

UNIT - 2

Transaction States

There are the following **six states** in which a transaction may exist:

- **Active (2) Partially Committed (3) Failed (4) Aborted (5) Committed (6) Terminated**

Active: The initial state when the transaction has just started execution.

Partially Committed: At any given point of time if the transaction is executing properly, then it is going towards its COMMIT POINT. The values generated during the execution are all stored in volatile storage.

Failed: If the transaction fails for some reason. The temporary values are no longer required, and the transaction is set to ROLLBACK. It means that any change made to the database by this transaction up to the point of the failure must be undone. If the failed transaction has withdrawn Rs. 100/- from account A, then the ROLLBACK operation should add Rs 100/- to account A.

Aborted: When the ROLLBACK operation is over, the database reaches the BFIM. The transaction is now said to have been aborted. Two options after it has been aborted:

- restart the transaction (can be done only if no internal logical error)
- kill the transaction

Committed: If no failure occurs then the transaction reaches the COMMIT POINT. All the temporary values are written to the stable storage and the transaction is said to have been committed.

Terminated: Either committed or aborted, the transaction finally reaches this state.

The whole process can be described using the following diagram:

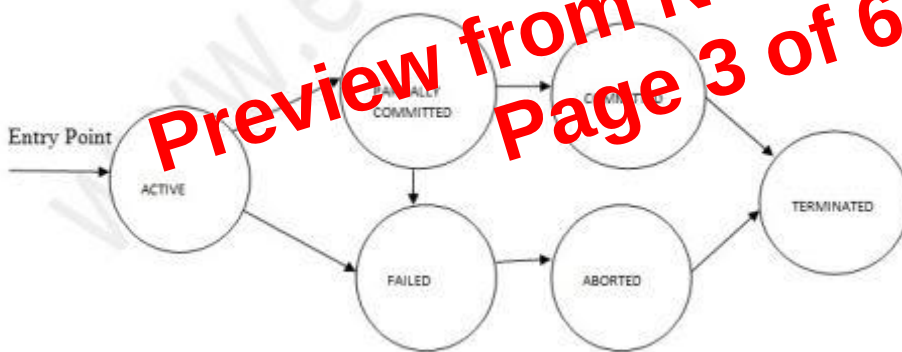


Figure 1-Transaction States

Concurrent Execution

A schedule is a collection of many transactions which is implemented as a unit. Depending upon how these transactions are arranged in within a schedule, a schedule can be of two types:

Serial: The transactions are executed one after another, in a non-preemptive manner.

Concurrent: The transactions are executed in a preemptive, time shared method.

In Serial schedule, there is no question of sharing a single data item among many transactions, because not more than a single transaction is executing at any point of time. However, a serial schedule is inefficient in the sense that the transactions suffer for having a longer waiting time and response time, as well as low amount of resource utilization.