UNIT 4

Lecture 39, 40 & 41
Serializability

Dinesh Kumar Bhawnani
Bhilai Institute of Technology, DURG

Schedule

- Schedule represent the ***chronological order*** in which instructions are executed in the system.
- Clearly, a schedule for a set of transactions must consist of all instructions of those transactions, and must preserve the order in which the instructions appear in each individual transaction.
- For example, in transaction *T*1, the instruction write(*A*) must appear before the instruction read(*B*), in any valid schedule.

T1
read (A)
A := A – 50;
write (A)
read (B)
B := B + 50;
write (B)

Serial Schedule (T1 followed by T2)

- Serial schedule consists of a sequence of instructions from various transactions, where the instructions belonging to one single transaction appear together in that schedule.
- Thus, for a set of *n* transactions, there exist *n*! different valid serial schedules.
- Serial schedules always preserve the consistency of the database.
- Serial schedules are recoverable schedules.
- Serial schedules are cascadeless schedules.

T1	T2
read (A)	
A := A – 50;	
write (A)	
read (B)	
B := B + 50;	
write (B)	
	read (A)
	A := A – 100;
	write (A)
	read (B)
	B := B + 100;
	write (B)

Serial Schedule (T2 followed by T1)

- Serial schedule consists of a sequence of instructions from various transactions, where the instructions belonging to one single transaction appear together in that schedule.
- Thus, for a set of *n* transactions, there exist *n*! different valid serial schedules.
- Serial schedules always preserve the consistency of the database.
- Serial schedules are recoverable schedules.
- Serial schedules are cascadeless schedules.

T1	T2
	read (A)
	A := A – 100;
	write (A)
	read (B)
	B := B + 100;
	write (B)
read (A)	
A := A – 50;	
write (A)	
read (B)	
B := B + 50;	
write (B)	

Concurrent Schedule

- In Concurrent schedule various operations of different transactions are executed in a interleaved manner, i.e. they are executed whenever they get the CPU time slice.
- Thus, for a set of *n* transactions, there exist more than *n*! different valid concurrent schedules.
- Concurrent schedules may or may not preserve the consistency of the database.
- Concurrent schedules may or may not be recoverable schedules.
- Concurrent schedules are not cascadeless schedules.

T1	T2
read (A)	
A := A – 50;	
write (A)	
	read (A)
	A := A – 100;
	write (A)
read (B)	
B := B + 50;	
write (B)	
	read (B)
	B := B + 100;
	write (B)

Concurrent Schedule

- In Concurrent schedule various operations of different transactions are executed in a interleaved manner, i.e. they are executed whenever they get the CPU time slice.
- Thus, for a set of *n* transactions, there exist more than *n*! different valid concurrent schedules.
- Concurrent schedules may or may not preserve the consistency of the database.
- Concurrent schedules may or may not be recoverable schedules.
- Concurrent schedules are not cascadeless schedules.

T1	T2
read (A)	
A := A – 50;	
	read (A)
	A := A – 100;
	write (A)
write (A)	
read (B)	
B := B + 50;	
write (B)	
	read (B)
	B := B + 100;
	write (B)

Serializable Schedule

- Serializable schedules are the concurrent schedules which are equivalent to some serial schedules.
- Thus, for a set of *n* transactions, there exist more than *n*! different valid serializable schedules.
- Serializable schedules always preserve the consistency of the database.
- Serializable schedules may or may not be recoverable schedules.
- Serializable schedules are not cascadeless schedules.

T1	T2
read (A)	
A := A – 50;	
write (A)	
	read (A)
	A := A – 100;
	write (A)
read (B)	
B := B + 50;	
write (B)	
	read (B)
	B := B + 100;
	write (B)

Conflict Serializability

- Let us consider a schedule S in which there are two consecutive instructions I_i and I_j, of transactions T_i and T_j , respectively ($i \neq j$).
- If I_i and I_j refer to different data items, then we can swap I_i and I_j without affecting the results of any instruction in the schedule.

