

# Tema 12: Transacciones

Introducción a la Ingeniería del Software y los Sistemas de Información I  
Ingeniería Informática – Tecnologías Informáticas  
Departamento de Lenguajes y Sistemas Informáticos



1. ¿Qué es una transacción?
2. Propiedades ACID
3. Problemas de concurrencia
4. Aislamiento de transacciones
5. Recuperación de transacciones en SGBDs

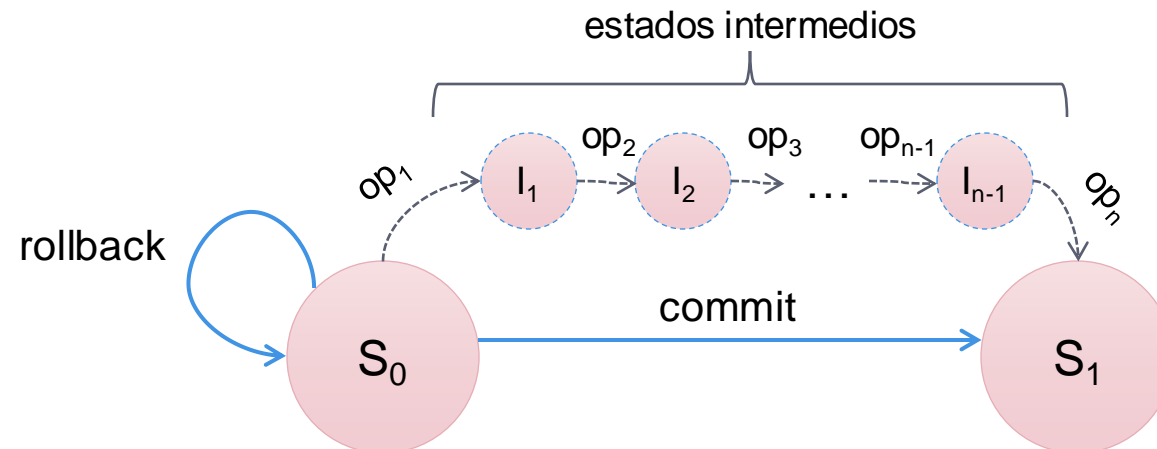
# ¿Qué es una transacción en una base de datos?

- Es una **secuencia de operaciones** ( $op_1..op_N$ ) que forman una **unidad lógica de trabajo** en una BD.
- Normalmente, se corresponden con la ocurrencia de algún **hecho** en el entorno del Sistema de Información del que se deba tener memoria.

