Date _____ Day _____

TRANSACTIONS "

" Logical unit of work that must be entirely completed or aborted."

→ If the database operations in a transaction donot update the database but only retrieve data, then it is called read-only transactions.

→ If there is an update operation then, read-write transaction.

→ Database users:

Single-user DBMS : Home computer

Multi-user DBMS : Airline reservation system

Multiprogramming : Concurrent execution. Interleaved.

→ Read & Write Operation :

read_item (x);

1. Find the address of the disk block that contains item X.
2. Copy that disk block into a buffer in main memory (if that block is not already in some main mem buffer). The size of the buffer is same as the disk block size.
3. Copy item X from the buffer to the program variable named X.

Date _____ Day _____

write_item (x)

1. Find the address of the disk block that contains item item X.
2. Copy that disk block into a buffer in main mem (if that disk block is not already in some main memory buffer).
3. Copy item X from the program variable named X into its correct location in the buffer.
4. Store the update disk block from buffer back to disk (either immediately or at some later point in time).

"CONCURRENCY CONTROL"

- Problems in Concurrency Of Transactions

Temporary Update Problem

- Dirty Read Problem: (Uncommitted value ko read krna)

T_1	T_2
Read (X)	
Write (X)	
⋮	Read(X)
	Comitt
rollback	

agr koi transaction value read krta hai jo ki database ki value na ho ki kisi uncomitted transcation ki thru likhi hai value ho usko dirty read kehte hai.

jab T_1 roll back krega to x ki purani value retain ho jayegi or T_2 x ki esi value read kr chuka hai jo DB mein hi nhi hai

→ isne X read to krlia lekin stupid age T_2 X ko change krde q ki usne abhi tak comitt nhi kraa hai.

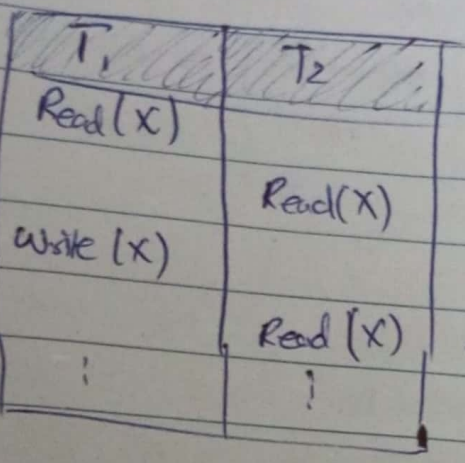
Page No.

Day

Solution: jo transaction dirty read kre wo doosri trans k read hi commit kre.

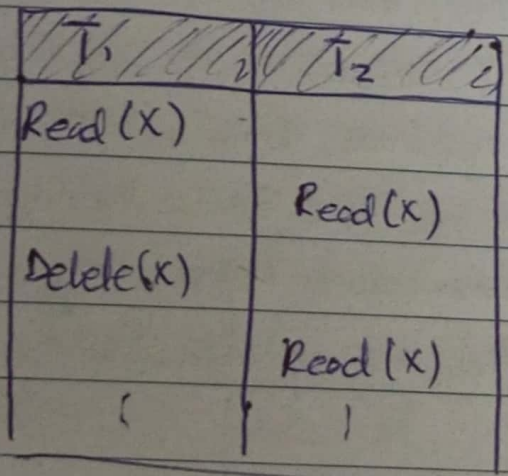
- Unrepeatable Read Problem: → jab ek variable ko do baar read krein or different values aayein.

DB x=10



T2 ne pehle x ki value 10 read ki or kuch der baad jab doobara read ki to wo 11 hogai

- Phantom Read Problem:

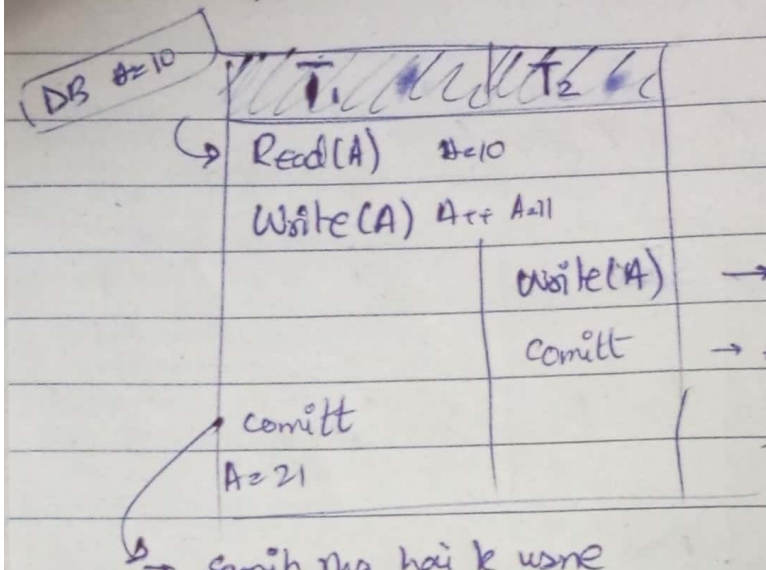


jab aap kisi variable ko read kro, or phir doobara is liye read nhi kar paao q k database ka structure change hogaya ho.

If a transaction write on a variable before ... blind write.

Date _____ Day _____

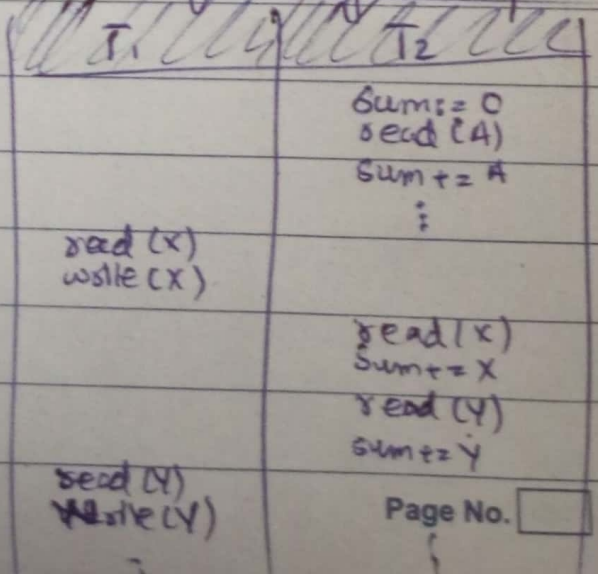
- Lost Update Problem (write - write conflict)



→ Samjh nhi hai k usne A ko sare kaa leken T_2 ne A ko overwrite kadia hai

- Incorrect Summary Problem:

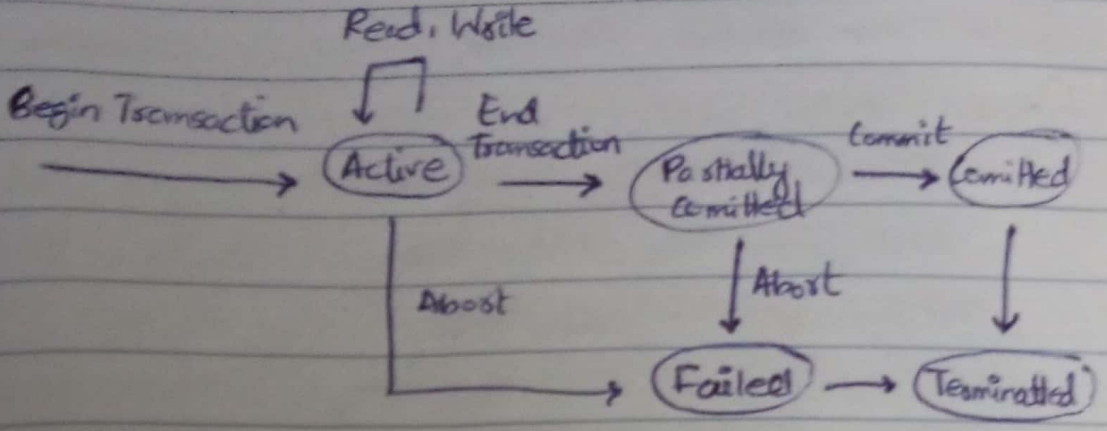
→ If one transaction is calculating an aggregate summary func on several database items while other trans are updating some of these items the aggregate func may calc some values before they are updated and others after they are updated.



Page No.

Date _____ Day _____

States of Transaction:



System Log:

- System log keeps track of transaction operations
- Sequential, append-only file
- Not affected by failure
- Log file backed up periodically.

