Unit - III
Transactions

Topics to cover:

1) Transaction Concepts
2) ACID properties
3) Schedule and Immediate update
4) Shadow paging
5) ARIES Algorithm
6) Serializability
7) Transaction Support in SQL
8) Need for Concurrency
9) Concurrency Control
10) Two phase locking
11) Time stamp
12) Multiversion
13) Multiple Granularity locking
14) Deadlock Handling
15) Recovery concepts
16) Recovery based on deferred

Transaction Concepts

*) They are set of logically related operation
*) They are a group of tasks
*) A transaction is an action or series of action
*) It is performed by a single user
Ex: Sending 800 from A to B

open_Account(A)	open Account(B)
old bal = X.bal	⇒ old bal = B.bal
new bal = old bal - 800	new bal = old bal + 800
X.bal = new bal	B.bal = new bal
close_Account(A)	close Account(B)

Operations of Transaction:

Read: It is used to read the value of A from the database and store them in buffer
write: It writes the value to DB from buffer
Commit: permanently changed
RollBack: undo changes by going to save point

ACID Properties

1) Atomicity: All or never, All operations performed either once or never
Abort: If a transaction aborts all changes are not visible
Commit: All changes are changed permenently

A = A - 100 B = B + 100
write(A) write(B)

2) Consistency: The updation must be consistent to avoid incorrectness
If money is sent the updation like — from A and B to b must be consistent.

3) Isolation: The data that is used by a transaction can not be used by other until that transaction is completed, until then the transaction has to wait.

4) 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 errors. When a transaction is completed then the database reaches a state known as the consistent state
*) This state can not be lost.

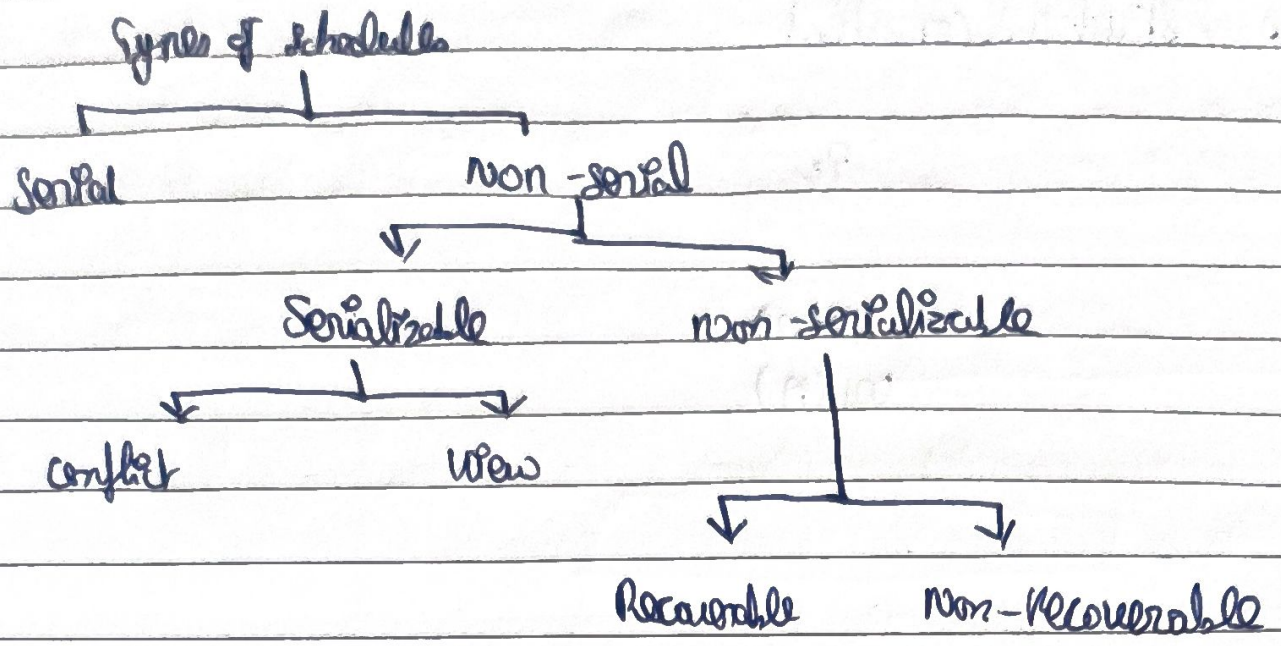
States of transaction:

1) Active state
2) partially committed state
3) committed state
4) failed state
5) Aborted state

