# Introducción a las Transacciones

*Autor: Sergio D'Arrigo*

# Motivación

Supongamos que estamos haciendo una transferencia de $ 1.000 de una cuenta “C1” a una cuenta “C2” de un mismo banco, el saldo original de C1 es $1.500 y el de C2 es $3.000

Una vez que ante la confirmación de una operación de transferencia el sistema realiza:

- Primero un débito a la cuenta origen
- Luego un crédito a la destino

- ¿Cuál sería el resultado esperado una vez finalizado?
  - ✓Saldo C1: $500
  - ✓Saldo C2: 4.000

# Motivación

Y si el resultado luego de haber confirmado fuese otro? Por ej…

- Saldo C1 = $500 y Saldo C2 = $3000
- Saldo C1 = 500 y saldo C2 = 4000, pero al día siguiente aparece Saldo C1=$1.500 y saldo C2= $3.000
- Saldo C1= $1.500 y saldo C2 = $3.000

¿qué piensan de estos resultados no esperados? ¿son aceptables?

¿Qué se les ocurre que pudo haber pasado para estos resultados no esperados?

# Motivación

- Y si en el mismo momento otro usuario le estuviese transfiriendo también $1000 a C2 desde la cuenta C3 (con saldo inicial $2000)
- Qué esperaríamos al haber terminado ambas operaciones (sin ninguna otra que afecte a esas cuentas?
- Saldo C1 = $500
- Saldo C2 = $5000
- Saldo C3 = $1000

Y si el resultado fuese C1:$500, C2:$4000 y C3:$1000?

Qué piensan? Qué se les ocurre que pudo haber pasado

# Transacciones

## Transacción

- Es un programa (o segmento de programa) en ejecución que forma una **unidad lógica de procesamiento** de la base de datos
  - Una transacción incluye una o más operaciones de acceso a la base de datos

- **Qué esperamos** de una transacción?
  - Que se ejecute completa o nada
  - Que genere los resultados como si estuviese ejecutando sola
  - Que los cambios aplicados a la base de datos perduren una vez que terminó la transacción
  - Que mantenga la consistencia de la Base de Datos

- Los bloques de código comprendidos en una transacción se especifican mediante en el código por sentencias del estilo BEGIN TRANSACTION y END TRANSACTION.

# Propiedades ACID de las transacciones

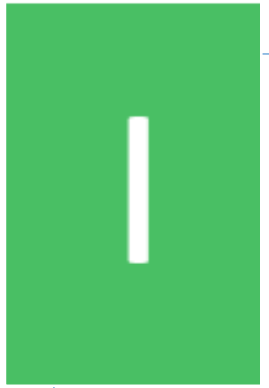
- Es deseable que un DBMS garantice las siguientes propiedades de las transacciones:


## Atomicidad

- Todas las operaciones de la transacción se reflejan adecuadamente en la base de datos, o ninguna lo hace.


## Consistencia

- La ejecución de una transacción en aislamiento (es decir, sin que ninguna otra transacción se ejecute simultáneamente) debe preservar la consistencia de la base de datos.




## **Aislamiento** (Isolation)

- Cada transacción debe aparecer como si se estuviera ejecutando sin que ninguna otra transacción se esté ejecutando al mismo tiempo


## Durabilidad

- Una vez que una transacción finaliza correctamente, todos los cambios realizados sobre la base de datos persisten a futuro, aún si ocurriesen fallos en el sistema.

# Data ítems

## Data Items

- Son los recursos primarios de la base de datos, que pueden ser accedidos por usuarios o programas al recuperar datos o modificar la base de datos.
- Al modelar el procesamiento de transacciones, las bases de datos se representan como una colección de data ítems nominados
- Un data ítem puede ser de diferente granularidad: un bloque, un registro de datos, un campo de un registro de datos.

## Operaciones sobre un data ítem X

- Read(X)
- Write(X)

# Operaciones relevantes a representar

## Begin

- Indica el comienzo de la transacción

## Read

- Indica una operación de lectura de un data ítem

## Write

- Indica una operación de escritura de un data ítem

## End

- Indica el fin de una transacción

## Commit

- Indica el fin exitoso de la transacción.
- Los **cambios** aplicados **se guardan de forma segura**, no pudiendo ser deshechos

## Rollback / Abort

- Indica el fin no exitoso de la transacción.
- Los **cambios** aplicados por la transacción **son deshechos**.

