Transacciones

Edward Núñez
nunezedward@uniovi.es

Procesamiento de transacciones

- Sistemas monousuario
- Sistemas multiusuario
- Concurrencia:
  - Concurrencia intercalada (una CPU)
  - Procesamiento paralelo (+de 1 CPU)

Procesamiento de transacciones

- **Transacción**: una unidad lógica de procesamiento de la base de datos que incluye una o más operaciones de acceso a la base de datos (lectura –select- o escritura- insert, delete, update-).

Ciclo de vida de una transacción

- Inicio
- Lecturas/escrituras de elementos de la BD
- Final (pueden necesitarse verificaciones)
- Confirmación (COMMIT)
- Anulación (ROLLBACK)

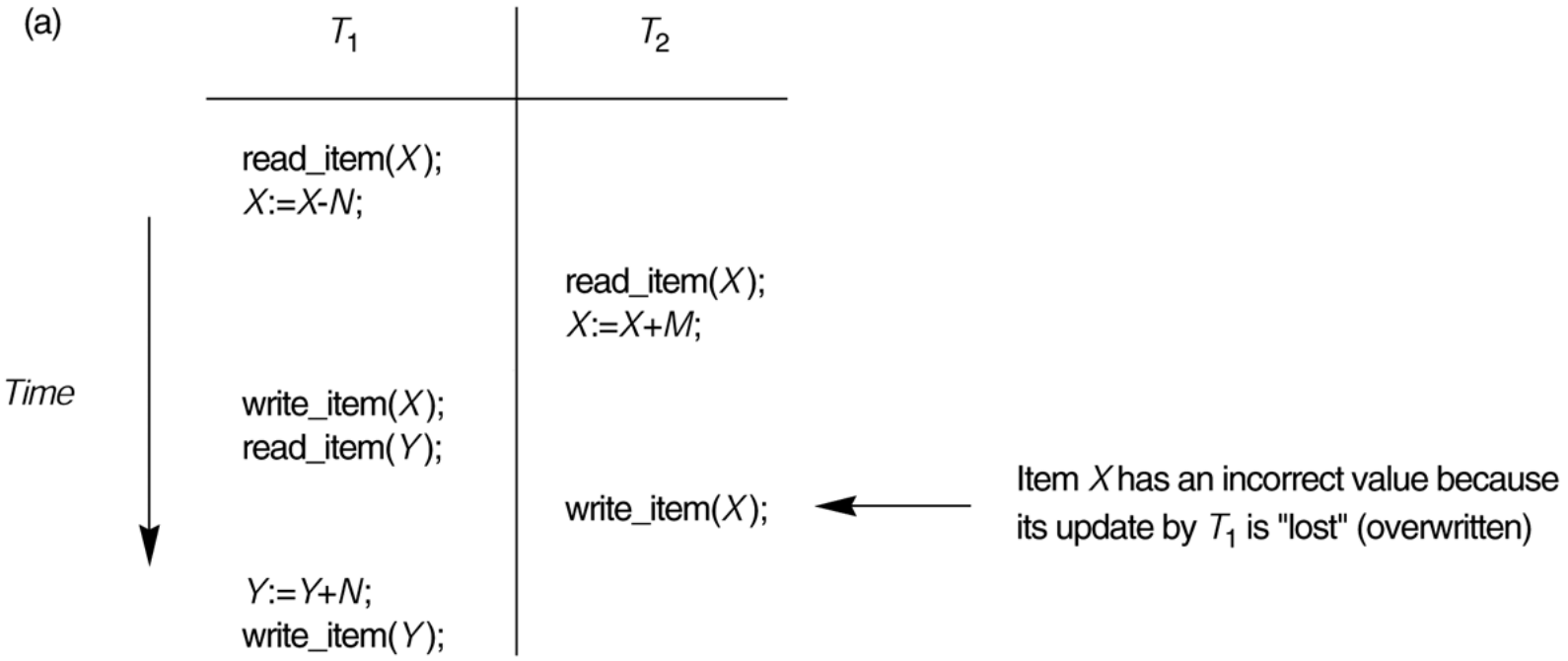
¿Control de concurrencia?

