Ingeniería Técnica en Informática
Escuela Universitaria de Informática
Universidad Politécnica de Madrid

Asignatura: **Administración de Bases de Datos**

Tema 5:
Proceso de Transacciones

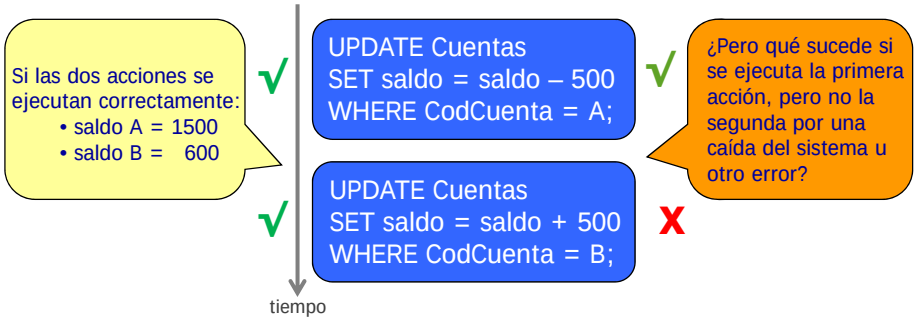
Pedro P. Alarcón Cavero
pedrop.alarcon@eui.upm.es

Febrero 2011

Contenido

1. Concepto de Transacción

- Ejemplo
  - Transferencia de 500€ de la cuenta A (con saldo 2000€) a la cuenta B (saldo 100€) de un mismo banco



Solución: ejecutar las dos operaciones como si fuesen una
- si se produce algún fallo, **anular** todas
- si no hay fallos, **confirmar** la ejecución de todas

Concepto de Transacción

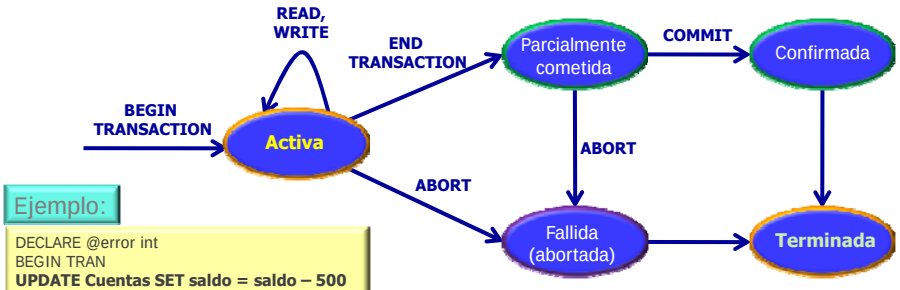
- Transacción
  - Unidad lógica de procesamiento secuencial compuesta por una o más acciones que se ejecutan en bloque sobre una base de datos (o todas, o ninguna)
  - Acciones sobre la base de datos
    - Read X --> Xi
    - Write Xi --> X
  - Sentencias
    - Begin/End Transaction (opcional en algunos SGBD)
    - Commit
      - Confirma (se hacen permanentes) los cambios producidos por la transacción en la BD
    - Rollback
      - Aborta la transacción, deshaciendo los cambios producidos por la transacción o por la parte ya ejecutada de ésta

Concepto de Transacción

- Propiedades deseables de una transacción (**ACID**)
  - **A**tomicity
    - Una transacción es atómica, o se realizan todas las acciones de la transacción o ninguna
    - Si la transacción no puede completarse, el módulo de recuperación tiene que deshacer sus efectos
  - **C**onsistency
    - Conservación de la consistencia de la base de datos tras su ejecución
    - Responsabilidad de los **programadores** o del módulo del SGBD que controla las restricciones de integridad
  - **I**solation
    - Aislamiento de una transacción respecto del resto de transacciones que se estén ejecutando en la BD. Sus operaciones no afectan ni se ven afectadas por otras.
    - Asegurado por el subsistema de control de concurrencia
  - **D**urability
    - Permanencia de sus efectos en la base de datos tras su ejecución
    - Responsabilidad del subsistema de recuperación del SGBD

