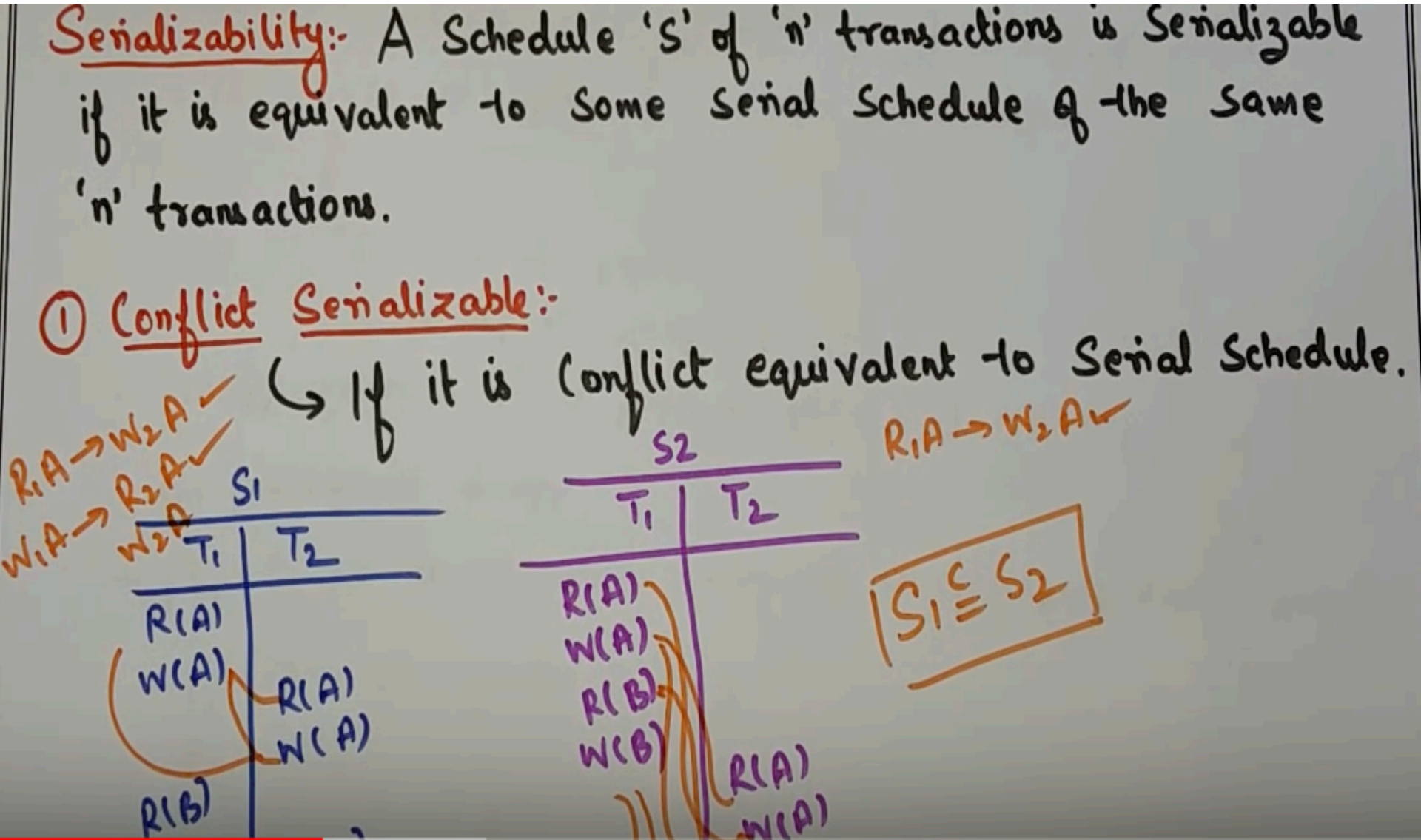
UNIT - 5

Serializability



Test for Conflict Serializability

Precedence Graph is used

- Let 'S' be a Schedule, Construct a directed graph known as precedence graph.
- Graph Consists of a pair of G= (V, E) where
  V: a set of Vertices
  E: Set of Edges

