TRANSACTION

- Why we study transaction?
  - According to general computation principle (operating system) we may have partially executed program, as the level of atomicity is instruction i.e. either an instruction is executed completely or not
  - But in DBMS view, user perform a logical work(operation) which is always atomic in nature i.e. either operation is execute or not executed, there is no concept like partial execution. For example, Transaction T_1 which transfer 100 units from account A to B

T_1
Read(A)
A = A-100
Write(A)
Read(B)
B = B+100
Write(B)

- In this transaction if a failure occurs after Read(B) then the final statue of the system will be inconsistent as 100 units are debited from account A but not credited in account B, this will generate inconsistency.
- Here for 'consistency' before (A + B) == after (A + B)"

T_1
Read(A)
A = A-100
Write(A)
Read(B)
B = B+100
Write(B)

What is transaction

- To remove this partial execution problem, we increase the level of atomicity and bundle all the instruction of a logical operation into a unit called transaction.

- So formally 'A transaction is a Set of logically related instructions to perform a logical unit of work'.

T_1
Read(A)
A = A-100
Write(A)
Read(B)
B = B+100
Write(B)

- As here we are only concerned with DBMS so we well only two basic operation on database

- **READ (X)** - Accessing the database item x from disk (where database stored data) to memory variable also name as X.

- **WRITE (X)** - Writing the data item from memory variable X to disk.

Q A transaction can include following basic database access operations: **(NET-JUNE-2011)**

(A) Read_item(X)

(B) Write_item(X)

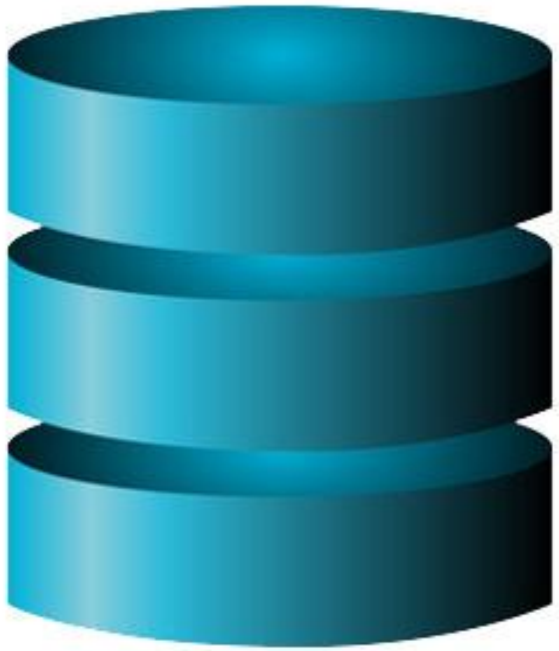
(C) Both (A) and (B)

(D) None of these

Desirable properties of transaction

- Now as the smallest unit which have atomicity in DBMS view is transaction, so if want that our data should be consistent then instead of concentrating on data base, we must concentrate on the transaction for our data to be consistent.

T_1
Read(A)
A = A-100
Write(A)
Read(B)
B = B+100
Write(B)



- Transactions should possess several properties, often called the **ACID** properties; to provide integrity and consistency of the data in the database. The following are the ACID properties:

A = Atomicity

C = Consistency

I = Isolation

D = Durability

- **Atomicity -** A transaction is an atomic unit of processing; it should either be performed in its entirety or not performed at all.

- It is the responsibility of ***recovery control manager / transaction control manager of DBMS*** to ensure atomicity

- **Consistency** - A transaction should be consistency preserving, meaning that if it is completely executed from beginning to end without interference from other transactions, it should take the database from one consistent state to another.

- The definition of consistency may change from one system to another. The preservation of consistency of database is the responsibility of programmers(users) or the DBMS modules that enforces integrity constraints.

- <u>**Isolation**</u> **-** A transaction should appear as though it is being executed in isolation from other transactions, even though many transactions are executing concurrently.

- That is, the execution of a transaction should not be interfered with by any other transactions executing concurrently.

- The isolation property of database is the responsibility of ***concurrency control manager of database***.

- **Durability -** The changes applied to the database by a committed transaction must persist in the database.

- These changes must not be lost because of any failure. It is the responsibility of ***recovery control manager of DBMS.***

Q NOT a part of the **ACID** properties of database transactions? **(GATE- 2016) (1 Marks)**

(a) Atomicity

(b) Consistency

(c) Isolation

(d) Deadlock-freedom

Q Consider the following transaction involving two bank accounts x and y. **(GATE - 2015) (1 Marks)**

```
read(x);
x: = x – 50;
write(x);
read(y);
y: = y + 50;
write(y)
```

The constraint that the sum of the accounts x and y should remain constant is that of

(A) Atomicity **(B)** Consistency

(C) Isolation **(D)** Durability

Q All Oracle transactions obey the basic properties of a database transaction. What is the name of the following property?
'All tasks of a transaction are performed or none of them are. There are no partial transactions.' **[Asked in Wipro NLTH 2021]**

