Chapter 09. Recuperación de fallos.Técnicas.

GBD / BDA

DEFINICIONES

- Recuperación de la BD: restablecimiento de un estado correcto de la BD (consistente) después que un fallo del sistema haya ocasionado que el estado actual sea inconsistente.

- ¿Quién se encarga de la recuperación? La recuperación la gestiona el módulo gestor de recuperación del SGBD

OBJETIVOS

- El sistema de recuperación se ocupa de que se cumplan dos de las propiedades ACID de las transacciones:

  - Atomicidad: se ejecutan todas las acciones o ninguna.
  - Durabilidad: cuando una transacción se confirma los cambios realizados deben permanecer en el sistema.

Fichero de log/diario

El sistema se mantiene al tanto de lo que hacen las transacciones mediante un fichero especial llamado diario o log.

Cada registro almacenado en el log se llama entrada, y será uno de los siguientes tipos.

<INICIAR, T> Indica que la transacción ha comenzado

<LEER, T, x> Indica que la transacción leyó el valor del elemento x de la BD.

<ESCRIBIR, T, x, valor_antiguo, valor_nuevo> Indica que la transacción ha cambiado el valor del elemento x; x contenía el valor_antiguo antes de la escritura y contendrá el valor nuevo después de la misma.

<COMMIT, T> Se confirma la transacción.

<ROLLBACK, T> La transacción ha sido anulada (abortada)

Funcionamiento del módulo de recuperación

- Los algoritmos de recuperación son técnicas para asegurar la consistencia de la BD y la atomicidad de las transacciones incluso en presencia de fallos.

- El sistema de recuperación se apoya en el log para realizar su función (que guarda todas las acciones realizadas.)

LOG: punto de control (o de sincronismo)

- Periódicamente se escribe en el log una entrada especial llamada punto de control, indicando que en ese momento el sistema escribe en el disco todos los *buffers* del SGBD que han sido modificados en la BD por las transacciones en curso (y que temporalmente estaban en memoria). En inglés *checkpoint.*

- El SBGD debe decidir el intervalo entre puntos de control (medido en tiempo o número de transacciones).

- Realizar un punto de control implica:
  1. Suspensión temporal de la ejecución de las transacciones.
  2. Escritura forzada de todos los *buffers* a disco.
  3. Escribir una entrada especial en el log [punto de control].
  4. Reactivar las transacciones en ejecución.

LOG: punto de control (o de sincronismo)

- Los puntos de control evitan tener que recorrer todo el log para la recuperación, lo que resultaría muy costoso y realizaría acciones innecesarias.

- Cuando se realiza (y registra) un punto de control, se escribe toda la información a disco y se guarda la información de las transacciones activas en ese momento.

Tipos de fallos

Fallos no catastróficos

Fallos de transacciones:

- Errores lógicos: una transacción no puede completarse por algún error interno a la misma.
- Errores del sistema: una transacción es abortada por el SGBD.

Fallos por imposición del control de la concurrencia.

Fallo del sistema (caída suave): errores de HW, SW, o de conexión.

Fallos catastróficos

- Fallo de disco
- Problemas físicos y catástrofes

Algoritmos de recuperación

Conceptualmente podemos distinguir dos **técnicas** principales para recuperarse frente a fallos no catastróficos:

- Actualización diferida
- Actualización inmediata

Las técnicas de actualización diferida no actualizan la BD hasta llegar al punto de confirmación.

En las técnicas de actualización inmediata las operaciones de una transacción modifican la BD hasta que la transacción se confirme.

Estudiaremos tres algoritmos que se basan en combinar acciones de las técnicas anteriores.

