

# Modelos y bases de datos

## Recuperación y concurrencia

CEIS

2022-1

# Agenda

## Introducción

Problemas Solución

Transacciones

Corrección

## Concurrencia

Problemas

Solución

## Recuperación

Problemas

Solución

## Aspectos finales

# Agenda

## Introducción

Problemas Solución

Transacciones

Corrección

## Concurrencia

Problemas

Solución

## Recuperación

Problemas

Solución

## Aspectos finales

# Agenda

## Introducción

Problemas Solución

Transacciones

Corrección

## Concurrencia

Problemas

Solución

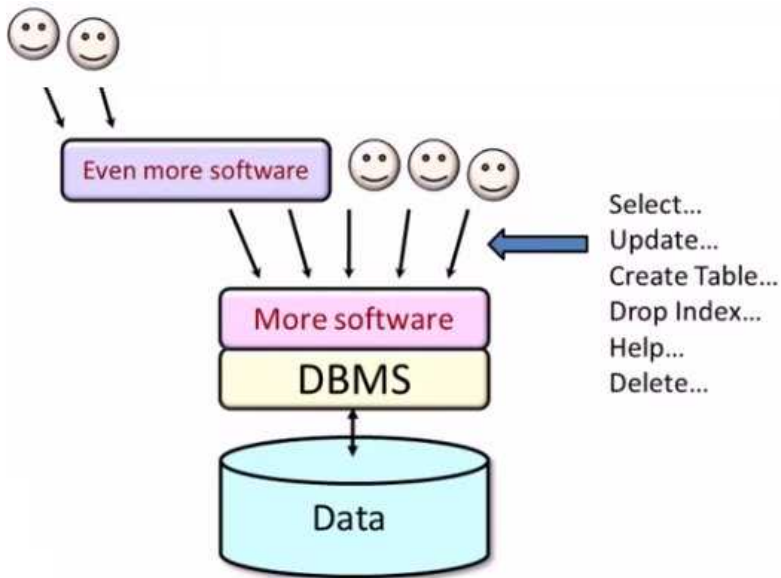
## Recuperación

Problemas

Solución

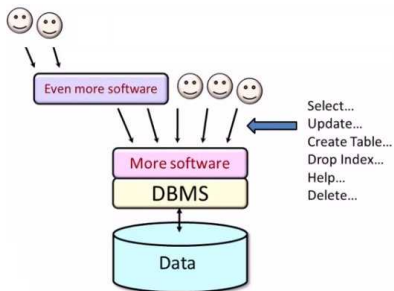
## Aspectos finales

# Problemas-Solución



# Problemas-Solución

:( Dos aplicaciones que están ejecutando simultáneamente usan los mismos datos

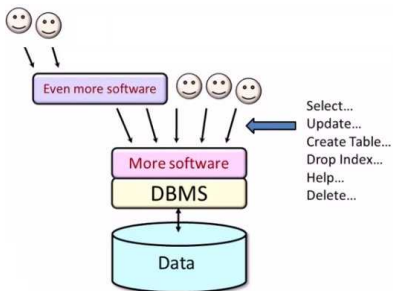


"MOOC Databases. Stanford."

:( El sistema falla cuando se está ejecutando

Retirar dinero en un cajero

# Problemas-Solución



"MOOC Databases. Stanford."

:( Dos aplicaciones que están ejecutando simultáneamente usan los mismos datos

Puede quedar la base de datos con datos incorrectos

## Concurrencia

¿Cómo garantizamos una ejecución concurrente **correcta**?

:( El sistema falla cuando se está ejecutando

Puede quedar la base de datos en un estado desconocido (correcto o incorrecto)

## Recuperación

¿Cómo volvemos a un estado **correcto** y conocido?

## ADMINISTRACION DE TRANSACCIONES

Retirar dinero en un cajero

# Agenda

## Introducción

Problemas Solución

**Transacciones**

Corrección

## Concurrencia

Problemas

Solución

## Recuperación

Problemas

Solución

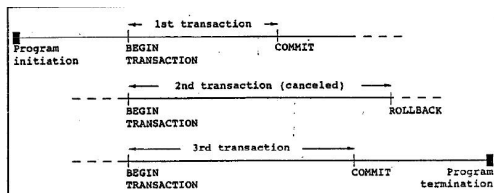
## Aspectos finales



# Transacciones

## Definición

## Contexto



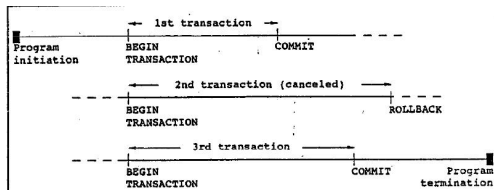
## Propiedades. ACID

# Transacciones

## Definición

Es una secuencia de operaciones que transforman un estado consistente en otro estado consistente. Unidad de trabajo lógica.

## Contexto



La ejecución de un programa es la ejecución de una secuencia de transacciones que pueden ser **concurrentes**

## Propiedades. ACID

- A Atomicidad
- C Consistencia
- I Aislamiento
- D Durabilidad

# Transacciones

## Transferencia

```
CREATE OR REPLACE  
PROCEDURE TRANSFERENCIA(origen IN CHAR, destino IN CHAR, valor IN NUMBER)  
IS  
BEGIN  
    UPDATE CUENTAS SET saldo=saldo+valor WHERE numero=destino;  
    UPDATE CUENTAS SET saldo=saldo-valor WHERE numero=origen;  
END;  
/
```

Hay un chequeo de saldo  $\geq 0$

- ¿Es correcto?

