Administración de Bases de Datos

Tema 4

Gestión y control de la concurrencia

Objetivos

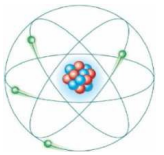
- Analizar el problema del control de la concurrencia en bases de datos.
- Conocer los mecanismos que emplea Oracle para el control de la concurrencia.

Transacciones y concurrencia

- Un programa de usuario puede realizar distintos tipos de operaciones, pero a nivel del SGBD, sólo nos interesan las de **lectura** y **escritura**.
- Una **transacción** es una serie de acciones llevadas a cabo por un único usuario o por un programa de usuario, que lee o actualiza el contenido de la base de datos.
- Ejemplo: una transferencia entre dos cuentas bancarias.
  - Parece que la operación está formada por una sola operación: “transferir un importe de la cuenta C1 a la cuenta C2”.
  - En realidad está formada por varias operaciones. Entre otras: restar de C1 y sumar en C2. Si en una de estas operaciones se produce un fallo, el saldo total sería incorrecto.
  - Es preferible que en caso de algún fallo, no tenga lugar ninguna operación.

- Las transacciones deben poseer 4 propiedades, conocidas como propiedades **ACID** (por sus siglas en ingles) y deben ser impuestas por los métodos de control de concurrencia y recuperación de los SGBD. Estas propiedades son:

• Atomicidad (Atomicity): se debe pensar en las transacciones como en un “todo”; o bien se ejecutan todas sus operaciones o bien no se ejecuta ninguna.

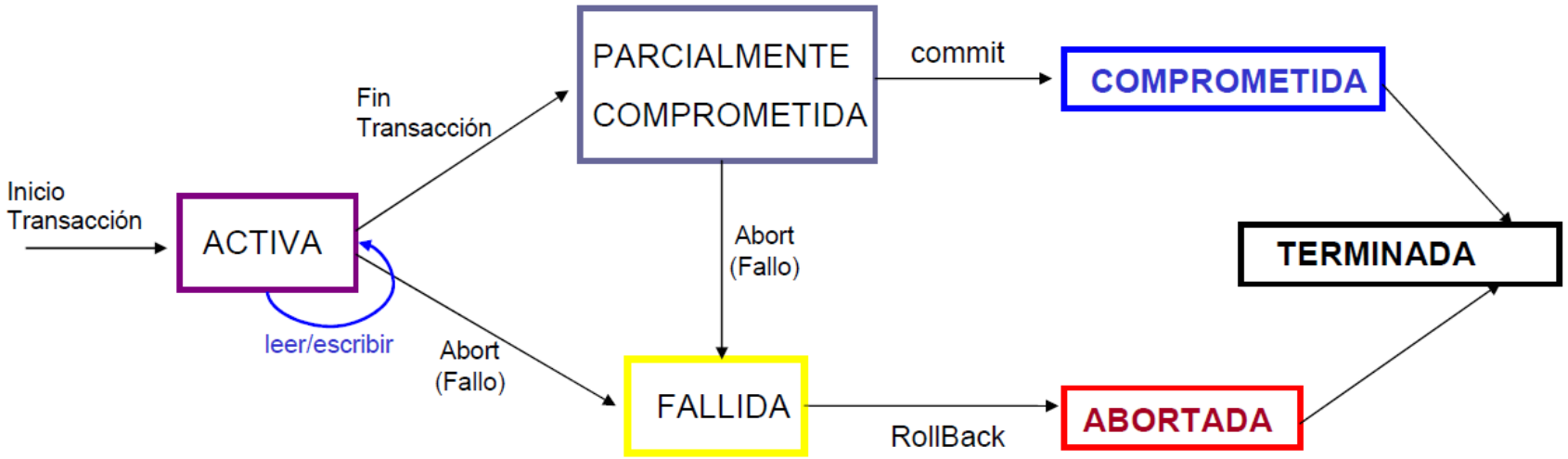


• Consistencia (Consistency): tras la ejecución de una transacción, la BD debe quedar en un estado consistente.

• Aislamiento (Isolation): las transacciones están protegidas de los efectos de la ejecución concurrente con otras transacciones.

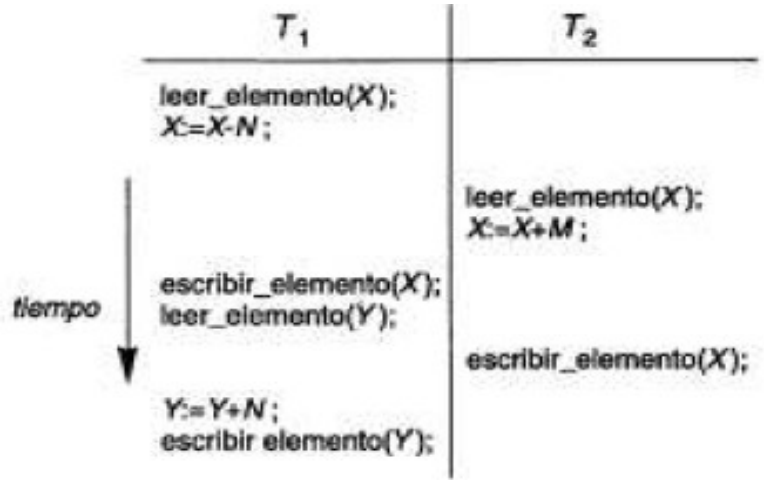
• Durabilidad (Durability): tras finalizar con éxito una transacción, los cambios realizados en la BD son permanentes, incluso si hay fallos en el sistema.

