

Bases de Datos II

Tema III.- Gestión de Transacciones

Índice

- Transacción Atómica
 - Unidad lógica de procesamiento de BD
 - Propiedades deseables (Atomicidad, Consistencia, Aislamiento, Durabilidad)
 - Planes (historias) de transacciones, Recuperabilidad
- Control Concurrency
 - Varias transacciones concurrentes se interfieren
 - Seriabilidad: secuencias de ejecución correctas
- Recuperación ante fallos por:
 - error de transacción (overflow, división por 0, interrupción del usuario, ...)
 - excepciones de transacción (NO \exists datos, saldo insuficiente, ...)
 - control concurrencia (viola la seriabilidad, bloqueo mortal, ...)
 - caída del sistema
 - fallo del disco
 - catástrofes físicas

Procesamiento de Transacciones

- Transacción Atómica
 - Unidad lógica de procesamiento de BD
 - Ejecución de un programa que LEE o ESCRIBE en la BD

Operaciones de Acceso:

leer_elemento (X) \Rightarrow X (disco) \rightarrow memoria \rightarrow X (variable)

escribir_elemento (X) \Rightarrow

X (disco) \rightarrow memoria \rightarrow X (disco)
X (variable) \rightarrow memoria

Ejemplo:

T1

leer_elemento (X)

X := X - N

escribir_elemento(X)

leer_elemento(Y)

Y := Y + N

escribir_elemento(Y)

Conjunto lectura = {X, Y}

Conjunto escritura = {X, Y}

¿Por qué se necesita Control de Concurrency?

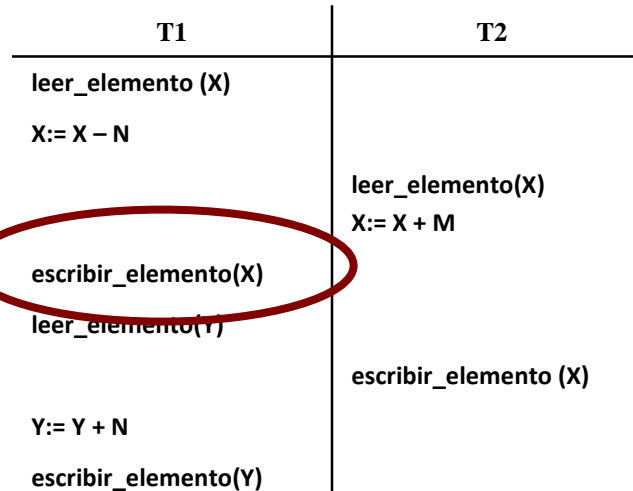
EJ: RESERVAS DE VUELOS:

T1: transfiere N reservas del vuelo X al vuelo Y

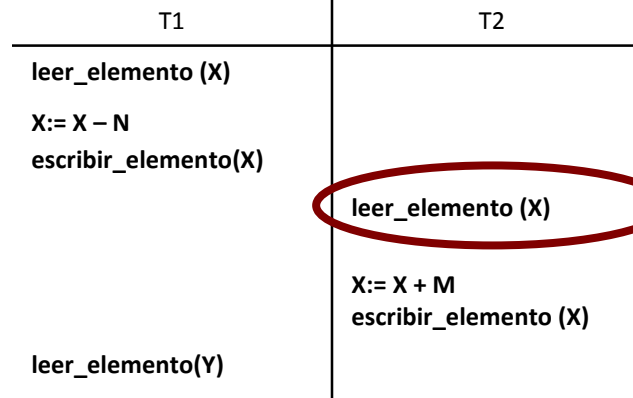
T2: añade M reservas al vuelo X

1) Actualización perdida

Se pierde!!



2) Actualización temporal (lectura sucia)

