ISSN 2349-6193
Impact Factor: 2.805

IJESMR
International Journal OF Engineering Sciences & Management Research

A DETAILED ANALYSIS OF ACID PROPERTY AND ISSUES DUE TO CONCURRENCY

Jyotiraditya Tripathi*[1], Bhawana Gautam[2] & Preeti Bala[3]
*[1]Central University of Punjab, Bathinda M.Tech. Student Centre for Computer Sci. & Tech
[2]Central University of Punjab, Bathinda M.Tech. Student Centre for Computer Sci. & Tech
[3]IMT –CDL Ghaziabad

Keywords: Transaction, concurrency control, serializable schedule, conflict and view equal.

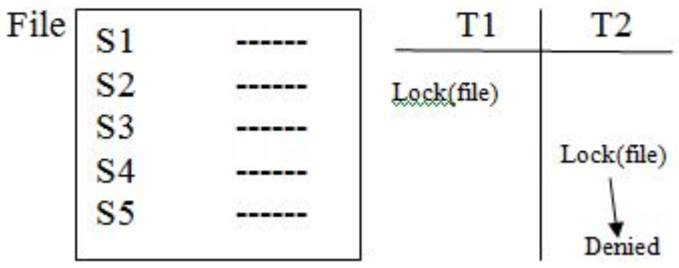
ABSTRACT

This paper presents a core concept of transaction. Basically transaction is a unit of interrelated operations to perform a unit of task. ACID property must be preserved during whole transaction where 'A' stands for atomicity, 'C' stands for consistency, 'I' stands for isolation and 'D' stands for durability. Durability is maintained by setting RAID (redundant array independent disk) architecture. Serial schedules are very good in terms of maintaining ACID property because risk in these type of schedules is very low but the throughput and degree of concurrency in serial scheduling is very low.

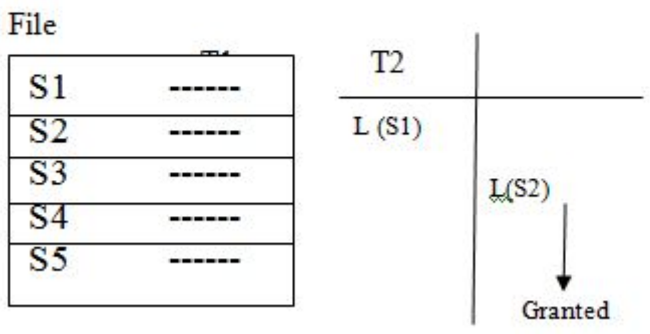
INTRODUCTION

Transaction is set of logically related operations to perform unit of work. Simultaneous execution of two or more transactions over the same database is known as concurrent execution. And number of transactions that can access same database simultaneously is termed as degree of concurrency.

To avoid error (inconsistency) in database because of simultaneous execution during transaction concurrency control approach is applied. Degree of concurrency of operating system over file level is very less because concurrency control over file level is achieved upon flat file system.



Degree of concurrency of operating system over file level is very less because concurrency control over file level uses flat file system. Whereas degree of concurrency is very high over record level because it uses DBMS file system for concurrency control.



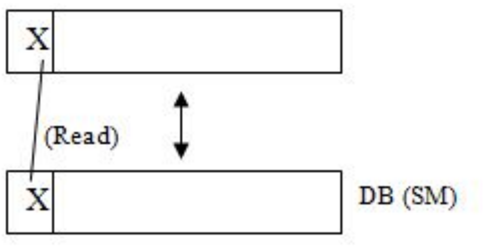
Degree of concurrency:

Record Level > Block level > File level

MAIN TRANSACTION OPERATION

X: Data item

Read (X): It is a process of accessing data item (X) from database (secondary memory) to main memory in order to know the value of (X).



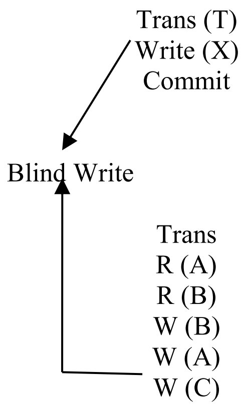
Write (X): Updating data item into database.

