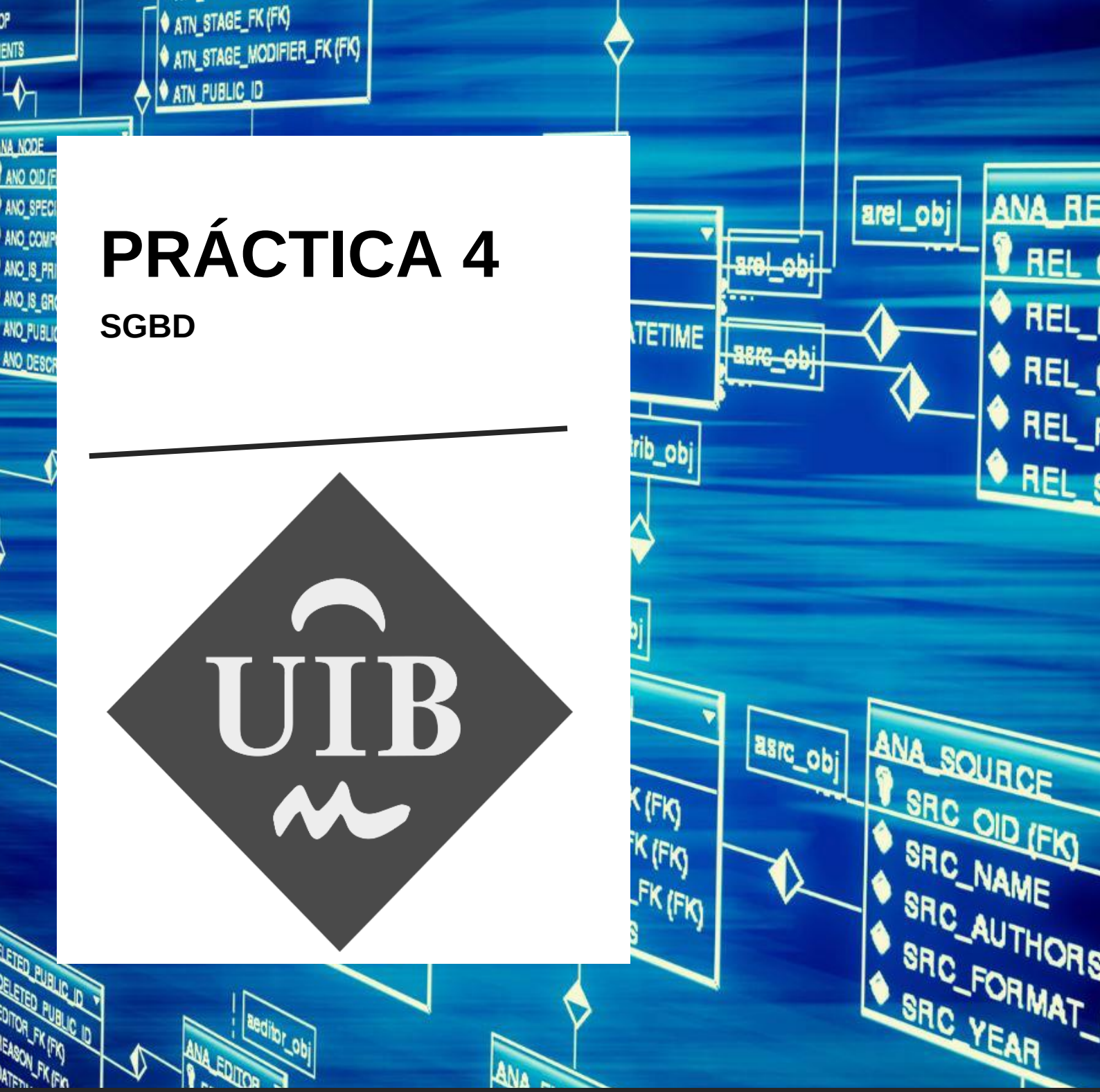

PRÁCTICA 4

SGBD

7 ENERO 2024

PRÁCTICA 4 – Undo & Redo log
Mario Ventura Burgos 43223476-J
Grado en Ingeniería Informática (GIN 3)

1. INTRODUCCIÓN

Esta práctica consta con dos partes en las que se nos pide lo siguiente:

PRIMERA PARTE

Se pide lo siguiente:

Begin transacción;
A := A * C;
B := D / 2;
C := A / D;
D := C;
commit;

Donde, los valores iniciales de A, B, C y D son:
A → 5
B → 2
C → 8
D → 10

Hacer la tabla de la aplicación de un diario Undo log, así como la explicación de todos y cada uno de los pasos que sean necesarios. Relacione los conceptos tanto con los procesos de los gestores, como de las partes de la memoria implicada.