Conflict Serializability

- However, if I_i and I_j refer to the same data item Q, then the order of the two steps may matter. Since we are dealing with only read and write instructions, there are four cases that we need to consider :
  1. I_i = read(Q), I_j = read(Q). The order of I_i and I_j does not matter, since the same value of Q is read by T_i and T_j , regardless of the order.
  2. I_i = read(Q), I_j = write(Q). If I_i comes before I_j, then T_i does not read the value of Q that is written by T_j in instruction I_j. If I_j comes before I_i, then T_i reads the value of Q that is written by T_j. Thus, the order of I_i and I_j matters.
  3. I_i = write(Q), I_j = read(Q). The order of I_i and I_j matters for reasons similar to those of the previous case.
  4. I_i = write(Q), I_j = write(Q). Since both instructions are write operations, the order of these instructions does not affect either T_i or T_j . However, the value obtained by the next read(Q) instruction of S is affected, since the result of only the latter of the two write instructions is preserved in the database. If there is no other write(Q) instruction after I_i and I_j in S, then the order of I_i and I_j directly affects the final value of Q in the database state that results from schedule S.

Conflict Serializability

- Thus, only in the case where both I_i and I_j are read instructions does the relative order of their execution not matter.
- We say that I_i and I_j **conflict** if they are operations by different transactions on the same data item, and at least one of these instructions is a write operation.
- If a schedule S1 can be transformed into a schedule S2 by a series of swaps of non-conflicting instructions, we say that S1 and S2 are **conflict equivalent**.
  - For e.g. Schedule S1 and S2 are conflict equivalent schedules.
- We say that a schedule *S* is **conflict serializable schedule** if it is conflict equivalent to a serial schedule.
  - For e.g. Schedule S1 is conflict serializable schedule because it is conflict equivalent to a serial schedule (T1 followed by T2).

Conflict Equivalent Schedule (Conflict Serializable)

S1	
T1	**T2**
read (A)	
A := A – 50;	
write (A)	
	read (A)
	A := A – 100;
	write (A)
read (B)	
B := B + 50;	
write (B)	
	read (B)
	B := B + 100;
	write (B)

S2	
T1	**T2**
read (A)	
A := A – 50;	
write (A)	
read (B)	
B := B + 50;	
write (B)	
	read (A)
	A := A – 100;
	write (A)
	read (B)
	B := B + 100;
	write (B)

S3	
T1	**T2**
	read (A)
	A := A – 100;
	write (A)
	read (B)
	B := B + 100;
	write (B)
read (A)	
A := A – 50;	
write (A)	
read (B)	
B := B + 50;	
write (B)	

View Serializability

- Consider two schedules S1 and S2, where the same set of transactions participates in both schedules. The schedules S1 and S2 are said to be **view equivalent** if three conditions are met:
  1. For each data item Q, if transaction T_i reads the initial value of Q in schedule S1, then transaction T_i must, in schedule S2, also read the initial value of Q.
  2. For each data item Q, if transaction T_i executes read(Q) in schedule S1, and if that value was produced by a write(Q) operation executed by transaction T_j , then the read(Q) operation of transaction T_i must, in schedule S2, also read the value of Q that was produced by the same write(Q) operation of transaction T_j .
  3. For each data item Q, the transaction (if any) that performs the final write(Q) operation in schedule S1 must perform the final write(Q) operation in schedule S2.

View Serializability

- For e.g. Schedule S1 and S2 are view equivalent schedules.
- We say that a schedule *S* is **view serializable schedule** if it is view equivalent to a serial schedule.
- For e.g. Schedule S4 is view serializable schedule because it is view equivalent to a serial schedule (T1 followed by T2).

View Equivalent Schedule (View Serializable)

S4	
T1	**T2**
read (A)	
A := A – 50;	
write (A)	
	read (A)
	A := A – 100;
	write (A)
read (B)	
B := B + 50;	
write (B)	
	read (B)
	B := B + 100;
	write (B)

S5	
T1	**T2**
read (A)	
A := A – 50;	
write (A)	
read (B)	
B := B + 50;	
write (B)	
	read (A)
	A := A – 100;
	write (A)
	read (B)
	B := B + 100;
	write (B)

S6	
T1	**T2**
	read (A)
	A := A – 100;
	write (A)
	read (B)
	B := B + 100;
	write (B)
read (A)	
A := A – 50;	
write (A)	
read (B)	
B := B + 50;	
write (B)	

Relation between Conflict and View Serializable schedule

- Every conflict serializable schedule is also view serializable, but there are view serializable schedules that are not conflict serializable.
- For. E.g. Schedule S7 is view serializable schedule because it is view equivalent to a serial schedule T1, T2, T3.
- Schedule S7 is not conflict serializable schedule because it is not conflict equivalent to any serial schedules.
- Every view serializable schedule which is not conflict serializable always contains **blind writes**.