Estados de una Transacción

- Diagrama de transición de estados



Ejemplo:

```
DECLARE @error int
BEGIN TRAN
UPDATE Cuentas SET saldo = saldo – 500
WHERE CodCuenta = A
SET @error = @@ERROR
IF (@error<>0) GOTO TrataError
UPDATE Cuentas SET saldo = saldo + 500
WHERE CodCuenta = B
SET @error = @@ERROR
IF (@error<>0) GOTO TrataError
COMMIT TRAN

TrataError:
IF @error <> 0 ROLLBACK TRAN
```

Read registro_cuentaA
Write registro_cuentaA

Read registro_cuentaB
Write registro_cuentaB

Parcialmente cometida

Abortada

Si los cambios producidos por la transacción pueden registrase en el **diario** pasa a estado Confirmada. En caso contrario se aborta.

2. Necesidad de controlar la concurrencia

- Las transacciones pueden ser correctas en sí mismas, pero al entremezclarlas el SGBD para ejecutarlas pueden producirse problemas:
  - Pérdida de operaciones
  - Introducción de inconsistencias
  - Imposibilidad de reproducir lecturas
  - Observación de inconsistencias
- La ejecución simultánea de transacciones debe ser correcta

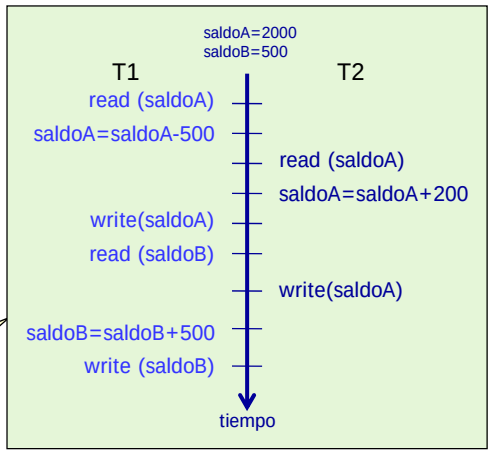
Solución sencilla pero ineficaz:
¡bloquear la base de datos entera**!**

Necesidad de controlar la concurrencia

- Pérdida de operaciones

T1 Transferir 500€ de cuenta A a B	T2 Ingresar 200€ en la cuenta A
read (saldoA)	read (saldoA)
saldoA=saldoA-500	saldoA=saldoA+200
write(saldoA)	write(saldoA)
read (saldoB)	
saldoB=saldoB+500	
write (saldoB)	

¿Es correcta la ejecución?

