

Barehjan Sirz (Book chap - 3, 4)

- ① Integrity constraints :
 - ① Primary key
 - ② Foreign key
 - ③ not null [मात्र रोकने वाले attribute]
 - ④ value predefined not null हैं]
 - ⑤ check : for checking purpose when insert query is executed.

For example: IMDB_Rating >= 5.0

Create Table Movie (

(attribute) → (Type) (SI - QD) ;

IMDB_Rating integer check (IMDB_Rating >= 0.0);

Check (IMDB_Rating >= 0.0);

A एकी attribute एवं domain (set of permitted values) set के रूप माम्

check ('genre' in ('Action', 'Fiction', - - -));

④ unique (It is not a key, it is a constraint)

যেটা এক value হিসাবে allow করব না)

for b. Create Table Movie (

unique (A1, A2); Attribute

যখন আমরা sure হিসেবে

করে primary key করব,

Unique → অব্যাপ্ত candidate

key এর মতো রথক selection

For ex: NID, phone no

④ Transaction : (Chap - 17) \rightarrow Collection of operations.

Real Life is a set of operations হচ্ছে আরেকে।

(একটা single query

; execute করে) considering a single

unit which is

indivisible / A

single unit of task

503f

Abul

101/2) mismatch P.P. student

Bulbul

Purpose: Database এর data item প্রোকে Access & update করবে but single unit task হিসেবে consider করবে,

To access Data items:

2 operations

(i) $\text{read}(D)$: Data item এর উপর read operation buffer

from DB Transfer(D) to a Variable(D) in Main Memory

(ii) $\text{write}(D)$: Variable(D) to a Main Memory

* Update করবে গরের বাটন Write করা

From to Transfer(D)

DB

From Variable(D)

in Main Memory

Buffer.

TI (Transaction I)

read(A);

$A = A - 101$; $A = ?$

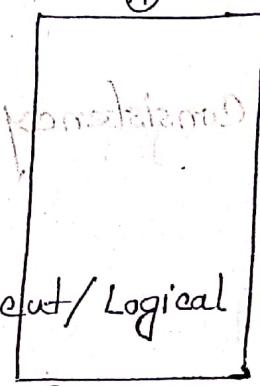
updated value Ram² write(A);

Power cut/Logical error/OS crash

write করা মানে Database

নির্ভুল হওয়া না, কাজ একটা Temporary file এ রাখত।

(A)



Power cut/Logical error/OS crash

read (B);

B := B + 101;

write (B);

Properties of Transaction:

ACID properties : // Board Viva + Job Viva

+ final semester.

A → Atomicity ([Atom টথকে এয়েছে ; Transaction
কে একটা indivisible property

ALL or None

হিসেবে চিন্তা করতে হবে])

operations

Network failure, power cut

problem দূর করায় জন্ম atomicity
property maintain করা এয়েজন।

Write করার পরে একটা

temporary log file এ রাখে।

C → Consistency

: Integrity constraints use করা

যদি

PK, FK, check

Integrity constraints + Application-dependent constraints.

এক application এর code কি develop

কৰছে set কৰে দেয় so
that data insert কৰাৰ গৱে

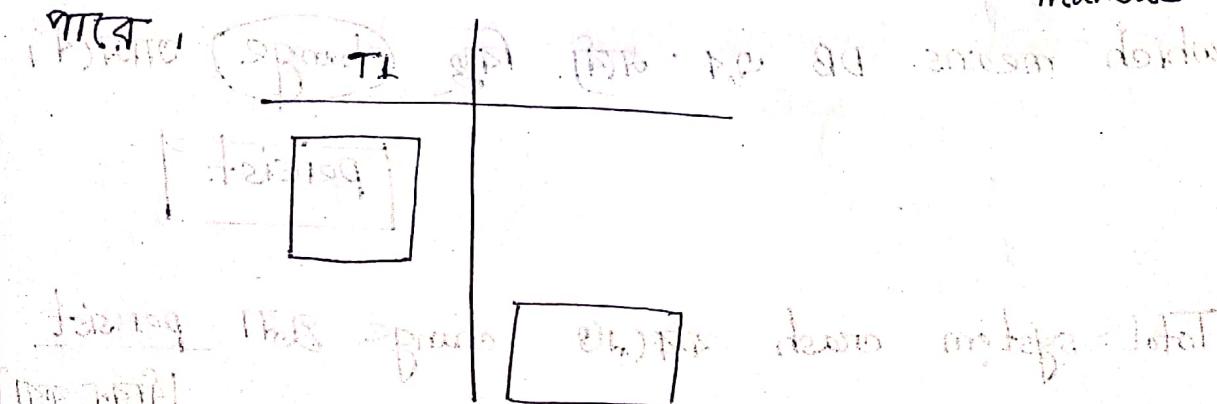
मन Inconsistency create न
करता है।

(DB আসেও যদি consistent হিলে *DB must be consistent at
কিছু operations কার্য পরের the end of each transaction
stage এও যদি consistent হিলে *DB must be consistent at
* একটা Transaction হবে atomic Transaction.

$I \rightarrow$ Isolation (প্রত্যক্ষ ট্রানজেকশন
(transaction isolation এ প্রকরণ)

D → Durability

Isolation $\xrightarrow{\text{Database DB এ একাত্মিক Transaction থাকত}}$



④ Concurrent execution of units for performance improvement

प्राचीन विज्ञान एवं विद्या के सम्बन्ध में अतिरिक्त प्रश्न एवं समस्या आने की स्थिति लागती है।

205 पारा । अतः वे constraints break RCT नियम ।

Q. अवधारणा का अभाव inconsistency

Concurrent execution of multiple transactions

Inconsistency

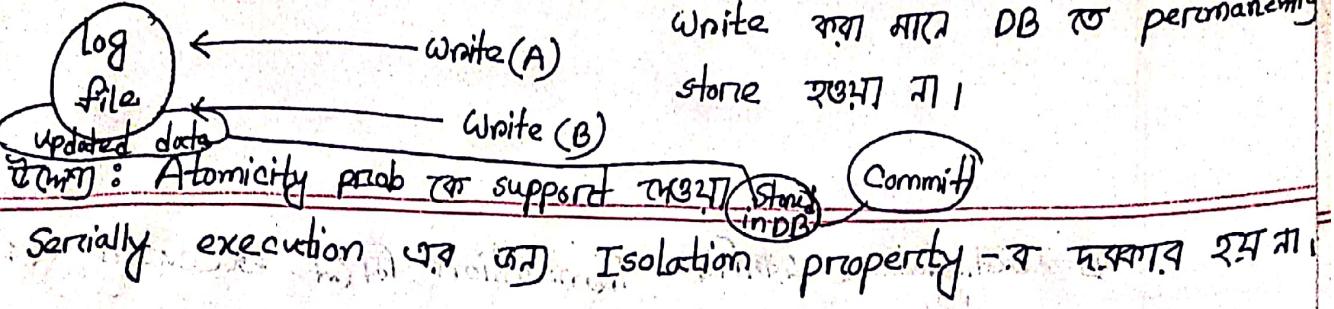
Inconsistency यह ना हो सके कारण अभी एकमात्र रूप
कारबैंड के लिए एक transaction पर अटोम द्वारा one
transaction पर नहीं लाभ लाना कहा जाता है। इसका कारण
कारबैंड (works like serial in execution)

