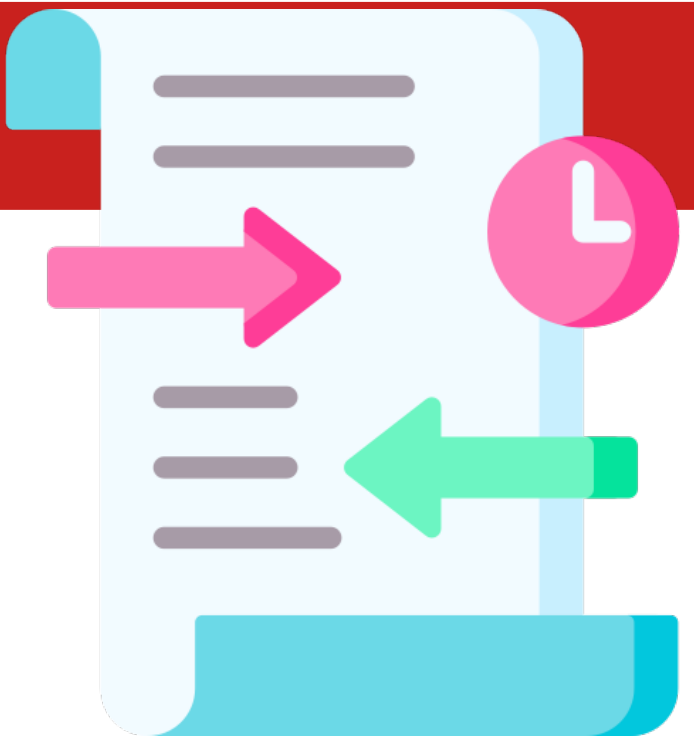
Bases de Datos I
Unidad VIII



Transacciones

https://www.flaticon.com/free-icons/relations

Introducción

- **En los sistemas de bases de datos se ejecutan operaciones habituales.**

- **Ejemplos de ello son:**
  - **Ordenes de venta**
  - **Reservas de pasajes de avión**
  - **Registros de empleados**
  - **Transferencias de dinero**
  - **Ventas con actualizaciones de stock**

Introducción

- **Muchas de las operaciones antes mencionadas, requieren ejecutar más de una sentencia SQL/DML para poder llevarlas a cabo.**

- **Veamos un pequeño ejemplo en el que para realizar una operación de transferencia bancaria, se requieren más de una sentencia SQL/DML.**

Introducción

- **Imaginemos siguiente tabla:**

```
CREATE TABLE cuenta(
  numero BIGINT PRIMARY KEY,
  tipo INTEGER REFERENCES tipo_cuenta,
  cliente INTEGER REFERENCES cliente,
  sucursal INTEGER REFERENCES sucursal,
  saldo NUMERIC(14,2));
```

- **Y la operación que debemos realizar, es una transferencia entre cuentas.**

Y la operación de transferencia...

```
PROCEDURE transferencia(origen BIGINT, destino BIGINT,
                                    monto NUMERIC(14,2))  AS
DECLARE
saldo NUMERIC(14,2);
BEGIN
  saldo = (SELECT saldo FROM cuenta WHERE numero = origen);
  IF(saldo >= monto)THEN
    BEGIN
    UPDATE cuenta SET saldo = saldo - monto WHERE numero = origen;
    UPDATE cuenta SET saldo = saldo + monto WHERE numero = destino;
    END
  ELSE
    EXCEPTION 'El saldo (' || saldo || ') es insuficiente para transferir ' || monto;
END;
```

Introducción

- **¿Que pasaría si después de descontar el monto a transferir de la cuenta origen y antes de incrementar el saldo de la cuenta destino, se produce algún fallo o error?**
- **¿En qué estado quedaría nuestra base de datos?**
- **Necesitamos alguna forma de agrupar las sentencias para llevar adelante la operación, y que actúen como una unidad.**

Introducción

- **¿Que pasaría si después de consultar el saldo de la cuenta origen y verificar que sea mayor o igual al monto de la transferencia, pero antes de descontarlo, otra operación lo modifica?**
- **¿En que estado quedaría nuestra base de datos?**
- **Necesitamos alguna forma de asegurarnos que una operación, no interfiera con otras operaciones.**

Transacción

- **¿Qué es una transacción?**

  Se llama transacción a una colección de operaciones de acceso a la base de datos, que forman una única unidad lógica de trabajo.

