Transaction

Transaction

It is a set of operations used to perform a logical unit of work

Types of operations

- Read
- Write
- Commit

Example

A=1000 B=2000

R(A)

A=A-500

W(A)

R(B)

B=B+500

W(B)

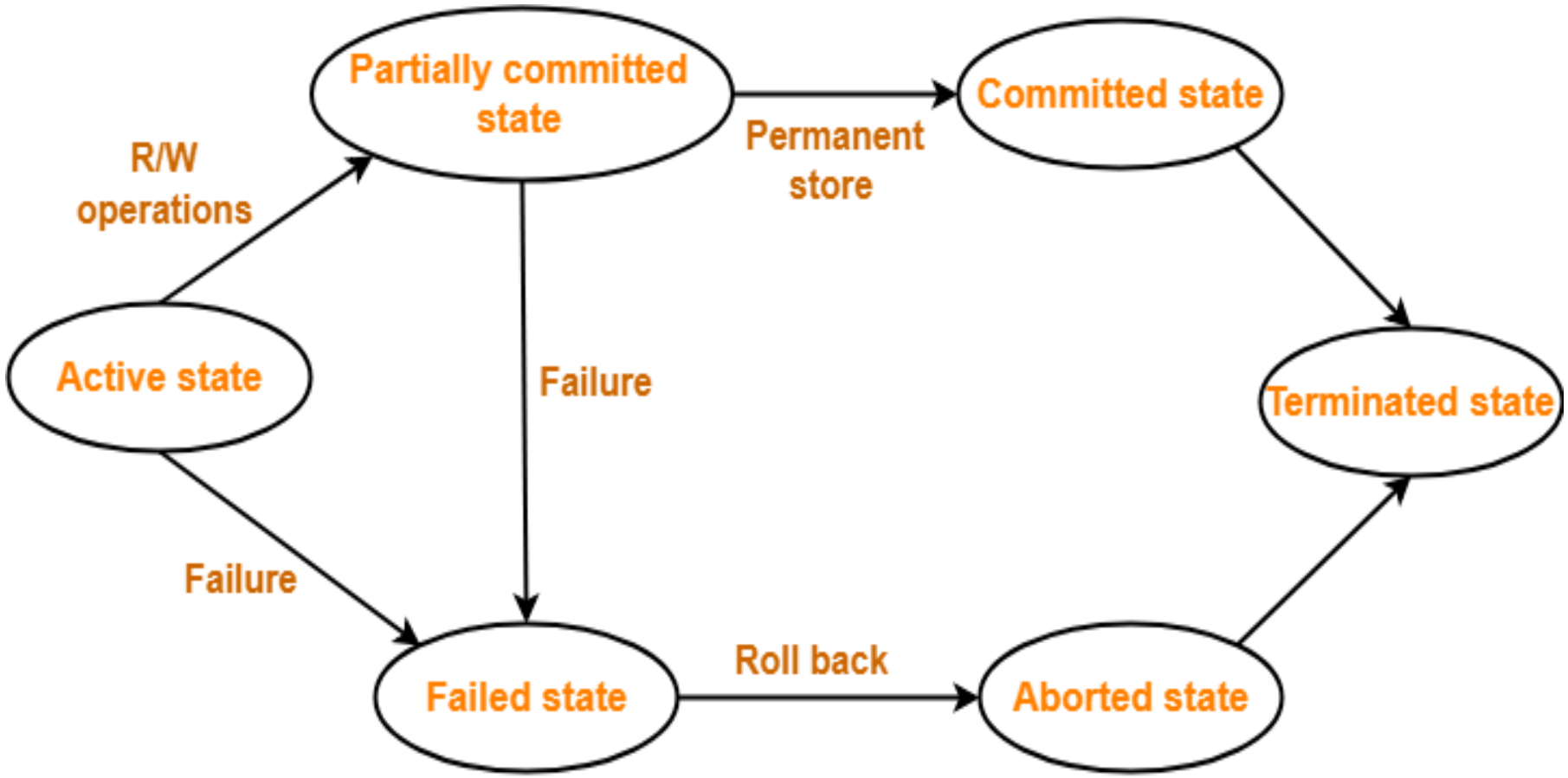
Commit----------A=500 B=2500

ACID Properties

- Atomicity
- Consistency
- Isolation
- Durability

Transaction States

- Active
- Partially Committed
- Committed
- Terminated
- Failed
- Abort



Transaction States in DBMS

Schedule

It is a chronological execution sequence of multiple transactions.

OR

Multiple transactions executed in which order or sequence that is schedule.

Example:- T1,T2,T3……Tn.

Types

- Serial
- parallel

Concurrency control protocol

Achieve serializability and recoverability-locking protocols.

1)Shared-Exclusive locking

-shared(S):-read only

-exclusive(X):-read and write both

eg. T1	T2
S(A)	X(A)
R(A)	R(A)
U(A)	W(A)
	U(A)

