Unidad III: Transacciones, Concurrencia y Recuperación.

Ejercicio 1

a)

<T3,start> de la linea 0
<T1,start>, de la linea 3
<T3,tax,2,3> de la linea 6
<T3,commit> de la linea 7
<T2,start> de la linea 8
<T2,tax,3,4> de la linea 12
<T2,commit> de la linea 13
<checkpoint start> de la linea 14, T1 todavía está activa
<T1,salary,1,2> de la linea 17
<checkpoint end> de la linea 18
<T1,commit> de la linea 19

b)

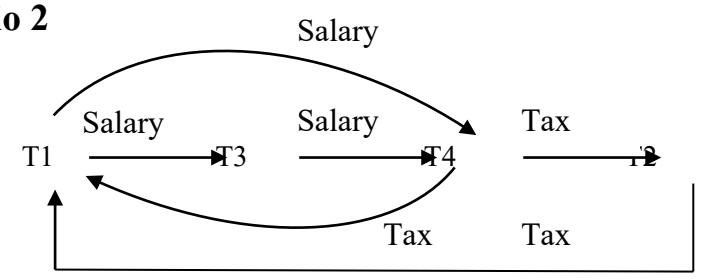
a. T1
b. T3

c)

a. T1
b. Ninguna

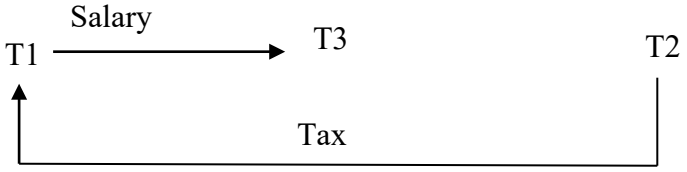
Ejercicio 2

a)



b) Ninguno, ya que existe un ciclo en este grafo.

c)



d) T2,T1,T3

Ejercicio 3

Para el registro a):

- No puede utilizarse para rehacer la base de datos (redo) porque el valor de D debería ser o 450 (T1 hace commit antes del checkpoint) o 18 que es escrito por T2.

Para el registro b):

- Si corresponde con un redo log, porque A puede ser 5,13 o 11, B puede ser 40, C puede ser 35 y D puede ser 4 y E 6 o 18, que corresponden con los posibles valores de la tabla.

Ejercicio 4

a) W 4 ó 7 , X 28 ó 17 , Y 5, Z 10

b) T2

c) W 4 ó 7 , X 14 ó 28 ó 17 , Y 15 ó 5 , Z 10 ó 20. Las transacciones sería T1 y T2.

Considerar la regla siguiente: Checkpoint obliga a volcar los bloques de memoria intermedia modificados en disco antes del Start Checkpoint, mientras en otra situación si el dato ha salido en memoria, puede que si o no (el sistema gestor de base de datos decidirá).

Ejercicio 5

La planificación anterior no es secuenciable en cuanto a conflictos ya que no se puede producir la planificación secuencial intercambiando pares de instrucciones no conflictivas de las transacciones. Como se puede ver en la planificación anterior, T1, T2 y T3 ejecutan el comando escribir(A). No se pueden intercambiar porque cambiará el valor que tiene A después de la ejecución de las tres transacciones. Por lo tanto, la planificación no es secuenciable en cuanto a conflictos.

La planificación P1 es secuenciable en cuanto a vistas (con respecto a la planificación secuencial P) si se cumplen tres condiciones, para todo elemento de datos Q:

(a) Si la transacción Ti lee el valor inicial de Q en P, debería hacerlo en P1.
(b) Si la transacción Tj ejecutar leer(Q) y el valor lo ha producido en P la transacción Ti (si hay), debería pasar lo mismo en P1.
(c) La transacción (si hay) que realice la última operación escribir(Q) en P debería hacerlo en P1.

La planificación secuencial para las transacciones T1, T2 y T3 sería:

T1	T2	T3
	leer (A) escribir (A)	
		escribir (A) leer (B)
leer (B) leer (A) escribir (A)		

Considerando las tres condiciones:
Para A:

a) T2 lee el valor inicial de A,
b) T1 lee el valor producido por T3,

c) T1 escribe el último valor.

