Concurrency Control

Mirunalini.P

SSNCE

May 22, 2022

Table of Contents

Session Objective

- Locking Mechanisms
- Types of Locks
- Two phase protocol

Locking Rules

- Transaction must request appropriate lock on a data item X before it reads or writes X.
- If T holds a write (exclusive) lock on X, it can both read and write X.
- If T holds a read lock on X, it can only read X.
- T must unlock all items that it holds before terminating (also T cannot unlock X unless it holds a lock on X).

Shared/Exclusive or Read/Write locks Rules

- A transaction T must issue the operation read lock(X) or write lock(X) before any read item(X) operation is performed in T.
- A transaction T must issue the operation write lock(X) before any write item(X) operation is performed in T.
- A transaction T must issue the operation unlock(X) after all read item(X) and write item(X) operations are completed in T.
- A transaction T will not issue a read lock(X) operation if it already holds a read lock or a write lock on item X. (exceptions: downgrading of lock from write to read)
- A transaction T will not issue a write lock(X) operation if it already holds a read lock or write lock on item X. (exceptions: upgrading of lock from read to write)

Conversion of Locks

- **Lock conversion**: A transaction that already holds a lock on item X is allowed under certain conditions to convert the lock from one locked state to another.
- **Upgrading Lock**: It is possible for a transaction T to issue a read lock (X) and then later to upgrade the lock by issuing a write lock (X) operation.
  - if Ti holds a read-lock on X, and no other Tj holds a read-lock on X then convert (upgrade) read-lock(X) to write-lock(X)
  - else force Ti to wait until all other transactions Tj that hold read locks on X release their locks
- **Degrading Lock** It is also possible for a transaction T to issue a write lock (X) and then later to downgrade the lock by issuing a read lock (X) operation.
- Either binary locks or read/write locks in transactions does not guarantee serializability of schedules.

Serial Schedules using Locks

T1	T2
read lock(Y);	read lock(X);
read item(Y);	read item(X);
unlock(Y);	unlock(X);
write lock(X);	write lock(Y);
read item(X);	read item(Y);
X := X + Y;	Y := X + Y;
write item(X);	write item(Y);
unlock(X);	unlock(Y);

Table 1: Transactions example using shared/Exclusive Locks

Initial values: X=20, Y=30 Result serial schedule T1 followed by T2 : X=50, Y=80 Result of serial schedule T2 followed by T1 : X=70, Y=50

Nonserializable schedule S that uses locks

T1	T2
read lock(Y);	
read item(Y);	
unlock(Y);	
	read lock(X);
	read item(X);
	unlock(X);
	write lock(Y);
	read item(Y);
	Y := X + Y;
	write item(Y);
	unlock(Y);
write lock(X);	
read item(X);	
X := X + Y;	
write item(X);	
unlock(X);	

Guaranteeing Serializability by Two-Phase locking

To guarantee Serializability an additional protocol called two phase locking protocol is proposed has more concerns in the positioning of locking and unlocking operations.

- A transaction is said to follow the two-phase locking protocol if all locking operations **(read lock, write lock)** precede the first unlock operation in the transaction.Transaction are in Two Phases:
- **Expanding or Growing Phase**: During which new locks on items can be acquired but none can be released;
- **Shrinking (second) phase**: During which existing locks can be released but no new locks can be acquired.
  With Lock Conversion:
- **Upgrading of locks:** (from read-locked to write-locked) must be done during the expanding phase.
- **Downgrading of locks:** (from write-locked to read-locked) must be done in the shrinking phase.

Two Phase Locking Protocol

- When transaction starts executing, it is in the locking phase, and it can request locks on new items
- A transaction may be blocked (forced to wait) if a lock request is not granted.
- Once the transaction unlocks an item, it starts its shrinking phase and can no longer request new locks.
- The combination of locking rules and 2-phase rule ensures serializable schedules

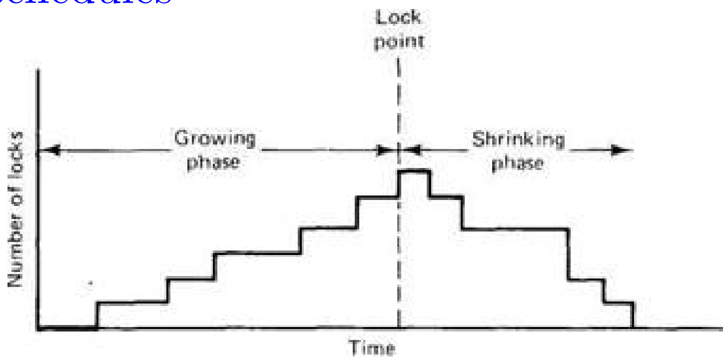

Figure 1: Two phase Locking Protocol

Two-Phase locking

T1
growing phase:
read lock(Y);
write$_{l}ock(X)$;
read item(Y);
read item(X);
X := X + Y;
write item(X);
Shrinking phase:
unlock(Y);
unlock(X);