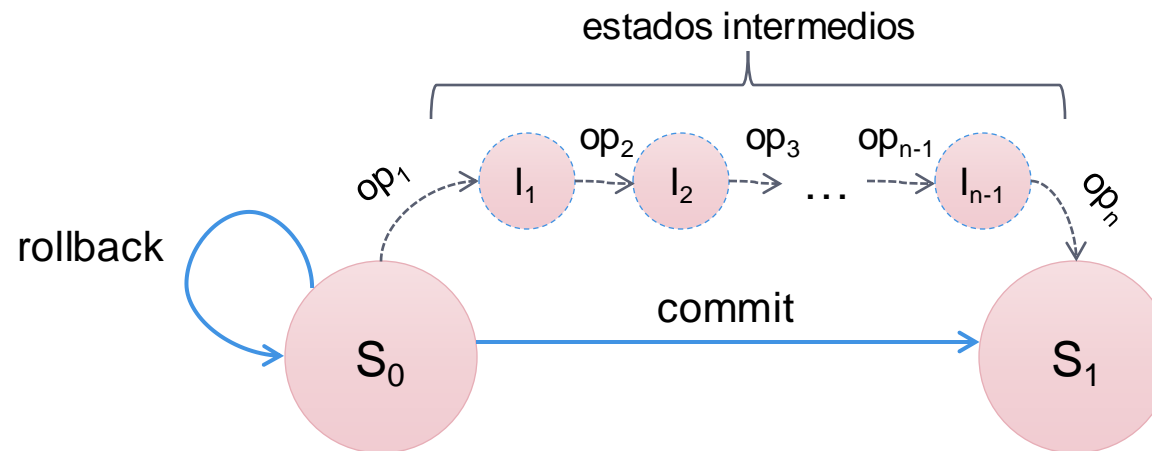




- **Atomicidad:** si se confirma una transacción (commit), se ejecutan todas las operaciones de la misma; si se cancela una transacción (rollback), no se ejecuta ninguna.
- No se puede ejecutar parcialmente, dejando la base de datos en un estado intermedio ( $I_i$ ).

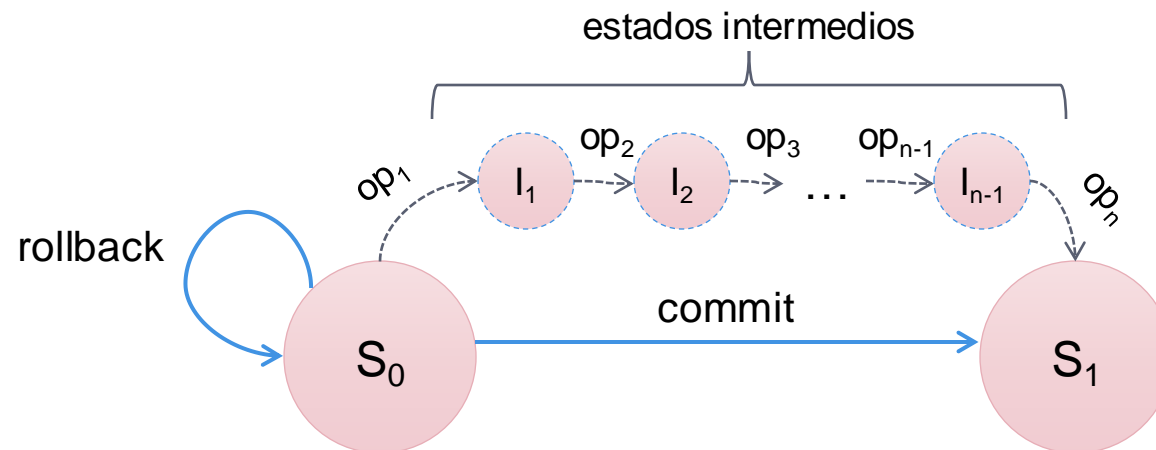


- **Consistencia:** a partir de un estado inicial consistente ( $S_0$ ), se deja la base de datos en otro estado consistente ( $S_1$ ), que respeta todas las reglas de integridad.
- Los estados intermedios ( $I_{1..n}$ ) no son necesariamente consistentes\*.