Update table 'R' set X=X+10;

Trans (T)
R (X)
X=X+10
W (X)
Commit

[Transaction executed successfully]

Update table 'R' set X=100:



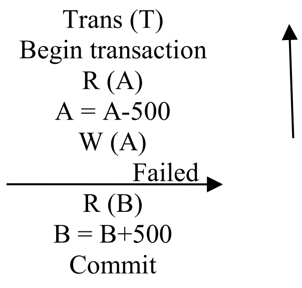
In transaction, blind write means updating the data item without reading (knowing) the value of that particular data item from database (secondary memory).

ACID PROPERTY AND ANALYSIS

This property must be restored by any transaction to maintain the integrity of data items that are involved in transaction.

A: Atomicity: If this property is maintained by transaction then it means transaction will execute all of its operations including commit or execute none of its operation (rollback).

Transferring Rs. 500 from 'A' to 'B', where 'A' is having Rs 1000 and 'B' is having Rs 2000 initially in their accounts.



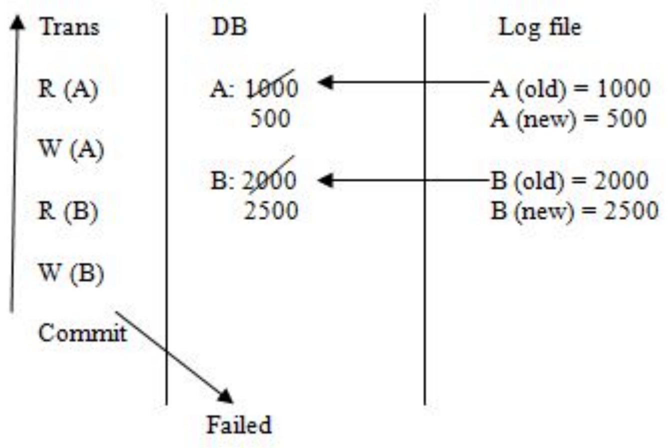
In above transaction Rs 500 was deduced from account ‘A’ and rest amount was updated in account ‘A’ but before adding the deduced amount to account ‘B’, transaction (T) failed. Because this transaction was atomic so that it rolled back otherwise data inconsistency may reflect in this transaction (means amount deduced from account ‘A’ but not added in account ‘B’).

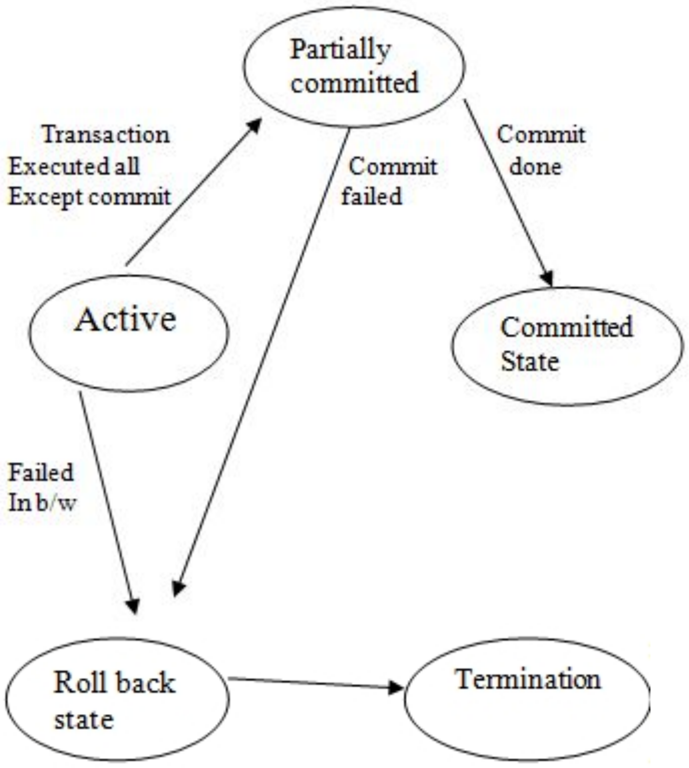
Reasons for transaction failure:

- Power failure.
- Software crash (DB overload).
- Hardware crash (HD crash).
- Concurrency controller may kill transaction.

