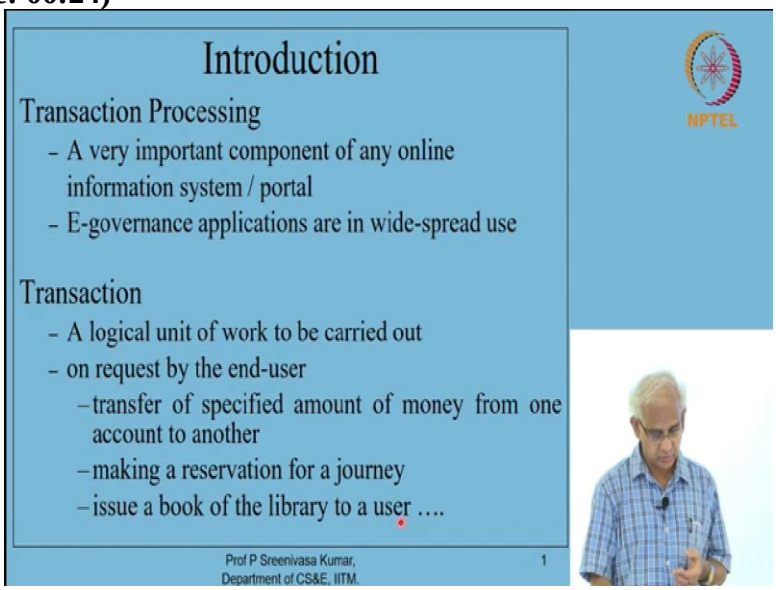
Database Systems
Prof. Dr. Sreenivasa Kumar
Department of Computer Science and Engineering
Indian Institute of Technology Madras

Lecture-36
ACID Properties and Operations in Transactions

(Refer Slide Time: 00:24)



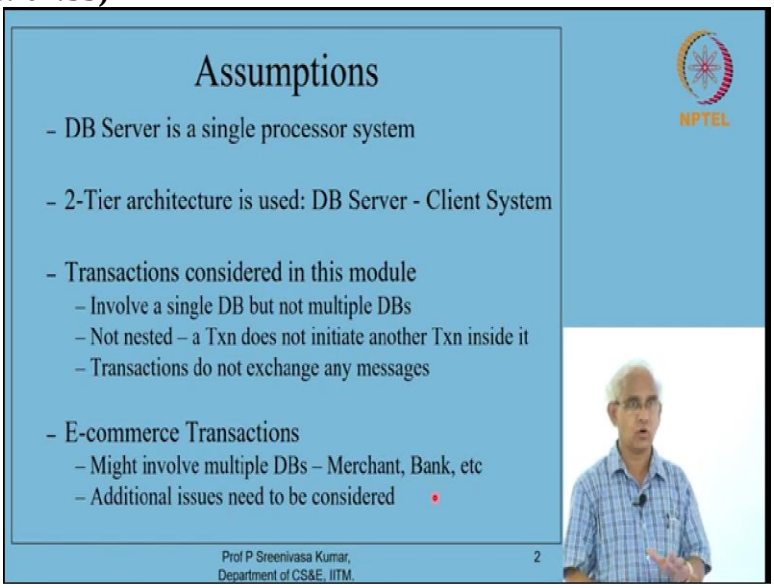
I am going to start off the transaction processing and concurrency control module, it happens to be the last module of this course. So, in this module, basically, I will be exposing you to some of the fundamental, important ideas in the transaction processing, sub field database systems. This is again a very, you know, as subfield of database systems. So we will only be touching some important and main, fundamental kind of ideas and there is really a lot more for you to study in this field then what I can actually be able to do in a introductory course like this.

So, transaction processing is a very important component of database systems. So, you can now see that many of our day to day you know functions are actually controlled by some transaction servers sitting somewhere you swipe your credit card and then response has to come in a few seconds only then you know your transaction to purchases it to otherwise a standing there. So, many of our activities are now controlled by transaction processing systems are very, important.

So, what exactly is a transaction is what we are trying to understand. And then we will see what are the issues in realizing these transaction processing so at this I will be giving more and more detailed definitions for this various things that are involved so at this stage, we can say that transaction is a logical unit of work to be carried out some piece of work just on the request by the end user of the system. So examples, of course, are many, you can say transfer of some specified amount of money from 1 account to another.

Making reservation for a journey issuing a book in the library for a particular user all of these things are logical pieces of work that are asked for by the end users, like the people at the counters and on or people operating elaborate enterprises now.

(Refer Slide Time: 02:53)



So, a few assumptions that we make in this discussion is that the database server is a single process system. Single processor system because we were talking about be the process waiting and then you know, transaction multiple transactions getting interleaved, and all that so you could have though so we should make it clear as to how many processes with that so we make an assumption that we have a single processor system.

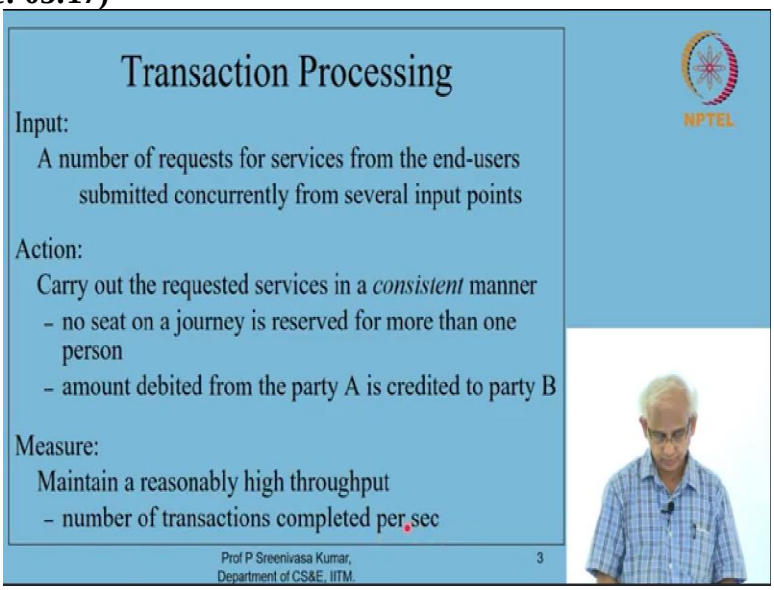
If you understand these issues, then we will be able to understand the issues later and will also assume that there is a 2 time architecture a database system and a client systems database server and client system. From there we will get requests for the processing of transactions. Now they,

the transactions that we are going to consider in this particular module will basically involve a single database, single database, but not multiple databases.

