Quiz on 2PL

Let's test our understanding of the two-phase locking mechanism.

Question 3

2PL guarantees serializability, but does its approach have any drawbacks/limitations?

Hide Answer

2PL is sufficient to ensure conflict serializability on its own. It creates schedules with an acyclic precedence graph. However, it is vulnerable to cascade aborts, which occur when one transaction fails, and the application must roll another back, causing a lot of wasted resources and effort.

3 of 3

Back

Analysis and Evaluatio...

Next

Introduction to Chubby

Mark as Completed

Question 1

Consider a transaction with the following operations: $read[a] \rightarrow write[b] \rightarrow commit$, follows this schedule for its execution:
$readlock[a],\ read[a],\ readunlock[a],\ writelock[b],\ write[b],\ writeunlock[b].$

Based on your understanding of the basic rules of 2PL, does the schedule above follow those rules?

Hide Answer

The schedule above violates the basic rule of 2PL that a transaction can acquire no lock once it has released even one lock. Since the schedule above states that the transaction $readunlocks[a]$ and then the transaction acquires a $writelock[b]$, it doesn't follow 2PL. A correct execution schedule can be as follows:
$readlock[a],\ read[a],\ writelock[b],\ write[b],\ readunlock[a],\ writeunlock[b].$

1 of 3

Question 2

Suppose there are three transactions, T1, T2, and T3, with timestamps 2, 4, and 6, respectively. How will the deadlock prevention techniques handle the following two cases?

1. T1 requesting an object held by T2
2. T3 requesting an object held by T2

Hide Answer

The two techniques will handle the requests in the following way.

1. Wait-die:
    - If T1 requests an object held by T2, then T1 will wait because old waits.
    - If T3 requests an object held by T2, then T3 will be terminated (die).
2. Wound-wait:
    - If T1 requests an object held by T2, then T2 will be aborted (wounded).
    - If T3 requests an object held by T2, then T3 will wait.

2 of 3

Question 3

2PL guarantees serializability, but does its approach have any drawbacks/limitations?

Hide Answer

2PL is sufficient to ensure conflict serializability on its own. It creates schedules with an acyclic precedence graph. However, it is vulnerable to cascade aborts, which occur when one transaction fails, and the application must roll another back, causing a lot of wasted resources and effort.