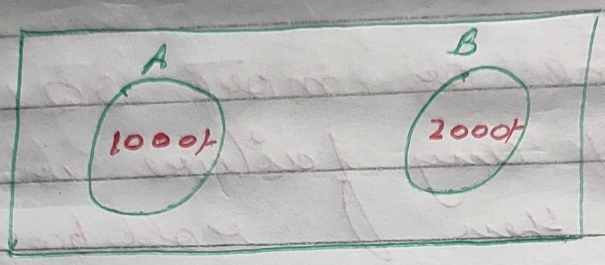
Date: _____ Page no: ___

Lecture = 12

Transactions

ABC Bank



→ ~~From~~ Task :- ₹50 from A to B

(i) Read (A)
(ii) A = A - 50;
(iii) Write (A)
(iv) Read (B)
(v) B = B + 50;
(vi) Write (B);

← 6 Logical Steps = 1 Transaction

Atomic
↳ they are considered to be a single task.
i.e. if any error in any one step
→ aborted

Date: Page no:

Transaction:- A unit of work done against the DB in a logical sequence.

→ one or more SQL statements.
→ If any failure in any step then rollback.
→ Either complete, or either aborted. (changes done or undone)

Acid Properties :-

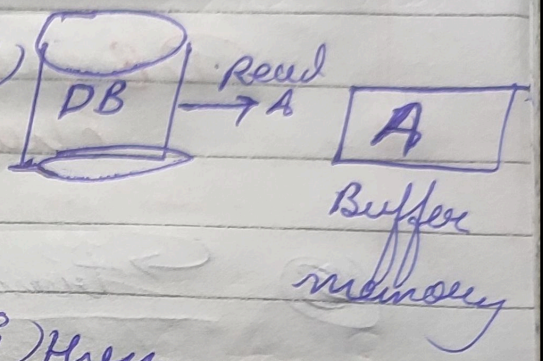
→ (1) To ensure data integrity, we need that DB maintains following 4 properties :-

Date: 13/Ap/25 Page no:

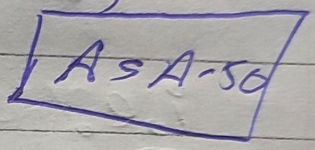
Acid properties:-

First we need to understand that 6 steps:-

(i) read (A)
(ii) A = A - 50
(iii) Write (A)
(iv) Read (B)
(v) B = B + 50
(vi) Write (B)

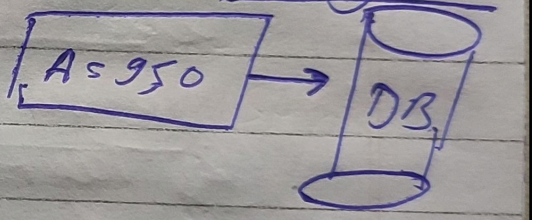
(i) 

(ii) Then,



(iii) storing in DB

Write (A) =



→ Commit:- All changes done in main (buffer) memory we store that changes in our DB.

Date: _____ Page no: ___

Now 4 Properties (ACID Properties)

(1) Atomicity :- Either all steps done & reflect in DB, if any failure / no changes & rollback.

(2) → transaction → successful
→ fail → old state recovery.

i.e.
DB maintains → old state
↓
Intermediate state
↓
success X
Rollback old state

(2) Consistency :- (1) Integrity constraints must be maintained ~~by~~ before & after transactions.

Date: _____ Page no: ___

(2) DB must be consistent after transaction happens.

(3) Isolation :-

Eg :- Banking System

T_1, T_2, T_3, T_n

Where T are Transaction

→ means, if I do GPay & net Banking same time then both must be isolated & no inconsistency.

$t=0$ → T_1 → GPAY

↳ read (A) = 1000

A = A - 50 = 950

Write A = 950

T_1 terminated then T_2 execute

$t=10$ → T_2 → Net Banking

↳ read (A) = 950

A = A - 50 = 900

Write (A) = 900

(t = time in milliseconds)

$T_1 \longrightarrow T_2$

first come → first execute → first terminate then second & so on.

Date: _____ Page no: __

* Isolation:- If multiple transaction executes concurrently, then system guarantees that for every pair of transactions T_i & T_j,

if appears to T_i that T_j finished execution before T_i started or vice versa. Thus each transactions can happen in the system in isolation. without interfering each other.

