Fundamentals of Database Systems

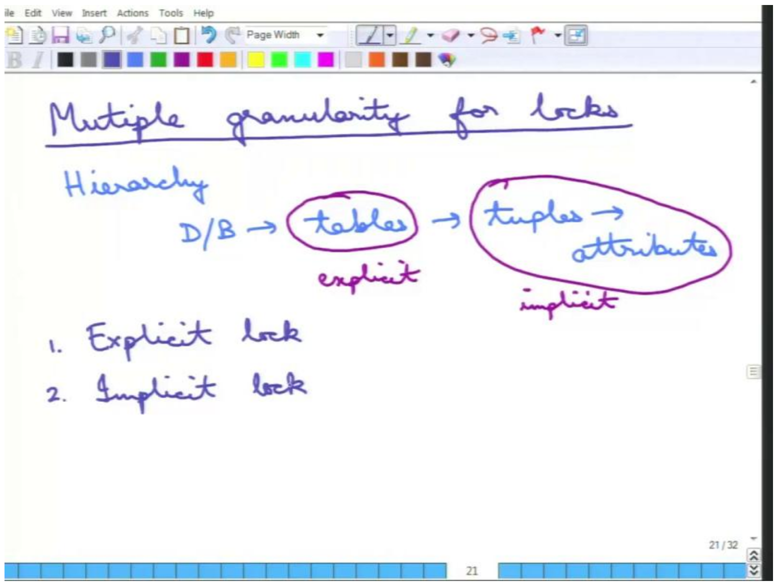
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 42
Concurrency Control: Multiple Granularity for Locks

We will next continue with some other kinds of issues in the Concurrency Control Protocols. So, we will return to locks, but more importantly, we will return to locks in what is called a Multiple Granularity. So, if you have noticed, then we have been understanding locks, we have been studying locks for various name data items. But, we have not bothered about, what the data items were actually.

Now, what happens in that, in some cases is that, suppose there is a table; that the entire table needs to be locked and then, within the table, there are records or tuples that needs to be locked again and within in a tuple, there may be an attribute that needs to be allowed. So, the lock is essentially applied at multiple granularity, so the lock may be applied to only an attribute or to a table or to a record or to a entire database and so on and so forth and then, there are some issues that we will discuss themselves.

(Refer Slide Time: 01:05)


Multiple granularity for locks
Hierarchy
D/B → tables → tuples → attributes
explicit
implicit
1. Explicit lock
2. Implicit lock

So, this is the issue of multiple granularity for locks, so multiple granularity. So, the granularity essentially means at which level the lock is applied. Now, what one may do is that, so the hierarchy, so if this takes into account the hierarchy of the database. So, the hierarchy may be there is a database, then there are different tables, then there are tuples or records within that, then there are attributes within that and so on and so forth. This is the hierarchy and it may be, locks may be applied at different levels.

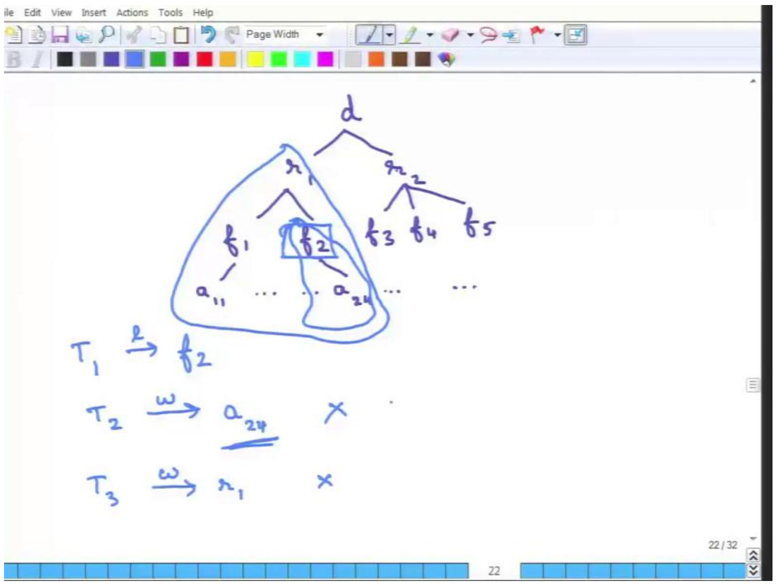
Now, the locking, when it is done at different levels, then the following thing happens. So, suppose a particular table is locked. Now, what does locking a particular table mean? It is essentially, even a particular table is locked that is an explicit lock, but implicitly what happens is that all the tuples and all the attributes in all the tuples are also assumed to be locked implicitly. Because, otherwise you can see what is the problem is that.