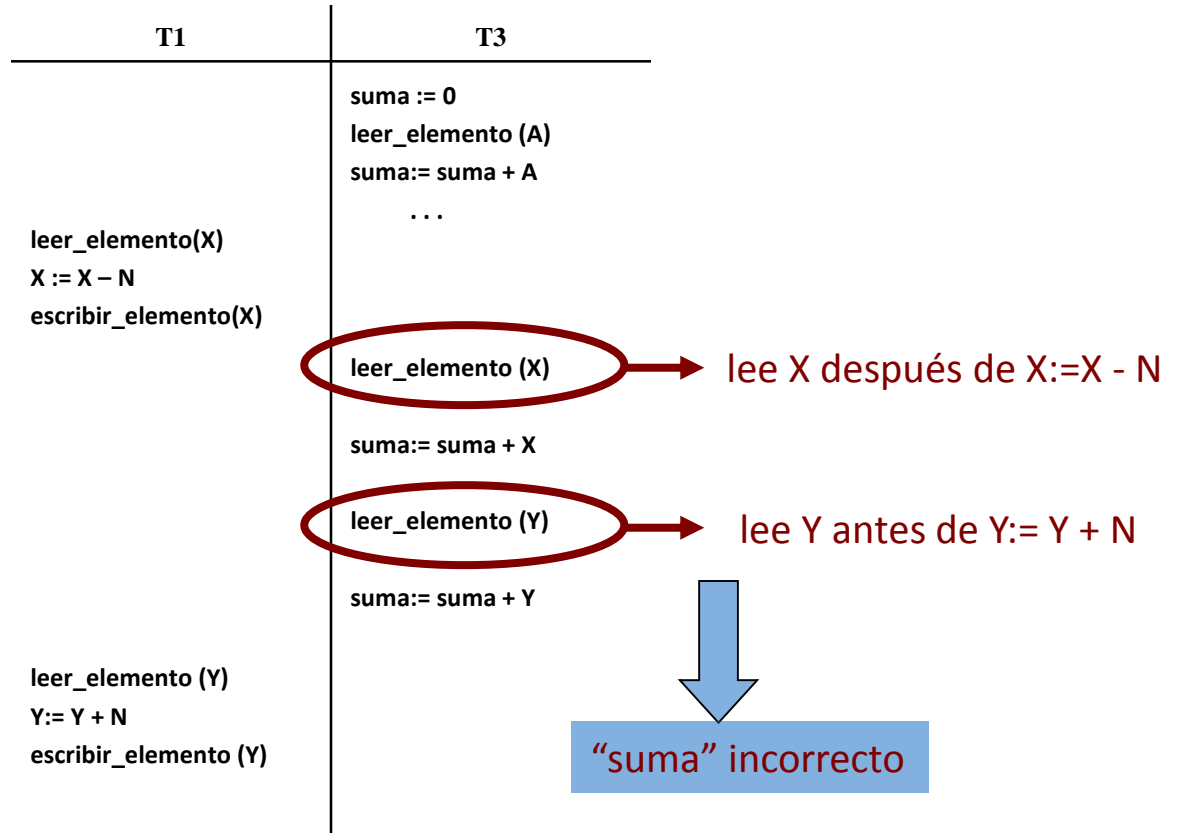


dato sucio

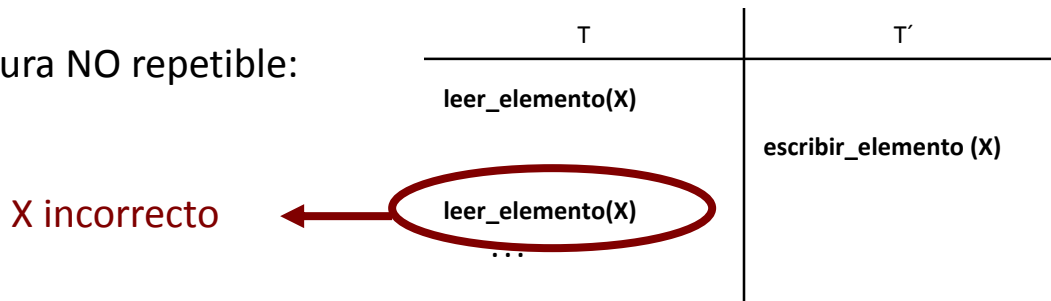
T1 falla \Rightarrow X debe volver a su antiguo valor
PERO T2 ya ha leído X

¿Por qué se necesita Control de Concurrency?

3) Resumen incorrecto

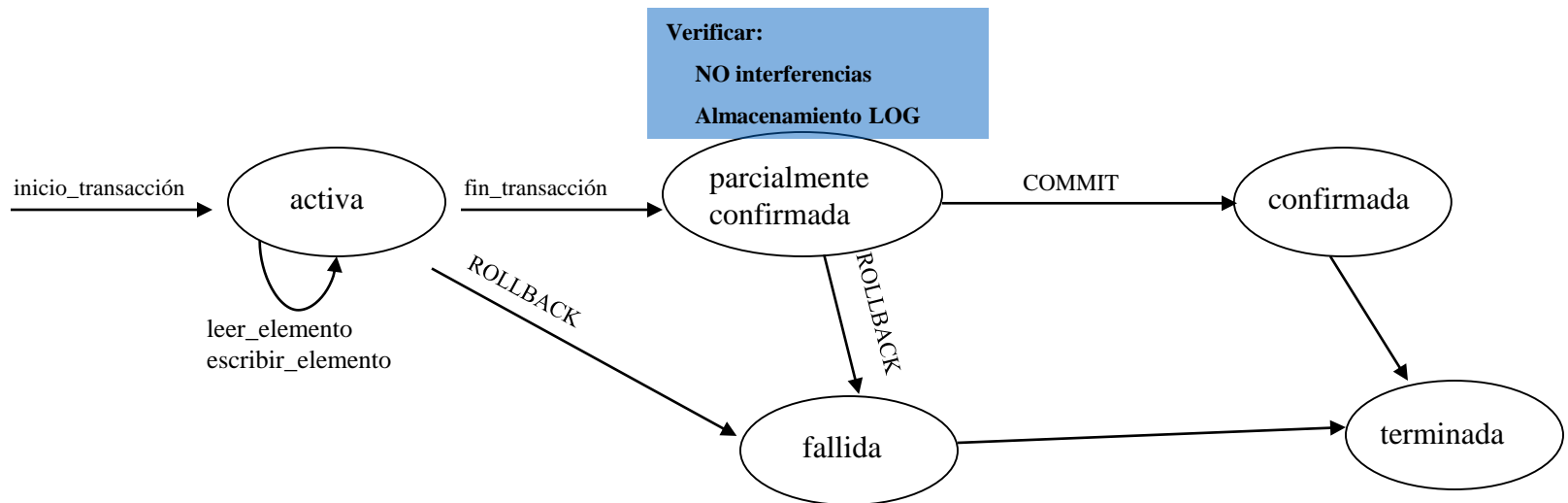


4) Lectura NO repetible:



Estado de una transacción

- Transacción Atómica
 - Ejecución de un programa que LEE o ESCRIBE en la BD
 - Operaciones
 - Inicio de Transacción:
 - **inicio_transacción**
 - Operaciones de Acceso:
 - **leer_elemento(X)**
 - **escribir_elemento(X)**
 - Terminación de Transacción:
 - **fin_transacción**
 - **confirmar (COMMIT)** terminó con éxito \Rightarrow confirmar cambios
 - **deshacer (ROLLBACK)** terminó sin éxito \Rightarrow cancelar cambios



Archivo de Log

- = {registros Log} que se mantienen temporalmente en memoria

Tipos de Registros Log:

[inicio_transacción, T]

inicio ejecución de T

[escribir_elemento, T, X, *valor_ant*, *valor_nuevo*]

cambio de X(disco)

[leer_elemento, T, X]

lectura del elemento X (disco)

[confirmar, T]

∀ escrituras se han ejecutado con éxito

∀ reg. Log de $T \rightarrow A$. Log (forzar la escritura)
se abortó la transacción T

[abortar, T]

∀ reg. Log de $T \rightarrow A$. Log (forzar la escritura)

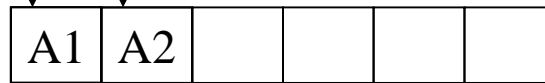
Funcionamiento de la memoria Oracle

ZONA DE MEMORIA (SGA)

