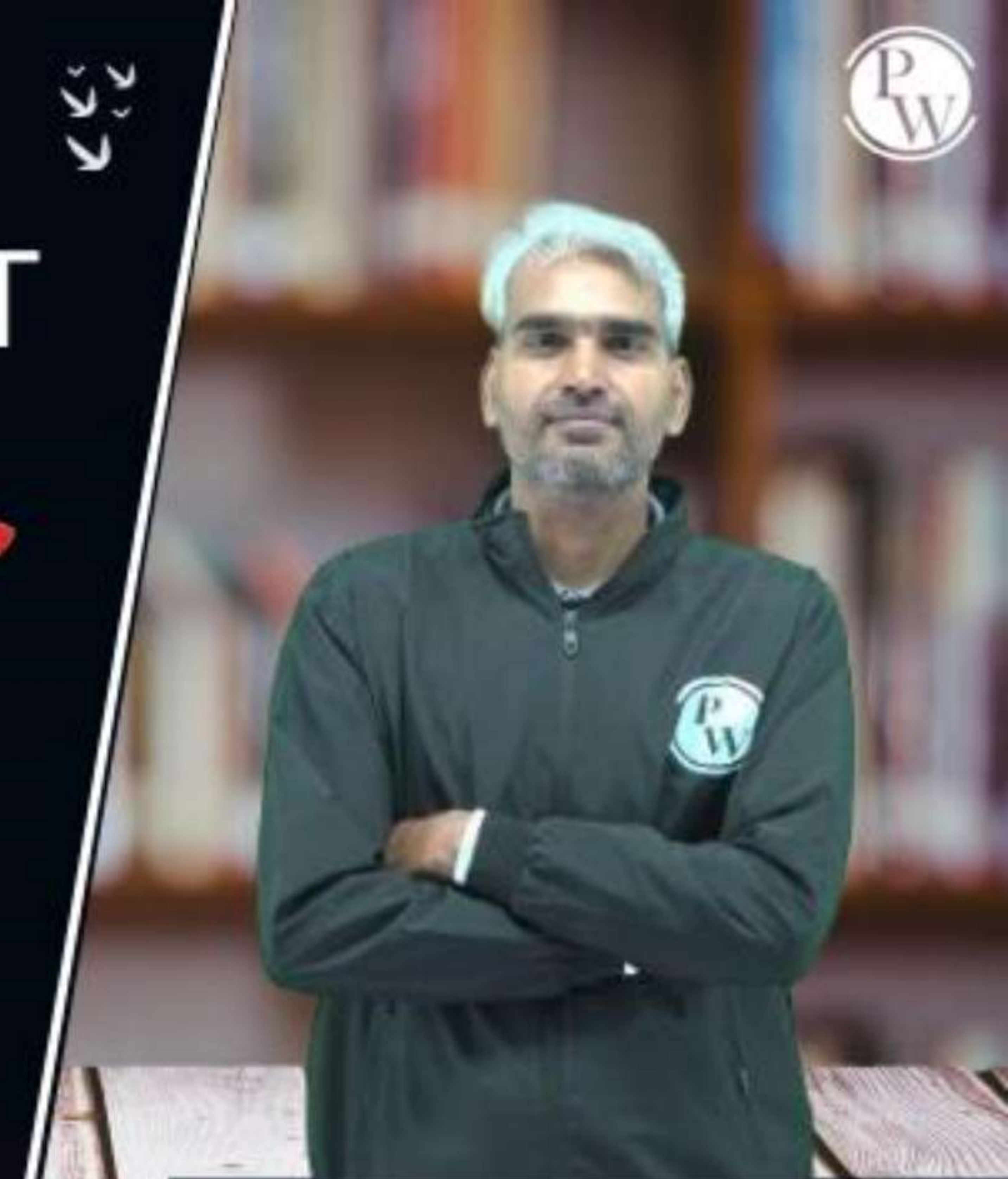
Computer Science & IT

Database Management System

Transaction & Concurrency control

Lecture No. 11



By- Vishal Sir

Recap of Previous Lecture

✓ **Topic** Basic 2PL and lock upgrading/downgrading

✓ **Topic** Problems possible with Basic 2PL

✓ **Topic** Strict 2PL

Slide

Topics to be Covered

- ✓ **Topic** Conservative 2PL
- ✓ **Topic** Rigorous 2PL
- **Topic** Time stamp ordering protocols
- **Topic** Read time stamp
- **Topic** Write time stamp

Slide

Topic : Different types of "Two phase locking protocols"

- ✓ Basic 2PL {we have already discussed}
- ✓ Strict 2PL {Avoid irrecoverability and Cascading Rollback problem. also lost update Problem
- ✓ Conservative 2PL
- Rigorous 2PL

Topic : Conservative 2PL

Conservative 2PL can be used to avoid deadlock

T_1	T_2
S(A)	
R(A)	
	X(B)
	W(B)
~~S(B)~~	
R(B)	
	~~X(A)~~
	W(A)

Transaction T_1 is holding the lock on data item A, and waiting for the lock on dataitem B

Transaction T_2 is holding lock on dataitem B and waiting for lock on dataitem A.

