TRANSACCIONES

TRANSACCIONES

Una transacción es una unidad de la ejecución de un programa que accede y posiblemente actualiza varios elementos de datos.

TRANSACCIONES

Atomicidad: Todas las operaciones de la transacción se realizan adecuadamente en la base de datos o ninguna de ellas.

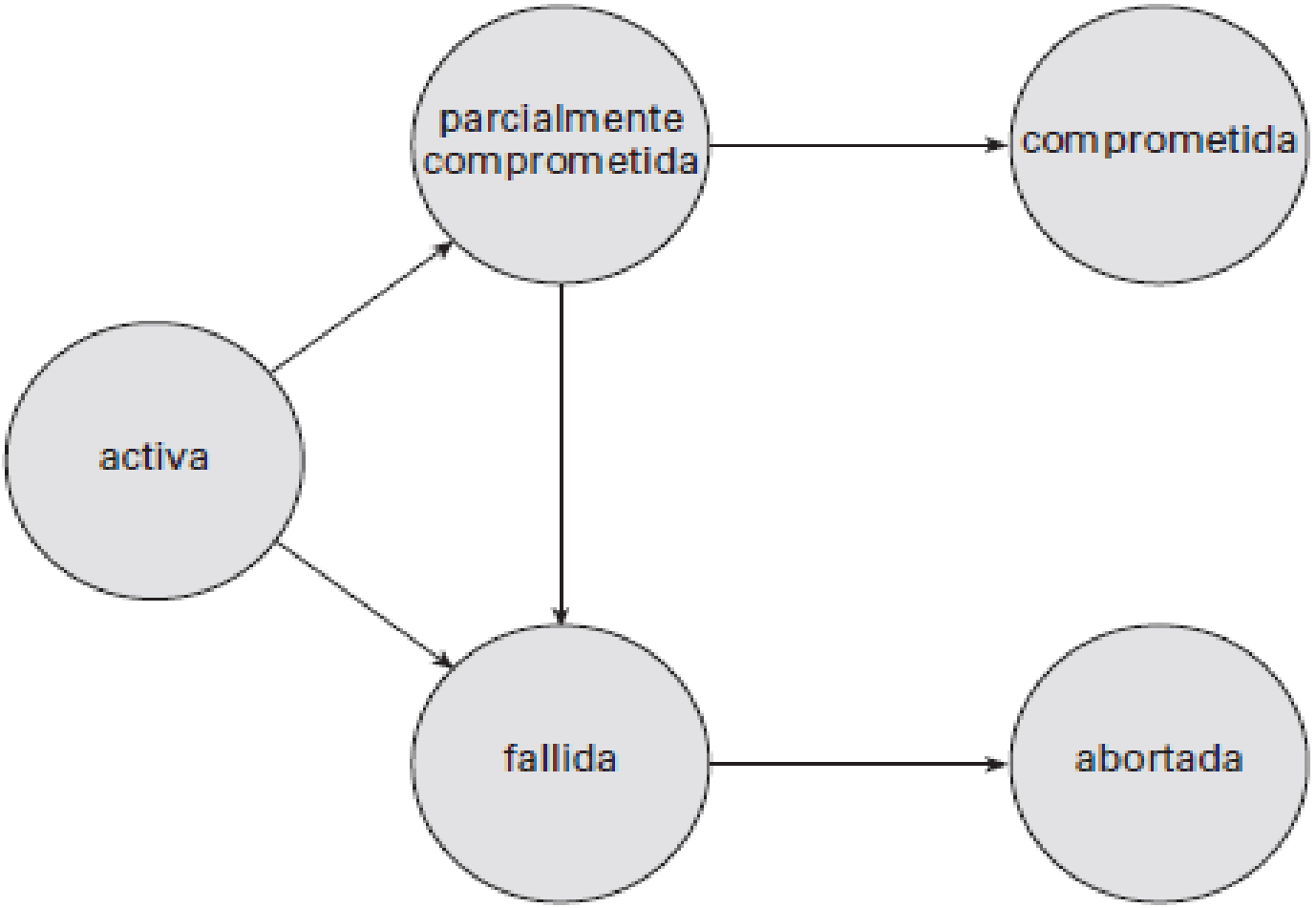
• **Consistencia**. La ejecución aislada de la transacción (es decir, sin otra transacción que se ejecute concurrentemente) conserva la consistencia de la base de datos.

TRANSACCIONES

• **Aislamiento.** Aunque se ejecuten varias transacciones concurrentemente, el sistema garantiza que para cada par de transacciones Ti y Tj, se cumple que para los efectos de Ti, o bien Tj ha terminado su ejecución antes de que comience Ti , o bien que Tj ha comenzado su ejecución después de que Ti termine. De este modo, cada transacción ignora al resto de las transacciones que se ejecuten concurrentemente
en el sistema.

