HW

① Serializability & 2PL

1.1

	T_1	T_2	T_3
t_1	read(x)		
t_2		read(y)	
t_3			write(x)
t_4		write(x)	
t_5	read(y)		

read(x) – In T_1, no subsequent writes to A, no new edge.
read(y) → In T_2, no subsequent writes to y, no new edge.
write(x) → x is subsequent read by T_2. new edge. $T_3 \to T_2$
read(x) → In T_2, no subsequent writes to x, no new edge.

Serializable → no cycles

1.2

	T_1	T_2	T_3
	S.lock(x)		
t_1	read(x)		
	S.lock(y)		
t_2	read(y)		
	X.lock(x)		
t_3	write(x)		
	X.lock(x)	S.lock(y)	
t_4	S.rl(x)	read(y)	
			X.lock(y)
t_5	write(x)		write(y)
	X.rl(x)		X.rl(y)
		S.lock(y)	
t_6		read(y)	
		S.rl(y)	

read(x) – No future write to x
read(y) → y is written in T_3 : $T_1 \to T_3$
write(x) → no future write to x
read(y) → y is written in T_3 : $T_2 \to T_3$
write(y) → y is read in T_2 : $T_3 \to T_2$

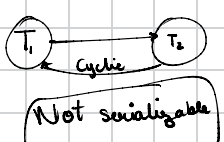
Follows 2PL

Not Serializable, precedence graph has cycles.

1.3

	T_1	T_2
	X.lock(x)	
t_1	write(x)	
	X.rl(x)	
t_2		S.lock(x)
		read(x)
		S.rl(x)
t_3	X.lock(x)	
	write(x)	
	X.rl(x)	

Doesn't follow 2PL because T_1 tries to get an x-lock on x after releasing it.



1.4

	T_1	T_2	T_3
	S.lock(x)		
t_1	read(x)		
		X.lock(x)	
t_2		write(x)	
t_3	write(x)		
t_4			S.lock(x)
			read(x)

t_1: read(x) → x has future write in T_2 : $T_1 \to T_2$
t_2: write(x) → x has future read in T_3 : $T_2 \to T_3$
t_3: write(x) → x has future read in T_3 : $T_1 \to T_3$
t_4: x doesn't have future write elsewhere

No Cycles = Serializable.

2) Degree of consistency:

T1 has a max degree of consistency of 0.
T2 has a max degree of consistency of 3.

3) Read Uncommitted:

* the isolation is allowing transactions to read uncommitted data from other transactions potentially leading to dirty reads.

* Equivalent degree of consistency: Degree 1: this is cause Read uncommitted allows the transaction to read data that has been read but not committed by the current transaction.

Read Committed:

* It ensures the transaction only reads data that has been committed by other transactions. It ensures that a transaction sees only consistent data & prevents any dirty reads, so it doesn't read uncommitted data.
* Equivalent degree of consistency: Degree 2- it aligns w/ the principle that transactions don't commit any writes until the end of the transaction, ensuring consistency & preventing partial updates from being visible to other transactions

Repeatable Read:

* Ensuring that once a transaction reads a piece of data, it will see the same data for the remainder of that transaction, even if other transactions are modified or inserted data.
* Equivalent degree of consistency: Degree 0 → doesn't overwrite dirty data of other xacts. Satisfying this because this degree prevents other transactions from modifying data reads by the current transaction.

Serializable

* this prevents all isolations. Creates a shared lock on all statements this follows a degree of consistency of 3. this degree satisfies by ensuring that data read by current transaction remains consistent until the transaction completes.

4) Recovery & Aries:

a) Action 0:
* Action: T1 updates P7, P7 brought into buffer.
* XT: {T1:00}
* DPT: {P7:00}

Action 1:
* Action: T0 updates P9, P9 brought into buffer, P9 flushed to disk
* XT: {T1:00, T0:10}
* DPT: {P7:00}

Action 2: LSN=20
* Action: T1 updates P8, P8 brought into buffer, P8 flushed to disk
* XT: {T1:20, T0:10}
* DPT: {P7:00}

Action 3: LSN=30
* Action: Begin checkpoint
* No change to XT & DPT

Action 4: LSN=40
* Action: end checkpoint
* No change to XT & DPT

Action 5: LSN=50
* Action: T1 updates P9
* XT {T1:50, T0:10}
* DPT {P7:00, P9:50}

Action 6: LSN=60
* Action: T2 updates P6
* XT {T1:50, T0:10, T2:60}
* DPT {P7:00, P9:50, P6:60}

Action 7: LSN=70

* Action: T1 updates P5
* XT {T1:70, T0:10, T2:60}
* DPT {P7:00, P9:50, P6:60, P5:70}

Action 8: LSN=80

* Action: T1 updates P7, P6 flushed to disk
* XT {T1:80, T0:10, T2:60}
* DPT {P7:00, P9:50, P5:70}

Action 9:

Restart

XT

trans ID	rec LSN
T1:	80
T0:	10
T2:	60

DPT

Page ID	last LSN
P7:	80

4B) the analysis

* It begins where the checkpoint ends.
* The analysis in this case starts at log time 4 with LSN=40

Time	LSN	Transaction table	Dirty Page table
4	40	T1: 20, T0: 10	P7: 00
5	50	T1: 50, T0: 10	P7: 00, P9: 50
6	60	T1: 50, T0: 10, T2: 60	P7: 00, P9: 50, P6: 60
7	70	T1: 70, T0: 10, T2: 60	P7: 00, P9: 50, P6: 60, P5: 70
8	80	T1: 80, T0: 10, T2: 60	P7: 00, P9: 50, P5: 70
9	Restart	Restart	Restart

* Analysis is starting here

4c) The redo phase starts at 00, that is because that is the earliest page LSN from the end_checkpoint.

Time	LSN	Log	Redo or Not
0	00	update : T1 updates P7	Redo
1	10	update: T0 updates P9	NOT
2	20	T1 updates P8	Not
3	30	begin checkpoint	Not
4	40	end checkpoint	Not
5	50	T1 updates P9	Redo
6	60	T2 updates P6	Not
7	70	T1 updates P5	Redo
8	80	T1 updates P7	Redo

5) Recovery and Aries

The ARIES and InnoDB Crash Recovery algorithms share striking similarities. They both encompass “Analysis” and “Redo” phases, which are crucial for data restoration. While ARIES maintains both an undo and redo log, InnoDB primarily relies on a redo log. Additionally, both algorithms address incomplete transactions through an undo mechanism. Notably, InnoDB offers a unique feature where, if all changes preceding the crash have been successfully flushed, it skips the redo phase altogether.