Unit-4

Transaction

The transaction is a set of logically related operation. It contains a group of tasks.

A transaction is a series of action or action. It is performed by a single user to perform operations for accessing the contents of the database.

Operations of Transaction

Following are the main operations of transaction:-

1) Read (x): It is used to read the value of x from the database and stores it in a buffer in main memory.

2) Write (x): Write operation is used to write the value back to the database from the buffer.

Let's take an example to debit transaction from an account which consists of following operation:

1) R(x);
2) X = X-500;
3) W(x);

Let's assum the value of x before starting the transaction is 4000.

→ The first operation reads x's value from database and stores it in a buffer.

→ The second operation will decrease the value of x by 500. So buffer will contain 3500.

→ The third operation will write the buffer's value to the database. So x's final value will be 3500.

But it may be possible that because of the failure of hardware, software or power, etc. that transaction may fail before finished all the operations in the set.

For example:- If in the above transaction, the debit transaction fails after executing operation 2 then x's value will remain 4000 in the database which is not acceptable by the bank.

To solve this problem, we have two important operations:

* Commit:- It is used to save the work done permanently.

* Rollback: It is used to undo the work done.

Transaction Property (ACID Property)

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

1) Atomicity
2) Consistency
3) Isolation
4) Durability

1) **Atomicity**:- It 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 unit 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 visible / not visible.

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

2) Consistency:

* The integrity constraints are maintained so that the database is consistent before and after the transaction.
* The execution of a transaction will leave a database in either its prior stable state or a new stable state.
* The consistent property of database states that every transaction sees a consistent database instance.
* The transaction is used to transform the database from one consistent state to another consistent state.

3) Isolation:

It shows 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.

The concurrency control subsystem of the DBMS enforced the isolation property

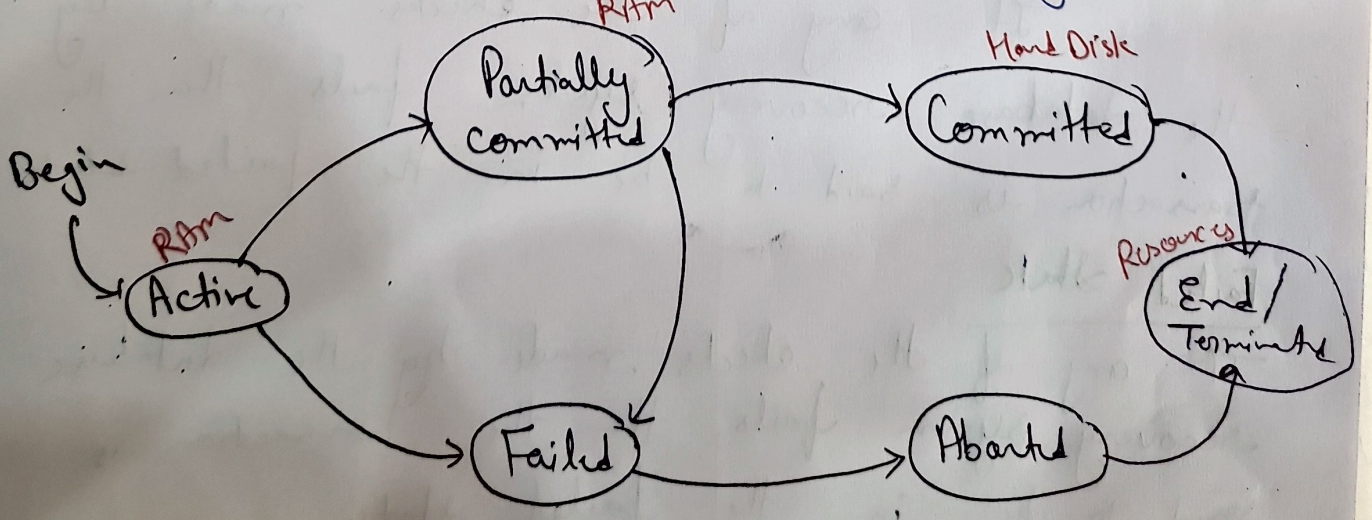
Durability :

The durability property is used to indicate the performance of the database's consistent state. It states that the transaction made the permanent changes.

