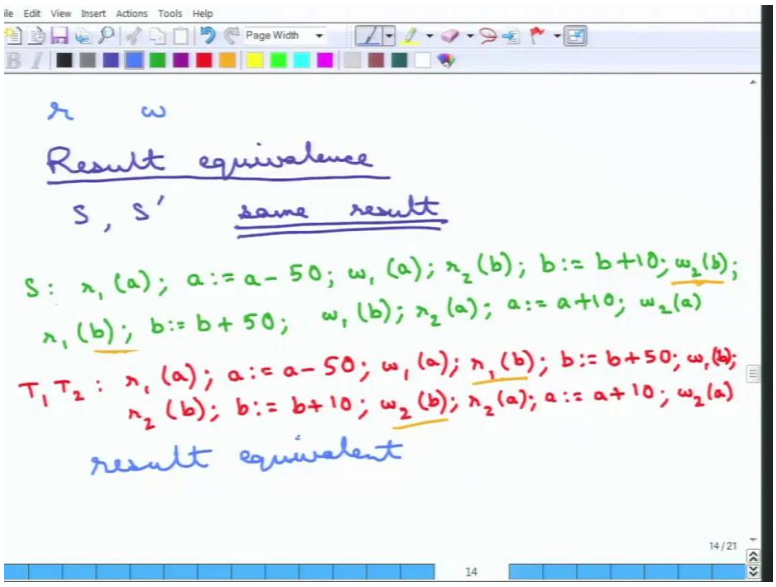
Fundamentals of Database Systems
Prof. Arnab Bhattacharya
Department of Computer Science and Engineering
Indian Institute of Technology, Kanpur

Lecture - 36
Schedules: Result Equivalence and Testing for Serializability

(Refer Slide Time: 00:15)



Now, one interesting thing or one important thing that you must have notice is that these serializability notions are just concerned about the write and read, but not the values that has been read or write. I mean of course, there is a constant they must be reading the same value excreta, but they do not analyse the result part say. So, now, extended there is another concept which is called result equivalence, so just like there is conflict equivalence and view equivalence there is something called a result equivalence.

Now, two schedules are set to be result equivalence to each other S and S dash are result equivalence to each other. If they produce the same result, if they somehow produce the same result, then they are set to be result equivalence of course, then it means that they are correct from the databases point of view if all the transactions in those schedules are success. Now, here is an example to highlight the point, is that suppose there is this schedule S, which says read one of a, then it actually does this following, now we need to worry about the result.

So, we need to know the values of a s and whatever things that we are writing. So, then it writes, then it reads b, then it does b is equal to b plus set 10, and then writes to b fine. And the other one is the following it is reads b, then it does b is equal to b plus 50, then it is w 1 b r 2 of a, then a is equal to a plus 10. And then, finally, so just note this is all just one schedule, so this is one serial S and this is another schedule T 1, T 2, which is the serial schedule, which if you write down this operation in the following manner, then it produces this thing.

So, we will write down all the operations of a first, so first all the operations of transaction one and then those of transaction two. Now, look at these two transactions and let us see whether they are conflict equivalent or not. Clearly, they are not conflict equivalence, because r 2 and w 1 b etcetera must happen in the correct order, so they are not conflict equivalent just to highlight, because this w 2 b and r 1 b must be in this order.

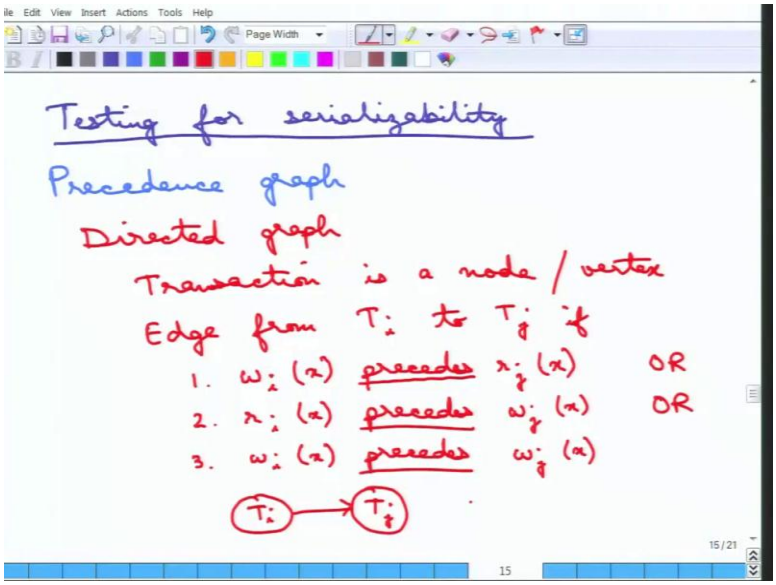
So, w 2 b and r 1 b conflict and here it is swapped the other way round. So, this is not conflict equivalent, this is also not view equivalent, because this r 1 b is reading the value that is produced by w 2 a, while as this r 1 b is reading the value something else. So, these are neither conflict equivalent nor view equivalent; however, very, very importantly these are result equivalent and that can be seen just by understanding the semantic of this two schedule.

