3. Transaction management

* Transaction :-
  - It is a set of operations used to perform a logical unit of works.
  - Transaction generally represent change in database

Imp for internal + main exam.

* ACID Properties of transaction :-
  Every transaction must follow these properties, in order to ensure the integrity of data.

1) Atomicity :- Either All or None.
   All the changes to the data must be performed successfully or not at all.

2) consistancy :-
   Data must be in consistent state before & after transaction.

3) Isolation :-
   No other process/transaction can change the data while the transaction is running. They can not change nobody the data

4) Durability :-
   The changes made by a transaction must persist. These changes must not be lost because of any failure.

* States of Transaction :-

P.T.O

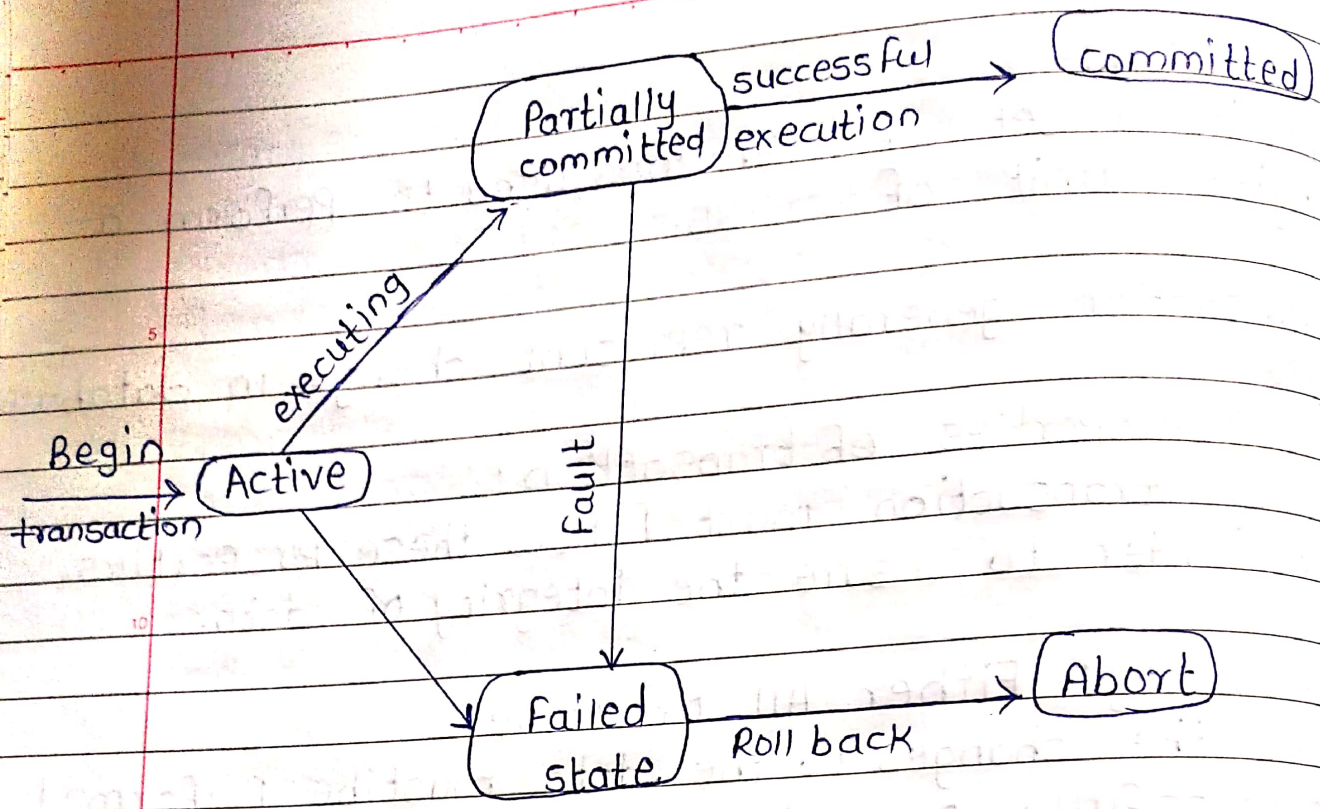


Fig. State of transaction

- When there is no failure occurs during the execution of transaction then that transaction completes it's execution successfully called as a committed transaction.

