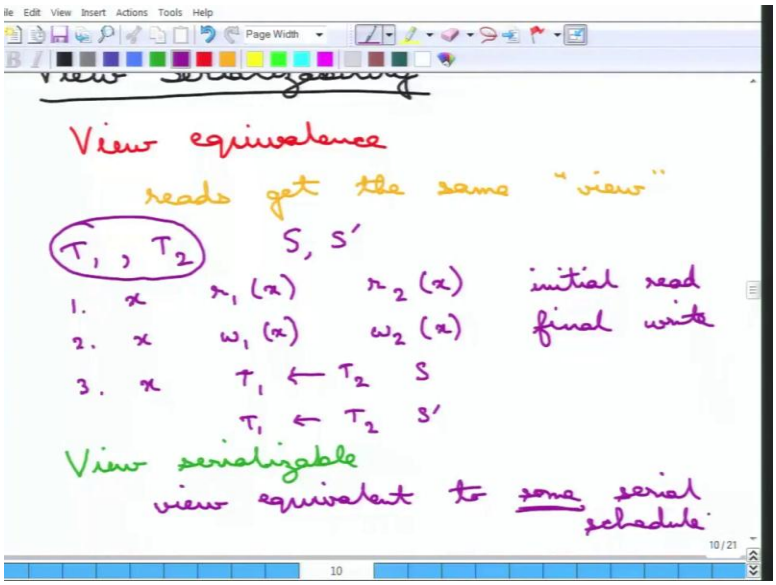
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 35
Schedules: View Serializability

We will continue with the notion of serializability, so after conflict serializability we will start on the View Serializability.

(Refer Slide Time: 00:19)



So, this is view serializability, so just like conflict serializability require the notion of conflict equivalence. So, we will require the notion of view equivalence, so two schedules whether they are view equivalent to each other or not that is the notion of view equivalence. So, essentially very roughly the notion of view equivalence is that if they reads and then get the same view that is they reads that same thing that is produced by the writes, so reads get the same view.

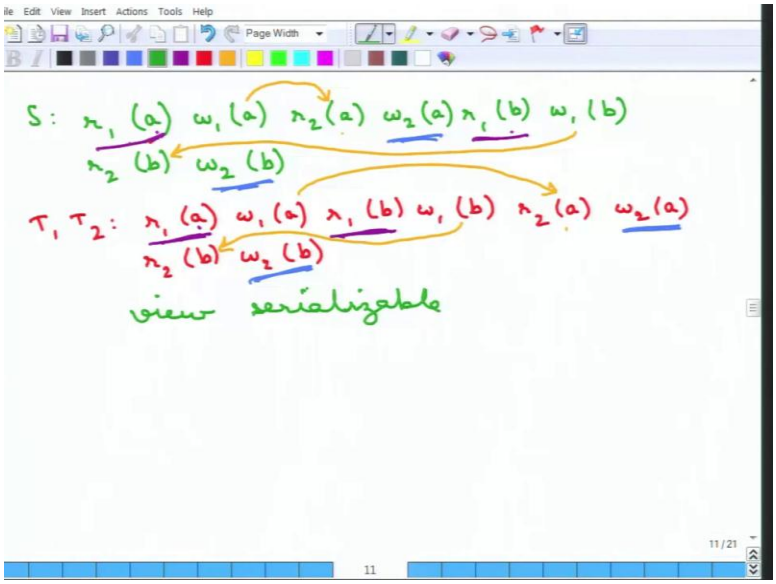
So, we will define the little bit more formally, where this view means essentially is the following. So, two transactions T 1 and T 2 are view equivalent if the following thing happens. So, for each data item x if there is a data item x transaction T 1 reads x and transaction T 2 reads the x, they read the same value of x. So, they read the same initial value, so this is the defining the initial value. So, the same initial value of x is being read by both the transactions that is number 1.

