

Lec 12

## Normalization

↳ nor. → DB optimization.

\* Functional dependency (FD)

- relationship b/w the primary key attribute of relation to that of other attribute of relation

determinant

$A \rightarrow B$   $\rightarrow$  dependent

$A$  is  $B$  determinant  $A \rightarrow C$

no such

Table

A	B	C	D
1	2	1	5

i.e. each  $A$  ki value has  
the corresponding 2 values  
in by  $A \rightarrow B$

e.g. emp

emp id / name / dept.

FD:

emp id  $\rightarrow$  emp name

emp id  $\rightarrow$  dept.

\* Types of FD

$\rightarrow$  Trivial FD

$A \rightarrow B$ ,  $B$  is subset of  $A$

e.g. {emp id, name}  $\rightarrow$  emp id

e.g.  $A \rightarrow A$ ,  $A \subseteq A$

$B \rightarrow B$ ,  $B \subseteq B$ .

→ Non Trivial FD

{Emp ID, name} → {Emp address}

$B \subseteq A$ , ~~which~~ subset.

⇒ NIT FD.

$A \rightarrow B, B \not\subseteq A, A \cap B = \text{NULL}$

\* RULES of FD (Armstrong's axioms)

→ Reflexive.

•  $X = \{a, b, c, d, e\}$

Subset  $Y = \{a, b, c\}$

$\Rightarrow Y$  is subset  $X$ .

$\Rightarrow X \rightarrow Y$

→ Augmentation.

$X \rightarrow Y$  (Given)  $R(X, Y, 2)$

Now, then

$XZ \rightarrow YZ$

→ valid

→ Transitivity.

$A \rightarrow B, B \rightarrow C$

$\Rightarrow A \rightarrow C$

→ Armstrong inference rules.

\* Derived rules.

• Union:  $yx \rightarrow y \& x \rightarrow z$   
then  $x \rightarrow yz$

$x \rightarrow x$

$\Rightarrow xy \rightarrow xy$

$x \rightarrow z$

$\Rightarrow yx \rightarrow yz$

$x \rightarrow x \& x \rightarrow yz$

$\Rightarrow x \rightarrow yz$

• Decomposition

$y \rightarrow yz$ , then  $x \rightarrow y \& x \rightarrow z$

• Pseudotransitivity

$yx \rightarrow y \& yw \rightarrow z$  then

$xw \rightarrow z$

Q) R(ABCDE)  
 FD :  $A \rightarrow B$   
 Given :  $A \rightarrow C$   
 $CD \rightarrow E$   
 $B \rightarrow D$   
 $E \rightarrow A$ .

Can  
 $CD \rightarrow AC$ ?

$CD \rightarrow E$  &  $E \rightarrow A \Rightarrow CD \rightarrow A$  (transitivity)

$C \subseteq CD \Rightarrow CD \rightarrow C$  (reflexive)

so,  $CD \rightarrow A$  &  $CD \rightarrow C$  UNION  
 $\Rightarrow CD \rightarrow AC$

\* Why normalization?  
 To avoid redundancy in DB not to store redundant data.  
 as insertion, deletion and update anomalies arises due to redundant data.

## → Anomalies

### • Insertion

id	name	age	Branch_code	Branch name	Branch HOD
1	A	18	1	CS	X
2	B	19	1	CS	X
3	C	18	2	ECE	X
4	D	20	3	ME	Z

if a new student comes but not yet selected branch

5	E	19	let null	let null	let null
---	---	----	-------------	-------------	-------------

Phir baad wal null ko update karo  
 ye h. anomaly

ye  $\lambda$  ki student ki existence or branch ki existence ph independent h

Now, if university added IT branch ph jis at the moment ph koi student hi nahi

- Deletion

agar id=4 ko delete kare ab wo passout hogga ph ye deletion ke karan ME branch ni hatt 849, as suppose uss Branch mae kewal wahi tha.

Deletion of data results in unintended loss of some other important data.

- updation

ref HOD of CS change to S. ph woh ab har student ke detail ko update karna padega.

id	name	age	BR. code

BR code	BR. name	HOD

- \* In normalization

Table  $\rightarrow$  decomposes into multiple tables.  
entill SRP is achieved.

$\downarrow$   
single responsibility principle.

\* types of normal forms

### → INF

- every relation cell must have atomic value.
- Relation must not have multivalued attribute.

e.g.