La ejecución concurrente de transacciones de forma no controlada puede dar lugar a los siguientes problemas:

- **Problema de la actualización perdida**
  - Sucede cuando se pierde la actualización hecha por una transacción T1 por la acción de otra transacción T2 sobre el mismo ítem.
- **Problema de la actualización temporal (o lectura sucia)**
  - Sucede cuando una transacción T2 lee un valor de un ítem dejado por otra transacción T1 que no hizo commit antes de que T2 leyera el item.
- **Problema del resumen incorrecto**
  - Una transacción T1 calcula una función agregada de resumen sobre varios registros, mientras otras transacciones actualizan dichos registros: puede que T1 considere algunos registros antes de que se actualicen y otros después de actualizarse

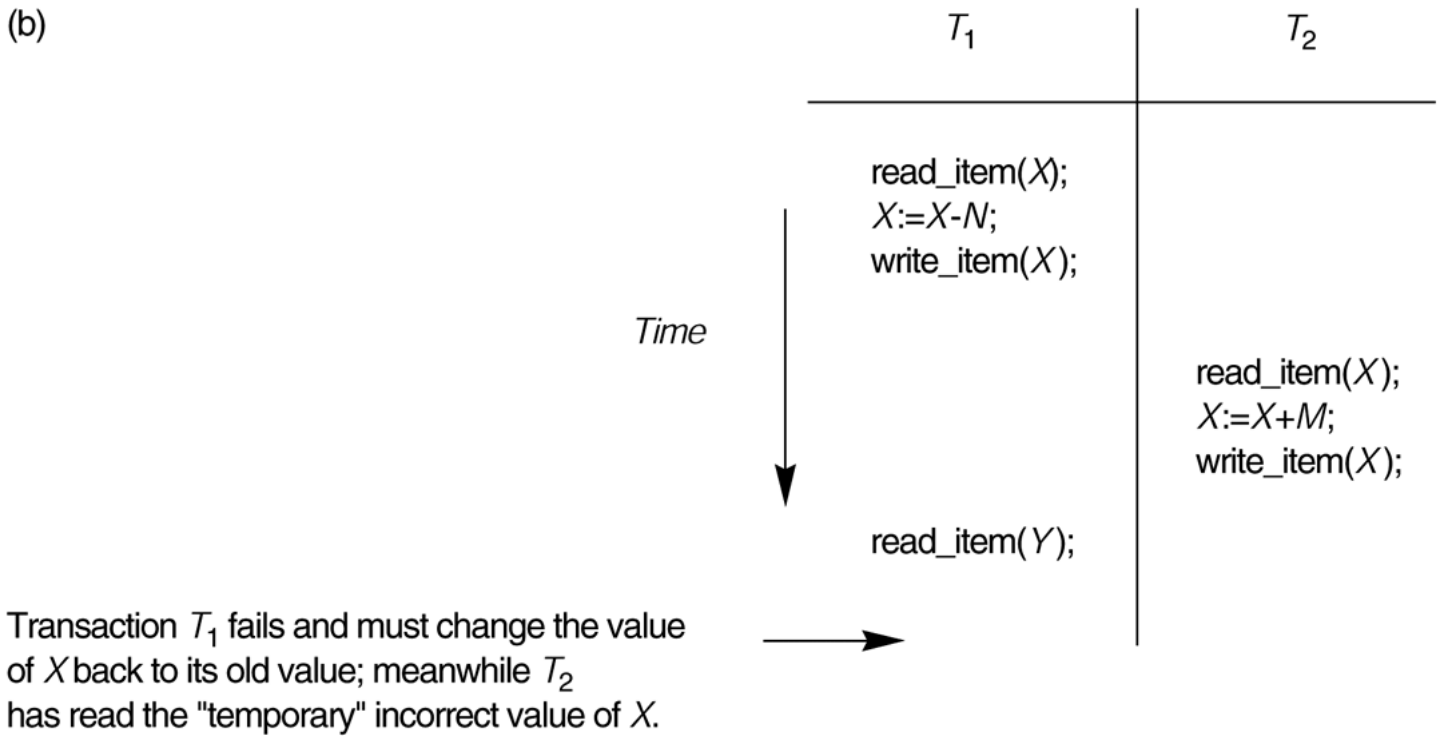
¿Control de concurrencia?

