



Tema.- Gestión de transacciones

*Fundamentos de Sistemas de Bases de Datos. Ramez A.
Elmasri, Shamkant B. Navathe [cap. 19]*

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 intercalan
- ▣ 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)
memoria \rightarrow X (disco)
X (variable)

EJ: RESERVAS DE VUELOS:

T1: transfiere N reservas del vuelo X al vuelo Y

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}

T2: añade M reservas al vuelo X

leer_elemento (X)
X := X + M
escribir_elemento(X)

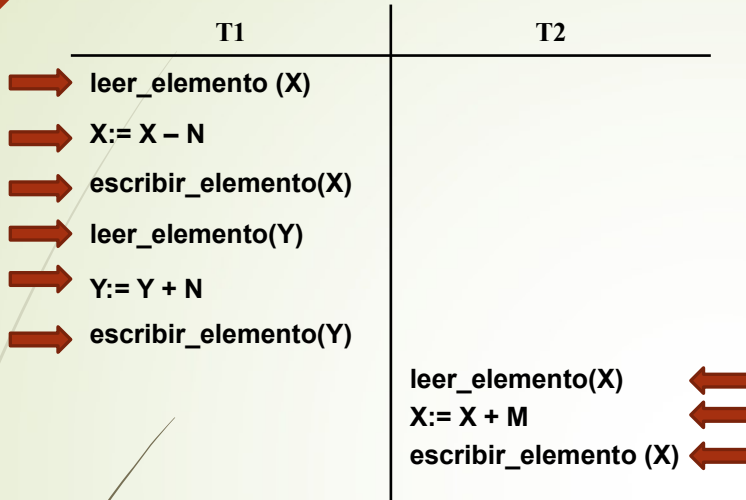
Conjunto lectura = {X}

Conjunto escritura = {X}

¿Por qué se necesita Control de Concurrency?

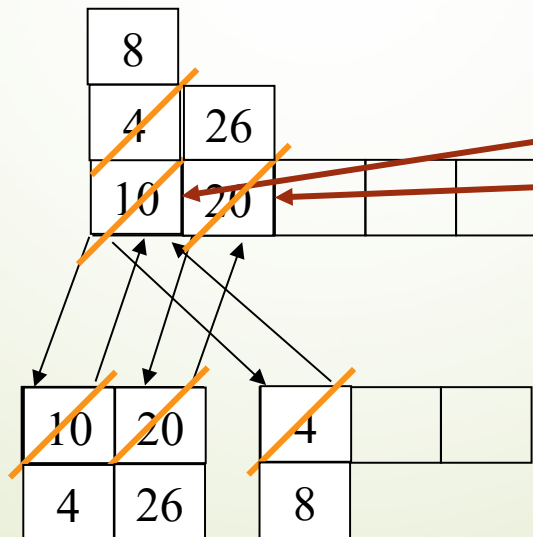
Ejemplo: Sin concurrencia T1-T2

X=10 Y=20 N=6 M=4



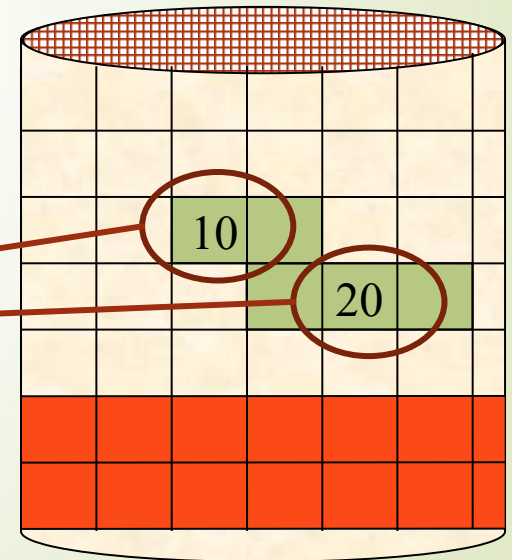
Buffer Datos
(SGA)

Zona PGA



Variables de T1

Variables de T2

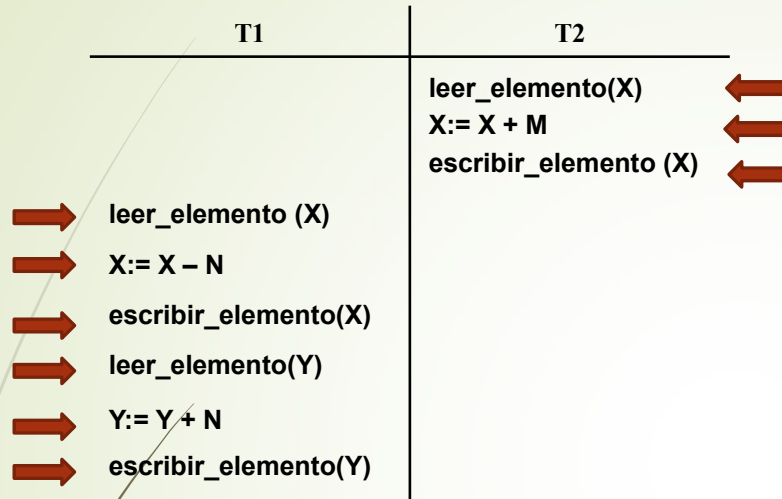


Archivo de Datos

¿Por qué se necesita Control de Concurrency?

