Database Management System (DBMS)

Lecture-39

Dharmendra Kumar

December 15, 2020

Schedule

A schedule is a sequence of instructions of all the transactions in which order these instructions will execute.
There are two types of schedule. (1) Serial schedule (2) Concurrent schedule
Serial schedule: The serial schedule is a type of schedule where one transaction is executed completely before starting another transaction. In the serial schedule, when the first transaction completes its cycle, then the next transaction is executed. **Concurrent schedule:** If interleaving of operations is allowed, then the schedule will be concurrent schedule.

Transaction

Example: Consider the following two transactions:-

Let T1 and T2 be two transactions that transfer funds from one account to another. Transaction T1 transfers $50 from account A to account B. It is defined as

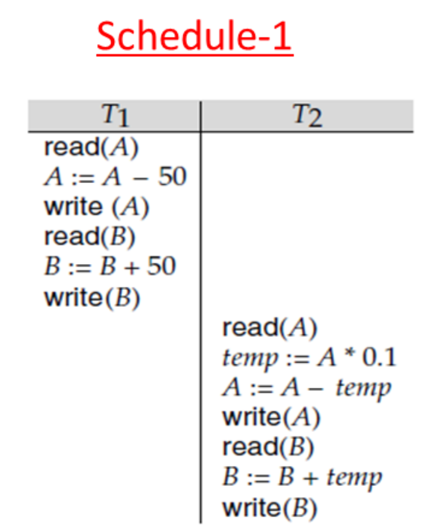
```
T1: read(A);
A := A - 50;
write(A);
read(B);
B := B + 50;
write(B).
```

Transaction T2 transfers 10 percent of the balance from account A to account B. It is defined as

```
T2: read(A);
temp := A * 0.1;
A := A - temp;
write(A);
read(B);
B := B + temp;
write(B).
```

Transaction

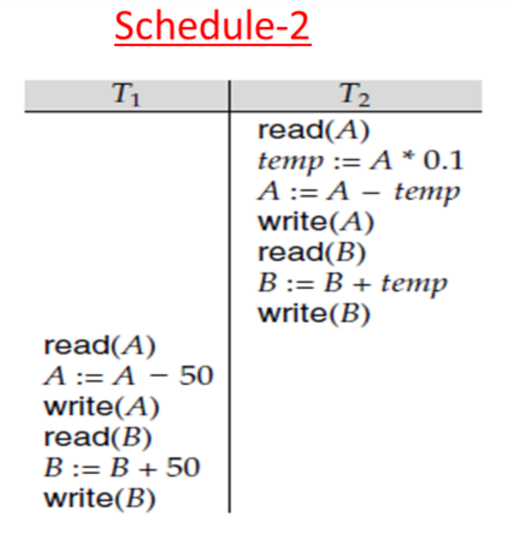
Now we make following four schedules for these transactions. schedule-1, schedule-2, schedule-3, and schedule-4.



Schedule-1

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)

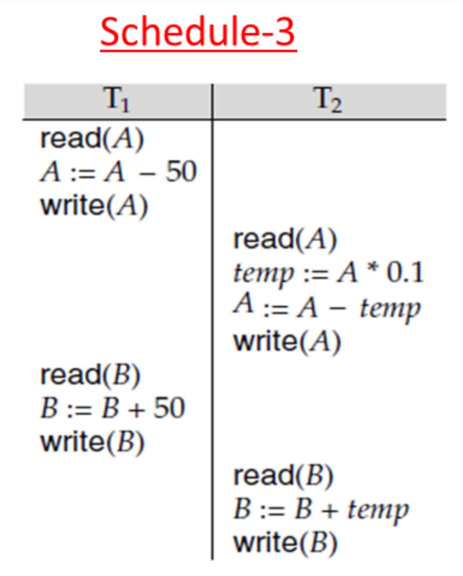
Transaction



Schedule-2

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)	

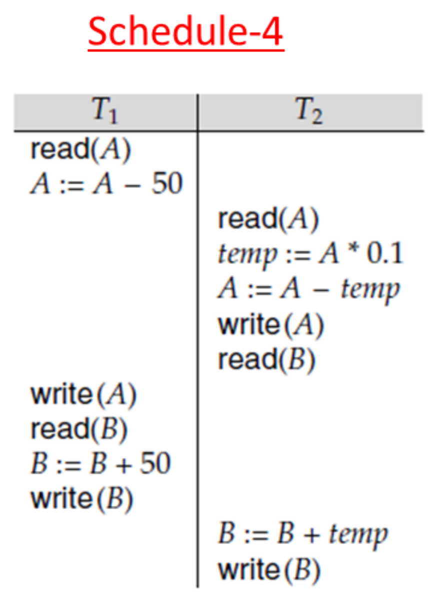
Transaction



Schedule-3

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)

Transaction



Schedule-4

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

To ensure consistency of database system, we must make a serializable schedule. Here, we will study two types of Serializability. These are

(1) Conflict Serializability

(2) View Serializability.

Conflict Serializability

Conflict Instructions:

Consider a schedule S in which there are two consecutive instructions I_i and I_j, of transactions T_i and T_j , respectively (i≠j). If I_i and I_j operates on same data item such as Q, then these instructions will be conflicting instructions if any one of the following is satisfied.

(1) I_i = read(Q), and I_j = write(Q).

(2) I_i = write(Q), and I_j = read(Q).