SEGUNDA PARTE

Supongamos que tenemos el siguiente estado transaccional:

- Begin Transaction; A := A * 2; B := B * 2; Commit;
- Checkpoint;
- Begin Transaction; A := A * B; B := B * 2; Commit;
- Begin Transaction; A := A * 2; B := B * 2; Rollback;
- Begin Transaction; A := A * 4; B := B * 4;
- Checkpoint;
- Begin Transaction; A := A * B; B := B * 2; Rollback;
- Begin Transaction; A := A * 2; B := B * 2; Commit;
- Commit; -- De la transacción que faltaba.

Los valores iniciales son:

- A = 2
- B = 4

Supongamos que se ejecutan todos los pasos exceptuando el último commit, ya que el sistema cae. El tipo de diario es Redo log.

Se pide:

1. Especifica el contenido del periódico.
2. Especifica los valores de A y B en disco en el momento de la caída.
3. Aplica y explica el proceso de recuperación del sistema.

2. PRIMERA PARTE

Los diarios Undo log cuentan con los siguientes registros:

- Inici(T)
- Update(T, Bloque, ValorAntiguo)
- Commit(T)
- Rollback(T)

Recordemos que en estos diarios solo podemos deshacer o volver atrás las operaciones, pero no podemos volver a realizarlas. Por tanto, si una transacción modifica un bloque, debe escribir en el diario una entrada para esta modificación antes de escribir la modificación en el disco. Si una transacción se confirma, debe escribir en el diario la operación de COMMIT después de que todos los bloques modificados por la transacción se escriban en disco.

Así pues, la tabla de aplicación de un diario Undo log para la transacción dada es la siguiente:

Paso	Acción	t	t2	Memoria Caché				Disco				Diario
				A	B	C	D	A	B	C	D	
1								5	2	8	10	
2	READ(A,t)	5		5				5	2	8	10	Inici(T)
3	READ(C,t2)	5	8	5		8		5	2	8	10	
4	t:=t*t2	40	8	5		8		5	2	8	10	
5	WRITE(A,t)	40	8	40		8		5	2	8	10	Update(T,A,5)
6	READ(B,t)	2	8	40	2	8		5	2	8	10	
7	READ(D,t)	10	8	40	2	8	10	5	2	8	10	
8	t:=t/2	5	8	40	2	8	10	5	2	8	10	
9	WRITE(B,t)	5	8	40	5	8	10	5	2	8	10	Update(T,B,2)
10	READ(A,t)	40	8	40	5	8	10	5	2	8	10	
11	READ(D,t2)	40	10	40	5	8	10	5	2	8	10	
12	t:=t/t2	4	10	40	5	8	10	5	2	8	10	
13	WRITE(C,t)	4	10	40	5	4	10	5	2	8	10	Update(T,C,8)
14	READ(C,t)	4	10	40	5	4	10	5	2	8	10	
15	WRITE(D,t)	4	10	40	5	4	4	5	2	8	10	Update(T,D,10)
16	Escriure log											
17	OUTPUT(A)	4	10	40	5	4	4	40	2	8	10	
18	OUTPUT(B)	4	10	40	5	4	4	40	5	8	10	
19	OUTPUT(C)	4	10	40	5	4	4	40	5	4	10	
20	OUTPUT(D)	4	10	40	5	4	4	40	5	4	4	Commit(T)
21	Escriure log											

A continuación, la explicación de todos los pasos realizados:

- **PASO 1** → Inicializamos los valores para A,B,C, y D del disco. Estos valores son: A=5, B=2, C=8, D=10.

- **PASO 2: READ(A,t)** → En el diario se anota "Inicio(T)", indicando el inicio de una nueva transacción T. El proceso Database Writer (DBWR de ahora en adelante) se invoca para que cargue el bloque A, de la memoria caché que se encuentra en la server global area (SGA de ahora en adelante), en la variable t, situada en el program global area (PGA de ahora en adelante). En

resumen, se podría entender como que "leemos el valor de A (5) y lo guardamos/cargamos en la variable t, que ahora tiene el valor 5".

