Transaction Module-5

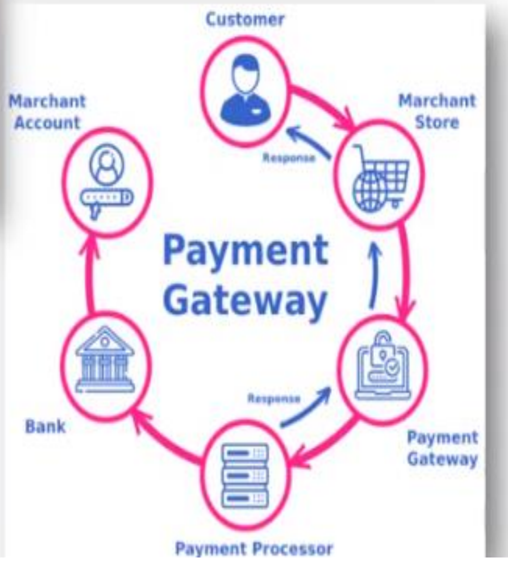
Prepared by,

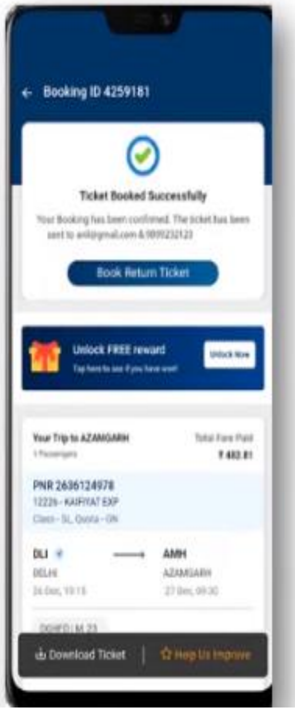
Dr Aruna M G

What is Transection?

- Transaction is a <u>set of operations which are all logically related.</u>
- Transaction is a <u>single logical unit of work</u> formed by a <u>set of operations.</u>

<u>Examples:</u>




Booking ID 4259181

Ticket Booked Successfully

Your Booking has been confirmed. The ticket has been sent to anil@gmail.com & 9899232123

Book Return Ticket

Unlock FREE reward
Tap here to see if you have won!
Unlock Now

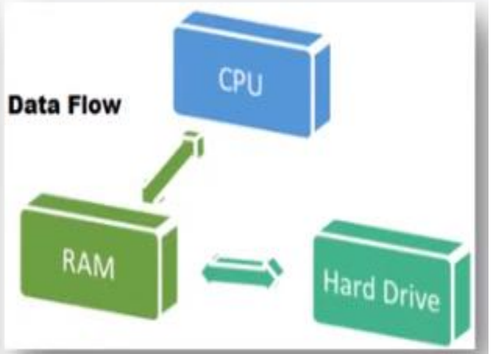
Your Trip to AZAMGARH
1 Passengers
Total Fare Paid
₹ 482.81

PNR 2636124978
12226 - KAIFIYAT EXP
Class - SL, Quota - GN

DLI → AMH
DELHI AZAMGARH
26 Dec, 19:15 27 Dec, 09:30

Download Ticket | Help Us Improve

- Database Connection
- DDL Commands
- DML Commands etc.

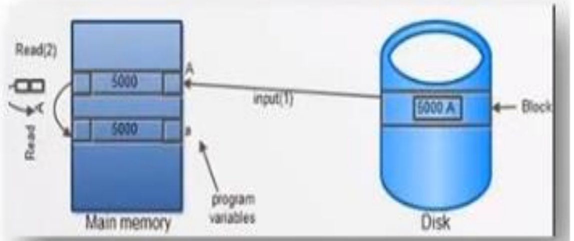


Operations in Transection

The main operations in a transaction are:

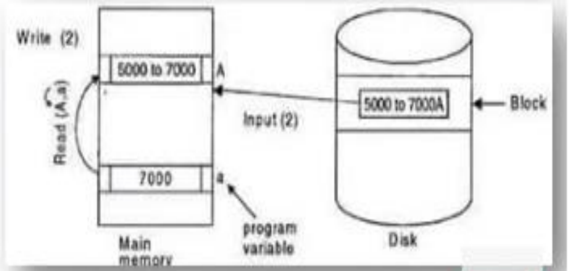
1. **Read Operations:**

- Read operation reads the data from the database and then stores it in the buffer in main memory.
- Example: Read(A) instruction will read the value of A from the database and will store it in the buffer in main memory.

