22/11/19

Transaction Processing

A transaction is a unit of a program that possibly access and update different data items present in a database in a all or none manner.

Banking Database

ACID Properties of Transaction Process:

(1) A : Atomicity :
C : Consistency
I : Isolation
D : Durability

Atomicity: Either all operation of the transaction are reflected properly in database or none.

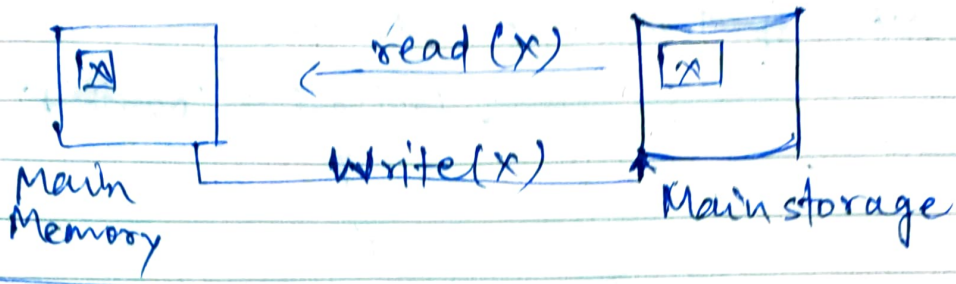
Consistency:

Isolation: When more than one transaction are processed concurrently.

Durability:

A simple Transaction Model:

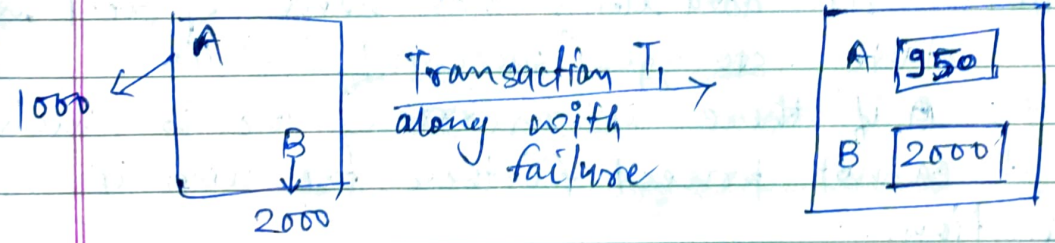
- read(x)
- write(x)




```
                A → B
T1 :    read (A)
        A := A - 50
        write(A)
        read (B)
        B := B + 50
        write (B)
```

← Transfering 50 Rs. from Acc. A to Acc. B.

① **Atomicity :-**



② **Consistency :-** The sum of the balance ~~should~~ before and after transaction must be same.

