Transacciones, Concurrencia y Aislamiento

REPOSITORIOS DE INFORMACION

Grado en Ingeniería Informática del Software

Escuela de Ingeniería Informática
Universidad de Oviedo

José Antonio Sánchez Sánchez
sanchezsjose@uniovi.es

Introducción

- Objetivos
  - Comprender y afianzar los conceptos de transacción, concurrencia y aislamiento
  - Conocer las diferentes posibilidades de configuración de un SGBD en función del grado de concurrencia que se quiera
  - Conocer los grados de aislamiento tanto de Oracle como HSQLdb
- Datos de Conexión
  - Oracle
    - IP: 156.35.94.99. Puerto: 1521, SID:DESA
    - Usuario=”UOXXXXXX” //Usuario corporativo
    - Password=“YYYYYY”
  - HSQLdb
    - jdbc:hsqldb:hsql://localhost
    - Usuario= “sa” //Usuario por defecto
    - Password = “”

Introducción (II)

- Preparación
  - HSQLdb
    - Arrancar servidor HSQLdb. “Base de datos (hsqldb) para Agencia de Viajes” (campus virtual) y ejecutar data/startup
    - Ejecutar data/runManagerSwing.bat (2 veces = 2 sesiones)
    - Menú Options → Desactivar “Autocommit mode” en ambas sesiones
  - Oracle
    - Arrancar SQL Developer y conectarse con usuario
    - Crear la siguiente tabla:
      - CREATE TABLE viajes (Destino VARCHAR2(200), Precio NUMBER(5));
    - Añadir los siguientes viajes:
      - Londres, 450
      - Paris, 320
      - Roma, 280
    - SQLDeveloper → Herramientas → Preferencias → Base de Datos → Avanzada – Verificar opción de “Confirmación Automática” desactivada
    - SQLDeveloper → Herramientas → Preferencias → Base de Datos → Hola de Trabajo – Verificar 2 primeras casillas activadas (“nueva hoja para conexiones no compartidas)
    - Abrir una nueva hoja de trabajo

Repaso

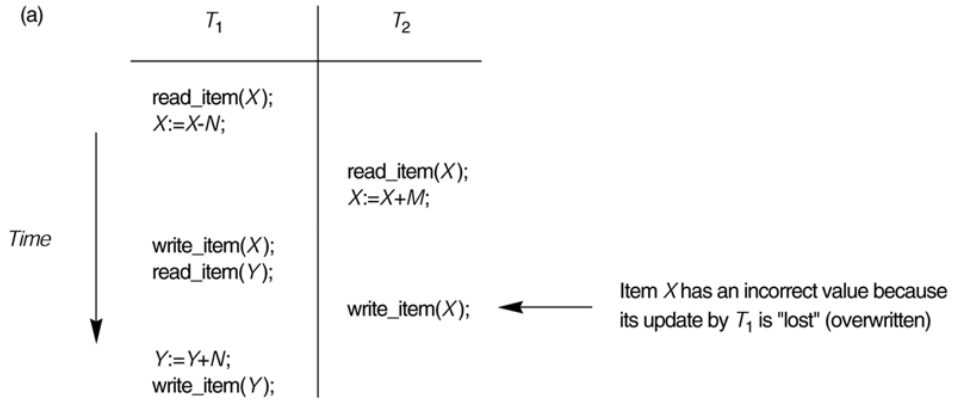
- **Transacción**: Unidad Lógica de procesamiento de la BD que incluye una o más operaciones de acceso (lecturas o escrituras)

- **Concurrencia**: Ejecución solapada en el tiempo de varias tareas de forma que se realizan de forma paralela y a la vez (o parcialmente a la vez)