a) Atomicity

b) Durability

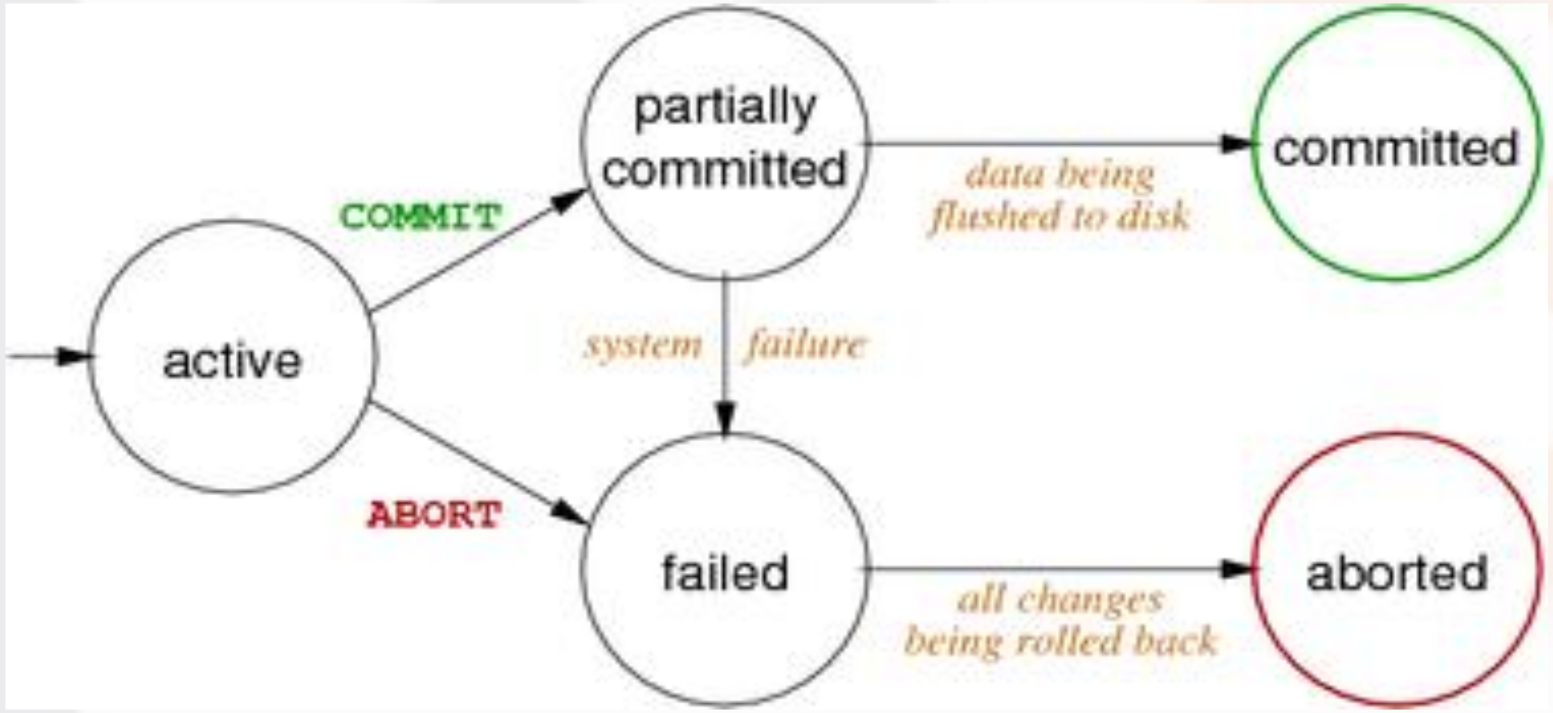
c) Consistency

d) Isolation

Break

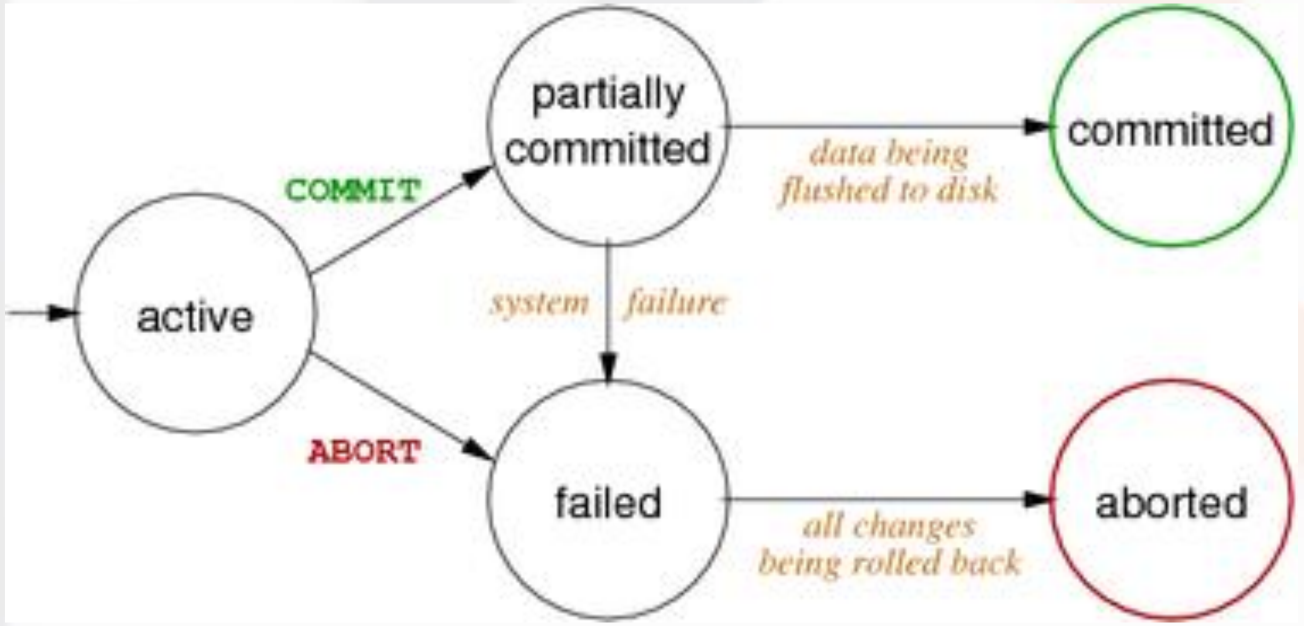
Transaction states

- **ACTIVE** - It is the initial state. Transaction remains in this state while it is executing operations.



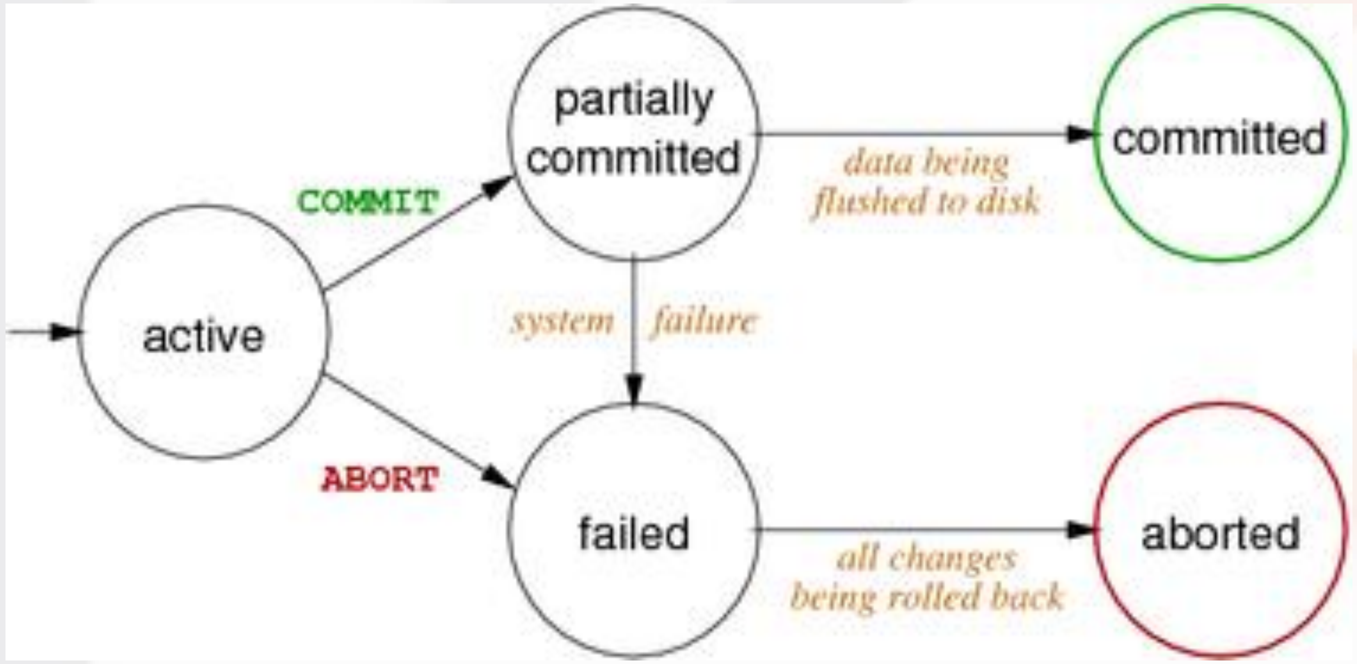
Transaction states

- **PARTIALLY COMMITTED** - After the final statement of a transaction has been executed, the state of transaction is partially committed as it is still possible that it may have to be aborted (due to any failure) since the actual output may still be temporarily residing in main memory and not to disk.



