UNIT 4 TRANSACTIONS

Transaction Concepts –ACID Properties – Schedules – Serializability – Concurrency Control – Need for Concurrency – Locking Protocols – Two Phase Locking – Deadlocks – Transaction Recovery – Save Points – Isolation Levels – SQL Facilities for Concurrency and Recovery

Recap

Transaction Concepts

ACID Properties

Schedules

Schedules

A sequences of instructions that specify the chronological order in which instructions of concurrent transactions are executed

- Successfully completes – **Execute Commit**
- Fails to Complete – **Execute Abort**

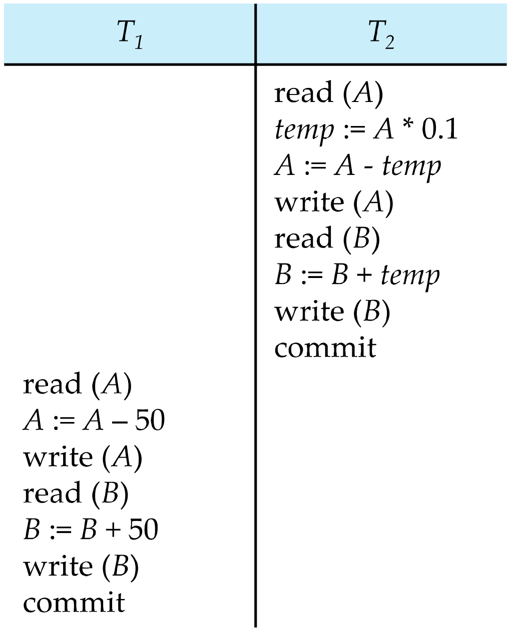
Schedule 1

- T_1 transfer $50 from A to B, and
- T_2 transfer 10% of the balance from A to B.
- A serial schedule - 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)	
commit	
	read (A)
	$temp := A * 0.1$
	$A := A - temp$
	write (A)
	read (B)
	$B := B + temp$
	write (B)
	commit

Schedule 2

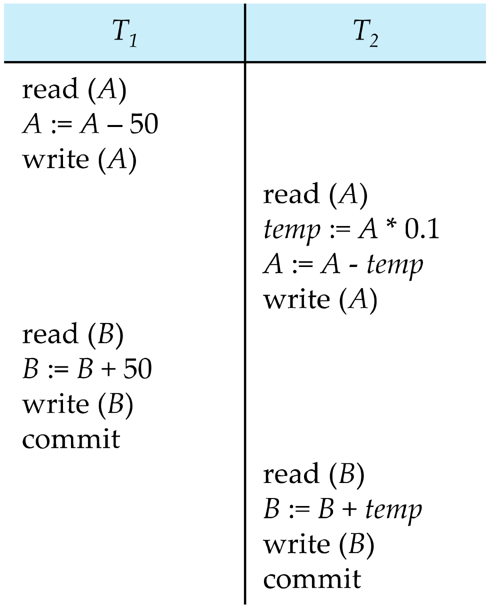
A serial schedule where T_2 is followed by T_1



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

Schedule 3

- T_1 and T_2 be the transactions defined previously not a serial schedule, but it is *equivalent* to Schedule 1