④ Durability : Transaction \longrightarrow successfully completed

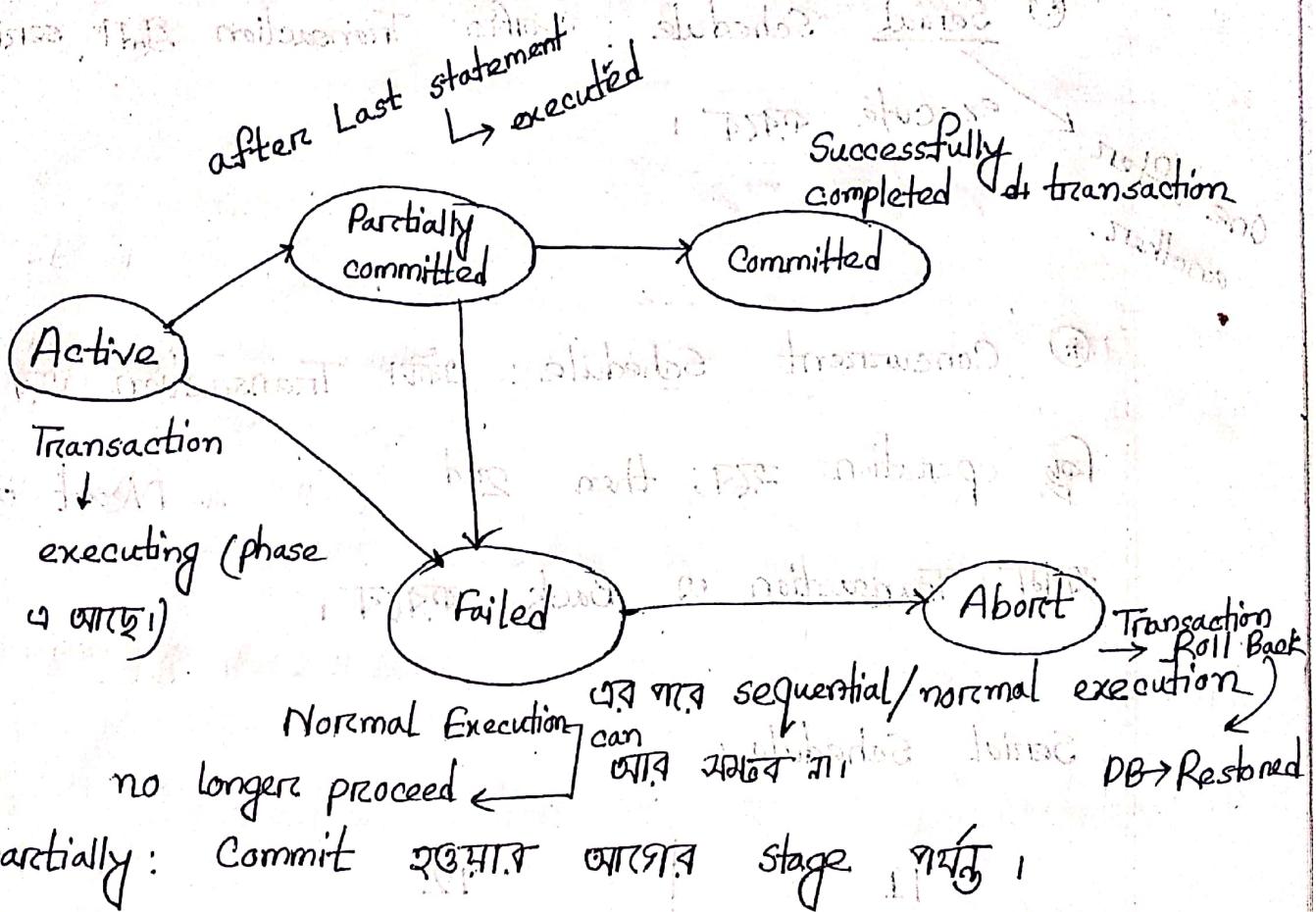
which means DB এবং মাফ্য স্বয়ে change আস্তি।

persist

Total system crash করলেও change এতো persist
(পুরো করা)
করা নামে,



অ্যালগরিদম Transaction এর stages : (অ্যালগরিদম Transaction State diagram : সময়ের অনুসরে এর মধ্যে স্থানান্তর stage এ বিবরণ করবে)



Abort : DB কে তাৰ আগেৰ অবস্থা/ পুরিয়ে দিতে হৈব।

Stage - ৫

(V) পুরিয়ে

(VI) পুরিয়ে

পুরিয়ে + পুরিয়ে = ১

(VII) পুরিয়ে

পুরিয়ে <-- (VIII) পুরিয়ে

পুরিয়ে <-- (IX) পুরিয়ে

④ Schedule : একাধিক Transaction মিলে execution প্রে
একটা sequence.

→ execution sequence of multiple Transactions

* Serial Schedule : একাধিক Transaction এন্টা serially
execute করবে।

one after
another.

* Concurrent Schedule : একটা Transaction প্রে
ক্ষুণ্ণ operation হবে; then 2nd " . Next অসার
তৃতীয় Transaction এ Back করবে।

Serial Schedule :

	T1	T2
Read (A)		
$A := A - 500$		
Write (A) $\rightarrow 4500$		
Read (B)		
$B := B + 500 \rightarrow 3500$		
Write (B) $\rightarrow 3500$		
Commit $\rightarrow DB$ এ সেল finally		
	read (A) $\rightarrow 4500$	
	temp: = $A * 0.1$	

$$A := A - \text{temp} \rightarrow 4500 - 450$$

$$\text{write}(A) \leftarrow 4050$$

$$\text{read}(B) \rightarrow 3500$$

$$B := B + \text{temp} \rightarrow 3500 + 450$$

$$\text{write}(B) \rightarrow 3950$$

Commit

$T_1 >> T_2 \text{ OR } T_2 >> T_1 \longrightarrow \text{Serial Execution.}$

$$\text{Suppose } A = 5000$$

$$B = 3000$$

$$\text{Transaction} \rightarrow A + B = 4050 + 3950$$

$$\text{Previously } A + B = 8000$$

বিশেষ করণের
concurrent execution

পারে।

এর মূল Inconsistency হল

$$5000 \rightarrow (A) \text{ begin}$$

$$(T_1) 1.0 * A = 1000$$

$$1000 - A = .A$$

$$5000 \leftarrow (A) \text{ end}$$

$$5000 \rightarrow (T_2)$$

$$5000 + B = 1000$$

$$5000 \rightarrow (S) \text{ begin}$$

$$5000 + B = 1000$$

$$5000 \rightarrow (Q) \text{ begin}$$

finished

একটা Transaction কাস্ট হবে, then Commit হবে, এরপর next transaction কাস্ট হয়ে কাস্ট হবে & commit হবে \rightarrow Serial Schedule. ($T_1 \gg T_2, T_2 \gg T_1$)

Problem in Serial Schedule: T_1 এর মধ্যে 100 operations
finished T_2 " " 3 "