Compatibility table:-

YES	NO
NO	NO

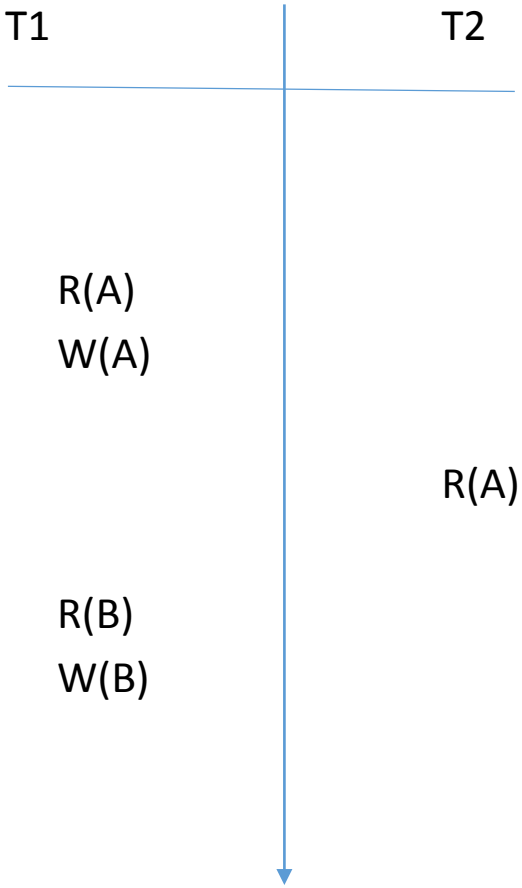
Advantages

- Simple protocol
- Simple programming
- Simple software

Problems

May not sufficient to produce only serializable schedule.

Example,

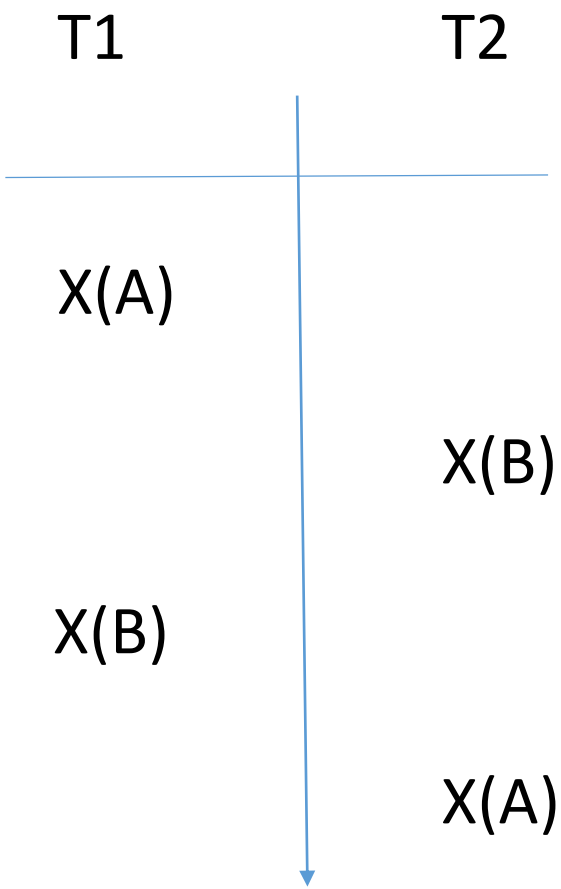


May not free from irrecoverability.