# Transacciones

## Transferencia

```
CREATE OR REPLACE
PROCEDURE TRANSFERENCIA(origen IN CHAR, destino IN CHAR, valor IN NUMBER)
IS
BEGIN
    UPDATE CUENTAS SET saldo=saldo+valor WHERE numero=destino;
    IF (SQL%ROWCOUNT=0) THEN
        RAISE_APPLICATION_ERROR(-20001,'No existe la cuenta destino');
    END IF;
    UPDATE CUENTAS SET saldo=saldo-valor WHERE numero=origen;
    IF (SQL%ROWCOUNT=0) THEN
        RAISE_APPLICATION_ERROR(-20001,'No existe la cuenta origen');
    END IF;
END;
/
```

Hay un chequeo de saldo  $\geq 0$

- ▶ ¿Es correcto?
- ▶ ¿Qué pasa si el sistema se cae después de realizarse la primera instrucción?

# Transacciones

## Transferencia

```
CREATE OR REPLACE
PROCEDURE TRANSFERENCIA(origen IN CHAR, destino IN CHAR, valor IN NUMBER)
IS
BEGIN
    UPDATE CUENTAS SET saldo=saldo+valor WHERE numero=destino; leer,actualizar,escribir
    IF {SQL%ROWCOUNT=0} THEN
        RAISE_APPLICATION_ERROR(-20001,'No existe la cuenta destino');
    END IF;
    UPDATE CUENTAS SET saldo=saldo-valor WHERE numero=origen; leer,actualizar,escribir
    IF {SQL%ROWCOUNT=0} THEN
        RAISE_APPLICATION_ERROR(-20001,'No existe la cuenta origen');
    END IF;
END;
/
```

Hay un chequeo de saldo  $\geq 0$

- ▶ ¿Es correcto?
- ▶ ¿Qué pasa si el sistema se cae después de realizarse la primera instrucción?
- ▶ ¿Qué puede pasar si hay dos transacciones concurrentes?

# Transacciones

## Elementos

### UN INICIO

#### 1. BEGIN TRANSACTION

Indica el inicio de la transacción

### DOS POSIBLES FINALES

#### 2. COMMIT

Indica la finalización de una transacción satisfactoria

Los cambios ahora son permanentes y visibles a todos los usuarios

#### 3. ROLLBACK

Indica la finalización de una transacción no satisfactoria

Los cambios no son válidos, nadie los verá

## Puntos

- ▶ **De sincronización** Las instrucciones COMMIT y ROLLBACK establecen puntos de sincronización; son puntos en que la base de datos debería estar en un estado de consistencia
- ▶ **De verificación** Escritura a disco, a determinados intervalos de tiempos prescritos

# Transacciones

## Transferencia

```
CREATE OR REPLACE
PROCEDURE TRANSFERENCIA(origen IN CHAR, destino IN CHAR, valor IN NUMBER)
IS
BEGIN
    UPDATE CUENTAS SET saldo=saldo+valor WHERE numero=destino;
    IF (SQL%ROWCOUNT=0) THEN
        RAISE_APPLICATION_ERROR(-20001,'No existe la cuenta destino');
    END IF;
    UPDATE CUENTAS SET saldo=saldo-valor WHERE numero=origen;
    IF (SQL%ROWCOUNT=0) THEN
        RAISE_APPLICATION_ERROR(-20001,'No existe la cuenta origen');
    END IF;
END;
/
```

- ▶ ¿BEGIN TRANSACTION?
- ▶ ¿COMMIT?
- ▶ ¿ROLLBACK?

# Transacciones

## Transferencia

```
CREATE OR REPLACE
PROCEDURE TRANSFERENCIA(origen IN CHAR, destino IN CHAR, valor IN NUMBER)
IS
BEGIN
  --BEGIN TRANSACTION
  UPDATE CUENTAS SET saldo=saldo+valor WHERE numero=destino;
  IF (SQL%ROWCOUNT=0) THEN
    RAISE_APPLICATION_ERROR(-20001,'No existe la cuenta destino');
  END IF;
  UPDATE CUENTAS SET saldo=saldo-valor WHERE numero=origen;
  IF (SQL%ROWCOUNT=0) THEN
    RAISE_APPLICATION_ERROR(-20001,'No existe la cuenta origen');
  END IF;
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    ROLLBACK;
    RAISE_APPLICATION_ERROR(-20999,SQLERRM);
END;
/
```



# Agenda

## Introducción

Problemas Solución

Transacciones

Corrección

## Concurrencia

Problemas

Solución

## Recuperación

Problemas

Solución

## Aspectos finales

# Corrección

De un programa

De base de base de datos

De conjunto de transacciones

# Corrección

## De un programa

:) Si cumple la precondition debe cumplir la poscondición

## De base de base de datos

:) Si cumple todas las restricciones de integridad definidas

## De conjunto de transacciones

:) Si es serializable

Produce el mismo resultado que **una** ejecución serial de las mismas transacciones

# Agenda

## Introducción

Problemas Solución

Transacciones

Corrección

## Concurrencia

Problemas

Solución

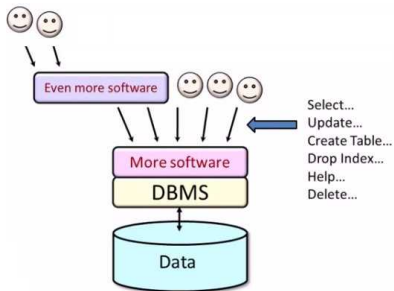
## Recuperación

Problemas

Solución

## Aspectos finales

# Problemas-Solución



"MOOC Databases. Stanford."

:( Dos aplicaciones que están ejecutando simultáneamente usan los mismos datos

Puede quedar la base de datos con datos incorrectos

Concurrencia

¿Cómo garantizamos una ejecución concurrente correcta?

ADMINISTRACION DE  
TRANSACCIONES

# Agenda

## Introducción

Problemas Solución

Transacciones

Corrección

## Concurrencia

Problemas

Solución

## Recuperación

Problemas

Solución

## Aspectos finales

# Problemas

## 1. Actualización perdida

Lectura no repetible

## 2. Dependencia no confirmada

Lectura sucia

## 3. Análisis inconsistente

Lectura no repetible

## 4.

Fantasma

# Problemas

## Actualización perdida

© 2011 - 2008 MIT - 2008 MIT - 2008 MIT - 2008 MIT - 2008 MIT

- ▶ Tenemos una cuenta  $t$  con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

¿Cuáles serían posibles resultados de una ejecución correcta?