- **PASO 3: READ(C,t2)** → Se invoca al proceso DBWR para cargar el bloque C (caché de la SGA) en la variable t2 (PGA), necesaria para no sobrescribir el valor de t y así poder realizar la operación esencial A*C. Puede entenderse como que se almacena en t2 el resultado de una lectura del valor de C por parte del proceso DBWR. Ahora C tiene el valor 8 asignado en la caché.

- **PASO 4: t:=t*t2** → El valor de t se multiplica por el de t2, obteniendo t=5*8=40. Es una operación intermedia y por tanto se realiza en la PGA. Esencialmente, se ejecuta la operación A=A*C;

- **PASO 5: WRITE(A,t)** → Invocamos al proceso servidor y el valor de A se ve modificado en la caché (ahora es 40) pero todavía no en el disco (sigue siendo 5) y observamos como el valor que teníamos en la PGA (t) pasa al SGA (A en la memoria caché). El protocolo para este tipo de diarios dicta que, si una transacción modifica un bloque, se debe escribir en el diario una entrada para esta modificación antes de escribir la modificación en el disco. Por ello, este paso corresponde con la escritura en el diario del registro Update(T,A,5), donde 5 representa el valor antiguo de A para la transacción T (recordemos que tras la operación explicada en el paso 4, ahora el valor de A es 40).

- **PASO 6: READ(B,t)** → El proceso DBWR cargará el valor del bloque B (situado en la caché de la SGA) y lo almacenará en la variable t, que se encuentra en la PGA y ahora tendrá el valor de B (2). Ahora B tiene el valor 2 asignado en la caché. Dado que se pretende hace la operación esencial B:=D/2, puede parecer que leer el valor de B no es necesario. Sin embargo, podemos prever que será necesario escribir un registro Update en el diario, que al ser de tipo Undo log, este registro tendrá la estructura "Update(T, BloqueID, ValorAntiguo)", donde *BloqueID* es "B" y *ValorAntiguo* es el valor de B previo a ese Update. Sin embargo, resulta imposible saber el valor antiguo de B sin hacer antes una lectura. Esta es la razón por la que se realiza esta lectura de B en el paso 6, de forma que podamos conocer el valor de B y almacenarlo, luego se pueda leer el valor de D, hacer la operación D/2 y escribir en el diario el registro Update con el antiguo valor de B. Esta lectura no sería necesaria si el diario fuese de tipo Redo log.

- **PASO 7: READ(D,t)** → Otra lectura realizada por parte del DBWR, que lee/carga el valor del bloque D y lo almacena en la variable t, situados en la caché de la SGA y en la PGA respectivamente.

- **PASO 8: t:=t/2** → Nuevamente, se trata de una operación intermedia y, por tanto, tiene lugar en la PGA. Dado que t tiene el valor leído de D, este paso representa la operación B:=D/2=10/2=5, cuyo resultado se guardará en t para posteriormente ser escrito en B.

- **PASO 9: WRITE(B,t)** → Se escribe el resultado de la operación D/2 (almacenado en t) en B. Para ello, el proceso servidor modifica en la caché el valor de B, que ahora es 5, pero todavía no en el disco, donde el valor sigue siendo 2. Nuevamente, tendremos que añadir al diario un registro Update dado que estamos actualizando el valor de un bloque ID. El bloque ID es B y su valor antiguo era 2, por eso en la tabla se puede ver como este paso corresponde con el registro Update(T,B,2).

- **PASO 10: READ(A,t**) → El proceso DBWR carga el valor del bloque A y lo almacena en la variable t, situados respectivamente en la caché de la SGA y en la PGA. Ahora t tendrá el valor 40 asignado.

- **PASO 11: READ(D,t2)** → Otra lectura por parte del proceso DBWR. Este proceso cargará el valor del bloque D, situado en la memoria caché de la SGA, en la variable t2, situada en la PGA. Una vez más, dado que queremos realizar la operación C:=A/D, es necesario tener el valor de A y el de D, y por tanto es necesario usar dos variables (t en el paso 10 y t2 en el paso 11).

- **PASO 12: t:=t/t2** → Se realiza la operación de división, que esencialmente representa la operación C:=A/D. El resultado de la división (40/10=4) se guardará en t para posteriormente ser cargado/escrito en la memoria caché de algún bloque.

