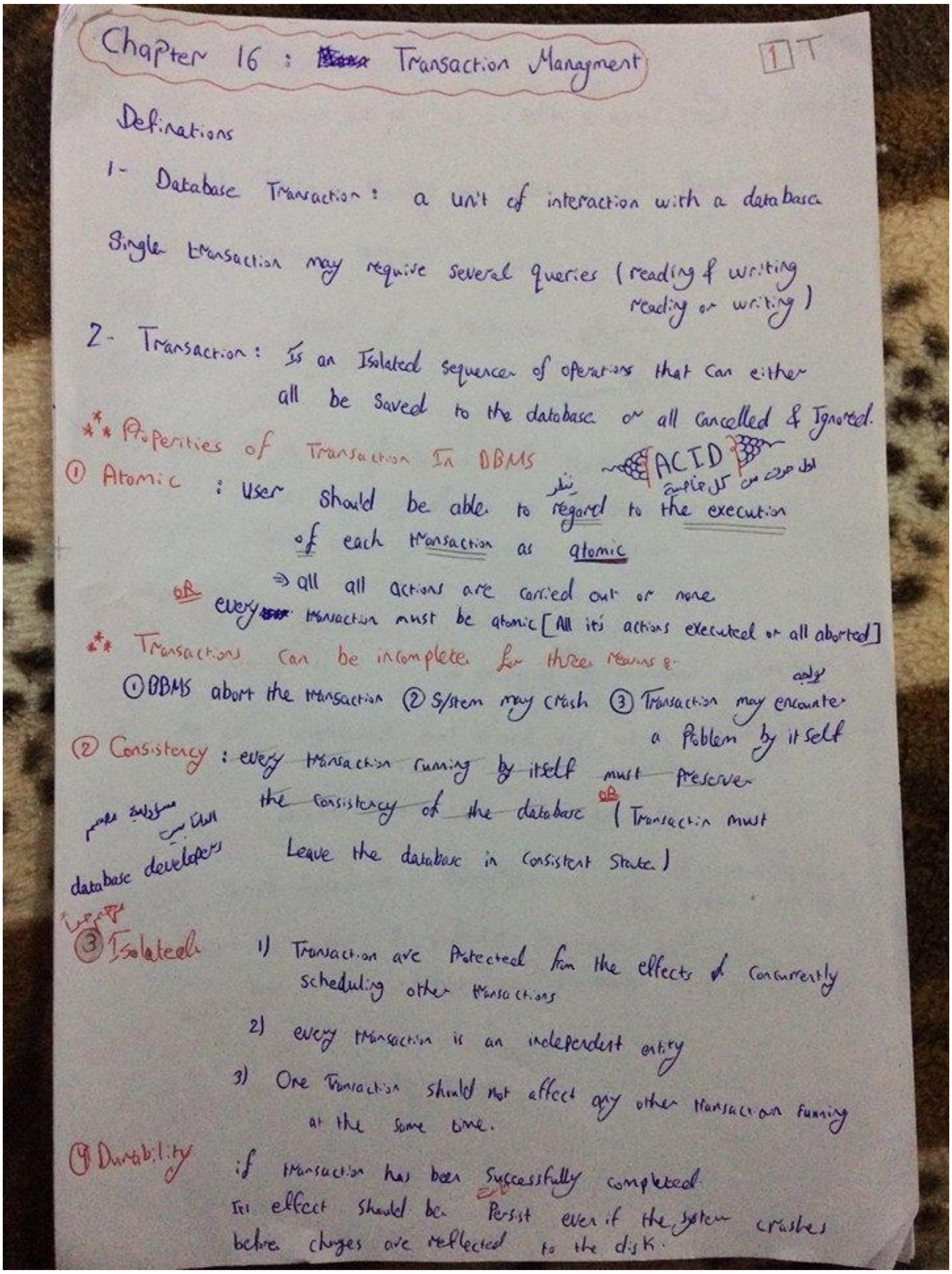


Chapter 16 : ~~[illegible]~~ Transaction Managment

$\boxed{1}$ T

Definations

1- Database Transaction : a unit of interaction with a database

Single transaction may require several queries (reading & writing
reading or writing)

2- Transaction : Is an Isolated sequence of operations that can either all be saved to the database or all cancelled & Ignored.