Para B:

a) T1 y T3 leen los valores iniciales.

Ejercicio 6

a)

El protocolo de bloqueo en dos fases consiste en:

1. Fase de crecimiento: una transacción puede obtener bloqueos pero no liberarlos.
2. Fase de decrecimiento: una transacción puede liberar bloqueos pero no obtener ninguno.

Según este protocolo se puede ver una posible solución (puede haber otras):

T1		T2		T3	
Fase	Instrucción	Fase	Instrucción	Fase	Instrucción
1	Bloquear_C(A) Leer(A) Bloquear_X(B) Leer(B) C=A+B (C=1500) C=C+A*0.1 (C=1600) A=A-A*0.1 (A=900) Subir(A) Escribir(A) Bloquear_X(C)				
2	Desbloquear(A)				
		1	C=0 Bloquear_X(A) Leer(A)		
	B=B-C Escribir(C); Desbloquear(C)				
			A=C+1000 Escribir(A);		
		2	Bajar(A)		
				1	Bloquear_C(A) Leer(A) Bloquear_X(C) Leer(C) C=C-A Escribir(C)
	D=A-1 Escribir(B) Desbloquear(B)				
			Desbloquear(A)		
				2	Desbloquear(C) Desbloquear(A)

b)

Para ver la estructura del registro histórico solamente hay que considerar los cambios que producirá cada transacción cada transacción en la base de datos:
A = 1000, B = 500 y C = 2000

T1	T2	T3	R.H	B.D.
			<T1 iniciada>	(A=1000,B=500,C=2000)

bloquear_C(A);
leer (A);
bloquear_C(B);
leer (B);
C:=A+B; (C=1500)
C:=C+(A*0.1);
(C=1600)
A:=A-(A*0.1);
(A=900)
subir(A);
escribir (A);				

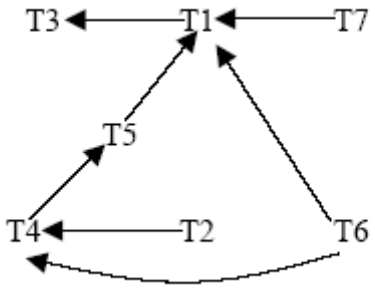
			<T1,A,1000,900>	A=900
bloquear_X(C); desbloquear(A);				
			<T2,iniciada>	
	C:=0; bloquear_X(A); leer (A); 900			
escribir (C);			<T1,C,2000,1600>	C=1600
desbloquear(C);				
	A:=C+1000; escribir(A);			
			<T2,A,900,1000>	A=1000
	bajar(A);			
			<T2,iniciada>	
		bloquear_C(A); leer (A); 1000		

		bloquear_X(C);
		leer(C); 1600
		C:=C-A;
		escribir(C);		

			<T3,C,1600,600>	C=600
B:=B-C; -1100 D:=A-1; -- escribir(B);				
			<T1,B,500,-1100>	B=-1100
desbloquear(B);				
			<T1,comprometida>	
	desbloquear(A);			
			<T2,comprometida>	
		desbloquear (C); desbloquear(A);		
			<T3,comprometida>	

Ejercicio 7

a)



b)

Tres posibles modificaciones que producirían interbloqueos serían:

- que transacción T1 está esperando datos que tiene transacción T2
- que transacción T3 está esperando datos que tiene transacción T4
- que transacción T3 está esperando datos que tiene transacción T7

Ejercicio 8

a) Dado que A = 1.100.000, B = 200 y D = 3.500.000

T1	T2	T3	Registro Histórico	Base de Datos
		Punto de revisión	Checkpoint	
leer (A) (A=1.100.000)			<T1 iniciada>	
C= 0.5				
		leer (D) (D=3.500.000)	<T3 iniciada>	
A=A*C (550.000)				
	leer (B)(B=200)		<T2 iniciada>	
		leer (B) (B=200)		
	leer (A) (A=1.100.000)			
		D=D + (B*1000) (D=3.700.000)		
		escribir(D)	<T3, D, 3.700.000,3.500.000> <T3 comprometida>	D = 3.700.000
	B= A+ B (B=1.100.200)			
	escribir (B) (1.100.200)		<T2, B, 1.100.200, 200> <T2 comprometida>	B = 1.100.200
				fallo
		Fallo		
escribir (A) (550.000)			<T1, A, 550.000,1.100.000> <T1 comprometida>	A=550.000