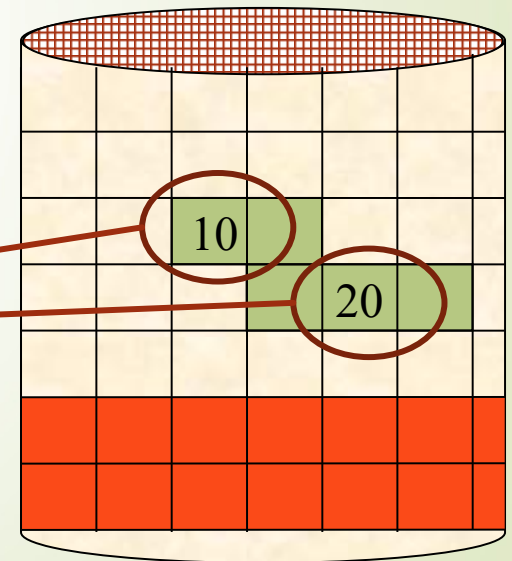
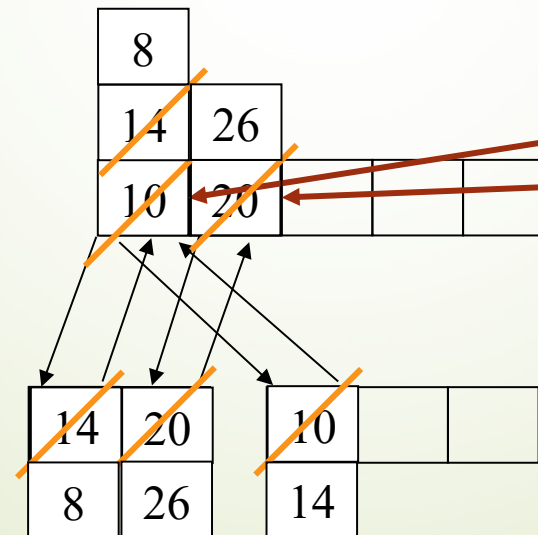
Ejemplo: Sin concurrencia T2-T1

X=10 Y=20 N=6 M=4



Buffer Datos
(SGA)

Zona PGA

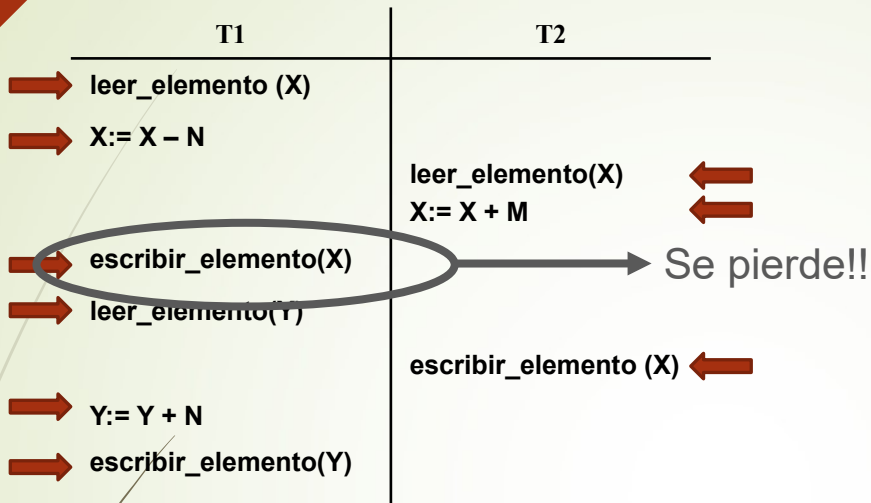


Archivo de Datos

¿Por qué se necesita Control de Concurrency?

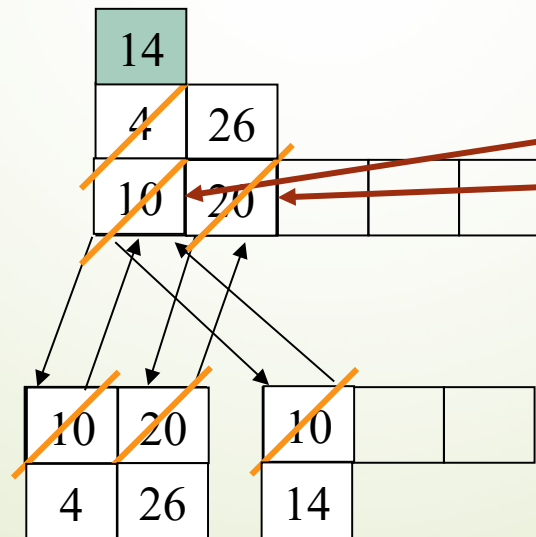
PROBLEMA 1) Actualización perdida

X=10 Y=20 N=6 M=4



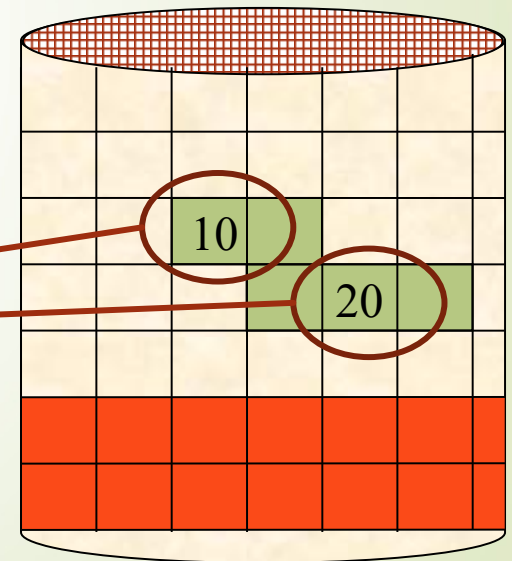
Buffer Datos
(SGA)

Zona PGA



Variables de T1

Variables de T2



Archivo de Datos

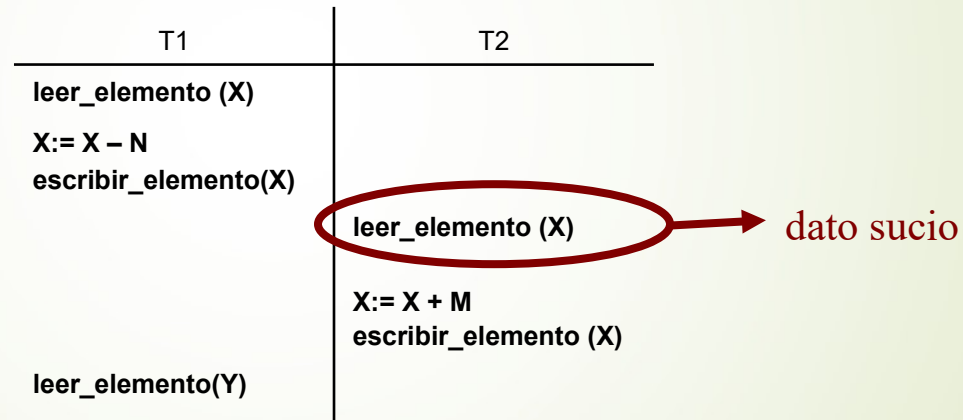
¿Por qué se necesita Control de Concurrency?

