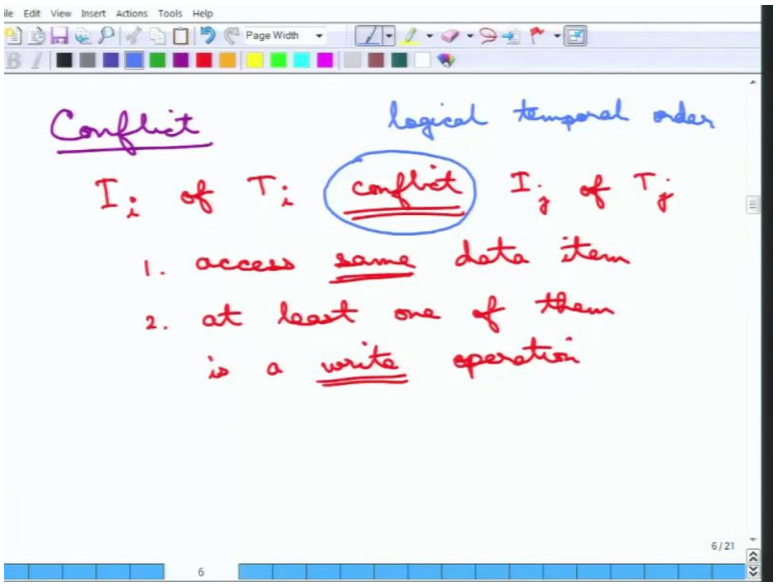
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 34
Schedules: Conflict Serializability

(Refer Slide Time: 00:09)



Our conflict so an operation, so an instruction I of transaction T i is set to conflict with another instruction I j of another transaction T j. If the following thing happens is that, they both access the same data item, this is very, very important, same data item and at least one of them is a write operation. So, now, one can intuitively see, why are they set to be conflicting, if you remember, there are this r a w, read after write, then write after write and write after read conflict.

Now, what is the problem, why are they set to be conflicting, this instruction I i and instruction I j of two different transactions, there must be of different transactions, number 1. Then, there must be accessing the same data item and one of them must be right. Now, what is why is this conflict is very simply the following thing, either suppose I i should have taken place before I j.

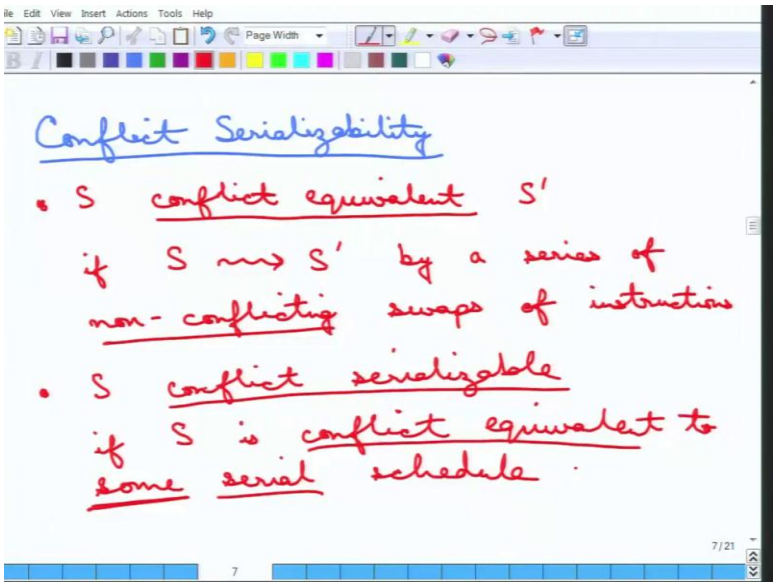
Now, they are conflicting, because in another schedule which is equivalent, this order of I i and I j must be preserved, it cannot happen that in the another schedule, I j is before I i and then, it is not equivalent. Why it is not equivalent? Because, you see that this read

write conflicts write, write or write read conflict will happen. So, that is why, they have set to be conflicting.

This is the essentially the definition of why they are set to be conflicting. So, intuitively these are conflict is this conflict, this enforces a logical temporal order on this instruction. This is the logical temporal order; they cannot be violated; that logical temporal order cannot be violated. And now, that is one thing, now on the other hand, the other side of it is that, if there are two instructions, which do not conflict.