Example,

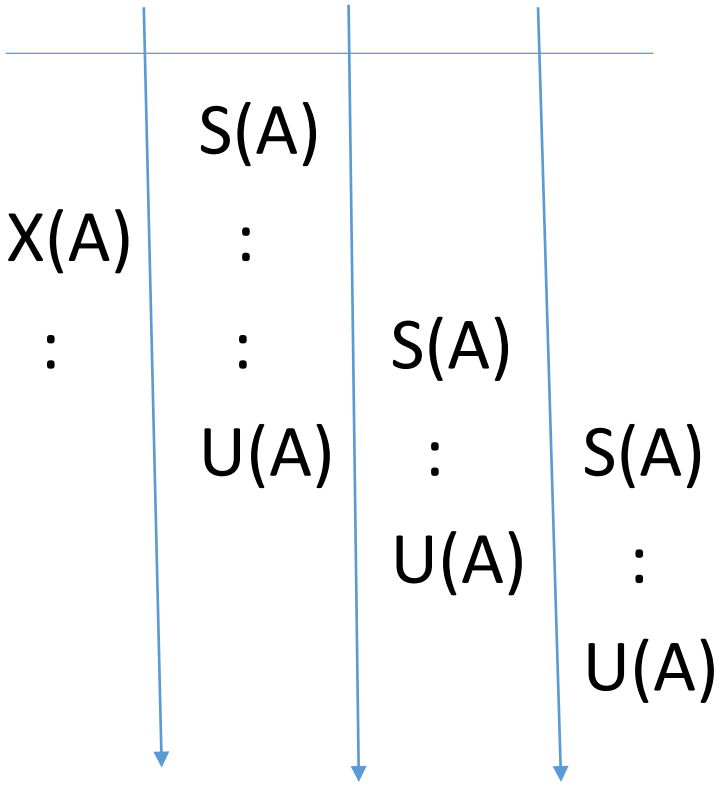
May not free from deadlock

Example,



May not free from starvation

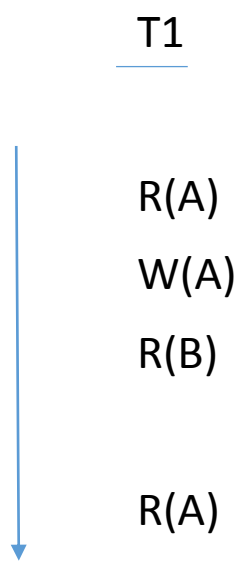
Example,



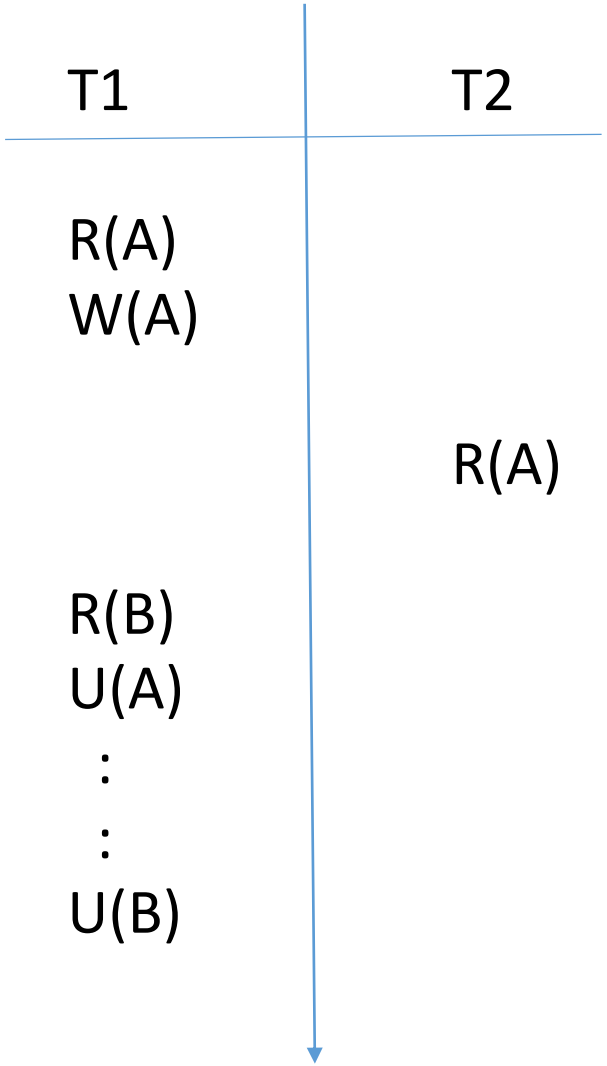
2-phase locking(2PL)

- Growing:-locks are acquired and no locks are released.
- Shrinking:-locks are released and no locks are acquired.

Example



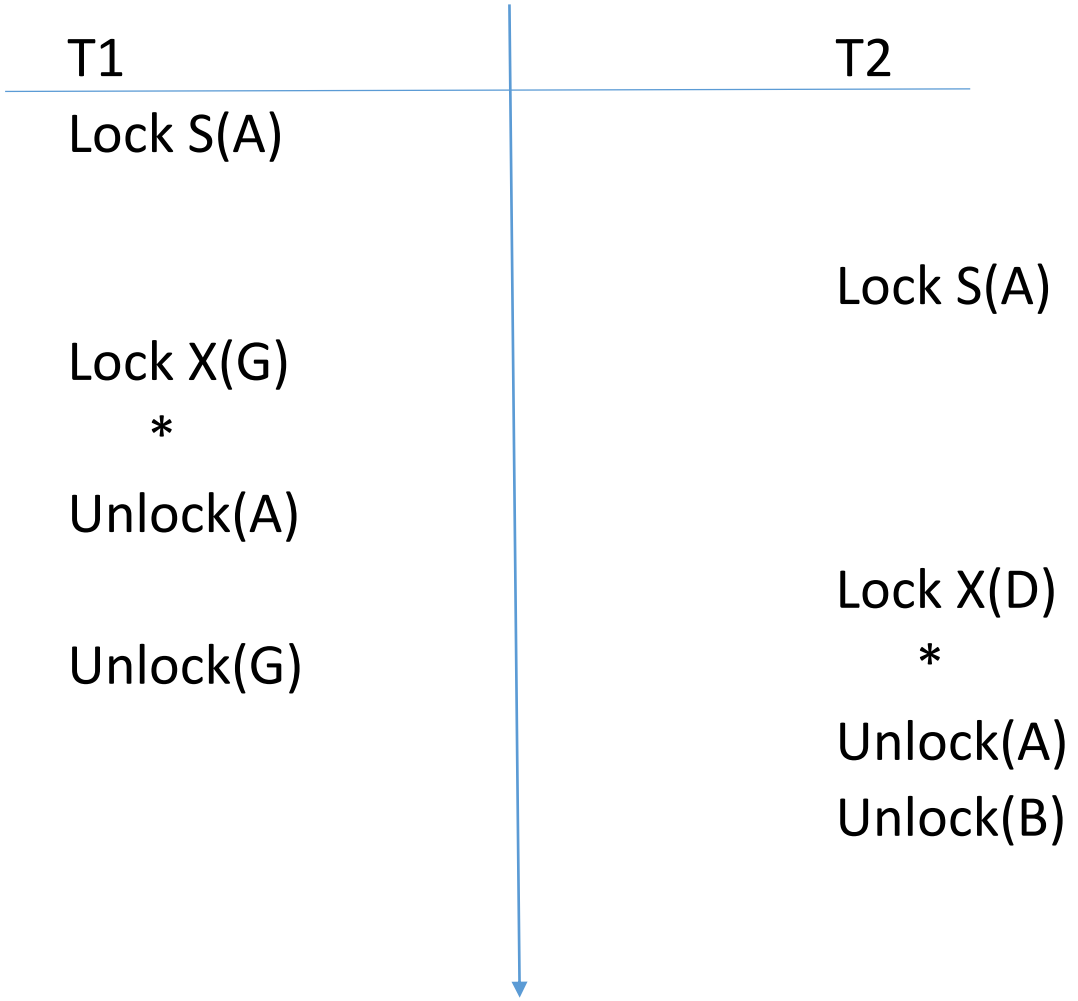
Example



Note

Any transaction that follows 2PL, it will always be serializable.