Another thing is a transaction does not contain another transaction inside it will make the sudden; there is no nesting of transactions. And transactions by themselves do not exchange any messages between them. They do not exchange any messages. They are all independent. Notice that in real life, one of our transactions might actually touch multiple databases. If you actually do a ecommerce transaction nowadays.

It might involve multiple databases, your merchant database, where you are thinking of the product, and then the payment gateway and your they actual a card system or a bank. They are all independent systems and they are all independent systems. They maintain their own derivative. And so a real attraction actually such as many so there are additional issues to be considered in kind in that kind of application now we will restrict our scope to try and understand how transaction processing happens on a single processor which is touching on the transaction that in just 1 database.

(Refer Slide Time: 05:17)



So, for the transaction processing system, there are a number of concurrent requests for these services. So, what are these services, these are application programs that have been developed by application programmers and are made available so that they can be invoked as part of these

transactions. So when you develop a for example, a library system, you will also develop as to what is a program that can that can issue a book or an item for it end user.

So it takes in the credentials of that end user you know kind of verifies at that particular person, you can be eligible to take a book and then modify the database to reflect the fact that this book has been issued to that particular person and then, you know price, back all these details to be transaction and then finally allows the person at the counter to actually give the book to you first. So, these are all various. So these are actually high level programs that are developed by application programmers.

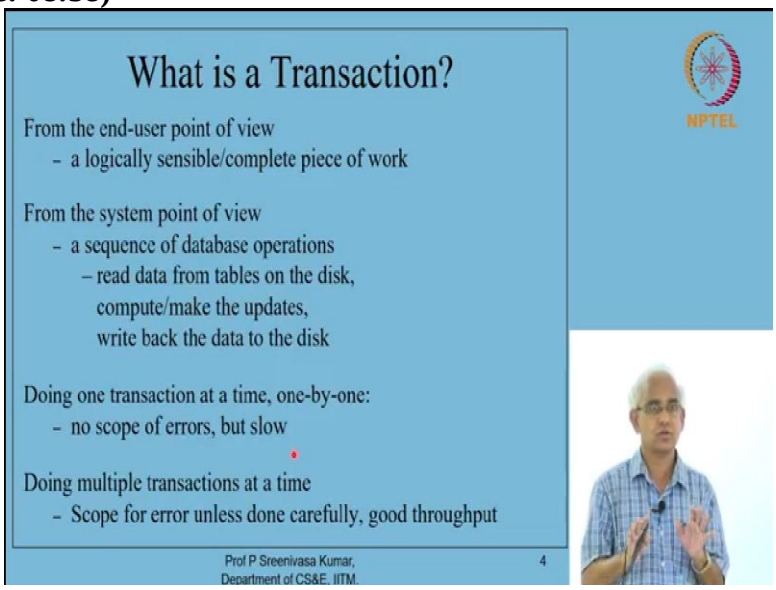
And so, you know, some of these programs are invoke concurrently by many people, so, it is a program. So, each invocation of that is a process invocation of that the process. So, this process gets initiated by multiple almost some sometimes simultaneously, then many people at you know like concurrently like, for example, you can imagine the railway reservation system. So there at any point of time, there are probably 100s of a few 1000 people trying to make the reservation.

So they are all invoking 1 particular kind of program which is to conduct it is this reservation and then ticket issue program. So, a number of requests for these services are submitted by the end users concurrently from several input points, several people will be operating from their various devices and the central system that has to carry out these requests in a consistent manner. So what is this consistency that we are talking about will depend on the actual system that we are operating?

So if we are actually operating a reservation system, no seat on a journey is supposed to be reserved for more than 1 person. Different dissenting requirement and the amount debited from party A is to be credited for party B if there is a transfer from A to B fast and then what is the measure is to maintain a reasonably high throughput is the number of transaction invocations that are completed in unit time by the server.

How fast it is this will decide how what is your response time or when you as an end user invoke some of these things will be made waiting for a long time what is the number of units number of transaction locations that are completed per unit time. So, try and maintain a high throughput.

(Refer Slide Time: 08:58)



Now let me read come back to this question. From an end user point of view, the transaction is just nothing but a logically sensible, complete piece of work. Now for progress for us the database system people, what is it exactly? Is it from the system point of view, it is actually a sequence of database operations, the sequence of a lot of database operations, read data from tables on the disk, compute make updates.

Prepare updates maybe you will have to read data from multiple tables, make the updates, write them all back to the disk all that so there are a lot of operations to be done. No 1 approach that I can adopt while handling transactions is to do 1 transaction at a time 1 invocation come, I will try I will do that completed and then take up another, but the cancel as we have been seen in the last module the disk operations that are involved or you know very slow compared to the memory of so this program is known then kind of halting.