S7		
T1	**T2**	**T3**
read (A)		
	write (A)	
write (A)		
		write (A)

S8		
T1	**T2**	**T3**
read (A)		
write (A)		
	write (A)	
		write (A)

Recoverability

- So far, we have studied what schedules are acceptable from the viewpoint of consistency of the database, assuming implicitly that there are no transaction failures. We now address the effect of transaction failures during concurrent execution.
- If a transaction T_i fails, for whatever reason, we need to undo the effect of this transaction to ensure the atomicity property of the transaction.
- In a system that allows concurrent execution, it is necessary also to ensure that any transaction T_j that is dependent on T_i (that is, T_j has read data written by T_i) is also aborted.
- To achieve this surety, we need to place restrictions on the type of schedules permitted in the system.

Recoverable Schedule

- A **recoverable schedule** is one where, for each pair of transactions T_i and T_j such that T_j reads a data item previously written by T_i, the commit operation of T_i appears before the commit operation of T_j.
- Schedule S9 is a non recoverable schedule.
- Schedule S10 is a recoverable schedule.

S9	
T1	**T2**
read (A)	
write (A)	
	read (A)
	Commit;
read (B)	
Commit;	

S10	
T1	**T2**
read (A)	
write (A)	
	read (A)
read (B)	
Commit;	
	Commit;

Cascadeless Schedule

- A **cascade-less schedule** is one where, for each pair of transactions T_i and T_j such that T_j reads a data item previously written by T_i, the commit operation of T_i appears before the read operation of T_j.
- Schedule S11 is not a cascade-less schedule because cascading rollbacks occur after the failure (between write(A) of T3 and commit; of T1).

S11		
T1	**T2**	**T3**
read (A)		
write (A)		
	read (A)	
	write (A)	
		read(A)
		write(A)
Commit;		
	Commit;	
		Commit;

Cascadeless Schedule

- A **cascade-less schedule** is one where, for each pair of transactions T_i and T_j such that T_j reads a data item previously written by T_i, the commit operation of T_i appears before the read operation of T_j.
- Schedule S12 is a cascade-less schedule because cascading rollbacks cannot occur after the failure.

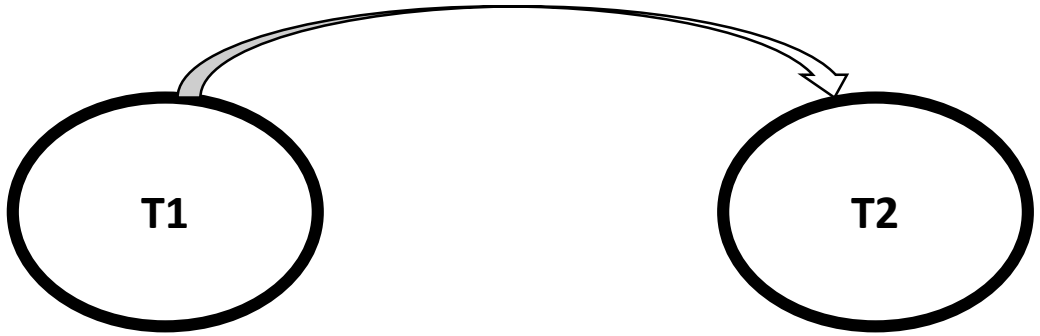
S12		
T1	**T2**	**T3**
read (A)		
write (A)		
Commit;		
	read (A)	
	write (A)	
	Commit;	
		read(A)
		write(A)
		Commit;

Test for conflict serializablity of a schedule

1. For each transaction T_i, participating in the schedule S, create a node labeled T_i in the precedence graph.
2. For each case where T_j executes a read(X) after T_i executes write(X), create an edge from T_i to T_j in the precedence graph.
3. For each case where T_j executes write(X) after T_i executes a read(X), create an edge from T_i to T_j in the graph.
4. For each case where T_j executes a write(X) after T_i executes a write(X), create an edge from T_i to T_j in the graph.
5. The schedule S is serializable if and only if there are no cycles in the graph. (i.e. it is acyclic).

