COURSE NAME: DBMS
COURSE CODE:23AD2102R

TOPIC:

TRANSACTION PROCESSING ISSUES AND TRANSACTION STATES

Session - 2

AIM OF THE SESSION

To familiarize students with the basic concept of **Transaction Management**

INSTRUCTIONAL OBJECTIVES

a) The concept of Transaction
b) Main Operations in a Transaction
c) Popular Examples of Transaction Processing
d) Transaction Processing Issues
e) Transaction States

LEARNING OUTCOMES

At the end of this session, you should be able to understand :

Transaction Processing Issues & States of Transaction Processing

SESSION OBJECTIVES

- At the end of the session the student will understand about

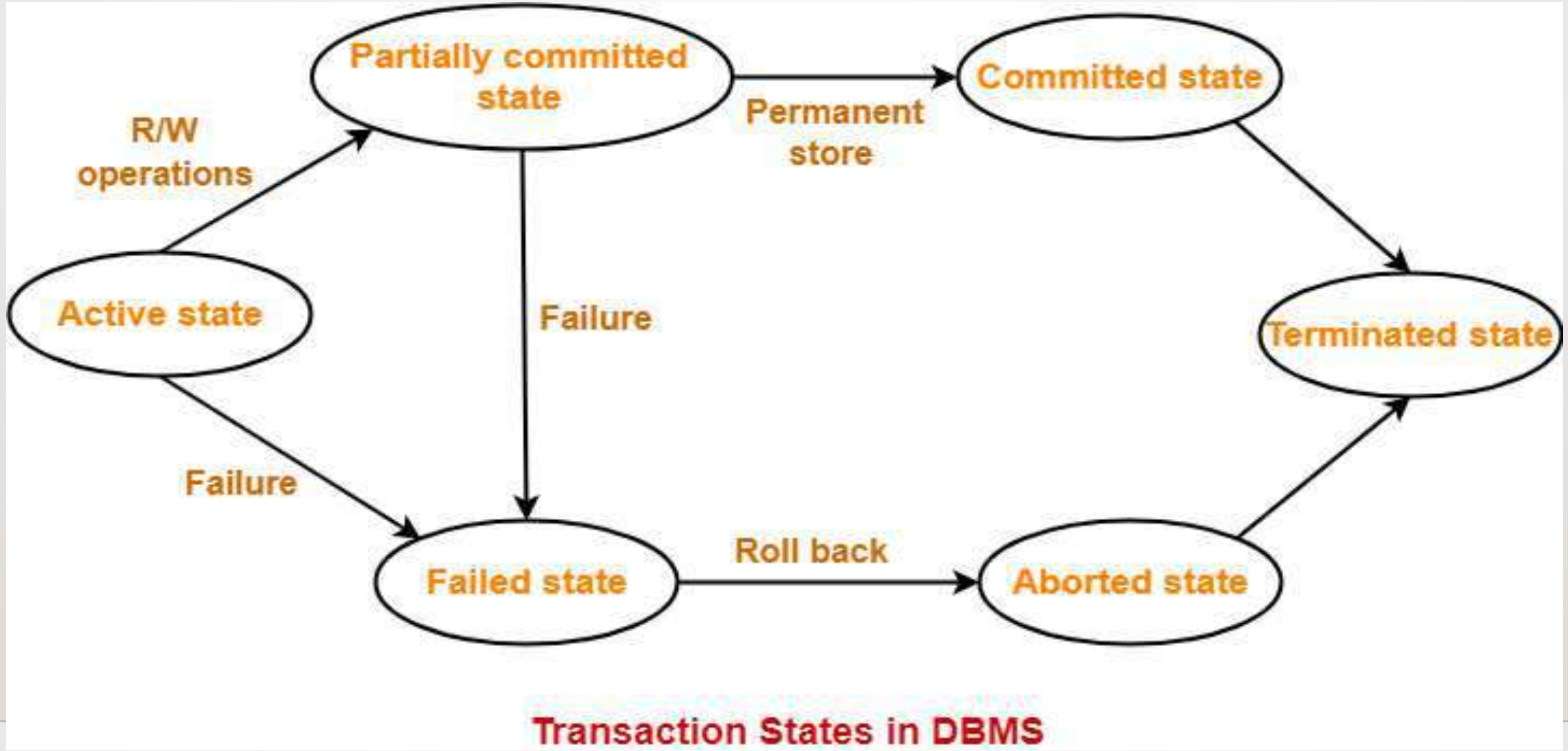
- Transaction States
- The Properties of Transactions
- ACID Properties

