Universidad Nacional del Nordeste

Facultad de Ciencias Exactas y Naturales y Agrimensura

Carrera: Licenciatura en Sistemas de Información

Cátedra: Base de Datos

Año: 2023

Proyecto de Estudio: Disparadores (TRIGGERS)

GRUPO N° 2

Integrantes:

Apellido y Nombre	DNI	LU
Alfonzo, Daiana	37891410	48270
Corrales, José Luis	32229297	56246
Garcia, Emir Ariel	43.333.263	49.889
Muñoz Leandro Manuel	40084126	56300

Índice

Capítulo I: Introducción 3

Capítulo II: Marco Conceptual o Referencial 4

2.1 Concepto de Triggers (Disparadores) 4

2.2 Estructura de un Trigger 4

2.3 Tipos de Triggers 5

2.4 Ventajas de Utilizar Triggers 5

2.5 Desventajas de Utilizar Triggers 6

Capítulo III: Metodología 7

Capítulo IV: Desarrollo del Tema/ Resultados 8

Capítulo V: Conclusiones 15

Capítulo VI: Bibliografía 17

Índice de Figuras

Figura 1 - Código de creación de la tabla auditoriaConsorcio 8

Figura 2 - Código de creación del TRIGGER auditoriaConsorcio – Operación Update 9

Figura 3 - Consulta de la tabla consorcio 9

Figura 4 - Update en tabla consorcio 9

Figura 5 - Actualización de la tabla auditoriaConsorcio 10

Figura 6 - Código de creación del TRIGGER auditConsorcio – Operación Delete. 10

Figura 7 - Prueba de borrado de un registro en tabla consorcio y registro en tabla auditoriaConsorcio 10

Figura 8 - Tabla auditoriaGasto y TRIGGER auditGasto – Operación Update 11

Figura 9 - SELECT de la tabla gasto 11

Figura 10 - Prueba de modificación en tabla gasto y registro en auditoriaConsorcio 11

Figura 11 - Código de creación del trigger de la tabla gasto para la operación DELETE 12

Figura 12 - Código de inserción de un gasto nuevo. 12

Figura 13 - Resultado luego de la inserción que se visualiza en el id 8002 12

Figura 14 - Código de eliminación del gasto 8002 12

Figura 15 - -Resultado luego de la eliminación visualizándo las tablas gasto y auditoriaGasto 12

Figura 16 - Código de creación de una tabla auxiliar auditoriaAdministrador para resguardar los datos actualizados de la tabla administrador. 13

Figura 17 - Código de creación del trigger de la tabla administrador para la operación UPDATE 13

Figura 18 - Código de inserción de un registro en la tabla administrador. 13

Figura 19 - Resultado luego de insertar 2 veces el registro anterior (fila 175 y 176). 13

Figura 20 - Código de actualización de los datos del registro 176 de la tabla administrador, cambiando nombre, teléfono y fecha de nacimiento 14

Figura 21 - Resultado de las tablas administrador y auditoriaAdministrador luego de la actualización. 14

Figura 22 - Código de creación del trigger de la tabla administrador para la operación DELETE 14

Figura 23 - Código de eliminación del registro 176 de la tabla administrador y mensaje de error lanzado por el trigger. 15
Figura 24 - Consulta de la tabla administrador en la que se visualiza el registro que se intentó eliminar anteriormente. 15

Capítulo I: Introducción

El propósito fundamental de esta investigación es proporcionar una comprensión en profundidad de los triggers en bases de datos y su relevancia en la gestión de datos. Los triggers son objetos esenciales en cualquier **Sistema de Gestión de Bases de Datos (SGBD)** y consisten en un conjunto de reglas predefinidas que se asocian a tablas específicas. Estas reglas se desencadenan automáticamente en la base de datos en respuesta a eventos particulares, como inserciones, actualizaciones o eliminaciones de registros. En otras palabras, los triggers automatizan acciones específicas en las tablas de la base de datos, lo que ahorra tiempo y mejora la eficiencia en la gestión de datos.