Durabilidad. Tras la finalización con éxito de una transacción, los cambios realizados en la base de datos permanecen, incluso si hay fallos en el sistema

ESTADOS DE UNA TRANSACCION

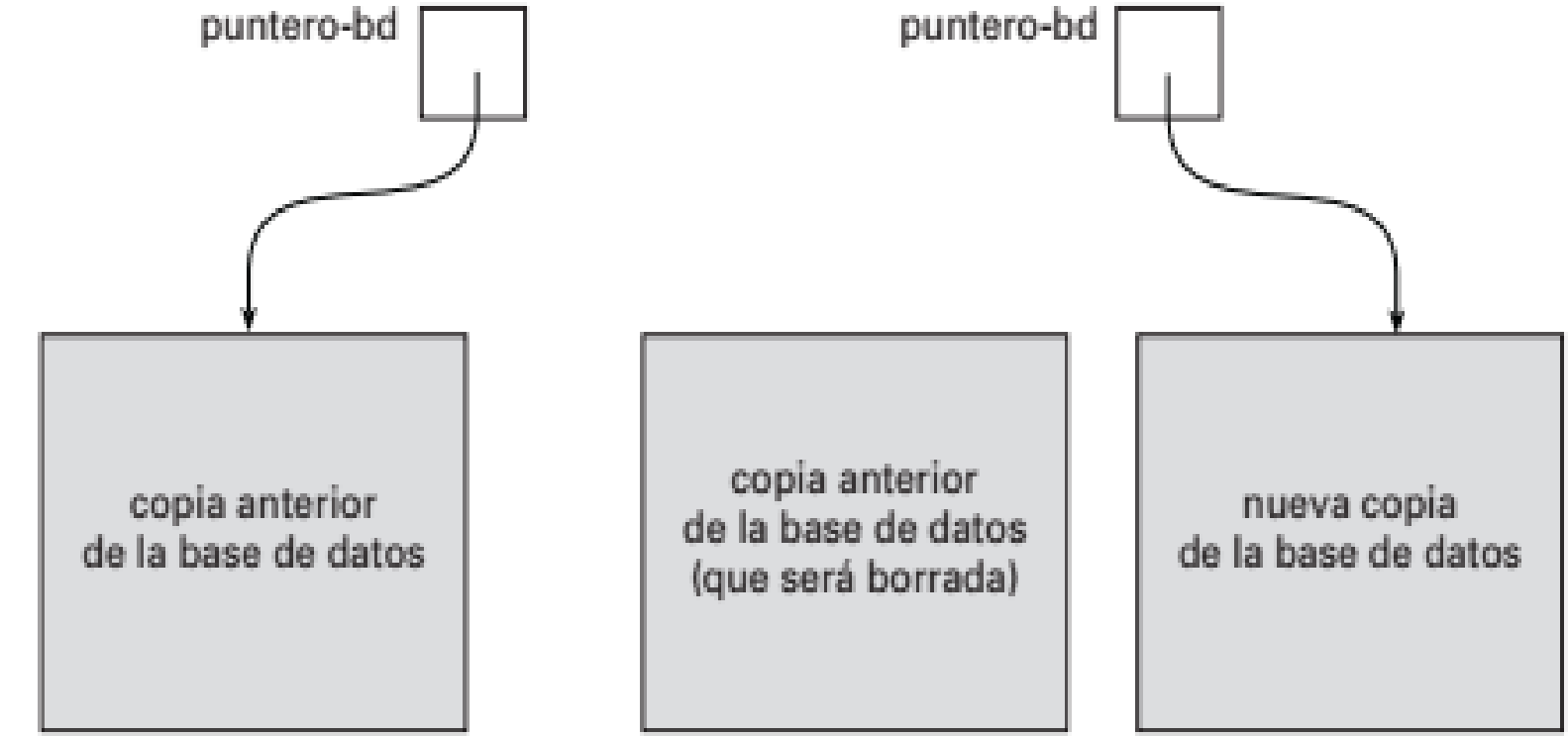


TRANSACCIONES

T_i: leer(A);
$A := A - 50$;
escribir(A);
leer(B);
$B := B + 50$;
escribir(B).

Puesto que una transacción es una **unidad** que conserva la consistencia, una ejecución secuencial de transacciones garantiza que se conserva dicha consistencia.

Atomicidad y durabilidad



(a) Antes de la actualización

(b) Después de la actualización

EJECUCIONES CONCURRENTES

- Productividad y ejecución de recursos
- Tiempo de espera reducido
- Multiprogramación

T_1: leer(A);
$A := A - 50$;
escribir(A);
leer(B);
$B := B + 50$;
escribir(B).

T_2: leer(A);
$temp := A * 0.1$;
$A := A - temp$;
escribir(A);
leer(B);
$B := B + temp$;
escribir(B).