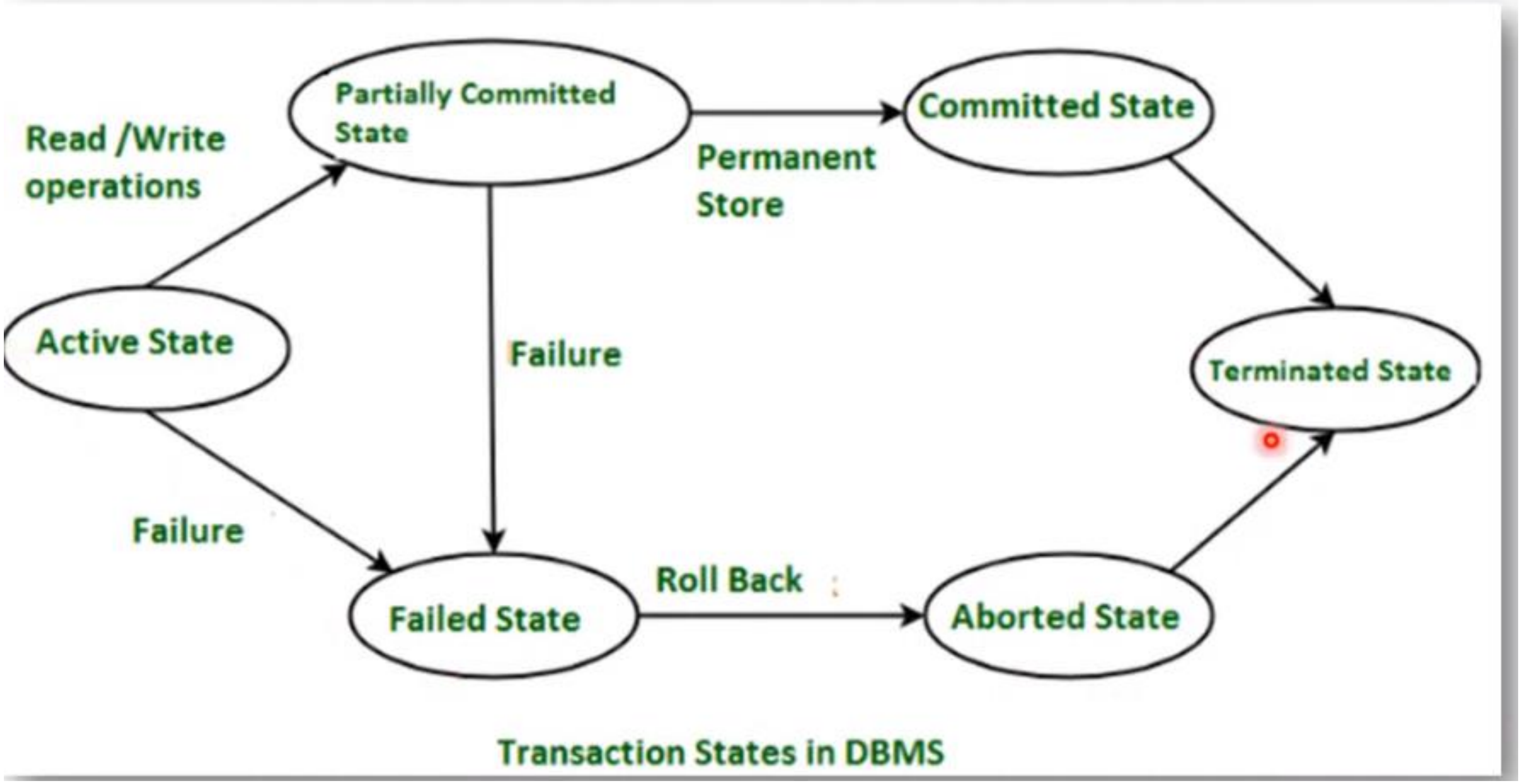


2. **Write Operations:**

- Write operation writes the updated data value back to the database from the buffer.
- Example: Write(A) will write the updated value of A from the buffer to the database.



Transection State

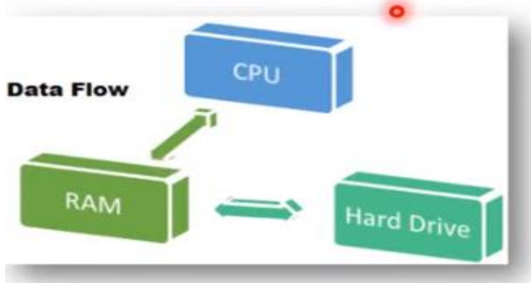


Transaction States in DBMS

1. Active State

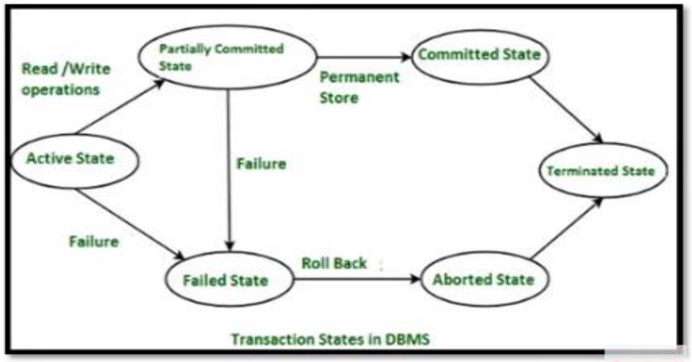
- This is the first state in the life cycle of a transaction.
- A transaction is called in an active state as long as its instructions are getting executed.
- All the changes made by the transaction now are stored in the buffer in main memory.