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.

The recovery subsystem of the database management system has the responsibility of Durability property.

States of Transaction :- In a database, the transaction can be one of the following states.



Active State:

The active state is the first state of every transaction. In this state, the transaction is being executed.

For example: Insertion or deletion or updating a record is done here. But all the records are still not saved to the database.

Partially Committed

In the partially committed state, a transaction executes its final operation, but the data is still not saved to the database.

Committed

A transaction is said to be in a committed state if it executes all its operations successfully. In this state, all the effects are now permanently saved on the database system.

Failed State

If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.

If any of the checks made by the database recovery system fails, then the transaction is said to be in the failed state.

arted:

If any of the checks fail and the transaction has reached a failed state then the database recovery system will make sure that the database is in its previous consistent state. If not then it will abort or roll back the transaction to bring the database into a consistent state.

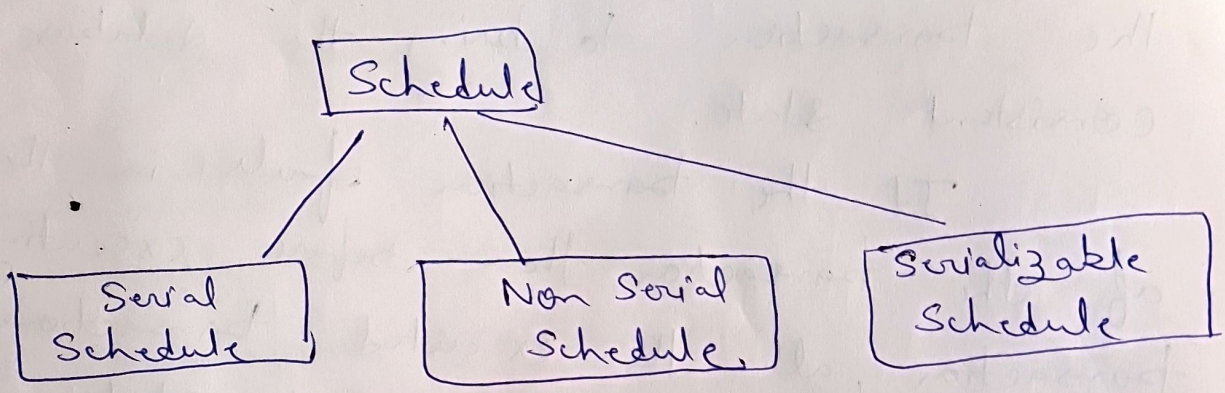
If the transaction fails in the middle of the transaction then before executing the transaction, all the executed transactions are rolled back to its consistent state.

After aborting the transaction, the database recovery module will select one of the two operations:-

1) Re-start the transaction
2) Kill the transaction

Schedule

A series of operation from one transaction to another transaction is known as schedule. It is used to preserve the order of the operation in each of the individual transaction.



1) Serial Schedule :-

The serial schedule is a type of schedule where one transaction is executed completly before starting another transaction. In the serial schedule, when the first transaction completes its cycle, then the next transaction is executed.

for eg.

T1	T2
read (A); A = A-N; write (A); read (B); B = B+N; write (B);	read (A) A = A+m write (A);

fig(1)
Schedule A

T1	T2
read (A) A = A-N; write (A); read (B); B = B+N; write (B);	read (A); A = A+m; write (A)

fig(2)
Schedule B

Suppose there are two transactions T1 and T2 which have some operations. If it has no interleaving of operations, then there are the following two possible outcomes:-

1) Execute all the operations of T1 which was followed by all the operations of T2.

2) Execute all the operations of T2 which was followed by all the operations of T1.

