unit-8

Transaction and Query processing

Introduction to Transaction:

The concept of transaction provides a mechanism for describing logical units of database processing, Transaction in DBMS are sets of operations performed to modify data, including insertion, updates, or deletions. Transactions in DBMS ensures the integrity of the database.

A transaction generally represents change in database.

we have different types of operations relating to a transaction. They are as follow:

(a) Read (x)

A read operation is used to read the value of 'x' from the database and store it in a buffer in the main memory for further action.

(b) Write (x)

A write operation is used to write the value back to the database from the buffer.

Eg:
1. R(x); // Read x
2. x = x - 500; // Deduct withraw value from x
3. W(X); // Write back to database.

let's initially value of X is 10000. i.e X =10000

- The first operation reads X's value from database and store it in buffer.
- The second operation will decrease the value of X by 500. so, buffer will contain 9500.
- The third operation will write the buffer's value to the database. So, X's final value will be 9500.

But it may be possible that because of the failure of hardware, software or power, etc. that transaction may fail before finished all the operations in the set.

In the above example (transaction), the debit transaction fails after executing operation 2 the X's value will remain 10,000 in the database which is not acceptable by the bank.

To solve this problem, we have two important operations: Commit & Rollback

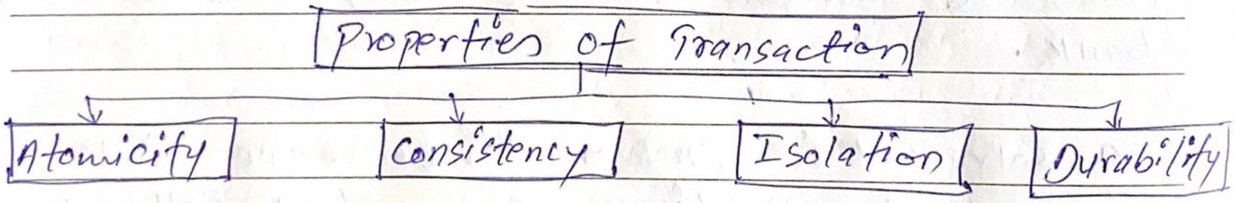
(C) Commit

To ensure that further operations of any other transaction are performed only after work of the current transaction is done, a commit operation is performed to the changes made by a transaction permanently to the database.

(d) Rollback:
This operation is interrupted in between due to any power, hardware, or software failure. It can be said that a rollback operation does undo the operations of transactions that were performed before its interruption to achieve a safe state of the database and avoid any kind of ambiguity or inconsistency.

We can say that
commit: is used to save the work done permanently.
Rollback: It is used to undo the work done.

Transactions ensure the integrity of the database by the following ACID properties:
i.e. ACID (Atomity, Consistency, Isolation, Durability)



1. Atomicity: By this, we mean that either the entire transaction takes place at once or doesn't happen at all. There is no midway i.e. transactions do not occur partially. Each transaction is considered as one unit and either runs to completion or is not executed at all. It involves the following two operations.

→ abort : If a transaction aborts, changes made to the database are not visible.

→ Commit: If a transaction commits, changes made are visible.

Transaction (T1)	Transaction (T2)
X = 500	Y = 200
Read (X)	Read(Y)
X = X - 100	Y = Y+100
write (X)	write(Y)
After: X = 400	After: Y = 300

If the transaction fails after completion of (T1) but before completion of T2 (after write(X) but before write (Y)), then the amount has been deducted from X but not added to Y. This results in an inconsistent database state. Therefore, the transaction must be executed in its entirety in order to ensure the correctness of the database state.

2. 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.

In the above example of transaction (T1 & T2)

Total sum before Transaction

$= X + Y$

$= 500 + 200 = 700$ ←

no loss

Total sum after transaction

$= X + Y$

$= 400 + 300 = 700$ ←

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

3. Isolation: It shows that the data which is used at the time of execution of a transaction can't be used by the second transaction until the first one is completed. This property ensures that multiple transactions can occur concurrently without leading to the inconsistency of the database state. Transaction occur independently without interference.

Eg. - X wants to transfer Rs. 100 from a/c 12345 to account 45678
- Y wants to withdraw Rs. 50 from 45678.

Isolation ensures that

*) Y's transaction can't see the changes made by X's transaction until X's transaction is committed.

*) Similarly, X's transaction can't see the changes made by Y's transaction until Y's transaction is committed.

We can conclude that, no transaction will affect the existence of any other transaction.

4. Durability: This property ensures that once the transaction has completed execution, the updates and modifications to the database are stored in and written to disk and they persist even if a system failure occurs. These updates now become permanent and are stored in non-volatile (i.e. Harddisk) memory.
Or we can say that, durability guarantees that the changes made by the Committed transaction are not lost, even in the event of a system failure.

Transaction states in DBMS :

In DBMS, a transaction

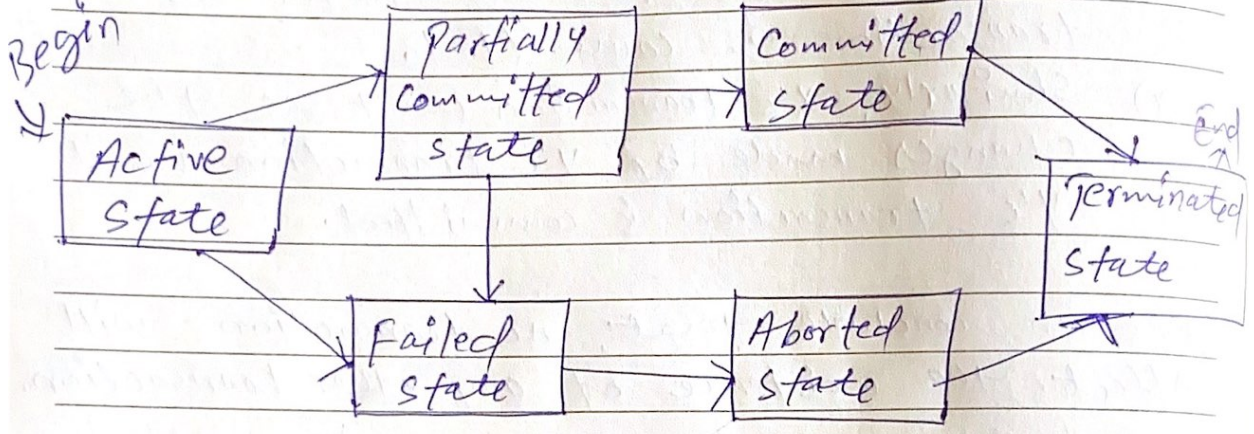


Fig: Transaction state in DBMS

(1.) Active state: when the instruction are running then the transaction is in active state. If all the 'read and write' operations are performed without any error then it goes to the "partially committed state". If any instruction fails, it goes to the "failed state".

(2.) Partially Committed: After completion of all the read and Write operation the changes are made in main memory or local buffer. If the changes are made permanent ~~men nently~~ on the Database then the state will change to "Committed State" and in case of failure it will go to the "failed state".

Failed state:

(3) If an error occurs during the execution of a transaction, it enters the failed state. In this state, the transaction can't proceed further, and the changes made by it are typically rolled back to maintain database consistency.

(4) Committed state: It is the state when the changes are made permanent on the Database and the transaction is complete and therefore terminated in the "terminated state".

(5) Aborted state: After a transaction has failed, it enters the aborted state. In this state, any changes made by the transaction are undone (rolled back), and the database returns to its state before the transaction began.

(6) Terminated: This is the final state of a transaction. once a transaction has been committed or aborted, it enters the terminated state, signifying that its execution is complete.