~~emp~~

id	name	Phone
1	A	88
2	B	12
2	B	99

Convert INF

emp

id	name	Phone
1	A	88
2	B	12, 99

### → 2NF

- Relation must be in INF
- There should not be any partial dependency.
  - All non prime attributes must be fully dependent on PK.
  - Non Prime attribute can not depend on part of PK

R (A B C D)

{ A B } → PR : A & B will make PK

A, B → prime attribute

C, D → Non prime

⇒ FD:  $A \rightarrow C$   $B \rightarrow C$   $\rightarrow$  Partial dependency.

Yeh kaha chahiye

✓  $AB \rightarrow C$  but  $B \rightarrow C$  mal B alone C.  
 $AB \rightarrow D$  Ko determine kar gahan

$\{A, B\} \rightarrow PK$ .

A	B
NULL	1
2	NULL
NULL	NULL
3	4

$A, B$  CX saath null rhi hoga as

$PK \leftarrow \{A, B\}$

$R(A, B, C, D)$

$A \xrightarrow{B} C$

$A \in C \quad B \in D$

~~Suppose~~  $B = NULL$

NOW  $B \rightarrow C$

$NULL \rightarrow C \quad \times$  not holds.

2NF Conversion.

$$R_1(A, B, D) \quad R_2(B, C)$$

$\downarrow$        $\downarrow$

PK.       $B \rightarrow C$

$AB \rightarrow D$

C.G.

<del>OK conversion not possible</del>		student name	Project name
Student ID	Proj IP	JACOB	GEO
589	P09	AVA	IOT
576	P07		

PK: {Student ID, Proj ID}

F.D.: student ID  $\rightarrow$  student name

Given project ID  $\rightarrow$  project name.

Non prime is being determined by part of PK.

2NF form

student

student ID	proj ID	student name
------------	---------	--------------

Project

project ID	project name
------------	--------------

3NF

- Relation must be in 2NF
- No transitivity dependency exists.
  - Non-prime attribute should not find a non-prime attribute.

R (A BC)

INF ✓

PK ∈ A<sup>3</sup>

2NF ✓      B & C fully  
dependent on PK.

F.D: -  $A \rightarrow B$        $B \rightarrow C$       } Transitivity dependency

A	B	C
a	x	x
b	x	x
c	x	y
d	2	y
e	2	y
f	3	z
g	3	z

$A \rightarrow$  Prime  
 $B \rightarrow$  C  
Non Prime      } Prime

3NF form

$R_1 \underline{(A B)}$      $R_2 \underline{(B C)}$

$R \underline{(ABC)}$   
 $A \rightarrow B$   
 $B \rightarrow C$

A	B
a	1
b	1
c	1
d	2
e	2
f	3
g	3

B	C
1	x
2	y
3	z

for  $R_1$   
INF ✓ atomic value ✓

2NF ✓ will not have  
as  $A \rightarrow B$  in  $R_1$  table &  $B \rightarrow C$  in  $R_2$  table ✓

3NF ✓  
 $R_1 P \rightarrow B$  prime & non prime &  $R_2 B \rightarrow C$   
prime & non prime ✓  
no dependency

BCNF (Boyce - codd normal form)

→ BCNF must be 3NF

FD:  $A \rightarrow B$ , A must be superkey & C.R.

we must not derive prime attribute from any

prime or non prime attribute.

~~HR → P~~

~~P → P~~

E.g

std-ID	subject	professor
101	Java	PJ
101	CPP	PC
102	Java	PJ2
103	C++	PC #
104	Java	PJ

- one student can enroll in multiple subjects.
- so each subject, a professor is assigned to student.
- multiple prof. can teach single subject.
- one prof. can teach only one subject.

## \* Fourth Normal form (4NF)

- Relation is 4AF if it is in  $X \rightarrow\!\!\! \rightarrow Y$   
- BCNF form

- has no multi-valued dependency.

i.e. for a dependency A, B if for a single value of A, multiple values of B exists, then the relation will be  $\checkmark$  801<sup>a</sup> multi-valued dependency.

vo. M, E1
vo. M, E2
vo. M, E3
vo. M, E1

vo. M1
vo. M2

E1
E2

## \* Fifth Normal form (5NF) or project join Normal form

If it is in

- 4NF & no contains any join dependency & joining should be lossless.

- 5NF is satisfied overall the tables are broken into many tables as possible in order to avoid redundancy.