Roll backing of transaction, if failure occurs at any point before commit operation, is done by recovery manager. Basically roll back means undo all modifications that are done by transaction which is already failed.

A transaction log is maintained by recovery management component in order to record all activities of transaction to perform roll back.





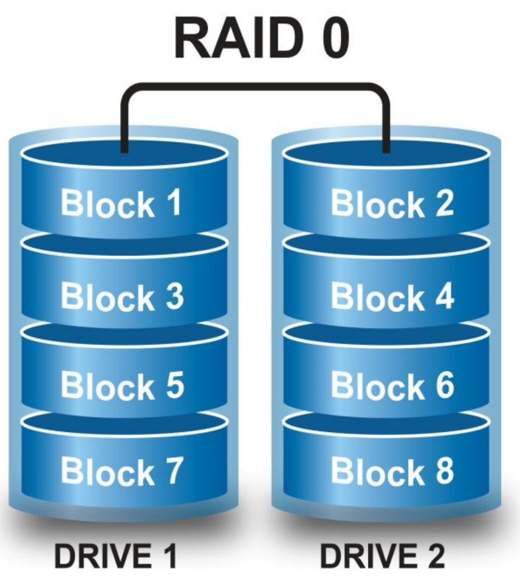
[Transaction Life Cycle]

Atomicity is maintained by recovery management component (RMC).

D: Durability: Durability property in transaction is maintained by recovery management component (RMC). Durability means transaction should able to roll back in any case of failure. This property is achieved by maintaining different type of RAID (Redundant Array of Independent Disks), depending upon importance of data.

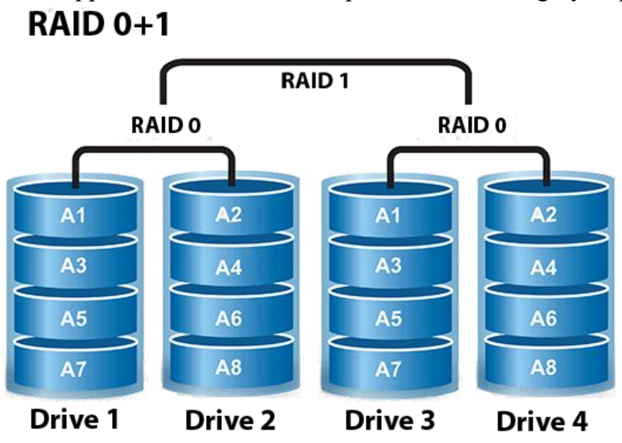
RAID Architecture

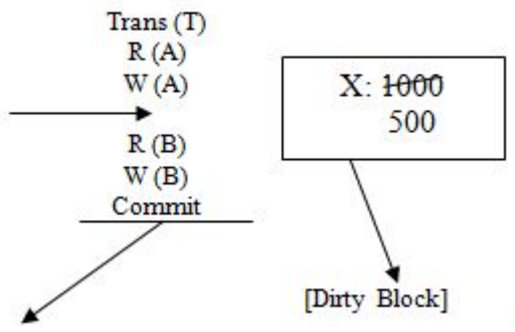
RAID – 0: This architecture is also known as stripe set or stripped volume. Basically it splits data across two or more disks uniformly without parity information. This type of architecture is not fault tolerant because it does not maintain any mirror image of database so that RAID – 0 is on high risk. Hence failure of one drive can cause entire array to fail. Mainly this setup is used to increase the performance of disk.



RAID – 1: This setup maintains exact one copy of data sets (mirror image) on two or more disks. In RAID – 1 setup any read request can be serviced by any array in disk. This architecture offers excellent read speed and write speed which is comparable to that of single drive. The main disadvantage of this architecture is that effective storage capacity is only half of disk because every data set has to be written twice. RAID – 1 is more durable than RAID – 0.