Actualización diferida: Algoritmo no deshacer/rehacer

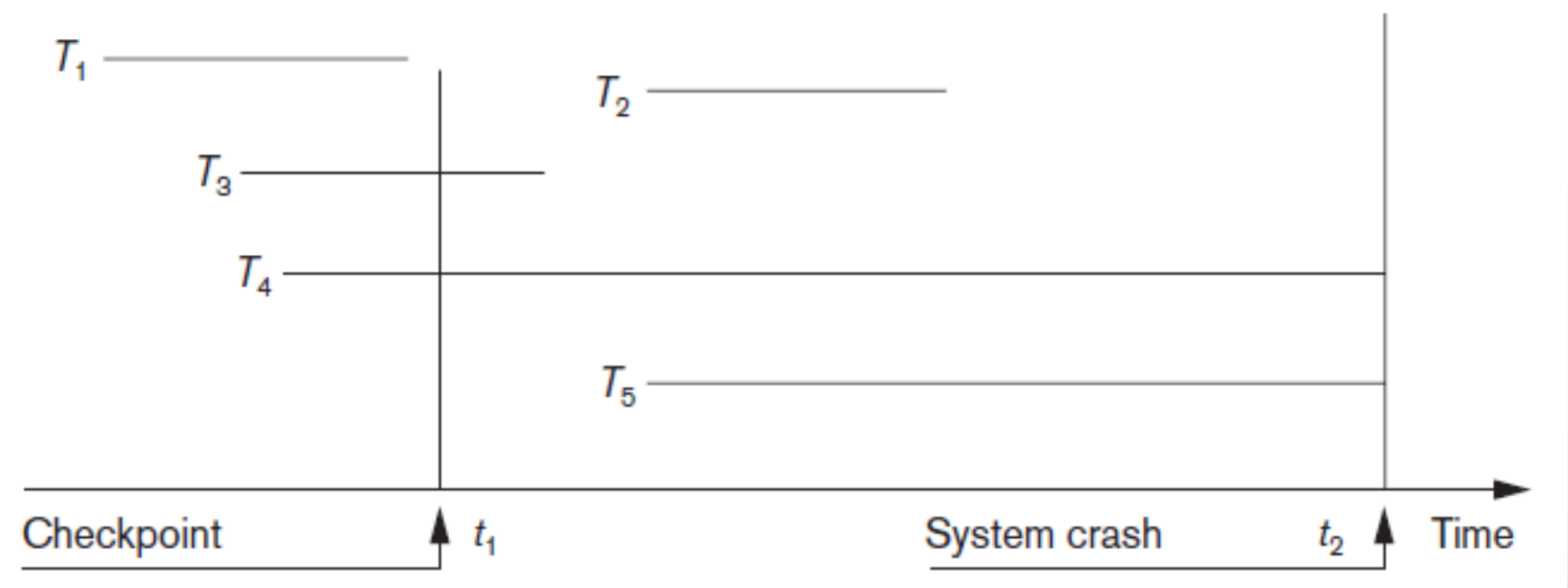
Procedimiento cuando ocurre un fallo

El algoritmo usa dos listas de transacciones: las confirmadas desde el último *checkpoint* (lista de confirmadas), y las transacciones activas (lista de activas). Rehace todas las transacciones confirmadas en el mismo orden en el cual fueron escritas en el log. Las transacciones activas no confirmadas son canceladas y reejecutadas de nuevo.

Actualización diferida: Algoritmo no deshacer/rehacer

- Con esta técnica nunca es necesario deshacer ninguna transacción después de un fallo, porque no han llegado a tener efecto.
- La escritura en disco de los datos actualizados se posterga hasta llegar al punto de confirmación de la transacción.
- Abortar transacciones es muy barato.
- Durante el proceso de recuperación después de una caída una transacción debe rehacerse si y solo si el log contiene las entradas de inicio y de confirmación de dicha transacción.
- Una transacción nunca leerá el valor de un elemento que es actualizado por otra transacción no confirmada, ya que los elementos permanecen bloqueados hasta que la transacción alcanza el punto de confirmación.

Actualización diferida: Algoritmo no deshacer/rehacer



Explica qué ocurre con cada transacción.

Actualización diferida: Algoritmo no deshacer/rehacer

T_1
read_item(A)
read_item(D)
write_item(D)

T_2
read_item(B)
write_item(B)
read_item(D)
write_item(D)

T_3
read_item(A)
write_item(A)
read_item(C)
write_item(C)

T_4
read_item(B)
write_item(B)
read_item(A)
write_item(A)

[start_transaction,T_1]
[write_item, T_1, D, 20]
[commit, T_1]
[checkpoint]
[start_transaction, T_4]
[write_item, T_4, B, 15]
[write_item, T_4, A, 20]
[commit, T_4]
[start_transaction, T_2]
[write_item, T_2, B, 12]
[start_transaction, T_3]
[write_item, T_3, A, 30]
[write_item,T_2, D, 25] ← System crash

T_2 and T_3 are ignored because they did not reach their commit points.

T_4 is redone because its commit point is after the last system checkpoint.

An example of recovery using deferred update with concurrent transactions :The READ and WRITE operations of four transactions and the System log at the point of crash.

¿Cuál es el valor de A, B, C y D?

Actualización diferida: Algoritmo no deshacer/rehacer

- Si la traza en tres instantes de tiempo al fallar es:

(a)	(b)	(c)
<T_0 start>	<T_0 start>	<T_0 start>
<T_0, A, 950>	<T_0, A, 950>	<T_0, A, 950>
<T_0, B, 2050>	<T_0, B, 2050>	<T_0, B, 2050>
	<T_0 commit>	<T_0 commit>
	<T_1 start>	<T_1 start>
	<T_1, C, 600>	<T_1, C, 600>
		<T_1 commit>

la recuperación en cada caso es como sigue

Actualización inmediata

- Cuando una transacción lleva a cabo una operación de actualización, la base de datos en disco es actualizada sin tener que esperar a que la transacción se haya confirmado.

- Existen dos algoritmos para esta técnica:
  - Algoritmo deshacer / no-rehacer
  - Algoritmo deshacer / rehacer

Actualización inmediata

- Estos algoritmos usan dos listas de transacciones: las activas y las confirmadas desde el último *checkpoint*.
- Para el primer caso (deshacer/no-rehacer): **Deshace** todas las operaciones de escritura de las transacciones que están en la lista de activas. Las operaciones deben ser deshechas en el orden inverso al que se escribieron en el log. **No Rehace** las transacciones que están en la lista de confirmadas.