Waiting time অনেক বেশি + resource প্লাট

Concurrent Schedule:

$T_1, T_2 \rightarrow$ Active

Schedule - 3.2

10% (B)

(A)

difference bet"

parallel: দুইজন একে

মাথে কান বায়ছে

concurrent: দুইজন "

মাথে কান করছে না

but একাধিক Transaction
at a time - ৭

active stage

৭ মিনিট,

T_1	T_2
read(A) $A := A - 500$ write(A) $\rightarrow 4500$ log file	500 \rightarrow (B) read(A) $\leftarrow 4500$ $temp := A * 0.1 (450)$ $A := A - temp$ write(A) $\rightarrow 4050$
read(B) $\leftarrow 3000$ $B := B + 500$ write(B) $\leftarrow 3500$ Commit	

read (B) $\leftarrow 3500$
 $B := B + \text{temp}$ 450
 write (B) $\rightarrow 3950$
 Commit
 *updated value হলো preventing inconsistency &
 Log file -> Log file
 মানবে,

Difference betn
Write & Commit,

mainly
atomicity

secure হবে। (For
সকল নির্বাচন, নির্বাচন করা)

transaction এর
সম্পর্কে operation successful
-by execute হয়েছে।

log file বাধা করে যেন atomicity
property কে support দিয়ে।

There is a chance

A $\leftarrow 5000$
B $\leftarrow 3000$

সম্ভূত Schedule শেষ হওয়ার পরে

A + B

4050 + 3950

= 8000 (Luckily মিল

হয়ে but নাও

মিলতে পারত

0053 ... 8

Cause Operating System

কোন transaction এর কয়টা operation
execute হওয়ার পরে CPU access
another transaction কে জাপনা

হচ্ছে দিবে:

$A \leftarrow 3000$

$B \leftarrow 5000$

T1

T2

read (A)

$$A := A - 500$$

update
2500

main memory
cause log file

DB cause

value T2

main memory
write (A) $\leftarrow 2500$

read (B) $\rightarrow 5000$

$$B := B + 500$$

write (B) $\rightarrow 5500$

Commit

B + A

5500 + 2500

mid

A $\rightarrow 2500$

B $\rightarrow 5500$

7800

read (A)

3000 (DB এর পাস)
temp := A * 0.1 (300)

$$A := A - temp$$

write (A) $\rightarrow 2700 \rightarrow$ log file

read (B) $\leftarrow 5000$

updated value

$$B := B + temp (5000 + 300)$$

$$\text{write (B)} = 5300$$

Commit

final value (concurrency to

মানের রেখা)

A এর value update হয়েছে যিনু write করার আগেই read
হয়েছে which means updated value টা পাইত না। Main

problem →

read (D)
write (D)

এবং sequence change হওয়ার জন্য

কামনা হলু

Conflict হলু যখন same data মিল কাত হওয়া

৩ 4th এ টা Combinations :

T1	T2
read(D)	
	read(D)

T1	T2
	read(D)
	read(D)

OS change হওয়ার
T2 এ আগে read
করবে then T1 এ
read করবে।

T1	T2
read(D)	
	write(D)

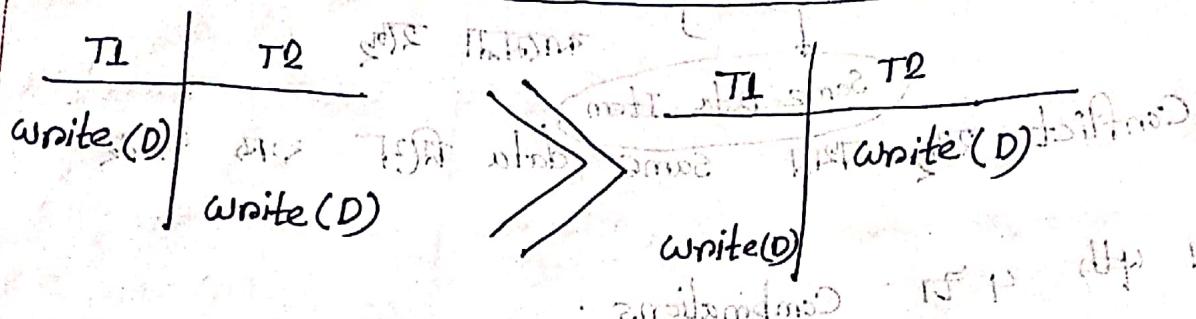
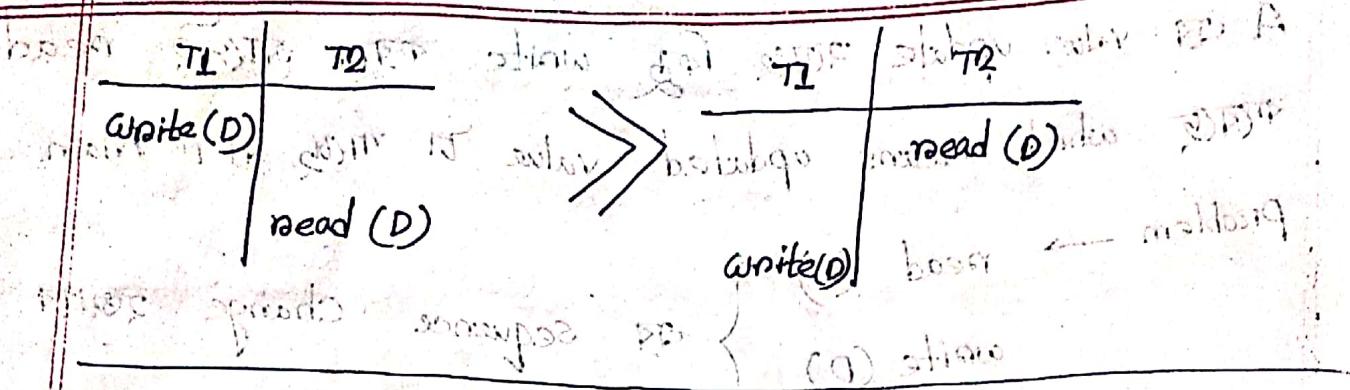
T1	T2
	read(D)
	write(D)

(Two instructions i. & ii. are said to be conflicting :

i. they must belong to different transactions

ii. they must operate on same data values.

iii. there must be atleast (one of them) write operation



PL/SQL (Procedural Language /
(ORACLE) Structured Language)

SQL - input → একাধিক Table
(Q) output → Table

SQL → declarative
what to do
how to do x

But there is a problem → মাঝে আমরা now এর
কোন একটা particular single variable নিয়ে কাজ

শর্করাতে স্টেটমেন্ট (Looping, Branching, Declaring variables, conditional statements - IF/ELSE) SQL এ আছে না।

Figure → Event handling methods আছে অনেক।

function (math for you) handle and handle small in

Assertion through হবে মান transaction এর basic rules এতে
মন violate না হয়,

Step by step execution depending on the type of
procedure → PL

What are the inherent benefits of it ?

error handle করা যায়, input/output variable নিয়ে কাজ দ্বা
যায়,

Structure of a Block :

Declaration part:

variable, cursor declare করব,

(varchar, char)

int, num

Begin : This is the executable section

exception: error handling → // (Optional Part)

end : mandatory.