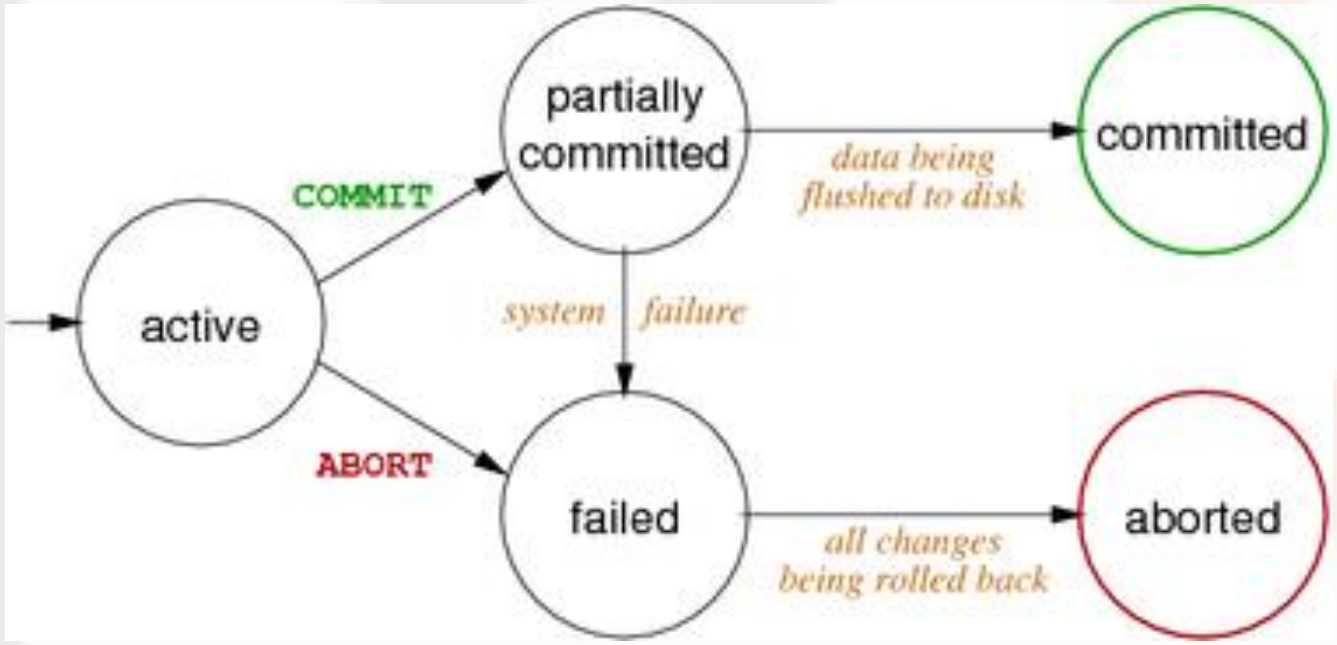
Transaction states

- **FAILED** - After the discovery that the transaction can no longer proceed (because of hardware /logical errors). Such a transaction must be rolled back



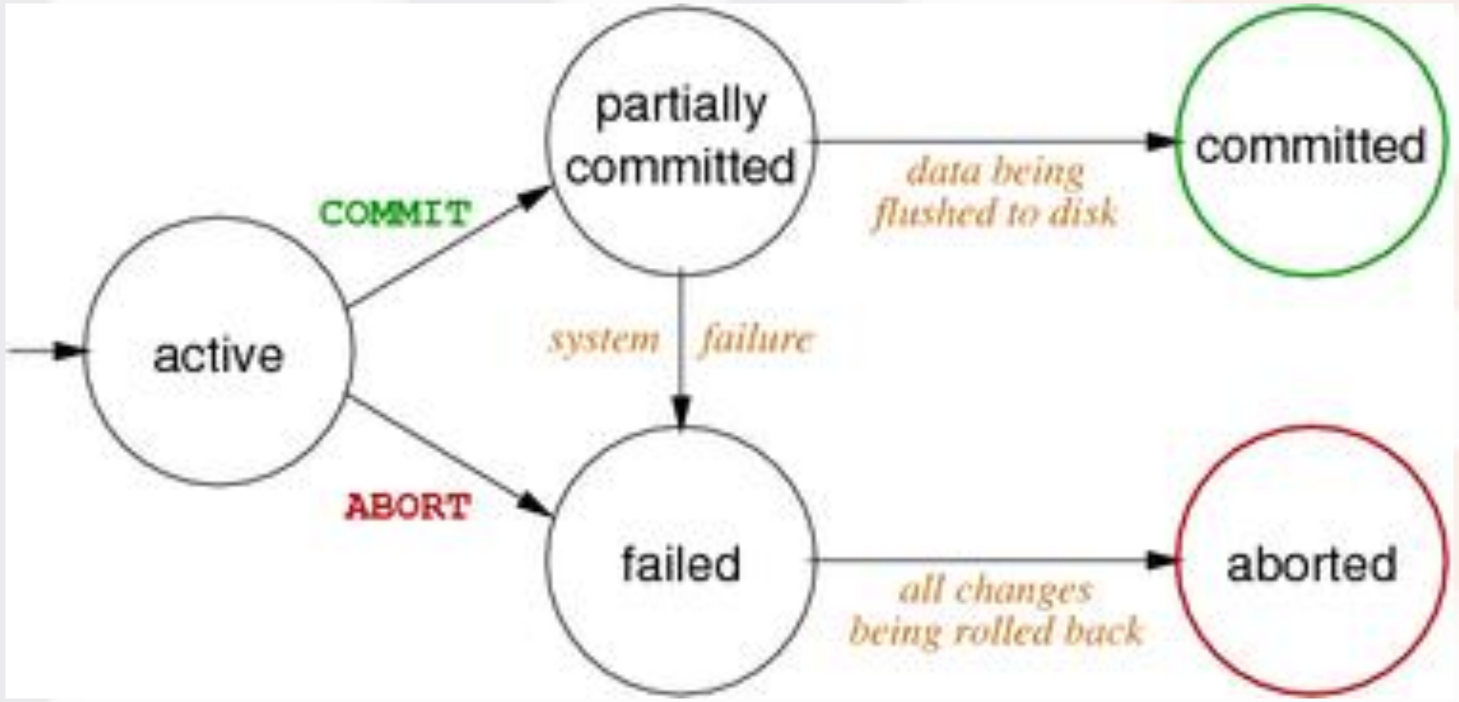
Transaction states

- **ABORTED** - A transaction is said to be in aborted state when the when the transaction has been rolled back and the database has been restored to its state prior to the start of execution.



Transaction states

- **COMMITTED** - A transaction enters committed state after successful completion of a transaction and final updation in the database



Q if the transaction is in which of the state that we can guarantee that data base is in consistent state

a) aborted

b) committed

c) both aborted & committed

d) none

Q Match:

Column I	Column II
1. Atomicity	A. Recovery Manager
2. Durability	B. Concurrency control manager
3. Isolation	C. Programmer
4. Consistency	