- Lossless join

## \* Indexes

CREATE INDEX indexname ON table\_name (column1, column2, ... , columnn);

table\_name (column1, column2, ... , columnn);

\* Lossless Decomposition

R	A	B	C
1	2	1	
2	2	2	
3	3	2	

$\rightarrow R, IA \cdot B)$

R <sub>1</sub>	A	B
1	2	
2	2	
3	3	

$\rightarrow R_2 (BC)$

R <sub>2</sub>	B	C
2		
2		2
3		2

$R_1$  &  $R_2$  ke sikh ok common  
attribute kara chahiye so  
decomposition

BC agar doosra was  $R_1 \& R_2$  ko join kara to phr oos common  
attribute ke base per hain.

and the value of C of the various A - l.

$\Rightarrow$  JOIN i.e natural join

select R<sub>2</sub>. C from R<sub>2</sub>.

Natural join

R<sub>1</sub>

where R<sub>1</sub>. A = ?;

natural join  
= cross product of tables  
+ C.

R <sub>1</sub>	R <sub>2</sub>
1 2 1	2 1 2
1 2 2	2 2 2
2 2 1	2 1 2
2 2 2	2 2 2
3 3 2	3 2 2

After join ✓  
R' (original table hain)  
but tuples badhiya)

A	B	C
1 2 1		
1 2 2		
2 2 1		
2 2 2		
3 3 2		

A job D h

so C K<sup>o</sup> value

1 or 2 aa hi which is wrong.

remove duplication of B

i.e. lossy decomposition

PK: { Std-ID, Subject } as dono loga tb ki  
professor kon h pta lagega

FD:

Std-ID, subject → Professor.

Professor → Subject (as koi ek prof. ko koi ek subject  
 hi padhna hota)

∴ prime  
 no prime

~~XY NP → prime~~

so tables not in BCNF

⇒ BCNF form.

student	
od	Pid
101	1
101	2
,	,
1	)

Prune		Prof	Sub	PB
Pid	Prof			
1	PS			
2	PC			
		Java		
		CPP.		

here P → NP

here  
 P → NP

⇒ BCNF  $\surd$

\* Advantages of Normalisation

- helps to minimise data redundancy.

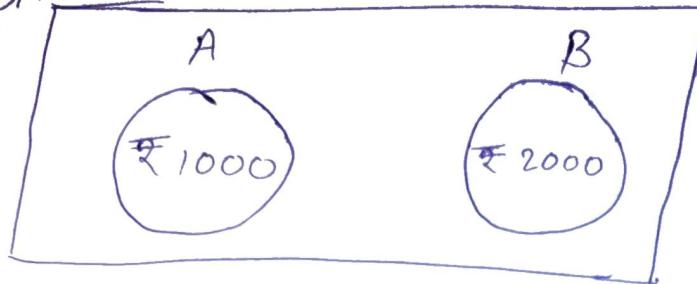
- Greater overall dB organisation.

- Data consistency is maintained in DB.

L-12

## ACID properties & Transactions

\* Transaction



Task P → ₹50 from Bank A/C A to A/C B

T<sub>e</sub> : Read(A)

A = A - 50;

Write(A); → 950 kanna

Read(B)

B = B + 50

Write(B)

} 6 steps

atomic  
they are  
considered to  
be a single task.

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

- It is a logical unit of work that contains one or more SQL

statements. The result of all these statements in a transaction either gets completed successfully

(all the changes made to the database are permanent)

or if at any point any failure happens it gets

rolled back (all the changes being done are undone).

## \* Acid Properties

→ to ensure integrity of data, we say that DB system maintain following properties of transaction.

### Atomicity

read(A)



write(A);



$T_i$  : read(A)

$$A' = A - 50$$

write(A);

read(B);

$$B' = B + 50;$$

write(B);

Real DB  
commit kare xe Baad.  
1950 → BB  
DB mei jata h.

### Atomicity

Either all operations of transactions are reflected properly in DB, or none are.

account

A

read 1000

$$950 = 1000 - 50$$

write(A)

950

B account

read 2000

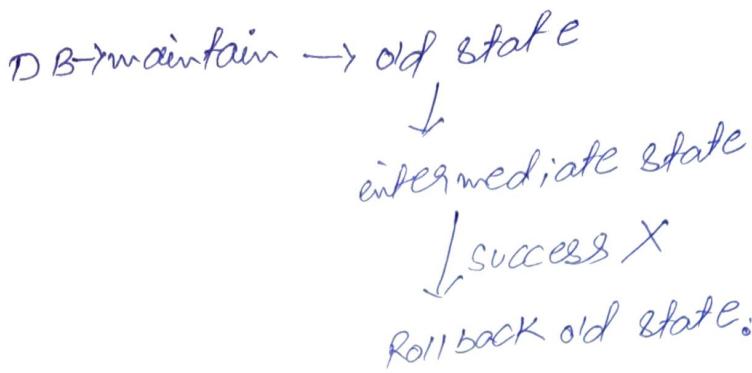
read 2000

$$2000 + 50$$

write(2050)

8050

transaction → successful  
→ fail → old state recovery.



