Paper Name: Database Management Systems

Topic: Transaction Management

Paper Code: BCA CC410

By Ms. Manisha Prasad

Head, Assistant Professor,

Department of Computer Science

Patna Women's College

TRANSACTION MANAGEMENT

A transaction is a set of logically related operations performing one task. For example, you are transferring money from your bank account to your friend's account, the set of operations would be like this:

1. Read your account balance
2. Deduct the amount from your balance
3. Write the remaining balance to your account
4. Read your friend's account balance
5. Add the amount to his account balance
6. Write the new updated balance to his account

This whole set of operations can be called a transaction. Although I have shown you read, write and update operations in the above example but the transaction can have operations like read, write, insert, update, delete.

In DBMS, we write the above 6 steps as follows:-
Lets say your account is A and your friend's account is B, you are transferring 10000 from A to B, the steps of the transaction are:

1. R(A);
2. A = A - 10000;
3. W(A);
4. R(B);
5. B = B + 10000;
6. W(B);

In the above transaction **R** refers to the **Read operation** and **W** refers to the **Write operation**.

Problems associated with Transactions

The main problem that can happen during a transaction is that the transaction can fail before finishing the all the operations in the set. This can happen due to power failure, system crash etc. This is a serious problem that can leave database in an inconsistent state. Assume that transaction fail after third operation (see the example above) then the amount would be deducted from your account but your friend will not receive it.

To solve this problem, we have the following two operations

Commit: If all the operations in a transaction are completed successfully then commit those changes to the database permanently.

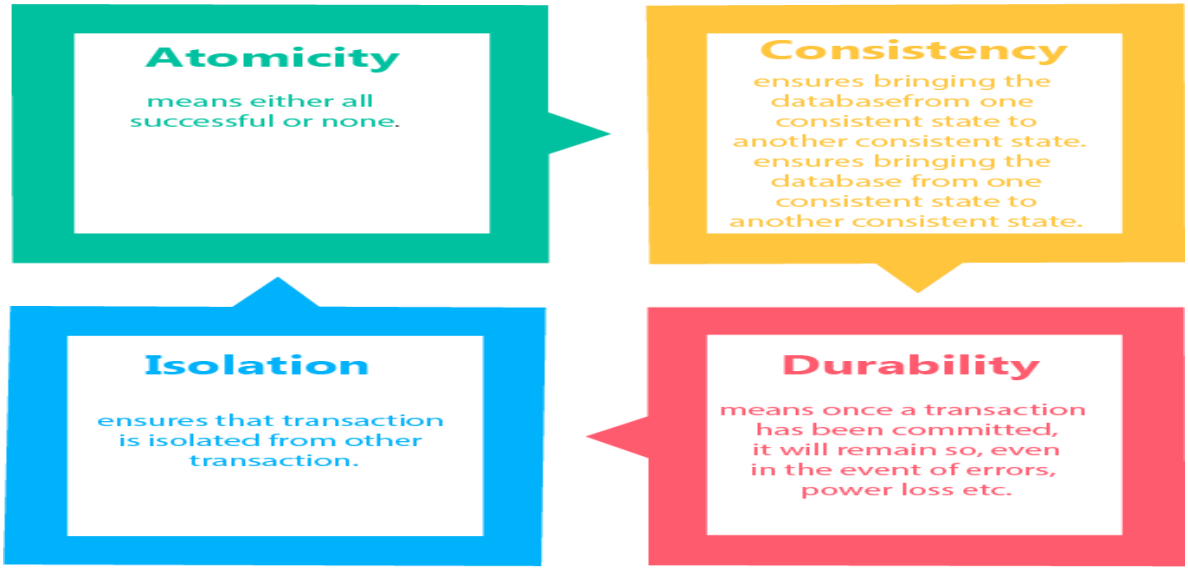
Rollback: If any of the operation fails then rollback all the changes done by previous operations.

Even though these operations can help us avoiding several issues that may arise during transaction but they are not sufficient when two transactions are running concurrently. To handle those problems we need to understand database **ACID properties**.

Transaction property

The transaction has the four properties Known as ACID property. These are used to maintain consistency in a database, before and after the transaction.

1. **A**-Atomicity
2. **C**-Consistency
3. **I**-Isolation
4. **D**-
   Durability



Atomicity

It states that all operations of the transaction must take place at and if not, the transaction is aborted. There is no midway, i.e., the transaction cannot occur partially. Each transaction is treated as one unit which should either run to completion or should not execute at all. Atomicity involves the following two operations:

Abort: If a transaction aborts then all the changes made are not visible.

Commit: If a transaction commits then all the changes made are visible.

Example: Let's assume that following transaction T consisting of T1 and T2. Let A account has Rs 600 and B has Rs 300. Transfer Rs 100 from account A to account B.