a) 1-a, 2–a, 3-b, 4–c

b) 1-a, 2-b,3-b,4-c

c) 1-a,2-a,3-b,4-b

d) none of these

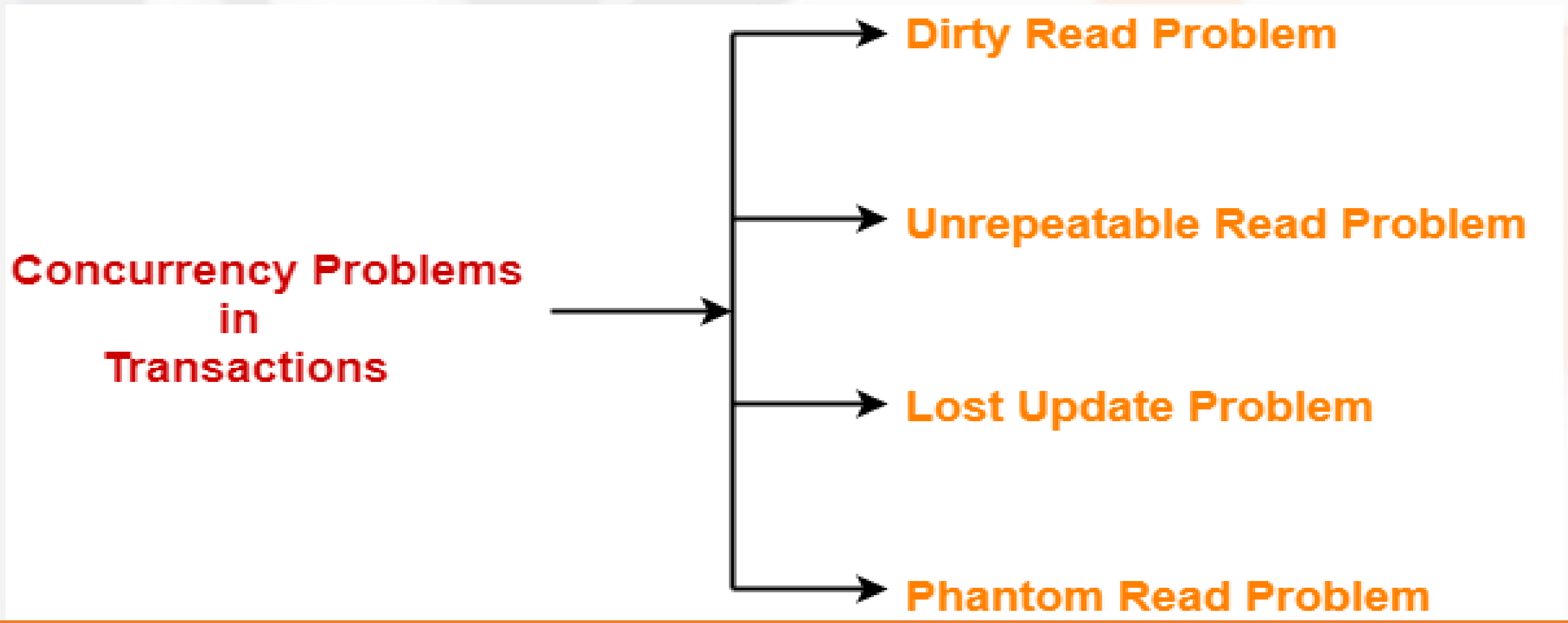
Break

Why we need concurrent execution

- Concurrent execution is necessary because-
  - It leads to good database performance , less weighting time.
  - Overlapping I/O activity with CPU increases throughput and response time.

PROBLEMS DUE TO CONCURRENT EXECUTION OF TRANSACTION

- But interleaving of instructions between transactions may also lead to many problems that can lead to inconsistent database.
- Sometimes it is possible that even though individual transaction are satisfying the acid properties even though the final statues of the system will be inconsistent.


Concurrency Problems in Transactions
Dirty Read Problem
Unrepeatable Read Problem
Lost Update Problem
Phantom Read Problem

Lost update problem / Write - Write problem-

- If there is any two write operation of different transaction on same data value, and between them there is no read operations, then the second write over writes the first write.

T_1	T_2
Read(A)	
Write(A)	
	Write(A)
	Commit
Commit	

Dirty read problem/ Read -Write problem

- In this problem, the transaction reads a data item updated by another uncommitted transaction, this transaction may in future be aborted or failed.
- The reading transactions end with incorrect results.

T_1	T_2
Read(A)	
Write(A)	
	Read(A)
	Commit
Abort	

Q The problem that occurs when one transaction updates a database item and then the transaction fails for some reason is ________. **(NET-JUNE-2012)**

(A) Temporary Select Problem

(B) Temporary Modify Problem

(C) Dirty Read Problem

(D) None

Unrepeatable read problem

- When a transaction tries to read a value of a data item twice, and another transaction updates the data item in between, then the result of the two read operation of the first transaction will differ, this problem is called, Non-repeatable read problem

T_1	T_2
Read(A)	
	Read(A)
	Write(A)
Read(A)	

Phantom read problem

T_1	T_2
Read(A)	
	Delete(A)
Read(A)	

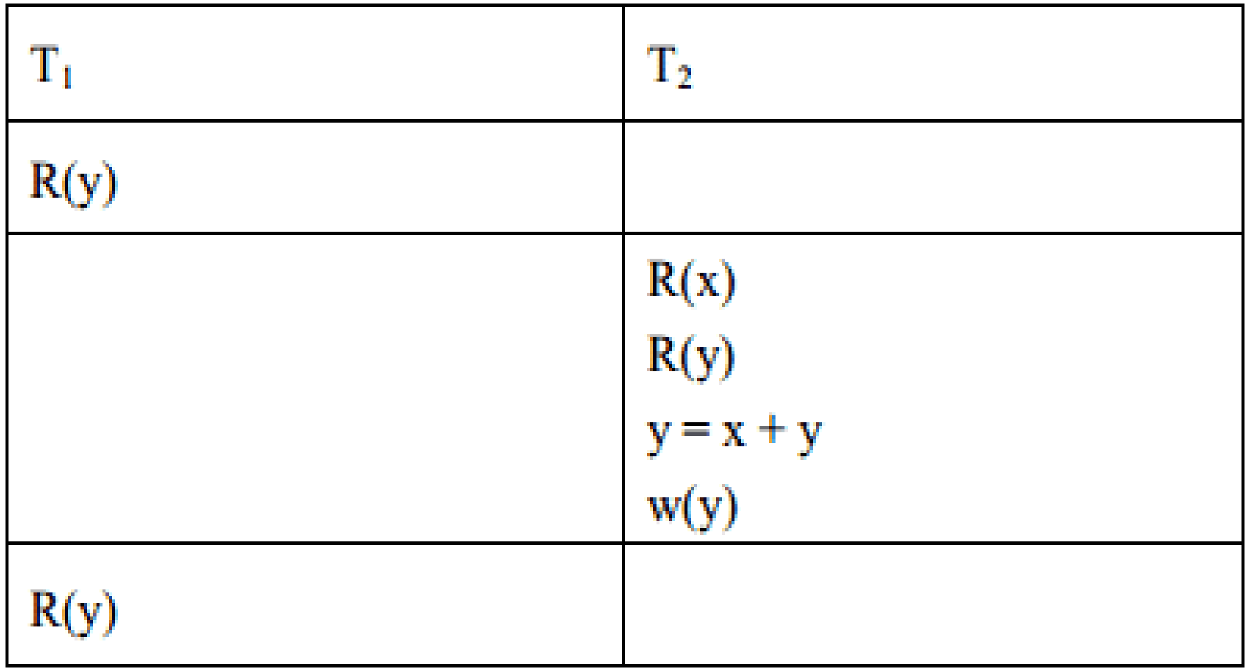
Q The following schedule is suffering from

a) Lost update problem

b) Unrepeatable read problem

c) Both A and B

d) Neither A and B



T_1	T_2
R(y)	

	R(x)
	R(y)
	y = x + y
	w(y)

R(y)	

Q Which of the following scenarios may lead to an irrecoverable error in a database system? **(GATE – 2008) (1 Marks)**

(A) A transaction writes a data item after it is read by an uncommitted transaction

(B) A transaction reads a data item after it is read by an uncommitted transaction

(C) A transaction reads a data item after it is written by a committed transaction

(D) A transaction reads a data item after it is written by an uncommitted transaction

