Handling distributed Transactions :

Transaction : Set of ops which needs to be performed or simply a group of tasks which needs to be performed against the DB.

4 props :

Atomicity : all ops in single Tx, should be successful or all should fail

Consistency : DB should be in consistent state before & After Tx.

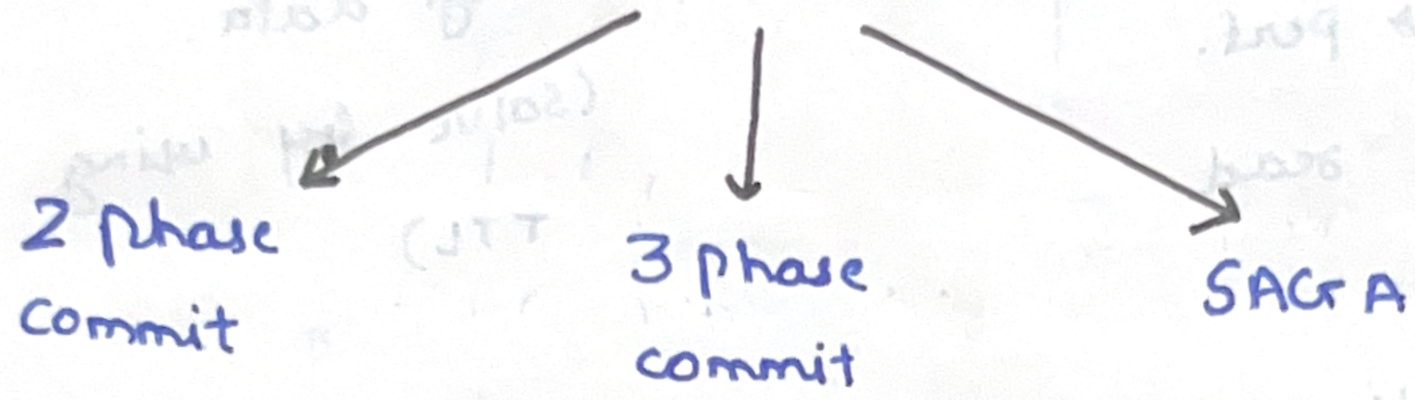
Isolation : due to Lock in DB, all transaction appear to be serial.

Durability : after Tx. completion, even if DB fails data should not be lost.

⊛ But if we have multiple databases involved in a Transaction, How would we manage ACID??

⇒ basically,

How to handle Tx. in distributed System?



2-phase commit :

if all the prepare vote returns OK, then only we move to phase - ②. (commit phase)

⇒ if any of the prepare is not OK then we abort all the Transaction (Rollback)

Possiblity of failures :

① TC Fails
② Participant fails

→ PREPARE Msg lost
→ Ok Msg Lost
→ Commit / Abort msg Lost

* TC and participats maints a log file wher for a Tx. there is information about current state

TC : Stores status of Tx & msg/vote from eath participant

P : Stores status of Tx & msg received from TC. and sent to TC as well.

Participant :

Prepare msg Lost ⇒ after timeout, abort the Tx, by Abort msg.

Ok msg Lost ⇒ TC: after TO, it will do Abort.

Commit / Abort msg Lost ⇒ participant cant take decision as it has to wait for TC.

after the service comes up, if Tx. is active there then it will TC for the status, and will do commit/abort based on status in Tc.

there is a timer with each Lock, so even if any of the service fails, lock would be released.