EJ: RESERVAS DE VUELOS:

T1: transfiere N reservas del vuelo X al vuelo Y

T2: añade M reservas al vuelo X

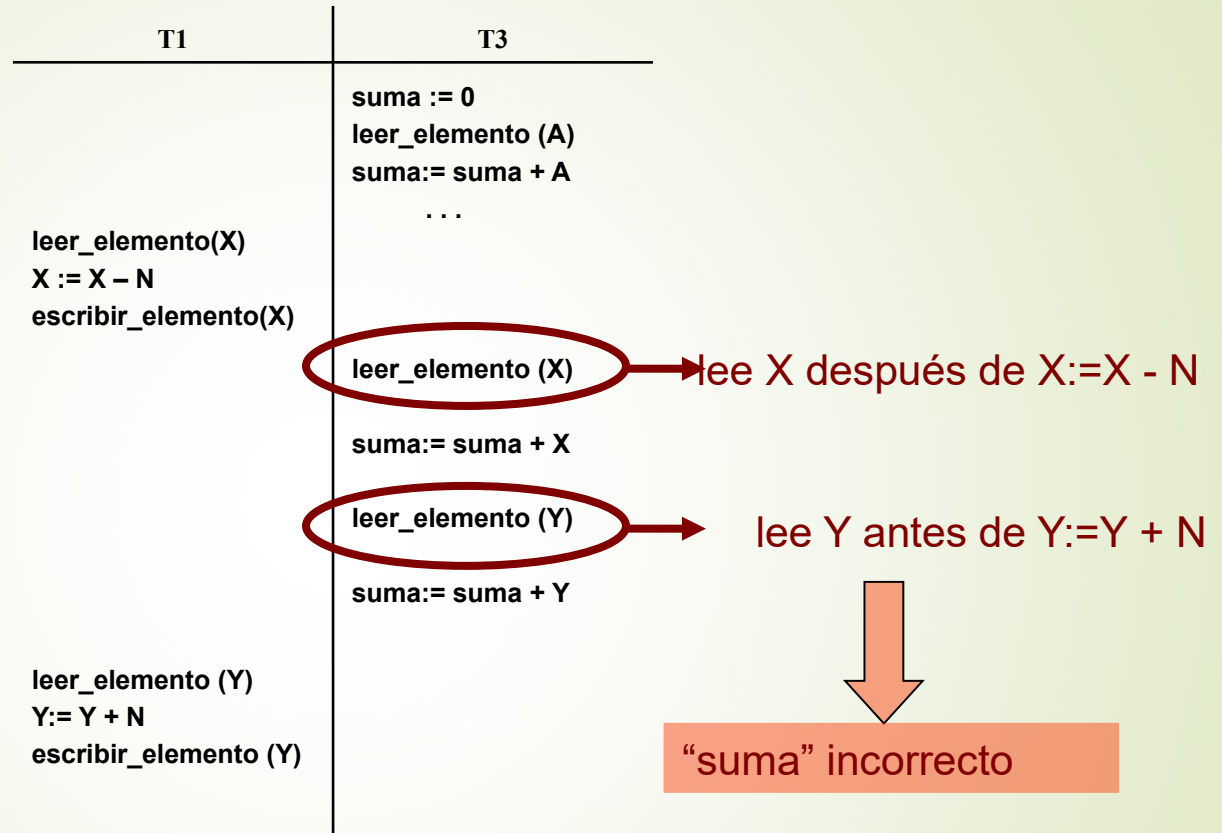
PROBLEMA 2) Actualización temporal (lectura sucia)



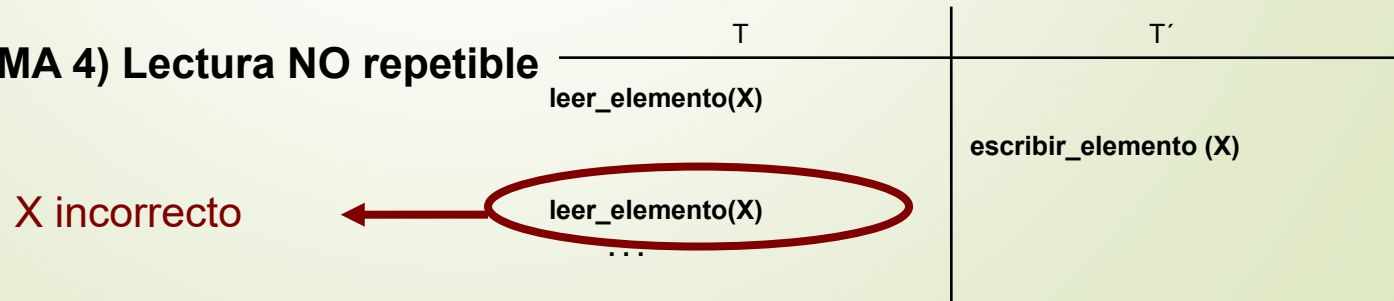
T1 falla \Rightarrow X debe volver a su antiguo valor
PERO T2 ya ha leído X

¿Por qué se necesita Control de Concurrency?