# Problemas

## Actualización perdida

Transaction A	Time	Transaction B
-		-

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

Ejecutemos paso a paso

# Problemas

## Actualización perdida

Transaction A	Time	Transaction B
RETRIEVE t	t1	

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

Ejecutemos paso a paso

# Problemas

## Actualización perdida

Transaction A	Time	Transaction B
— RETRIEVE t —	t1   t2	— — RETRIEVE t

- ▶ Tenemos una cuenta *t* con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

Ejecutemos paso a paso

# Problemas

## Actualización perdida

Transaction A	Time	Transaction B
-		-
RETRIEVE t	t1	-
-		-
-	t2	RETRIEVE t
-		-
UPDATE t	t3	-

- ▶ Tenemos una cuenta *t* con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

Ejecutemos paso a paso

# Problemas

## Actualización perdida

Transaction A	Time	Transaction B
-		-
-		-
RETRIEVE t	t1	-
-		-
-	t2	RETRIEVE t
-		-
UPDATE t	t3	-
-		-
-	t4	UPDATE t
-		-

- ▶ Tenemos una cuenta *t* con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

Ejecutemos paso a paso

# Problemas

## Actualización perdida

Transaction A	Time	Transaction B
-		-
-		-
RETRIEVE t	t1	-
-		-
-	t2	RETRIEVE t
-		-
UPDATE t	t3	-
-		-
-	t4	UPDATE t
-		-

- ▶ Tenemos una cuenta t con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

¿Es correcto?

# Problemas

## Actualización perdida

Transaction A	Time	Transaction B
-		-
-		-
RETRIEVE t	t1	-
-		-
-	t2	RETRIEVE t
-		-
UPDATE t	t3	-
-		-
-	t4	UPDATE t
-		-

- ▶ Tenemos una cuenta t con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

¿Es correcto? ¡No es correcto! Lectura no repetible

# Agenda

## Introducción

Problemas Solución

Transacciones

Corrección

## Concurrencia

Problemas

Solución

## Recuperación

Problemas

Solución

## Aspectos finales



# Bloqueo

## Idea

Cuando una transacción quiere asegurar que el objeto en el que está interesada (normalmente una tupla) no cambiará mientras la está usando adquiere un **bloqueo** sobre ese objeto.

## Tipos de bloqueo

- ▶ Exclusivo (X)
- ▶ Compartido (S)

# Bloqueo

## Idea

Cuando una transacción quiere asegurar que el objeto en el que está interesada (normalmente una tupla) no cambiará mientras la está usando adquiere un **bloqueo** sobre ese objeto.

## Tipos de bloqueo

### ► Exclusivo (X)

Si la transacción A pone un bloqueo exclusivo X sobre la tupla t,

- se rechazará la petición de cualquier otra transacción B para un bloqueo de cualquier tipo sobre t

### ► Compartido (S)

Si la transacción A pone un bloqueo compartido S sobre la tupla t,

- se rechazará la petición de cualquier otra transacción B para un bloqueo X sobre t

- se otorgará una petición de cualquier otra transacción para un bloqueo S sobre t

# Bloqueo

## Idea

Cuando una transacción quiere asegurar que objeto en el que está interesada (normalmente una tupla) no cambiará mientras la está usando adquiere un **bloqueo** sobre ese objeto.

## Tipos de bloqueo

- ▶ Exclusivo (X)
- ▶ Compartido (S)

## Completar tabla

	X	S	-	Tiene
X				
S				
Solicita				

# Bloqueo

## Idea

Cuando una transacción quiere asegurar que objeto en el que está interesada no cambiará mientras la está usando adquiere un **bloqueo** sobre ese objeto.

## Granularidad del bloqueo

- ▶ Tupla

Normalmente es una tupla

- ▶ Tabla

- ▶ Base de datos

# Transacciones

Propiedades. ACID. [I] Aislamiento. Niveles.

1. READ UNCOMMITTED : Lectura no registrada
2. READ COMMITTED : Lectura registrada
3. REPEATABLE READ : Lectura repetible
4. SERIALIZABLE : Serializable

# Transacciones

Propiedades. ACID. [I] Aislamiento. Niveles.

1. READ UNCOMMITTED : Lectura no registrada  
sin bloqueos
2. READ COMMITTED : Lectura registrada  
Bloqueo exclusivo para actualizar
3. REPEATABLE READ : Lectura repetible  
Bloqueo exclusivo para actualizar y compartido para leer
4. SERIALIZABLE : Serializable  
Bloqueo exclusivo para actualizar y compartido para leer  
Bloqueos de rango en las consultas (no cambiar WHERE)

# Bloqueo

## Protocolo: Lectura repetible

Este es el protocolo normal de acceso a los datos

## Operaciones - bloqueo

### ► Para leer

Una transacción que desea recuperar una tupla primero debe adquirir un bloqueo compartido S sobre la tupla

(S) Compartido = De lectura

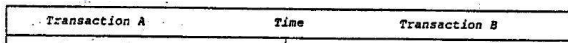
### ► Para escribir

Una transacción que desea actualizar una tupla primero debe adquirir un bloqueo exclusivo X sobre la tupla

(X) Exclusivo = De escritura

# Con bloqueos: Lectura repetible

## Actualización perdida



- ▶ Tenemos una cuenta  $t$  con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

¿Cuáles serían posibles resultados de una ejecución correcta?



