Concurrency Control Protocols

Concurrency control protocols allow multiple transactions to happen while ensuring they are conflict/view serializable, recoverable, and sometimes cascadeless. These protocols enforce rules to prevent non-serializable schedules.

Types of Lock-Based Protocols

Simplistic Lock Protocol

The simplistic lock protocol requires that a transaction must obtain a lock on every data item it accesses before reading or writing. Once the transaction completes all its operations, it releases all the locks.

Characteristics:

Easy to implement with a simple rule: acquire a lock before accessing a data item.

Uses a single type of lock, limiting concurrent access to data items.

Advantages:

Simple to understand and implement.

Maintains data consistency by preventing concurrent access to the same data item.

Disadvantages:

Limits concurrency and can lead to performance bottlenecks.

Does not address deadlocks, where transactions wait indefinitely for each other.

Pre-Claiming Lock Protocol

The pre-claiming lock protocol mandates that a transaction must declare and obtain all the locks it will need before any operations are performed.

Characteristics:

A transaction must acquire all required locks at the beginning or wait until they are all available.

Helps prevent deadlocks by avoiding cyclic dependencies.

Advantages:

Prevents deadlocks by acquiring all locks upfront.

Straightforward to understand and implement.

Disadvantages:

Holding all locks can reduce concurrency, leading to potential performance issues.

Locks may be held longer than necessary, leading to inefficient resource use.

Two-Phase Locking (2PL)

The Two-Phase Locking protocol divides transaction execution into two phases: the growing phase and the shrinking phase.

Growing Phase: Transactions can acquire locks but cannot release any.

Shrinking Phase: Transactions can release locks but cannot acquire new ones.

Characteristics:

Guarantees that transactions are serializable.

Can lead to deadlocks, similar to simplistic protocols.

Advantages:

Preserves database consistency through serializability.

Ensures transactions do not interfere with each other.

Disadvantages:

Can lead to deadlocks.

Performance overhead due to locking phases.

Strict Two-Phase Locking (Strict-2PL)

This is a variant of the 2PL protocol where a transaction must hold all its exclusive locks until it commits or aborts.

Characteristics:

Exclusive locks are retained until the transaction completes.

Shared locks can be released before the transaction commits.

Advantages:

Prevents cascading aborts, ensuring consistency.

Simplifies recovery as uncommitted changes are not visible to other transactions.

Disadvantages:

Can lead to deadlocks, similar to basic 2PL.

Reduced concurrency due to holding exclusive locks.

Conclusion

Locking protocols are essential for ensuring consistency and isolation of transactions in a multi-user database. By understanding and properly implementing shared locks, exclusive locks, and various locking protocols like Two-Phase Locking, databases can effectively manage concurrent transactions, prevent conflicts, and maintain data integrity.