So, suppose a table is locked; that means, the table needs to be read, etcetera and if the tuple is not locked, if the tuple is not supposed to be locked, then what may happen is that some other transaction may come and writes to the tuple. So, then this problem of summarization, problem of maintaining, statistics, etcetera may get harm. So, that is why these are also supposed to be locked, but these are not locked in explicitly, this is locked in an implicit mode.

So, then essentially there are they; that means, there are two kinds of locks which is an explicit lock and then there is an implicit lock and locking something, explicitly locks everything under it in a implicit mode. So, the granularity of locking essentially is that, this can be in a fine granularity; that means, this is lower in the hierarchy, so something like attributes are being locked, etcetera and; that means, one can assume higher concurrency, because if only an attribute is locked, other parts of the tuple can be accessed, other tables can be accessed, etcetera.

But, it is a high locking overhead, because every attribute is being locked, etcetera. On the other hand if it is a coarse granularity, then something coarse such as tables, etcetera is locked. So, it reduces the concurrency, but the locking overhead is low.

(Refer Slide Time: 03:30)


d
r_1
r_2
f_1
f_2
f_3
f_4
f_5
a_{11} ... ... a_{24} ... ...
$T_1 \xrightarrow{r} f_2$
$T_2 \xrightarrow{w} a_{24}$ ×
$T_3 \xrightarrow{w} r_1$ ×
22 / 32
22

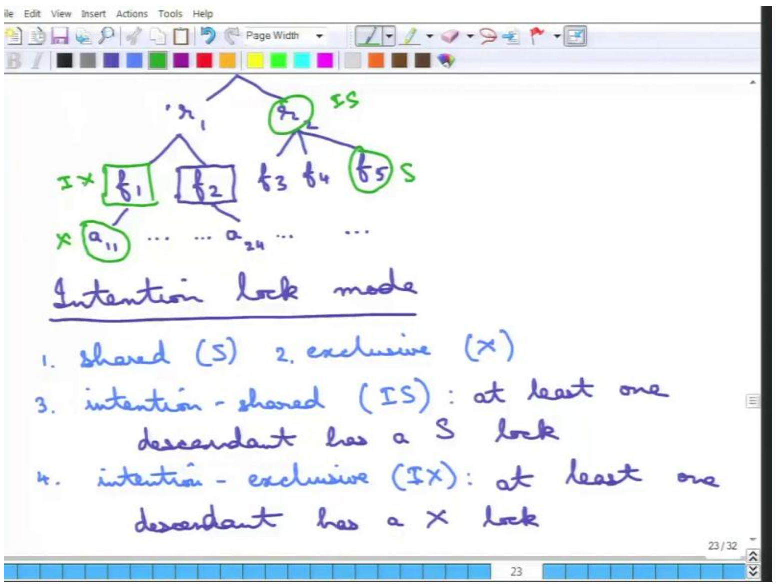
Anyway, so here is an example of a hierarchy, so this is the database d and then, under it, there are assumed, there are two tables r 1 and r 2, under r 1, this is just an illustrating example. So, suppose there are these tuples f 1 and f 2 and under r 2, there are this f 3, f 4 and f 5 and under f 1, there are these attributes a 11, etcetera and so on and so forth. So, this is the thing that to be done.

Now, suppose there is a transaction T 1, T 1 has locked tuple f 2, so T 1 has locked f 2, now T 2, so this has locked already, now T 2 wants to lock a 2 4. What is going to happen is that, this should not be allowed, because even though f 2 is locked explicitly, this attribute a 24 is locked implicitly. So, this request cannot be granted, because this is an implicit lock that is been taken up a 2 4. So, that is because, it has locked f 2, so this goes to because f 2 is locked, everything under it is supposed to be locked as way. But, suppose T 3 wants to lock r 1, should it be allowed, T 3 wants to lock r 1; that is also not allowed, because if T 3 locks r 1, then it means that T 3 will be locking this entire thing.

