



Universidad César Vallejo

Pregrado

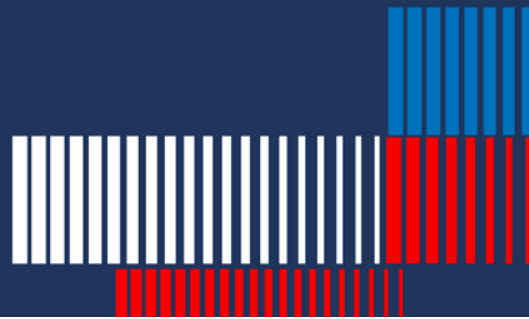
Gestión de Datos e Información II

30
años

Licenciada por
SUNEDU
para que puedas
salir adelante

SESIÓN 04

TRIGGERS (DISPARADORES)





- ✓ Un **"trigger"** (disparador o desencadenador) es un tipo de procedimiento almacenado que se ejecuta cuando se intenta modificar los datos de una tabla (o vista).
- ✓ Se definen para una tabla (o vista) específica.
- ✓ Se crean para conservar la integridad referencial y la coherencia entre los datos entre distintas tablas.
- ✓ Si se intenta modificar (agregar, actualizar o eliminar) datos de una tabla en la que se definió un disparador para alguna de estas acciones (inserción, actualización y eliminación), el disparador se ejecuta (se dispara) en forma automática.



- ✓ Un trigger se asocia a un evento (inserción, actualización o borrado) sobre una tabla.

La diferencia con los procedimientos almacenados del sistema es que los triggers:

- ✓ No pueden ser invocados directamente; al intentar modificar los datos de una tabla para la que se ha definido un disparador, el disparador se ejecuta automáticamente.
- ✓ No reciben y retornan parámetros.
- ✓ Son apropiados para mantener la integridad de los datos, no para obtener resultados de consultas.



- ✓ Los disparadores, a diferencia de las restricciones "**check**", pueden hacer referencia a campos de otras tablas.
- ✓ **Por ejemplo**, puede crearse un trigger de inserción en la tabla "ventas" que compruebe el campo "stock" de un artículo en la tabla "articulos"; el disparador controlaría que, cuando el valor de "stock" sea menor a la cantidad que se intenta vender, la inserción del nuevo registro en "ventas" no se realice.



- ✓ Los disparadores se ejecutan **DESPUÉS** de la ejecución de una instrucción **"insert"**, **"update"** o **"delete"** en la tabla en la que fueron definidos. Las restricciones se comprueban **ANTES** de la ejecución de una instrucción **"insert"**, **"update"** o **"delete"**. Por lo tanto, las restricciones se comprueban primero, si se infringe alguna restricción, el desencadenador no llega a ejecutarse.
- ✓ Los **triggers** se crean con la instrucción **"create trigger"**. Esta instrucción especifica la tabla en la que se define el disparador, los eventos para los que se ejecuta y las instrucciones que contiene.



Los **triggers** definen dos tablas especiales que contienen toda la información que necesitamos: **inserted y deleted**. Ambas son subconjuntos de la tabla que contiene el **trigger**, justamente con los registros que nos interesan: los afectados por la sentencia desencadenante.

INSERTED contiene los registros con los nuevos valores para triggers que se desencadenan con sentencias **INSERT** (nuevos registros) y **UPDATE** (nuevo valor para registros actualizados).

DELETED, por su parte, contiene los registros con los viejos valores para triggers que se desencadenan con sentencias **DELETE** (registros borrados) y **UPDATE** (valor anterior para los registros actualizados).



En un trigger definido como **AFTER INSERT** sólo dispondremos de la tabla inserted, en uno definido como **AFTER DELETE** solamente tendremos la tabla deleted, mientras que, finalmente, ambas tablas estarán disponibles en triggers definidos para ejecutarse tras un **UPDATE** con **AFTER UPDATE**, pudiendo consultar así de los valores antes y después de actualizarse los registros correspondientes.



```
CREATE TRIGGER NOMBREDISPARADOR  
ON NOMBRETABLA  
FOR EVENTO- INSERT, UPDATE O DELETE  
AS  
SENTENCIAS
```




```
CREATE [ OR ALTER ] TRIGGER [ schema_name . ]trigger_name
ON { table }
[ WITH <ddl_trigger_option> [ ,...n ] ]
{ FOR | AFTER } { [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
AS { sql_statement [ ; ] [ ,...n ] }
<ddl_trigger_option> ::=
[ NATIVE_COMPILATION ]
[ SCHEMABINDING ]
[ EXECUTE AS clause ]
```



```
CREATE TRIGGER mensaje_pasajero  
ON pasajero  
FOR INSERT, UPDATE  
AS  
    PRINT 'Pasajero actualizado correctamente';  
GO
```



```
create trigger deltitle
on titles
for delete
as
if (select count(*)
    from deleted, salesdetail
    where salesdetail.title_id = deleted.title_id) > 0
begin
    rollback transaction
    print 'No puedes eliminar un título con ventas.'
end;
```



```
create trigger DIS_ventas_insertar
on ventas for insert
as
declare @stock int
select @stock= stock from libros
      join inserted on inserted.codigolibro=libros.codigo
      where libros.codigo=inserted.codigolibro;
if (@stock>=(select cantidad from inserted))
      update libros set stock=stock-inserted.cantidad
      from libros join inserted on inserted.codigolibro=libros.codigo
      where codigo=inserted.codigolibro
else
begin
      raiserror ('Hay menos libros en stock de los solicitados para la venta', 16, 1)
      rollback transaction
end;
```



¿QUÉ HEMOS APRENDIDO HOY?

Pregrado

Ingeniería de
Sistemas



Para que reflexionen y entiendan la importancia de los temas tratados y el mejoramiento de su propio proceso de aprendizaje.



Universidad **César Vallejo**

Licenciada por Sunedu
para que puedas salir adelante