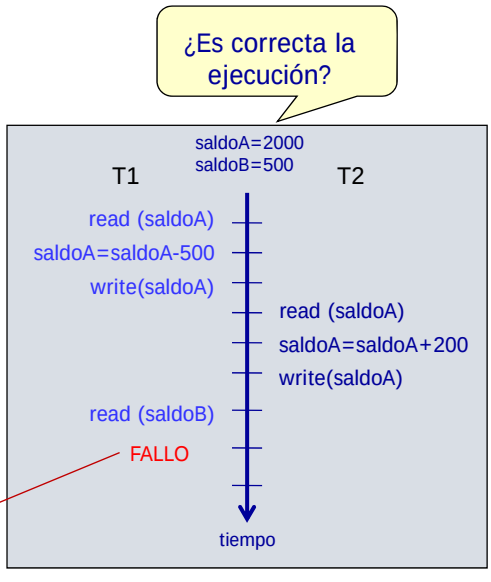


Contenido
Concepto
Necesidad
Planes
Bloqueos
Gestión
SQL

Necesidad de controlar la concurrencia

- Lectura sucia

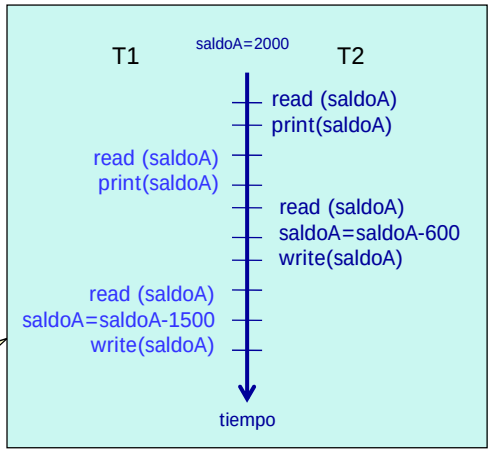
T1 Transferir 500€ de cuenta A a B	T2 Ingresar 200€ en la cuenta A
read (saldoA)	read (saldoA)
saldoA=saldoA-500	saldoA=saldoA+200
write(saldoA)	write(saldoA)
read (saldoB)	
saldoB=saldoB+500	
write (saldoB)	



Necesidad de controlar la concurrencia

- Imposibilidad de reproducir lecturas

T1 Retirar 1500€ de cuenta A	T2 Retirar 600€ en la cuenta A
read (saldoA)	read (saldoA)
print saldoA	print saldoA
(comprobar si hay saldo suficiente)	(comprobar si hay saldo suficiente)
read (saldoA)	read (saldoA)
saldoA=saldoA-1500	saldoA=saldoA-600
write (saldoA)	write(saldoA)

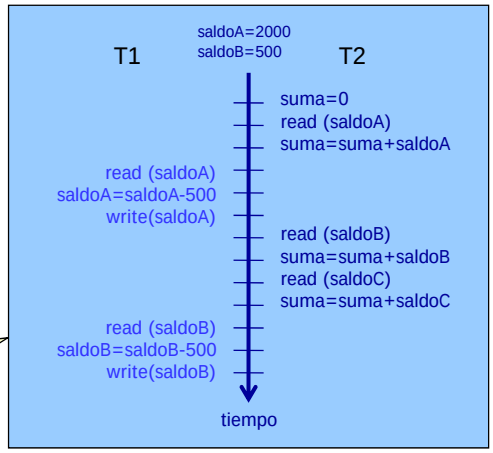


Necesidad de controlar la concurrencia

- Resumen Incorrecto

T1 Transferir 500€ de cuenta A a B	T2 Obtener el saldo de la Entidad
read (saldoA)	suma = 0
saldoA=saldoA-500	read (saldoA)
write(saldoA)	suma=suma+saldoA
read (saldoB)	read (saldoB)
saldoB=saldoB+500	suma=suma+saldoB
write (saldoB)	……

¿Es correcta la ejecución?



3. Planes y recuperabilidad

- Plan
  - Una secuencia de las operaciones realizadas por un conjunto de transacciones concurrentes que preserva el orden de las operaciones en cada una de las transacciones individuales

Plan serie (sucesión)

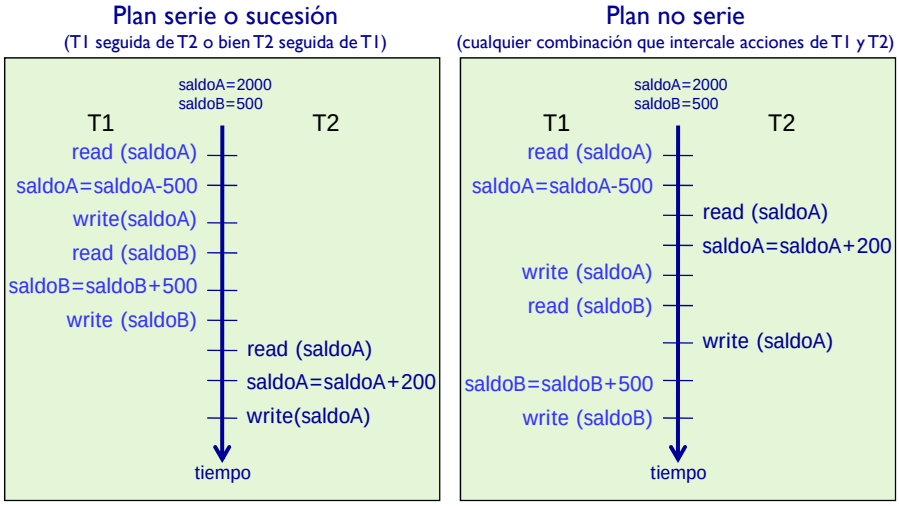
- *Una planificación en la que las operaciones de cada transacción se ejecutan consecutivamente sin que se entrelacen operaciones de otras transacciones*
- *Ausencia de conflictos al no producirse concurrencia*

