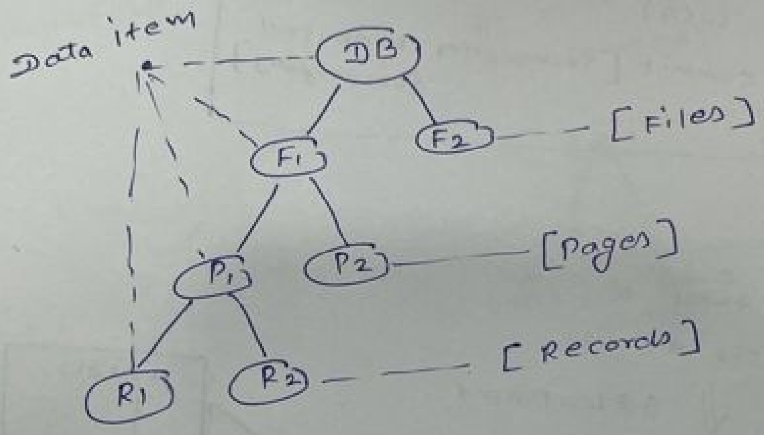
* Transaction and Concurrency Control →

Lec - 1
DBMS.
5-Nov-2021

Transaction :- set of operations use to perform unit of work.-

[process (or) program is in O.S
same as Transaction]

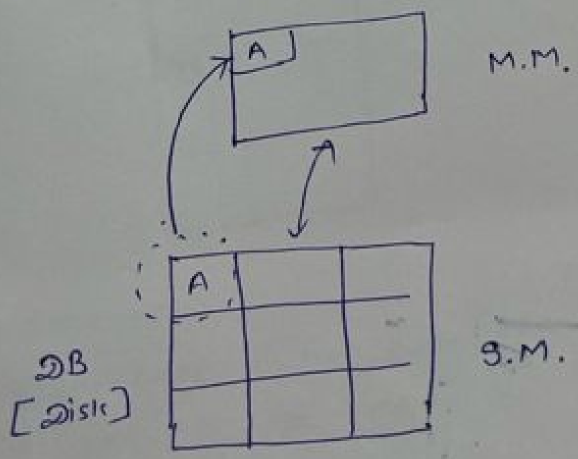
Data item :- [Database element], shared resources required for many transaction.



Main operations in Transaction :-

A: Data Item

a) R(A)
Read (A) :- accessing data item (A)
From DB [secondary memory] to program Variable.
[Main Memory]



(b) write (A) => Updation of data item, A in DB.
[S.M. Block]

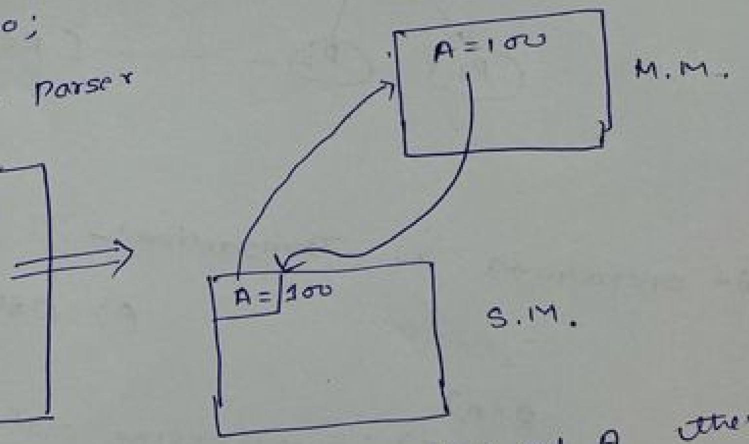
Trans (T₁)
R(A)
A = A - 1000
W(A).

=> update table R
set A = A+10;
⇓ SQL parser.

Trans. (T₁)
R(A)
A = A+10;
W(A)
commit [Transaction Executed successfully]

=> update table R
set A = 100;
⇓ SQL parser

Trans(T₂)
W(A)
commit

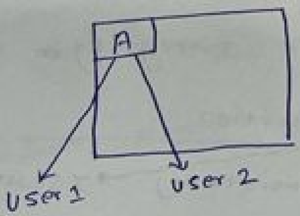


* Don't need to use previous value of A then only W(A) is sufficient for (A = 100, set)

✯ Concurrency Control :- Method to avoid inconsistency [error] becoz of concurrent execution of two or more transaction over same DB.

