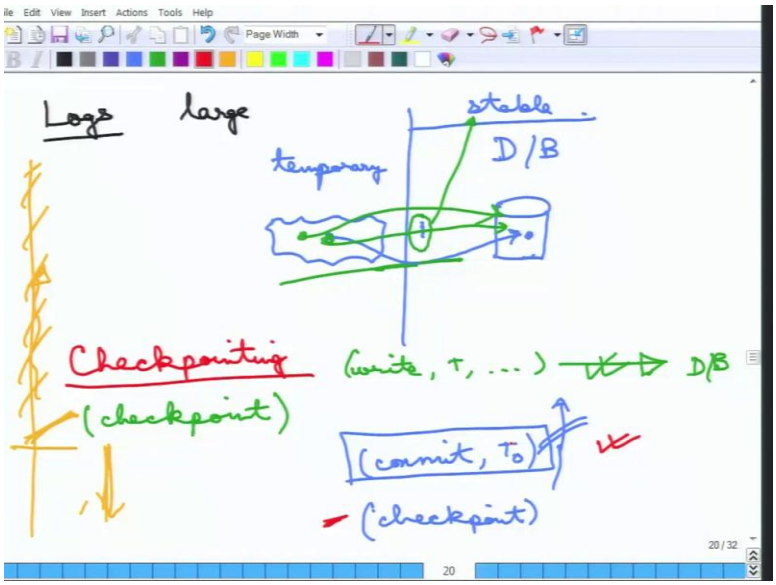
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 32
Recovery Systems: Check pointing and Shadow Paging

Now, one problem that happens is that.

(Refer Slide Time: 00:11)



So, the logs can becomes very large, so there is very large logs, because every all the transaction that we written in logs etcetera, etcetera, etcetera and suddenly one transaction has finishes. Now, it does not make sense to go over all the logs for some very old transactions, but to maintain the correctness there is to be way of saying that if the log is written for a particular transaction, all the writes in the transaction have actually gone to the database, the database has actually are seeing the effect of this, so why is it.

Now, just to highlight my point, what is being done is that the following, so there is the database which has contained the actual values of this data items and then there is a temporary copy of the database. So, all the writes are temporarily changed in this, there is a temporary copy and the log records only the log, it has an entry, the log it only saved if there is change here. Now, that does not mean that this is actually propagated to the

database and does been done and this the stable storage. So, this part is the stable storage, so this will never be lost.

So, the question is if there is many, many, many, many, log entries then there are many, many, many, many changes in the temporary thing, but it is not sure which temporary things that gone to the database, that is the well idea. So, the logs when go very, very long and searching in the logs and finding on which transactions have started, but not committed which has all those things become very much time consuming. Now, to prevent all of that, an idea is used which is called a check pointing; I am sure many of you have heared this idea of check pointing.

Check pointing essentially is trying to make sure that whenever a check point is being taken on the log, everything after the check point is correct. Now, correct meaning the following thing is that, so there is a very large log record and a check point is taken. Now, this means that everything that is on the log is being put to this stable storage. So, the log after that point is being put to the stable storage, more importantly all the pending writes up to the check point, all the pending writes are also fleshed to the database.

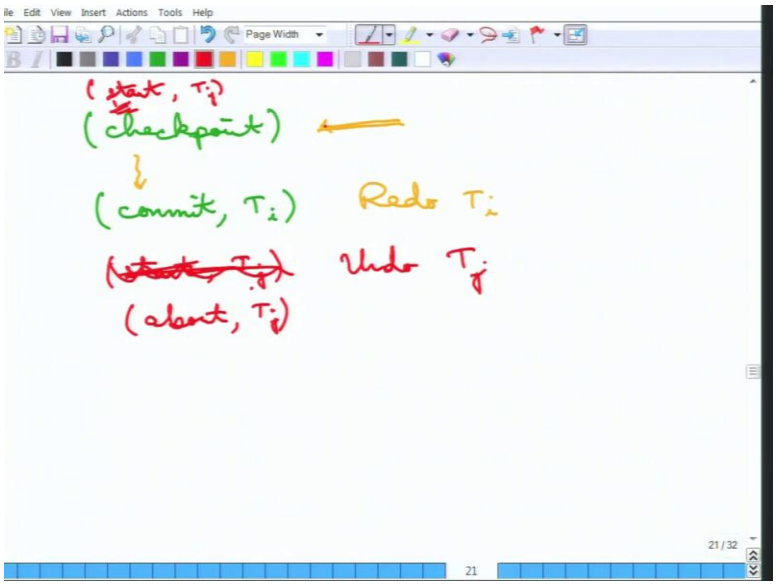
So; that means, if I say there is an entry which says check point in the log, so if check point is being done, so if there is the write entry for some transactions before the check point, this effect as actually go on to the database, this is make sure. How it is been make sure? It is being actually forced to the database, so before the check point is being done, it is actually force in the database, the log etcetera. The log is also maintain in the stable storage, the log record also goes, the log is also return on the stable storage and all the write records in the log has gone to the database, it is make sure before the check point entry is being made.