- But if any failure occurs then that transaction will no longer execute with it's normal execution called as Aborted (failed) transaction which needs to be Rollback.

1) Active state :-
When transaction begin it's execution /normal execution, Read, write statements are executed on database items.

2) Partically committed:-
When transaction execute it's final read, write

statement successfully then it enters into Partically committed state. But changes made by the transaction at this state are not Permanently stored into the database.

3) committed:-
When transaction executed successfully from start to end without any fault or error then that transaction is called as committed transaction. All changes made by committed transaction will permanently stored into the database.

4) Failed state:-
When transaction executes it's read, write operation or even transaction executes it's last instruction, at that point it enters into failed state because of some failure or fault. such failed transaction will not normally execute. hence that transaction needs to roll back.

5) Abort state:-
Failed transaction needs to remove from database. but before the transaction has been remove from the system, all the updates (changes) made by that transaction must be restore (undo) into database.

* operations of transaction:-

1) Begin:-
Starts the execution of transaction.

2) Read or write :-
This operation reads the value of data items from database or writes the value of data items into the database.

3) Commit :-
This singals the successful end (execution) of transaction & All the changes made by the transaction will be saved into the database & cannot be undone.

4) Roll back :-
unsuccessfully transaction (failed transaction) needs to be remove from database by restoring all the updates made by that transaction.

5) End :-
When transaction executes all read, write, operation successfully upto it's last instruction then we can say that there is end of transaction, but it is necessary to check whether the transaction is committed or not.

* Problem related with concurrent Execution of transaction :-

* concurrent Execution :-
It implies interleaved ving execution of operations of a transaction.

* Benefits :-
i) Helps in reducing waiting time.

ii) improves system throughput & resource utilization.

Definition :- system throughput :-
Average no. of transaction executed successfully in given time is called as system throughput.

* Problems :-

① Lost update problem (W-W conflict) :-
occurs when two transaction access the same data item, have their operations interleaved in such a way that makes the value of the database item incorrect.

T1	T2
R(A)	
A := A - 50	
	R(A)
	A := A + 100
W(A)	
	W(A)

update is lost

2) Temporary update (Dirty Read) Problem (W-R-conflict) :-
occurs when one transaction updates a database items & then the transaction fails, but it's update is read by some other transaction.

T1	T2
R(A)	
A := A + 20	
W(A)	
	R(A)
	A := A + 10
	W(A)
R(B)	
Failure	commit

Roll back

update value

3) Unrepeatable Read (R-W conflict):-
If a transaction 'Ti' reads an item value twice & the item is changed by another transaction 'Ti' in between the two read operation. Hence 'Ti' receives different values for it's two Read operation of the same item.

T1	T2
R(A)	
	R(A)
	A:=A+2000
	W(A)
R(A)	

4) Incorrect Summary Problem:-
If one transaction is calculating an aggregate summary function on a no. of records, while other transaction is updating some of these records, the aggregate function may calculate some values before they are updated & others after they are updated results in incorrect summary

T1	T2
	sum:=0
	R(A)
	sum:=sum+A
	R(4)
	sum:=sum+4
R(4)	
4=4+100	
W(4)	

* Schedules :-

A schedules of n transaction T1,T2 ---- Tn is an ordering of operations of the transactions in chronological order.

OR

when several transactions are executing concurrently, then the order of execution of various instructions is known as schedules.

* Types of schedules :-

1) Serial schedules :-

A schedule in which set of transactions are executed from start to end one by one is called as serial schedule:

T1	T2
Read (x)	
x := x+10	
write (x)	
	Read (x)
	x := x+50
	write (x)

2) Non-serial schedule:-

A schedule in which actions of different transactions are interleaved in a concurrent manner. so, that there is intermixing of instruction sequence in given schedule.

T1	T2
R(x)	
	R(x)
x := x+10	
	x := x+50
write (x)	

3) Non-recoverable schedule:-
Non-recoverable schedule is S<T1,T2>.
For each pair of transaction T1&T2 such that transaction T2 reads the database item which has been previously written by T1 appears T2 will execute commit operation before comm of t1.