Application that requires very high availability of data uses RAID – 1 architecture. This set up is used mainly in accounting, payroll and financial applications. Hardware implementation is highly required.



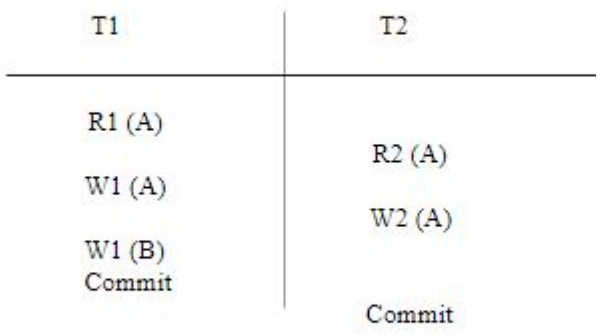


[All dirty blocks turn into clean blocks.]

Dirty blocks are maintained during transaction to store data that are modified by uncommitted transaction. These blocks turn into clean blocks after commit operation.

I: Isolation: In transaction isolation property defines a degree to which a transaction must be isolated from data modifications done by other transactions and resources.

Schedule: It is simply time order sequence of two or more transactions



T1	T2
R1 (A)	
	R2 (A)
W1 (A)	
	W2 (A)
W1 (B)	
Commit	
	Commit

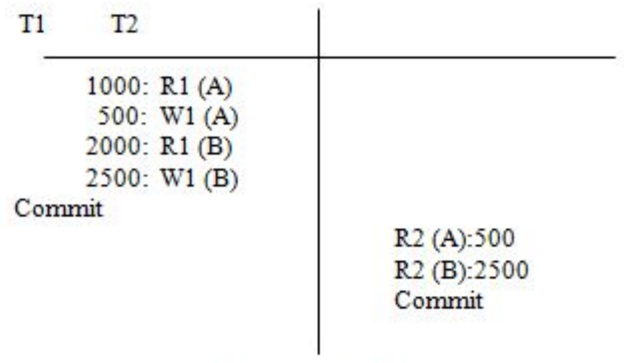
S: R1 (A) R2 (A) R2 (B) W1 (A) W2 (A) W1 (B) Commit (T1)
Commit (T2)

Serial Schedule: In this approach of transaction execution only after commit/roll back of current transaction other transaction begins.

T1: Transferring Rs 500 from ‘A’ to ‘B’ [R1 (A) W1 (A) R1 (B) W1 (B)]

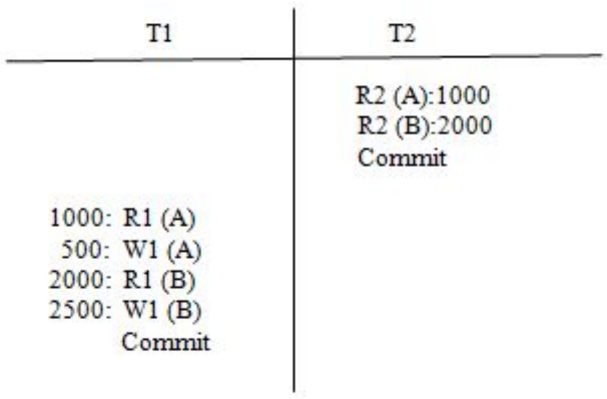
T2: Display (A+B) balance [R2 (A) R2 (B)]

IJESMR
International Journal OF Engineering Sciences & Management Research



T1	T2
1000: R1 (A) 500: W1 (A) 2000: R1 (B) 2500: W1 (B) Commit	
	R2 (A):500 R2 (B):2500 Commit

[Serial: T1→T2]