PROBLEMA 3) Resumen incorrecto



PROBLEMA 4) Lectura NO repetible



Estado de una transacción

Operaciones

Inicio de Transacción:

`inicio_transacción`

Operaciones de Acceso:

$I_T(X)$ `leer_elemento(X)`

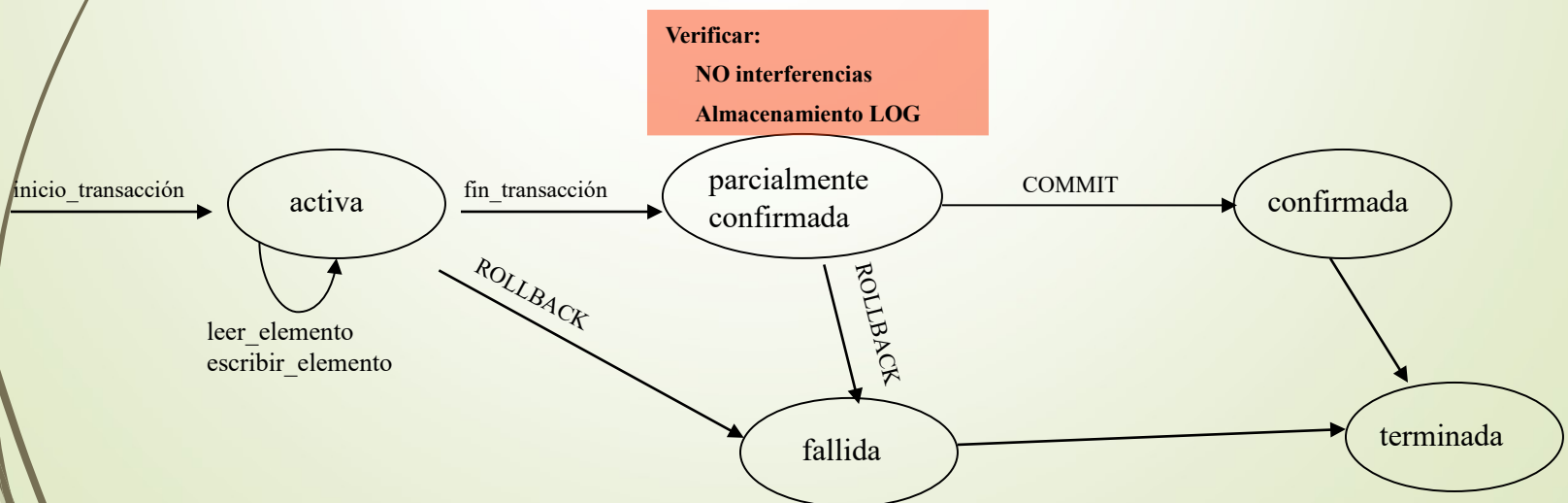
$e_T(X)$ `escribir_elemento(X)`

Terminación de Transacción:

`fin_transacción`

c_T `confirmar (COMMIT)` terminó con éxito \Rightarrow confirmar cambios

a_T `deshacer (ROLLBACK)` terminó sin éxito \Rightarrow cancelar cambios



Archivo de Log

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

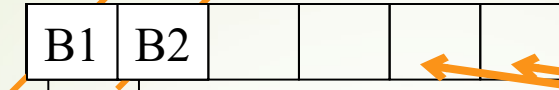
Tipos de Registros Log:

[inicio_transacción, T]	inicio ejecución de T
[escribir_elemento, T, X, <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]	falló o se abortó la transacción T \forall reg. Log de $T \rightarrow A$. Log (forzar la escritura)

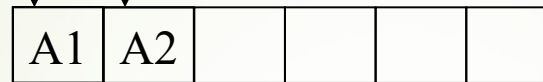
Funcionamiento de la memoria Oracle

ZONA DE MEMORIA (SGA)

Buffer Datos



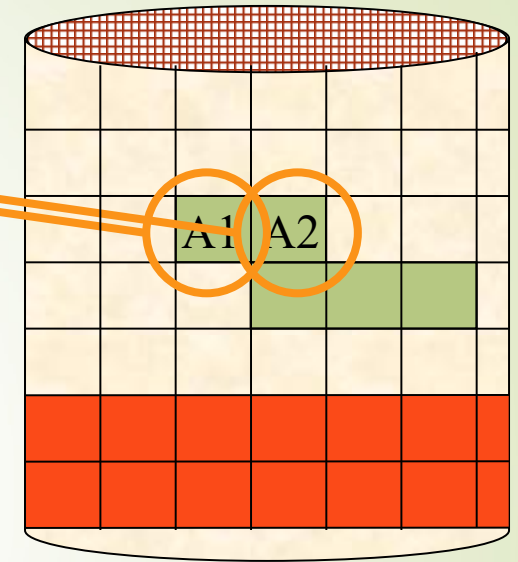
Buffer Rollback



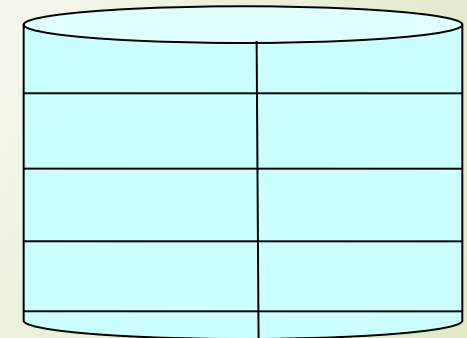
Buffer de Log

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

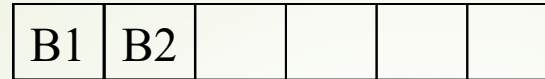


Archivo de Log

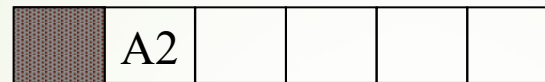
Funcionamiento de la memoria Oracle

ZONA DE MEMORIA (SGA)

