unacademy

Transaction Concept

Comprehensive Course on DBMS For CS & DA

Vijay Kumar Agarwal • Lesson 36 • Sept 5, 2024

unacademy

Computer Science & IT Engineering

Transaction Concept

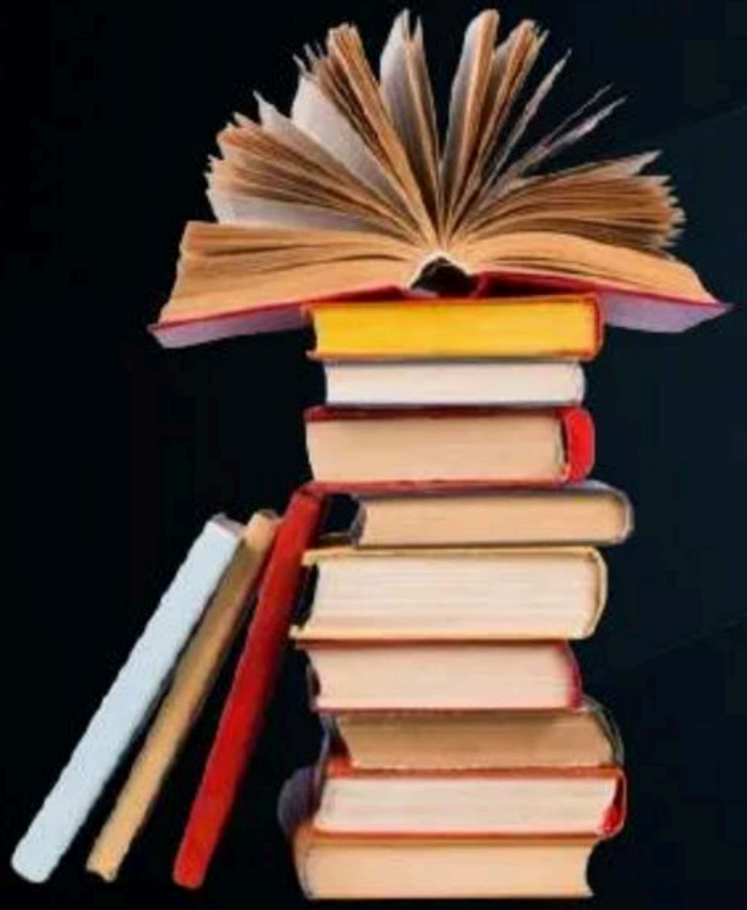
Lecture : 36 Database Management System

By- Vijay Agarwal Sir

unacademy

Topics

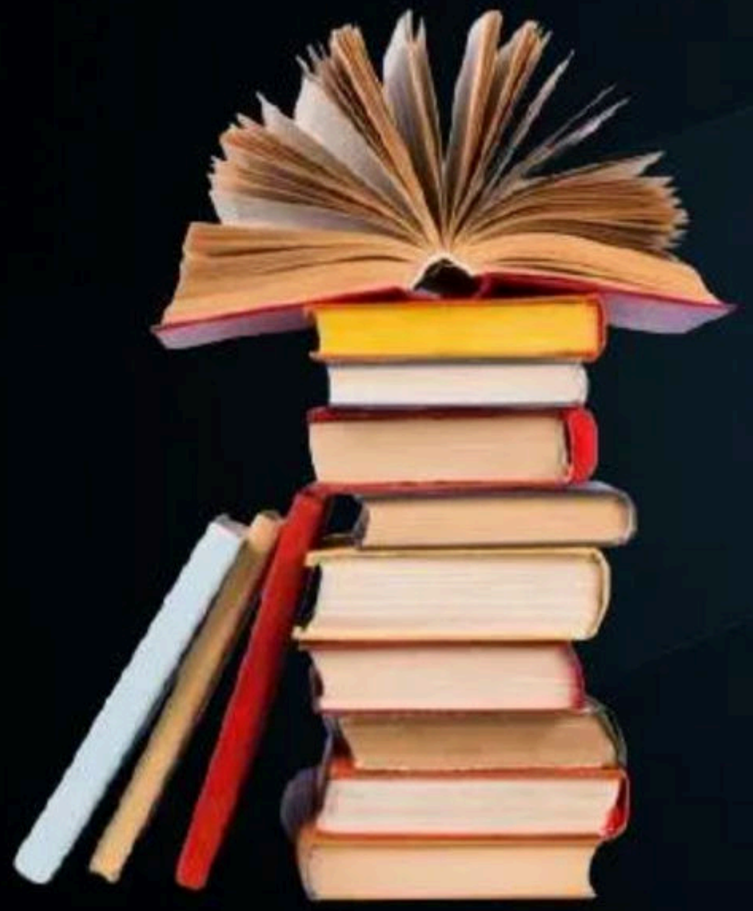
to be covered



1. Transaction Concept
2. ACID Properties
3. Schedule
4. Types of Schedule

unacademy

Recap of previous Lecture



1. B+ Tree & GATE PYQ's
2. NULL Value Concept & Questions

unacademy

▲ 1 • Asked by Sunitesh

"You are one of those teacher who can make an impact on any kind of student"

I am greatful to be your student.

Transaction Concept

- A transaction is a unit of program execution that accesses and possibly updates various data items.
- E.g. Transaction to transfer Rs 500 from account A to account B:
  1. read (A)
  2. A : = A – 500
  3. write(A)
  4. read(B)
  5. B: = B + 500
  6. write(B)

ACID Properties

- A transaction is a unit of program execution that accesses and possibly updates various data items. To preserve the integrity of data the database system must ensure:
  - A. Atomicity
  - C. Consistency
  - I. Isolation
  - D. Durability

ACID Properties

- **Atomicity:** Either all operations of the transaction are properly reflected in the database or none are.
- **Consistency:** Execution of a transaction in isolation preserves the consistency of the database.
- **Isolation:** Although multiple transactions may execute concurrently, each transaction must be unaware of other concurrently executing transactions. Intermediate transaction results must be hidden from other concurrently executed transactions.

- That is, for every pair of transactions T_i and T_j, it appears to T_i that either T_j, finished exection before T_i started, or T_j started execution after T_i finished.

- Durability: After a transaction completes successfully, the changes it has made to the database persist, even if there are system failures.