Necessary Cond^n for deadlock

① Mutual Exclusion

② No Preemption

③ Hold-&-wait

④ Circular wait

If any of the necessary Cond^n can be dis-satisfied then there will be no deadlock

* Conservative 2PL will dis-satisfy the "Hold & Wait" Condition by "Hold or Wait"

either hold all the locks required for its execution (or) wait for all the locks to become available at the same time (while not holding any lock)

In Conservative 2PL the transaction will request for all the locks required for its execution before starting its execution

(1) If all the requested lock are granted, then transaction will start its execution, and we are sure that the transaction will not have to wait for any lock during its execution {∴ Can not be involved in deadlock}

i.e. transaction will hold all the required resources and wait for none

(2) If atleast one of the requested lock is not granted, then transaction will release all the granted locks as well, and it will wait for all the locks to become available at a time {Can not be involved in deadlock}

i.e. transaction is not holding any lock and waiting for all

Hold or Wait

∴ No deadlock

* **Conservative 2PL:** → ① it ensures serializability ✓

② it avoids deadlock ✓

└ {but chances of starvation increases}

→ In Conservative 2PL we only define the order in which locks will be acquired {i.e. before starting the Execution of transaction}, but we do not define the position to unlock the acquired locks!

∴ Locks can be unlocked at any time

└ and Hence irrecoverability & Cascading rollback problems are possible with Conservative 2PL.

Topic : Different types of "Two phase locking protocols"

- ✓ Basic 2PL {we have already discussed}
- ✓ Strict 2PL {Avoid irrecoverability and Cascading Rollback problem. also lost update Problem
- ✓ Conservative 2PL {it Avoids deadlock}
- ✓ Rigorous 2PL

Topic : Rigorous 2PL

Rigorous 2PL = Basic 2PL + Every lock (both shared & Exclusive) Can be unlocked only after the Commit of that transaction

i.e

T_1	T_2
S(A)	
X(B)	
Commit	
U(A)	
U(B)	
	S(B)/X(B)/X(A)