Buffer Datos



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
T1		Confirmar			

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

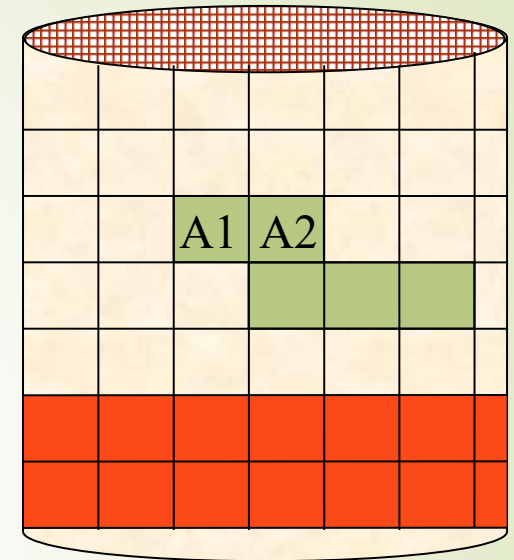
2) T2 cambia el dato D2=A2 al valor B2

3) T1 confirmado

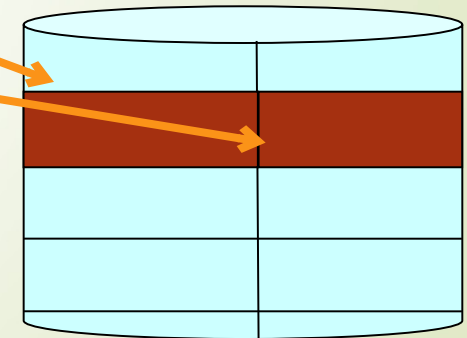
=> escribir Reg. Log

=> mover Reg. Log de T1 a Archivo Log

=> liberar Buffer Rollback



Archivo de Datos

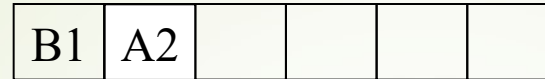


Archivo de Log

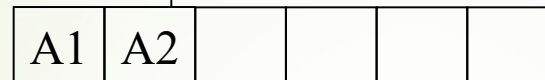
Funcionamiento de la memoria Oracle

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		Abortar			

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

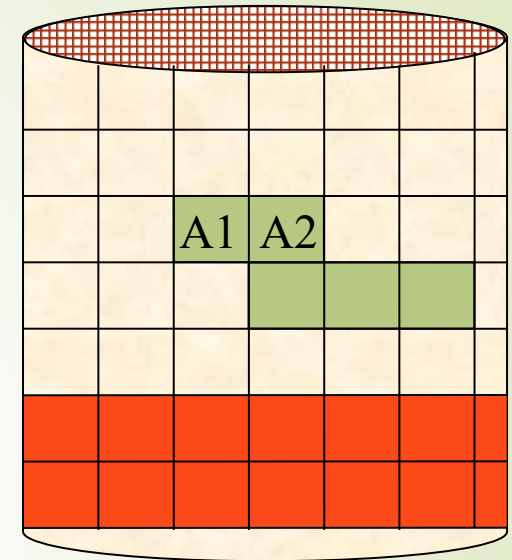
2) T2 cambia el dato D2=A2 al valor B2

3) T2 anulado

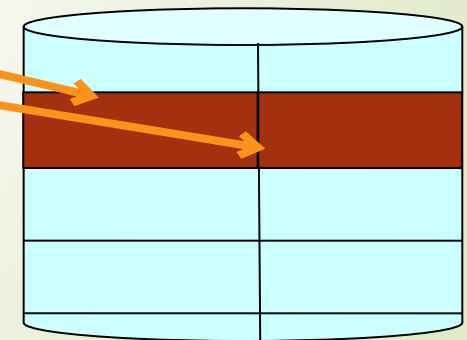
=> escribir Reg. Log

=> mover Reg. Log de T2 a Archivo Log

=> mover Buffer Rollback a Buffer Datos



Archivo de Datos



Archivo de Log

Archivo de Log

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] \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 (*m* minutos, N transacciones confirmadas)

almacena una lista con T activas en el instante checkpoint

SI FALLO \Rightarrow T terminadas antes del último checkpoint NO se repiten

T terminadas después se repiten (REDO) \Rightarrow rastrear A. Log hacia adelante
cambiando X al *valor_nuevo*

T NO terminadas se anulan (UNDO) \Rightarrow rastrear A. Log hacia atrás
restableciendo X al *valor_ant*

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
CHECKPOINT					

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

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

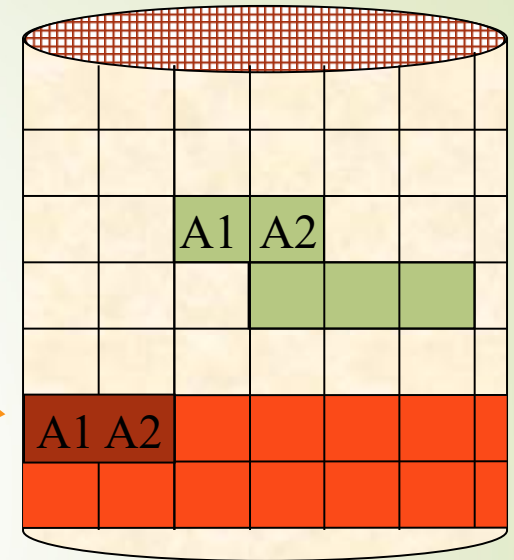
3) CHECKPOINT

=> escribir Reg. Log

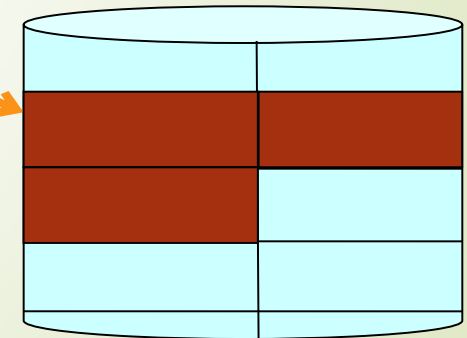
=> mover Buffer Rollback a disco

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

=> mover Reg. Log a Archivo Log



Archivo de Datos



Archivo de Log

Propiedades deseables de las transacciones

Atomicidad (**A**tomicity)

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

Consistencia (**C**onsistency)

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

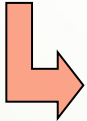
Aislamiento (**I**solation)

- la ejecución de una transacción no debe interferir en otras transacciones concurrentes
- lo impone el MECANISMO DE CONTROL DE CONCURRENCIA

Permanencia (**D**urability)