Algorithm for Creation of Graph

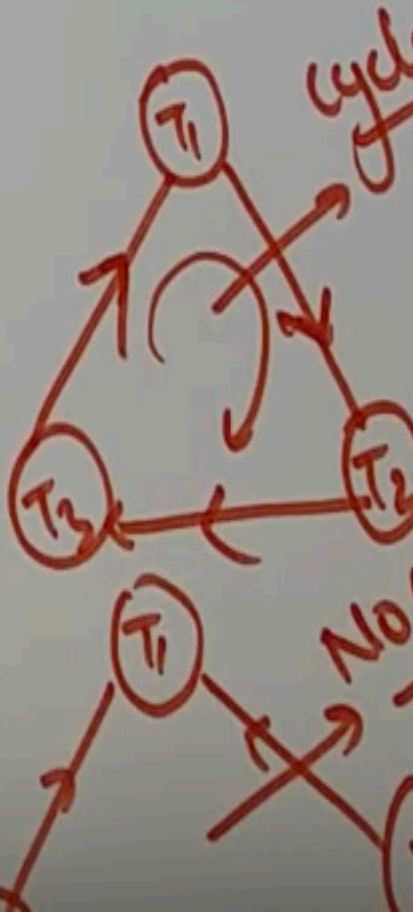
(i) Create a Node for each transaction

(ii) A directed edge, $T_i \rightarrow T_j$, if T_j reads a value of an item written by T_i.

iii) Directed edge $T_i \rightarrow T_j$, if T_j writes a value into item after it has been read by T_i.

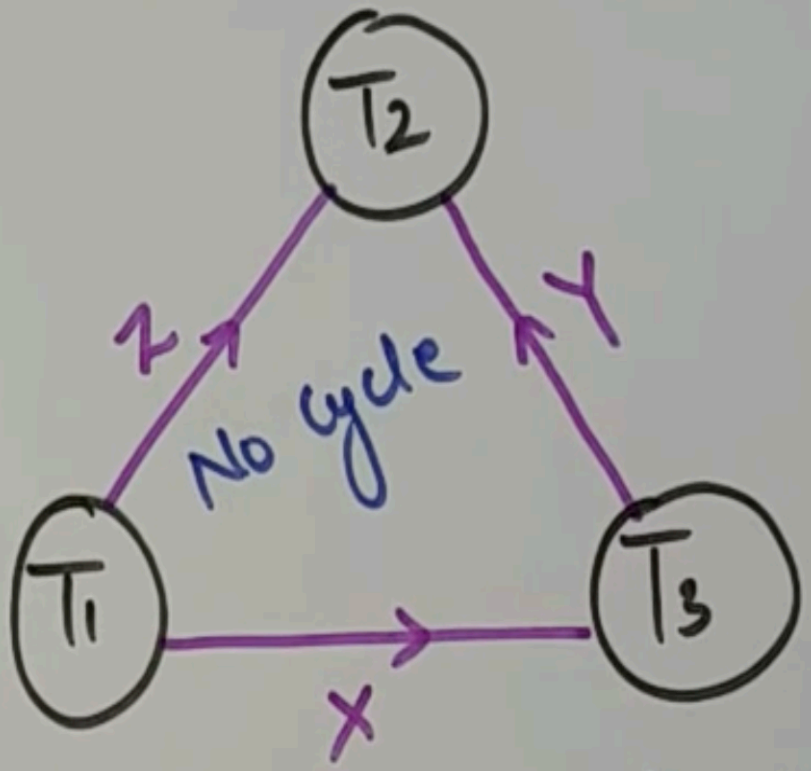
) Directed edge, $T_i \rightarrow T_j$, if T_j write after T_i write.

