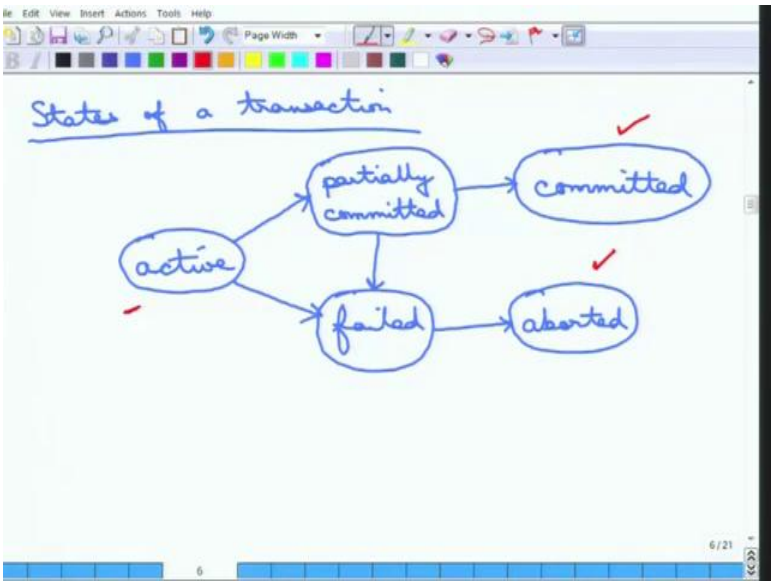
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 29
Database Transactions: States and Systems

The states of a transaction, a transaction can be in a various state.

(Refer Slide Time: 00:12)



Now, what do you mean by state is that the status of the transaction, whether it has completed, whether it has started, etcetera this is called the states of a transaction and the state of a transaction can be essentially explained using this diagram. So, a transaction can be active; that means, the transaction is running, is executing somehow. So, this can be in active state, then there is something called a partially committed state.

So, what is a partially committed state? So, the partially committed state, a transaction is said to be in a partially committed state, if the last statement of the transaction has been executed successfully, but the transaction has not itself declared to be completing successfully. So, the last statement has been done, but it is still not declared to be correct that is a partially committed.

Then of course, there is a committed which means after the partially committed thing, the transaction essentially completes everything that needs to be done and then, declares itself to be successfully completed. So, committed essentially means successfully

completed that is the committed state, then of course, there is a failed state, the transaction has failed because of different reasons, power system, crash, etcetera whatever, all those things and the system error, etcetera and that is a failed state.

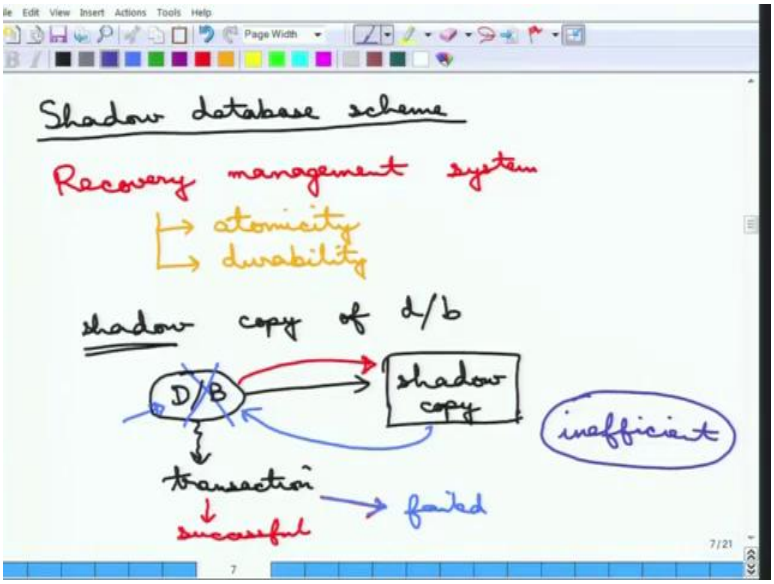
And then after the transaction fails, it has no other way undo, aborts. So, abort meaning a transaction must undo all the things that it has done and should roll back to the state before it started. So, if a transaction that transfer of money aborts, what will happen is that the transaction should roll back to the original among that a and b had, so that is the abort.