Now, here is the point, why is it that a check point entry will now help is that, now if the system crashes all that the log needs to do is to find the last check point, it is sure that everything below before that is correct. So, it needs to worry only after of things after the check point. So, even in the log is very large, everything up to this larger part of this thing, where the check point has been taken is correct, so it does not needs to bother about it, so that is the whole point and just to take it a little bit further.

So, suppose there is the transaction commit T 0 after which there is a check point is being return. So; that means,... So, that fact that there is check point meaning that all the

write operations for this below this has been correctly return that means all the operations of T 0 have been correctly return. So, T 0 does not need to be bothered about any more, this is all done just, because there is a check point entry. So, T 0 is nothing is to be done, so no redoing needs to be done for T 0 that is fine.

(Refer Slide Time: 04:19)



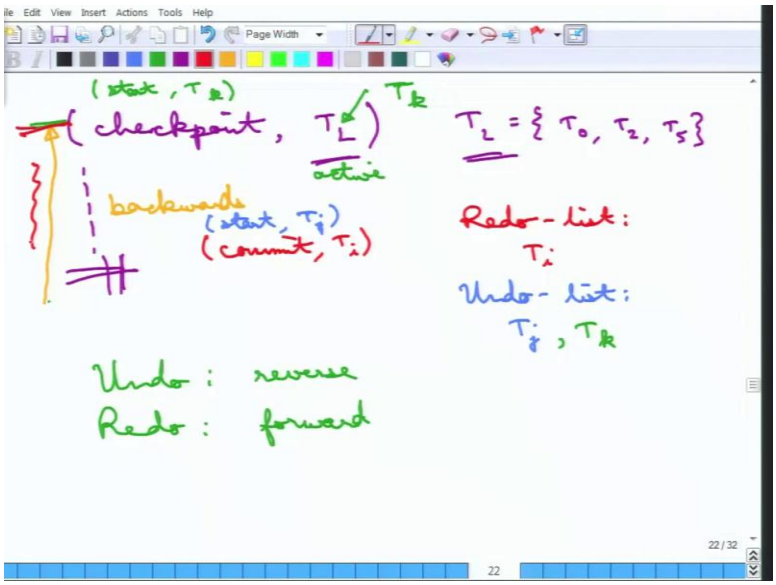
But, there may be other problem in the sense that something there may be a check point entry and then after some point in time, there is a commit to some transaction say T i. So, now, what it is being guaranteed is that every write up to check point is being done. But, there may be write entries after the check point and before the commit event has been done. So, T i is not guaranteed that all the entries, all the writes of T i is being done correctly. So, T i must be redone this cannot be avoided, so T i must be redone that is the thing.

And then, anything suppose there is another thing where there is a start of T j and of course, there is no, the start of T j may be here, the start of T j may be here. And, but there is no commit entry of T j that must be undone. Because, again if the check point only guarantees that everything has been retained correctly, but. So, there may be some write operation before the check point and then the T j may about here.

So, this instead of this thing this may be aborted here or it may not even write abort. So, T j must be undone, because the check point only guaranties that the writes are gone to there. So, it must be undone, so that is the thing, so check points... So, any transaction

just to summarize any transaction that has committed before the check point entries being made, does not need to be bothered about it. Because, all it is writes of gone to correct, anything that has committed after the check points needs to be redone and any other transaction that has started, but does not no committed entry needs to be undone; even if it is before the check point that is the whole idea.