Buffer Datos



Buffer Rollback

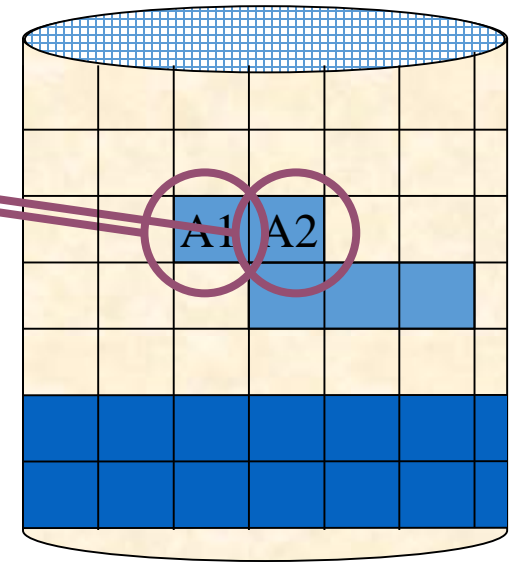


Buffer de Log

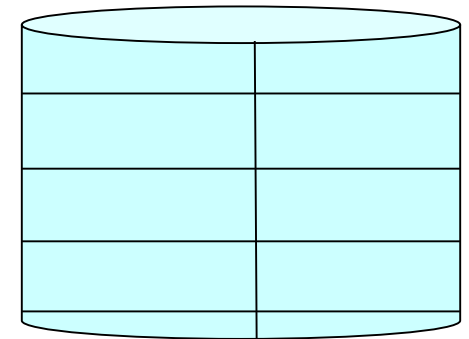
Tr.	Fecha	Estado	Dirección	Viejo	Nuevo
T1		En curso	tabla1.D1	A1	B1
T2		En curso	tabla2.D2	A2	B2

1) T1 cambia el dato D1=A1 al valor B1

2) T2 cambia el dato D2=A2 al valor B2



Archivo de Datos



Archivo de Log

Funcionamiento de la memoria Oracle

ZONA DE MEMORIA (SGA)

Buffer Datos

B1	B2				
----	----	--	--	--	--

Buffer Rollback

A1	A2				
----	----	--	--	--	--

Buffer de Log

Tr.	Fecha	Estado	Dirección	Viejo	Nuevo
T1		En curso	tabla1.D1	A1	B1
T2		En curso	tabla2.D2	A2	B2
T1		Validado			

1) T1 cambia el dato D1=A1 al valor B1

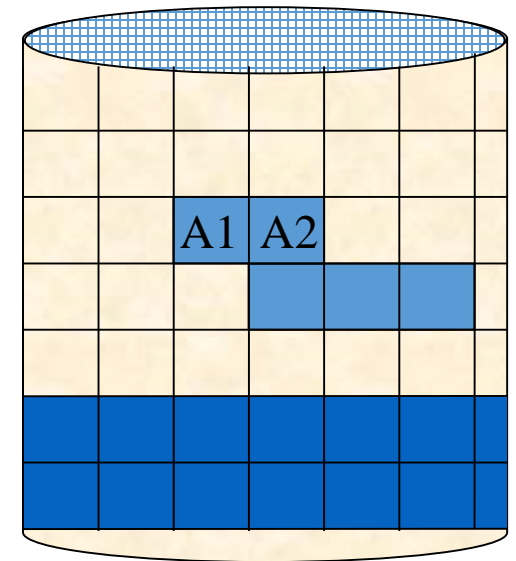
2) T2 cambia el dato D2=A2 al valor B2

3) T1 confirmado

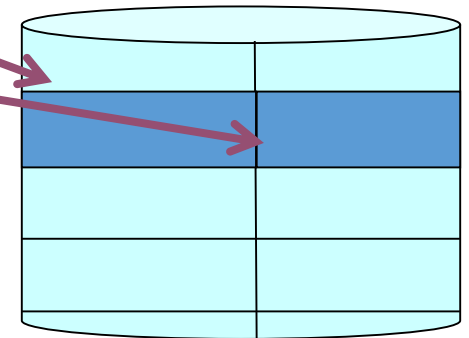
=> escribir Reg. Log

=> mover Reg. Log de T1 a Archivo Log

=> liberar Buffer Rollback



Archivo de Datos

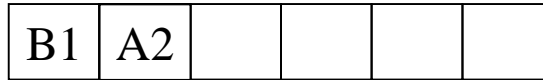


Archivo de Log

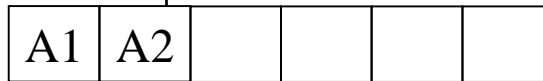
Funcionamiento de la memoria Oracle

ZONA DE MEMORIA (SGA)

Buffer Datos



Buffer Rollback



Buffer de Log

Tr.	Fecha	Estado	Dirección	Viejo	Nuevo
T1		En curso	tabla1.D1	A1	B1
T2		En curso	tabla2.D2	A2	B2
T2		Anulado			

1) T1 cambia el dato D1=A1 al valor B1

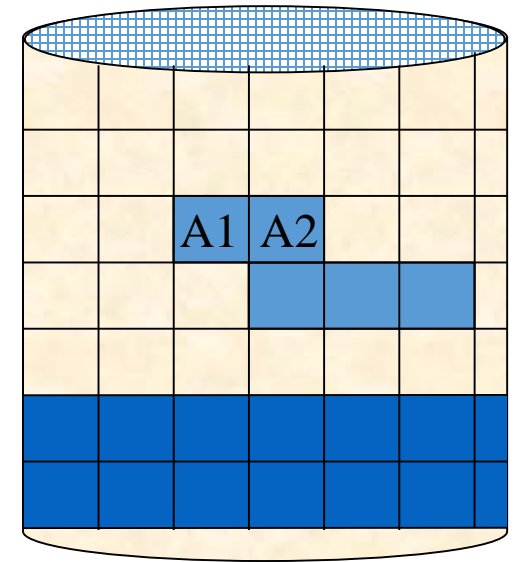
2) T2 cambia el dato D2=A2 al valor B2

3) T2 anulado

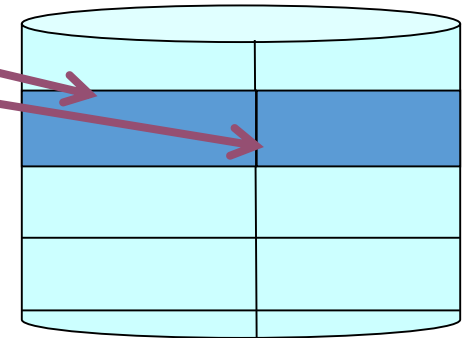
=> escribir Reg. Log

=> mover Reg. Log de T2 a Archivo Log

=> mover Buffer Rollback a Buffer Datos



Archivo de Datos



Archivo de Log

Archivo de Log

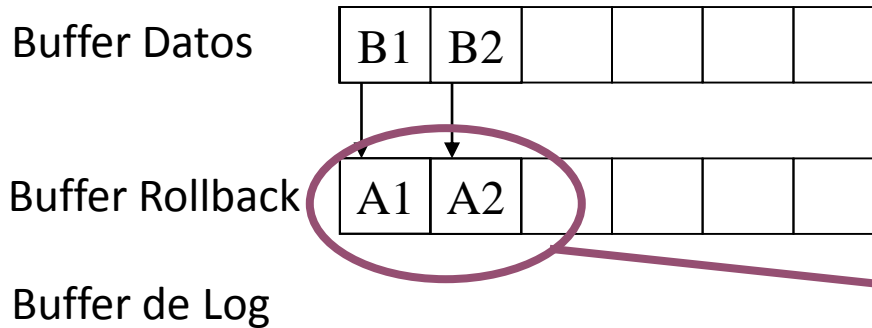
- = { registros Log } que se mantienen temporalmente en memoria

