­­­

“Transacciones”

“Taller de Base Datos”

**Ingeniería en Sistemas Computacionales.**

PRESENTA:

Guízar Morales Jonathan

**Professor: Leonardo Martinez Gonzales**

JIQUILPAN, MICHOACÁN, NOVIEMBRE 2024

Contenido

[1.- ¿Qué es una transacción en bases de datos? 3](#_Toc182327447)

[2.- ¿Cómo puedo aplicar transacciones en mi proyecto? 3](#_Toc182327448)

[3.- ¿Cuáles son los tipos de bloqueos en bases de datos? 3](#_Toc182327449)

[4.- ¿Niveles de atracción y cómo funcionan? 4](#_Toc182327450)

# 1.- ¿Qué es una transacción en bases de datos?

Una transacción es una unidad indivisible de operaciones que se ejecutan en una base de datos. Es un conjunto de instrucciones (como consultas, inserciones, o actualizaciones) que se agrupan para realizarse en conjunto. Para que una transacción sea exitosa, todas las operaciones incluidas deben completarse correctamente; si alguna falla, todas deben revertirse, manteniendo la integridad de los datos. Las transacciones deben cumplir con las propiedades ACID (Atomicidad, Consistencia, Aislamiento, Durabilidad) para garantizar que la base de datos mantenga un estado coherente.

# 2.- ¿Cómo puedo aplicar transacciones en mi proyecto?

En el proyecto, implementé transacciones para asegurar la integridad de los datos en operaciones críticas. Utilicé transacciones SQL para agrupar varias operaciones de actualización, de forma que, si alguna de ellas fallaba, todas se revertían automáticamente para mantener la base de datos consistente. Empecé cada transacción con **BEGIN TRANSACTION**, ejecuté las operaciones necesarias, y usé **COMMIT** para confirmar los cambios si todas las operaciones se realizaban correctamente. En caso de error, utilicé **ROLLBACK** para deshacer los cambios y evitar datos incompletos o inconsistentes. Esto garantizó que todas las transacciones fueran atómicas, consistentes y seguras.

# 3.- ¿Cuáles son los tipos de bloqueos en bases de datos?

Los bloqueos en bases de datos ayudan a controlar el acceso concurrente a los datos, asegurando que las transacciones se ejecuten sin conflictos. Existen varios tipos de bloqueos, según el nivel de precisión y la intención de uso:

**S (Compartido)**: Permite que otros usuarios accedan al recurso para realizar lecturas, pero no para modificarlo.

**Sch-S (Compartido de esquema)**: Permite bloqueos compartidos en el esquema de la base de datos.

**IS (Intención Compartida)**: Señala la intención de establecer un bloqueo compartido en una parte de un recurso más grande.

**Bloqueos exclusivos**:

**X (Exclusivo)**: Evita que otros usuarios accedan al recurso mientras se realiza una operación de escritura o modificación.

**IX (Intención Exclusiva)**: Indica la intención de establecer bloqueos exclusivos en partes del recurso.

Los bloqueos pueden aplicarse en diferentes niveles:

**BD** (Base de Datos): Bloquea toda la base de datos.

**TAB** (Tabla): Bloquea una tabla completa.

**PAG** (Página de datos): Bloquea una página de datos específica.

**KEY** (Fila específica): Bloquea una fila específica que está siendo modificada.

Estos bloqueos permiten que múltiples transacciones coexistan sin interferir unas con otras, manteniendo la integridad y consistencia de los datos en sistemas multiusuario.

# 4.- ¿Niveles de atracción y cómo funcionan?

**READ COMMITTED**: Establece bloqueos compartidos solo durante cada lectura, permitiendo que otras transacciones realicen cambios entre lecturas. Esto puede causar lecturas no repetibles (mismos datos leídos de manera diferente en una misma transacción).

**READ UNCOMMITTED**: No usa bloqueos para lectura, lo cual permite lecturas sucias (lecturas de datos que pueden estar en proceso de cambio). Prioriza rendimiento sobre consistencia.

**REPEATABLE READ**: Bloquea los datos leídos hasta el final de la transacción, evitando lecturas no repetibles, pero no evita lecturas fantasmas (nuevos datos agregados por otras transacciones).

**SERIALIZABLE**: Garantiza lecturas consistentes bloqueando rangos completos de datos, evitando lecturas sucias, no repetibles y fantasmas. Es el nivel más seguro, pero también el menos eficiente en multiusuario.

**SNAPSHOT**: Utiliza versiones de filas en vez de bloqueos, permitiendo que una transacción acceda a una versión consistente de los datos al momento de iniciar, evitando lecturas sucias, no repetibles y fantasmas.

**READ COMMITTED SNAPSHOT**: Combina READ COMMITTED y SNAPSHOT, usando versiones de filas. Evita bloqueos de lectura y escritura, pero permite lecturas no repetibles y fantasmas.