(Refer Slide Time: 06:06)



Now, just to make it even more efficient, this check point can be augmented by the list of transaction that are active at that point. So, instead of just writing check point, so it can also say there is a list of entries, so T L is the list of entries. So, T L essentially is a list, so it can say T 0 is active, T 2 is active, T 5 is active and so on and so forth. So, this is the list of the entries, so this list is written as part of the check point entry. Now, after that there are some more log entries etcetera and suppose there is a crash here.

So, once the crash happens the log is read backwards up to the check point and the... So, this is read backward this is very importantly, this is read backwards up to the check point and the following thing is chunk, if there is a commit T i is found in this backward scan then there is a redo list, where T i is added to the redo list. Now, on this scanning backward if suppose on the other hand somebody founds the start of T j then there is an undo list, where T j is added.

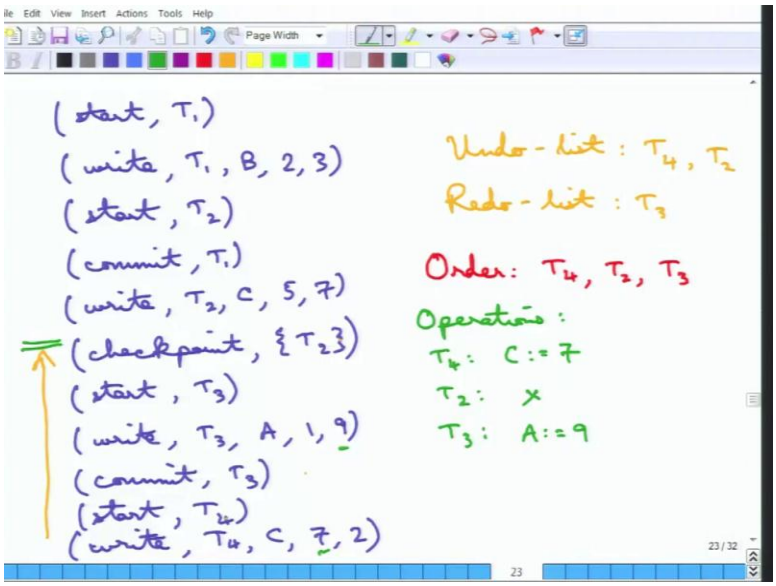
And suppose there is some other T k which is started earlier have, now known that scanning the log goes only up to the check point. However, what will happen is that

since T k started before the check point and does not committed, T k must appear somewhere in this T l. So, this is must be the active in the list of active transactions, so T k must be appearing somewhere in that. So, T k is not found using this scanning, but T k is in the active list, then T k is also added to the undoing list.

Of course, if T k has not been in the redo list; that means, T k there is no commit T k entry of this, if there is commit T k entry then T k would have been added to the redo list then nothing needs to be done. But, otherwise T k needs to be added to the undo list, so this is the way how the redo list and the undo list have been made and then we follow the same thing. So, undo's are fast done in a reverse order and then the redo's are being done in a forward order.

So, the transactions in the undo list has first revert it back in the reverse order has the appear in the log and the transactions in the redo list as then done in the forward order as they appear in the log and one more thing only the operations after the check point. So, this is rate only up to the check point. So, only the operations after the check point needs to be either undone or redone that is it, because before that everything has been either undone or redone correctly.

(Refer Slide Time: 09:14)



And here is an example to see, suppose there is a start T 1 then there is write T 1, B, 2, 3 there is a starting of T 2 and then T 1 commits then there is a write by transaction T 2 to another named item C the value 7 then there is a check point. Now, this check point will

essentially say the list of active transaction which is only T 2 at this point. So, the check pointing the heavy operation, the check pointing is done lots of things needs to be checked.