Block (1) : → declare part

Allowed command : Select, insert, update, delete

DML

Data definition statements like alter . . .

PL/SQL is not case sensitive.

Variables and Types : (Table এর data type এর মতৰ)

DECLARE

 price NUMBER ;

 myBeen VARCHAR (20) ;

We can also declare boolean variables.

④ %Type

Program run কৰাৰ কৰ্য . run দিতে হবে,

VARIABLE

x Number NUMBER

SQL → row by
row করে
করে।

delete

④ message print করা পায় DBMS.OUTPUT.PUT_LINE package

Accessing database করার জন্য SQL ফিল্টার

Select e,f into a,b
(projection)

→ assignment এর মত ব্যবহার করে

⑤ Control Flow:

⑥ Loop:

for loop দুটি ধরণ হল

<loop_body> : PL/SQL

end loop;

প্রথম স্বর্গের PL/SQL এর উদাহরণ দেখো।

⑦ FOR :

FOR < var > IN <start> < finish >

(alihoda)

(base)

initialization

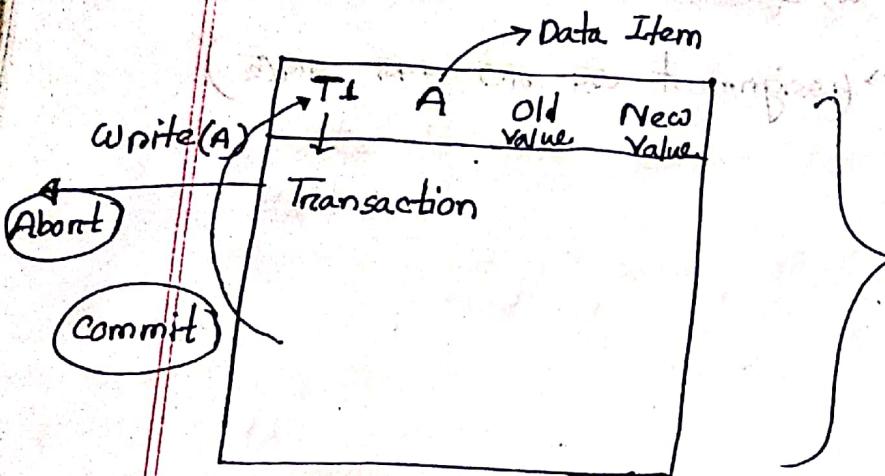
(process)

message, raise_error ; assignments

exit loop;

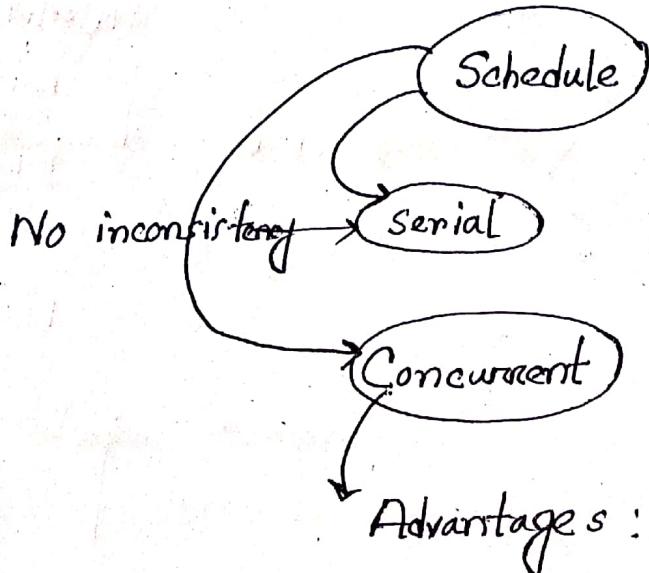
loop condition, loops

* log file use করা হয় atomicity property support দ্বারা
জন্য।



Track রাখা হবে, so
that power cut হলেও যদি
আমের অবস্থান ফিরে পার
which is called Abort

Successfully commit হলে T1 এর change (রাখার
কোনো দরকার নাই)

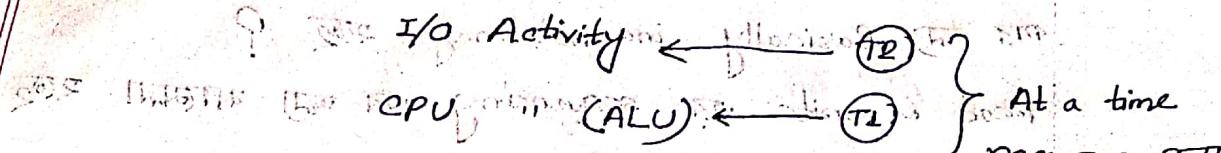


Advantages : T1 এর অনেক operation
T2 " কর (waiting time

* Reduced waiting
time অনেক কম)

Concurrently execute করলে
average waiting time T_1
~~বার্জ মান~~

Operation 2 করতে হয়;



* Resource utilization & Improved
throughput.

(concurrently execute করলে করবে)

pipeline

→ এই concept ব্যবহার করা হচ্ছে এখানে

used হচ্ছে কোনি
idle করে থাকা
নাগার কর,

Drawbacks of Concurrent Schedule:

Schedule - 4.1

আগের ফল :

T_1

read (A)

$A := A - 500$

write (A)

T_2 (A). update

Round Robin scheduling

Schedule - 4.2

(A). update

update

write (A)

read (B)

$B := B + 500$

write (B)