Tipos de Registros Log:

[inicio_transacción, T]	inicio ejecución de T
[escribir_elemento, T, X, <i>valor_ant</i> , <i>valor_nuevo</i>]	cambio de X(disco)
[leer_elemento, T, X]	lectura del elemento X (disco)
[confirmar, T]	\forall escrituras se han ejecutado con éxito
	\forall reg. Log de $T \rightarrow A$. Log (forzar la escritura)
[abortar, T]	se abortó la transacción T
	\forall reg. Log de $T \rightarrow A$. Log (forzar la escritura)
[checkpoint] (punto de control)	\forall datos (memoria) de T confirmadas \rightarrow disco
	\forall reg. Log \rightarrow A. Log (forzar la escritura)
se realiza periódicamente (<i>m</i> minutos, N transacciones confirmadas)	
almacena una lista con T activas en el instante checkpoint	
SI FALLO \Rightarrow	T <u>terminadas</u> antes del último checkpoint NO se repiten
	T <u>terminadas</u> después se repiten (REDO) \Rightarrow rastrear A. Log hacia adelante cambiando X al <i>valor_nuevo</i>
	T <u>NO terminadas</u> se anulan (UNDO) \Rightarrow rastrear A. Log hacia atrás restableciendo X al <i>valor_ant</i>

Funcionamiento de la memoria Oracle

ZONA DE MEMORIA (SGA)



Tr.	Fecha	Estado	Dirección	Viejo	Nuevo
T1		En curso	tabla1.D1	A1	B1
T2		En curso	tabla2.D2	A2	B2
CHECKPOINT					

1) T1 cambia el dato D1=A1 al valor B1

2) T2 cambia el dato D2=A2 al valor B2

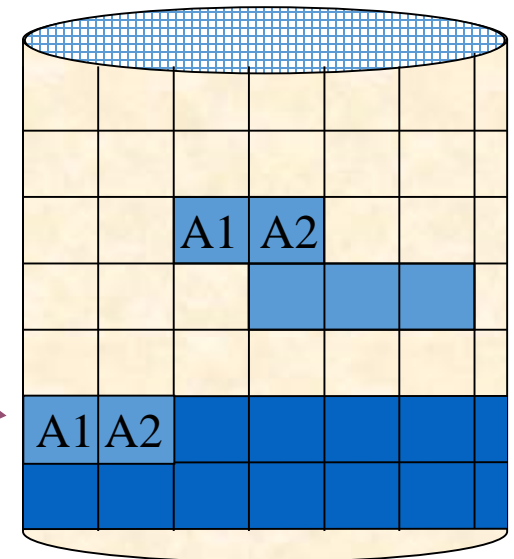
3) CHECKPOINT

=> escribir Reg. Log

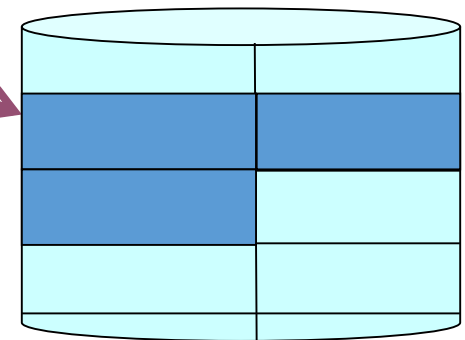
=> mover Buffer Rollback a disco

=> mover Buffer de Datos de trans.confirmdas a disco (no hay)

=> mover Reg. Log a Archivo Log



Archivo de Datos



Archivo de Log

Propiedades deseables de las transacciones

□ Atomicidad

- \forall operaciones o ninguna
- es responsabilidad del MECANISMO DE RECUPERACIÓN

□ Consistencia

- pasar de un estado consistente (cumple \forall restricciones) a otro
- es responsabilidad del programador que las transacciones sean independientes


□ Aislamiento

- la ejecución de una transacción no debe interferir en otras transacciones concurrentes
- lo impone el MECANISMO DE CONTROL DE CONCURRENCIA
- niveles:
 - 0 \Rightarrow NO sobrescribe lecturas sucias
 - 1 \Rightarrow NO actualizaciones perdidas
 - 2 \Rightarrow 1 + NO lecturas sucias
 - 3 \Rightarrow 2 + lecturas repetibles

□ Permanencia

- las transacciones confirmadas NO se pierden
- es responsabilidad del MECANISMO DE RECUPERACIÓN

Planes de transacciones

- Orden de ejecución de las operaciones $\subset \{T_1, T_2, \dots, T_n\}$
 - $\subset \forall$ operaciones (incluso [confirmar, T] [abortar, T])
 - = orden de las operaciones que en T_i
 - si \exists operaciones en **conflicto** \Rightarrow secuencial
 -  $\left\{ \begin{array}{l} \in \neq T_i \\ \text{acceso al} = X \\ \text{una operación es escritura} \end{array} \right.$

Planes según su recuperabilidad

- 1) RECUPERABLE \Rightarrow T confirmada nunca se tiene que deshacer (rollback)

Ti **no finaliza** mientras no hayan finalizado las Tj que escriben antes elementos leídos por Ti

ej: $I_1(X)$ $e_1(X)$ $I_2(X)$ $I_1(Y)$ $e_2(X)$ c_2 a_1 \rightarrow NO recuperable

ej: $I_1(X)$ $e_1(X)$ $I_2(X)$ $I_1(Y)$ $e_2(X)$ $e_1(Y)$ c_1 c_2 \rightarrow Recuperable



GARANTIZA NO anulación de Ti confirmadas

PERO...