Example


T1
T2
Lock S(A)
Lock S(A)
Lock X(G)
*
Unlock(A)
Lock X(D)
Unlock(G)
*
Unlock(A)
Unlock(B)

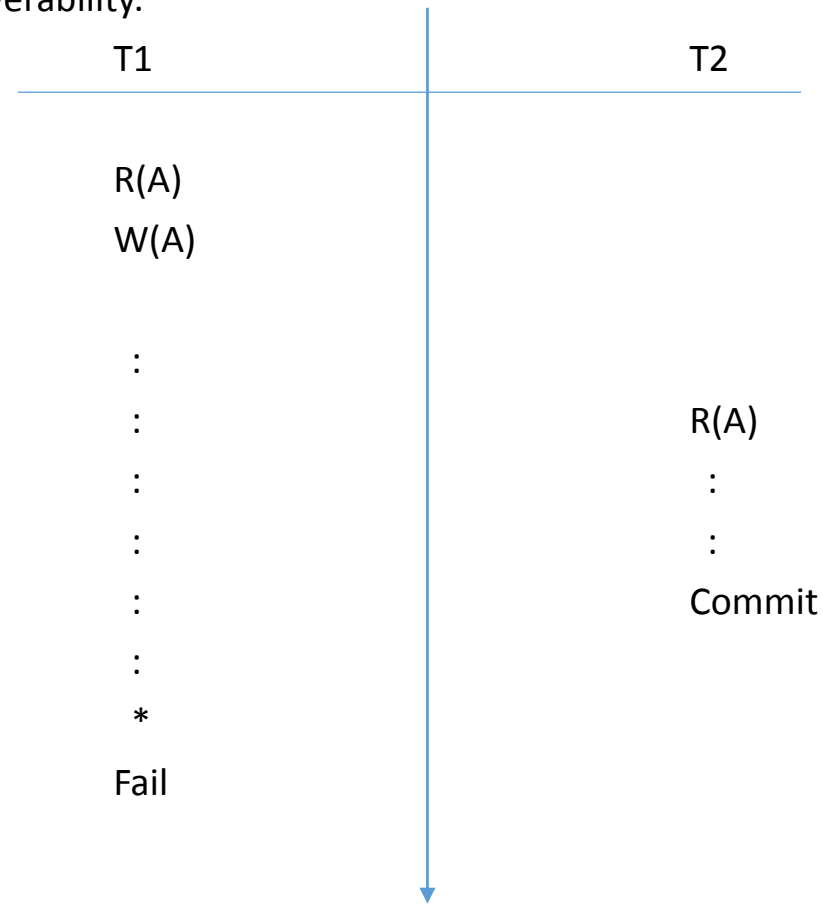
Advantages:-

Always ensure serializability

Drawbacks:-

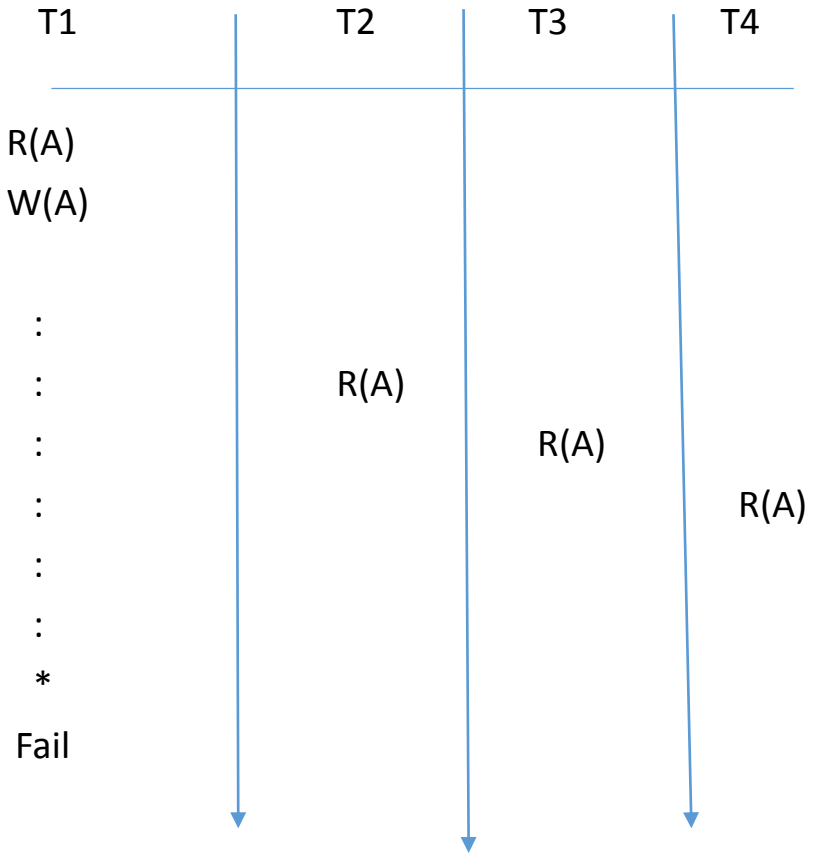
- May not free from irrecoverability.