TRANSACTION STATES

- A transaction goes through many different states throughout its life cycle.
- These states are called as **transaction states**.
- Transaction states are as follows-

1. ***Active state***
2. ***Partially committed state***
3. ***Committed state***
4. ***Failed state***
5. ***Aborted state***
6. ***Terminated state***

TRANSACTION STATES



TRANSACTION STATES

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.

2. Partially Committed State-

- After the last instruction of transaction has executed, it enters into a **partially committed state**.
- After entering this state, the transaction is considered to be partially committed.
- It is not considered fully committed because all the changes made by the transaction are still stored in the buffer in main memory.

TRANSACTION STATES

3. Committed State-

- After all the changes made by the transaction have been successfully stored into the database, it enters into a **committed state**.
- Now, the transaction is considered to be fully committed.

NOTE-

- After a transaction has entered the committed state, it is not possible to roll back the transaction.
- In other words, it is not possible to undo the changes that has been made by the transaction.
- This is because the system is updated into a new consistent state.
- The only way to undo the changes is by carrying out another transaction called as **compensating transaction** that performs the reverse operations.

TRANSACTION STATES

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**.

5. Aborted State-

- After the transaction has failed and entered into a failed state, all the changes made by it have to be undone.
- 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**.

TRANSACTION STATES

6. Terminated State-

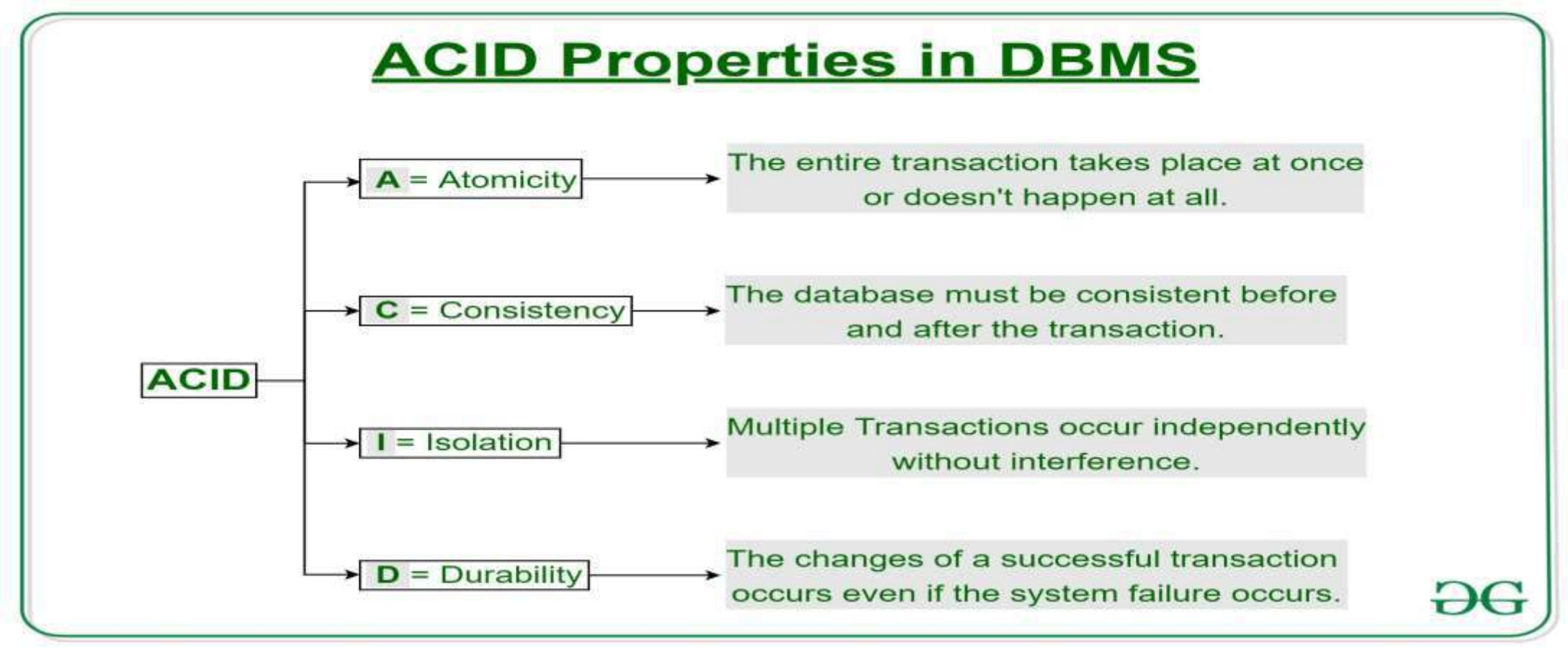
- This is the last state in the life cycle of a transaction.
- After entering the committed state or aborted state, the transaction finally enters into a **terminated state** where its life cycle finally comes to an end.

