May 1st 23

DBMS Theory Assignment

PA - 20
Krishnaraj P.T
A1
1032210888

Q.1. What are the problems of a lock based protocol in DBMS?

→ Why are they used?

(*) They are used in DBMS to maintain consistency and ensure transaction Isolation.

(*) They are a concurrency control mechanism.

(*) They prevents conflicts in concurrent transactions.

(*) This is done by making use of "shared" and "exclusive" lock based systems.

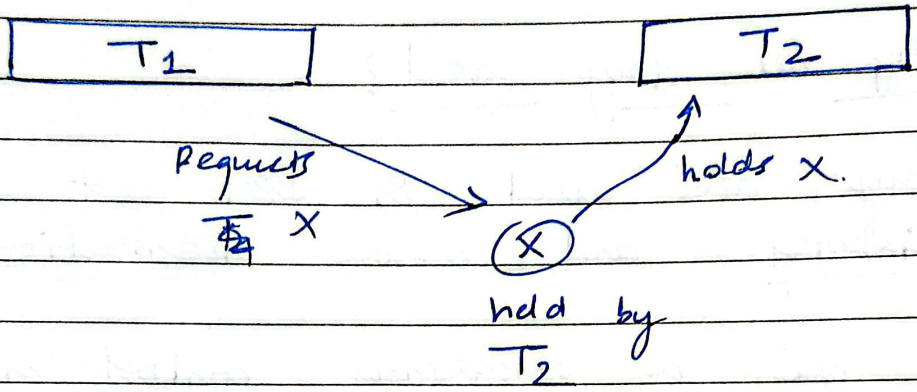
→ What are the possible problems they can cause?

(1) Deadlocks: They can easily result in deadlocks when 2 or more transactions are blocked, because they are waiting for each other's locks to be released.

(2) Starvation: A transaction may also be blocked indefinitely if it keeps waiting

to acquire a lock, but is unable to do so as the other transaction is continuously holding it back.

This results in starvation.



(3) Reduced Concurrency: The whole point of using transactions at the same time is faster processing. The assumption is that both T_1 and T_2 are ~~happening~~ simultaneously running;

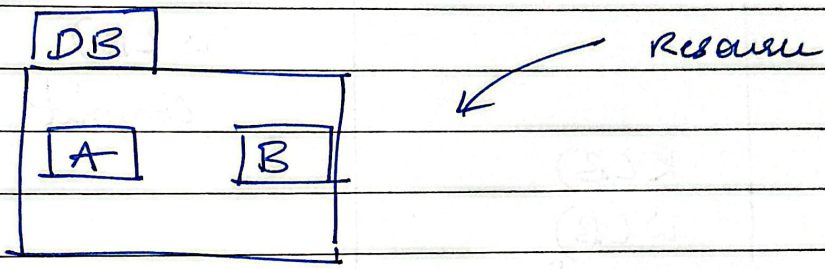
but this falls apart if they both are just waiting for locks to be released and shared between each other.

As this is an inevitable consequence of lock based protocols, it reduces the efficiency and concurrency of processing transactions.

(4) Lock contention:

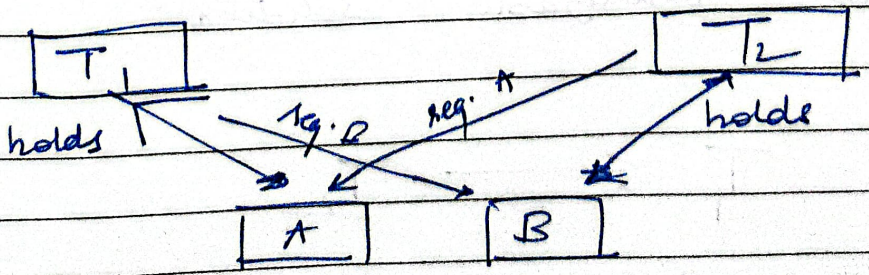
→ These protocols call for contention of shared resources leading to decreased performance and increased resource utilization.

(*) eg: consider a database of a bank with 2 accounts A and B.



→ Say T_1 wants to transfer 100 ₹ from A to B. It therefore needs to read and write both A and B
→ so it needs X-locks on both.

→ Say T_2 needs to transfer 40 ₹ from B to A. Similar scenario now leads to both T_1 and T_2 holding on to their primary resource



→ This leads to a simple deadlock.

Q.2 Example of a serializable schedule with two transactions such that the order in which the transactions commit is different from the serialization order.

→ Consider:

T_1	T_2
R(A)	
W(A)	
	R(B)
	W(B)
	Commit
R(B)	
W(B)	
Commit	

The serializable schedule would look like $T_1 \rightarrow T_2$

T_1 then T_2

so R(A) W(A) R(B) W(B) commit,
R(B) W(B) commit

But order of Transactions committing is T_2 before T_1 which is different from its serializable solution.

$T_1 \leftarrow T_2$

precedance graph allows serializability