3) In the fig(1), Schedule A shows the serial schedule where T1 followed by T2.

4) In the fig(2), Schedule B shows the serial schedule, where T2 followed by T1.

2) Non-Serial Schedule

* If interleaving of operations is allowed then there will be non-serial schedule.

* It contains many possible orders in which the system can execute the individual operations of the transactions.

* In the given figure (3) and (4), Schedule C and Schedule D are the non-serial schedule. It has interleaving of operations.

T_1	T_2
read (A);	
A = A-N;	
	read (A);
	A = A+M;
write (A);	
read (B)	
	write (A);
B = B+N;	
write (B);	

fig (3)

Schedule C

T1	T2
read (A)	
A = A-N	
write (A);	
	read (A);
	A = A+M;
	write (A);
read (B);	
B = B+N;	
write (B)	

fig (4)

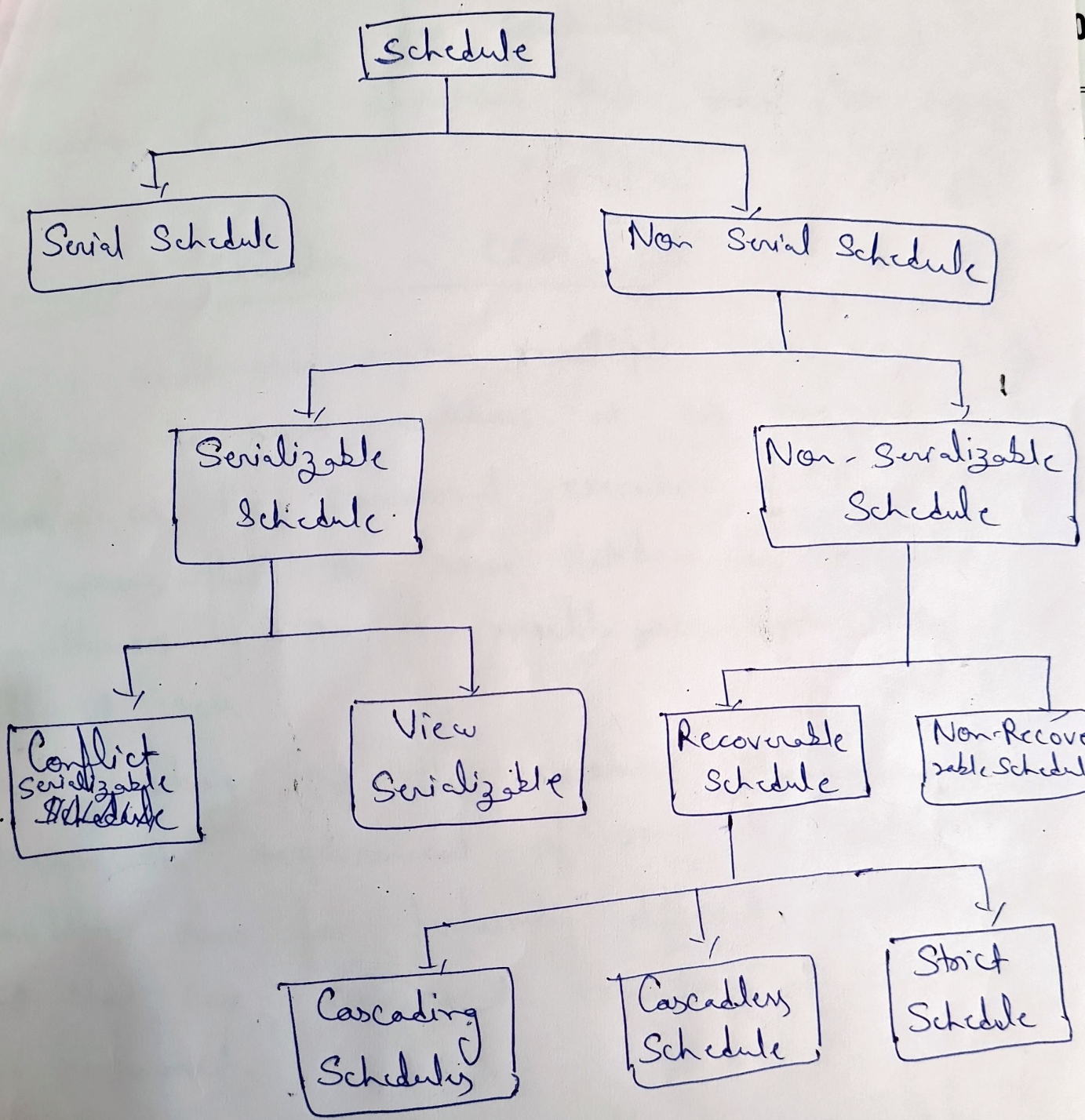
Schedule D

3) Serializable Schedule

* The serializability of schedules is used to find non-serial schedules that allow the transaction to execute concurrently without interfering with one another.
* It identifies which schedules are correct when executions of the transaction have interleaving of their operations.
* A non serial schedule will be serializable if its result is equal to the result of its transactions executed serially.

write (A);

Types of Schedules in DBMS



DBMS Concurrency Control

Concurrency Control is the management procedure that is required for controlling concurrent execution of the operations that take place on a database.

Concurrent Execution in DBMS

In a multi-user system, multiple users can access and use the same database at one time, which is known as the concurrent execution of the database. It means that the same database is executed simultaneously on a multi-user system by different users.

While working on the database transactions, there occurs the requirement of using the database by multiple users for performing different operations, and in that case, concurrent execution of the database is performed.

The thing is that the simultaneous execution that is performed should be done in an interleaved manner, and no operation should affect the other executing operations, thus maintaining the consistency of the database. Thus, on making the concurrent execution of the transaction operations, there occur several challenging problems that need to be solved.