Commit

Round Robin scheduling

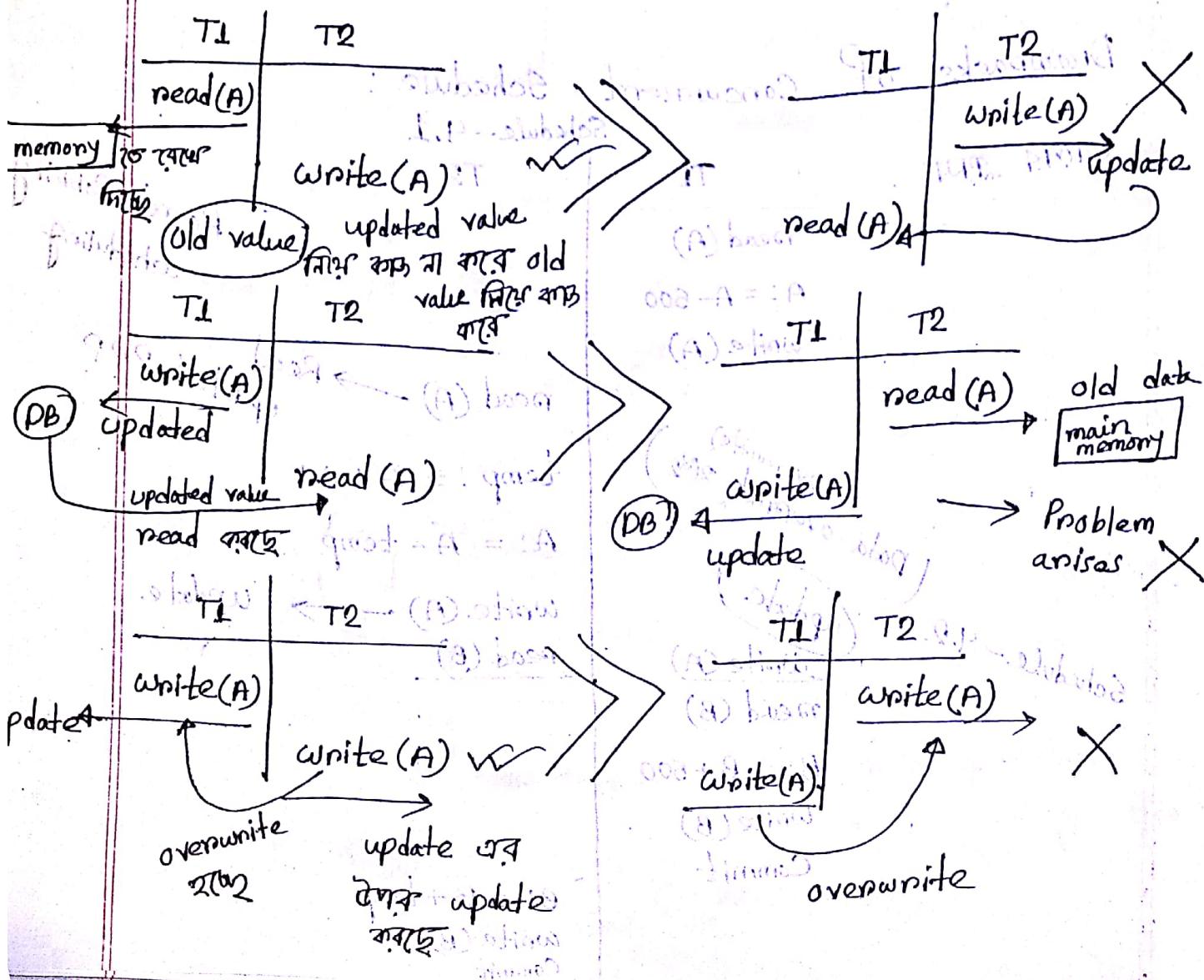
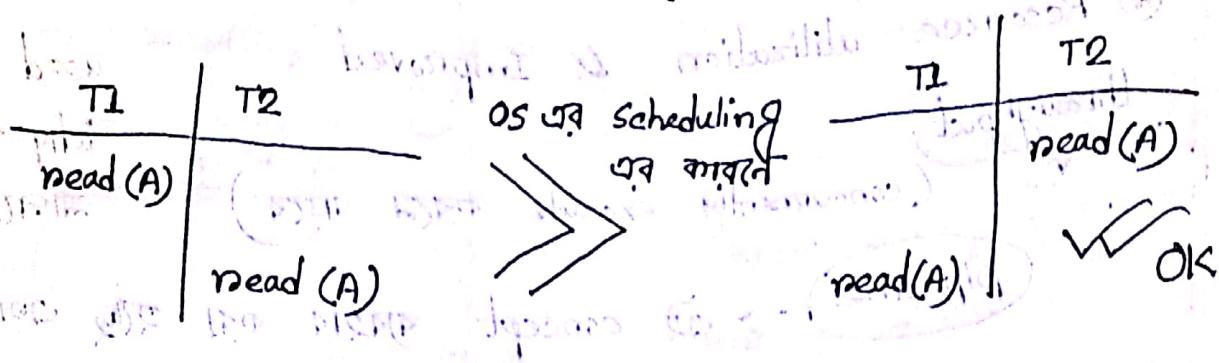


CamScanner

Concurrency এবং performance improved \rightarrow যদি

কোর করা Basically inconsistency হচ্ছে ?

read & write এর sequencing এর জন্য মাত্রা হচ্ছে ।



At least একটা write same data item নিয়ে খাতু করলে কোম্পিউটা

সত্ত্ব ।

④ Conflicting Instructions :

Summary :

Consecutive Instructions

of Different Transactions

→ atleast one write operation

→ working on same data item

* Concurrent Schedule এবং কোনো এই conflicting Instructions হলাই inconsistency এবং ক্ষয় দায়ি ।

মাত্র non-conflicting হ্য তখন
যদি swap করতে পারি ।

⑤ Consecutive Instructions of Different Transactions

whether conflicting

(swap কো মুক্তি) ① don't conflict

→ can be swapped

→ swap করে একটা equivalent schedule পাও / to
form an equivalent Schedule.

T1

read(A)

write(A)

read(B)

write(B)

T2

read(A)

write(A)

read(B)

write(B)

Consecutive & Non-conflicting so
we can swap them

Schedule 4.1

After swapping we get which is an

equivalent schedule
(same as above)

T1

T2

read(A)

write(A)

read(A)

read(B)

write(A)

write(B)

read(B)

write(B)

4.1

Schedule (4.1)

(4.1.1)

④ Conflict equivalent Schedule

(4.1.1) is a

of Schedule (4.1)

Conflict Serialization Schedule

Again swapping:

T1	T2
read(A)	
write(A)	
read(B)	
	read(A)
	write(A)
write(B)	
	read(B)
	write(B)

Schedule

(4.1.2)

(4.1.1)

T1	T2
read(A)	
write(A)	
read(B)	
write(B)	
	read(A)
	write(A)

Conflict Schedule (4.1.2)

Conflict

inconsistency

read(B)

write(B)

কোন একটা

concurrent schedule মধ্যে a series of

Swapping এর পরে serial schedule $\xrightarrow{2,1}$ → no

inconsistency will arise in that case. (when

can I say a schedule is conflict serializable)

Logically equivalent to serial schedule.

