RAMAIAH
Institute of Technology

M.S. Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of Computer Science and Engineering

Course Name: Database Systems

Course Code: CS52

Credits: 3:1:0

UNIT 5

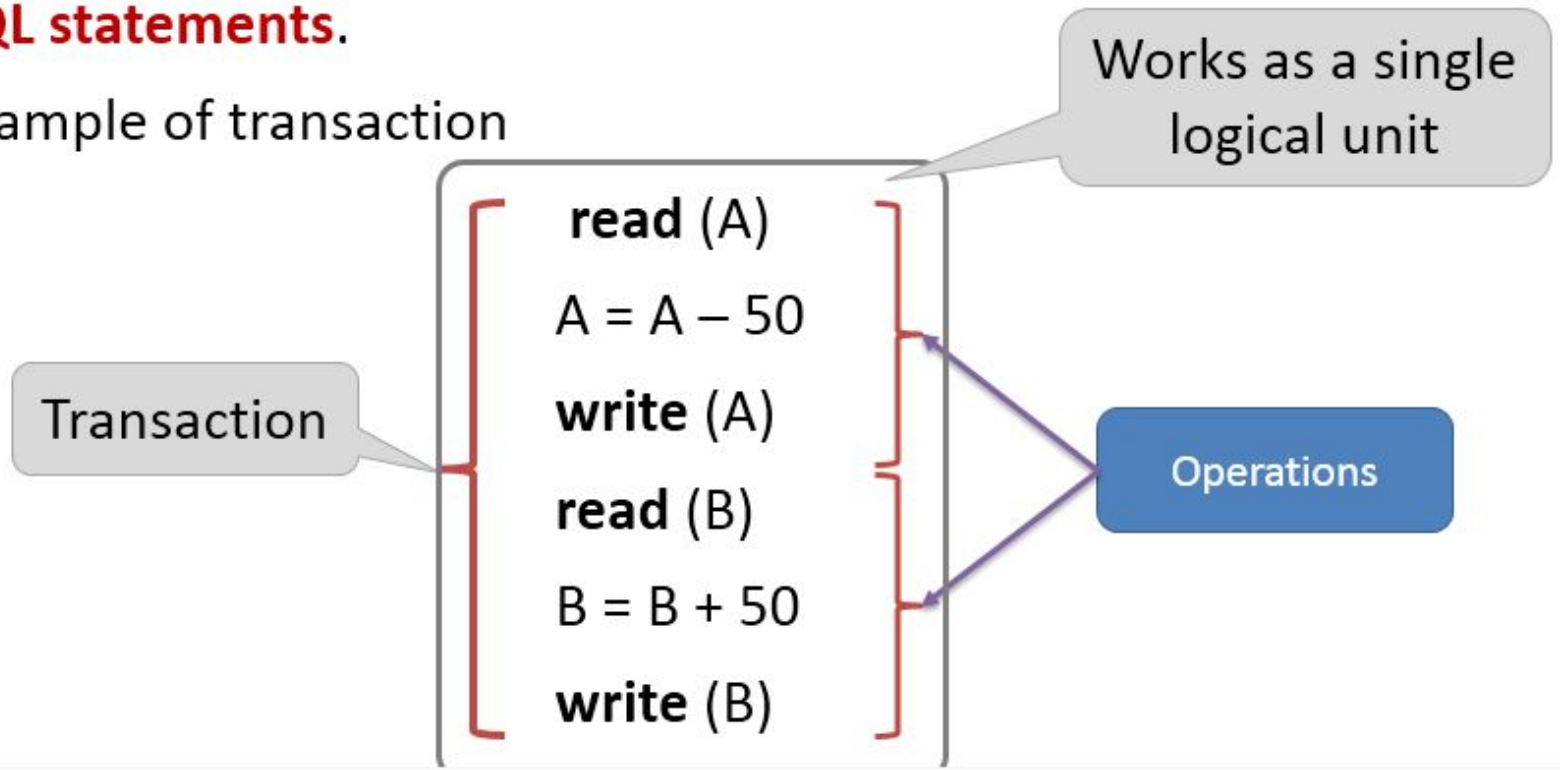
Term: October 2021 – February 2022

Faculty:
Dr. Sini Anna Alex

RAMAIAH

What is transaction?

