NOMOR 1

S : T1:r(x), T2:w(x), T3:r(x), T1:r(y), T4:r(z), T2:w(y), T1:r(v), T3:w(v), T4:r(v), T4:w(y), T5:w(y), T5:w(z)

A. Apakah penjadwalan tersebut merupakan penjadwalan yang view-serializable? Jelaskan dan buktikan!

T1	T2	T3	T4	T5
R(X)				
	W(X)			
		R(X)		
R(Y)				
			R(Z)	
	W(Y)			
R(V)				
		W(V)		
			R(V)	
			W(Y)	
				W(Y)
				W(Z)

Sebuah schedule dapat dikatakan view serializable jika schedule tersebut view equivalent dengan beberapa schedule serial. Penjadwalan view-serializable adalah penjadwalan yang hasil akhirnya dapat diperoleh dari penjadwalan serial yang setara, tanpa memperhatikan urutan operasinya, asalkan setiap transaksi melihat tampilan data yang sama seperti dalam jadwal serial.

Langkah-langkah untuk menentukan view-serializability:

1) Initial Reads:
   Transaksi yang membaca nilai awal dari suatu item data dalam penjadwalan serial harus tetap membaca nilai yang sama dalam penjadwalan.
2) Final Writes:
   Transaksi yang menulis nilai akhir dari suatu item data dalam penjadwalan serial harus tetap menulis nilai akhir yang sama dalam penjadwalan.
3) Update Writes:
   Jika transaksi Ti membaca nilai item data yang telah diubah oleh transaksi Tj, maka transaksi Tj harus terjadi sebelum transaksi Ti

Analisa

1) Initial Reads:

T1 membaca x pertama kali.
T3 membaca x setelah T2 menulis x.
T1 membaca y pertama kali.
T4 membaca z pertama kali.
T1 membaca v pertama kali.
T4 membaca v setelah T3 menulis v.

2) Final Writes:
   T5 menulis y terakhir.
   T5 menulis z terakhir.
3) Update Writes:
   T2 menulis x setelah membaca nilai awal.
   T3 menulis v setelah membaca nilai awal.
   T4 menulis y setelah membaca nilai sebelumnya.
   T5 menulis y dan z terakhir.

Kesimpulan:
Penjadwalan ini adalah view-serializable karena semua kondisi untuk view-serializability terpenuhi. Hasil akhir dari transaksi dalam jadwal ini dapat diperoleh dengan urutan serial yang setara, tanpa mengubah tampilan data yang dilihat oleh setiap transaksi.

.

B. Apakah penjadwalan tersebut merupakan penjadwalan yang conflict-serializable? Jelaskan dan buktikan!

Penjadwalan conflict-serializable adalah penjadwalan yang dapat diubah menjadi jadwal serial dengan mempertahankan urutan konflik. Dua operasi dikatakan berkonflik jika memenuhi tiga kondisi:

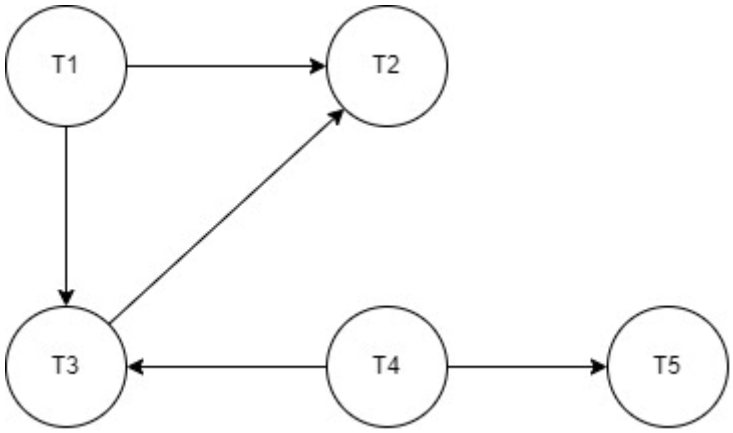
1) Mereka dioperasikan oleh transaksi yang berbeda.
2) Mereka mengakses item data yang sama.
3) Setidaknya satu dari operasi adalah operasi tulis (write).

