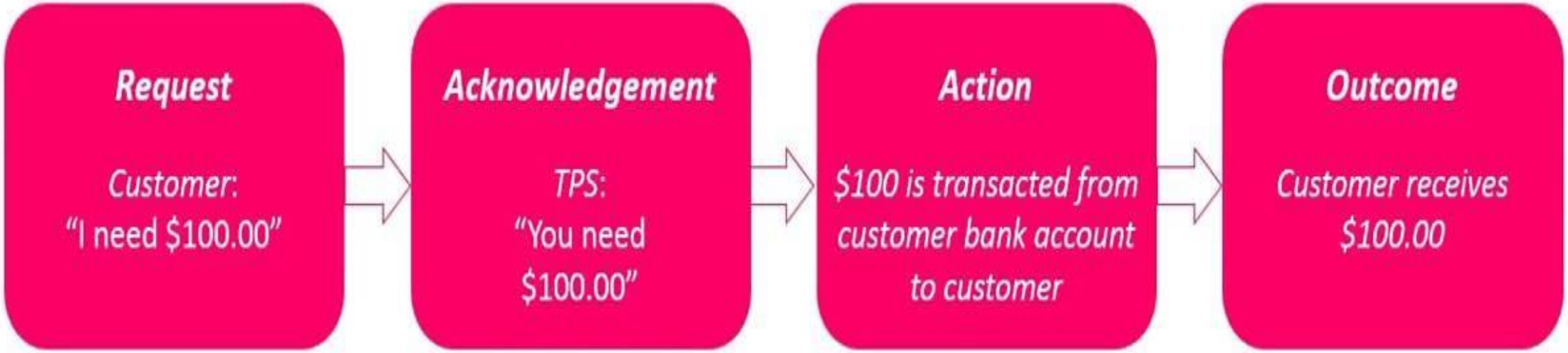
Lecture 38

THE NORTHCAP UNIVERSITY

Transaction Processing

What is a Transaction?

An example of a Financial Transaction:



Request

Customer:
"I need $100.00"

Acknowledgement

TPS:
"You need $100.00"

Action

$100 is transacted from customer bank account to customer

Outcome

Customer receives $100.00

The Transaction Processing System:

Collecting: ATM, EFTPOS, POS Terminal, PC or Mobile Device

Organising & Analysing: DBMS and Server Software

Processing: CPU, Batch & Real Time processing

Storing / Retrieving: HDD, Server, Cloud

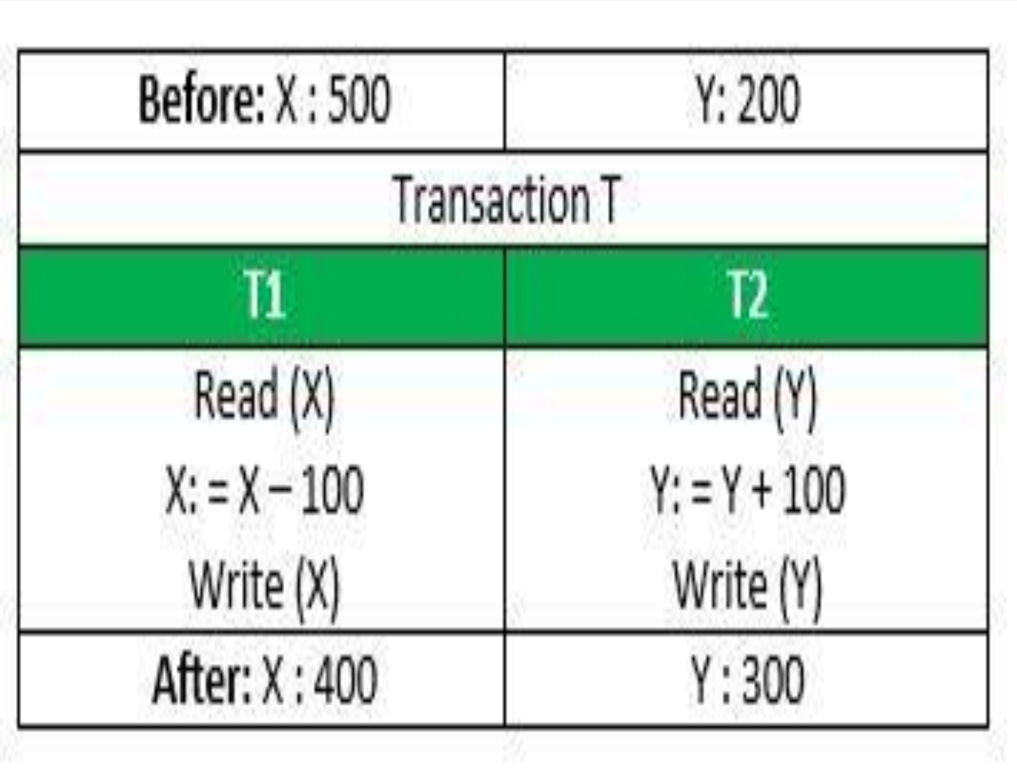
Displaying: ATM, EFTPOS, POS Terminal, PC or Mobile Device