- A transaction is a **sequence of operations performed as a single logical unit of work**.
- A transaction is a **logical unit of work that contains one or more SQL statements**.
- Example of transaction



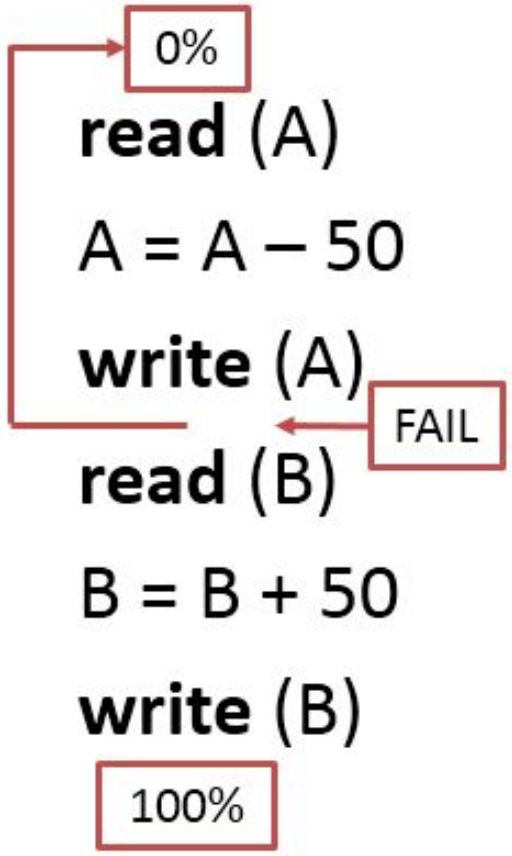
RAMAIAH
Institute of Technology

ACID properties of transaction

- Atomicity **(Either transaction execute 0% or 100%)**
- Consistency **(database must remain in a consistent state after any transaction)**
- Isolation **(Intermediate transaction results must be hidden from other concurrently executed transactions)**
- Durability **(Once a transaction completed successfully, the changes it has made into the database should be permanent)**

ACID properties of transaction

- **Atomicity**
  - This property states that a **transaction must be treated as an atomic unit**, that is, **either all of its operations are executed or none**.
  - **Either transaction execute 0% or 100%**.
  - For example, consider a transaction to transfer Rs. 50 from account A to account B.
  - In this transaction, if Rs. 50 is deducted from account A then it must be added to account B.



ACID properties of transaction

- **Consistency**
  - The **database must remain in a consistent state after any transaction**.
  - If the database was in a consistent state before the execution of a transaction, it must remain consistent after the execution of the transaction as well.
  - In our example, total of A and B must remain same before and after the execution of transaction.

A=500, B=500

A+B=1000

read (A)

A = A – 50

write (A)

read (B)

B = B + 50

write (B)

A=450, B=550

A+B=1000

ACID properties of transaction

- **Isolation**
  - **Changes occurring in a particular transaction will not be visible to any other transaction until it has been committed**.
  - **Intermediate transaction results must be hidden from other concurrently executed transactions**.
  - In our example once our transaction starts from first step (step 1) its result should not be access by any other transaction until last step (step 6) is completed.

read (A)

A = A – 50

write (A)

read (B)

B = B + 50

write (B)

ACID properties of transaction

- **Durability**
  - After a transaction completes successfully, **the changes it has made to the database persist (permanent)**, even if there are system failures.
  - Once our transaction completed up to last step (step 6) its result must be stored permanently. It should not be removed if system fails.

A=500, B=500

read (A)

A = A – 50

write (A)

read (B)

B = B + 50

write (B)

A=450, B=550

What is schedule?

