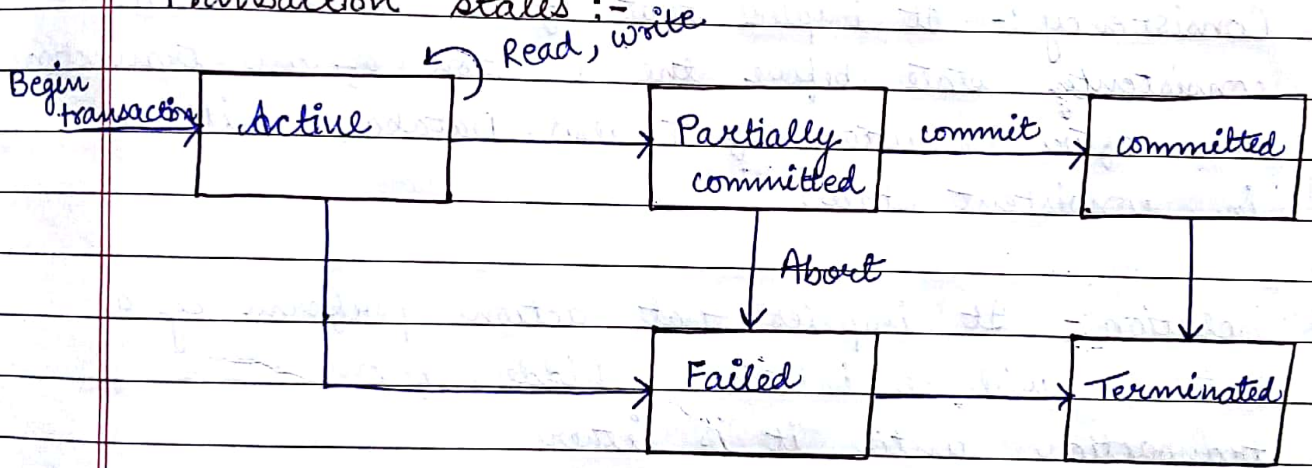
Unit 1

Transaction Management :-

- A transaction is a unit of program execution that access & possibility updates various data items.
- It has sequences of several operations that transform a consistent state of the database into another consistent state without necessarily perserving consistency of all intermediate points

Transaction states :-



1. Active :- Initial state, when the transaction is executing
2. Partially committed :- When the last statement has finished execution.
3. Failed :- On discovery that normal execution can no longer proceed.
4. Aborted :- After a rollback is performed from a failed transaction.

5. Committed :- After successful completion.

6. Terminated :- Either committed or aborted and failed.

—X—

ACID property of transaction :-

1. Atomicity :- It implies that it will run to completion as an individual unit (atomic unit).
* At the end of a unit of transaction either
   1. There is no change in database.
   2. Database has been changed in a consistent manners.

2. Consistency :- It implies that if a database was in consistenty state before the execution of an transaction then after execution of T also. Database will be in consistent state.

3. Isolation :- It implies that action perform by a Database will be isolated or hidden from other transaction until it is either
   1. execute or,
   2. terminate.

4. Durability :- It implies that updates made by a trans-action will be reflected in Database on termination of committed action. Therefore any loss or failure after committed action will ~~be~~ not cause loss of update of transaction.

Concurrency control :-

* In multiprogramming environment, programs are executed simultaneously or at the same time.

* Resource are shared among multiple user or program.

* DBMS packages allow many user / application programs to access data in a concurrent manners.

T_1	T_2	Permanent state of DB
		Initially
Read (A)		
A = A - 100		A is still 200, A's value in DB will be changed only write (A) command.
	Read (A) (T_2 will read the value of A as zero).	
	A = A * 100	
Write (A)		A = 100 (Modification done)
	write (A)	A = 20000 (as written by T_2)

* The concurrent usage of a single database can lead to inconsistencies in a Database.

* P1 ⇒ Suppose T_1 & T_2 are simultaneously accessing a data item in a Database & are modifying its value. Then update

made by T1 lost. This is k/a lost update problem of concurrency.

★ P2 ⇒ Inconsistent read problem :-

- Suppose A and B represent 2 different account in a bank.
- Let us assume T1 transfer Rs 100 from A to B.
- That is T1 should reduce Rs 100 from A & add to B
- But at same time another T_2 is running.
- T_2 has job of calculating total amount in A & B accounts.

T_1	T_2	Permanent state of DB
		Initially, A = 200, B = 500 & Sum = 0.
Read (A)		
	Read(A)	
A = A − 100		
write (A)		A = 100, B = 500, Sum = 0.
	Sum = Sum + A (Sum = 200 Now)	A = 100, B = 500, Sum = 0 (Value of sum has not been saved in DB.
Read (B)		

B = B + 100		
write(B)		A = 100, B = 600, sum = 0.
	Read (B)	
	Sum = Sum + B	
	write (sum)	A = 100, B = 600 sum = 800.