[A schedule is Conflict Serializable if and only if precedence graph is acyclic.]



Ques1.) check for conflict Serializability.

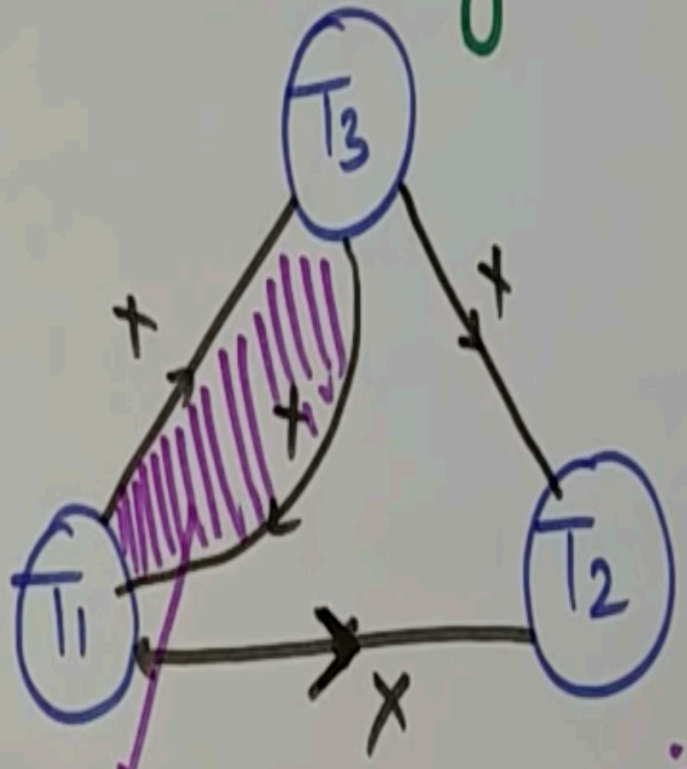
T_1	T_2	T_3
R(x)		
	R(z)	
R(z)		
		R(x)
		R(Y)
		W(x)
	R(Y)	
	W(z)	
	W(Y)	



Conflict Serializable. ✓

Ques 2.) Check for Conflict Serializability.

T_1	T_2	T_3
R(x)		
		R(x)
		W(x)
W(x)		
	R(x)	



Cycle.

Not Conflict Serializable.

View Serializability Two Schedules 'S' and 'S'' are **view equivalent** if the following conditions are met:

(i) For each data item Q, if Ti reads an initial value of Q in Schedule S, then Ti must in S' also reads an initial value of Q.

(ii) if Ti executes Reads Q in S, and that value was produced by Tj (if any), then Ti must in Schedule S' also reads the value of Q that was produced by Tj.

(iii) For each data item Q, the transaction that perform the final Write(Q) operation in Schedule S must also perform the final Write(Q) in Schedule S'.

Lock based protocol

Concurrency Control:- It is the process of managing Simultaneous execution of transactions in a shared database, to ensure th Serializability of transactions.

Purpose of Concurrency Control

(i) To enforce Isolation → One tr^n is not interleaving with another tr^n.

(ii) To preserve database Consistency

(iii) To resolve read-write and write-write Conflicts. {Conflicting operations}.

Concurrency Control Techniques:-

① **Lock-Based Protocol:** A Lock gurantees exclusive use of a data item to a current transaction.

- To Access Data item (Lock Acquire)
- After Completion of transaction (Release Lock)

[all data items must be accessed in a mutually exclusive manner.]