Problems with Concurrent Execution

In a database transaction, the two main operations are READ and WRITE. So there is a need to manage these two operations in the concurrent execution of the transactions as if these operations are not performed in an interleaved manner, and the data may become inconsistent. So the following problems occur with the Concurrent Execution of the operations.

1) Lost Update Problems (W-W Conflict)

The problem occurs when two different database transactions perform the read/write operations on the same database items in an interleaved manner (i.e. concurrent Execution) that makes the values of the items incorrect hence making the database inconsistent.

Eg. Consider the diagram where two transactions T_X and T_Y, are performed on the same account A where the balance of account is $300.

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. There comes the read-write conflict between both transactions.

Eg: Consider two transaction Tx and Ty which are performing read/write operations on account A where the available balance of A is 30.

time	Tx	Ty
t1	Read(A)	—
t2	A = A+50	—
t3	write(A)	—
t4	—	Read(A)
t5	Server Down Rollback	—

t1 → Tx reads A = 300
t2 → Tx add 50 to A i.e. A = 350
t3 → Tx write updated value of A = 350
t4 → Ty reads value of A = 350
t5 → Tx rollback due to server problem
So Now A = 300
But the value for account A remains

Time	T_x	T_y
t_1	Read (A)	
t_2	A = A-50)	—
t_3	—	Read (A)
t_4	—	A = A+100
t_5	—	
t_6	write (A)	—
t_7		write (A)

$t_1 \rightarrow$ Tx reads the value of A = 300

$t_2 \rightarrow$ Tx deducts 50 from A i.e. A = 250

$t_3 \rightarrow$ Ty reads the value of A = 300

$t_4 \rightarrow$ Ty add 100 on A i.e. A = 8400

$t_6 \rightarrow$ Tx write the value of A = 250
Ty didn't update the value yet

$t_7 \rightarrow$ Ty write the value of A = 400
Now the value of Tx is lost

Hence data becomes incorrect, and database set to inconsistent.

so for Ty as committed, which is the dirty read and therefore known as the Dirty Read Problem.