### \* Consistency

- Integrity constraint must be maintained before and after transaction.
- DB must be consistent after transaction happens.

ie agar DB total ₹ 3000 h ft operation ke baad bhi ₹ 3000 hi hona chahiye.

### \* Isolation

Banking System.

T<sub>1</sub>, T<sub>2</sub>, T<sub>3</sub> - - -

if wo agar hr w Gpay se payment kare aur netbanking se bhi transaction kare to wo dono isolated honge.

T<sub>1</sub> (G pay)

₹ 100 read (A) → 1000 CK Baath

$$A: A - 50 = 950$$

₹ 950/-

+10 T<sub>2</sub> (Netbanking)

read K91, ya

read (A) → 1000

$$1000 - 50 = 950$$

B → 50 + 50

It aise B ko double so chlo gya so KP  
possible nahi.

T<sub>b</sub> & T<sub>i</sub> success/failure nahi hogi. ∵ T<sub>b</sub> ko  
start bhi nahi hogi thaise isolation hogao jaega.

- Even though multiple transactions may execute concurrently, the system guarantees that for every pair of transactions T<sub>i</sub> and T<sub>j</sub>, it appears to T<sub>i</sub> that either T<sub>j</sub> finished execution before T<sub>i</sub> started or T<sub>j</sub> started execution after T<sub>i</sub> finished. Thus, each transaction is aware of other transactions executing concurrently in system.
- Multiple transactions can happen in the system in isolation, without interfering each other.

### \* Durability

- After transaction completes successfully, the changes it has made to the database persist even if there are system failures.

T<sub>i</sub> → SUCCESS



DB updates permanent (persist)



even if there is a system failure after T<sub>i</sub> completes trans.

→ recovery - management component.

read(A)

$A = A - 50$

write(A)

Read(B)

$B = B + 50$

write B.

→ log (S0)

→ log (writerA)

$A = 950$

$B = 2050$

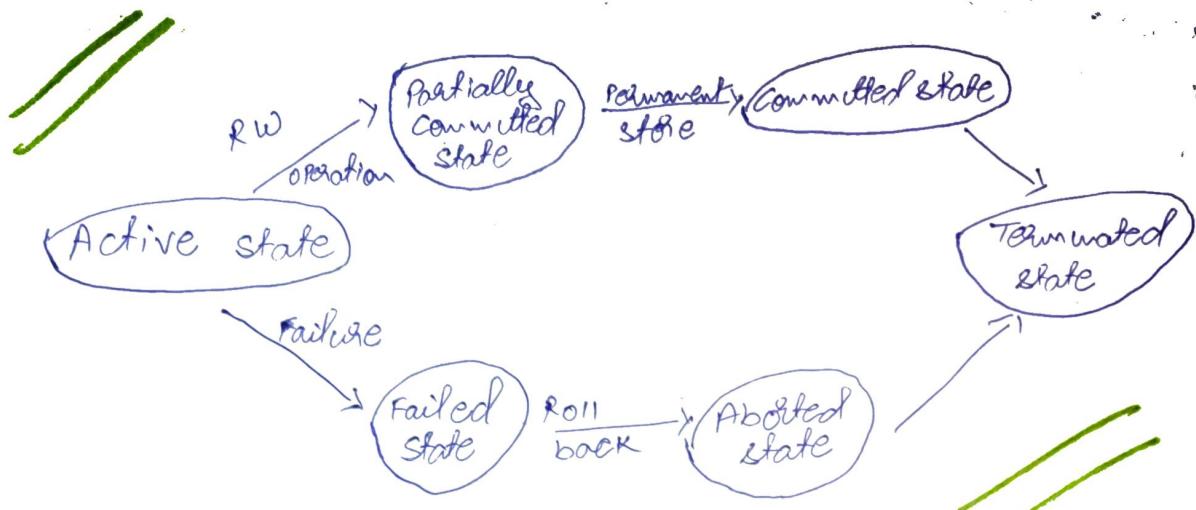
DB file  
while loop



main memory

Save failure area  
by

## \* Transaction States .



### 1. Active state

The very first state of life cycle of transaction, all the RW operations are being performed. If they execute without any error the T comes to partially committed state. Although if any error occurs then it leads to a failure.