Types of Locks

Shared Lock
- Lock-S
- Read data value

Exclusive Lock
- LOCK-X
- Both Read and

Compatibility b/w Lock Modes

	S	X
S	✓T	XF
X	XF	XF

Eg. of Lock Protocol

T_1	T_2
Lock-X(B)	
R(B) -	→ Exclusive lock
B-50	
W(B) -	
Unlock(B)	
	Lock-S(B) } Shared lock
	R(B) -
	Unlock(B)

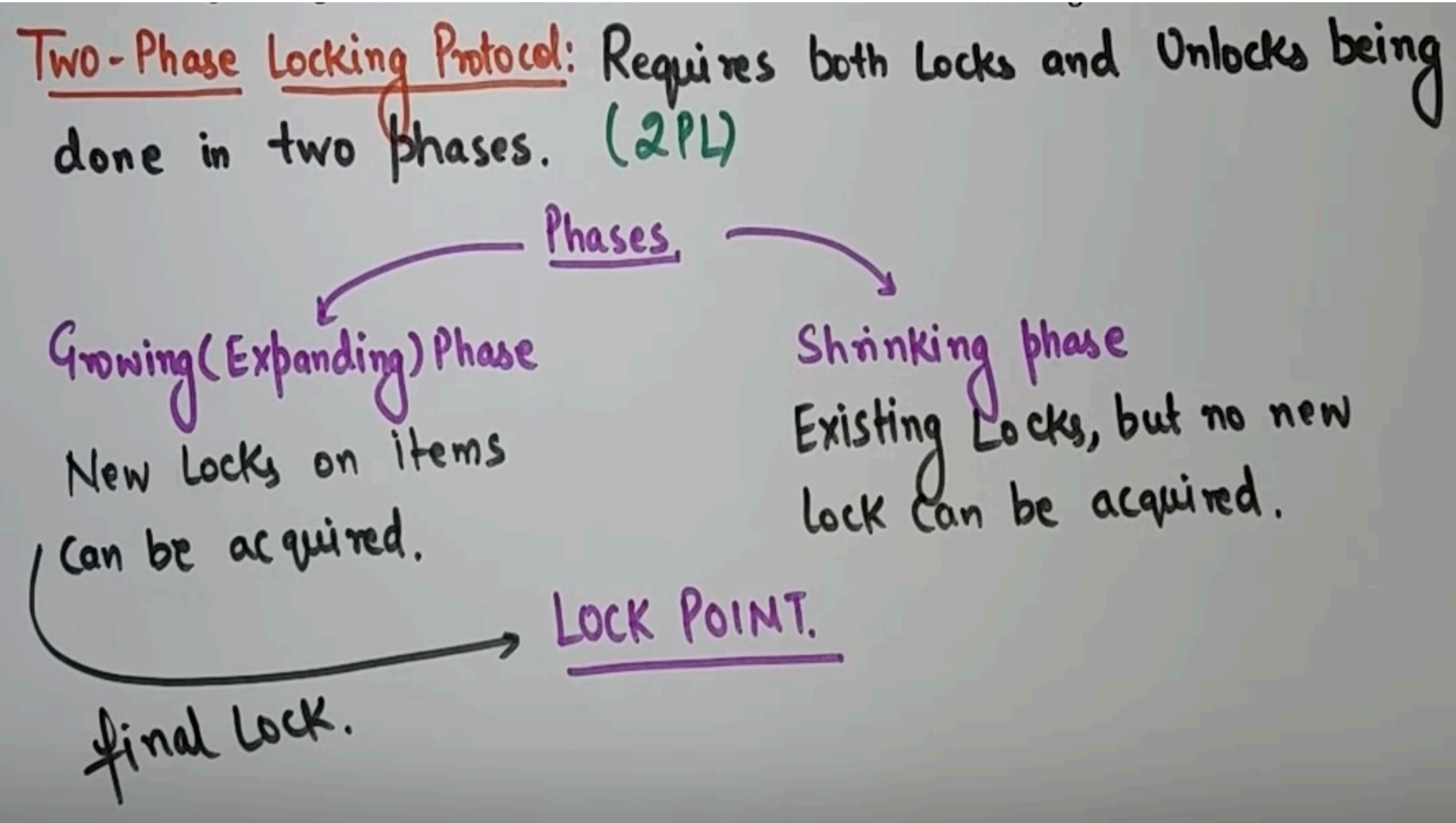
Note:- Any no. of transactions Can hold Shared lock on an item but Exclusive lock can by one transaction

Conversion of Locks:-

i) upgrading → R-lock → write-lock.

ii) Downgrading → Write lock - Read-lock.

