Diberikan urutan kedatangan instruksi transaksi T1, T2, dan T3 ke DBMS berikut. Transaksi dimulai tepat sebelum instruksi pertama pada transaksi tersebut.
Keterangan: R(*Q*) adalah *read* data *Q* dan W(*Q*) adalah *write* pada data *Q* untuk transaksi terkait.

T1	T2	T3
R(A)		
	R(A)	
		R(B)
R(B)		
		W(C)
	W(C)	
R(C)		
commit		
	R(D)	
		W(B)
		commit
	W(D)	
	commit	

Tuliskan langkah-langkah untuk mengeksekusi jadwal di atas sampai semua transaksi tuntas dieksekusi dengan ***multiversion timestamp ordering protocol*** dan jelaskan setiap tahapan eksekusi, termasuk versi data yang dihasilkan. Sebutkan dan jelaskan apa yang terjadi pada tiap transaksi: apakah *abort* atau *commit*. Jika terjadi *abort*, jelaskan bagaimana proses *rollback* dilakukan. Jika dibutuhkan, *timestamp* (Tx,Ty,Tz) = (1,2,3).
Asumsikan: Jika terjadi *abort* yang menyebabkan sebuah transaksi di-*rollback*, DBMS akan memprioritaskan eksekusi ulang dari instruksi transaksi tersebut hingga bagian instruksi yang menyebabkan *abort*.

<Qx, RTS, WTS> = data Q versi x dengan R-timestamp = RTS dan W-timestamp = WTS

Versi data saat jadwal dimulai:
<A0, 0, 0>
<B0, 0, 0>
<C0, 0, 0>
<D0, 0, 0>

T1 (TS = 1)	T2 (TS = 2)	T3 (TS = 3)	
R(A) <A0, 1, 0>			T1 get <A0, 0, 0> and update A0
	R(A) <A0, 2 , 0>		T2 get <A0, 1, 0> and update A0
		R(B) <B0, 3, 0>	T3 get <B0, 0, 0> and update B0

R(B) <B0, 3, 0>			T1 get <B0, 3, 0>
		W(C) <C1, 3, 3>	T3 get <C0, 0, 0> and create C1
	W(C) <C2, 2, 2>		T2 get <C0, 0, 0> and create C2
R(C) <C0, 1, 0>			T3 get <C0, 0, 0> and update C0
commit			
	R(D) <D0, 2, 0>		T1 get <D0, 0, 0> and update D0
		W(B) <B1, 3, 3>	T3 get <B0, 3, 0> and create B1
		commit	
	W(D) <D1, 2, 2>		T3 get <D0, 2, 0> and create D1
	commit		

<C0, 0, 0>¶
<D0, 0, 0>¶
¶

T1·(TS=1)¤	T2·(TS=2)¤	T3·(TS=3)¤	·¤	¤
·R(A)·····<A0,1,0>¤	·¤	·¤	T1·gets·<A0,0,0>·and·updates·A0¤	¤
·¤	R(A)·····<A0,2,0>¤	·¤	T2·gets·<A0,1,0>·and·updates·A0¤	¤
·¤	·¤	R(B)·····<B0,3,0>¤	T3·gets·<B0,0,0>·and·updates·B0¤	¤
R(B)·····<B0,3,0>¤	·¤	·¤	T1·gets·<B0,3,0>·¤	¤
·¤	·¤	W(C)·····<C1,3,3>¤	T3·gets·<C0,0,0>·and·creates·C1¤	¤
·¤	W(C)·····<C2,2,2>¤	·¤	T2·gets·<C0,0,0>·and·creates·C2¤	¤
R(C)·····<C0,1,0>¤	·¤	·¤	T1·gets·<C0,0,0>·and·updates·C0¤	¤
commit¤	·¤	·¤	¤	
·¤	R(D)·····<D0,2,0>¤	·¤	T1·gets·<D0,0,0>·and·updates·D0·¤	¤
·¤	·¤	W(B)·····<B1,3,3>¤	T3·gets·<B0,3,0>·and·creates·B1¤	¤
·¤	·¤	commit¤	·¤	¤
·¤	W(D)·····<D1,2,2>¤	·¤	T3·gets·<D0,2,0>·and·creates·D1·¤	¤
·¤	commit¤	·¤	·¤	¤

¶
¶

Diberikan urutan kedatangan instruksi transaksi T1, T2, dan T3 ke DBMS berikut. Transaksi dimulai tepat sebelum instruksi pertama pada transaksi tersebut.
Keterangan: R(*Q*) adalah *read* data *Q* dan W(*Q*) adalah *write* pada data *Q* untuk transaksi terkait.