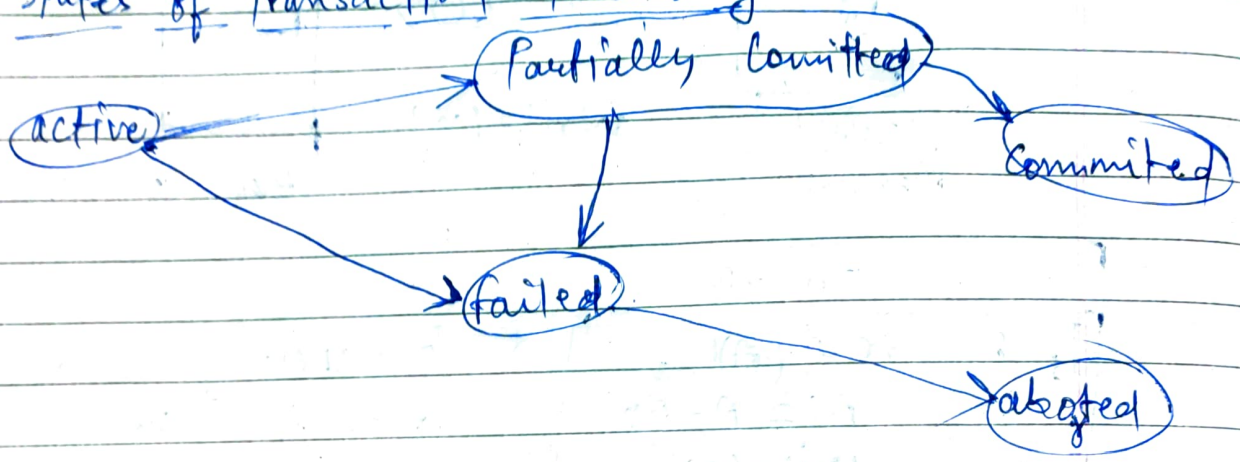
③ **Durability :-**

11/8/21

Volatile storage
Non-Volatile storage

Stable storage

States of Transaction Processing:-



- active: when it is executing its instruction.
- Partially Committed: when it has finished its last instruction but yet to update the data items in the stable storage.
- failed:
- Committed: When data items has been updated in stable storage. (Successful Execution)
- failed: if there is an error due to which it cannot proceed further for normal execution.
  - (1) Logical error
  - (2) Hardware / Network failure
  - (3) Security issues
- Aborted: when its effect has been rolled back. (Unsuccessful Execution)

A transaction is said to be terminated if it has either committed or aborted.

Concurrency:

(1) Better throughput / Better resource utilization

Throughput: Number of transaction completed per unit time.

(2) Reduce waiting time:-

T2: read(A)
temp := A X 0.1
A := A - temp
write(A)
read(B)
B := B + temp
write(B)

Serial Schedule:

Schedule - I

T_1	T_2
read(A)	
A := A - 50	
write(A)	
read(B)	
B := B + 50	
write(B)	
commit	
	read(A)
	temp := A X 0.1
	A := A - temp
	write(A)
	read(B)
	B := B + temp
	write(B)
	commit

Date
Page No

Q. How to generate concurrent schedule ?

Schedule – II

A ← 1000
B ← 2000

T_1	T_2
read(A) → 1000	
A := A − 50	
Write(A) → 950	
	read(A) → 950
	temp := A × 0.1
	95 ← A := A − temp
	Write(A) ← 855
read(B) → 2000	
B := B + 50	
Write(B) → 2050	
Commit	
	read(B) → 2050
	B := B + temp
	Write(B) → 2145
	Commit

final value of A = 855
" " of B = 2145

Total Valu = 3000

① Schedule II is a concurrent schedule which preserves the consistency property.

Q. Is it the case that every concurrent schedule ~~maintai~~ preserves the consistency property ?

Schedule: III

T_1	T_2
read (A) → 1000	
A := A - 50	
	read (A) ← 1000
	temp := A × 0.1
	A := A - temp
	Write (A) → 900
	read (B) → 2000
Write (A) → 950	
read (B) → 2000	
B := B + 50	
Write (B) → 2050	
Commit	
	B := B + temp
	Write (B) → 2150
	Commit

final value of A = 950
B = 2150
sum = 3050

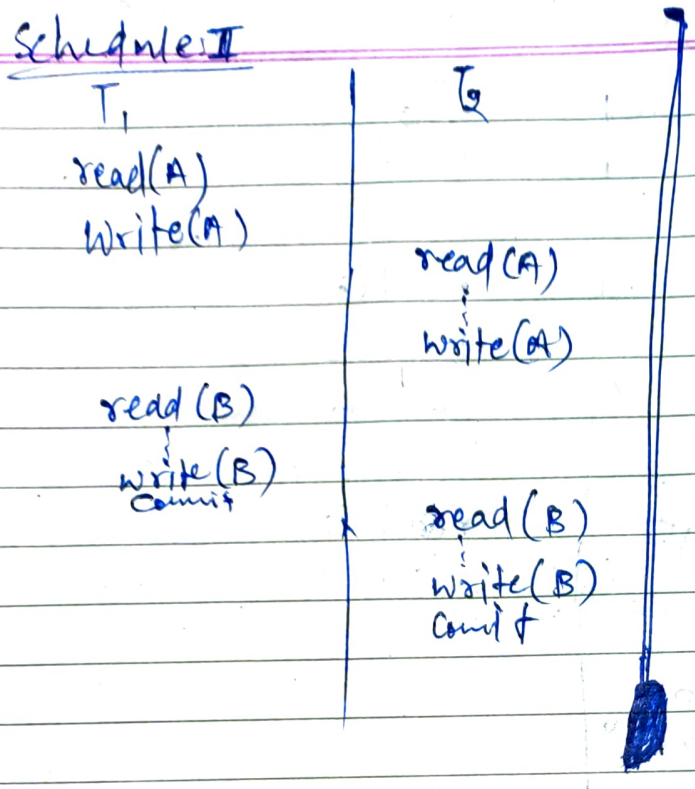
Schedule III leads to the database to an inconsistent state.