So, suppose T 3 wants to lock r 1, it is not going to be allowed, because if T 3 locks r 1; then it means, this T 3 is going to lock this entire thing, which means that the lock at f 2 again clashes with that at T 1. So, it is not going to be allowed, so essentially what is being meant is that, if f 2 is locked, all it is descendents are locked, as well as all it is ascendants up to the database cannot be taken a lock on any of it is ascendants up to the database. And none of it is descendents in it is sub tree cannot be locked either, so that is the thing about implicit locks.

So, r 1 could not be locked, even though, there is no such implicit lock on r 1, but r 1 still cannot be locked, because locking r 1 would actually mean an implicit lock on f 2. So, that would clash with the explicit lock of T 1, so that is why, it is not going to be allowed, so that is the thing. So, this you can see that, this is clearly going to be a little inefficient, etcetera.

(Refer Slide Time: 05:47)



Intention lock mode

1. shared (S) 2. exclusive (X)
3. intention-shared (IS): at least one descendant has a S lock
4. intention-exclusive (IX): at least one descendant has a X lock

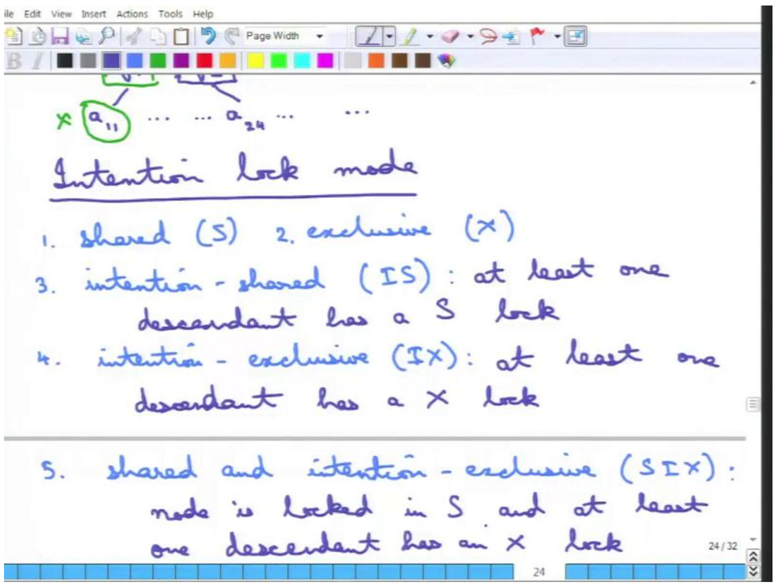
So, what is being done is that, there is something called an intention lock mode; that is being used. So, intention locking mode, intention lock mode that is used, so that is essentially what we have been discussing. So, if f 2 is locked, then all it is ascendants are in the intention lock mode. So, what is ascendant is in the intention mode, so it is not an implicit lock, but it is an intention mode lock, so that is the intention mode.

So, we have been studied, we had already studied this shared lock S and the exclusive lock X; that we had studied for the read write. If you remember, there was the lock matrix as well S, X, etcetera. The intention lock mode introduces three additional locks, so this is 1 and 2, but it introduces three more locks, the first one is called an intention shared lock, so this is the IS lock.

So, what is an intention shared lock? This means that, at least one descendant has a S lock. So, at least one descendant has a S lock, so then it is called an intention shared lock. So, what does this mean is that, suppose this is locked in S mode, then r 2 is supposed to be in an IS mode, because at least one of it is descendant is in a S lock. So, this is why the intention lock mode is being used.

Then, similar to IS there is something called an IX, which is intention exclusive lock mode IX. So, this is the definition is similar, so at least one descendant has an X lock. So, that is why, so this is called an intention exclusive, because this is really not an exclusive mode, but it is an intent of an exclusive mode, because some descendant has a X lock. So, again this is easy to see, what can be the example, so suppose, this is X mode, then this is in IX mode.

