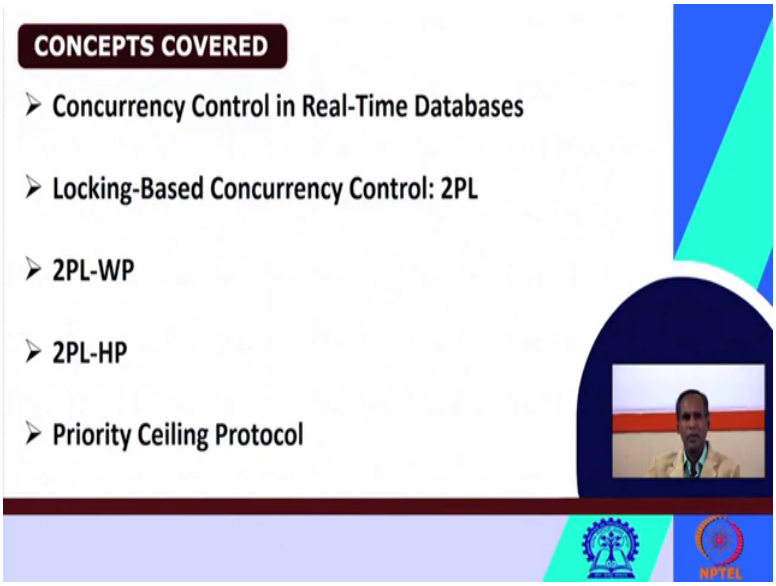
Real Time Systems
Professor Durga Prasad Mohapatra
Department of Computer Science and Engineering
National Institute of Technology, Rourkela
Lecture 60
Locking Based Concurrency Control in Real - Time Databases

(Refer Slide Time: 00:15)

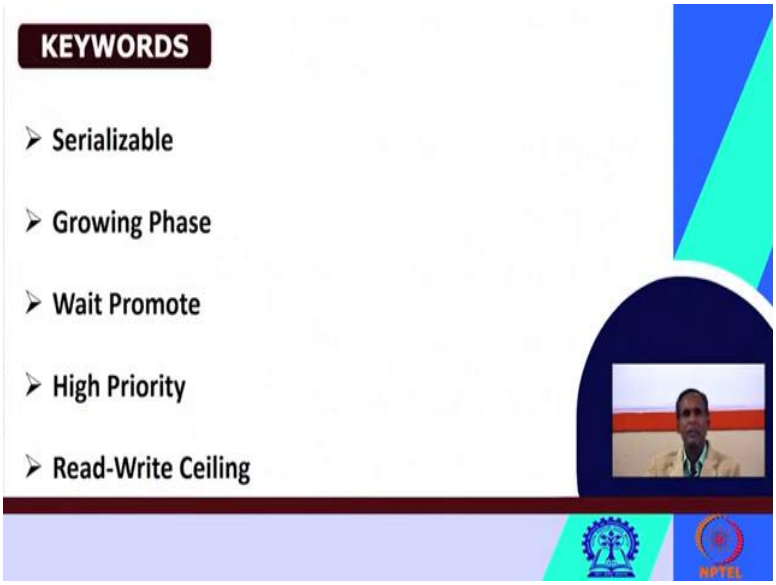
Good afternoon to all of you so, in this lecture, we will discuss some of the Locking Based Concurrency Control in Real Time Databases. How, concurrency control can be achieved in real time databases that we will see.

(Refer Slide Time: 00:32)



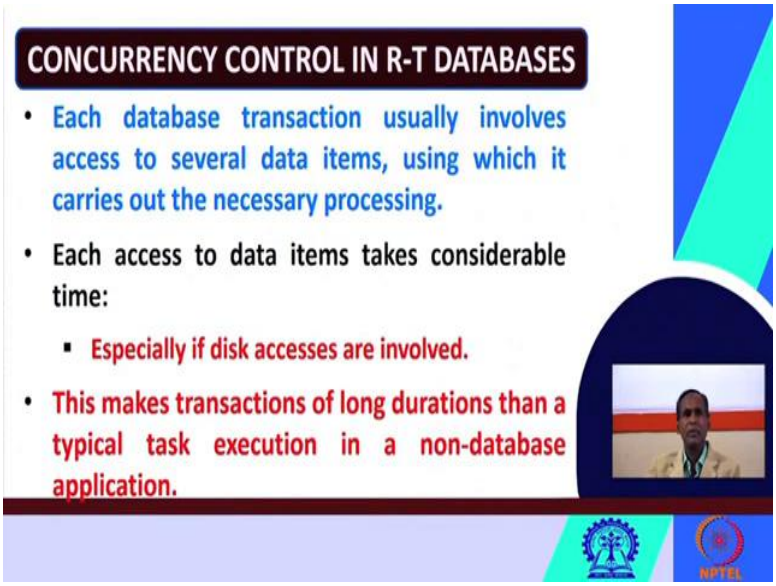
So, we will first see how concurrency control can be achieved in real time databases, then we will see some of the locking-based concurrency control protocols, such as the 2PL then variations of 2PL like 2PL a WP, 2PL HP and priority ceiling protocol.

(Refer Slide Time: 00:48)



These things we will discuss, these are the keywords we will use such as Serializable transactions, Growing phase, Wait Promote, High Priority and Read Ceiling, Write Ceiling, maybe Read Write Ceiling, etcetera.

(Refer Slide Time: 01:02)



Let us see first what do you mean by concurrency control in real time databases, you know each database transaction, usually involves the access to several data items, whenever you are saying some data base transaction, each database transaction, usually involves the access to

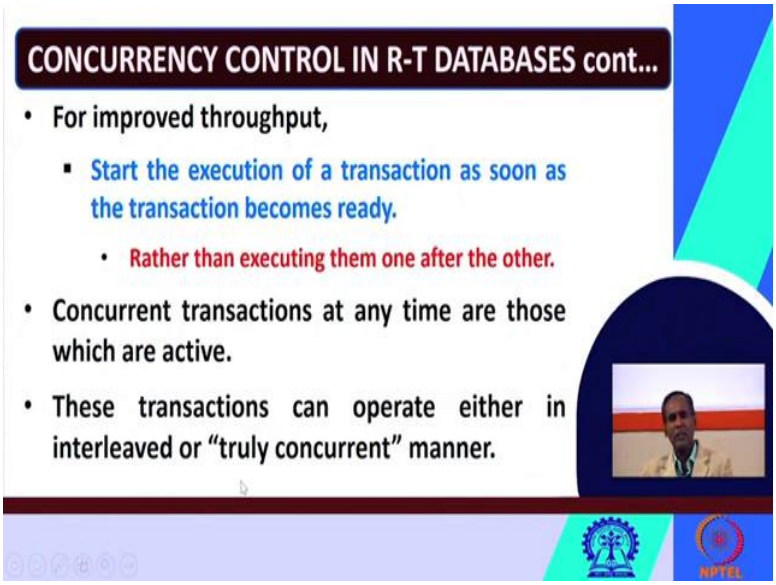
several data items. Using which it carries out the necessary processing, on every database transaction it usually involves access to several data items, present in the database, using which it carries out the necessary processing.

Each access to data items takes some amount of time. Whenever you are trying to or a transaction tries to access the data items obviously it takes some considerable amount of time, and this time will be very much large, if you are using disk accesses.

Last class I have already told you, the difference between using disk accesses and using disk storage devices and storing the data in main memory. So, if you are using disk accesses obviously it will take a huge amount of time to access the data. So, each access to data items takes a considerable amount of time, significant amount of time especially if you are using disk accesses, this makes the transactions of long durations, then a typical task execution in non-database applications.