Number 2, if there is the data item x and the final writes the return values that x is return to by transaction 1 and the final write by transaction 2, the final writes are also the same. So, the initial reads are same and the final writes are same and the third one is essentially what to we have been studying for the conflict serializability etcetera is that for the data item x, if T 1 reads the value that is produced by T 2, then that should not be changed in the two schedules.

So, if there are two schedules that involve this T 1 and T 2 then there may not be changed. So, that is the notion of view serializability just to go once more, so there are two schedules S and S dashed that involve these two transactions is just for our sack of understanding, in generally it can include too many number of transactions. But, if both of them starts from the same transactions that reads the value of x then of course, they get the same initial value that is read, then that is fine.

Otherwise, even if the two transactions read the two different things, then they must have the same initial read and the same final write, these are the two schedules should have the same final write and then the transaction ((Refer Time: 03:00)) every x if T 1 reads the values that is produced by T 2 in transaction S the same must happen in transaction S dashed as well. This is view equivalent and the schedule is said to be view equivalent if the following three conditions happened and if it is called view serializable, if it is view equivalent to some serial schedule. So, that is the thing if it is view equivalent to some serial schedule that is the same like the conflict equivalent definitions. So, to some serial schedule that is the whole point.

(Refer Slide Time: 03:45)



So, coming to the example, so suppose this is the example r 1 a then w 1 a then r 2 a, w 2 a r 1 b, w 1 b then r 2 b and w 2 b. So, the question we need to test whether this is view serializable or not. So, let us test it with T 1, T 2 just like what we have been doing earlier. So, what is T 1, T 2 we will write now all the operations of T 1 first which is r 1 a, w 1 a then r 1 b, w 1 b and then that of T 2, so then this is r 2 a, w 2 a, r 2 b and w 2 b. So, now, we need to see if they follow those three conditions.

So, first of all they read the same value of a, so for a this is r 1 is reading it and r 1 is reading it here, so that is the same. So, if you going that to the definition, if this happens that transaction 1 read this one says and transaction 2 reads then this is not following the view equivalence, neither is these following the view equivalence. So, the same transaction, if the transaction 1 reads the value then it must be the transaction 1 reads that here, if the transaction 1 writes the final value then it must be that the same transaction writes, so that must happen.

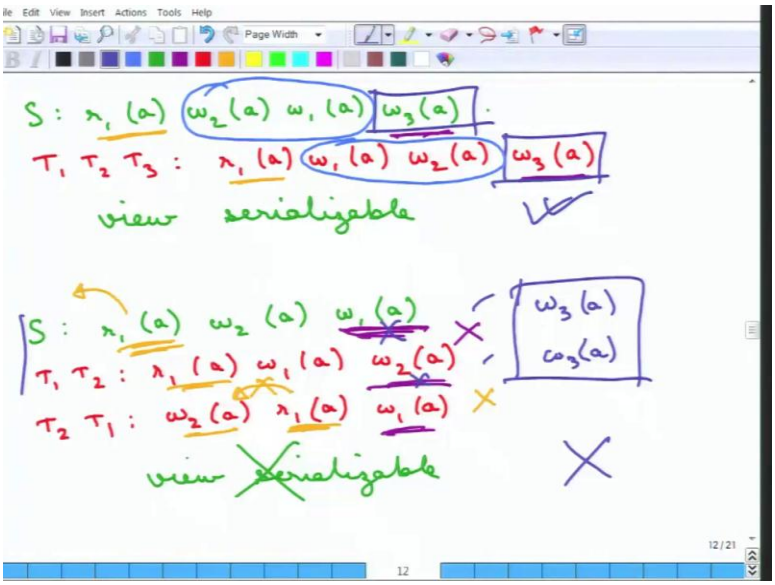
So, r 1 a and r 1 a that follows that is not a problem, now we have to test it for each data items. So; that means, this way only tested it for data item a, we need to also test it for data item b, now data item b is first read by r 1. So, and then it is also first read by r 1, so that is also fine. So, the initial read conditions are correct, now let us test for the final write conditions. So, the final write conditions of a is by w 2 a if this s and in this serial

schedule T 1, T 2 it is again by w 2 a. So, that is correct same for b it is w 2 b and w 2 b, so that is also correct.

