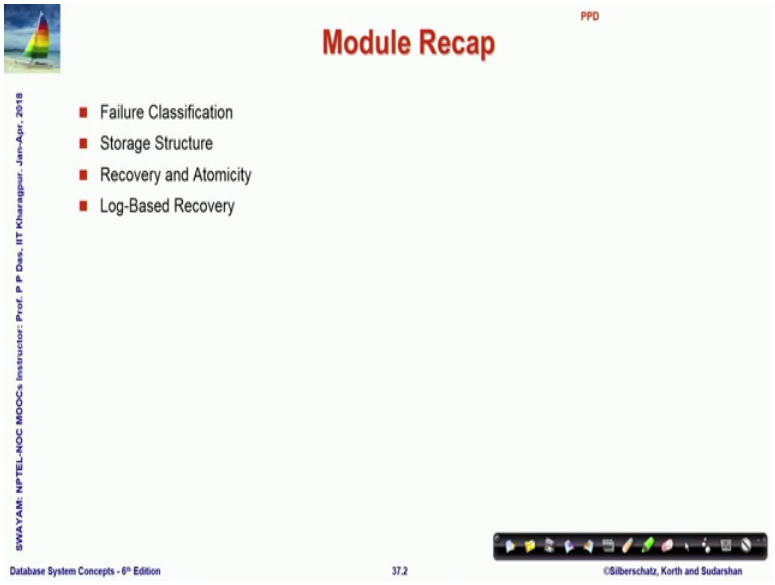
Database Management System
Prof. Partha Pratim Das
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 37
Recovery/2

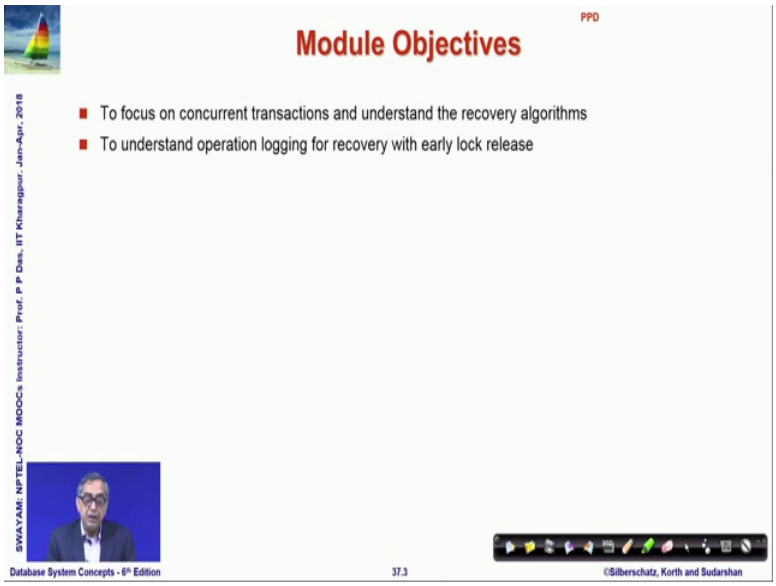
Welcome to module 37 of Database Management Systems. We have been discussing about Database Recovery. This is the second and concluding part of the Database Recovery.

(Refer Slide Time: 00:25)



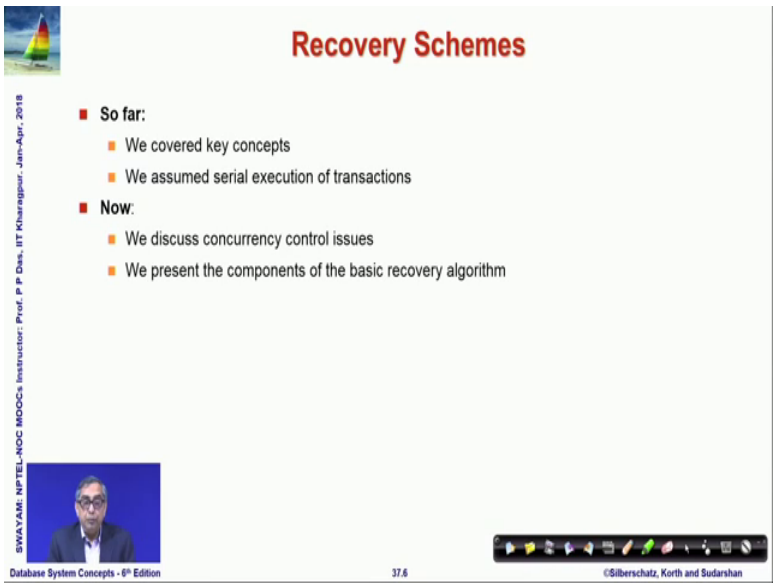
We have earlier discussed about failure classification, storage structures and significantly the log based recovery mechanism.

(Refer Slide Time: 00:34)



In this, we will focus on concurrent transactions and understand the recovery algorithm for them and we will understand the operation of logging for recovery with early lock release. We will learn about another kind of logging mechanism; so, the recovery algorithm.

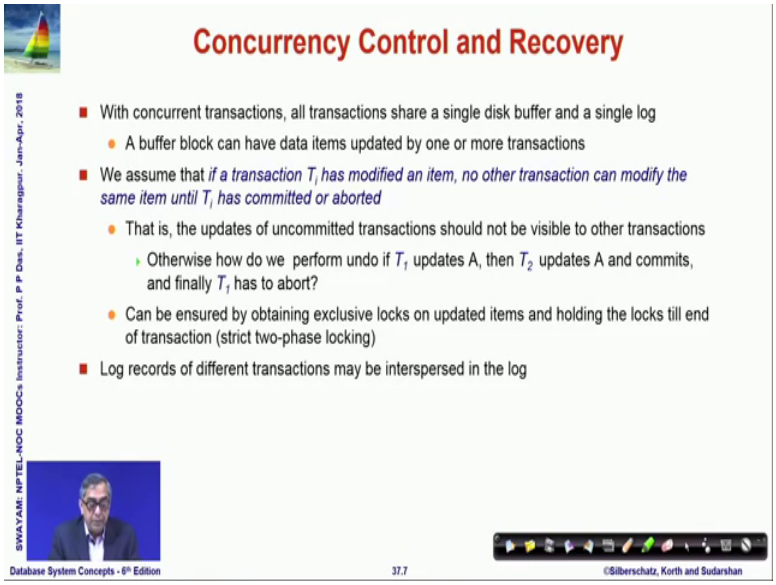
(Refer Slide Time: 00:51)



So, what we have seen so far are we have learned the basic concept of recovery and logging and we have assumed the serial execution of transactions and now we discussed the Concurrency Control issues. So, now, we will assume that there are multiple

transactions operating at the same time and the components that are required for the recovery of those.