Concurrent

Schedule

Serial

Schedule

Conflict equivalent to

(i) read

(ii) write

(iii) read

(iv) write

Conflicting Instructions

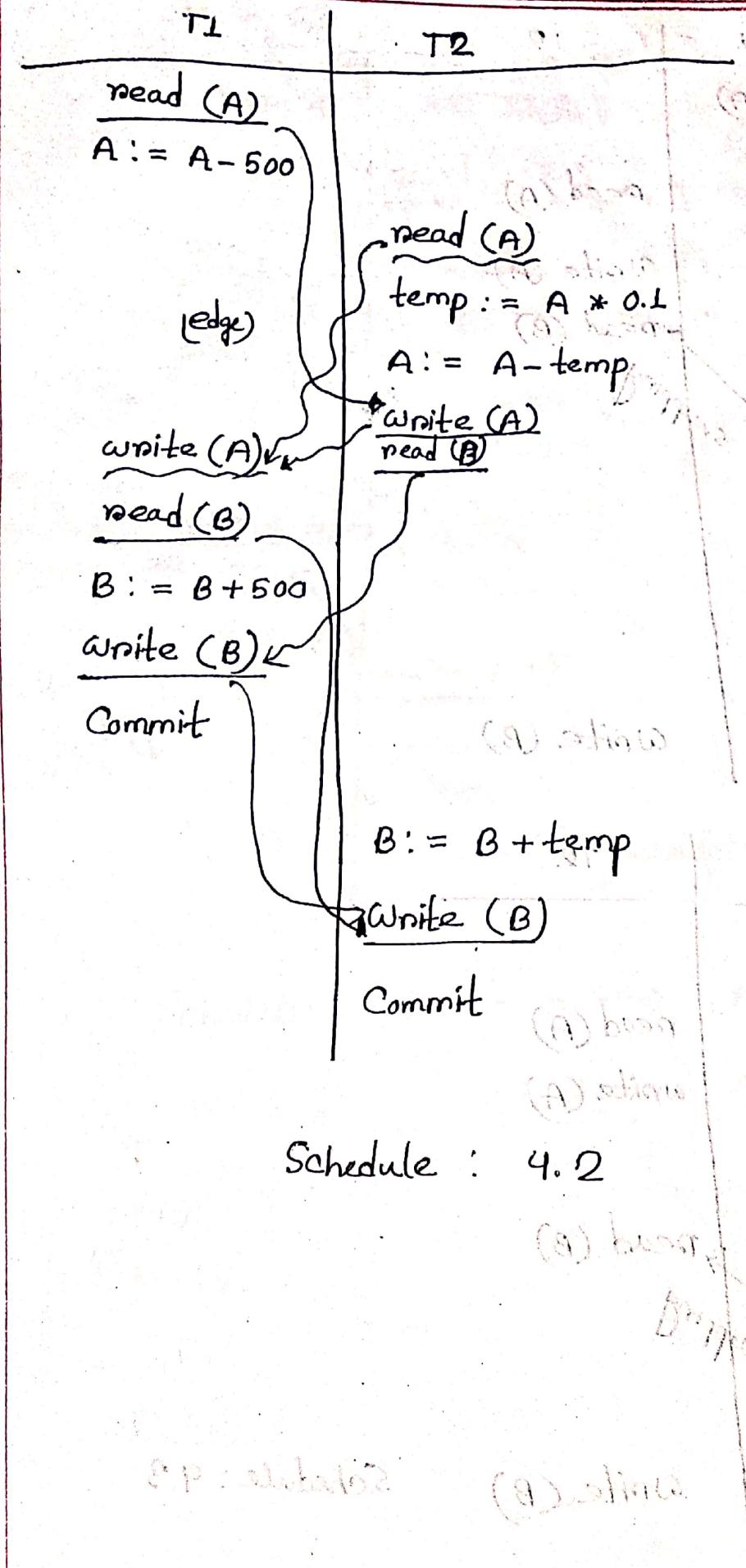
Conflict equivalent Schedule

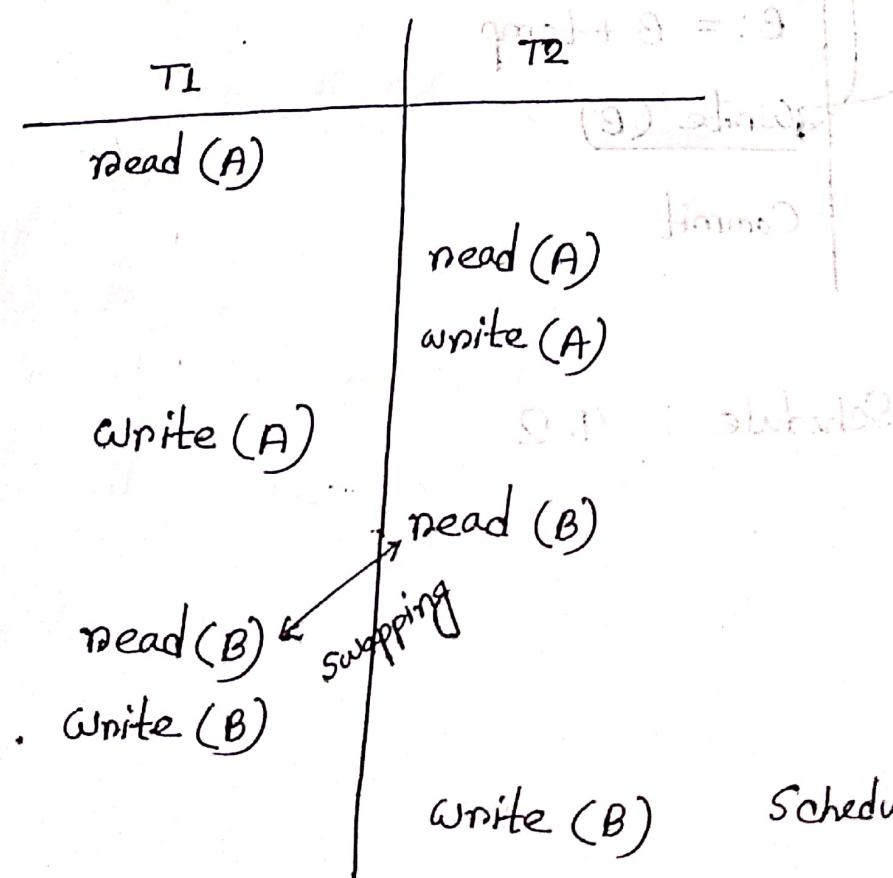
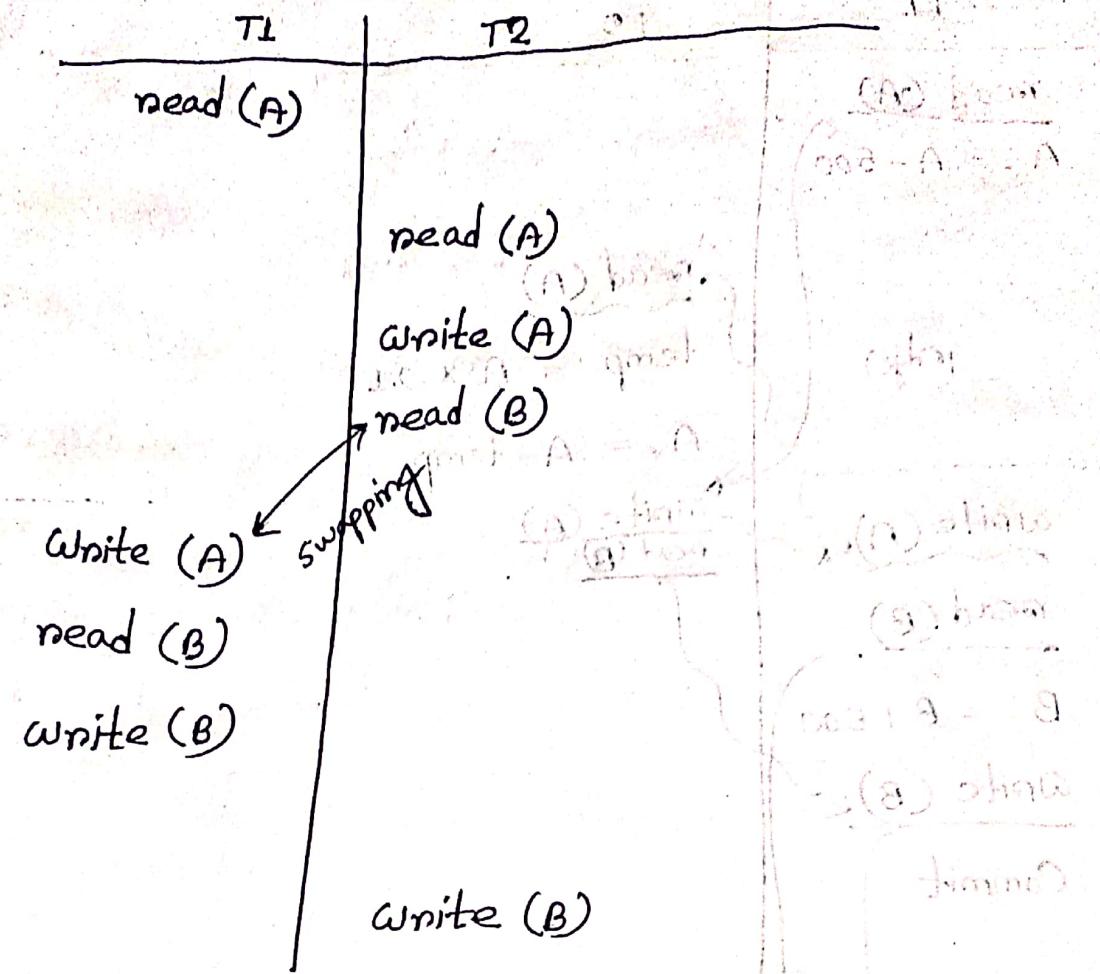
Conflict Serializable "