★ Sum = 800 but it actually should be 700.

Concurrency controls :-

1. Serializability :-

★ The transection execution are not serial, because the operation of the transection T_1 and T_2 are not executed consecutively.

★ The execution would have been serial, if operations of each T are executed one after the other, without any interland operations from the other T.

★ If every T executes serially without interference from other T, the result are always correct.

T_1	T_2
Read (A)	
A = A - 100	
Write (A)	
Read (B)	
B = B + 100	
Write (B)	
	Read (A)
	Sum = Sum + A
	Read (B)
	Sum = sum + B
	Write (Sum)

Fig. → Serial execution of transaction

2. Recoverability :-

★ Recovery from T failures is easy. Such schedules are called recoverable schedules.

★ A schedules is recoverable, if no T in a schedule commits, until all the T in schedule that have written a data item which T reads are commits.

★ Fig 1. is not Recoverable because T_2 reads data item A from T_1 & commits But T_1 abort after, thus invalidating the value of A read by T_2.

★ For schedule to be recoverable, T_2 should not commit until T_1 commits. If T_1 abort, T_2 should also abort.

T_1	T_2
Read (A)	
A = A − 100	
Write (A)	
	Read (A)
	A = A + 200
Read (B)	
	write (A)
	commit
Abort.	

Fig.1 → Non Recoverable Transaction schedules.

T_1	T_2
Read (A)	
A = A − 100	
write (A)	
	Read (A)
	A = A + 200
Read (B)	
	write (A)
Abort	
	Abort

Fig → Recoverable transaction schedules.

—X—

Concurrency control schemes:-

Locking :- A data item can be locked by a T in order to prevent this data item from being accessed and modified by other T.

Locking Manager :- The part of DataBase which is responsible for locking or unlocking data items.

Locks (2 types)

Exclusive Locks	Shared locks
★ T which want to read as well as modify a data item must make exclusive lock on that data item.	★ T which want to read item only & dont't modify it can make shared lock on that data item.
★ No other T can lock that data item once it is exclusively lock.	

concepts of two phase locking :-

★ It has two phases → growing phase
→ contracting phase

Growing phase :- No. of lock ↑ from 0 to max for a T.

contracting phase :- No. of lock ↓ from max to zero.

★ The T must first acquire locks on all the accquired data items.

★ It cann't unlock a data item unless it has locked all data items it needs for execution of T.

* Once a T start releasing locks, it is not allowed to request any further locks.

Transaction T_1	Transaction T_2.
Lock x (A)	
Read (A)	
A = A − 100	
write (A)	
Lock x (B)	
unlock (A)	
Read (B)	
B = B + 100	
write (B)	
unlock (B)	
	Lock x (Sum)
	Sum = 0
	Lock (A)
	Read (A)
	Sum = Sum + A
	locks (B)
	Read (B)
	Sum = Sum + B
	write (sum)
	unlock (A)
	unlock (B)
	unlock (sum).

Fig. phase locking of T_1 & T_2.

Concept of two phase locking :-
Two phase locking are of following types :-

1. Basic two phase locking

2. Conservation two phase locking :- A T lock all the required data items before a T is being executed. If any of data item it needs can't be locked the T doesn't lock any item instead it waits until all the item required are available for locking.

3. Strict two phase locking :- A T doesn't release any of its exclusive (write) locks until it commits or aborts.

4. Rigorous two phase locking :- A T doesn't release any of its locking (exclusive or shared) until it commits or abort.

Deadlock :-

★ A deadlock is said to occur when there is circular chain of T, each waiting for the release of a data item held by the next transaction in the chain.

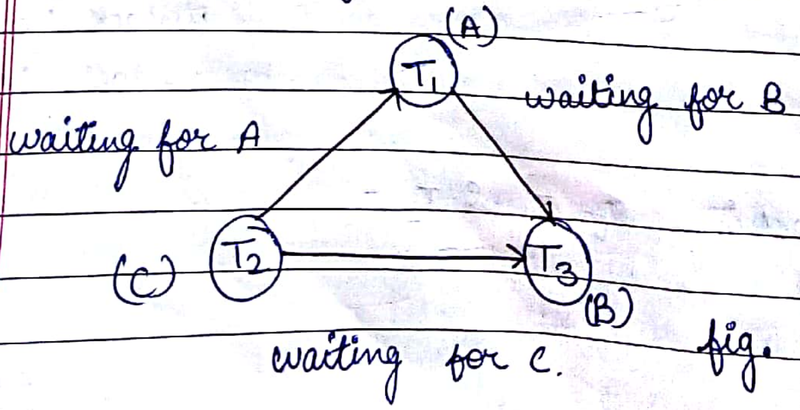


fig. Deadlock occurring situation

A deadlock would occurs under the following 4 condition

1. A data item can be accessed by only one T at a time. Another T can n't use a data item unless it is released by the T which locked it.