(Refer Slide Time: 01:10)

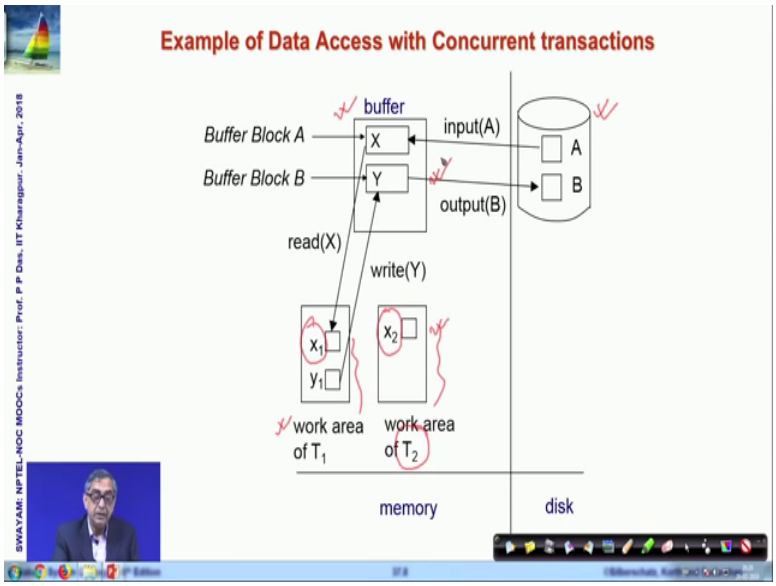


So, with Concurrent Transaction, all transactions share we already know that every transaction is a private work area that assumption stays, but we talked about a system buffer area.

So, that system buffer area the that area would be common for all different transactions, also the log area would be common for all transactions. So, now, the in the buffer area the, data rights are or reads or writes are done for different transactions and in the log the different logs of different transactions are fixed up. So, we make one assumption that if a transaction has modified an item, then no other transaction can modify that same item unless that transaction is committed or aborted.

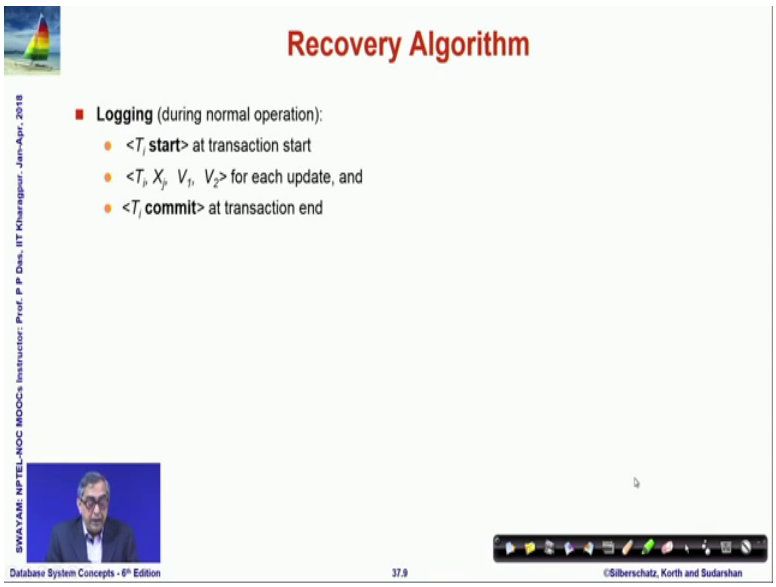
So, which means that kind of when the transaction modifies the item it holds a lock and that lock is held till the end of the transaction and this is a I mean if we if we think back in terms of our locking protocol, this is a strict locking protocol that we are talking of. This is important for recovery because if we did not have this, then it is possible that multiple updates to the same rate item are done by multiple transactions. So, if we have to undo that then we will not know we were which one it to be undone with. So, that is the basic problem. So, we will assume an exclusive lock in this case and log records will be written interspersed as we have already saw.

(Refer Slide Time: 02:43)



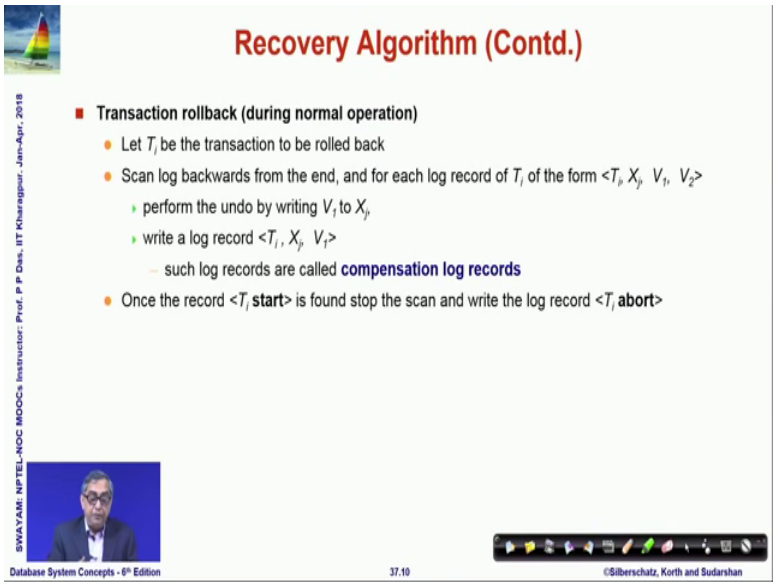
So, in terms of our storage access mechanism, the same eh earlier diagram, this is here is a disk, here is a buffer, the buffer is common and the private work area. So, now, in addition to T 1, we have another transaction T 2 with it is own buffer area, but so, the x has been written in T 1, has been read in T 1 as x 1, x has also been read in T 2 as x 2 and each are concurrently making changes in that private work area, but they are using the same system buffer area for the for writing the output back to the disk or reading directly from the disk. So, this is a model that we will go with.

(Refer Slide Time: 03:30)



So, what is the recovery algorithm, first is logging and the logging structure remains same; the start transaction log, the update transaction log and the commit transaction log as before.

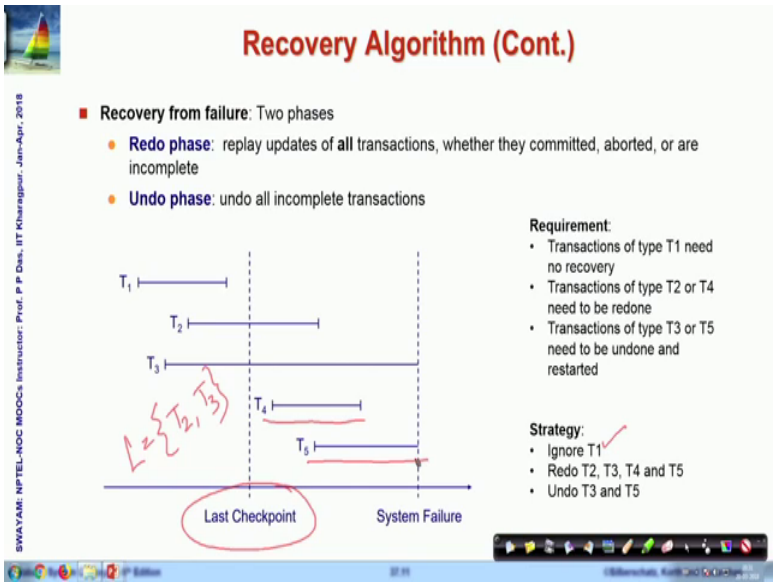
(Refer Slide Time: 03:43)



When you have to do a transaction rollback during normal operation; so, for that transaction T i to be rolled back, what we will need to do it is a rollback. So, undo has to happen. So, scan will scan the log backwards from the end and for each log record update log record, we will restore the original value for which was written over and we will write a compensation log record as before and going backwards in this way when we come across the start log record, then we will stop that scan and write a abort log record in that place.