3) Unrepeatable Read Problem (W-R Conflict)

Also known as the Inconsistent Retrievales Problem that occurs when in a transaction, two different values are read for the same database item.

Eg. Consider Tx and Ty, performing R/W operations of account A having value 300.

time	Tx	Ty
t1	Read(A)	—
t2	—	Read(A)
t3	—	A = A+100
t4		Write(A)
t5	Read(A)	—

t1 → Tx reads A = 300
t2 → Ty reads A = 300
t3 → Ty Update A by 100, A = 400
t4 → Ty write A = 400
t5 → Tx reads A = 400

It means that within the same transaction Tx, it reads two diffrent value of A, i.e. 300 initially and after updation reads 400. It is an unrepeatable

read and is therefore known as the Unrepeatable read problem.

Thus in order to maintain consistency the database and avoid such problems that take place in concurrent execution, management is needed, and that is the concept of Concurrency Control comes into role.

Concurrency Control

Concurrency Control is the working concept that is required for controlling and managing the concurrent execution of database operations and thus avoiding the inconsistence in the database. Thus, for maintaining the concurrency of the database, we have the concurrency control protocols.

Concurrency Control Protocols

1) Lock-Based Concurrency Control Protocol
2) Time-stamp Concurrency Control Protocol
3) Validation-Based Concurrency Control Protocol

ializable Schedule

Serializability is a term in which that is a property of the system that describes how the different process operates the shared data. If the result given by the system is similar to the operation performed by the system, then in this situation, we call that system serializable.

Types of Serializability

In DBMS, all the transaction should be arranged in a particular order, even if all the transaction is concurrent. If all the transaction is not serializable, then it produces the incorrect result.

In DBMS, there are different types of serializable. Each type of serializable has some advantages and disadvantages. The two most common types of serializable are view serializability and conflict serializability.

1) Conflict Serializability

Conflict Serializability is a type of conflict in serializability that operates the same data that should be executed in a particular order and maintains the consistency of the database. In DBMS, each transaction has some unique value, and every transaction of the database is based on that unique value of the database.

This unique value ensures that no two operations having the same conflict value are executed concurrently.

There is some conditions for the conflict serializability of the database. These are as below:-

* Both

S1

T1	T2
R(A)	
W(A)	
	R(A)
	W(A)
R(B)	

Non Conflict Pair then swap

⇒

S1

T1	T2
R(A)	
W(A)	
	R(A)
R(B)	
	W(A)

Non Conflict Pair then swap

S1

T1	T2
R(A)	
W(A)	
R(B)	
	R(A)
	W(A)

⇒ This is Equivalent to Schedule S2

Both are Conflict Equivalent Schedule

Q.

T1	T2
R(A)	
W(A)	
	R(A)
W(A)	
R(B)	

Conflict Pair then No Swapping

Conflict pair No Swapping

find Out Conflict Equivalent if Exist

so

so here No Conflict Equivalent Exist

Finding Conflict Equivalent Schedule

Eg:

Schedule 1

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

Schedule 2

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

R(A) - R(A) → Non Conflict Pair
R(A) - W(A)
W(A) - R(A) } → Conflict Pair
W(A) - W(A)

R(B) R(A)
W(B) R(A) } → Non Conflict Pair
R(B) W(A)
W(A) W(B)

Method:- Check Adjacent Non-Conflict Pair, if exist then swap the position of them

Note:- If any schedule is having Conflict Equivalent Schedule.

$$S \xrightarrow{CE} S'$$

then that schedule will be a Serializable Schedule.

There are other methods also exist to check conflict Equivalent Schedule.

Q.