(Refer Slide Time: 08:18)

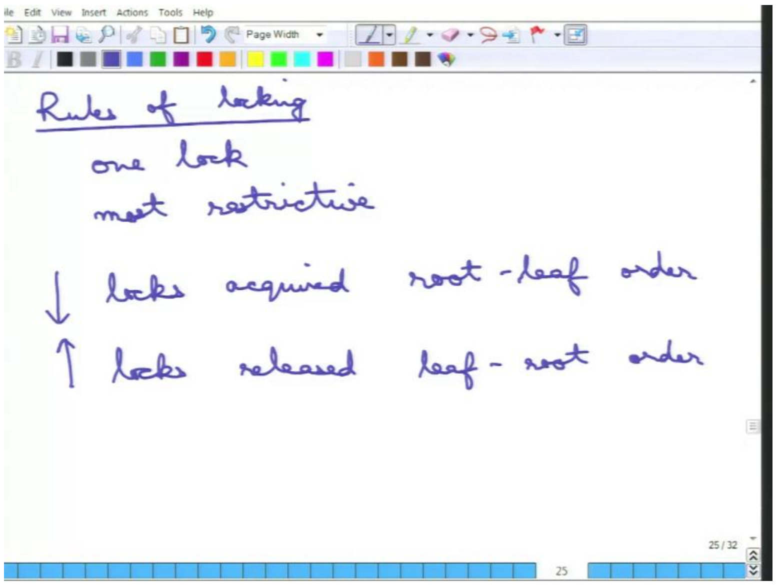

Intention lock mode

1. shared (S) 2. exclusive (X)
3. intention - shared (IS) : at least one descendant has a S lock
4. intention - exclusive (IX) : at least one descendant has a X lock

5. shared and intention - exclusive (SIX) : node is locked in S and at least one descendant has an X lock

Very, very importantly, there is a 5th kind of lock, which is called a shared and intention exclusive lock, this is a both shared and intention exclusive lock, this is called the SIX lock. This is a famous rather an unfamous lock, this is a SIX lock. So, shared and intention exclusive lock, this thing is that, the node itself is in is locked in S mode and at least one descendant has X node. So, this is what a shared this one is in X lock, X mode, so this is a shared intensive lock. So, the shared intensive lock is a little bit complicated, we will come to that.

(Refer Slide Time: 09:19)



Rules of locking

one lock

most restrictive

$\downarrow$ locks acquired root - leaf order

$\uparrow$ locks released leaf - root order

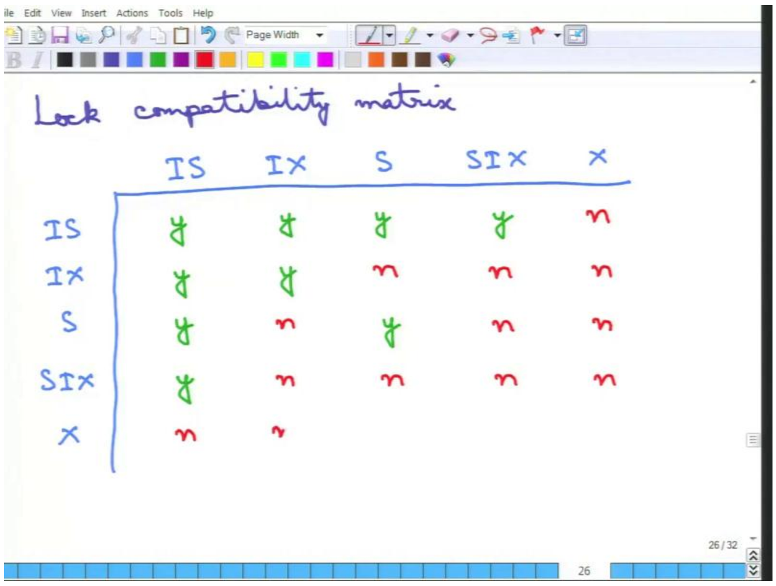
But, before that, let us go through the rules of locking. Using this now, we have five locks in our hand and so we must have the rules of locking, we must know the rules of locking. So, the first rule, actually some simple rule is that a transaction may obtain only one lock at a time, on an entity at a time. So, that means, if a transaction wants a lock on one data item, it can only get one lock, it cannot get two locks, so it cannot for an example get an S lock and an X lock, it should get only one lock. Now, the question is, which lock will it get? It will get the most restrictive lock.

So, it should get only one lock and that is the most restrictive lock that it should get. So, if it wants both S and X, it should get the most restrictive, which means, it should get X. And every lock must be given notice of all low level locks, so that means that whenever something, whenever transaction locks a data item, it sends the notice up to all the ascendants.

