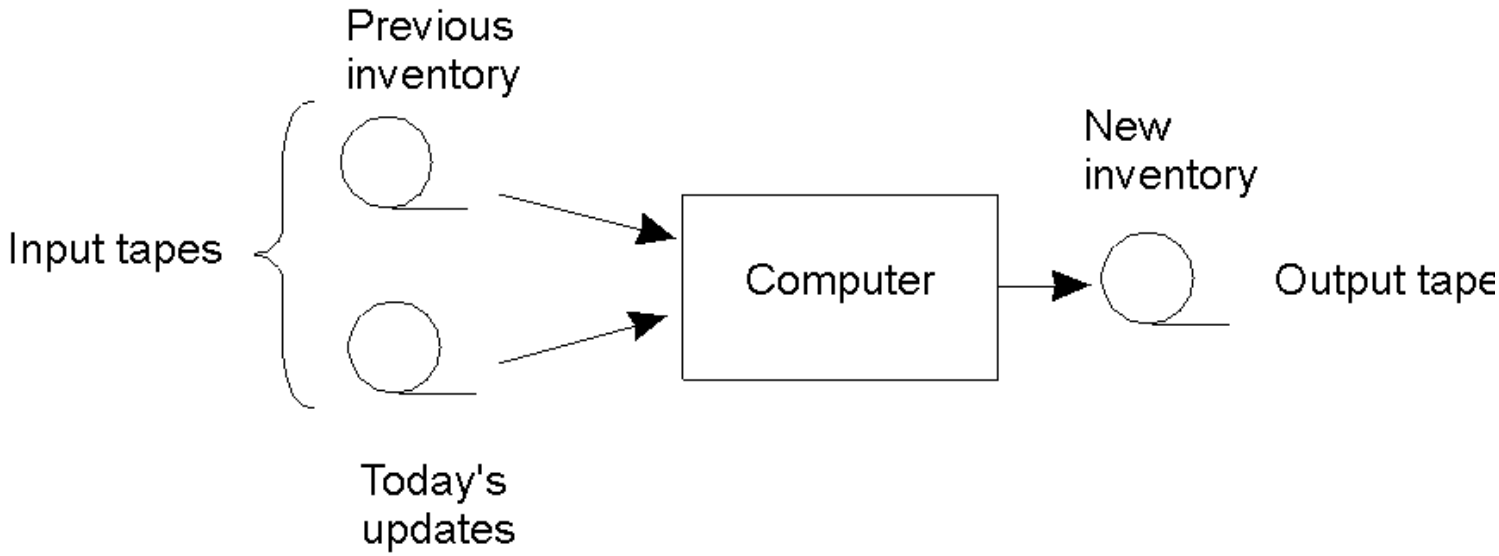
Transacciones Distribuidas

Transacciones Distribuidas

Modelo Transaccional

- La actualización de una cinta maestra es tolerante a las fallas.



Transacciones Distribuidas

¿Qué es una transacción?

- Base de datos
- Sistema de archivos
- Objetos
- Cliente

Transacciones Distribuidas

Transacciones Atómicas

Propiedades

- Serializabilidad: Las transacciones concurrentes no interfieren unas con otras.
- Atomicidad: Desde el mundo exterior la transacción es indivisible. (todo o nada).
- Permanencia: Una vez que la transacción se ejecuta los cambios son permanentes.

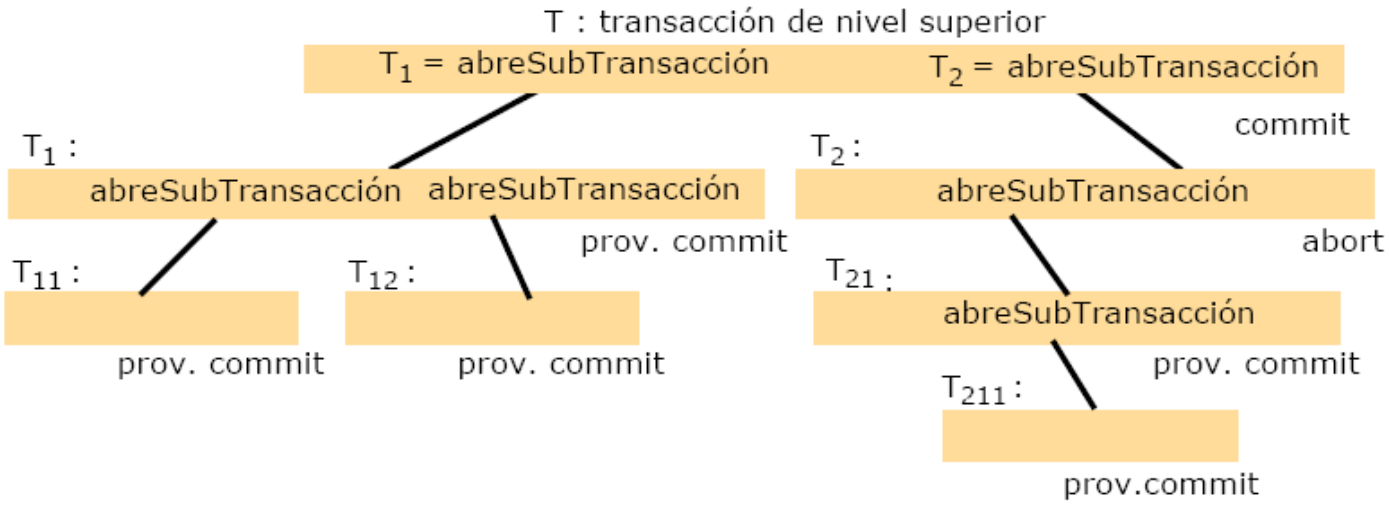
Transacciones Distribuidas

Tipos de Transacciones

- Planas
- Anidadas

Transacciones Distribuidas

Transacciones Anidadas.