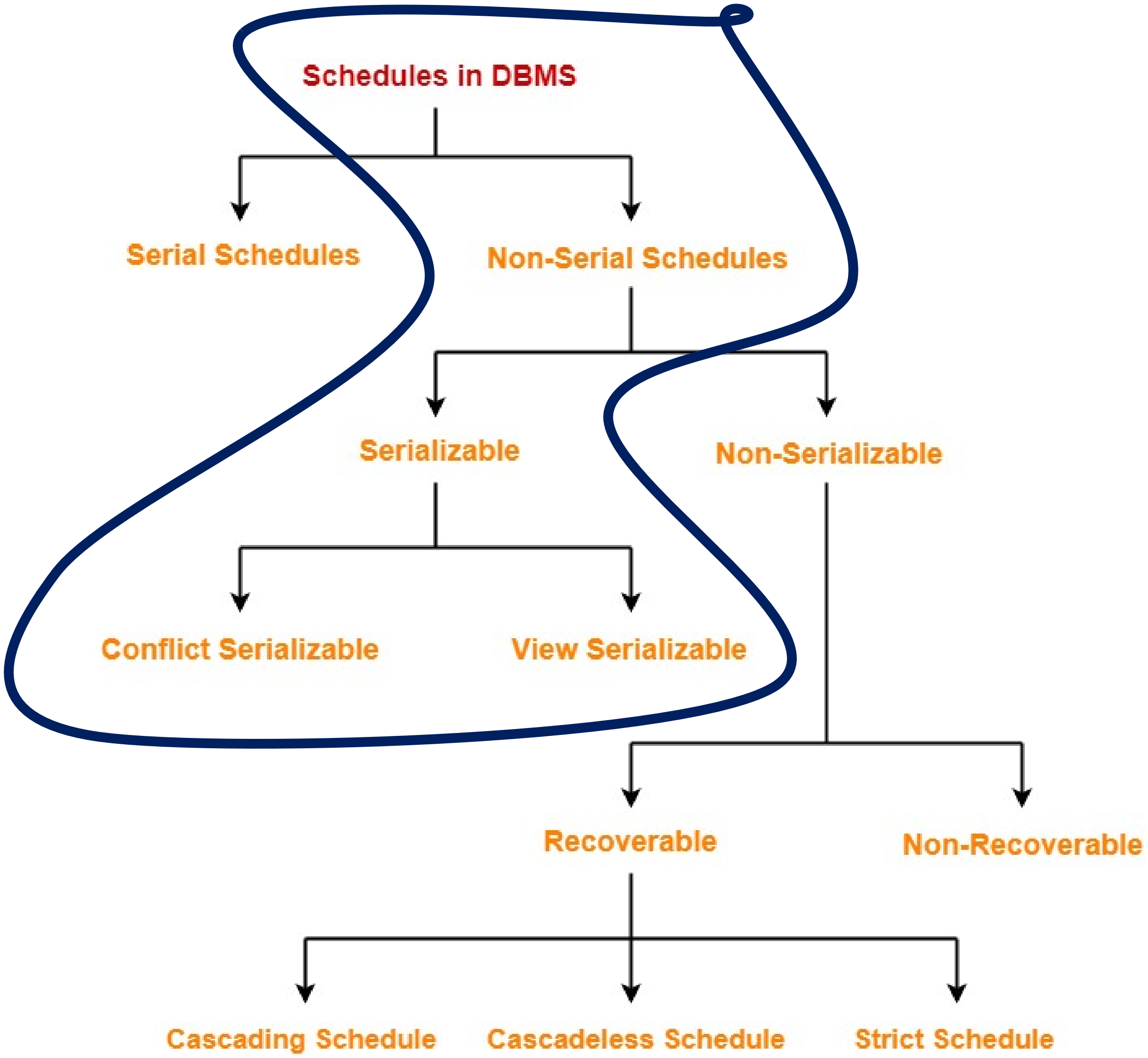
T_1	T_2
read (A)	
$A := A - 50$	
write (A)	
	read (A)
	$temp := A * 0.1$
	$A := A - temp$
	write (A)
read (B)	
$B := B + 50$	
write (B)	
commit	
	read (B)
	$B := B + temp$
	write (B)
	commit

Schedule 4

- 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)	
commit	
	B := B + *temp*
	write (B)
	commit

Inconsistency



Schedules in DBMS

Serial Schedules

Non-Serial Schedules

Serializable

Non-Serializable

Conflict Serializable

View Serializable

Recoverable

Non-Recoverable

Cascading Schedule

Cascadeless Schedule

Strict Schedule

Serial Schedules	VS	Serializable Schedules
No concurrency is allowed. Thus, all the transactions necessarily execute serially one after the other.		Concurrency is allowed. Thus, multiple transactions can execute concurrently.
Serial schedules lead to less resource utilization and CPU throughput.		Serializable schedules improve both resource utilization and CPU throughput.
Serial Schedules are less efficient as compared to serializable schedules. (due to above reason)		Serializable Schedules are always better than serial schedules. (due to above reason)

Serializability

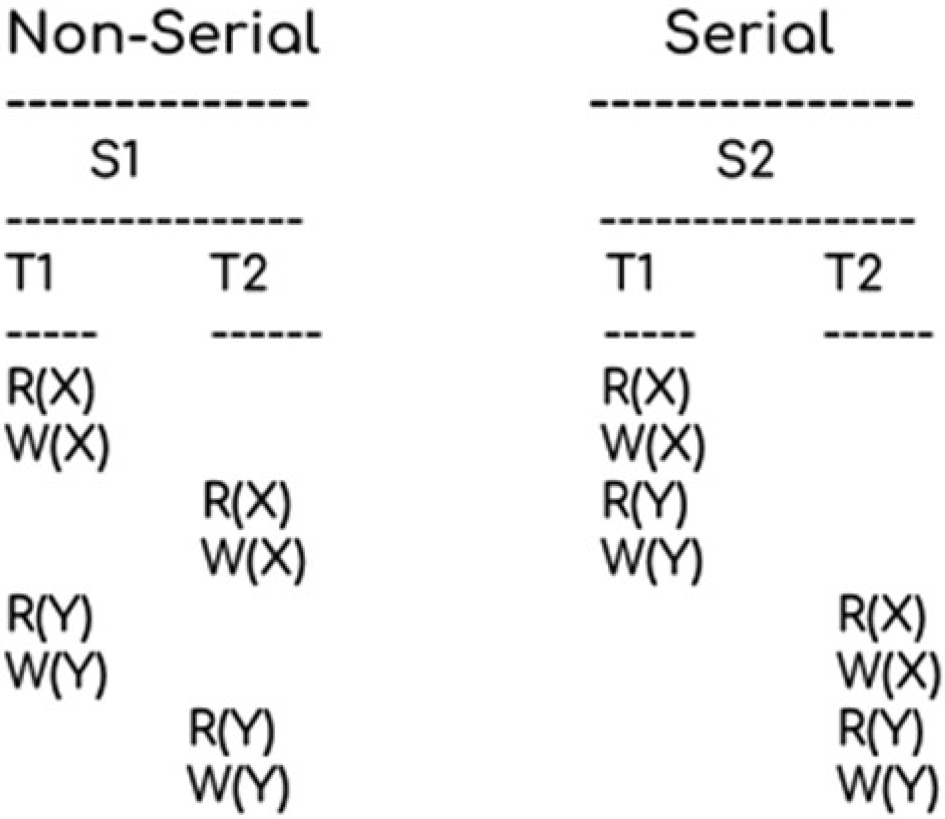
Read (A)
A=A-5
Write (A)