Actualización inmediata

- Para el segundo caso (deshacer/rehacer): **Deshace** todas las operaciones de escritura de las transacciones que están en la lista de activas. Las operaciones deben ser deshechas en el orden inverso al que se escribieron en el log. **Rehace** todas las operaciones de escritura de las transacciones que están en la lista de confirmadas, en el orden en el cual fueron escritas en el log.

Actualización inmediata

- ¿Cómo se lleva a cabo el procedimiento de deshacer?

Deshacer una operación de escritura consiste en examinar su entrada en el registro [escribir, T, x, valor_antiguo, valor nuevo] y establecer el valor del elemento X en la base de datos al valor antiguo.

Deshacer un número de operaciones de escritura de una o más transacciones del registro debe proceder en el orden inverso al orden en que se escribieron las operaciones en el registro.

Consideraciones

- El algoritmo deshacer/rehacer es el más comúnmente empleado por los SGBD.
- Las operaciones de deshacer y de rehacer son idempotentes.
- Se realizan primero las operaciones de deshacer y luego las de rehacer.

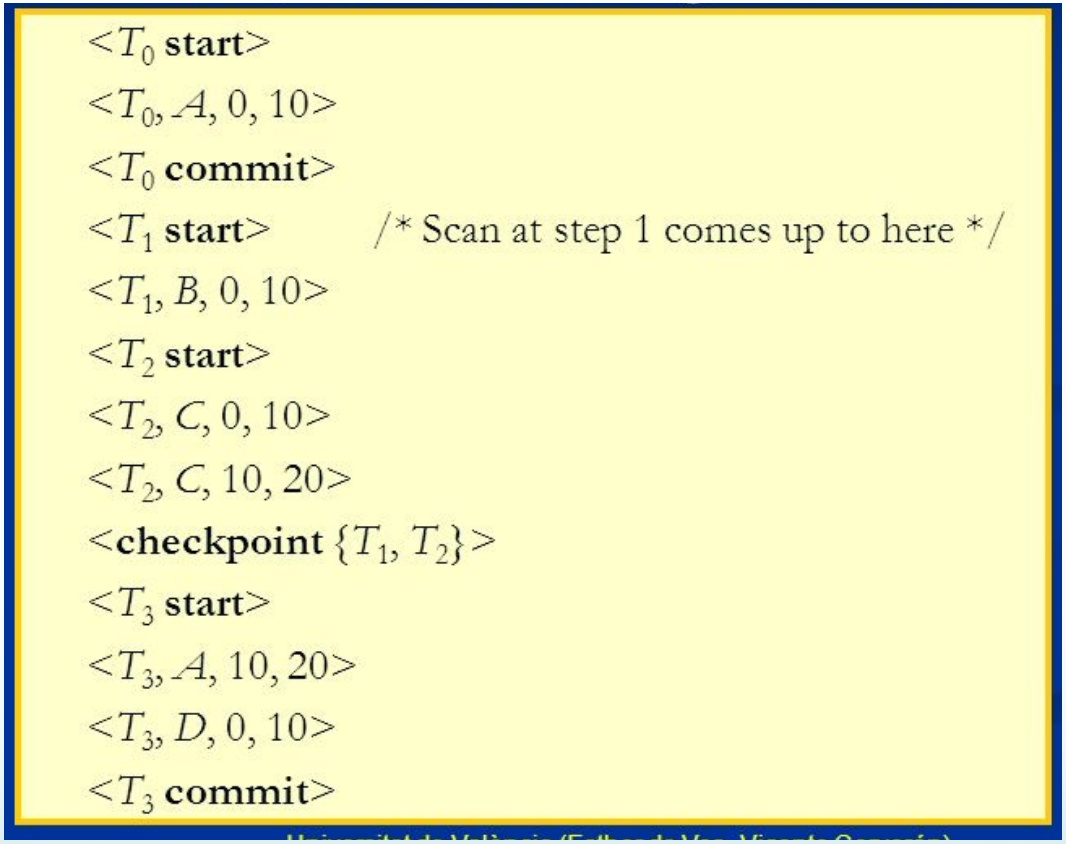
Ejemplo algoritmo deshacer/rehacer

Si el log en tres instantes de tiempo al fallar es:

<T_0 start>	<T_0 start>	<T_0 start>
<T_0, A, 1000, 950>	<T_0, A, 1000, 950>	<T_0, A, 1000, 950>
<T_0, B, 2000, 2050>	<T_0, B, 2000, 2050>	<T_0, B, 2000, 2050>
	<T_0 commit>	<T_0 commit>
	<T_1 start>	<T_1 start>
	<T_1, C, 700, 600>	<T_1, C, 700, 600>
		<T_1 commit>

¿Cómo es la recuperación en cada caso?

Ejemplo algoritmo deshacer/rehacer



¿Cuál es el valor de A, B, C y D después del fallo?