Transmitting / Receiving: Wired and Wireless media

TRANSACTION CONCEPT

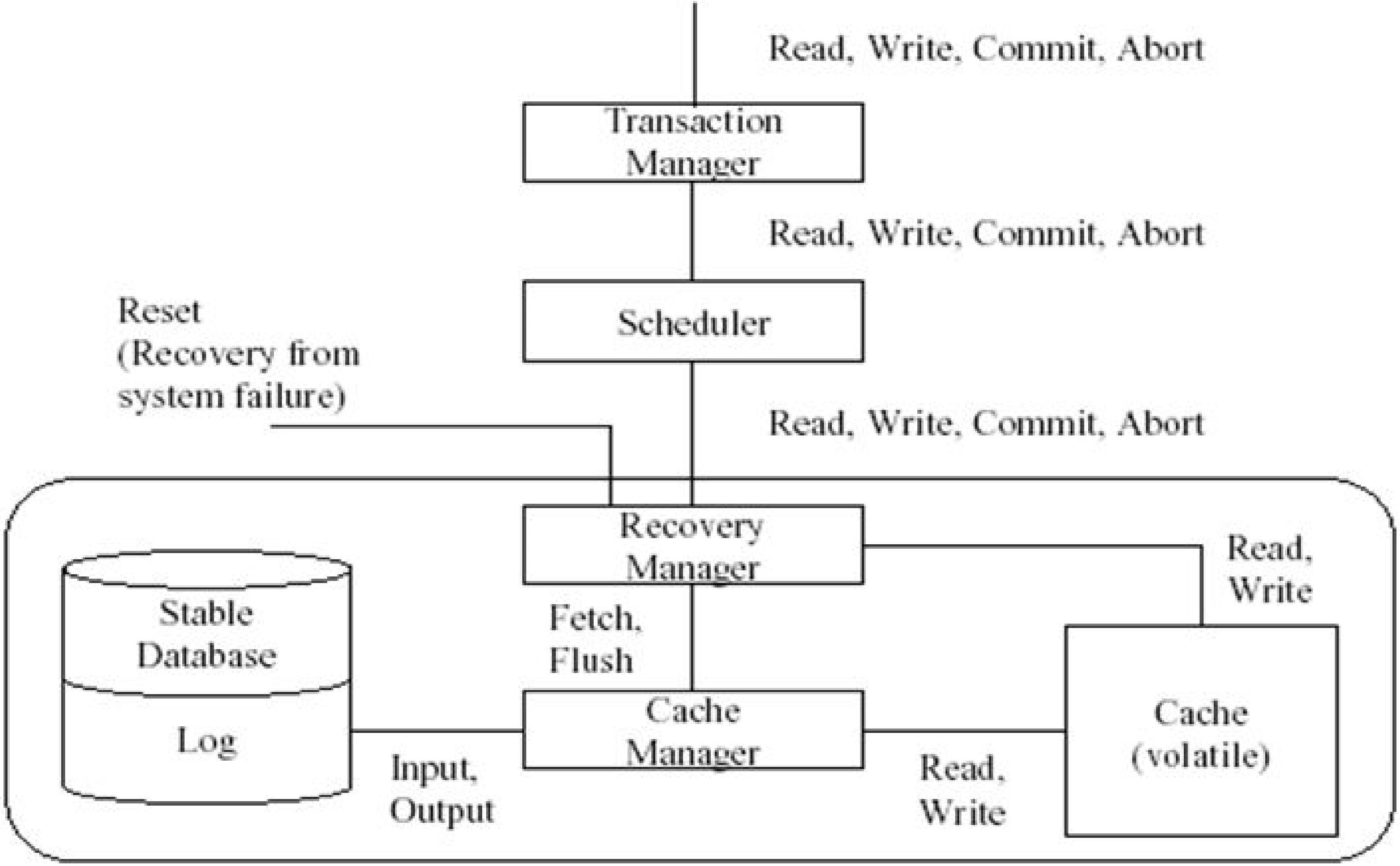
- A **transaction** is a *unit* of program execution that accesses and possibly updates various data items.
- E.g. transaction to transfer $100 from account X to account Y:

1. **read**(X)
2. $X := X - 100$
3. **write**(X)
4. **read**(Y)
5. $Y := Y + 100$
6. **write**(Y)