- A schedule is a **process of grouping the transactions** into one and **executing them in a predefined order**.
- A schedule is the **chronological (sequential) order** in which **instructions are executed** in a system.
- A schedule is required in a database because **when some transactions execute in parallel, they may affect the result of the transaction**.
- Means if one transaction is updating the values which the other transaction is accessing, then the order of these two transactions will change the result of another transaction.
- Hence a schedule is created to execute the transactions.

RAMAIAH
Institute of Technology

Transaction Support in SQL

Possible violation of serializabilty:

Isolation level	Type of Violation: Dirty read	nonrepeatable read	phantom
READ UNCOMMITTED	yes	yes	yes
READ COMMITTED	no	yes	yes
REPEATABLE READ	no	no	yes
SERIALIZABLE	no	no	no

RAMAIAH Institute of Technology

Characterizing Schedules based on Recoverability

Schedules classified on recoverability:

Recoverable schedule:

- One where no transaction needs to be rolled back.
- A schedule S is recoverable **if no transaction T in S commits** until all transactions T' that have written an item that T reads have committed.

Cascadeless schedule:

- One where **every transaction reads only the items** that are **written by committed transactions.**

RAMAIAH Institute of Technology

Characterizing Schedules based on Recoverability

Schedules classified on recoverability (contd.):

Schedules requiring cascaded rollback:

- A schedule in which **uncommitted transactions that read an item from a failed transaction** must be rolled back.

Strict Schedules:

- A schedule in which a **transaction can neither read or write an item X until the last transaction that wrote X has committed**.

Characterizing Schedules Based on Recoverability

A shorthand notation for describing a schedule uses the symbols ***b*, *r*, *w*, *e*, *c*, and *a*** for the operations

begin_transaction, read_item, write_item, end_transaction, commit, and abort, respectively.

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

Same as S_a *except 2 commit operations*

S_b: $r_1(X)$; $r_2(X)$; $w_1(X)$; $r_1(Y)$; $w_2(X)$;c_2; $w_1(Y)$;c_1

S_b *is recoverable , even though it suffers from lost update problem*

S_c: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$; c_2;a_1;

Is not recoverable because T2 reads item X from T1, and the T2 commits before T1 commits.

RAMAIAH

Serial schedule

- A serial schedule is one in which **no transaction starts until a running transaction has ended**.
- Transactions are executed one after the other.
- This type of schedule is called a serial schedule, as transactions are executed in a serial manner.

Example of serial schedule

T1	T2
Read (A)	
Temp = A * 0.1	
A = A - temp	
Write (A)	
Read (B)	
B = B + temp	
Write (B)	
Commit	
	Read (A)
	A = A - 50
	Write (A)
	Read (B)
	B = B + 50
	Write (B)
	Commit

RAMAIAH
Institute of Technology

Interleaved schedule

- Schedule that **interleave the execution of different transactions**.
- Means **second transaction is started before the first one could end** and **execution can switch between the transactions back and forth**.

RAMAIAH Institute of Technology

Example of interleaved schedule

T1	T2
	Read (A)
	Temp = A * 0.1
	A = A - temp
Read (B)	Write (A)
B = B + temp	
Write (B)	
Commit	
	Read (A)
	A = A - 50
	Write (A)
Read (B)	
B = B + 50	
Write (B)	
Commit	

Serializability

- A schedule is serializable **if it is equivalent to a serial schedule**.
- In serial schedules, **only one transaction is allowed to execute at a time** i.e. no concurrency is allowed.
- Whereas in serializable schedules, **multiple transactions can execute simultaneously** i.e. concurrency is allowed.
- Types (forms) of serializability
  1. Conflict serializability
  2. View serializability

RAMAIAH Institute of Technology

Conflicting instructions

- Let I_i and I_j be two instructions of transactions T_i and T_j respectively.