Then, they can on inter change and they can be placed anyone before any other one and then, that is not a problem. So, that is the other side of it and that is what is we are going to study in much more detail.

(Refer Slide Time: 02:49)



So, conflict serializability, so this is the conflict and we have studied serializability. So, the next thing what we will do is we will study, what is called the conflict serializability. So, first of all a schedule is conflict equivalent is called conflict equivalent to another schedule is dashed, if it can be transformed to S dash by a series of non-conflicting swaps of instructions.

So, non-conflicting swaps of instructions. So, what does it mean? So, suppose either schedule S and then, there are some instructions we need of course and then, let us pick any two instructions within this. And if that do not conflict and they cannot be inter

change. So, let us inter change it and let us keep on doing this and suppose of by the process of this, one gets to S dashed from it is S.

Then, S and S dashed are conflict equivalent, because what we are essentially done is that, they have ensured that if two instructions do not conflict, they have been swap. So, on the other hand which also means that, if two instructions are conflict, their logical order has been maintained, so these two schedules are equivalent in the sense of the conflicting instructions, the conflicting instructions maintain the same order, so that is why, they are called conflict equivalent.

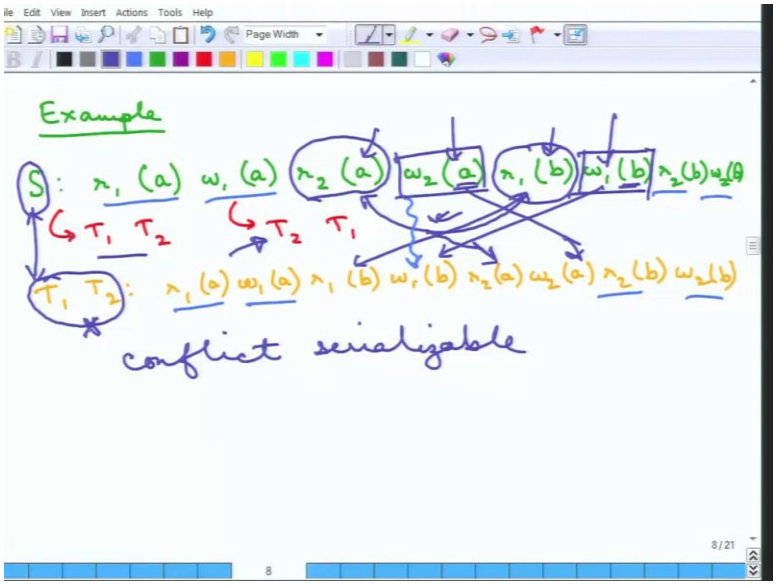
So, that is the definition of conflict equivalent. Now, this is the first definition, now S is set to be conflict serializable, this is conflict equivalent, conflict serializable. This is the definition again, if S is conflict equivalent to some serial schedule, so once more S is conflict serialable, it is given to some serial schedule.

So, it does not need to be conflict equivalent to all the serial schedule, but they must exist at least one serial schedule that, if is conflict equivalent to and then, it is called conflict serializable; that is the definition of conflict serializability. Now, why are we studying it, because if S, if a schedule S is a conflict serializable that is means that, this it is equivalent some serial schedule.

That means, it will preserve the database consistency, because we know that if the transactions are, if the transactions take place one after another serially, then the preserve the database consistency. Because, each transaction by itself preserve the database consistency, so if transaction T 1 happens, then it preserve the database consistency and then, T 2 happens, then it is also preserve the transaction, then it also preserve the database consistency.

So, if there is an another schedule, which even though is interlay T 1 and T 2 is conflict equivalent to it, then it also preserves the database consistency.

(Refer Slide Time: 06:17)



So, now let us see an example, so suppose S is the following, so we will is the same notation and we will even wrote the semicolon. So, the first question is this conflict serializable, so that is what we will trying to answered. So, that means, to say the whether this is conflict serializable or not, it must be equivalent to either. So, there only two transaction T 1 and T 2, so it must conflict equivalent to either T 1, T 2, or to T 2, T 1.