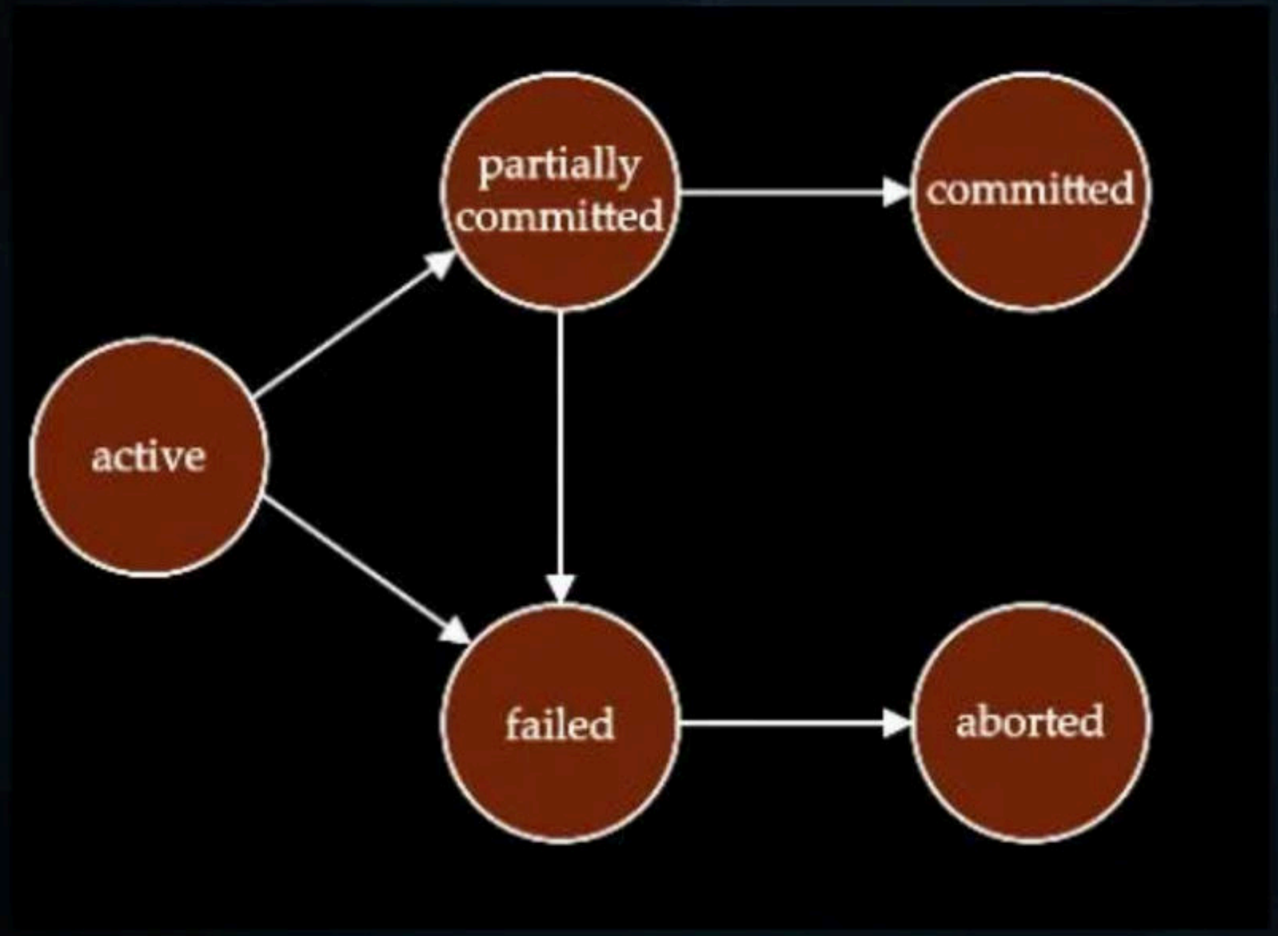
Transaction State

- Active : the initial state; the transaction stays in this state while it is executing.
- Partially committed : after the final statement has been executed.
- Failed: after the discovery that normal execution can no longer proceed.
- Aborted: after the transaction has been rolled back and the database restored to its state prior to the start of the transaction. Two options after it has been aborted:

- Restart the transaction
  - Can be done only if no internal logical error
- Kill the transaction

- Committed: After successful completion.

Transaction State (Cont.)



Schedules

- Schedule: a sequences of instructions that specify the chronological order in which instructions of concurrent transactions are executed.
  - A schedule for a set of transactions must consist of all instructions of those transactions
  - Must preserve the order in which the instructions appear in each individual transaction.
- A transaction that successfully completes its execution will have a commit instructions as the last statement
  - By default transaction assumed to execute commit instruction as its last step

- A transaction that fails to successfully complete its execution will have an abort instruction as the last statement.