### 2. Partially committed state.

After transaction is executed the changes are saved in the buffer in the main memory. If the changes made are permanent on the DB the state will transfer to committed state & if there is any failure, the T will go to failed state.

### 3. Committed state.

new updates are made permanent on DB, Then the T is said to be in committed state. Roll back can't be done from the committed state. New consistent state is achieved at this stage.

#### 4. Failed state

when T is being executed & some failure occurs due to this it is impossible to continue the execution of T.

#### 5. Aborted state

when T reaches the failed state, all the changes made in the buffer are reserved. After that the T will back completely. T reaches abort state after roll back, DB's state prior to the T is achieved.

#### 6. Terminated state

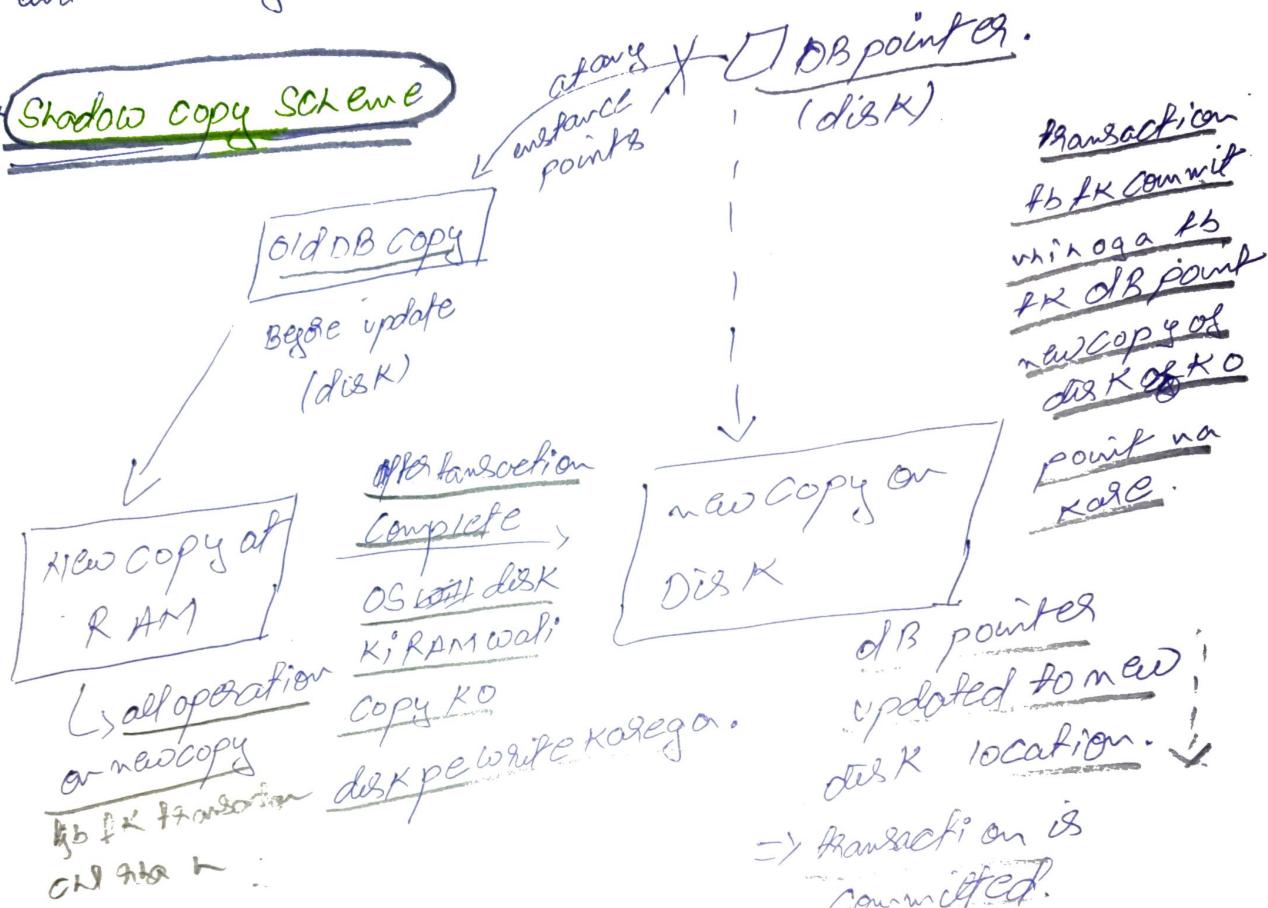
A transaction is said to have terminated if has either committed or aborted.

