1. Good Evening
2. We begin at 9:07 pm
3. Topic - Transactions

Agenda

1. Introduction
2. ACID
3. Start, Commit, Rollback, Demo
4. Isolation Levels - 4 levels
5. Demo
6. Deadlocks

Task [Money Transfer A → B]

Case1 [Crashes & Failures]

transferMoney (A, B, d)

1. x = Read(A) ✓
2. x = x - d ✓
3. Write (A, x) ✓ → crash/failure
4. x = Read(B)
5. x = x + d
6. Write (B, x)

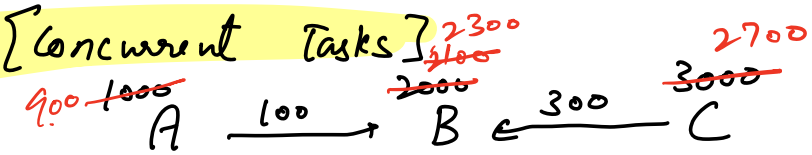
A (~~1000~~ 900) — 100 → B (2000)

x = 1000

$x = x - 100$

Write(A, x)

Case 2 [Concurrent Tasks]



1.	x = Read(A)	1000		1.	x = Read(C) 3000
2.	$x = x - 100$	900		2.	$x = x - 300$ 2700
3.	Write(A, x)			3.	Write(C, x)
4.	x = Read(B)	2000		4.	x = Read(B) 2000
5.	$x = x + 100$	2100		5.	$x = x + 300$ 2300
6.	Write(B, x)			6.	Write(B, x)

Case 1 → Failures / Crashes can lead to data inconsistency.

Case 2 → Concurrent tasks can lead to data inconsistency.

Transactions

↳ Set of operations to be run together.

ACID

A - Atomicity

C → Consistency

I → Isolation

D → Durability

A (Atomicity) → All or nothing
↳ A transaction should be indivisible

C (Consistency) → State of data before & after the transaction should be consistent.

I (Isolation) → One transaction should not be impacted by other transactions in the system.

D (Durability) → Once the transaction finishes successfully the changes should not be lost

DBMS should make sure that changes got written to hard-disk at the end of transaction

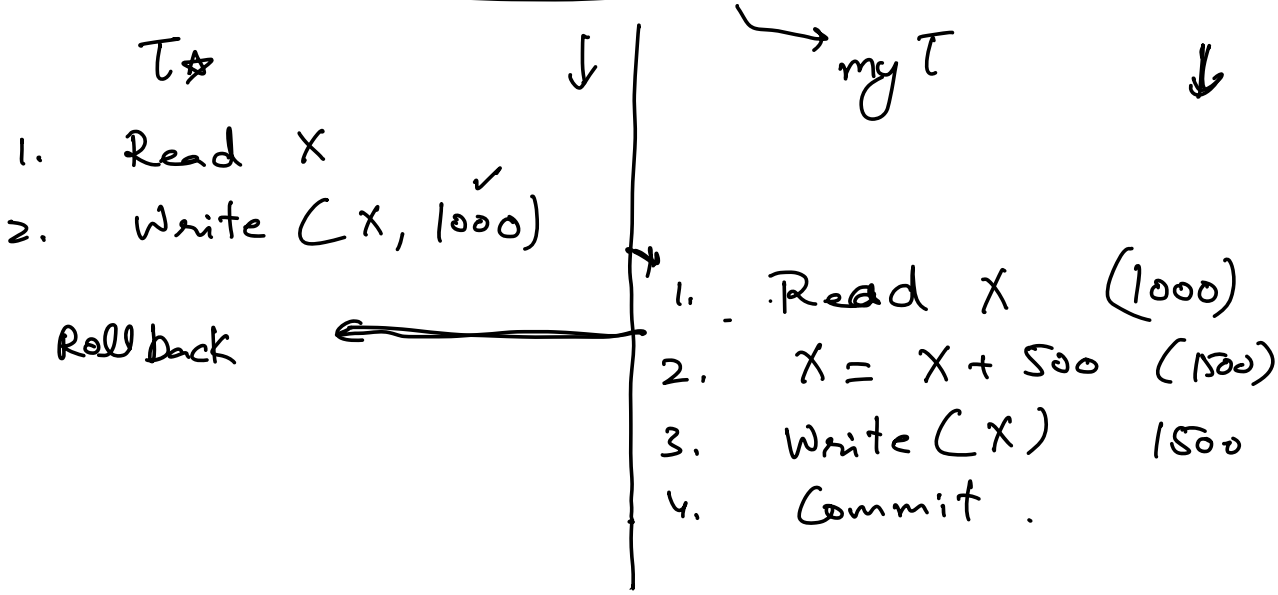
keywords → Start, Commit, Rollback [Failure]