If you are using these disk accesses and you are accessing the data items, then this makes the transactions of long durations, then the transactions they will become very long durations, then a typical task execution, that is happening in a non-database application. So, that is accesses data items maybe if you are using dick accesses it takes a huge amount of time. So, this will make the transactions of very long durations.

(Refer Slide Time: 02:48)



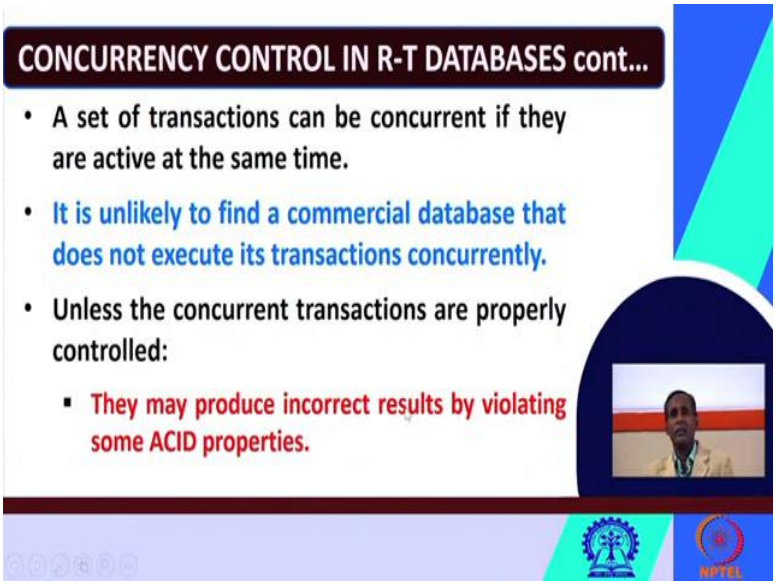
So, for improved throughput, what we have to do, for improved throughput, we have to start the execution of a transaction as soon as the transaction becomes ready. So, for improved throughput, what we should do, we should start the execution of a transaction as early as

possible as soon as possible or as soon as this transaction becomes ready. So, whenever a transaction is becoming ready, we should immediately start the execution of that transaction instead of executing it or executing the transactions one after another.

So, rather than executing the transactions one after another, in a sequential manner what you should do, you should start the execution of a transaction as soon as the transaction becomes ready. The concurrent transactions at any time are those which are active, what do may concurrent transactions, concurrent transactions are those transactions, which are active at any point of time. So, you consider any point of time say, T is equal to Tn, so at that time how many transactions are active, we call them as concurrent transactions. These transactions can operate either in interleaved manner or in true concurrent manner.

So, these concurrent transactions they can operate they can be executed either in an interleaved manner or in a true concurrent manner, that means, parallelly simultaneously they can be executed.

(Refer Slide Time: 04:11)



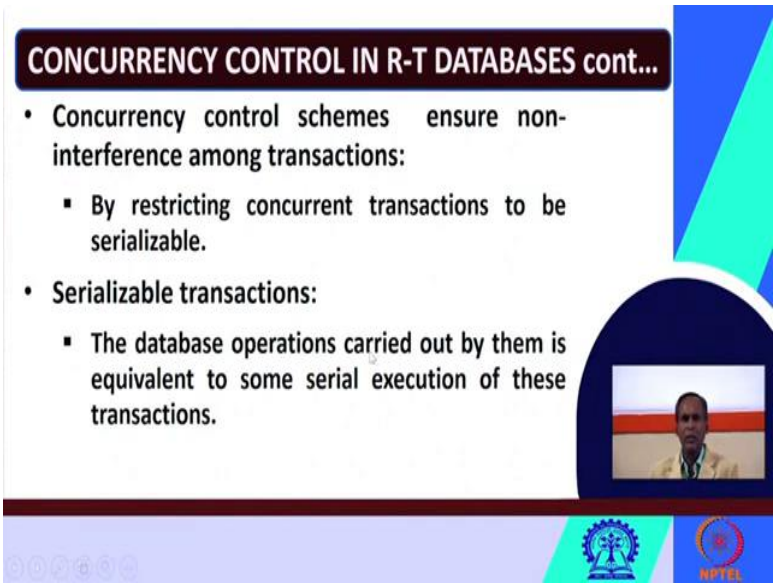
A set of transactions can be concurrent if they are active at the same time, so when you say that a set of transactions are concurrent, a set of transactions can be concurrent, if they are active at the same time. It is very much unlikely to find a commercial database which does not execute its transactions concurrently.

Nowadays, almost it is very much unlikely to find a commercial database which does not use, which does not execute its transactions concurrently. So, unless the concurrent transactions are

properly controlled, what problem may occur, they may produce incorrect results by violating some ACID properties.

So, if the concurrent transactions are not properly controlled, then these concrete transactions may produce some incorrect results by violating some of the ACID properties, ACID properties already we have discussed in the last class.

(Refer Slide Time: 05:01)



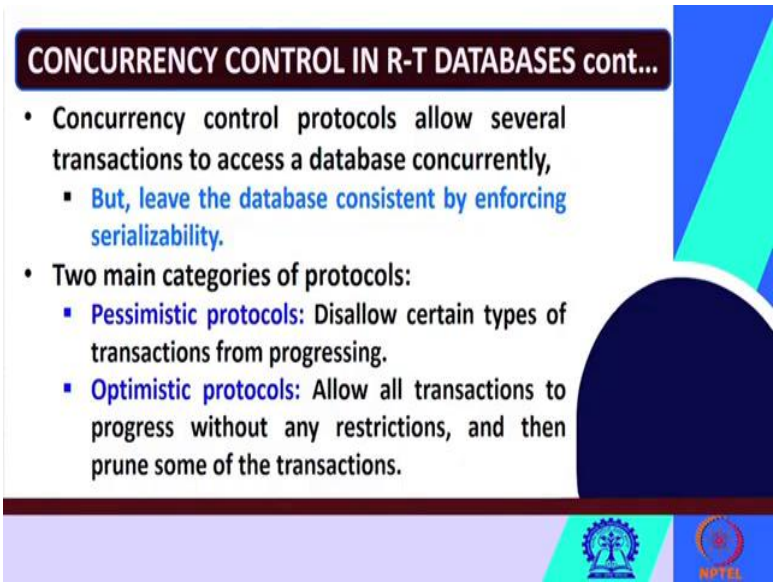
Now, let us see why you will use these concurrent, control schemes or concurrent control protocols, how they will help us, the concurrent control schemes or concurrency control protocols, they ensure non-interference among transactions, so if, if you require that the transactions they are not in, they are not interfering with each other, then we have to use concurrency control schemes or concurrence control protocols. How, by restricting concurrent transactions to be serializable.

So, these concurrency control schemes they restrict the concurrent transactions just to be serializable and, in this way, they can ensure that these transactions are not interfering among each other.

What do you mean by serializable transactions? So, serializable transactions means the database operations they are carried out by them is equivalent to some serial execution of these transactions. So, I hope this kind of things like serializable, serializable transactions etcetera, you have already read in the database paper, I am just quickly revising it in case of serializable transactions, the database operations are carried out by the serializable transactions, is equivalent to some serial execution of these transactions, as if about these transactions are

serially executed. So, in serializable transactions the database operations which are carried out by them is equivalent to some serial execution of these transactions.