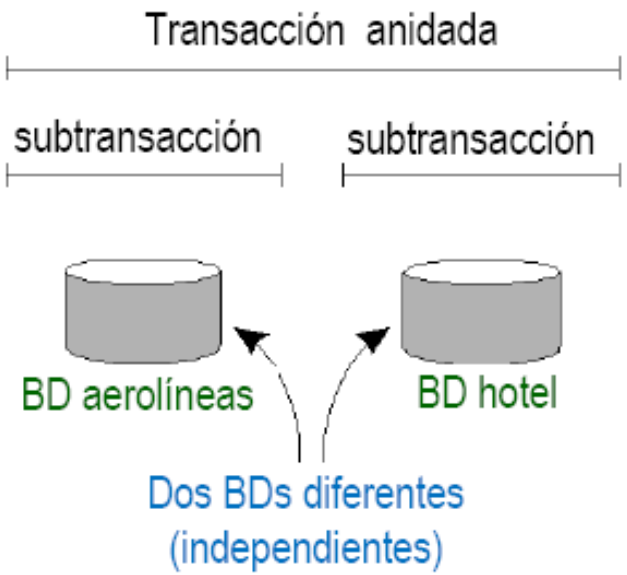


Transacciones Distribuidas

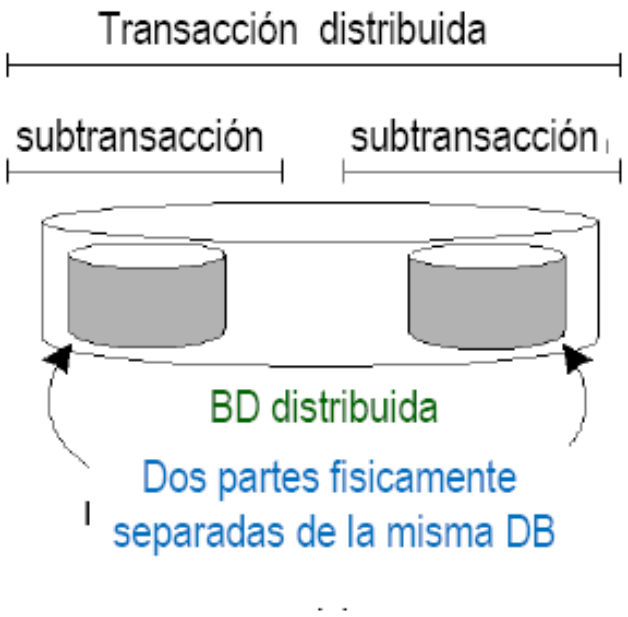
- Se usa el término **transacciones distribuidas** para referirse a **transacciones planas o anidadas que acceden a objetos administrados por múltiples servidores**.
- Cuando una transacción distribuida llega a su fin, la propiedad de atomicidad de las transacciones requiere que todos los servidores involucrados produzcan el **commit** (consumación) de la transacción o todos ellos la abortan.
- La manera en que el coordinador logra esto, depende del protocolo elegido.

Transacciones Distribuidas

- Transacciones Distribuidas



Una transacción anidada



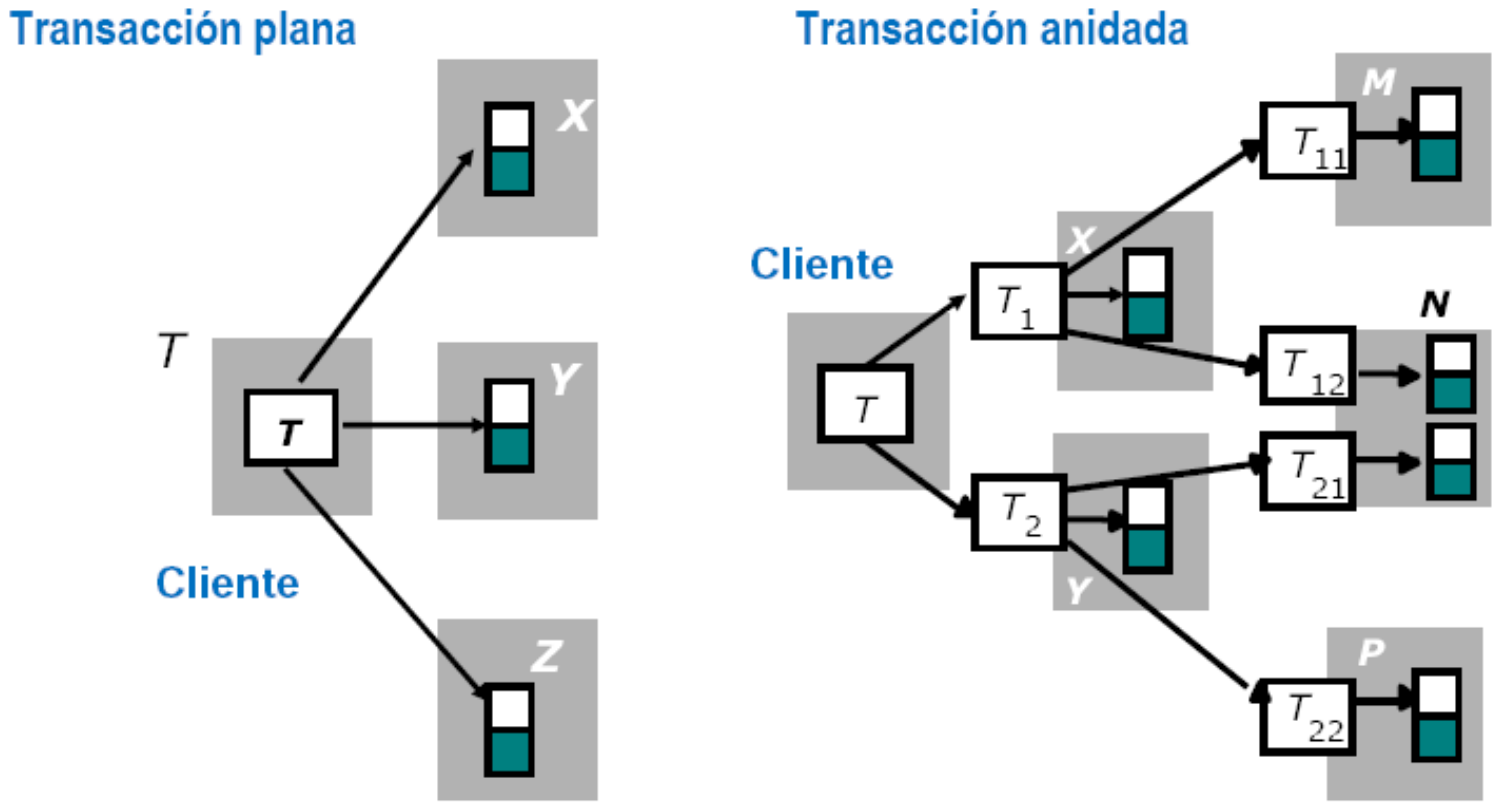
Una transacción distribuida

Transacciones Distribuidas

Transacciones distribuidas planas y anidadas

- En una **transacción plana**, el cliente hace requerimientos a más de un servidor. Cada transacción accede a los objetos en los servidores secuencialmente.
- El cliente de la transacción plana espera completar todos sus requerimientos antes de pasar a la próxima.
- En una **transacción anidada**, la transacción de mayor nivel puede abrir subtransacciones y, a su vez cada subtransacción puede abrir otras en niveles más bajos de anidamiento.