Break

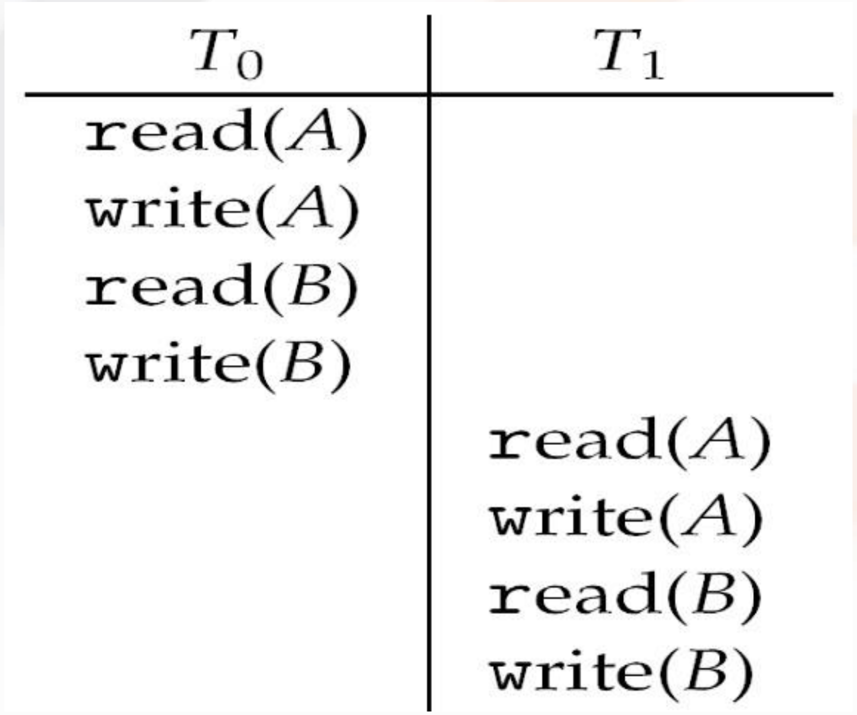
Solution is Schedule

- When two or more transaction executed together or one after another then they can be bundled up into a higher unit of execution called schedule.

- A **schedule** of *n* transactions $T_1, T_2, \ldots, T_n$ is an ordering of the operations of the transactions. Operations from different transactions can be interleaved in the schedule *S*.

- However, schedule for a set of transaction must contain all the instruction of those transaction, and for each transaction T_i that participates in the schedule *S*, the operations of T_i in *S* must appear in the same order in which they occur in T_i.

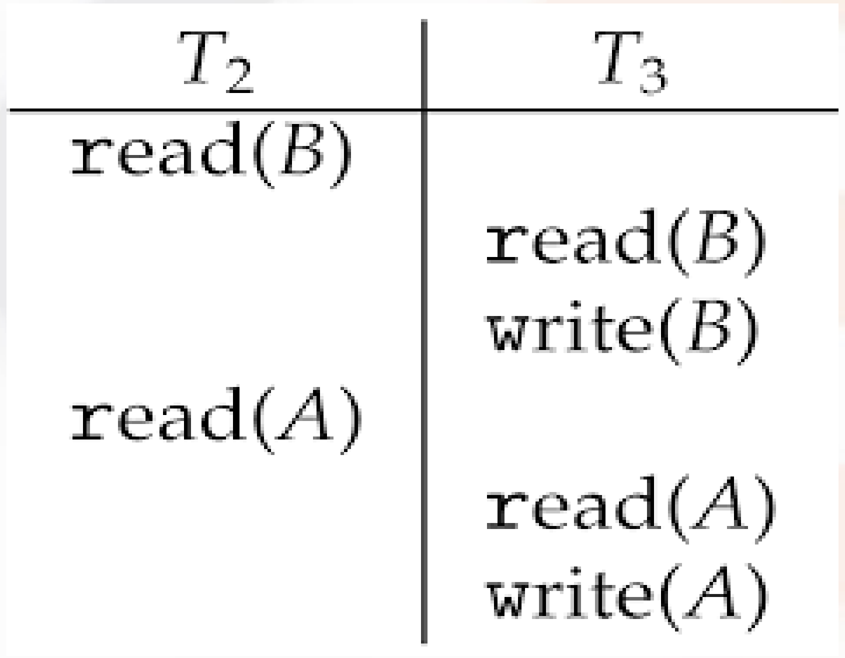
- **Serial schedule** - A serial schedule consists of sequence of instruction belonging to different transactions, where instructions belonging to one single transaction appear together. Before complete execution of one transaction another transaction cannot be started.



T_0	T_1
read(A)	
write(A)	
read(B)	
write(B)	
	read(A)
	write(A)
	read(B)
	write(B)

- For a set of n transactions, there exist **n!** different valid serial schedules. Every serial schedule lead database into consistent state. Throughput of system is less.

- **Non-serial schedule** - A schedule in which sequence of instructions of a transaction appear in the same order as they appear in individual transaction but the instructions may be interleaved with the instructions of different transactions i.e. concurrent execution of transactions takes place.



T_2	T_3
read(B)	
	read(B)
	write(B)
read(A)	
	read(A)
	write(A)

- So the number of schedules for n different transaction T_1,T_2,T_3,---------T_N where each transaction conations n_1, n_2, n_3, -----------,n_4 respectively will be.
- $\{(n_1 + n_2 + n_3 + \text{-----------} + n_n)!\} / (n_1!\ n_2!\ \ n_3!\ \text{-----------} n_n!)\}$
- $\{(n_1 + n_2 + n_3 + \text{-----------} + n_n)!\} / (n_1!\ n_2!\ \ n_3!\ \text{-----------} n_n!)\} - n!$

- **Conclusion of schedules**
  - We do not have any method to proof that a schedule is consistent, but from the above discussion we understand that a serial schedule will always be consistent, so if somehow we proof that a non-serial schedule will also have same effects as of a serial schedule that we get a proof that, this particular non-serial schedule will also be consistent “find those schedules that are logically equal to serial schedules”.

  - For a concurrent schedule to result in consistent state, it should be equivalent to a serial schedule. i.e. it must be serializable.

- On the basis of SERIALIZABILITY
  - Conflict serializable
  - View serializable
  - Result Equivalent
- On the basis of RECOVERABILITY
  - Recoverable
  - Cascadeless
  - Strict

T_1	T_2
R(A)	
	R(B)

T_1	T_2
	R(B)
R(A)	

T_1	T_2
R(A)	
	W(A)

T_1	T_2
	W(A)
R(A)	

T_1	T_2
R(A)	
	W(B)

T_1	T_2
	W(B)
R(A)	

T_1	T_2
	R(A)
W(A)	

T_1	T_2
W(A)	
	R(A)

T_1	T_2
R(A)	
	R(A)

T_1	T_2
	R(A)
R(A)	

T_1	T_2
	W(A)
W(A)	

T_1	T_2
W(A)	
	W(A)

SERIALIZABILITY

- ***Conflicting instructions*** **-** Let I and J be two consecutive instructions belonging to two different transactions T_i and T_j in a schedule S, the possible I and J instruction can be as-
  - I= READ(Q), J=READ(Q) ->*Non-conflicting*
  - I= READ(Q), J=WRITE(Q) ->*Conflicting*
  - I= WRITE(Q), J=READ(Q) ->*Conflicting*
  - I= WRITE(Q), J=WRITE(Q) ->*Conflicting*

- So, the instructions I and J are said to be conflicting, if they are operations by different transactions on the same data item, and at least one of these instructions is a write operation.

Conflict equivalent – if one schedule can be converted to another schedule by swapping of non-conflicting instruction then they are called conflict equivalent schedule.