So, the ascendants must be know, in that something has happened in the descendants, otherwise it cannot get this intention lock mode, so that is there. So, the locks are acquired in a root to leaf order. So, from top to bottom; that means and the locks are released in a leaf to root order. Little bit of thinking will tell you why that is necessary is that, before the locks are released, some other transaction may come and take another locks.

For all those cases of conflicts do not happen, the locks are acquired in a root to leaf; that means, the highest lock that is needed is first got; that means, that everything under this is supposed to be locked and then, the next lock in the level is acquired and so on and so forth. On the other end, when it is released, it is released in the opposite order, because otherwise what may happen is that, suppose a higher leaf is released, then anybody checking the higher leaf would think, higher node will think that everything else below is not locked; that is wrong. So, it is released in the leaf to root order. So, these are the rules of locking, but very, very importantly the one thing that is left is that, the actual locking matrix.

(Refer Slide Time: 11:45)



Lock compatibility matrix

	IS	IX	S	SIX	X
IS	y	y	y	y	n
IX	y	y	n	n	n
S	y	n	y	n	n
SIX	y	n	n	n	n
X	n	n			

Now, we have five kinds of locks and the lock compatibility matrix must be noted for all of them. So, this is the lock compatibility matrix between all these five kinds of locks. So, I am going to draw the lock compatibility matrix in a moment, so the locks are the following, so this is a is lock then there is an IX lock, then there is an S lock, then there is SIX lock, then there is an X lock, this is one thing.

And on the rows, there is the same thing IS, then this is IX, then this is S, then there is SIX lock and then, there is a X. The reason I need to writing this is the least restrictive to most restrictive. Let us complete the lock compatibility matrix and then, we will understand what this is why I am saying this. So, this is S; that means, if one transaction gets an IS lock on one data item, another transaction can then get an IS lock on that same data item.

This is also y; that means, IS and IX also do not conflict IS and S do not conflict, IS and SIX do not conflict. However, is and X do conflict; that means that, if a transaction is holding an intention shared lock on one data item, another transaction cannot get an X lock on it, so that is very important. So, no that this is no, next we continue with the IX to IS is S, this we have already seen and IX to IX is also S, if both transactions can hold an IX log; that means, some of the one of the descendants is in X for both the things and we will worry about it later, which descendent etcetera.

But, otherwise this is no problem; IX to S is however no. So, one cannot get a shared lock on a node, if one of it is descendent is in a exclusive mode, because what does it mean, why is this a problem, a shared lock on everything means that, it is probably reading that entire table etcetera. Then, IX meaning, there is something inside is locked for writing.

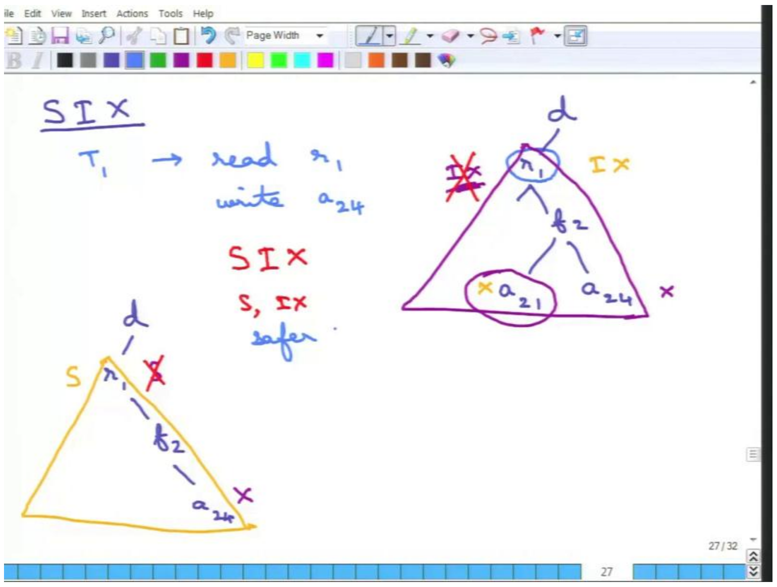
Now, that means, that if the writing is being done, while it is reading then the summary etcetera will be wrong, that is why, this is not allowed. Now, IX to SIX is also not allowed and IX to X of course, not allowed. Next comes here the S lock, which is the shared lock, now shared to IS is y; there is no problem, shared to IX we have already seen, this is no. Then, shared to shared will be yes, both of them can get a shared lock, no problem.

