Computer Science & IT

Database Management System

Transaction & concurrency control

Lecture No. 09

By- Vishal Sir

Recap of Previous Lecture

Topic — Basic 2PL

Topic — Strict 2PL

Slide

Topics to be Covered

- **Topic** Conservative 2PL
- **Topic** Rigorous 2PL
- **Topic** Basic time stamp ordering protocol
- **Topic** Time stamp ordering protocol with Thomas write rule

Slide

Topic : Problems possible with Basic 2PL

A schedule that is allowed to execute using basic 2PL protocol may suffer from,

1. Irrecoverability { Can be solved by Strict-2PL }
2. Deadlock { Can be solved by Conservative-2PL }
3. Starvation { No solution }

Topic : Irrecoverability with Basic 2PL

S → It is a Conflict serializable schedule which is allowed to execute using Basic 2PL But it suffer from irrecoverability.

T_1	T_2
X(A)	
R(A)	
W(A)	
X(B)	
S(C)	
U(A)	
	S(A)
	R(A)
R(B)	
W(B)	
U(B)	
	S(B)
	R(B)
	U(A)
	U(B)
	Commit
R(C)	
U(C)	
Commit	

Unsafe → { R(C), U(C) }

Uncommitted Read

T_2 depends on T_1
&
T_2 Commit before T_1

∴ Irrecoverable Schedule

Precedence graph

$T_1 \rightarrow T_2$

Acyclic
∴ C.S.S.

Topic : Deadlock with Basic 2PL

S

T_1	T_2
S(A)	
R(A)	
	X(B)
	W(B)
denied → S(B)	
R(B)	
	X(A) ← denied
	W(A)

T_1 depends on T_2 for resource

T_2 depends on T_1 for resource

Time out

Time out

Dependency graph w.r.t resource

Cyclic dependency graph

∴ Deadlock.

Topic : Starvation with Basic 2PL

	T_1	T_2	T_3	T_4	T_5
		S(A)			
denied because of T_2 →	X(A)				
Time Out			S(A)		
		U(A)			
denied because of T_3 →	X(A)				
Time out			U(A)		
				X(A)	
denied because of T_4 →	X(A)				
Time out				U(A)	
					S(A)
denied because of T_5 →	X(A)				

If this keeps happening then we say that transaction T_1 is under Starvation

Topic : 2PL Classification

There are different versions of 2PL

1. Basic 2PL (Already done)
2. Strict-2PL
3. Conservative-2PL
4. Rigorous-2PL

Topic : Strict 2PL

Basic-2PL restriction + Strict-Recoverability $Cond^n$ = Strict-2PL

A transaction T can request for a lock on any data item only if it has not performed any Unlock operation

+

T_1	T_2
W(A)	
Commit/Rollback	
	R(A)/W(A)

=

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

It will Ensure Serializability + It will Ensure Strict Recoverability = It will Ensure serializability as well as Strict recoverability

Topic : Strict 2PL

- Strict-2PL is a 2PL protocol with the restriction that Every Exclusive lock acquired by any transaction Can be unlocked only after the Commit operation of that transaction

Shared locks can be unlocked at any time as per the restriction of 2PL

Strict-2PL

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

It will ensure serializability as well as Strict recoverability

Topic : Strict 2PL

Strict-2PL is
- → Free from
  1. Irrecoverability
  2. Cascading rollback problem
  3. Lost-update problem
- → Not free from
  1. Deadlock
  2. Starvation

Topic : Conservative 2PL ⟹ Used to avoid deadlock

* In conservative 2PL, we will dis-satisfy "Hold & wait" by "Hold or Wait"

Necessary Cond^n for Deadlock
1. Mutual Exclusion
2. No preemption
3. Hold & Wait
4. Circular Wait

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

i.e. transaction will hold all the locks & Wait for None

* If all the locks requested by the transaction are granted then only transaction will start its execution, in this case transaction will not have to depend on any other transaction for locks during its execution

No other transaction will depend on it

i.e, tanasaction will wait for the locks and holds none

* If any one of the requested lock is not granted, then it will release all the granted locks as well {i.e, it will not hold any lock}, and it will go into time out, once time out period is over it will again request for all locks

Topic : Conservative 2PL

* In conservative 2PL, we will dis-satisfy "Hold & Wait" by "Hold or Wait"

* Conservative 2PL will only define the order in which locks can be acquired {i.e, before starting the execution of transaction}, but it does not define the order in which locks must be released.