Let T_1 transfer 100 Rs from A to B, and T_2 transfer 10% of the balance from A to B.

Schedule 1

T_1	T_2
read (A)	
A: = A – 100	
write (A)	
read (B)	
B: = B + 100	
write (B)	
commit	
	read (A)
	temp := A * 0.1
	A := A – temp
	write (A)
	read (B)
	B := B + temp
	write (B)
	Commit

$S_1 < T_1\ T_2>$

Serial schedule in which T_1 is followed by T_2:

Schedule 2

T_1	T_2
	read (A)
	temp := A * 0.1
	A := A – temp
	write (A)
	read (B)
	B := B + temp
	write (B)
	Commit
read (A)	
A: = A – 100	
write (A)	
read (B)	
B: = B + 100	
write (B)	
commit	

$S_2 < T_2\ T_1>$

serial schedule where T_2 is followed by T_1

unacademy

Schedule 3

T_1	T_2
read (A)	
A: = A – 100	
write (A)	
	read (A)
	temp := A * 0.1
	A := A – temp
	write (A)
read (B)	
B: = B + 100	
write (B)	
commit	
	read (B)
	B := B + temp
	write (B)
	Commit

C_1

Schedule 4

T_1	T_2
read (A)	
A: = A – 100	
	read (A)
	temp := A * 0.1
	A := A – temp
	write (A)
write (A)	
read (B)	
B: = B + 100	
write (B)	
commit	
	read (B)
	B := B + temp
	write (B)
	Commit

C_2

Serial Schedule

- After Commit of one transaction, begins (Start) another transaction.
- Number of possible serial Schedules with 'n' transactions is "n!"
- The execution sequence of Serial Schedule always generates consistent result.