Langkah-langkah untuk menentukan conflict-serializability:

1) **T1 (x) berkonflik dengan T2 (x)**:
   (Read-Write Conflict) T1 membaca nilai x, kemudian T2 menulis nilai x. Konflik ini terjadi karena operasi baca dan tulis dilakukan pada item data yang sama oleh transaksi yang berbeda.
2) **T3 (x) berkonflik dengan T2 (x)**:
   (Read-Write Conflict) T3 membaca nilai x, kemudian T2 menulis nilai x. Konflik ini terjadi karena operasi baca dan tulis dilakukan pada item data yang sama oleh transaksi yang berbeda.
3) **T1 (y) berkonflik dengan T2 (y)**:
   (Read-Write Conflict) T1 membaca nilai y, kemudian T2 menulis nilai y. Konflik ini terjadi karena operasi baca dan tulis dilakukan pada item data yang sama oleh transaksi yang berbeda.
4) **T1 (v) berkonflik dengan T3 (v)**:

(Read-Write Conflict) T1 membaca nilai v, kemudian T3 menulis nilai v. Konflik ini terjadi karena operasi baca dan tulis dilakukan pada item data yang sama oleh transaksi yang berbeda.

5) **T4 (y) berkonflik dengan T5 (y)**:
   (Write-Write Conflict) T4 dan T5 keduanya menulis nilai y. Konflik ini terjadi karena operasi tulis dilakukan pada item data yang sama oleh transaksi yang berbeda.
6) **T4 (z) berkonflik dengan T5 (z)**:
   (Read-Write Conflict) T4 membaca nilai z, kemudian T5 menulis nilai z. Konflik ini terjadi karena operasi baca dan tulis dilakukan pada item data yang sama oleh transaksi yang berbeda.
7) **T4 (v) berkonflik dengan T3 (v)**:
   (Read-Write Conflict) T4 membaca nilai v setelah T3 menulis nilai v. Konflik ini terjadi karena operasi baca dan tulis dilakukan pada item data yang sama oleh transaksi yang berbeda.



Pada graph konflik di atas, tidak ada siklus yang terlihat. Ini berarti bahwa penjadwalan ini adalah conflict-serializable.

Kesimpulan:
Penjadwalan tersebut adalah conflict-serializable karena graph konflik yang dihasilkan bebas dari siklus. Kita dapat menyusun ulang transaksi sesuai dengan urutan yang tertera pada graph konflik untuk menjaga semua konflik tanpa menyebabkan deadlock atau ketidaksesuaian.

NOMOR 2

T31: read(A); read(B); if A = 0 then B := B + 1; write(B);

T32: read(B); read(A); if B = 0 then A := A + 1; write(A);

A. Tambahkan perintah lock dan unlock pada kedua transaksi tersebut sehingga memenuhi two-phase locking protocol.

Two-Phase Locking Protocol adalah metode untuk memastikan serializability dalam penjadwalan transaksi dengan menggunakan kunci. Protokol ini memiliki dua fase:

1) Growing Phase: Transaksi hanya dapat mengakuisisi kunci dan tidak dapat melepaskan kunci.

   Pada fase ini, transaksi mengakuisisi kunci untuk semua item data yang akan diakses. Misalnya, dalam T31, kunci pada A dan B diakuisisi sebelum melakukan operasi baca dan tulis. Dalam T32, kunci pada B dan A diakuisisi sebelum melakukan operasi baca dan tulis.

2) Shrinking Phase: Transaksi hanya dapat melepaskan kunci dan tidak dapat mengakuisisi kunci.

   Pada fase ini, transaksi mulai melepaskan kunci setelah semua operasi baca dan tulis selesai dilakukan. Setelah operasi selesai, kunci pada B dan A dilepaskan dalam urutan terbalik dari saat mengakuisisinya untuk memastikan tidak ada kunci yang masih dipegang saat transaksi berakhir.

T31	T32
Lock(A)	
S(A)	
R(A)	
Lock(B)	
R(B)	
	Lock(B)
	S(B)
	R(B)
if A = 0 then B := B + 1	Lock(A)
X(B)	S(A)
W(B)	R(A)
Unlock(B); Unlock(A)	if B = 0 then A := A + 1
	X(A)
	W(A)
	Unlock(A); Unlock(B)

B. Apakah eksekusi dari kedua transaksi tersebut akan mengakibatkan deadlock? Jelaskan dan buktikan!

Deadlock terjadi ketika dua atau lebih transaksi saling menunggu satu sama lain untuk melepaskan kunci yang diperlukan untuk melanjutkan eksekusi. Deadlock dapat terjadi jika:

1) Transaksi T1 mengunci item A dan menunggu item B, sementara transaksi T2 mengunci item B dan menunggu item A.
2) Tidak ada transaksi yang dapat melanjutkan eksekusi karena keduanya saling menunggu.