Commit: If both succes, Tx. is successful

If one fails, we cancel the resv. and abort.
Lock

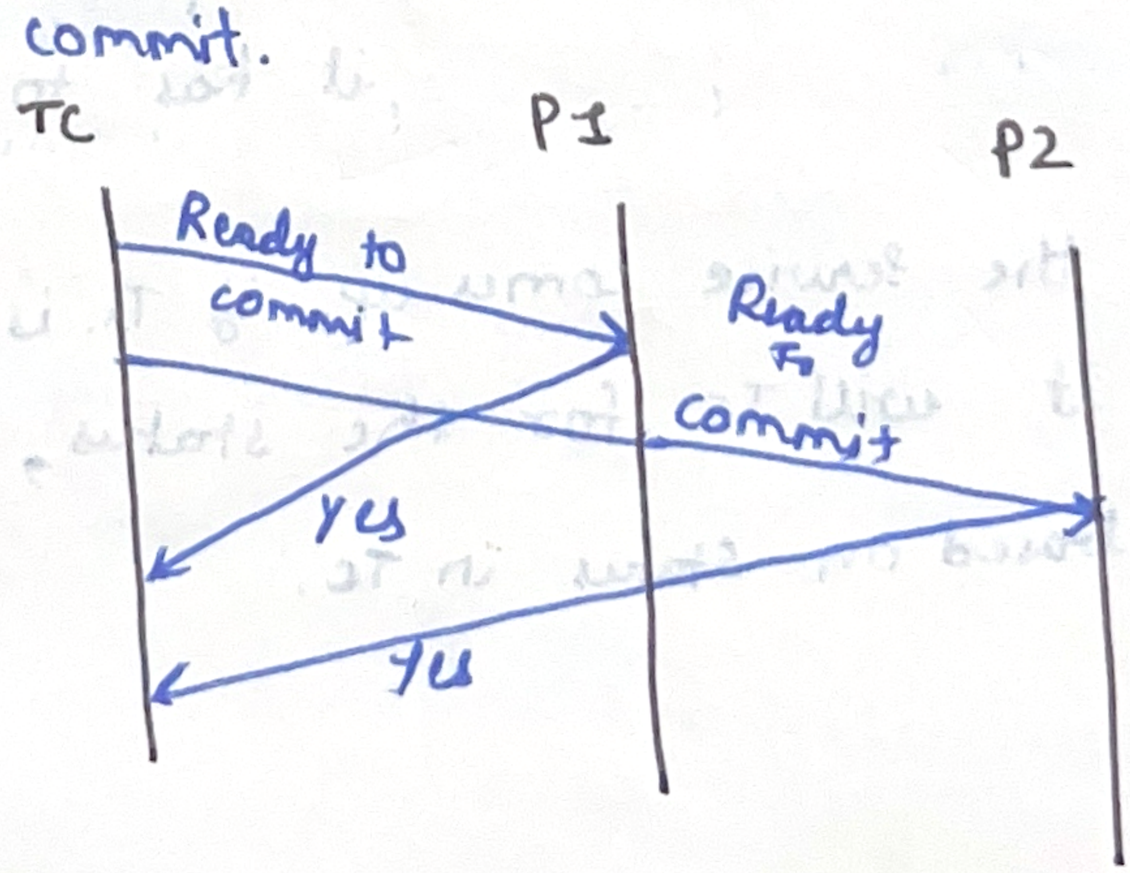
ideally time to bring up the failed service or solve the network issue would be less than timer timeout, so as soon as service comes up, it would complete the pending Tx.

If Tc fails, the Tx. would not be completed on client side as well and eventually timer cancels the reservation.

3 phase commit:

to solve the blocking problem, 3-phase commit adds an extra step to inform every participant that we are ready to commit.

Phase ②:



Phase ② advantages: this makes all the participants aware that we can commit/abort now, so everyone persists correct information, if the participant fails in this process then Tx. would be Aborted (due to failure).

If TC fails during Phase ③; Participants talk to each other and collectively decide to commit or Abort.

If TC fails during Phase ②; Participants waits till timeout for pre-commit msg, otw they abort the Transaction.

⇒ all the ops are commited at similar time and reduces/eliminats inconsistency in Tx.

SAGA: Async handling

⇒ in a long list of participants, each participants are sent query and as soon as we receive the OK, we commit on that participant, to avoid blocking. If any of them fails, we execute Rollback among already commited ones.