L-13 HOW to implement: sustainability & Durability?

- Durability?

  - Recovery mechanism component of DBMS supports atomicity and durability.

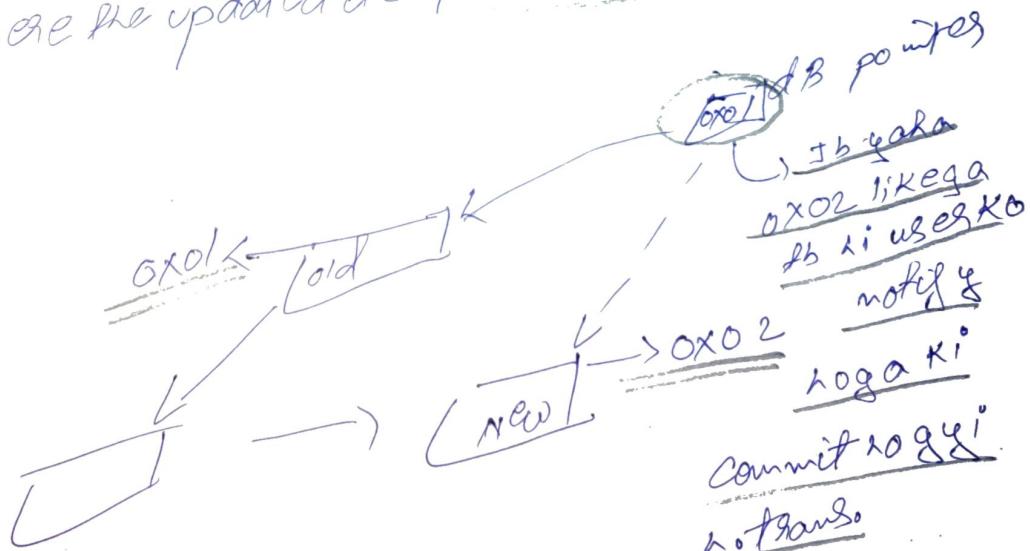
## \* Shadow copy Scheme



- Based on making copies of DB (aka shadow copies)
  - Assumption only one transaction ( $T$ ) is active at a time.
  - A pointer called DB pointer is maintained on the disk; which at any instant points to current copy of DB.

- All further updates are done on new DB copy leaving the original copy (Shadow copy) untouched.
- If at any point the T has to be aborted the system deletes the newcopy and old copy is not affected.
- If T success, if is committed as,
  - OS will make sure all the pages of the new copy of DB are written on the disk.
  - DB system updates the db pointer to point to the new copy of DB.
  - New copy is now the current of DB.
  - The old copy is deleted.

So the T is said to have been committed at the point where the updated db-pointer is written to disk.

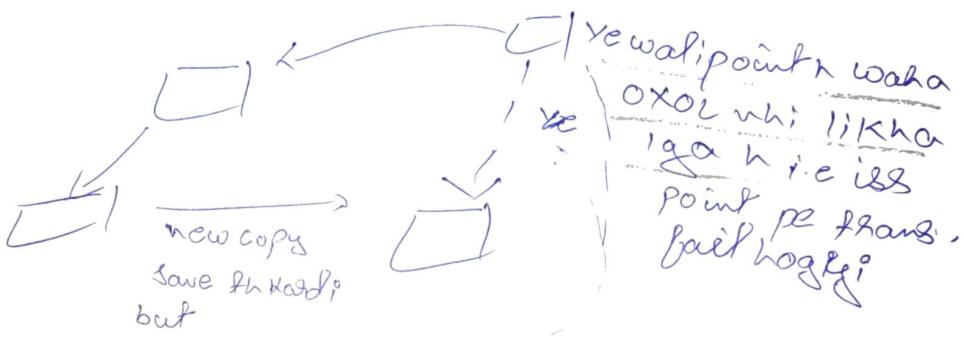


## Atomicity

- If T fails at any time before db pointer is updated, the old content of DB are not affected.
- T abort can be done by just deleting new copy of DB.
- Hence, after all updates are affected or none.

## Durability

- Suppose, system fails at any time before the updated db pointer is written to disk.



- Then the system restarts, it will read db pointer & will thus see the original content of DB and none of the effects of T will be visible.
- Trans. is allowed to be successful only when db pointer is updated.

$\text{Q} \leftarrow \text{0X02}$  ab system crash hogga aise  
 $\text{Q} \rightarrow \text{0X01}$  ye db pointer waala th point karna tha math or db  
will be ga

No. 00

Suppose if system fails after db pointer has been updated.  
Before that all the pages of new copy were written to disk. Hence,  
when system restarts it will read new DB copy.

