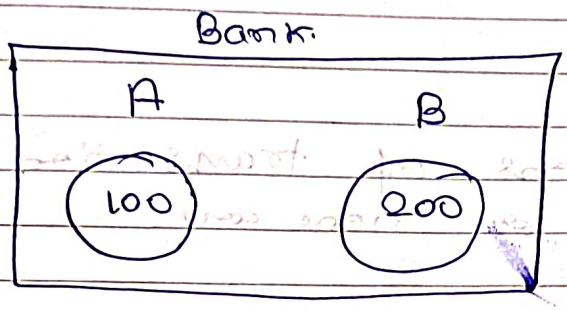
ACid Properties & Transactions in DBMS

Transactions :-



- Task 1 :- ₹50 from Bank A/c A to A/c B.

```
read (A)
A = A-50;
Write (A);
read (B)
B = B+50;
Write (B)
```

→ All 6 steps must have to be atomic. (They either should complete or ~~re~~ revert / rollback.

Transaction

1. A unit of work done against the DB in a logical sequence.
2. Sequence is very important in transaction.

Acid Properties

To ensure integrity of the data, we require that the DB system maintain the following below properties.

(1) Atomicity

Either all operations of transaction are reflected properly or none are.

(2) Consistency

Integrity constraints must be maintained before and after transaction.

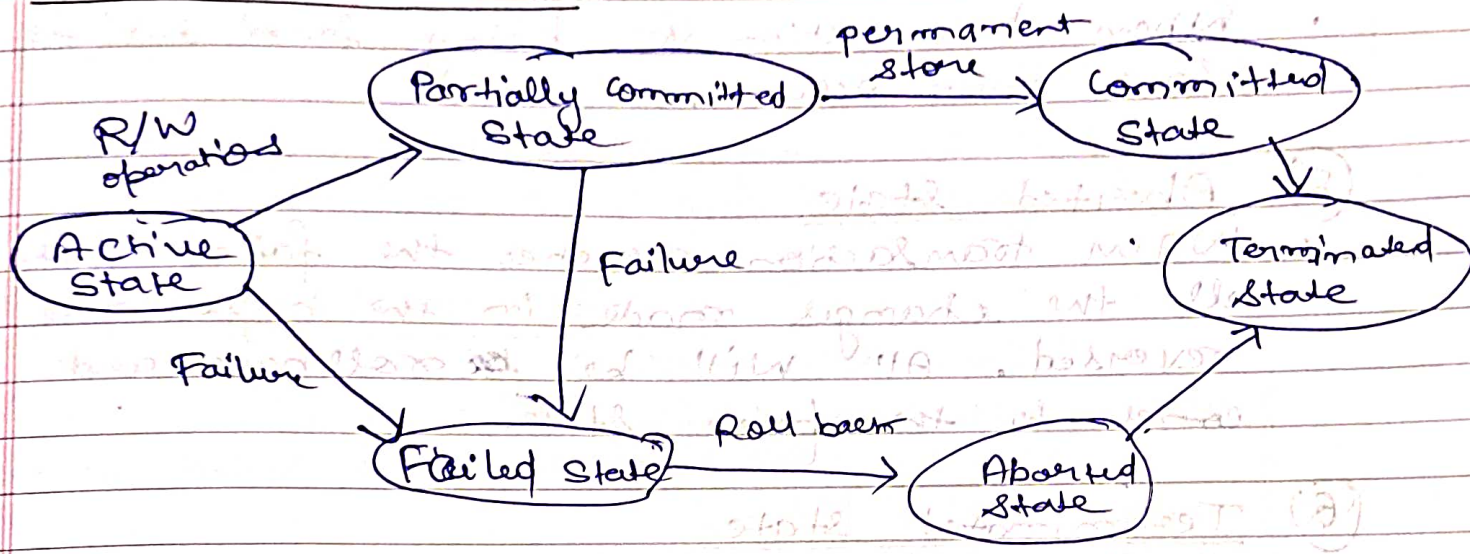
(3) Isolation

If multiple transactions are happening at the same time, the system guarantees that, for every pair transactions A and B, either A finished execution before B or B finished execution and then A goes. Each transaction is unware of other transactions. [But other waits if one transaction is going].

(4) Durability

After completion of transaction, the changes it has made to the database will be permanent.

- Transaction states



① Active state

- The first state of the life-cycle of the transaction, all the read and write operations are being performed here. If all goes fine then it comes to "partially committed state" or if it fails it comes to "failed state".

② Partially committed state

- After transaction is executed the changes are saved in the buffer of the main memory. If changes are permanent on the DB then the state will transfer to the committed state or if it fails, it will go to the failed state.

③ Committed state

- When updates are made permanent on the DB. Then the transaction is said to be in the committed state. Rollback can't be done from the committed states.

④ Failed State

- When transaction is being failed due to some error occurs.

⑤ Aborted State

- When transaction reaches the failed state, all the changes made in the buffer are reversed. All will be rollback and comes to the prior state.