*** Properities of Transaction In DBMS — ACID اول حرف من كل خاصية

① Atomic : User should be able to regard (ننظر) to the execution of each transaction as atomic

⇒ all all actions are carried out or none

OR every ~~[illegible]~~ transaction must be atomic [All its actions executed or all aborted]

*** Transactions can be incomplete for three reasons :

① DBMS abort the transaction ② System may crash ③ Transaction may encounter (يواجه) a Problem by itself

② Consistency : ~~every transaction running by itself must preserve the consistency of the database~~ OR (Transaction must Leave the database in consistent State.)

مسؤولية معظم الكاتبين
database developers

③ Isolated معزول

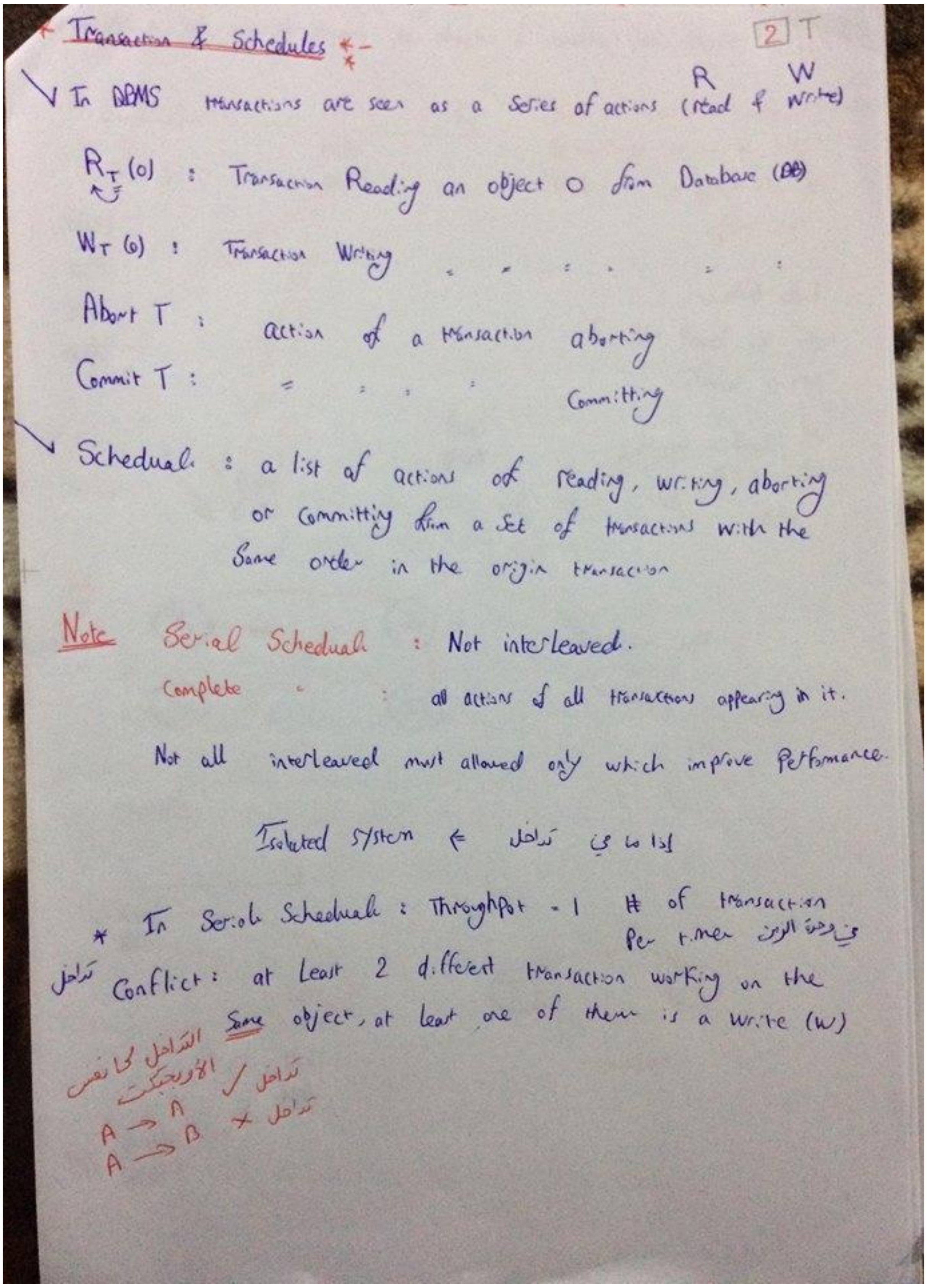
1) Transaction are Protected from the effects of concurrently scheduling other transactions