EJECUCIONES CONCURRENTES

T_1	T_2
	leer(A)
	$temp := A * 0.1$
	$A := A - temp$
	escribir(A)
	leer(B)
	$B := B + temp$
	escribir(B)
leer(A)	
$A := A - 50$	
escribir(A)	
leer(B)	
$B := B + 50$	
escribir(B)	

T_1	T_2
leer(A)	
$A := A - 50$	
escribir(A)	
leer(B)	
$B := B + 50$	
escribir(B)	
	leer(A)
	$temp := A * 0.1$
	$A := A - temp$
	escribir(A)
	leer(B)
	$B := B + temp$
	escribir(B)

Dos planificaciones diferentes para las transacciones T1 y T2

Transacciones Concurrentes

T_1	T_2
leer(A)	
$A := A - 50$	
escribir(A)	
	leer(A)
	$temp := A * 0.1$
	$A := A - temp$
	escribir(A)
leer(B)	
$B := B + 50$	
escribir(B)	
	leer(B)
	$B := B + temp$
	escribir(B)

El componente de *gestión de* ***control de concurrencia*** de la base de datos es el responsable de manejar transacciones concurrentes.

EJECUCIONES CONCURRENTES

- Cuando varias transacciones se ejecutan **concurrentemente** en la base de datos, puede que deje de conservarse la ***consistencia*** de los datos. Es, por tanto necesario que el sistema controle la interacción entre las transacciones concurrentes.

- **Una planificación** captura las acciones clave de las transacciones que afectan a la ejecución concurrente, tales como las operaciones leer y escribir, a la vez que se abstraen los detalles internos de la ejecución de la transacción

EJECUCIONES CONCURRENTES

Es necesario que toda planificación producida por el procesamiento concurrente de un conjunto de transacciones tenga el efecto equivalente a una planificación en la cual esas transacciones se ejecutan secuencialmente en un cierto orden. Un sistema que asegure esta propiedad se dice que asegura la secuencialidad.

AISLAMIENTO

Bloqueo de transacciones

Una transacción realiza un bloqueo en la base de datos antes de comenzar y lo libera después de haberse comprometido. Mientras una transacción mantiene el bloqueo no se permite que ninguna otra lo obtenga y todas ellas deben esperar hasta que se libere el bloqueo.

TRANSACCIONES EN SQL

Commit (Compromete la transacción)

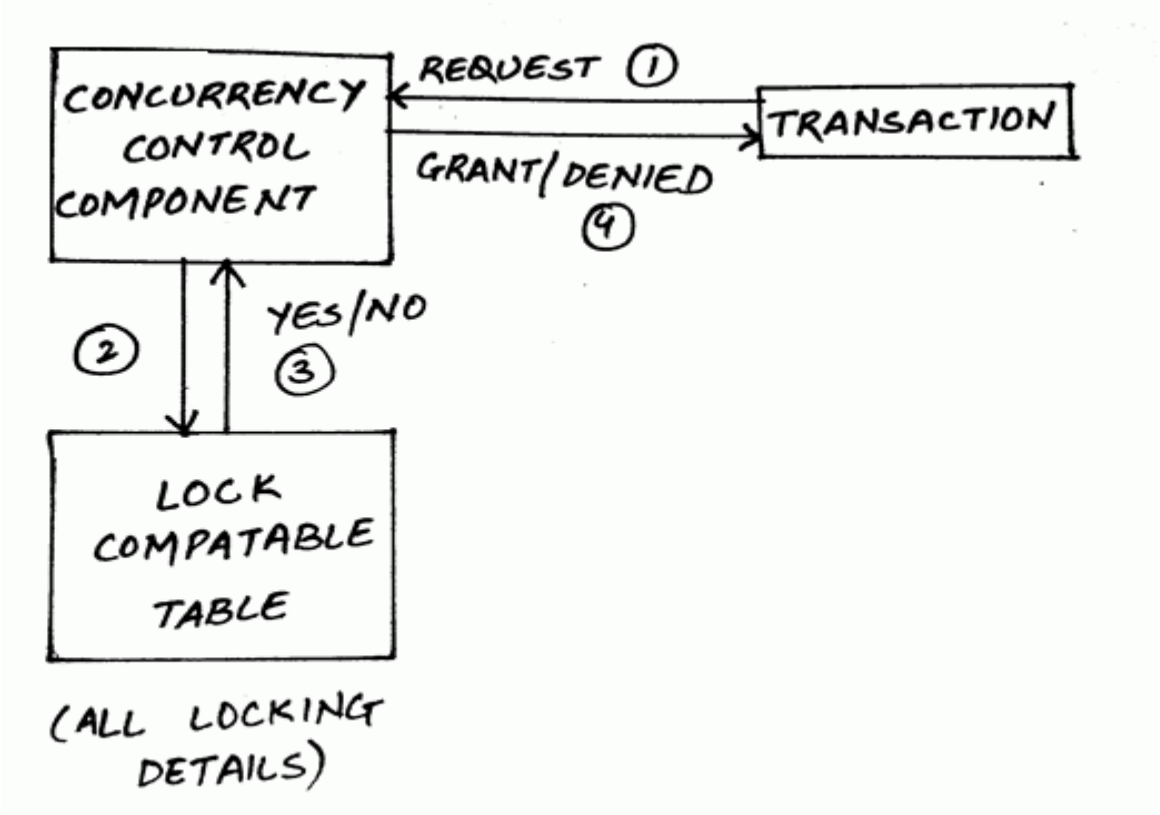
Rollback (Se regresa a un estado anterior)