So, it is exactly same to what we did.

(Refer Slide Time: 04:21)

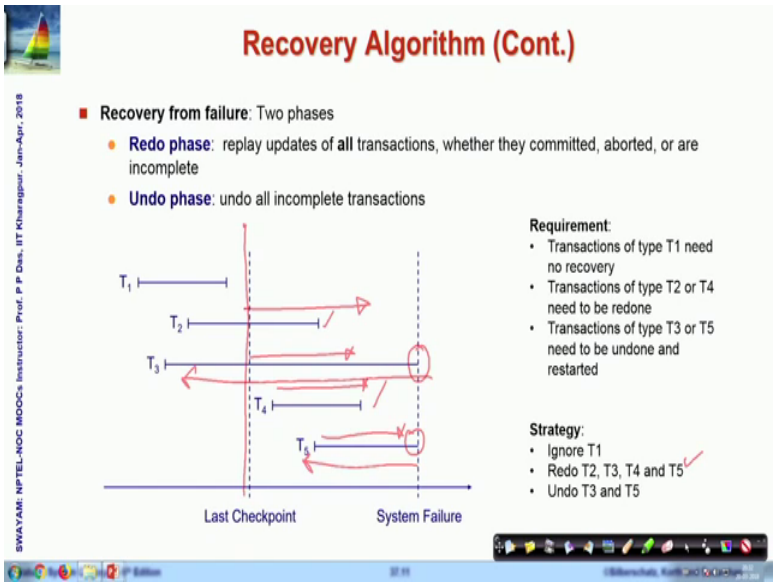


So, now let us look into the actual Recovery Algorithm. So, the transaction rollback has no difference. So, in the Recovery Algorithm, what we do we have a recovery phase and where we replay updates of all transactions. So, we make sure that all transactions whatever they did they those are done again. So, after the failure we recover from the failure. So, we up do all that again whether they are committed, whether they are aborted, whether they are incomplete in every case and then we keep track of what are the transactions which did not complete and for them we do an undo phase. So, here I am showing another example here.

So, this is the last checkpoint where eh all updates I mean freezing the updates, everything was output to the disk the log as well as the data item updates were put to the disk and the set of transactions that are live during that time well execution in that time were recorded. So, if we look at that set L in this case, then it will be T 2, T 3 these two transactions.

So, we can we have already seen that our strategy would be that we will ignore T 1 because it had completed before the last checkpoint. T 2 and T 3 were ongoing and then T 4 has started after checkpoint and committed before that, T 5 started after checkpoint, but was also active was also in execution when system failed. So, our strategy would be, that we will assume as if this, this whole thing as is redone.

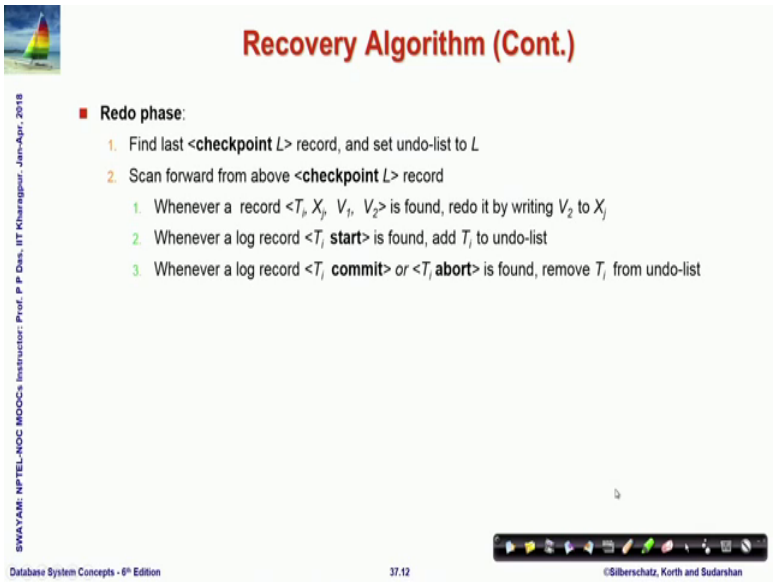
(Refer Slide Time: 06:06)



So, T 2, T 3, T 4, T 5 all these log records exist. So, we will follow through them and redo all of them. If we redo all of them then naturally we come across T 3 and T 5 which cannot proceed further because the system had could not proceed further because. So, we do not know in terms of the log what would have happened to them because the system had failed.

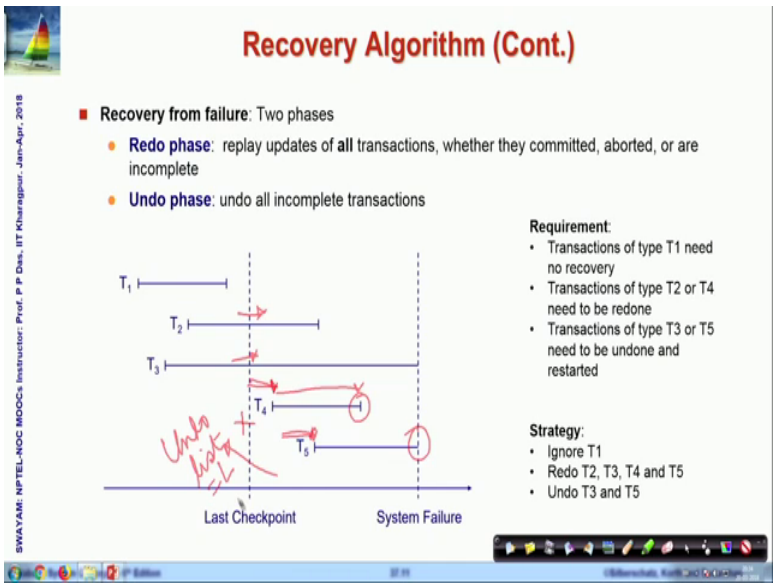
So, after having done this then, we do an undo phase where we undo this, but naturally the effect of these will remain. Now you can question that this could have been done in a more smart way, do we really need to redo everything and then undo some parts of that, that is a override in terms of that which is true, but this just makes the whole algorithm simple and over it actually is not very hard.