But, this SIX lock cannot be here within the X lock and X is no of course, then comes the SIX lock, the SIX lock is this is yes, then with IX, SIX lock cannot be shared. So, then SIX lock cannot be shared with S as well, SIX lock cannot be shared with SIX lock itself and nothing can be shared with X. And finally, X lock cannot be shared with anything. Now, if you notice this matrix is symmetric, as it should be, because it does not matter which transaction is on the row and which transaction is on the column, but here are the things.

So, couple of important things to notice that, X lock cannot be shared with anything, if something is being exclusively locked for writing; that means, that is supposed to be written and; that means, it cannot be shared with anything else. And if something is being exclusive and something has been intentionally locked.

So, in the intention mode of shared lock; that means, something below it is only read and even this is not being read, then it can share with most of the locks except of course, the X lock, because something inside this being written. So, that cannot be done, this is the lock compatibility matrix that one must remember to get an understanding. Now, it is probably easier to understand all those locks except the SIX locks. So, what it exactly is the SIX lock, the SIX lock is something little bit more interesting etcetera.

(Refer Slide Time: 15:55)



So, let us go over the SIX lock in a little bit more detail. So, what does SIX lock, why a SIX lock needed, how it is useful. Now, suppose there is a transaction T 1; that wants to read r 1, this is reading r 1, but modify, but it wants to write a 24. So, if you remember this was one is being done etcetera and then, there was an f 2 under which there was an a 24.

So, transaction T 1 wants to read r 1 and write to a 4, now if transaction T 1 wants to read and it writes to a 4, then what happens is that, how on which mode will it lock r 1, so that is the question. So, the thing is that, if r 1 is locked in the let us say r 1 is being locked in the IX mode, so that is the first thing r 1 is locked in the IX mode, which means that, because there is something X going on here it can be IX mode.

Now, which means, that any other transaction that wants to lock r into get this IX mode. Now, let us check up these things, IS to IX is y, so that means, any other transactions, which wants to lock r 1 in a IX mode may also be granted, so that means, some other transaction may also lock this r 1 into IX mode. So, this is unsafe, why because if the some other transaction is locking this, then what may happen is that some other transaction may be actually modifying a 21, which is unsafe, because T 1 is supposed to be reading the entire r 1; that is why, that is T 1 is reading r 1, so that is these thing.

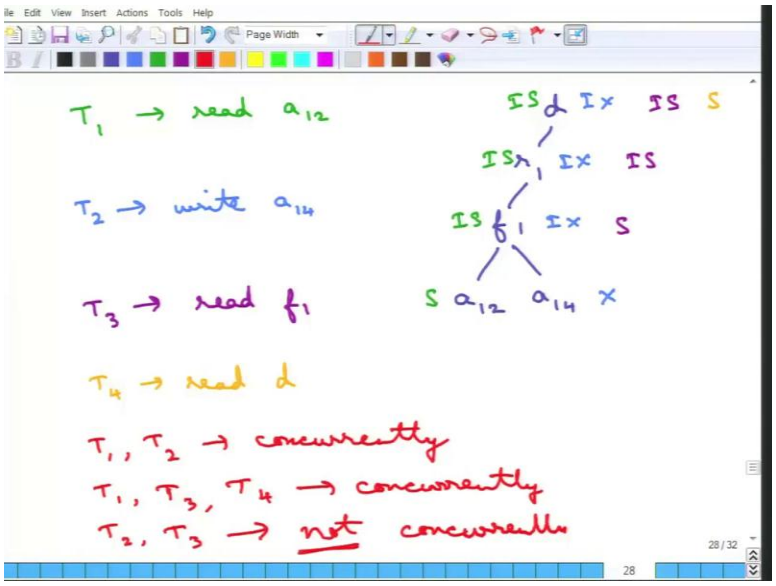
So, that means, this is unsafe, this is where the problem will happen. So, it should not allow it into IX mode. So, that is the first part of it and what about if it only locks it in an S mode, so suppose on the other hand, the same example. So, let me redraw the example here once more, this is r 1, then this is f 2, this is a 24. Now, what if this is locked only in the S mode, because it is only reading, then some other transactions may also come and lock it in a S mode.