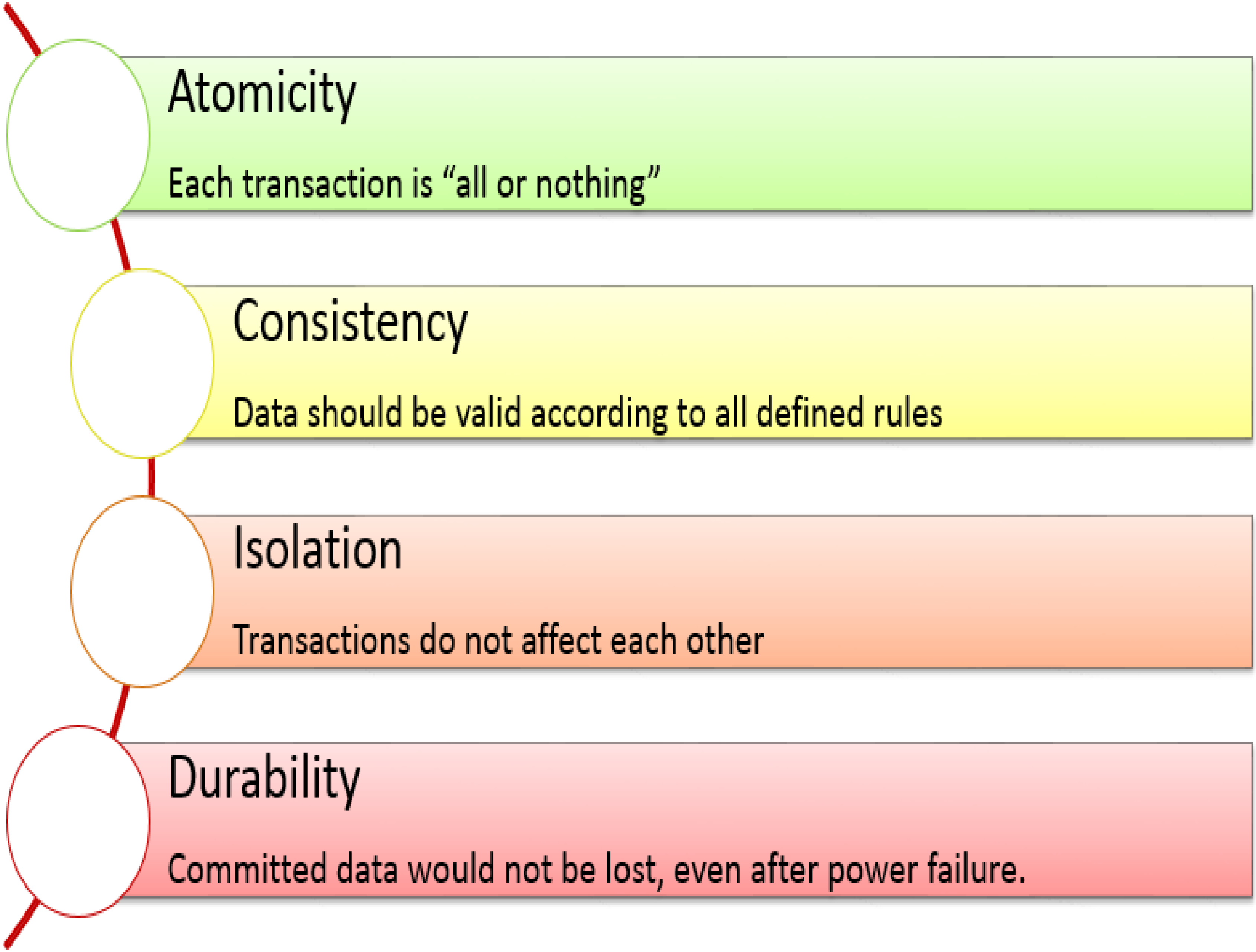


Before: X : 500	Y: 200
Transaction T	
T1	T2
Read (X) X: = X – 100 Write (X)	Read (Y) Y: = Y + 100 Write (Y)
After: X : 400	Y : 300

DBMS Transaction Subsystem



ACID Properties



Atomicity

Each transaction is “all or nothing”

Consistency

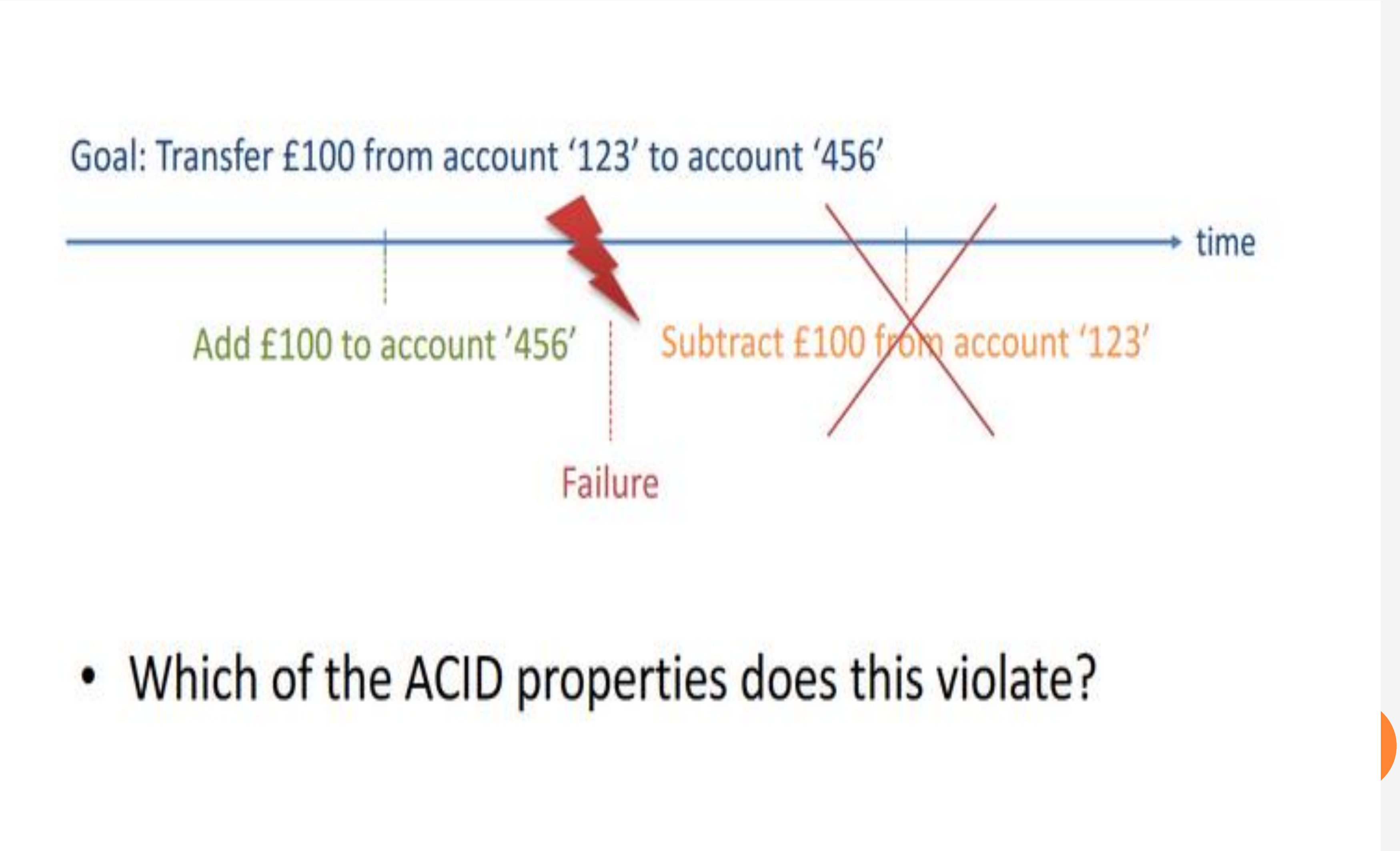
Data should be valid according to all defined rules