Examples:



Operating System

ATM



2. Partially Committed State

- After last instruction of transaction has executed, it enters into a partially committed state.
- After completion of all the read & write operation changes are made in main memory or buffer.
- If the changes are made permanent on the Database then the state will change to "committed state" and in case of failure it will go to the "failed state".

3. Committed State

- A transaction is said to be in a committed state if it executes all its operations successfully.
- After all the changes made by the transaction have been successfully stored into the database.
- Now, the transaction is considered to be fully committed.

4. Failed State

- When a transaction is getting executed in the active state or partially committed state and some failure occurs.
- Due to which it becomes impossible to continue the execution, it enters into a failed state.
- Like, Electricity off, Battery down, Server error, No Internet Connection

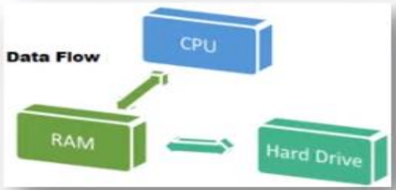
5. Aborted State

- After the transaction has failed and entered into a failed state.
- After aborting the transaction, the database recovery module will select one of the two operations: 1. Re-start the transaction 2. Kill the transaction
- To undo the changes made by the transaction, it becomes necessary to roll back the transaction.
- After the transaction has rolled back completely, it enters into an aborted state.

6. Terminated State

- This is the last state in the life cycle of a transaction.
- After entering the committed state or aborted state, the transaction finally enters into a terminated state where its life cycle finally comes to an end.
- The system is consistent and ready for new transaction and the old transaction is terminated.

Examples:



Operating System

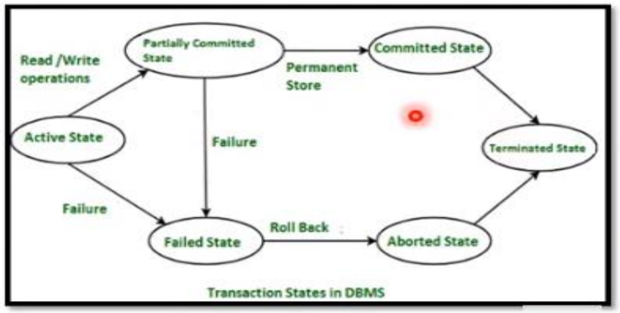
$T2$

$R(A)$
$W(A)$
$R(B)$
$W(B)$
$Com.$

ATM TRANSACTION

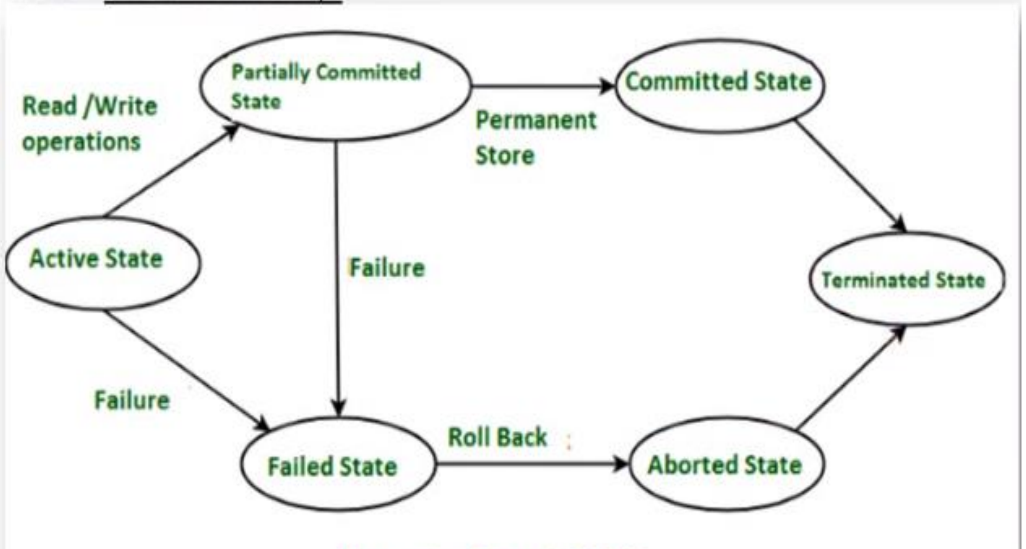
TERMINAL #	[illegible]
SEQUENCE #	[illegible]
DATE	15:18 08/10/2016
CARD NUMBER	[illegible]
CUSTOMER NAME	JOHN EMPTY
REQUESTED AMOUNT	$100.00
TERMINAL FEE	$1.25
TOTAL AMOUNT	$101.25

