***Properties of Transaction***

***DESIRABLE PROPERTIES OF TRANSACTION***

**Transactions should possess several properties. These are often called the ACID properties, and they should be enforced by the concurrency control and recovery methods of the DBMS. The following are the ACID properties:**

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

**The atomicity property requires that we execute a transaction to completion. It is the responsibility of the transaction recovery subsystem of a DBMS to ensure atomicity. If a transaction fails to complete for some reason, such as a system crash in the midst of transaction execution, the recovery technique must undo any effects of the transaction on the database.**

**Eg:**

**Consider the case of funds transfer from account A to account B.**

**A.bal -= amount;**

**B.bal += amount;**

**A.bal -= amount;**

**CRASH**

**…**

**…**

**RECOVERY**

**A.bal += amount; -- Rollback**

**2. Consistency preservation: A transaction is consistency preserving if its complete execution take(s) the database from one consistent state to another.**

**The preservation of consistency is generally considered to be the responsibility of the programmers who write the database programs or of the DBMS module that enforces integrity constraints. Recall that a database state is a collection of all the stored data items (values) in the database at a given point in time. A consistent state of the database satisfies the constraints specified in the schema as well as any other constraints that should hold on the database. A database program should be written in a way that guarantees that, if the database is in a consistent state before executing the transaction, it will be in a consistent state after the complete execution of the transaction, assuming that no interference with other transactions occurs**

**Eg:**

**Consider the case of funds transfer from account A to account B.**

**A.bal -= amount;**

**B.bal += amount;**

**B.bal += amount;**

**A.bal -= amount (FAILS!! A’s balance is 0)**

**B.bal -= amount; -- Rollback**

**3. Isolation: A transaction should appear as though it is being executed in isolation from other transactions. That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently.**

**Isolation is enforced by the concurrency control subsystem of the DBMS. If every transaction does not make its updates visible to other transactions until it is committed, one form of isolation is enforced that solves the temporary update problem and eliminates cascading rollbacks. There have been attempts to define the level of isolation of a transaction. A transaction is said to have level 0 (zero) isolation if it does not overwrite the dirty reads of higher-level transactions. A level 1 (one) isolation transaction has no lost updates; and level 2 isolation has no lost updates and no dirty reads. Finally, level 3 isolation (also called true isolation) has, in addition to degree 2 properties, repeatable reads.**

**Eg:**

**Consider the case of funds transfer from account A to account B.**

**Transaction T1:**

**A.bal -= amount; (Let A’s balance become 0 after this…)**

**B.bal += amount;**

**Transaction T2:**

**A.bal -= amount2;**

**Net effect should be either T1,T2 (in which case T2 fails) or**

**T2,T1 (in which case T1 fails)**

**4. Durability or permanency: The changes applied to the database by a committed transaction must persist in the database. These changes must not be lost because of any failure.**

**Finally, the durability property is the responsibility of the recovery subsystem of the DBMS.**

**Eg:**

**Consider the case of funds transfer from account A to account B.**

**Account A should have a balance of amount**

**Transaction T1:**

**A.bal -= amount;**

**B.bal += amount;**

**Commit**

**Account A should have a balance of 0**