

Computer Science & IT

Database Management System



Transaction & concurrency control

Lecture No. 01



By- Vishal Sir

Recap of Previous Lecture



Topic

Practice questions



Topic

Domain relational calculus (DRC)



Topics to be Covered



- ✓ Topic Transaction
- ✓ Topic ACID properties
- ✓ Topic Atomicity





Topic : Transaction



An ordered sequence of logically related operations that are used to perform a particular task is called a transaction

Read (A) :- This operation is used to read the value of dataitem 'A' from database (ie, disk) to main memory

Write (A) :- The current value of dataitem A that is present in main memory will be updated into the database

★ Let 'A' & 'B' represent the amount in two different account 'A' & 'B' respectively

• Consider, initially $A=1000$ & $B=0$

• Transaction: To transfer 500/- from A to B

Transaction

Read (A)

$A = A - 500$

Write (A)

Read (B)

$B = B + 500$

Write (B)

Commit

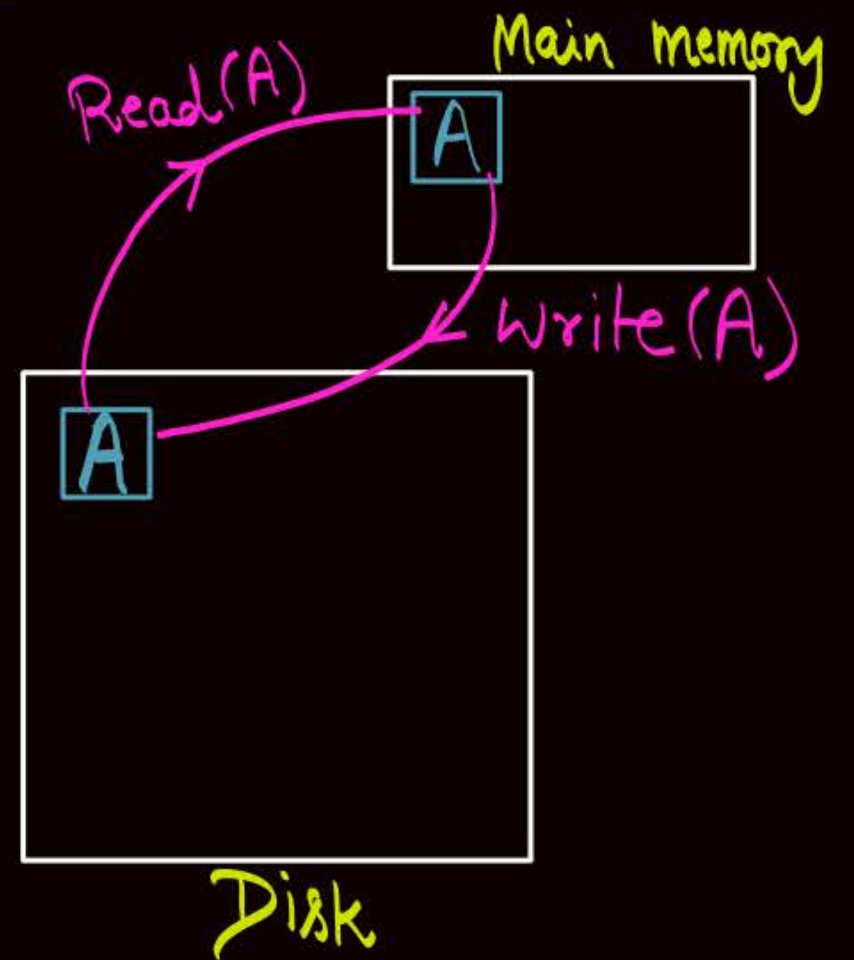
This two operations are not important from database point of view.

← Because of this opⁿ a Copy of dataitem A will be created in Main Mem.

← This opⁿ will be performed in Main Mem

← Current value of dataitem A from main mem. (i.e. $A=500$) will be updated in the database

← Transaction Committed means, transaction executed Successfully from Users Point of View.





Topic : Blind write



Read(A) \equiv R(A)
Write(B) \equiv W(B)

If transaction T updates (write) any dataitem in the database without reading that dataitem from database, then it is called a "blind write" opⁿ.