(Refer Slide Time: 06:34)



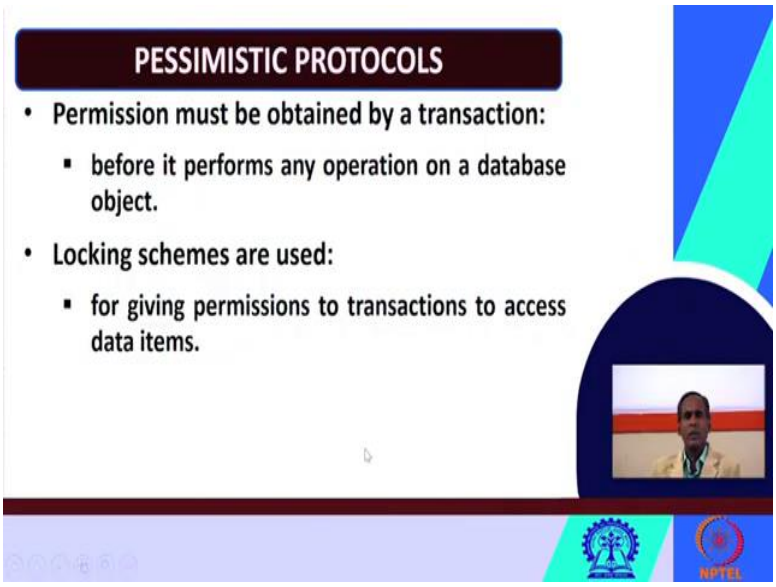
Now let us see this concurrency control protocols what do they do? The concurrency control protocols, they allow several transactions to access a database concurrently. So, by using concurrency control protocols, you can ensure that several number of transactions they can access a database concurrently, but they leave the database consistent by enforcing serializability. But even if several for transactions they are assessing the database, simultaneously, concurrently but, they will not leave the database inconsistent, they will leave the database consistent that will not be suffering from any problem.

The database that will be left as consistent, how, by enforcing serializability. By what is serializability I have just told here, so by following this concept of serializability, by enforcing serializability, these what transactions they leave the database consistent. Even if they are assessing or even if several transactions, they are assessing the database concurrently.

So, the concurrency control protocols, they permit several transactions to access a database concurrently or simultaneously without what making inconsistent the database. But they leave the database consistent by enforcing the serializability, there are two main categories of concurrency control protocols, number one Pessimistic protocols, number two Optimistic protocols, the difference between these two protocols are as follows, in pessimistic protocols, they do not allow certain types of transactions from progressing. Or they disallow certain types of transactions from progressing, optimistic protocols these, protocols allow all the transactions to progress without any restrictions.

There are no restrictions, all the transactions can be allowed to progress and then prune some of the transactions. So, first these protocols allow all the transactions to progress, without putting any restrictions and later on these transactions they prune some of the transactions, they cut or they what reject some of the transactions they prune some of the transactions.

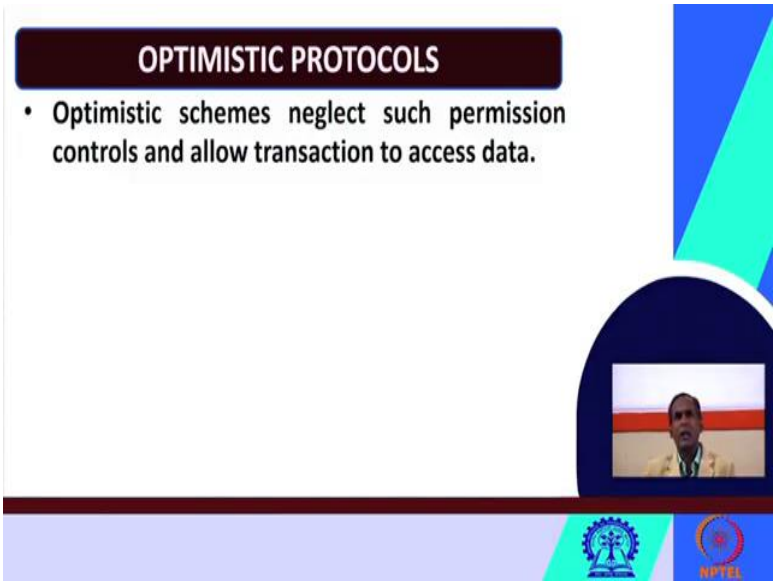
(Refer Slide Time: 08:44)



So, let us see about little bit about pessimistic protocols, I have already told you in pessimistic protocols they do not allow all the transactions to progress, so, in case of pessimistic protocols the permission must be obtained by a transaction before it performs any operation on a database object. In case of pessimistic protocols, you have to take permission or the permission must be taken by a transaction, before it can perform any operation on a database object and in pessimistic protocol also some locking schemes are used for giving permission to transactions to access the data items.

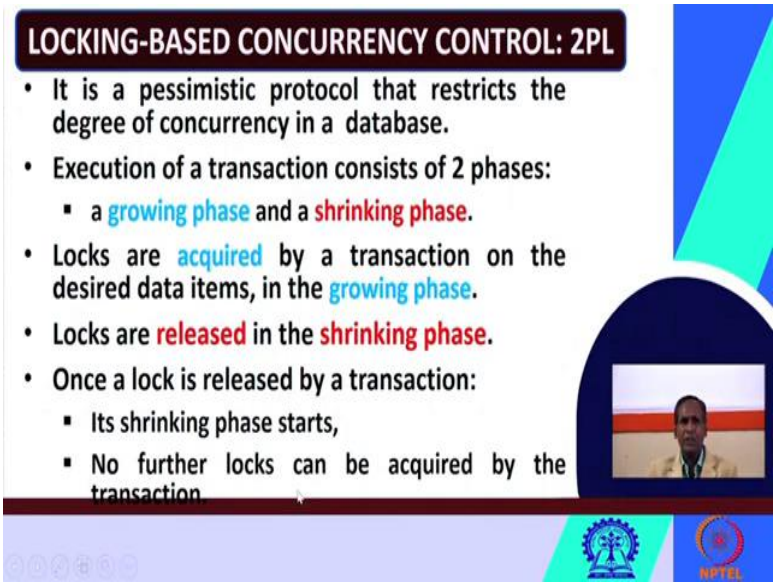
So, here some locking mechanisms are used to give permissions to the transactions in order to access the data items.

(Refer Slide Time: 09:28)



But in optimistic protocols the optimistic schemes they neglect such permission controls and allow transaction to access data, in optimistic protocols they neglect the such types of permission controls and they allow the transaction to access the data.

(Refer Slide Time: 09:45)



Now, we will see some of this first, we will see the pessimistic protocols I have already told you these details of the pessimistic protocols. Now, today we will see some of the pessimistic protocols and next class we will see some of the optimistic protocols. So, we will first see the locking-based concurrency control protocols, how it can be achieved by 2PL so 2PL is a pessimistic protocol it is transferred 2PL, 2P for phase, L for locking, 2 phase locking protocol

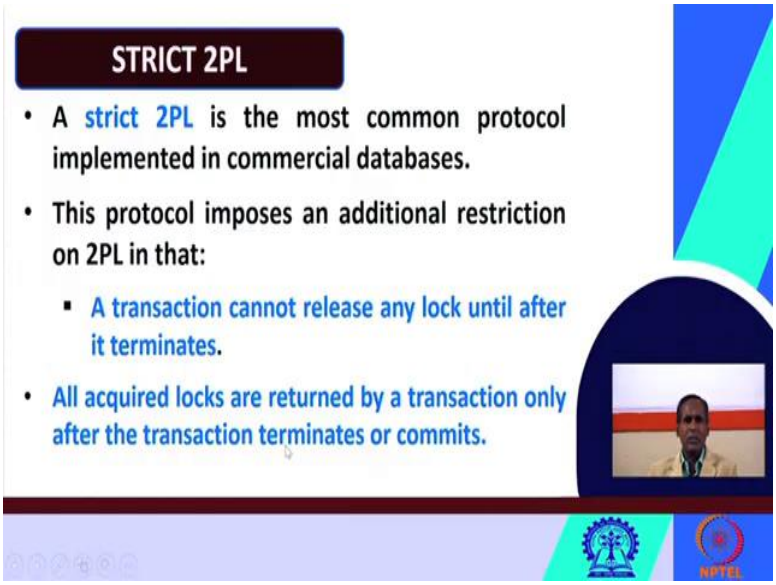
it is a pessimistic protocol. Which restricts the degree of a concur, the degree of concurrence in a database.