T_1	T_2
R(A)	
A=A-50	
	R(B)
	B=B+50
R(B)	
B=B+50	
	R(A)
	A=A+10

T_1	T_2
	R(B)
	B=B+50
R(A)	
A=A-50	
R(B)	
B=B+50	
	R(A)
	A=A+10

Q Consider a schedule of transactions T_1 and T_2:
Here, RX stands for "Read(X)" and WX stands for "Write(X)". Which one of the following schedules is conflict equivalent to the above schedule?**(Gate-2020) (2 Marks)**

S	
T_1	T_2
RA	
	RB
	WB
RC	
	RD
WD	
	WC
WB	
Commit	
	Commit

a)

S	
T_1	T_2
	RB
	WB
	RD
RA	
RC	
WD	
WB	
	WC
Commit	
	Commit

b)

S	
T_1	T_2
RA	
RC	
WD	
WB	
	RB
	WB
	RD
	WC
Commit	
	Commit

c)

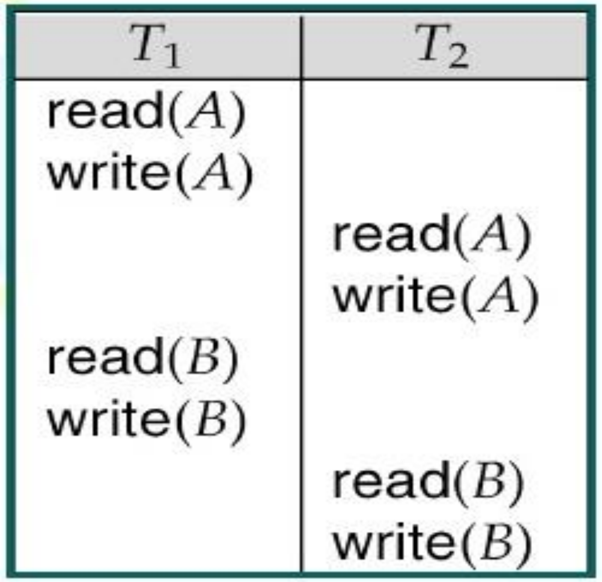
S	
T_1	T_2
RA	
RC	
WD	
	RB
	WB
	RD
WB	
	WC
Commit	
	Commit

d)

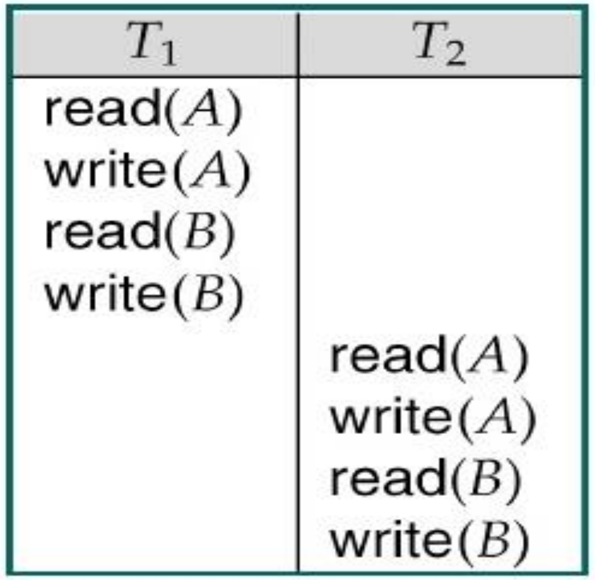
S	
T_1	T_2
	RB
	WB
	RD
	WC
RA	
RC	
WD	
WB	
Commit	
	Commit

CONFLICT SERIALIZABLE

- The schedules which are conflict equivalent to a serial schedule are called conflict serializable schedule.
- If a schedule S can be transformed into a schedule S’ by a series of swaps of non- conflicting instructions, we say that S and S’ are ***conflict equivalent.***
- A schedule S is ***conflict serializable***, if it is conflict equivalent to a serial schedule.



T_1	T_2
read(A)	
write(A)	
	read(A)
	write(A)
read(B)	
write(B)	
	read(B)
	write(B)



T_1	T_2
read(A)	
write(A)	
read(B)	
write(B)	
	read(A)
	write(A)
	read(B)
	write(B)

Procedure for determining conflict serializability of a schedule

- It can be determined using PRECEDENCE GRAPH method:
- A precedence graph consists of a pair G (V, E)
- V= set of vertices consisting of all the transactions participating in the schedule.
- E= set of edges consists of all edges $T_i \rightarrow T_{j,}$ for which one of the following conditions holds:
    - T_i executes write(Q) before T_j executes read(Q)
    - T_i executes read(Q) before T_j executes write(Q)
    - T_i executes write(Q) before T_j executes write(Q)

- If an edge $T_i \rightarrow T_j$ exists in the precedence graph, then in any serial schedule S' equivalent to S, Ti must appear before $T_{j.}$

- **If the precedence graph for S has no cycle, then schedule S is conflict serializable, else it is not.** This cycle detection can be done by cycle detection algorithms, one of them based on depth first search takes **$O(n^2)$** time.

- The serialializibility order of transactions of equivalent serial schedule can be determined using **topological order** in a precedence graph.

Break

Q Let $r_i(z)$ and $w_i(z)$ denote read and write operations respectively on a data item z by a transaction T_i. Consider the following two schedules.

- $S_1: r_1(x) r_1(y) r_2(x) r_2(y) w_2(y) w_1(x)$
- $S_2: r_1(x) r_2(x) r_2(y) w_2(y) r_1(y) w_1(x)$

Which one of the following options is correct? **(GATE - 2021) (2 Marks)**

a) S_1 is conflict serializable, and S_2 is not conflict serializable

b) S_1 is not conflict serializable, and S_2 is conflict serializable

c) Both S_1 and S_2 are conflict serializable

d) Neither S_1 nor S_2 is conflict serializable

Q Let $r_i(z)$ and $w_i(z)$ denote read and write operations respectively on a data item z by a transaction T_i. Consider the following two schedules.

- $S_1: r_1(x)r_1(y)r_2(x)r_2(y)w_2(y)w_1(x)$
- $S_2: r_1(x)r_2(x)r_2(y)w_2(y)r_1(y)w_1(x)$

Which one of the following options is correct? **(GATE - 2021) (2 Marks)**

a) S_1 is conflict serializable, and S_2 is not conflict serializable

b) S_1 is not conflict serializable, and S_2 is conflict serializable

c) Both S_1 and S_2 are conflict serializable

d) Neither S_1 nor S_2 is conflict serializable

S_1	
T_1	T_2
R(X)	
R(Y)	
	R(X)
	R(Y)
	W(Y)
W(X)	

S_2	
T_1	T_2
R(X)	
	R(X)
	R(Y)
	W(Y)
R(Y)	
W(X)	

Q Consider the following four schedules due to three transactions (indicated by the subscript) using read and write on a data item X, denoted by r(X) and w(X) respectively. Which one of them is conflict serializable? **(NET-NOV-2018)** **(GATE - 2014) (2 Marks)**

S_1: $r_1(X); r_2(X); w_1(X); r_3(X); w_2(X)$

S_2: $r_2(X); r_1(X); w_2(X); r_3(X); w_1(X)$

S_3: $r_3(X); r_2(X); r_1(X); w_2(X); w_1(X)$

S_4: $r_2(X); w_2(X); r_3(X); r_1(X); w_1(X)$

(a) S_1 **(b)** S_2 **(c)** S_3 **(d)** S_4

Q Consider the following four schedules due to three transactions (indicated by the subscript) using read and write on a data item X, denoted by r(X) and w(X) respectively. Which one of them is conflict serializable? **(NET-NOV-2018)** **(GATE - 2014) (2 Marks)**

S_1: $r_1(X)$; $r_2(X)$; $w_1(X)$; $r_3(X)$; $w_2(X)$

	S_1	
T_1	T_2	T_3
R(X)		
	R(X)	
W(X)		
		R(X)
	W(X)	

S_2: $r_2(X)$; $r_1(X)$; $w_2(X)$; $r_3(X)$; $w_1(X)$

	S_2	
T_1	T_2	T_3
	R(X)	
R(X)		
	W(X)	
		R(X)
W(X)		

S_3: $r_3(X)$; $r_2(X)$; $r_1(X)$; $w_2(X)$; $w_1(X)$

	S_3	
T_1	T_2	T_3
		R(X)
	R(X)	
R(X)		
	W(X)	
W(X)		

S_4: $r_2(X)$; $w_2(X)$; $r_3(X)$; $r_1(X)$; $w_1(X)$

	S_4	
T_1	T_2	T_3
	R(X)	
	W(X)	
		R(X)
R(X)		
W(X)		

(a) S_1 (b) S_2 (c) S_3 (d) S_4

Q Consider following schedules involving two transactions :

$S_1 : r_1(X); r_1(Y); r_2(X); r_2(Y); w_2(Y); w_1(X)$

$S_2 : r_1(X); r_2(X); r_2(Y); w_2(Y); r_1(Y); w_1(X)$

Which of the following statement is true ? **(NET-JAN-2017)** **(GATE - 2007) (2 Marks)**

(a) Both S_1 and S_2 are conflict serializable.