- A Schedule s of n transaction is serializability, if its equivalent to some serial schedule of the same transaction
- Transaction – set of instruction that perform single logical operation in the database
- When a multiple transactions are running concurrently, then there to be sequence in which the operations are to be performed

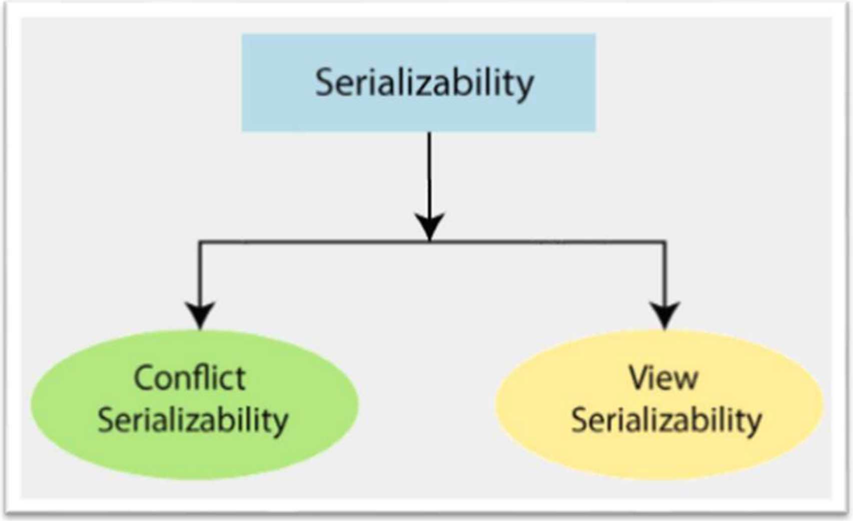
Serializability

- Serializability is a property of a system describing how different processes operate on shared data.
- A system is serializable if its result is the same as if the operations were executed in some sequential order, meaning there is no overlap in execution

Types of Serializability



```
Non-Serial                          Serial
--------------                      ---------------
     S1                                  S2
--------------                      ---------------
T1          T2                      T1          T2
-----       ------                  -----       ------
R(X)                                R(X)
W(X)                                W(X)
            R(X)                    R(Y)
            W(X)                    W(Y)
R(Y)                                            R(X)
W(Y)                                            W(X)
            R(Y)                                R(Y)
            W(Y)                                W(Y)
```

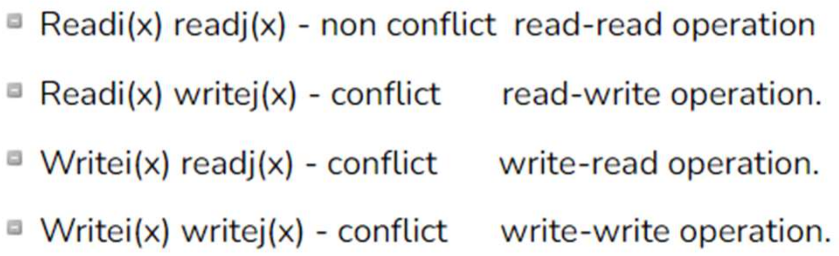


Conflict Serializability

- Conflict serializability orders any conflicting operations in the same way as some serial execution.
- A pair of operations is said to conflict if they operate on the same data item and one of them is a write operation.

Conflict Serializability

- A schedule is called conflict serializable if it can be transformed into a serial schedule by swapping non-conflicting operations.
- Two operations are said to be conflicting if all conditions satisfy:
  - They belong to different transactions
  - They operate on the same data item
  - At Least one of them is a write operation



- Readi(x) readj(x) - non conflict read-read operation
- Readi(x) writej(x) - conflict read-write operation.
- Writei(x) readj(x) - conflict write-read operation.
- Writei(x) writej(x) - conflict write-write operation.

View Serializability

- A schedule is view-serializability if it is viewed equivalent to a serial schedule.
- The rules it follows are as follows −
  - T1 is reading the initial value of A, then T2 also reads the initial value of A.
  - T1 is the reading value written by T2, then T2 also reads the value written by T1.
  - T1 is writing the final value, and then T2 also has the write operation as the final value.

Summary

- Schedules
- Serializability

Upcoming Session

- Concurrency Control
- Need for Concurrency

Thank you