Because it is waiting for the block to come to me and memory and so that it can do some changes and then continue doing something more and etc. So, if you do 1 invocation at a time, then you throughput is definitely going to be very, low because each of these operations involved in each

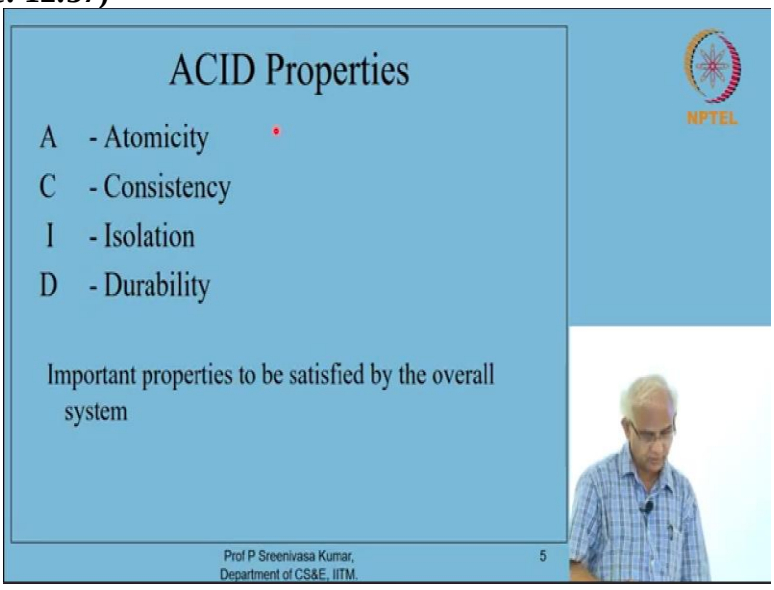
of these transactions involve multiple disk operations and the disk operations are an order of magnitude slower compared to the big number.

So, but then so what is you so we do have to take multiple transactions probably even the same invocate in with multiple invocations of the same transaction and then actually do them all concurrently. This like an operating system does several processes concurrently with transaction the management system also has to do several processes concurrently in order to increase. So, there comes the scope for possible errors unless we do it carefully.

Unless we do so, operations of various transactions are going to be equally there is a single processor system. So, while it is doing some part of the work for some transaction, it has asked for a discrete and so, you do you temporarily suspend the process and then start taking another thing and then continue doing that and then something else this follow is asked for a discrete. So, you have that fellow out another transaction it is all going on.

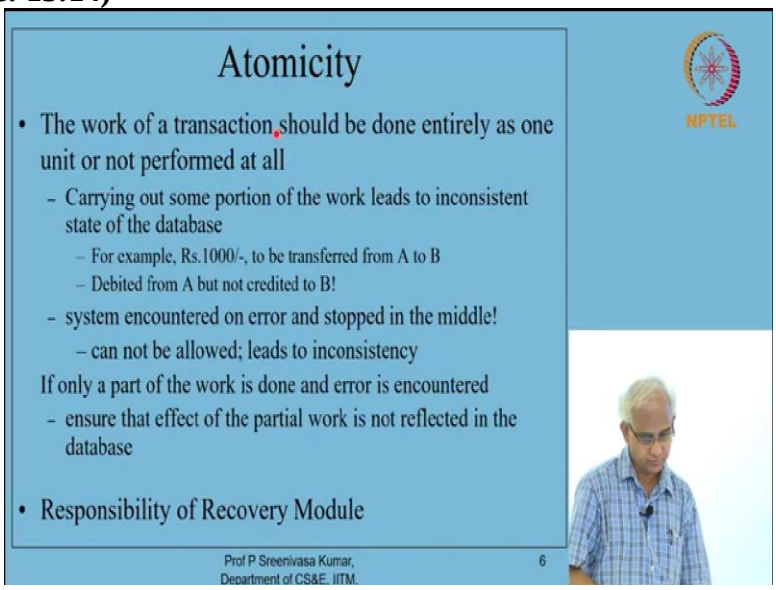
But then you know each of these things are actually doing lots of disk reads and then if they are, if 2 of these if 2 of these connections are actually trying to do some updates on the same database item and things like that, then we are actually a little bit of a trouble unless we control as to how they do. So, it has to be done carefully. This is what we will examine in this particular module.

(Refer Slide Time: 12:37)



So in this context, in the context of a transaction processing system, we typically talk about these 4 important properties. These are called acid, acid atomicity, consistency, isolation, durability; these are 4 important properties that we talked about. And they are somehow they are always called ACID properties. So let us look at the ACID properties.

(Refer Slide Time: 13:14)



So what atomicity basically says that the work of a particular transaction should be done, it is in its entirety. The work of a transaction should be done in its entirety or we should create the transaction as though it is atomic to either do this and entire work or do not do anything at all. Even though the request system do not perform anything or perform the whole thing either so you can see that why is this very important? It is very important because carrying out some portion of the work obviously leads to inconsistent state of the database.

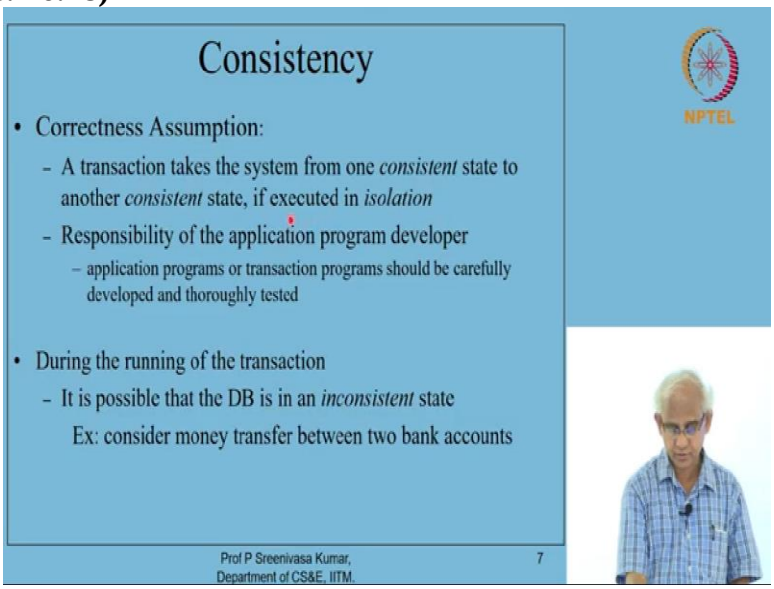
For example, if you are transferring some 1000 rupees from upon A to B and you reduce it from A account and actually not credited to B account, and then you know you finish the program has finished for some reason, then it databases you know is that the inconsistency it is supposed to be credited to the B's account. So, part of the work if it is done and if you stopped due to some encountered some kind of error, then we have then atomicity is not guaranteed.