LOG RECORDS:

- [start_transaction T]
- [write_item, T, X, old_value, new_value]
  Indicates that transaction T has changed value of DB item X
- [read_item, T, X]
- [commit, T]
- [abort, T]

Date _____ Day _____

Buffer Replacement Policies:

→ In DBMS, the buffer rep policies determine how data pages are loaded into and removed from the buffer cache / buffer pool.

↳ It is a part of memory used to temporarily store pages read from DB's disk storage.

↳ Accessing data from buffer is much faster.

↳ The cache will hold the disk pages currently being processed.

(1) Domain Seperation Method

→ Efficiently handles disk pages types by segregating buffers.

→ Buffer are divide into 3 domains

i- Index Pages

ii- Data file pages

iii- Log file pages

Each domain allocated fix no of buffers.

→ Buffers within a domain are replaced when full, without impacting other domains.

(2) Hot Set Method

→ Optimized for queries with repeated page scans

→ Identifies a hot set of pages frequently accessed during query execution & avoids replacement of

Date _____ Day _____

→ Reduce I/O oper for sepetetive queries.

② DBMIN Method:

→ Uses predective model to manage buffer replacement for specific DB opes.
→ Predicts the pattern of page seferences for operations
→ Calculates the locality set (set of pages likely to be accessed) using the Query Locality Set Model (QLSM)
→ Refetches and retains the locality set to prevent page faults.

ACIDS Properties:

① Atomicity: All operations of a trans must be completed if not, the transaction is aborted.

Consistency: Before trans start and after tran end the sum of total money must be same.

Isolation: Data used during transaction cannot be used by second trans until the 1st is completed

Durability: Ensures that once trans is comitted, it cannot be undone or lost. (All changes in DB should be permanent)

Serializability: Ensures that the schedule for the concurrent execution of several trans should yield consistent results.

Day

- **Levels Of Isolation:**
  - Level 0 → doesn't overwrite dirty reads of higher level trans
  - Level 1 → has no lost updates
  - Level 2 → has no lost updates & no dirty reads
  - Level 3 → (true) isolation → has repeatable reads in addition to level 2 properties

- **Shedule:** A shedule (or history) S of n transaction $T_1, T_2, \ldots T_n$ is an ordering of the operation of the transaction.

- **Conflicting Conditions:**
  → Two conflicting operations in a shedule if satisfy all three of the following conditions are said to be in conflict:
    - ↳ Oper. belong to different transaction
    - ↳ Oper. access the same item X
    - ↳ At least one of the operation is a write_item()

  → Two operations conflict if changing their order results in a different outcome.

S_1		
T_1	T_2	
R(A)		No two operations at same time.
	R(B)	Trans will work in interleafe fashion.
	W(B)	
R(B)		
	R(A)	
W(B)		
	W(A)	

Page No.

Day

- SERIAL SCHEDULE:

T_1	T_2
	R(B)
	W(B)
	R(A)
	W(A)
R(A)	
W(A)	
R(B)	
W(B)	

→ Serial schedule gives consistent results.

→ Total no. of combination for serial schedule $n!$ (n = total no. of transactions)

- NON SERIAL SCHEDULE:

→ This is efficient but doesnt provide consistent results due to conflicts

T_1	T_2
R(A)	
W(A)	
	R(B)
	W(B)
R(B)	
	R(A)
W(B)	
	W(A)

→ Total no. of combinations

$$\frac{(n_1 + n_2 + \dots n_n)!}{n_1! \cdot n_2! \dots n_n!} - n!$$

→ For making non-serial shedule consistent we will try to convert non-serial → serial.

→ For this purpose we will swap the operations of two transaction only if there are non-conflicting.

→ Conflict Serializability: When a non-serial schedule is converted to a serial schedule by swapping of non-conflicting operations.

Conflict serializability shedules are guaranteed to be consistent.

Page No.

Date

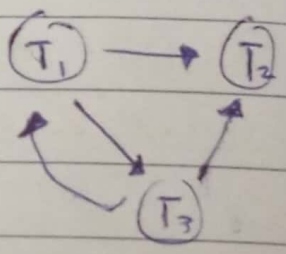
• For Checking if Shedule is Conflict Serializable?

(1) First we will draw 3 nodes (as we have 3 transactions)

(2) Now we will see from top to bottom and check for conflicting operations.

