Lecture 40

THE NORTHCAP UNIVERSITY

Solutions of Problems in Concurrent Schedule

Irrecoverable Schedules-

A transaction performs a dirty read operation from an uncommitted transaction

And commits before the transaction from which it has read the value

Transaction T1	Transaction T2
R(A)	
W(A)	
	R(A) //Dirty Read
	W(A)
	Commit
Rollback	

Irrecoverable Schedule

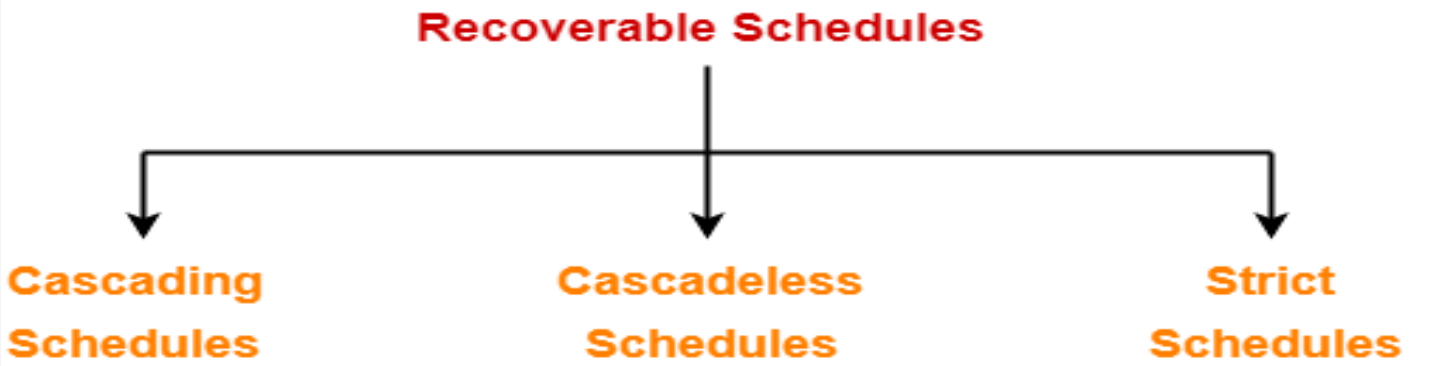
RECOVERABLE SCHEDULES-

A transaction performs a dirty read operation from an uncommitted transaction

- And its commit operation is delayed till the uncommitted transaction either commits or roll backs

Thumb Rule: No dirty read means a recoverable schedule.

Transaction T1	Transaction T2
R(A)	
W(A)	
	R(A) //Dirty Read
	W(A)
Commit	
	Commit //Delayed
Recoverable Schedule	



THE NORTHCAP UNIVERSITY

Cascading Schedule-

If in a schedule, failure of one transaction causes several other dependent transactions to rollback or abort, then such a schedule is called as a **Cascading Schedule** or **Cascading Rollback** or **Cascading Abort**.

It simply leads to the wastage of CPU time.

Cascading Rollback.

The failure of transaction T1 causes the transaction T2 to rollback.

The rollback of transaction T2 causes the transaction T3 to rollback.

The rollback of transaction T3 causes the transaction T4 to rollback.

T1	T2	T3	T4
R(A)			
W(A)			
	R(A)		
	W(A)		
		R(A)	
		W(A)	
			R(A)
			W(A)
Failure			

Cascading Recoverable Schedule

CASCADELESS SCHEDULE-

Cascadeless schedule allows only committed read operations.

Therefore, it avoids cascading roll back and thus saves CPU time.

T1	T2	T3	T4
R(A)			
W(A)			
Commit			
	R(A)		
	W(A)		
	Commit		
		R(A)	
		W(A)	
			Commit

Cascadeless Schedule

STRICT SCHEDULE-

Strict schedule allows only committed read and write operations.

Clearly, strict schedule implements more restrictions than cascadeless schedule.