Transacciones Distribuidas

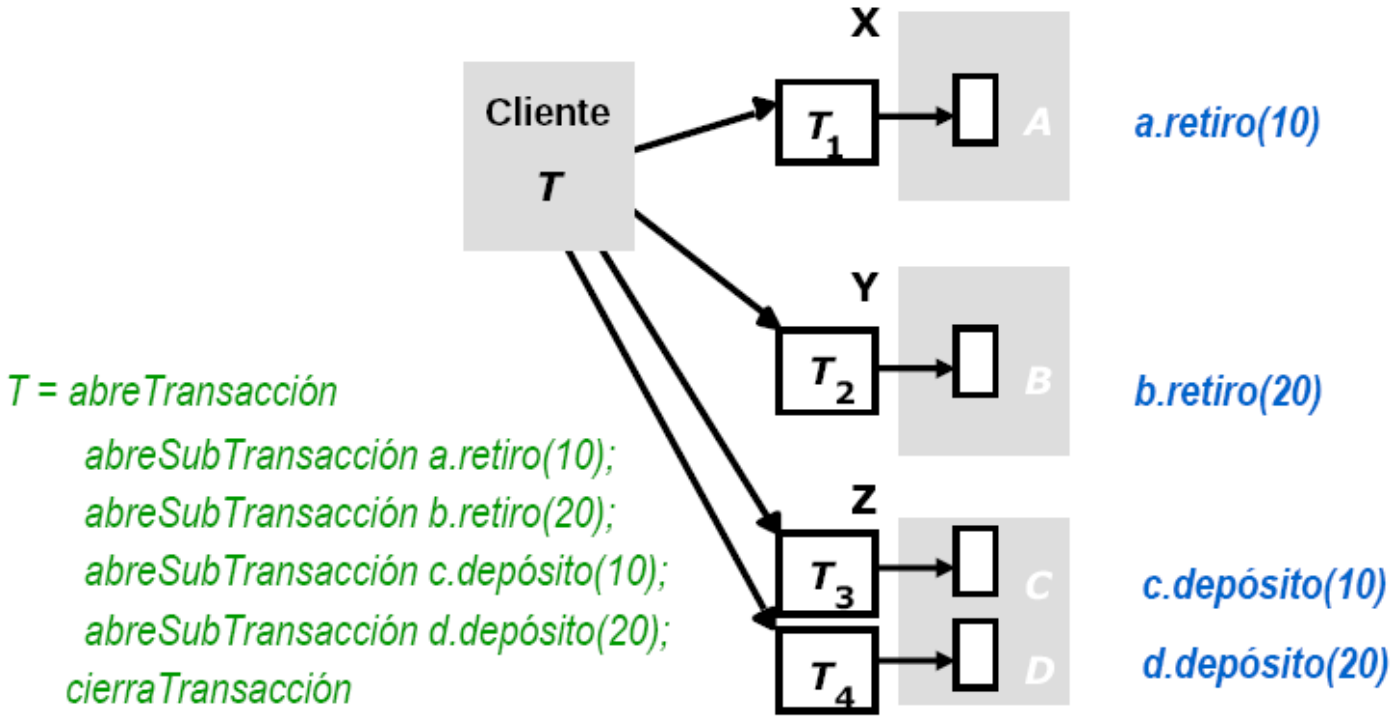


Transacciones Distribuidas

Ejemplo de una transacción bancaria anidada

- Sea una transacción distribuida donde el cliente transfiere $10 de la cuenta A a C y $20 de B a D.
- Las cuentas A y B están separadas en servidores X e Y y las cuentas C y D están en el servidor Z.
- Si la transacción se estructura como un conjunto de cuatro transacciones anidadas, los cuatro requerimientos (dos depósitos y dos retiros) pueden correr en paralelo y el efecto total es lograr mayor rendimiento que una transacción simple ejecutando las cuatro operaciones secuencialmente.

Transacciones Distribuidas



Transacciones Distribuidas

El Coordinador de una Transacción Distribuida

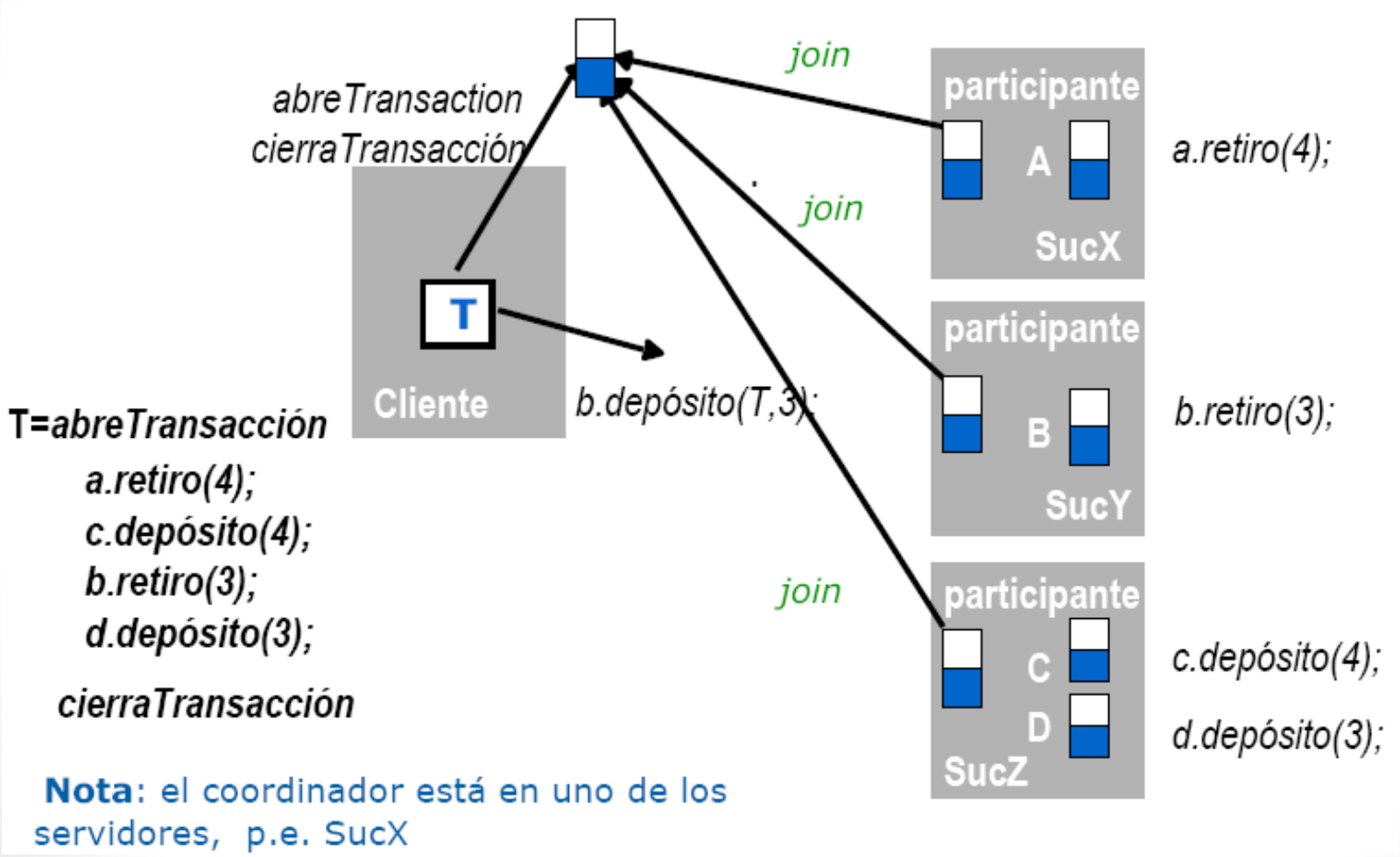
- Los servidores que ejecutan requerimientos que son parte de una transacción distribuida necesitan poder comunicarse con otros para coordinar sus acciones cuando la transacción **commits**.
- Un clienta comienza la transacción enviando un requerimiento **abreTransacción** a un coordinador en algún servidor.
- El coordinador que es contactado lleva adelante la **abreTransacción** y retorna un identificador al cliente (éste debe ser único).
- El coordinador que abre la transacción se convierte en el coordinador para la transacción distribuida.