- Diagrama de transición de estados de una transacción:



Interferencia de transacciones

- Cuando varias transacciones se ejecutan de forma intercalada, el orden en el que las operaciones se llevan a cabo constituye lo que se conoce como **plan de las transacciones**.
- Un plan P de *n* transacciones T1, T2,…Tn, es un ordenamiento de las operaciones de las transacciones sujeto a la restricción de que, para cada transacción Ti que participe en P, las operaciones de Ti en P deben aparecer en el mismo orden que en Ti.



- **Serializabilidad**: los efectos de un conjunto de transacciones simultáneas deben dar el mismo resultado que si se ejecutaran en serie las transacciones individuales y que si cada una de las transacciones tuviera un uso exclusivo del sistema.

- Tipos de planes:

  - Plan en serie: no intercala las operaciones de diferentes transacciones.
  - Plan equivalente: si para cualquier estado de la BD, el efecto de ejecución de un plan es idéntico al de la ejecución de otro.
  - Plan serializable: aquel que es equivalente a un plan en serie.

Problemas de ejecución concurrente

- **Ejemplo**: reserva de asientos en una línea aérea.

  • T1: cancela N reservas de un vuelo (nº de asientos reservados en el vuelo X) y reserva el mismo número de asientos en otro vuelo Y.

  • T2: reserva M asientos en el vuelo X.

Operaciones:

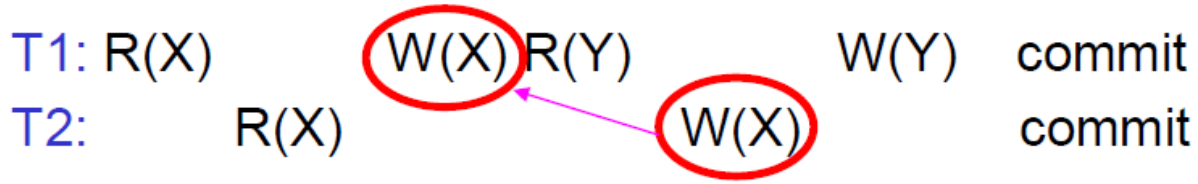
T1: X := X-N; Y := Y+N

R(X); W(X); R(Y); W(Y);

T2: X := X+M

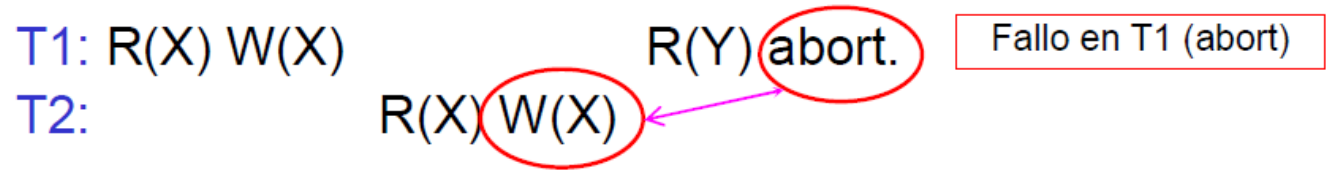
R(X); W(X);

- Problema 1 (**actualización perdida**): se pierde el cambio que ha afectado a una operación de escritura.



La modificación realizada por T1 es sobrescrita por T2.

- Problema 2 (**lectura sucia**): alguna transacción lee datos y posteriormente se produce un fallo.



T2 ha leído datos “sucios”, incorrectos o no comprometidos, lo que puede producir inconsistencia.

- Problema 3 (**lectura no repetible**): una transacción lee dos veces el mismo dato y obtiene diferentes valores en cada lectura.

T1: R(X) W(X) commit

T2: R(X) R(X) commit

- Problema 4 (**resumen incorrecto**): una transacción está realizando una operación de agregación y otra transacción está modificando las fuentes sobre las que se está realizando la función de agregación.

T3: R(A)W(s) ... R(X)W(s) R(Y)W(s) ... R(Z);W(s) commit

T4: R(X)W(X) R(Y)W(Y) commit

Caso particular: **fantasmas**. Una transacción selecciona datos en función de unos criterios de selección y otra transacción escribe algún dato que cumple esos criterios. Si se repite la misma selección se obtiene un conjunto de datos distinto.

- **Nivel de aislamiento**: el establecimiento del nivel de aislamiento de una transacción ayuda a evitar diversas anomalías de datos.

- Niveles de aislamiento:

  •Read uncommited: es el nivel más permisivo. Una transacción que se ejecuta con este nivel puede ver valores que otra transacción ha escrito, o deja ver valores que otra haya borrado, a pesar de que ésta no haya hecho commit.

  •Read commited: este nivel evita lecturas sucias, ya que la transacción sólo podrá leer datos que han sido reafirmados por el commit de otra transacción.

  •Repeatable read: este nivel evita el problema de lecturas no repetibles.

  •Serializable.

- Relajación del nivel de aislamiento: hay situaciones en las que es conveniente limitar el nivel de aislamiento de las transacciones para mejorar la concurrencia (sobre todo si se sabe que las interferencias no se van a producir realmente o no es importante que se produzcan).

CONFLICTO / NIVEL AISLAMIENTO	ACTUALIZACION PERDIDA	LECTURA NO CONFIRMADA	LECTURA NO REPETIBLE Y ANALISIS INCONSISTENTE (EXCEPTO FANTASMAS)	FANTASMAS
READ UNCOMMITED	SI	NO	NO	NO
READ COMMITED	SI	SI	NO	NO
REPEATABLE READ	SI	SI	SI	NO
SERIALIZABLE	SI	SI	SI	SI

SI= Si Protege / **NO=** No Protege