T1	T2
W(A)	
Commit/Rollback	
	R(A)/W(A)
Strict Schedule	

PRACTICE PROBLEMS

PROBLEM-1

T1	T2
R(A)	
	R(A)
	W(A)
	Commit
W(A)	
Commit	

PRACTICE PROBLEM

SOLUTION

T1	T2
R(A)	
	R(A)
	W(A)
	Commit
W(A)	
Commit	

- Strict Recoverable Schedule
- Not Serializable
- RW Problem

PRACTICE PROBLEMS

PROBLEM-2

T1	T2
R(A)	
	R(A)
	W(A)
W(A)	
Commit	
	Commit

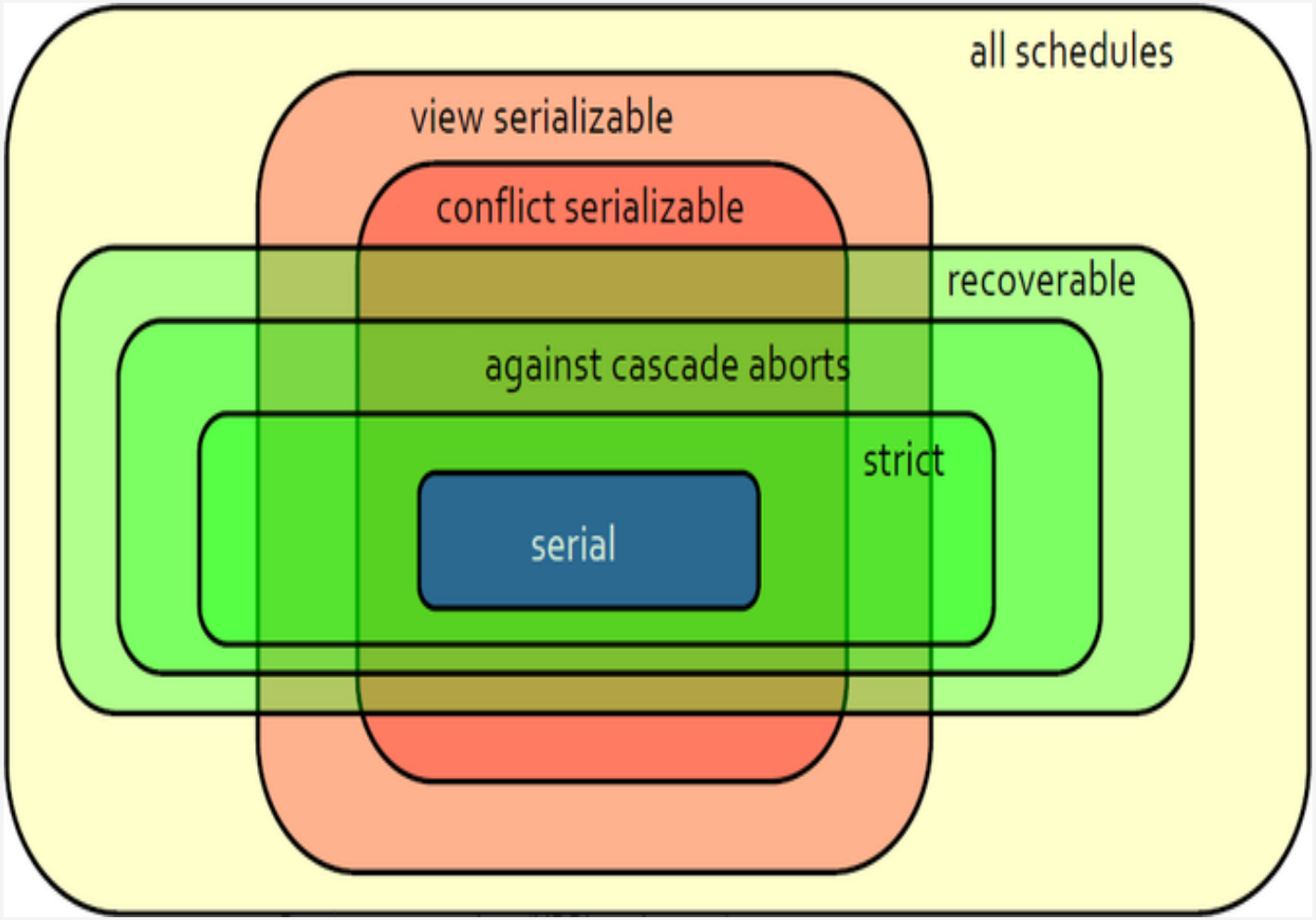
PRACTICE PROBLEM

SOLUTION

T1	T2
R(A)	
	R(A)
	W(A)
W(A)	
Commit	
	Commit

- Recoverable
- Cascadeless Rollback
- Not Strict Recoverable

Characterizing Schedules through Venn Diagram



Is the given schedule free from cascading rollback problem or not?

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-100;	A=4000			A=5000
W(A);	A=4000			A=4000
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
Failure Point				
Commit;				
		Commit;		

Solution

- Above table shows a schedule with two transactions, T1 reads and writes A and that value is read and written by T2.
- But later on, T1 fails. So we have to rollback T1. Since T2 has read the value written by T1, it should also be rollbacked.
- So, this schedule is not free from cascadeless rollback problem.

Is the given schedule Cascading Rollback Schedule?

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-100;	A=4000			A=5000
W(A);	A=4000			A=4000
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
Failure Point				
Commit;				
		Commit;		

SOLUTION

- Table shows a schedule with two transactions, T1 reads and writes A and that value is read and written by T2.
- But later on, T1 fails. So we have to rollback T1. Since T2 has read the value written by T1, it should also be rollbacked.
- As it has not committed, we can rollback T2 as well. So it is recoverable with cascading rollback.
- Recoverable with cascading rollback: If Tj is reading value updated by Ti and commit of Tj is delayed till commit of Ti , the schedule is called recoverable with cascading rollback.

Is the given schedule cascadeless recoverable Schedule?

T1	T1's buffer space	T2	T2's Buffer Space	Database
				A=5000
R(A);	A=5000			A=5000
A=A-100;	A=4000			A=5000
W(A);	A=4000			A=4000
Commit;				
		R(A);	A=4000	A=4000
		A=A+500;	A=4500	A=4000
		W(A);	A=4500	A=4500
		Commit;		

SOLUTION

- Table 3 shows a schedule with two transactions, T1 reads and writes A and commits and that value is read by T2.
- But if T1 fails before commit, no other transaction has read its value, so there is no need to rollback other transaction.
- So, this is a cascadeless recoverable schedule.

THANKS!!