**1. Define a transaction. Bring out the desirable / ACID properties of a transaction ?**

**Transaction**:Logical unit of database processing that includes one or more access operations (read -retrieval, write - insert or update, delete).A transaction (set of operations) may be stand-alone specified in a high level language like SQL submitted interactively, or may be embedded within a program.

**ACID Properties of a transaction**

1. **Atomicity**: A transaction is an atomic unit of processing; it is either performed in its entirety or not performed at all.

– Transaction recovery subsystem ensures atomicity

1. **Consistency preservation**: A correct execution of the transaction must take the database from one consistent state to another.

– Programmers must ensure consistency preservation.

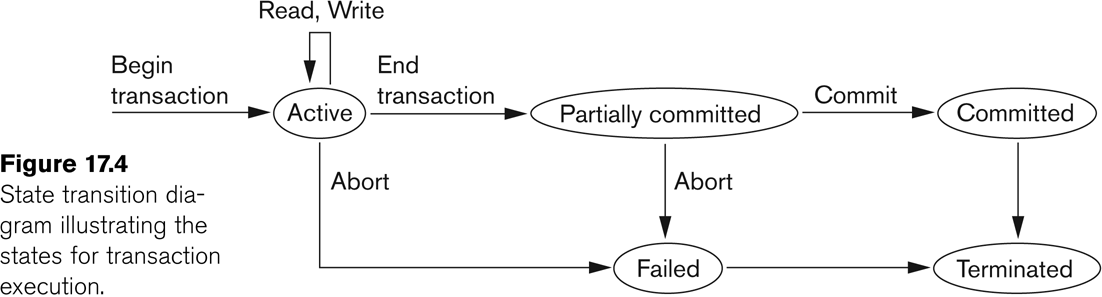
1. **Isolation**: A transaction should not make its updates visible to other transactions until it is committed; this property, when enforced strictly, solves the temporary update problem and makes cascading rollbacks of transactions unnecessary.

– Concurrency control subsystem ensures atomicity

1. **Durability or permanency**: Once a transaction changes the database and the changes are committed, these changes must never be lost because of subsequent failure.

– Transaction recovery subsystem ensures atomicity

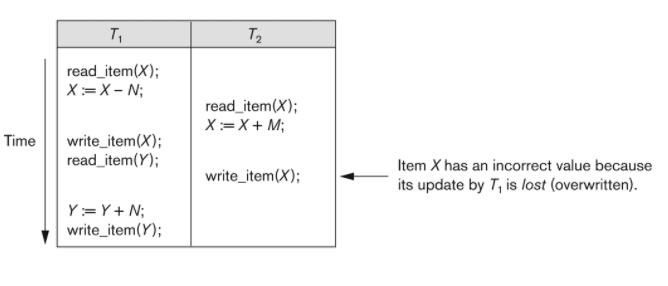
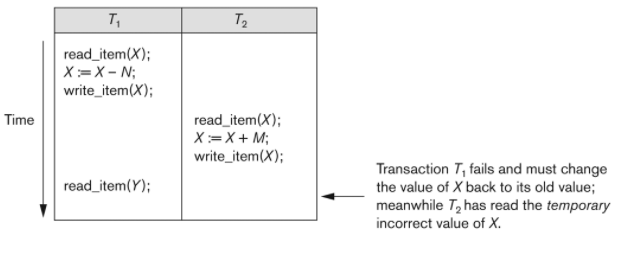
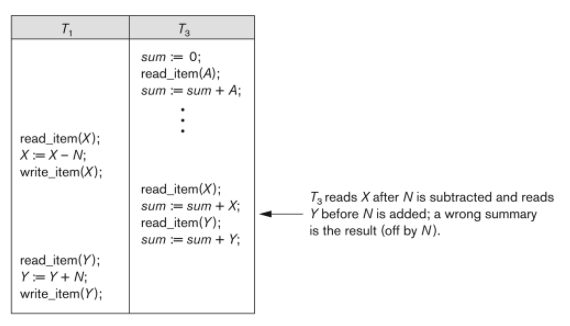
**2. With a neat diagram, write and explain the state transition diagram illustrating the states for transaction execution.**