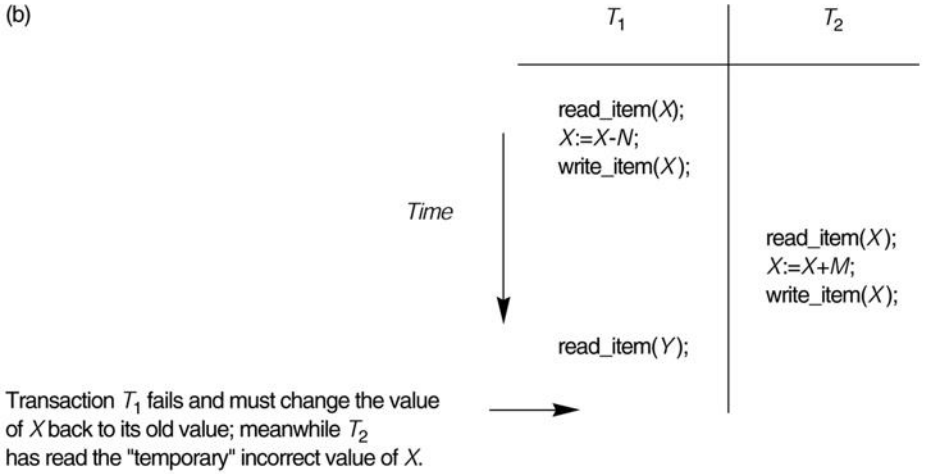
- **Aislamiento**: Modo de aislamiento de transacciones para obtener una concurrencia eficiente

Repaso – Concurrencia

- Problema de la actualización perdida

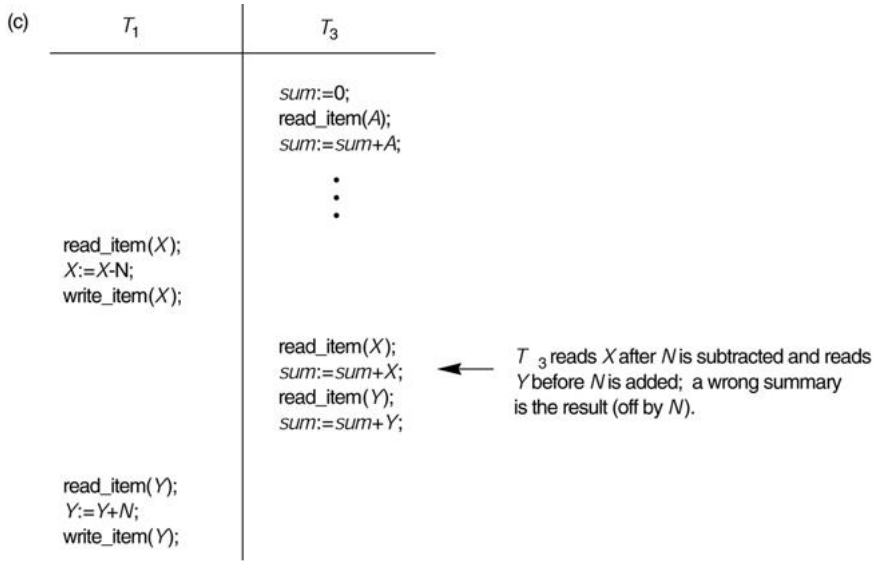


- Problema de la lectura sucia



Repaso – Concurrencia (II)

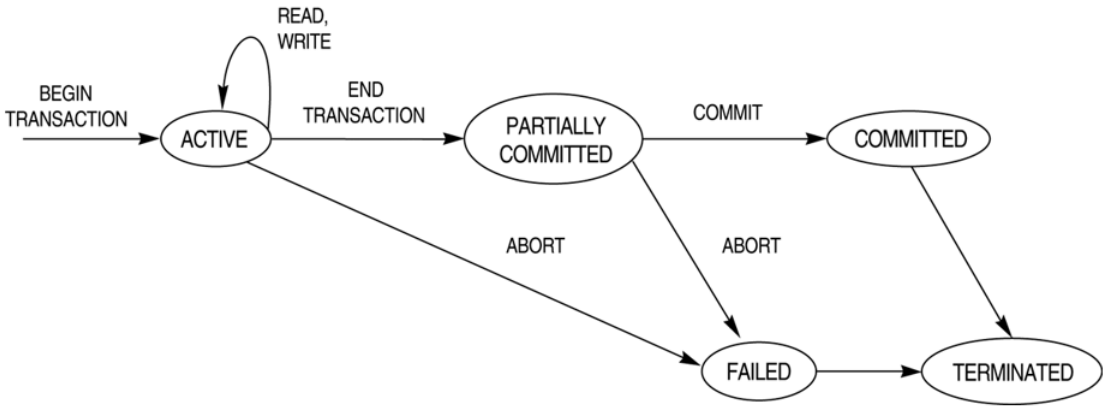
- Problema de resumen incorrecto



- Fallos:
  - Fallo del equipo (caída del sistema)
  - Error de transacción o sistema
  - Error local o condiciones de excepción detectadas en la transacción
  - Imposición de control de concurrencia (cuidado con niveles de aislamiento)
  - Fallo de disco
  - Problemas físicos/catástrofes