So, essentially that point is this that S should be allowed by the database, because it finally, produces the same result as the serial schedule T1, T2, but there is very, very hard for the database engine to figure it out and unless it does semantic analysis. So, semantic is the meaning of the transactions, it is not just blindly whether it is read or write and who reads from, which other transaction reads from which other values etcetera.

But, actually analysing the value, that if you do a minus 50 first and then replace 10 that is the same as doing a plus 10 first, and then a minus 50 or if you do b plus 50 first and b minus this thing first, then it does not matter. So, then that records as semantic analysis and which is of course, not done by the database, because it can be very, very hard as you see. So, although you can see that theoretical there is a notion of result equivalence, database in use do not do that.

So, it is essentially just a theoretical notion of equivalence and the database only go, only try to go up to conflict serializability and view serializability. Now having said all of these things, let us see that how does the database find out or whether, what is the algorithm to find out whether a schedule is serializable or not. I mean, we have been doing with paper and pen etcetera.

(Refer Slide Time: 05:17)



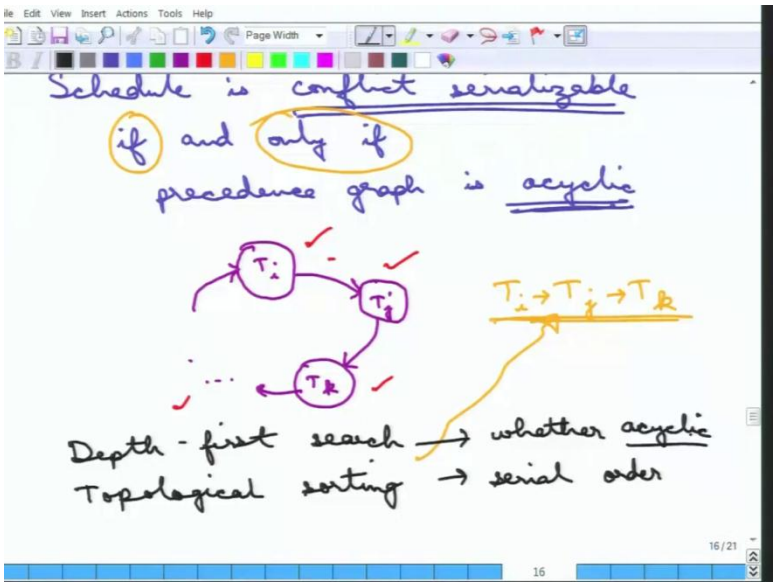
So, essentially what we are trying to say is testing for serializability little. So, whether there can be an automated algorithm for doing this or what can we done about it, testing for serializability requires the following notion. So, what is done is first of all this is something called a precedence graph is created, so this is notion of precedence graph. Now, I am assuming all of you are familiar with the graph data structure, essentially graph has certain nodes and there are edges that connect the nodes, which mean certain things. Now, we will see what the thing is that, so this is the directed graph.

So, here in this thing, so this is a directed graph and each transaction, so every transaction in their schedule is a node, transaction is a node or vertex whatever way you want to say it is the same thing. And then, there is an edge from transaction T i to transaction T j note that edges are directed, because it is a directed graph. If the, this thing happen if any of that following three condition happen is that the w i x precedes r j x.

So, what are we trying to say here is that this is the important condition precedes. So, precedes meaning w i x must be happening before r j x this or r i x precedes w j x or, now you can guess the third condition. It is w i x precedes w j x, essentially it just encodes the three conflicts the read after write, write after read and write after write. So, essentially, so if there are two, so between transaction T i and T j if there are two instructions that can conflict, so essentially write read, read write and write, write.

And if transaction if the operation of transaction i happens before that of j, then transaction i has a directed edge to transaction j that is how you create the precedence graph. And now it should be clear why it is called a precedence graph. Now, given this precedence graph, so first of all given a schedule one can create the precedence graph.

(Refer Slide Time: 08:17)



Now, given this precedence graph a schedule is conflict serializable, so it is a particular schedule is conflict serializable if an only if, so this is if and only if this is important it not just if, so this is both and necessary and sufficient condition. If and only if the precedence graph is acyclic, so which has got no cycles, so this is conflict serializable if and only if the precedence graph is acyclic. So; that means, if there is cycle in the precedence graph, so essentially let us trying to understand, what does the cycle means.