So, by using this protocol you can restrict the degree of concurrency in your database, as its name suggest 2 phase locking so, the execution of a transaction it consists of 2 phases first phase is the growing phase and the next phase is the shrinking phase, so what happens in growing phase, in growing phase locks are acquired, because we want that it is a log based concurrency control protocol. So, the transaction has to acquire the lock, so in the growing phase the locks are acquired by transaction on the desired data items, if a transaction requires to access data item d1, then it has to acquire the lock.

After acquiring the lock, you can access the data item d1, so in the growing phase, locks are acquired by a transaction on the desired data items, and then when its job is over the transaction completes its processing, then it can release the locks, when locks are released, locks are released in the shrinking phase. So, in the growing phase locks are acquired by a transaction on the desired database and acquiring the locks the transaction can do its processing and after its processing is over then, the locks are released and normally locks are released in a phase called as the shrinking phase.

So, there are two phases I have already told you in the growing phase, locks are acquired and in this shrinking phase locks are released, once a lock is released by the transaction, then this shrinking phase starts and no further locks can be acquired by the transaction. So, after once it has released this lock then the shrinking phase has started and after that no further locks can be acquired by the transaction. This is how this 2PL protocol this works or this 2L scheme it works.

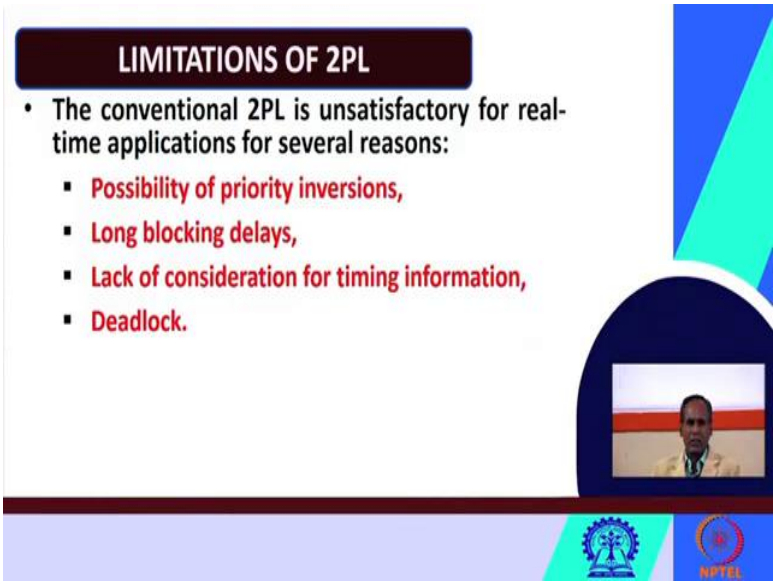
(Refer Slide Time: 12:12)



So, there is a variation of this 2PL scheme that is called as strict 2PL so, a strict 2PL is the most common protocol, which is implemented in the commercial databases and this protocol as I have already told, this is a variation or extension of this normal 2PL, let us see what does it do. This protocol imposes a special restriction or an additional restriction on the 2PL, what is that restriction, it says that a transaction cannot release any lock until after its determinates, so until a transaction terminates, until it has completed all these processing, it cannot release the lock.

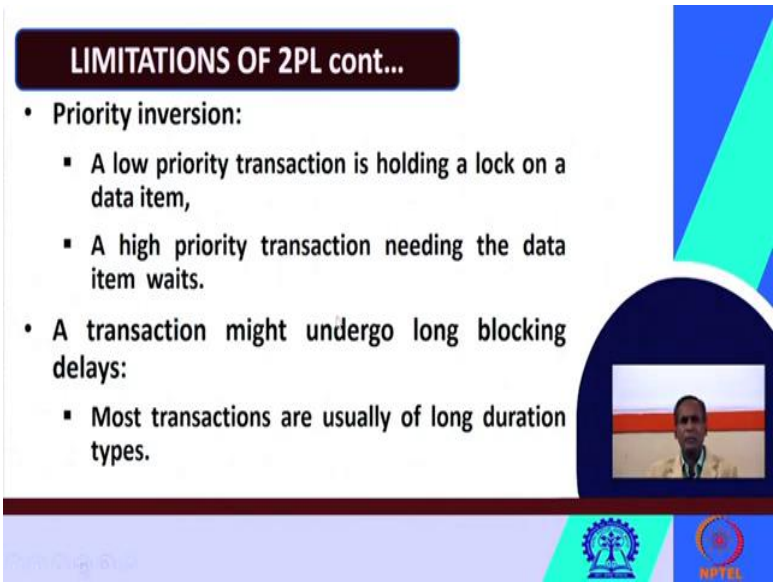
So, the additional restriction in strict 2PL is that, a transaction cannot release any lock until after it terminates. All required locks are returned by a transaction only after the transaction terminates or commits, this I have already told you. So, all the locks that a transaction has acquired, they can be returned only after the transaction has terminated or the transaction terminates or commits. Then only all the locks, all the acquire locks can be returned.

(Refer Slide Time: 13:18)



What are the limitations of 2PL, there are many limitations of 2PL the conventional or the ordinary 2PL is unsatisfactory for real-time applications due to the following reasons. If you are using 2PL there is a possibility of having priority inversions, what is priority inversion you have already known earlier, it can lead to long blocking delays, there may be lack of consideration for timing information and there is a chance of getting deadlocks. These are the limitations of this ordinary 2PL.

(Refer Slide Time: 13:49)

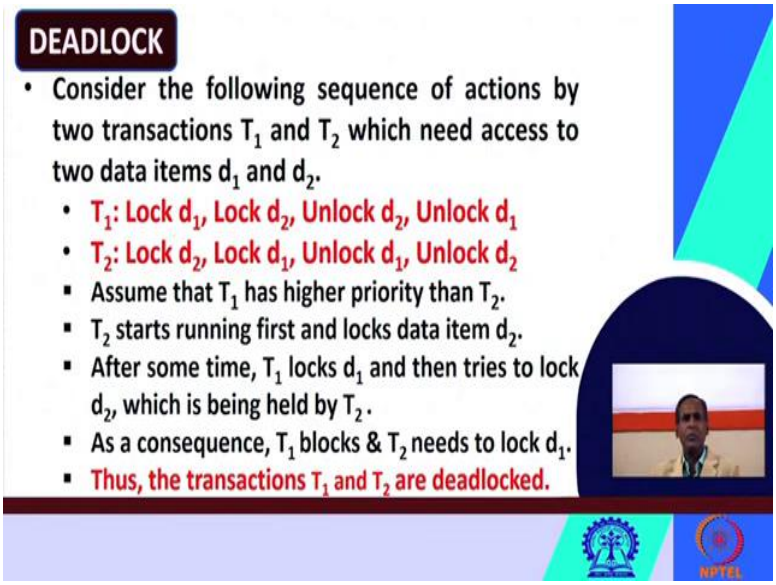


Now, let us see first one is what priority inversions, what does it say, a low priority transaction is holding a lock on a data item and when a high priority transaction it needs the data item, then it has to wait. This is known as priority inversion you have already known earlier.