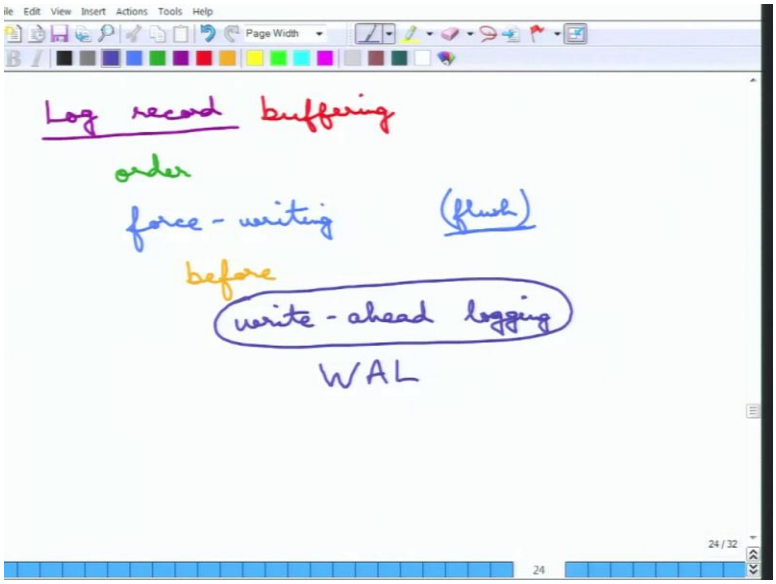
So, start T 3 then there is a write of T 3 for value A, 1, 9 then there is a commit of T 3 then there is a start of T 4 and finally, there a write of T 4 for C, 7, 2. So, these are entire list of log record that will see, so what happens is that. So, the first thing to find out is that, so then there is a crash at this point. So, now if the first thing to find out is what is the list of undo and redo list, so the undo list is the following.

So, undo list and the redo list, so the undo list is, it is scan backward there is a start before that is being noted. So, that is added to the undo list correct and then it goes to the check point and sees that there is an active transaction T 2, so that is also added to the undo list, because that is not in the redo list. So, the redo list if it commits and it finds commit T 3, so the redo list is at the list is only T 3.

Now, note that T 1 is not appearing at all, because T 1 committed before the check point that is it. So, then the order of operations will be the undo list first and then the redo list. So, there are two things in the undo list how will be done it is based on the when it started etcetera. So, T 4 is first undone then T 2 is undone and finally, T 3 is undone. So, this is the order in which this things will be done and the order of operations will be the following.

So, for T 4 the operation if the C is return back to the old value which is 7, it is return back to the old value and there is a only operations for T 4 for T 2 nothing appears yes. So, no operations is being done, even though it has return something back, because you see the point it goes only up to the check point and does the redoing and undoing of only those operations that appear up to the check point and for T 3 the redoing of this thing needs to be done. So, A is return the new value of 9, so that is the way this whole scheme which check pointing takes place, we will have too more small things to covered.

(Refer Slide Time: 12:49)



So, first thing is a very small issue of log records, so the logs are also retain to the stable storage etcetera. And so every time are database draw the write operation, read operations something is being return to the log. Now, the question is the log is maintain as a temporary list in the main memory and the log needs to be return to the database or not to the database, the log needs to be return back to the some stable storage.

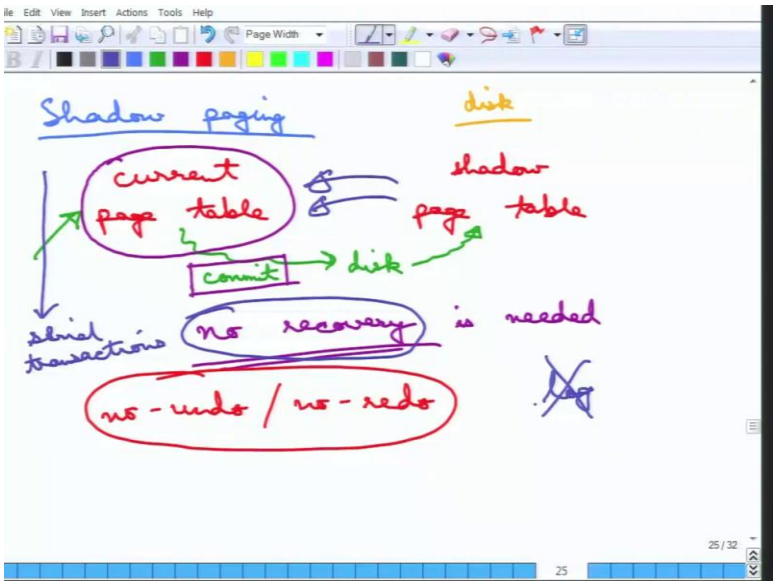
So, how many times will the log be return, so there is the concept of log record buffering, which means that not every time a log record is being done, the log is return back to the stable storage. A couple of records the buffered together and then this is the output to the stable storage. The reason is the same reason that we saw all those why the inter disk page is access together. So, all the log records that fit into one disk block is first filled up and then the disk block is return back to the disk on the stable storage that is the way.