ATM



Transection State Revision

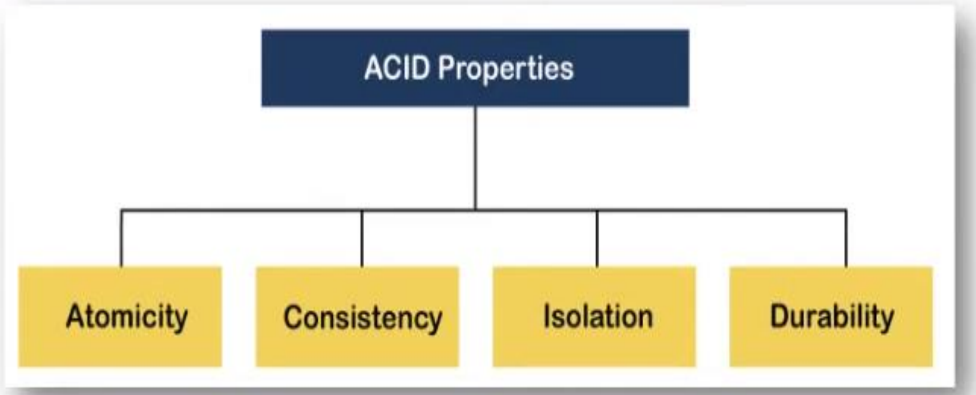
- **Active:** Transaction is <u>executing.</u>
- **Failed:** Transaction <u>fails</u> to complete successfully.
- **Abort:** changes made by transaction are cancelled (<u>roll back</u>).
- **Partially Commit:** Final statement of <u>transaction is executed</u>.
- **Commit:** Transaction completes its execution <u>successfully.</u>
- **Terminated:** Transaction is <u>finished.</u>



Transaction States in DBMS

About ACID Properties

- It is important to ensure that the <u>database remains consistent before and after the transaction.</u>
- To <u>ensure the consistency of database</u>, certain <u>properties are followed by all the transactions</u> occurring in the system.
- These <u>properties are called as ACID Properties</u> of a transaction.

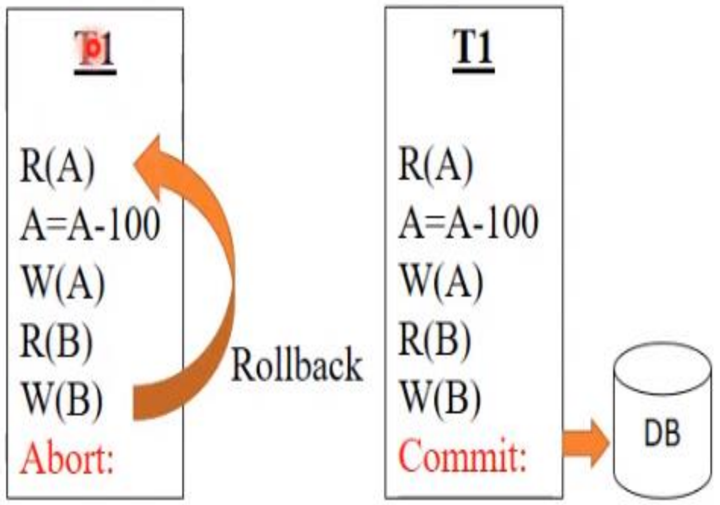


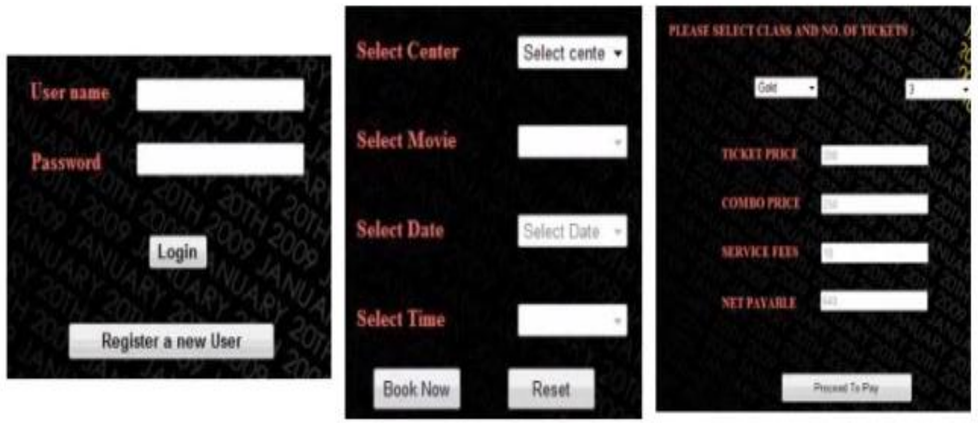
1. Atomicity

➢ **(ALL or NONE)**

- This property ensures that either the transaction occurs completely or it does not occur at all.
- In other words, it ensures that no transaction occurs partially.
- It is the responsibility of Transaction Control Manager to ensure atomicity of the transactions.
- The operations either get Abort or Commit.