- **PASO 13: WRITE(C,t)** → Se escribe en C el resultado de la operación, que se encuentra almacenado en t. Para ello, el proceso servidor modifica en la caché el valor de C, que ahora es 4, pero todavía no en el disco, donde el valor sigue siendo 8. Una vez más, este paso corresponde con la escritura en el diario de un registro de Update de la forma Update(T,C,8), indicando que se actualiza en la transacción T el valor de C, que antes era 8.

- **PASO 14: READ(C,t)** → El proceso DBWR cargará el valor del bloque C de la memoria caché de la SGA en la variable t situada en la PGA, de forma que ahora t tiene asignado el valor 4.

- **PASO 15: WRITE(D,t)** → Se llama al proceso servidor para que escriba en D el valor de t, que contiene el valor del bloque C leído en el paso anterior por parte del proceso DBWR. Como en todos los casos anteriores, las acciones de WRITE implican la escritura en diario de un registro Update con el valor antiguo del bloque modificado. Por ello, puede verse en la tabla como este paso coincide con la escritura en diario del registro Update(T,D,10), ya que el valor de D antes era 10.

- **PASO 16: Escriure log** → Se invoca al proceso Log Writer (LOGWR de ahora en adelante), que consolida los valores del diario en el disco.

- **PASO 17: OUTPUT(A)** → Se invoca al proceso DBWR para registrar en el disco el valor del bloque A. Este proceso escribirá el bloque con el valor modificado en disco.

- **PASO 18 OUTPUT(B)** → Se invoca al proceso DBWR para registrar en el disco el valor del bloque B. Este proceso escribirá el bloque con el valor modificado en disco.

- **PASO 19 OUTPUT(C)** → Se invoca al proceso DBWR para registrar en el disco el valor del bloque C. Este proceso escribirá el bloque con el valor modificado en disco.

- **PASO 20: OUTPUT(D)** → Se invoca al proceso DBWR para registrar en el disco el valor del bloque D. Este proceso escribirá el bloque con el valor modificado en disco. Este paso coincide con la escritura en el diario del registro Commit(T), indicando que se da por finalizada la transacción. De ahora en adelante se sabe que todos los registros que intervienen en la transacción son correctos, dado que esta transacción ha acabado correctamente.

- **PASO 21: Escriure log** → Ahora que la transacción ha acabado, se vuelven a consolidar los valores del diario en disco mediante el proceso LOGWR, dado que la primera vez que se escribió el log, todavía no se había escrito el registro de commit en el diario y, por tanto, este commit todavía no ha sido consolidado en disco.

3. SEGUNDA PARTE

Se plantea la siguiente situación transaccional:

- Begin Transaction; A := A * 2; B := B * 2; Commit;
- Checkpoint;
- Begin Transaction; A := A * B; B := B * 2; Commit;
- Begin Transaction; A := A * 2; B := B * 2; Rollback;
- Begin Transaction; A := A * 4; B := B * 4;
- Checkpoint;
- Begin Transaction; A := A * B; B := B * 2; Rollback;
- Begin Transaction; A := A * 2; B := B * 2; Commit;
- Commit; -- De la transacción que faltaba.

Los valores iniciales son:

- A = 2
- B = 4

A continuación, se detallará la solución propuesta para cada una de las cuestiones planteadas:

1.1 Especificar el contenido del diario

En este caso, debido a que se trata de 6 transacciones que se ejecutan consecutivamente con checkpoints en ciertos puntos, el valor en disco obtenido tras acabar una transacción se "guardará" de cara a la próxima transacción. Es decir, una transacción comenzará con los valores en disco de A y B que la transacción anterior ha escrito. Por ello creamos unos "identificadores" para cada transacción, que serán T1, T2, T3, T4, T5 y T6 para las transacciones de 1 a 6 respectivamente, y los identificadores CH1 y CH2 para los checkpoints 1 y 2 respectivamente. Podemos afirmar que T2 depende directamente de T1 (en cuanto a que los valores de A y B usados por T2 son los escritos por T1), T3 de T2, y así sucesivamente.