Plan no serie

- *Una planificación en la que las operaciones de un conjunto de transacciones concurrentes están entrelazadas*
- *Pueden producirse conflictos: perdidas de operación, inconsistencias*

Planes y recuperabilidad

- Ejemplos



Planes y recuperabilidad

- Objetivo
  - Evitar la ausencia de conflictos al ejecutar los "planes no serie" sobre la BD
- Plan serializable
  - Un plan de n transacciones es serializable si produce el mismo resultado que una sucesión cualquiera de las n transacciones participantes
  - Condición suficiente para garantizar la ausencia de conflicto en "planes no serie" que se ejecuten en la BD
  - ¿Cómo?
    - Aplicando técnicas de control de concurrencia

Planes y recuperabilidad

- Enfoques para el control de concurrencia
  - Enfoque conservador o pesimista
    - Asumen la existencia de conflictos, se toman las medidas necesarias durante la ejecución, aumentando los tiempos
    - Técnicas más comunes
      - Marcas temporales
      - Bloqueos
  - Enfoque optimista
    - Asumen la ausencia de conflictos
    - Los posibles conflictos se resuelven al finalizar la ejecución de las transacciones
    - Si una transacción modifica unos datos, se vuelven a leer para comprobar si han variado desde que se leyeron. Y si han sido modificados, el programa debe tratar el "error"

4. Bloqueos

- Objetivo
  - Permitir únicamente la ejecución simultánea de operaciones compatibles
- Evitan la generación de ejecuciones incorrectas
- Transacciones con operaciones conflictivas sobre un objeto esperan
- Modo de operación
  - Clase que caracteriza a una operación
  - Determinan compatibilidades entre operaciones
  - Modos de operación clásicos: **lectura** y **escritura**
  - Se extienden para optimizar los algoritmos de bloqueo

4. Bloqueos

- Protocolo de bloqueo
  - Sobre un gránulo, y simultáneamente, se permite ejecutar operaciones con modos compatibles
  - Protocolo que indica el acceso a un gránulo, que puede llegar a ser compartido, caracterizado por petición de autorización para realizar una operación, y señales que indican la finalización de la operación
  - Es necesario conocer: el comienzo, modos de operación y el final

LOCK (gránulo, modo)

UNLOCK (gránulo)

Bloqueos

- Algoritmos de bloqueo
  - Sobre un gránulo sólo se deben ejecutar operaciones compatibles
  - Información necesaria
    - Gránulo
    - Modos de las operaciones asignadas
    - Modos de las operaciones solicitadas
  - Matriz de compatibilidades para lectura y escritura
    - Permite determinar compatibilidades entre modos de operación

$$C = \begin{array}{c} \begin{matrix} m_1 & m_2 \end{matrix} \\ \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \end{array} \begin{matrix} m_1 \\ m_2 \end{matrix}$$