(3) I_i = write(Q), and I_j = write(Q).

In all other cases, instructions I_i and I_j will be non-conflicting.

Conflict Serializability (cont.)

Conflict equivalent: Two schedules S and S' are said to be conflict equivalent if a schedule S can be transformed in to a schedule S' by using swapping of non-conflicting instructions.

Conflict serializable: A schedule S is said to be conflict serizaliabe if it is conflict equivalent to a serial schedule.

Some examples:

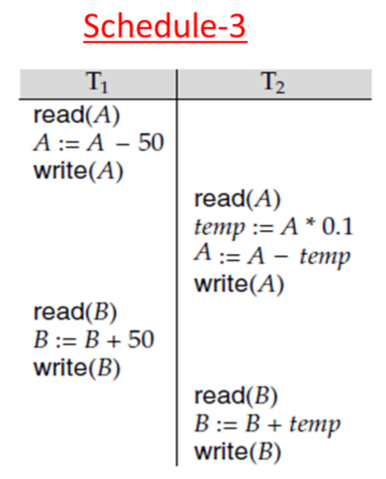
Example: Check schedule-1, schedule-2, schedule-3 and schedule-4 are conflict serializable or not.

Solution:

(1) Clearly schedule-1 and schedule-2 are serial schedules,therefore these schedules are conflict serializable.

Transaction

(2) Consider schedule-3 i.e.



Schedule-3

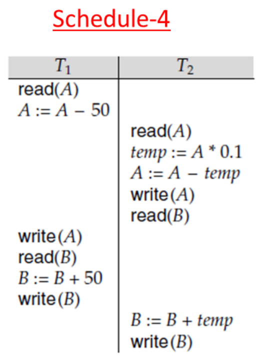
T_1	T_2
read(A)	
$A := A - 50$	
write(A)	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
read(B)	
$B := B + 50$	
write(B)	
	read(B)
	$B := B + temp$
	write(B)

Transaction

- In this schedule, instruction read(B) in transaction T_1 is non-conflicting with instructions read(A) and write(A) in transaction T_2, therefore we can swap instructions read(B) in T_1 and write(A)in T_2. Similarly, we can swap instructions read(B) in T_1 and read(A)in T_2.
- Similarly, instruction write(B) in transaction T_1 is non-conflicting with instructions read(A) and write(A) in transaction T_2, therefore we can swap instructions write(B) in T_1 and write(A)in T_2. Similarly, we can swap instructions write(B) in T_1 and read(A)in T_2.
- Therefore, using swapping, schedule-3 can be transformed in to a serial schedule-1 i.e. $T_1 T_2$. Therefore schedule-3 is conflict serializable.

Transaction

(3) Consider schedule-4 i.e.



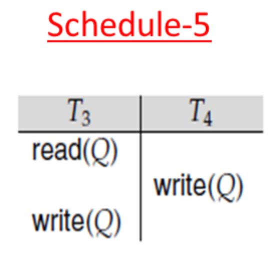
Schedule-4

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)

In this schedule, instruction write(A) in transaction T_1 and instruction write(A) in transaction T_2 are conflicting instructions, therefore, we can not swap these two instructions. Therefore, schedule-4 can not be transformed in to any serial schedule. Hence, schedule-4 is not conflict serializable.

Transaction

Example: Consider the following schedule-5:-



Schedule-5

T_3	T_4
read(Q)	
	write(Q)
write(Q)	

Is this schedule conflict serializable?

Transaction

Solution:

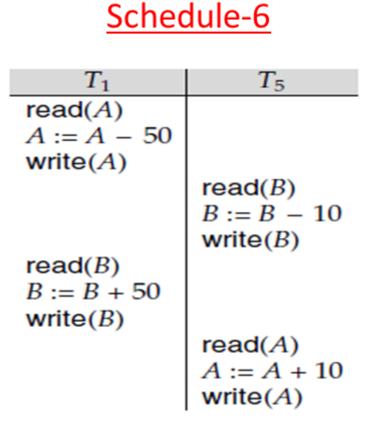
Clearly, in this schedule, instruction read(Q) in transaction T_3 and instruction write(Q) in transaction T_4 are conflicting instructions, therefore, we can not swap these two instructions.

Similarly, instruction write(Q) in transaction T_3 and instruction write(Q) in transaction T_4 are conflicting instructions, therefore, we can not swap these two instructions.

In this situation, this schedule can not be transformed into any serial schedule. Therefore, the schedule-5 is non-conflict serializable.

Transaction

Example: Consider the following schedule-6:-



Schedule-6

T_1	T_5
read(A)	
$A := A - 50$	
write(A)	
	read(B)
	$B := B - 10$
	write(B)
read(B)	
$B := B + 50$	
write(B)	
	read(A)
	$A := A + 10$
	write(A)

Is this schedule conflict serializable?

Transaction

Solution:

In this schedule, instruction read(B) in transaction T_1 and instruction write(B) in transaction T_5 are conflicting instructions, therefore, we can not swap these two instructions. Hence, this schedule can not be transformed into serial schedule $T_1 T_5$.

Similarly, instruction write(A) in transaction T_1 and instruction read(A) in transaction T_5 are conflicting instructions, therefore, we can not swap these two instructions. Hence, this schedule can not be transformed into serial schedule $T_5 T_1$.

Therefore, this schedule-6 is non-conflict serializable.