So, if such a situation comes, because a situation comes up we have to stop, then you must keep this commitment of atomicity in mind and then you know, ensure that the effect of the partial work that you have done is actually not reflected in the database at all. So, you must ensure that

even though you have proceeded further in this transaction, but stop somewhere in the middle, then you should see that whatever changes that you have done or not actually permanent to there in the database.

When you so, this particular atomicity is a very important property and actually we will see that the recovery module of the transaction processing system of Recovery Manager module of the system will actually handle this. So, why would such a thing happen? It is possible that there is a system error and it processes have been suspended and you have to restart the system or there is a power outage or something like that happened stop. So, how exactly this is done, we will see as we proceed.

(Refer Slide Time: 16:18)



Now here is the other property called the consistency of? So, this is both to do as a assumption about the correctness of individual transactions programs. So the assumption here is that an application program which represents this transaction takes the system the database system, the database system on the system database from 1 consistent state and starts from a consistent state and then takes it in a consistent state that is the assumption that we make provided that the transaction is executed in its entirety.

And is also executed in isolation as though it is running all by itself and it completely runs. If that happens, then the transits we make an assumption that the transaction actually takes the system the database from 1 consistent state to another consistency. This is a fair assumption to

make and whose responsibilities is this it is obviously the application programmers responsible when you write a program to do some transfer of amount A from account A to account B.

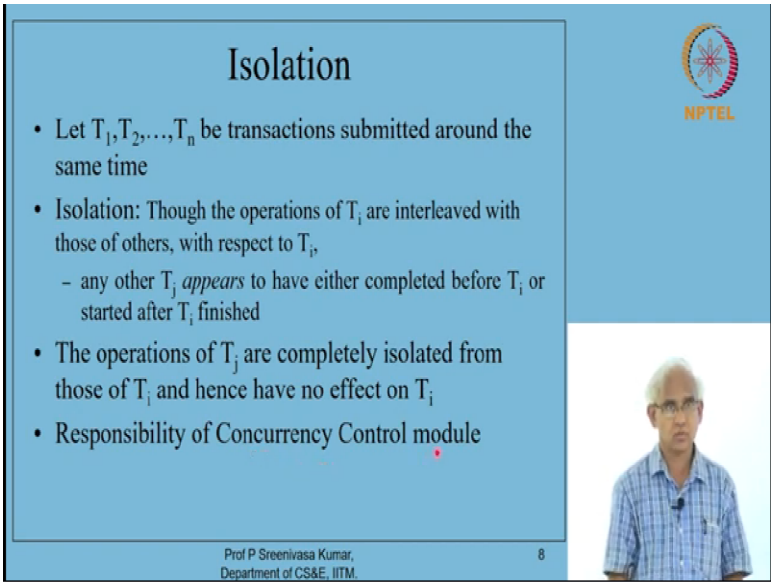
So, you are implementing the transfer functionality, then it is your response to actually if you are guaranteed that your transaction is going to be run in its entirety, it is your responsibility to ensure that you have done the deduction on the addition. And you are also given me assurance that you are done, you are isolated you are running you know all by yourself, no, there is no disturbance or some other transaction does not run in.

So, this is the responsibility of the application. So, we kind of transaction processing systems assume that the transaction programs have been developed carefully, and then thoroughly tested. So that we can make this assumption about the transaction program but that if we have a consistent database, the running of this particular program will take the database and the consistent state, of course, in between, it might in some sense, go through a little bit temporary inconsistent state because for example, you are transferring it.

So, you are at some point of time directing 1000 rupees from base account. And after some time only we are good actually are so in between if you watch it this is a little bit, but then we are not bothered about that we are bothered about because our atomicity you know, assumption says that we are going to run each of these transactions in their entirety all do not own them at all. So we assume that if it is run executed in isolation and completely.

It will take the data, get the database from 1 consistent, consistent state is a very pair of them to make and it is the responsibility of the application programmers and that is what actually the role of the application programmers is also very critical in the entire system, you have to ensure that these programs are thoroughly tested and checked off.

(Refer Slide Time: 20:09)

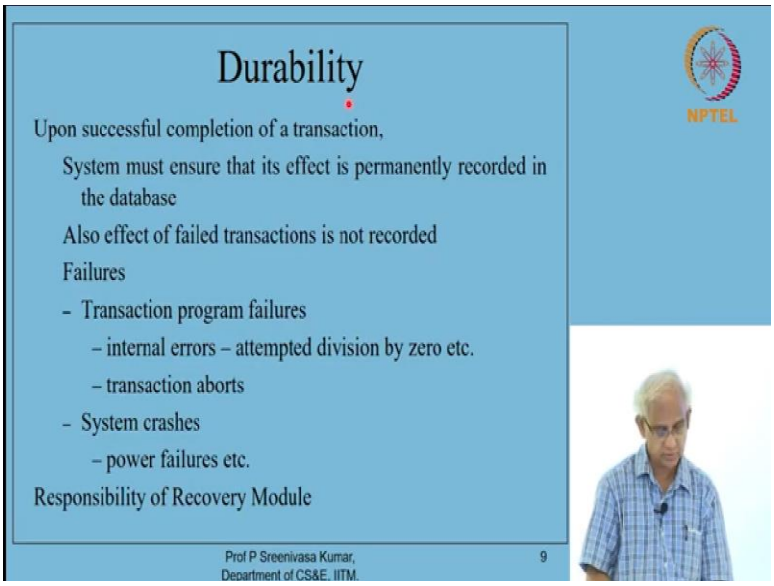


So, here comes another very important property called isolation. This is also very, you know very much required. So let us say there are some especially in the context of this system having to run many processes concurrently. So, in that context so let us say there are some n number of transactions submitted around the same time to the system. So what the isolation property asks for is that though the operations of this particular.