Can we make a concurrent schedule which is equivalent to a serial schedule??

Serializable Schedule

How to determine whether two schedule are equivalent?

Schedule: II



Conflict ~~Serializability~~ Serializability:

T_i	T_j
read(Q)	
	write(Q)

Case: I — T_i's statement is scheduled first and the T_j's statement is scheduled later.

Case: II — T_j's statement is scheduled first and T_i's statement is scheduled later.

T_i T_j [Q] database

T_i	T_j	
T_i : read(Q)	T_j : read(Q)	→ No Conflict
T_i : read(Q)	T_j : write(Q)	
T_i : write(Q)	T_j : read(Q)	
T_i : write(Q)	T_j : write(Q)	

↓

Conflict

We say that two instructions are non-conflicting if the order of execution does not make any difference.
This is only possible when both are read statement.

S → → →

swapping means changing the order of execution of two non-conflicting statement.

A schedule is said to be conflict serializable if we can obtain a serial schedule by a series of swap operations between two non-conflicting statements.

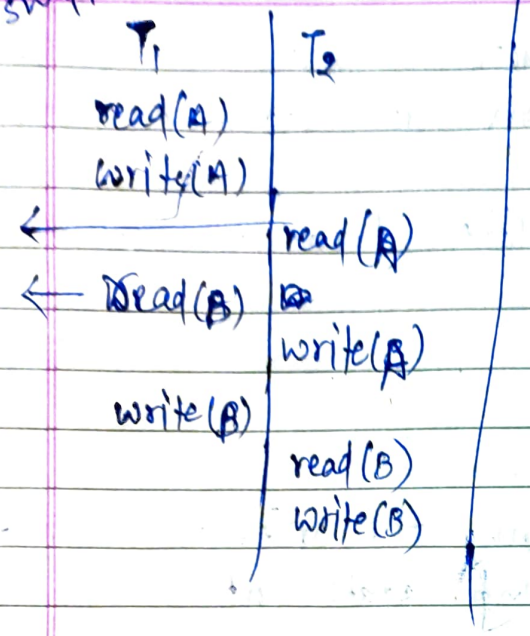
(1) If T_i and T_j operates on two different data items.
(2) If T_i and T_j operates on same data items but both of them are read statement.