Transacciones Distribuidas

El Coordinador de una Transacción Distribuida

- Cada uno de los servidores que administra un objeto accedido por la transacción es un participante en la transacción, se llamará participante.
- Los participantes son responsables en la cooperación con el coordinador para llevar adelante el protocolo de commit de la transacción.
- Durante el progreso de la transacción, el coordinador registra los participantes y éstos registran al coordinador.
- Ejemplo: Un cliente cuya transacción (plana) involucra las cuentas A, B, C y D en los servidores SucX, SucY y SucZ.
- La transacción T del cliente transfiere $3 de la cuenta B a D y $4 de A a C.

Transacciones Distribuidas

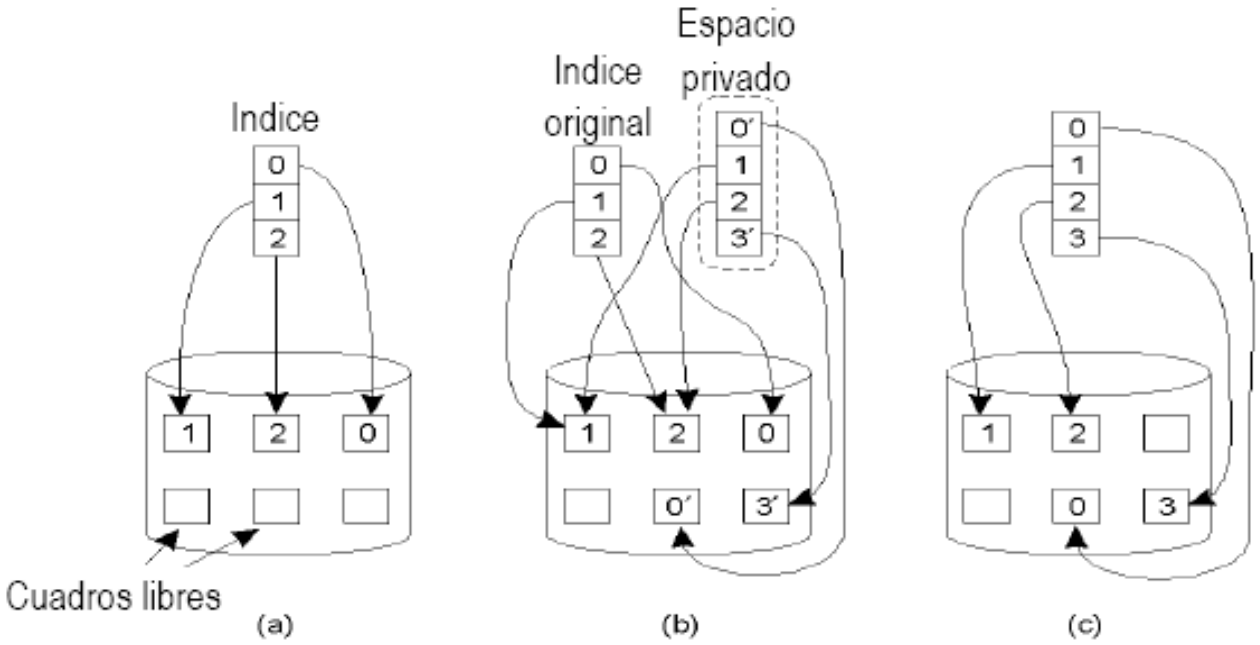


Transacciones Distribuidas

Implementación - Espacio privado

a) Índice y bloques de disco (3) de un archivo

b) La situación luego que una transacción ha modificado el bloque 0 y c) Luego del commit



Transacciones Distribuidas

Escritura con bitácora

a) Una transacción

b) – d) La bitácora luego de ejecutar la sentencia

	Bitácora	Bitácora	Bitácora
x = 0;			
y = 0;			
BEGIN_TRANSACTION;			
x = x + 1;	[x = 0 / 1]	[x = 0 / 1]	[x = 0 / 1]
y = y + 2		[y = 0/2]	[y = 0/2]
x = y * y;			[x = 1/4]
END_TRANSACTION;			
(a)	(b)	(c)	(d)

Transacciones Distribuidas

Protocolos de Commit atómicos

- Una transacción llega a su fin cuando los requeri-mientos del cliente han llegado a commit o abort.

Los protocolos pueden ser:

- Commit de una fase atómico
- Commit de dos fases atómico
- Commit de tres fases atómico

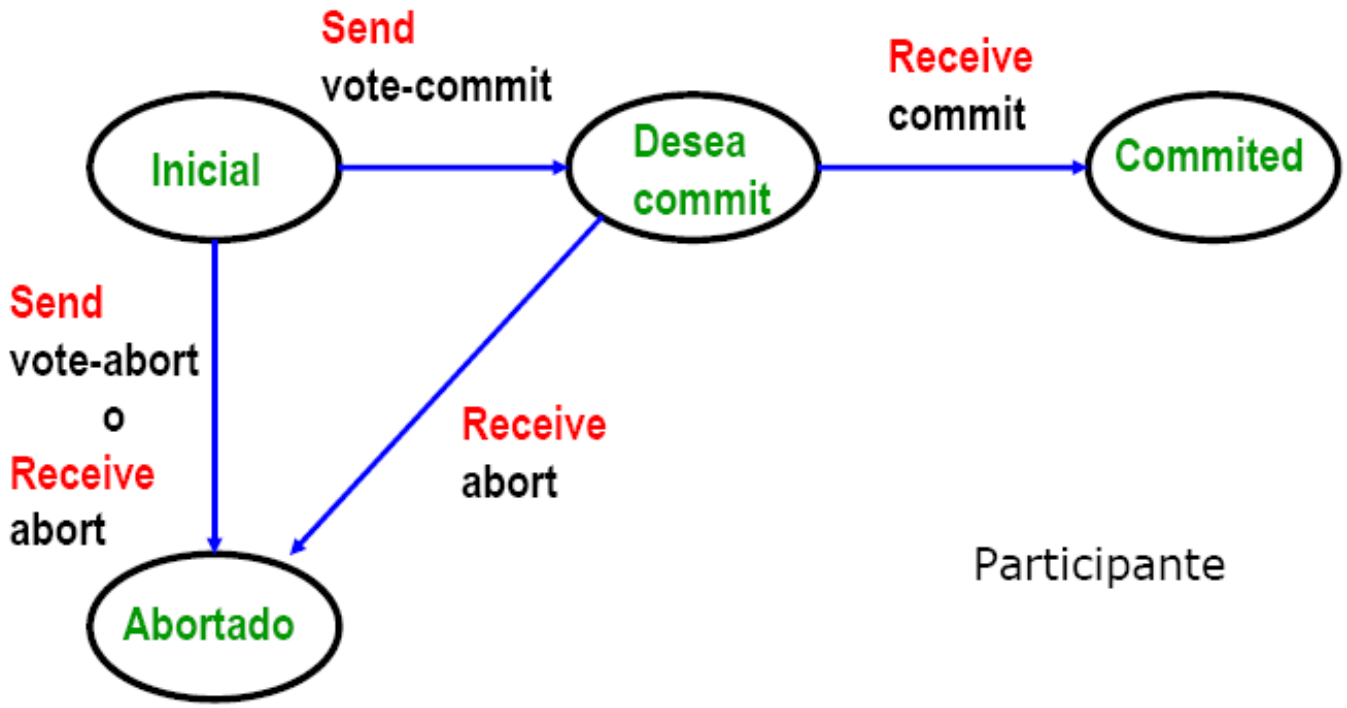
Transacciones Distribuidas

Commit de una fase atómico

- Una manera simple de completar una transacción en forma atómica por el coordinador es comunicar el requerimiento de commit o abort a todos los participantes de la transacción y mantenerse repitiendo el requerimiento hasta que todos ellos tengan conocimiento de que todos lo han llevado a cabo.

