TRANSACTION PROCESSING

Dr.M.Venkatesan
Assistant Prof,CSE,NITK

Transaction Management

- A transaction is a program unit whose execution may change contents of a database.

Database before transaction → Database after transaction (consistent state)

After transaction operation

‘Or’

- Transaction is used to represent a logical unit of database processing that must be completed in its entirely to ensure correctness.

Example:

Bank(db) – 10,000
Debit 5000

R(a) → current
A-5000
W(a)

Transaction

ACID properties of Transaction

- **<u>Atomicity</u>(ALL or None):**

Ensure that a transaction will run to completion as an individual unit, at the end of which either no changes have occurred to the database, or database has been changed in a consistent manner.

Example:

<u>Transaction(Ti):</u>

Read(A,a)

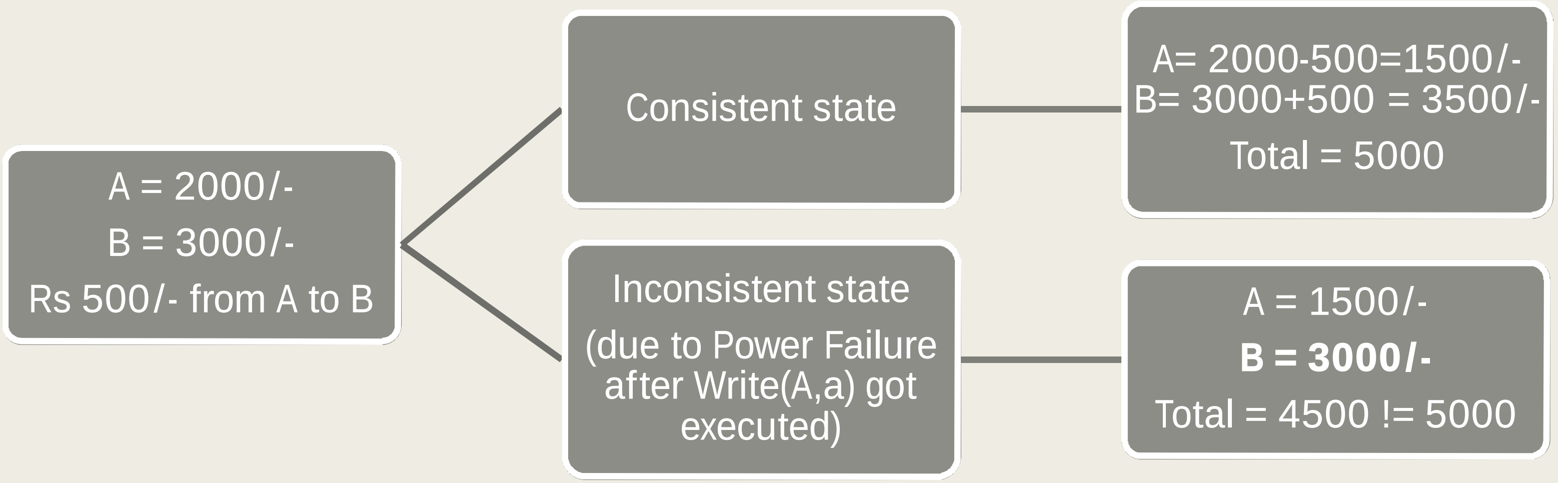
A – 500

Write(A,a)

Read(B,b)

B = b + 500

Write(B,b)



Cont'd..

- **<u>Consistency </u>(Correctness):**

Ensures that if the database was in a consistent state before the start of the transaction, then or termination, the database will also be in a consistent state.

Sum(A,B) before transaction = Sum(A,B) after transaction

Cont'd..

- **<u>Isolation:</u>**

Indicates that the actions performed by a transaction will be isolated or hidden from outside the transaction until it terminates.

Transaction T1	Transaction T2
R(A) A – 500 W(A)	
	R(A) R(B) Print(a,b)→4,500
R(B) B = B + 500 W(B)	

Initially
A = 1500
B = 3000

- **<u>Durability</u>(Commit):**

All updates done by a Transaction must become permanent (or)

Ensures that the commit action of a transaction on its termination will be reflected in the table

Concurrent Execution:

- It implies interleaving execution of operations of a transaction.
- Benefits

i. Helps in reducing waiting time.

ii. Improved throughput and resource utilization.

Schedule:

It represents the order in which instructions of a transaction are executed

T1 → T2 → T3 ⟹ schedule

Problems with Concurrent Execution

- Lost update Problem(W – W conflict)

Occurs when two transactions that accesses the same database items have their operations interleaved in a way that makes the value of the database item incorrect.

Update is lost

Transaction (T1)	Transaction(T2)
R(A) → 1000 A = A -50 → 950	
	R(A) → 1000 A = A + 100 → 1100
W(A) → 950	
	W(A) → 1100

Actual value: 1000 – 50 = 950
950 + 100 = 1050

Cont’d:

- **Temporary update(Dirty Read) Problem [W – R Conflict]**

Occurs when one transaction updates a database item and then the transaction fails, but its update is read by some other transaction.