Isolation

Transactions do not affect each other

Durability

Committed data would not be lost, even after power failure.



Goal: Transfer £100 from account '123' to account '456'

- Which of the ACID properties does this violate?

Isolation

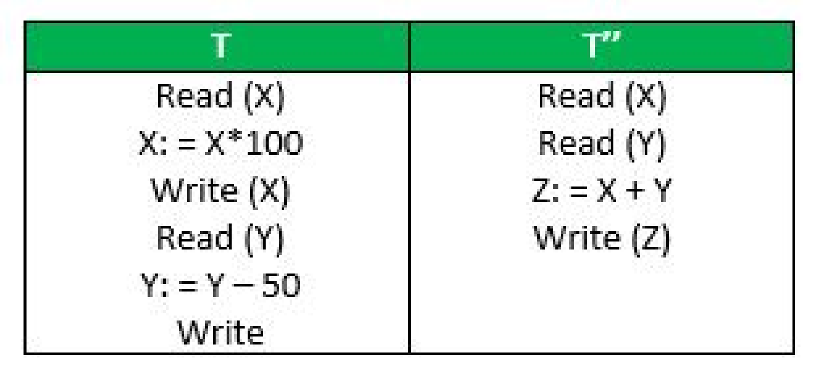
- This property ensures that multiple transactions can occur concurrently without leading to inconsistency of database state.

- Transactions occur independently without interference.

- Changes occurring in a particular transaction will not be visible to any other transaction until that particular change in that transaction is written to memory or has been committed.

- This property ensures that the execution of transactions concurrently will result in a state that is equivalent to a state achieved these were executed serially in some order.