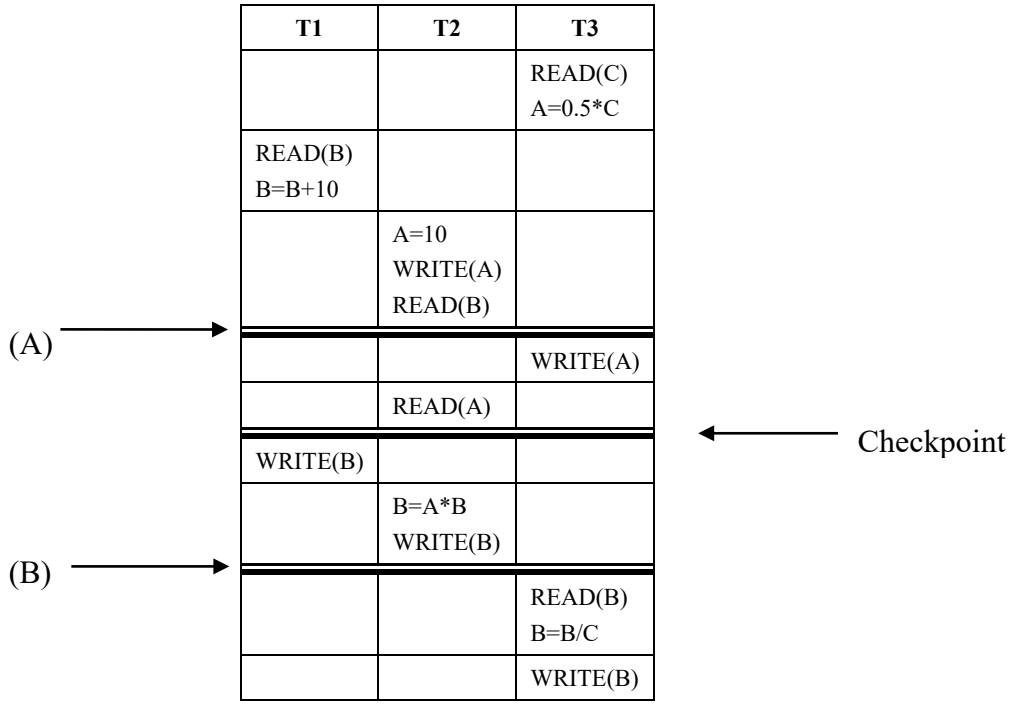
Por error, en el registro del log se encuentra valor nuevo, valor antiguo y deberá ser al revés.

b) En el caso que hay un fallo (donde está indicado en la tabla), hay que empezar el proceso de recuperación desde el punto de revisión, porque lo que había ocurrido antes ya está incorporado en una copia de la base de datos en memoria estable. Aquí, las dos transacciones T2 y T3 se han comprometido y hay que rehacerles. Transacción T1 no se ha

comprometido, pero hay que deshacerla aunque hasta el momento no ha modificado la base de datos.

Ejercicio 9

Se tiene la siguiente planificación, donde inicialmente A=100, B=200 y C=300. Suponer que cada transacción acaba justo con la última sentencia de ella.