(b) S_1 is conflict serializable and S_2 is not conflict serializable.

(c) S_1 is not conflict serializable and S_2 is conflict serializable.

(d) Both S_1 and S_2 are not conflict serializable.

Q Consider following schedules involving two transactions :

$S_1 : r_1(X); r_1(Y); r_2(X); r_2(Y); w_2(Y); w_1(X)$

T_1	T_2
r(x)	
r(y)	
	r(x)
	r(y)
	w(y)
w(x)	

$S_2 : r_1(X); r_2(X); r_2(Y); w_2(Y); r_1(Y); w_1(X)$

T_1	T_2
r(x)	
	r(x)
	r(y)
	w(y)
r(y)	
w(x)	

Which of the following statement is true ? **(NET-JAN-2017)** **(GATE - 2007) (2 Marks)**

(a) Both S_1 and S_2 are conflict serializable.

(b) S_1 is conflict serializable and S_2 is not conflict serializable.

(c) S_1 is not conflict serializable and S_2 is conflict serializable.

(d) Both S_1 and S_2 are not conflict serializable.

Q Consider the following transactions with data items P and Q initialized to zero: **(GATE-2012) (2 Marks)**

```
T1: read (P);
    read (Q);
    if P = 0 then Q: = Q + 1;
    write (Q);
```

```
T2: read (Q);
    read (P);
    if Q = 0 then P: = P + 1;
    write (P);
```

Any non-serial interleaving of T_1 and T_2 for concurrent execution leads to

(A) A serializable schedule

(B) A schedule that is not conflict serializable

(C) A conflict serializable schedule

(D) A schedule for which a precedence graph cannot be drawn

Q Consider the following transactions with data items P and Q initialized to zero: **(GATE-2012) (2 Marks)**

T_1: read (P);
read (Q);
if P = 0 then Q: = Q + 1;
write (Q);

T_1
r(P)
r(Q)
w(Q)

T_2
r(Q)
r(P)
w(P)

T_2: read (Q);
read (P);
if Q = 0 then P: = P + 1;
write (P);

Any non-serial interleaving of T1 and T2 for concurrent execution leads to

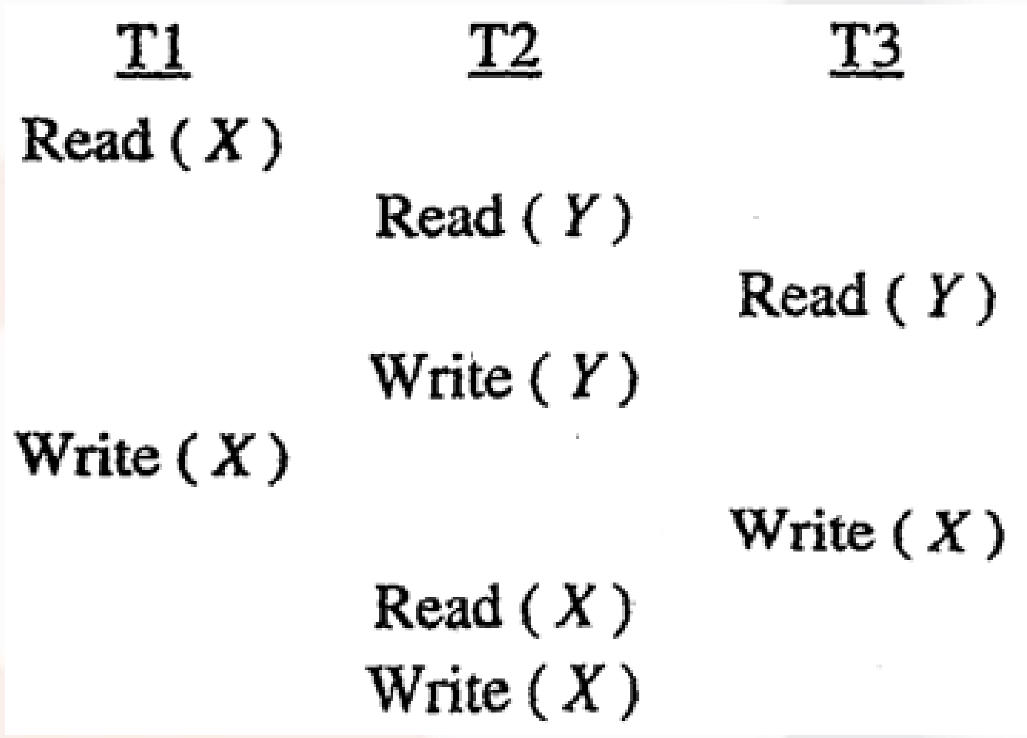
(A) A serializable schedule

(B) A schedule that is not conflict serializable

(C) A conflict serializable schedule

(D) A schedule for which a precedence graph cannot be drawn

Q Consider the following schedule for transactions T_1, T_2 and T_3: **(GATE – 2010) (2 Marks)**
Which one of the schedules below is the correct serialization of the above?



T1	T2	T3
Read (X)		
	Read (Y)	
		Read (Y)
	Write (Y)	
Write (X)		
		Write (X)
	Read (X)	
	Write (X)	

(A) T_1->>T_3->>T_2

(B) T_2->>T_1->>T_3

(C) T_2->>T_3->>T_1

(D) T_3->>T_1->>T_2

Q Consider two transactions T_1 and T_2 which form schedules S_1, S_2, S_3 and S_4 as follows: -
Which of the above schedules is conflicts serializable? **(GATE - 2009) (2 Marks)**

a) Only S_1 **b)** Both S_1 and S_2 **c)** Both S_1 and S_4 **d)** Both S_3 and S_4

T_1: R_1 [A], W_1 [A], W_1 [B]

T_2: R_2 [A], R_2 [B], W_2 [B]

S_1: R_1[A], R_2 [A], R_2[B], W_1[A], W_2[B], W_1[B]

S_2: R_1[A], R_2 [A], R_2[B], W_1[A], W_1[B], W_2 [B]

S_3: R_2[A], R_1 [A], R_2[B], W_1[A], W_1[B], W_2[B]

S_4: R_1[A], R_2 [A], R_2[B], W_1[B], R_2[B], W_2[B]

Q Consider two transactions T_1 and T_2 which form schedules S_1, S_2, S_3 and S_4 as follows: -
Which of the above schedules is conflicts serializable? **(GATE - 2009) (2 Marks)**

a) Only S_1 **b)** Both S_1 and S_2 **c)** Both S_1 and S_4 **d)** Both S_3 and S_4

T_1: R_1 [A], W_1 [A], W_1 [B]

T_2: R_2 [A], R_2 [B], W_2 [B]

S_1: R_1[A], R_2 [A], R_2[B], W_1[A], W_2[B], W_1[B]

S_1	
T_1	T_2
R(A)	
	R(A)
	R(B)
W(A)	
	W(B)
W(B)	