2PL protocol

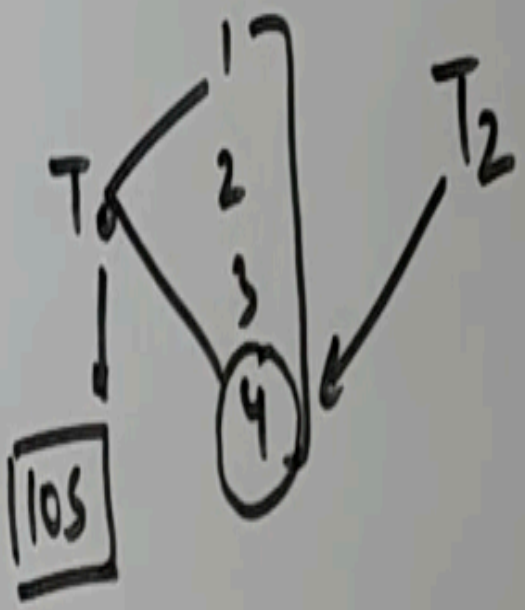


2PL Protocol enforces Serializability but may reduce Concurrency due to the following reasons.

↳ (i) Holding Lock Un-necessarily.

(ii) Locking too early.

(iii) Penalty to other trⁿ.



Variations of 2PL Locking Protocol

Conservative (Static) 2PL

↳ Acquire all lock before it starts

↳ Release all locks after commit.

- Avoids Cascading Rollback.
- Deadlock Free

Strict 2PL

Exclusive lock can't be released until commit.

→ Helps in Cascadeless schedule.

Rigrous 2PL

Shared / Exclusive Can't be released Until Commit.

→ Avoids Cascading Rollback.

Deadlock May Occur.

Distributed databases

Recovery in Distributed Database More complicated than in a Centralized System. Failures related to distributed db are :→ loss of message, failure of site at which Subtransaction is running, failure of Comm^{n} Link.

2-Phase Commit protocol

Recovery System must ensure ATOMICITY. (ALL or NONE)

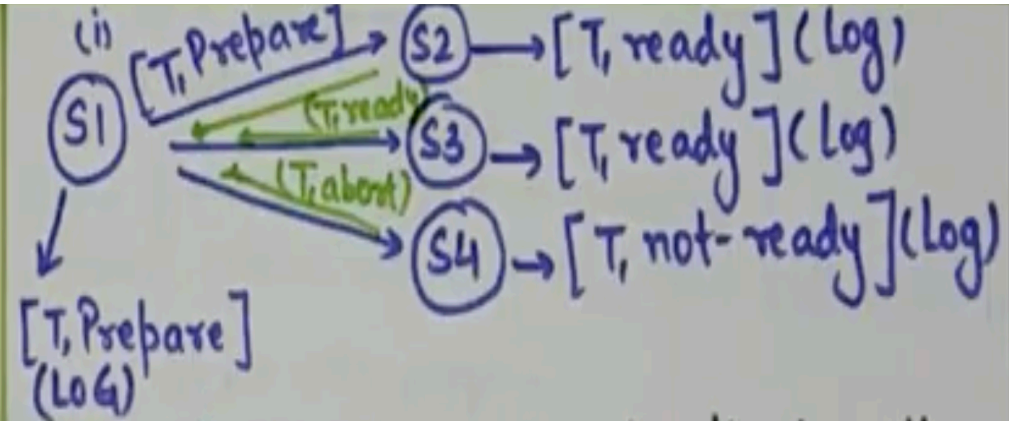
Commit Protocol:

(i) Two-Phase Commit: Two Phases of Commit Protocol → Voting Phase, Decision Phase

Transaction T is initiated at Site S1, (C.S)

S2, S3 and S4] P.S → S2, S3, S4

VOTING PHASE: In this, participating sites Vote on whether they are ready to Commit the transaction or not.