- **Las operaciones pueden ser consultas, inserciones, modificaciones o eliminaciones (SQL/DML).**

Transacciones

- **Un SGBD debe asegurar que la ejecución de las transacciones se realice adecuadamente a pesar de la existencia de fallos.**
- **Solo puede tener dos finales posibles:**
  1) **Se ejecuta la transacción completa.**
  2) **No se ejecuta nada en absoluto.**

Propiedades ACID

- **Para asegurar la integridad de los datos se necesita que el sistema de base de datos mantenga las siguientes propiedades de las transacciones, conocidas por el acrónimo ACID:**
  - **Atomicity (Atomicidad)**
  - **Consistency (Consistencia)**
  - **Isolation (Aislamiento)**
  - **Durability (Durabilidad)**

Atomicidad (ACID → Atomicity**)**

- **Una transacción se ejecuta exactamente una vez y tiene carácter "atómico" es decir, el trabajo se realiza en su totalidad o no se realiza en ningún caso.**

- **La responsabilidad de asegurar la atomicidad de las transacciones es del gestor de base de datos, con su componente de gestión de transacciones.**

Consistencia (ACID → Consistency**)**

- **La “consistencia” requiere que los datos involucrados en una transacción se mantengan en términos de semántica.**
- **Una transacción toma una base de datos consistente, hace operaciones sobre ella y debe dejarla en un estado consistente.**
- **La responsabilidad de asegurar la consistencia es del programador, con la asistencia del gestor de base de datos (restricciones).**

• **Aislamiento (**ACID → Isolation**)**

- **Una transacción es una unidad de aislamiento, permitiendo que transacciones concurrentes se comporten como si cada una fuera la única transacción que se ejecuta en el sistema.**
- **Las responsabilidad de asegurar el aislamiento es del gestor de base de datos, con su componente de control de concurrencia.**

Durabilidad (ACID → Durability)

- **Tras la finalización con éxito de una transacción, los cambios realizados en la base de datos, quedan almacenados de forma permanente, incluso si hay fallos en el sistema.**
- **Las responsabilidad de asegurar la durabilidad es del gestor de base de datos, con su componente de persistencia a disco y su componente de gestión de recuperaciones.**

Transacciones en SQL (SQL/TCL)

- **El lenguaje SQL proporciona algunas sentencias para el control de las transacciones:**

  START TRANSACCION

  COMMIT

  ROLLBACK

- **También se permite el anidamiento de transacciones o sub-transacciones:**

  SAVEPOINT

  RELEASE

  ROLLBACK

START TRANSACTION

- **Sintaxis:**

  START TRANSACTION [READ WRITE | READ ONLY];

- **La sentencia permite indicar al SGBD que se inicia una nueva transacción. Las sentencias que se ejecuten a continuación, van a formar parte de la misma unidad lógica.**
- **Los modificadores permiten indicar si la transacción es de lectura escritura o sólo lectura. Por defecto, lectura escritura.**

SET TRANSACTION

- **Sintaxis:**

  SET TRANSACTION [READ WRITE | READ ONLY];

- **La sentencia permite modificar o establecer las características de la transacción.**

- **Debe ser la primera sentencia que se ejecute después del inicio de la transacción, ya que no se puede modificar sus características a mitad de la misma.**

COMMIT

- **Sintaxis:**

  COMMIT [WORK];

- **Termina la transacción en curso, indicando que las modificaciones realizadas debe ser comprometidas.**
- **Una vez se ejecuta la sentencia COMMIT, se asegura la durabilidad de las modificaciones, y las mismas no pueden perderse o ser deshechas.**

ROLLBACK

- **Sintaxis:**

  ROLLBACK [WORK];

- **Termina la transacción en curso, indicando que las modificaciones realizadas debe ser deshechas.**
- **Una vez se ejecuta la sentencia ROLLBACK, se asegura la no durabilidad de las modificaciones realizadas por la transacción.**

