TRABAJO PRÁCTICO – Modulo 7

Transacciones y control de concurrencia

Alumno: Ignacio Figueroa – 45.406.120

Tecnicatura Universitaria en Programación – UTN

Materia: Base de Datos

Comisión: 8

Objetivo:

Comprender y aplicar los conceptos de **integridad, consistencia, disponibilidad, transacciones, propiedades ACID y control de concurrencia** en bases de datos.

Parte 1 – Fundamentos de la integridad y consistencia

Integridad de datos

La integridad garantiza que los datos sean correctos, coherentes y válidos en todo momento dentro de una base.

Evita errores, duplicaciones o inconsistencia

Tipos de integridad:

Tipo	Descripción	Ejemplo
Integridad de entidad	Cada registro debe tener un identificador único.	`PRIMARY KEY (id_cliente)`
Integridad referencial	Los datos relacionados deben existir en ambas tablas.	Un cliente no puede tener un `CodigoPostal` que no exista en `localidades`.
Integridad de dominio	Cada campo solo puede contener valores válidos según su tipo o restricción.	`edad INT CHECK (edad >= 18)`

Consistencia

Una base de datos es consistente cuando pasa de un estado valido a otro valido después de una operación.

Ejemplo:

SaldoA + SaldoB = constante

SI algo falla en medio, se deshace y los saldos quedan igual.

Disponibilidad:

La disponibilidad implica que la base pueda ser accedida y usada cuando se la necesita.

Depende de:

- Copias de respaldo (backups)
- Replicación entre servidores
- Alta disponibilidad (HA) y mantenimiento programado.

Parte 2 – Transacciones y Propiedades ACID

Transacción:

Una transacción es una unidad completa de trabajo que debe ejecutarse todo o nada.

Ejemplo: Mover dinero entre cuentas, registrar ventas, etc

Características:

- Agrupa varias operaciones SQL
- Puede confirmarse o revertirse
- Mantiene la coherencia del sistema

Propiedades ACID

Propiedad	Descripción	Ejemplo	Qué pasa si se viola
Atomicidad	La transacción se ejecuta completa o no se ejecuta.	Si una transferencia falla al descontar, también se cancela el depósito.	Se pierde dinero o se duplica.
Consistencia	Los datos pasan de un estado válido a otro.	No se permite saldo negativo si no hay descubierto.	Base incoherente.
Aislamiento	Cada transacción actúa como si	Dos usuarios no pueden modificar	Lecturas sucias o datos corruptos.

	fuera la única.	el mismo registro a la vez.	
Durabilidad	Una vez confirmada, la transacción persiste aunque se apague el servidor.	COMMIT guarda los cambios en disco.	Los datos se pierden ante un corte.

Ejemplo de transacción SQL

START TRANSACTION;

UPDATE cuentas SET saldo = saldo - 500 WHERE id = 1;

UPDATE cuentas SET saldo = saldo + 500 WHERE id = 2;

COMMIT;

- `START TRANSACTION` o `BEGIN` → inicia la transacción.
- `COMMIT` → confirma los cambios.
- `ROLLBACK` → deshace todo si algo falla.

Ejemplo de rollback:

START TRANSACTION;

UPDATE cuentas SET saldo = saldo - 500 WHERE id = 1;

-- Supongamos que falla la siguiente línea

UPDATE cuentas SET saldo = saldo + 500 WHERE id = 999;

ROLLBACK;

Ninguna cuenta se modifica, sino que se mantiene la atomicidad.

Parte 3 – Control de Concurrencia

Concurrencia

Es la **ejecución simultánea de múltiples transacciones** sobre la misma base. En entornos multiusuario, el desafío es evitar conflictos o inconsistencias.

Problemas de concurrencia communes

Problema	Descripción	Ejemplo
Lectura sucia	Se leen datos que aún no fueron confirmados.	Usuario A modifica un saldo y Usuario B lo lee antes del COMMIT.
Lectura no repetible	Una transacción vuelve a leer un dato que cambió.	B consulta un saldo dos veces y A lo modifica entre medio.
Lectura fantasma	Aparecen nuevos registros entre dos lecturas.	Una consulta de facturas muestra diferentes resultados al repetirse.
Actualización perdida	Dos usuarios actualizan el mismo registro y uno pisa el cambio del otro.	

Niveles de aislamiento

Nivel	Problemas evitados	Comentario
READ UNCOMMITTED	Ninguno	Más rápido, menos seguro
READ COMMITTED	Evita lecturas sucias	Nivel por defecto en MySQL
REPEATABLE READ	Evita lecturas sucias y no repetibles	Predeterminado en InnoDB
SERIALIZABLE	Evita todos los problemas	Más seguro pero más lento

Ejemplo para establecer nivel de aislamiento:

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;

START TRANSACTION;

Mecanismos de control de concurrencia

Método	Descripción	Cómo previene conflictos
Bloqueo (Locking)	Impide que dos transacciones modifiquen el mismo dato al mismo tiempo.	Un UPDATE bloquea la fila hasta COMMIT o ROLLBACK.
Ordenamiento por marca de tiempo (Timestamp)	Asigna una marca temporal a cada transacción y las ejecuta en orden.	Evita que las más nuevas sobreescriban las viejas.
Control multiversión (MVCC)	Cada transacción ve su propio “snapshot” de los datos.	Permite lecturas sin bloqueo (usado en PostgreSQL, MySQL InnoDB).
Método	**Descripción**	**Cómo previene conflictos**

Ejemplo de bloqueo:

SELECT * FROM cuentas WHERE id = 1 FOR UPDATE;

Bloqueo mutuo (Deadlock)

Ocurre cuando dos transacciones esperan recursos que la otra tiene bloqueados.

Ejemplo:

- A bloquea cuenta1 y espera cuenta2
- B bloquea cuenta 2 y espera cuenta1

Solución: detección automática y **rollback** de una transacción.

Parte 4 – Aplicaciones Reales y Preguntas de Investigación

Aplicaciones del mundo real

Área	Ejemplo de uso de transacciones
Bancos	Transferencias y pagos deben ser atómicos.
E-commerce	Confirmación de pedido, actualización de stock y cobro.
Sistemas de reservas	Evitar doble reserva del mismo asiento.

Pregunta 1 – Estrategias de concurrencia

Concurrencia pesimista: bloquea registros antes de operar.
Útil en sistemas con muchos conflictos (banca, contabilidad).

Concurrencia optimista: permite operar y valida al confirmar.
Ideal en sistemas con pocas colisiones (consultas, reportes).

Bloqueos de intención: avisan a la base qué tipo de bloqueo se va a aplicar, evitando deadlocks.

Pregunta 2 – Implementación ACID en MySQL

MySQL (InnoDB):

- Cumple con las propiedades ACID
- Niveles de aislamiento: READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE.
- Predeterminado: REPETEABLE READ.
- Usa MVCC y bloqueos de fila de aislamiento.

Conclusión

En esta unidad se comprendieron los principios que garantizan la **integridad, consistencia y confiabilidad** de los datos mediante transacciones y control de concurrencia, aplicando los conceptos ACID y los mecanismos utilizados por MySQL e InnoDB.