Example:- `



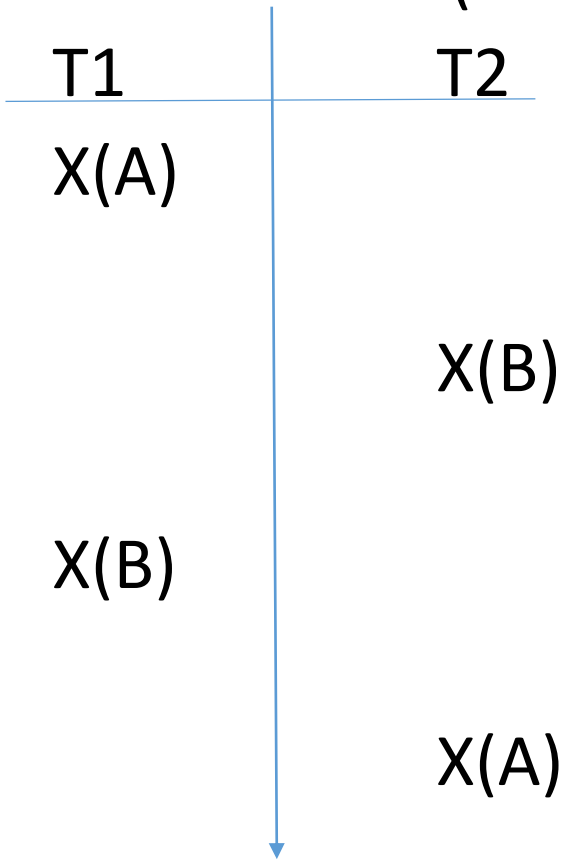
- Not free from cascading rollback

Example:-



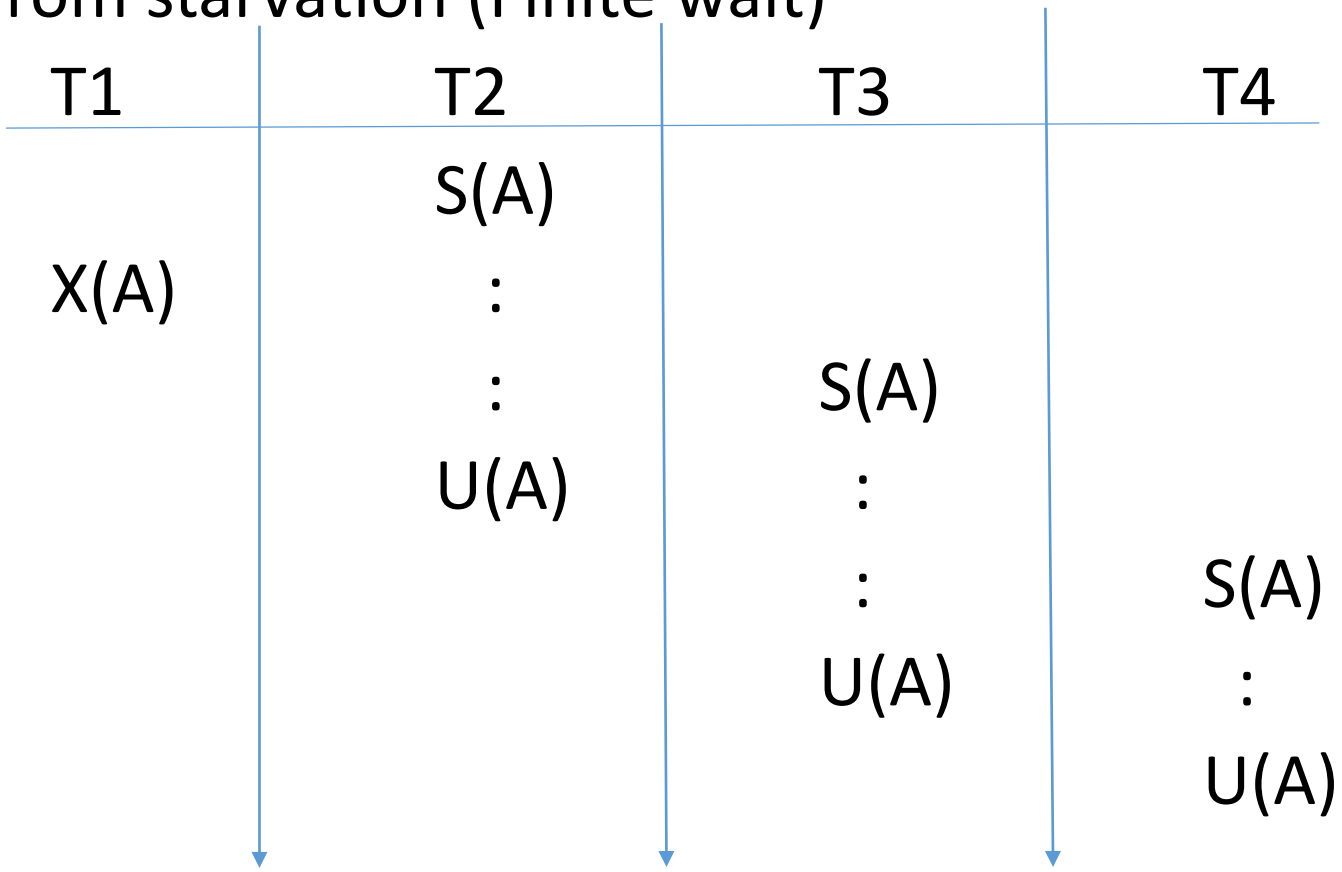
- Not free from deadlocks(Infinite wait)

Example:-



- Not free from starvation (Finite wait)