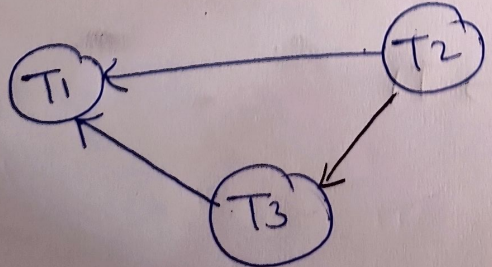
T1	T2	T3
R(X)		
		R(y)
		R(x)
	R(y)	
	R(z)	
		W(y)
	W(z)	
R(z)	..	
W(x)		
W(z)		

R-W
W-R
W-W

*Check Conflict Pairs in other transaction and draw Edges

Vertex = No. of Transaction

First Make Precedence Graph



* Check if any loop or cycle exist in graph.

→ if No loop / No Cycle Exist

↓

then schedule is Conflict Serializable

↓

then schedule is also Serializable

↓

If Schedule is Serializable then then schedule is Consistent too

There are Six Possibility

$3! = 6$ (Because total transaction = 3)

T1 → T2 → T3

T1 → T3 → T2

T2 → T1 → T3

✓ T2 → T3 → T1

T3 → T2 → T1

T3 → T1 → T2

} to check which Possibility Exists.

for this check Graph and find Indegree of Every Vertex (No Incoming Edge)

T2 → T3 → T1

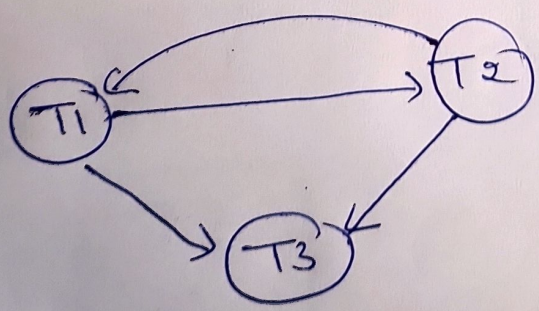
flict Serializable

T1	T2	T3
	R(y)	
	R(z)	
	W(z)	
		R(y)
		R(x)
		W(y)
R(x)		
R(z)		
W(x)		
W(z)		

Q:

T1	T2	T3
R(A)		
	W(A)	
W(A)		
		W(A)

To check whether Schedule is Conflict Serializable or not



Because loop Exist then it is Non-Conflict Serializable

Serializability

A schedule will view serializable if it is view equivalent to a serial schedule.

* If a schedule is conflict serializable, then it will be view serializable.
* The view serializable which does not conflict serializable contains blind writes.

View Equivalent

Two schedule S1 and S2 are said to be view equivalent if they satisfy the following conditions-

1) Initial Read

An initial read of both schedules must be the same. Suppose two schedule S1 and S2. In schedule S1, if a transaction T1 is reading the data item A, then in S2, transaction T1 should also read A.

T1	T2
Read(A);	
	write(A);

S1

T1	T2
	write(A);
Read(A);	

S2

Above two schedules are view equivalent because initial read operation in S1 is done by T1 and in S2, it is also done by T1.

2) Updated Read

In schedule S1, if Ti is reading A which is updated by Tj then in S2 also, Ti should read A which is updated by Tj.

T1	T2	T3
W(A);		
	W(A);	
		R(A);

S1

T1	T2	T3
	W(A);	
W(A);		
		R(A);

S2

Above two schedules are not view equal because in S1, T3 is reading A updated by T2 and in S2, T3 is reading A updated by T1.

3) Final Write

A final write must be the same between both the schedules. In schedule S1, if a transaction T1 updates A at last then in S2, final writes operations should also be done by T1.