So, when suppose, a low priority transaction is holding a lock on a particular data item d1 then a high priority transaction need to comes, it needs the data item the same data item d1 which is logged by a low priority transaction. Then, what will happen, the high priority transaction has to wait, so this is known as priority inversion. So, if you are using 2PL ordinary 2PL there is a possibility that priority inversion may occur. A transaction might undergo long blocking delays, most transactions are usually of long duration types.

So, it is possible that a transaction it might undergo long blocking delays the blocking delays varies more, so transaction might undergo lung blocking delays and you know why, because most of the transactions are usually of long duration types. So, in this 2PL most of the transactions are usually of long duration types, so transaction might undergo long blocking delays.

(Refer Slide Time: 15:01)



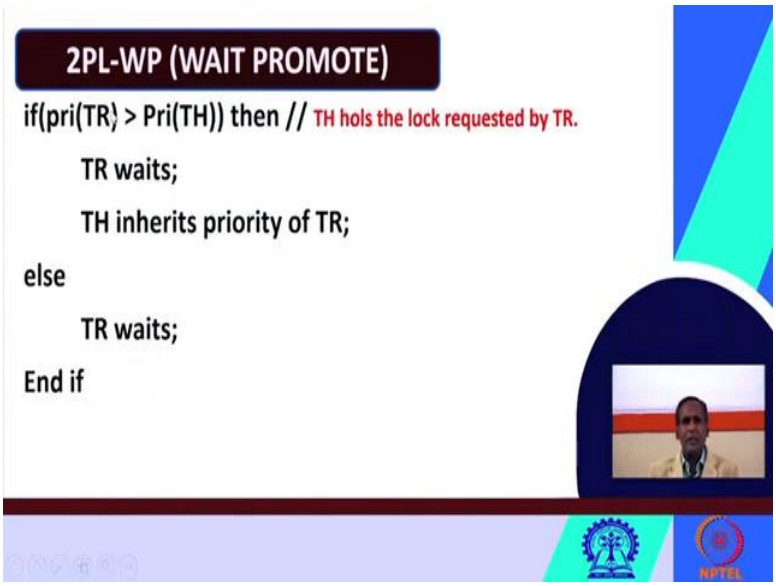
Then the third one, then the other one is, there might be a possibility of having deadlock let us see how a deadlock situation might arise in case of a simple 2PL. So, let us consider the following sequence of actions by two transactions T1 and T2. Suppose, there are two transactions T1 and T2 and they need both of the transactions, they need access to two data items d1 and d2.

So, first what happens and let us assume that T1 has a higher priority than the transaction T2, let us assume that, T2 it is a lower priority transaction than T1 and T2 first arrives and it is running, it starts running fast. So, T2 is a T2 starts running fast and it locks item d2. So, T2 locks the data item d2, after some time, the T1 transaction arrives and it locks the data item d1, no problem. Then, after some time T1 finds that it requires the data item d2 and hence, then it

tries to lock the data item d2 and d2 is now currently held by which one, T2. So, what will happen so as a consequence, what will happen, T1 blocks, T1 will be blocked.

And T2 now, needs to lock d1, and after sometime T2, it has already held T2 after some time, it needs to lock also to which one d1. But, d1 is held up by what, T1, so what will happen, both the transactions now there will be deadlock. So, both the transactions T1 and T2 are deadlocked, because T1 needs d2, which is held up by T2 and T2 needs d1 which is held up by T1. So, this is a deadlock situation, both the transactions T1 and T2 are deadlock. So, this is how the ordinary 2PL protocol it may lead to deadlock.

(Refer Slide Time: 16:56)



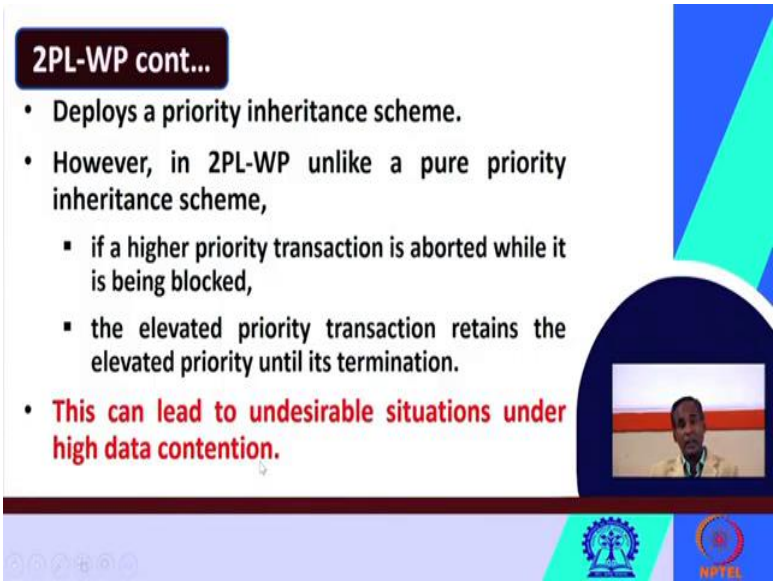
Then you will see one variance of ordinary 2PL, we call as 2PLWP, it overcomes some of the problems of the ordinary 2PL, WP stands for weight promote, the algorithm looks like this here it will first compare the priority of the TR and priority of TH. TH means not holds it should be holds, TH holds the lock request by a TR, so this transaction TH it holds the lock and which is requested by TR. So, TH has come early, so it has hold the lock on a data item and the TR it also request for the same data item, which is held by TH.

So, in this case what will happen, if priority of this TR is greater than priority of TH, then TR waits, because nothing can be done. TH has already hold this data item, so if a priority of TH has come first and it has to hold the data item. So, even if TR has a higher priority than TH, but it has come later on so, it has to wait.

If a priority of TR is greater than priority of TH then TR waits, and what will happen then TH will inherit the priority of TR. Because, TH needs to complete then, TH inherits the priority of

TR, else what will happen, if a priority of TR is less than priority of TH. No problem simple TR has to wait, no inheritance ration is required simply TR has to wait. So, in this way this 2PL WP works.

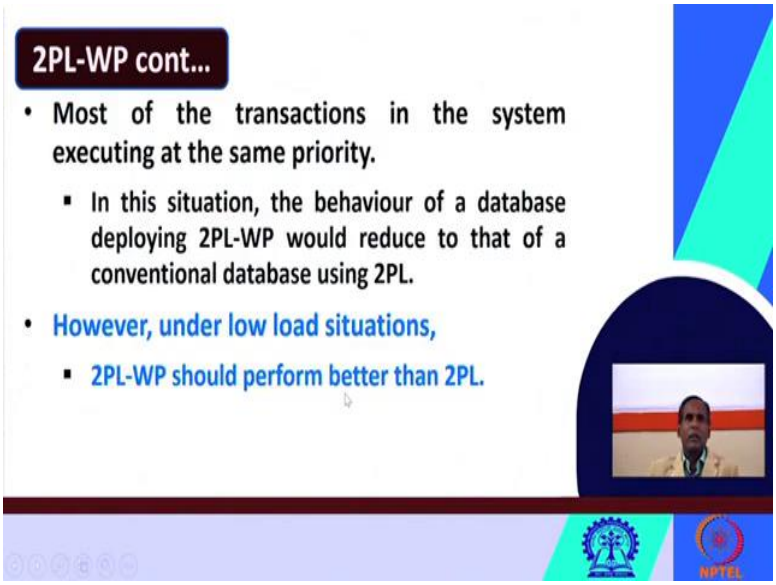
(Refer Slide Time: 18:31)



So, I have already told you 2 PL WP to employ a mechanism scheme called priority inheritance scheme. If TR is having higher priority than TH, then TH will inherit the priority of TR, so 2 PLWP it deploys a priority inheritance scheme, however in 2PL WP unlike a pure priority inheritance scheme, there is a difference. What happens if a high priority transaction is aborted, while it is being blocked then the elevated priority transaction it returns the elevator priority until its termination. Please see how this inheritance scheme it is little bit different than the pure priority inheritance scheme.