ACID PROPERTIES-INTRODUCTION

- For successful transaction on a database, it MUST satisfy these ACID properties.
- ACID Properties are important to maintain the database Integrity.
- These days all the Third Party Applications like Banking Apps/Paytm/Gpay/PhonePay etc., implements at the backend.

TRANSACTION PROPERTIES

The transaction has the four properties. These properties used to maintain consistency in a database, before and after the transaction.



ATOMICITY

Either all Successful or ***None***

Entire Transaction MUST Complete or Revert the database to the old state

ATOMICITY

- Example: Let's assume that following transaction T consisting of Transaction T1 and Transaction T2. A consists of Rs 600 and B consists of Rs 300. Transfer Rs 100 from account A to account B.

Transaction T1

Read(A)

A:= A-100

Write(A)

Transaction T2

Read(B)

B:= B+100

Write(B)

After completion of the transaction, A consists of Rs 500 and B consists of Rs 400

BUT……..

- What happens, if Transaction T2 fails to execute after Transaction T1 completes execution……

Problem : Amount will be deducted from A but not reflected/shown in account B. It shows ***inconsistency*** of database.

To overcome the above problem and ensure the correctness of database state, the transaction must be executed in entirety i.e both Transaction T1 and Transaction T2 MUST be executed or None of the Operations MUST be executed. This is called as Atomicity.

ATOMICITY

- Therefore, Atomicity states that all operations of the transaction take place at once if not, the transaction is aborted.
- There is no midway, i.e., the transaction cannot occur partially.
- Each transaction is treated as one unit and either run to completion or is not executed at all.

- Atomicity involves the following two operations:

Abort: If a transaction aborts then all the changes made are not visible.

Commit: If a transaction commits then all the changes made are visible.

WHAT HAPPENS IF SOME OPERATION FAILS AND HENCE THE TRANSACTION FAILS……?

- The DBMS will ***Roll Back*** the operations and start executing the operations again.
- Example : Network /Power Failure after entering PIN number in an ATM. We will start the transaction afresh from the insertion of the card…

CONSISTENCY

- The execution of a transaction will leave a database in either its prior stable state or a new stable state
- That means the integrity constraints are maintained so that the database is consistent before and after the transaction.
- The transaction is used to transform the database from one consistent state to another consistent state.

CONSISTENCY-EXAMPLE

In the previous example, the total amount must be maintained before or after the transaction.

- ***Total before T occurs = 600+300=900***
- ***Total after T occurs= 500+400=900***

- Therefore, the database is consistent.
- In the case when T1 is completed but T2 fails, then inconsistency will occur because of mismatch of balance amounts in the respective accounts.