Examples:





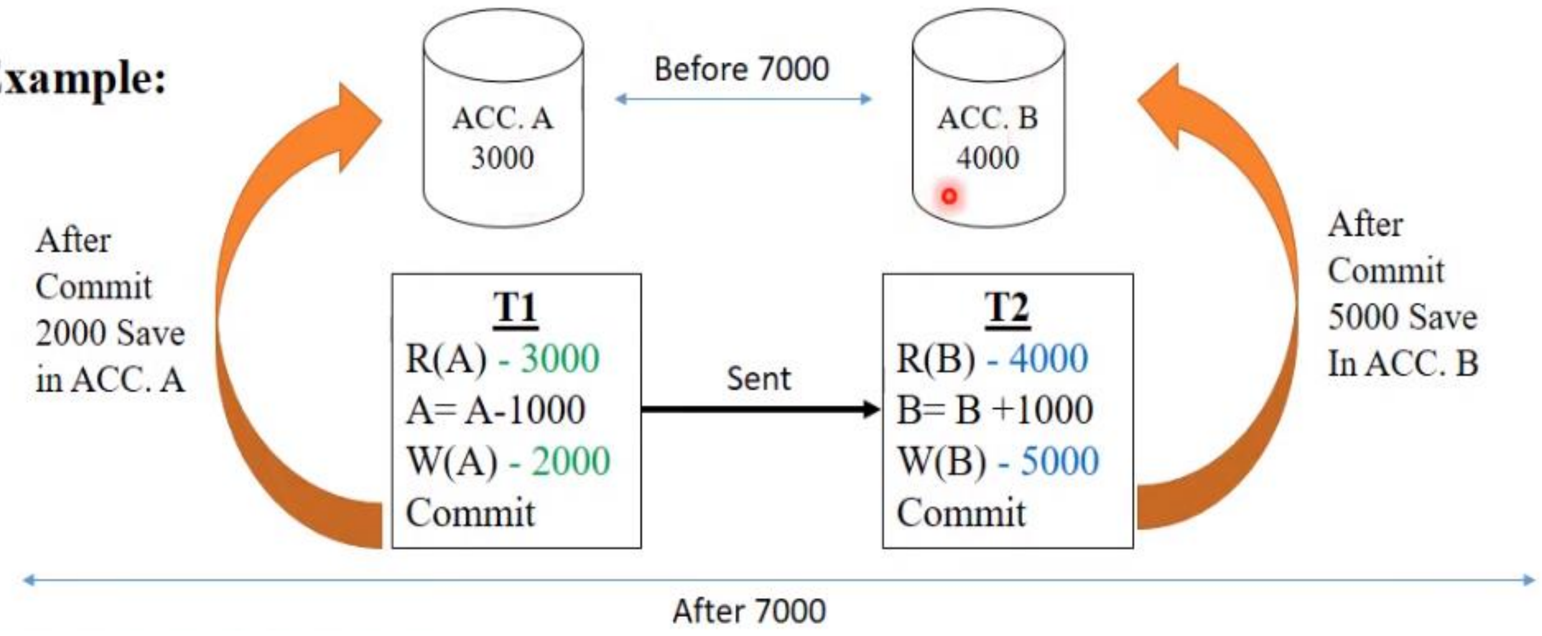
Movie Ticket Booking

2. Consistency

➤ **(Before & After the Transection, Sum same)**

- This property ensures that integrity constraints are maintained.
- It ensures that the database remains consistent before and after the transaction.
- It is the responsibility of DBMS application programmer to ensure consistency of the database.

Example:

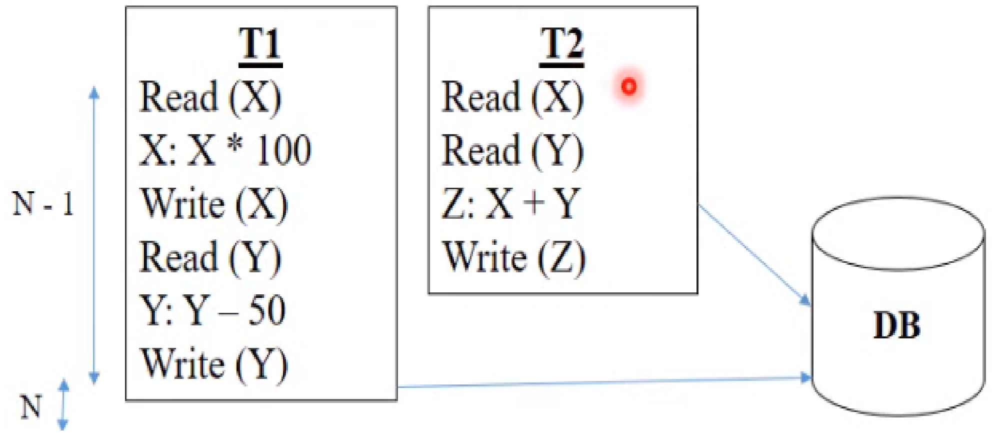


3. Isolation

➢ **(Multiple Transections Occur Independently)**

- The term 'isolation' means separation.
- It means if two operations are being performed on two different databases, they may not affect the value of one another.
- In the case of transactions, when two or more transactions occur simultaneously, the consistency should remain maintained.
- Any changes that occur in any particular transaction will not be seen by other transactions **until** the change is not committed in the memory.

Example:



4. Durability

➢ **(Successful Transection / Permanently Data Stored even if failure occures)**

- Durability ensures the permanency of something.
- This property ensures that all the changes made by a transaction after its successful execution are written successfully to the disk.
- It also ensures that these changes exist permanently and are never lost even if there occurs a failure of any kind.
- The COMMIT command must be used every time we make changes.
- It is the responsibility of Recovery Manager to ensure durability in the database.

About Concurrency Control

- Multiple users can access and use the same database at one time, which is known as the concurrent execution of the database.
- It ensures that Database transactions are performed concurrently and accurately.
- It confirms that produce correct results without violating data integrity of the respective Database.

Example:

✓Two people buy a movie ticket for the same movie and the same show time.

✓There is only one seat left, for the movie show in that particular theatre.

✓Concurrency Problem occur.

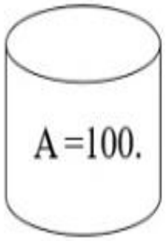
✓Without concurrency control it is not possible to buy ticket.

✓It provides a ticket to the buyer who has completed the transaction process first.

Concurrency Control Problems

1. **Lost Update Problem (W – W Problem):** It occur when multiple transactions select the same row and update the row based on the value selected.

Example:



T1	T2
R(A)	

A = A-50
A=50	

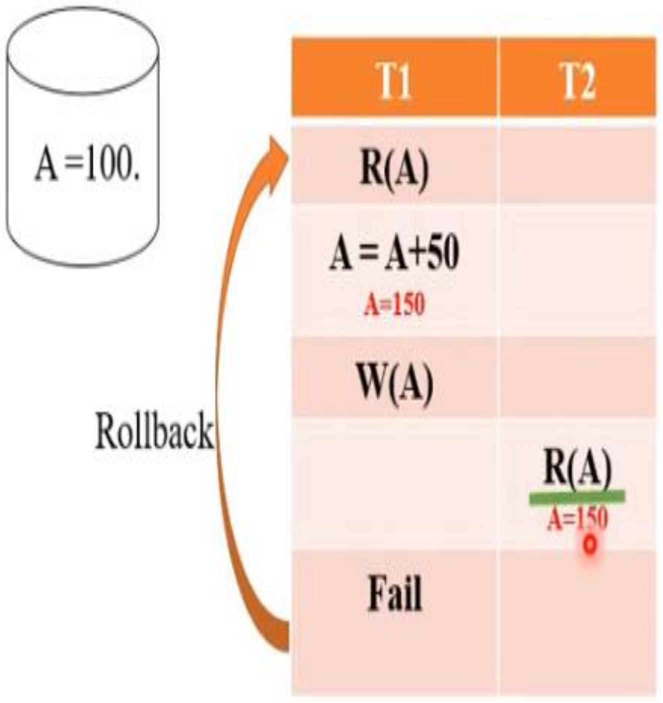
	R(A)

	A=A+100
	A=200

W(A)	
	W(A)

2. **Dirty Read Problem:** When one transaction updates an item of the database, and somehow the transaction fails, and before the data gets rollback, the updated database item is accessed by another transaction.

Example:



T1	T2
R(A)	

A = A+50
A=150	

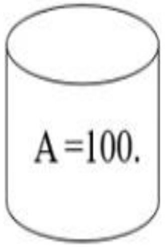
W(A)	

	R(A)
	A=150

Fail	

3. Unrepeatable Read Problem: Also known as Inconsistent Retrievals Problem that occurs when in a transaction, two different values are read for the same database item.