Trans ko commit bol dia, bolve ke bad system restart  
hoga tb durability maintain karne ki db mai changes  
hoga n persist kare.

or yaha persist ho ghan.

- agar  $0X01 \rightarrow 0X02$  is ke bich mai fail hogga tb  
The implementation is dependent on write to the db pointer,  
being atomic.

LUCKILY, disk system provide atomic update to entire block or at  
least a disk sector.       $\hookrightarrow$  update hogga yahi hogga

So, we make sure db pointer lies entirely in a single sector.  
by storing dbpointer at the beginning of a block.

- Inefficient, as entire DB is copied for every Trans.  
method

\* Log-based Recovery systems.

- The log is a sequence of records. Log of each trans is maintained in some stable storage so that if any failure occurs, then it can be recovered from there.
  - If any operation is performed on the data base, then it will be recorded in the log.

A  
To : read(A)  
A: A-SO.  
write(A)  
B: B+SO  
write(B).

Log B (stable storage)

L TO start } variable

L TO, A, 950 } new value

L TO, B, 2050 }

L TO commit }

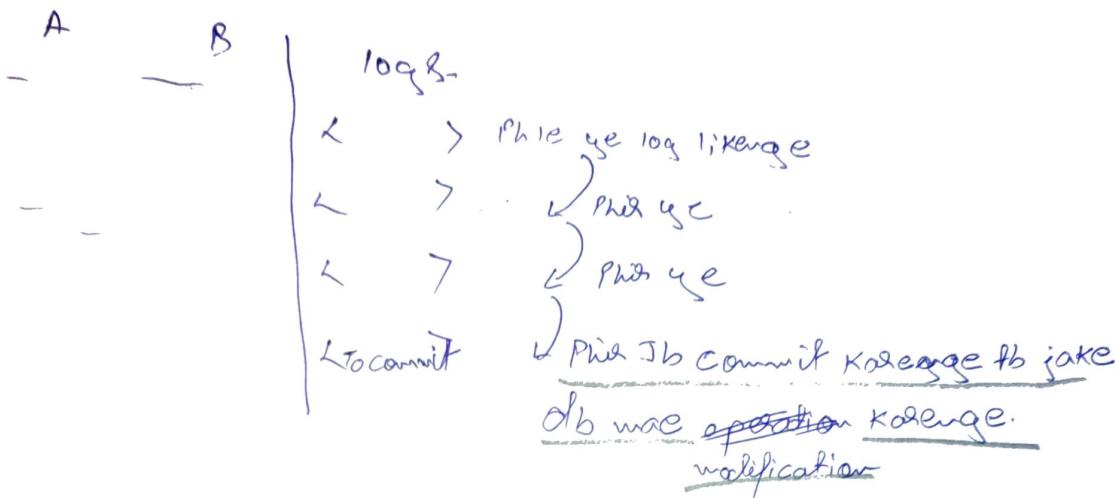
- but the process of storing the logs should be done before the actual trans. is applied to the db.

Log file likeage phis broadmace operation likeage.

// stable storage: is a classification of computer ~~as~~ data storage technology that guarantees atomicity for any given write operation and allows software to be written that is robust against some hardware and power failures.

## Deferred DB modification.

- Ensuring atomicity by recording all DB modifications in the log but deferring the execution of all the write operations until the final action of the T has been executed.

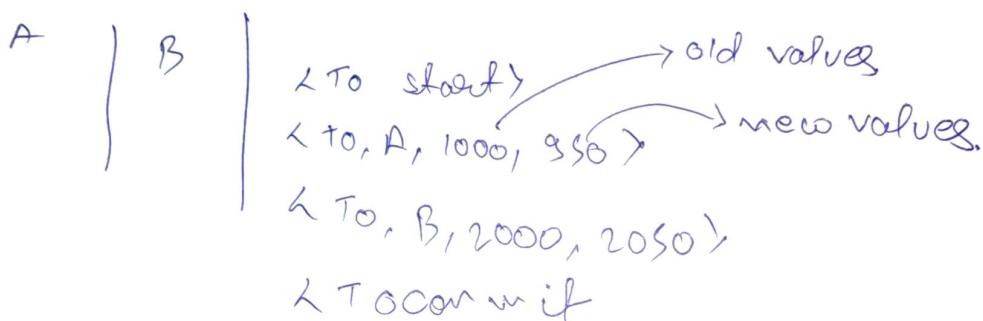


- Log-info is used to execute deferred writes when T is completed.
- If system crashed before the Trans. completes, & if T is aborted, the info in the logs are ignored.
- If Trans. completes, the records associated to it in the log file are used in executing the deferred writes.
- If failure occurs while this updating is taking place, we perform Redo
  - wasas se log ko read karna ga or save op. file se karna ga