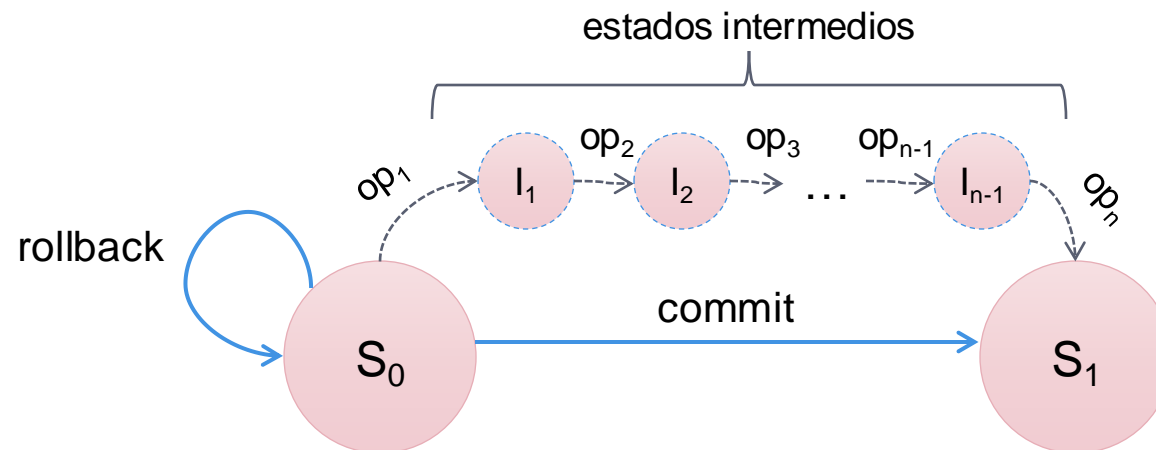


\*Por eso es necesario que sea atómica, para no dejar la base de datos en un estado inconsistente.

- **Aislamiento:** la transacción se realiza como si fuera la única ejecutándose en la base de datos en ese momento.
- Los estados intermedios ( $I_{1..n}$ ) no deben ser visibles para otras transacciones.
- Es la propiedad **menos desarrollada** en los SGBD.

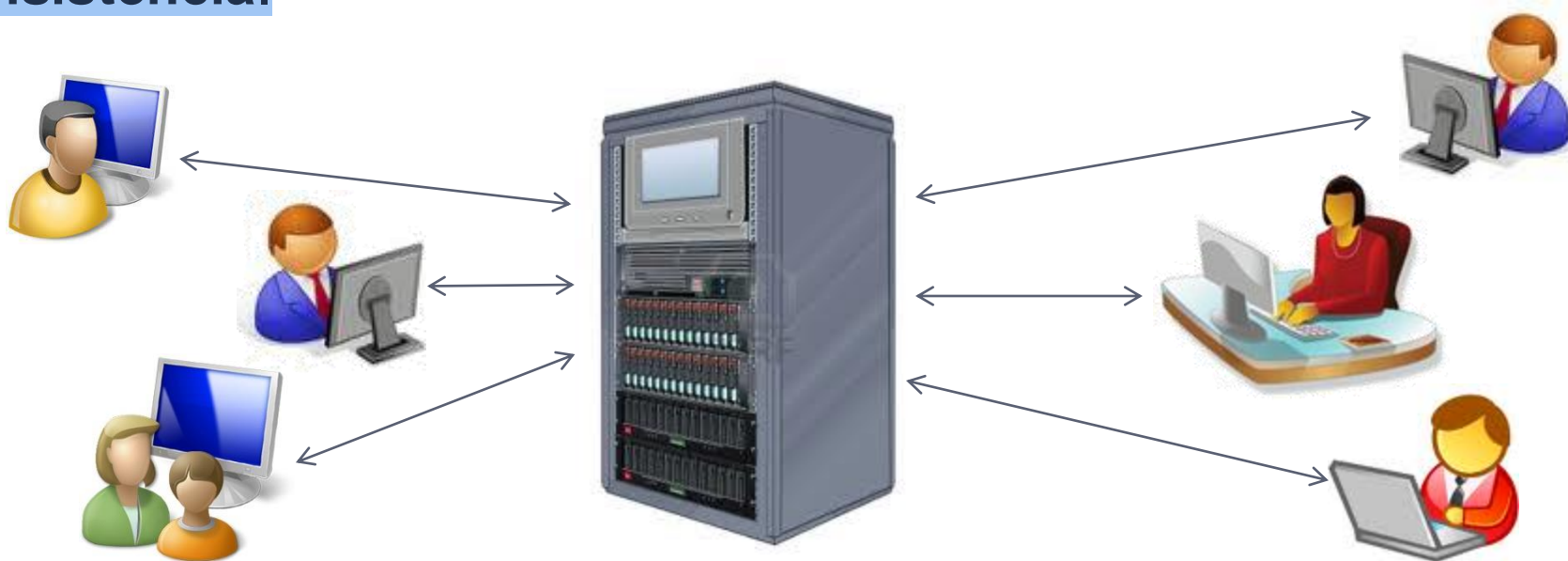


- **Durabilidad:** si se confirma una transacción (**commit**), los cambios en la base de datos se hacen **permanentes**.
- El SGBD es responsable de que los cambios sean permanentes, incluso aunque se produzca un **fallo**.



