

Model question Paper

Section B

Question 5. Answer

Deadlock: Deadlock is an unwanted situation that arises in a shared resource environment, where a process indefinitely waits for a resource that is held by another process.

For example, assume a set of transactions $\{T_0, T_1, T_2, \dots, T_n\}$. T_0 needs a resource X to complete its task. Resource X is held by T_1 , and T_1 is waiting for a resource Y , which is held by T_2 . T_2 is waiting for resource Z , which is held by T_0 . Thus, all the processes wait for each other to release resources. In this situation, none of the processes can finish their task. This situation is known as a deadlock.

Prevention:

To prevent any deadlock situation in the system, the DBMS aggressively inspects all the operations, where transactions are about to execute. The DBMS inspects the operations and analyzes if they can create a deadlock situation. If it finds that a deadlock situation might occur, then that transaction is never allowed to be executed.

There are deadlock prevention schemes that use timestamp ordering mechanism of transactions in order to predetermine a deadlock situation. These are deadlocks :

Wait-Die Scheme

In this protocol, if a transaction requests to lock a resource (data item), which is already held with a conflicting lock by another transaction, then one of the two possibilities may occur –

- If $TS(T_i) < TS(T_j)$ – that is T_i , which is requesting a conflicting lock, is older than T_j – then T_i is allowed to wait until the data-item is available.
- If $TS(T_i) > TS(T_j)$ – that is T_i is younger than T_j – then T_i dies. T_i is restarted later with a random delay but with the same timestamp.

This scheme allows the older transaction to wait but kills the younger one.

Wound-Wait Scheme

In this protocol, if a transaction requests to lock a resource (data item), which is already held with conflicting lock by some another transaction, one of the two possibilities may occur –

- If $TS(T_i) < TS(T_j)$, then T_i forces T_j to be rolled back – that is T_i wounds T_j . T_j is restarted later with a random delay but with the same timestamp.

- If $TS(T_i) > TS(T_j)$, then T_i is forced to wait until the resource is available.

This protocol, allows the younger transaction to wait; but when an older transaction requests an item held by a younger one, the older transaction forces the younger one to abort and release the item.