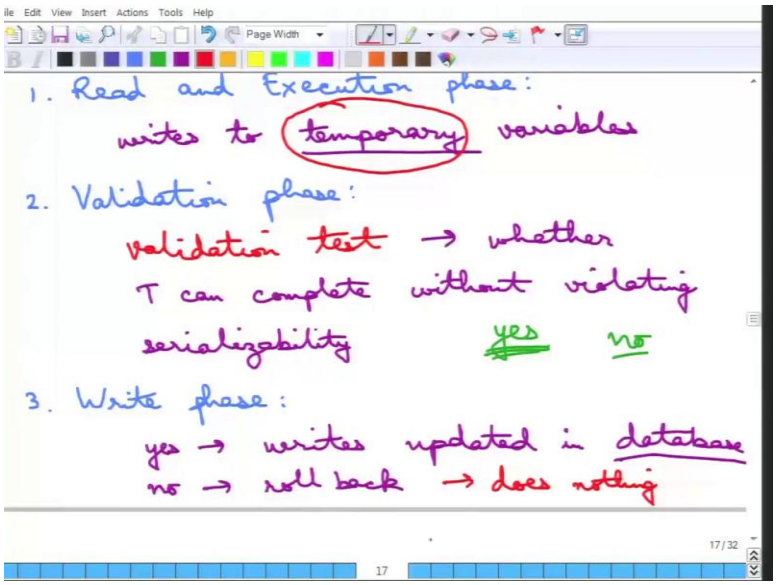
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 41
Concurrency Control: Validation Based Protocol

We will continue with the Concurrency Control Protocols, next we will be studying this Validation Based Protocol.

(Refer Slide Time: 00:16)



So, this is called the validation or certification based protocol. So, what happens in this is that, the transaction or all the transactions in a schedule has three distinct phases, the first phase is called the read and execution phase. So, what happens in this phase is that, all the transaction in the schedule, read the variables from the database and execute on them, but write on the temporary values in those databases. So, it only writes to temporary variables, not to the actual database values.

So, only the temporary values of this are modified; that is part of this transaction, this part of the schedule, so nothing is being changed in the actual database, so this is just a temporary variable, this is important. In the second phase, this, the second phase is called the validation phase, which is why this protocol is called a validation based protocol. What happens in this validation phase is that, now, it is actually being checked whether

this writes that where going through in the first phase in the execution phase and read phase can be done without violating the serializability.

So, this validation test is done, there is something called a validation test. So, the validation test essentially checks whether variables can be written, whether it can be written without violating the serializability, whether transaction can complete without violating the serializability. So, whether all those writes that are think are correct in the sense of, they are correct in the sense of serializability. So, without violating the serializability, so that is what the test is being done in this second phase, which is the validation phase.

And in a third phase, the third phase is simply called the write phase. Now, there are two things that can happen in the second phase, so the transaction checks whether it can complete without violating this serializability. So, if it can complete without violating the serializability, then in this third phase this writes are actually updated to the database. So, if it is yes, then writes are actually updated to the database, writes updated in database.

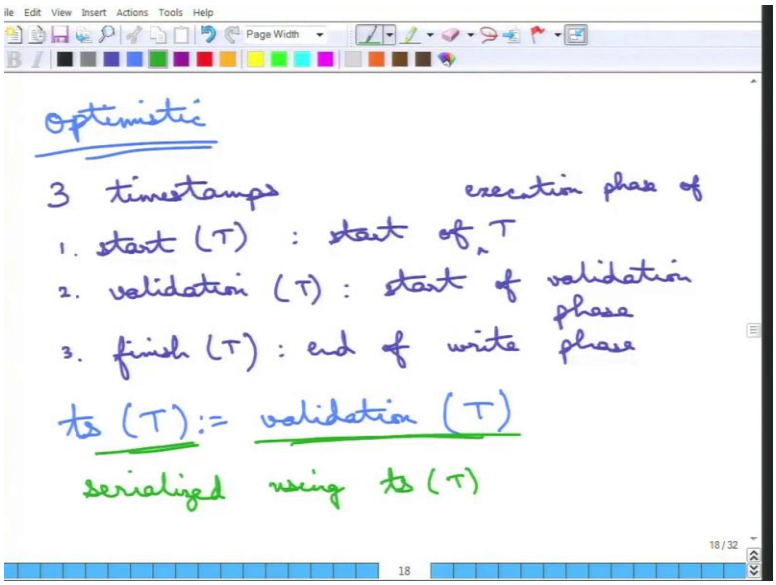
So, that is essentially; that means that all the writes that were in the temporary variables are now actually propagated to the database. On the other hand, this violating serializability may also answer no, which means that if the transaction proceeds, then it will violate the serializability. So; that means, it should not proceed, so essentially this is ((Refer Time: 03:21)) to roll back, so the transaction must roll back.

Now, what does this roll back means is that, so the transaction has not write anything to the database. Now, actually notice that the transaction has not returned anything to the database, anyway it has written only to the temporary variables. So, the roll back actually meaning, it does anything. It just simply, can does not updates the writes in the database, so it is just nothing. So, there is no problem, that way in the roll back phase, but it does the way to understand is that.