{i.e, in Conservative 2PL, an Exclusive lock acquired by a transaction may be unlocked, before the commit opn of that transaction}

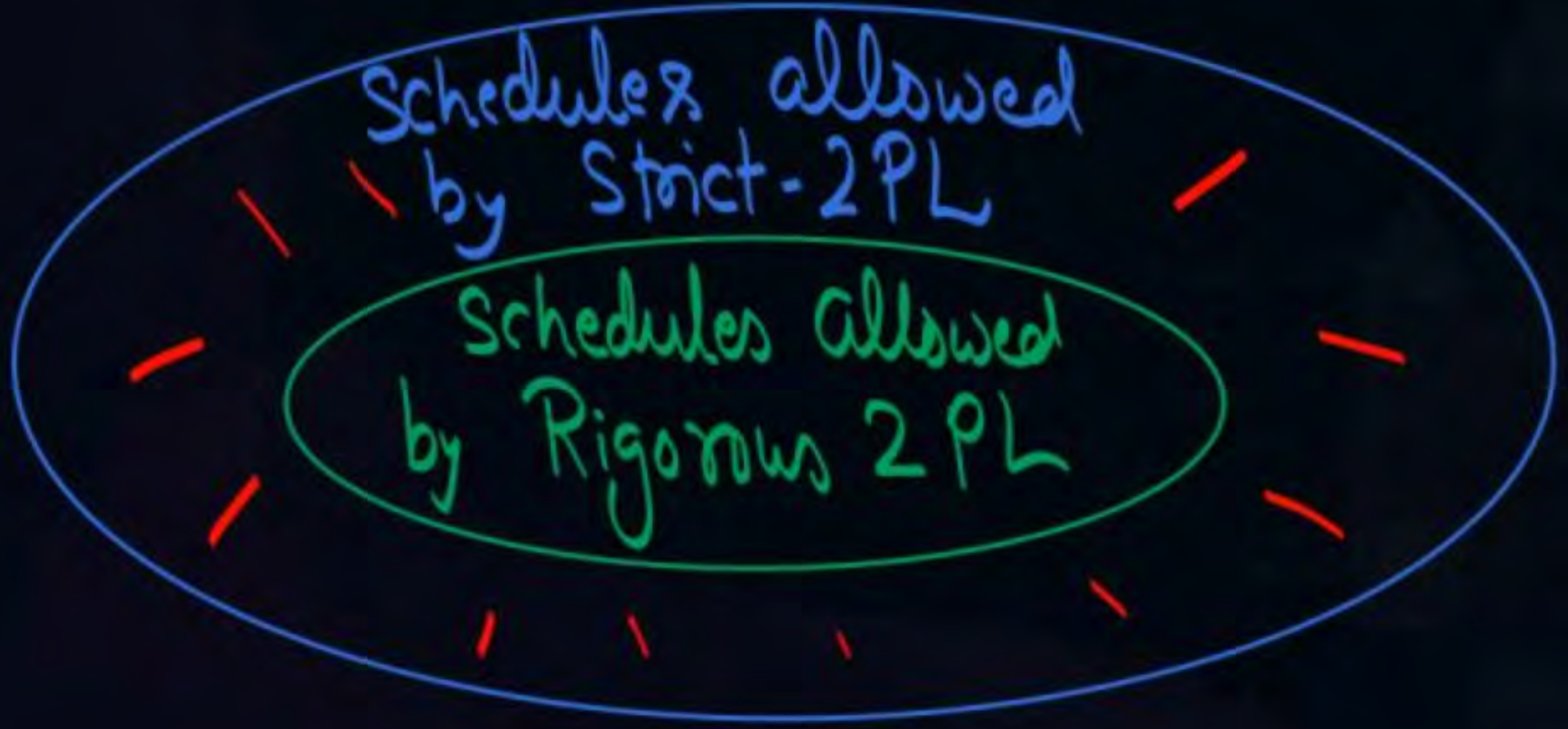
Hence, Conservative 2PL may suffer from irrecoverability, Cascading rollback and lost-update problem.

Conservative 2PL is

- free from
  1. Deadlock
- Not free from
  1. Starvation
  2. Irrecoverability
  3. Cascading Rollback
  4. Lost-update Problem

Topic : Rigorous 2PL

* In Rigorous 2PL, both Shared (S) as well as Exclusive (X) locks must be unlocked after the Commit opⁿ of transaction



Rigorous 2PL is

Free from,
1. Irrecoverability
2. Cascading Rollback
3. Lost-update Problem

Not free from,
1. Deadlock
2. Starvation

Implementation of Rigorous-2PL is Easy

↓

We just need to unlock all the locks after Commit.

Time Stamp Ordering Protocols

Topic : Time stamp ordering protocols

* There are two different versions of time stamp ordering protocol
  1. Basic time stamp ordering protocol
  2. Time stamp ordering protocol with Thomas Write Rule

* Time Stamp :-

Time stamp is a unique value assigned by DBMS to each transaction in ascending order.

* Let T_1 and T_2 are two transactions in the system, such that

$$\underset{TS(T_1)}{\text{Time stamp of } T_1} < \underset{TS(T_2)}{\text{Time stamp of } T_2}$$

then, T_1 is the old transaction
&
T_2 is the younger transaction

- Read time stamp of data item 'A'

  RTS(A) : It is the highest time stamp value among the time stamps of the transactions that has Performed Read(A) opⁿ successfully.