2. The T which requires a data item continues to wait for release by other T which locked it.

3. There is no provision of pre-emtion of data items.

4. A circular wait condition occurs.

wait for graph:- is a directed graph that contain node and directed arcs.

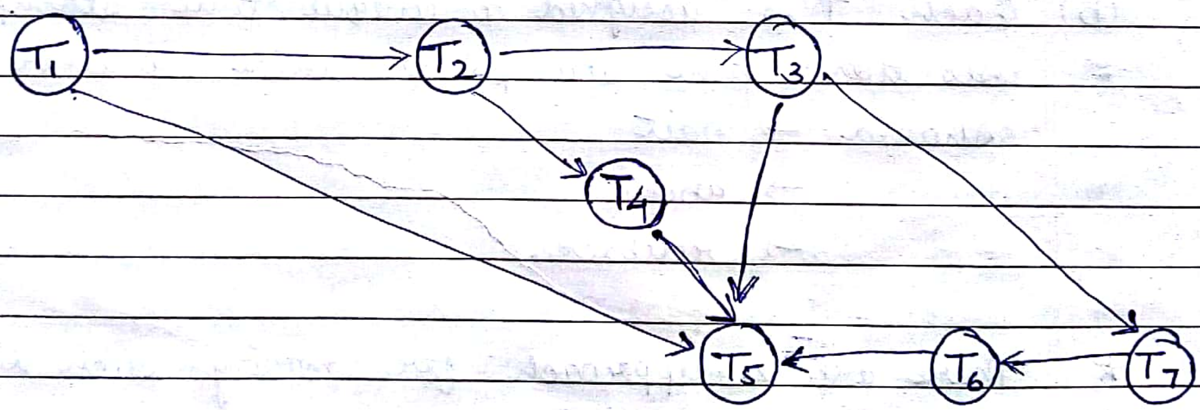


fig. wait for graph which cycle and hence deadlock.

→ T_1 is waiting for data item locked by T_2 & T_5.
→ T_2 is waiting for data item locked by T_3 & T_4.
→ T_3 is waiting for data item locked by T_5 & T_7.
→ T_4 is waiting for data item locked by T_5
→ T_6 is waiting for data item locked by T_5
→ T_7 is waiting for data item locked by T_6.

→ T_5 is waiting for data item locked by T_3.

Deadlock Detection:- The presence of cycle in the wait for graph is the reason for deadlock to occurs.

Deadlock recovery:- To recover from deadlock, the cycle in the wait for graph must be broken.

Deadlock Avoidance:- Deadlock can be avoided by using following methods.

(a) Lock all the required data items at the beginning of T.

(b) Each T is assigned a unique time stamp the system uses these time stamps to decide whether the T should → wait
→ abort
→ rollback.

★ There are 2 approach for making such division :-

1. Wait - die :- If requesting T is older than T which is holding the lock on required data item, the requesting T is allowed to waiting otherwise aborted / roll back.

2. Wound wait:- If requesting T is older than T holding locking then younger T will be aborted and roll back (younger T is abort by older T) otherwise it waits.

- X —

concurrency control based on Timestamps:-

★ Each transaction Ti is association with a unique time-stamp TS (Ti).

★ The timestamp is association as soon as Begin - transaction is encountered.

★ Timestamps are assigned in a increasing order
(i) T_1 is given $TS(T_1)$
(ii) T_2 is given $TS(T_2)$
where $TS(T_2) > TS(T_1)$

★ Two timestamp value are associated with each data item
→ W- timestamp (A)
→ R- timestamp (A)

W- timestamp (A):- is equal to largest timestamp of a T, that has executed write (A) operation successfully.

R- timestamp (A):- is equal to largest timestamp of a T, that has executed Read (A) operation successfully.

The timestamp- ordering algorithm or protocol works as follows:-

(a) If a transaction Ti tries to execute a read (A) command then
(i) If timestamp of Ti, TS (Ti) is less than W- timestamp (A), then read operation is rejected or roll-back.

(ii) TS(Ti) > W – timestamp (A), then read operation is executed & R – timestamp (A) is set to TS (Ti).

(b) If a transaction Ti tries to execute a write(A) commond, then
(i) If time stamp of Ti TS(Ti) < R – timestamp(A) then write operation is rejected or roll-back.

(ii) If TS(Ti) < W – timestamp (A), write operation is rejected or rollback.

(iii) In all other cases, write operation is executed & W – timestamp (A) is set of TS(Ti).

X

Validation Based protocol:-

★ In this scheme, updates in T are not directly applied to DB items until the reaches its ends.

★ At the end of T execution, a validation phase checks whether any of T's update violate serialazability.
(a) If serializability is not violated T is committed and DB is updated.
(b) If voileted, T is aborted and restart later.

★ There are 3 phase
(a) Read phase – A T can read value of committed data item from DB.
(b) Validation phase – checking is performed to ensure serializabity
(c) Write phase – If validation phase is successful T update are applied to DB otherwise discarded.

X