And the records are of course, first in the order in which they appear in the log. So, this is in the order of list appear in the log and then there is concept of force writing is used, force writing meaning everything that is return that needs to be return is forcefully return back to the disk. So, this is in the if you know this programming languages see etcetera this called as flash operations. So, flash means that it is surely gone to the stable storage, so that is the flash.

So, this force writing of the log record is being used and even, so if there are some commit entry etcetera that is also flashed and all the log records. So, if there is the block of data that is return to the data base all the log records ((Refer Time: 14:36)) it must be done before. So, this log must be flashed before the actual, the writes are go on true, so that is the thing. So, the log records retain before the actual thing that is why this is sometimes called are write ahead logging.

That means, before the logging before the actual write is done, the log is being retain ahead. So, this is the write ahead logging or this is sometimes known as the WAL rule, W A L, because it is the write ahead logging, so that is all the thing about this log.

(Refer Slide Time: 15:23)



Then there is one more type of data base recovery scheme that can be followed, which is called the shadow paging and although we talk about it very briefly last time, a little bit more details on this is as follows. So, the shadow paging is essentially there are two copies of the data base are maintained. Now, we will assume that there are two page tables in the OS type. So, essentially forgetting about page tables the details of it there are two copies of the data base are maintained and the database is broken up into disk blocks or pages.

So, these pages the list of this pages are maintained and then there are two copies, the first one is the current page table, page mean that disk block and the other one is the shadow page table. So, there are the two copies are from maintained, the shadow page

table is maintained on the disk, this is maintain on the disk and no changes at done to it. So, no changes are done on the shadow page table, so whenever something needs to be changed the corresponding page in the current page table is what is being changed.

Now, so all the update etcetera goes to the current page table, now whenever the transaction commits all the changes that have been made to the current page table, all those things are flash to the disk. So, this is all flash to the disk, so what does flashing to the disk means, the flashing the disk means these are return to the corresponding pages in the shadow page table, which is on the disk. So, that is being flash to the shadow pages, the shadow page table actually modified only when there is the commit in a transaction commits, so that is the commit operation that is been done.

And if this is followed then actually what can be shown is that there is no recovery is needed, if that is the simple scheme no recovery is needed. What it means to say no recovery is needed? Because, you see what happens is when a transaction fails what happens all the copies or all these things are in the current page table which is will be lost that is it there is no, the shadow page table context all the efforts of only the committed transactions.

So; that means, nothing needs to be changed for the shadow page table and only the current page table is what is be changed. So, that is why this is also called a no undo slash no redo scheme, because nothing needs to be undone, nothing needs to be redone. However, at the end of all of these things the shadow page table is the new correct things. So, that is the thing, so this what is the this seems to be very useful, because no recovery needed, but there are some practical problems that happens with this first of all there are too many pages that needs to be copied.

So, the efficient is not much the commit overwrite also may be to high, because lots of things may be needed to the shadow page table, whenever a transaction commits and the things is serial if really only happens for serial transactions. So, the transaction must come one after another, if the transactions have been inter list then this scheme may not fault. So, this only what is the serial transactions, the advantages of course, is that no recovery is needed.

And so there are no logs, no override of writing the logs, etcetera, etcetera, etcetera, so logs are not readied. So, there is no override of writing the log, so that aims the topic on

data base recovery management systems and we studied some couple of scheme when using logs and which check pointing etcetera and finally, the shadow paging, we will next covered the important issue of schedules.