1. Mostrar el estado del registro histórico para esta planificación utilizando los esquemas (R) redo y (U-R) undo-redo log

(R)	
LOG:	MEMORIA:
<T3 START>	
<T1 START>	
<T2 START>	
<T2 A, 10>	
(A)	
<T3, A, 150>	
<Start Checkpoint T1,T2,T3>	
<End Checkpoint>	
<T1, B, 210>	
<T1 COMMIT>	
	B = 210
<T2, B, 2000>	
<T2 COMMIT>	
	A = 10

(U-R)	
LOG:	MEMORIA:
<T3 START>	
<T1 START>	
<T2 START>	
<T2, A, 100, 10>	
	A = 10
(A)	
<T3, A, 10, 150>	
	A = 150
<Start Chekpoint T1,T2,T3>	
<End Checkpoint>	
<T1, B, 200, 210>	
	B = 210
<T1 COMMIT>	
<T2, B, 210, 30000>	
	B = 30000

B = 2000	
(B)	
<T3, B, 6.67>	
<T3 COMMIT>	
	A = 150
	B = 6.67

<T2 COMMIT>
(B)
<T3, B, 30000, 100>
B = 100
<T3 COMMIT>

2. Si el sistema se cae en el punto (A), para cada caso ¿Qué estado tendría la base de datos en memoria y en disco y cómo se recuperaría el sistema?

   De los diagramas anteriores de MEMORIA se tiene además en DISCO:

(R)	
MEMORIA:	DISCO:
A = 100	A = 100
B = 200	B = 200
C = 300	C = 300
Como no se ha escrito nada en disco (por ser diferida) y no haberse consolidado ninguna transacción:	
• NO hay que hacer nada	

(U-R)	
MEMORIA:	DISCO:
A = 10	A = 100 ó 10
B = 200	B = 200
C = 300	C = 300
Como se ha podido escribir en disco (por ser inmediata) y no haberse consolidado ninguna transacción:	
• Hay que deshacer T2 (y formalmente se debería deshacer T1 y T3 también, aunque no han modificado nada)	

3. Si el sistema se cae en el punto (B), para cada caso ¿Qué estado tendría la base de datos en memoria y en disco y cómo se recuperaría el sistema?

   De los diagramas anteriores de MEMORIA se tiene además en DISCO:

(R)	
MEMORIA:	DISCO:
A = 10	A = 100 ó 10
B = 2000	B = 200 ó 210 ó 2000
C = 300	C = 300
Como se ha podido escribir (al ser diferida) y haberse consolidado T1 y T2:	
• Hay que rehacer T1 y T2	

(U-R)	
MEMORIA:	DISCO:
A = 150	A = 150
B = 30000	B = 200 ó 210 ó 30000
C = 300	C = 300
Como se ha podido escribir (al ser inmediata), no haberse consolidado T3 y haberse consolidado T1 y T2:	
• Hay que deshacer T3	
• Hay que rehacer T1 y T2	

4. Si no estuviese el checkpoint presente, qué efecto tendría en los puntos (A) y (B)

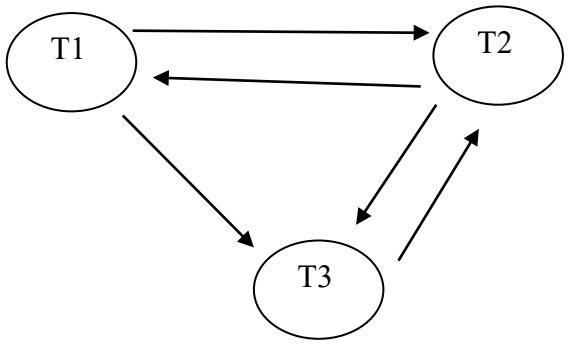
   De los diagramas anteriores de MEMORIA se tiene además en DISCO:

(R)	
MEMORIA:	DISCO:
A = 100	A = 100
B = 200	B = 200
C = 300	C = 300
Pero por ser diferida y no haberse consolidado ninguna transacción:	
• NO se vuelca nada a disco	
• Rehacer T1 y T2.	
• Los valores son los mismos que el apartado 2.	

(U-R)	
MEMORIA:	DISCO:
A = 150	A = 100 ó 10 ó 150
B = 30000	B = 200 ó 210 ó 30000
C = 300	C = 300
Pero por ser inmediata y no haberse consolidado ninguna transacción:	
• Cualquier posible valor en RAM + valor inicial.	
• Hay que deshacer T3 y rehacer T1 y T2.	