- valor 1 si m_i y m_j son compatibles
- valor 0 si m_i y m_j son incompatibles

Bloqueos

- Matriz de compatibilidad con modos extendidos
  - m_1: consulta protegida
  - m_2: consulta protegida
  - m_3: actualización no protegida
  - m_4: actualización protegida
  - m_5: consulta exclusiva
  - m_6: actualización exclusiva

$$C = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

- Tabla de bloqueos
  <nombre_gránulo, tipo_LOCK, número_lecturas>

Bloqueos

- Protocolos de bloqueo en dos fases
  - Bloquea primero todos los gránulos, ejecuta operaciones y luego desbloquea todos
  - Toda ejecución completa de un conjunto de transacciones con bloqueo en dos fases es serializable
  - Limita la cantidad de concurrencia al realizar bloqueos tempranos y liberaciones tardías de recursos
  - Variaciones del bloqueo en dos fases
    - Bloqueo básico
    - Bloqueo conservativo o estático --> bloquea antes de comenzar
    - Bloqueo estricto --> desbloquea después de terminar totalmente

Bloqueos

- Problemas
  - Interbloqueo (deadlock)
  - Esperada indefinida (livelock)

    una transacción no entra mientras exista otra
  - Hambruna (starvation)

    El módulo controlador siempre mata a la misma transacción para resolver un interbloqueo

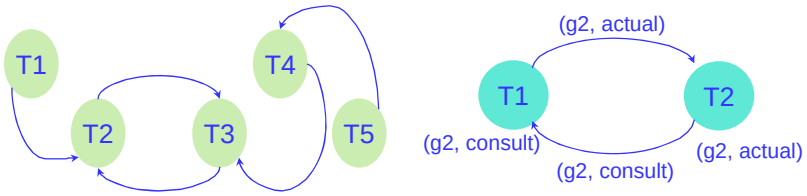
Bloqueos

- Interbloqueo
  - Imposibilidad de realizar simultáneamente dos operaciones no compatibles sobre el mismo gránulo
  - Dos o más transacciones quedan en espera de un recurso
  - La finalización de otras transacciones no supone una solución

T1	T2
LOCK (g1, actualización)	
	LOCK (g2, consulta)
LOCK (g2, actualización) **espera**	
	LOCK (g1, actualización) **espera**

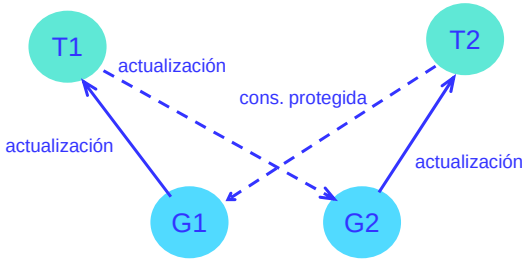
Bloqueos

- Interbloqueo: Grafos de espera
  - Nodos: transacciones concurrentes
  - Arcos: relación "espera a" . Ocurre si una transacción espera por un gránulo que tiene bloqueada otra
  - Interbloqueo si y solo si el grafo tiene un camino cerrado



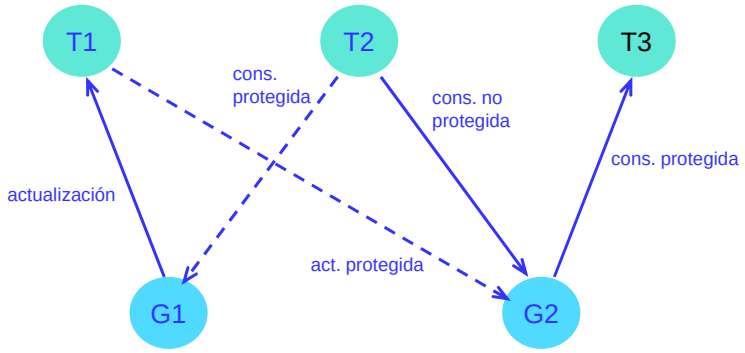
Bloqueos

- Interbloqueo: Grafos de asignaciones
  - Nodos: transacciones y gránulos
  - Arco continuo: de Gi a Tp si y solo si Tp ha obtenido el bloqueo de Gi en al menos un modo de operación
  - Arco discontinuo: de Tp a Gi, si Tp ha solicitado el bloqueo de Gi