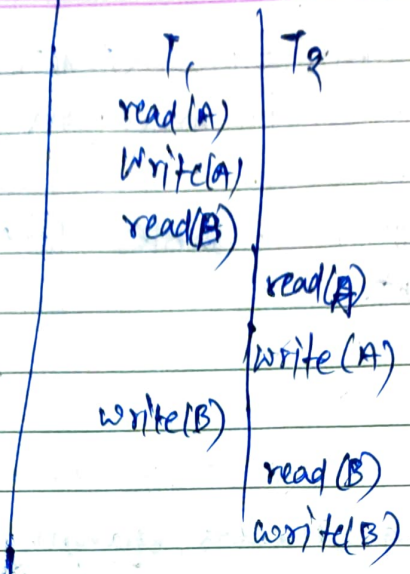
schedule: II

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

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

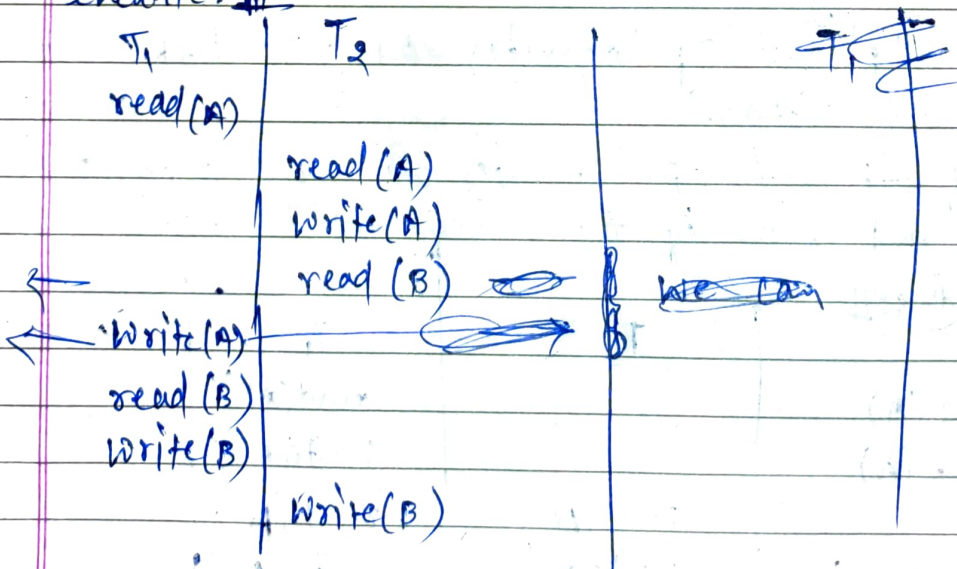
swapping two statement





From schedule II by a series of swap operation we are able to reach to a serial schedule. Hence schedule II is conflict serializable.

Schedule: III



schedule III is not conflict serializable.

29/11/24

- If there are two transaction T_i and T_j working on two different data items P (by T_i) and Q (access by T_j) Then there absolutely no conflict.
- If — — T_i and T_j ---one data item P and operation is read (R) then there no conflict.
- If — — T_i and T_j — — one data item P and one of operation is write (W) then there is conflict.

If a schedule S can be transformed into a schedule S' by a series of swap operation of non-conflicting statements then we call S and S' are conflict equivalent.

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

Que Given a schedule How to test whether it is conflict seridlizable or not ?

Solution:

Consider for the given Schedule S we construct a directed graph G(V, E) : Precedence Graph

V(G) → set of Transaction

$V(G) = \{T_1, T_2 \cdots T_n\}$

Date
Page No.

⊙ In G1 there is a directed edge $T_i \rightarrow T_j$ if any one of the following three cases arrives:

(A) T_i executes a write(Q) statement statement before T_j executes a read(Q) statement.

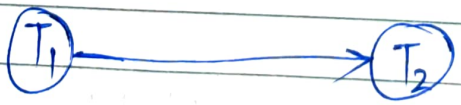
(B) T_i — a read(Q) — before
T_j — a write(Q) —.

(C) T_i — a write(Q) — before
T_j — a write(Q) —.

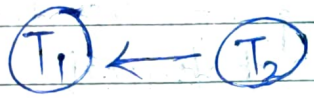
Ex

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

for this schedule precedence Graph ?



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



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

Time requirement will be proportional to number of data items involved.

If the presedence graph contains a cycle then the schedule is not conflict serializable.

How to detect a cycle in a directed graph?

Graph Traversal

$G(V, E)$

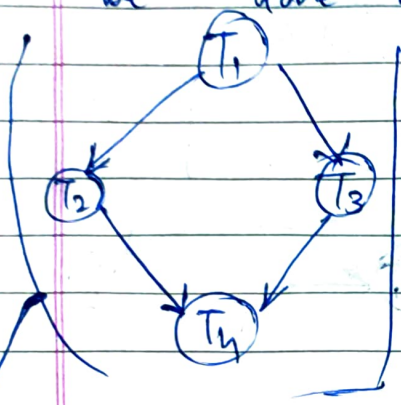
Test for conflict serializability :-

Step 1: Construct the presedence Graph:

Step 2: Execute a graph traversal algorithm for detecting to check whether there is a cycle or not.