DECISION PHASE: In this, the Coordinator Site decides whether the transaction can be Commited or has to be aborted.

↳ i) [Ready, T] message from all P.S

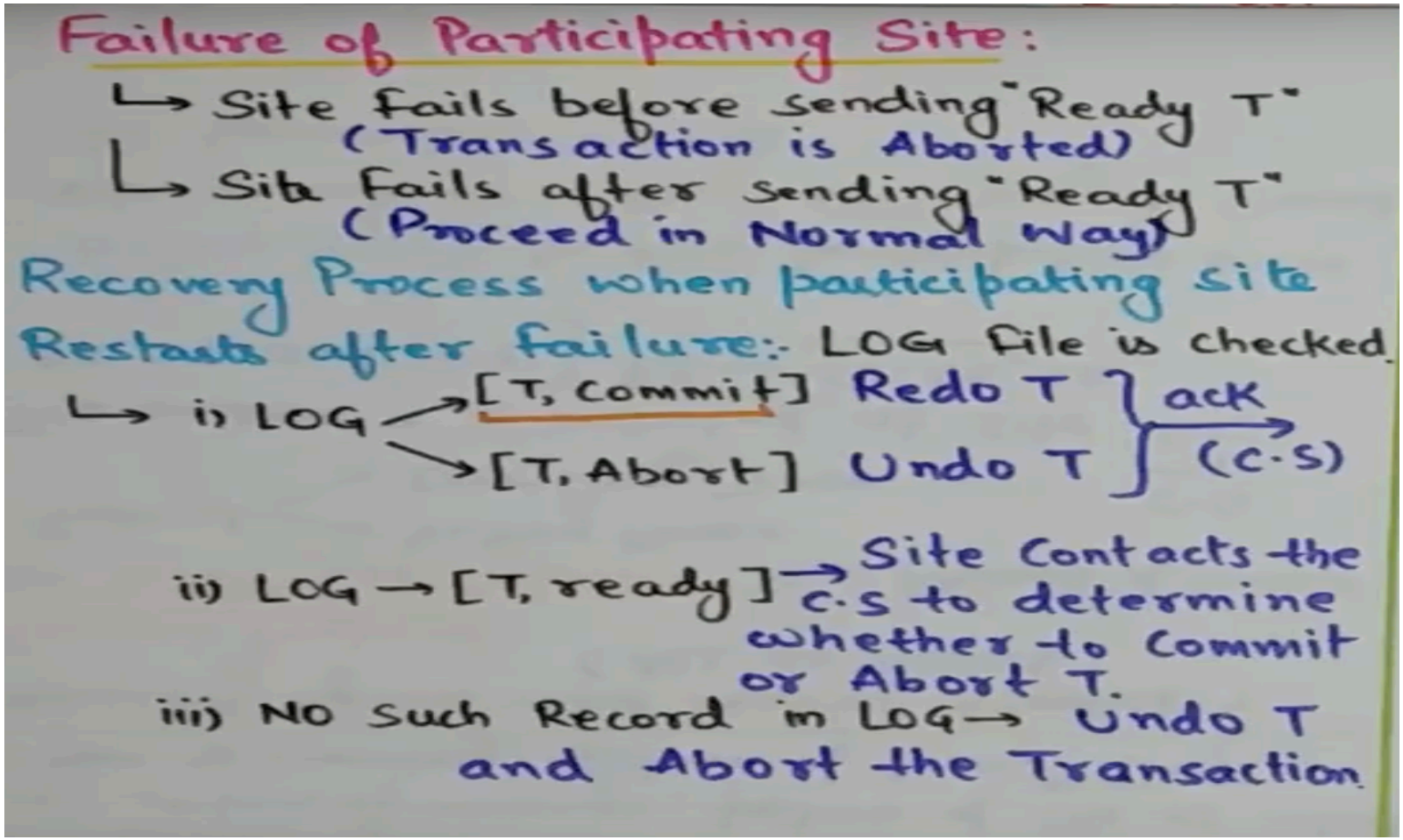
↳ Commit

S1 → S2 → S3 → [T, Commit]