****

Transaction states:

1. **Active state**
   1. A transaction goes into an active state immediately after it starts execution, where it can issue READ and WRITE operations.
2. **Partially committed state**
   1. When the transaction ends,it moves to the partially committed state. At this point, some recovery protocols need to ensure that a system failure will not result in an inability to record the changes of the transaction permanently
3. **Committed state**
   1. Once this check is successful, the transaction is said to have reached its commit point and enters the committed state.
   2. Once a transaction is committed, it has concluded its execution successfully and all its changes must be recorded permanently in the database.
4. **Failed state**
   1. However, a transaction can go to the failed state if one of the checks fails or if the transaction is aborted during its active state. The transaction may then have to be rolled back to undo the effect of its WRITE operations on the database.
5. **Terminated State** 
   1. The terminated state corresponds to the transaction leaving the system. The transaction information that is maintained in system tables while the transaction has been running is removed when the Transaction terminates.

**Explain the problems which may encounter when transactions run concurrently with examples**

* **The Lost Update Problem**
  + This occurs when two transactions that access the same database items have their operations interleaved in a way that makes the value of some database item incorrect.
  + 
* **The Temporary Update (or Dirty Read) Problem** 
  + This occurs when one transaction updates a database item and then the transaction fails for some reason (see Section 17.1.4).
  + The updated item is accessed by another transaction before it is changed back to its original value.
  + 
* **The Incorrect Summary Problem**
  + If one transaction is calculating an aggregate summary function on a number of records while other transactions are updating some of these records, the aggregate function may calculate some values before they are updated and others after they are updated.
  + 
* **The Unrepeatable Read Problem.** 
  + where a transaction T reads the same item twice and the item is changed by another transaction T between the two reads. Hence, T receives different values for its two reads of the same item.
  + This may occur, for example, if during an airline reservation transaction, a customer inquires about seat availability on several flights. When the customer decides on a particular flight, the transaction then reads the number of seats on that flight a second time before completing the reservation, and it may end up reading a different value for the item.

**Write a note on binary locks and explain with an example.**

Binary Locks. A binary lock can have two states or values: locked and unlocked (or 1 and 0, for simplicity).

A distinct lock is associated with each database item X.

If the value of the lock on X is 1, item X cannot be accessed by a database operation that requests the item.

If the value of the lock on X is 0, the item can be accessed when requested, and the lock value is changed to 1.

We refer to the current value (or state) of the lock associated with item X as lock(X). Two operations, lock\_item and unlock\_item, are used with binary locking.

A transaction requests access to an item X by first issuing a lock\_item(X) operation. If LOCK(X) = 1, the transaction is forced to wait.

If LOCK(X) = 0, it is set to 1 (the transaction locks the item) and the transaction is allowed to access item X.

When the transaction is through using the item, it issues an unlock\_item(X) operation, which sets LOCK(X) back to 0 (unlocks the item) so that X may be accessed by other transactions.

**What is two phase protocol? How does it guarantee serializability?**

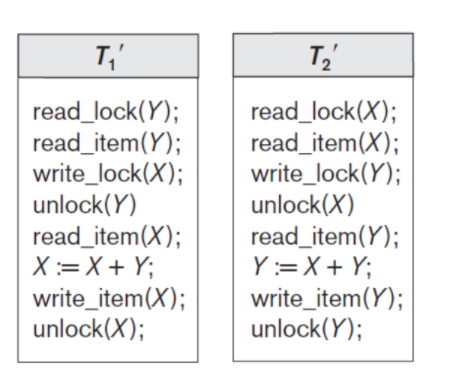
➢A transaction is said to follow the two-phase locking protocol if all locking operations

(read\_lock, write\_lock) precede the first unlock operation in the transaction.