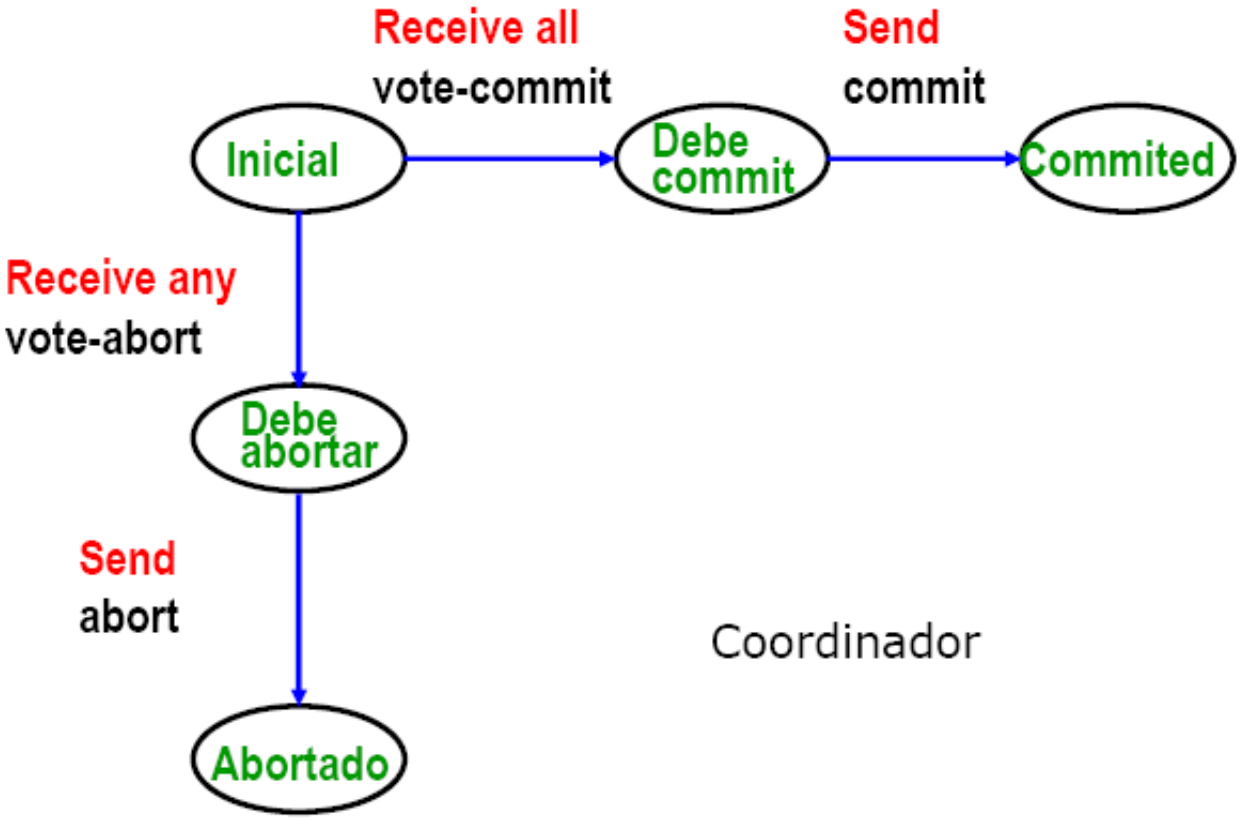
Transacciones Distribuidas

Estados de commit de una sola fase (participante)



Transacciones Distribuidas

Estados de commit de una sola fase (coordinador)



Transacciones Distribuidas

Bloqueo de transacciones

- Pueden ocurrir cuando hay fallas en la red.
- La situación se puede dar con una transacción que no ha fallado sin embargo está bloqueada esperando un commit o abort.
- ¿Qué decisiones podría tomar T_i cuando “detecta” (timeout) que está bloqueada?
  - commit sin autorización
  - abortar

Transacciones Distribuidas

Commit de dos fases atómico

- Hay voto seguido de decisión.
- Se reduce la posibilidad de bloqueo.
- Diferencia con el anterior: si la transacción no recibe respuesta (timeout) entra en un estado de recuperación.
- Para evitar el bloqueo la T_i hace un help-me a los demás participantes.

Transacciones Distribuidas

Cuando un participante recibe un help-me:

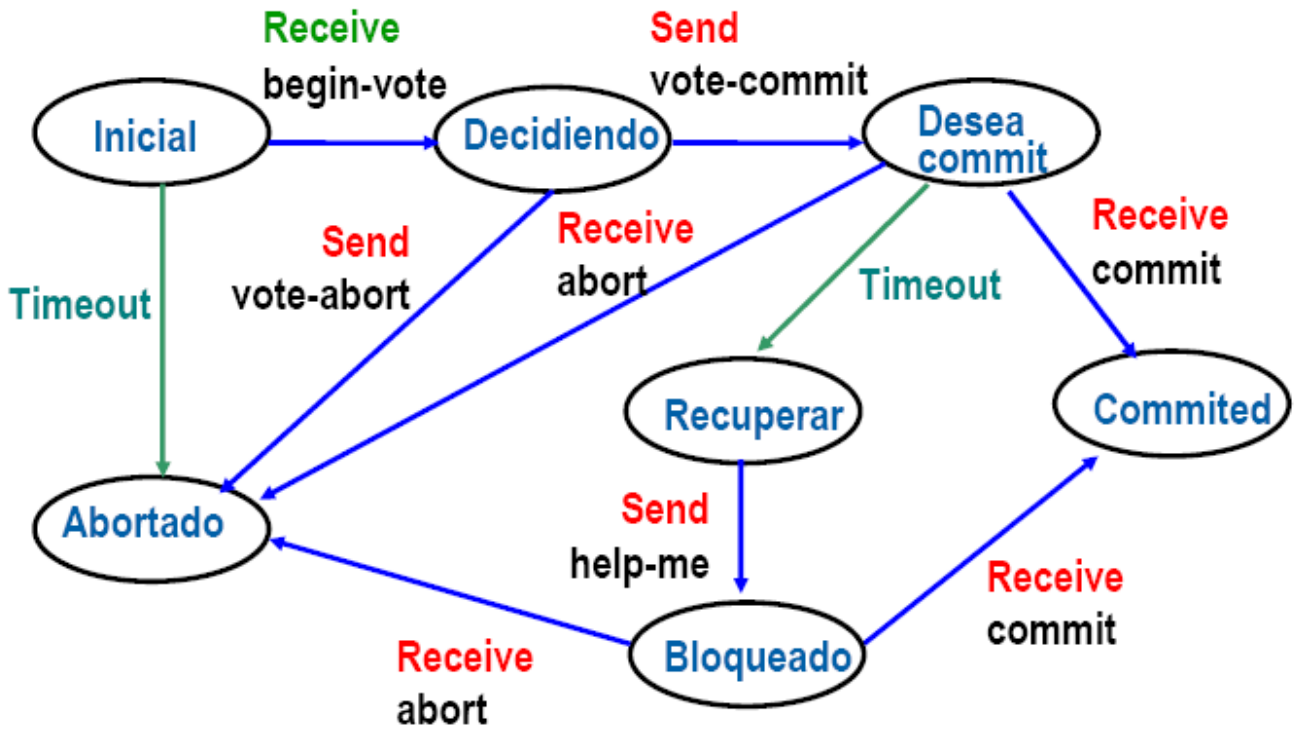
- Un participante en estado **committed** replica **commit**. Lo hace en forma segura porque ya recibió el **commit** del coordinador.
- Un participante en estado **abortado** envía abort porque ya sabe que la transacción va a abortar.
- Si el participante no ha votado aún (en estado **inicial**) puede ayudar a resolver el problema si decide arbitrariamente abortar (envía abort y vote-abort)
- Si el participante está en esperando **commit** no puede resolver el problema, **no responde**.