# Con bloqueos: Lectura repetible

## Actualización perdida

Transaction A	Time	Transaction B
RETRIEVE t (acquire S lock on t)	t1	

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

Ejecutemos paso a paso

# Con bloqueos: Lectura repetible

## Actualización perdida

Transaction A	Time	Transaction B
-		-
RETRIEVE t	t1	-
(acquire S lock on t)		-
-		-
-	t2	RETRIEVE t
-		(acquire S lock on t)
-		-

- ▶ Tenemos una cuenta *t* con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

Ejecutemos paso a paso

# Con bloqueos: Lectura repetible

## Actualización perdida

Transaction A	Time	Transaction B
-		-
RETRIEVE t	t1	-
(acquire S lock on t)		-
-		-
-	t2	RETRIEVE t
-		(acquire S lock on t)
UPDATE t	t3	-
(request X lock on t)		-
wait		-

- ▶ Tenemos una cuenta *t* con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

Ejecutemos paso a paso

# Con bloqueos: Lectura repetible

## Actualización perdida

Transaction A	Time	Transaction B
-		-
RETRIEVE t	t1	-
(acquire S lock on t)		-
-		-
-	t2	RETRIEVE t
-		(acquire S lock on t)
UPDATE t	t3	-
(request X lock on t)		-
wait		-
wait	t4	UPDATE t
wait		(request X lock on t)
wait		wait
wait		wait
wait		wait

- Tenemos una cuenta *t* con saldo 1.000.000
- La transacción A es una consignación de 250.000
- La transacción B es un retiro de 500.000

Ejecutemos paso a paso

# Con bloqueos: Lectura repetible

## Actualización perdida

Transaction A	Time	Transaction B
-		-
RETRIEVE t	t1	-
(acquire S lock on t)		-
-		-
-	t2	RETRIEVE t
-		(acquire S lock on t)
UPDATE t	t3	-
(request X lock on t)		-
wait		-
wait	t4	UPDATE t
wait		(request X lock on t)
wait		wait
wait		wait
wait		wait

- Tenemos una cuenta *t* con saldo 1.000.000
- La transacción A es una consignación de 250.000
- La transacción B es un retiro de 500.000

Ejecutemos paso a paso

# Con bloqueos: Lectura repetible

## Actualización perdida

Transaction A	Time	Transaction B
-		-
RETRIEVE t	t1	-
(acquire S lock on t)		-
-		-
-	t2	RETRIEVE t
-		(acquire S lock on t)
UPDATE t	t3	-
(request X lock on t)		-
wait		-
wait	t4	UPDATE t
wait		(request X lock on t)
wait		wait
wait		wait
wait		wait

- Tenemos una cuenta *t* con saldo 1.000.000
- La transacción A es una consignación de 250.000
- La transacción B es un retiro de 500.000

¿Es correcto?

# Con bloqueos: Lectura repetible

## Actualización perdida

Transaction A	Time	Transaction B
-		-
RETRIEVE t	t1	-
(acquire S lock on t)		-
-		-
-	t2	RETRIEVE t
-		(acquire S lock on t)
UPDATE t	t3	-
(request X lock on t)		-
wait		-
wait	t4	UPDATE t
wait		(request X lock on t)
wait		wait
wait		wait
wait		wait

- Tenemos una cuenta *t* con saldo 1.000.000
- La transacción A es una consignación de 250.000
- La transacción B es un retiro de 500.000

¿Es correcto? **Bloqueo mortal**

# Bloqueo mortal

¿Cómo se soluciona?

¿Cómo se detecta?



# Bloqueo mortal

## ¿Cómo se soluciona?

Seleccionar una de las transacciones del bloqueo, **la víctima** y deshacerla, liberando sus bloqueos,

## ¿Cómo se detecta?

### 1. Mantener grafo de espera

¿Qué información guardamos?

¿Qué buscamos?

### 2. Control de tiempo

¿Qué información guardamos?

¿Qué buscamos?

# Agenda

## Introducción

Problemas Solución

Transacciones

Corrección

## Concurrencia

Problemas

Solución

## Recuperación

Problemas

Solución

## Aspectos finales

# Agenda

## Introducción

Problemas Solución

Transacciones

Corrección

## Concurrencia

Problemas

Solución

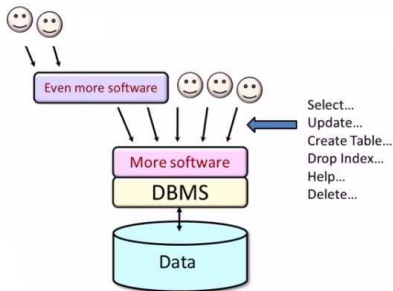
## Recuperación

Problemas

Solución

## Aspectos finales

# Problemas-Solución



"MOOC Databases. Stanford."

:( Dos aplicaciones que están ejecutando simultáneamente usan los mismos datos

Puede quedar la base de datos con datos incorrectos

## Concurrencia

¿Cómo garantizamos una ejecución concurrente **correcta**?

:( El sistema falla cuando se está ejecutando

Puede quedar la base de datos en un estado desconocido (correcto o incorrecto)

## Recuperación

¿Cómo volvemos a un estado **correcto** y conocido?

## ADMINISTRACION DE TRANSACCIONES

# Problemas

Fallas locales

Fallas globales

# Problemas

## Fallas locales

- ▶ Falla de operación

## Fallas globales

- ▶ Falla del sistema
- ▶ Falla del medio

# Agenda

## Introducción

Problemas Solución

Transacciones

Corrección

## Concurrencia

Problemas

Solución

## Recuperación

Problemas

Solución

## Aspectos finales

# Solución

## Redundancia

Garantizar que cualquier parte de la información que contiene el sistema puede ser recuperada a partir de otra información guardada redundantemente en otro lugar del sistema

- ▶ Bitacora

Mantiene registro de los cambios que cada transacción ha hecho a la base de datos