Repaso – Transacciones

- Transacción: Unidad atómica de trabajo que se realiza por completo o no se efectúa en absoluto
- Estados:
  - Activa
  - Parcialmente confirmada
  - Confirmada
  - Fallida
  - terminada



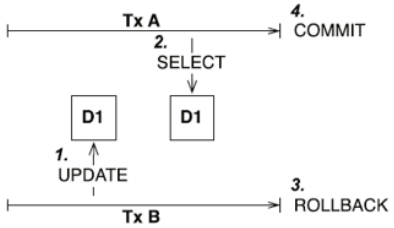
- Propiedades → **ACID**
  - Atomicidad (**A**tomicity)
  - Conservación de Consistencia (**C**onsistency)
  - Aislamiento (**I**solation)
  - Durabilidad o Permanencia (**D**urability)

Repaso – Transacciones en SQL

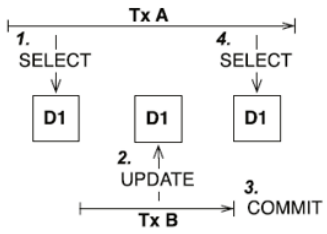
- Transacción SQL siempre es atómica → Aunque falle
- En SQL no hay un comienzo de transacción explícito
- Toda transacción debe terminar explícitamente con un COMMIT o un ROLLBACK
- Sentencia SET TRANSACTION → READ ONLY | READ WRITE
- Aislamiento ISOLATION LEVEL
  - READ UNCOMMITTED
  - READ COMMITTED
  - REPEATABLE READ
  - SERIALIZABLE

Repaso – Aislamiento

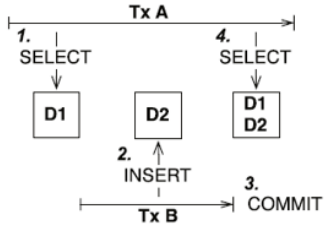
- Violaciones del Aislamiento:
  - **Lectura sucia**: Lectura de información no confirmada por otra transacción
  - **Lectura no repetible**: Lectura de información que puede ser modificada con lo que si se vuelve a leer la información será distinta
  - **Lectura fantasma**: Lectura de conjunto de filas en la que se puede insertar información nueva → Si se repite la sentencia aparecen nuevos datos que no estaban en la primera lectura



- *Dirty read:* TrxA lee datos que luego desaparecen por rollback



- *Unrepeatable read:* Durante txA txB es más rápida y modifica datos que vuelve a necesitar txA



- *Phantom read:* Durante txA txB inserta (o modifica) nuevos datos que txA va a detectar más tarde repitiendo la consulta (u otra que depende de ellos)

Repaso – Ajuste de Aislamiento

- Necesario ajustar bien el aislamiento que queremos ofrecer en la aplicación
- Sólo se puede probar cuando el sistema está en carga → producción
- Nivel muy restrictivo → Ralentiza mucho el acceso → Alta consistencia de datos
- Nivel muy bajo → Acceso rápido → Aparecen muchos datos inconsistentes difíciles de depurar
- **En búsqueda del equilibrio… ¿Cuál escoger? → Revisar Teoría**

Repaso – Soporte Transacciones en SQL

Nivel de aislamiento / Efecto de lectura	Lectura sucia	Lectura no repetible	Lectura fantasma
Read Uncommitted	Es posible	Es posible	Es posible
Read Committed	-	Es posible	Es posible
Repeatable Read	-	-	Es posible
Serializable	-	-	-