(3) As jese $R(X)_{T_1}$ ko lenge or baaqi neeche ki operations se conflict check krenge. So the conflict will be $W(X)_{T_3}$. This means that T_1 should finish before T_3. And we will draw an edge from $T_1 \rightarrow T_3$.

T_1	T_2	T_3
$R(X)$		
		$R(Z)$
		$W(Z)$
	$R(Y)$	
$R(Y)$		
	$W(Y)$	
		$W(X)$
	$W(Z)$	
$W(X)$		



▲ Precedence Graph

- Then select $R(Z)_{T_3}$ it conflicts from $W(Z)_{T_2}$. So an edge $T_3 \rightarrow T_2$
- Then $W(Z)_{T_3}$ conflict with $W(Z)_{T_2}$ no need to show multiple edges in same direc.
- Then $R(Y)_{T_2}$ no conflict
- Then $R(Y)_{T_1}$ conflict with $W(Y)_{T_2}$ edge $T_1 \rightarrow T_2$. (This indicates that T_1 must finish before T_2)
- Then $W(Y)_{T_2}$ no conflict
- Then $W(X)_{T_3}$ conflict with $W(X)_{T_1}$ $T_3 \rightarrow T_1$

Page No.

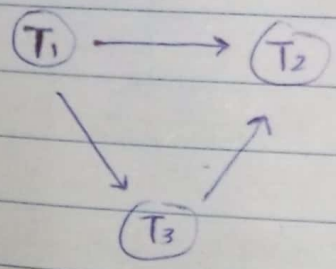
Day

→ Noow stop when we get a cycle. Because there are two edges $T_1 \rightarrow T_3$ & $T_3 \rightarrow T_2$. which can't be serialized.

→ Or we stop when all oper finishes.

Q#02:

T_1	T_2	T_3
R(x)		
	R(y)	
		R(y)
	w(y)	
w(x)		
		w(x)
	R(x)	
	w(x)	



As no cycle so it is conflict serializable!

- Now Find the order of serial scheduling

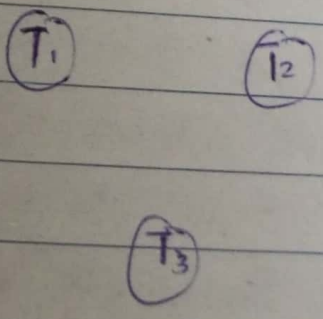
↳ Check for the vertex with no incomming edge. As they will be 1st to execute. Here it is T_1

↳ And the one having most incomming edge will be last to execute. Here is is T_2

↳ So order becomes $T_1 \rightarrow T_3 \rightarrow T_2$

Q#03:

T_1	T_2	T_3
R(a)		
	R(b)	
		R(c)
		w(c)
	w(b)	
w(a)		



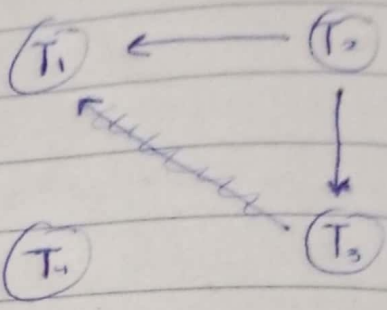
It is conflict serializable and can be arranged in any order as no edges.

Page No.

Date Day

Q#04 :

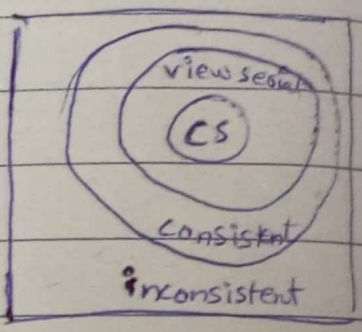
T_1	T_2	T_3	T_4
	R(x)	W(x)	
		C	
W(x)			
C	W(y)		
	R(z)		
	C		R(x)
			R(y)
			C



$T_2 \rightarrow T_1 \rightarrow T_3 \rightarrow T_4$

- Here $W(X)_{T_3}$ will not conflict with $W(X)_{T_1}$ or $R(X)_{T_4}$ because T_3 has been comitted. Therefore after comit it has no existence.

⇨ VIEW SERIALIZABILITY :



→ It is also way to find consistent schedule.

→ If a schedule is Conflic Seslizable then it is also view ser→.