Example

S : $R_1(A)$ $W_1(A)$ Commit (T_1) $R_2(A)W_2(A)$ commit (T_2).

Advantage

- Serial Schedule always produce correct result (integrity guaranteed) as no resource sharing.

Disadvantage

- Less degree of concurrency.
- Through put of system is low.
- It allows transactions to execute one after another.

Serializable Schedule

A Schedule is serializable Schedule if it is equivalent to a Serial Schedule.

(i)Conflict Serializability

(ii)View Serializability

Serializability

- Basic Assumption: Each transaction preserves database consistency.
- Thus, serial execution of a set of transactions preserves database consistency.
- A (possibly concurrent) schedule is serializable if it is equivalent to a serial schedule. Different forms of schedule equivalence give rise to the notions of:
  1. Conflict serializability
  2. view serializability

Conflict Serializability

- 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.
- We say that a schedule S is conflict serializable if it is conflict equivalent to a serial schedule.

Conflict Serializability (Cont.)

- Schedule 3 can be transformed into Schedule 6, a serial schedule where T_2 follows T_1, by series of swaps of non-conflicting instructions. Therefore Schedule 3 is conflict serializable.

Schedule 3

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

Schedule 6

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

Conflict Serializability (Cont.)

- Example of a schedule that is not conflict serializable:

T_3	T_4
read (Q)	
	write (Q)
write (Q)	

- We are unable to swap instructions in the above schedule to obtain either the serial schedule < T3, T4 >, or the serial schedule < T4, T3 >

Conflict Serializability (Cont.)

- Example of a schedule that is not conflict serializable:

T_3	T_4
read (Q)	
	write (Q)
write (Q)	

- We are unable to swap instructions in the above schedule to obtain either the serial schedule < T3, T4 >, or the serial schedule < T4, T3 >

Conflict Serializable

A schedule is said to be conflict serializable if it is conflict equivalent to a serial schedule.

Same conflicting operation order in C_1 & S_1

$\therefore$ Its $\{C_1\}$ conflict is conflict serializable.

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

C_L

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

S_L

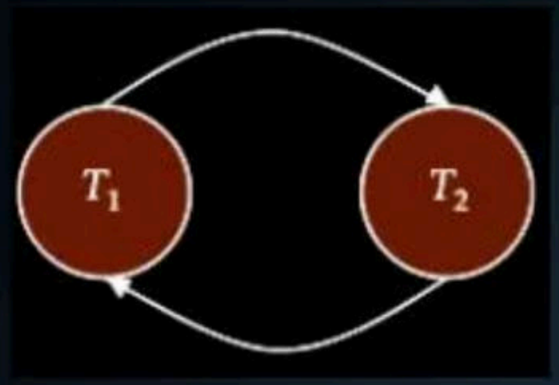
Conflicting Instructions

- Instructions l_i, and l_j of transactions T_i and T_j respectively, conflict if and only if there exists some item Q accessed by both l_i, and l_j, and at least one of these instructions wrote Q.
  1. l_i, = read(Q), l_j = read(Q). l_i and l_j don't conflict.
  2. l_i, = read(Q) l_j = write(Q). They conflict.
  3. l_i, = write(Q) l_j = read(Q). They conflict
  4. l_i= write(Q) l_j = write(Q). They conflict
- Intuitively, a conflict between l_i and l_j forces a (logical) temporal order between them.
  - If l_i, and l_j are consecutive in a schedule and they do not conflict, their results would remain the same even if they had been interchanged in the schedule.