- Write time stamp of dataitem 'A'

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

* Initially,

$$\left.\begin{aligned} RTS(A) &= 0 \\ WTS(A) &= 0 \end{aligned}\right\} \text{for all dataitems 'A'}$$

eg: 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)
Initially →					0	0
After R(A) opn of transaction T_1	R(A)				10	0
			R(A)		30	0
		W(A)			30	20
		R(A)			30	20
				W(A)	30	40
			W(A)		30	40

Topic : Basic Time stamp ordering protocol

(B.T.S.O.P.)

* A schedule is allowed to execute using B.T.S.O.P. if and only if schedule is a Conflict serializable schedule and conflict equivalent serial schedule is based on time stamp ordering of the transaction

eg Consider the Schedule 'S' with time stamp ordering as specified along with transactions

S

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

We can observe that the time stamp ordering is

$$TS(T_2) < TS(T_1) < TS(T_3)$$

i.e; $T_2 \rightarrow T_1 \rightarrow T_3$

Schedule 'S' will be allowed to Execute using B.T.S.O.P. if and only if schedule 'S' is a Conflict serializable schedule and Conflict Equivalent serial schedule is $T_2 \rightarrow T_1 \rightarrow T_3$

(i.e; Based on Time Stamp Ordering)

▸ **Basic time stamp ordering protocol Conditions:-**

· Let T_1 & T_2 are two transactions such that $\underset{\text{old}}{TS(T_1)} < \underset{\text{young}}{TS(T_2)}$

① When transaction T_1 issue a Read(A) opⁿ

(i)

T_1	T_2
	R(A)
R(A)	

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time Stamp ordering is $T_1 \rightarrow T_2$

If transaction T_1 is allowed to perform this R(A) operation, then also the behaviour of this schedule will be Conflict Equivalent to Serial Schedule based on time Stamp ordering of transactions (i.e, $T_1 \rightarrow T_2$)

∴ Transaction T_1 is allowed to perform this R(A) opⁿ

· Read - Read opⁿ will never Create any problem

· If $RTS(A) > TS(T_1)$, then T_1 is allowed to perform R(A) opⁿ

(ii)

T_1	T_2
	W(A)
R(A)	

Rollback T_1

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time Stamp ordering is $T_1 \rightarrow T_2$

If transaction T_1 is allowed to perform this R(A) operation then the behaviour of this schedule will not be Conflict Equivalent to Serial Schedule based on time Stamp ordering of transactions (i.e, $T_1 \rightarrow T_2$)

∴ T_1 is not allowed to perform this R(A) opⁿ and we will rollback transaction T_1.

▸ If $WTS(A) > TS(T_1)$, then T_1 is not allowed to perform this R(A) opⁿ & Rollback T_1.

Basic time stamp ordering protocol Conditions:-

Let T_1 & T_2 are two transactions such that $TS(T_1) < TS(T_2)$

② When transaction T_1 issue a Write(A) opn

(i)

T_1	T_2
	R(A)
W(A)	

Rollback

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time stamp ordering is $T_1 \rightarrow T_2$

If transaction T_1 is allowed to perform this W(A) opn then behaviour of schedule will not be Conflict Equivalent to serial schedule based on time stamp ordering (i.e. $T_1 \rightarrow T_2$)
∴ T_1 is not allowed to perform this W(A) opn, hence rollback T_1

* If transaction T_1 issue a W(A) opn and if $RTS(A) > TS(T_1)$, then Rollback T_1

(ii)

T_1	T_2
	W(A)
W(A)	

Rollback

$TS(T_1) = 10$
$TS(T_2) = 20$
∴ Time stamp ordering is $T_1 \rightarrow T_2$

If transaction T_1 is allowed to perform this W(A) opn then behaviour of schedule will not be Conflict Equivalent to serial schedule based on time stamp ordering (i.e. $T_1 \rightarrow T_2$)
∴ T_1 is not allowed to perform this W(A) opn, hence rollback T_1

* If transaction T_1 issue a W(A) opn and if $WTS(A) > TS(T_1)$, then Rollback T_1

Basic time stamp ordering Protocol Condition :-

Let T_1 & T_2 are two transactions s.t. $TS(T_1) < TS(T_2)$

ie, T_1 is older than T_2

1. Let T_1 issue Read(A) opⁿ :-

   (a) If $WTS(A) > TS(T_1)$, then rollback T_1

   (b) Otherwise T_1 is allowed to perform this R(A) opⁿ
   ∴ T_1 will perform this R(A) opⁿ, and set
   $RTS(A) = Max(RTS(A), TS(T_1))$

2. Let T_1 issue Write(A) opⁿ :-

   (a) If $RTS(A) > TS(T_1)$, then Rollback T_1

   & (b) If $WTS(A) > TS(T_1)$, then Rollback T_1

   (c) Otherwise, T_1 will Perform this W(A) opⁿ, and set
   $WTS(A) = TS(T_1)$

THANK - YOU

Slide