

Basics of Locking

EECS 484 Handout

Locking is a mechanism commonly used by systems to control access to shared resources by concurrently running users. In the context of a DBMS, these shared resources are data objects, and the users are transactions.

Locking is typically implemented using a lock manager, which records which objects are locked, by whom, and in what mode. When a transaction wishes to use a particular object (to read or write), it must request a lock from the lock manager. After it is done with the object, it releases the lock by again notifying the lock manager. In certain cases, the lock manager is not able to immediately grant a lock when it is requested (e.g., if it is held by another transaction). In this case, the lock manager maintains a queue of transactions waiting for the lock.

It is important to also recognize that some data items can be shared simultaneously between transactions (e.g., transactions T1 and T2 both want to read object X), but in other cases it is necessary for a transaction to have an exclusive lock (e.g., T1 wants to write to X). This motivates the need for multiple lock modes. In this case, the idea is that, if a transaction requests a lock on an object in a mode that is incompatible with an existing lock on that object, then it must wait on the lock queue until the existing lock is released. For shared (S) and exclusive (X) lock modes, the following table indicates compatibility:

	S	X
S	Yes	No
X	No	No

In a DBMS, the goal is to develop a locking protocol that guarantees a schedule with desirable properties (e.g., serializability, recoverability, avoid cascading aborts). Two common protocols are two-phase locking (2PL) and strict two-phase locking (Strict 2PL).

Strict Two-Phase Locking (Strict 2PL)

1. If a transaction T wants to read object X, it requests a shared lock on X. If it wants to write X, it requests an exclusive lock.
2. All locks requested by a transaction are held until the transaction is completed (commits or aborts), at which point the locks are released.

It can be shown that Strict 2PL guarantees schedules that are serializable, recoverable, and that avoid cascading aborts.

Two-Phase Locking (2PL)

2PL relaxes Strict 2PL slightly. A transaction need not hold all locks until completion, but once it has released a lock, it may not request any more locks. 2PL is guaranteed to produce schedules that are serializable.