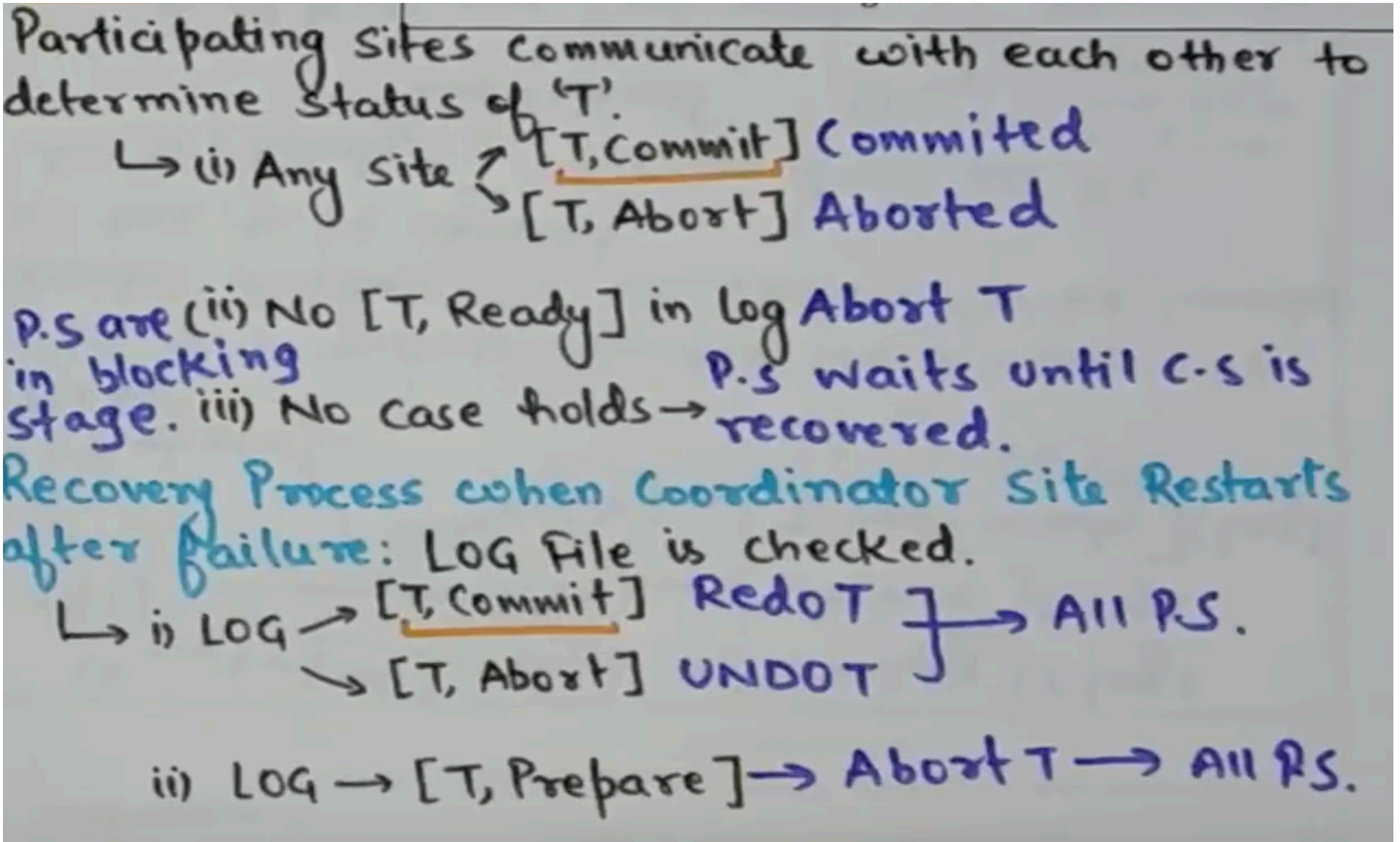
ii) At least one [not-ready, T], Abort the transaction T.

