Transaction property

- The transaction has the four properties.
- These are used to maintain consistency in a database, before and after the transaction.

Property of Transaction

1. Atomicity Consistency
2. Isolation Durability

Atomicity

- It states that all operations of the transaction take place at once if not, the transaction is aborted.
- There is no midway, i.e., the transaction cannot occur partially.
- Each transaction is treated as one unit and either run to completion or is not executed at all.

Atomicity involves the following two operations:

Abort: If a transaction aborts then all the changes made are not visible.

Commit: If a transaction commits then all the changes made are visible.

In order to ensure correctness of database state, the transaction must be executed in entirety.

Consistency

- The integrity constraints are maintained so that the database is consistent before and after the transaction.
- The execution of a transaction will leave a database in either its prior stable state or a new stable state.
- The consistent property of database states that every transaction sees a consistent database instance.
- The transaction is used to transform the database from one consistent state to another consistent state.

Isolation

- It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.

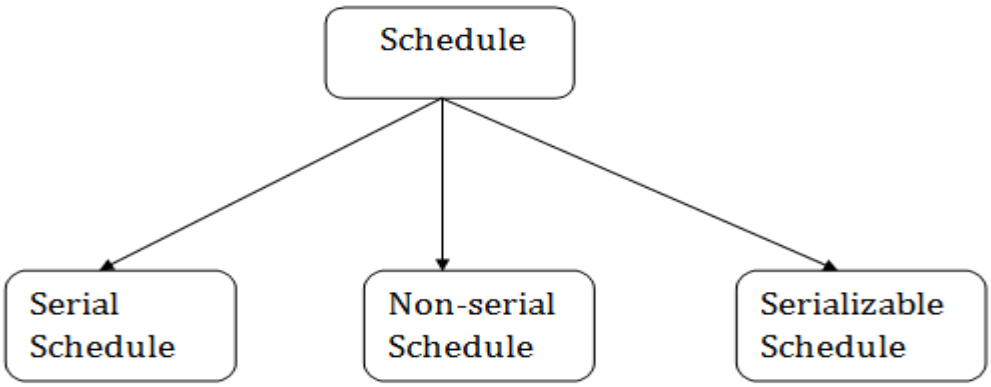
- In isolation, if the transaction T1 is being executed and using the data item X, then that data item can't be accessed by any other transaction T2 until the transaction T1 ends.
- The concurrency control subsystem of the DBMS enforced the isolation property.

Durability

- The durability property is used to indicate the performance of the database's consistent state.
- It states that the transaction made the permanent changes.
- They cannot be lost by the erroneous operation of a faulty transaction or by the system failure.
- When a transaction is completed, then the database reaches a state known as the consistent state.
- That consistent state cannot be lost, even in the event of a system's failure.
- The recovery subsystem of the DBMS has the responsibility of Durability property.

Schedule

- A series of operation from one transaction to another transaction is known as schedule.
- It is used to preserve the order of the operation in each of the individual transaction.



Serial Schedule

- It is a type of schedule where one transaction is executed completely before starting another transaction.
- In the serial schedule, when the first transaction completes its cycle, then the next transaction is executed.

Non-serial Schedule

- If interleaving of operations is allowed, then there will be non-serial schedule.
- It contains many possible orders in which the system can execute the individual operations of the transactions.

Serializable schedule

- The serializability of schedules is used to find non-serial schedules that allow the transaction to execute concurrently without interfering with one another.
- It identifies which schedules are correct when executions of the transaction have interleaving of their operations.
- A non-serial schedule will be serializable if its result is equal to the result of its transactions executed serially.

Failure Classification

Transaction failure System crash Disk failure

Transaction failure

- The transaction failure occurs when it fails to execute or when it reaches a point from where it can't go any further.
- If a few transaction or process is hurt, then this is called as transaction failure.

Reasons for a transaction failure:

Logical errors: If a transaction cannot complete due to some code error or an internal error condition, then the logical error occurs.

Syntax error: It occurs where the DBMS itself terminates an active transaction because the database system is not able to execute it.

For example, The system aborts an active transaction, in case of deadlock or resource unavailability.