- Problema de la actualización perdida



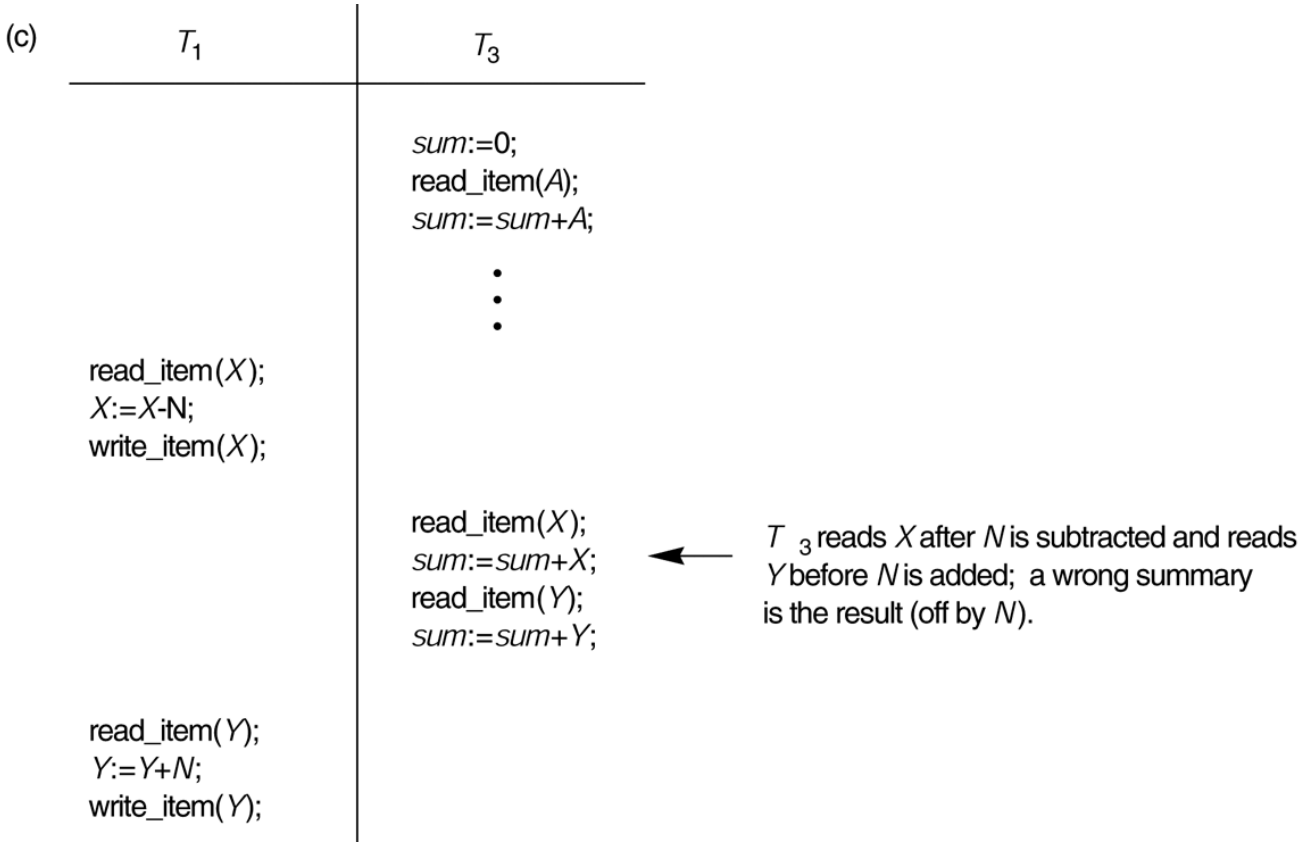
¿Control de concurrencia?

- Problema de la lectura sucia



¿Control de concurrencia?

- Problema del resumen incorrecto



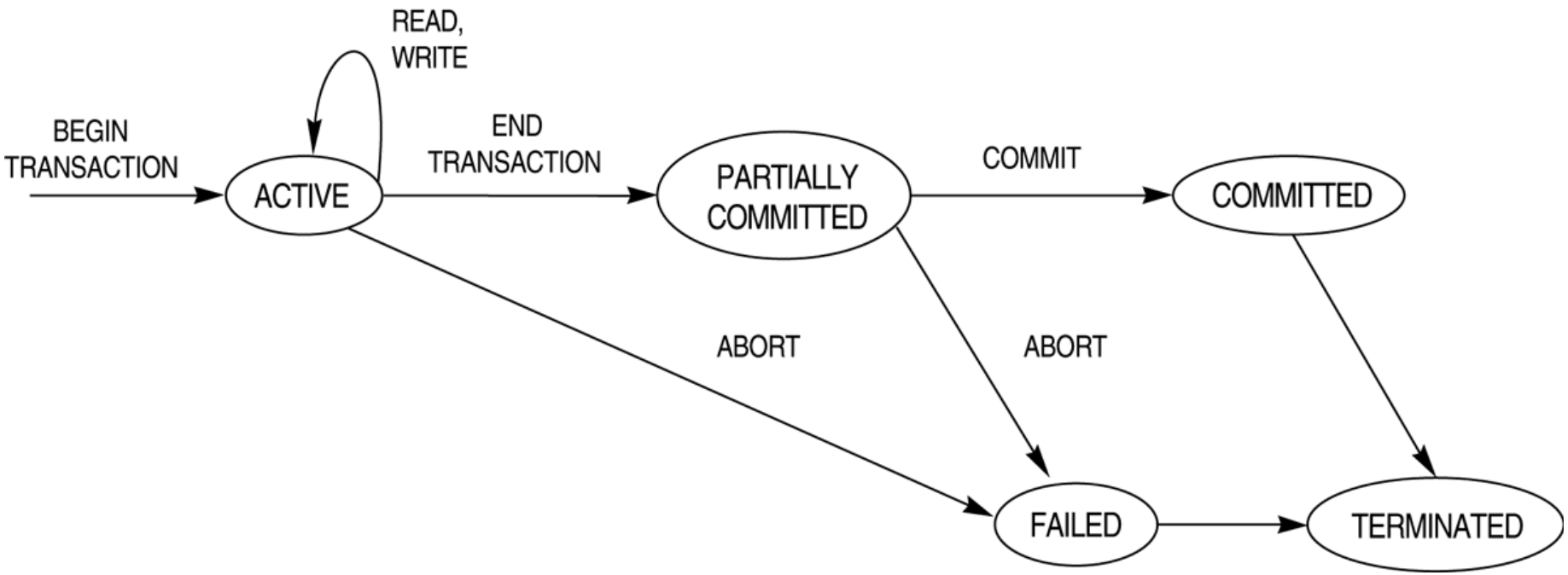
¿Recuperación?