Test for conflict serializablity of a schedule

S13	
T1	**T2**
read (A)	
A := A – 50;	
write (A)	
	read (A)
	A := A – 100;
	write (A)
read (B)	
B := B + 50;	
write (B)	
	read (B)
	B := B + 100;
	write (B)

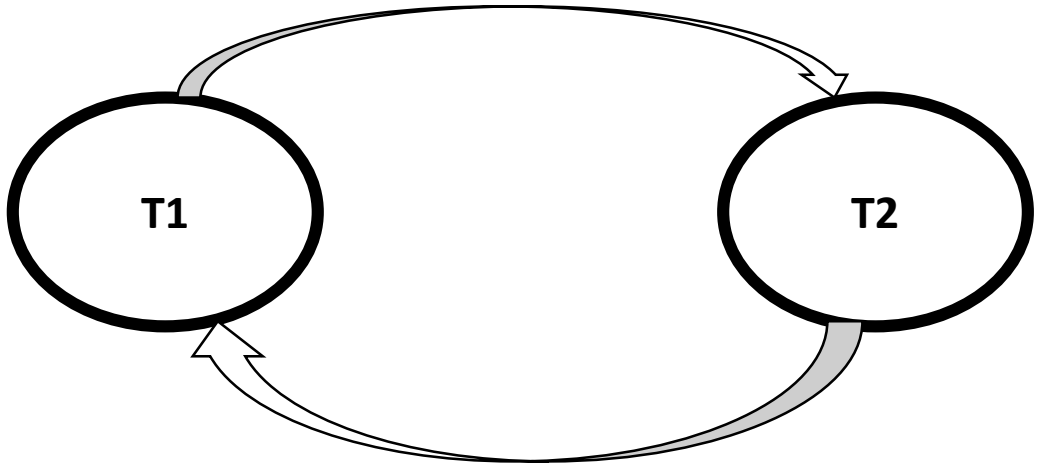


Precedence Graph of Schedule S13

- The Schedule S13 is a conflict serializable schedule because the precedence graph is acyclic.
- A **serializability order** of the transactions can be obtained through **topological sorting**, which determines a linear order consistent with the partial order of the precedence graph. There are, in general, several possible linear orders that can be obtained through a topological sorting.
- It is conflict equivalent to a serial schedule T1 → T2.

Test for conflict serializablity of a schedule

S14	
T1	**T2**
read (A)	
A := A – 50;	
	read (A)
	A := A – 100;
	write (A)
write (A)	
read (B)	
B := B + 50;	
write (B)	
	read (B)
	B := B + 100;
	write (B)

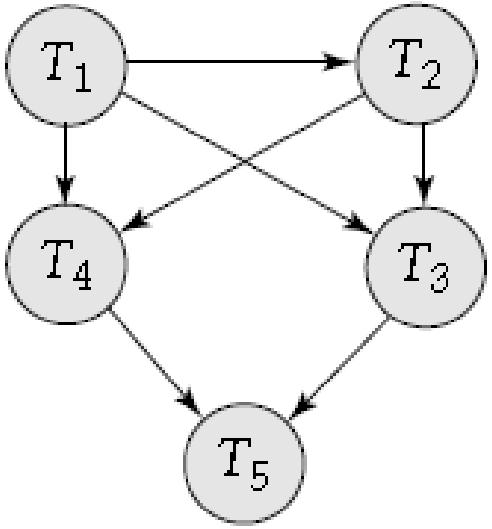


Precedence Graph of Schedule S14

The Schedule S14 is not a conflict serializable schedule because the precedence graph contain a cycle (T1→T2→T1).

Test for conflict serializablity of a schedule

Q. Consider the precedence graph below. Is the corresponding schedule conflict serializable? Explain your answer.



Sol : The following schedule is conflict serializable schedule, because given precedence graph is acyclic. It's equivalent serial schedules are

(i) T1 → T2 → T3 → T4 →T5, and

(ii) T1 → T2 → T4 → T3 → T5.

Test for conflict serializablity of a schedule

Q. Consider the following schedules :

S1 = R1(X); R2(Z); R1(Z); R3(X); R3(Y); W3(Y); R2(Y); W2(Z); W3(Y).

S2 = R1(X); R2(X); R3(X); R1(Z); R2(Y); W1(X); W2(Z); W3(Y).

(i) Create transaction table for each.

(ii) Draw precedence graph for S1 & S2.

(iii) Check whether S1 and S2 are conflict serializable or not.

Transaction Table

Given schedules are

S1 = R1(X); R2(Z); R1(Z); R3(X); R3(Y); W3(Y); R2(Y); W2(Z); W3(Y).