(4) Durability :- After transaction succeed, the changes it has made to the DB persist, even if there are system failures.

i.e. T_i → success → then DB update permanent
↳ all step success

Date: _____ Page no: ___

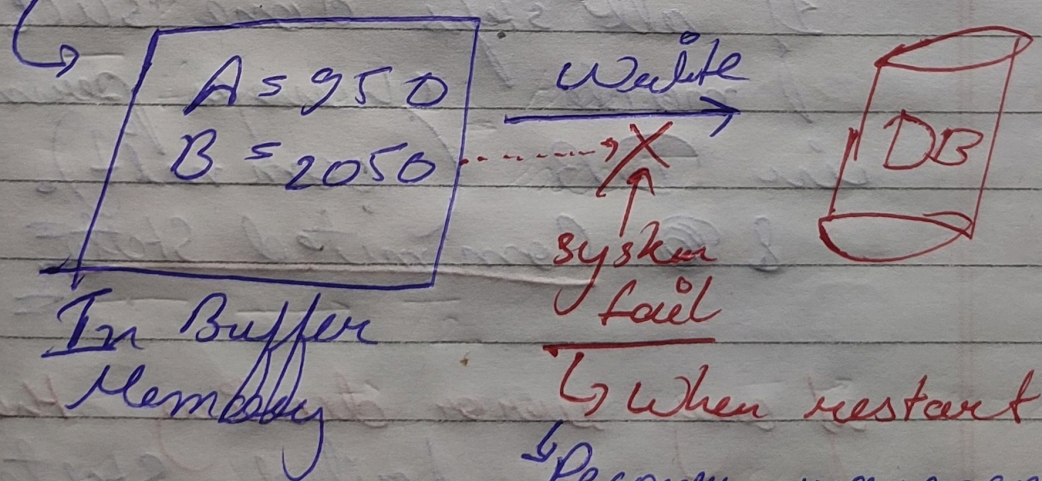
even if system fails but after transaction.

→ DB durability ensure by,

↳ Recovery management component.

i.e.,

1 read(A) ⟶ logs generate logs
2 A = A - 50 ⟶
3 Write(A) ⟶ ↳ logs (write A)
4 Read B ⟶ ↳ (Read(B)) log
5 B = B + 50 ⟶ —
6 Write B ⟶ —



In Buffer Memory

↳ When restart

↳ Recovery management check logs & write on DB

Date: _____ Page no: ___

States of Transaction
(Life cycle of transaction)

(1) Active State :- Read & Write Performed in main memory (buffer memory).

Then,

- if successed then T. comes to Partially committed state.
- If any failure then T. goes to failed state.

(2) Partially committed state :-
If all steps done success then made changes permanent in DB (commit).
& Commited state.

if any error during then go to failed state.

Date: Page no:

(3) Committed State :- When changes done in DB. Then transaction is done permanent & no rollback now.

(4) Failed State :- Any ~~to~~ kind of failure during active or partially committed State ; by which Transaction not possible.

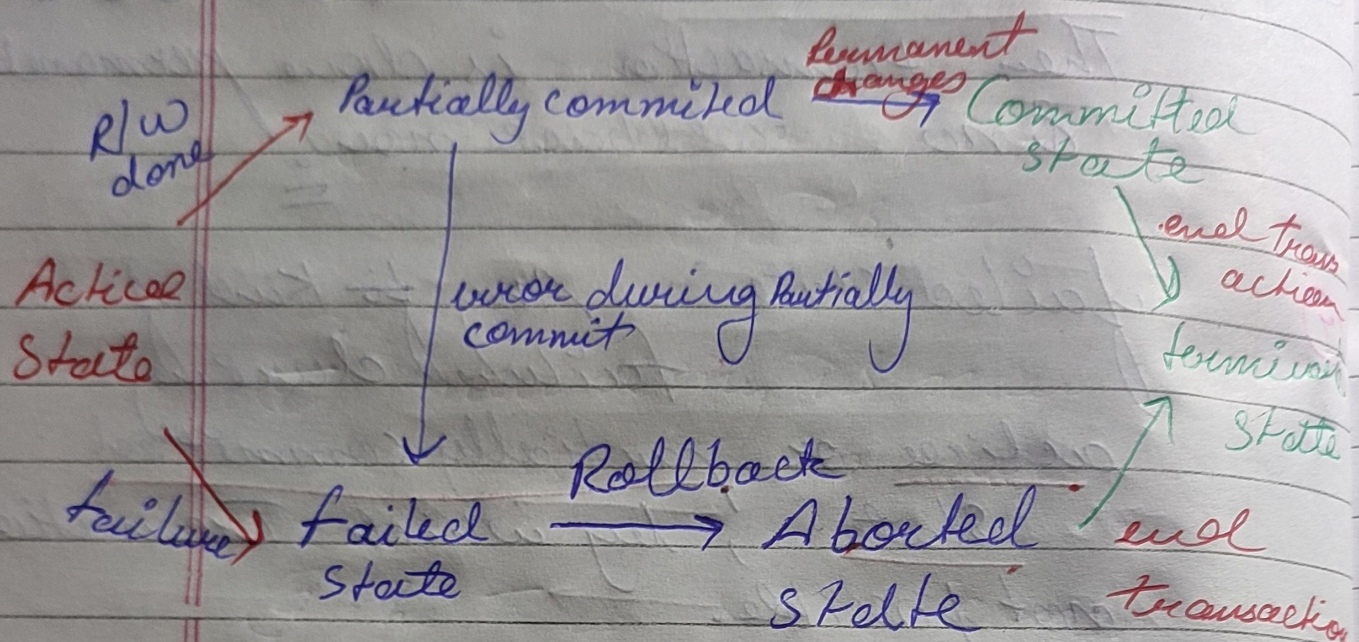
Then,

(5) ~~RollBack~~ Aborted State :- Undo all changes. & all changes Rollback & Transaction reach Aborted State.