So, there is T i that has an edge to T j, then that has an edge to some other thing let us take T k excreta and that has an edge to something else dot, dot, dot finally, there is an edge back to T i. Now, what does this edge T i to T j mean T i to T j essentially mean is

that T i and T j conflict and the order of T i the instruction of T i must happen before that T j. Then, the next edge essentially means that the, the instruction of T j some instruction of T j must happen before T k.

And similarly happens that the instructions of T k must happen before T i; that means, that you can think of any serial order, let us a T i T j T k. Now, that is going to violate the last condition the T k not happening before T i. Now, let us choose the other way round suppose its T k T i T j, then T k to T I is fine T i to T j is fine, but then T j should be happening before T k.

So, essentially if there is a cycle, then this essentially saying that T i must be before T j it must be before T k it must be before something else must be before T i, which have never happen which kept; that means, there is no serial schedule possible, which means that if there is a cycle it is not conflict serializable.

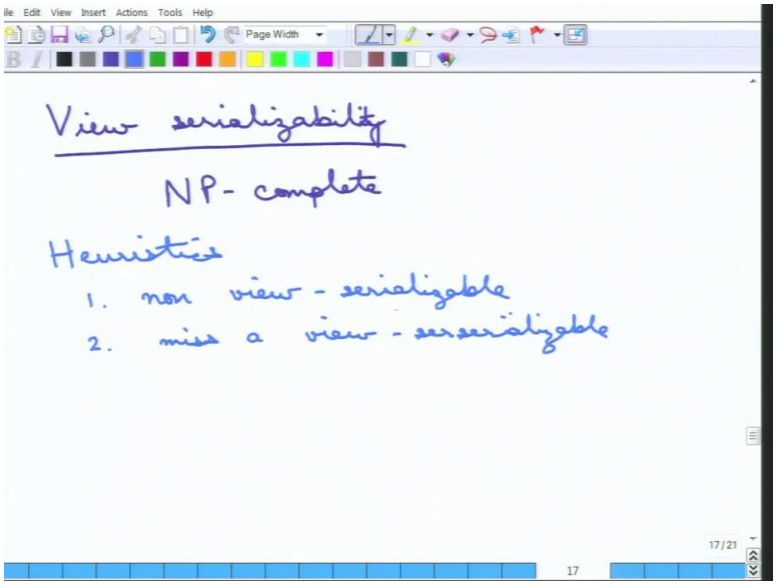
And the other way round can also shown is that if it is conflict serializable, then if as schedule is conflict serializable; that means, that it is equivalent to some order T i T j T k excreta it is conflict serializable it is equal to some serial schedule; that means, the only the edges that can be present it from T i to T j and T j to T k they cannot be any package from T k T i; otherwise it not equivalent; that means, they cannot be any cycle.

So, that is why this both if and only if, so this both necessary and sufficient condition. Now, the question then become is the given a schedule one can produce precedence graph and if the graph contain cycles, then it is not conflict serializable otherwise it is. So, the essentially the question now goes in to how do you find if a directed graph the precedence graph contains cycles or not.

Now, for that there is an algorithm, which is called a depth first search if you start the depth first search from vertex if there is a cycle it can find it out. So, assuming there is no cycle if one runs another algorithm, which is called a topological sorting the topological sorting will essentially if a serial order, so topological sorting will actually produce as serial order of the transaction. So, that means, topological order may actually produce something like T i T j T k so; that means, this is equivalent to T i T j T k and the depth first search can find out whether there are acyclic or not, so whether the graph is acyclic or not.

So, these are the two algorithms and if you are familiar with graph algorithm, then this is they are otherwise look after just move this step.

(Refer Slide Time: 12:07)



So, this is about conflict serializability, now testing for view serializability is a much, much harder problem and actually there is a complexity clause of problem called the N P complete serializability. So, testing for view serializability is actually N P complete problem N P complete problems are essentially hard problem. So, suppose to hard is not clear whether they hard or not nobody else found a good efficient algorithm for them yet, so N P complete problem is a typically sort of as hard problem.

So, testing for view serializability is a hard problem and the database is really do not, because it can take a long, long time to the actually solve this N P complete problem. So, the database engines generally do not try to test for view serializability in the exact notions. So, what it will try to do is again it resorts to the concept of heuristics, so what is does is that this is some practical heuristics it is try to catch all non view serializable schedules.

So, it can quickly catch if a schedule is non view serializable, but then of course; that means, that it can miss a actually view serializable thing. So, it catches all non view serializable thing, but can miss a view serializable. So, what is the effect of what I am

trying to saying is that given are schedule if it is non view serializable there is a practical algorithm some heuristic will actually say it is non view serializable.

Bu, it may also happen that it declares of view serializable schedule to be non-view serializable as well that is the mistake that it does. Because, the correct precedence N P complete and that will take a long, long time to finish, so the practical way of doing it is that doing a mistaken only one side. So, it may mean a view serializable schedule, but if a schedule is non view serializable it will surely catch it.