TRANSACCIONES EN SQL

```
BEGIN TRANSACTION:

        UPDATE cuentas set balance = balance -100000
         WHERE cuenta_id=120;

         IF @@error <> 0
                ROLLBACK

         ELSE

                        COMMIT
END;
```

CONTROL DE CONCURRENCIA

- El encargado de gestionar la concurrencia en una base de datos es el módulo de Control de Concurrencia.



CONTROL DE CONCURRENCIA

Protocolos basados en el Bloqueo

Una forma de asegurar la ***secuencialidad*** es exigir que el acceso a los elementos de datos se haga en ***exclusión mutua***; es decir, mientras una transacción accede a un elemento de datos, ninguna otra transacción puede modificar dicho elemento

Un ***bloque***o es un mecanismo comúnmente utilizado para resolver problemas de sincronización de acceso a información compartida, acopiada en una base de datos

GESTOR DE BLOQUEOS

Un gestor de bloqueos se puede implementar como un proceso que recibe mensajes de transacciones y envía mensajes de respuesta.

GESTOR DE BLOQUEOS

Bloqueo de dos fases:

El protocolo de bloqueo de dos fases permite que una transacción bloquee un nuevo elemento de datos sólo ha desbloqueado ningún otro elemento de datos. (Bloqueo - Desbloqueo)

CONTROL DE CONCURRENCIA

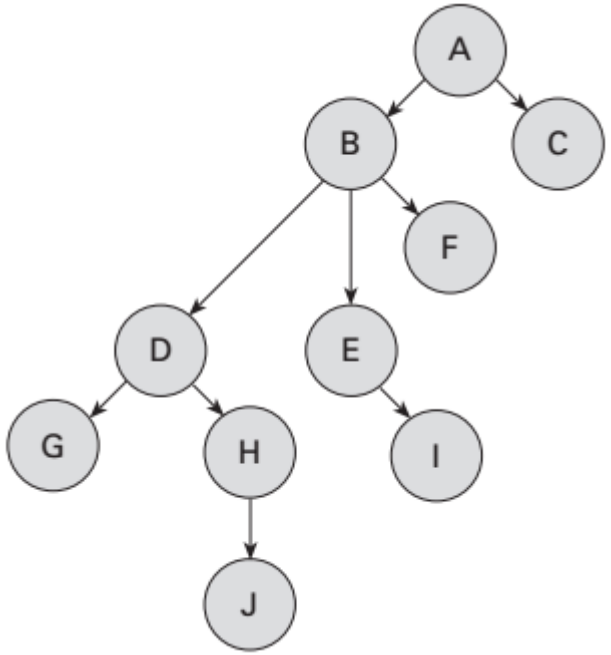
Bloqueos.

Compartido. Si una transacción Ti obtiene un bloqueo en modo compartido (denotado por C) sobre el elemento Q, entonces Ti puede leer Q pero no lo puede escribir.

Exclusivo. Si una transacción Ti obtiene un bloqueo en modo exclusivo (denotado por X) sobre el elemento Q, entonces Ti puede tanto leer como escribir Q.

GESTOR DE BLOQUEOS

Protocolos basados en Grafos



El primer bloqueo de Ti puede ser sobre cualquier elemento de datos.

Posteriormente, Ti puede bloquear un elemento de datos Q sólo si Ti está bloqueando actualmente al padre de Q.

Los elementos de datos bloqueados se pueden desbloquear en cualquier momento.

Ti no puede bloquear de nuevo un elemento de datos que ya haya bloqueado y desbloqueado anteriormente.

GESTOR DE BLOQUEOS

Protocolos basados marcas temporales

A toda transacción Ti del sistema se le asocia una única marca temporal fijada, denotada por **MT**(Ti). El sistema de base de datos asigna esta marca temporal antes
de que comience la ejecución de Ti. Si a la transacción Ti se le ha asignado la marca temporal MT(Ti) y una nueva transacción Tj entra en el sistema, entonces
MT(Ti) < MT(Tj).

- Usar el valor del reloj del sistema
- Usar un contador lógico