SI anulación de Ti NO confirmadas

ej: $I_1(X)$ $e_1(X)$ $I_2(X)$ $I_1(Y)$ $e_2(X)$ $e_1(Y)$ a_1 a_2 \rightarrow anulación en cascada

2) EVITA ANULACIÓN EN CASCADA

Ti **no lee** X mientras no hayan finalizado las Tj que escriben X antes

ej: $I_2(X)$ (y todas las operaciones posteriores en T2) se pospone hasta a_1 ó c_1

3) PLAN ESTRICTO

Ti **no lee ni escribe** X mientras no hayan finalizado las Tj que escriben X antes

ej: $e_1(X)$ $e_2(X)$ a_1 \rightarrow No Plan Estricto

Sol.- $e_2(X)$ se pospone hasta a_1 ó c_1



restauración de la imagen anterior
(valor de X antes de Ti abortada)

Indicar si los planes siguientes son **recuperables** (RE), **evitan anulación en cascada** (AC) y/o **estrictos** (ES). Si no se puede decir si un plan pertenece a cierta clase según las acciones indicadas, explicarlo brevemente

Las acciones aparecen en orden consecutivo dentro de cada plan. Si no aparecen las confirmaciones de las transacciones, asumir que los *commit* / *abort* se encuentran después de todas las acciones indicadas.

	PLAN	RE	AC	ES
1	I1(X) I2(X) e1(X) e2(X)			
2	e1(X) I2(Y) I1(Y) I2(X)			
3	I1(X) I2(Y) e3(X) I2(X) I1(Y)			
4	I1(X) I1(Y) e1(X) I2(Y) e3(Y) e1(X) I2(Y)			
5	I1(X) e2(X) e1(X) a2 c1			
6	I1(X) e2(X) e1(X) c2 c1			
7	I2(X) e3(X) c3 e1(Y) c1 I2(Y) e2(Z) c2			
8	I1(X) e2(X) c2 e1(X) c1 I3(X) c3			

RECUPERABLE: Ti **no finaliza** mientras no hayan finalizado las Tj que escriben antes elementos leídos por Ti

EVITA ANULACIÓN EN CASCADA: Ti **no lee** un elemento mientras no hayan finalizado las Tj que escriben X antes

PLAN Estricto: Ti **no lee ni escribe** un elemento mientras no hayan finalizado las Tj que escriben X antes

	PLAN	RE	AC	ES
1	I1(X) I2(X) e1(X) e2(X)	S	S	
2	e1(X) I2(Y) I1(Y) I2(X)	(a)		
3	I1(X) I2(Y) e3(X) I2(X) I1(Y)	(b)		
4	I1(X) I1(Y) e1(X) I2(Y) e3(Y) e1(X) I2(Y)	(c)		
5	I1(X) e2(X) e1(X) a2 c1	S	S	
6	I1(X) e2(X) e1(X) c2 c1	S	S	
7	I2(X) e3(X) c3 e1(Y) c1 I2(Y) e2(Z) c2	S	S	S
8	I1(X) e2(X) c2 e1(X) c1 I3(X) c3	S	S	S

(a) se cumple si T1 finaliza antes de T2

(b) se cumple si T3 finaliza antes de T2

(c) se cumple si T3 finaliza antes de T2

Serialización de los planes

- Determina qué planes son “correctos”

1) PLAN EN SERIE:

- \forall operaciones de T se ejecutan consecutivamente (sin intercalación)
- Si T_i 's *independientes* (consistencia) $\Rightarrow \forall$ plan serie es CORRECTO
- Problema: limita la concurrencia

2) PLAN NO EN SERIE:

- 📦 Intercala operaciones entre T_i 's
- 📦 SERIALIZABLE \Rightarrow EQUIVALENTE a un PLAN EN SERIE \Rightarrow correcto



~~por resultados~~ \Rightarrow produce el mismo estado en la BD
para cada dato se aplican = operaciones y en = orden

Def. 2.1) **por conflictos**
2.2) **por vistas**

Ejemplo:

P1
leer_elemento (X)
 $X := X + 10$
escribir_elemento(X)

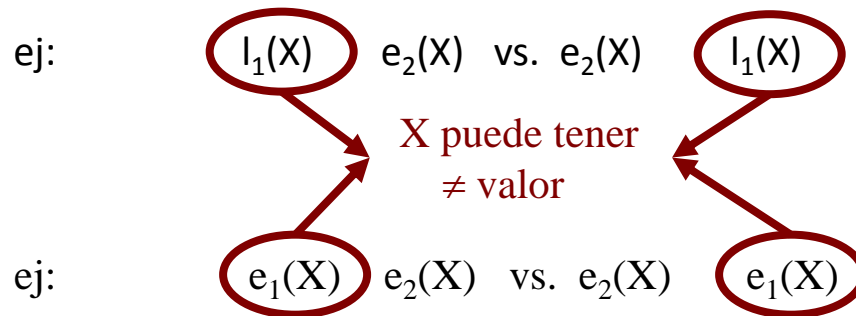
P2
leer_elemento (X)
 $X := X * 1.1$
escribir_elemento(X)

} equivalentes por resultado si **$X=100$**

Serialización de los planes

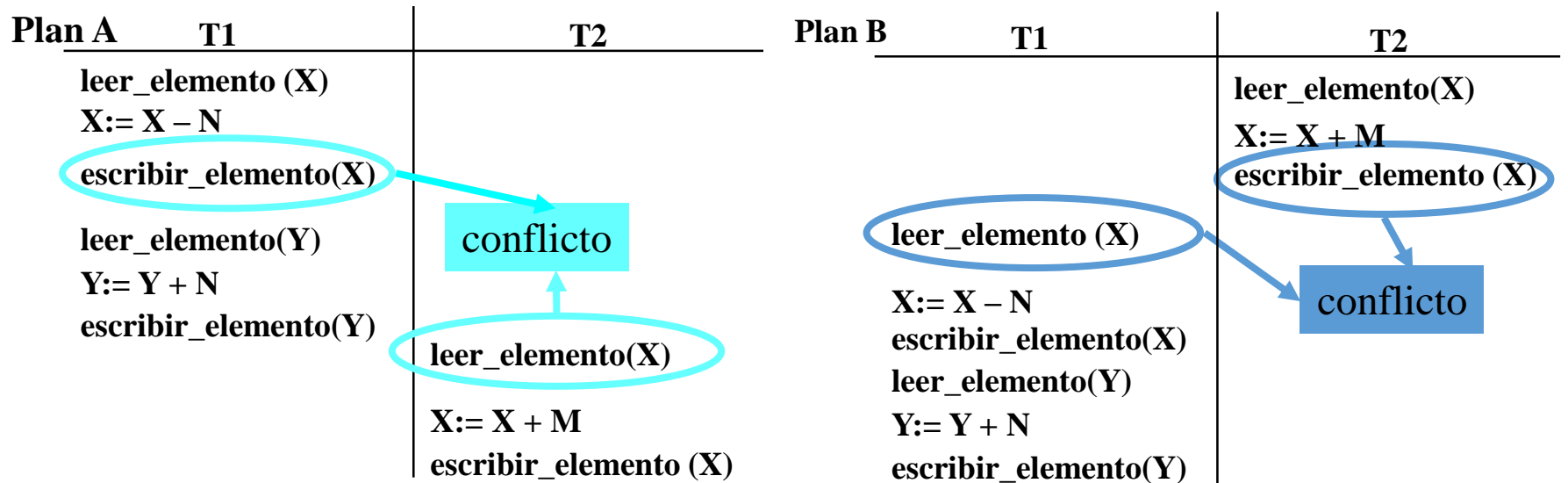
2.1) Equivalencia POR CONFLICTOS:

El orden de las operaciones en conflicto es = en ambos planes



P NO en serie es SERIALIZABLE POR CONFLICTOS \Rightarrow
es EQUIVALENTE POR CONFLICTOS a P' en serie
que se obtiene reordenando las operaciones NO en conflicto

Planes Serializables por Conflictos



¿¿ D equivalente (por conflictos) a A ??

SI porque:

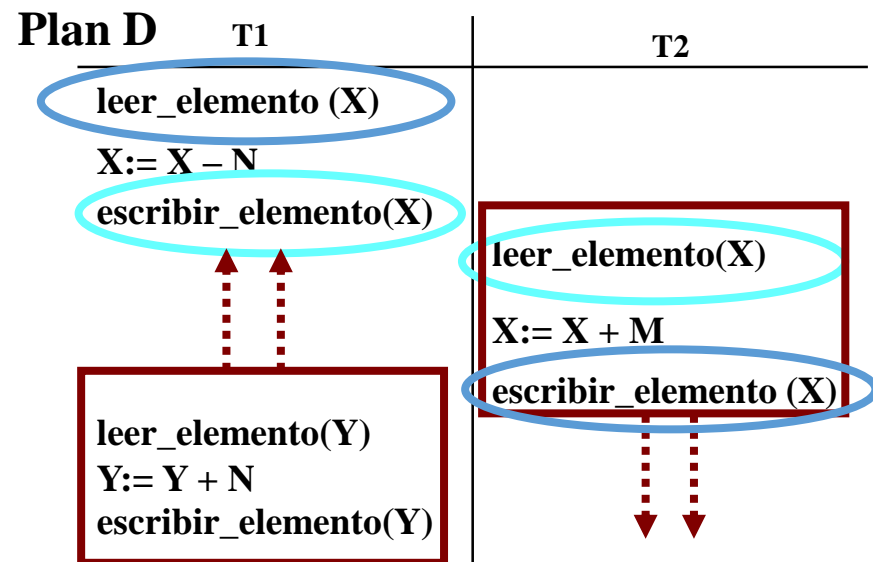
- $l_2(X)$ posterior a $e_1(X)$



**D es SERIALIZABLE
por conflictos**

¿¿ D equivalente (por conflictos) a B ??

NO



Planes Serializables por Conflictos

Plan A	T1	T2
	<code>leer_elemento(X)</code> <code>X := X - N</code> <code>escribir_elemento(X)</code> <code>leer_elemento(Y)</code> <code>Y := Y + N</code> <code>escribir_elemento(Y)</code>	<code>leer_elemento(X)</code> <code>X := X + M</code> <code>escribir_elemento(X)</code>

Diagram illustrating a conflict between Plan A and Plan B. A red oval highlights `escribir_elemento(X)` in Plan A and `leer_elemento(X)` in Plan B. A red box labeled **conflicto** is connected to both operations by red arrows.

Plan B	T1	T2
	<code>leer_elemento(X)</code> <code>X := X - N</code> <code>escribir_elemento(X)</code> <code>leer_elemento(Y)</code> <code>Y := Y + N</code> <code>escribir_elemento(Y)</code>	<code>leer_elemento(X)</code> <code>X := X + M</code> <code>escribir_elemento(X)</code>

Diagram illustrating a conflict between Plan A and Plan B. A red oval highlights `leer_elemento(X)` in Plan B and `escribir_elemento(X)` in Plan A. A red box labeled **conflicto** is connected to both operations by red arrows.

Plan C	T1	T2
	<code>leer_elemento(X)</code> <code>X := X - N</code> <code>escribir_elemento(X)</code> <code>leer_elemento(Y)</code> <code>Y := Y + N</code> <code>escribir_elemento(Y)</code>	<code>leer_elemento(X)</code> <code>X := X + M</code> <code>escribir_elemento(X)</code>

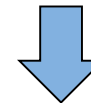
Diagram illustrating a conflict between Plan A and Plan B. A red oval highlights `leer_elemento(X)` in Plan B and `escribir_elemento(X)` in Plan A. A red box labeled **conflicto** is connected to both operations by red arrows.

C NO equivalente (por conflictos) a A pq:

- $l_2(X)$ anterior a $e_1(X)$

C NO equivalente (por conflictos) a B pq:

- $l_1(X)$ anterior a $e_2(X)$



C NO es SERIALIZABLE
(por conflictos)

Prueba de Serialización por Conflictos

- Construir un GRAFO DE PRECEDENCIA $G = (N, A)$ siendo:
 - $N = \{T_1, T_2, \dots, T_n\}$
 - $A = \{a_1, a_2, \dots, a_m\}$ donde $a_p = (T_i \rightarrow T_j)$
 - \Rightarrow 1 oper. de T_i aparece antes que una *operación en conflicto* de T_j
 - \Rightarrow en un plan en serie T_i debe aparecer antes de T_j

ALGORITMO:

1. crear un nodo $\forall T_i$ de P
2. $\dots e_i(X) \dots l_j(X) \dots \Rightarrow T_i \rightarrow T_j$
3. $\dots l_i(X) \dots e_j(X) \dots \Rightarrow T_i \rightarrow T_j$
4. $\dots e_i(X) \dots e_j(X) \dots \Rightarrow T_i \rightarrow T_j$
5. P es SERIALIZABLE (por conflictos) \Leftrightarrow el grafo NO tiene ciclos
6. el plan en serie equivalente (por conflictos) se obtiene siguiendo las aristas (ordenación topológica)

