:
Even if they click at same time, DB ensures only **one booking succeeds**, the other gets failed message.
Both transactions run independently without corrupting data.

D → Durability

- Once a transaction is completed successfully, its changes are permanent.
- Even if power fails or system crashes, the committed data remains safe.

Example:
If money transfer is completed and receipt is generated,
even if server restarts after 2 seconds — money will still be credited.
No loss of committed data.



1. Active State

- This is the **starting phase of a transaction**.
- All read and write operations happen here.
- If everything runs properly → it moves to **Partially Committed** state.
- If any error occurs → it moves to the **Failed** state.

Example: Money is being deducted from your account – process is active.

2. Partially Committed State

- Here, the transaction has finished executing all steps.
- Changes are temporarily saved in **main memory (buffer)**.
- If these changes are successfully written to database → it becomes **Committed**.
- If something goes wrong → it goes to **Failed State**.

Example: Money has been deducted from A but not yet added permanently to B.

3. Committed State

- All changes become **permanent** in the database.
- Rollback is **not possible** after this.
- Database is now in a **new consistent state**.

Example: Money is deducted from A and added to B successfully – transaction completed.

4. Failed State

- This happens when a transaction faces an error before completion.
- Execution cannot continue further from here.

Example: Server crashed after deducting money but before crediting – failure occurred.

5. Aborted State

- After reaching Failed State, system **rolls back** all changes.
- Database returns to the state **before the transaction started**.
- After rollback completes, the transaction is called **Aborted**.

Example: Money deduction reversed → both accounts return to original balance.

6. Terminated State

- Final stage of transaction.
- A transaction reaches this state if it **committed successfully** or **aborted after rollback**.

Example: Either transfer completed or cancelled – process is fully closed.