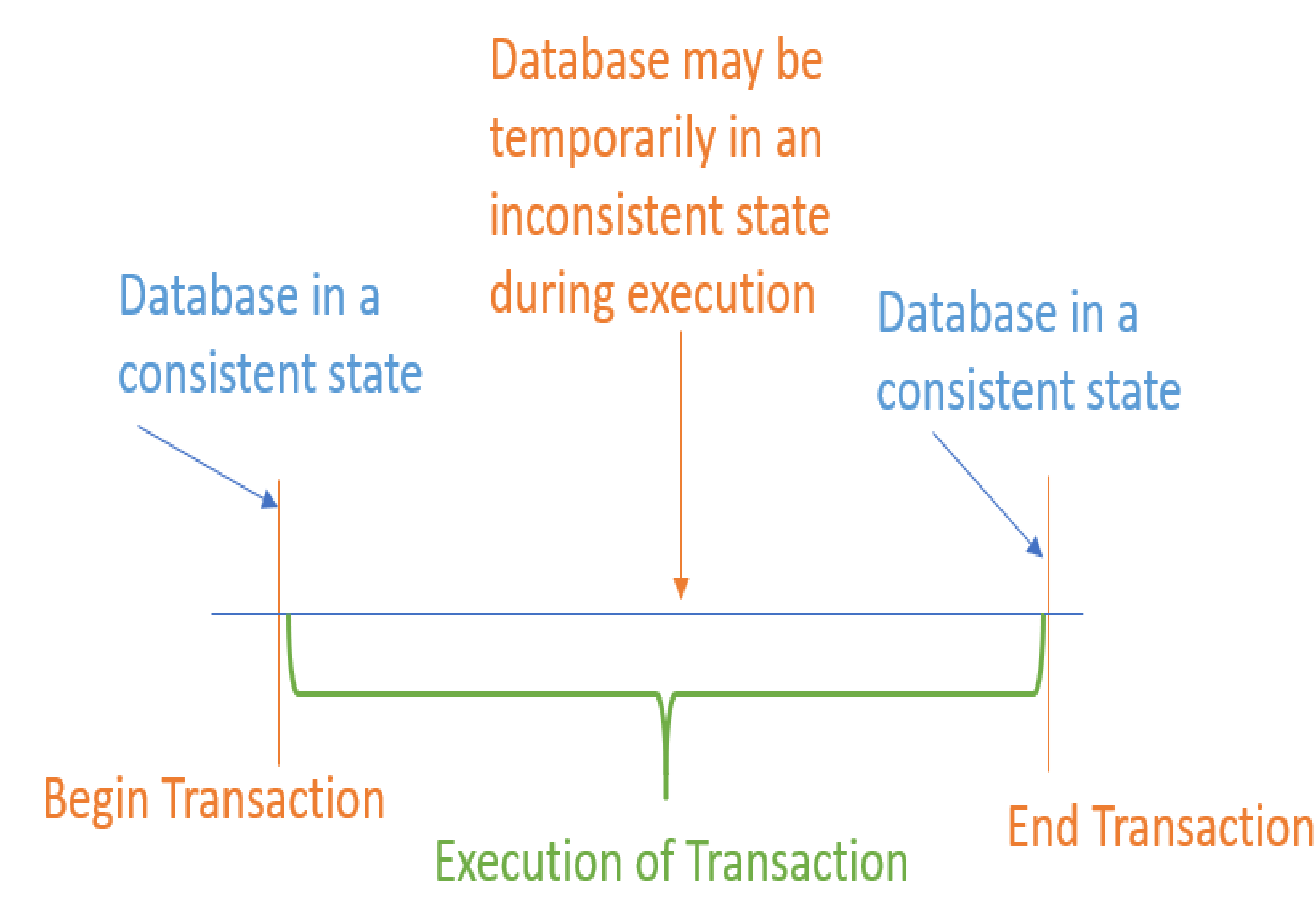
- Let **X**= 500, **Y** = 500.
  Consider two transactions **T** and **T”.**



T	T”
Read (X)	Read (X)
X: = X*100	Read (Y)
Write (X)	Z: = X + Y
Read (Y)	Write (Z)
Y: = Y – 50	
Write	

- Suppose **T** has been executed till **Read (Y)** and then **T”** starts. As a result , **T”** reads correct value of **X** but incorrect value of **Y.**

- Hence, transactions must take place in isolation and changes should be visible only after a they have been made to the main memory.