Ejemplo transacciones

```
PROCEDURE transferencia(origen BIGINT, destino BIGINT, monto NUMERIC(14,2))  AS
DECLARE
saldo NUMERIC(14,2);
BEGIN
  START TRANSACTION READ WRITE;
  saldo = (SELECT saldo FROM cuenta WHERE numero = origen);
  IF(saldo >= monto)THEN
    BEGIN
    UPDATE cuenta SET saldo = saldo - monto WHERE numero = origen;
    UPDATE cuenta SET saldo = saldo + monto WHERE numero = destino;
    COMMIT;
    END
  ELSE
   BEGIN
    ROLLBACK;
    EXCEPTION 'El saldo (' || saldo || ') es insuficiente para transferir ' || monto;
    END;
END;
```

SAVEPOINT

- **Sintaxis:**

  SAVEPOINT nombre;

- **Establece un nuevo punto de rescate dentro de la transacción actual.**
- **Un punto de rescate es una marca especial dentro de una transacción que permite que todas las sentencias que se ejecutan después de que se estableció, pueden revertirse, restaurando el estado de la transacción a lo que era en el momento del punto de rescate.**

RELEASE

- **Sintaxis:**

  RELEASE [SAVEPOINT] nombre;

- **Destruye un punto de rescate previamente definido, en la transacción actual.**
- **La destrucción de un punto de rescate hace que no esté disponible como un punto de reversión, no deshace las modificaciones de las sentencias ejecutadas después del establecimiento del punto de rescate.**

ROLLBACK

- **Sintaxis:**

  ROLLBACK [WORK] TO [SAVEPOINT] nombre;

- **Hacer retroceder todas las modificaciones que se han ejecutado después del establecimiento del punto de rescate.**
- **Implícitamente destruye todos los puntos de rescate que se establecieron después de que el punto nombrado.**

Aislamiento

- **El SGBD debe asegurarse del aislamiento y la conservación de la consistencia:**
  - **Los efectos de una transacción no deben verse influenciados por otras transacciones concurrentes.**
  - **La ejecución completa (y confirmada) de una transacción, aún en concurrencia con otras, debe asegurar el mantenimiento de la consistencia de la BD.**
  - **Una transacción debe ejecutarse como si fuera la única transacción que estuviera ejecutándose en la BD en ese preciso momento.**

Posibles efectos no deseados

- **EL estándar define tres posibles efectos no deseados de lectura:**
  - **Lectura sucia**
  - **Lectura no repetible**
  - **Lectura fantasma**

Lectura sucia

- **Dadas dos transacciones concurrentes T1 y T2.**

- **Se produce una lectura sucia cuando:**
  - **Una transacción T1 puede leer la actualización de T2 que todavía no ha confirmado.**
  - **Si T2 aborta, T1 habría leído un dato incorrecto.**

Ejemplo: Lectura sucia

T1	T2
START TRANSACTION;	START TRANSACTION;
	UPDATE cuenta SET saldo = saldo + 10 WHERE numero = 1;
SELECT saldo FROM cuenta WHERE numero = 1;	
	ROLLBACK;
...	

Lectura no repetible

- **Dadas dos transacciones concurrentes T1 y T2.**

- **Si una transacción T1 consulta dos veces un mismo dato, y en medio una transacción T2 lo modifica y compromete, T1 verá valores diferentes para el dato en cada consulta.**

Ejemplo: Lectura no repetible

T1	T2
START TRANSACTION;	START TRANSACTION;
SELECT saldo FROM cuenta WHERE numero = 1;	
	UPDATE cuenta SET saldo = saldo + 10 WHERE numero = 1;
	COMMIT;
SELECT saldo FROM cuenta WHERE numero = 1;	
...	

Lectura fantasma

- **Dadas dos transacciones concurrentes T1 y T2.**

- **Una transacción T1 puede leer un conjunto de filas (que cumplan una condición).**
- **Si una transacción T2 inserta una fila que también cumple la condición, y T1 repite la lectura verá una fila que previamente no existía (una fila fantasma).**

Ejemplo: Lectura fantasma

T1	T2
START TRANSACTION;	START TRANSACTION;
SELECT saldo FROM cuenta WHERE saldo BETWEEN $100 AND $200;	
	INSERT INTO cuenta VALUES(1000, …, $155);
	COMMIT;
SELECT saldo FROM cuenta WHERE saldo BETWEEN $100 AND $200;	
...	