T1	T2	T3
R(A)		
	R(A)	
		R(B)
R(B)		
		W(C)
	W(C)	
R(C)		
commit		
	R(D)	
		W(B)
		commit
	W(D)	
	commit	

Tuliskan langkah-langkah untuk mengeksekusi jadwal di atas sampai semua transaksi tuntas dieksekusi dengan ***multiversion two-phase locking protocol*** dan jelaskan setiap tahapan eksekusi, termasuk versi data yang dihasilkan.

T1: read-only txn
T2 dan T3: update txns

TS-counter at start = 0
Versi saat start: A0, B0, C0, D0

T1	T2	T3	
R(A)			T1(TS) = 0, A0
	R(A)		L-S(A), A0
		R(B)	L-S(B), B0
R(B)			T1(TS) = 0, B0
		W(C)	L-X(C), Result: C locked

	W(C)		Wait L-X(C) dari T2
R(C)			TS(T1) = 0, C0
commit			
	R(D)		Wait T2
		W(B)	Upgrade L-X(B), B locked
		commit	TS-counter = 1, C0 → C1, B0 → B1, UL(C), UL(B)
	W(C)		L-X(C), Result: C locked
	R(D)		L-S(D), D0
	W(D)		L-X(D), Result: D locked
	commit		TS-counter = 2, C1 → C2, D0 → D2, UL(C), UL(D)

Multiversion two-phase locking protocol

T1: read-only txn; T2 and T3: update txns

TS-counter at start = 0

Versions at start: A0, B0, C0, D0

T1	T2	T3	
			TS(T1)=0, A0
R(A)			Lock-S(A), A0
	R(A)		Lock-S(B), B0
		R(B)	TS(T1)=0, B0
R(B)			Lock-X(C), Result: C∞
		W(C)	Wait for Lock-X(C) of T2
	W(C)		TS(T1)=0, C0
R(C)			
commit			Wait for T2
	R(D)		Upgrade Lock-X(B), B∞
		W(B)	TS-counter = 1, C0 → C1, B0 → B1, Unlock(C), Unlock(B)
		commit	Lock-X(C), Result: C∞
	W(C)		Lock-S(D), D0
	R(D)		Lock-X(D), Result: D∞
	W(D)		TS-counter = 2, C1 → C2, D0 → D2, Unlock(C), Unlock(D)
	commit		

Diberikan urutan kedatangan instruksi transaksi T1, T2, dan T3 ke DBMS berikut. Transaksi dimulai tepat sebelum instruksi pertama pada transaksi tersebut.
Keterangan: R(*Q*) adalah *read* data *Q* dan W(*Q*) adalah *write* pada data *Q* untuk transaksi terkait.
Asumsikan nilai dari tiap data di awal A=10; B=20; C=0; D=30

T1	T2	T3
R(A)		
	R(A)	
		R(B)
R(B)		
		C:=50
		W(C)
	C:=35	
	W(C)	
R(C)		
commit		
	R(D)	
		B:=B-0.1*C
		W(B)
		commit
	D:=0.2*D+A	
	W(D)	
	commit	

Tuliskan langkah-langkah untuk mengeksekusi jadwal di atas sampai semua transaksi tuntas dieksekusi dengan ***snapshot isolation*** (dengan *first-committer wins*) dan jelaskan setiap tahapan eksekusi, termasuk versi data yang dihasilkan dan nilai dari tiap data. Sebutkan dan jelaskan apa yang terjadi pada tiap transaksi: apakah *abort* atau *commit*. Jika terjadi *abort*, jelaskan bagaimana proses *rollback* dilakukan. Sebutkan pula versi nilai data di akhir seluruh transaksi.

Snapshot Isolation (First-Committer Wins)

Content saat start:
A0 = 10
B0 = 20
C0 = 0
D0 = 30

T1	T2	T3	
R(A)			A0 = 10

	R(A)		A0 = 10
		R(B)	B0 = 20
R(B)			B0 = 20
		C := 50	
		W(C)	C3 = 50
	C := 35		
	W(C)		C2 = 35
R(C)			C0 = 0
commit			commit sukses
	R(D)		D0 = 30
		B := B - 0.1 * C	
		W(B)	B3 = 20 - 0.1 * 50 = 25
		commit	commit sukses, C3 = 50, B3 = 25
	D := 0.2 * D + A		D2 = 0.2 * 30 + 10 = 16
	W(D)		D2 = 10
	commit		commit gagal, serialization error → T2 abort dan rollback
	R(A)		A0 = 10
	C := 35		

	W(C)		C2 = 35
	R(D)		D0 = 30