✫ Degree of Concurrency :- # of transaction can be executed concurrently over same data base. [
* It is useful when more degree of concurrency required for any DB.

⇒ Concurrent Execution :- Simultaneous execution of two or more transaction over some data item (A)

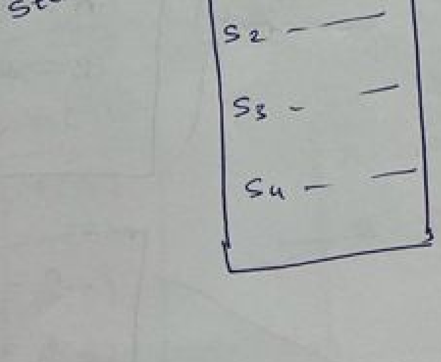


T_1	T_2
R(A)	
	W(A)
W(A)	

[Inconsistency may occur]

* Concurrency control over flat files [OS File] :

OS Controller : [File treated as resource)



⇒ U_1 : update S_1 +
U_2 : update S_2

T_1	T_2
lock (stud.txt)	lock (stud. txt) ⇓ (wait) (denied)

*
⇒ Because concurrency control over "File" is O.S. which result less degree of concurrency.

⇒ DBMS Concurrency Controller :- Resource is Record
↓
one of the data item

student

S_1
S_2
S_3
S_4
S_5

} lock the data at record level.

T_1	T_2
lock(s_1)	
	lock(s_2)
	⇑ grants.

⇒ Becoz Concurrency control over "Record" level degree of concurrency high.

* To preserve Integrity of Transaction should satisfy ACID properties:

atomicity → consistency → Isolation → Durability.

⇒ A: Atomicity :- Execute all operations including commit (100%) or Execute none of the operations ~~before~~ [Rollback].

Ex: T_1 : Trans 500 Rs from A to B

Assume balance of
A = 1000
B = 2000

⇒ Trans (T_1)
R(A)
A = A − 500;
W(A)
failed.→* R(B)
B = B + 500;
W(B)
Commit

100% done {
A = ~~1000~~ 500
B = 2000 + 500
= 2500

A = ~~1000~~ 500
B = 2000
inconsistent database!

Transaction failure becoz of:

(i) power failure

(ii) S/W crash :- DBMS down
- O.S. Restarted

(iii) H/W crash - Disk crashed
- N/W failure (linkdown)

(iv) OS/DBMS, may kills trans.

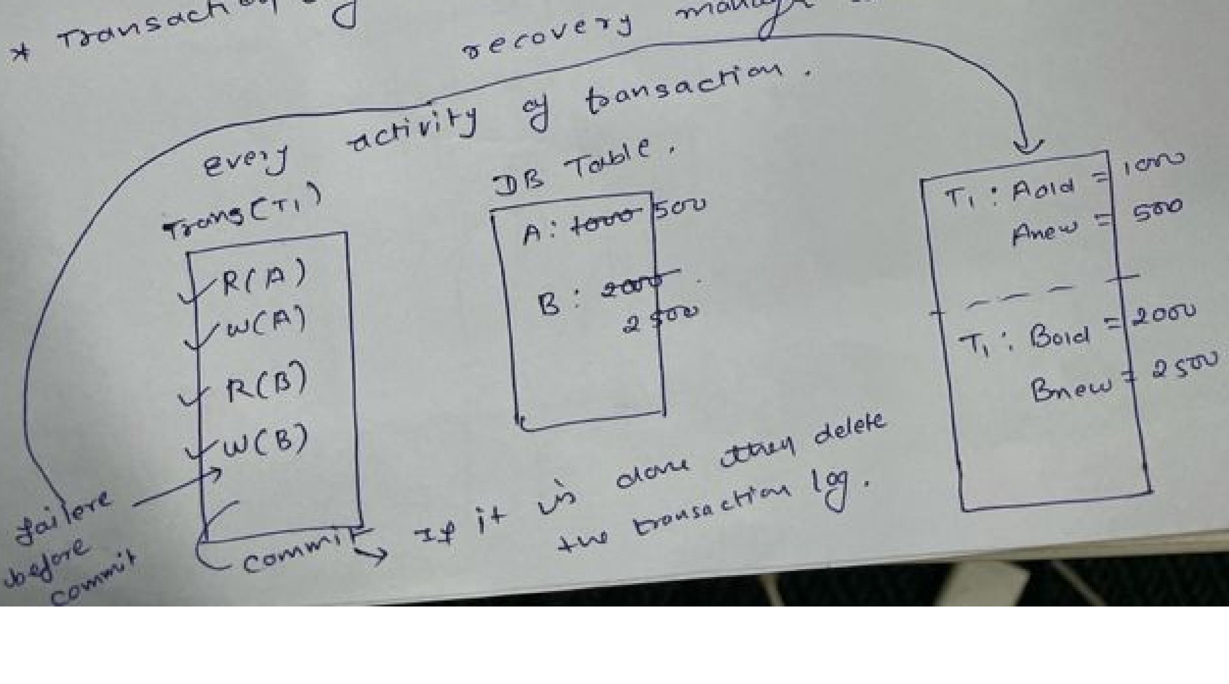
=> Recovery Mgmt. Component :- Recovery manager Rollback transaction