# Problemas de concurrencia

- **Múltiples usuarios simultáneos** del sistema de Información **provocan la** ejecución de múltiples **transacciones concurrentes.**
- Si el aislamiento no está garantizado por el SGBD, **puede haber problemas de consistencia.**





## Actualización perdida (lost update)

- Unas transacciones sobrescriben las actualizaciones de otras.

Transacción 1	Transacción 2
$x_1 = \text{LEER}(X);$	
	$x_2 = \text{LEER}(X);$ $x_2 = x_2 + 1;$ <b>ESCRIBIR</b> ( $X, x_2$ ); <b>COMMIT</b> ;
$x_1 = x_1 + 1;$ <b>ESCRIBIR</b> ( $X, x_1$ ); <b>COMMIT</b> ;	

se pierde

## Lectura sucia (dirty read)\*

- Se lee un valor que está siendo modificado por otra transacción que no ha finalizado y que podría cancelarse o fallar.

Transacción 1	Transacción 2
$x_1 = \text{LEER}(X);$ $x_1 = x_1 + 1;$ <b>ESCRIBIR</b> ( $X, x_1$ );	
	$x_2 = \text{LEER}(X);$ <b>ESCRIBIR</b> ( $X, x_2$ ); <b>COMMIT</b> ;
<b>ROLLBACK</b> ;	

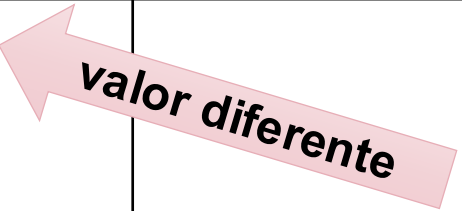
lectura sucia

\*Denominado también *dependencia no confirmada*.

## Lectura no repetible (non-repeatable read)

- Se obtienen lecturas diferentes del mismo valor durante la misma transacción.

Transacción 1	Transacción 2
$x_1 = \text{LEER}(X);$ $\text{IMPRIMIR}(x_1);$	
	$x_2 = \text{LEER}(X);$ $x_2 = x_2 + 1;$ $\text{ESCRIBIR}(X, x_2);$ $\text{COMMIT};$
$x_1 = \text{LEER}(X);$ $\text{IMPRIMIR}(x_1);$ $\text{COMMIT}$	



valor diferente

## Lectura fantasma (phantom read)

- Se obtienen lecturas diferentes de un conjunto de tuplas durante la misma transacción.

Transacción 1	Transacción 2
$\{x_i\} = \text{LEER}(\{X\});$ $\text{IMPRIMIR}(\{x_i\});$	
	$\{x_j\} = \text{LEER}(\{X\});$ $\{x_j\} = \{x_j\} \cup \{x_k\};$ $\text{ESCRIBIR}(\{X\}, \{x_j\});$ $\text{COMMIT};$
$\{x_i\} = \text{LEER}(\{X\});$ $\text{IMPRIMIR}(\{x_i\});$ $\text{COMMIT}$	

una tupla (fantasma) más

## ¿Cómo evitar los problemas de concurrencia?

- Ejecutando las transacciones **secuencialmente**:

- Reduce mucho el rendimiento de un SGBD.

$$T_1 \rightarrow T_2 \rightarrow T_3 \rightarrow \dots \rightarrow T_n$$

- Ejecutando las transacciones **concurrentemente** pero con un **plan de ejecución secuenciable**:

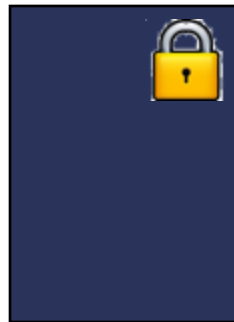
- Debe tener el mismo efecto final que si las transacciones se ejecutaran secuencialmente (**secuenciabilidad**).
  - Normalmente, el SGBD usa **bloqueos**, marcas de tiempo (timestamps) o versiones múltiples de los datos para garantizar la secuenciabilidad.