5. ¿Es secuenciable en conflictos y en vistas la planificación anterior? ¿Por qué? Si lo es, ¿Cuáles serían las posibles planificaciones anteriores?

Para ver si es secuenciable en conflictos, se construye el diagrama de precedencia y se mira si hay ciclos. Nos fijamos en los writes de los elementos y las instrucciones que están en conflicto con ellos. El diagrama es:



Como se puede ver hay ciclos, luego no es secuenciable en conflictos. Para ver si es en vistas hay que mirar si se puede obtener una planificación serie que cumpla las tres reglas:

1. Si Ti lee el valor inicial de la base de datos, en la planificación serie equivalente Ti también lo lee.
2. Si Ti lee un valor escrito por Tj, en la planificación serie equivalente también se cumple.
3. Si Ti escribe el valor final de un elemento de datos, en la planificación serie equivalente también.

Estas tres reglas hay que verificarlas para los tres elementos de datos A,B y C, y en cuanto haya uno que no lo cumpla, la planificación ya no es secuenciable en vistas. Si nos fijamos en el elemento B, se puede ver que T3 escribe el valor final de la base de datos, luego T3 debe de ser la última ya que las tres transacciones escriben el valor de B. Y además T3 lee un valor escrito por T2, luego T2

debe de ir delante de T3, luego para el elemento B, el posible orden sería: T1,T2,T3, pero se puede ver que tanto T1 como T2 leen el valor inicial de la basde de datos, luego este orden hace que T2 no lea el valor inicial de B, luego no existe orden entre T1,T2 y T3 para que cumplan las tres condiciones → No es secuenciable en vistas.

6. ¿Tiene rollbacks en cascada la planificación anterior? ¿Por qué?

Suponiendo que después de la última instrucción de cada transacción se hace el commit, nos fijamos si alguna transacción lee un valor escrito por otra sin que haya realizado el commit la que escribe. Eso pasa entre T3 (escribe A) y T2 (lee A) sin que T3 haya hecho commit, luego tiene rollbacks en cascada

7. ¿Es recuperable la planificación anterior? ¿Por qué?

Para ello nos fijamos si una transacción lee un valor escrito por otra y nos fijamos cuando hacen los commit; si la que lee lo hace antes de la que escribe el valor, no es recuperable. Eso se cumple entre T3 (escribe A) y T2 (lee A) y T2 hace el commit antes que T3, luego n es recuperable.

8. Utilizando un protocolo de bloqueo de dos fases (normal). ¿Cuál sería el orden de ejecución de las transacciones?

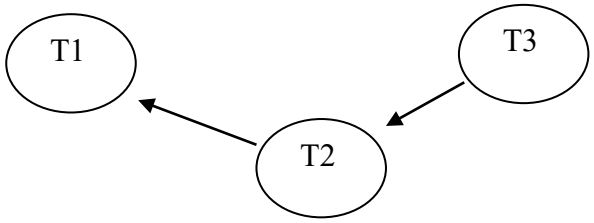
El protocolo de bloqueo de dos fases riguroso adquiere los bloqueos compartidos (si sólo lee el elemento de datos) o bloqueos exclusivos (si tiene que escribir el elemento de datos) antes de seguir con la operación correspondiente. Teniendo en cuenta eso, se tiene que construir una lista enlazada con los tipos de bloqueos pedidos y asignados de cada elemento de datos. Así se puede ver si la transacción continua, se bloquea o se produce un interbloqueo. Hay tres elementos de datos, luego la lista que se va construyendo es:

A→T2(Exclusivo,6)→T3(Exclusivo,8,se bloquea, no continua)

B→T1(Exclusivo,3)→T2(Exclusivo,7, se bloquea,no continua)

C→T3(Compartido,1)

Luego en este momento, T1 es la que está en ejecución, y T2 espera a T1, y T3 espera a T2. El grafo de espera es el siguiente en este momento:



No hay interbloqueo ya que no hay ciclo, y por lo tanto primero se ejecutará T1, cuando haga COMMIT, se sueltan sus bloqueos, comenzará T2 y T3 en espera, para cuando T2 haga COMMIT, comenzará T3. Luego el orden serie que se ha producido es T1, T2, T3