Testing for Serializability

- Testing for conflict serializability.
  - Consider some schedule of a set of transactions T_1, T_2, ...T_n
  - Precedence graph — a direct graph where the vertices are the transactions (names).
  - We draw an arc from T_i to T_j if the two transaction conflict, and T_i accessed the data item on which the conflict arose earlier.
  - We may label the arc by the item that was accessed.

Example:



A schedule is conflict serializable if and only if its precedence graph is acyclic.

NOTE: CNC [Cycle not conflict serializable]

Q.

S: $R_1(A)$ $W_1(A)$ $R_2(A)$ $W_2(A)$ $R_1(B)$ $W_1(B)$ $R_2(B)$ $W_2(B)$

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

Q. $R_1(A)\ R_2(A)\ W_2(A)\ W_1(A)\ R_1(B)\ W_1(B)\ R_2(B)\ W_2(B)$

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

Q. $R_1(A)\ W_1(A)\ R_2(B)\ W_2(B)\ R_1(B)\ W_1(B)\ R_2(A)\ W_2(A)$

Serializability Order

Important Point 1:

1. If S_1, S_2 Schedule are conflict equal then precedence graph of S_1 and S_2 must be same.
2. If S_1 and S_2 have same precedence graph then S_1 and S_2 may or may not conflict equal.

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

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

S_2: $r_1(X); r_2(X); r_2(Y); W_2(Y); r_1(Y); w_1(X)$

[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 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?

[2014(Set-1): 2 Marks]

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

Q. Let $r_i(z)$ and $w_i(z)$ denote read and write operations respectively on a data item 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? [MCQ: 2021: 2M]

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

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

S_2: $r_1(X); r_2(X); r_2(Y); W_2(Y); r_1(Y); w_1(X)$

[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.2

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?

[2014(Set-1): 2 Marks]

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

MCQ Q.3

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(Y) ; r2(Z) ; w2(Z)
T3: r3(Y); 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?

[GATE-2014-CS: 2M]

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.

Q.4 Let $r_i(z)$ and $w_i(z)$ denote read and write operations respectively on a data item 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? [MCQ: 2021: 2M]

(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.5 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.

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? [2022: 2 Marks]

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 $T_3 \rightarrow T_1 \rightarrow T_4 \rightarrow T_2$

Q.6 Consider the following transaction involving two bank accounts 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 [2015(Set-2): 1 Marks]

A. Atomicity

B. Consistency

C. Isolation

D. Durability

MCQ Q.7

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

[GATE-2016-CS: 1M]

A Atomicity

B Consistency

C Isolation

D Deadlock-freedom

MCQ Q.8

Suppose a database schedule S involves transaction T_1 ....., T_n. 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?

[GATE-2016-CS: 2M]

A Topological order

B Depth-first order

C Breadth-first order

D Ascending order of transaction indices

MCQ Q.9

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

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

[GATE-2010-CS: 2M]

T1	T3	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) T2 → T3 → T1

D) T3 → T 1 → T 2

MCQ Q.10

Consider two transactions T_1 and T_2, and four schedules S_1, S_2, S_3, S_4 of T_1 and T_2 as given below:

T_1: $R_1[x]$ $W_1[x]$ $W_1[y]$;

T_2: $R_2[x]$ $R_2[y]$ $W_2[y]$;

S_1: $R_1[x]$ $R_2[x]$ $R_2[y]$ $W_1[x]$ $W_1[y]$ $W_2[y]$;

S_2: $R_1[x]$ $R_2[x]$ $R_2[y]$ $W_1[x]$ $W_2[y]$ $W_1[y]$;

S_3: $R_1[x]$ $W_1[x]$ $R_2[x]$ $W_1[y]$ $R_2[y]$ $W_2[y]$;

S_4: $R_2[x]$ $R_2[y]$ $R_1[x]$ $W_1[x]$ $W_1[y]$ $W_2[y]$;

Which of the above schedules are conflict serializable?

[GATE-2009-CS: 2M]

A S_1 and S_2

B S_2 and S_3

C S_3 only

D S_4 only

unacademy

THANK YOU!

Here’s to a cracking journey ahead!