(Refer Slide Time: 06:55)



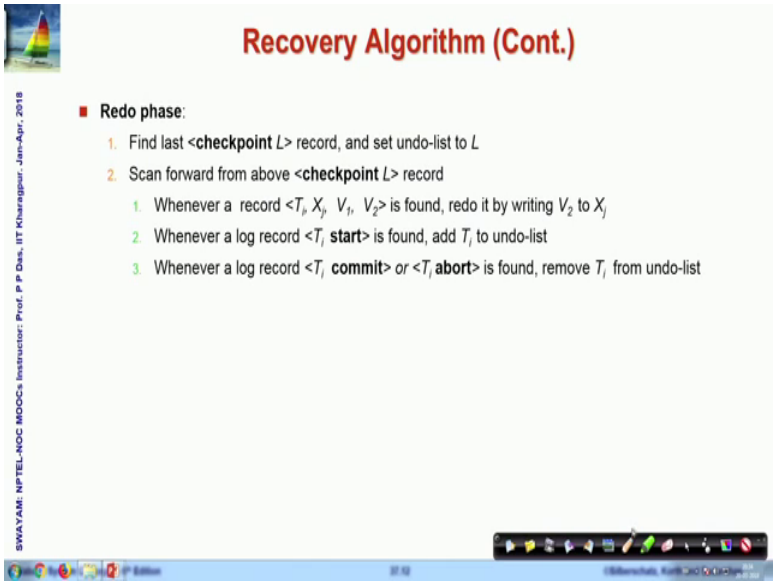
So, we are doing the redo phase, even the redo phase you will find the check point and you will scan forward from the checkpoint record and as you scan forward from the checkpoint record; if you have an update, you will simply redo which means V 2, will again be written to X j and when you find a start transaction, then you do not know. Just look at this point carefully; if you find a start transaction for example, when you are working on this, suppose you come across a start transaction here, you will come across the start transaction transactions start here.

(Refer Slide Time: 07:31)



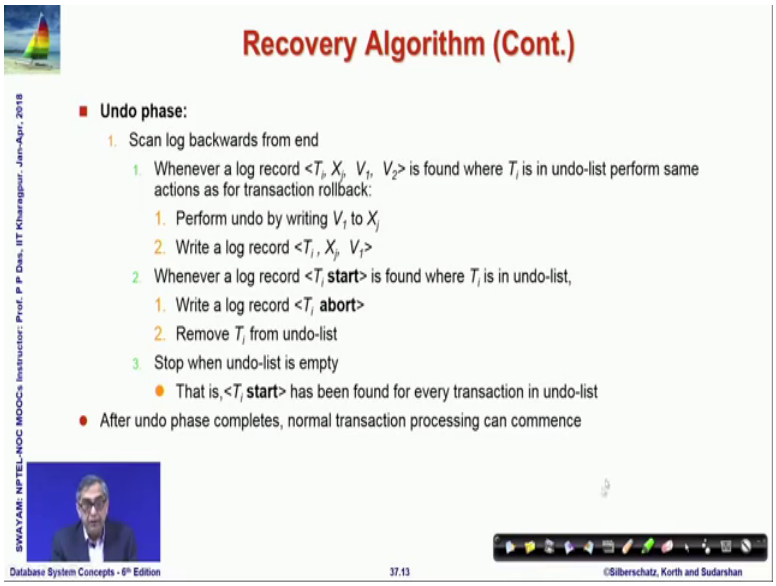
So, whenever you get that, then you put this into the undo list. Initially your undo list is L because they were going on. So, you do not know that they could finish all that still need to be undone and when you come across a new start, you add that to the undo list and then the rest of it is simple. So, you keep on going this way, if you find that the commit has happened or abort has happened, you remove that from the undo list, but if you do not find that then that stays in the undo list. So, you know if you, if you proceed from in this direction in the redo phase, you know and that way when you have scanned the whole log, you know what are the transactions which still need to be undone. So, that is a simple strategy that is followed.

(Refer Slide Time: 08:26)



So, ma whenever you have a log record start, then you put it to the undo list and whenever you get a log record which is committed abort which says that before the system failure the transaction actually had either committed that it finished everything or you had to roll back, then you remove that from the undo list. So, what will be left out, at the end will be the undo list of transactions that need to be undone subsequently.

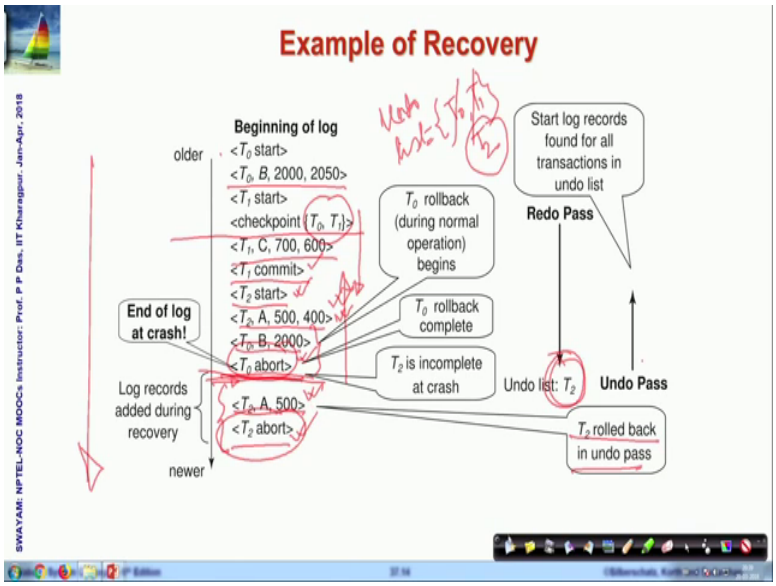
(Refer Slide Time: 08:53)



In the undo phase, in the undo phase, you go backwards because it is undo. So, what you will do is a very similar. So, if you have an update record, then you undo in the ma transaction which is in undo list then you do exactly in terms of transaction rollback that you write the old value and write a redo only log record. Now when you find going backwards, when you find ti start; so you know that this is a starting point of the transaction and the transaction is in undo list. So, it came across because it could not be it was on the undo list. So, which means that it could not be completed and therefore, you have found the start. So, this is where your undo operation is over. So, you write a abort log record and once you have written that, then you are done with the transaction. So, you remove that from the undo list and in this way, you will continue till your undo list is becomes empty.

Once it becomes empty, so, then you have found that T i start for all transactions in the undo list and there is nothing more to do. So, after undo phase completes normal transaction processing can comment. So, your failure recovery from the failure is already taken care of.

(Refer Slide Time: 10:14)



So, here are certain examples which you could check out, here are a start. So, this is the how that this is the order in which the transactions are going and this is the crash point, these is where it failed and mind you. So, this is where and this is where our checkpoint is. So, at checkpoint you can see that T 0 and T 1 are; what are your candidates? So, when you start in the redo phase, you start from this point because before that everything has been done. You naturally, you come across this. So, you redo this which means you again actually change C from it is old value 700 to 600 and then T 1 commits. As T 1 commits, you know that this transaction is done with.