So, which means that, what happens if this is locked in the S mode, then some other transaction may come and also lock it in the S mode because S to S is allowed. So, then; that means, that although there is an X here, these transaction the other transaction when it is reading this entire thing will have a problem, because this is being modified. So, S lock cannot be allowed as well.

So, the abstract is that neither IX mode should be allowed nor S mode should be allowed, now as a compromise, what is being done is that, a new mode is being invented new mode is being designed, which is called the SIX mode. That is it wants to say that this is both X, this is both IX and S. So, SIX mode as you can see the name this is S and IX both; that means, that I want that this transaction is saying that, this is being read completely.

So, that is why, which is S, but something inside it also be written that is why it is IX. So, that is why, this is a SIX mode. So, this SIX mode this SIX lock is essentially a compromise from IX and S and this is safer SIX mode is much safer than either as S mode or an IX mode.

(Refer Slide Time: 19:33)



$T_1 \rightarrow$ read a_{12}

$T_2 \rightarrow$ write a_{14}

$T_3 \rightarrow$ read f_1

$T_4 \rightarrow$ read d

$T_1, T_2 \rightarrow$ concurrently

$T_1, T_3, T_4 \rightarrow$ concurrently

$T_2, T_3 \rightarrow$ not concurrently

So, this is the thing, let us now do some example. So, suppose T 1 wants to read a 12. So, it simply wants to read a 12. So, then what is the thing it applies, it applies this in S mode, then this IS mode, this is IS mode and this is IS mode simply that is fine done that is the first example. The second example suppose T 2 wants to write a 14, so what it should be doing is that, it is going to lock this in X mode, then this is IX mode, this is IX mode and this is IX mode.

Then, suppose the third example is that there is a transaction T 3, which wants to read f 1, so that is one it is doing it wants to read f 1. So, what it does is that, it locks f 1 in S mode and then, r 1 in IS mode and d in IS mode. So, that is again very simple and the fourth example is that T 4 wants to read the entire database, so read d. So, the only thing that it does is that, it just locks d in the S mode and that is what it is being done.

So, these are all the four locks, then doing go going ahead, let us do a little bit analysis of which examples, I mean of all these transactions, which can be going through concurrently. So, now, the thing is T 1, T 2 can if they execute concurrently T 1 and T 2. So, T 1 is trying to read a 12 and T 2 is trying to write a 14. So, can they go ahead concurrently, yes they can, because there is no problem, because this is an S and X in a different things and this is IS and IX. If we go back to our compatibility matrix IX and IS is compatible.

So, there is no problem with this. So, T 1 and T 2 can be done concurrently. So, there is no problem with that. Then, let us think of some other kind of operations. So, let us say let us now argue about T 1 may be T 3 and T 4, what happen what can be done about them. So, T 1 is this is all in S IS mode, T 3 is this is in S mode and these are in IS mode and this is S. So, can they go ahead, so we have to check is versus S which is S. So, there is no problem. So, this can also go through concurrently.

So, there is no issue with them, because these are either in IS and S modes and they do not conflict, they are compatible with each other. So, this can go ahead concurrently. Now, let us say T 2 and T 3, we will just going to do some examples T 2 and T 3. So, T 2 this is in S IS, IX mode and T 3 is S is IX. So, now, can they go ahead concurrently, we need to check and S with IX is no; that means, that this cannot go through. So, there is a conflict here S with IX cannot be done.

So, this cannot be read one transaction cannot be reading f completely, because something inside it is being executed. So, this cannot go ahead concurrently this cannot go ahead concurrently this is not compatible T 2 and T 3 are not compatible. And let us just do one more example, which is T 2 and T 4, can they go ahead concurrently, the thing is that T 2 is IX. So, this is the same problem S with IX.

So, again they cannot go ahead concurrently that is it can be understood from the… So, if one remembers the lock compatibility matrix, then this can be done. Even, otherwise one does not remember to really memorize the compatibility matrix, the compatibility matrix is quite intuitive and if one draws this diagram, it is probably easier to understand why S, X is not compatible with IX.

Because, as you see in this example, if somebody is reading f 1; that means, it is trying to read everything, including a 14; that is why it is getting a lock S on this, while some other transaction is writing on this. So, which is of course, got a read write conflict and that cannot be allowed. So, that is why these things can be done.