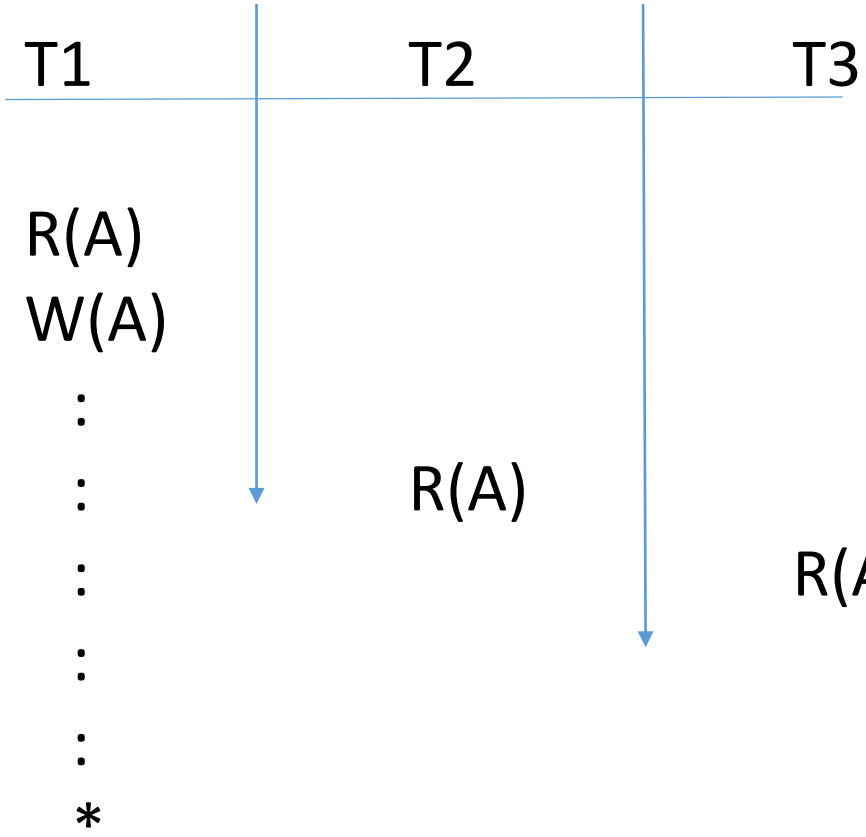
Example:-

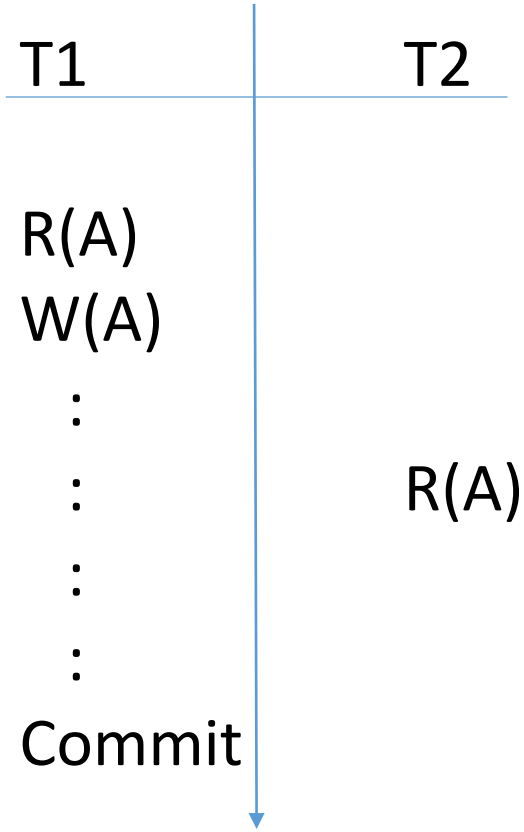


Extensions of basic 2PL

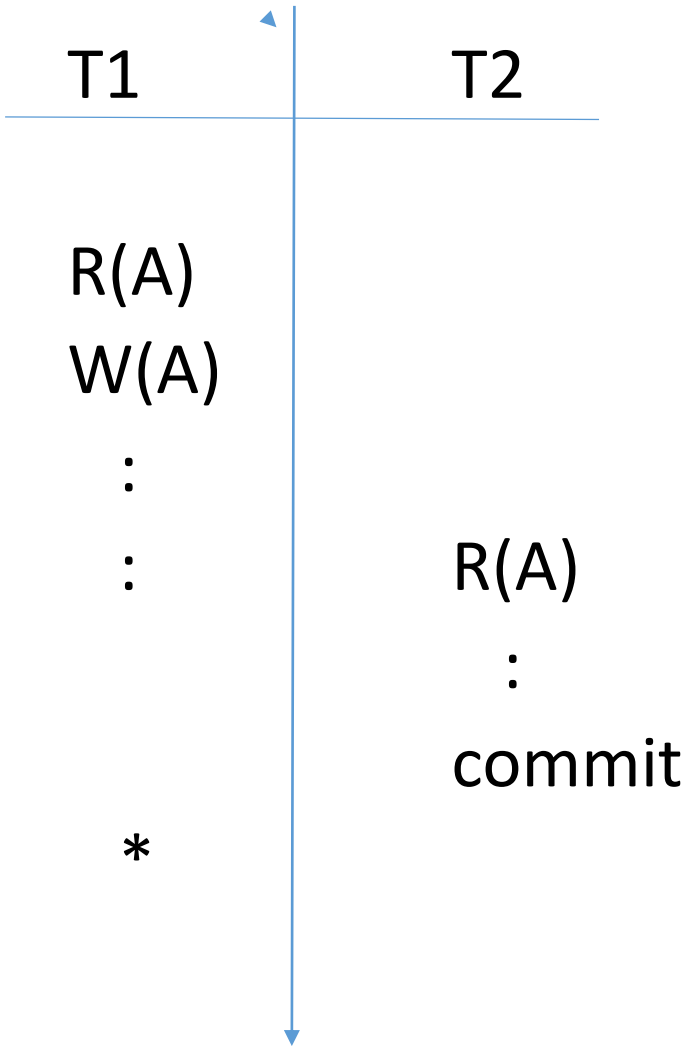
- Strict 2PL:-It should satisfy the basic 2PL and all exclusive locks should hold until commit/Abort.
- Rigorous 2PL:-It should satisfy the basic 2PL and all shared, Exclusive locks should hold until commit/Abort. It is more restricted than strict 2PL.