- ▶ Copias de respaldo

Archivar la base de datos a un segundo disco



# Solución

## Punto de sincronización

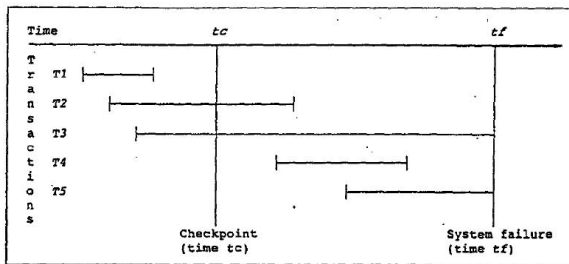
- ▶ Las instrucciones COMMIT y ROLLBACK establecen puntos de sincronización; son puntos en que la base de datos debería estar en un estado de consistencia

## Puntos de verificación

- ▶ Escritura a disco, a determinados intervalos de tiempos prescritos

# Solución

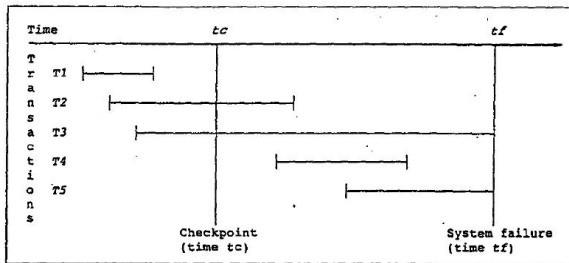
¿Cuáles son las acciones?



- ¿Qué queda listo?
- ¿Qué se debe rehacer?
- ¿Qué se debe deshacer?

# Solución

## ¿Cuáles son las acciones?



### ► ¿Qué queda listo?

Todo el trabajo realizado por las transacciones terminaron antes del punto de verificación

### ► ¿Qué se debe rehacer?

Todo el trabajo realizado por las transacciones que terminaron satisfactoriamente antes de la caída

Recuperación hacia adelante

### ► ¿Qué se debe deshacer?

Todo el trabajo realizado por las transacciones que no terminaron antes de la caída

Recuperación hacia atrás

# Agenda

## Introducción

Problemas Solución

Transacciones

Corrección

## Concurrencia

Problemas

Solución

## Recuperación

Problemas

Solución

## Aspectos finales

# Problemas

## Example

Actualización perdida

Transaction A	Time	Transaction B
-		-

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

# Problemas

## Example

Actualización perdida

Transaction A	Time	Transaction B
RETRIEVE t	t1	

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

# Problemas

## Example

Actualización perdida

Transaction A	Time	Transaction B
— — RETRIEVE t — —	t1   t2	— — — — RETRIEVE t

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

# Problemas

## Example

Actualización perdida

Transaction A	Time	Transaction B
-		-
RETRIEVE t	t1	-
-	t2	RETRIEVE t
-	t3	-
UPDATE t		-

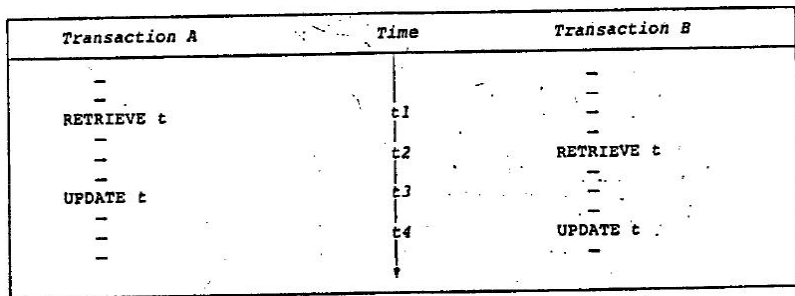
- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000



# Problemas

## Example

### Actualización perdida

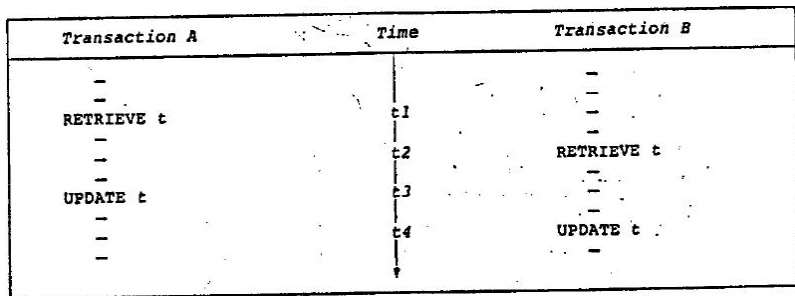


- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

# Problemas

## Example

### Actualización perdida



- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

¡No es correcto! Lectura no repetible

# ¿Solución?

## Actualización perdida

<i>Transaction A</i>	<i>Time</i>	<i>Transaction B</i>

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

# ¿Solución?

## Actualización perdida

Transaction A	Time	Transaction B
-	-	-
-	-	-
RETRIEVE t	t1	-
(acquire S lock on t)		-

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

# ¿Solución?

## Actualización perdida

Transaction A	Time	Transaction B
-	-	-
RETRIEVE t	t1	-
(acquire S lock on t)	-	-
-	-	-
-	t2	RETRIEVE t
-	-	(acquire S lock on t)
-	-	-

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

# ¿Solución?

## Actualización perdida

Transaction A	Time	Transaction B
-		-
RETRIEVE t	t1	-
(acquire S lock on t)		-
-		-
-	t2	RETRIEVE t
-		(acquire S lock on t)
UPDATE t	t3	-
(request X lock on t)		-
wait		-

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

# ¿Solución?

## Actualización perdida

Transaction A	Time	Transaction B
-		-
RETRIEVE t	t1	-
(acquire S lock on t)		-
-		-
-	t2	RETRIEVE t
-		(acquire S lock on t)
UPDATE t	t3	-
(request X lock on t)		-
wait	t4	UPDATE t
wait		(request X lock on t)
wait		wait
wait		wait
wait		wait
wait		wait

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

# ¿Solución?

## Actualización perdida

Transaction A	Time	Transaction B
-		-
RETRIEVE t	t1	-
(acquire S lock on t)		-
-		-
-	t2	RETRIEVE t
-		(acquire S lock on t)
UPDATE t	t3	-
(request X lock on t)		-
wait	t4	UPDATE t
wait		(request X lock on t)
wait		wait
wait		wait
wait		wait
wait		wait