Niveles de aislamiento

- **El estándar de SQL define 4 diferentes niveles de aislamiento:**
  - **READ UNCOMMITTED (Lectura no comprometida)**
  - **READ COMMITTED (Lectura comprometida)**
  - **REPEATABLE READ (Lectura repetible)**
  - **SERIALIZABLE (Serializable)**

READ UNCOMMITTED

- **Este nivel de aislamiento, no provee ningún tipo de aislamiento.**
- **En el pueden producirse:**
  - **Lecturas sucias**
  - **Lecturas no repetibles**
  - **Lecturas fantasmas**
- **Una transacción con este nivel de aislamiento, es interferida por todas las transacciones concurrentes, hayan comprometido o no.**

READ COMMITTED

- **Este nivel de aislamiento, solo puede consultarse aquello que esté comprometido.**
- **En el pueden producirse:**
  - **Lecturas no repetibles**
  - **Lecturas fantasmas**
- **Una transacción con este nivel de aislamiento, es interferida por todas las transacciones concurrentes, que hayan comprometido.**

REPEATABLE READ

- **Este nivel de aislamiento, solo puede consultarse aquello que esté comprometido, y si se repite la consulta se obtiene lo mismo.**
- **En el pueden producirse:**
  - **Lecturas fantasmas**
- **Una transacción con este nivel de aislamiento, es interferida por todas las transacciones concurrentes, que inserten y comprometan.**

SERIALIZABLE

- **Es el mayor nivel de aislamiento.**
- **Una transacción No puede realizar actualizaciones sobre datos que otra transacción ha consultado (las distintas transacciones se ejecutan como si fueran una tras otra sin afectarse entre si) .**
- **En este nivel de aislamiento no se produce ninguno de los efectos no deseados de lectura.**

Niveles de aislamiento en SQL/TCL

- **Para especificar el nivel de aislamiento, debemos hacerlo al iniciar la transacción, con las sentencias:**

START TRANSACTION

SET TRANSACTION

SET TRANSACTION

- **Sintaxis:**
  - **SET TRANSACTION ISOLATION LEVEL [SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED] [READ WRITE | READ ONLY];**

- **Ya hablamos sobre esta sentencia, ahora simplemente mostramos que también permite establecer el nivel de aislamiento.**

Ejemplo transacciones

```
PROCEDURE transferencia(origen BIGINT, destino BIGINT, monto NUMERIC(14,2))  AS
DECLARE
saldo NUMERIC(14,2);
BEGIN
  START TRANSACTION ISOLATION LEVEL SERIALIZABLE READ WRITE;
  saldo = (SELECT saldo FROM cuenta WHERE numero = origen);
  IF(saldo >= monto)THEN
    BEGIN
    UPDATE cuenta SET saldo = saldo - monto WHERE numero = origen;
    UPDATE cuenta SET saldo = saldo + monto WHERE numero = destino;
    COMMIT;
    END
  ELSE
    BEGIN
    ROLLBACK;
    EXCEPTION 'El saldo (' || saldo || ') es insuficiente para transferir ' || monto;
    END;
END;
```

Aclaración final

- **¿Todos los SGBD implementan todos los niveles de aislamiento?**
  - **La respuesta es NO. Aunque la mayoría si lo hace.**
- **¿Que pasa si establecemos un nivel de aislamiento que el SGBD no tiene implementado?**
  - **Se establece el inmediato superior implementado**
- **Por eso el único obligatorio es SERIALIZABLE.**

PostgreSQL

- **Solo implementa los niveles de aislamiento:**
  - **READ COMMITTED**
  - **REPEATABLE READ**
  - **SERIALIZABLE**

- **En el nivel de lectura repetible, no pueden producirse lecturas fantasmas, pero las modificaciones producidas por ella misma la afectan.**

Bibliografía

- **SQL-99 Complete, Really. 1999. Peter Gulutzan y Trudy Pelzer.**
- **Elmasri R. y Navathe Sh. (2000). Sistemas de Bases de Datos. Conceptos Fundamentales. Segunda Edición. Addison Wesley Longman de México, S. A.**