Now, the third condition is essentially saying that every read that is produce by another write must be consistence, so must be the same. So, what are the read's have to produce etcetera, so let us see. So, transaction 2 reads the value a that is produced by transaction once write, so that condition must be present here. So, if r 2 is read that must be produced by transaction 1. So, if r 2 reads then the same a that must be produced, so that is also valid here.

And similarly r 2 b is following, what is return by w 1 b here and same as s, so r 2 b will follow what is return by w 1 b. So, then this is view equivalent to this transaction s 1, s 2 so; that means, this is e is view serializable, because this is view equivalent to this transactions T 1, T 2. So, that is the first example.

(Refer Slide Time: 07:07)



Now, let us take another example which is the following, this is r 1 a then w 2 a then there is a w 1 a then w 3 a. So, if you remember this must not conflict serializable, but this previous example that we did also conflict serializable. Now, let us test it with T 1, T 2, T 3. So, what is T 1, T 2, T 3? This is r 1 a then w 1 a then w 2 a then w 2 a. So, the initial read conditions by transaction 1 here and transaction 1 here, so that is fine.

So, final writes r by w 3 a here and w 3 a here, so that that is also correct and then there are no more reads by either the schedule is or by the serial schedule T 1, T 2, T 3. So, the third condition is also honoured trivial, so this is actually view serializable, this is very, very important to note is that. So, this is view serializable although if you remember the example of conflict serializability this was not conflict serializable, because we swap these two, so this conflict we swap.

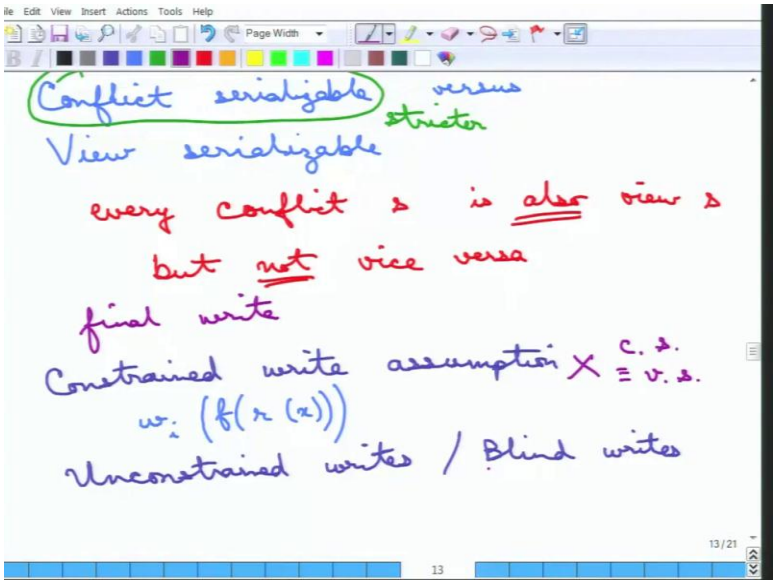
But, here it does not matter, so why does not matter, so here is something that why view serializability is useful. So, why it is sometimes more useful than conflict serializable is that look at the middle two conditions. So, w 2 a and then w 1 a and versus w 1 a and then w 2 a all though none of the writes actually matter. Why does it not matter? Because, both of them it actually super seated by a third write by transactions 3 w 3 a.

So, the condition for view equivalent essentially say that the final write is what wants to be bothered about and the final writes are both produced by transaction 3. So, they are all correct, so there is nothing problem here, but just we will come to this point even little while again, but before that I just want to show you another example just to high light what is happening on here. So, suppose this is another schedule which is simply this, now let us test whether this is view serializable or not, this is r 1 a, w 1 a, w 2 a.