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

Planes de transacciones

- ▮ Orden de ejecución de las operaciones $\subset \{T1, T2, \dots, Tn\}$
 - ▮ $\subset \forall$ 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 últimas Tj que escriben antes elementos leídos por Ti

ej: $I_1(X)$ $e_1(X)$ **$I_2(X)$** $I_1(Y)$ $e_2(X)$ $e_1(Y)$

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

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

\rightarrow No Plan Estricto

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

Ejemplos planes según su recuperabilidad

Para cada uno de los planes siguientes, indicar si son recuperables (RE), evitan anulación en cascada (AC), estrictos (ES), del siguiente modo:

- En caso **afirmativo** escribir **S** en el recuadro correspondiente
- 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* 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			

Ejemplos planes según su recuperabilidad

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

EVITA ANULACIÓN EN CASCADA: Ti **NO LEE** X mientras no hayan finalizado las Tj que escriben X antes

PLAN ESTRICTO: Ti **NO LEE NI ESCRIBE** X mientras no hayan finalizado las Tj que escriben X antes

	PLAN	RE	AC	ES
1	l1(X) l2(X) e1(X) e2(X)			
2	e1(X) l2(Y) l1(Y) l2(X)			
3	l1(X) l2(Y) e3(X) l2(X) l1(Y)			
4	l1(X) l1(Y) e1(X) l2(Y) e3(Y) e1(X) l2(Y)			
5	l1(X) e2(X) e1(X) a2 c1			
6	l1(X) e2(X) e1(X) c2 c1			
7	l2(X) e3(X) c3 e1(Y) c1 l2(Y) e2(Z) c2			
8	l1(X) e2(X) c2 e1(X) c1 l3(X) c3			

(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

Seriabilidad de los planes

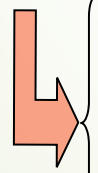
- Determina qué planes son “correctos”

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

2.1) por conflictos
2.2) Def por vistas

Ejemplo:

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

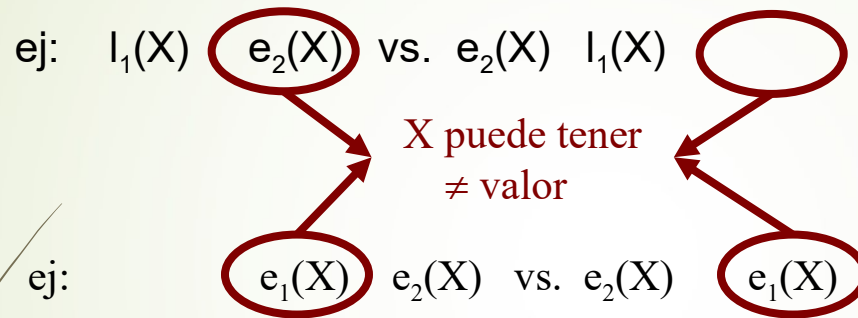
P2
leer_elemento (X)
 $X := X * 1.1$
escribir_elemento(X)

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

Seriabilidad de los planes

2.1) 2 planes son **EQUIVALENTES 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

Planes Serializables por Conflictos

Plan A

leer_elemento (X)

X := X - N

escribir_elemento(X)

leer_elemento(Y)

Y := Y + N

escribir_elemento(Y)

T2

conflicto

leer_elemento(X)

X := X + M

escribir_elemento (X)

Plan B

T1

leer_elemento (X)

X := X - N

escribir_elemento(X)

leer_elemento(Y)

Y := Y + N

escribir_elemento(Y)

T2

leer_elemento(X)

X := X + M

escribir_elemento (X)

conflicto

¿¿ 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

Plan D

leer_elemento (X)

X := X - N

escribir_elemento(X)

leer_elemento(Y)

Y := Y + N

escribir_elemento(Y)

T2

leer_elemento(X)

X := X + M

escribir elemento (X)

Planes Serializables por Conflictos

Plan A

leer_elemento (X)

X:= X - N

escribir_elemento(X)

leer_elemento(Y)

Y:= Y + N

escribir_elemento(Y)

T2

conflicto

leer_elemento(X)

X:= X + M

escribir_elemento (X)

Plan B

T1

leer_elemento (X)

X:= X - N

escribir_elemento(X)

leer_elemento(Y)

Y:= Y + N

escribir_elemento(Y)

T2

leer_elemento(X)

X:= X + M

escribir_elemento (X)

conflicto

Plan C

leer_elemento (X)

X:= X - N

escribir_elemento(X)

leer_elemento(Y)

Y:= Y + N

escribir_elemento(Y)

T2

leer_elemento(X)

X:= X + M

escribir_elemento (X)

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 Seriabilidad 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)

Planes Serializables por Conflictos

Plan A

leer_elemento (X)

$X := X - N$

escribir_elemento(X)

leer_elemento(Y)

$Y := Y + N$

escribir_elemento(Y)

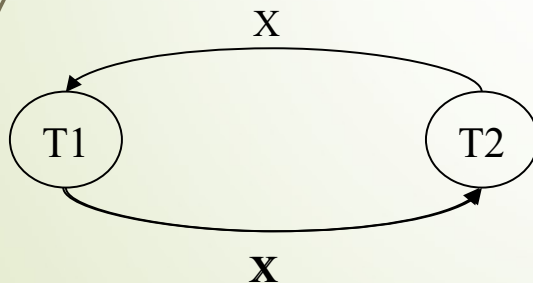
T2

conflicto

leer_elemento(X)

$X := X + M$

escribir_elemento (X)



Plan B

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 D

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)

plan serie equivalente: A = T1 T2

Planes Serializables por Conflictos

Plan A

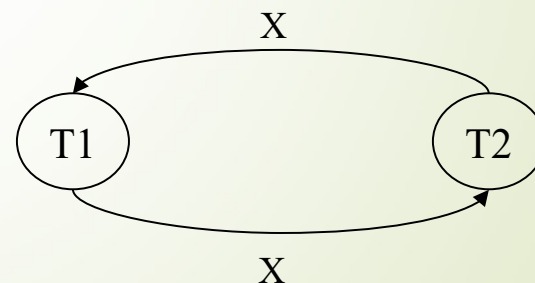
	T2
leer_elemento (X)	
X:= X - N	
escribir_elemento(X)	
leer_elemento(Y)	
Y:= Y + N	
escribir_elemento(Y)	
	leer_elemento(X)
	X:= X + M
	escribir_elemento (X)