Example:



T1	T2

R(A)
A = 100	

	R(A)

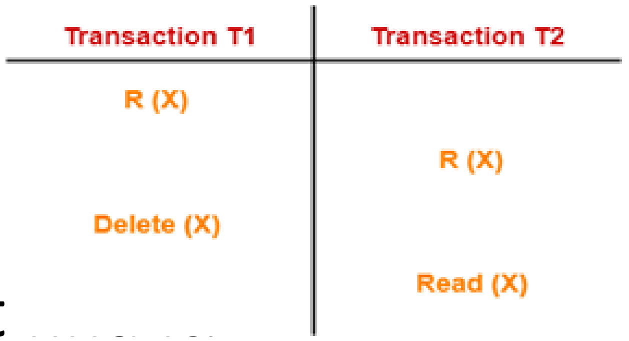
	A=A+100
	A = 200

	W(A)

R(A)
A = 200	

4.Phantom Read Problem

- It occurs when a transaction reads a variable once from the buffer and when it tries to read that same variable again, an error occurs saying that the variable does not exist.
  - T1 reads X.
  - T2 reads X.
  - T1 deletes X.
  - T2 tries reading X but does not



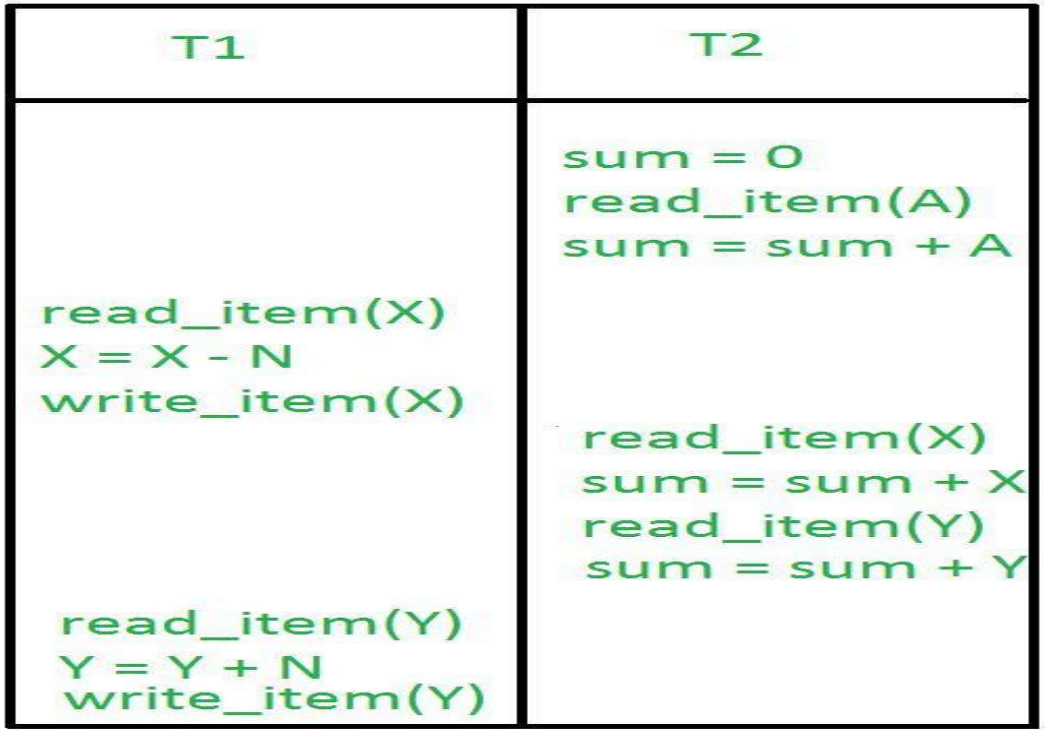
Transaction T1	Transaction T2
R (X)	
	R (X)
Delete (X)	
	Read (X)

Use of Concurrency Control Methods

1. To apply Isolation through mutual exclusion between conflicting transactions
2. To resolve read-write and write-write conflict issues
3. To preserve database consistency through constantly preserving execution obstructions
4. The system needs to control the interaction among the concurrent transactions.
5. Concurrency control helps to ensure serializability.
6. Maintain integrity of the database.

5. Incorrect Summary Problem:

- Consider a situation, where one transaction is applying the aggregate function on some records while another transaction is updating these records. The aggregate function may calculate some values before the values have been updated and others after they are updated.
- **Example:**