S_2: R_1[A], R_2 [A], R_2[B], W_1[A], W_1[B], W_2 [B]

S_2	
T_1	T_2
R(A)	
	R(A)
	R(B)
W(A)	
W(B)	
	W(B)

S_3: R_2[A], R_1 [A], R_2[B], W_1[A], W_1[B], W_2[B]

S_3	
T_1	T_2
	R(A)
R(A)	
	R(B)
W(A)	
W(B)	
	W(B)

S_4: R_1[A], R_2 [A], R_2[B], W_1[B], R_2[B], W_2[B]

S_4	
T_1	T_2
R(A)	
	R(A)
	R(B)
W(B)	
	R(B)
	W(B)

Q Consider the transactions T_1, T_2, and T_3 and the schedules S_1 and S_2 given below. **(GATE - 2006) (2 Marks)**

T_1: $r_1(X); r_1(Z); w_1(X); w_1(Z)$

T_2: $r_2(Y); r_2(Z); w_2(Z)$

T_3: $r_3(Y); r_3(X); w_3(Y)$

S_1: $r_1(X); r_3(Y); r_3(X); r_2(Y); r_2(Z);w_3(Y); w_2(Z); r_1(Z); w_1(X); w_1(Z)$

S_2: $r_1(X); r_3(Y); r_2(Y); r_3(X); r_1(Z);r_2(Z); w_3(Y); w_1(X); w_2(Z); w_1(Z)$

Which one of the following statements about the schedules is TRUE?

(A) Only S_1 is conflict-serializable.

(B) Only S_2 is conflict-serializable.

(C) Both S_1 and S_2 are conflict-serializable.

(D) Neither S_1 nor S_2 is conflict-serializable.

Q Consider the transactions T_1, T_2, and T_3 and the schedules S_1 and S_2 given below. **(GATE - 2006) (2 Marks)**

T_1: $r_1(X)$; $r_1(Z)$; $w_1(X)$; $w_1(Z)$

T_2: $r_2(Y)$; $r_2(Z)$; $w_2(Z)$

T_3: $r_3(Y)$; $r_3(X)$; $w_3(Y)$

S_1: $r_1(X)$; $r_3(Y)$; $r_3(X)$; $r_2(Y)$; $r_2(Z)$;$w_3(Y)$; $w_2(Z)$; $r_1(Z)$; $w_1(X)$; $w_1(Z)$

S_2: $r_1(X)$; $r_3(Y)$; $r_2(Y)$; $r_3(X)$; $r_1(Z)$;$r_2(Z)$; $w_3(Y)$; $w_1(X)$; $w_2(Z)$; $w_1(Z)$

Which one of the following statements about the schedules is TRUE?

(A) Only S_1 is conflict-serializable.

(B) Only S_2 is conflict-serializable.

(C) Both S_1 and S_2 are conflict-serializable.

(D) Neither S_1 nor S_2 is conflict-serializable.

	S_1	
T_1	T_2	T_3
R(X)		
		R(Y)
		R(X)
	R(Y)	
	R(Z)	
		W(Y)
	W(Z)	
R(Z)		
W(X)		
W(Z)		

	S_2	
T_1	T_2	T_3
R(X)		
		R(Y)
	R(Y)	
		R(X)
R(Z)		
	R(Z)	
		W(Y)
W(X)		
	W(Z)	
W(Z)		

Q Consider three data items D_1, D_2 and D_3 and the following execution schedule of transactions T_1, T_2 and T_3. In the diagram, R(D) and W(D) denote the actions reading and writing the data item D respectively. **(GATE – 2003) (2 Marks)**

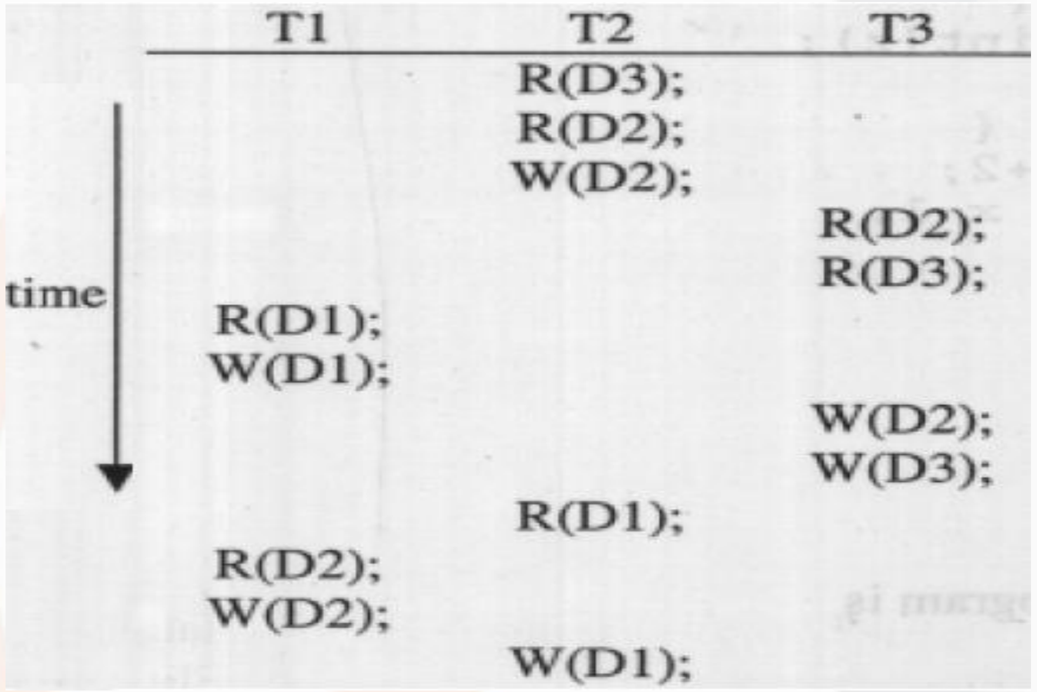
Which of the following statements is correct?

(A) The schedule is serializable as T_2; T_3; T_1

(B) The schedule is serializable as T_2; T_1; T_3

(C) The schedule is serializable as T_3; T_2; T_1

(D) The schedule is not serializable



T1	T2	T3
	R(D3);	
	R(D2);	
	W(D2);	
		R(D2);
		R(D3);
R(D1);		
W(D1);		
		W(D2);
		W(D3);
	R(D1);	
R(D2);		
W(D2);		
	W(D1);	

Q Suppose a database schedule S involves transactions T_1, T_2, ...,T_n. Consider the precedence graph of S with vertices representing the transactions and edges representing the conflicts. If S is serializable, which one of the following orderings of the vertices of the precedence graph is guaranteed to yield a serial schedule? **(NET-NOV-2017)** **(GATE- 2016) (1 Marks)**

(a) Topological order

(b) Depth - first order

(c) Breadth - first order

(d) Ascending order of transaction indices

Let $R_i(z)$ and $W_i(z)$ denote read and write operations on a data element z by a transaction T_i, respectively. Consider the schedule S with four transactions. **(GATE- 2022) (2 Marks)**

$$S: \; R_4(x)R_2(x)R_3(x)R_1(y)W_1(y)W_2(x)W_3(y)R_4(y)$$

Which one of the following serial schedules is conflict equivalent to S?

A. $T_1 \rightarrow T_3 \rightarrow T_4 \rightarrow T_2$
B. $T_1 \rightarrow T_4 \rightarrow T_3 \rightarrow T_2$
C. $T_4 \rightarrow T_1 \rightarrow T_3 \rightarrow T_2$
D. $T_3 \rightarrow T_1 \rightarrow T_4 \rightarrow T_2$