Any particular transaction T i are actually going to be interleaved for, the performance say for the sake of you know, getting better performance are going to be interleaved with those of others. So with respect to this transaction, any other transaction T j any other transaction T j appears to have either completely done before T i appears to have here completed before T i are going to start after this that means essentially you are giving you are able to get an impression that T i will run you know as 1 unit without interference from other.

So the operations of so with respect to T i every other operation every other transaction T j appears to have completed before T i are started after T i finish. So, in some sense what we are saying here Is that the operations of T j are in some sense completely isolated from those of the T i and hence will not have any effect on the T i. So, who is this is actually the responsibility of the concurrency control module is where there is going to be a concurrency control module in the entire subsystem. So that is going to take care of.

(Refer Slide Time: 22:25)

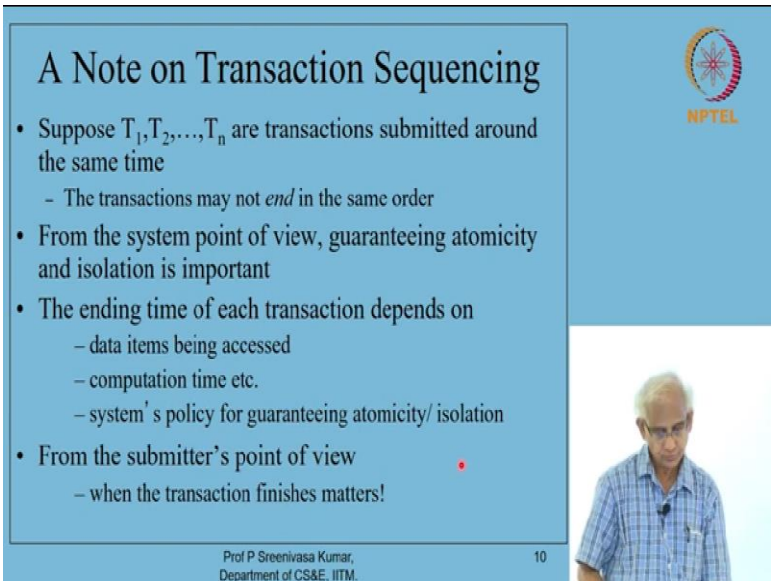


Finally, we have what is called durable acid properties so durable. What durability says is that upon successful completion of a transaction, the system must ensure that the effect of this particular transaction is permanently recorded in the database. Durable permanently also, if there are any failed transactions, the effect of those failed transactions should not be in the database for both these things are very important.

So, durability basically says this. So, where do failures come? Of course failures come because transaction programs might by themselves fail because of some internal errors. For example, attempted division by 0, then you have to stop the program. So, that we also will discuss what about transaction of ours above and system might crash due to various reasons, fluctuation in power failure in redundant systems my systems might crash.

So, in this in the face of all these failures it is the system's responsibility to say that if this transaction is successfully completed, then the effect of the transaction is formally recorded in the database and this is the responsibility of the recovery management module recording. So how exactly does the recording module happened to do this? We will see, all that details, how can it do this.

(Refer Slide Time: 24:26)

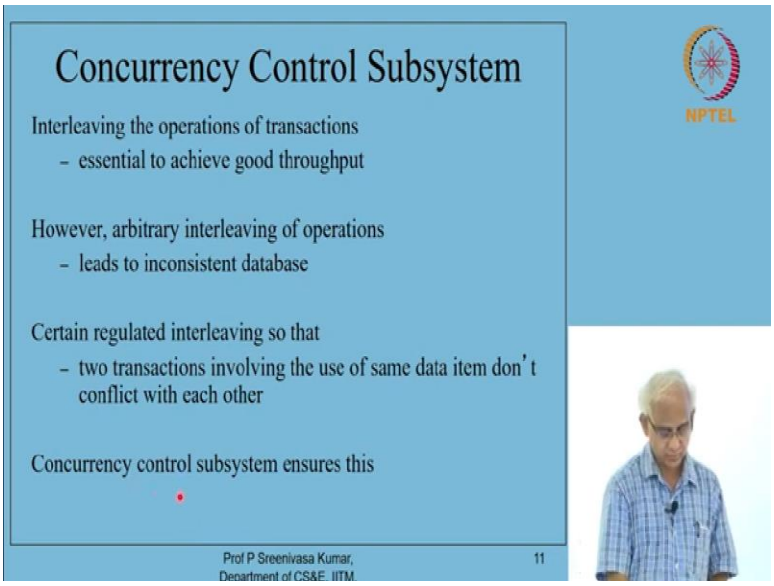


So while we are discussing these issues here is a small again, note about this transaction sequencing. Suppose some n numbers of these transactions are submitted around the same time or they were actually end in the same order because each of these transaction is actually doing different kind of work. So if something one of these and that you may have to wait longer for some discrete or you know, for the computation it is doing and things like that. So we do not know when they will end they may not end in the same.

So actually from the system, our database management system point of view, the guaranteeing atomicity and isolation are the most important things for us. Whereas from an end user point of view from the end user point of view, submitter's point of view when the transaction finishes, of course matters because that is that might decide whether he is getting the seat or not, you know, he is getting not.

So you have to keep all these things in mind. So the end time is actually not we cannot give guarantees about the end time, but what we can do guarantees as a system is atomicity and isolation. And what is so the ending time actually depends on all these factors saying what is the policy for guaranteeing.