➢Such a transaction can be divided into two phases:

**expanding or growing (first) phase**, during which new locks on items can be acquired but none can be released;

**shrinking(second) phase**, during which existing locks can be released but no new locks can be acquired.



➢If lock conversion is allowed, then upgrading of locks (from read-locked to write-locked) must be done during the expanding phase, and downgrading of locks (from write-locked to read-locked) must be done in the shrinking phase. Hence, a read\_lock(X) operation that downgrades an already held write lock on X can appear only in the shrinking phase.

Serializability is mainly an issue of handling write operation. Because any inconsistency may only be created by write operation. Multiple reads on a database item can happen parallely. 2-Phase Locking protocol restricts this unwanted read/write by applying exclusive lock. Moreover, when there is an exclusive lock on an item it will only be released in shrinking phase. Due to this restriction there is no chance of getting any inconsistent state.

**Consider a database with objects X and Y and assume that there are two transactions T1 and T2. Transaction T1 reads objects X and Y and then writes object X. Transaction T2 reads objects X and Y and then writes objects X and Y.**

(**a) Give an example schedule with actions of transactions T1 and T2 on objects X and Y that results in a write-read conflict.**

**(b) Give an example schedule with actions of transactions T1 and T2 on objects X and Y that results in a read-write conflict.**

**(c) Give an example schedule with actions of transactions T1 and T2 on objects X and Y that results in a write-write conflict.**

**(d) For each of the three schedules, show that Strict 2PL disallows the schedule.**

Answer

Given that

T1 = r1(X), r1(Y), w1(X) T2 = r2(X), r2(Y), w2(X), w2(Y)

we design the following schedules which have the asked conflicts.

(a) Write-read conflict (which necessarily means reading uncommitted data, aka dirty read which results in cycle in the precedence graph):

S1 = r2(X), r2(Y),**w2(X), r1(X),** r1(Y), w1(X), w2(Y)

The WR conflict is written in bold. Note that, it is not necessary to have the W and R statement immediately after each other to create a WR conflict.

(b) Read-write conflict (which necessarily means unrepeatable read which causes cycle in the precedence graph):

S2 = r2(X), r2(Y), w2(X), r1(X), **r1(Y), w2(Y)**, w1(X)

The RW conflict is written in bold.

(c) Write-write conflict (which necessarily means overwriting uncommitted data or blind writes resulting in cycle in the precedence graph):

S3 = r2(X), r2(Y), r1(X), r1(Y), **w2(X), w1(X**), w2(Y)

The WW conflict is written in bold.

(d) If we use strict 2PL the locking and unlocking will be done in two phases and all the exclusive locks will be held till the transaction commits or aborts and shared locks can be released any time during the second phase. If we apply strict 2PL, only serializable schedules will be allowed and all the three schedules listed above will be disallowed. Let us denote taking exclusive lock by x and shared lock by s, and release of lock by u, and commit by c. So, following will be the execution of the schedules where strict 2PL will ensure serializability by not granting locks which may create conflicts.

S1 = s2(X), r2(X), s2(Y), r2(Y), x2(X), w2(X), s1(X)-request not granted, x2(Y), w2(Y), u2(X), u2(Y), c2, s1(X), r1(X), s1(Y), r1(Y), x1(X), w1(X), u1(X), u1(Y), c1.

S2 = s2(X), r2(X), s2(Y), r2(Y), x2(X), w2(X), s1(X)-request not granted, x2(Y), w2(Y), u2(X), u2(Y), c2, s1(X), r1(X), s1(Y), r1(Y), x1(X), w1(X), u1(X), u1(Y), c1.

S3 = s2(X), r2(X), s2(Y), r2(Y), s1(X)-request not granted, x2(X), w2(X), x2(Y), w2(Y), u2(X), u2(Y), c2, s1(X), r1(X), s1(Y), r1(Y), x1(X), w1(X), u1(X), u1(Y), c1.