T1	T2
Read(A) A:= A-100 Write(A)	Read(B) Y:= Y+100 Write(B)

After the completion of the transaction T, A will have Rs 500 and B will have of Rs 400.

If the transaction T fails after the completion of transaction T1, but before completion of transaction T2, then the amount will be deducted from A but not added to B. This shows

the inconsistent database state. In order to ensure correctness of database state, the transaction must be executed in entirety.

Consistency

The integrity constraints are maintained so that the database is consistent before and after the transaction.The execution of a transaction will leave a database in either its prior stable state or a new stable state. The transaction is used to transform the database from one consistent state to another consistent state.

For example: The total amount must be maintained before or after the transaction.

1. Total before T occurs = 600+300=900
2. Total after T occurs= 500+400=900

Therefore, the database is consistent. In the case when T1 is completed but T2 fails, then inconsistency will occur.

Isolation

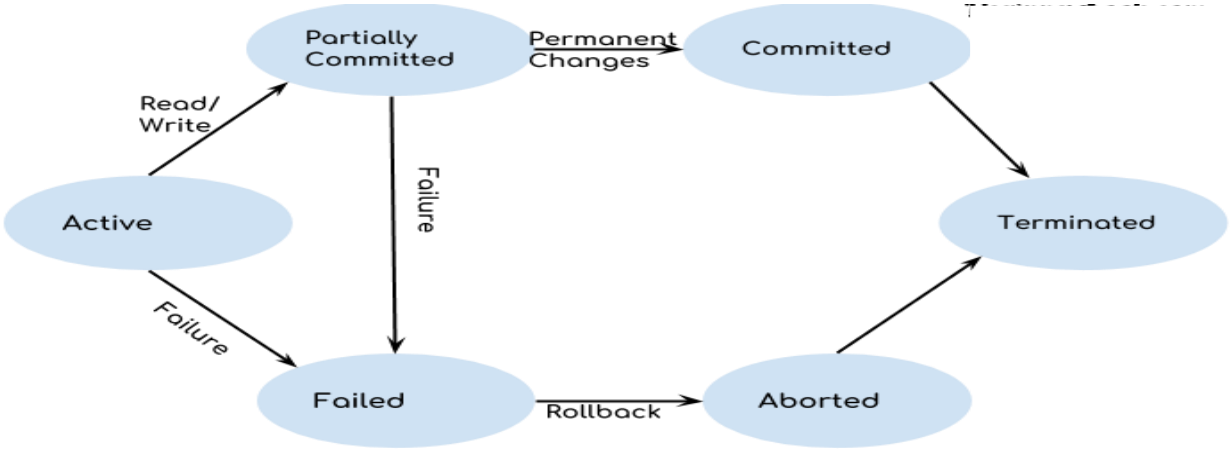
It shows that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.In isolation, if the transaction T1 is being executed and using the data item X, then that data item can't be accessed by any other transaction T2 until the transaction T1 ends. The Concurrency Control subsystem of the DBMS enforces the isolation property.

Durability

The durability property is used to indicate the performance of the database's consistent state. It states that the transaction made the permanent changes. They cannot be lost by the erroneous operation of a faulty transaction or by the system failure. When a transaction is completed, then the database reaches a state known as the consistent state. That consistent state cannot be lost, even in the event of a system's failure. The Recovery Management subsystem of the DBMS has the responsibility of Durability property.

Transaction States

A transaction in DBMS can be in one following states:-



DBMS Transaction States Diagram

Active State

A transaction is a sequence of operations. If a transaction is in execution then it is said to be in active state. It doesn't matter which step is in execution, until unless the transaction is executing, it remains in active state.

Failed State

If a transaction is executing and a failure occurs, either a hardware failure or a software failure then the transaction goes into failed state from the active state.

Partially Committed State

As we can see in the above diagram that a transaction goes into "partially committed" state from the active state when there are read and write operations present in the transaction.A transaction contains number of read and write operations. Once the whole transaction is successfully executed, the transaction goes into partially committed state where we have all the read and write operations performed in the local memory(or buffer) instead of the actual database. The reason why we have this state is because a transaction can fail during execution so if the changes are made in the actual database instead of local memory, database may be left in an inconsistent state in case of any failure. **This state helps to rollback the changes made to the database in case of a failure during execution.**

Committed State

If a transaction completes the execution successfully then all the changes made in the local memory during **partially committed** state are permanently stored in the database. It is evident in the above diagram that a transaction goes from partially committed state to committed state when everything is successful.

Aborted State

If a transaction fails during execution then the transaction goes into a failed state. The changes made into the local memory (or buffer) are rolled back to the previous consistent state and the transaction goes into aborted state from the failed state