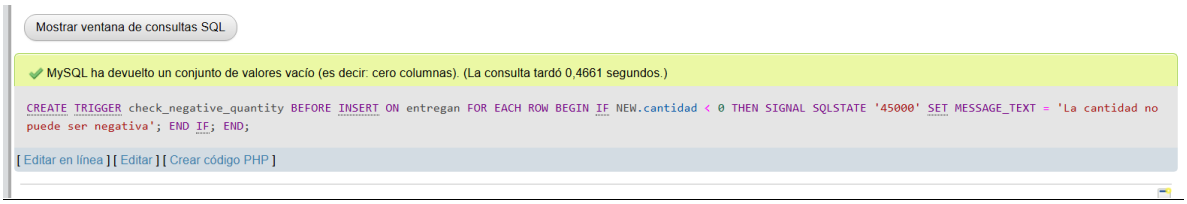
Laboratorio 28 - Triggers

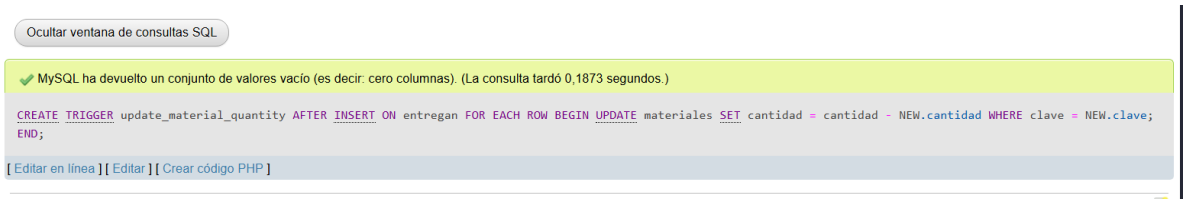
1.-**Descripción:** Este trigger asegura que las cantidades de material entregadas no puedan ser negativas. Si un usuario intenta insertar una fila con una cantidad negativa, la transacción fallará y se lanzará un mensaje de error personalizado.

```
CREATE TRIGGER check_negative_quantity

BEFORE INSERT ON entregan

FOR EACH ROW

BEGIN

 IF NEW.cantidad < 0 THEN

  SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'La cantidad no puede ser negativa';

 END IF;

END$$

COMMIT;
```



2.- **Descripción: Descripción:** Este trigger asegura que la cantidad de material disponible en la tabla materiales se reduzca cada vez que se registre una nueva entrega en la tabla entregan. Se toma la clave del material entregado (de la tabla entregan) y se resta la cantidad entregada de la tabla materiales.

```
CREATE TRIGGER update_material_quantity

AFTER INSERT ON entregan

FOR EACH ROW

BEGIN

 UPDATE materiales

 SET cantidad = cantidad - NEW.cantidad

 WHERE clave = NEW.clave;

END$$
```



Pregunta 1: ¿Qué utilidad tiene un trigger (ventajas)?

Los triggers ofrecen varias ventajas:

1. **Automatización de procesos**: Permiten automatizar tareas, como actualizaciones de registros, sin la intervención directa de la aplicación o del usuario.
2. **Integridad de los datos**: Aseguran que las reglas de negocio se apliquen de manera constante, por ejemplo, asegurando que los valores insertados o actualizados sean válidos.
3. **Reducción de errores**: Previenen errores comunes, como la inserción de valores no válidos o la actualización de datos de manera incorrecta, asegurando la calidad de los datos.
4. **Mejora de rendimiento**: Ejecutan las operaciones automáticamente en la base de datos, lo que puede mejorar el rendimiento y reducir el código adicional que necesita la aplicación para realizar estas tareas.

Pregunta 2: ¿Tipos de triggers?

Los triggers se dividen según el tipo de evento que los activa y el momento en que se ejecutan:

1. **BEFORE Triggers**:
    - Se ejecutan **antes** de que la operación de inserción, actualización o eliminación se complete.
    - Se utilizan comúnmente para validar datos antes de que se modifiquen.
2. **AFTER Triggers**:
    - Se ejecutan **después** de que la operación se haya completado.
    - Son útiles para realizar acciones adicionales, como actualizaciones o auditorías después de que se ha realizado un cambio.
3. **INSTEAD OF Triggers**:
    - Se usan en lugar de la operación original. En vez de realizar la operación solicitada, el trigger ejecuta su propio conjunto de operaciones.
    - Usado típicamente en vistas.

4. **Triggers DML (Data Manipulation Language)**:
    - Activados por operaciones de **inserción**, **actualización** o **eliminación**.
    - Ejemplos: BEFORE INSERT, AFTER UPDATE.
5. **Triggers DDL (Data Definition Language)**:
    - Se activan por operaciones que cambian la estructura de la base de datos, como crear o eliminar tablas.
    - Ejemplo: AFTER CREATE TABLE.

Pregunta 3: ¿En qué casos NO son de utilidad?

Los triggers no siempre son adecuados para todas las situaciones. Algunas razones por las cuales podrían no ser útiles incluyen:

1. **Alta complejidad**: Los triggers pueden volverse complejos y difíciles de mantener cuando se usan en exceso o con operaciones muy intrincadas. Esto puede llevar a una sobrecarga innecesaria.
2. **Impacto en el rendimiento**: Los triggers pueden afectar el rendimiento de la base de datos, especialmente si se ejecutan en grandes volúmenes de datos. Como son procesos automáticos que se ejecutan cada vez que ocurre una operación de DML, pueden ralentizar las transacciones.
3. **Dificultad para depurar**: Los errores en los triggers pueden ser difíciles de rastrear y depurar, ya que se ejecutan de manera automática y no siempre es obvio cuándo o por qué fallaron.
4. **Dependencia del motor de base de datos**: Los triggers están muy ligados al motor de la base de datos que los soporta, lo que puede dificultar la portabilidad del sistema a otras bases de datos.
5. **Ciclo de operaciones no deseadas**: Si los triggers no están bien diseñados, pueden generar ciclos infinitos de operaciones. Por ejemplo, si un trigger de actualización vuelve a disparar otro trigger que también actualiza los mismos datos, esto puede llevar a un ciclo de actualizaciones y bloqueos