Ejercicio 10

El siguiente fragmento muestra las operaciones y los ***intentos*** de bloqueo que realizan una serie de transacciones (T1 a T6) sobre un conjunto de elementos de datos (A a E). Los pasos representados ***no*** indican que se adquiera el bloqueo, sino sólo que se intenta realizar el mismo (XLOCK significa bloqueo exclusivo y SLOCK significa bloqueo compartido):

Secuencia	Transacción	Acción
1	T1	XLOCK A
2	T2	XLOCK E
3	T1	SLOCK B
4	T4	SLOCK C
5	T5	SLOCK B
6	T3	SLOCK D
7	T3	XLOCK A
8	T6	XLOCK A
9	T4	SLOCK B
10	T2	XLOCK D
11	T1	SLOCK E
12	T5	READ B

Suponiendo que inicialmente no hay establecido ningún bloqueo en el sistema, determinar:

1. El grafo de espera de las transacciones.

   En este fragmento hay 6 transacciones que intentan obtener bloqueos sobre 5 elementos de datos (A, B, C, D, E). Para cada elemento de datos, el orden de petición de bloqueo es el siguiente:

   A → T1(X,1) → T3(X,7) → T6(X,8)

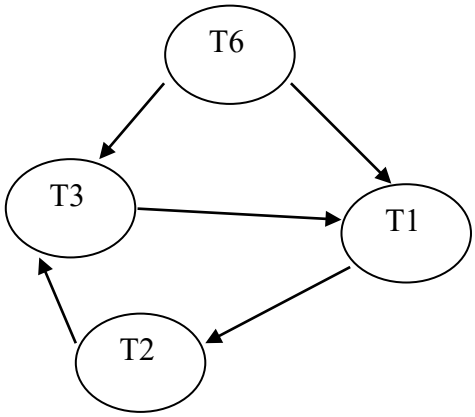
   B → T1(S,3) → T5(S,5) → T4(S,9)

   C → T4(S,4)

   D → T3(S,6) → T2(X,10)

   E → T2(X,2) → T1(S,11)

   Donde X indica bloqueo exclusivo y S bloqueo compartido, y el número indica la secuencia donde se realiza la petición. De lo anterior se puede deducir que:



2. Las transacciones que se encuentran bloqueadas al terminar dicho fragmento.

   Del diagrama anterior se obtiene que las transacciones bloqueadas son T1, T2, T3 y T6.

3. Las transacciones que no se encuentran bloqueadas.

De forma análoga, las transacciones que no se encuentran bloqueadas son T4 y T5.

4. Al final del fragmento anterior, ¿qué bloqueos serían concedidos a las transacciones y sobre qué elementos de datos?

T1 tendría concedido bloqueo X sobre A

T1 tendría concedido bloqueo S sobre B

T5 tendría concedido bloqueo S sobre B

T4 tendría concedido bloqueo S sobre B

T4 tendría concedido bloqueo S sobre C

T3 tendría concedido bloqueo S sobre D

T2 tendría concedido bloqueo X sobre E

5. ¿Existe alguna situación de interbloqueo (*deadlock*)?. Si se produce esa situación, ¿cómo se podría deshacer el mismo? Razonar la respuesta.

Hay un interbloqueo por el ciclo que hay entre T1, T2 y T3. Si se deshace una de las transacciones se rompe la situación de interbloqueo. Dado que T1 accede a 3 elementos de datos y T2 y T3 sólo a 2 elementos, a falta de otra información, las mejores candidatas son T2 ó T3.