# Diagrama de Estados de las Transacciones

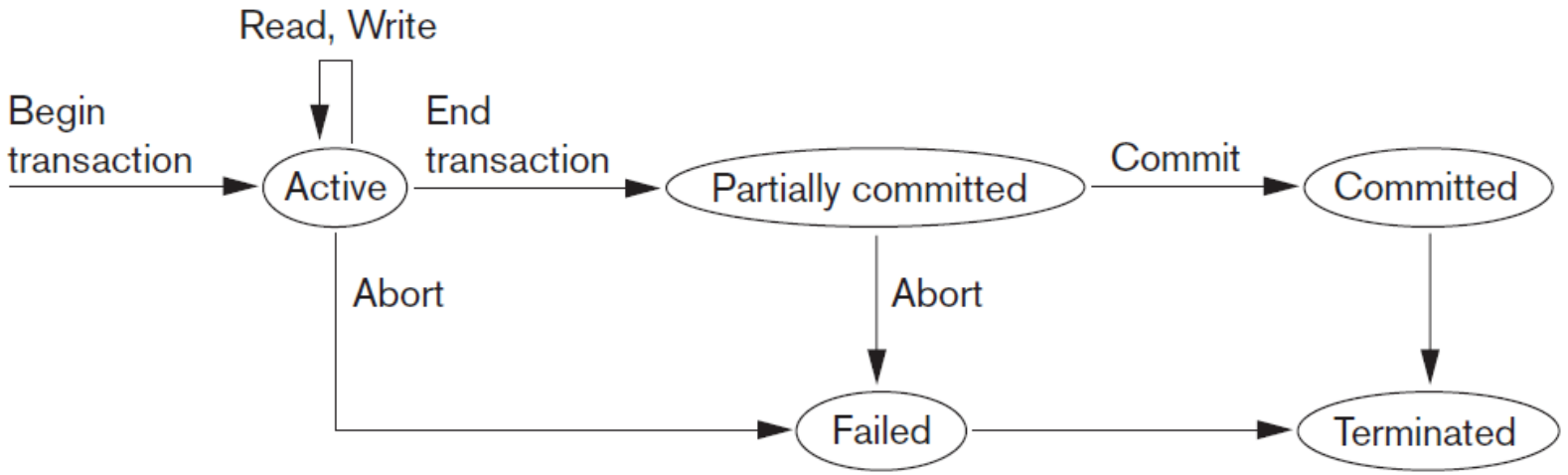
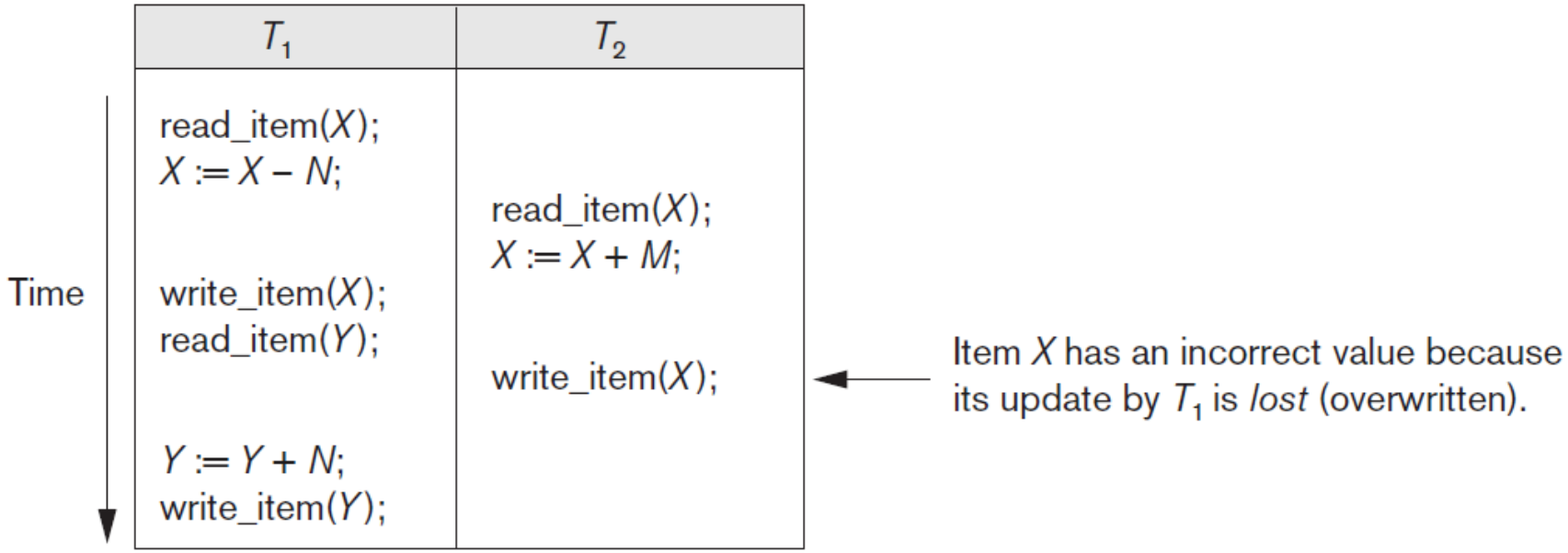