- En Oracle sólo existen dos niveles de aislamiento:
  - READ COMMITTED (por defecto)
  - SERIALIZABLE

- Comando para cambiar el nivel

```
SET TRANSACTION ISOLATION LEVEL {READ COMMITTED|SERIALIZABLE}
```

Repaso – Transacciones en JDBC

- Por defecto el objeto "*Connection*" está en modo "*AutoCommit*"

  `setAutoCommit(True|False)`

- Métodos para controlar transacciones:
  - commit → Valida la transacción
  - savepoint → Crea un punto de salvaguarda
  - rollback → Deshace la transacción completa o desde un savepoint

- Métodos para controlar nivel aislamiento (depende de SGBD)

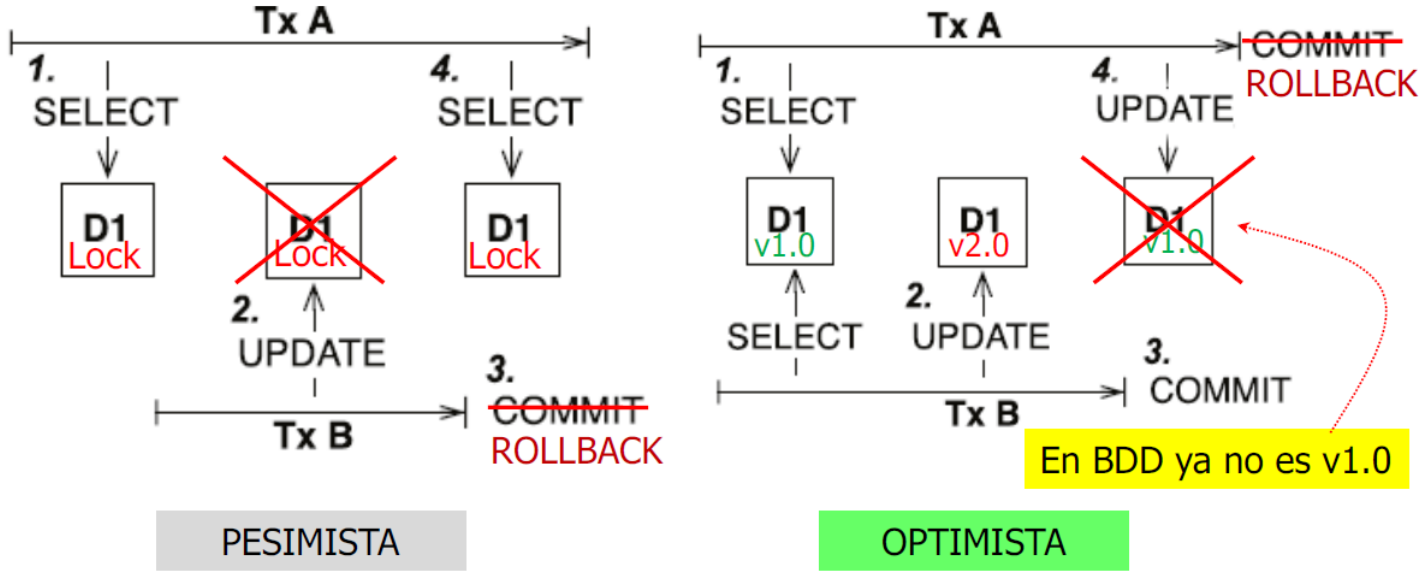
  `setTransactionIsolation(nivel aislamiento)`

  `getTransactionIsolation()` → Consulta nivel actual

  - *connection.TRANSACTION_READ_UNCOMMITTED*
  - *connection.TRANSACTION_READ_COMMITTED*
  - *connection.TRANSACTION_REPEATABLE_READ*
  - *connection.TRANSACTION_SERIALIZABLE*

Repaso – Control de Concurrencia

- Protocolos pesimistas (prevención)
  - Técnicas de bloqueo (locks)
  - Marcas de tiempo (time-stamping)
  - Multiversión
- Protocolos optimistas (corrección)
  - Validación o certificación



Repaso – Transacciones en JDBC (II)