Database may be temporarily in an inconsistent state during execution

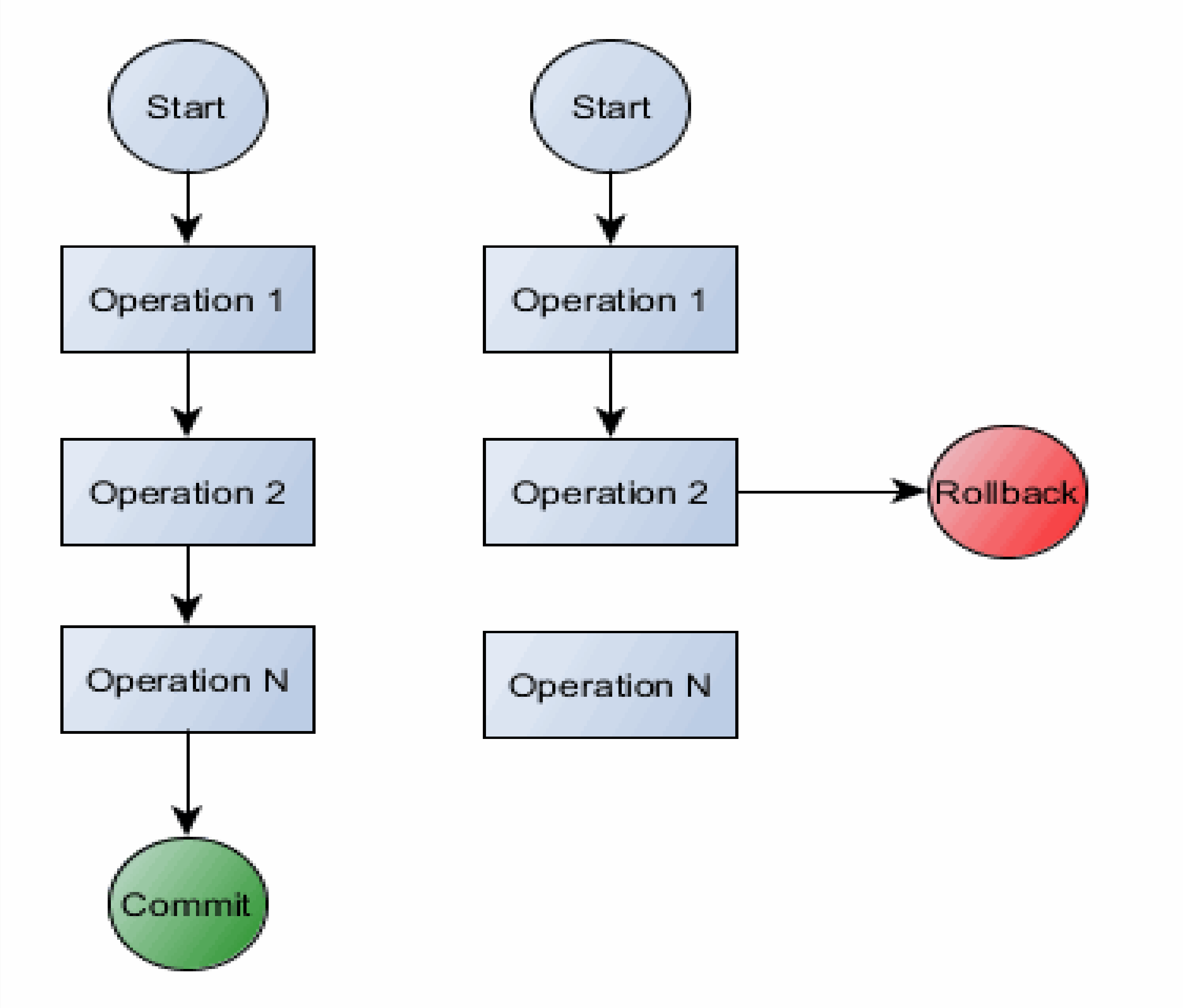
Database in a consistent state

Database in a consistent state

Begin Transaction

Execution of Transaction

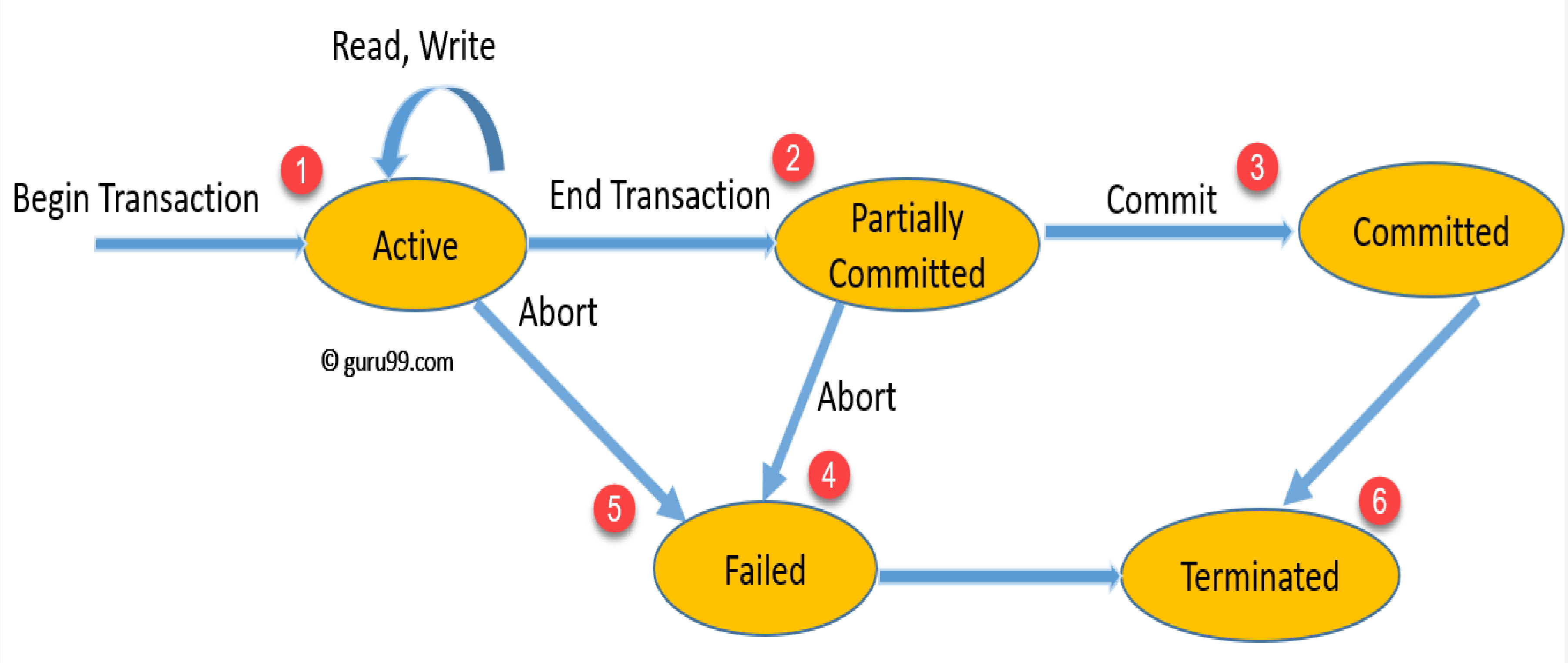
End Transaction


Start
Operation 1
Operation 2
Operation N
Commit
Start
Operation 1
Operation 2
Rollback
Operation N

Durability

This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if system failure occurs

Transaction State



SCHEDULES

- Transactions hold many queries, schedules hold many transactions.
- The basic syntax of a schedule is:

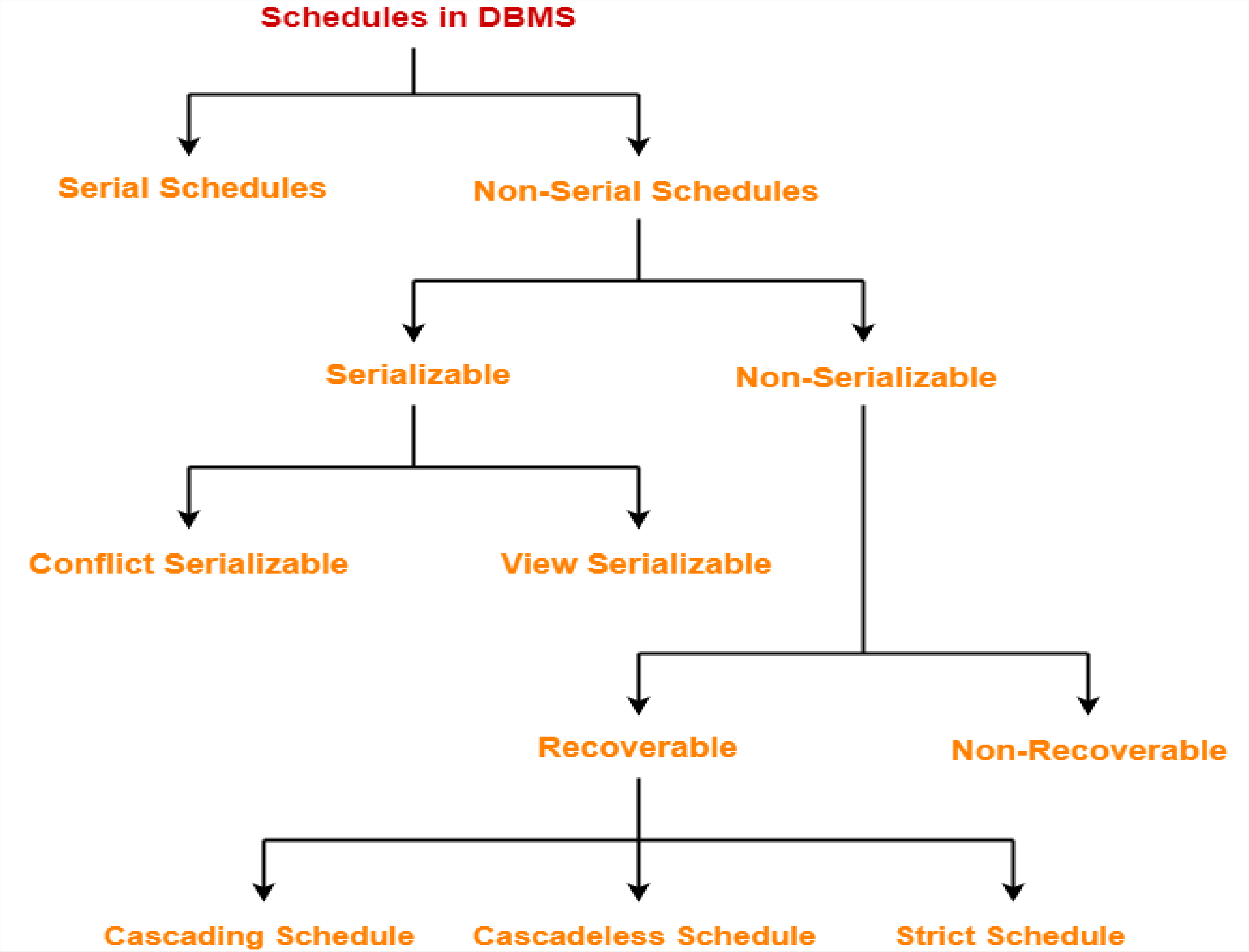
S—The schedule itself.

R—Read an item.