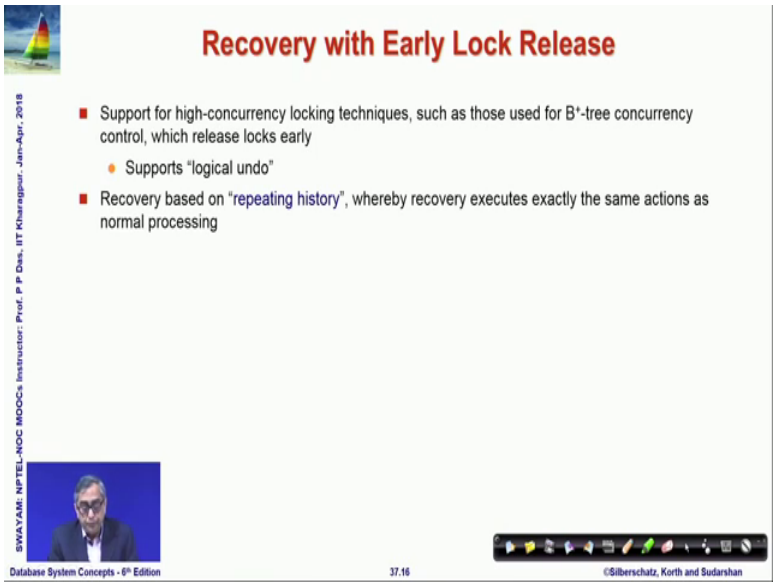
So, you remove that. So, your undo list undo list at the beginning is T 0, T 1, but going in the forward direction when you come across T 1 commit, you naturally that from your undo list, then you come across T 2 start. So, you know that another transaction is starting now. So, it may be you do not know whether it come could complete or you could not. So, you add that to the undo list then give effect to this update, then if for T 0 we have you have a rollback record that is because T 0 actually you can see that T 0 has aborted. So, the change that T 0 had done earlier this had to be rolled back, this rollback is a normal transaction rollback, this is not because of the failure. So, this mm rollback had happened and this is where the rollback is complete.

So, T 2 e, T 0 is also completed. So, T 0 after this is taken care of, then in the redo phase T 0 is also complete and this in where you reach the crash point. So, your undo list is left

with only T 2. So, now, when you have done this, so when you have taken done the redo here that T 2 which is ongoing is there, then you write this log record. So, this these log records are written during recovery not during the original transaction and T 2 had to abort because of the system failure.

So, T 0 support was due to the transaction rollback, but T 2 s abort is because of the system failure. So, T 2 is rolled back in the can be rolled back in the undo phase. So, once this has been done, then you do the undo phase starting with T 2 and then you go backwards as you go backwards here. So, you will undo this, this is what you write you come across T 2 and naturally you have rollback. So, you write T 2 abort. This is how the actual rollback can happen and you can see that now the with this redo undo phase you can always bring back the database to a consistent state and these transactions are executing concurrently and therefore, your log record is a intermix of the log record of different transactions. Now the last that ma we would like to talk about is Recovery with Early Lock Release.

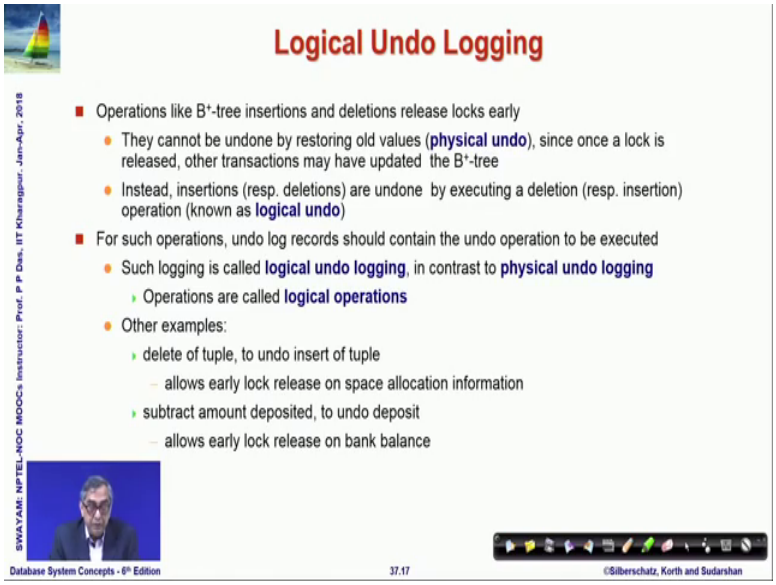
(Refer Slide Time: 13:55)



What this means is well, so far we have talked about recovery which is only in terms of data update, single data updates. So, I mean if I want to recover I can just you know write back the old data, but this is not true in case of many other situations for example, if you are inserting a record in a B-tree, then it is not enough only to undo that because you cannot undo and get back the same.

As you can understand that if you make inserts or deletes in the B-tree, if you are made an insert then the structure of the B-tree itself has changed and after that several other inserts, deletes may have happened. So, if you now want to just go back and undo this particular insert in terms of values, it is not possible to do that. So, when you want to do that, so you cannot do really kind of repeating the history kind of strategy.

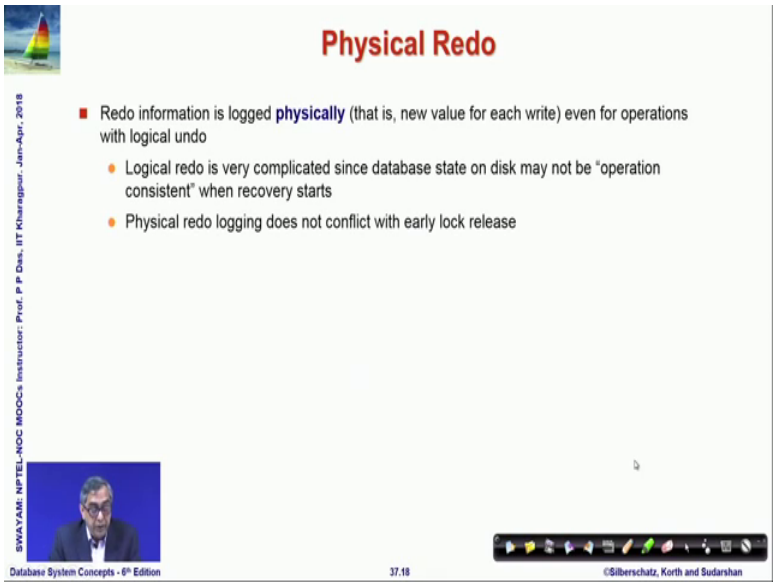
(Refer Slide Time: 14:53)



So, what you have to do is do some kind of an undo which is logical. So, so far the undo was physical that, you wrote this, you change this value by this value. So, your undo is a physical. So, you restore the original value and your undo is done here. It is logical that is for the operation that you have performed, you try to find out a matching operation which creates the similar effect as of undo. So, if you have inserted, then you your undo is a corresponding delete of that record. If you have incremented by 10 then you can say that your corresponding undo is a decrement by 10. So, that is what is known as the logical undo and it is logical undo is a very good option in case of delete of, insert delete of people.