Schedule

*) A transaction is a set of actions
*) When multiple transaction has to run concurrently, the sequence is used they are executed one by one, this is called Schedule

*) Scheduling is used to avoid concurrency problems



Serial:

*) As the name says each transaction is executed sequentially in serial order without affecting other, it is also called non-interleaved execution

*) Serial schedules are always recoverable, cascades, strict and Consistent
A serial schedules will always give correct results

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

The number of serial schedule $= n!$

Non-Serial Schedule:

*) Multiple transactions are executed concurrently without waiting for the other

* Transactions proceeds without completion of other
* If it either interleaved or mixed, they are not recoverable cascales, strict and consistent

T1	T2
R1(A)	
W2(A)	R2(A)
	W1(A)
R3(A)	
W3(A)	

Serializability

* It ensures transactions are executed serially
* It has two types → Conflict Serializability
→ View Serializability

Conflict Serializability:

* It occurs if two transactions are conflicting
* Means both accessing the same data item
* One of them is a write operation
* A schedule is a conflict serializable if one can reorder to make them execute serially

EX: T1 : Read(A) write(A)
T2 : write(A)

Operation	Transaction
Read(A)	T1
write(A)	T2
write(A)	T2

This is conflicting

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

They can be reordered as

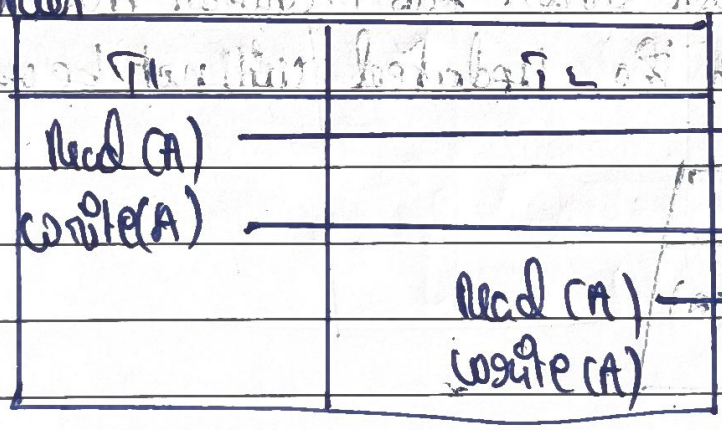
T1	T2
Read(A)	
write(A)	
Read(A)	
	Read(A)
	write(A)

View Serializability:

*1 Two schedules S1 and S2 are said to be view equivalent if they satisfy the following conditions

1) Initial read
2) update read
3) Final Read

*1 Single for each transaction the data must be accessed in same order



Concurrency

*If same data is accessed by different users at the same time, then it leads to Concurrency problems
*To control them Concurrency Control is used to fix errors

Problems in Concurrent execution:

1) Lost update problem

T1	T2
R(A)	
W(A)	
R(B)	R(A)
	W(A)
B:=A+200	R(B)
	B:=A+10
W(B)	
	W(B)

Thus one of the updates will be lost.

2) Dirty Read problem:

*When a data is updated but not committed due to Server error then in another transaction is accessed then the updated will not be used

T1	Tc
write(A)	read(A)
Server error	

3] Unrepeatable Read problem:

*) When one reads once, and the other updates, next time if the old user reads the data will be different

T1	T2
R(A)	
	W(A)
	A=A+100
	W(A)
R(A)	

Concurrency protocol

*) It ensures ACID property of transactions
*) The types are
   i) Lock Based Concurrency protocol.
   ii) Time Stamp CCP
   iii) validation based CCP.

Lock Based protocol

i) Simplistic lock protocol
ii) Pre claiming lock protocol
iii) Two phase locking protocol
iv) Strict two phase locking protocol

M T W T F S S

Page No.: YOUVA

Date:

1) Simplistic lock protocol : (out of Syllabus)

*) It locks all transactions before inserting, updating and deleting and then the transactions are unlocked

2) Pre-claiming lock protocol : (out of Syllabus)

*) It asses transactions and determines which data elements require lock

*) The transaction commence if locks are obtained, when ever the transaction is finished the lock is released

3) Two phase locking protocol: (Syllabus)

It has two phase:

1) Growing phase : New locks are acquired on data items but none of these ~~block~~ locks can be released

2) Shrinking phase : The existing locks are released

Ex	T1	T2
G	lock S(A)	
		lock-S(A)
	lock X(B)	
	-	-
S	unlock (A)	
		lock-X(C)
	unlock (B)	
		unlock(A)
		unlock(C)