Transaction 'T'

R(A)

R(C)

W(A)

R(B)

W(D)

W(C)

R(D)

← it is not a blind write opⁿ, because of

← it is a blind write opⁿ

← Not a blind write

Consider two transactions T_1 & T_2

$\frac{T_1}{R(A)}$
 $R(B)$

$\frac{T_2}{R(B)}$
 $R(A)$

} T_1 & T_2 are
two different
transactions

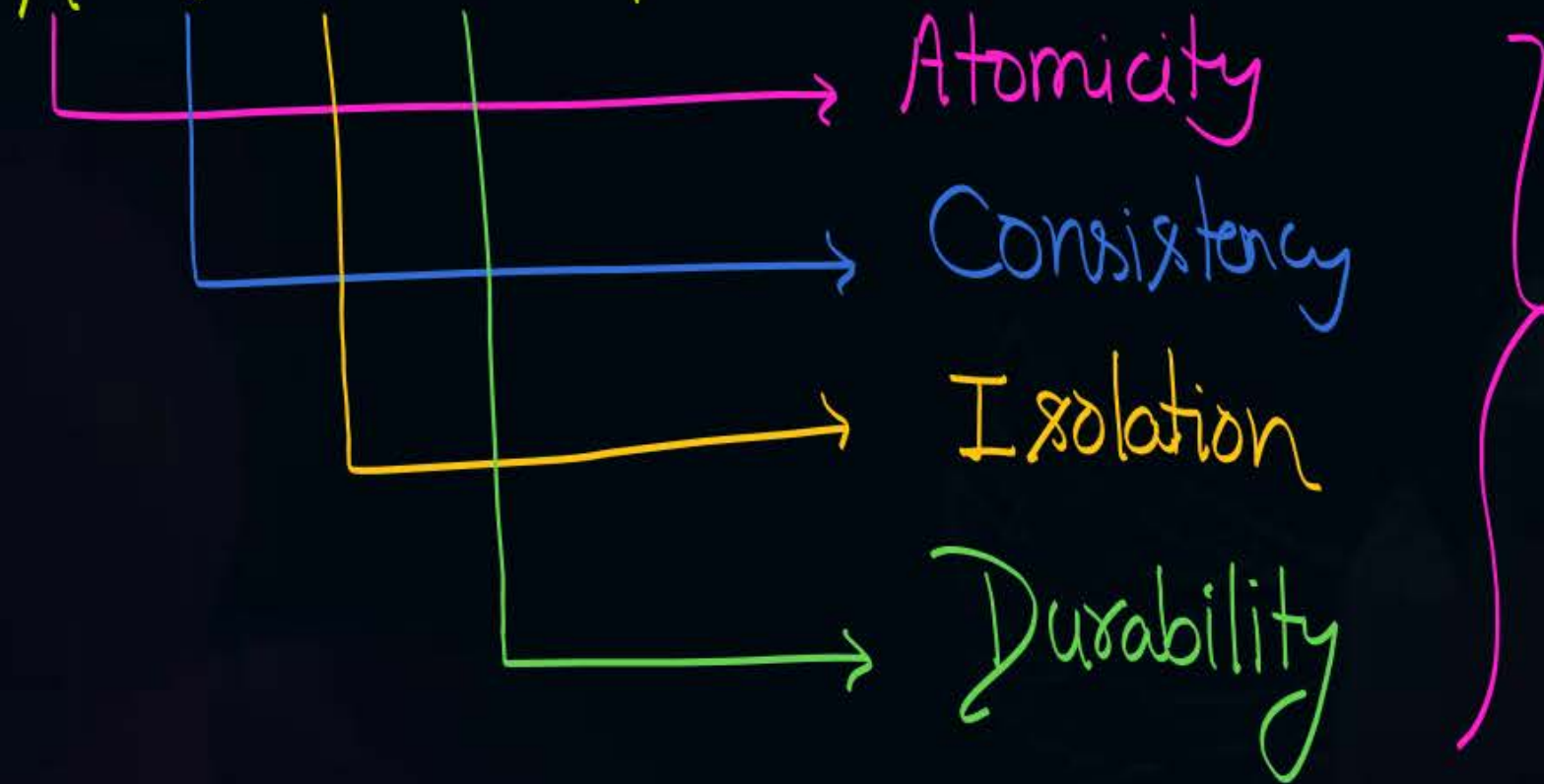
⇓
We can not change
the position of
the operations
within the transaction



Topic : ACID properties



* For the integrity of the database
ACID properties must be satisfied





Topic : Atomicity



* Atomicity states that, either execute all operations of a transaction or none of them.