Plan C

	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 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)	

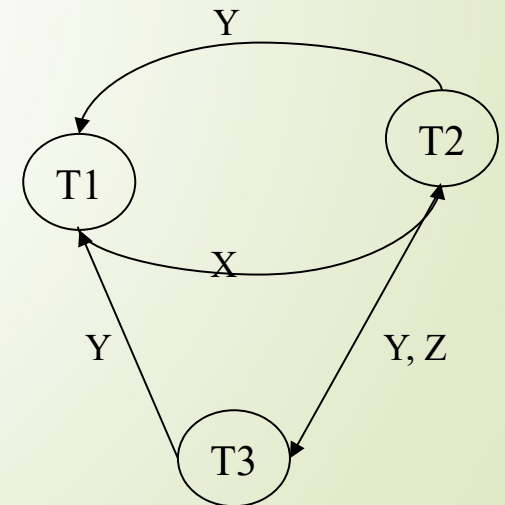
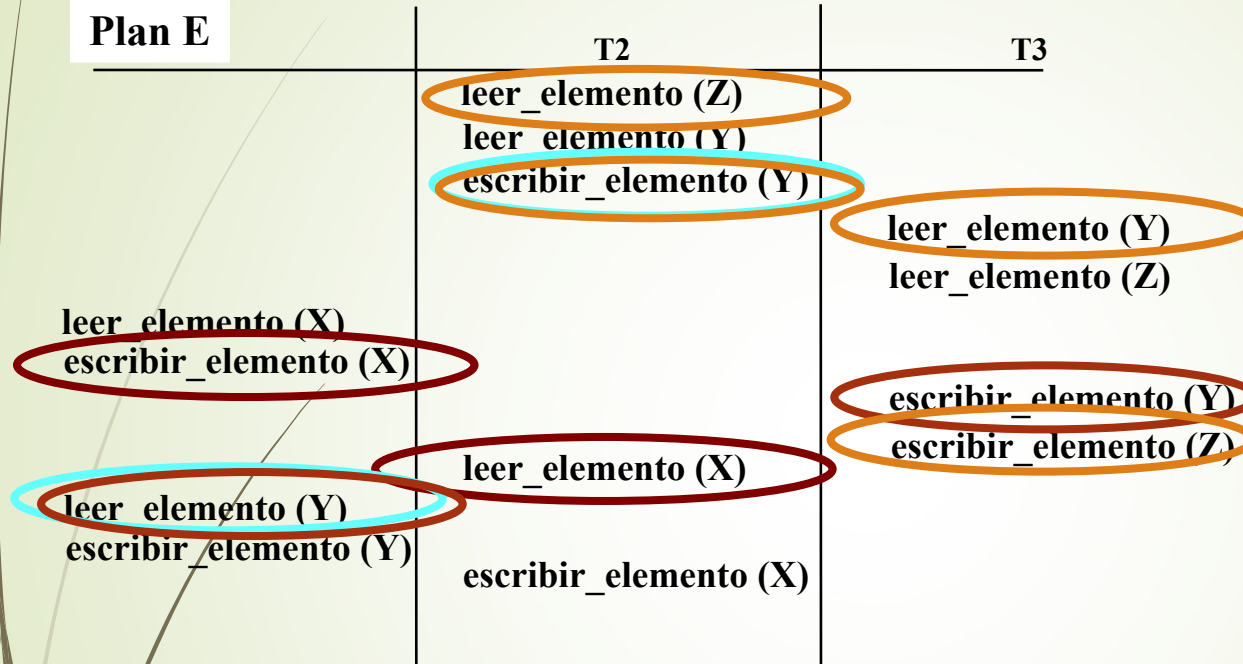


Plan C

plan serie equivalente: ninguno

Planes Serializables por Conflictos

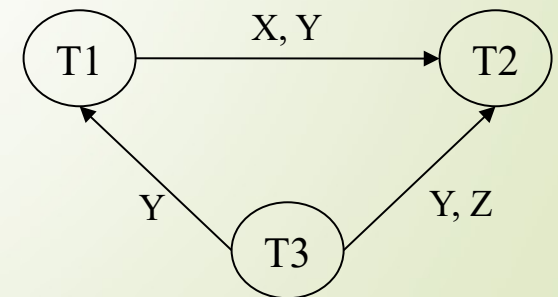
Plan E



plan serie equivalente: ninguno

Planes Serializables por Conflictos

Plan F	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

Seriabilidad de los planes

2.2) P y P' son **EQUIVALENTES POR VISTAS** \Rightarrow

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

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

(\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

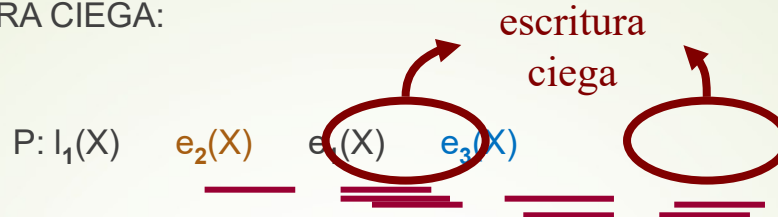
Planes Serializables por Vistas

Permiten ESCRITURA CIEGA:

T1: $I_1(X)$ $e_1(X)$

T2: $e_2(X)$

T3: $e_3(X)$



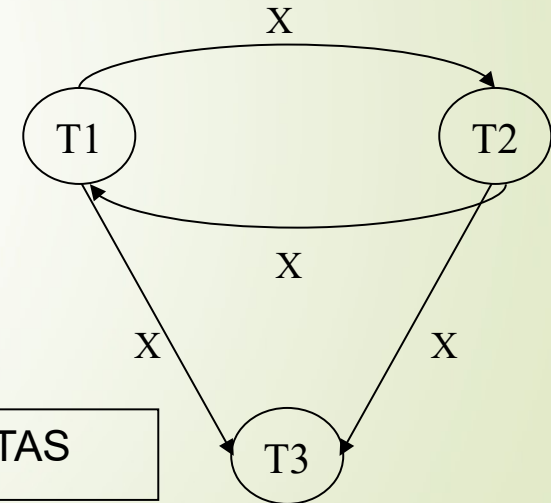
P es SERIALIZABLE POR VISTAS (equivalente por vistas a $P' = \{T1, T2, T3\}$)

T1T2T3: $I_1(X)$ $e_1(X)$ $e_2(X)$ $e_3(X)$

porque $I_1(X)$ es limpia en ambos planes y

$e_3(X)$ es la última escritura en ambos planes

pero NO ES SERIALIZABLE POR CONFLICTOS



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

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

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

(ESCRITURA RESTRINGIDA)

Ejemplo

Indicar si los planes siguientes son **serializables por conflictos** (SC) y/o **serializables por vistas** (SV)
Si no se puede decir si un plan pertenece a cierta clase según las acciones indicadas, explicarlo brevemente.