El contenido del diario para cada transacción es el siguiente:

- Begin Transaction; A := A * 2; B := B * 2; Commit;

Paso	Acción	Variable t	Memoria Caché		Disco		Diario
			A	B	A	B	
1					2	4	
2	READ(A,t)	2	2		2	4	Inici(T1)
3	t=t*2	4	2		2	4	
4	WRITE(A,t)	4	4		2	4	Update(T1,A,4)
5	READ(B,t)	4	4	4	2	4	
6	t=t*2	8	4	4	2	4	
7	WRITE(B,t)	8	4	8	2	4	Update(T1,B,8)
8							Commit(T1)
9	Escriure log						
10	OUTPUT(A)	8	4	8	4	4	
11	OUTPUT(B)	8	4	8	4	8	

- **CH1:** Dado que no hay transacciones activas, no se escribe nada entre los paréntesis escritos en las entradas del diario. El checkpoint inicia y acaba “de inmediato” dado que T1 ha acabado y se ha confirmado y, de momento, no hay ninguna otra transacción activa.

Paso	Acción	Variable t	Memoria Caché		Disco		Diario
			A	B	A	B	
12	Inicio del checkpoint 1				4	8	<Inici PC()>
13	Fin del checkpoint 1				4	8	<Fi Pc>

- Begin Transaction; A := A * B; B := B * 2; Commit;

Paso	Acción	Variable t	Variable t2	Memoria Caché		Disco		Diario
				A	B	A	B	
14						4	8	
15	READ(A,t)	4		4		4	8	Inici(T2)
16	READ(B,t2)	4	8	4	8	4	8	
17	t:=t*t2	32	8	4	8	4	8	
18	WRITE(A,t)	32	8	32	8	4	8	Update(T2,A,32)
19	READ(B,t2)	32	8	32	8	4	8	
20	t2:=t2*2	32	16	32	8	4	8	
21	WRITE(B,t2)	32	16	32	16	4	8	Update(T2,B,16)
22								Commit(T2)
23	Escriure log							
24	OUTPUT(A)	32	16	32	16	32	8	
25	OUTPUT(B)	32	16	32	16	32	16	

- Begin Transaction; A := A * 2; B := B * 2; Rollback;

Paso	Acción	Variable t	Memoria Caché		Disco		Diario
			A	B	A	B	
26					32	16	
27	READ(A,t)	32	32		32	16	Inicio(T3)
28	t=t*2	64	32		32	16	
29	WRITE(A,t)	64	64		32	16	Update(T3,A,64)
30	READ(B,t)	16	64	16	32	16	
31	t=t*2	32	64	16	32	16	
32	WRITE(B,t)	32	64	32	32	16	Update(T3,B,32)
33							Rollback(T3)
34	Escriure log						

- Begin Transaction; A := A * 4; B := B * 4;

Paso	Acción	Variable t	Memoria Caché		Disco		Diario
			A	B	A	B	
35					32	16	
36	READ(A,t)	32	32		32	16	Inicio(T4)
37	t=t*4	128	32		32	16	
38	WRITE(A,t)	128	128		32	16	Update(T4,A,128)
39	READ(B,t)	16	128	16	32	16	
40	t=t*4	64	128	16	32	16	
41	WRITE(B,t)	64	128	64	32	16	Update(T4,B,64)

- **CH2:** Independientemente del tipo de PC, entre el inicio y el final se deberán escribir todos los bloques modificados por transacciones ya confirmadas pero que aún no se han escrito en disco. En este caso, el punto de control 2 (o checkpoint 2) no tiene que escribir nada ya que T1 y T2 ya se han confirmado con Commit y han escrito en disco, T3 ha acabado con Rollback y T4 no se ha confirmado y aún sigue activa.

Paso	Acción	Variable t	Memoria Caché		Disco		Diario
			A	B	A	B	
42	Inicio del checkpoint 2				32	16	<Inici PC(T4)>
43	Fin del checkpoint 2				32	16	<Fi PC>

- Begin Transaction; A := A * B; B := B * 2; Rollback;

Paso	Acción	Variable t	Variable t2	Memoria Caché		Disco		Diario
				A	B	A	B	
44						32	16	
45	READ(A,t)	32		32		32	16	Inicio(T5)
46	READ(B,t2)	32	16	32	16	32	16	
47	t:=t*t2	512	16	32	16	32	16	
48	WRITE(A,t)	512	16	512	16	32	16	Update(T5,A,512)
49	READ(B,t2)	512	16	512	16	32	16	
50	t2:=t2*2	512	32	512	16	32	16	
51	WRITE(B,t2)	512	32	512	32	32	16	Update(T5,B,32)
52								Rollback(T5)
53	Escriure log							

- Begin Transaction; A := A * 2; B := B * 2; Commit;