- Tipos de fallos:
  1. Fallo del ordenador (caída del sistema)
  2. Error de la transacción o del sistema
  3. Errores locales o condiciones de excepción detectadas por la transacción
  4. Imposición de control de concurrencia
  5. Fallo del disco
  6. Problemas físicos/catastrofes

Transacciones

- Una transacción es una unidad atómica de trabajo que se realiza por completo o bien no se efectúa en absoluto.
- Estados de una transacción
  - Activa
  - Parcialmente confirmada
  - Confirmada
  - Fallida
  - Terminada

Transacciones - estados



Transacciones - propiedades

- Las propiedades deseables de una transacción son (**ACID**):
  - Atomicidad (**A**tomicity)
    - Las operaciones que componen una transacción deben considerarse como una sola
    - Se debe ejecutar todo (commit) o nada (rollback)
  - Conservación de la consistencia (**C**onsistency)
    - Una operación nunca deberá dejar datos inconsistentes.
    - Se pasa de un estado consistente a otro
  - Aislamiento (**I**solation)
    - No lee resultados intermedios de transacciones no terminadas, No interfiere con otra.
  - Durabilidad o permanencia (**D**urability)
    - En caso de éxito (commit), los cambios son persistentes, incluso si hay fallos de otras

Soporte transacciones en SQL

- Una transacción SQL se considera siempre que es atómica, tanto si termina su ejecución sin errores, como si falla y deja la base de datos sin modificar.
- En SQL no hay un comienzo de transacción explícito.
- Toda transacción debe terminar explícitamente con un COMMIT o un ROLLBACK.

Soporte transacciones en SQL

- Sentencia SET TRANSACTION
  - Modo de acceso:
    - READ ONLY | READ WRITE
  - Nivel de aislamiento ISOLATION LEVEL:
    - READ UNCOMMITTED
    - READ COMMITTED
    - REPEATABLE READ
    - SERIALIZABLE