A = 100

Transaction T1	Transaction T2
R(A) → 100 A = A + 20 → 120 W(A)	
	R(A) → 120 A = A + 10 → 130 W(A) → 130
R(B)	
	Commit (130) (110)

Updated value

Power failure Roll Back

Cont'd:

- **Unrepeatable Read[W – R conflict]**

If a transaction 'Ti' reads an item value twice and the item is changed by another transaction 'Tj' in between the two read operation. Hence 'Ti' receives different values for its two Read operation of the same item.

A = 1000

Transaction T1	Transaction T2
R(A) → 1000	
	R(A) A = A + 2000, A = A – 2000 W(A)
R(A) → 3000	

Cont’d:

- Incorrect Summary Problem:

If one transaction is calculating an aggregate summary function on a no. of records, the aggregate function may calculate some values before they are updated and others after they are updated – results in incorrect summary.

A = 100, Y = 200

Transaction T1	Transaction T2
	Sum = 0 R(A) Sum = Sum + A , Sum = 100 R(Y) Sum = Sum + Y, Sum = 300
R(Y) → 200 Y = Y +100 → 300	

Schedule

- A schedule ‘s’ of n transactions T1,T2,.........., Tn is an ordering of operations of the transactions in chronological order.

Ti(x → y operation)Tj
In any schedule Ti(x) → Tj(y)

T1	T2
R(X)	
	W(X)

T1 → T2

When several transactions are executing concurrently then the order of execution of various instructions is known as Schedule.

Types of Schedule

- **Serial Schedule:**

Does not interleave the actions of any operations of different transactions.

Always ensure a Consistent state.

INITIALLY A =100

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

T1 → T2
T1: A = 100 + 50 =150
T2: A = 150 + 100 = 250

T2 → T1
T2: A = 100 + 100 = 200
T1: A = 200 + 50 = 250

Non - Serial

T1	T2
R(A) A = A + 50	
. . .	R(A) A = A + 100
W(A) → 150	. . .
	W(A) → 200

Cont’d:

- **Complete Schedule:**

If the last operation of each transaction is either abort or commit.

T1	T2
R(A)	
	R(A)
W(A)	
COMMIT	
	W(A) ABORT

COMPLETE SCHEDULE

Cont’d:

- **Recoverable Schedule:**

Is one where for each pair of transactions (Ti, Tj), such that Tj reads a data item that was previously written by Ti, then the commit operation of Ti should appear before commit operation of Tj.

T1	T2
R(a) A + 50 W(A)	
	R(A) A + 100 W(A)
R(B) W(B) COMMIT	

ROLL BACK

failure
ROLL
BACK

Cont'd..

- **Cascade less Schedule:**

Is one where for each pair of transaction(Ti,Tj) such that Tj reads a data item written by Ti, then the commit operation of Ti should appear the read operation of Tj.

Cascading rollback

T1	T2	T3
R(A) W(A)		
	R(A) W(A)	
		R(A) W(A)
COMMIT		
	COMMIT	
		COMMIT

fail

T1	T2	T3
R(A) W(A) COMMIT		
	R(A) W(A) COMMIT	
		R(A) W(A) COMMIT

Cont’d:

- **<u>Strict Schedule:</u>**

If a value written by a transaction cannot be read or overwritten by another transaction until the transaction is either aborted or committed.

T1	T2
R(A) W(A) COMMIT	
	W(A) R(A)

Every Strict Schedule is both Recoverable and cascade less.

Serializability

- A schedule ‘s’ of ‘n’ transaction is serializable if it is equivalent to some serial schedule of the same ‘n’ transactions.
- **Conflict Serializable:**

If it is conflict equivalent to serial schedule

R1A→ W2A
W1A → R2A
W2A

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

R1A → W2A

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

$S1 \overset{C}{=} S2$

Test for Conflict Serializability

- Precedence graph is used
- Let ‘s’ be a schedule, construct a directed graph known as precedence graph.
- Graph consists of a pair of G = (V,E) where V: a set of verties, E: a set of edges
- Algorithm for creating graph:

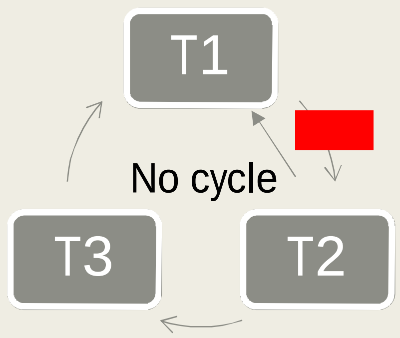
i. Create a node for each transaction

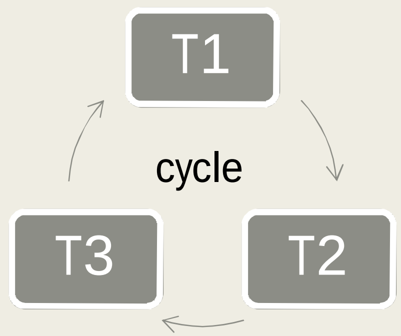
ii. A directed edge, Ti → Tj, if Tj reads a value of an item written by Ti.

iii. Directed edge Ti → Tj, if Tj writes a value into item after it has been read by Ti.

iv. Directed edge, Ti → Tj, if Tj write after Ti write.