(6) Terminated State :- Transaction ends when committed or either aborted.

Date: _____ Page no: ___

Dig. of Transaction State :-



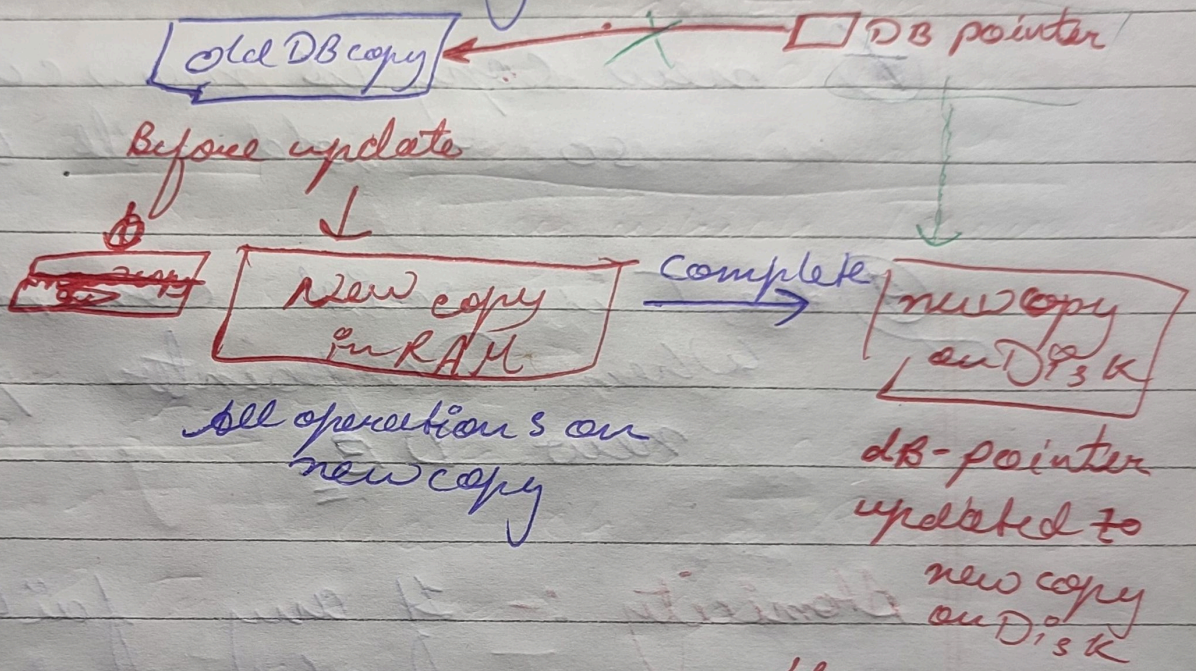
Date. 15 April Page no:

Lec-13

How to implement atomicity & Durability.

ways:-

① Shadow Copy Scheme



→ ⓐ Making copies of DB in ram. then T. committed

ⓑ One T. at a time.

ⓒ T_i that wants of update DB first create complete copy of DB.

ⓓ If T is aborted ~~must~~ new copy in RAM will deleted.

Date: _____ Page no: ___

③ If T is successed
↳ all pages of new copy are written on disk & DB pointer to the new copy of DB in disk.

④ new copy is now current so old will be deleted

⑤ the T is said committed when DB pointer points new DB.

Atomicity :- If any failure old changes are stored.
Hence all changes reflection or error null

Durability :- DB pointer points previous DB in disk until new DB is

Date:_____ Page no:___

commited pointer by DB pointer.

→ Only when DB pointer committed.

If system fails after DB pointer update so no issue it is pointing new DB.
It can read new copy.

- The pointer update is also atomic.

- Inefficient, as entire DB is copied for every transaction.

(2) Log Based Recovery :-

(a) Each T is maintained in some stable storage

(b) first write log then do that step.

Date:_____ Page no:__

Types:-

① Deferred DB modification:-
(टालना)

→ @ first logs are written along with transaction steps., then DD is modified on the basis of logs.

• So if system fails before commit then all steps have rollbacked by ignoring that logs.

or if T completed then write DB in deffered way.

Durability:- if there is commit in logs then Disk will write else: rollback & if while writing disk fails then

Date:_____ Page no:___

use redo (re-read) logs & made changes.

(2) Immediate DB modifications:-

→ log → step → write

→ DB written at active Talong with process of T.

→ If's log contains new values, along with old values. So if T fails then roll Back using logs old value.

If T completes and system crashes, then new value field is used to redo T having commit log.

Date: _____ Page no: ___

(3) Check Point :-

If lacks of transactions so memory of log is lot so we create checkpoints :-

$C_1 \rightarrow \Sigma T_n \rightarrow$ commit ↑

$C_2 \rightarrow \Sigma T_n \rightarrow$ commit ↑

C_n