There are two problems

① We can not execute all the operations of a transaction at once, they will be executed in a sequence one after another

② If we start executing the operations of a transaction, then we can not guarantee that all operations of that transaction will get executed without any failure

- * Let 'A' & 'B' represent the amount in two different account 'A' & 'B' respectively
- Consider, initially $A = 1000$ & $B = 0$
- Transaction: To transfer 500/- from A to B

Transaction

Read (A)

$A = A - 500$

Write (A)

Read (B)

$B = B + 500$

Write (B)

Commit

* failure occur
in between

- If we start executing the operations of a transaction, and if failure occur without executing its commit operation, then atomicity is violated

Note:-

To ensure atomicity, if transaction failed before executing its commit operation successfully, then we must undo all the changes performed by the transaction before its failure.

{ If we are able to do so, then it will be similar to the situation when none of the operation of the transaction is executed }

To ensure atomicity, "Recovery management Component" is used



Topic : Recovery management component

★ Recovery management Component is responsible for "undo" and "redo" operation

{ "undo" opn will be performed w.r.t. failed transaction

"redo" opn will be performed w.r.t. Committed transaction

↳ The process of "undoing" the changes performed by a failed transaction is called "Rollback"

For performing "Undo" and "Redo" operations, "Transaction log" is maintained by Recovery Management Component



Topic : Transaction log

- Transaction log is used to record the activities performed by the transactions

- * Let 'A' & 'B' represent the amount in two different account 'A' & 'B' respectively
- Consider, initially $A=1000$ & $B=0$
- Transaction: To transfer 500/- from A to B

Transaction (T_1)

Read (A)

$A = A - 500$

Write (A)

Read (B)

$B = B + 500$

Write (B)

Commit

* failure occur in between

Transaction Log

Start T_1 :

$(T_1, A, 1000, 500)$

If transaction T_1 fails at this point, then using transaction log we will rollback the value of dataitem A to its old value i.e. $A=1000$

Consider the following representation

(T_i, X, V_1, V_2)

Transaction id

dataitem

Old value of dataitem 'X' in database

New value of dataitem 'X' updated by transaction ' T_i '



Topic : Transaction log

★ Transaction log is used to record the activities performed by the transactions

★ We can recover from a failure only if "transaction log" is available

⇒ ∴ Transaction log is maintained by recovery management Component in the secondary memory (disk).

Note:

Either we can maintain a separate transaction log for each ongoing transaction
(Or) we can maintain a unified transaction log

And, Consider the following "transaction log"

Consider: (T_i, X, V_1, V_2)
 Transaction no. $\rightarrow T_i$
 data item $\rightarrow X$
 Old value $\rightarrow V_1$
 new value $\rightarrow V_2$

"Undo"

Redo

let system fails at this point

T_1 : Start
 $(T_1, A, 200, 120)$
 $(T_1, B, 100, 150)$
 T_2 : Start
 $(T_2, C, 250, 75)$
 Check point
 $(T_1, D, 90, 140)$
 $(T_2, E, 500, 290)$
 T_2 : Commit
 T_3 : Start
 $(T_3, F, 280, 100)$
 failure

The operations performed before the check point have been updated into database

If there is any transaction that committed before the last check point then we don't need to worry about those transaction

If any transaction/s committed after the last check point, then we need to "redo" only those operations of that transaction that appears after last check point

$\rightarrow T_1$ & T_3 were the ongoing transactions {i.e. failure occurred before their commit opn}

\rightarrow We need to "Undo" all the operations of ongoing transactions {i.e. T_1 & T_3 }

$\rightarrow T_2$ committed after last check point, \therefore We need to "redo" the opn of transaction T_2 that appears after the last check point



2 mins Summary



✓ **Topic**

Transaction

✓ **Topic**

ACID properties

✓ **Topic**

Atomicity

Topic

THANK - YOU