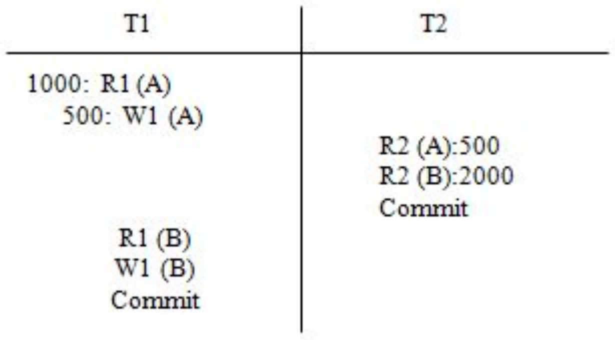


T1	T2
	R2 (A):1000 R2 (B):2000 Commit
1000: R1 (A) 500: W1 (A) 2000: R1 (B) 2500: W1 (B) Commit	

[Serial: T2→T1]

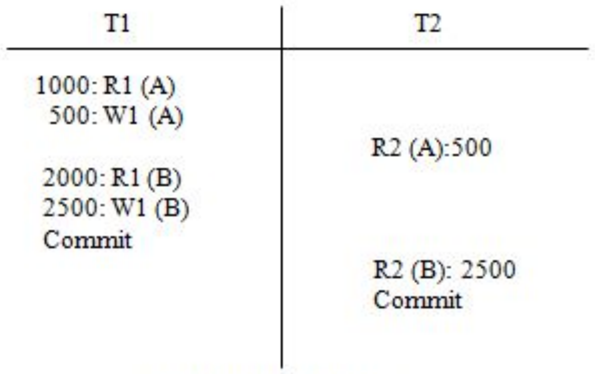
All serial schedules are consistent but also offers less degree of concurrency, less throughput, more response time and less resource utilization.

Concurrent Schedule: To increase the performance in terms of degree of concurrency, throughput, response time and resource utilization, this type of scheduling is applied. In this approach, two or more transactions are executed in interleaved (simultaneous) manner. But there is chance of inconsistency if concurrent scheduling is followed. It means all concurrent schedules are not inconsistent. Concurrency control component is responsible for avoiding inconsistency in concurrent schedules. If result of concurrent schedule transaction is not equal to serial schedule then there is chance of inconsistency



T1	T2
1000: R1 (A)	
500: W1 (A)	
	R2 (A):500
	R2 (B):2000
	Commit
R1 (B)	
W1 (B)	
Commit	

[Inconsistent Schedule]

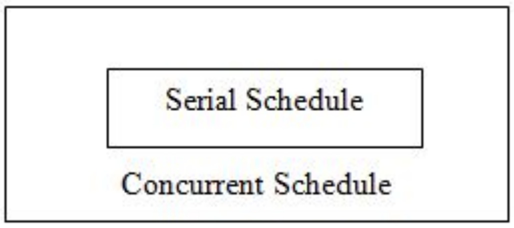


T1	T2
1000: R1 (A)	
500: W1 (A)	
	R2 (A):500
2000: R1 (B)	
2500: W1 (B)	
Commit	
	R2 (B): 2500
	Commit

[Consistent Schedule]

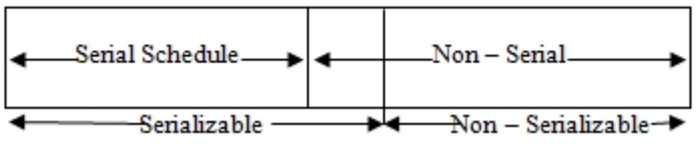
If concurrent execution of two or more transaction result is equal to result of any serial schedule. Then we say that isolation property is preserved. Hence in 2nd concurrent schedule, isolation property is preserved.

Therefore from above analysis, it can be concluded that, concurrent schedules are superset of serial schedules.



Types of Schedules

a) Serial Schedule: Non – serial execution is not allowed.
b) Serializable: Non – serial execution is allowed but must be equal to any serial schedule.
c) Non – Serializable: These schedules are not equal to any schedule, so ACID property is not preserved.