**Figure 20.4**
State transition diagram illustrating the states for transaction execution.

# Problemas potenciales por concurrencia

**Actualización perdida** (lost update)

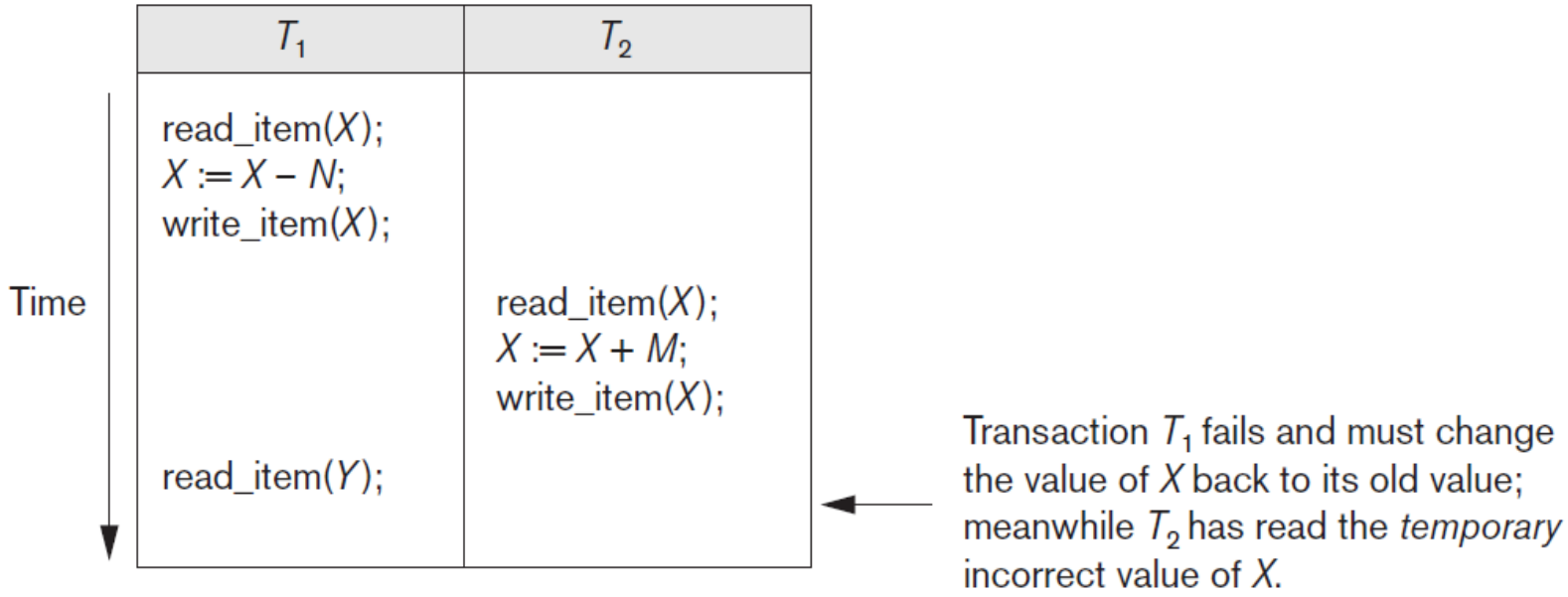
(a)



# Problemas potenciales por concurrencia

- **Actualización perdida** (lost update)
- **Lectura Sucia** (Temporary update o dirty read)

(b)



# Problemas potenciales por concurrencia

- **Actualización perdida** (lost update)
- **Lectura Sucia** (Temporary update o dirty read)
- **Resumen incorrecto** (Incorrect Summary)

(c)

| $T_1$ | $T_3$ |
|---|---|
| | sum := 0;<br>read_item(A);<br>sum := sum + A;<br>. . . |
| read_item(X);<br>X := X – N;<br>write_item(X); | |
| | read_item(X);<br>sum := sum + X;<br>read_item(Y);<br>sum := sum + Y; |
| read_item(Y);<br>Y := Y + N;<br>write_item(Y); | |

← $T_3$ reads $X$ after $N$ is subtracted and reads $Y$ before $N$ is added; a wrong summary is the result (off by $N$).

# Problemas potenciales por concurrencia

- **Actualización perdida** (lost update)
- **Lectura Sucia** (Temporary update o dirty read)
- **Resumen incorrecto** (Incorrect Summary)

- Necesitamos **control de concurrencia**!!!

# Clasificación de fallas

- **Tipos de fallas**

## de Transacción

- Error Lógico
  - Ej input erróneo, overflow
- Error de Sistema
  - Ej deadlock

## de Sistema

- Ej malfuncionamiento de hardware, o bug en SO o SGBD

## de Disco

- Ej, un bloque de disco pierde contenido, o caen las cabezas de disco en una operación

- La ocurrencia de fallas durante la ejecución pueden dejar la base de datos en un estado inconsistente
- Necesitamos mecanismos de **recuperación ante fallas**!!!

# Bibliografía


- *Elmasri R. & Navathe S. (2016) Fundamentals of Database Systems (7ma. Ed.) Pearson, Cap. 20*
- *Silberschatz A., Korth H. & Sudarshan S.(2020) Database System Concepts (7ma. Ed.), Mc.Graw Hill, Cap. 17*
- *Ramakrishnan r. & Gehrke J. (2003) Database System Concepts (3ra. Ed.), Mc.Graw Hill, Cap. 16*