(Refer Slide Time: 26:25)



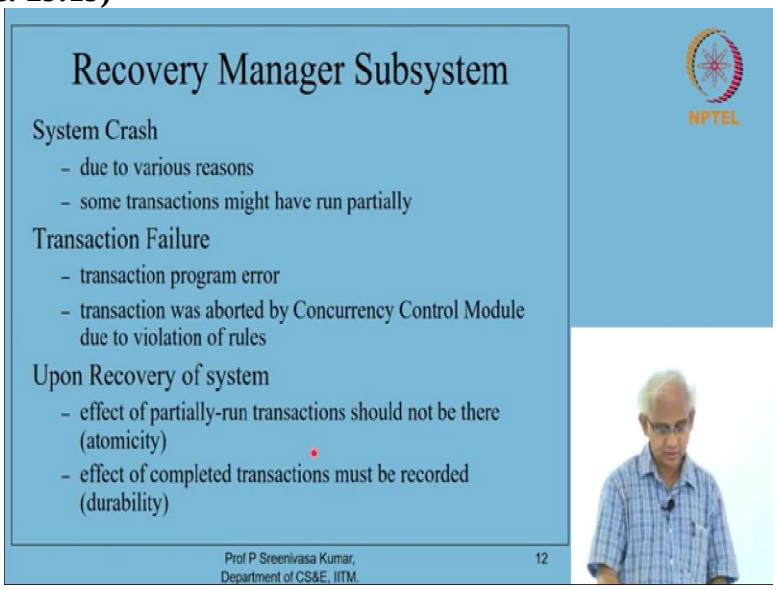
Now, let us come to the concurrency control subsystem. Interleaving of operations of these transaction processes is very important. Otherwise, we will lose out on the performance the throughput will be very low. However, we cannot do arbitrary interleaving because it might lead to some inconsistent state of the database. So, what is it that we have to do? So we have to do some kind of regulation. In this installation, we cannot allow all possible interleavings of process operations to be performed on the system.

Especially when to transactions you know involving the use of the same data item or trying to run then they are going to conflict with each other. And so, the concurrency control subsystem has to detect this kind of situations, and then actually ensure that such kind of thing does not happen. It is kind of complex. So, it requires us to kind of have a very detailed model of how these transactions get interleave. And then steady these we will later introduce what is called it is scheduled for transaction operations.

That is a particular interleaving and then we are to study this properties of those interleavings and then find out what are the best interleavings that can be allowed and what are other should not be allow it. So, they come there are multiple ways of achieving this purpose of concurrency control controlling the concurrency such that desired results come in the transaction system. We will in this sequence of lectures I will discuss a lock based concurrency control mechanical there

are what are also called timestamp based concurrency control mechanisms. But we will try to we will focus on locking base concurrency control.

(Refer Slide Time: 29:19)



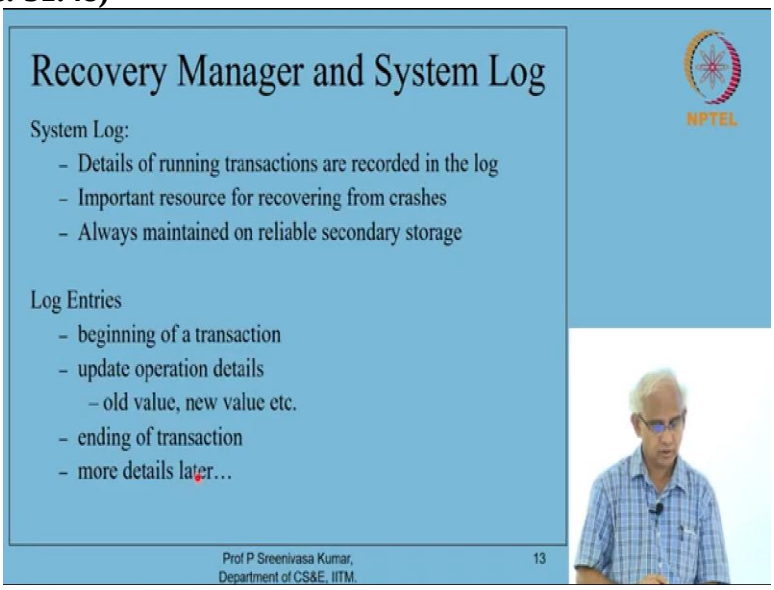
Now, again coming back to the recording managers subsystem, system crashes cannot be avoided. So, some transactions might have done it partially in a partial manner. So because of that they would have also made some changes to the database and transaction pages also can may occur for example, errors in the transaction and you are trying to transfer amount for example, A account does not have sufficient money inside.

And sometimes the transaction might actually be aborted by the system itself. The concurrency control module that we will discuss, we are going to see might actually decide that this particular transaction is causing trouble for us, because it is violating certain rules, it might deciding any time we will do abort this runner and kind of restart it. So in the face of all these things, the Recovery Manager has to ensure durable Recovery Manager Accenture both durability and performance.

So how does it do? Is the question so, what exactly needs to effect of any partially run transactions should not be there system and the effect of completed transactions must be there. Now, the big trick in this entire thing is to actually have the control with you with the system as to when to say that you have finished a transaction you know, keeps doing some work. So, it will

ask for saying that, yes, I have finished you know but then you have to keep some control on when exactly you, kind of that we will see that as we go along.

(Refer Slide Time: 31:48)



So, the Recovery Manager what users is what is called a system log. System log in a very important tool for the Recovery Manager so all the some amount of details of the running transactions if you admit a transaction it is running, then you keep track of certain details of the running transaction in the log. And this log is always maintained in a reliable secondary storage. So that Recovery Manager can make use of this log in order to recover the system or in order to restart it under so these log entries, the log entries.