Example:-





Example:-



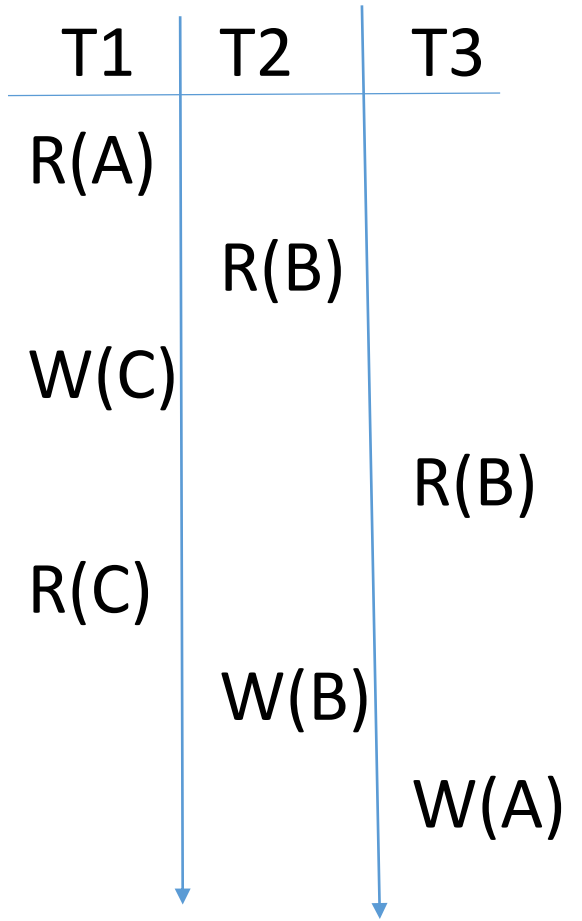
Advantages:-

- Cascadeless
- Recoverable

Drawbacks:-

- Deadlock
- Starvation

Example-Timestamp ordering protocol



	A	B	C
RTS	0	0	0
WTS	0	0	0

Pessimistic concurrency control protocol

- Transaction delay
- Transaction rollback
- Unnecessary overhead

Validation based or optimistic concurrency control protocol

- No checking is done while the transaction is executing.
- It is based on assumption that majority of database operations do not conflict.
- A transaction proceeds without restrictions until the transaction gets commit.
- It is then checked to see whether it has come into conflict with other transactions.
- When a conflict occurs, a transaction is aborted.

Three phases

- Read phase
- Validation phase
- Write phase

Read Phase

- At the start of this phase, transaction ti is associated with a timestamp start(Ti).
- Ti reads the values of data items from the database and these values are then stored in the temporary local copies of the data items kept in the workspace of Ti.
- All modifications are performed on these temporary local copies of the data items without updating the actual data item of the database.

Validation Phase

- At the start of this phase, transaction Ti is associated with a timestamp validation(Ti).
- The system performs a validation test when Ti decides to commit.
- This validation is performed to find whether the modification made to be temporary local copies can be copied to the database without violating serializability.
- If Ti conflict with any other concurrently executing transaction, ti is rolled back, its work place is cleared and Ti is restarted.

Write Phase

- The system copies the modifications made by Ti in its workspace to the database only if it succeeds in the validation phase.
- At the end of this phase, Ti is associated with a timestamp finish(Ti).

Validation test for a transaction Tj

- For each transaction Ti with

  Ts(Ti) < Ts(Tj)

Either one of the following condition holds:-

- Finish(Ti) < start(Tj)

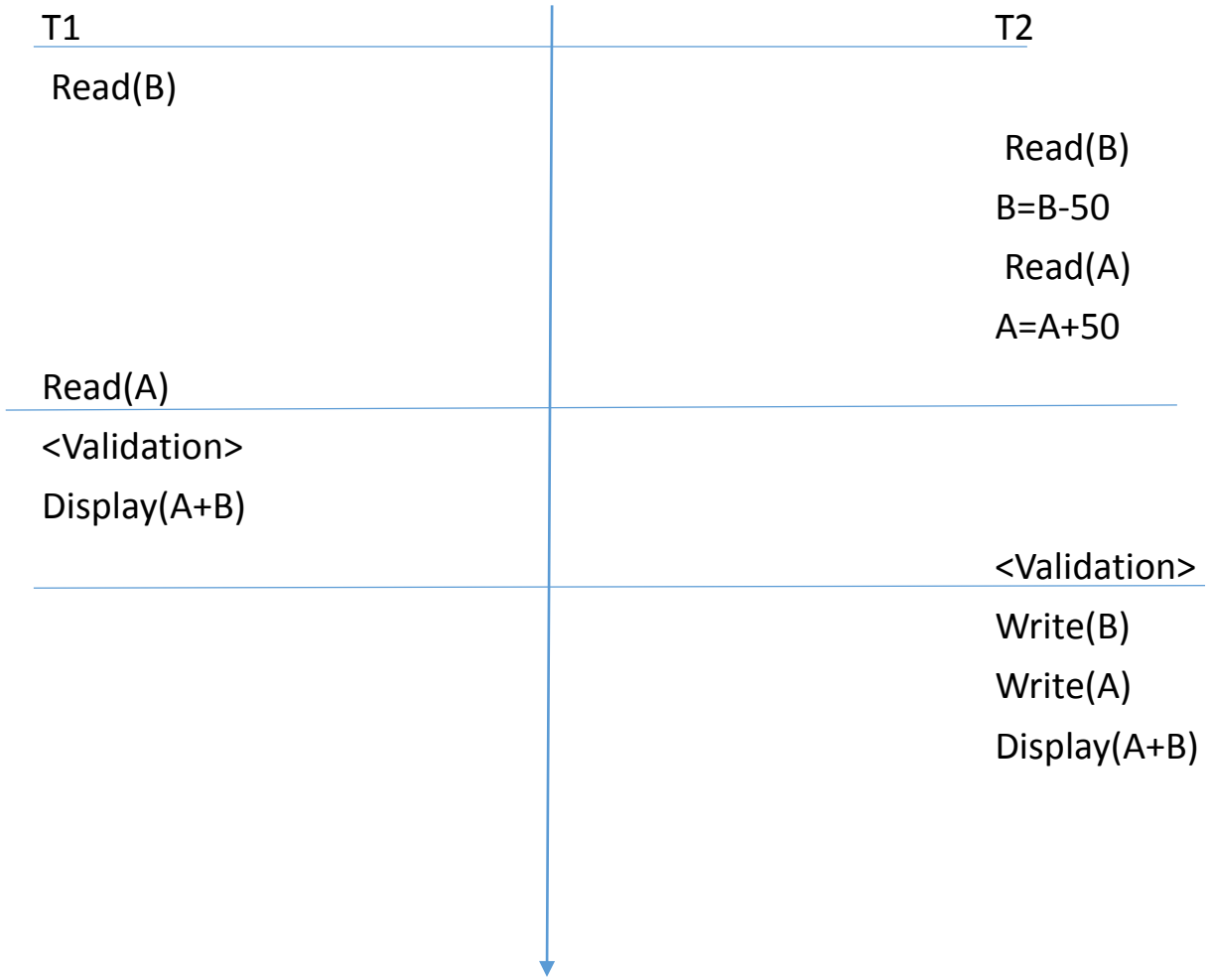
Transaction Tj starts before Ti finishes (start(Tj) < finish(Ti)):-