CONCURRENT TRANSACTIONS

Imagine how Train/Air Reservation System transactions are being done……

A database is a shared resource accessed. It is used by many users and processes concurrently.

Execution of multiple transactions simultaneously is called as Concurrent Transactions execution.

For example, the banking system, railway, and air reservations systems, stock market monitoring, supermarket inventory, and checkouts, etc.

ISOLATION

- This property ensures that multiple transactions can occur simultaneously without causing any inconsistency.
- During execution, each transaction feels as if it is getting executed alone in the system.
- A transaction does not realize that there are other transactions as well getting executed parallelly.
- Changes made by a transaction becomes visible to other transactions only after they are written in the memory.
- The resultant state of the system after executing all the transactions is same as the state that would be achieved if the transactions were executed serially one after the other.
- It is the responsibility of concurrency control manager to ensure isolation for all the transactions.

ISOLATION

- It states that the data which is used at the time of execution of a transaction cannot be used by the second transaction until the first one is completed.
- In isolation, if the transaction T1 is being executed and using the data item X, then that data item can't be accessed by any other transaction T2 until the transaction T1 ends.
- In the given example, the amount Rs 600 and Rs 300 will be the data items and while Rs 100 is being subtracted from Rs 600, the data item Rs 600 should NOT be accessed by any other transaction.
- That means, the data item Rs 600 should be kept ISOLATED from other transactions, till the actual transaction completes.

ISOLATION

- Isolation doesn't necessarily means that the transaction MUST not be deliberately kept isolated, but even though the transaction is executed in concurrency, the execution should behave as if that single transaction is being executed at that time.

- ***Therefore, the database system must take special actions to ensure that transactions operate properly without interference from concurrently executing database statements***

DURABILITY

- It states that the transaction once made, then the changes MUST be permanent.
- They cannot be lost by the erroneous operation of a faulty transaction or by the system failure.
- When a transaction is completed, then the database reaches a state known as the consistent state.
- That consistent state cannot be lost, even in the event of a system's failure.
- Here, in the given example, the balance in A's account MUST show Rs 500 and balance of B MUST show Rs 400 , till another successful transaction being executed on the data items Rs 500 and Rs 400.

CONCURRENCY PROBLEMS

- When multiple transactions execute concurrently in an uncontrolled or unrestricted manner, then it might lead to several problems.
- Such problems are called as **concurrency problems**.