In case of the 2 PLWP if a higher priority transaction is aborted, while it is being blocked then what will happen, the elevated priority transaction, it will retain this elevated priority until its termination. So, until it is terminated, the elevated priority transaction, it will retain the elevated priority. So, this mechanism can lead to undesirable situations under high data contention, this feature can lead to undesirable situations under high data contention.

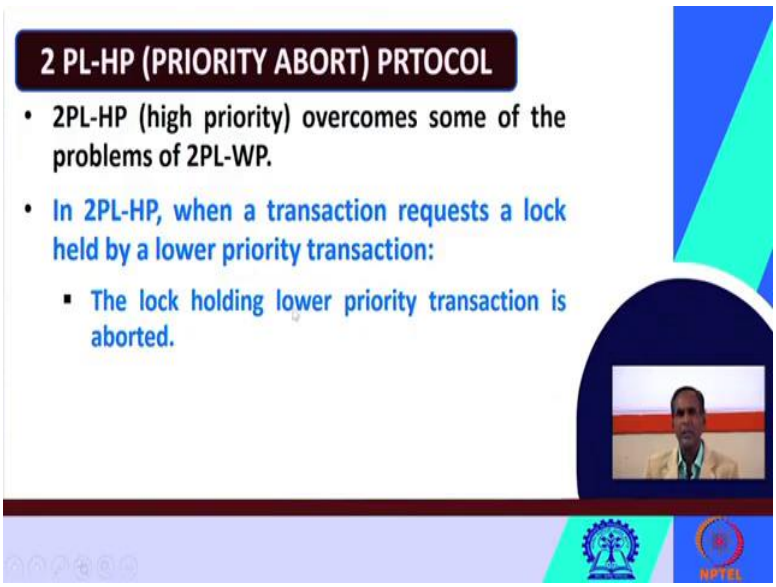
(Refer Slide Time: 19:48)



So, most of the transactions in the system they are executing at the same priority, so what in the consequence we are discussing in a 2PL WP, most of the transactions present in the system, they execute at the same priority. So, in this situation what will happen the behavior of a database, which is deploying the 2PL WP, it could reduce to that of a conventional databases using 2PL. So, when most of the transactions in the system they are executing are the same priority, in this situation the behavior of the database deploying, and this protocol 2PL WP protocol, it will simply reduce to that of a conventional database using 2PL.

However, under load situations 2PL WP should perform better than 2PL, so when the load situation is low, when there is very low load then 2PL WP it should perform better than 2PL.

(Refer Slide Time: 20:45)

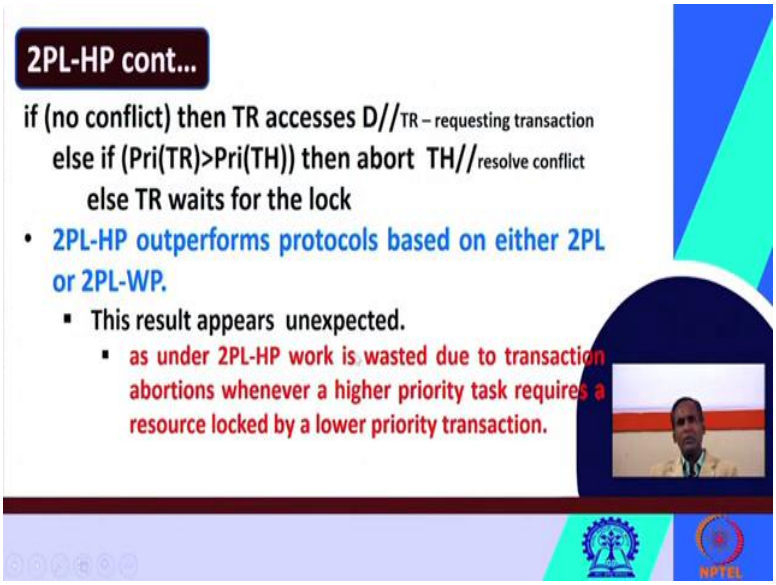


Then the next variation of 2PL protocol that is, 2PL HP is normally stands for the priority abort protocol, HP stands for high priority. So, 2PL HP it overcomes some of the problems of 2PL WP, so in 2PL WP when a transaction request a lock which is held by a lower priority transaction, then the lock holding lower prior transaction is aborted. Please see what I am saying here, this high priority is taken into account.

In this 2 PL HP protocol, whenever a transaction it request a lock which is held by a low priority transaction, in that case the lock holding lower price transaction, because the lower price transaction is holding the lock, in this case the lock holding low price transaction is aborted.

So, priority is given to whom, so more importance is given to the high priority transaction, so even if a low priority transaction is holding the lock, so when a high priority transaction request for this lock the low priority transaction, which holds the lock it has to be aborted. In 2 PL HP when a transaction request a lock held by a low priority transaction, then the lock holding the low priority transaction is aborted.

(Refer Slide Time: 22:01)



So, the pseudo code for this 2PL HP is as follows, if there are no conflicts, then TR accesses D, I have already told you TR is this requesting transaction, its price is more than TH, TH is the what transaction holding the lock if no conflict then TR accesses the data item, TR is the requesting transaction else that means if some conflict is there, then we have to check the priority of TR, and priority of TH.

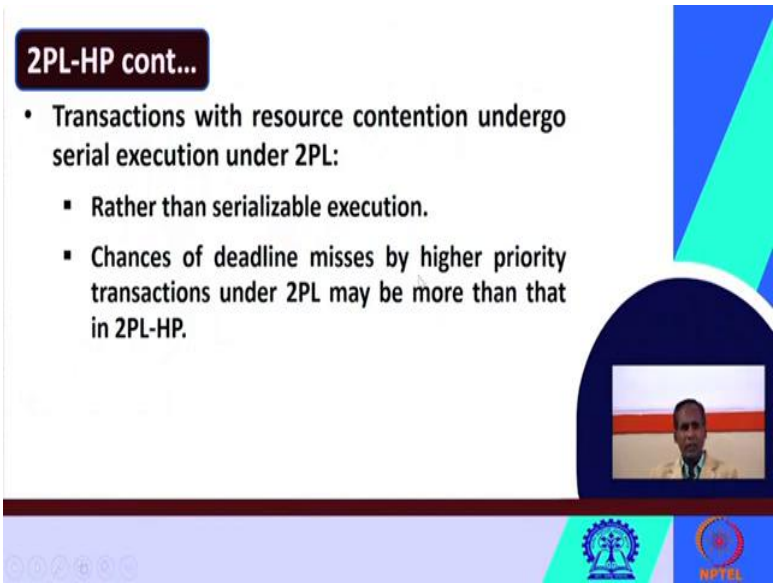
If priority of TR is greater than priority of TH, where TR means transaction requesting the lock and TH means transaction holding the lock, else if the priority of the transaction requesting the lock if its priority is greater than the priority of the transaction holding the lock.

