Unit 4 Transaction

Definition
A **transaction** is a set of logically related operations.
A transaction is an action or series of actions that are being performed by a single user or application program, which reads or updates the contents of the database.
For example, you are transferring money from your bank account to your friend's account; the set of operations would be like this:

Simple Transaction Example

1. Read your account balance
2. Deduct the amount from your balance
3. Write the remaining balance to your account
4. Read your friend's account balance
5. Add the amount to his account balance
6. Write the new updated balance to his account

This whole set of operations can be called a transaction. Although I have shown you read, write and update operations in the above example but the transaction can have operations like read, write, insert, update, delete.

Lets say your account is A and your friend's account is B, you are transferring 10000 from A to B, the steps of the transaction are:

a) Read(balance of A)
b) A=A- 10000;
c) Write(A);
d) Read(B)
e) B=B+10000;
f) Write(B)

Transaction failure in between the operations
We should understand what are the problems associated with it.
Problem
That can happen during a transaction is that the transaction can fail before finishing the all the operations in the set.

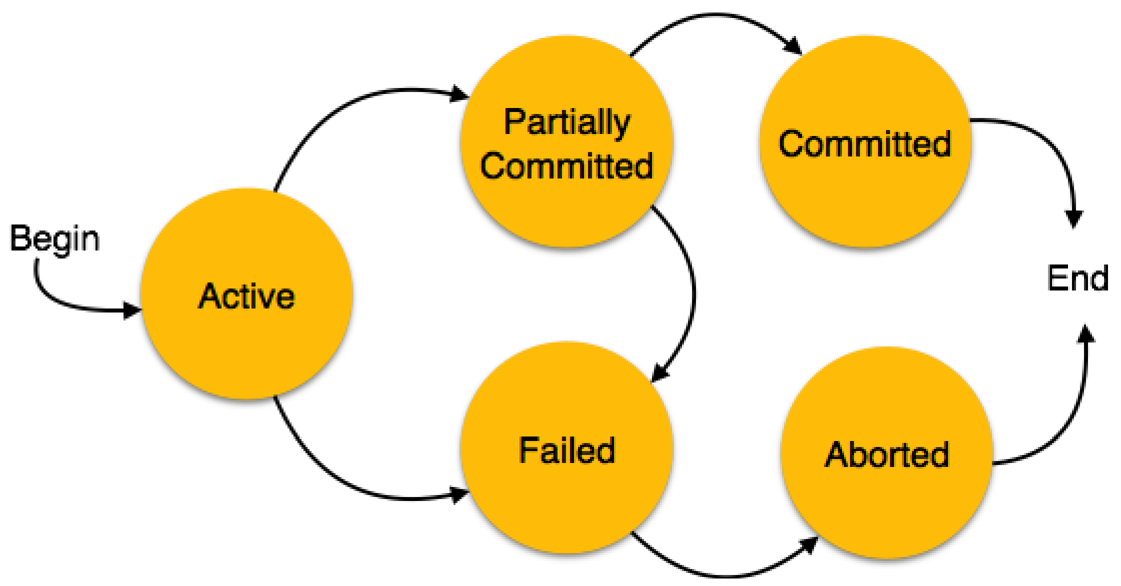
This can happen due to power failure, system crash etc. This is a serious problem that can leave database in an inconsistent state. Assume that transaction fail after (c) operation then the amount would be deducted from your account but your friend will not receive it.

To solve this problem, we have the following two operations
Commit: If all the operations in a transaction are completed successfully then commit those changes to the database permanently.
Rollback: If any of the operation fails then rollback all the changes done by previous operations.

Even though these operations can help us avoiding several issues that may arise during transaction but they are not sufficient when two transactions are running concurrently.
To handle those problems we need to understand database ACID properties.

States of Transactions

A transaction in a database can be in one of the following states –



- **Active** − In this state, the transaction is being executed. This is the initial state of every transaction.
- **Partially Committed** − When a transaction executes its final operation, it is said to be in a partially committed state.
- **Failed** − A transaction is said to be in a failed state if any of the checks made by the database recovery system fails. A failed transaction can no longer proceed further.
- **Aborted** − If any of the checks fails and the transaction has reached a failed state, then the recovery manager rolls back all its write operations on the database to bring the database back to its original state where it was prior to the execution of the transaction. Transactions in this state are called aborted. The database recovery module can select one of the two operations after a transaction aborts −
    - Re-start the transaction
    - Kill the transaction

- **Committed** − If a transaction executes all its operations successfully, it is said to be committed. All its effects are now permanently established on the database system.

Properties of Transaction:

The four basic are in combination termed as ACID properties.

- **Atomicity**: The 'all or nothing' property. A transaction is an indivisible entity that is either performed in its entirety or will not get performed at all. This is the responsibility or duty of the recovery subsystem of the DBMS to ensure atomicity.

- **Consistency**: A transaction must alter the database from one steady-state to another steady state. This is the responsibility of both the DBMS and the application developers to make certain consistency

- **Isolation**: Transactions that are executing independently of one another is the primary concept followed by isolation. In other words, the frictional effects of incomplete transactions should not be visible or come into notice to other transactions going on simultaneously.

.

- **Durability**: The effects of an accomplished transaction are permanently recorded in the database and must not get lost or vanished due to subsequent failure. So this becomes the responsibility of the recovery sub-system to ensure durability.