2) every transaction is an independent entity

3) One Transaction should not affect any other transaction running at the same time.

④ Durability

if transaction has been successfully completed Its effect should be [illegible] Persist even if the system crashes before changes are reflected to the disk.



* ## Transaction & Schedules *-

2 T

In DBMS transactions are seen as a series of actions (read & write)

R W

$R_T(O)$: Transaction Reading an object O from Database (DB)

$W_T(O)$: Transaction Writing " " " " " "

Abort T : action of a transaction aborting

Commit T : " " " " Committing

Scheduale : a list of actions of reading, writing, aborting or committing from a set of transactions with the same order in the origin transaction

Note Serial Scheduale : Not interleaved.

Complete " : all actions of all transactions appearing in it.

Not all interleaved must allowed only which improve Performance.

Isolated system ⇐ اذا ما في تداخل

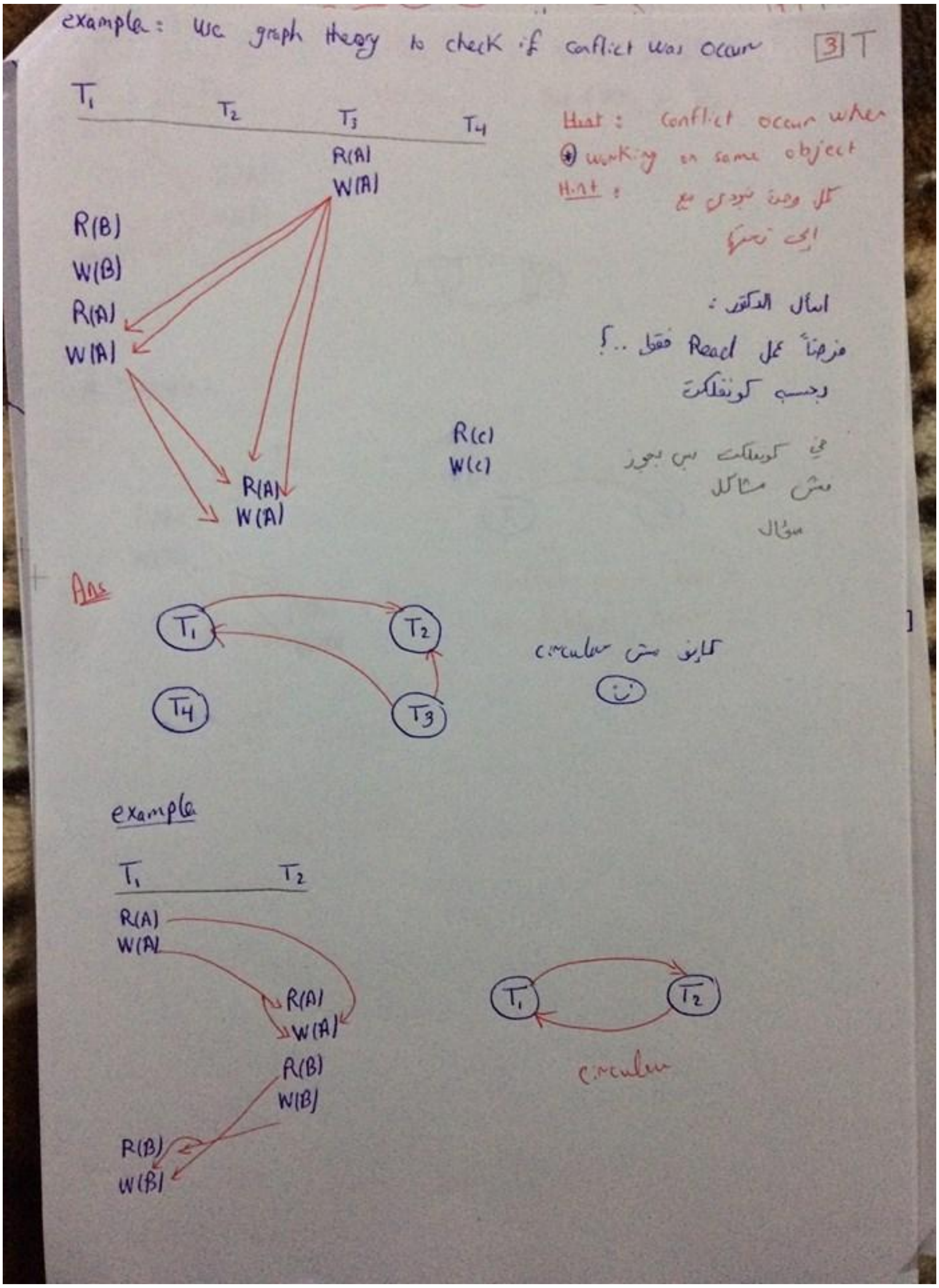