S1 → [T, Abort]

2PC (contd.)


Failure of Participating Site:
Site fails before sending "Ready T"
(Transaction is Aborted)
Site fails after sending "Ready T"
(Proceed in Normal Way)
Recovery Process when participating site Restarts after failure:- LOG file is checked.
i) LOG → [T, commit] Redo T
[T, Abort] Undo T
ack (C·S)
ii) LOG → [T, ready] → Site Contacts the C·S to determine whether to Commit or Abort T.
iii) NO Such Record in LOG → Undo T and Abort the Transaction

Failure of coordinators site



3-Phase Commit protocol

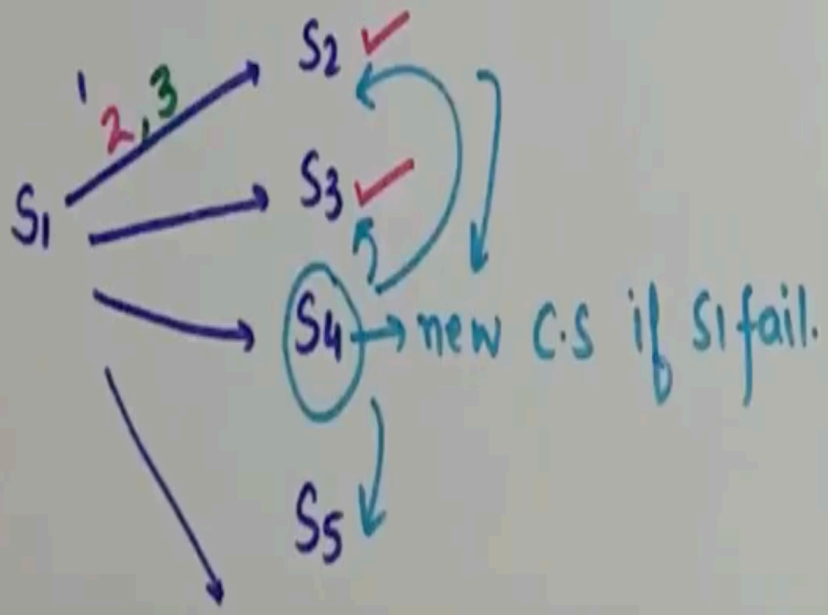
Three-Phase Commit: (3PC) is an extension of the two phase commit protocol that avoids blocking even if the coordinator site fails during recovery.

Conditions to Avoid Blocking:

↳ i) No N/w Partitioning

ii) Atleast there is one available Site.

iii) At most 'K' site fail, where 'K' is some predetermined no. (K=2)



1) [T, Prepare]
2) [T, Pre-Commit] } → Phase b/w voting and decision.
3) [T, Commit]

Theory:- C.S sends [T, Prepare] message and receive votes from P.S. Then it Sends [T, Pre-Commit] msg and after ensuring that 'K' Sites Knows about the decision to Commit, actual [T, Commit] msg is sent.

if C.S fails, new C.S is Chosen which Communicates with remaining sites to check whether old C.S had decided to Commit. It Can be checked easily if any one of 'K' sites has [T, Pre-Commit] msg.

[BLOCKING IS AVOIDED, BUT OVERHEAD is ↑]