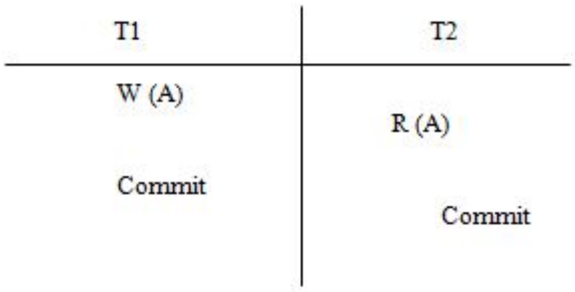
C: Consistency: Transaction which is executed by user should be logically correct. It means data should be consistent in every phase of transaction. Any database that is involved in transaction must change affected data only in allowed ways.

CONCURRENT SCHEDULE ISSUES

There are several problems that arise due to concurrent execution, these are listed below:

- Read – Write Problem
- Write – Read Problem
- Write – Write Problem

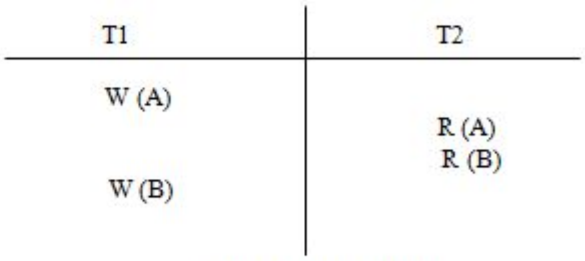
Dirty Read or Uncommitted Read: If two transactions, say T1, T2, are involved in schedule, 'S' and transaction T2 is reading data item 'A' which is modified by uncommitted transaction T1 then this read operation by T2 is known as dirty read operation.



T1	T2
W (A)	
	R (A)
Commit	
	Commit

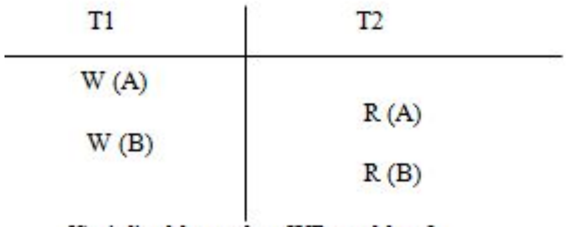
Write – Read Problem: This problem arises due to these two conditions that are listed below:

- Schedule 'S' must be in non – serializable condition.
- At least one uncommitted read should be present in schedule 'S'.



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

[Non – Serializable]
[WR Problem]



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

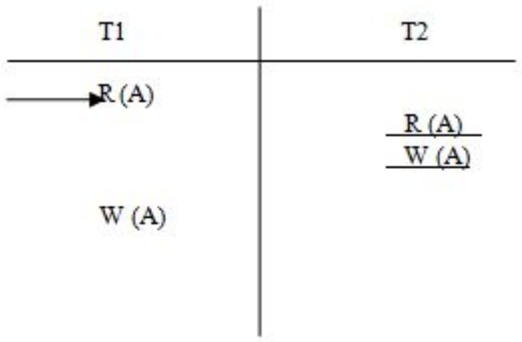
[Serializable and no WR problem]

Simultaneous Read – Write operation means transaction T2 is updating 'A' which is read by uncommitted transaction 'T1'.

Read – Write Problem: This problem arises in concurrent execution due to these two conditions:

- Schedule 'S' is non – serializable.
- Some simultaneous RW operations exist in 'S'

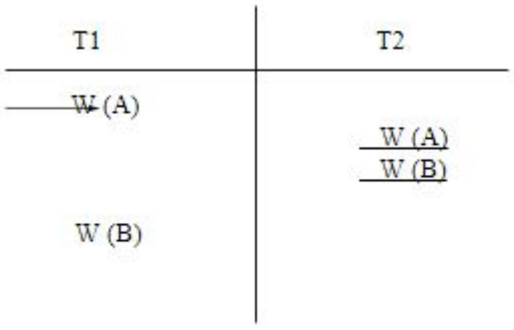
IJESMR
International Journal OF Engineering Sciences & Management Research



[Non – Serializable RW Problem]