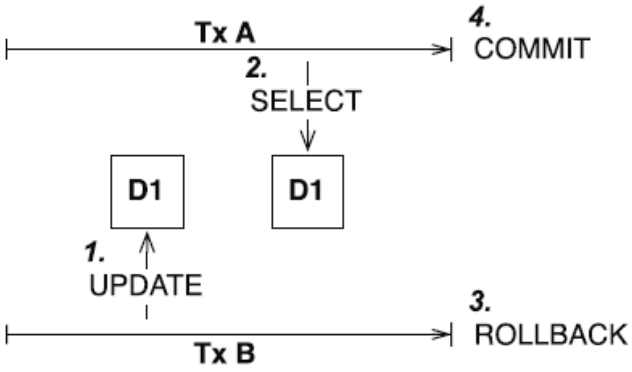
Soporte transacciones en SQL

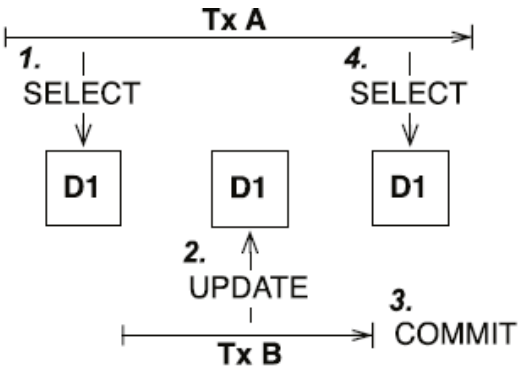
- Nivel de aislamiento ISOLATION LEVEL:
  - **READ UNCOMMITTED**: No afectan los bloqueos producidos por otras conexiones a la lectura de datos.
  - **READ COMMITTED:** Una transacción no podrá ver los cambios de otras conexiones hasta que no hayan sido confirmados o descartados.
  - **REPEATABLE READ:** Garantiza que los datos leídos no podrán ser cambiados por otras transacciones, durante esa transacción.
  - **SERIALIZABLE:** No se permitirá a otras transacciones la inserción, actualización o borrado de datos utilizados por una transacción. Los bloquea mientras dura la misma

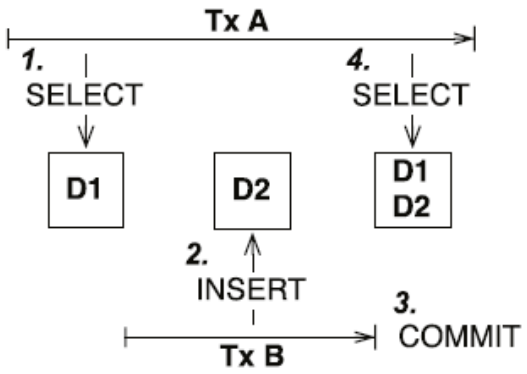
Soporte transacciones en SQL

- Posibles violaciones del aislamiento:
  - **Lectura sucia**: en una transacción se puede leer información no confirmada de otra transacción
  - **Lectura no repetible**: en una transacción se puede leer información que puede ser modificada con lo que si vuelve a leerla será distinta
    - **Second lost update**: Caso especial de unrepeatable read. La actualización de txB es sobreescrita por la de txA
  - **Lectura fantasma**: en una transacción se leen un conjunto de filas de una tabla en la que se puede insertar información nueva, con lo que si se repite la sentencia aparecerán filas que no estaban la primera vez

Soporte transacciones en SQL







- *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)

Soporte transacciones en SQL

Isolation level	**Tipo de Violación**		
Nivel de aislamiento	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	-	-	-

Soporte en Diversos Sistemas

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

SET TRANSACTION ISOLATION LEVEL {READ COMMITTED|SERIALIZABLE}

- PostgreSQL ya tiene todos menos READ UNCOMMITED
- MySQL (InnoDB) soporta todos (la opción por defecto es repeatable read)

SET [SESSION | GLOBAL] TRANSACTION ISOLATION LEVEL {READ UNCOMMITTED | READ COMMITTED | REPEATABLE READ | SERIALIZABLE}

Table 7–4 Read Committed Versus Serializable Transaction

Operation	Read Committed	Serializable
Dirty write	Not Possible	Not Possible
Dirty read	Not Possible	Not Possible
Non-repeatable read	Possible	Not Possible
Phantoms	Possible	Not Possible
Compliant with ANSI/ISO SQL 92	Yes	Yes
Read snapshot time	Statement	Transaction
Transaction set consistency	Statement level	Transaction level
Row-level locking	Yes	Yes
Readers block writers	No	No
Writers block readers	No	No
Different-row writers block writers	No	No
Same-row writers block writers	Yes	Yes
Waits for blocking transaction	Yes	Yes
Subject to "can't serialize access" error	No	Yes
Error after blocking transaction aborts	No	No
Error after blocking transaction commits	No	Yes

Ajuste del nivel de aislamiento

- Si se pone mal nos lleva a problemas:
  - Sólo se van a notar cuando el sistema está en carga
    - Cuando ya estamos en producción
  - Si se pone nivel muy restrictivo se ralentiza mucho el acceso
  - Si se pone muy bajo aparecerán datos inconsistentes difíciles de depurar
- Hay que buscar el equilibrio

¿Cuál escoger?