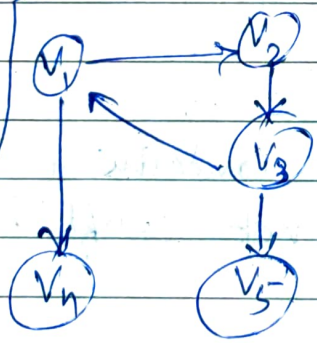
#

The order of conflict serializability can be done by performing by topological sorting.



Given a directed acyclic graph with the vertex set $\{v_1, v_2 \dots v_n\}$ the ordering of vertices is called a topological sorting. if for every $(v_i, v_j) \in E(G)$ then

$$P(v_i) < P(v_j)$$



for directed cyclic graph it cannot have a topological sorting.

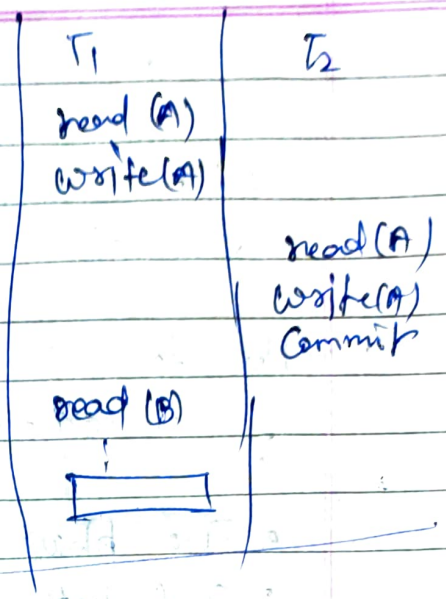
→ Topological Sorting : $T_1 \to T_2 \to T_3 \to T_4$

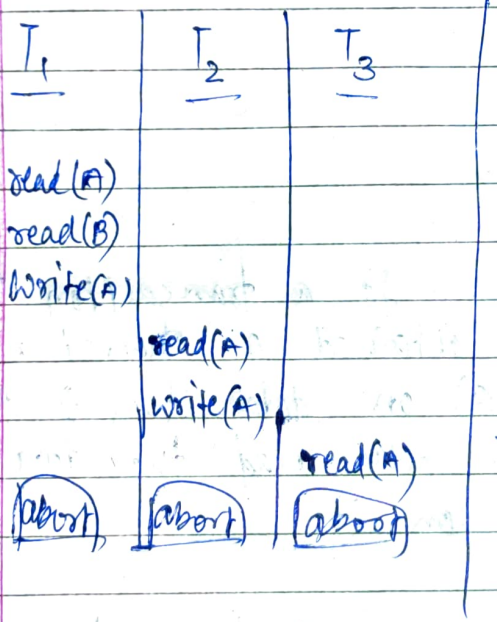
$T_1 \to T_3 \to T_2 \to T_4$

1/12/21

Such schedule are called non-recoverable schedules.

We will not let T_2 commit unless T_1 commits.





The phenomena in which a single transaction failure leads to a series of transaction roll backs. It is called cascading-roll back.

Cascadeless schedule:

A " " is one where each pair of transaction 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 option of T_j.

- Multiple transaction are getting executed + Transaction may fail.

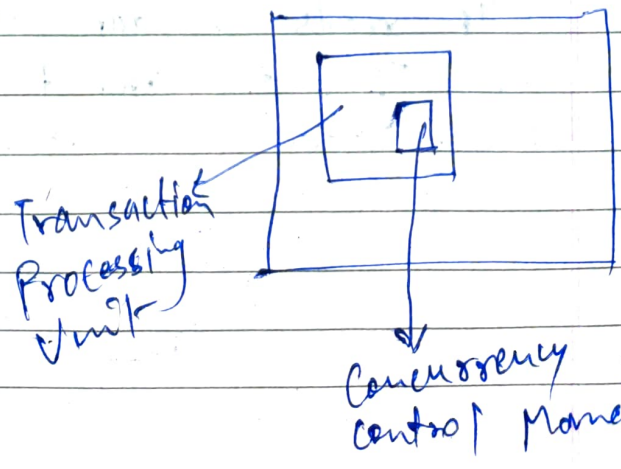
Q. How to maintain concurrency in presence of transaction failure ?