[ISOLATION Levels & Concurrency

Concurrency vs Consistency

1. Read Uncommitted
2. Read Committed
3. Repeatable Read → Default
4. Serializable

Read Uncommitted X = ~~2000~~ 1500 ✓



Should we ever use Read Uncommitted?

↳ Financial Abblication X

- Booking.com — X
- Social Media → Maybe if data is not critical

☆ Pros of using read uncommitted
- High Performance

Read Committed ✓

→ Solves dirty read

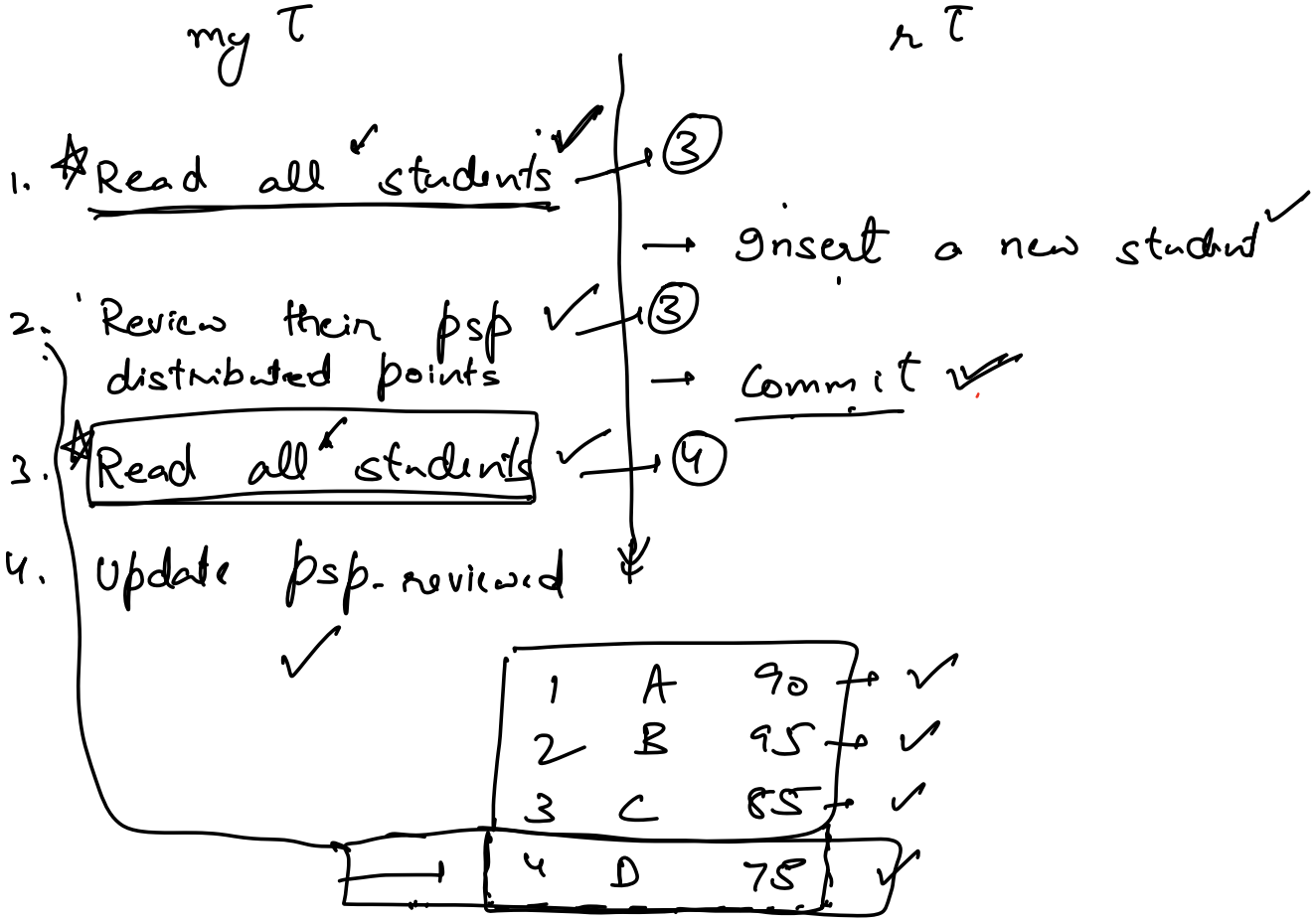
$X = \cancel{2000}$ 2500 ✓

rT (X=1000)		my T (X+=500)	
1. Read X	2000		
2. Write (X, 1000)	1000 ✓		
		1. Read X ✓	2000
3. Rollback		2. X = X + 500 ✓	2500
		3. Write (X)	2500
		4. Commit	✓

→ Problem with read committed [Non-repeatable Read]

myT → Read all students
Increase the points with ≥ 90 psp
Update all students, that psp is reviewed