So, the critical thing about this is that, it first executes, assuming that everything will be correct and then, it validates whether that assumption that everything will be correct was valid or not. If it is valid, so the validation phase says yes, then; that means, that everything was actually correct, then it just simply goes and writes to the database; that is the writes are updated in the database. Otherwise if everything is not valid, it cannot

write anything to the database. But, it has actually not written anything to the database, anyway it has written only to the temporary variables and nothing needs to be done.

(Refer Slide Time: 04:28)



Now, this is also sometimes called an optimistic concurrency control protocol, this is sometimes also known as an optimistic concurrency control protocol. Why it is called optimistic? Because, the transaction actually does all the operations with the hope, that the validation phase will pass successfully, with the hope that nothing is wrong with the hope that everything is well, so that is why it is called an optimistic.

So, for the transaction to actually do these things, there are generally three time stamps are maintained. So, three time stamps are maintained to do the actual validation step, so the three timestamps for each transaction is the start timestamp for the transaction, this is the start timestamp and second is the validation timestamp, this is the start of the validation phase.

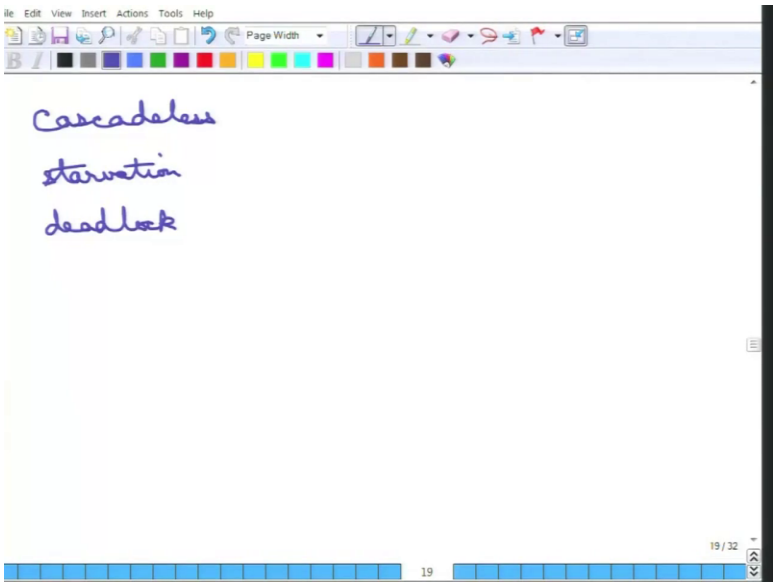
Now, this is the start of the transaction, this is the start of the validation phase of the transaction, so that is the validation timestamp, start of validation phase and finally, there is a finish timestamp, which essentially says that the end of the write phase. So, everything has gone to the database, actually being done correctly to the database. So, this is start of T, which is you can also say this is start of the execution phase of T; that is the same as saying that is the first phase, so that is the same thing as saying part of the execution.

So, these are the three timestamps that the transaction uses in this optimistic concurrency protocol or the validation based protocol. Now, timestamp of the transactions, so there is a timestamp that is given to the transaction, which is set to the validation timestamp of this. So, the timestamp, when somebody talks about the timestamp of the transaction; that is essentially the timestamp of the validation phase of the transaction, so which is essentially equal to the validation timestamp.

Now, the transactions everything is correct, then this is serialized, if everything is correct, then the serialization is done using this validation, using this timestamp, so using this timestamp T. So, that serialization is done using this timestamp T, which is the same as saying that the serialization is done using the validation timestamp. So, what does it mean essentially is that, suppose there are two transactions in the schedule T 1 and T 2 and T 1 enters the validation phase earlier than T 2 and suppose that there is no, nothing wrong in those, so the validation steps are performed correctly.

Then, the serialization order of this, of the transaction in the schedule is T 1, then T 2, because T 1's validation timestamp is earlier than the validation timestamp of T 2. So, this increases the concurrency, etcetera.

(Refer Slide Time: 07:22)



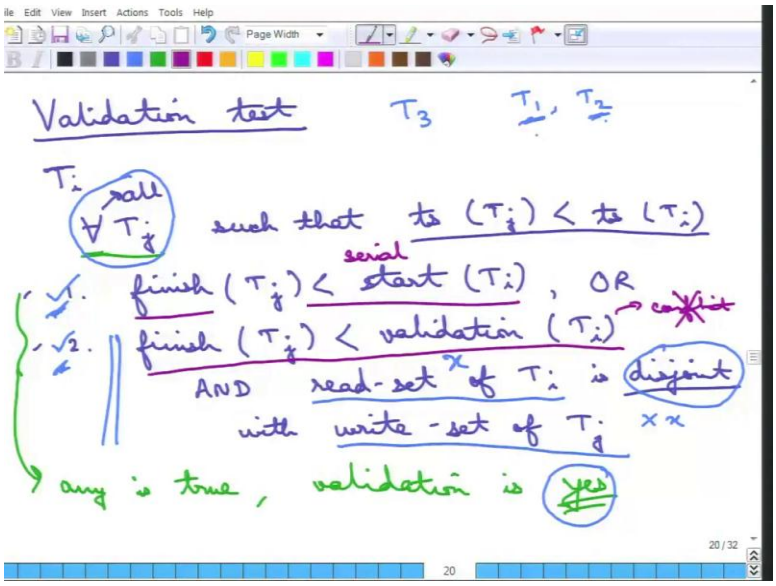
This can be also shown to be cascadeless, this is shown to be cascadeless. The reason why this is cascadeless is that, if a transaction fails, essentially it will not pass the validation phase. So, it is actually not going to write anything, so nothing needs to be roll