1. I_i = read(Q), I_j = read(Q)

   I_i and I_j don't conflict

T1	T2
read (Q)	
	read (Q)

T1	T2
	read (Q)
read (Q)	

2. I_i = read(Q), I_j = write(Q)

   I_i and I_j conflict

T1	T2
read (Q)	
	write(Q)

T1	T2
	write(Q)
read (Q)	

3. I_i = write(Q), I_j = read(Q)

   I_i and I_j conflict

T1	T2
write(Q)	
	read (Q)

T1	T2
	read (Q)
write(Q)	

4. I_i = write(Q), I_j = write(Q)

   I_i and I_j conflict

T1	T2
write(Q)	
	write(Q)

T1	T2
	write(Q)
write(Q)	

Conflicting instructions

- Instructions **I_i and I_j conflict** if and only if there exists some item **Q accessed by both Ii and Ij**, and **at least one of these instructions wrote Q**.
- If **both the transactions access different data item** then they are **not conflict**.

Conflict serializability

- If a given **schedule can be converted into a serial schedule by swapping its non-conflicting operations**, then it is called as a conflict serializable schedule.

Conflict serializability

- Example of a schedule that is not conflict serializable:

T1	T2
Read (A)	
	Write (A)
Read (A)	

- We are **unable to swap instructions** in the above schedule to obtain either the serial schedule < T1, T2 >, or the serial schedule < T2, T1 >.

RAMAIAH Institute of Technology

Characterizing Schedules Based on Serializability (cont'd.)

Testing for serializability of a schedule

1. For each transaction T_i participating in schedule S, create a node labeled T_i in the precedence graph.
2. For each case in S where T_j executes a read_item(X) after T_i executes a write_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
3. For each case in S where T_j executes a write_item(X) after T_i executes a read_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
4. For each case in S where T_j executes a write_item(X) after T_i executes a write_item(X), create an edge ($T_i \rightarrow T_j$) in the precedence graph.
5. The schedule S is serializable if and only if the precedence graph has no cycles.

RAMAIAH Institute of Technology

Constructing the Precedence Graphs

Constructing the precedence graphs for schedules A and D to test for conflict serializability.

- (a) Precedence graph for serial schedule A.
- (b) Precedence graph for serial schedule B.

(a)

Time	T_1	T_2
↓	read_item(X);	
	X:=X-N;	
	write_item(X);	
	read_item(Y);	
	Y:=Y+N;	
	write_item(Y);	
		read_item(X);
		X:=X+M;
		write_item(X);

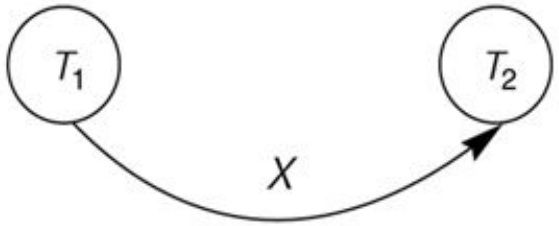
Schedule *A*

(b)

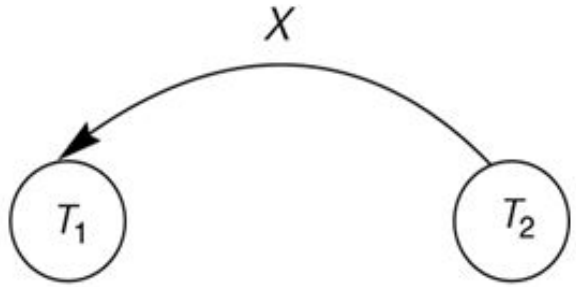
Time	T_1	T_2
↓		read_item(X);
		X:=X+M;
		write_item(X);
	read_item(X);	
	X:=X-N;	
	write_item(X);	
	read_item(Y);	
	Y:=Y+N;	
	write_item(Y);	

Schedule *B*

(a)



(b)



Constructing the precedence graphs for schedules A and D to test for conflict serializability
(c) Precedence graph for schedule C (not serializable).
(d) Precedence graph for schedule D (serializable, equivalent to schedule A).