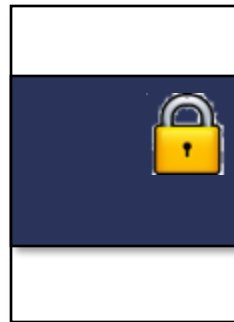
## Gránulo de bloqueo

- Un **gránulo** es una unidad que puede ser bloqueada por una transacción.
- Dependiendo del SGBD, el gránulo puede ser una tabla, un rango de filas, una sola fila, una columna de una fila, etc.
- Si una transacción intenta bloquear un gránulo ya bloqueado, **espera** hasta su liberación.

**Tabla**



**Rango**



**Fila**



**Columna**

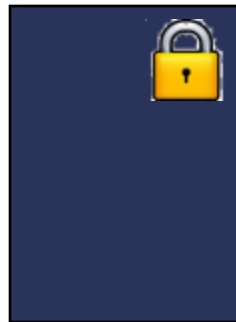




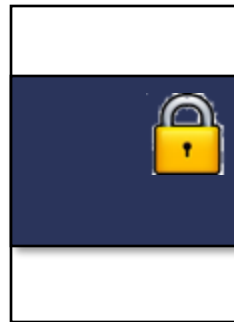
## Tipos de bloqueos

- **Para lectura** (compartido): un gránulo puede estar bloqueado para lectura por varias transacciones; los intentos de bloqueo exclusivo deben esperar.
- **Para escritura** (exclusivo): un gránulo puede estar bloqueado para escritura por una sola transacción; los demás intentos de bloqueo deben esperar.

**Tabla**



**Rango**



**Fila**



**Columna**



SQL-92 define cuatro niveles de aislamiento dentro de una transacción:

- **READ\_UNCOMMITTED**: permite leer datos no confirmados de otras transacciones.
- **READ\_COMMITTED**: sólo permite leer datos confirmados antes de que se ejecute una consulta. Suele ser el valor por defecto.
- **REPEATABLE\_READ**: sólo permite leer datos confirmados antes de que comience la transacción.
- **SERIALIZABLE**: garantiza la secuenciabilidad de la transacción con todas las demás transacciones concurrentes; si no puede garantizarlo, la transacción falla.



- Se debe especificar al comienzo de la transacción:

**SET TRANSACTION ISOLATION LEVEL nivel;**

















- El nivel por defecto suele ser **READ\_COMMITTED**, que ofrece un buen rendimiento.
- Para solucionar el problema de la actualización perdida con nivel **READ\_COMMITTED**, algunos SGBD permiten **bloquear para escritura** al hacer un **SELECT** mediante la cláusula **FOR UPDATE:**

**SELECT \* FROM** tabla








**WHERE** condición

**FOR UPDATE;**

 **bloquea para escritura**

Nivel de aislamiento	Problemas de concurrencia			
	Lost update	Dirty read	Non-repeat. read	Phantom read
READ_UNCOMMITTED				
READ_COMMITTED	 *			
REPEATABLE_READ				
SERIALIZABLE				

\* Salvo que se use la cláusula FOR UPDATE (ver transparencia anterior).

Nivel de aislamiento	Bloqueos		
	Bloqueo Escritura	Bloqueo Lectura	Bloqueo de Rango
READ_UNCOMMITTED			
READ_COMMITTED			
REPEATABLE_READ			
SERIALIZABLE			

- Mantiene el bloqueo hasta el final de la transacción.
- Mantiene el bloqueo hasta el final de la operación.