Two-Phase locking Protocol - EXAMPLE

The transactions T2 was put in the waiting state untill the transaction T2 unlocks the data base item A
Serializiablity is achieved by the transactions $T1- > T2$

T1	T2
growing phase:	
write_lock(A);	
read_item(A);	
write_item(A);	
	read_lock(A);
	read_item(A);
	Waiting
read_lock(B);	
read_item(B);	
Shrinking phase:	
unlock(A);	
unlock(B);	
	resumed:

Two phase Locking protocol - serializability

- X must remain locked by T until all items that the transaction needs to read or write have been locked; only then can X be released by T.
- Another transaction seeking to access X may be forced to wait, even though T is done with X;
- Conversely, if Y is locked earlier than it is needed, another transaction seeking to access Y is forced to wait even though T is not using Y yet.
- **Theorem:** If every transaction in a schedule follows the 2PL rules, the schedule must be serializable.
- The 2PL suffers from the problem of **deadlock and cascading rollback**

Two phase Locking protocol - Example

Figure 22.4
Transactions T_1' and T_2', which are the same as T_1 and T_2 in Figure 22.3, but follow the two-phase locking protocol. Note that they can produce a deadlock.

T_1'	T_2'
read_lock(Y);	read_lock(X);
read_item(Y);	read_item(X);
write_lock(X);	write_lock(Y);
unlock(Y)	unlock(X)
read_item(X);	read_item(Y);
$X := X + Y$;	$Y := X + Y$;
write_item(X);	write_item(Y);
unlock(X);	unlock(Y);

Figure 2: Two Phase locking protocol

2pl-Deadlock

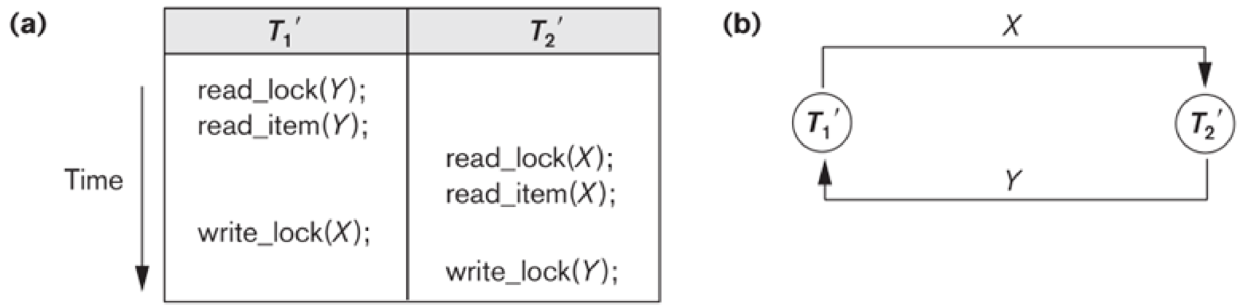

Figure 22.5
Illustrating the deadlock problem. (a) A partial schedule of T_1' and T_2' that is in a state of deadlock. (b) A wait-for graph for the partial schedule in (a).

Figure 3: deadlock

Types of Two-phase Locking

Two-phase policy generates different locking algorithms

- **Basic 2PL:** Transaction locks data items incrementally, results in deadlock.
- **Strict 2PL:** A more stricter version of Basic algorithm where unlocking of exclusive write is performed after a transaction terminates (commits or aborts and rolled back).
- **Conservative 2PL:** Prevents deadlock by locking all desired data items before transaction begins execution.
- **Rigorous 2PL** : A more restrictive variation of strict 2PL is rigorous 2PL which also guarantees strict schedules where unlocking of exclusive write or read is performed after a transaction terminates.

Strict 2PL

- The most popular variation of 2PL is strict 2PL, which guarantees strict schedules
- In this variation, a transaction T does not release any of its exclusive (write) locks until after it commits or aborts.
- Hence, no other transaction can read or write an item that is written by T unless T has committed, leading to a strict schedule for recoverability.
  - Policy - Release write locks only after terminating. Transaction is in expanding phase until it ends (may release some read locks before commit).
  - Property: NOT a deadlock-free protocol but no cascading rollback
  - Practical : Possible to enforce recoverability.

Rigorous 2PL

- A transaction T does not release any of its locks (exclusive or shared) until after it commits or aborts.
- Does not unlock any of its items until after it terminates (by committing or aborting), so the transaction is in its expanding phase until it ends.
- Behaves similar to Strict 2PL except it is more restrictive, but easier to implement since all locks are held till commit.
- No cascading rollback and deadlock may happen

Conservative 2PL

- Requires a transaction to lock all the items(atomic manner) it accesses before the transaction begins execution, by predeclaring its read-set and write- set.
- If any of the predeclared items needed cannot be locked, the transaction does not lock any item; instead, it waits until all the items are available for locking.
  - Policy: Lock all that you need before reading or writing.
  - Transaction is in shrinking phase after it starts. Property: Conservative 2PL is a deadlock-free protocol
  - Practical : Difficult to use because of diffculty in predeclaring the read-set and write-set.