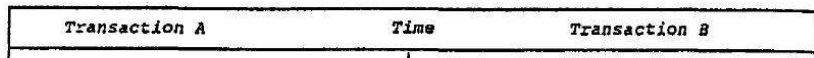
- ▶ Tenemos una cuenta *t* con saldo 1.000.000
- ▶ La transacción A es una consignación de 250.000
- ▶ La transacción B es un retiro de 500.000

Bloqueo mortal



# Problemas

## Dependencia no confirmada



- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consulta el saldo
- ▶ La transacción B trata de hacer un retiro de 500.000 pero se arrepiente

# Problemas

## Dependencia no confirmada

Transaction A	Time	Transaction B
-		-
-		-
-	t1	UPDATE t

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consulta el saldo
- ▶ La transacción B trata de hacer un retiro de 500.000 pero se arrepiente

# Problemas

## Dependencia no confirmada

Transaction A	Time	Transaction B
-	-	-
-	-	-
-	t1	UPDATE t
-	-	-
RETRIEVE t	t2	-

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consulta el saldo
- ▶ La transacción B trata de hacer un retiro de 500.000 pero se arrepiente

# Problemas

## Dependencia no confirmada

Transaction A	Time	Transaction B
-		-
-		-
-	t1	UPDATE t
-		-
RETRIEVE t	t2	-
-		-
-	t3	ROLLBACK
-		

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consulta el saldo
- ▶ La transacción B trata de hacer un retiro de 500.000 pero se arrepiente

# Problemas

## Dependencia no confirmada

Transaction A	Time	Transaction B
-		-
-		-
-		-
-	t1	UPDATE t
RETRIEVE t	t2	-
-	t3	-
-		ROLLBACK
-		

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consulta el saldo
- ▶ La transacción B trata de hacer un retiro de 500.000 pero se arrepiente

¡ No es correcto!. Lectura sucia

# ¿Solución?

## Dependencia no confirmada

Transaction A	Time	Transaction B

- ▶ Tenemos una cuenta **t** con saldo 1.000.000
- ▶ La transacción A es una consulta de saldo
- ▶ La transacción B es un retiro de 500.000

# ¿Solución?

## Dependencia no confirmada

Transaction A	Time	Transaction B
-		-
-		-
-	t1	UPDATE t (acquire X lock on t)

- Tenemos una cuenta **t** con saldo 1.000.000
- La transacción A es una consulta de saldo
- La transacción B es un retiro de 500.000

# ¿Solución?

## Dependencia no confirmada

Transaction A	Time	Transaction B
-	-	-
-	-	-
-	-	-
-	-	-
RETRIEVE t (request S lock on t)	c2	UPDATE t (acquire X lock on t)
-	-	-

- Tenemos una cuenta **t** con saldo 1.000.000
- La transacción A es una consulta de saldo
- La transacción B es un retiro de 500.000



# ¿Solución?

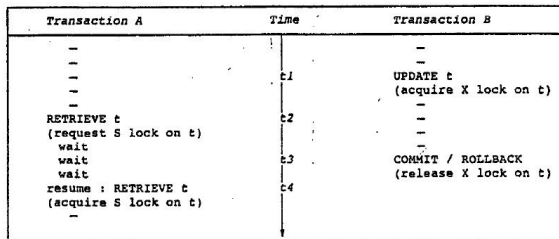
## Dependencia no confirmada

Transaction A	Time	Transaction B
-		-
-		-
-	t1	UPDATE t (acquire X lock on t)
-		-
RETRIEVE t (request S lock on t)	t2	-
wait		-
wait	t3	COMMIT / ROLLBACK (release X lock on t)
wait		

- Tenemos una cuenta **t** con saldo 1.000.000
- La transacción A es una consulta de saldo
- La transacción B es un retiro de 500.000

# ¿Solución?

## Dependencia no confirmada



- Tenemos una cuenta **t** con saldo 1.000.000
- La transacción A es una consulta de saldo
- La transacción B es un retiro de 500.000

# ¿Solución?

## Dependencia no confirmada

Transaction A	Time	Transaction B
-		-
-		-
-	t1	UPDATE t (acquire X lock on t)
-		-
RETRIEVE t (request S lock on t)	t2	-
wait		-
wait	t3	COMMIT / ROLLBACK (release X lock on t)
wait		
resume : RETRIEVE t (acquire S lock on t)	t4	
-		

- Tenemos una cuenta **t** con saldo 1.000.000
- La transacción A es una consulta de saldo
- La transacción B es un retiro de 500.000

¡Es correcto!

# Problemas

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# Problemas

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-		-
-		-
RETRIEVE ACC 1 :	tl	-
sum = 40		-

- La transacción A está calculando el saldo de las cuentas
- La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# Problemas

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-		-
RETRIEVE ACC 1 :	t1	-
sum = 40		-
-		-
RETRIEVE ACC 2 :	t2	-
sum = 90		-

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# Problemas

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-		-
RETRIEVE ACC 1 :	c1	-
sum = 40		-
-		-
RETRIEVE ACC 2 :	c2	-
sum = 90		-
-		-
-	c3	RETRIEVE ACC 3

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# Problemas

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
RETRIEVE ACC 1 : sum = 40	t1	
RETRIEVE ACC 2 : sum = 90	t2	
	t3	RETRIEVE ACC 3
	t4	UPDATE ACC 3 : 30 → 20

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1



# Problemas

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-		-
RETRIEVE ACC 1 :	t1	-
sum = 40		-
-		-
RETRIEVE ACC 2 :	t2	-
sum = 90		-
-		-
-	t3	RETRIEVE ACC 3
-		-
-	t4	UPDATE ACC 3 :
-		30 → 20
-		-
-	t5	RETRIEVE ACC 1