back in that case, so that is the thing about these transactions. However, it can suffer from starvation, now this, one must understand once this timestamps are established, the timestamps of the transaction are established using this validation thing, then the timestamp based ordering protocols can be used.

Something similar to the timestamps based ordering protocol can be used to serialize it, so the starvation may happen. In the sense, that if a transaction cannot finish, it restarts with a new timestamp and then, it must again check, etcetera and then, it may again not finished, because some other transaction has come through again and so on and so forth.

So, it keeps on starting, it keeps on restarting and keeps on rolling back or keeps on aborting, that is why it can be starved. Although, theoretically there is no deadlock, because these are timestamps based protocols and there is always a strict order of transactions. Between two transactions, there is always a strict order based on the timestamp of the transaction, so there is no dead lock. But; however, as we can see, this can have starvation, now what exactly at the validation test that is being done.

(Refer Slide Time: 08:41)



So, let us now move on to the details of the validation test. So, what how this validation phase actually does, we have been talking about the validation phase etcetera, what exactly are this validation test. So, for a transaction T i there are two conditions, that are checked for all transactions T j, so we are worried about transaction T i and for all transaction T j; such that, so the time stamps of T j is less than the time stamp of T i; that

means, all transactions that started earlier the two things are checked, the first thing is that finish T j finish is less than start of T i; that is the first check.

The second check is that the finish T j is earlier than the validation of validation phase of T i and this is very importantly and the read set. So, earlier I say in a moment what a read set is, but let me just write this, now the read set of T i is disjoint with disjoint with write set of T j. If either of this condition is true, then the validation passes, so if anyone is true any of this condition is true, then validation phase says yes the validation is yes otherwise it is no.

So, what does this mean let us go over these two conditions one by one, so we are worried about transaction T I, now note for this is for all transactions T j. So, this is for all transaction Tj it is not for a single transaction T j, so what for all transactions Tj, that has the time stamp earlier than the time stamp of the transaction T I, that we are worried about, it must happen that T j finish before T i started.

Now, what happens if T j finishes before T i started of course, if T j finishes remember that the finish T j that time stamp is when tj has written everything to the database, so if the T j is completed. Now, if T i starts after T j, then of course, everything is validated nothing is wrong, because T j was supposed to complete earlier than T i and T j finish before even T i started. So, that is fine that is this one is probably easier to understand, why the validation is correct, then.

The second condition or the condition is that T j finished before the validation started, now couple of things is happening. So, T j finish before validation started, but before validation of T i started, what is the T i phase, that it is into it is the execution of the read phase of T i so; that means, T i is actually reading from the database reading some variables from the database and writing to the temporary copies of it, but it is actually reading.

Now, what may happen is that think of an item x, so T i reads x, now it may happen that T j was supposed to write x and T j wrote x after T i did, so there is a problem correct. So, it cannot say that it has validated, because it is not clear when T j wrote. But, the and condition says that the read set of T i is disjoint with the write set of T j; that means, if T i reads x from the database T j does not write to x.

So, T j it is disjoint; that means, this they do not share anything so; that means, if x is read by T i, then x is not written by T j; that means, if T i read some variable y; that is guaranteed that T j has not written to y; that is why it is called disjoint so; that means, that the reading of y by T i is correct, because there is two transactions T i earlier than earlier to it which has read which has written to it that way also this is correct.

So, that is why, even if this condition is met, then the validation is also correct and, so that is why it can written validation as yes. So, this the two conditions, that must be these are the two conditions, so either of one of them must be happening for the validation to be passing. Just to remind you once more that this has to be for all T j, now so; that means, that, so suppose you are worried about the transaction T 3 and before transaction T 3 there is transaction T 1 and T 2.

Now, for each of these transactions either condition one or condition two must hold, so for T 1 and T 2, so it may happen the T 1 for T 1 transact condition one holds and for T 2 condition two holds, so that is fine. Either of these conditions must hold for T 1 and T 2 only, then the validation test is supposed to pass correctly. Now, what is the outcome of all of these things if these condition happens this is essentially just a serial I mean this is T j finishes and then, T i starts, so that is just essentially saying this a serial.

And the second condition says that the reads of T i are not affected by the writes of T j, so there is no conflict so; that means, there is no conflict, so that is why these two are correct. So, that is the whole thing about validation test this is how the validation test is done and that is how the validation based protocol concurrency based protocol goes through.