- Finish(Ti) < Validation(Tj)

Transaction Tj starts before Ti finishes:-

- Validation(Ti) < Validation(Tj)

Example


T1
T2
Read(B)
Read(B)
B=B-50
Read(A)
A=A+50
Read(A)
<Validation>
Display(A+B)
<Validation>
Write(B)
Write(A)
Display(A+B)

Deadlock handling

- Prevention
- Detection

Deadlock prevention

It ensures that deadlock never happens.

Deadlock prevention techniques:-

- Wait-die technique
- Wound-wait technique

Wait-die

(a) If a transaction Ti request for a resource that is locked by transaction Tj,

if Ts(Ti) < Ts(Tj) then

Ti is allowed to wait until resource is available for execution.

Wait-die

(b) If Ti is older transaction and has hold some resources with it and Tj is waiting for it, then Tj is rolled back (dies).

if Ts(Ti) < Ts(Tj) then

Tj is rolled back and restarted with very minute delay with same timestamp.

Hence, this technique allows the older transaction to wait but kills(dies) the younger transaction.

Wound-wait

(a) If transaction Ti request for a resource that is locked by transaction Tj,

if Ts(Ti) < Ts(Tj) then

Tj is rolled back (i.e wounded by Ti)

Wound-wait

(b) If transaction Tj(younger) requesting a resource which is held by Ti(old),

if Ts(Ti) < Ts(Tj) then Tj waits

Advantages of both techniques:-

- Starvation is avoided.
- A transaction with smallest timestamp is not rolled back.

Disadvantage:-

- The request to acquire a lock on a data item held by another transaction does not necessary involve a deadlock.
- Therefore, unnecessary rollbacks may occur in both wait-die and wound-wait techniques.

Deadlock detection using wait-for graph

- It is suitable for smaller databases.
- To detect deadlock, the system maintains a wait-for graph, which consists of nodes and directed edges.
- Nodes of graph represent currently executing transactions.
- Directed edges exist from one node to another if a transaction is waiting for another transaction to release a lock.
- If this graph contains a cycle, it indicates the deadlock in the system.

Failure classification

- Transaction failure
- System crash/Computer failure
- Disk failure
- Physical problem and environmental disasters

Transaction failure

A transaction has to abort when it fails to execute or when it reaches a point from where it can’t go any further is called transaction failure.

Reasons:-

-Logical error

-system error

Storage structure

- Volatile
- Nonvolatile
- Stable

Stable

a mythical form of storage that survives all failures.

approximated by maintaining multiple copies on distinct nonvolatile media.

- Maintain multiple copies of each block on separate disks
  - copies can be at remote sites to protect against disasters such as fire or flooding.
- Failure during data transfer can still result in inconsistent copies: Block transfer can result in
  - Successful completion
  - Partial failure: destination block has incorrect information
  - Total failure: destination block was never updated
- Protecting storage media from failure during data transfer (one solution):
  - Execute output operation as follows (assuming two copies of each block):
    1. Write the information onto the first physical block.
    2. When the first write successfully completes, write the same information onto the second physical block.
    3. The output is completed only after the second write successfully completes.

stable

- Copies of a block may differ due to failure during output operation. To recover from failure:

  First find inconsistent blocks:

  1. *Expensive solution*: Compare the two copies of every disk block.
  2. *Better solution*:
     - Record in-progress disk writes on non-volatile storage (Non-volatile RAM or special area of disk).
     - Use this information during recovery to find blocks that may be inconsistent, and only compare copies of these.
     - Used in hardware RAID systems

Recovery methods-Log based recovery

- Deferred database modification
- Immediate database modification

Deferred database modification

T1	
R(A)	<T1,start>
A=A+100	<T1,A,200>
W(A)	<T1,B,400>
R(B)	<T1,commit>
B=B+200	
W(B)	
COMMIT	

Example

<T1,start>

<T1,A,200>

<T1,B,400>

<T1,commit>

<T2,start>

<T2,C,500>

Immediate database modification

T1	
R(A)	<T1,start>
A=A+100	<T1,A,100,200>
W(A)	<T1,B,200,400>
R(B)	<T1,commit>
B=B+200	
W(B)	
COMMIT	

Example

<T1,start>

<T1,A,1000,2000>

<T1,B,5000,6000>

<T1,commit>

<T2,start>

<T2,C,700,800>