* In Serial Scheduale : Throughput = # of transaction per time في وحدة الزمن

تداخل Conflict : at Least 2 different transaction working on the Same object, at least one of them is a write (W)

التداخل على نفس الاوبجكت

A → A تداخل ✓

A → B تداخل ✗



example: use graph theory to check if conflict was occur $\boxed{3}$ T

T_1	T_2	T_3	T_4
		R(A)	
		W(A)	
R(B)			
W(B)			
R(A)			
W(A)			
			R(C)
			W(C)
	R(A)		
	W(A)		

Hint : conflict occur when
⊛ working on same object

Hint : كل وحدة بتودي مع
إلي بعدها

اسأل الدكتور :
فرضاً عمل Read فقط ..؟
وحسب كونفلكت

في كونفلكت بس بجوز
مش مشاكل
سؤال

Ans

$$\begin{tikzcd}
T_1 \arrow[rr, bend left] & & T_2 \\
T_4 & & T_3 \arrow[llu, bend right] \arrow[u]
\end{tikzcd}$$

كاينو مش circular

example

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

$$\begin{tikzcd}
T_1 \arrow[r, bend left] & T_2 \arrow[l, bend left]
\end{tikzcd}$$

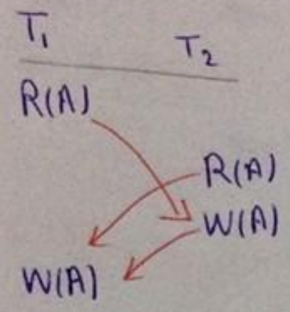
circular

$\boxed{4}$ T

example:

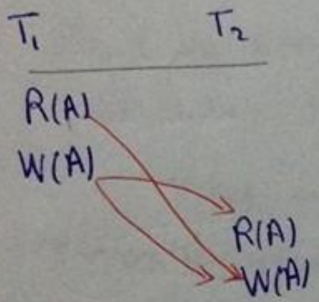
Read : تأثر على Write

Write : تأثر على Read & Write



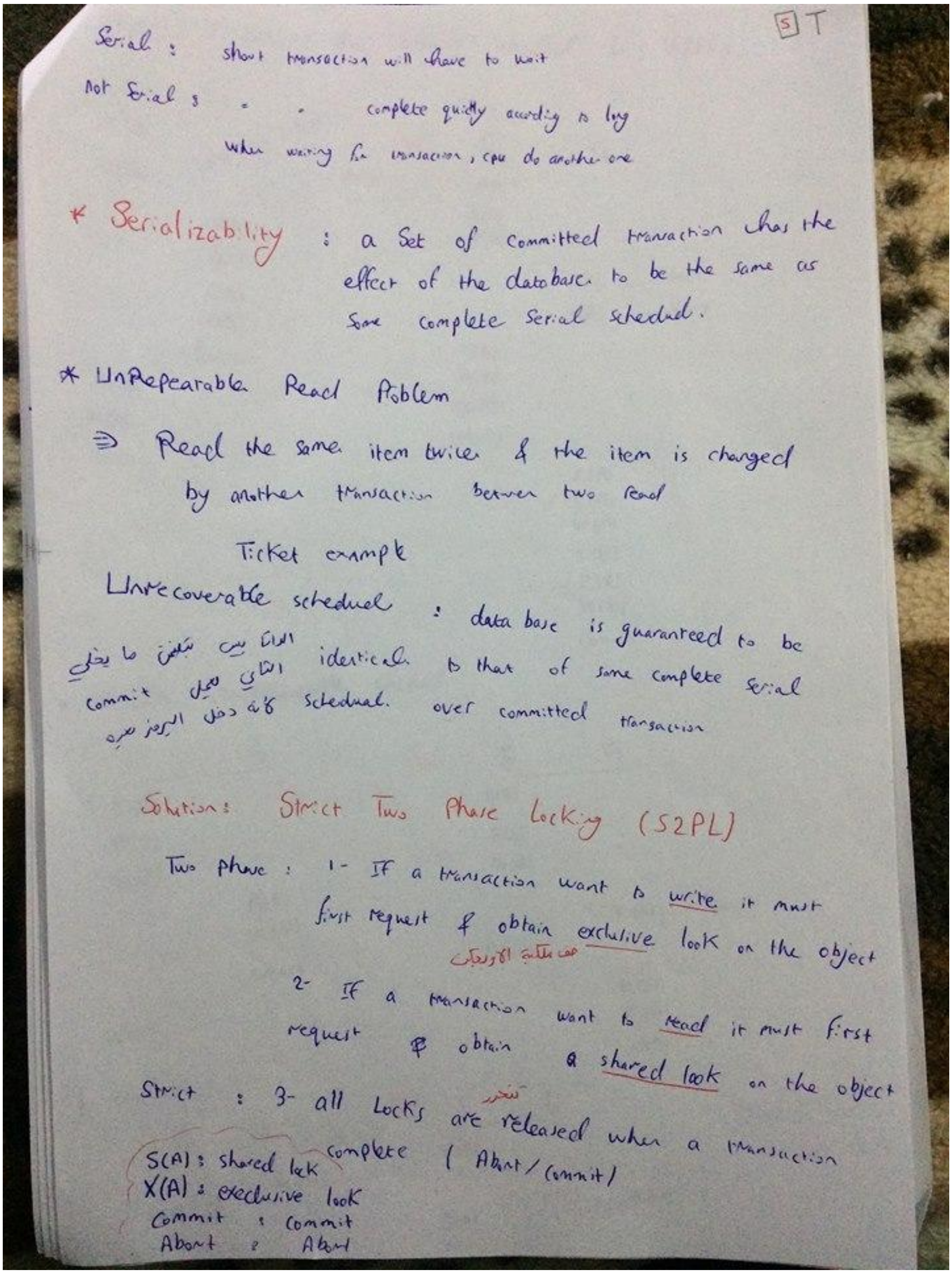
$$\begin{tikzcd} T_1 \arrow[r, bend left] & T_2 \arrow[l, bend left] \end{tikzcd}$$