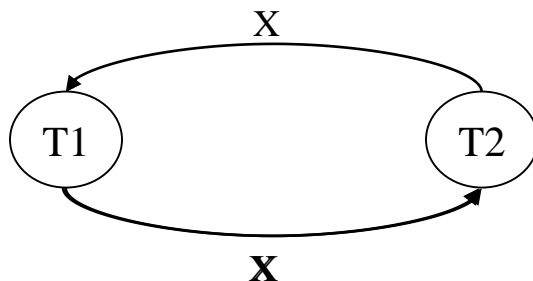
Prueba de Serialización por Conflictos

Plan A	T1	T2
leer_elemento (X)		
X := X - N		
escribir_elemento(X)		
leer_elemento(Y)		
Y := Y + N		
escribir_elemento(Y)		
		leer_elemento(X)
		X := X + M
		escribir_elemento (X)

conflicto

Plan B	T1	T2
		leer_elemento(X)
		X := X + M
		escribir_elemento (X)
	leer_elemento (X)	
	X := X - N	
	escribir_elemento(X)	
	leer_elemento(Y)	
	Y := Y + N	
	escribir_elemento(Y)	

conflicto



Plan A

plan serie equivalente: A = T1 T2

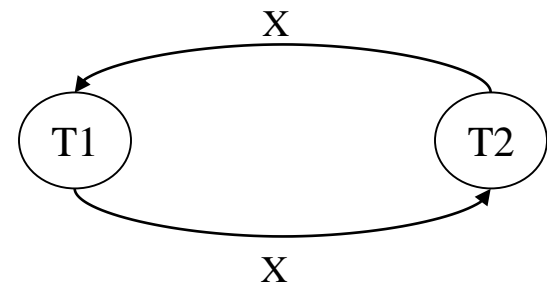
Plan D	T1	T2
	leer_elemento (X)	
	X := X - N	
	escribir_elemento(X)	
		leer_elemento(X)
		X := X + M
		escribir_elemento (X)
	leer_elemento(Y)	
	Y := Y + N	
	escribir_elemento(Y)	

Planes Serializables por Conflictos

Plan A	T1	T2
	leer_elemento (X)	
	X:= X - N	
	escribir_elemento(X)	
	leer_elemento(Y)	
	Y:= Y + N	
	escribir_elemento(Y)	
		leer_elemento(X)
		X:= X + M
		escribir_elemento (X)

Plan B	T1	T2
		leer_elemento(X)
		X:= X + M
		escribir_elemento (X)
	leer_elemento (X)	
	X:= X - N	
	escribir_elemento(X)	
	leer_elemento(Y)	
	Y:= Y + N	
	escribir_elemento(Y)	

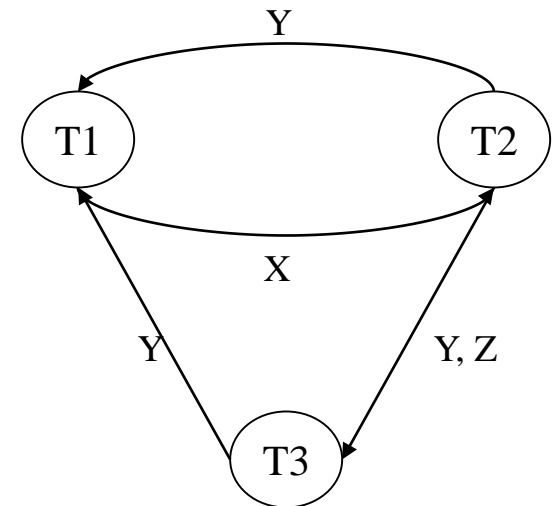
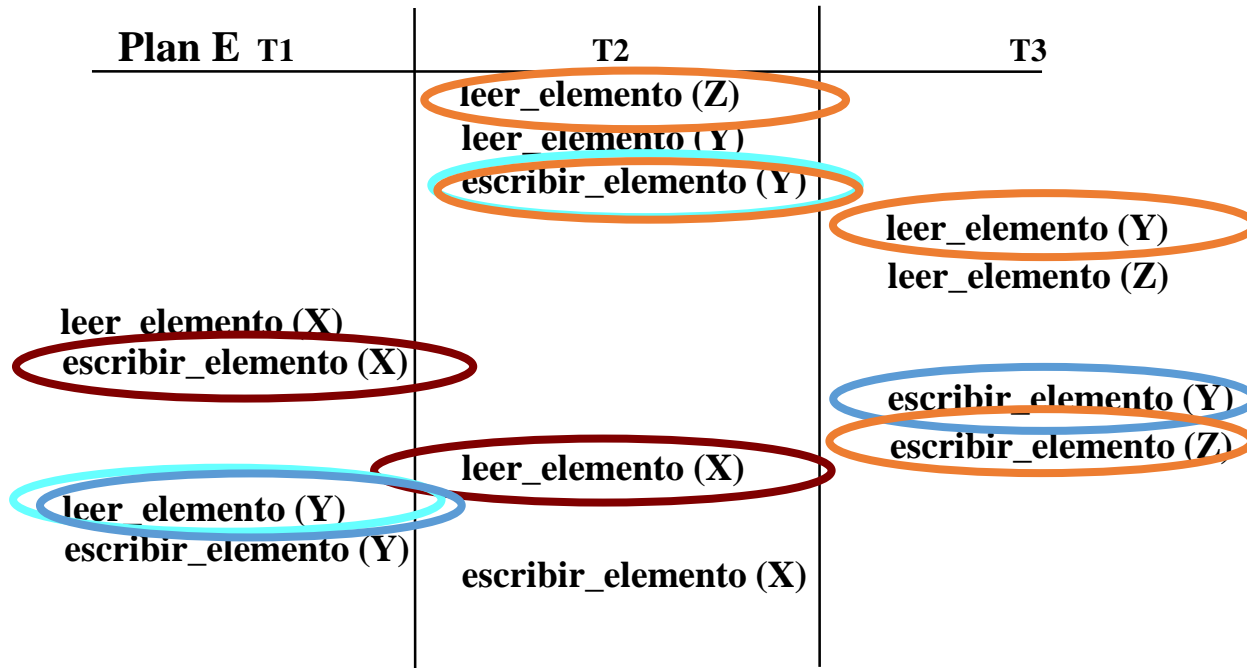
Plan C	T1	T2
	leer_elemento (X)	
	X:= X - N	
		leer_elemento(X)
	escribir_elemento(X)	
		X:= X + M
	leer_elemento(Y)	
		escribir_elemento (X)
	Y:= Y + N	
	escribir_elemento(Y)	