W—write an item

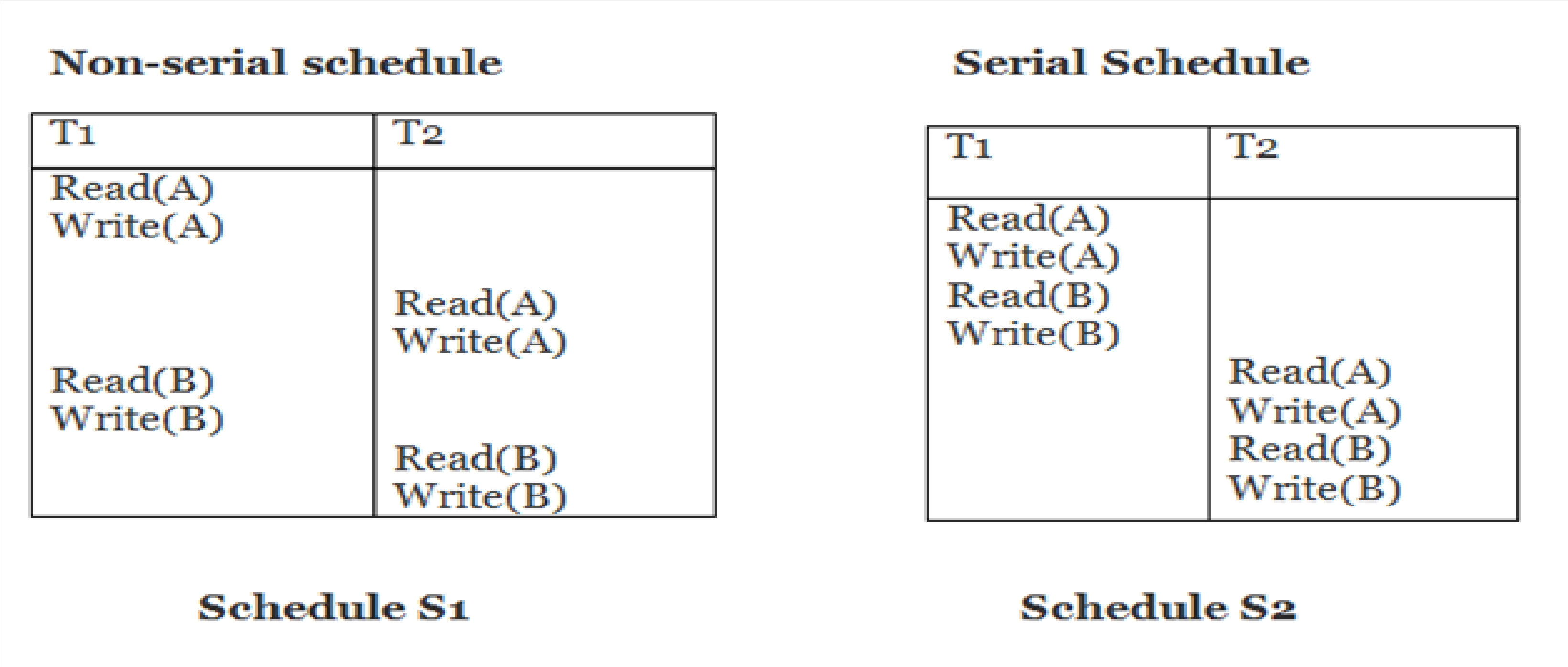
C—commit an item

A—abort the item


Schedules in DBMS
Serial Schedules
Non-Serial Schedules
Serializable
Non-Serializable
Conflict Serializable
View Serializable
Recoverable
Non-Recoverable
Cascading Schedule
Cascadeless Schedule
Strict Schedule

Serial Schedules

- All the transactions execute serially one after the other.
- When one transaction executes, no other transaction is allowed to execute.



Non-serial schedule

T1	T2
Read(A)	
Write(A)	
	Read(A)
	Write(A)
Read(B)	
Write(B)	
	Read(B)
	Write(B)

Schedule S1

Serial Schedule

T1	T2
Read(A)	
Write(A)	
Read(B)	
Write(B)	
	Read(A)
	Write(A)
	Read(B)
	Write(B)

Schedule S2

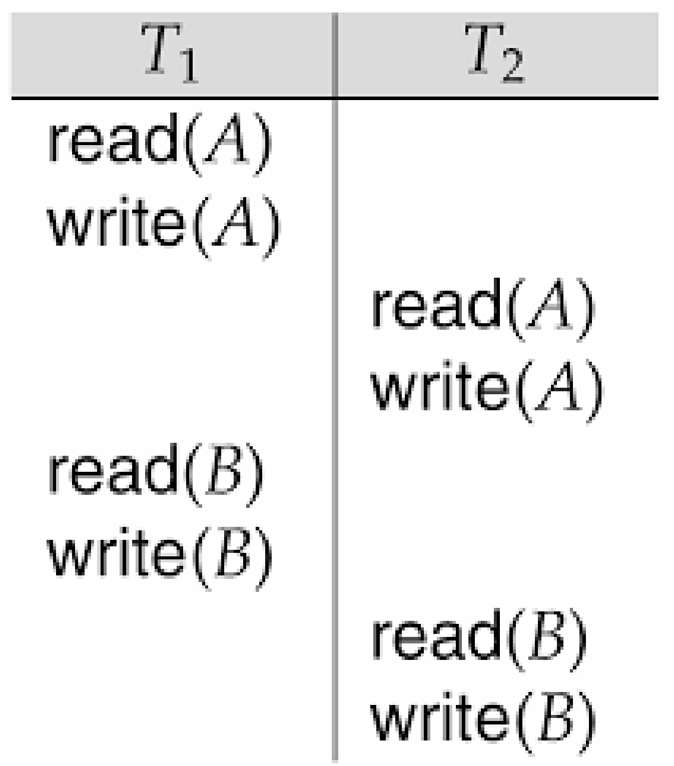
Concurrent Schedules

When multiple transactions execute concurrently in an uncontrolled or unrestricted manner

The schedule no longer need to be serial

OS may execute one transaction and then concurrently execute the 2 transaction and then switch back to 1 one, and so on.

Several execution sequence, the various instructions may be interleaved.



T_1	T_2
read(A)	
write(A)	
	read(A)
	write(A)
read(B)	
write(B)	
	read(B)
	write(B)

Serial Schedule

Advantage:

- It always gives guarantee for data consistency.

Disadvantages:

- High average waiting time.
- Low response time
- low throughput

Concurrent Schedule

Advantages :

- Reduce waiting time.
- improve response time and throughput.

Disadvantages

- Possible data inconsistency.
- some time too much context switching

Thanks!!