Transacciones Distribuidas

- Otra modificación es que el coordinador envía un **beginvote** a todos los participantes.
- Si la recuperación del estado **desea commit** es necesaria, cada transacción debe o necesita conocer los restantes participantes.

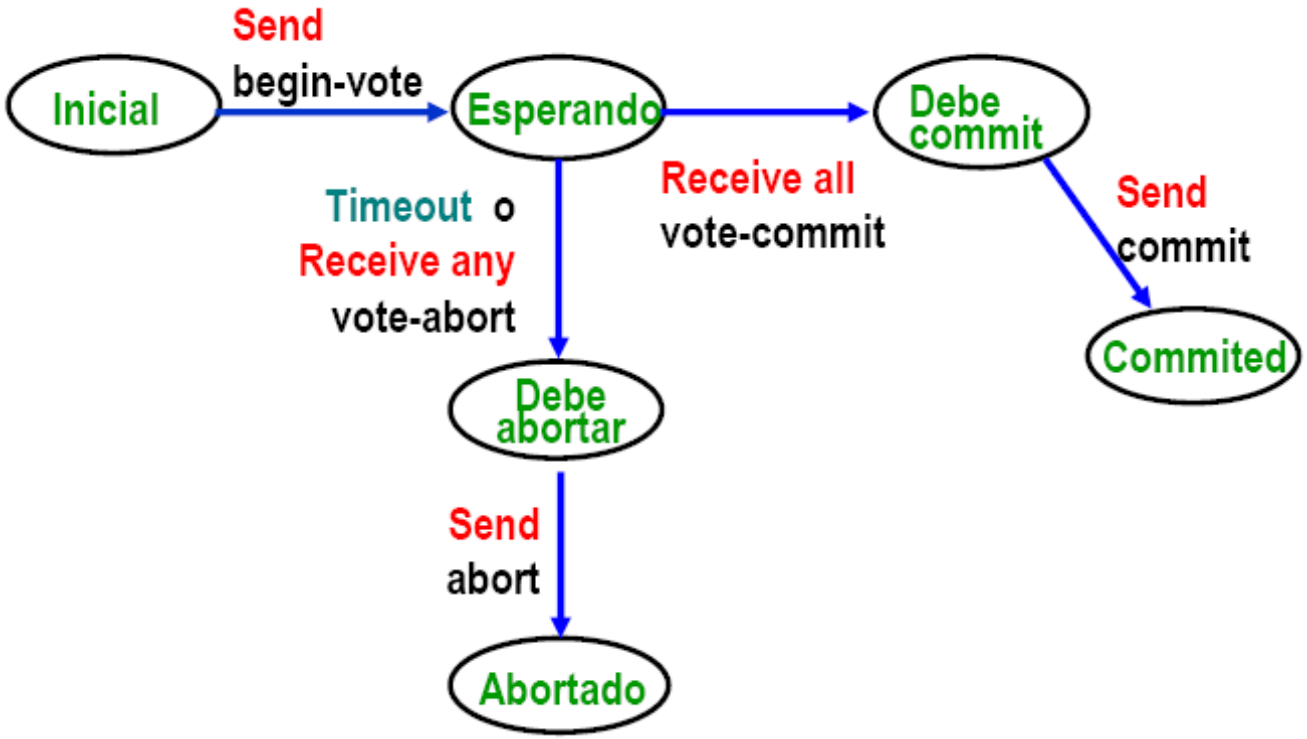
Transacciones Distribuidas

Estados de commit de dos fases (participante)



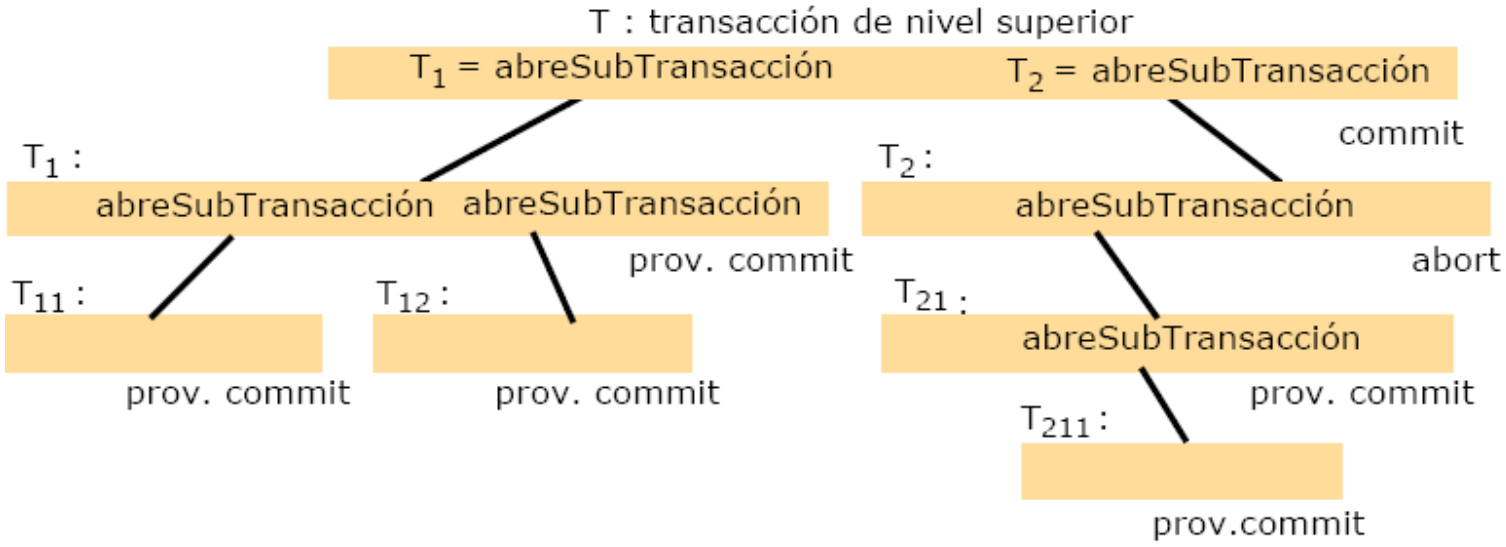
Transacciones Distribuidas

Estados de **commit** de dos fases (coordinador)