Then you have to abort TH, no other way, resolve the conflict, by aborting the TH otherwise, that means if a priority of TR is not less than priority of TH, then TR has to wait. Then TR waits for the lock. 2 PL HP outperforms, protocols based on the either the 2PL or this. So, 2PL HP protocol it outperforms the protocols which are based on either 2PL or 2PL WP.

This result appears unexpected so, as I have already told you that the 2PLHP it outperforms this 2PL ordinary 2PL or 2PLWP, this result appears unexpected, as under 2PLHP the work is wasted due to transaction abortions. Because, here you aborting a transaction and in that case some work is wasted.

So, as under 2 PL HP some work is wasted due to this transaction abortions, whenever a high priority task requires a resource which is locked by a low priority transaction. So, this result appears on unexpected because, here some of the work will be wasted due to these abortions of the transactions.

(Refer Slide Time: 24:02)

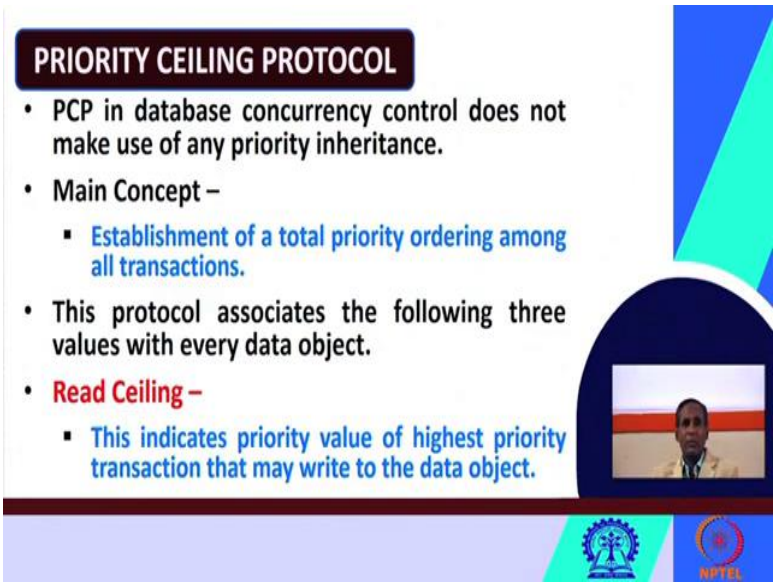


The transactions with resource contention undergo serial execution under 2PL, the transactions with resource contention, they might undergo serial execution under 2PL rather than serializable execution.

The transactions with resource contention they may undergo a serial execution, where under normal ordinate 2PL rather than serialization execution, chances of deadline miss by high

priority transactions under 2PL may be more than that in 2PL HP. So, if we will compare to in the ordinary 2PL and 2PL HP, then the chances of the deadline miss by the high priority transactions. It may be more than that of 2PL HP. So, in case of what PL the chances of deadline miss by the priority task, may be more than that in 2PL HP protocols.

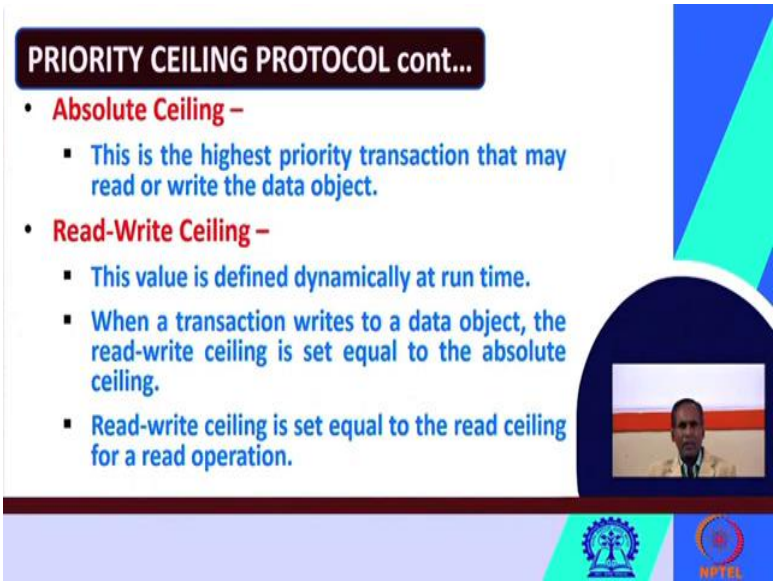
(Refer Slide Time: 24:53)



We will see the last protocol that is the priority ceiling protocol or PCP, you have already known PCP in earlier, what lectures. PCP in database concurrency control does not make use of any priority inheritance, please remember. This priority ceiling protocol in case of database concurrency control it does not make use of any priority inheritance, the main concept in priority ceiling protocol is that.

Establishment of a total priority ordering among all transactions, so it establishes a total priority ordering among all the transactions. So, in priority ceiling protocol, it establishes a total priority ordering among all the transactions, this protocol associates three values with every data objects. This priority ceiling protocol in this real time databases it associates three values with every data object.

(Refer Slide Time: 25:52)



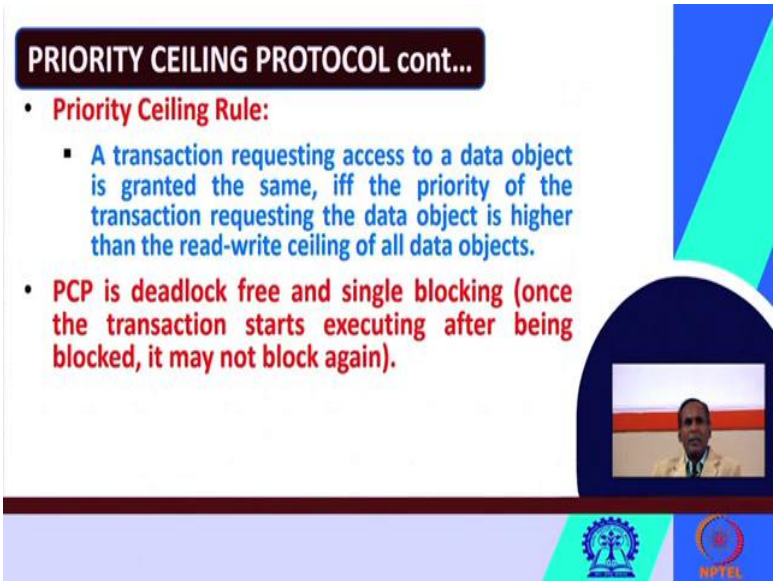
One is read ceiling, another is absolute ceiling, another is read write ceiling. Let us see in read ceiling what is happening this indicates priority value of highest priority transaction that may write to the data object. So, read ceiling means it indicates the priority value of the highest priority transaction, which may write to the data objects. So, that is why the name is what it is known as read ceiling.

Then absolute ceiling so, this is the highest priority transaction, that may read or write both, in absolute ceiling this is the highest prior transaction that may either read or writing the data object, and the last one is read write ceiling, this value is defined dynamically at runtime. So, this read write ceiling this can be defined dynamically during runtime. When a transaction writes to a data object the read write ceiling is set equal to the absolute ceiling.

So, whenever a transaction tries to write to a data object what will happen, the read write ceiling value will be set equal to the absolute ceiling whereas I have already told you absolute ceiling is the highest priority transaction, which may read or write the data object.

Read write ceiling is set equal to the read ceiling for a read operation, so when you are what performing the read operation, the read write ceiling is made equal to it set equal to the read ceiling for a read operation what is the read ceiling already you have shown earlier.