Now of course, we can draw this state diagram is that a transaction starts from active and it can go to partially committed state or it can go to failed. From a partially committed state the transaction can go to the committed state, but unfortunately it can also go to the failed state. So, after the last statement has of the transaction is successfully done, there may be still some failures.

And we will see why these failures can happen, there is a log needs to be retain, etcetera some more operation needs to be done by the database and during any of those operations, it may fail and of course, from a failed state there is no other way, we went to abort. So, now, we see essentially once the transaction starts from active, it will ends up in either committed or aborted state, there is no other way, so that is thing.

So, either a transaction commits, which makes it is completed everything successfully or it aborts; that means, it is failed somewhere and it must roll back to the database state, where before it started. So, that is the thing about states of a transaction. So, the next we will go over this certain schemes of this transactions and how to manage transactions etcetera, etcetera.

(Refer Slide Time: 03:14)



The first one is called a shadow database scheme. So, shadow database, let us explain the term what is it. So, essentially these are all form part of recovery management. So, why does transaction needs to recovers? So, when a transaction aborts it needs to recover, which means it needs to ensure that the database recovers to the original state. So, if A's money has been debited without crediting B and then, the transaction fails, then A's money must be credited back, the same amount, etcetera.

So, recovery is the one that is ensures that atomicity is maintained; it also ensures the durability has maintained. So, recovery has some more rules, so if the transaction said it has been committed, then the durability essentially says that, well even it is for some reasons the B's account has not be reflected with the correct thing, then recovery system will ensure that B will have the actual, the B's account will reflect the correct balance, if necessary by running the transaction again.

But, making sure of course, that A's money is not debited twice, but we will discuss all those things later, but this essentially that the point of recovery management is these two things. So, the shadow database scheme is the recovery management scheme, so recovery management this is called a recovery management system and the shadow database scheme is one of them and what does the shadow database scheme does is that, it enforces atomicity by maintaining a shadow copy of the database.

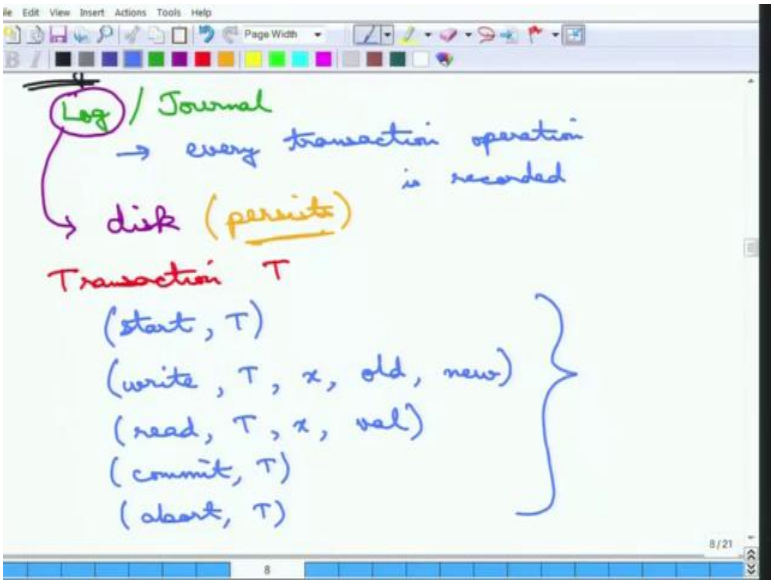
Now, what is a shadow copy? As we said earlier, there is a mirror or shadow copy which is essentially an entire replica of the database and before any transaction starts. So, this is the idea, so this is the database, where this is the shadow copy is being made, this is the shadow copy. So, this means that entire database is the exact replica of this thing and now a transaction is run beyond this database, this is the transaction is being run.