## Immediate DB modification

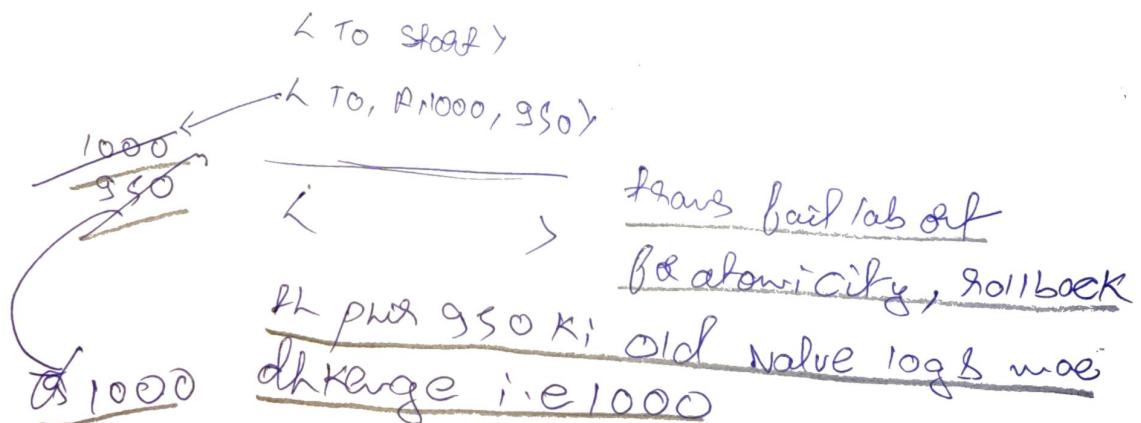
- DB modi to be output to the DB while the T is still in active state.
- DB modi written by ~~the~~ active T are called uncommitted modifications.

operation ko log karega pr phir just baad hi use write kar denge.



### For durability

In the event crash or T failure system uses old value field of log records to restore modified values.



- update takes place only after log records in a stable storage

- Failure handling.

System failure before Trans completes OR if Trans aborted, then old value field is used to undo the Trans.

If Trans completes and System Crashes then new value field is used to redo Trans having Commit logs in the log.

L      >  
L      >  
L      >  
< To Commit > ~

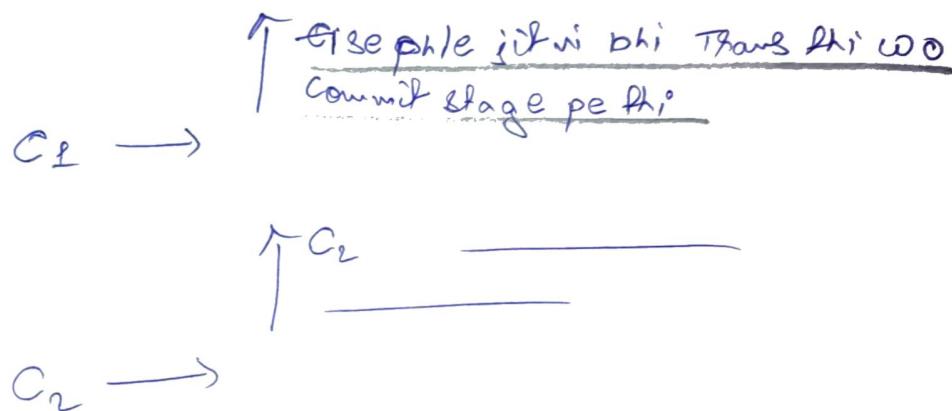
log ke andar ikh chuke h trans complete logyan h. DB  
mai update kare the abhi db gallogya th DB ke  
andar actual commit whi hoga n abhi's  
th Durability ke liye

Fir jb bhi system wapas se restart se hoke  
recover hoga wo aake dhaga ki ye <ToCommit>  
hui thi ki whi yaha th ikha h logya thi th..  
trans ko redo karenge

// check point

If there are 1000s of log then storage will keep getting full i.e. not possible in real time as cost etc increases

So use check point log based recovery method.



Aagar trans fail hofi krt wapas check point pe recover log jata h