if transaction failed, before commit.

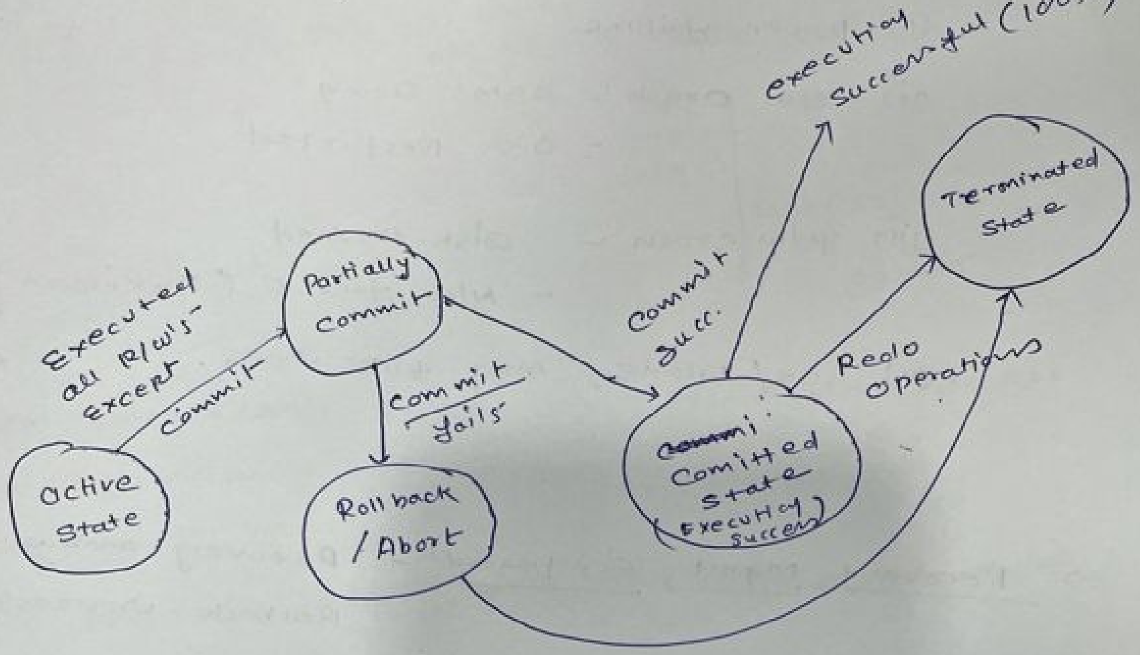
=> Rollback (Abort) => undo DB modification that are done by transaction which is failed.

[Reset database into previous consistent state].

* Transaction log => It is a file maintain by recovery manager which records every activity of transaction.



* Transaction States :-


execution successful (100%)
Terminated State
Partially Commit
Executed all R/W's except commit
Commit succ.
Commit fails
Redo operations
Active State
Rollback / Abort
~~Commi~~ Committed State (Execution success)