A concurrent Schedule which is conflict

equivalent to a Serial Schedule.

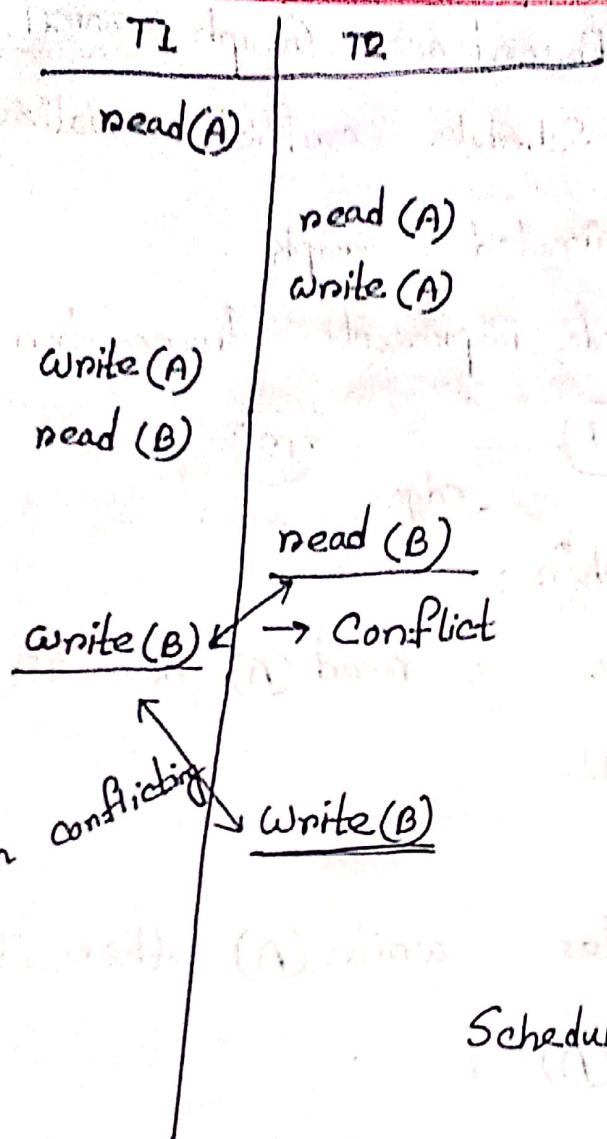
$\xrightarrow{(i) \text{ read}}$ No inconsistency
 $\xrightarrow{(ii) \text{ write}}$ problem.





write (B)

Schedule: 9.3



Schedule : 4.4

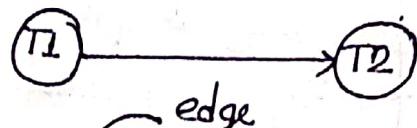
Schedule : 4.2 \longrightarrow Not conflict \Rightarrow Serializable

Schedule (এই তর্ফে consistency problem হবে)

Easier way : Precedence Graph :: কোনো পক্ষে
Schedule conflict serializable কি না,

* Directed Graph

* Node represents transaction



* when :

1. T_1 executes a read (A) then T_2 executes a write (A).

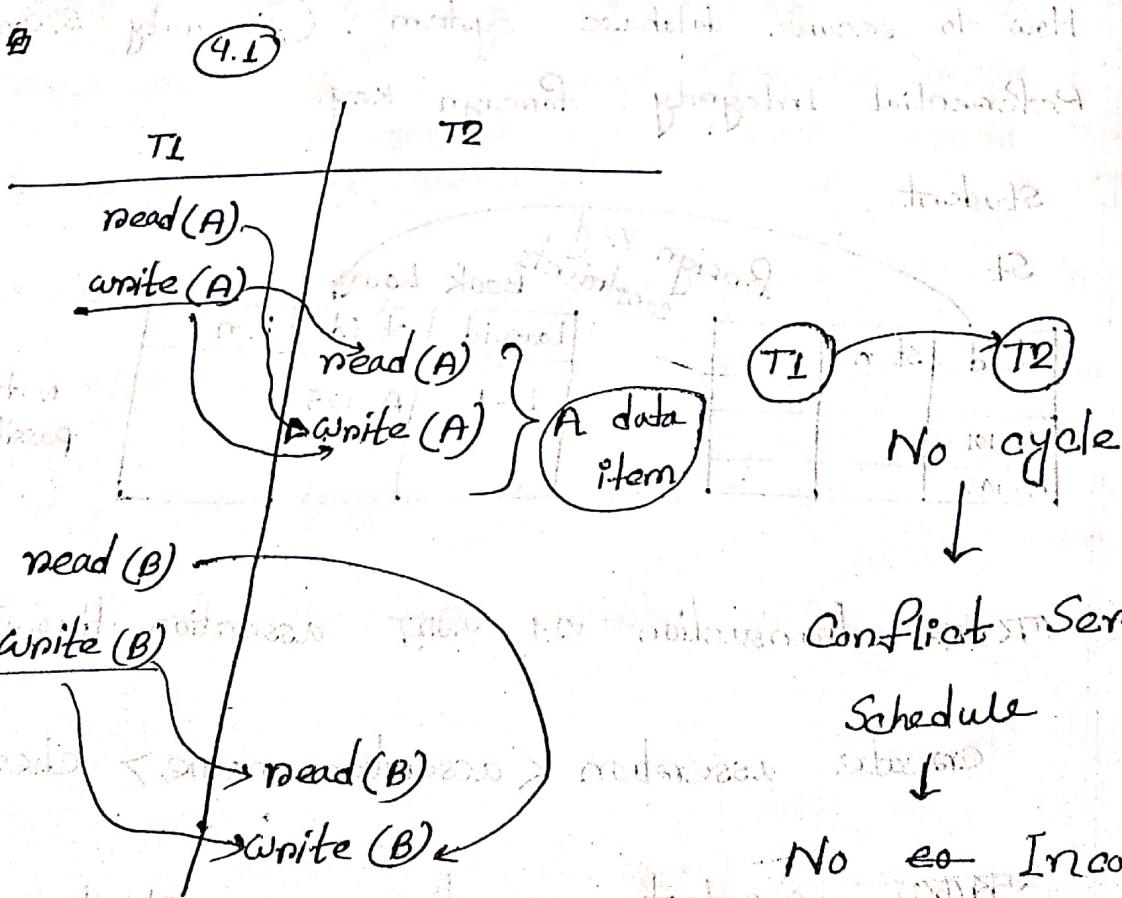
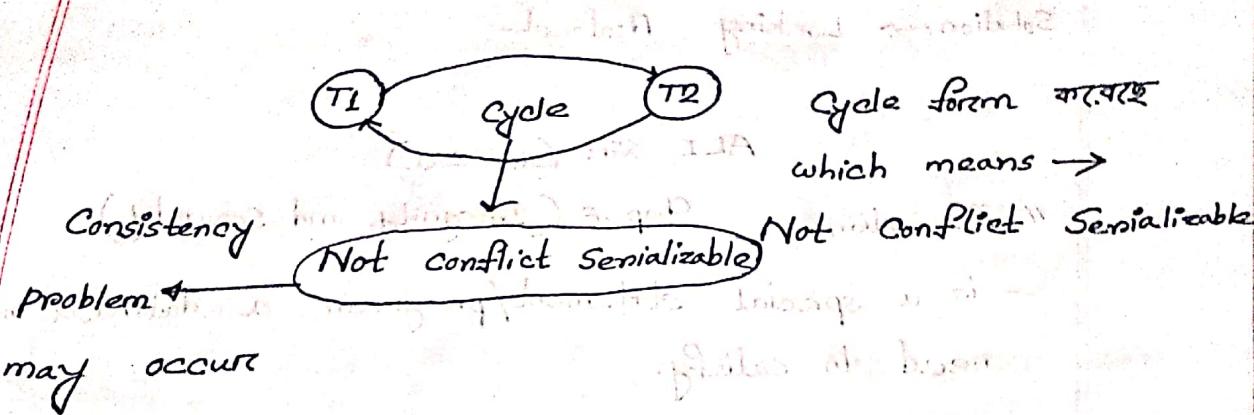
or,