- Two phase locking Protocol
- Snap shot Isolation

Lock Based Protocols:

① shared mode lock : If a transaction T_i has obtained a shared mode lock (denoted by S) on data item Q then T_i can only read the data item but cannot write on it.

② Exclusive Mode lock :- If a transaction T_i has obtained a Exclusive mode lock (denoted by X) then T_i can do both read and write.



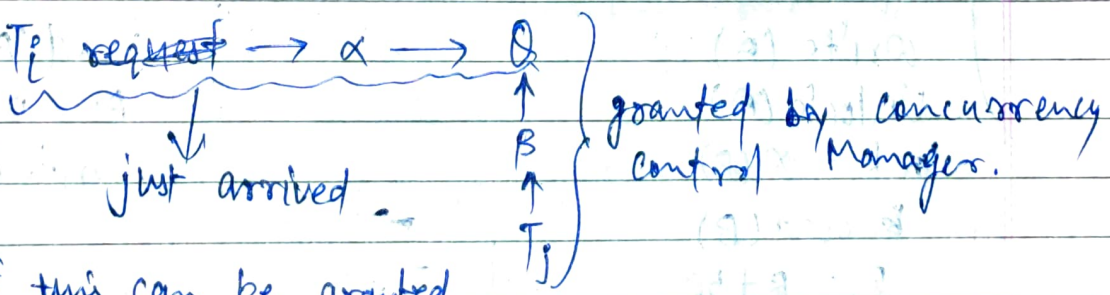
T_i → write operation
data item (Q)

- T_i has to make a request to the concurrency control Manager for having a lock in appropriate mode (exclusive mode).

- Upon having the request the concurrency control Manager ~~either~~ may grant it or ~~may~~ not.
why ?

Given a set of lock modes we can define the notion of compatibility.

① Let α and β two arbitrary lock modes.



If this can be granted immidiately ~~even if~~
The locks modes α and β are compatible with each other.

T_i \ T_j	S	X
S	True	False
X	False	False

The compatibility of locks can be represented in the form of a boolean matrix.

T_i

lock - S(Q) : T_i is requesting a shared mode lock on the data item Q.

lock - X(Q) : T_i is requesting a exclusive mode lock on the data item Q.

Unlock (Q) : ~~A transaction~~ T_i releases its lock on Q.

T_1 : Transfering 50 rs/- from Acc A to B.

T_1

```
lock - X(A)
read (A)
A := A - 50
write (A)
Unlock (A)
lock - X (B)
read (B)
B := B + 50
write (B)
Unlock (B)
commit
```

```
Read (A)
A := A - 50
write (A)
read (B)
B := B + 50
write (B)
```

T_2 :

```
lock - S(A)
read (A)
Unlock (A)
lock - S(B)
read (B)
Unlock (B)
display (A+B)
Commit
```

2/12/21

Date
Page No.

A possible schedule for concurrent execution of T_1 & T_2 :

T_1	T_2	Concurrency Control Manager
lock-X(A)		
		grant-X(A, T_1)
read(A)		
A := A - 50		
write(A)		
Unlock(A)		
	lock-S(A)	
		grant-S(A, T_2)
	read(A)	
	Unlock(A)	
	lock-S(B)	
		grant-S(B, T_2)
	read(B)	
	Unlock(B)	
	display(A+B)	
lock-X(B)		
		grant-X(B, T_1)
read(B)		
B := B + 50		
Write(B)		
Unlock(B)		

To finish the execution of both the transaction T_1 and T_2:

() T_1 is holding the lock on the data item A in an incompatible mode and this is required by T_2.

() T_2 — — — — — — — — — — — — — — — — and ~~this is~~

None of T_1 & T_2 can finish their execution. This situation is called ~~sta~~ DEADLOCK.

If the system is DEADLOCKED then it must roll back one of the transactions.

Que. Which transaction we should choose??

Goal is to minimize the amount of rework.

Deadlock ✓ Vs. Inconsistent state ✗
↓
Incorrect result will be stored in database.

Que. Can we develop a locking protocol (policy) such that the possibility of entering into a deadlocked state is very very less??