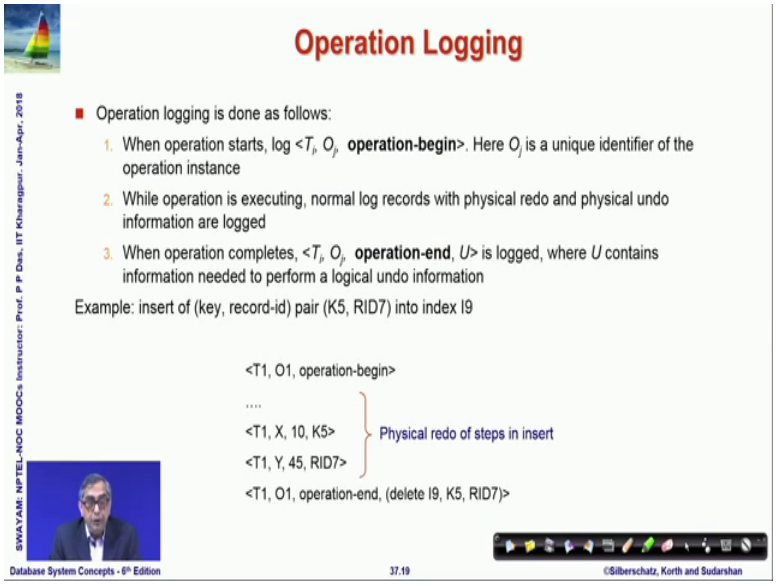
So, if you have deleted a people undo to insert, if you have subtracted then undo to undo deposit to go forward and so on.

(Refer Slide Time: 15:55)



So, a redo information is logged physically, so new values for each right even for operations which are logically, which has logical undo. So, you do not do a logical redo I mean, I will not go into the details of why this is not done, but it simply makes things very complicated. So, physical redo is always physical and you can show that physical redo does not prohibit this kind of operations that we are trying to do, but the logical redo is not used. We will only use logical undo operation.

(Refer Slide Time: 16:40)



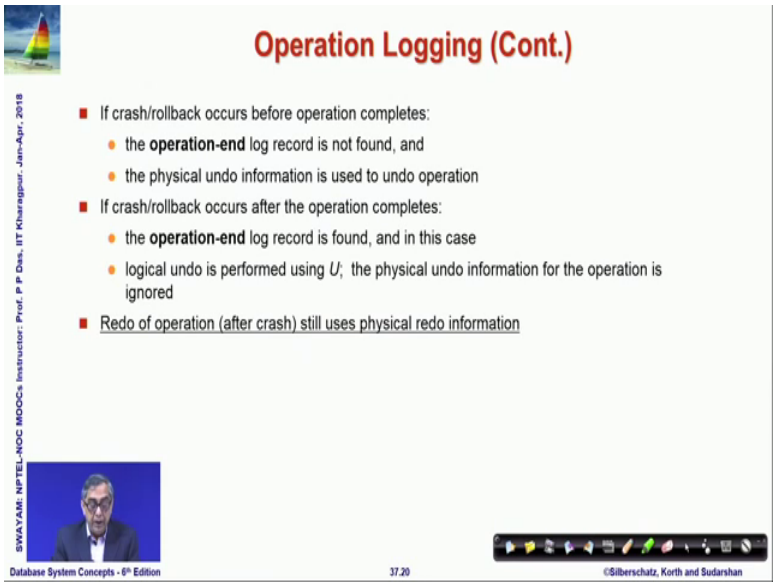
So, how do you log for such a logical undo operation, what you do is instead of now. So, now, it is an operation it may not be an a single value update. So, it is not captured in terms of one you know log record, but it could be a number of log records which have actually done 3, 4 different changes to make that operation happen and you want to actually define an undo for that operation. So, when you start this. So, you start with a log which says that what is the transaction and what is the operation. So, you put an identifier to the operation and then you write operation begin.

So, you know this is where operation has started, then all the things that are happening for this operation while the operation is executing then you write normal log records with physical redo physical information. All these logs are written and when this operation ends mind you, this is a particular operation you are talking of. So, not the whole transaction whole transaction will continue when that particular operation ends, then you write an operation in record and along with that you write, what is a logical, what is a logical undo information you put that in.

So, let us have a look at the example. So, suppose your operation is insert of a key record pair, so, let us say this is the key record pair and into index I 9. So, this operation starts here and then there are several steps to be done; for example, you will have to say if X is on the key value which had 10 and is becoming K 5, you will have a physical update undo record of this. If Y is the record id which is RID 7, then it y changes from 45 to. So, these are all physical redo steps in insert. So, these are the different instructions in terms of this broad operation and when you are done with all that then your operation ends and you write this undo information. So, insert of, so you had insert of this record with index 9.

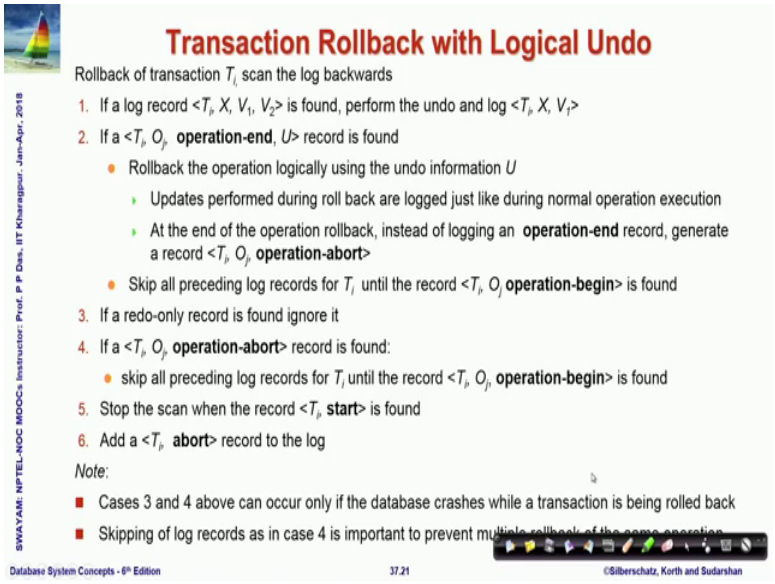
So, now you do write your what will be the undo, to delete that from index I 9, to delete this key record ID pair. So, this is a whole locking that we do. So, you can make use of this undo operation in terms of your recovery process.

(Refer Slide Time: 19:22)



So, if the crash or rollback occurs before the operation completes, then operation and log record is not found you will not find it. So, you do not know what is the undo operation. So, in that case the physical undo information is will be used to undo, but if we have a crash on rollback that happens after an operation completes, then the operation end log will be available and in this case we will use the undo operation that is given in the operation end log record and do a logical undo. And we will ignore all the physical undo information that the operation that that we will find in the log records. Redo of course will still use the physical redo information which is there.