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# Problemas

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-		-
RETRIEVE ACC 1 :	c1	-
sum = 40		-
-		-
RETRIEVE ACC 2 :	c2	-
sum = 90		-
-	c3	RETRIEVE ACC 3
-	c4	-
-		UPDATE ACC 3 :
-		30 → 20
-	c5	-
-	c6	RETRIEVE ACC 1
-		-
-		UPDATE ACC 1 :
-		40 → 50

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# Problemas

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-	-	-
RETRIEVE ACC 1 :	t1	-
sum = 40	-	-
-	-	-
RETRIEVE ACC 2 :	t2	-
sum = 90	-	-
-	-	-
-	t3	RETRIEVE ACC 3
-	-	-
-	t4	UPDATE ACC 3 :
-	-	30 → 20
-	-	-
-	t5	RETRIEVE ACC 1
-	-	-
-	t6	UPDATE ACC 1 :
-	-	40 → 50
-	-	-
-	t7	COMMIT

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# Problemas

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-	-	-
RETRIEVE ACC 1 :	t1	-
sum = 40	-	-
-	-	-
RETRIEVE ACC 2 :	t2	-
sum = 90	-	-
-	-	-
-	t3	RETRIEVE ACC 3
-	-	-
-	t4	UPDATE ACC 3 :
-	-	30 → 20
-	-	-
-	t5	RETRIEVE ACC 1
-	-	-
-	t6	UPDATE ACC 1 :
-	-	40 → 50
-	-	-
-	t7	COMMIT
-	-	-
RETRIEVE ACC 3 :	t8	-
sum = 110, not 120	-	-
-	-	-

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

¡No es correcto! Lectura no repetible

# ¿Solución?

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# ¿Solución?

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-	-	-
RETRIEVE ACC 1 : (acquire S lock on ACC 1) sum = 40	t1	-
		-
		-

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# ¿Solución?

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-	-	-
RETRIEVE ACC 1 :	t1	-
(acquire S lock on ACC 1)	-	-
sum = 40	-	-
-	-	-
RETRIEVE ACC 2 :	t2	-
(acquire S lock on ACC 2)	-	-
sum = 90	-	-

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# ¿Solución?

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-		-
-		-
RETRIEVE ACC 1 :	t1	-
(acquire S lock on ACC 1)		-
sum = 40		-
-		-
RETRIEVE ACC 2 :	t2	-
(acquire S lock on ACC 2)		-
sum = 90		-
-		-
-	t3	RETRIEVE ACC 3
		(acquire S lock on ACC 3)

- La transacción A está calculando el saldo de las cuentas
- La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1



# ¿Solución?

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-	-	-
RETRIEVE ACC 1 :	t1	-
(acquire S lock on ACC 1)	-	-
sum = 40	-	-
-	-	-
RETRIEVE ACC 2 :	t2	-
(acquire S lock on ACC 2)	-	-
sum = 90	-	-
-	-	-
-	t3	RETRIEVE ACC 3
-	-	(acquire S lock on ACC 3)
-	-	-
-	t4	UPDATE ACC 3
-	-	(acquire X lock on ACC 3)
-	-	30 → 20

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# ¿Solución?

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-	-	-
RETRIEVE ACC 1 :	t1	-
(acquire S lock on ACC 1)	-	-
sum = 40	-	-
-	-	-
RETRIEVE ACC 2 :	t2	-
(acquire S lock on ACC 2)	-	-
sum = 90	-	-
-	-	-
-	t3	RETRIEVE ACC 3
-	-	(acquire S lock on ACC 3)
-	-	-
-	t4	UPDATE ACC 3
-	-	(acquire X lock on ACC 3)
-	-	30 → 20
-	-	-
-	t5	RETRIEVE ACC 1
-	-	(acquire S lock on ACC 1)

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# ¿Solución?

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-	-	-
RETRIEVE ACC 1 :	t1	-
(acquire S lock on ACC 1)	-	-
sum = 40	-	-
-	-	-
RETRIEVE ACC 2 :	t2	-
(acquire S lock on ACC 2)	-	-
sum = 90	-	-
-	-	-
-	t3	RETRIEVE ACC 3
-	-	(acquire S lock on ACC 3)
-	-	-
-	t4	UPDATE ACC 3
-	-	(acquire X lock on ACC 3)
-	-	30 → 20
-	-	-
-	t5	RETRIEVE ACC 1
-	-	(acquire S lock on ACC 1)
-	-	-
-	t6	UPDATE ACC 1
-	-	(request X lock on ACC 3)
-	-	wait

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# ¿Solución?

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-	-	-
RETRIEVE ACC 1 :	t1	-
(acquire S lock on ACC 1)	-	-
sum = 40	-	-
-	-	-
RETRIEVE ACC 2 :	t2	-
(acquire S lock on ACC 2)	-	-
sum = 90	-	-
-	-	-
-	t3	RETRIEVE ACC 3
-	-	(acquire S lock on ACC 3)
-	-	-
-	t4	UPDATE ACC 3
-	-	(acquire X lock on ACC 3)
-	-	30 → 20
-	-	-
-	t5	RETRIEVE ACC 1
-	-	(acquire S lock on ACC 1)
-	-	-
-	t6	UPDATE ACC 1
-	-	(request X lock on ACC 3)
-	-	wait
RETRIEVE ACC 3 :	t7	wait
(request S lock on ACC 3)	-	wait
wait	-	wait
wait	-	wait

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

# ¿Solución?

## Análisis inconsistente

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-	-	-
RETRIEVE ACC 1 :	t1	-
(acquire S lock on ACC 1)	-	-
sum = 40	-	-
-	-	-
RETRIEVE ACC 2 :	t2	-
(acquire S lock on ACC 2)	-	-
sum = 90	-	-
-	-	-
-	t3	RETRIEVE ACC 3
-	-	(acquire S lock on ACC 3)
-	-	-
-	t4	UPDATE ACC 3
-	-	(acquire X lock on ACC 3)
-	-	30 → 20
-	-	-
-	t5	RETRIEVE ACC 1
-	-	(acquire S lock on ACC 1)
-	-	-
-	t6	UPDATE ACC 1
-	-	(request X lock on ACC 3)
-	-	wait
RETRIEVE ACC 3 :	t7	wait
(request S lock on ACC 3)	-	wait
wait	-	wait
wait	-	wait

- ▶ La transacción A está calculando el saldo de las cuentas
- ▶ La transacción B está transfiriendo 10 de la cuenta ACC3 a la cuenta ACC1

**BLOQUEO MORTAL**

# Problemas

¿?