Now, if some problem happens during the transaction, then what happens is that, so now, there are two cases, if it is successful, if everything is successful, then essentially the shadow copy is being updated with the new thing. If on the other hand it fails, so let me use a different color for this if from the other hand it fails, then this database is essentially removed and the shadow copies now taken as the correct database, because this is where the transaction started.

So, the transaction meaning, were the operations are being done on the database, so there are two ways, either it succeeds or it fails. If it succeeds; that means, the new database the current database that it is working upon is the correct copy, so the shadow copy is updated. If it is fails on the other hand, then it means that the shadow copy on which it was started is the correct copy. So, then the shadow copies copied to the actual database, so this is the shadow database scheme.

Now, as you can see this is highly inefficient. Why is this highly inefficient? Because, you are coping the entire database every time a transaction starts, essentially it is not been done there are some database points ((Refer Time: 06:33)), but it is you can see that this is very, very inefficient this can be extremely inefficient. And the other important, more important problem is that when there are concurrent transactions happening, so when there are more than one transactions happening, it really cannot efficiently handled those more than one transactions in a concurrent manner. So, that is the problem with shadow database scheme for the recovery management.

(Refer Slide Time: 06:58)



So, the important thing that the recovery management them uses is something called a log, this is l o g log, log is sometimes also called, this is log or sometimes also called a journal by a database. So, this essentially keeps track of every operation that is done by a transaction. So, every transaction operation, every operation needs a transaction is recorded here in the log. So, this is like saying whenever you do anything, you maintain a copy of that or you write it down in the logs.

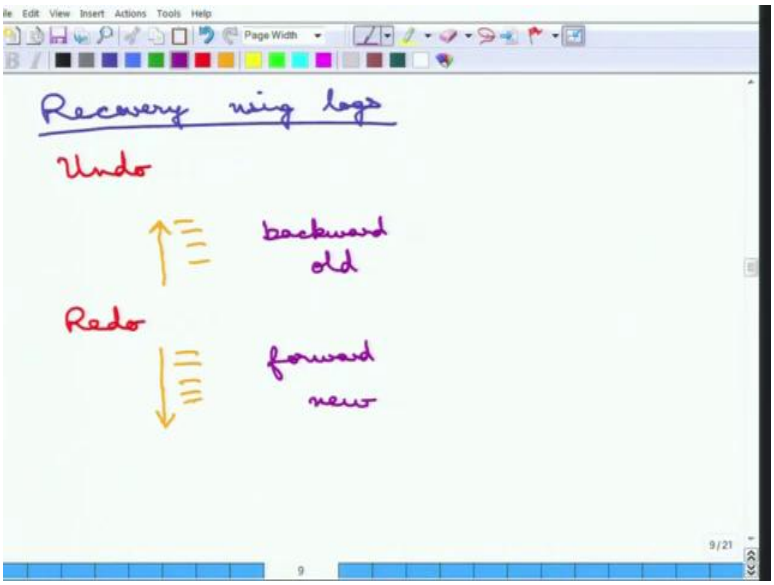
So, if you say that transaction one reads A, so that is being written in the log, the transaction one has read A, then transaction one writes B. So, that is also being maintained in the log, etcetera. So, the log if one reads the log, then the one can of the entire understanding of which transaction operated and which data item and then in which order very importantly, so that is the purpose of the log. So, log very important is that log is maintained on the disk so; that means, log essentially what is the idea is a log is persists.

So, the log can be read again and again has been written as it, because it is persists that is one thing and then log is a periodically backup to our ((Refer Time: 08:15)). So, that it is really, really persistence you can go back to any log, any position in the log as we put once, so that is because of this persist issue it is been written on the disk. Now, for a particular transaction T, suppose there is the transaction T, the couple of things are maintained in the log.