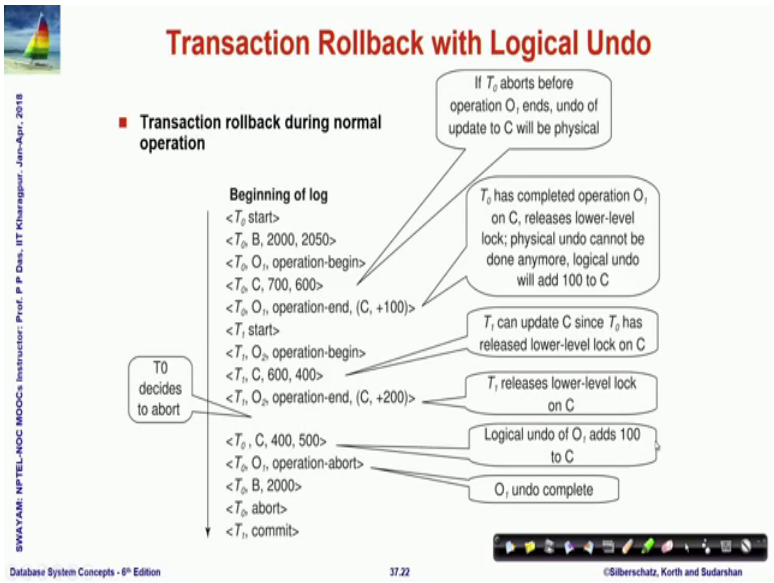
(Refer Slide Time: 20:10)



So, if we look into the actual if we if we look into the transaction rollback with logical undo. So, if I have an update record which is naturally physical, and then we can perform the undo which is as we did last time the creating a redo only record. If I find an operation end record, then to rollback we will pick up the logical undo information from you and we will perform that operation. At the end of that we will certainly write the operation abort record to show to mark that this operation has been aborted.

So, if we have a redo only record, then we will ignore it and if we find an operation abort, then we will skip all the records that were found till the beginning. Naturally, you can you can you can understand that 3 and 4 will not happen in a normal course of transaction, it will happen only when the failures have happened during recovery and at the end we will add T i abort record to the log. So, the critical thing to remember in this that whenever we are doing operation hmm unlogging, we are doing undoing based on the operation logging then since once we get the operation ends since we know what the undo information is, we have to make sure that through the undo process we actually ignore the physical undo records that exist in the log and just go with the operation case. So, this these are the notes I just mentioned it ok.

(Refer Slide Time: 22:15)



So, this is an example which you will have to spend some time and understand with care. So, you can see that a transaction T 0 has started, this is where it has done a physical update, is a physical undo record and then it does an operation. Of course, it is a simple

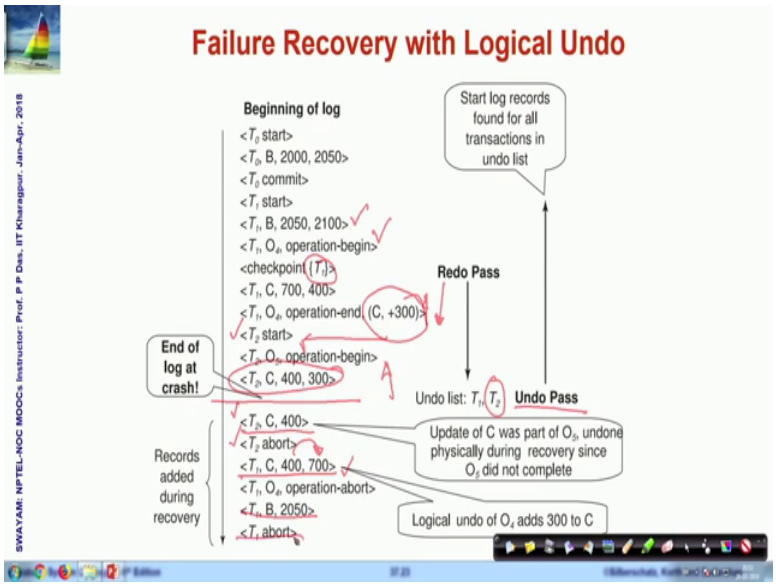
operation which changes the value of c from 700 to 600. So, naturally, so it has decremented by 100. So, your undo operation here is incrementing by 100.

So, if T 0 aborts here, if T 0 aborts somewhere here you know before your operation end has happened then naturally the undo will have to be based on this physical undo structure. So, you will have to replace 600 by 700, but if it happens after this, then this is the case if it is completed the operation and then the failure happens, then you will do the logical undo that is whatever the value of C is you will just logically add 100 to that. But that means, that when you go backwards from here to find the begin, you will actually have to ignore this physical undo record because you have already given effect to that in terms of the operation undo that you are doing.

So, this is the basic difference. Here are different subsequent examples on that and you can you can see that well here after the operation end has happened, then possibly it has released the T 0 has released a lock on C 1. So, T 1 has again taken the log. T 1 has again done the updates. So, then it releases that and T 0 at this point might decide to abort; if T 0 aborts, then this logical undo of O 1 this operation will add, it had to add 100. So, it adds now this C had become 400, now it is adding 100 back. So, C becomes 500 and then the operation is finished. So, you write operation abort and O 1 undo of O 1 gets completed, but you still have after going backwards in this, you still have this record which was directly updated.

So, these are the undo transactions, undo lock record for that where B is being restored from 2050 to 2000 and you record the transaction abort for T 0, T 1 eventually has committed here. So, this is how the transaction rollback will happen when logical undo is also used and this is a very powerful way to take care of that.

(Refer Slide Time: 25:13)

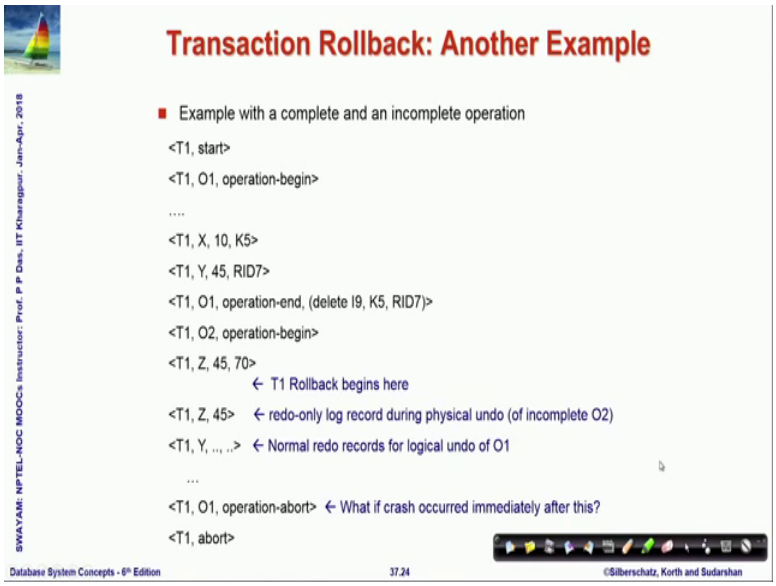


This is similarly another illustration for doing the failure recovery for with logical undo. So, here is the undo is a re redo phase that you are seeing here, this is where the end of log at the crash these are redo phase because these are check point where T 1 was there. So, at the end of redo T 1 if you if you. So, you are starting to redo from here. So, you have done operation end. T 1 has not finished T 2 has started. So, you have added T 2 to the undo list and when the crash has happened both of them are on the undo list. So, they have to go through that undo pass. So, we undo T 2, C, 400. So, this is what this is this is how you will go.