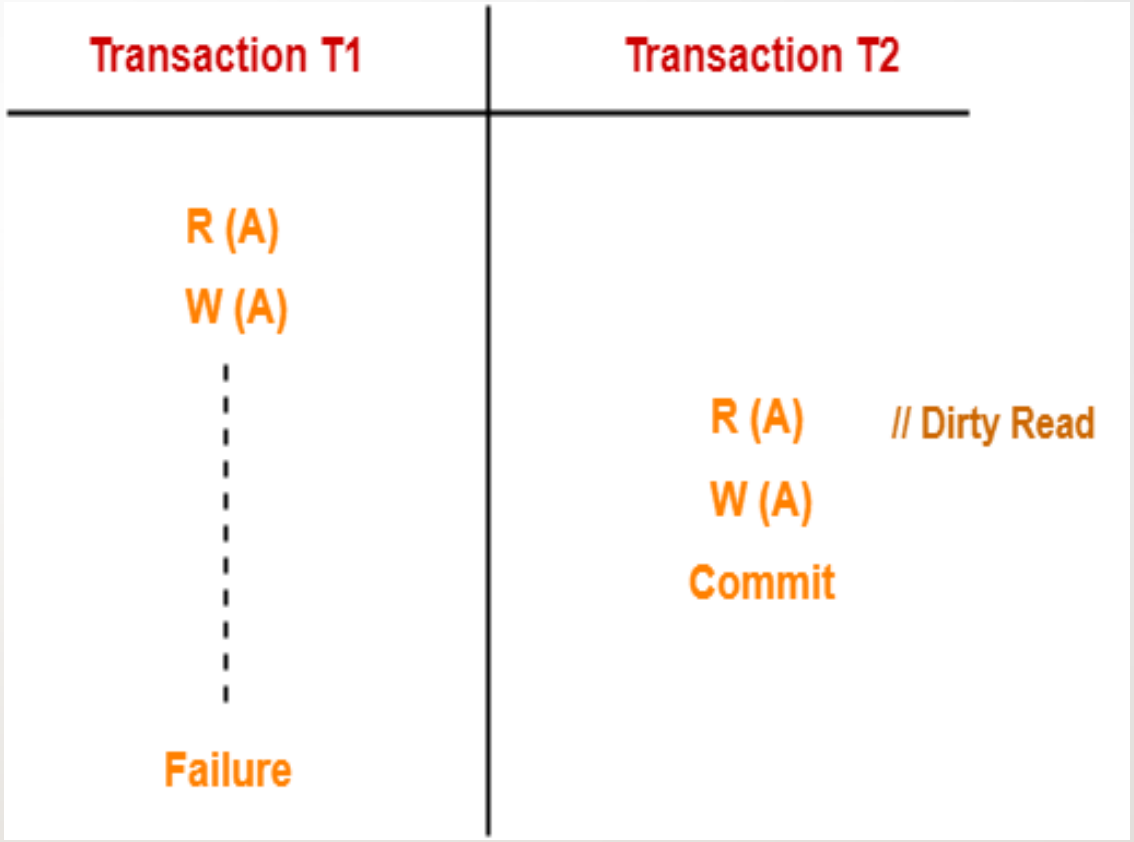
DIRTY READ PROBLEM(W-R CONFLICT)

- The dirty read problem occurs *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.*
- *Also called as Write-Read(W-R Conflict) between the transactions.*

DIRTY ***READ*** PROBLEM

Reading the data written by an uncommitted transaction is called as dirty read.

Here,
1.T1 reads the value of A.
2.T1 updates the value of A in the buffer.
3.T2 reads the value of A from the buffer.
4.T2 writes the updated the value of A.
5.T2 commits.
6.T1 fails in later stages and rolls back



In this example,
•T2 reads the dirty value of A written by the uncommitted transaction T1.
•T1 fails in later stages and roll backs.
•Thus, the value that T2 read now stands to be incorrect.
•Therefore, database becomes inconsistent.

DIRTY *READ* PROBLEM-EXAMPLE 2

- **Consider two transactions T_X and T_Y in the below diagram performing read/write operations on account A where the available balance in account A is $300:**

Time	T_x	T_y
t_1	READ (A)	——
t_2	A = A + 50	——
t_3	WRITE (A)	——
t_4	——	READ (A)
t_5	SERVER DOWN ROLLBACK	——

DIRTY READ PROBLEM

DIRTY READ PROBLEM EXAMPLE 2

- At time t1, transaction T_X reads the value of account A, i.e., $300.
- At time t2, transaction T_X adds $50 to account A that becomes $350.
- At time t3, transaction T_X writes the updated value in account A, i.e., $350.
- Then at time t4, transaction T_Y reads account A that will be read as $350.
- Then at time t5, transaction T_X rollbacks due to server problem, and the value changes back to $300 (as initially).
- But the value for account A remains $350 for transaction T_Y as committed, which is the dirty read and therefore known as the Dirty Read Problem.

UNREPEATABLE READ PROBLEM

- This problem occurs when a transaction gets to read unrepeated i.e. different values of the same variable in its different read operations even when it has not updated its value.

Here,

T1 reads the value of X (= 10 say).

T2 reads the value of X (= 10).

T1 updates the value of X (from 10 to 15 say) in the buffer.

T2 again reads the value of X (but = 15).