T1	T2
R(A)	
A:=A+10	
W(A)	
	R(A)
	A:=A+20
	W(A)
	commit
R(B)	
B:=B-5	
W(B)	
commit	

4) Recoverable schedule :-

T1	T2
R(A)	
A:=A+10	
W(A)	
commit	
	R(A)
	A:=A+20
	W(A)
	commit

In recoverable schedule S<T1,T2>, for each pair of transaction Ti & T2 such that T2 reads the database items which has been Previously written by Ti & will execute commit operations after Ti execute commit operation.

5) complete schedule :-
schedule contain either commit or abort for each transaction whose actions are listed in it.

T1	T2
R(A)	
A:=A+10	
w(A)	
commit	
	R(A)
	A:= A+20
	w(A)
	abort

6) Strict schedule :-
In Strict schedule, transaction cannot read or write value of database item until the last transaction writes the values of database items as well as commited successfully. strict schedule is more restrictive schedule than any other schedul

T1	T2
R(A)	
A:=A+10	
w(A)	
commit	
	R(A)
	A:= A+20
	w(A)
	commit

Cascading Rollback / Abort :-

When series of transactions are inter dependent on each other for database items, then a single transaction failure will create the problem of rollbacking series of transaction which depends on each other called as cascading rollback.

T1	T2	T3
R(A)		
A:= A+10	Roll back	Roll back
W(A)		
Rollback	R(A)	
Fails	R(B)	
	A:=A+B	
	W(A)	
		R(A)
		R(C)
		A:=A+C
		W(A)
commit		
	commit	
		commit

Cascadless Schedule :-

It is a schedule which avoid cascading roll in such a way that the next transaction can only read database item if and only if the last transaction writes that database item as well as commits successfully.

T1	T2	T3
R(A)		
A:= A+10		
W(A)		
commit		
	R(A)	
	R(B)	
	A:= A+B	
	W(A)	
	commit	
		R(A)
		R(C)
		A:= A+C
		W(A)
		commit

Imp * serializability :-

A schedule 's' of 'n' transaction is serializability if it is equivalent to some serial schedule of the same 'n' transaction.

Types :-

① conflict serializability :-

If conflict equivalent is equals to serial schedule then it is called as conflict serializable schedule

conflicting operations :-

1] R(x) - W(x)

2] W(x) - R(x)

3] W(x) - W(x)

Test for conflict serializability Precedence graph is used

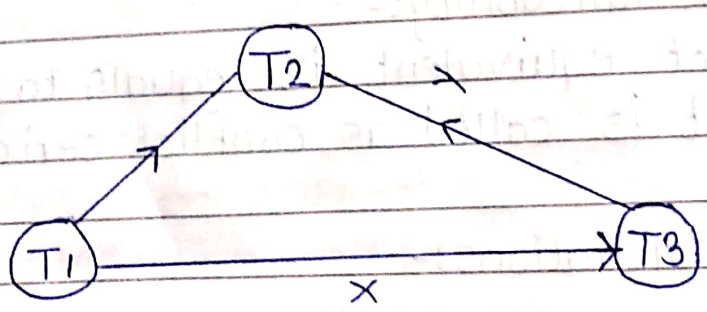
Let 's' be a schedule, construct a directed graph known as precedence graph.
Graph consist of a pair of G = (V, E)
where V: a set of vertices
E: set of edges.

A schedule is conflict serializable if and only if Precedence graph is acyclic. means in graph there is no loop/cycle is present.

Ex.1)

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

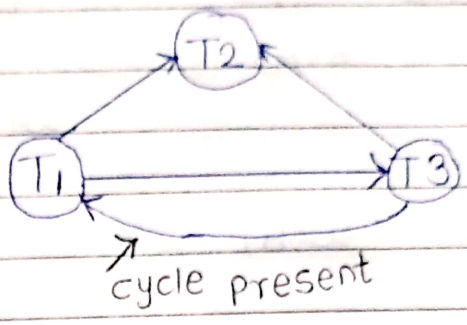
Precedence graph →



In above graph, no cycle is present.
so, Given schedule is conflict serializable.

Ex 2.

