# ¿Que son las transacciones?

Es una secuencia de una o mas operaciones de base de datos que se ejecutan como una unidad indivisible, también conocido como atomicidad

Garantiza que las consultas se ejecuten de manera consistente es decir que cumplan todas las operaciones o ninguna de ellas

Revertir

Ejecución de operaciones

Inicio de transacción

Parcialmente Incompleto

Involucrar (Fin de transacción)

Para realizar un proceso en una base de datos, se inicializa una transacción desde la aplicación. Durante el proceso, se ejecutan varias operaciones, como cargar datos del usuario (nombre, contraseña o características específicas). En ocasiones, pueden ocurrir errores, como una contraseña incorrecta o no coincidente; Si esto sucede, la transacción debe revertirse. Si no se detecta ningún error, la transacción se considerará incompleta hasta que todas las operaciones necesarias se verifiquen correctamente. Incluso después de la verificación, la transacción puede cancelarse y revertirse. Si todo es exitoso, la transacción finaliza correctamente.

# ¿Cómo aplico las transacciones a mi proyecto?

En SQL, las transacciones son bloques de instrucciones que se ejecutan como una unidad, de modo que todas las operaciones dentro de la transacción deben completarse exitosamente para que los cambios se confirmen en la base de datos. Si alguna operación falla, la transacción puede revertirse para evitar inconsistencias.

* Iniciar la transacción: usando `BEGIN TRANSACTION` o `START TRANSACTION`
* Ejecutar las operaciones (INSERT, UPDATE, DELETE, etc.) que deseas agrupar en la transacción.
* Confirmar la transacción con `COMMIT`
* Revertir la transacción con `ROLLBACK` en caso de error

En bases de datos, los bloqueos son mecanismos que controlan el acceso concurrente a los recursos, como tablas o filas, para mantener la integridad de los datos.

### 1. **Bloqueo compartido**

* Permite que múltiples transacciones lean el mismo recurso de recurso
* No permite que ninguna transacción realice cambios en el recurso hasta que todos los bloqueos concurrentes
* Es útil para consultas de solo lectura, evitando conflictos con otras lecturas.

2. **Bloqueo exclusivo**

* Permite que solo una transacción tenga acceso de escritura al recurso, bloqueando cualquier otra transacción de lectura o escritura hasta que se librere del bloqueo
* Se usa cuando

### 3. **Bloqueo de actualización (Update Lock, U)**

* Este tipo de bloqueo se utiliza para evitar un "deadlock" bloqueo mutuo.
* Permite a una transacción leer datos con la intención de actualizarlos más adelante. Si se necesita una actualización, el bloqueo de actualización se convierte en un bloqueo exclusivo.
* Solo permite un bloqueo de actualización a la vez, para que otra transacción no intente actualizar el mismo recurso simultáneamente.

### 4. **Bloqueo intencional**

* Los bloqueos intencionales indican que una transacción tiene la intención de colocar un bloqueo más específico en una parte del recurso, como una fila dentro de una tabla.
* Hay varios tipos de bloqueos intencionales:
  + **Intención compartida (IS)**
  + **Intent Exclusive (IX**
  + **Shared Intent Exclusive (SIX)**

5. **Bloqueo de fila o registro (Row Lock)**

* Afecta una sola fila dentro de una tabla, permitiendo un mayor control y eficiencia al no bloquear toda la tabla.
* Este bloqueo es muy útil en bases de datos de alta concurrencia.

### 6. **Bloqueo de tabla**

* Bloquea toda la tabla en lugar de filas específicas, y puede ser de tipo compartido o exclusivo.
* Es menos común en sistemas transaccionales de alta concurrencia y suele usarse en operaciones de mantenimiento o procesos de carga masiva.

### 7. **Bloqueo de página**

* Afecta una "página" de la base de datos, que contiene varias filas.
* Este bloqueo es menos granular que un bloqueo de fila, pero puede ser más eficiente en algunos casos.

### 8. **Bloqueo de nivel de esquema (Schema Lock)**

* Se aplica a la estructura o esquema de la base de datos y tiene dos tipos:
  + **Bloqueo de modificación de esquema (Sch-M)**
  + **Schema Stability Lock (Sch-S)**

### 9. **Bloqueo de rango de clave**

* Se aplica en sistemas que usan índices y es útil para operaciones que necesitan mantener el orden de las filas.
* Ayuda a evitar inconsistencias en rangos de datos ingresados.

# ¿Niveles de aislamiento y cómo funcionan?

Los niveles de aislamiento en bases de datos definen el grado de visibilidad que tienen las transacciones sobre los cambios realizados por otras transacciones concurrentes. Cuanto mayor es el nivel de aislamiento, menor es la probabilidad de enfrentar problemas de concurrencia, pero también puede reducir el rendimiento.

* **Read Uncommitted (Lectura no confirmada):**

Es el nivel más bajo de aislamiento. Permite que una transacción lea datos que aún no han sido confirmados por otras transacciones, lo que se conoce como "lectura sucia" (dirty read).

* **Read Committed (Lectura confirmada):**

Permite que una transacción lea solo los datos que ya han sido confirmados por otras transacciones, evitando lecturas sucias.

* **Repeatable Read (Lectura repetible):**

Garantiza que, si una transacción lee un dato, ese dato no podrá ser modificado por otras transacciones hasta que la transacción original finalice, evitando así las lecturas no repetibles.

* **Serializable:**

Es el nivel más alto de aislamiento y asegura que las transacciones se ejecuten como si fueran en serie (una tras otra), lo que evita cualquier tipo de problema de concurrencia (lecturas sucias, lecturas no repetibles y fantasmas).