Ejercicio 11

Considerar los planes P_1 y P_2 que se muestran a continuación:

P_1: $r_1(X)$, $r_2(Z)$, $r_1(Z)$, $r_3(X)$, $r_3(Y)$, $w_1(X)$, $w_3(Y)$, $r_2(Y)$, $w_2(Z)$, $w_2(Y)$, c_1, c_2, c_3

P_2: $r_1(X)$, $r_2(Z)$, $r_3(X)$, $r_1(Z)$, $r_2(Y)$, $r_3(Y)$, $w_1(X)$, c_1, $w_2(Z)$, $w_3(Y)$, $w_2(Y)$, c_3 ,c_2

Donde:

$r_n(A)$, indica la lectura de la transacción n del elemento de datos A,
$w_n(A)$, indica la escritura de la transacción n del elemento de datos A.
c_n, indica la operación COMMIT de la transacción n.

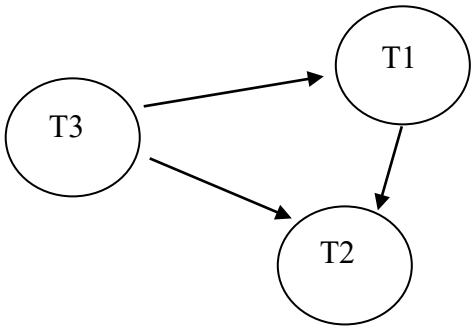
Justificar razonando:

a) Si cada plan es o no es serializable en cuanto a conflictos y en cuanto a vistas. En caso de ser serializables obtener los posibles planes equivalentes serie.

Para el primer plan P1

En este caso si realiza el grafo de precedencia se puede ver que no hay ciclos, luego es un plan serializable en cuanto a conflictos y por lo tanto también en cuanto a vistas.

	T1	T2	T3
1	R(X)		
2		R(Z)	
3	R(Z)		
4			R(X)
5			R(Y)
6	W(X)		
7			W(Y)
8		R(Y)	
9		W(Z)	
10		W(Y)	
11	C		
12		C	
13			C

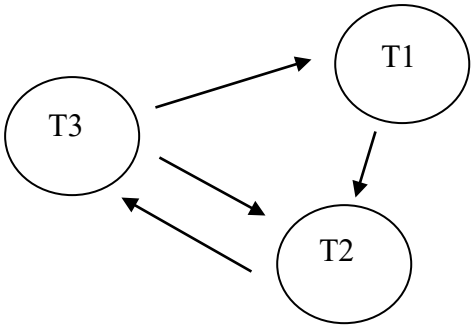


Posible plan serie: T3, T1, T2 en conflictos y vistas

Para el segundo plan P2

	T1	T2	T3
1	R(X)		
2		R(Z)	
3			R(X)
4	R(Z)		
5		R(Y)	
6			R(Y)
7	W(X)		
8	C		
9		W(Z)	
10			W(Y)
11		W(Y)	
12			C
13		C	

En este caso si realiza el grafo de precedencia se puede ver que hay ciclos, luego es un plan no serializable en cuanto a conflictos.



Para el caso de secuencialidad en vistas, hay que encontrar un plan serie S' de tal manera que se cumpla que en esa planificación una Ti lee el valor inicial de Q, también se hace en S', si se produce una lectura de un valor Q debido a una transacción, ocurra lo mismo en S' y que si una transacción escribe el último valor de Q, también se haga en S'. En este caso para el elemento Y no se cumple entre las transacciones T2 y T3

b) Si cada plan tiene o no tiene rollbacks en cascada.

No hay rollbacks en cascada si una transacción Tj lee un elemento de datos escrito por Ti, y el commit de Ti aparece antes que la lectura de Tj.

La planificación P1 tiene rollbacks en cascada ya que T2 lee un dato (Y) escrito por T3 antes del commit de T3.

La planificación P2 no tiene rollbacks en cascada ya que todas leen datos al principio sin ser producidos por escrituras de otras transacciones.

c) Si cada plan es o no es recuperable.

Un plan es recuperable si la transacción Tj lee un elemento escrito por Ti, el commit de Ti aparece antes que el de Tj. Por lo tanto, todo plan que no tiene rollbacks en cascada es recuperable.

El plan P2 es recuperable por no tener rollbacks en cascada, y el plan P1 no es recuperable ya que T2 lee un elemento escrito por T3 y el commit de T2 es anterior que el de T3.