System Crash

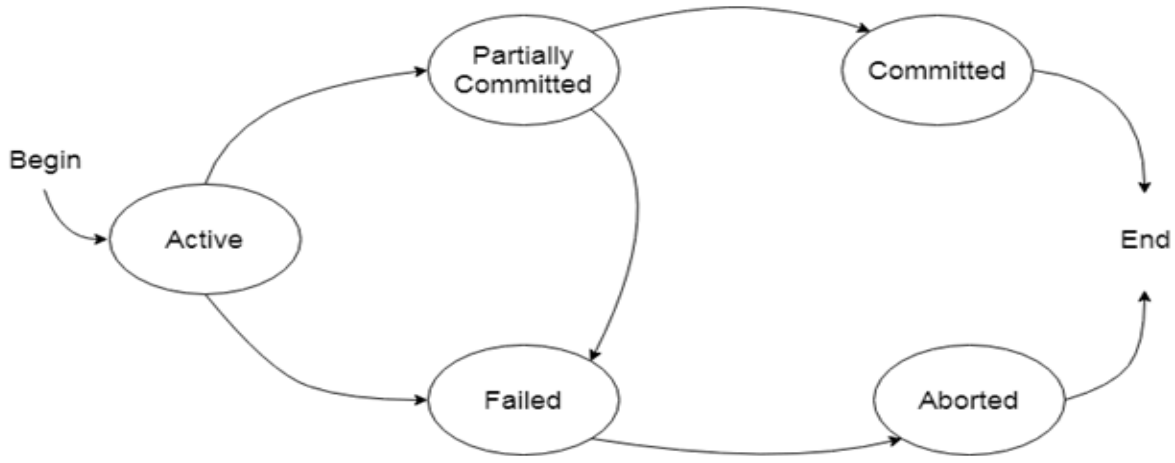
- System failure can occur due to power failure or other hardware or software failure.
  Example: Operating system error.

Fail-stop assumption: In the system crash, non-volatile storage is assumed not to be corrupted.

Disk Failure

- It occurs where hard-disk drives or storage drives used to fail frequently.
- It was a common problem in the early days of technology evolution.
- Disk failure occurs due to the formation of bad sectors, disk head crash, and unreachability to the disk or any other failure, which destroy all or part of disk storage.

States of Transaction



Active state

- The active state is the first state of every transaction.
- In this state, the transaction is being executed.

For example: Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

Partially committed

- In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database.
- In the total mark calculation example, a final display of the total marks step is executed in this state.

Committed

- A transaction is said to be in a committed state if it executes all its operations successfully.
- In this state, all the effects are now permanently saved on the database system.

Failed state

- If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.
- In the example of total mark calculation, if the database is not able to fire a query to fetch the marks, then the transaction will fail to execute.

Aborted

- If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state.
- If not then it will abort or roll back the transaction to bring the database into a consistent state.
- If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.
- After aborting the transaction, the database recovery module will select one of the two operations:

Re-start the transaction Kill the transaction

Denormalization

- Denormalization is a technique used by database administrators to optimize the efficiency of their database infrastructure.

- This method allows us to add redundant data into a normalized database to alleviate issues with database queries that merge data from several tables into a single table.
- The denormalization concept is based on the definition of normalization that is defined as arranging a database into tables correctly for a particular purpose.

> ***NOTE:*** **Denormalization does not indicate not doing normalization. It is an optimization strategy that is used after normalization has been achieved.**

Pros of Denormalization

- Enhance Query Performance
- Make database more convenient to manage
- Facilitate and accelerate reporting

Cons of Denormalization

- It takes large storage due to data redundancy.
- It makes it expensive to updates and inserts data in a table.
- It makes update and inserts code harder to write.
- Since data can be modified in several ways, it makes data inconsistent. Hence, we'll need to update every piece of duplicate data.
- It's also used to measure values and produce reports.

How is denormalization different from normalization?

- Denormalization is a technique used to merge data from multiple tables into a single table that can be queried quickly.

- Normalization, on the other hand, is used to delete redundant data from a database and replace it with non-redundant and reliable data.

- Denormalization is used when joins are costly, and queries are run regularly on the tables.

- Normalization is typically used when a large number of insert/update/delete operations are performed, and joins between those tables are not expensive.