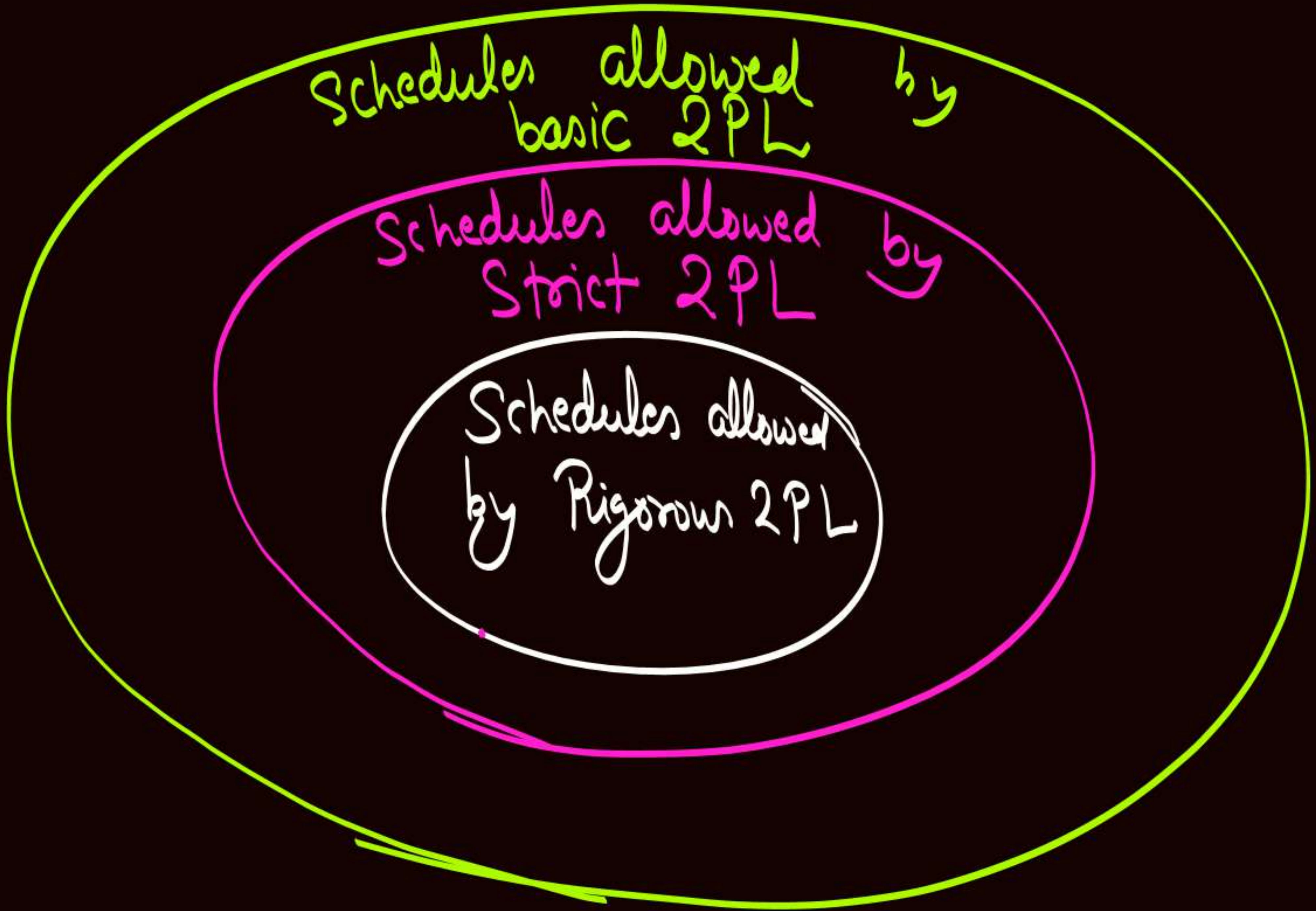
Rigorous 2PL: —

1. Ensures serializability
2. Avoid irrecoverability, Cascading rollback, and lost update problem
3. Every lock will be unlocked only after Commit operation, ∴ Implementation is easy

→ We only define the position at which locks will be unlocked

i.e., locks can be requested during the execution of transaction

→ and hence hold & wait is possible & Hence deadlock & Starvation is still possible


Schedules allowed by basic 2PL
Schedules allowed by Strict 2PL
Schedules allowed by Rigorous 2PL

Note:-

* 2PL ensures serializability, but only a subset of conflict serializable schedule are allowed to execute using 2PL,

There are many other serializable schedules which are not allowed by 2PL.

∴ We need some other Concurrency Control protocol.

Hence, Timp stamp ordering protocols are defined

Topic : Time stamp ordering protocols

There are two types of time stamp ordering Protocols

- ① Basic time stamp ordering protocol (B.T.S.O.P.)
- ② Thomas Write time stamp ordering protocol (T.W.T.S.O.P.)

(or)

Time Stamp ordering protocol with Thomas Write rule

Time Stamp :- Time stamp is a unique value assigned to each transaction by database management system.

Time stamps are assigned in ascending order.

- Let T_1 & T_2 are two transactions.

If Time stamp of transaction T_1 < Time stamp of transaction T_2

ie. $T.S.(T_1) < TS(T_2)$

then T_1 is old transaction
& T_2 is young transaction

① Read time stamp of dataitem A :

RTS(A)

It is the highest time stamp value among the time stamps of transactions that has performed the Read(A) Operation successfully.

Initially

$RTS(A) = 0$

② Write time stamp of dataitem A :

WTS(A) It is the highest time stamp value among the time stamps of transactions that has performed the Write(A) Operation successfully.

Initially
WTS(A) = 0

RTS(A) & WTS(A) :→

$TS(T_1)=10$ $TS(T_2)=20$ $TS(T_3)=30$ $TS(T_4)=40$

T_1	T_2	T_3	T_4	RTS(A)	WTS(A)
				0	0
R(A)				10	0
		R(A)		30	0
	W(A)			30	20
	R(A)			30	20
			W(A)	30	40
		W(A)		30	40

↓ time increasing

2 mins Summary

- ✓ **Topic** Conservative 2PL
- ✓ **Topic** Rigorous 2PL
- ✓ **Topic** Time stamp ordering protocols
- ✓ **Topic** Read time stamp
- ✓ **Topic** Write time stamp

Slide

THANK - YOU

Slide