Transacciones Distribuidas

Commit para transacciones Anidadas



Transacciones Distribuidas

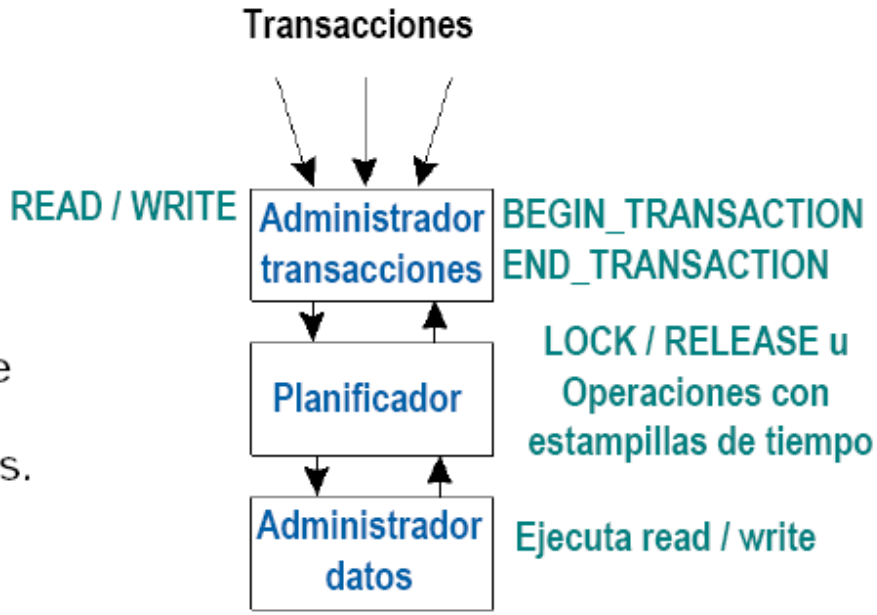
Commit de tres fases atómico

- Un participante no commit hasta que todos los participantes commit y además hasta que no sabe que todos los participantes también lo saben.

Transacciones Distribuidas

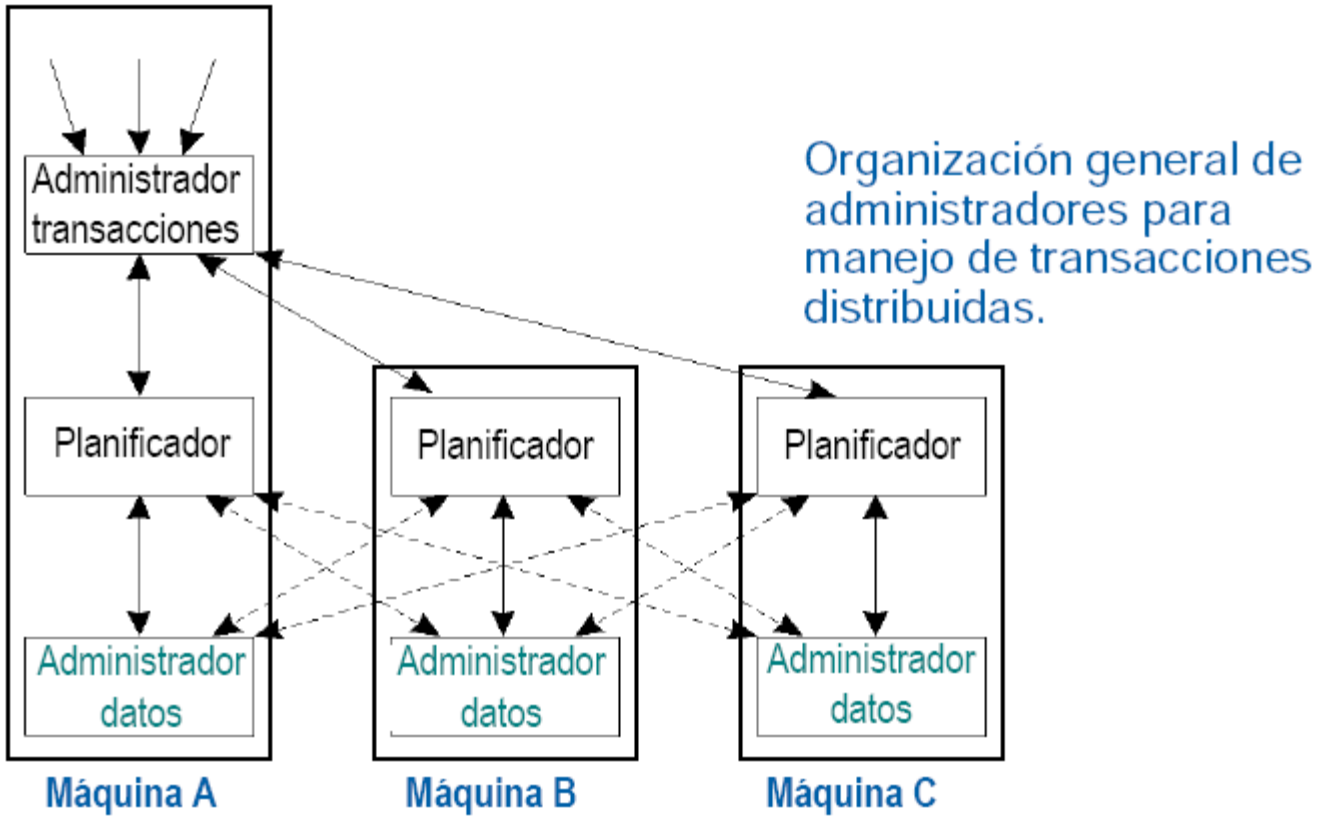
Control de Concurrencia

Organización general de administradores para manejo de transacciones.



Transacciones Distribuidas

Control de Concurrencia



Organización general de administradores para manejo de transacciones distribuidas.

Transacciones Distribuidas

Control de Concurrencia en Transacciones

Distribuidas

- Cada servidor administra un conjunto de objetos y es responsable de mantener la consistencia cuando son accedidos por transacciones concurrentes.
- Cada servidor aplica un control de concurrencia a sus objetos.
- Esto quiere decir que si la transacción T es antes que la U en el acceso conflictivo a un objeto en un servidor entonces deben ser hechas en tal orden en todos los servidores cuyos objetos son accedidos en manera conflictiva por T y U.