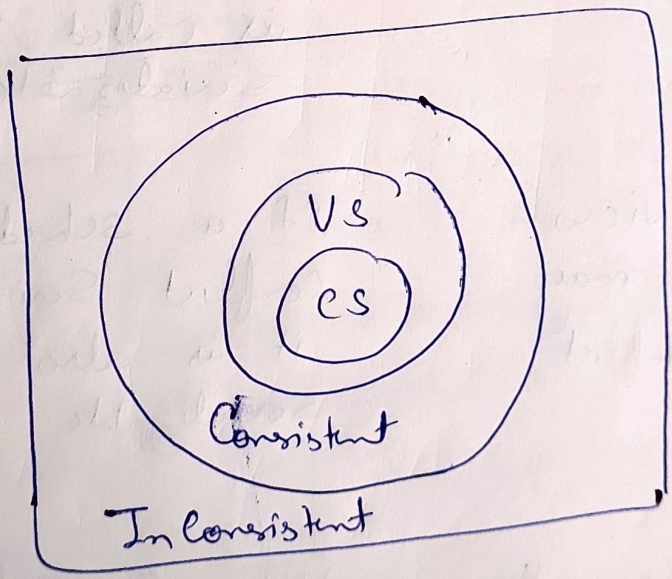
T1	T2	T3
W(A);	R(A);	W(A);

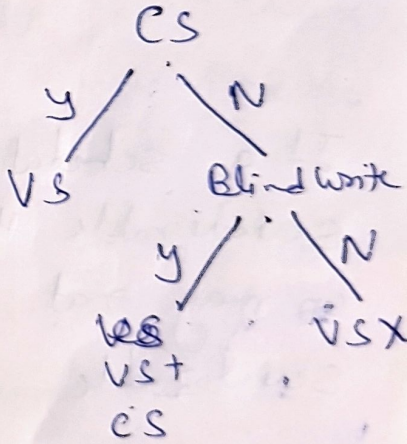
S1

T1	T2	T3
W(A);	R(A);	W(A);

S2

Above two schedule is view equal because final write operation in S1 is done by T3 and in S2, the final write operation is also done by T3.





Difference Between Conflict and View Serializab

Conflict Serializability

Two schedules are said to be conflict Equivalent if all the conflicting operations in both the schedule get executed in the same order. If a schedule is a conflict Equivalent to its serial schedule then it is called Conflict Serializable Schedule.

If a schedule is view serializable then it may or may not be conflict serializable

Conflict Equivalence can be easily achieved by reordering the operations of two transactions therefore, conflict Serializability is Easy to achieve.

View Serializability

Two Schedules are said to be view equivalent if the order of initial read, final write and update operations is the same in both the schedules. If a schedule is view equivalent to its serial schedule then it is called View Serializable Schedule.

If a schedule is conflict Serializable then it is also view Serializable schedule.

View Equivalence is rather difficult to achieve as both transactions should perform similar actions in a similar manner. Thus, View Serializability is difficult to achieve.

any transaction that performs a dirty read operation from an uncommitted transaction and also its committed operation becomes delayed till the uncommitted transaction is either committed or rollback, such type of schedules is called as Recoverable Schedules.

Types of Recoverable Schedules

1) Cascading Schedules
2) Cascadeless Schedules
3) Strict Schedules

1) Recoverable Schedules

T1	T2
R(x)	
W(x)	
	W(x)
	R(x)
Commit	
	Commit

Here Transaction T2 is reading value written by transaction T1 and the commit of T2 occurs after the commit of T1. Hence it is a Recoverable Schedule.

1) Cascading Schedules

A Cascading Rollback is a type of rollback in which if one transaction fails, then it will cause rollback of other dependent transactions. The main disadvantage of Cascading Rollback is

that it can cause CPU time wastage.

Eg.

T1	T2	T3	T4
R(A)			
W(A)			
	R(A)		
	W(A)		
		R(A)	
		W(A)	
			R(A)
			W(A)
failure			

2) Cascadeless Schedule

When a transaction is not allowed to read data until the last transaction which has written it is committed or aborted, these type of schedules are called Cascadeless schedules.

Eg.

T1	T2
R(x)	
W(x)	
	W(x)
Commit	
	R(x)
	Commit

Here the updated value of x is read by Transaction T2 only after the commit of Transaction T1. Hence the schedule is cascadeless Schedule.