Now, let us see, whether it is conflict equivalent to T 1 or T 2. So, first of all T 1 or T 2 is the schedule. So, what is the T 1, T 2, you will write down all the instructions of T 1 first in the order that they appear that cannot be violated, so r 1 b, then w 1 b and then, all the instructions of T 2. Now, we will see that, if we can arrive at this from S through series of non-conflicting swaps.

So, now, what is the first thing? So, first thing is that, let us you have to do it. So, this is fine, there is no problem with this to, similarly these two have no problem, because they have to been starting and they have to been ending So, now what has been swapped is essentially and consider that r to A and w r 1, b has been swapped, because this r 1 b has come here and r 2 a has come here.

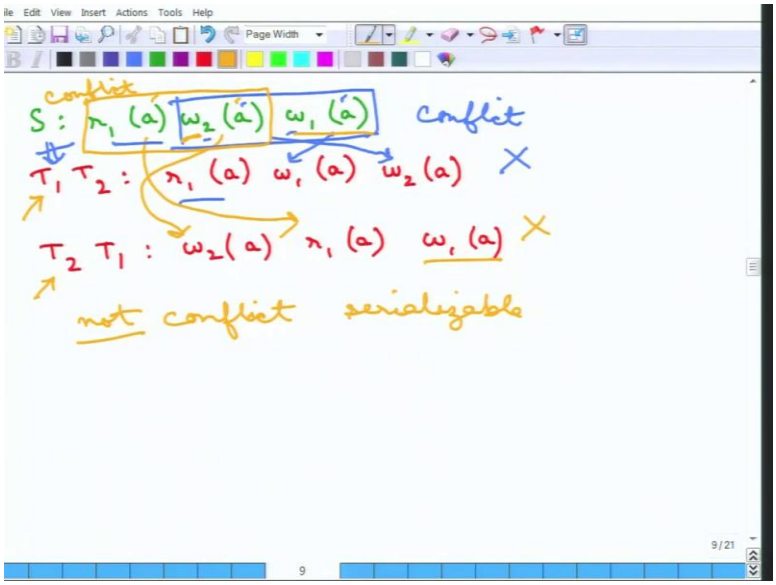
So, are these two instructions non-conflicting, yes, because they do not operate on this same data item, very simply, so these are non-conflicting and these swap is fine. And

how about this 2, then this two have also been swapped w 2, a has been swapped with w 1 b, because w 1 b is coming here and t 2 a is coming here. Are these two non-conflicting again it is yes, because these two separate data item.

So, that is means that S and T 1 and T 2 are equivalent; that means S is conflict serializable, so the answer to this is S is conflict serializable. So, there is no problem with this as for as database consistency is concerned. Now, known that we only need to test with T 1 and T 2 and not T 2 and T 1, because, it has to be conflict equivalent to one of them that is it.

So, since it is already processed in the first case it is fine, it need not be check with T 2 and T 1. If this one is not equivalent to T 1 and T 2, then it need to checked with T 2 and T 1, only when T 1 and T 2 did not pass that is the thing.

(Refer Slide Time: 09:32)



Now, let us take another example, which is the following, this is simply r 1 a w 2 a and w 1 a. This is a very short schedule and let us see, whether this is equivalent. So, what is T 1and T 2 is r 1 a, w 1 a, w 2 a. Now, let us test whether S is equivalent T 1and T 2, this is fine, but these two has been swap. So, these two these as been swapped is this a non-conflicting swap, can we do that, let us test it, they operate on the same data item and one of them the right.

So, that means, these two conflict, these two instructions conflict, so this swap is not a good swap so; that means, this is not equivalent to T 1 and T 2. But, we still cannot conclude anything about S; we need to test it with T 2 and T 1 as well. So, what is T2 and T 1, T 1 is simply w 2 a, then r 1 a and then w 1 a, let us see, whether this is equivalent to S.

So, once more w 1 aits a last operation, so nothing to worried about, but r 1 a and w 2 a has been swap. So, let us see if that swap can be happening now r 1e a and w 1 a are on the same data item and one of them is a right. So, this also conflicts. So, this is not a non-conflicting swap. So, this is also not equivalent. So, S is neither equivalent T 1and T 2 nor it is equivalent T 2 and T 1; that means, it is not conflict serializable.

So; that means, S will not preserve the database consistency in some manner. So, in the sense of conflict serializability, so S is not conflict serializable that is the concept of conflict serializability.