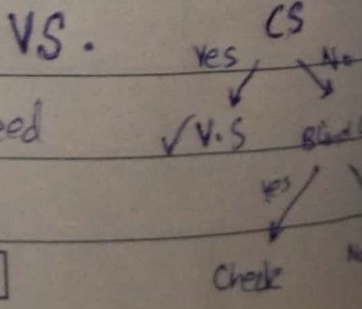
→ To find out that a Schedule is view serializable or not is NP Complete (complex) Problem

→ So 1st we find CS then it is automatically VS.

→ If a schedule is NOT CS then if it is a VS then it should have a "blind write".

→ If no blind write then not VS.

→ If it has blind write then we need to check.



Date _____ Day _____

How To check?

→ 1st we need to have all possible combinations for ~~a~~ serial schedule for S.

→ ~~Agar koi kisi phir hamare~~ Like if we have 3 Transaction T_1 T_2 T_3 then there can be 6 combination for serial schedule.

→ Phir ek ek koke hum non-Serial wale ki serial ~~se~~ ke sath view equivalent check krna hai.

If a non-serial is view equivalent to a serial then the non-serial schedule is view serializable.

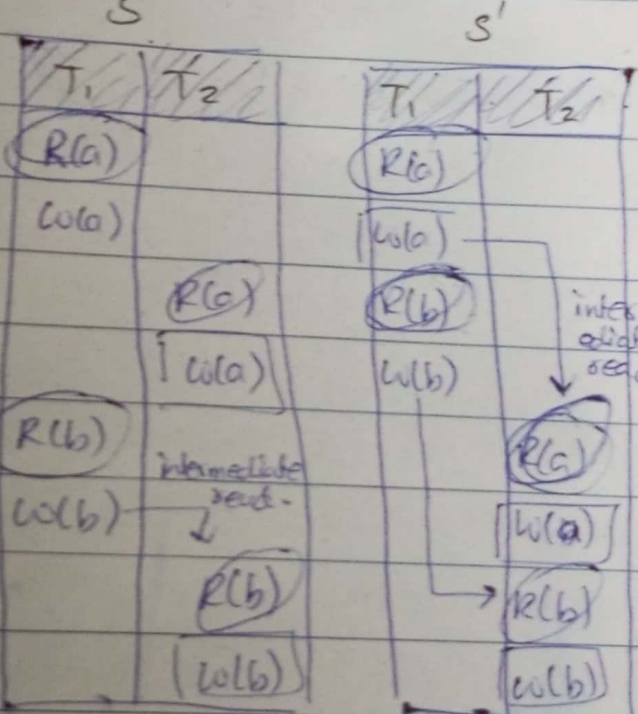
→ There are 3 conditions for view equivalency.

↳ We will check initial reads for every variable. If in schedule S T_1 has initial read 'a' then in S' only T_1 should read it 1st ~~same for B~~.

↳ Same rule for initial write.

↳ Agr schedule S mein koi transaction T_i read krta kisi T_j transaction ki likhi hui value ko to S' mein bhi yehi hona chahiye. (Only for intermediate reads)

S mein $W(b)_{T_1}$ ki value $R(b)_{T_2}$ krta hai to S' mein bhi esa hi hai.



Page No. ☐

Date 5 Day

T_1 T_2 T_3 ✓
T_1 T_3 T_2
T_2 T_1 T_3
T_2 T_3 T_1
T_3 T_1 T_2
T_3 T_2 T_1
} ✗ no need to check.

T_1	T_2	T_3
R(a)		
	w(a)	
w(a)		
		w(a)

$T_1 \rightleftarrows T_2$ cycle ⇒ not CS.

$T_1 \rightarrow T_3$, $T_2 \rightarrow T_3$

There is also two blind write
$w(a)_{T_2}$ & $w(a)_{T_3}$

1st serial S'

T_1	T_2	T_3
R(a)		
w(a)		
	w(a)	
		w(a)

→ initial read ✓
→ final write ✓
→ intermediate read ✓ (no intermediate reads)

khud bnaya hai ye

So S is view equivalent to S' and as S' is serial so S is also serial.

Date _____ Day _____

Recoverable Schedules :

→ CS ya VS hone k baad bhi agr system failure koi ho to bhi DB inconsistent state mein pohonch sakta hai.