Transaction T1	Transaction T2
R (X)	
	R (X)
W (X)	
	R (X) // Unrepeated Read

- ***In this example,***
- ***T2 gets to read a different value of X(=15) in its second reading.***
- ***T2 wonders how the value of X got changed because according to it, it is running in isolation.***

LOST UPDATE PROBLEM

- This problem occurs when multiple transactions execute concurrently and updates from one or more transactions get lost.
- Here,

1. T1 reads the value of A (= 10 say).
2. T2 updates the value to A (= 15 say) in the buffer.
3. T2 does blind write A = 25 (write without read) in the buffer.
4. T2 commits.
5. When T1 commits, it writes A = 25 in the database.

Transaction T1	Transaction T2
R (A)	
W (A)	
⋮	W (A)
	Commit
Commit	

- **In this example,**
- **T1 writes the over written value of X in the database.**
- **Thus, update from T1 gets lost.**

- ***This problem occurs whenever there is a write-write conflict.***
- ***In write-write conflict, there are two writes one by each transaction on the same data item without any read in the middle.***

PHANTOM READ PROBLEM

- This problem occurs when a transaction reads some variable from the buffer and when it reads the same variable later, it finds that the variable does not exist.
- Here,

1. T1 reads X.
2. T2 reads X.
3. T1 deletes X.
4. T2 tries reading X but does not find it.

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

- ***In this example,***
- ***T2 finds that there does not exist any variable X when it tries reading X again.***
- ***T2 wonders who deleted the variable X because according to it, it is running in isolation.***

AVOIDING CONCURRENCY PROBLEMS

- To ensure consistency of the database, it is very important to prevent the occurrence of above problems.
- **Concurrency Control Protocols** help to prevent the occurrence of above problems and maintain the consistency of the database.

SUMMARY OF THE SESSION

- For successful transaction on a database, it MUST satisfy FOUR properties.
- ACID(Atomicity, Consistency, Isolation and Durability)
- Atomicity ensures all the operations of a transaction be executed ***either all Successful*** or ***None***.
- Consistency ensures the data will ***remain consistent*** before and after the completion of the transaction.
- Isolation ensures the data item be separated from the other transactions while one transaction is operating on it.
- Durability ensures the updates after the successful transaction remain permanently(till next transaction starts)

SUMMARY OF SESSION

- Collections of operations that form a single logical unit of work
- The transaction is a set of logically related operation. It contains a group of tasks.
- A transaction is an action or series of actions.
- A transaction is performed by a single user to perform operations for accessing the contents of the database.
- Every Transaction starts with ***read ( )*** operation and ends with ***write( )*** Operation, followed by ***commit*** operation.
- Transaction processing is designed to maintain database integrity (the consistency of related data items) in a known, consistent state.

SELF-ASSESSMENT QUESTIONS

1. A True Transaction updates the Database

(a) True
(b) False
(c) Can't Say
(d) None of the Above

2. At the end of every transaction, every transaction concludes with

(a) Read ( ) Operation
(b) Commit ( ) Operation
(c) Roll Back ( ) Operation
(d) None of the Above

TERMINAL QUESTIONS

1. Discuss about the Concept of Transaction Processing
2. List out the Steps of Operations in a Transaction Processing
3. Why a simple Operation on a database can not be a Transaction ?
4. Discuss about various Issues during a Transaction ?
5. Draw the diagram of Transaction States and Explain ?

REFERENCES FOR FURTHER LEARNING OF THE SESSION

Reference Books:

1. "Database Management Systems" by Raghu Ramakrishnan and Johannes Gehrke - This is a comprehensive textbook that covers both theoretical and practical aspects of DBMS, including data modeling, relational algebra, SQL, query optimization, transaction management, and distributed databases.

2. "Fundamentals of Database Systems" by Ramez Elmasri and Shamkant Navathe - This is another popular textbook that covers the fundamental concepts of DBMS, including database design, normalization, SQL programming, transaction management, and concurrency control.

Sites and Web links:

1. https://www.geeksforgeeks.org/shadow-paging-dbms/
2. https://www.tutorialspoint.com/what-is-shadow-paging-in-dbms
3. https://whatisdbms.com/shadow-paging-in-dbms/

THANK YOU

(DEEMED TO BE UNIVERSITY)

Team – DBMS