(c)

T_1	T_2
read_item(X);	
X:=X-N;	
	read_item(X);
	X:=X+M;
write_item(X);	
read_item(Y);	
	write_item(X);
Y:=Y+N;	
write_item(Y);	

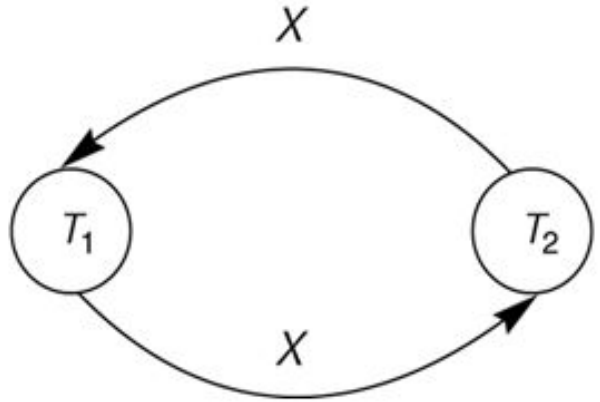
Time

Schedule C

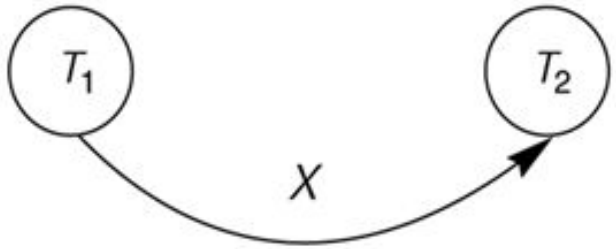
T_1	T_2
read_item(X);	
X:=X-N;	
write_item(X);	
	read_item(X);
	X:=X+M;
	write_item(X);
read_item(Y);	
Y:=Y+N;	
write_item(Y);	

Schedule D

(c)



(d)



Another example of serializability Testing

Figure 17.8
Another example of serializability testing. (a) The read and write operations of three transactions T_1, T_2, and T_3. (b) Schedule E. (c) Schedule *F*.

(a)

Transaction T_1
read_item(X);
write_item(X);
read_item(Y);
write_item(Y);

Transaction T_2
read_item(Z);
read_item(Y);
write_item(Y);
read_item(X);
write_item(X);

Transaction T_3
read_item(Y);
read_item(Z);
write_item(Y);
write_item(Z);

Another Example of Serializability Testing

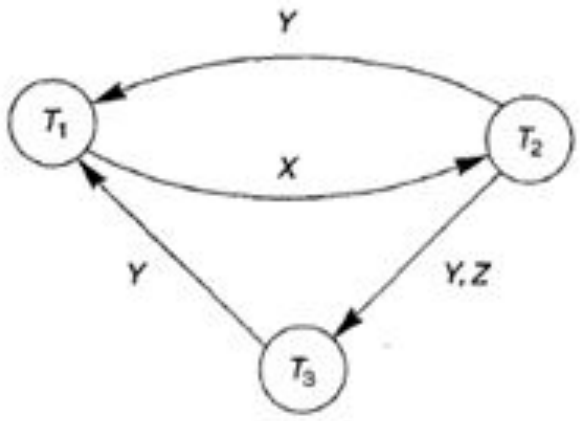
Figure 17.8
Another example of serializability testing. (a) The read and write operations of three transactions T_1, T_2, and T_3. (b) Schedule E. (c) Schedule *F*.

(b)

Time

Transaction T_1	Transaction T_2	Transaction T_3
	read_item(Z);	
	read_item(Y);	
	write_item(Y);	
		read_item(Y);
		read_item(Z);
read_item(X);		
write_item(X);		
		write_item(Y);
		write_item(Z);
	read_item(X);	
read_item(Y);		
write_item(Y);		
	write_item(X);	

Schedule E



Equivalent serial schedules

None

Reason

cycle $X(T_1 \rightarrow T_2)$, $Y(T_2 \rightarrow T_1)$
cycle $X(T_1 \rightarrow T_2)$, $YZ(T_2 \rightarrow T_3)$, $Y(T_3 \rightarrow T_1)$