2. T_1 executes write (A) then T_2 executes a read (A)

or,

3. T_1 executes write (A) then T_2 executes write (A)

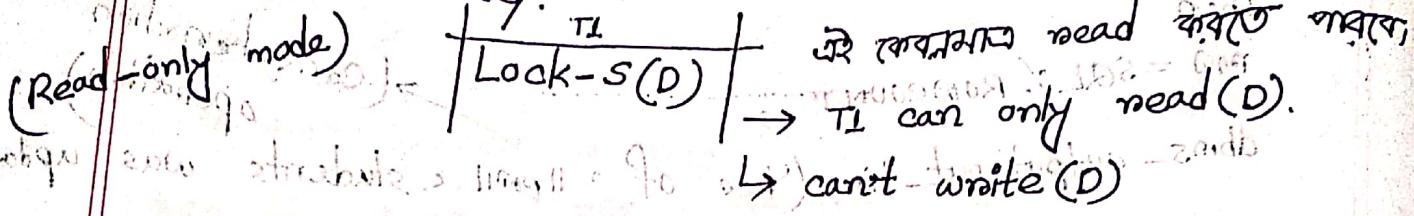
In Schedule : 4.2 →



How to solve this inconsistency problem → Locking protocol

④ Locking Modes :

(i) Shared mode:



(ii) Exclusive mode : both read(D) & write(D)

→ can read (D) and write (D) quid

~~Lock-X(D)~~

* Shared | যন্তা ইয়েছ এই ক্ষেত্রে, পারিবারিক Transaction এর

data item কে shared mode এ Lock করতে হাবে,

* Exclusive অপ্রতু Mutual exclusion প্রযোগ, যখন

II D data item କେବେ exclusive mode ଏ lock କରିବେ
ତଥା ଅନ୍ୟ କୋଣ କେବେ read / write କରାଯାଇବା
ପାରିବେ ନା ।

যদি shared mode এ Lock করা আছে
 & অন্য কোটি,, ,, এ ন করতে পারে।
 Locking করে নাই।

Concurrency-control Manager : Compatibility - Matrix এর

মার্গিত্ব measure দ্বারা Valid
 Lock কী না।
 Concurrency-control Manager

T1	T2	Grant-X(A, T1)
Lock-X(A)	Request(Locking request)	
Initial status: A = 3000 B = 1000		
read(A)		
A := A - 500		
(2500) write(A)		
unlock(A)		
problem arises: early unlocking		
read(A) 2500	Lock-S(A) → wait (wait করতে হবে এতক্ষণ Next সম্ভু না T1 A এর unlock Grant-S(A, T2) করবে) & T1 এর শেষ access করে দিব।	
Lock-S(B) →	Grant-S(B, T2)	
read(B) 1000		
Display(A+B) 3500		
unlock(A)		
unlock(B)		
Lock-X(B)		
Lock-X(B) - - - - - → Grant-X(B, T1)		
read(B)		
B := B + 500		
write(B)		
unlock(B)		
Commit		

← Lock
point

→

Compatibility - matrix : which represents Lock compatibility

यावे की ना।

		Requesting	
		T _j	S
T _i holding		X	X
		S	True compatibility
		X	False (Lock बरत)

ज्ञान योग्यता (Locking / Unlocking करने का) problem face

अब बरव ना एन ना, किंतु Locking protocol में कैसे

करते हैं।

Locking Protocol

* Two-phase Locking Protocol :
(2 PL)

Two phases:

ensures

Serializability
(Advantage of
2 PL)

(ii) Growing phase এমন একটা phase যেখানে phase এলা

grow করবে।

↳ can only obtain Lock

(প্রতিমাত্র
Lock)
কর্তৃয় থাব

↳ can't release any Lock

Lock ← Last Lock যেখানে করে ফেললাম → Last Lock obtained
point: (ii) Shrinking Phase:

(Growing
phase এবং শোষ &
shrinking phase এবং
ডাটা)

↳ can only release Lock

↳ can't obtain any new Lock

In previous example: T1 এর লকে:

Lock unlock (A)

unlock (B) &

Lock-X(B) এর পরে Lock
point হুস্ত থাব।

ফলে কোনো বিরুদ্ধে inconsistency problem হবে না।

କିମ୍ବା ଏବଳରେ କିମ୍ବା problem ରୁକ୍ଷତା ଯାତ୍ରା 2 PL ଟାଇ

Cascading Rollback problem :

ଅଛି କିମ୍ବା କିମ୍ବା କିମ୍ବା

ଅଛି କିମ୍ବା କିମ୍ବା କିମ୍ବା

ଅଛି କିମ୍ବା କିମ୍ବା

ଅଛି କିମ୍ବା କିମ୍ବା

କିମ୍ବା କିମ୍ବା କିମ୍ବା

(A) ଅବଳରେ କିମ୍ବା

(B) ଅବଳରେ କିମ୍ବା

(C) A-B କିମ୍ବା

(D) A-B କିମ୍ବା