→ Os ese schedule ko irrecoverable shedule bolte hai.

→ Os recoverable wo jo in-between failure ko bhi handle krenge and guarantees consistency of DB.

→ If there is no dirty read then schedule is always recoverable.

DB → A = 10 LB = 10

T_1	T_2
R(A)	
A = A + 10	
W(A)	
	R(A)
	A = A - 5
	W(A)
	Comitt
failure	
Comitt	

Irrecoverable

→ ~~Isi exam~~ Is eg mein T_2 jo A ki value read krra hai wo T_1 ki likhi hoi hai to jab tak T_1 comitt nhi krleta T_2 ko bhi comitt nhi krna chahiye

T_1	T_2
:	
	:
Comitt	
	Comitt

→ Only changed the ordes of comitt.

→ Jis order mein dirty read hoa hai usi order mein comitt bhi hoga.

Means that T_2 ne T_1 ki ~~value~~ copy se read ~~read~~ kia hai to wo comitt bhi uske baad hi krega

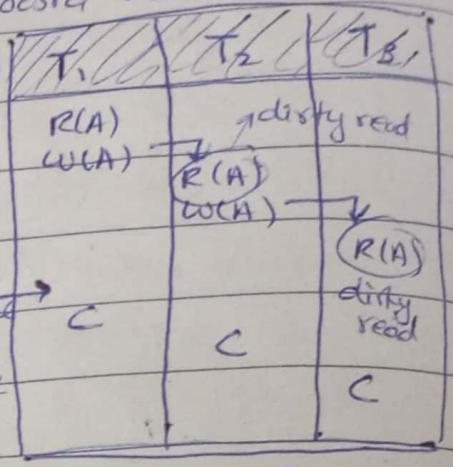
Page No.

Date _____ Day _____

▷ CASCADLESS SCHEDULE :

→ jab ek transaction ko dekh kr doosra roll back hoe to cascading roll back (but this is not efficient)

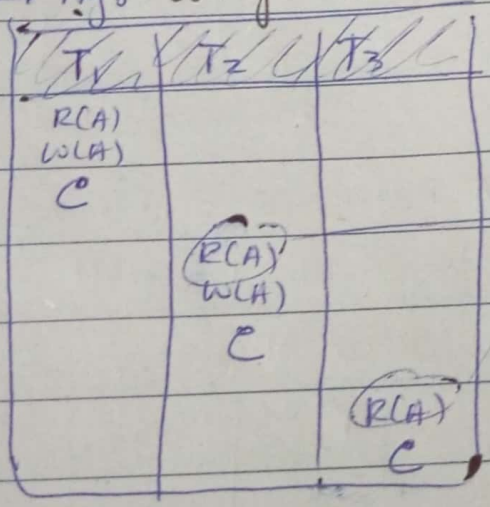
→ A schedule is cascadeless if it donot have cascading roll backs.



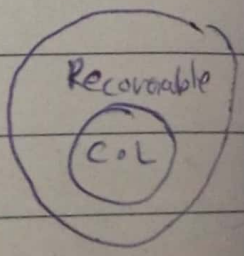
if there is failure then all of them roll back bcz of dirty read

→ degree of concurrency decr.

→ Agr dirty read hai to cascading roll back hoga hi.

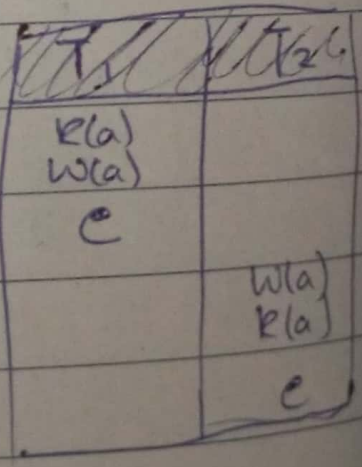


→ abye dirty read nhi bcz it is reading from a comitted value directly from DB.



▷ STRICT SCHEDULE :

Agr koi transaction kisi data item par write operation ek baar perform krde to jabtak we comitt nhi kreg tab tak us data item par koi doosri trans read/write nhi kr sakti.



Strict ka matlab zoori nhi k serial hi hoga!

Date _____ Day _____

T_1	T_2
R(a)	
W(a)	R(b)
C	W(b)
	R(a)
	C

← This is also strict but not serial.