⑥ Terminated State

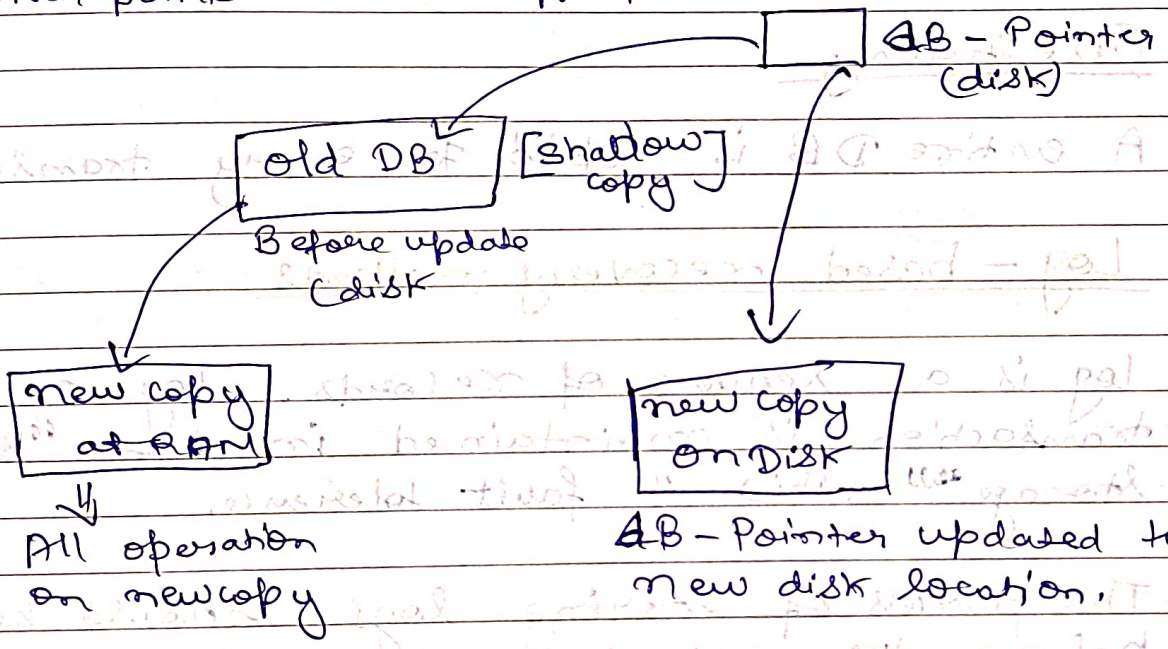
- A transaction is said to have terminated if it has either committed or aborted.

How to implement Atomicity and durability in Transactions

- Recovery Mechanism component of DBMS supports atomicity and durability.

(1) Shadow-Copy Scheme

- Based on making copies of DB.
- A pointer called db-pointer is maintained on disk, which points to current copy of DB.



- Transaction T, wants to update DB first creates a complete copy of DB.
- All further updates are done on new DB copy leaving original copy (Shadow copy) untouched.
- If T success, the system updates the db-pointer to point to the new copy of DB.
- If T fails, before db-pointer update, the old content of DB are not affected.

Note :-
Here the db-pointer is atomic as it lies entirely in a single sector.

- **Atomicity**
- original content of DB does not affect if T fail
- **Durability**
- If system fails before db-pointer update, the original content of DB does not affected. If it fails after the db-pointer update, all the operations are saved on the disk.
- **Inefficiency**

A entire DB is copied for every transaction.

(2) Log-based recovery methods

- log is a sequence of records. log of each transaction is maintained in some "Stable Storage" which is fault tolerence.
- The process of storing logs should be done before the actual transaction is applied in the database.
- We can implement this system by two ways :-

(1) Deferred DB Modifications.

- Here in this system we write logs after we perform that part of transaction and then we do changes in the DB.

- After all parts complete, we write the commit in the log. Then after we write it to the DB. (That's why it is deffered or delay writing).

- E.g. we do transaction from A/C A (₹100) of ₹50 to a/c B (₹200).

A	B	logs
₹100	₹200	
read (A)		< To Start>
A = A-50		< To, A, 50 >
Write (A)		
B = B+50		< To, B, 250 >
Write (B)		< To commit>

- (1) If System crashed before transaction completes, the information in logs are ignored.
- If transaction completes, we do the deffered writes.
- If failure occur while updating the DB, we perform redo by using logs.

(2) Immediate DB Modifications.

- Here transaction steps performs, we write logs and also updating it to the DB in the same sequence.
- Here in the log files, we also store the

previous values of transaction to use it if system crashes ; to restore to its previous state.

<To Start>
<To, A, 100, 50>
<To, B, 200, 250>
<To Commit>

(3) Check points

- A declare point where DBMS was in a consistent state and all transactions were committed.
- Upon reaching the checkpoint, the log file is destroyed by saving its update to the database.
- The new log file is created with upcoming execution operations of the transaction.

(8) Why need checkpoints ?

⇒ Whenever transaction logs are created in real-time environment, it eats up lots of storage space.