Another Example of Serializability Testing

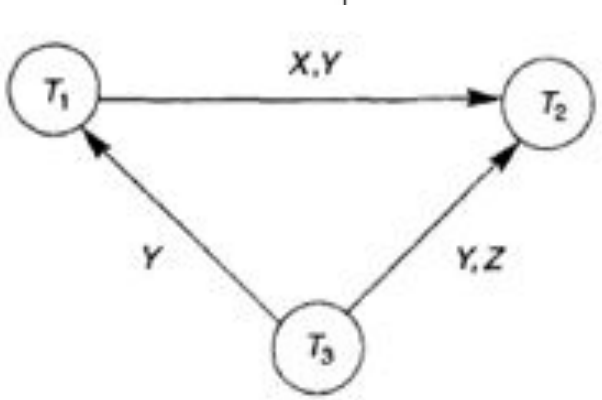
Figure 17.8
Another example of serializability testing. (a) The read and write operations of three transactions T_1, T_2, and T_3. (b) Schedule E. (c) Schedule *F*.

(c)

Time

Transaction T_1	Transaction T_2	Transaction T_3
		read_item(Y);
		read_item(Z);
read_item(X);		
write_item(X);		
		write_item(Y);
		write_item(Z);
	read_item(Z);	
read_item(Y);		
write_item(Y);		
	read_item(Y);	
	write_item(Y);	
	read_item(X);	
	write_item(X);	

Schedule F



Equivalent serial schedules

$T_3 \rightarrow T_1 \rightarrow T_2$

RAMAIAH Institute of Technology

Isolation levels in Transaction Processing

A transaction isolation level is defined by the following phenomena –

Dirty Read – A Dirty read is the situation when a transaction reads a data that has not yet been committed. Reading a value that was written by a transaction which failed.

S_b: $r_1(X)$; $w_1(X)$; $r_2(X)$; $r_1(Y)$; $w_2(X)$;c_2; $w_1(Y)$;c_1

Non Repeatable read – Non Repeatable read occurs when a transaction reads same row twice, and get a different value each time. Allowing another transaction to write a new value between multiple reads of one transaction.

- A transaction T1 may read a given value from a table. If another transaction T2 later updates that value and T1 reads that value again, T1 will see a different value.

Phantom Read – Phantom Read occurs when two same queries are executed, but the rows retrieved by the two, are different.

Based on these phenomena, The SQL standard defines four isolation levels :

RAMAIAH
Institute of Technology

Isolation levels in Transaction Processing

Read Uncommitted – Read Uncommitted is the lowest isolation level. In this level, one transaction may read not yet committed changes made by other transaction, thereby allowing dirty reads. In this level, transactions are not isolated from each other.

Read Committed – This isolation level guarantees that any data read is committed at the moment it is read. Thus it does not allows dirty read. The transaction holds a read or write lock on the current row, and thus prevent other transactions from reading, updating or deleting it.

RAMAIAH
Institute of Technology

Isolation levels in Transaction Processing

Repeatable Read – This is the most restrictive isolation level. The transaction holds read locks on all rows it references and writes locks on all rows it inserts, updates, or deletes. Since other transaction cannot read, update or delete these rows, consequently it avoids non-repeatable read.

Serializable – This is the Highest isolation level. A *serializable* execution is guaranteed to be serializable. Serializable execution is defined to be an execution of operations in which concurrently executing transactions appears to be serially executing.

Which of the following Schedules is conflict serializable? For each serializable schedule , determine equivalent serial schedules

a). r1(X); r3(X); w1(X); r2(X); w3(X);

b). r1(X); r3(X); w3(X); w1(X); r2(X);

c). r3(X); r2(X); w3(X); r1(X); w1(X);

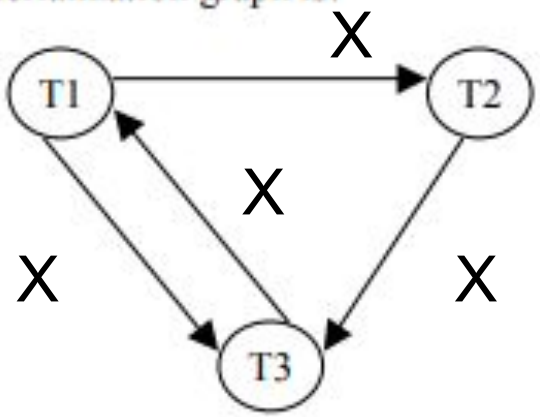
d). r3(X); r2(X); r1(X); w3(X); w1(X);