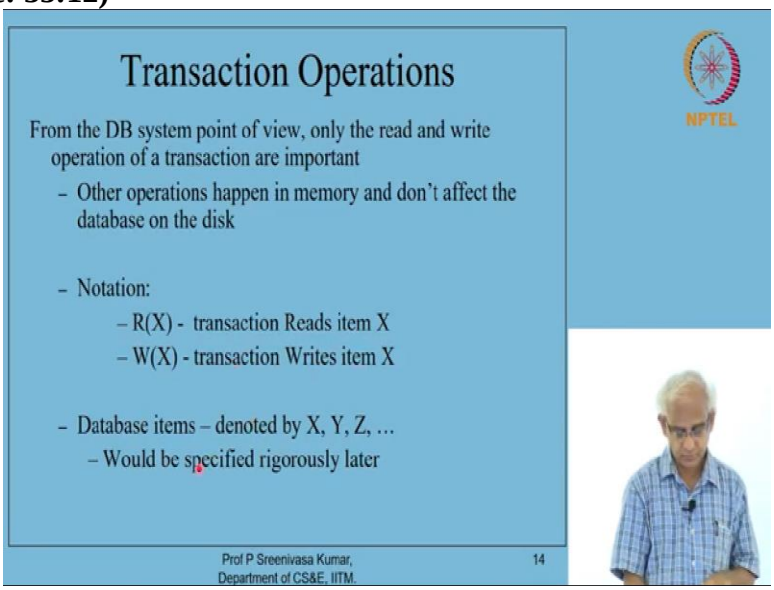
Now will have when the particular transaction started running the beginning of it and surprisingly, we have not really go after the process after that, in the sense that whatever it is doing especially update operations is doing, we have to keep track of. So, what are the updated operations is doing? What is the old value, what is the new value from the existing value is there and it is going to change that and the might new value, sometimes we will keep track of the old value and the new value, etc. and when is the transaction ending?

And so, we kind of keep very close track of how the transaction is process proceeding and what are the various things that is doing in a system log, so that at any point of time if the system crashes, we have sufficient information to kind of bring the database back to a consistent state

would be a state where you have admitted saying that you are certain transaction has finished then the effect of the transaction is permanently in the database.

And if you have not done that, then you do not give any commitment for the founder and such a transaction has not actually done any damage to the database itself it does not guarantee that such is the critical nature of this particular operation. So how exactly the recovery happens will depend on the kind of logs that we maintain and what is the protocol that we use for method will use for recovery. So, all this will discuss when we come to the towards the end of this month. So these are the various interesting issues regarding transaction processing.

(Refer Slide Time: 35:12)



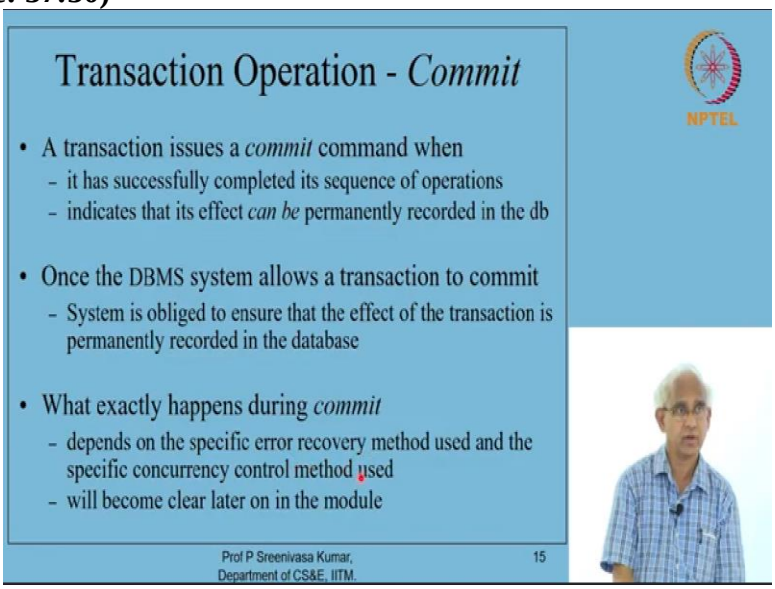
Now let us try to focus a little bit on what exactly are these operations? From a system point, DB system point of view from the programmer, you are actually going to use C or C++, Java and then write this, you know, detail C operations and then invoking SQL commands and all that.
But then those things are all going to be replaced by library calls that implement those queries, etc. Finally, it all boil down to a bunch of reads from the database updates and writing data. So those are the things that we will focus now on this way.

So from a DB system point of view, especially for the transaction processing server activity, we only need the read and write operations of a transaction on to the database. And all the other operations happen in memory. And so actually do not affect the database on the disk. So let us

introduce some notation for this. So R of X is a transaction reads some item X. I am going to actually make this a little bit more detail later on as we go through these lectures.

But for the moment now let us assume that the R of X is a transaction reads transaction reads item X, what exactly is an item? I mean it depends on you know, what we choose. So, it is a normal practice to choose that item as a disk block WX is transaction writes this item X so, all these database items we will specify them as X Y Z.

(Refer Slide Time: 37:30)



Now, so, read write among, in addition to the read write operations, the transaction also does what is called commit and above operations. So what is coming the transaction issues a commit command when it has successfully completed its sequence of operations and is ready to kind of indicate that it indicates that the effect can be permanently recorded in variables the effect can be permanently recorded in the db. So, this commit is a it is going to make a system call it is a kind of system.

So, by making the system invoking this commit come call, what the transaction, the individual transaction the process that is going to indicate saying that I have finished my work, I finished my work and all the changes that I have done, actually what some of the changes it has done might be in memory some of the changes it has done might be in than this it all depends because we are going to have a buffer you are this you are studied paging.

Paging mechanism in operating systems so, disk pages this blocks are going to be brought in to buffer memory buffer space in the buffer pages are going to be given to the processes and so, the processes will keep making changes. So, some of the changes might be in memory some of the changes might have gone to the disk all this is happening. Now, it is in this context the transaction we as a process it when it looks a commit.