G

S

Deadlocks:

In two phase locking it means when both transaction requires data for each other but they are locked

T1 | T2
lock A(B) | lock B(A)

deadlock is occurred

Blocking:

If one transaction holds a lock for a long time, other transactions that need the same lock are waiting long and blocked

Time stamp ordering Protocol

* It a method of concurrency control
* Transactions are serialized with time stamps
* When ever actions are performed the time stamp is stored

Ex:

T1 Read X
T1 read X → RTS(X) = TS(T1)
T2 write X → WTS(X) = TS(T2)

RTS → Read Time stamp
TS → Time stamp

Rule:

Allows only if time stamp is newer and rejects old time stamp

*) If T older than last write → Reject T (Abort T)
If T is older → Reject T (Abort T)

M T W T F S S

Page No.: YOUVA

Date:

Multiversion Concurrency Control:

When ever a data is updated it does not replace old but rather is a new version of it.

Deadlock Handling

A System is in deadlock state if there exists a set of transactions such that every transaction in the set is waiting for another transaction in the set.

Deadlock prevention

*) For a longer DB, deadlock prevention method is suitable

*) A deadlock can be prevented if the resources are allotted in such a way that deadlock never occurs

*) The DBMS analyzes the operations whether they can create deadlock situation or not, if they do the transaction is never allowed to be executed

Two techniques: i) wait, die ii) would wait

i) older transaction needs a data item held by younger transaction - older transaction waits
younger transaction needs a data item held by older transaction - younger transaction dies

ii) older transaction need a data item held by younger younger dies

Younger transaction needs a data item held by older transaction - younger dies

M T W T F S S

Page No.:

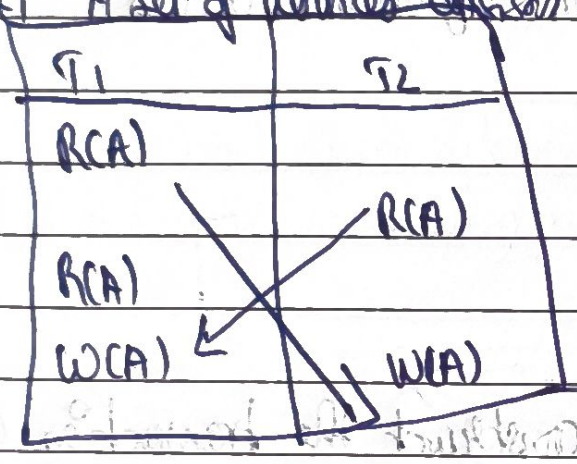
Date:

YOUVA

Deadlock detection:

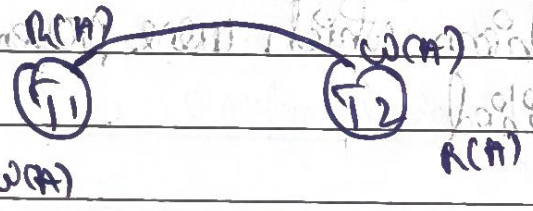
*) It is done by wait for graph method
*) If deadlock occurance is detected, then the system must try to recover from it.
*) In this method, a graph is created based on the transaction and their locks
*) If the created graph has a cycle, then there is a deadlock
*) The system keeps checking the graph if there is any cycle in the graph $(G=(V,E))$
*) A set of vertices consists of all transactions in the system

Ex



Step1: Draw vertices for all transaction

Step 2: Find Read write pair



Step 2: Repeat

Step3: Since cycle is detected deadlock is found

M T W T F S S

Page No.: YOUVA

Date:

ARIES Algorithm

ARIES → Algorithm for Recovery Isolation Exploiting Semantics

*) It is a recovery algorithm
*) Algorithm for Recovery Isolation Exploiting Semantics
*) It is based on write Ahead log protocol (WAL)
*) When database crashes during some transaction processing, we have the logs that got saved to the disk
*) ARIES has 3 phases that occur in following orders
   i) Analysis
   ii) Redo
   iii) Undo

Analysis:

*) Scan the log from start to reconstruct the transaction and dirty page table
*) Dirty page contain data that has been changed but not yet saved in disk
*) Active transactions which were present at the time of crash are identified

Redo:

*) It is started only after completing analysis
*) The log is read forward and each update is redone

Undo:

*) It is started after Redo process
*) The log is scanned backward and updates to corresponding active transaction are done

Advantages: It is flexible, concurrency control protocol is supported
Independent recovery of every page