Paper Name: Database Management Systems

Topic: Checking for Serializability

Paper Code: BCA CC410

By Ms. Manisha Prasad

Head, Assistant Professor,

Department of Computer Science

Patna Women's College

How to check serializability of a schedule

In real time applications we prefer Concurrency of transactions as they lead to faster response time, less waiting time and better efficiency. And when we talk about concurrency, it means execution of Non serial schedules. But we also understand that Non serial schedules sometimes lead to inconsistency in the databases.

Before learning to check the serializability of a Non serial schedule, some important points to recapitulate :-

1. A transaction is a **set of instructions** performing a logical unit of work.
2. The instructions of the transactions execute in a particular order to accomplish the task.
3. A set of **one or more than one transaction** executing in a particular order is known as a Schedule.
4. If the order of execution of Transactions within the schedule is one by one i.e. when one transaction finishes then next transaction starts, it is known as Serial schedule. However if there is interleaving of execution of multiple transactions i.e. there will be context switching of instruction execution between multiple transactions of the schedule, it is known as Non serial schedule.
5. A Serial schedule always ensures consistency of the system but it leads to Poor performance of the system as Response Time will be slow.
6. A Non serial schedule may or may not ensure the consistency of the system but leads to high efficiency of the system .

The aim of this discussion is to somehow deduce the mechanism for ensuring Concurrency along with Consistency

So the idea is if we can convert a non serial schedule to its equivalent Serial schedule, we can say that consistency will be maintained because Serial schedules always ensure consistency.

Let us take an example of a Non serial Schedule S1.

This schedule consists of two transactions T1 and T2, executing in an interleaved manner. First T1 begins, which reads the value of data item A and then writes it. Then execution T2 begins. This transaction also reads the value of A and writes it. Again a context switch occurs and transaction T1 resumes which now reads the value of data item B and writes it. After this again context switch occurs and now transaction T2 resumes which reads and subsequently writes to the value of B.

Schedule S1 :

	T1		T2
1	R(A)		
2	W(A)		
		3	R(A)
		4	W(A)
5	R(B)		
6	W(B)		
		7	R(B)
		8	W(B)

The Serial equivalent of the Schedule S1 can be Schedule S2 or S3 as follows :

Schedule S2 :

T1	T2
R(A)	
W(A)	
R(B)	
W(B)	
	R(A)
	W(A)
	R(B)
	W(B)

OR

Schedule S3 :

T1	T2
	R(A)
	W(A)
	R(B)
	W(B)
R(B)	
W(B)	
R(A)	
W(A)	

How to convert a Non Serial Schedule to a Serial Schedule ?????

To convert a Non serial schedule to a serial schedule, we will have to shift or change the order of execution of some of the instructions of the participating transactions. But the swapping of instructions cannot be done randomly. It has to follow certain rules.

The rules are :-

1. The order of execution two instructions within the same transaction cannot be swapped.
2. The order of execution two instructions of two different transactions can be swapped if they are **Non conflicting** instructions.

So, while swapping the instructions we have to check whether the pair of instructions, which we want to swap, are in conflict with each other. If they are conflicting with each other then we cannot swap those pair of instructions and if they do not conflict with each other then they can be swapped.

Conflicting Instructions

Two operations/instructions are said to be in conflict, if they satisfy **all** the following three conditions:

1. Both the operations should belong to different transactions.
2. Both the operations are working on same data item.
3. At least one of the operation is a write operation.

Examples:-

Example 1: Operation W(A) of transaction T1 and operation R(A) of transaction T2 are conflicting operations, because they satisfy all the three conditions mentioned above. They belong to different transactions, they are working on same data item X, one of the operation in write operation.

Example 2: Similarly Operations W(A) of T1 and W(A) of T2 are conflicting operations.

Example 3: Operations W(A) of T1 and W(B) of T2 are non-conflicting operations because both the write operations are not working on same data item so these operations don't satisfy the second condition**.**

Example 4: Similarly R(A) of T1 and R(A) of T2 are non-conflicting operations because none of them is a write operation

Example 5: Similarly W(A) of T1 and R(A) of T1 are non-conflicting operations because both the operations belong to same transactionT1

Schedule S1 :

	T1		T2
1	R(A)		
2	W(A)		
		3	R(A)
		4	W(A)
5	R(B)		
6	W(B)		
		7	R(B)
		8	W(B)

In the example, if we are allowed to swap the instructions 3 and 4 with instructions 5 and 6, then it will become a serial schedule like S2.

Step 1 : We see that instruction 4 and 5 are non conflicting because they are working on different data item, so they can be swapped

	T1		T2
1	R(A)		
2	W(A)		
		3	R(A)
4	R(B)		
		5	W(A)
6	W(B)		
		7	R(B)
		8	W(B)

Step 2 : We see that instruction 3 and 4 are non conflicting because they are working on different data item, so they can be swapped

	T1		T2
1	R(A)		
2	W(A)		
3	R(B)		
		4	R(A)
		5	W(A)
6	W(B)		
		7	R(B)
		8	W(B)

Step 3 : We see that instruction 5 and 6 are non conflicting because they are writing on different data item, so they can be swapped

	T1		T2
1	R(A)		
2	W(A)		
3	R(B)		
		4	R(A)
5	W(B)		
		6	W(A)
		7	R(B)
		8	W(B)

Step 4 : We see that instruction 4 and 5 are non conflicting because they are working on different data item, so they can be swapped

	T1		T2
1	R(A)		
2	W(A)		
3	R(B)		
4	W(B)		
		5	R(A)
		6	W(A)
		7	R(B)
		8	W(B)

Finally we are able to derive a Serial equivalent schedule of the Non serial schedule S1by swapping of Non conflicting instructions. So we can say that S1 is a serializable schedule.

Let us check whether a schedule is conflict serializable or not. If a schedule is conflict Equivalent to its serial schedule then it is called Conflict Serializable schedule.

Example 1:

Let us consider this schedule:

T1	T2
R(A)	
R(B)	
	R(A)
	R(B)
	W(B)
W(A)	

To convert this schedule into a serial schedule we must have to swap the R(A) operation of transaction T2 with the W(A) operation of transaction T1.

However we cannot swap these two operations because they are conflicting operations, thus we can say that this given schedule is not Conflict Serializable.

Example 2:

Let us take another example:

T1	T2
R(A)	
	R(A)
	R(B)
	W(B)
R(B)	
W(A)	

Let us swap non-conflicting operations:

Step 1 : After swapping R(A) of T1 and R(A) of T2 we get:

T1	T2
	R(A)
R(A)	
	R(B)
	W(B)
R(B)	
W(A)	

Step 2 : After swapping R(A) of T1 and R(B) of T2 we get:

T1	T2
	R(A)
	R(B)
R(A)	
	W(B)
R(B)	
W(A)	

Step 3 : After swapping R(A) of T1 and W(B) of T2 we get:

T1	T2
	R(A)
	R(B)
	W(B)
R(A)	
R(B)	
W(A)	

We finally got a serial schedule after swapping all the non-conflicting operations so we can say that the given schedule is Conflict Serializable.