SCHEDULES:

Schedule is a sequences of instructions that specify the chronological order in which instructions of concurrent transactions are executed

- A schedule for a set of transactions must consist of all instructions of those transactions
- must preserve the order in which the instructions appear in each individual transaction.

Serial Schedule

It is a schedule in which transactions are aligned in such a way that one transaction is executed first. When the first transaction completes its cycle, then the next transaction is executed. Transactions are ordered one after the other. This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

Schedule 1

- Let *T*1 transfer 50 from *A* to *B*, and *T*2 transfer 10% of the balance from *A* to *B.*
- A serial schedule in which *T*1 is followed by *T*2 :

T_1	T_2
read(A) $A := A - 50$ write (A) read(B) $B := B + 50$ write(B)	read(A) $temp := A * 0.1$ $A := A - temp$ write(A) read(B) $B := B + temp$ write(B)

Schedule 2

- A serial schedule where *T2* is followed by *T*1

T_1	T_2
	read(A) $temp := A * 0.1$ $A := A - temp$ write(A) read(B) $B := B + temp$ write(B)
read(A) $A := A - 50$ write(A) read(B) $B := B + 50$ write(B)	

Schedule 3

• Let *T*1 and *T*2 be the transactions defined previously. The following schedule is not a serial schedule, but it is *equivalent* to Schedule 1.

T_1	T_2
read(A) $A := A - 50$ write(A)	
	read(A) $temp := A * 0.1$ $A := A - temp$ write(A)
read(B) $B := B + 50$ write(B)	
	read(B) $B := B + temp$ write(B)

Schedule 4

The following concurrent schedule does not preserve the value of (*A* + *B*).

T_1	T_2
read(A) $A := A - 50$	
	read(A) $temp := A * 0.1$ $A := A - temp$ write(A) read(B)
write(A) read(B) $B := B + 50$ write(B)	
	$B := B + temp$ write(B)

SERIALIZABILITY

- When multiple transactions are running concurrently then there is a possibility that the database may be left in an inconsistent state.
- Serializability is a concept that helps us to check which schedules are serializable. A serializable schedule is the one that always leaves the database in consistent state.

• Serializability is the classical concurrency scheme.

• It ensures that a schedule for executing concurrent transactions is equivalent to one that executes the transactions serially in some order.

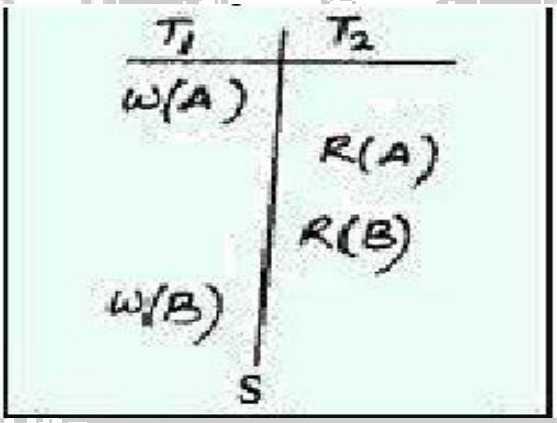
Serializable schedule

If a schedule is equivalent to some serial schedule then that schedule is called Serializable schedule