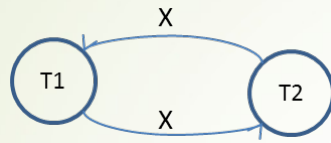
Las acciones aparecen en el orden consecutivo dentro de cada plan. En caso de que no aparezcan las confirmaciones de las transacciones, asumir que los *commit* / *abort* deberán encontrarse después de todas las acciones indicadas.

	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		

Ejemplo

P: I1(X) I2(X) e1(X) e2(X)

¿SERIALIZABLE POR CONFLICTOS?



Conclusión:

Existe un ciclo, por lo que no es serializable por conflictos

¿SERIALIZABLE POR VISTAS?

T1T2: I1(X) e1(X) I2(X) e2(X). No es equivalente, dado que en el plan en serie la lectura I2(X) es sucia, mientras que en el plan a comparar es limpia.

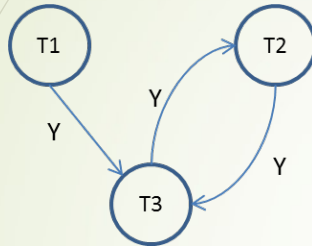
T2T1: I2(X) e2(X) I1(X) e1(X). No es equivalente, dado que en el plan en serie la lectura I1(X) es sucia, mientras que en el plan a comparar es limpia.

Conclusión: **NO es serializable por vistas**, dado que no existe ningún plan en serie equivalente a P por vistas.

Ejemplo

P: I1(X) I1(Y) e1(X) I2(Y) e3(Y) e1(X) I2(Y)

¿SERIALIZABLE POR CONFLICTOS?



Conclusión:

Existe un ciclo, por lo que no es serializable por conflictos

¿SERIALIZABLE POR VISTAS?

T1T2T3: I1(X) I1(Y) e1(X) e1(X) I2(Y) I2(Y) e3(Y) No es equivalente, dado que en el plan en serie la segunda lectura I2(Y) es limpia, mientras que en el plan a comparar es sucia.

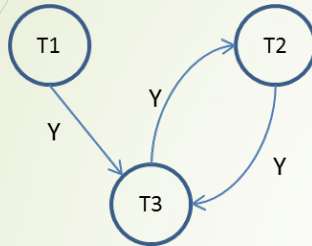
T2T1T3: I2(Y) I2(Y) I1(X) I1(Y) e1(X) e1(X) e3(Y). No es equivalente, dado que en el plan en serie la segunda lectura I2(Y) es limpia, mientras que en el plan a comparar es sucia.

T3T1T2: e3(Y) I1(X) I1(Y) e1(X) e1(X) I2(Y) I2(Y) No es equivalente, dado que en el plan en serie las lecturas I1(Y) y la primera I2(Y) son sucias, mientras que en el plan a comparar son limpias.

Ejemplo

P: I1(X) I1(Y) e1(X) I2(Y) e3(Y) e1(X) I2(Y)

¿SERIALIZABLE POR CONFLICTOS?



Conclusión:

Existe un ciclo, por lo que no es serializable por conflictos

¿SERIALIZABLE POR VISTAS?

T3T2T1: e3(Y) I2(Y) I2(Y) I1(X) I1(Y) e1(X) e1(X) No es equivalente, dado que en el plan en serie las lecturas I1(Y) y la primera I2(Y) son sucias, mientras que en el plan a comparar son limpias.

T1T3T2: I1(X) I1(Y) e1(X) e1(X) e3(Y) I2(Y) I2(Y) No es equivalente, dado que en el plan en serie la primera lectura I2(Y) es sucia, mientras que en el plan a comparar es limpia.

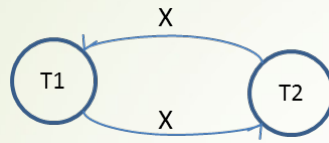
T2T3T1: I2(Y) I2(Y) e3(Y) I1(X) I1(Y) e1(X) e1(X). No es equivalente, dado que en el plan en serie la segunda lectura I2(Y) es limpia, mientras que en el plan a comparar es sucia.

Conclusión: **NO es serializable por vistas**, dado que no existe ningún plan en serie equivalente a P por vistas.

Ejemplo

P: I1(X) e2(X) e1(X) a2 c1

¿SERIALIZABLE POR CONFLICTOS?



Conclusión:

Existe un ciclo, por lo que no es serializable por conflictos

¿SERIALIZABLE POR VISTAS?

T1 T2: I1(X) e1(X) c1 e2(X) a2 Es equivalente porque la lectura de X por parte de T1 es limpia en ambos planes. Además, la última escritura, e1(X), es la misma (dado que T2 se anula).

T2 T1: e2(X) a2 I1(X) e1(X) c1 Es equivalente porque la lectura de X por parte de T1 es limpia en ambos planes (dado que T2 se anula). Además, la última escritura, e1(X), es la misma.

Conclusión: **Es serializable por vistas**, teniendo en cuenta que T2 se anula.

Bibliografía



- ▣ *Fundamentos de Sistemas de Bases de Datos. Ramez A. Elmasri, Shamkant B. Navathe* [cap. 19]
- ▣ *Fundamentos de Bases de Datos. A. Silberschatz. McGraw-Hill* [cap. 15]