Plan C

plan serie equivalente: ninguno

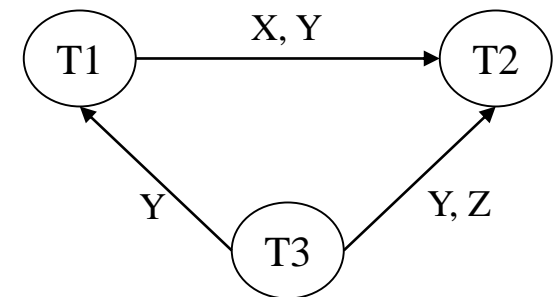
Planes Serializables por Conflictos



plan serie equivalente: ninguno

Planes Serializables por Conflictos

Plan F	T1	T2	T3
	leer_elemento (X) escribir_elemento (X)		leer_elemento (Y) leer_elemento (Z)
	leer_elemento (Y) escribir_elemento (Y)	leer_elemento (Z)	escribir_elemento (Y) escribir_elemento (Z)
		leer_elemento (Y) escribir_elemento (Y)	
		leer_elemento (X) escribir_elemento (X)	



plan serie equivalente: T3 → T1 → T2

Serialización de los planes

2.2) P y P' son equivalentes POR VISTAS \Rightarrow

1. $\forall Ti \in P \Rightarrow Ti \in P'$

2. $\forall \left\{ \begin{array}{l} l_i(X) \\ e_j(X) \ l_i(X) \end{array} \right\} \text{ en } P \Rightarrow \left\{ \begin{array}{l} l_i(X) \\ e_j(X) \ l_i(X) \end{array} \right\} \text{ en } P'$

(\forall lectura lee el resultado de la misma escritura en ambos planes,
es decir, P y P' perciben la **MISMA VISTA**)

3. $e_k(Y)$ última escritura de Y en P $\Rightarrow e_k(Y)$ última escritura de Y en P'
(**MISMO ESTADO FINAL** de la BD)

P NO en serie es SERIALIZABLE POR VISTAS \Rightarrow
es EQUIVALENTE POR VISTAS a P' en serie

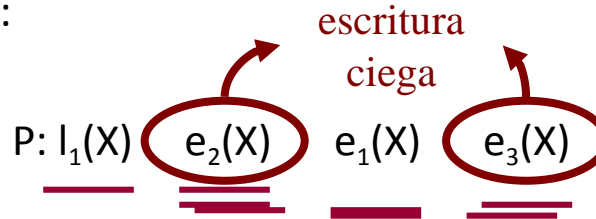
Planes Serializables por Vistas

- Permiten ESCRITURA CIEGA:

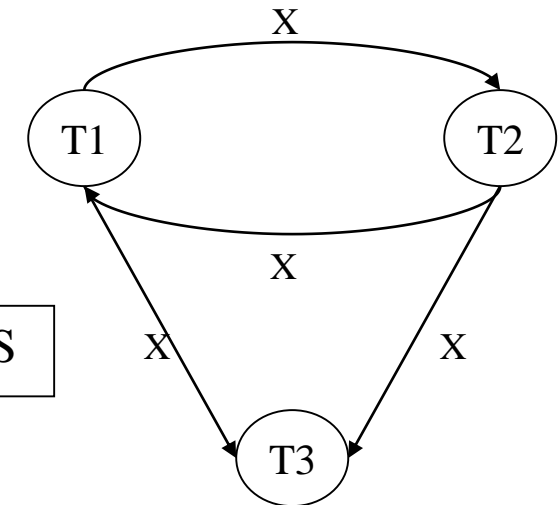
T1: $l_1(X)$ $e_1(X)$

T2: $e_2(X)$

T3: $e_3(X)$



P es SERIALIZABLE POR VISTAS (equivalente por vistas a $P' = \{T1, T2, T3\}$)
pero NO SERIALIZABLE POR CONFLICTOS



$\forall P$ serializable por CONFLICTOS es serializable por VISTAS

$\forall P$ serializable por VISTAS es serializable por CONFLICTOS \Leftrightarrow

$\forall e_i(X) \Rightarrow \exists l_i(X)$, y depende SOLO de $l_i(X)$

(ESCRITURA RESTRINGIDA)

Ejemplo

Indicar si los planes siguientes son **serializables por conflictos** (SC) y/o **serializables por vistas** (SV) marcando con una X el recuadro correspondiente en caso afirmativo.

Si no se puede decir si un plan pertenece a cierta clase según las acciones indicadas, explicarlo brevemente (para ello, en el recuadro correspondiente poner un número y explicar debajo de la tabla).

Las acciones aparecen en el orden consecutivo dentro de cada plan. En caso de que no aparezcan las confirmaciones de las transacciones, asumir que los *commit* / *abort* deberán encontrarse después de todas las acciones indicadas.

	PLAN	SC	SV
1	I1(X) I2(X) e1(X) e2(X)		
2	e1(X) I2(Y) I1(Y) I2(X)		
3	I1(X) I2(Y) e3(X) I2(X) I1(Y)		
4	I1(X) I1(Y) e1(X) I2(Y) e3(Y) e1(X) I2(Y)		
5	I1(X) e2(X) e1(X) a2 c1		
6	I1(X) e2(X) e1(X) c2 c1		
7	I2(X) e3(X) c3 e1(Y) c1 I2(Y) e2(Z) c2		
8	I1(X) e2(X) c2 e1(X) c1 I3(X) c3		

Solución

Indicar si los planes siguientes son **serializables por conflictos** (SC) y/o **serializables por vistas** (SV) marcando con una X el recuadro correspondiente en caso afirmativo.

	PLAN	SC	SV
1	I1(X) I2(X) e1(X) e2(X)		
2	e1(X) I2(Y) I1(Y) I2(X)	(1)	S
3	I1(X) I2(Y) e3(X) I2(X) I1(Y)	(2)	S
4	I1(X) I1(Y) e1(X) I2(Y) e3(Y) e1(X) I2(Y)		
5	I1(X) e2(X) e1(X) a2 c1		(3)
6	I1(X) e2(X) e1(X) c2 c1		
7	I2(X) e3(X) c3 e1(Y) c1 I2(Y) e2(Z) c2	S	S
8	I1(X) e2(X) c2 e1(X) c1 I3(X) c3		

(1) el plan en serie equivalente es T1 T2

(2) el plan en serie equivalente es T1 T3 T2

(3) si se tiene en cuenta que T2 se anula, puede ser SV porque es equivalente a T1 T2

Bibliografía

- *Fundamentos de Sistemas de Bases de Datos. Ramez A. Elmasri, Shamkant B. Navathe.*
- *Fundamentos de Bases de Datos. A. Silberschatz. McGraw-Hill*