T1	T2

	sum = 0
	read_item(A)
	sum = sum + A

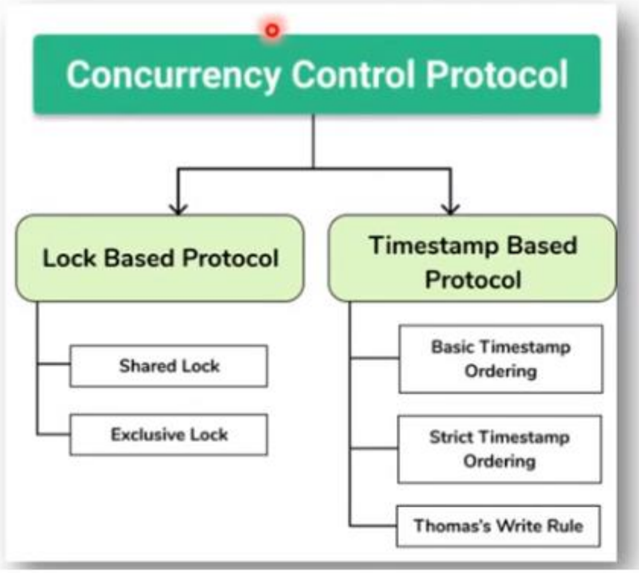
read_item(X)
X = X - N
write_item(X)	

	read_item(X)
	sum = sum + X
	read_item(Y)
	sum = sum + Y

read_item(Y)
Y = Y + N
write_item(Y)	

Concurrency Control

- Concurrency Control is the working concept that is required for controlling and managing the concurrent execution of database operations.
- It avoiding the inconsistencies in the database.
- The concurrency control protocols ensure the atomicity, consistency, isolation, durability and serializability of the concurrent execution of the database transactions



Lock Based Protocol

- It is very essential in concurrency control which controls concurrent access to a data item.
- It ensures that one transaction should not read and write record while another transaction is performing a write operation on it
- A lock is a data variable which is associated with a data item.
- This lock signifies that operations that can be performed on the data item.
- All lock requests are made to the concurrency-control manager.
- Transactions proceed only once the lock request is granted.

Types of Lock Based Protocol:

1. Shared Lock (S)
2. Exclusive Lock (X)

Example:

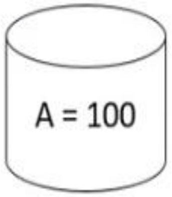
✓In traffic light signal that indicates stop and go, when one signal is allowed to pass at a time and other signals are locked.

✓In the same way in a database transaction, only one transaction is performed at a time meanwhile other transactions are locked

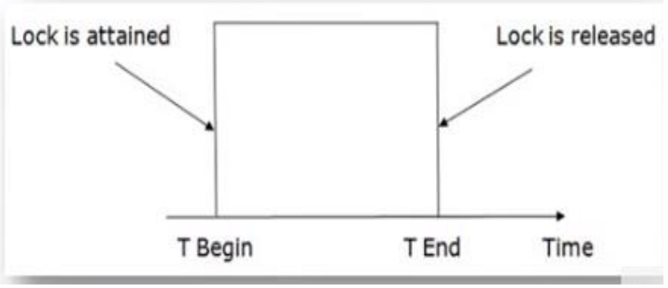
Shared Lock

- It is also known as a Read-only lock.
- Shared locks can only read without performing any changes to it from the database.
- Other transections also read same data item but can't update it until read is completed.
- Shared Locks are represented by S.

Example:



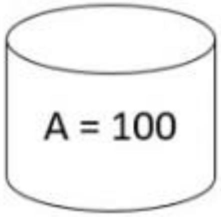
T1	T2
Lock S(A)	
Read (A) A = 100	Read (A) A = 100
Unlock S(A)	
	A = A+50
	W(A) A = 150



Exclusive Lock

- The data item can be both reads as well as written by the transaction.
- In this lock, multiple transactions do not modify the same data simultaneously.
- Exclusive Locks are represented by X.

Example:



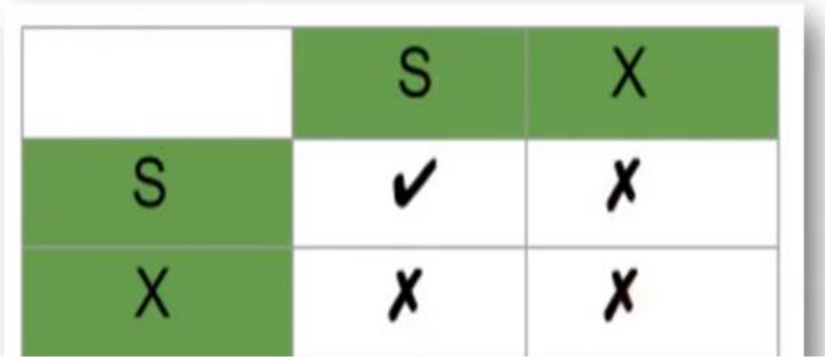
T1	T2
Lock X(A)	
Read (A) A = 100	
A = A + 50	
Write (A) A = 150	
Unlock X(A)	
	Read (A) A = 150
	A = A + 20
	Write (A) A = 170

Lock Compatibility Matrix

- The graphs shows that if two transactions only read the same data object they do not conflict.
- But if one transaction writes a data object and another either read or write the same data object, then they conflict with each other.

➢**S (Shared Lock) -> Read**

➢**X (Exclusive Lock) -> Read & Write**



	S	X
S	✓	✗
X	✗	✗