- Depende de las necesidades, pero:
  - *Read uncommitted* nunca (sólo expertos)
  - *Serializable* no es frecuente
    - ¿Realmente me van a afectar los *phantom reads*? → pocas trx son así
  - La duda mayor está entre *read committed* y *repeatable read*
    - Read commited no protege del *second lost update* y sí puede ser importante
    - Repetible read sí protege frente a second lost update
      - Pero no lo tienen todas las BBDD (Oracle no lo tiene…)

Read commited puede servir…

- Casi todas las BBDD tienen *read committed* por defecto
- Con bloqueos pesimistas se consigue *repetible read*
- Con control optimista se puede evitar el *second lost update*
  - Con tener la BDD en *read committed* por defecto sirve para el 90% si se añaden estos controles a la aplicación

En JDBC

- Por defecto la Connection está en modo AutoCommit. Para cambiarlo se usa el método setAutoCommit(TRUE/FALSE)
- Métodos de control de transacción
  - commit (valida), savepoint (crea un punto de salvaguarda) y rollback (deshace la transacción completa o desde un savepoint)

En JDBC

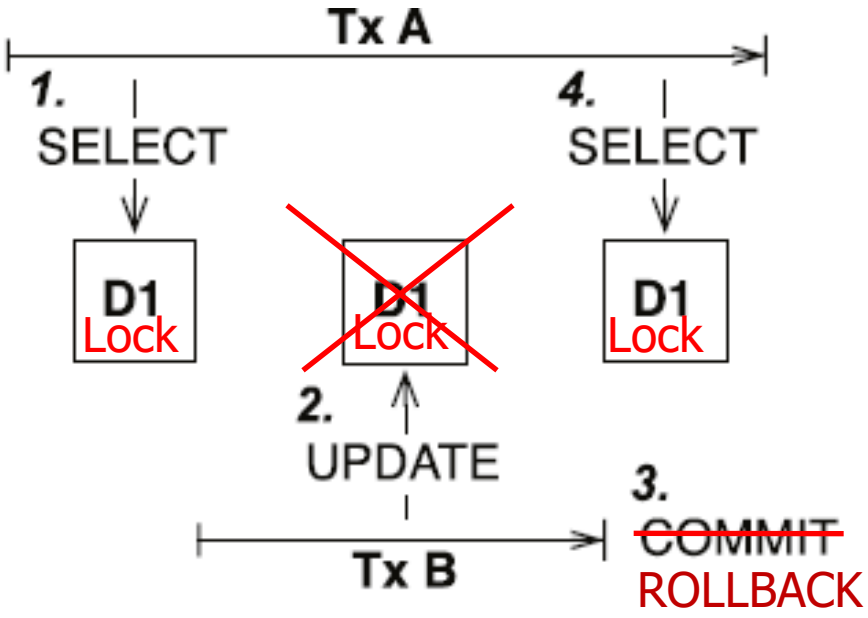
- Métodos para controlar el nivel de aislamiento
  - setTransactionIsolation(nivel de aislamiento)
    - connection.TRANSACTION_READ_UNCOMMITTED
    - connection.TRANSACTION_READ_COMMITTED
    - connection.TRANSACTION_REPEATABLE_READ
    - connection.TRANSACTION_SERIALIZABLE
- Depende de cada SGBD los niveles que se soportan

Control de Concurrencia

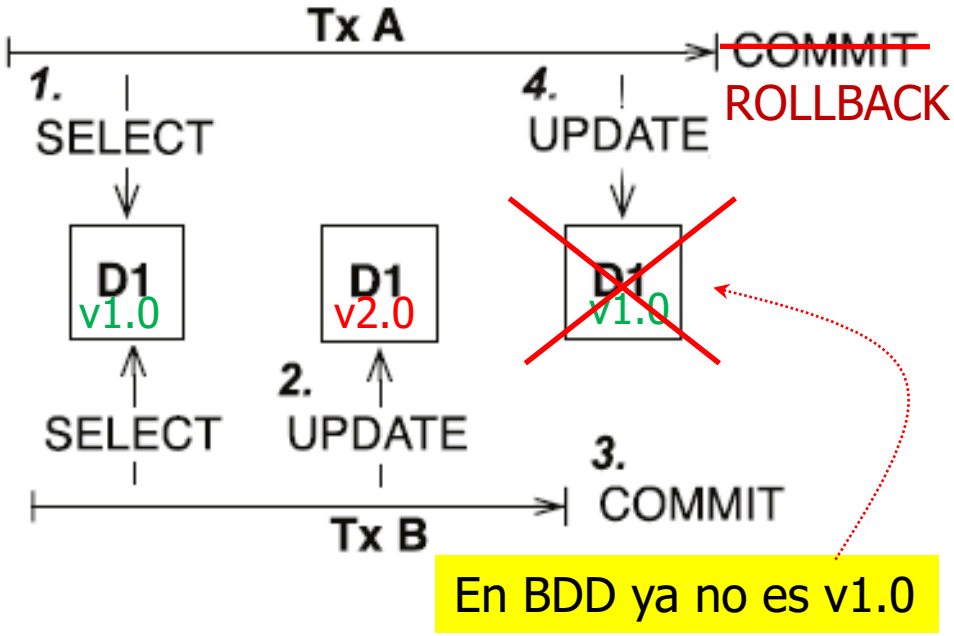
Control de Concurrencia

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

Control pesimista y optimista



PESIMISTA



OPTIMISTA

Control de concurrencia Técnicas de bloqueo (I)