S2 = R1(X); R2(X); R3(X); R1(Z); R2(Y); W1(X); W2(Z); W3(Y).

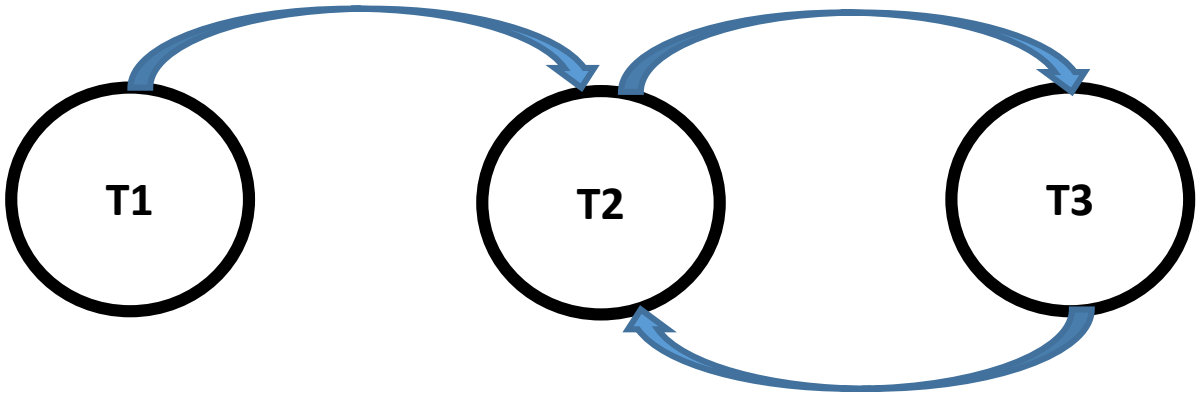
The transaction table for S1 and S2 are

S1		
T1	**T2**	**T3**
R(X)		
	R(Z)	
R(Z)		
		R(X)
		R(Y)
		W(Y)
	R(Y)	
	W(Z)	
		W(Y)

S2		
T1	**T2**	**T3**
R(X)		
	R(X)	
		R(X)
R(Z)		
	R(Y)	
W(X)		
	W(Z)	
		W(Y)

Precedence Graph

S1		
T1	T2	T3
R(X)		
	R(Z)	
R(Z)		
		R(X)
		R(Y)
		W(Y)
	R(Y)	
	W(Z)	
		W(Y)

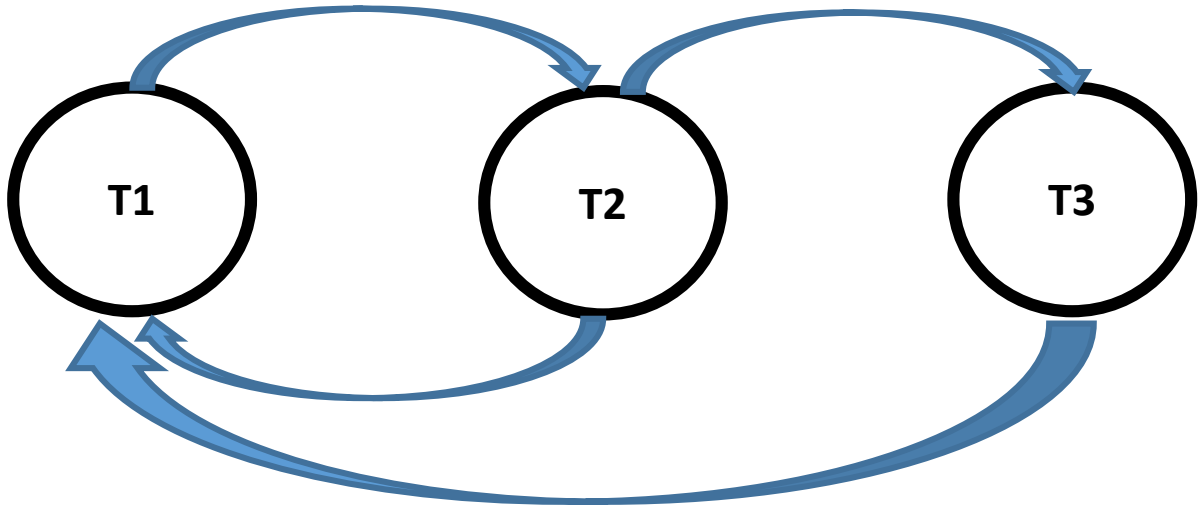


Precedence Graph (S1)

The Schedule S1 is not conflict serializable schedule because it contains a cycle
T2 → T3 → T2.