Bloqueos

- Interbloqueo: Grafos de asignaciones
  - Existe interbloqueo si y solo si el grafo de asignaciones presenta un camino cerrado (condición necesaria pero no suficiente)



5. Gestión de Concurrencia

- Algoritmos de control de accesos
  - Algoritmos de ordenamiento inicial
  - Algoritmos de bloqueo
  - Algoritmos optimistas
- Importancia de la granularidad
  - Compromiso entre gestión y esperas
- Gestión de colas de transacciones
  - Decidir a qué transacción hay que dar mayor prioridad
- Auditorías al monitor de transacciones

6. Transacciones en SQL

- Modos de transacción
  - Confirmación automática
    - Cada instrucción individual es una transacción
  - Explícita
    - inicio
      - `BEGIN {TRAN|TRANSACTION} [nombre]`
    - final
      - `COMMIT {TRAN|TRANSACTION} [nombre]`
      - `ROLLBACK {TRAN|TRANSACTION} [nombre]`
  - Implícita
    - Se inicia implícitamente cuando se ha completado la transacción anterior, y finaliza con cualquiera de las siguientes acciones:
      - `COMMIT WORK`
      - `ROLLBACK WORK`
      - inicio de sesión

Transacciones en SQL

- Niveles de aislamiento (Isolation)
  - Lectura no confirmada
    - Una transacción puede ver cambios realizados por otra transacción aún no cometida (**Problema de lectura sucia**).
    - No la soportan muchos SGBDs
  - Lectura confirmada
    - Una transacción puede ver los cambios realizados por otras transacciones ya cometidas (**Problema de lectura no repetible**)
  - Lectura repetible
    - Si se ejecuta un mismo SELECT dentro de la transacción se obtiene el mismo resultado. Pero también se obtiene las filas que pueda insertar otra transacción, entre dos ejecuciones del mismo SELECT (**Problema de filas fantasmas**)
  - Serializable
    - Si se ejecuta un mismo SELECT más de una vez dentro de la transacción se obtiene siempre el mismo resultado

Transacciones en SQL

- Estándar SQL
  - SET TRANSACTION
    - READ UNCOMMITED
      - Permite lectura sucia
    - READ COMMITED
      - Permite lectura no repetible
    - REPETEABLE READ
      - Permite filas fantasmas
    - SERIALIZABLE
      - El nivel más restrictivo, ejecuciones más lentas

Nivel de aislamiento

Transacciones en SQL

- ORACLE
  - Transacción
    - SET TRANSACTION
      { { READ { ONLY | WRITE }
      | ISOLATION LEVEL
      { SERIALIZABLE | READ COMMITTED }
      | USE ROLLBACK SEGMENT rollback_segment
      }
      [NAME string]
      | NAME string
      } ;
    - COMMIT; SAVE POINT punto; ROLLBACK | {ROLLBACK punto};
    - Niveles de aislamiento
      - Read Only: Lectura repetible (Read Consistency)
      - Read Write: Lectura repetible (valor por defecto)
      - Serializable: serializable
      - Read Commited: lectura confirmada

Transacciones en SQL

- ORACLE
  - Bloqueo de tabla
    - LOCK TABLE IN modo NOWAIT;
      - modo
        - ROW SHARE
          - Permite acceso concurrente pero no permite bloqueo exclusivo sobre la tabla
        - ROW EXCLUSIVE
          - Permite acceso concurrente pero no permite bloqueo compartido
        - SHARE
          - Permite consultas sobre la tabla pero no Update
        - EXCLUSIVE
          - Permite consultas pero no permite ninguna otra actividad sobre la tabla
      - NOWAIT
        - Devuelve el control sin tener que esperar a que la tabla esté desbloqueada
    - No se aplica bloqueo entre lecturas y escrituras
    - Las sentencias INSERT, UPDATE o DELETE realizan un bloqueo ROW EXCLUSIVE de las filas afectadas por el WHERE
    - Las sentencias COMMIT y ROLLBACK desbloquean las filas