* example:



$$\begin{tikzcd} T_1 \arrow[r, bend left] & T_2 \end{tikzcd}$$

Conflict exisit but no Problem occur



[5] T

Serial : short transaction will have to wait

Not Serial : " " complete quickly according to long

when waiting for transaction, cpu do another one

* Serializability : a Set of committed transaction has the effect of the database to be the same as some complete serial schedual.

* UnRepeatable Read Problem

⇒ Read the same item twice & the item is changed by another transaction between two read

Ticket example

Unrecoverable schedule : data base is guaranteed to be identical to that of some complete serial schedual. over committed transaction

الداتا بيس شغلين ما يخلي الثاني يعمل commit كأنه دخل اليوزر مره

Solution: Strict Two Phase Locking (S2PL)

Two phase : 1- If a transaction want to write it must first request & obtain exclusive look on the object

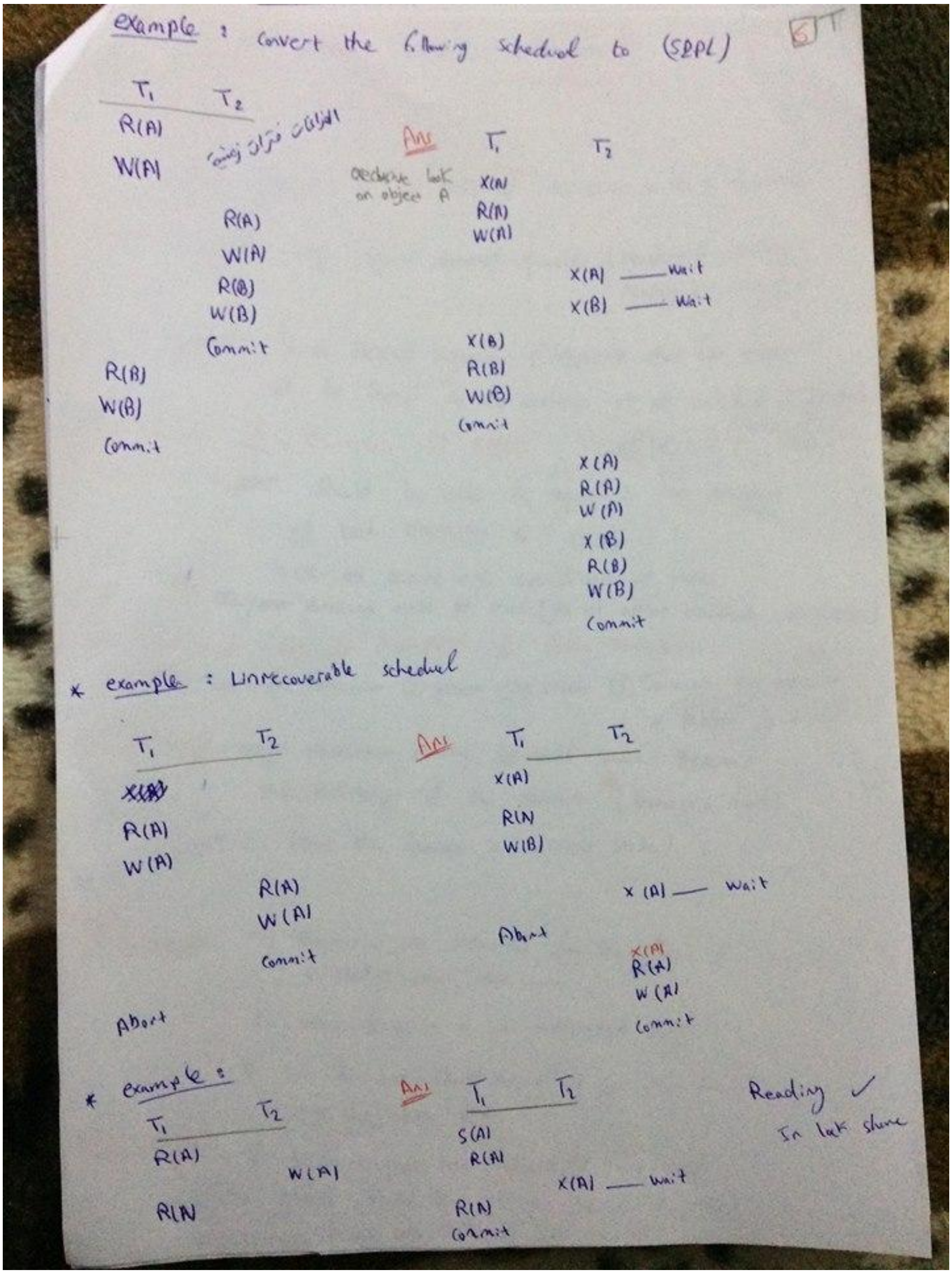
ملكية الاوبجكت

2- If a transaction want to read it must first request & obtain a shared look on the object

Strict : 3- all Locks are released when a transaction complete (Abort / Commit)

تتحرر

S(A) : shared lock
X(A) : exclusive look
Commit : Commit
Abort : Abort