- Técnicas utilizadas en la mayoría de los SGBD comerciales actuales.
- Idea: bloquear ítems de datos para evitar su acceso concurrente desde múltiples transacciones (sincronizar el acceso)
- Bloqueo (lock): Variable asociada a un ítem de datos, que describe su estado con respecto a las operaciones permitidas
- Riesgos: bloqueo mortal (deadlock), inanición (starvation)

Control de concurrencia Técnicas de bloqueo (II)

- Bloqueo binario
  - Muy restrictivo. Una única transacción puede poseer un bloqueo en un momento dado.
  - Operaciones lock() y unlock()
- Bloqueo de modo múltiple
  - Bloqueos compartidos (read locks - Lectura)
  - Bloqueo exclusivos (write locks - escritura).
  - Un elemento bloqueado para **lectura** tiene un ***bloqueo compartido*** por varias transacciones.
  - Un elemento bloqueado para **escritura** tiene un ***bloqueo exclusivo*** (sólo una transacción).
  - Permiten accesos simultáneos en modo lectura

Control de Concurrencia Técnicas de bloqueo (III)

- Conversión de bloqueos
  - **Promover**. Lectura→escritura. Posible si es la única que tiene el bloqueo de lectura en ese momento
  - **Degradar**. Escritura→lectura. Siempre es posible ya que es una única transacción que tiene el bloqueo

Control de Concurrencia Técnicas de bloqueo (IV)

- Bloqueo en dos fases (B2F)
  - **Fase de expansión (o crecimiento):** se pueden adquirir bloqueos pero no se pueden liberar. *Solicitud locks*
  - **Fase de contracción:** se pueden liberar bloqueos pero no se pueden adquirir nuevos. *Realización de unlocks*
  - Garantiza la serializabilidad →evita la necesidad de comprobarla
    - Pero puede limitar la concurrencia (restrictivo) → impide algunos planes serializables

Control de Concurrencia Técnicas de bloqueo (V)

- Variaciones del B2F
  - Básico. No previene el interbloqueo
  - Conservador. La transacción debe bloquear todos los elementos a los que va a acceder antes de comenzar a ejecutarse. Previene el interbloqueo, pero no es práctico
  - Estricto. No se libera ningún bloqueo exclusivo hasta después de terminar la transacción
  - Riguroso. No se libera ningún bloqueo (exclusivo o compartido) hasta después de terminar la transacción

Control de Concurrencia Técnicas de bloqueo (VI)

- Problemas
  - Interbloqueo (o bloqueo mortal). Se produce cuando cada transacción de un conjunto está esperando por algún elemento bloqueado por otra transacción del mismo conjunto.
  - Espera indefinida/Inanición (starvation). Una transacción no puede continuar por tiempo indeterminado, mientras otras pueden continuar normalmente.

Control de Concurrencia Técnicas de bloqueo(VII)

- Solución al interbloqueo
  - Prevención:
    - Bloquear por adelantado todo lo que se va a utilizar
    - Algoritmo de no espera (no-wait)
    - Algoritmo de espera cautelosa
    - Tiempo limitado
    - Ordenación de transacciones por Marcas de tiempo
      - **Esperar-morir:** si T1 < T2, entonces T1 puede esperar; de otro modo T1 aborta y se reinicia mas tarde con la misma marca de tiempo **Transacciones + viejas esperan por otras + jovenes**.
      - **Herir-esperar**: si T1 < T2 entonces se T1 hiere a T2(que aborta y se reinicia posteriormente con la misma marca de tiempo. **Transacciones + jóvenes esperan por otras + viejas**

Control de Concurrencia Técnicas de bloqueo(VII)

- Solución al interbloqueo (II)
  - Detección y recuperación
    - Se detecta mediante la construcción de un grafo de espera (se buscan ciclos)
    - Se recupera mediante la selección de víctimas y abortando la transacción.
      - La transacción que lleva menos tiempo ejecutándose
      - La transacción que está bloqueando un mayor número de transacciones
      - La transacción que menos veces se ha abortado
      - ...
    - Uso de timeouts

Control de Concurrencia Técnicas de bloqueo (VII)

- Espera indefinida (Inanición/Starvation)
  - Puede ocurrir si el esquema de espera es injusto, asignando a unas transacciones más prioridad que a otras.
  - Se puede solucionar con algoritmos justos (por ejemplo el uso de una lista FIFO)

Control de Concurrencia Granularidad del bloqueo

- Se puede bloquear a diversos niveles:
  - Campo
  - Registro
  - Tabla
  - Bloque
  - Fichero
- Granularidad gruesa: menor gestión de bloqueos pero menor concurrencia
- Granularidad fina: mayor gestión de bloqueos pero mayor concurrencia

Control de Concurrencia Oracle

- Oracle mantiene la concurrencia (y garantiza la coherencia de la información) mediante un modelo de consistencia multiversión y varios tipos de bloqueos y transacciones.