Paso	Acción	Variable t	Memoria Caché		Disco		Diario
			A	B	A	B	
54					32	16	
55	READ(A,t)	32	32		32	16	Inici(T6)
56	t:=t*2	64	32		32	16	
57	WRITE(A,t)	64	64		32	16	Update(T6,A,64)
58	READ(B,t)	16	64	16	32	16	
59	t:=t*2	32	64	16	32	16	
60	WRITE(B,t)	32	64	32	32	16	Update(T6,B,32)
61							Commit(T6)
62	Escriure log						
63	OUTPUT(A)	32	64	32	64	16	
64	OUTPUT(B)	32	64	32	64	32	

Paso	Acción	Variable t	Memoria Caché		Disco		Diario
			A	B	A	B	
65	CAÍDA DEL SISTEMA PREVIA AL COMMIT DE T4						

1.2 Valores de A y B en el momento de la caída

Como se puede observar en la tabla correspondiente a la transacción anterior (T6), en el momento en el que sucede la caída del sistema previa al commit, los valores de A y B son:

- A = 64
- B = 32

Los valores de A y B se deben a que, tal y como se puede ver en las tablas, el proceso Database Writer (DBWR) consolida en los pasos 63 y 64 de la transacción T6 (OUTPUT(A) y OUTPUT(B) respectivamente) los valores de A y B en disco justo antes de la caída del sistema en el paso 65.

1.3 Aplicar y explicar el proceso de recuperación del sistema

El proceso de recuperación del sistema en un diario de tipo Redo log consta de 2 pasos o fases:

1. **Fase de análisis** → En esta fase se determina la lista de transacciones confirmadas y que pueden tener cambios pendientes de aplicar. El objetivo de esto es establecer el punto del diario a partir del cual debemos empezar a aplicar los cambios aprovechando los registros de Update.
2. **Fase de Redo (roll forward)** → Esta segunda fase consiste en recorrer el diario desde el punto establecido en la fase de análisis y, para cada anotación de Update(T,BloqueID,ValorNuevo), si T es una transacción confirmada, se escribe el valor en el bloque.

Si aplicamos este protocolo a nuestro caso concreto, sucede lo siguiente:

1. **Fase de análisis** → Se determina la lista de transacciones confirmadas con cambios pendientes de aplicar y el punto del diario a partir del cual empezar a aplicar los cambios. En nuestro caso tenemos las siguientes transacciones en el siguiente estado:
    - T1: Transacción confirmada con COMMIT y sin cambios pendientes de aplicar.
    - T2: Transacción confirmada con COMMIT y sin cambios pendientes de aplicar.
    - T3: Transacción que acaba en ROLLBACK.
    - T4: Transacción que sigue activa.
    - T5: Transacción que acaba en ROLLBACK.
    - T6: Transacción confirmada con COMMIT y sin cambios pendientes de aplicar.

Por tanto, las únicas transacciones que podrían ser "candidatas" serían T1, T2 y T6, que son las únicas transacciones confirmadas. Sin embargo, todas ellas son transacciones sin cambios pendientes de aplicar. Como consecuencia, y dado el estado transaccional proporcionado, y la información de los checkpoints, el punto del diario a partir del cual empezar a aplicar los cambios en este caso es después de que T6 ya haya finalizado y haya sido confirmada, es decir, después del Commit de T6. Por lo tanto, el punto de inicio para aplicar los cambios en la posterior fase de Redo sería después de la confirmación de T6, es decir, posterior al paso 64 (último estado consistente la base de datos).

2. **Fase de Redo (roll forward)** → Dado que T6 es la última transacción y ya ha sido confirmada y no tiene cambios pendientes de aplicar, y teniendo en cuenta que no hay ningún registro de Update(T,Bloque,Valor) que haya sido escrito en el diario después de que acabe T6, se puede concluir que no habrá cambios pendientes por aplicar después de T6. Por lo tanto, en la fase de Redo (roll forward), no se realizarán cambios adicionales, ya que todos los cambios realizados por T6 ya están confirmados y aplicados en la base de datos.

   Pese a que T4 no ha acabado, no podemos volver atrás para rehacer todas las operaciones desde el inicio de T4, ya que eso implicaría deshacer las operaciones confirmadas y consolidadas por parte de T6 y eso no es posible en un diario de tipo Redo log. En este tipo de diarios, no tenemos anotaciones en el diario del valor antiguo de los bloques que modifica T6, únicamente tenemos información sobre el valor nuevo que se le dio, y por tanto deshacer las operaciones no es una opción.