Precedence Graph

S2		
T1	T2	T3
R(X)		
	R(X)	
		R(X)
R(Z)		
	R(Y)	
W(X)		
	W(Z)	
		W(Y)



Precedence Graph (S2)

The Schedule S2 is not conflict serializable schedule because it contains cycles
T1 → T2 → T1, and
T1 → T2 → T3 → T1.

GATE Questions

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?

(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

[GATE 2021]

GATE Questions

Let S be the following schedule of operations of three transactions T1, T2 and T3 in a relational database system:

R2(Y),R1(X),R3(Z),R1(Y)W1(X),R2(Z),W2(Y),R3(X),W3(Z)

Consider the statements P and Q below:

P : S is conflict-serializable.

Q : If T3 commits before T1 finishes, then S is recoverable.

Which one of the following choices is correct?

(A) Both P and Q are true

(B) P is true and Q is false

(C) P is false and Q is true

(D) Both P and Q are false

[GATE 2021]

GATE Questions

Consider a schedule of transactions T1 and T2:

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

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?

A.

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

B.

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

C.

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

D.

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

[GATE 2020]

GATE Questions

Two transactions T_1 and T_2 are given as:

T_1 : $r_1(X)$, $w_1(X)$, $r_1(Y)$, $w_1(Y)$

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

where $r_i(V)$ denotes a read operation by transaction T_i on a variable V and $w_i(V)$ denotes a *write* operations by transaction T_i on a variable V. The total number of conflict serializable schedules that can be formed by T1 and T2 is _____.

[GATE 2017]

GATE Questions

Which one of the following is NOT a part of the ACID properties of database transactions?

(A) Atomicity

(B) Consistency

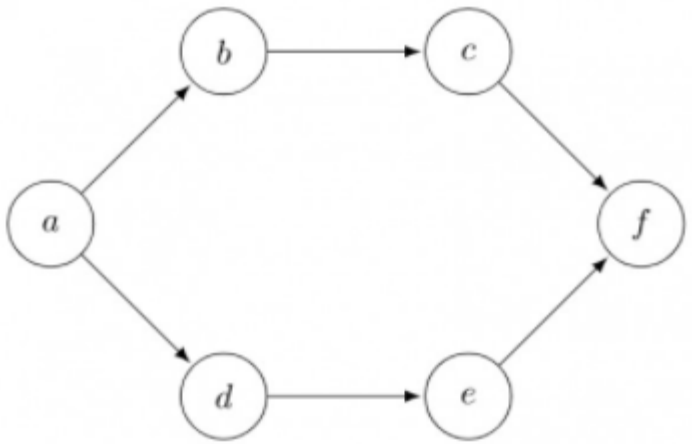
(C) Isolation

(D) Deadlock-freedom

[GATE 2016]

GATE Questions

Consider the following directed graph :



The number of different topological orderings of the vertices of the graph is _____.

[GATE 2016]

GATE Questions

Suppose a database schedule S involves transactions T1, . . . , Tn. Construct 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?

(A) Topological order

(B) Depth-first order

(C) Breadth-first order

(D) Ascending order of transaction indices

[GATE 2016]

GATE Questions

Consider the following database schedule with two transactions, T_1 and T_2.

$$S = r_2(X); r_1(X); r_2(Y); w_1(X); r_1(Y); w_2(X); a_1; a_2$$

Where $r_i(Z)$ denotes a read operation by transaction T_i on a variable Z, $w_i(Z)$ denotes a write operation by T_i on a variable Z and a_i denotes an abort by transaction T_i.

Which one of the following statements about the above schedule is TRUE?

(A) S is non-recoverable

(B) S is recoverable, but has a cascading abort

(C) S does not have a cascading abort

(D) S is strict

[GATE 2016]

GATE Questions

Consider the following transaction involving two bank account x and y.

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

[GATE 2015]

GATE Questions

Consider the following partial Schedule S involving two transactions T1 and T2. Only the read and the write operations have been shown. The read operation on data item P is denoted by read(P) and the write operation on data item P is denoted by write(P).

Time	Transaction-id	
	$T1$	$T2$
1	$read(A)$	
2	$write(A)$	
3		$read(C)$
4		$write(C)$
5		$read(B)$
6		$write(B)$
7		$read(A)$
8		$commit$
9	$read(B)$	

Suppose that the transaction T1 fails immediately after time instance 9. Which one of the following statements is correct?