So, this is the first thing you undo and then naturally you have come to the beginning of T 2 start. So, you abort and you are going back again and you are trying to do this. Why are you doing this because when you go back to undo from this point you come across this operation end which tells you that the undo operation has to happen by incrementing C by 300. So, C which had become 400 is now incremented by 300. You come to the check point which is the end here in terms of the operation begin and naturally you declare operation abort and going back further this is what you had when transaction T 1 had started.

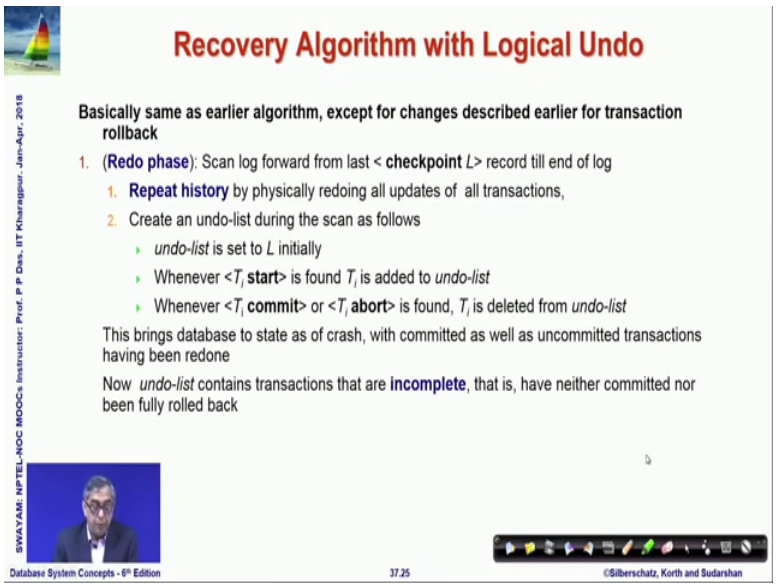
So, you undo that. That is again a physical undo and finally, t one aborts. So, this is how in both cases of transaction rollback as well as in terms of the failure the recovery can be done with the logical undo process.

(Refer Slide Time: 27:10)



Here I have given another example. I will not go through it step by step. So, at a arts that you go through that following the same logic and convince yourself that you understand that how this transaction rollbacks with physical undo as well as logical undo is taking place.

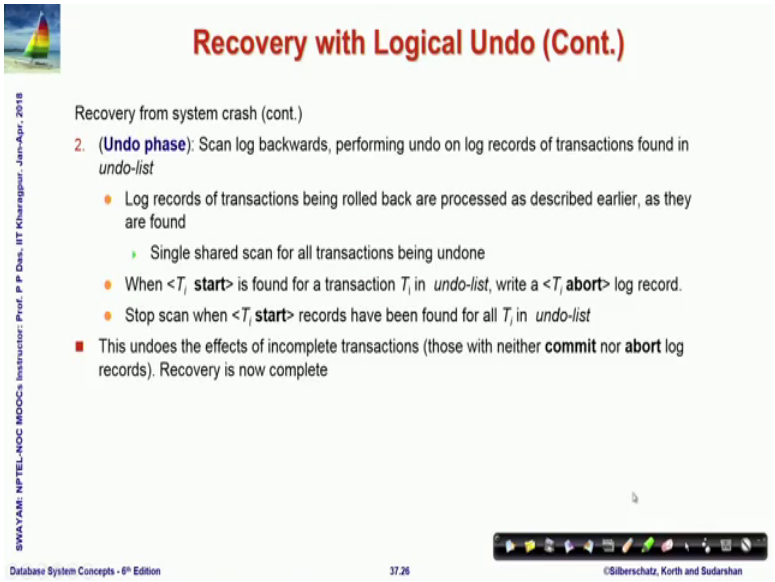
(Refer Slide Time: 27:27)



So, ma with this Recovery Algorithm with logical undo will look very similar to what we have already done with the physical undo redo and though that is what we have stated

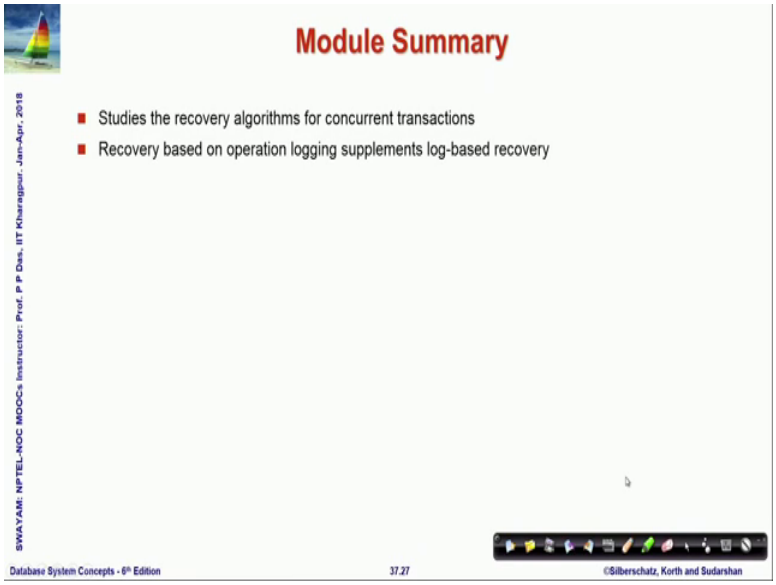
here, there is no nothing significantly new. So, I expect that you should be able to go through these steps.

(Refer Slide Time: 27:52)



And those will be clear to you again we have a 2 phase recovery of redo phase and the undo phase and we make use of the operation undo, logical undo as and when it is possible and when that is and when we do that, we ignore all physical undo records and when it is not possible, then we lose the physical undo records and that is how the recovery can be achieved.

(Refer Slide Time: 28:19)



So, in this module we have expose ourselves with the Recovery Algorithms now for concurrent transactions as well and we have shown that how recovery can be done using operational logging, operations logging and making sure that really the database may not need to hold on to a lock for a long time on the data item and delay other transactions, but if it can define the undo operation on the on the data on the data item, then it can release that log early and use that logging mechanism operation logging mechanism to recover the data.