- ¿Cómo se comportan los escenarios de los problemas de concurrencia en cada nivel de aislamiento?
- ¿Puede ocurrir un interbloqueo de dos transacciones?



## Tipos de fallos en bases de datos

- **Fallo de una orden al SGBD**
  - Violación de reglas de integridad, tipos incorrectos, nombres de tablas o columnas erróneos, errores de sintaxis, división por cero, etc.
  - Lo detecta el SGBD y lo debe tratar la aplicación mediante el tratamiento de excepciones oportuno.
- **Fallo de transacción**
  - Error de programación, interbloqueo, fallo de una orden al SGBD, etc.
  - Puede detectarlo tanto el SGBD como la aplicación, aunque el tratamiento (ROLLBACK) es responsabilidad del SGBD.



## Tipos de fallos en bases de datos

- **Fallo del sistema**

- Fallo de hardware, problemas de alimentación eléctrica, etc. sin que el medio de almacenamiento se vea afectado.
- El SGBD es responsable de su recuperación mediante el uso de bitácoras (**logs**).
- Los logs siempre se escriben en disco **antes** que las transacciones, para poder recuperarlas después.



## Tipos de fallos en bases de datos

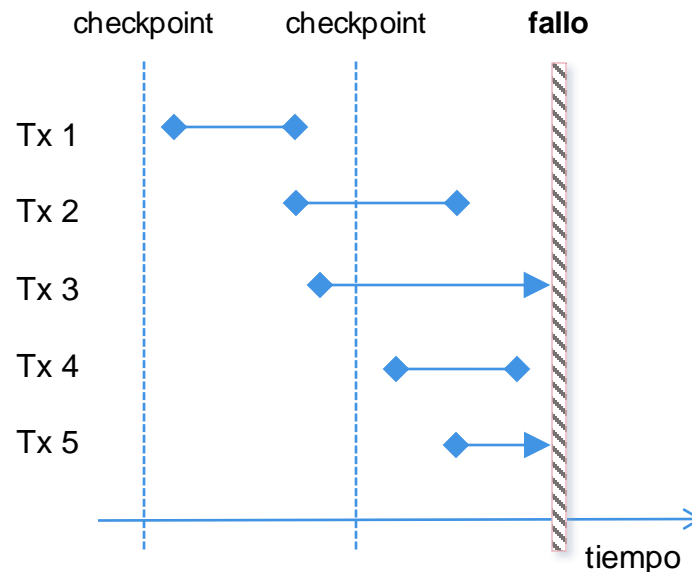
- **Fallo del medio**

- Fallo del hardware del medio de almacenamiento por avería, desastre natural, sabotaje, etc.
- Fallo de programación que ha llevado a la pérdida permanente de datos al sobrescribirlos o borrarlos.
- La recuperación es responsabilidad de los administradores de la base de datos, normalmente mediante copias de seguridad (**backups**).



## Algoritmos de recuperación de transacciones

- Cada cierto tiempo, el SGBD para el procesamiento de transacciones y vuelca todos los datos en memoria en el sistema de almacenamiento: **checkpoint**.



Tx 1: no hacer nada, está en disco.

Tx 2: rehacer, completa pero no en disco.

Tx 3: deshacer, incompleta.

Tx 4: rehacer, completa pero no en disco.

Tx 5: deshacer, incompleta.



- Entender el concepto de **transacción** en bases de datos y sus propiedades deseables.
- Conocer los problemas provocados por la **concurrency** en bases de datos y sus posibles soluciones.
- Conocer los **niveles de aislamiento** de transacciones en bases de datos SQL.
- Conocer los distintos tipos de **fallos** en bases de datos y la forma de **recuperarse** de ellos.

# Tema 12: Transacciones

Introducción a la Ingeniería del Software y los Sistemas de Información I  
Ingeniería Informática – Tecnologías Informáticas  
Departamento de Lenguajes y Sistemas Informáticos