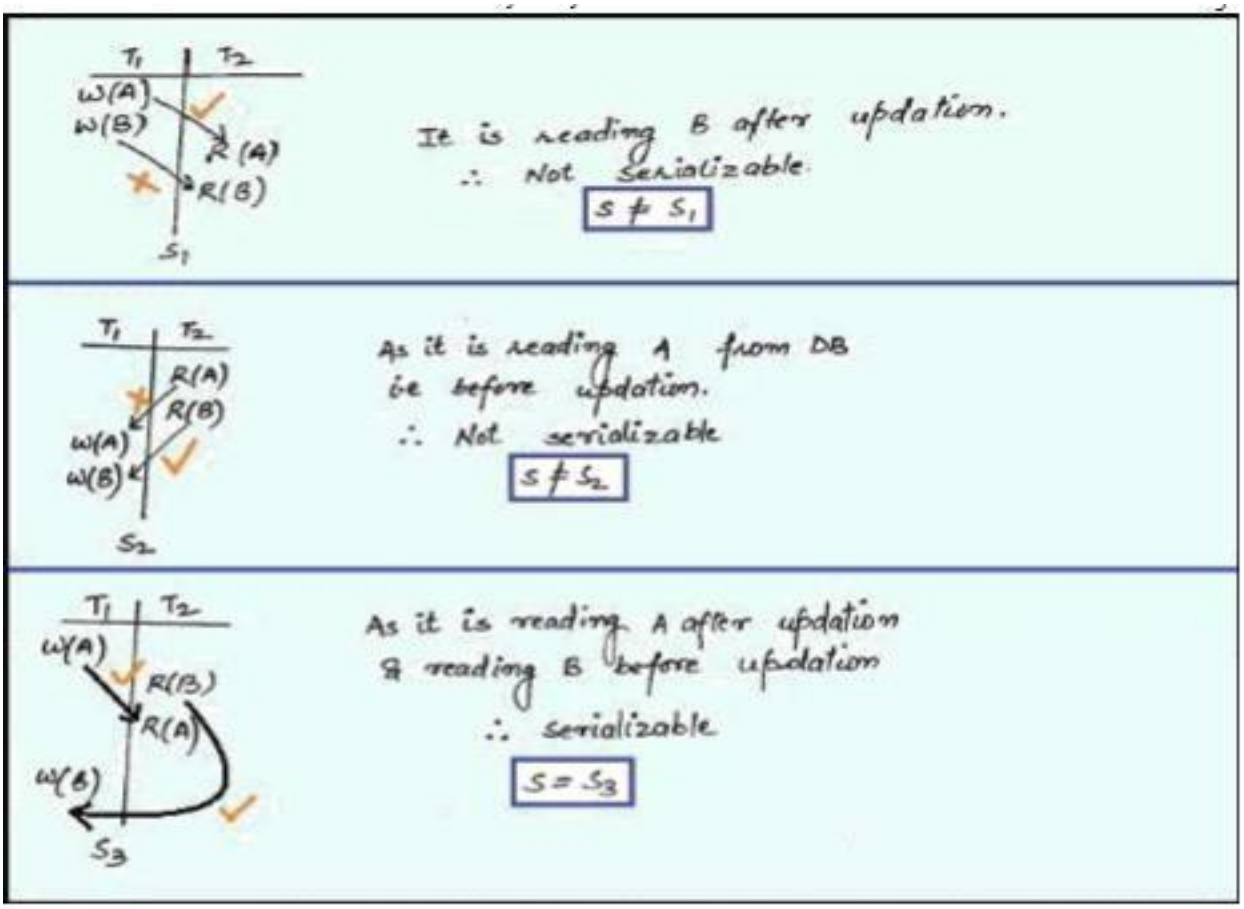
Let us consider a schedule S.

What the schedule S says ?

- Read A after updation.
- Read B before updation.



Let us consider 3 schedules S1, S2, and S3. We have to check whether they are serializable with S or not ?



Types of Serializability

1. Conflict Serializability
2. View Serializability

1. Conflict Serializable

Any given concurrent schedule is said to be Conflict Serializable if and only if it is Conflict equivalent to one of the possible serial schedule.

Two schedules would be conflicting if they have the following properties

- Both belong to separate transactions.
- Both accesses the same data item.
- At least one of them is "write" operation.

Conflicting Instructions :

li and lj of transactions Ti and Tj respectively, conflict if they are operations by different transaction on the same data item, and at least one of these instruction is write operation.

1. li = read(Q), lj = read(Q). li and lj don‘t conflict.

2. li = read(Q), lj = write(Q). They conflict.

3. li = write(Q), lj = read(Q). They conflict

4. li = write(Q), lj = write(Q). They conflict

Two schedules having multiple transactions with conflicting operations are said to be conflict equivalent if and only if

- Both the schedules contain the same set of Transactions.
- The order of conflicting pairs of operation is maintained in both the schedules.
- If a schedule S can be transformed into a schedule S´ by a series of swaps of non-conflicting instructions, we say that S and S´ are conflict equivalent.
- We say that a schedule S is conflict serializable if it is conflict equivalent to a serial schedule
- Schedule 3 can be transformed into Schedule 6, a serial schedule where T2 follows T1, by series of swaps of non-conflicting instructions. Therefore Schedule 3 is conflict serializable.

Schedule 3

T_1	T_2
read(A)	
write(A)	
	read(A)
	write(A)
read(B)	
write(B)	
	read(B)
	write(B)

Schedule 6

T_1	T_2
read(A)	
write(A)	
read(B)	
write(B)	
	read(A)
	write(A)
	read(B)
	write(B)

2. View Serializable

Any given concurrent schedule is said to be View Serializable if and only if it is View equivalent to one of the possible serial schedule.

Let S and S´ be two schedules with the same set of transactions. S and S´ are view equivalent if the following three conditions are met, for each data item Q,

1. If in schedule S, transaction Ti reads the initial value of Q, then in schedule S' also transaction Ti must read the initial value of Q.

2. If in schedule S, transaction Ti executes read(Q), and that value was produced by transaction Tj (if any), then in schedule S' also transaction Ti must read the value of Q that was produced by the same write(Q) operation of transaction Tj .

3. The transaction (if any) that performs the final write(Q) operation in schedule S must also perform the final write(Q) operation in schedule S'.