AMRITA VISHWA VIDYAPEETHAM

श्रद्धावान् लभते ज्ञानम्

Version 1.0

TWO-PHASE LOCKING PROTOCOL

Let's revisit

1. Let's go back to understand what concurrency control is..
2. Now, let's again go back to understand what locking is..

Now, that you are done with understanding what concurrency and locking is.. Let us learn something new!!

Two Phase Locking protocol!!

Two phase locking protocol

1. One protocol that ensures serializability is the two-phase locking protocol.
   a. Serializability : When multiple transactions are running concurrently then there is a possibility that the database may be left in an inconsistent state.
   b. Serializability is a concept that helps us to check which schedules are serializable. A serializable schedule is the one that always leaves the database in consistent state.
2. This protocol requires that each transaction issue lock and unlock requests in two phases:
   a. Growing phase. A transaction may obtain locks, but may not release anylock.
   b. Shrinking phase. A transaction may release locks, but may not obtain any new locks.
3. Initially, a transaction is in the growing phase. The transaction acquires locks as needed. Once a transaction releases a lock, it enters in the shrinking phase and it cannot issue more lock requests.

Example

- Transaction given below is a two phase transaction.
  - lock X(B);
  - read(B);
  - B:= B-50;
  - write(B);
  - lock X(A);
  - read(A);
  - A:=A+50;
  - write(A);

  Growing phase

  - unlock(B);
  - unlock(A);

  Shrinking phase

But why???
To achieve serializability
How??

T1	T2
L(A)	
W(A)	L(A)
U(A)	
	L(A)
	W(A)...

Waiting for T1 to release A

Lock Point

T1	T2
L(A)	
W(A)	L(C)
L(B)	W(C)
W(B)	L(D)
	W(D)
U(A)	R(C)
	U(C)
	U(D)

Lock Point of T1

Lock Point of T2

Advantages

1. Always ensures serializability

Disadvantages

1. Not free from irrecoverability
2. Not free from deadlock
3. Not free from starvation
4. Not free from cascading rollback

Not free from irrecoverability

T1	T2
L(A)	
W(A)	
U(A)	
	L(A)
	R(A)
	Commit
Imagine T1 is rollbacked, so T2 needs to be rolled back, but its not recoverable because T2 has committed.	

Not free from deadlock

T1	T2
L(A)	L(B)
W(A)	
L(B)	
	L(A)
In the growing phase, T1 and T2 are going to be in deadlock because both are now waiting infinitely.	

Not free from starvation

T1	T2
L(A)	
W(A)	
	L(A)
	waiting...
Imagine T1 is holding A for long period, then T2 is in starvation - finite waiting but waiting is there.	

Not free from cascading rollback

T1	T2	T3	T4
L(A)			
W(A)			
U(A)			
	L(A)		
	R(A)		
	U(A)		
		L(A)	
		R(A)	
		U(A)	
			L(A)
			R(A)
Rollback			U(A)
Imagine T1 is rollbacked, so T2, T3 and T4 needs to be rolled back			

AMRITA VISHWA VIDYAPEETHAM

Strict two phase protocol

1. This protocol requires not only that locking be two phase, but also that all exclusive-mode locks taken by a transaction be held until that transaction commits.
2. This requirement ensures that any data written by an uncommitted transaction are locked in exclusive mode until the transaction commits, preventing any other transaction from reading the data.
3. Rollback is cascadeless
4. Recoverable.

T1	T2
X(A)	
W(A)	
Not possible	
	S(A)
	Waiting
Commit	
U(A)	

Recoverable

T1	T2
X(A)	
W(A)	
	S(A) - Not Possible
Commit	
U(A)	
In this case, even T2 is not even be able to access A because without committing T1 will not release A. So it is recoverable.	

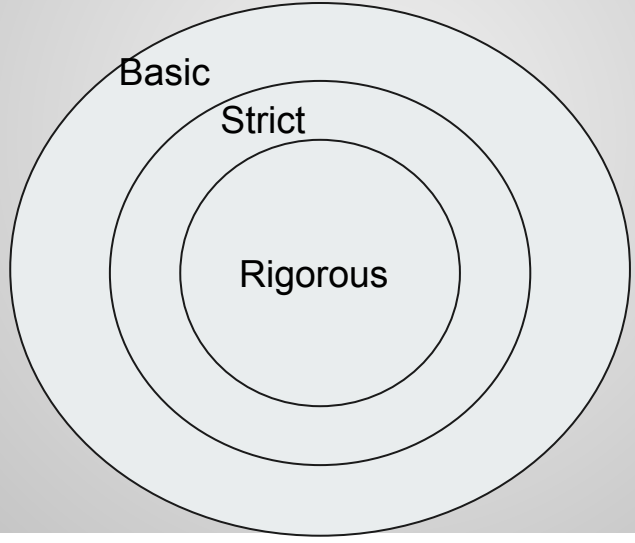
Cascadeless Rollback

T1	T2
X(A)	
W(A)	
	S(A)
	Waiting….
Commit	
U(A)	
	S(A)
	R(A)
So, after committing only T1 releases A, so even if T2 is reading data of A, it will be consistent.	

Rigorous Two Phase protocol

1. Another variant of two-phase locking is the rigorous two-phase locking protocol, which requires that all locks be held until the transaction commits.
2. No much advantages from strict 2 phase, but it is more restrictive.

So, if we conclude,



What next??

More refinement of Two-phase protocol by lock conversions - upgrading shared lock to exclusive, downgrading exclusive lock to shared lock - as we require. Let's discuss on that in the next video

Thank you !!