Control de Concurrencia Oracle

- Coherencia en la lecturas. Todos los datos leídos provienen del mismo momento temporal aunque otras transacciones terminen durante la ejecución.
  - A nivel de sentencia. Garantizada siempre
  - A nivel de transacción. Se garantiza cuando el isolation level es serializable
- Para garantizar la coherencia Oracle usa rollback segments (hasta la version 8) o undo tablespace (a partir de la 9)

Control de Concurrencia Oracle

- Bloqueos en oracle
  - Dos niveles
    - Exclusivos
    - Compartidos
  - Dos tipos
    - A nivel de fila
    - A nivel de tabla

Control de Concurrencia Oracle

- Tipos de bloqueo
  - A nivel de fila TX
  - A nivel de tabla TM. Puede ser de distintos modos:
    - RS (row share)
    - RX (row exclusive)
    - S (share)
    - SRX (share row exclusive)
    - X (exclusive)

Control de Concurrencia Oracle

SQL Statement	Mode of Table Lock	Lock Modes Permitted? RS	RX	S	SRX	X
SELECT...FROM *table*...	none	Y	Y	Y	Y	Y
INSERT INTO *table* ...	RX	Y	Y	N	N	N
UPDATE *table* ...	RX	Y*	Y*	N	N	N
DELETE FROM *table* ...	RX	Y*	Y*	N	N	N
SELECT ... FROM *table* FOR UPDATE OF ...	RS	Y*	Y*	Y*	Y*	N
LOCK TABLE *table* IN ROW SHARE MODE	RS	Y	Y	Y	Y	N
LOCK TABLE *table* IN ROW EXCLUSIVE MODE	RX	Y	Y	N	N	N
LOCK TABLE *table* IN SHARE MODE	S	Y	N	Y	N	N
LOCK TABLE *table* IN SHARE ROW EXCLUSIVE MODE	SRX	Y	N	N	N	N
LOCK TABLE *table* IN EXCLUSIVE MODE	X	N	N	N	N	N

RS: row share
RX: row exclusive
S: share
SRX: share row exclusive
X: exclusive

*Yes, if no conflicting row locks are held by another transaction. Otherwise, waits occur.

JDBC: Connection

- La interfaz java.sql.Connection
  - Representa sesiones
  - Permite el control de transacciones
    - Commit, Rollback, AutoCommitMode
    - Nivel de aislamiento

Transacciones en JDBC

- Una transacción es un conjunto de sentencias que se deben completar o anular en su totalidad.
- Una transacción se debe validar (commit) o descartar (rollback).
- Las conexiones (Connection) están por defecto en modo autoCommit, es decir, las sentencias se auto-confirman llamando a commit implícitamente cuando finalizan (por lo tanto, cada sentencia es una transacción).

Transacciones

- Para modificar el comportamiento de autocommit se utiliza el método (de Connection)
  - setAutoCommit(true|false)
- Cuando está desactivado es necesario invocar a los métodos commit o rollback para terminar una transacción (y de forma automática comenzar otra)

Transacciones

```
conn.setAutoCommit(false);

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

Transacciones

- Existen puntos de salvaguarda: puntos hasta los que se puede hacer un rollback.
- No tienen que soportarlo todos los drivers
- Para controlarlos están los métodos
  - setSavepoint()/setSavepoint(nombre) que crea un savepoint (con o sin nombre)
  - releaseSavepoint(savepoint): libera un savepoint y los siguientes
  - Rollback(savepoint): realiza un rollback anulando todas las instrucciones desde ese savepoint

Transacciones

```
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();
```

Transaciones - Aislamiento

- Existen 4 niveles de aislamiento. Depende del SGBD los que soporta.
- Se configura a nivel de conexión mediante el método
  - setTransactionIsolation(tipo)
    - Connection.TRANSACTION_READ_UNCOMMITTED
    - Connection.TRANSACTION_READ_COMMITTED
    - Connection.TRANSACTION_REPEATABLE_READ
    - Connection.TRANSACTION_SERIALIZABLE
- Mediante getTransactionIsolation() podemos consultar en qué nivel nos encontramos

Transacciones