- Ejemplo modificación *AutoCommit* y ejecución de transacción completa (finalizada por *commit* o *rollback*)

```
conn.setAutoCommit(false);

Statement stat = conn.createStatement();
stat.executeUpdate(comando1);
stat.executeUpdate(comando2);
...
conn.commit(); o bien conn.rollback();
```

Repaso – Transacciones en JDBC (III)

- Ejemplo Transacción con punto de salvaguarda (savepoint)

```
conn.setAutoCommit(false);
Statement stat = conn.createStatement();
...
stat.executeUpdate(comando1);
stat.executeUpdate(comando2);
Savepoint savept =conn.setSavepoint();
stat.executeUpdate(comando3);
stat.executeUpdate(comando4);
...
if (.....) conn.rollback(savept);//se deshacen 3 y 4 – y posteriores- quedan 1 y 2
conn.releaseSavepoint(savept);
...
conn.commit();
```

Ejercicios Transacciones – Comportamiento de SGBDs

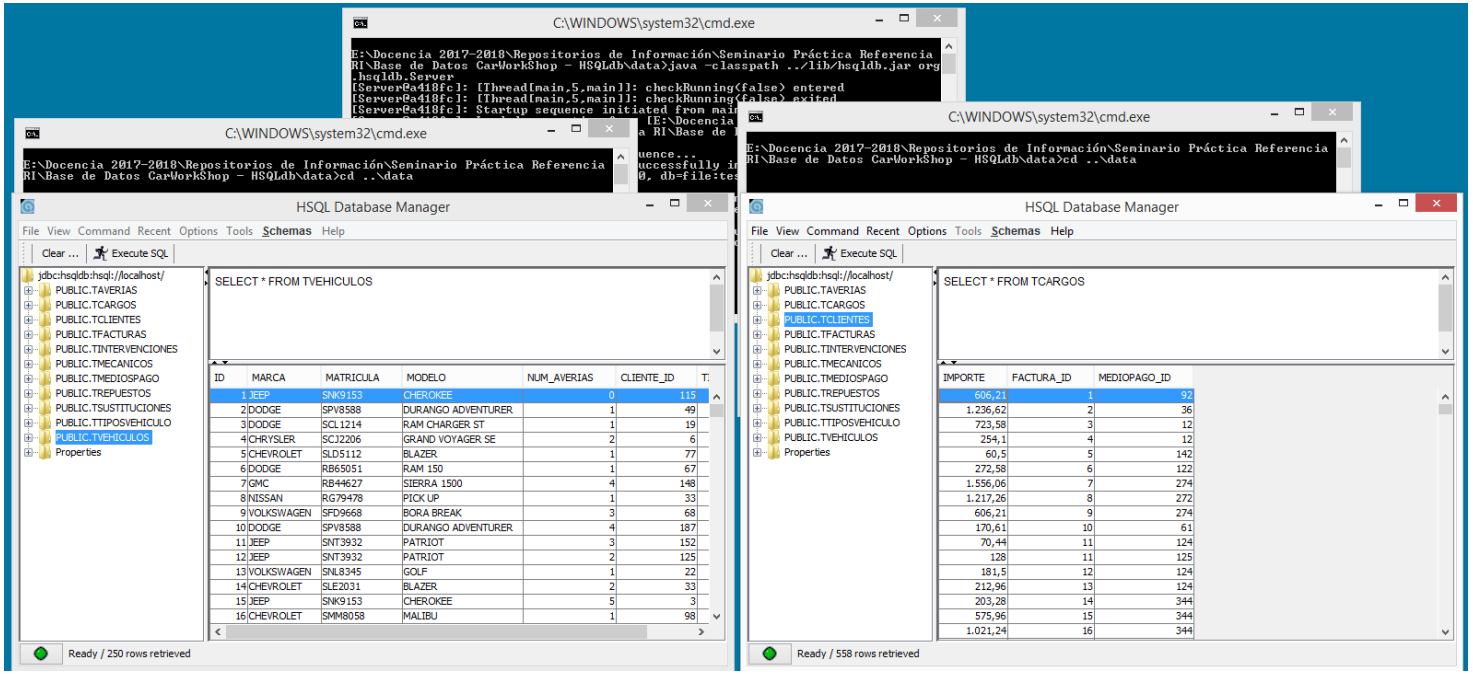
- Estudiar el comportamiento de los SGBDs ante diferentes efectos implementando diferentes niveles de aislamiento
  - HSQL (LOCKS) con READ COMMITTED y SERIALIZABLE
  - HSQL (MVCC) con READ COMMITTED y SERIALIZABLE
  - Oracle con READ COMMITTED y SERIALIZABLE