The first operation I mean it maintained is that let say start, so start is the operation in which transaction started, so it is a start T, so that is maintained. So, this is one kind of operation that is maintained, then what it is maintained is that, this is a write operation for the transaction T for the named item x, but the old value is this and then, new value is this. So, this is also maintained in the log.

Now, you see how detailed this is, so it essentially says that transaction T has return the value on x and it replace the old value with the new. Then, it also says read, so transaction T has read the value of x and the value that it is read is this value. So, it has read, essentially the when the transaction T read x, the value was val. Then, of course, there are this commit T and abort T. So, these are the five operations that the transaction writes on the log. So, these are the five operation that the transaction writes on the logs and how do we recover using logs, so the recovery using logs is done in the following manner.

(Refer Slide Time: 09:46)



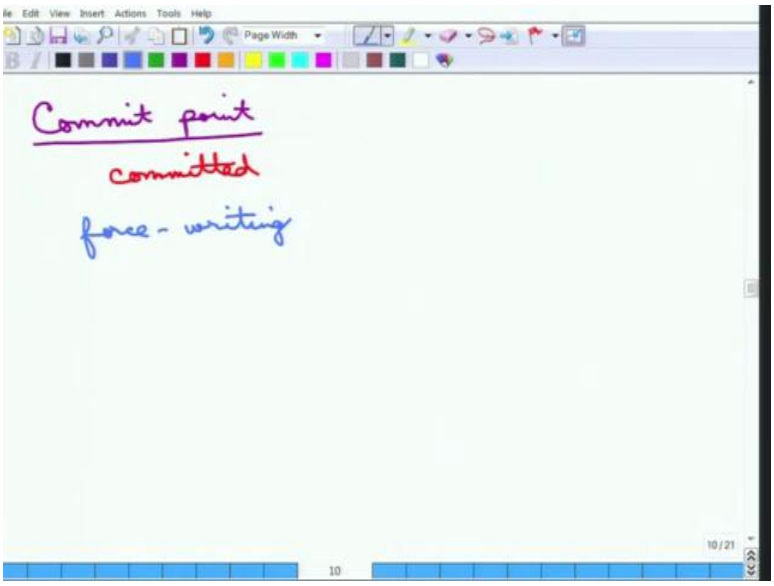
Now, we will try to build a recovery system using logs, recovery using this logs that we just explained. So, the first thing is that, there are two important operations, the first operation is called undo. What is the undo essentially? Suppose A's account has been debited, so how does one undo, essentially the log will say that A was read, this was the value that was read, then transaction T wrote A and this was the old value, this was the new value, etcetera.

So, if these all those things are maintained in the log, then to undo the log can be travels in a reverse manner in chronological order; that means, this was the latest in time point and this was the earlier in time point. It can be reading the reverse manner and each of this can be undone, so the old value might be written back etcetera, etcetera; that is the way of doing the undo.

And then, there is another operation, which is the read T, which is essentially there is a metalize of that. So, the transaction was supposed to do something, but it could not and for durability etcetera, it make sure that these values are there. So, it is read in the correct order the forward order, this is the called backward order, this is the forward order and then, those things are said to the new values.

So, once more, so in the undo the log is read in a backward order and the write values are going to their old once and in the redo, it is read in the forward order and the new rights are updated to the new value. So, this is essentially how the logs are used to recover from a problem in the transaction.

(Refer Slide Time: 11:30)



And then, a transaction is said to reach it is commit point, now this is not commit, this is the commit point. So, the definition of that commit point is essentially, when all that transaction, all the operation have been done correctly and they have been recorded in the logs. So, both the things must happening, all the transaction all the operations, then

the transaction have been executed successfully plus all of them have been recorded in the log.

So, if the recording in the log is not taken place, then it does not said to reach the commit point, the transaction does not read the punishment, unless all of those are recorded in the log. So, now, beyond this commit point, so once the transaction reaches the commit point and after that, it is said to be committed; that means, once this is written on the log and everything has been done successfully, then the transaction is said to be committed.