**Types of data updates in DBMS.**

Immediate Update: As soon as a data item is modified in cache, the disk copy is updated.

Deferred Update: All modified data items in the cache is written either after a transaction ends its execution or after a fixed number of transactions have completed their execution.

Shadow update: The modified version of a data item does not overwrite its disk copy but is written at a separate disk location.

In-place update: The disk version of the data item is overwritten by the cache version.

**Why concurrency control is needed? Discuss the types of problems that**

**might be encountered if transactions are allowed to run concurrently.**

Why Concurrency Control is needed:

The Lost Update Problem

◦ This occurs when two transactions that access the same database items have their operations

interleaved in a way that makes the value of some database item incorrect.

The Temporary Update (or Dirty Read) Problem

◦ This occurs when one transaction updates a database item and then the transaction fails for some

reason.

◦ The updated item is accessed by another transaction before it is changed back to its original value.

The Incorrect Summary Problem

◦ If one transaction is calculating an aggregate summary function on a number of records while

other transactions are updating some of these records, the aggregate function may calculate some

values before they are updated and others after they are updated.

**Write the procedure for recovery based on immediate update in multiple user environment.**

**Describe the procedure of two phase locking? Will two phase locking result in serializable schedule? Will two phase locking result in deadlock? Demonstrate with an example.**

**What is the system log used for? What are the typical kinds of records in a system log? What are transaction commit points, and why are they important?**

**The System Log**

◦ Log or Journal: The log keeps track of all transaction operations that affect the values

of database items.

◦ This information may be needed to permit recovery from transaction failures.

◦ The log is kept on disk, so it is not affected by any type of failure except for disk or

catastrophic failure.

◦ In addition, the log is periodically backed up to archival storage (tape) to guard against

such catastrophic failures.

◦ T in the following discussion refers to a unique transaction-id that is generated

automatically by the system and is used to identify each transaction:

**◦ Types of log record:**

◦ [start\_transaction,T]: Records that transaction T has started execution.

◦ [write\_item,T,X,old\_value,new\_value]: Records that transaction T has changed the value

of database item X from old\_value to new\_value.

◦ [read\_item,T,X]: Records that transaction T has read the value of database item X.

◦ [commit,T]: Records that transaction T has completed successfully, and affirms that its

effect can be committed (recorded permanently) to the database.

◦ [abort,T]: Records that transaction T has been aborted.

**Commit Point:**

◦ A transaction T reaches its commit point when all its operations that access the

database have been executed successfully and the effect of all the transaction

operations on the database has been recorded in the log.

◦ Beyond the commit point, the transaction is said to be committed, and its effect is

assumed to be permanently recorded in the database.

◦ The transaction then writes an entry [commit,T] into the log.

**Write the procedure for recovery based on immediate update in multiple user environment.**

Immediate update –

In the immediate update, the database may be updated by some operations of a transaction before the transaction reaches its commit point. However, these operations are recorded in a log on disk before they are applied to the database, making recovery still possible. If a transaction fails to reach its commit point, the effect of its operation must be undone i.e. the transaction must be rolled back hence we require both undo and redo. This technique is known as undo/redo algorithm.

**Consider a database with objects X and Y and assume that there are two transactions T1 and T2. Transaction T1 reads objects X and Y and then writes object X . Transaction T2 reads objects X and Y and then writes objects X and Y.**

**i) Give an example schedule with actions of transactions T1 and T 2 on objects X and Y that results in a write-read conflict.**

**ii) Give an example schedule with actions of transactions T1 and T 2 on objects X and Y that results in a read-write conflict.**

**iii) Give an example schedule with actions of transactions T1 and T2 on objects X and Y that results in a write-write conflict.**

**iv) For each of the three schedules, show that Strict 2PL disallows the schedule.**