- Cambiar el modo de control de transacciones en HSQL
  - `SET DATABASE TRANSACTION CONTROL LOCKS | MVCC`

- Cambio de niveles de aislamiento
  - `SET TRANSACTION ISOLATION LEVEL XXXX` (hay que ejecutar antes commit o rollback)

Ejercicios Transacciones – Comportamiento de SGBDs (II)

- Ejecución de Secuencias de Acciones
- Apertura de 2 sesiones en el SGBD que estemos probando



- *Nota*: Reestablecer los valores como al principio para las siguientes pruebas

- Ejemplo de Acciones a Ejecutar

a. Lectura sucia

a.1)

SESION 1 (Ej: Agencia de viajes)	SESION 2 (Ej: Central de reservas)
SET TRANSACTION ISOLATION LEVEL XXXXXXX	
	UPDATE viajes SET precio = precio + 10
SELECT * from viajes	

(¿Qué ocurre en los diferentes niveles de aislamiento?)
Finaliza la transacción de la sesión 2 con rollback y la de la primera con commit/rollback

a.2)

SESION 1 (Ej: Agencia de viajes)	SESION 2 (Ej: Central de reservas)
SET TRANSACTION ISOLATION LEVEL XXXXXXX	
SELECT * from viajes	
	UPDATE viajes SET precio = precio + 10
SELECT * from viajes	

(¿Y ahora, ocurre lo mismo que antes?)
Finaliza la transacción de la sesión 1 con commit/rollback y la de la segunda con rollback.

Ejercicios Transacciones – Comportamiento de SGBDs (III)

- Otras acciones a ejecutar
  - Lectura no repetible
  - Lectura fantasma
- Fichero de Resultados editable para anotar la solución

b. Lectura no repetible

b.1)

SESION 1 (Ej: Agencia de viajes)	SESION 2 (Ej: Central de reservas)
SET TRANSACTION ISOLATION LEVEL XXXXXXX	
SELECT precio FROM viajes WHERE destino = 'Paris' **(1. ¿Qué precio se ve?)**	
	UPDATE viajes SET precio = 350 WHERE destino = 'Paris'
	Commit
SELECT precio FROM viajes WHERE destino = 'Paris' **(2. ¿Qué precio se ve?)**	
Commit	
SELECT precio FROM viajes WHERE destino = 'Paris' **(3. ¿Qué precio se ve ahora? ¿Por qué?)**	

c. Lectura fanstasma

SESION 1 (Ej: Agencia de viajes)	SESION 2 (Ej: Central de reservas)
SET TRANSACTION ISOLATION LEVEL XXXXXXX	
SELECT * FROM viajes WHERE precio BETWEEN 250 AND 350	
	INSERT INTO viajes VALUES ('Madrid',260)
	Commit
SELECT * FROM viajes WHERE precio BETWEEN 250 AND 350	

(¿Qué viajes se ven? ¿Qué ocurre en los diferentes niveles de aislamiento?)

Ejercicios Transacciones – JDBC

- Realización de una aplicación Java que use SGBD Oracle.
- Consulta sobre datos de una tabla para mostrarlos por consola en bucle infinito
- Cambio de nivel de aislamiento en distintas ejecuciones del programa (Recuerda establecer autocommit = false tanto en aplicación como en SQL Developer)
- Mientras tanto, desde SQL Developer crea/elimina/modifica registros en dicha tabla y observa el funcionamiento de los distintos niveles de aislamiento:
  - En READ_COMMITTED: los datos solo aparecen y desaparecen después de hacer un *commit*
  - En SERIALIZABLE: cuando hacemos consulta y se ha modificado un registro que devuelve esa consulta, se sigue observando el valor original pese a que se haya validado (*commit*) su modificación

¿Alguna Duda?