RAMAIAH Institute of Technology

Solution

a). *r1(X); r3(X); w1(X); r2(X); w3(X);*

The serialization graph is:

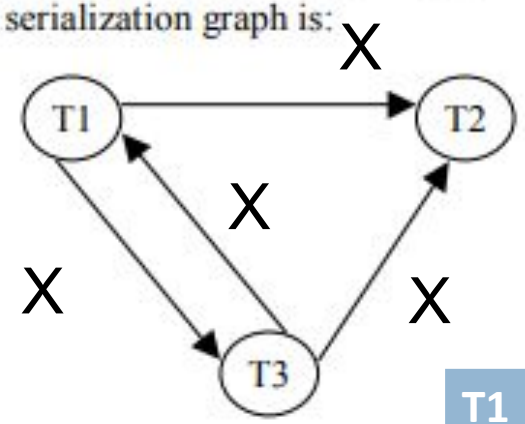


Not serializable.

T1	T2	T3
R(X)		
		R(X)
W(X)		
	R(X)	
		W(X)

b). *r1(X); r3(X); w3(X); w1(X); r2(X);*

The serialization graph is:



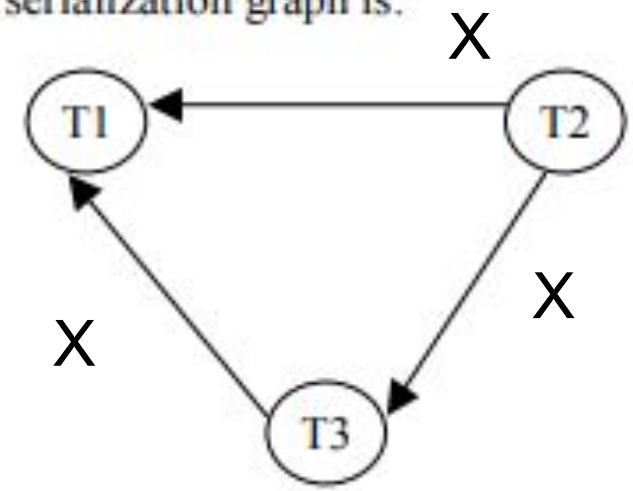
Not serializable.

T1	T2	T3
R(X)		
		R(X)
		W(X)
W(X)		
	R(X)	

Solution

c). *r3(X); r2(X); w3(X); r1(X); w1(X);*

The serialization graph is:



T1	T2	T3

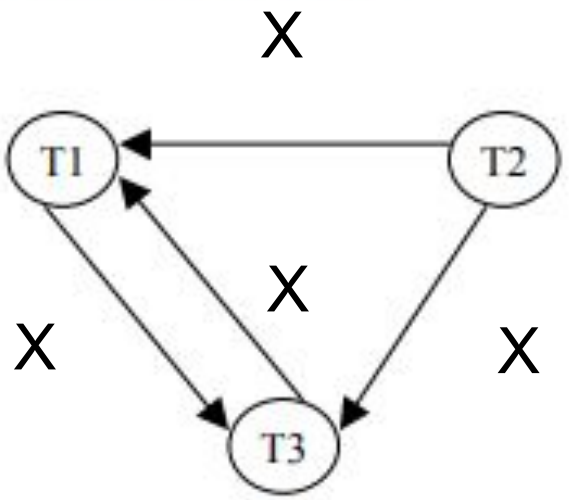
Serializable.

The equivalent serial schedule is: *r2(X); r3(X); w3(X); r1(X); w1(X);*

Solution

d). *r3(X); r2(X); r1(X); w3(X); w1(X);*

The serialization graph is:



Not serializable.

T1	T2	T3

RAMAIAH
Institute of Technology

Thank you