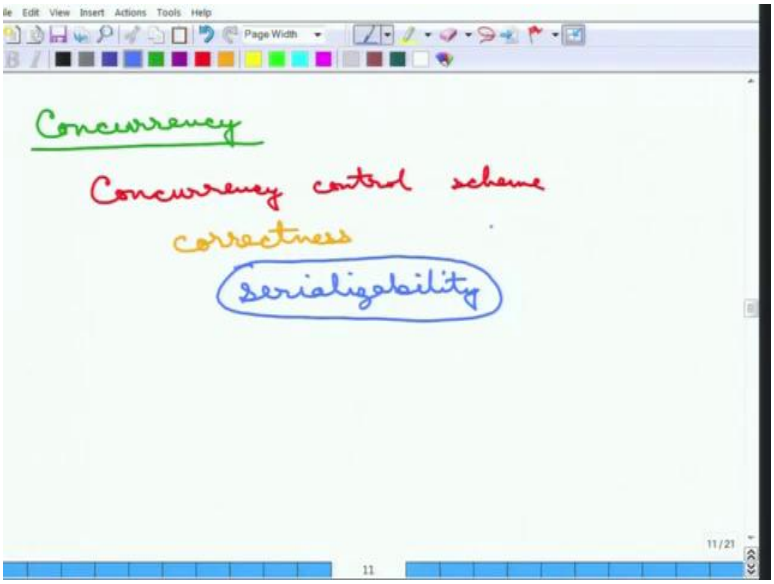
That means, that the transaction now cannot say that, I will undo etcetera it has do only redo and make sure that, the effects of the transaction are permanent in the database. So, that is the commit point and then, once that is done the commit T increase made in the log. So, this is the commit T and on the other hand, if there is an abort T, then that means, that undo operation have to be done and that is the easy to understand.

So, if A away once a transaction reaches the commit point, the logs etcetera must be written on the disk, the logs must be persist in the disk. All the transactions that are done, must also be persistence on the disk. Now, here is the point is that, how does transactions happen, so the other database commits this transaction is that, the brings the necessary data item from the disk to the main memory and changes the value in the main memory, because that is say only it can do.

So, when it is says to there is a commit point, it has to be makes sure that all of these write, which is suppose to take place on the disk has taken place on the disk. So, if necessary, there is a disk flash operation; that means, all the things that are in the disk buffer etcetera is actually gone to the disk. So, the actually the disk contains the new copy; that is called a forced writing.

So, if necessary to reach the commit point everything that is changed on the main memory must been force written on the disk, this is to make sure that the disk contains the correct versions momentum, what is that. So, this ensure that the redo operations can be done successfully. So, this is a point of recovery of transactions.

(Refer Slide Time: 13:54)



Now, there is another thing, which is called a concurrency issue of transactions. Concurrency essentially means that more than one transactions are been executed in the database and there may be accessing the same data item, otherwise there is no problem. Anyway why does concurrency is useful is that the of course, increase the time to did. So, this is much more efficient etcetera and so it increases the utilization of the processer and the disk and CPU and all those things and the average response time the average completion time for transactions is reduced, so that is the thing.

But, there may be problem in the sense is that, one is trying to read the data item, while the other transaction is trying to write. So, the concurrency controls schemes, so this is called a concurrency just like we had recovery systems, this is the concurrency control schemes. Must ensure the correctness of the these, correctness meaning, if one is trying to read a value that it should read, concurrency control schemes ensure that it reads that correct value. So, this is called a concurrency control scheme.

So, the important operation of this is the correctness, it must ensure the correctness of the two transactions; that is the important thing and to do that, we define those motion of serialize abilities serialize abilities is the formal way of studying whether concurrent transactions are correct or not. So, serialize ability is the actual formal way of doing this. So, that is the introduction of transaction and what transactions are and a little bit of log etcetera. Next, we will go over the recovery management systems in much more detail.