(A) T2 must be aborted and then both T1 and T2 must be re-started to ensure transaction atomicity

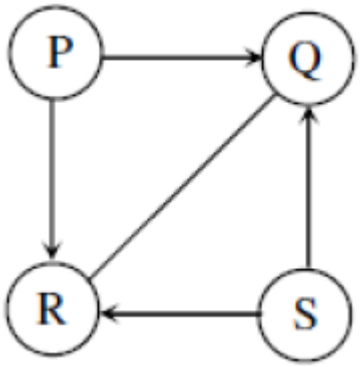
(B) Schedule S is non-recoverable and cannot ensure transaction atomicity

(C) Only T2 must be aborted and then re-started to ensure transaction atomicity

(D) Schedule S is recoverable and can ensure atomicity and nothing else needs to be done **[GATE 2015]**

GATE Questions

Consider the directed graph given below.



Which one of the following is **TRUE?**

(A) The graph does not have any topological ordering

(B) Both PQRS and SRQP are topological orderings

(C) Both PSRQ and SPRQ are topological orderings.

(D) PSRQ is the only topological ordering.

[GATE 2014]

GATE Questions

Consider the following four schedules due to three transactions (indicted by the subscript) using read and write on a data item x, denoted r (x) and w (x) respectively. Which one of them is conflict serializable?

(A) $r_1(x)$; $r_2(x)$; $w_1(x)$; $r_3(x)$; $w_2(x)$

(B) $r_2(x)$; $r_1(x)$; $w_2(x)$; $r_3(x)$; $w_1(x)$

(C) $r_3(x)$; $r_2(x)$; $r_1(x)$; $w_2(x)$; $w_1(x)$

(D) $r_2(x)$; $w_2(x)$; $r_3(x)$; $r_1(x)$; $w_1(x)$

[GATE 2014]

GATE Questions

Consider the following schedule S of transactions T1, T2, T3, T4:

T1	T2	T3	T4
	Reads(X)		
		Writes(X)	
		Commit	
Writes(X)			
Commit			
	Writes(Y)		
	Reads(Z)		
	Commit		
			Reads(X)
			Reads(Y)
			Commit

Which one of the following statements is CORRECT?

(A) S is conflict-serializable but not recoverable

(B) S is conflict-serializable but is recoverable

(C) S is both conflict-serializable and recoverable

(D) S is neither conflict-serializable nor is it recoverable **[GATE 2014]**

GATE Questions

Consider the transactions T1, T2, and T3 and the schedules S1 and S2 given below.

T1 : r1 (X) ; r1 (z) ; w1 (X) ; w1 (z)

T2 : r2 (X) ; r2 (z) ; w2 (z)

T3 : r3 (X) ; r3 (X) ; w3 (Y)

S1: r1(X); r3(Y); r3(X); r2(Y); r2(Z); w3(Y); w2(Z); r1(Z); w1(X); w1(Z)

S2: r1(X); r3(Y); r2(Y); r3(X); r1(Z); r2(Z); w3(Y); w1(X); w2(Z); w1(Z)

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

(A) Only S1 is conflict-serializable.

(B) Only S2 is conflict-serializable.

(C) Both S1 and S2 are conflict-serializable.

(D) Neither S1 nor S2 is conflict-serializable.

[GATE 2014]