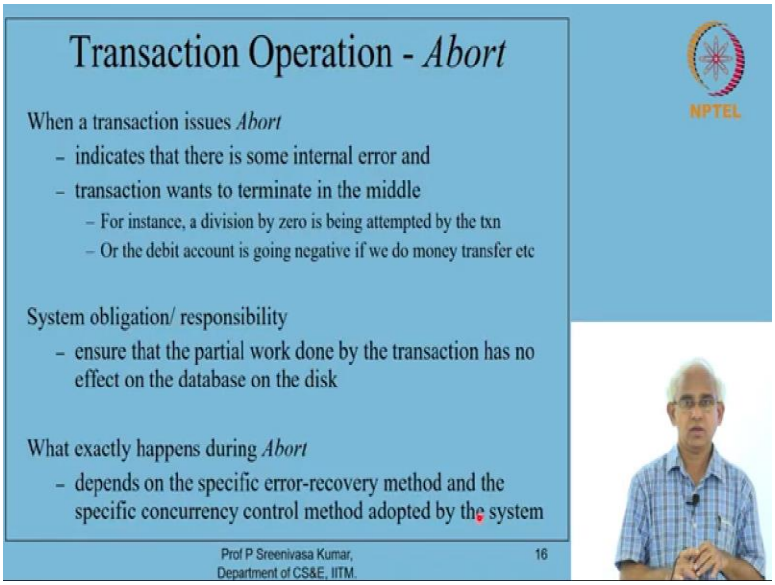
What it is saying is that I have successfully completed and I have completed the sequence of operations and whatever I have done can be permanently recorded in database now once the DBMS system you know you say S for this commit command. So, in the transaction individual process is asking for a commit it is invoking a commit operation. So the commit is a system call. So that comes to the RDBMS runtime system.

When the RDBMS runtime system says yes to this, yes, I agree, then the system is obligated, or obliged to ensure that the effect of the transaction is permanently recorded in the database. So what exactly happens during commit this will of course just before the commit you know of the transaction might you know this is might thing so it all depends so this ultimately it is going to be something called a it commits log records.

We will see that later on so the commit log record is what actually you know has to be returned to the log only then you know we will say that the actual in the transaction has indeed completed its law. So it actually write now it is a bit difficult to say this but we will see what are the exactly the error recovery methods specific error recovery methods that we have adopted based on that it will become clear.

Actually as to what happens in game so but it is important to remember that we committed a system also it comes to you the Alleghenies runtime system so you can take all the precautions that are necessary in order before you say yes because once you say yes you are coming. That is the interesting thing about they come so we will see exactly how commit gets implemented in various under different territory protocols.

(Refer Slide Time: 42:29)



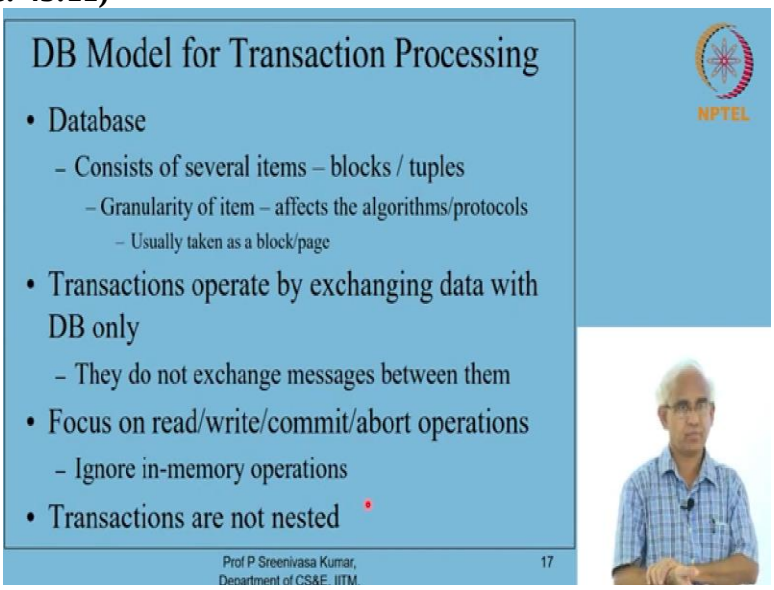
Now the other operation is the transaction abort is also and like this I did read write commit abort you are the high level operators that we can think abort basically indicates that there has been some internal error there have been some internal error and the transaction the process so as a programmer you do you know kind of been checking some things in between while running your production so you as a application programmer and art in under certain circumstances.

I did that this projection cannot proceed further and so you insane and I would not abort maybe you have encountered some you know attempted division by 0 or you know you trying to do some debate and they actually to carry it out they did it this one to become negative or something like that so you have deducted some condition between your application and then you have decided that and you as a programmer have decided that I cannot proceed with this so you each you like to issue and abort the quest.

When you issue an abort request then you are kind of indicating to the system saying that there has been some issue with me I mean there have been some issue with this particular transaction process so do not record anything that I have done you know in permanently database. So once it you say s for abort then invert it the system's obligation responsibility is that the partial work done by this particular transactions has no effect on it turn on the database along with this again.

What exactly happens during your bond depends on the specific error the query method use on the specific concurrency control my turn or doctor the so these are the various transaction high level operation done banking so the transactions are actual application programs but we are focusing from our point of view we will just focus on the and be done the issue of committing above.

(Refer Slide Time: 45:11)



We will from a transaction processing point of the kind of model the way up is that the database basically consists of several items usually it is taken as a block or a page is the granularity of the item is taken as a as block you could of course also take tuples or render team and it granularity that mean person will obviously affect all this algorithm protocol so we cannot really go into too much details here and the transactions operate by exchanging data with the DB alone.

They are not supposed to exchange information among them they cannot not exchange messages with me we make it this assumption and then we focus basically on page right commit award operations and of course ignore all things they are happening ultimately what comes to the (())(46:19) what is really leading it otherwise is not there to wench on surrender and transactions of course are not tested. So good I will continue this read more details about this in the next lecture we will focus on concurrency control and how do we model the concurrency and then how do we control that and then we will discuss generate.