Transacciones Distribuidas

Locking

- En una transacción distribuida un lock sobre un objeto es mantenido localmente (en el mismo servidor).
- El administrador local de locks puede decidir si otorga un lock o hace que la transacción que lo requirió espere.
- Sin embargo no puede liberar ningún lock mientras la transacción que los tiene no haya terminado (commit o aborto).
- Cuando es usado el locking para control de concurrencia, los objetos permanecen con el lock y no están disponibles para otras transacciones durante el protocolo de commit atómico.

Transacciones Distribuidas

Locking

- Dado que los administradores de locks en diferentes servidores otorgan locks independiente de los demás, es posible que diferentes servidores impongan diferente orden sobre las transacciones.
- Considerese el siguiente entrelazado de las transacciones T y U en los servidores X e Y:

T	*U*
Write(A) en *X* lock *A*	
	Write(B) en *Y* lock *B*
Read(B) en *Y* espera *U*	
	Read(A) en *X* espera por *T*

Transacciones Distribuidas

- Se tiene T antes que U en un servidor y U antes que T en el otro.
- Estos órdenes diferentes pueden llevar a una dependencia cíclica entre transacciones que deriva en una situación de interbloqueo.
- Cuando se detecta una condición de interbloqueo las transacciones son abortadas.
- En este caso el coordinador será informado y abortará las transacciones en los participantes involucrados.

Transacciones Distribuidas

- Control de Concurencia por Estampillas de Tiempo
- En transacciones distribuidas se requiere que cada coordinador genere una única estampilla de tiempo global.
- Esta estampilla de tiempo es dada al cliente por el primer coordinador accedido por la transacción.
- La estampilla de tiempo de la transacción es pasada al coordinador de cada servidor en los cuales se realizan operaciones de la transacción.
- Los servidores son conjuntamente responsables de que las transacciones distribuidas sean realizadas de manera serial.

Transacciones Distribuidas

- Por ejemplo: si la versión de un objeto accedido por la transacción U llega al commit después de la versión accedida por T en un servidor, entonces si T y U acceden al mismo objeto en otros servidores, deben llegar al commit en el mismo orden.
- Para lograr el mismo orden en todos los servidores, los coordinadores deben estar de acuerdo en el ordenamiento de sus estampillas de tiempo.
- Una estampilla de tiempo consiste de un par:

  <estampilla de tiempo local,identificador del servidor>
- Se necesita sincronización de relojes.

Transacciones Distribuidas

- Cuando el ordenamiento por estampillas de tiempo es usado para el control de concurrencia los conflictos son resueltos cuando cada operación es efectuada.
- Si la resolución del conflicto requiere que una transacción sea abortada, el coordinador será informado y abortará las transacciones de todos los participantes.

Transacciones Distribuidas

- Control de Concurrencia Optimista
- Cada transacción es validada antes de permitir el commit.
- Una transacción distribuida es validada por una colección de servidores independientes, cada uno de los cuales valida las transacciones que acceden a sus objetos propios.
- Sea las transacciones T y U entrelazadas, las cuales acceden a los objetos A y B en los servidores X e Y respectivamente:

T		U	
Read(A)	en X	*Read(B)*	en Y
Write(A)		*Write(B)*	
Read(B)	en Y	*Read(A)*	en X
Write(B)		*Write(A)*	

Transacciones Distribuidas

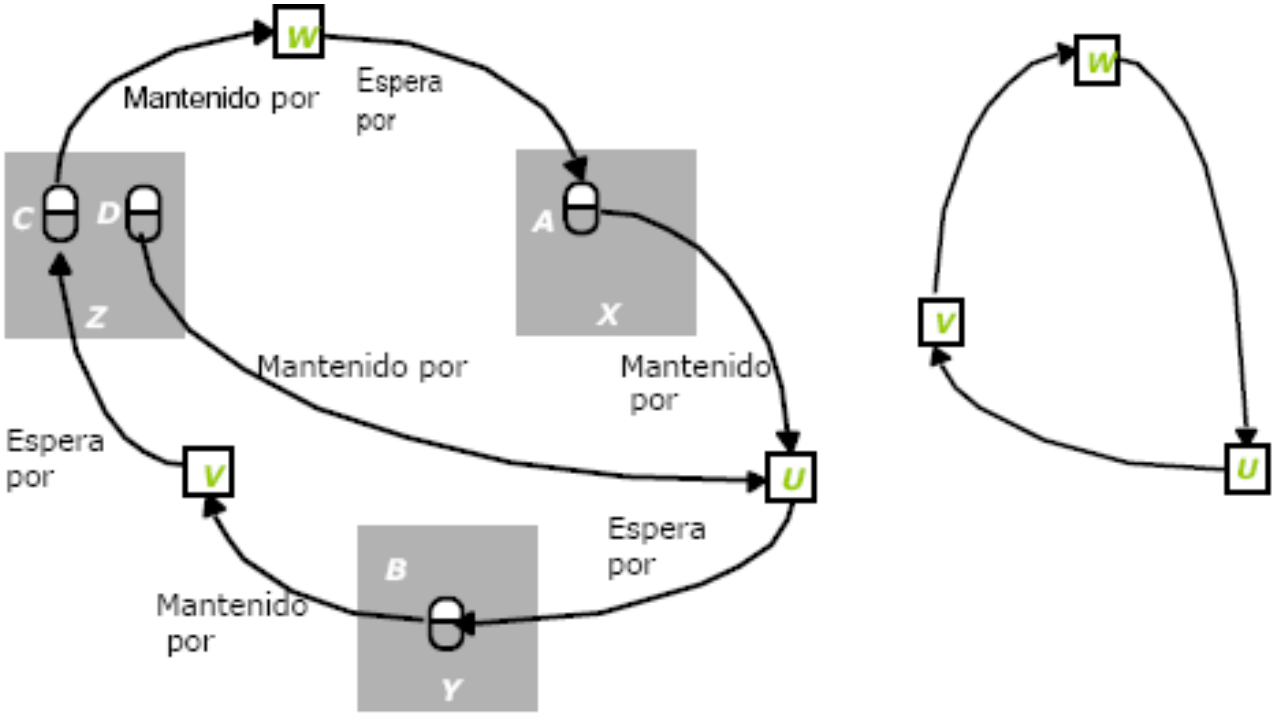
- Las transacciones acceden a los objetos en el orden T antes que U en el servidor X y U antes que T en el servidor Y.
- Si se supone que T y U empiezan la validación al mismo tiempo, el servidor X valida T primero y el servidor Y valida U primero.
- Solo una transacción puede realizar la fase de validación y actualización a la vez.
- En el ejemplo, cada servidor está inhabilitado de validar la otra transacción hasta que la primera haya completado.
- Esto es un interbloqueo de commit.

Transacciones Distribuidas

- Ejemplo de interbloqueo distribuido

U	V	W
d.deposita(10) *lock D*		
	b.deposita(10) *lock B en Y*	
a.deposita(20) *lock A en X*		
		c.deposita(30) *lock C en Z*
b.extrae(30) *espera en Y*		
	c.extrae(20) *espera en Z*	
		a.extrae(20) *espera en X*

Transacciones Distribuidas



- Fuente JRA 2009