GATE Questions

Consider the following transactions with data items P and Q initialized to zero :

```
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 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 precedence graph cannot be drawn **[GATE 2012]**

GATE Questions

Consider the following schedule for transactions T1, T2 and T3 :

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

Which one of the schedules below is the correct serialization of the above?

(A) T1→T3→T2

(B) T2→T1→T3

(C) T2→T3→T1

(D) T3→T1→T2

[GATE 2010]

GATE Questions

Consider two transactions T1 and T2, and four schedules S1, S2, S3, S4 of T1 and T2 as given below :

T1: R1[X] W1[X] W1[Y]

T2: R2[X] R2[Y] W2[Y]

S1: R1[X] R2[X] R2[Y] W1[Y] W1[Y] W2[Y] S2: R1[X] R2[X] R2[Y] W1[Y] W2[Y] W1[Y]

S3: R1[X] W1[X] R2[X] W1[Y] R2[Y] W2[Y] S4: R2[X] R2[Y] R1[X] W1[X] W1[Y] W2[Y]

Which of the above schedules are conflict serializable?

(A) S1 and S2

(B) S2 and S3

(C) S3 only

(D) S4 only

[GATE 2009]

GATE Questions

Consider the following schedules involving two transactions. Which one of the following statements is TRUE?

S1: r1(X); r1(Y); r2(X); r2(Y); w2(Y); w1(X)

S2: r1(X); r2(X); r2(Y); w2(Y); r1(Y); w1(X)

(A) Both S1 and S2 are conflict serializable.

(B) S1 is conflict serializable and S2 is not conflict serializable.

(C) S1 is not conflict serializable and S2 is conflict serializable.

(D) Both S1 and S2 are not conflict serializable.

[GATE 2007]

GATE Questions

Which of the following scenarios may lead to an irrecoverable error in a database system?

(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

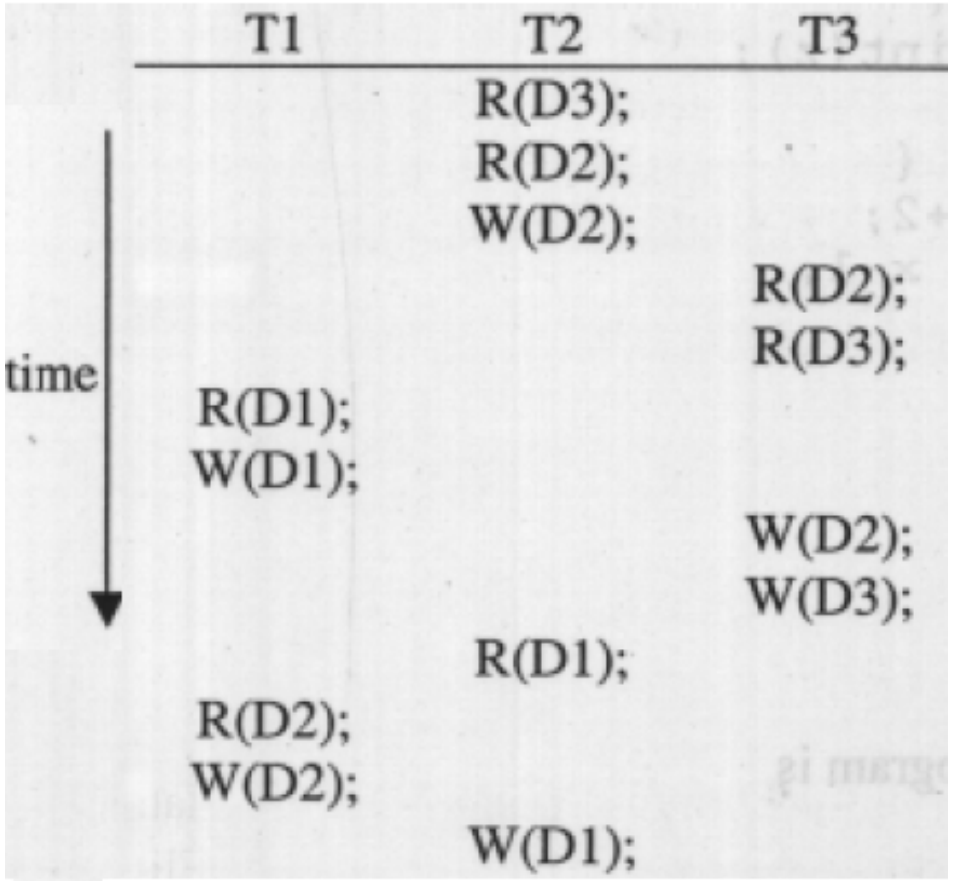
(C) A transaction reads a data item after it is written by a committed transaction

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

[GATE 2003]

GATE Questions

Consider three data items D1, D2 and D3 and the following execution schedule of transactions T1, T2 and T3. In the diagram, R(D) and W(D) denote the actions reading and writing the data item D respectively.



Which of the following statements is correct?

(A) The schedule is serializable as T2; T3; T1

(B) The schedule is serializable as T2; T1; T3

(C) The schedule is serializable as T3; T2; T1

(D) The schedule is not serializable

[GATE 2003]

For Video lecture on this topic please subscribe to my youtube channel.

The link for my youtube channel is

https://www.youtube.com/channel/UCRWGtE76JlTp1iim6aOTRuw?sub_confirmation=1