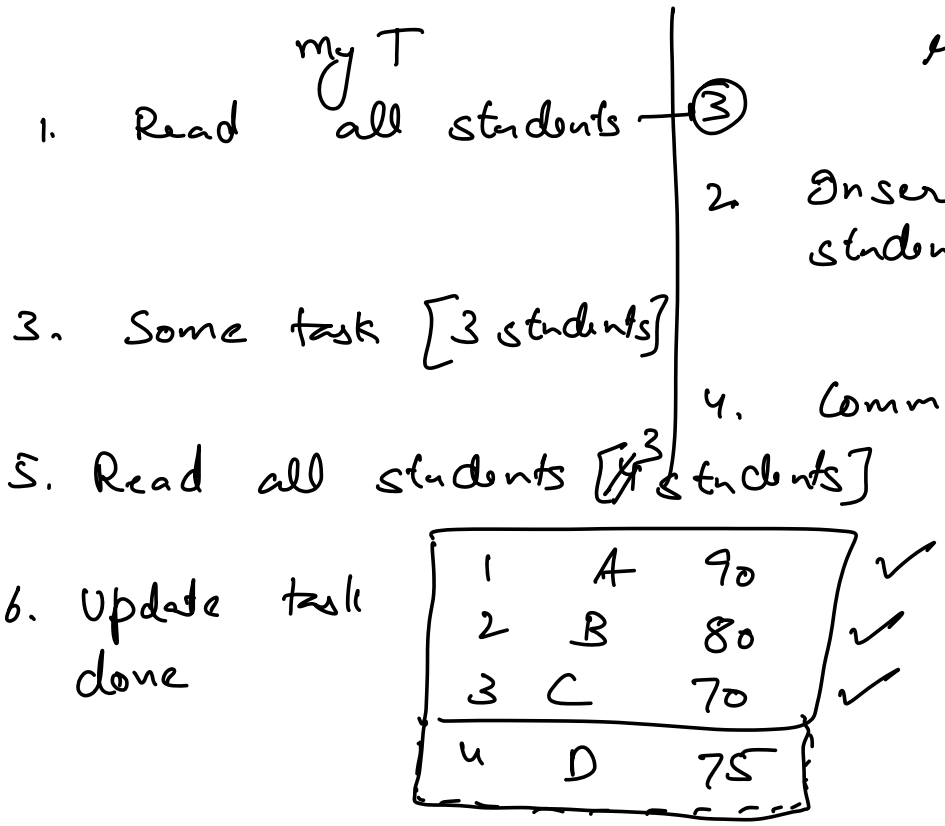
rT → Insert a new student



Read Uncommitted → dirty reads
Read Committed → ✓
Non-repeatable read

REPEATABLE READ

→ Solve non-repeatable problem



1. Read uncommitted → Dirty Read ✓
2. Read Committed → Non-repeatable read ✓
3. Repeatable Read [Default]

10:48 – 10:56

Demo → RUC, RC, RR ✓
→ Serializable
→ Demo
→ Deadlocks
→ Demo

[Problem which even repeatable read can't solve

READ SERIALIZABLE

A $\xrightarrow{100}$ B $\xleftarrow{200}$ C