Now, first of all the read conditions are all correct the writes; however, are not correct, this is w 1 a and this is w 2 a. So, this is not correct, so this s is not view equivalent to T 1, T 2. Now, let us test it with T 2, T 1, T 2, T 1 is your w 2 a then r 1 a, w 1 a, now here although the final writes are the same as it happens that the initial read is also the same. But, what it happens is that this is not view serializable, because this read has been produce by some other write; however, this read is being produced by this w 2 a. So, this is not allowed, so over all this is not view serializable.

Now, so that critical different between these two examples is that following is this w 3 a here recover the w 3 a between these two things, there is a final write here, there was a final write which essentially elapse to super seed both this writes. So, this final write condition was by some other transaction which remains the same for both the schedules. So, that is why this was view serializable why this was not, so that is the very important part of what to understand what view serializability is.

(Refer Slide Time: 11:03)



And now to understand it in more detail ((Refer Time: 11:04)) conflict serializability. So, what we are going to study is next is conflict serializable, let us say conflict serializability schedule verses view serializable schedule. So, as we solve the two examples, the first example was that it was conflict serializable and view serializable both, the second was not conflict serializable, but was view serializable and the third one was neither conflict serializable nor view serializable.

So, this is not by accident as it happens that we can we have this following rule is that every conflict serializable schedule is also view serializable I am just abbreviating serializable by s. Now, this is true, so if something is conflict serializable it must be view serializable and you can actually prove it formally, but if you thing for a little bit and look at the intuition of it, you will see that this is correct. However, of course, we saw an example that the wise versa is not true.

So; that means, that not every view serializable schedule is conflict serializable and we already saw an example. So, what it actually means is that conflict serializability is that stricter condition than view serializability, so conflict serializable if actually more strict, so this is stricter than view serializability. Now, we can see what is the usefulness of view serializability is that they are all certain schedules which are not conflict serializable.

Because, conflict serializability is stricter condition, but they are view serializable, so if one enforces the conflict serializability those schedules are not going to be allowed by the database engine. However, if one enforces only the view serializability they can be allowed by the database engine and in the end it does not matter, because the final writes are the same. So, initial leads etcetera are of course, the same the reads as produce by writes are also the same, but very, very importantly the final writes are the same.

So, essentially it all boils down to this final write, so the final write is the, with important condition that everything based onto. Now, there is an interesting concept called the constrained write assumption, which is related to this constrained write assumption. So, the constrained write assumption, essentially says that every write is constrained by the value that it has rate. So, write if there is a write by a transaction i, it must depend on what the value has been rate.

So, every transaction s, so they it must be a function of whatever it has read so; that means, that the write before or transaction tries to write it must read it. So, that because the write is the function of the read. So, the other way round is that, so if this is not failure then we get what we call unconstrained writes. So; that means, a transaction simply writes without reading unconstrained writes are also sometimes called blind writes.

Why it is called a blind write? Because, it is blindly write it does not care about what it has rate it etcetera, it does not need to be write read, so just writes. So, this unconstrained writes are blind writes. So, if constrained write assumptions are not allowed then... So, let me just check it if these are not allowed then conflict serializability is equivalent to view serializability, so this is sometime needed to be understood.

So, the reason for this is that constrain write assumption look at the example essentially the difference between the view serializability and conflict serializability is that view serializability can allow certain write to write mismatches as we saw in the example between w 1 a and w 2 a. Because, there is a final write which does not depend on anyone of them and just simply writes, so that is the blind write. Now, if one removes the blind writes one can show that actually you can been shown this conflict serializability is equivalent to view serializability.

Now, to turn it around; that means, that suppose there is a schedule which is view serializable, but not conflict serializable. So, then it must happen that schedule has blind writes. Because, so the only difference between view serializable and conflict serializable to allow the schedules as view serializable and not conflict serializable is those blind writes. So, view serializable schedules, which are not conflict serializable must have blind writes are this unconstrained writes, so that is the thing to note.

So, this unconstrained writes and blind writes is very important, so if there is... So, under this constrains write assumption, if there is only constrain writes if no blind writes are allowed then conflict serializability is equal to view serializability. So, that is about the two serializable notions of conflict and view serializability.