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



∴ Given schedule is not conflict serializable

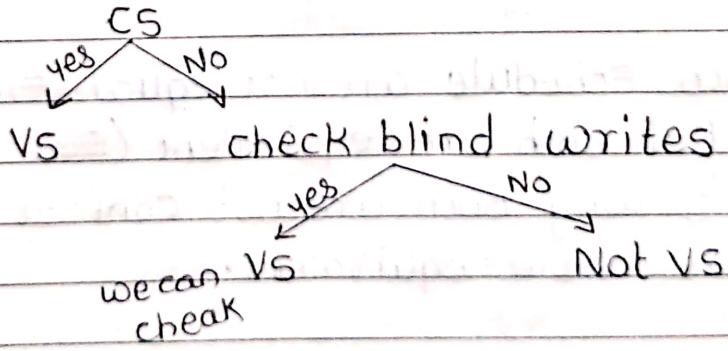
* View serializability :-

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 seriablizable which does not conflict serializable contains blind writes.

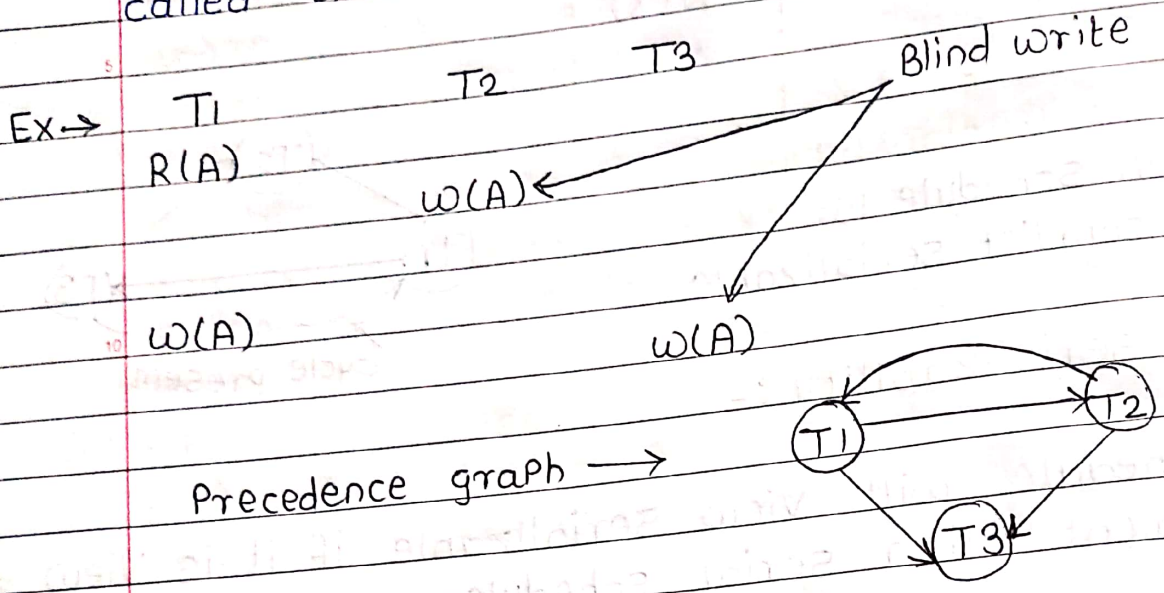
ex. Schedule (S)



* Blind write :- If any of the transaction Performs.

write operation without executing read operation for specific data item, such write operation called Blind write.

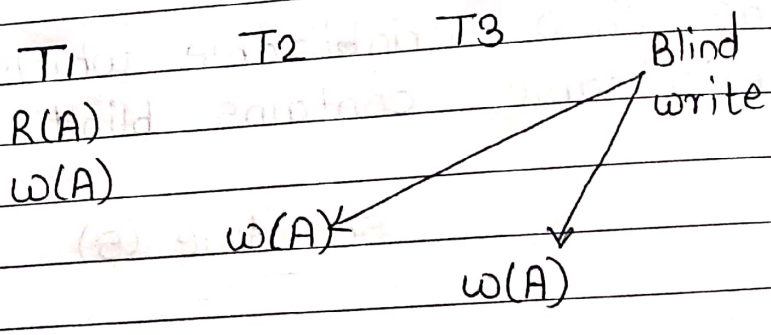
Ex→



Precedence graph →

∴ Schedule is not conflict serializable.

~~View~~ ~~Serializ~~



∴ Both schedule are not equal (=).
But both are equivalent (≡).
That's why Both are not conflict equivalent, but are view equivalent.

∴ The above schedule is view serializable.