Terjadi deadlock karena

T31 mengunci A (`lock(A)`) dan T32 mengunci B (`lock(B)`).
T31 mencoba mengunci B tetapi harus menunggu karena T32 telah mengunci B.
T32 mencoba mengunci A tetapi harus menunggu karena T31 telah mengunci A.

1) **T31 mengunci A dan menunggu B:** T31 sudah mengunci A dan tidak dapat melanjutkan eksekusi karena B dikunci oleh T32.
2) **T32 mengunci B dan menunggu A:** T32 sudah mengunci B dan tidak dapat melanjutkan eksekusi karena A dikunci oleh T31.

Deteksi dan Penyelesaian Deadlock oleh DBMS:

DBMS menggunakan algoritma deteksi dan resolusi deadlock untuk mengatasi situasi deadlock. Dua algoritma umum yang digunakan adalah:

1) **Wound-Wait Protocol:**
   Jika transaksi yang lebih tua (berdasarkan timestamp) meminta kunci yang dipegang oleh transaksi yang lebih muda, transaksi yang lebih muda akan di-bunuh (rollback) dan transaksi yang lebih tua akan memperoleh kunci.
   Contoh: Jika T31 dimulai lebih dahulu, maka T32 akan di-bunuh dan T31 akan memperoleh kunci B.
2) **Wait-Die Protocol:**
   Jika transaksi yang lebih muda meminta kunci yang dipegang oleh transaksi yang lebih tua, transaksi yang lebih muda akan di-bunuh (rollback). Jika transaksi yang lebih tua meminta kunci yang dipegang oleh transaksi yang lebih muda, transaksi yang lebih tua akan menunggu.
   Contoh: Jika T32 dimulai lebih dahulu dan menunggu kunci A yang dipegang oleh T31, maka T32 akan di-bunuh.

NOMOR 3

Phase Analysis:
LSN 20 Tambahkan (T1,20) ke TT dan (P1,20) ke DPT
LSN 30 Tambahkan (T2,30) ke TT dan (P2,30) ke DPT
LSN 40 Tambahkan (T3,40) ke TT dan (P3,40) ke DPT
LSN 50 Ubah status T2 ke C
LSN 60 Ubah (T3,40) ke (T3,60)
LSN 70 Remove T2 dari TT
LSN 80 Ubah (T1,20) ke (T1,70) dan Tambahkan (P5,70) ke DPT
LSN 90 No action
Pada akhir phase analysis, Tabel transaksi terdiri dari: (T1,80), and (T3,60).
Table dirty page terdiri dari: (P1,20), (P2,30), (P3,40), and (P5,80)

Phase REDO
Phase Redo dimulai dari LSN20 (minimum recLSN di DPT).
LSN 20 mengecek apakah P1 pageLSN > 10 atau tidak.
LSN 30 Redo perubahan P2
LSN 40 Redo perubahan P3
LSN 50 No action
LSN 60 Redo perubahan P2
LSN 70 No action
LSN 80 Redo perubahan P5
LSN 90 No action

Phase UNDO
LSN 80 Undo perubahan P5. tambahkan CLR: Undo T1 LSN 80, set undonextLSN = 20. Tambahkan 20 ke ToUndo.
LSN 60 Undo perubahan pada P2. Tambahkan CLR: Undo T3 LSN 60, set undonextLSN = 40. Tambahkan 40 ke ToUndo.
LSN 40 Undo perubahan pada P3. Tambahkan CLR: Undo T3 LSN 40, T3 end
LSN 20 Undo perubahan pada P1. Tambahkan CLR: Undo T1 LSN 20, T1 end

Log setelah Recovery:
LSN 00 begin checkpoint
LSN 10 end checkpoint
LSN 20 update: T1 writes P1
LSN 30 update: T2 writes P2
LSN 40 update: T3 writes P3
LSN 50 T2 commit, prevLSN = 30
LSN 60 update: T3 writes P2, prevLSN = 40
LSN 70 T2 end, prevLSN = 50
LSN 80 update: T1 writes P5, prevLSN = 20

LSN 90 T3 abort, prevLSN = 60
LSN 100 CLR: Undo T1 LSN 80, undonextLSN= 20
LSN 110 CLR: Undo T3 LSN 60, undonextLSN= 40
LSN 120,125 CLR: Undo T3 LSN 40, T3 end.
LSN 130,135 CLR: Undo T1 LSN 20, T1 end.