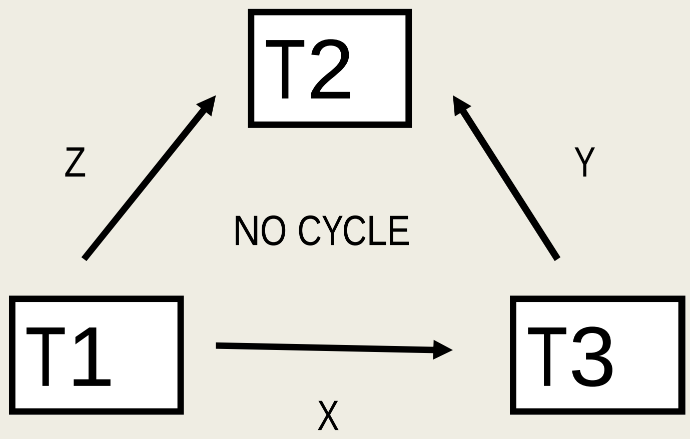
A schedule is conflict serializable if and only if precedence graph is acyclic





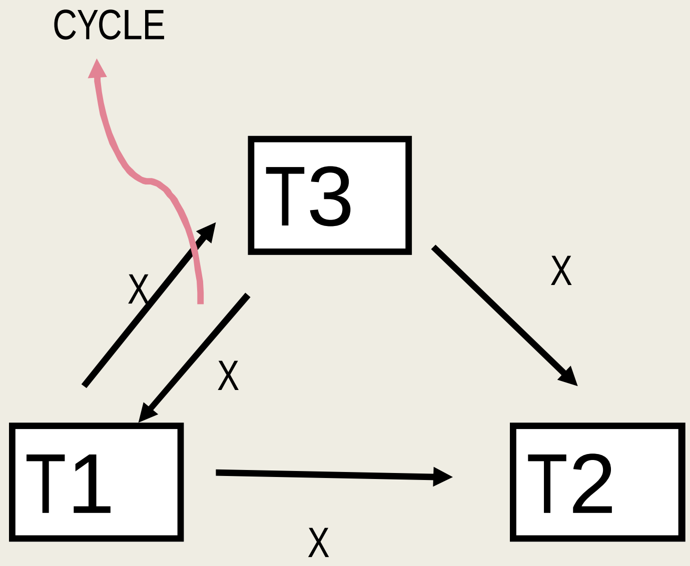
Check for Conflict Serializable

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



Check for conflict Serializability

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



Non conflict serializable

View Serializability

- Two schedules ‘s’ and ‘s1’ are view equivalent if the following conditions are met:

i. For each data item Q, if Ti reads an initial value of Q in schedule S, then Ti must in s1 also reads an initial value of Q

ii. If Ti executes reads Q in s, and that value was produced by Tj(if any), then Ti must in schedule s1 also reads the value of Q that was produced by Tj.

iii. For each data item Q, the transaction that perform the final write(Q) operation in schedule s must also perform the final write(Q) in schedule s1.

A schedule is view serializable if it is view equivalent to a serial schedule

Example of View Serializability

s1

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

s2

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

s3

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

$S1 \overset{v}{=} s2$

$S1 \overset{v}{\neq} s3$

■ **<u>Conflict Operations</u>**: operations are said to be conflicting if

i. Belong to different transactions

ii. Access to same database item ‘A’.

iii. Atleast one of them is a write operation → R W, W R, W W

R R – NOT CONFLICT

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

S1:
R(A) → 100
A = A + 10 → 110
W(A)

S2:
R(A) → 100
A * 1.1 → 110
W(A)

- **<u>Equivalent Schedule:</u>** Two schedules ‘s1’and ‘s2’ are said to be equivalent schedule if they produce the same final database state.

  *) **Result equivalent schedule:** Produce same final DB state for same initial values of data.

- **Conflict Equivalent:**

Two schedules are said to be conflict equivalent if ll conflicting operations in both the schedule must be executed in the same order.

Question1) check for conflict equivalent
S1: R1(A), R2(B), W1(A), W2(B)
S2: R2(B), R1(A), W2(B), W1(A)

S1

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

S2

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

S1 $\overset{C}{=}$ S2

Question2) check for conflict equivalent
S1: R1(A), W1(A), R2(B), W2(B), R1(B)
S2: R1(A), W1(A), R1(B), R2(B), W2(B)

S1 (W2B → R1B)

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

S2 (R1B → W2B)

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

S1 $\overset{C}{\neq}$ S2