Ejercicio 12

Considerar el siguiente log de una base de datos:

1	<START T1>
2	<T1, A, 22,17>
3	<START T2>
4	<START T3>
5	<T2,C,42,31>
6	<T3,E,9,14>
7	<T1,B,13,12>
8	<T1,A,17,8>
9	<T2,D,6,3>
10	<COMMIT T1>
11	<START T4>
12	<T4,F,7,5>
13	<T2,D,3,1>
14	<START CHECKPOINT (T2,T3,T4)>
15	<T2,C,31,2>
16	<T4,F,5,16>
17	<START 75>
18	<T3,E,14,15>
19	<T5,G,29,30>
20	<COMMIT T2>
21	<END CHECKPOINT>
22	<T5,G,30,32>
23	<ABORT T3>

24	<T4,H,22,11>
25	<COMMIT T5>
26	<COMMIT T4>

Considerar los siguientes escenarios de fallo:

a. El sistema falla justo antes de la escritura de la línea 9 en disco
b. El sistema falla justo antes de la escritura de la línea 16 en disco
c. El sistema falla justo antes de la escritura de la línea 20 en disco
d. El sistema falla justo antes de la escritura de la línea 21 en disco.
e. El sistema falla justo antes de la escritura de la línea 23 en disco.
f. El sistema falla justo antes de la escritura de la línea 25 en disco.
g. El sistema falla justo antes de la escritura de la línea 26 en disco.
h. El sistema falla justo después de la escritura de la línea 26 en disco.

a) ¿Qué valores contendrá la base de datos para cada elemento de datos A,B,C,D,E,F,G y H en disco después de que se realice con éxito la recuperación del sistema?.

En este caso estamos interesados en los valores de los elementos de datos después de una recuperación con éxito por parte del sistema gestor de base de datos. Según ello, cada elemento contendrá el valor original ó el último valor escrito por la transacción que ha realizado con éxito un commit (escritura del registro commit en el log). Los valores obtenidos estarán reflejados tanto en memoria como en disco. Se ha realizado una recuperación del sistema.

Según eso las acciones a realizar por la base de datos es:

a) Deshacer T3,T1,,T2

b) Deshacer T2,T3,T4 y rehacer T1

c) Deshacer T2,T3, T4, T5 y rehacer T1

d) Deshacer T3, T4, T5 y rehacer T1,T2

e) Deshacer T3, T4, T5 y rehacer T2

f) Deshacer T3, T4, T5 y rehacer T2

g) Deshacer T3, T4 y rehacer T2,T5

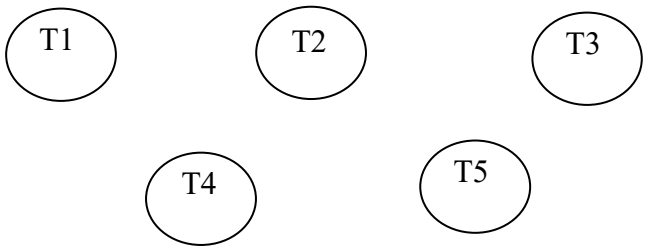
h) Deshacer T3 y rehacer T2,T5,T4

Según eso se puede obtener esta tabla:

Fallo	A	B	C	D	E	F	G	H
a)	22	13	42	6	9	7	29	22
b)	8	12	42	6	9	7	29	22
c)	8	12	42	6	9	7	29	22
d)	8	12	2	1	9	7	29	22
e)	8	12	2	1	9	7	29	22
f)	8	12	2	1	9	7	29	22
g)	8	12	2	1	9	7	32	22
h)	8	12	2	1	9	16	32	11

b) Obtener el grafo de precedencia del fragmento del log anterior y todas las planificaciones equivalentes en serie.

En este caso se puede ver que el grafo de precedencia no tiene arcos ya que las transacciones no acceden a ningún elemento en común. Por lo tanto son nodos aislados.



Según lo anterior, se podría dar cualquier combinación posible en el orden de las transacciones para formar planificaciones equivalentes serie en conflictos. Por lo tanto, habría: 5! Planificaciones o lo que es lo mismo 120 planificaciones posibles.