(Refer Slide Time: 27:19)

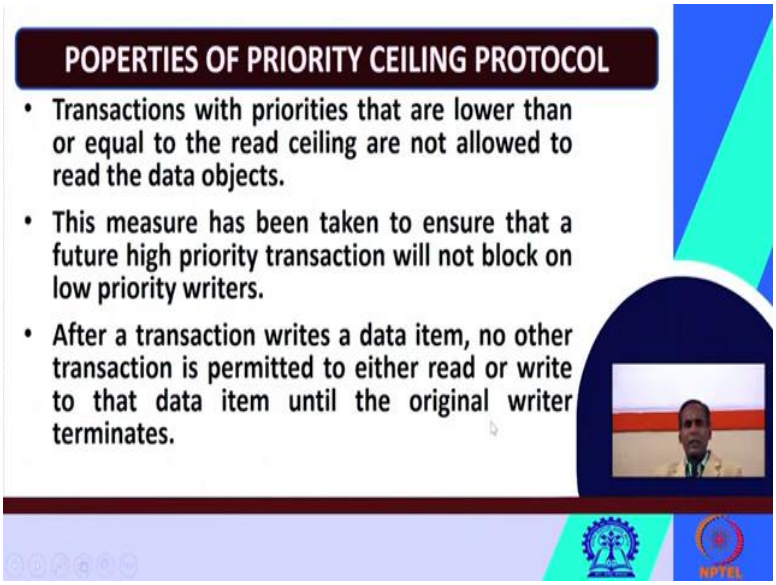


So, now let us see what is the priority ceiling rule what does it say, priority ceiling rule says that a transaction requesting access to a data object is granted the same if the following condition is satisfied. What is the condition? If the priority of the transaction requesting the data object is higher than the read write ceiling of all that objects, I am repeating again what is the priority ceiling rule a transaction requesting access to a data object, it is granted the same if and only if the following condition is satisfied.

What is the condition? For this, the priority of the transaction which request the object it should be higher than the read write ceiling of all the data objects. Then we said that the transaction which request access to that object it is granted the same.

Please remember that PCP is deadlock free and single blocking, so PCP we have already seen earlier 2PL one of the drawbacks that it may suffer from deadlock, but PCP is deadlock free and it is single blocking, single blocking you have already read earlier, single blocking means what, once this transaction starts executing after being blocked, it will not block again. So, once the transaction starts executing after being blocked further it will not be blocked again, it may not block again.

(Refer Slide Time: 28:48)

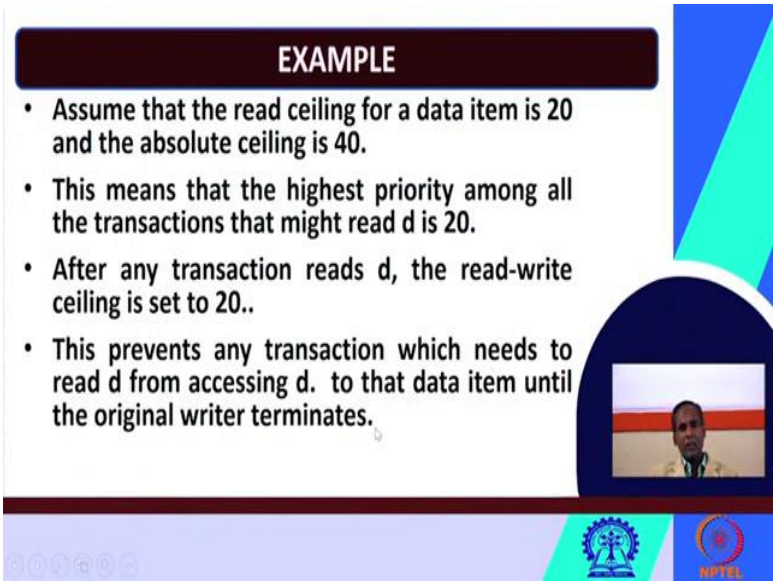


So, let us quickly see some of the properties of priority ceiling protocol, in priority ceiling protocols, the transactions with priorities which are lower than or equal to the read ceiling are not allowed to read the data objects. Please remember the transactions with the priorities which are lower than or equal to the ceiling, read ceiling these are not allowed to read the data objects. Why, this measure has been taken, this measure has been taken or this constraint has been put in order to ensure that a future high priority transaction, it will not block on low priority writers. So, what this condition I have told.

The transactions to the priorities that are lower than or equal to the read ceiling, they are not allowed to read the data objects, why, this provision has been made, this provision is made for the following reason, a high priority transaction which may come in future will not block on low priority writers. After a transaction writes a data item no other transaction is permitted to either read or write to that data item. So, after a transaction it writes a data item no other transaction is permitted to either read or write to that data item until the original writer terminates.

So, suppose a transaction writes a data item, no other transaction will be allowed to either read that data item or write to that data item until the original writer is terminates. So, these are some of the properties of the priority ceiling protocol.

(Refer Slide Time: 30:24)

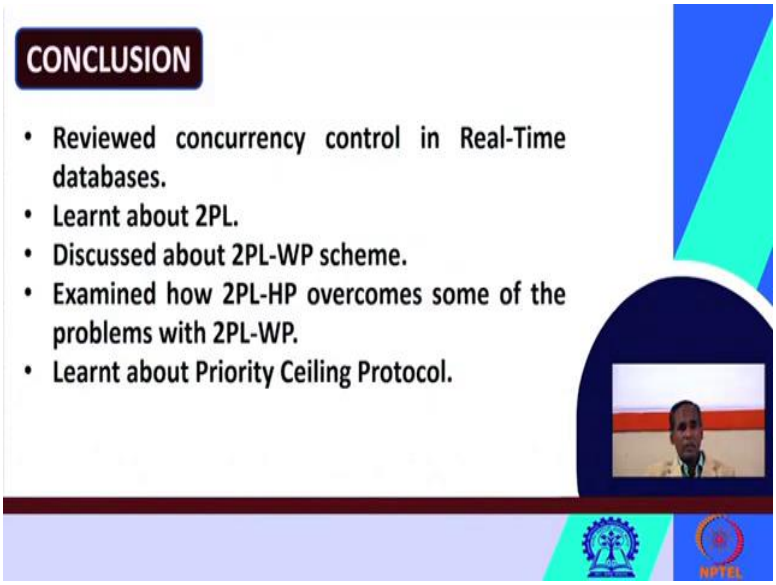


We can quickly take a small example, assume that the read ceiling for a data item is 20, and the absolute ceiling is 40. So, what is read ceiling, what is absolute ceiling I have already explained here. So, let us assume that the read ceiling for a data item is 20 and the absolute ceiling is 40. So, what does it mean this means that the highest priority among all the transactions that might be read d is 20, for this example where the read ceiling for a data item is 20 and the data, the absolute ceiling is 40?

So, one important conclusion we can draw, what is that conclusion this means that, the highest priority among all the transactions that might read the data item d is 20. After any transaction reads d then read-write ceiling is set to 20, so after any transaction it reads this data value, this data item d then d read write ceiling is set to 20. This feature prevents any transaction which needs to read this data item d from accessing d to that data item until the original writer terminates.

So, the original writer terminates, any transaction or any other transaction cannot, what read d from accessing d so, in this way we can prevent any transaction which needs to read the d from accessing the data item d. Until the original writer terminates.

(Refer Slide Time: 32:01)



So, today we have discussed the concurrency control in real-time databases, we have discussed about the ordinary two-phase locking protocol, we have also discussed some variations of 2PL like 2PL WP, 2PL HP etcetera. We have also explained how 2PL HP overcome some of the problems of 2PL WP and finally we have discussed the fundamental concepts of priority ceiling protocol in real-time databases.

(Refer Slide Time: 32:26)

We have taken these concepts from these two books, thank you very much.