~~1000~~ ~~+200~~

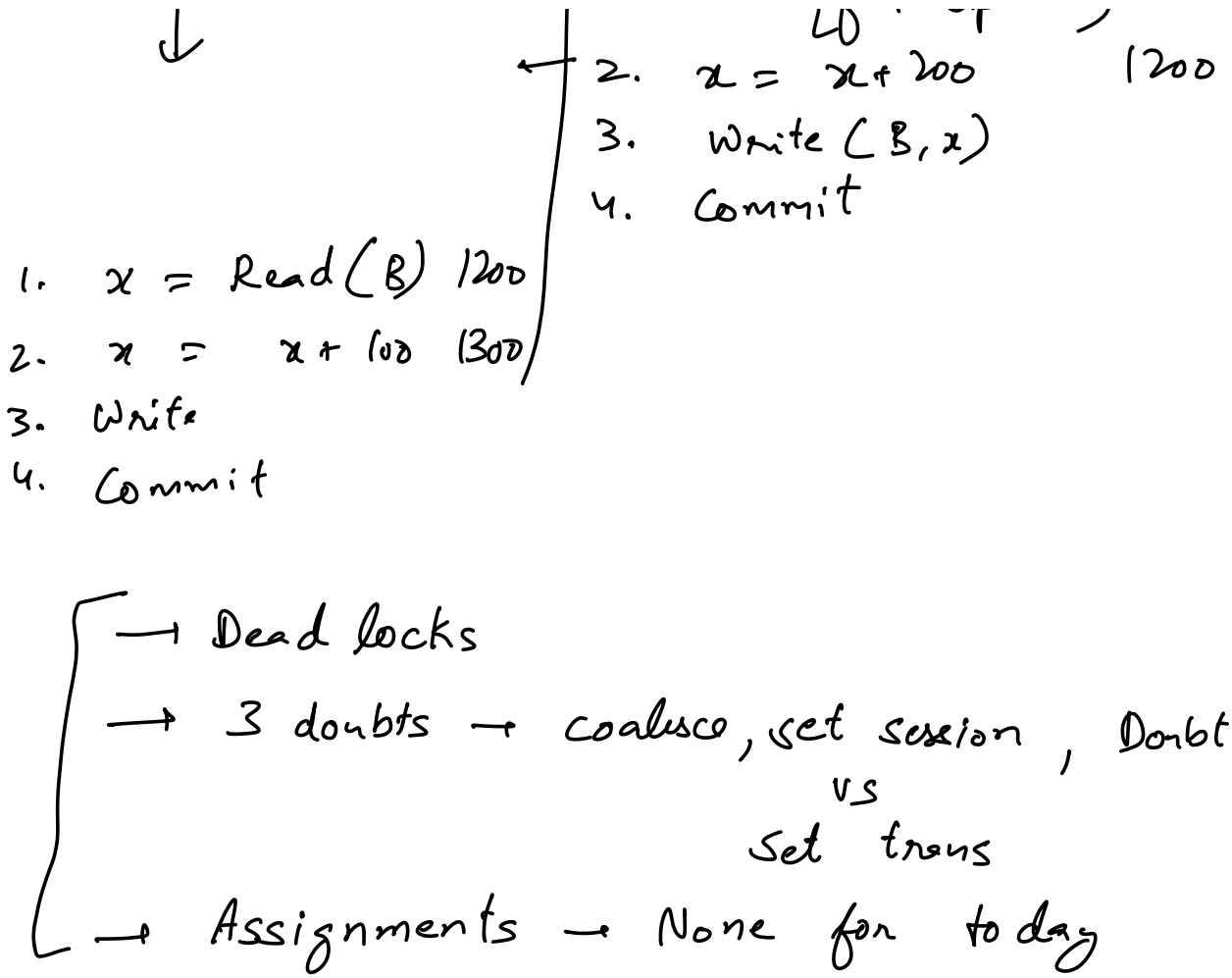
myT (A → B)	1100	rT (C → B) ✓	
✓ 1. x = read(B)	1000	1. x = read(B)	1000
		2. x = x + 200	1200
2. x = x + 100	1100	3. Write (B, x)	
3. Write (B, x)		4. Commit	
4. Commit			

~~1200~~ 1300
~~1000~~

A $\xrightarrow{100}$ B $\xleftarrow{200}$ C

Read, Serializable

1. x = Read with lock (B) Select [for update] 1000



- Dead locks
- 3 doubts → coalesce, set session, Doubt vs set trans
- Assignments → None for today