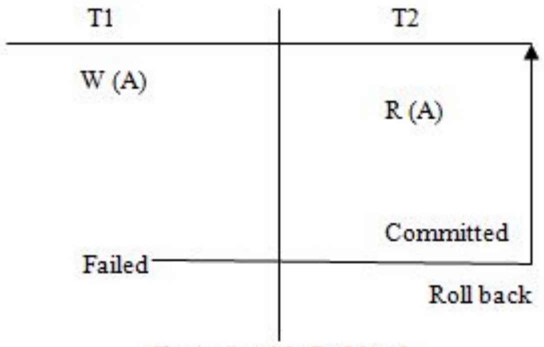
Simultaneous Write – Write operation means T2 updates ‘A’ which is modified by uncommitted transaction ‘T1’.
Write – Write Problem: This problem arises in concurrent execution due to these two conditions:

- Schedule ‘S’ is non – serializable.
- Some simultaneous’ WW’ operations exist in ‘S’



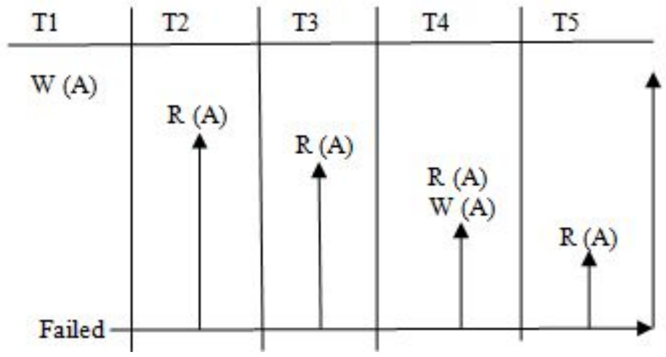
[Non – Serializable WW Problem]

To avoid RW, WR, WW problems, restriction of non – serializable is done by concurrency control component.
Irrecoverable Problem: When truncation T2 reads data item ‘A’ which is modified by uncommitted transaction T1 and after committing of transaction T2, T1 fails just before commit, this condition force to rollback transaction ‘T2’ which has already committed. This condition comes under irrecoverable problem.

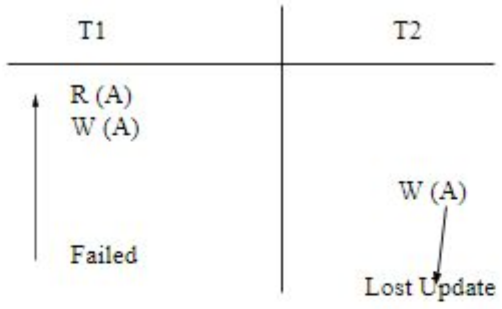


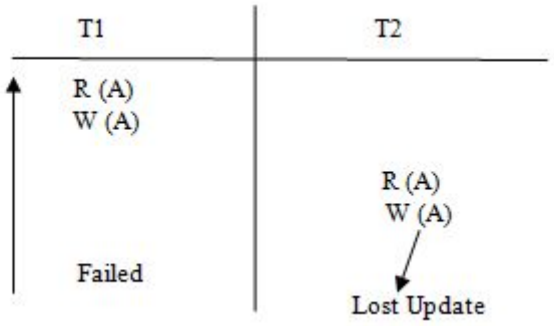
[Irrecoverable Problem]

transaction it forces to roll back other transactions. This condition comes under cascading roll back problem. The main disadvantage of this is wastage of CPU execution time and increased I/O cost



Lost Update Problem: This problem occurs if simultaneous WW operation is allowed in schedule

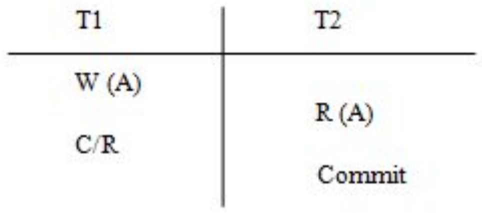




a) Recoverable Schedule: Schedule 'S' is recoverable iff:
- No uncommitted reads in 'S'.