<u>example</u> : Convert the following schedual to (SPPL) 6

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

الاجراءات متزامنة زمنياً

<u>Ans</u>

exclusive lock on object A

T1	T2
X(A)	
R(A)	
W(A)	
	X(A) —— Wait
	X(B) —— Wait
X(B)	
R(B)	
W(B)	
Commit	
	X(A)
	R(A)
	W(A)
	X(B)
	R(B)
	W(B)
	Commit

* <u>example</u> : Unrecoverable schedual

T1	T2
~~X(A)~~	
R(A)	
W(A)	
	R(A)
	W(A)
	Commit
Abort	

<u>Ans</u>

T1	T2
X(A)	
R(A)	
W(B)	
	X(A) —— wait
Abort	
	X(A)
	R(A)
	W(A)
	Commit

* <u>example :</u>

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

<u>Ans</u>

T1	T2
S(A)	
R(A)	
	X(A) —— wait
R(A)	
Commit	

Reading ✓
In lock share

Example

	T_1	T_2
	X(A)	
	R(A)	
	W(A)	
		X(B)
		R(B)
		W(B)
Wait ←	X(B)	
		⋮
		X(A) → wait

A ~~&~~ B are two different object

Previous example called " Dead Lock "

The solution for this problem is " system Time out "

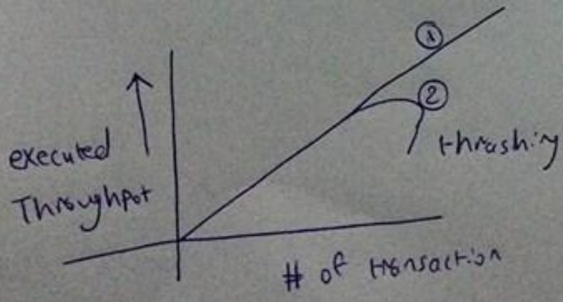
بعد فترة معينة السيستم بعمل Abort

Notes

$$\begin{tikzcd} T_1 \arrow[r, bend left] & T_2 \arrow[l, bend left] \end{tikzcd}$$

this circle Means ⇒ Dead Lock

⇒ We have to delete the transaction that cause this Loop



① Normal (No Problem)

② Problem occur

⇒ Losing data

Solution : Determine # of users Who Can access