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-		-

- ▶ La transacción A está calculando primero la suma y luego el promedio de las cuentas
- ▶ La transacción B está adicionando una nueva cuenta ACC4 100

# Problemas

¿?

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
RETRIEVE ACC 1 :	cl	
RETRIEVE ACC 2 :		
RETRIEVE ACC 3 :		
sum		

- ▶ La transacción A está calculando primero la suma y luego el promedio de las cuentas
- ▶ La transacción B está adicionando una nueva cuenta ACC4 100

# Problemas

¿?

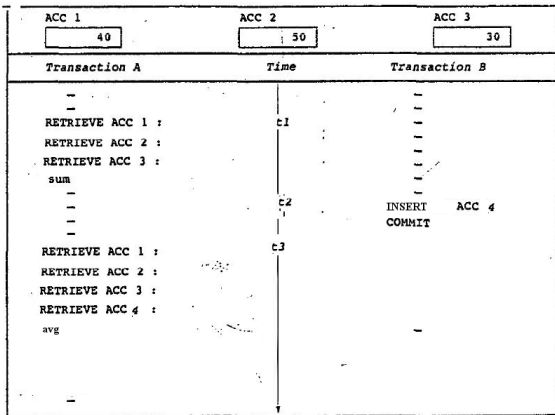
ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
RETRIEVE ACC 1 :	t1	
RETRIEVE ACC 2 :		
RETRIEVE ACC 3 :		
sum		
	t2	
		INSERT ACC 4
		COMMIT

- ▶ La transacción A está calculando primero la suma y luego el promedio de las cuentas
- ▶ La transacción B está adicionando una nueva cuenta ACC4 100



# Problemas

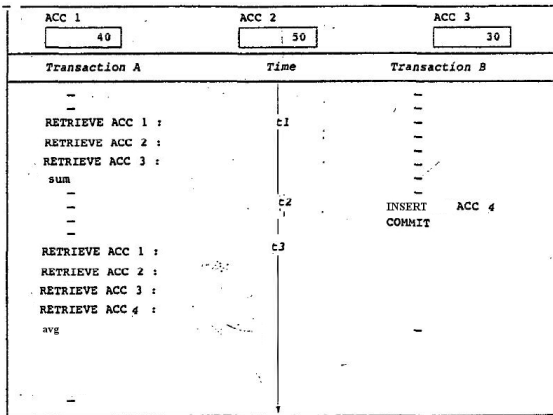
¿?



- ▶ La transacción A está calculando primero la suma y luego el promedio de las cuentas
- ▶ La transacción B está adicionando una nueva cuenta ACC4 100

# Problemas

¿?



- ▶ La transacción A está calculando primero la suma y luego el promedio de las cuentas
  - ▶ La transacción B está adicionando una nueva cuenta ACC4 100
- ¡No es correcto! Fantasmas.

# ¿Solución?

¿?

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
-	-	-

- ▶ La transacción A está calculando primero la suma y luego el promedio de las cuentas
- ▶ La transacción B está adicionando una nueva cuenta ACC4 100

¿?

- ◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ≡ ↺ 🔍 ↻

# ¿Solución?

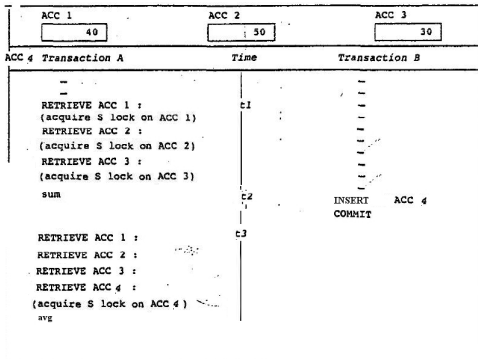
¿?

ACC 1	ACC 2	ACC 3
40	50	30
Transaction A	Time	Transaction B
— — RETRIEVE ACC 1 : (acquire S lock on ACC 1) RETRIEVE ACC 2 : (acquire S lock on ACC 2) RETRIEVE ACC 3 : (acquire S lock on ACC 3) sum	 c1	— — — — — — — —

- La transacción A está calculando primero la suma y luego el promedio de las cuentas
- La transacción B está adicionando una nueva cuenta ACC4 100

# ¿Solución?

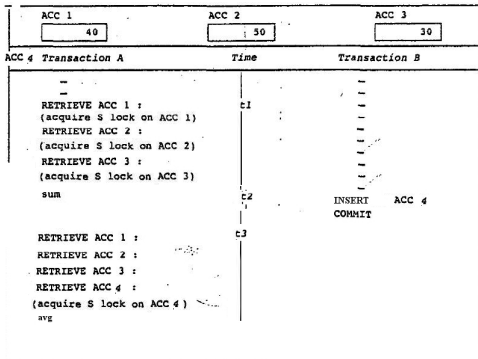
¿?



- ▶ La transacción A está calculando primero la suma y luego el promedio de las cuentas
- ▶ La transacción B está adicionando una nueva cuenta ACC4 100

# ¿Solución?

¿?



- ▶ La transacción A está calculando primero la suma y luego el promedio de las cuentas
- ▶ La transacción B está adicionando una nueva cuenta ACC4 100

¡No es correcto! Fantasmas.