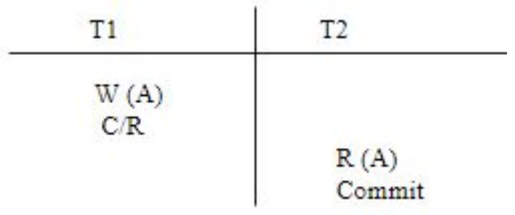
OR

If transaction T2 reads 'A' which is updated by transaction T1. Then commit of T2 must be delayed till commit/roll back of transaction T1



Because of this restriction recoverable schedule is not free from WW, WR, RW, cascading roll back and lost update problem.

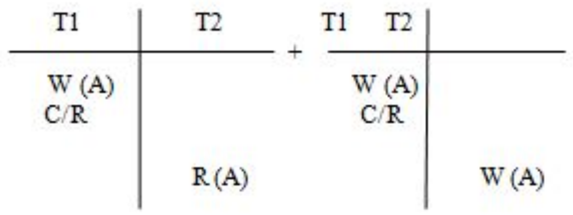
Cascade less Roll back Recoverable Schedule: If transaction T1 writes 'A' then read operation of 'A' by other transaction T2 should be delayed till C/R of transaction T1



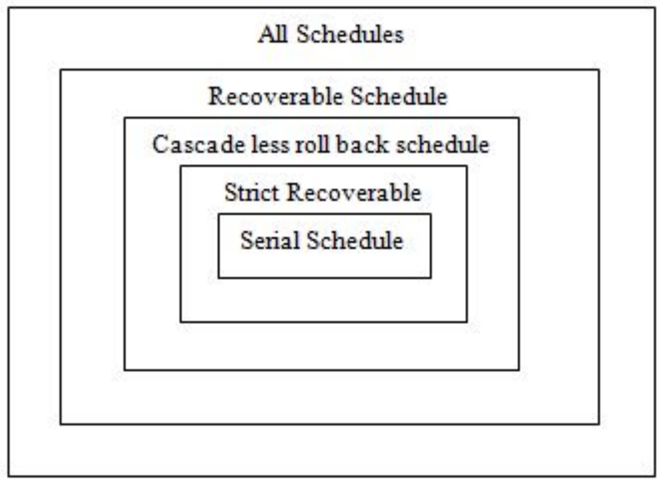
Cascade less roll back schedules are free from irrecoverability, WR problem and cascading roll back.
But these type of schedules are not free from RW, WW and lost update problem.

a) Strict Recoverable Schedule: If transaction T1 performs W (A) and if other transaction T2 requests for R (A)/W (A) should be delayed till commit or roll back of T1.

[Cascade less roll back Schedules]+[No lost update]



Strict recoverable schedules are free from irrecoverabilty, cascading roll back, lost update, WR and WW problem. But this is not free from RW problem.



CONCLUSION

From above explanation it can be concluded that in concurrent schedules it is very important to maintain ACID property. All serial schedules are consistent but they provide very less degree of concurrency and resource utilization is very low hence to improve the resource utilization and degree of concurrency transactions are executed in concurrent manner but there are chances of inconsistency in concurrent schedules so that to maintain consistency there are some concurrency control protocols for resource allocation like two phase locking protocol, timestamp ordering protocol.

REFERENCES

1. *'Database System Concepts',Sixth EditionAviSilberschatz Henry F. KorthS. Sudarshan.*
2. *Transaction management: https://www.ijarcsse.com/docs/papers/Volume_3/7_July2013/V3I7-0526.pdf*
3. *Fundamentals of DBMS, Lakhanpal Publisher, III edition (2008).*
4. *Transaction management: https://docs.oracle.com/cd/B19306_01/server.102/b14220/transact.html*
5. *R. G. Tiwari, S. B. Navathe, and G. J. Kulkarni, "Towards Transactional Data Management over the Cloud," 2010 Second Int. Symp. Data, Privacy, E-Commerce, pp. 100–106, Sep. 2010*