Los triggers, o disparadores, son scripts en lenguaje SQL que se utilizan ampliamente en sistemas de bases de datos como MySQL, PostgreSQL y, en este caso, SQL Server. Su función principal es optimizar la gestión de bases de datos al permitir que muchas operaciones se realicen de manera automática, eliminando la necesidad de intervención humana. Además de su funcionalidad automatizada, los triggers también desempeñan un papel fundamental en la mejora de la seguridad y la integridad de los datos. Esto se logra a través de la programación de restricciones y verificaciones que minimizan errores y mantienen la sincronización de la información.

Por lo tanto, esta investigación se propone demostrar y enseñar el propósito y el uso de los triggers en un motor de base de datos. Exploraremos su importancia en la operatividad de una base de datos, abordando temas como los tipos de triggers, su creación, modificación y eliminación. Nuestro enfoque se centrará en el motor de base de datos SQL Server, lo que permitirá a los lectores obtener una comprensión sólida de esta tecnología fundamental en la gestión de datos.

Capítulo II: Marco Conceptual o Referencial

2.1 Concepto de Triggers (Disparadores)

Un trigger, también conocido como disparador en español, es un conjunto de sentencias SQL que se ejecutan de forma automática cuando se produce un evento que modifica una tabla. Estos eventos se relacionan con la manipulación de datos almacenados, es decir, cuando se ejecutan sentencias INSERT, UPDATE o DELETE. Lo que hace que los triggers sean interesantes es su capacidad para programarse de manera que se ejecuten antes o después de estas operaciones.

Para ilustrar este concepto, consideremos un ejemplo: *Supongamos que tenemos una tabla de inventario en una base de datos y deseamos realizar un seguimiento de los productos agotados. Podríamos crear un trigger que se ejecute automáticamente después de una operación de eliminación (DELETE) en la tabla de ventas. Este trigger podría verificar si un producto está agotado y, en caso afirmativo, actualizar la tabla de inventario para reflejar ese cambio. Esto se hace automáticamente cada vez que se elimina un registro de ventas, asegurando que la información de inventario esté siempre actualizada.*

2.2 Estructura de un Trigger

La estructura y el funcionamiento de un trigger pueden resumirse en tres pasos:

1. Se produce una llamada de activación al código que se ejecutará.
2. Se aplican las restricciones necesarias para realizar la acción, como verificar una condición o nulidad.
3. Una vez verificadas las restricciones, se ejecuta la acción basada en las instrucciones recibidas en el primer punto.

Para comprender mejor la sintaxis de un trigger, aquí hay un ejemplo simple genérico:

```
CREATE TRIGGER nombre_disparador
BEFORE / AFTER
INSERT / UPDATE / DELETE
ON nombre_tabla
FOR EACH ROW
BEGIN
-- Código del trigger
END;
```

Explicación de cada parámetro:

CREATE TRIGGER: nombre_disparador: Se utiliza para crear un nuevo trigger o cambiar el nombre de uno existente.

BEFORE / AFTER: Define cuándo se ejecutará el trigger, ya sea antes o después de un evento específico.

INSERT / UPDATE / DELETE: Describe la acción que se desea llevar a cabo en la tabla.

ON nombre_tabla: Especifica la tabla a la que se asocia el trigger.

FOR EACH ROW: Esta declaración se refiere a los triggers de filas, lo que significa que se ejecutarán cada vez que se modifique una fila.

BEGIN-END: Aquí se especifica el código que se ejecutará como parte del trigger.

Y aquí hay un ejemplo que hemos creado para la aplicación de los scripts:

```
CREATE TRIGGER trg_auditConsorcio_update
ON dbo.consorcio
AFTER UPDATE
AS
BEGIN
      -- Registrar los valores antes de la modificación en una tabla auxiliar
        INSERT INTO auditoriaConsorcio
        SELECT *, GETDATE(), SUSER_NAME(), 'Update'
        FROM deleted;
END;
```

2.3 Tipos de Triggers

Los triggers SQL son funciones almacenadas que se ejecutan inmediatamente cuando ocurren eventos específicos. Se asemejan a la programación basada en eventos y pueden ser de varios tipos:

- **Disparadores DML (Lenguaje de Manipulación de Datos):** Estos se activan en respuesta a comandos DML como INSERT, UPDATE y DELETE. También se conocen como "Disparadores a nivel de tabla".
- **Disparadores DDL (Lenguaje de Definición de Datos):** Estos permiten ejecutar código en respuesta a cambios en la estructura de la base de datos, como la creación o eliminación de tablas. Son conocidos como "Disparadores a nivel de base de datos".
- **Disparadores de inicio de sesión:** Se invocan cuando se produce un evento de inicio de sesión, como el inicio de sesión, cierre de sesión o apagado del sistema. Se utilizan para auditar el acceso al sistema y gestionar la identidad de los usuarios.
- **Disparadores CLR (Common Language Runtime):** Estos son triggers únicos que aprovechan la tecnología .NET para ejecutar código. Son útiles cuando se necesitan cálculos complejos o interacción con entidades no relacionadas con SQL.

2.4 Ventajas de Utilizar Triggers

El uso de triggers proporciona varias ventajas importantes:

- Validación de datos: Los triggers permiten validar valores que no se pueden verificar mediante restricciones, lo que garantiza la integridad de los datos.
- Ejecución de reglas de negocios: Pueden aplicarse reglas empresariales complejas de manera automatizada.
- Automatización de acciones complejas: Los triggers permiten realizar acciones sofisticadas en respuesta a eventos específicos.
- Registro de cambios: Ayudan a rastrear y mantener un registro de los cambios realizados en una tabla, a menudo utilizando una tabla de registro.

2.5 Desventajas de Utilizar Triggers

Sin embargo, el uso de triggers conlleva ciertas desventajas:

- Pérdida de visibilidad: Al ejecutarse automáticamente, puede ser difícil rastrear y depurar sentencias SQL.
- Sobrecarga del servidor: Un uso excesivo o incorrecto de triggers puede causar respuestas lentas del servidor y aumentar la carga de trabajo.
- Complejidad adicional: Los triggers pueden agregar complejidad al diseño de la base de datos. A medida que aumenta el número de triggers, el mantenimiento y la comprensión del sistema pueden volverse más difíciles.
- Riesgo de bucles infinitos: Si no se configuran correctamente, los triggers pueden generar bucles infinitos, donde uno dispara otro en un ciclo sin fin. Esto puede bloquear la base de datos y causar un rendimiento deficiente.
- Desencadenadores ocultos: Los triggers pueden no ser obvios para los desarrolladores y administradores que no estén familiarizados con la base de datos. Esto puede llevar a sorpresas inesperadas y dificultades para diagnosticar problemas.
- Dificultad en la migración y la transferencia: Los triggers pueden complicar la migración de datos entre bases de datos o sistemas, ya que es necesario asegurarse de que los triggers se activen correctamente en el nuevo entorno.
- Entre otros.

Capítulo III: Metodología

En este capítulo se presenta el plan seguido o las acciones llevadas a cabo para realizar el trabajo, las dificultades encontradas y cualquier otra información que proporcione la idea de cómo se realizó el trabajo.

Para llevar a cabo este Trabajo Práctico, se han seguido los siguientes pasos:

En reunión virtual mencionada se había definido la implementación de triggers de auditorías para las tablas (Consorcio, Gasto y Administrador) que responderán a cada operación de actualización (UPDATE) o eliminación (DELETE). Estos triggers registrarán en tablas auxiliares los valores de los registros antes de ser modificados o eliminados, junto con la fecha y hora de la operación, así como el usuario de la base de datos que realizó la acción. Esta acción tiene como objetivo mantener un registro detallado de las modificaciones realizadas en las tablas y proporcionar trazabilidad.

Además, se ha definido un trigger específico que impedirá la ejecución de una operación DELETE en la tabla (Administrador). Cuando se intente eliminar un registro en esta tabla, se emitirá un mensaje de error y no se permitirá la operación. Esto se ha implementado para garantizar la integridad y seguridad de los datos en la tabla Administrador.

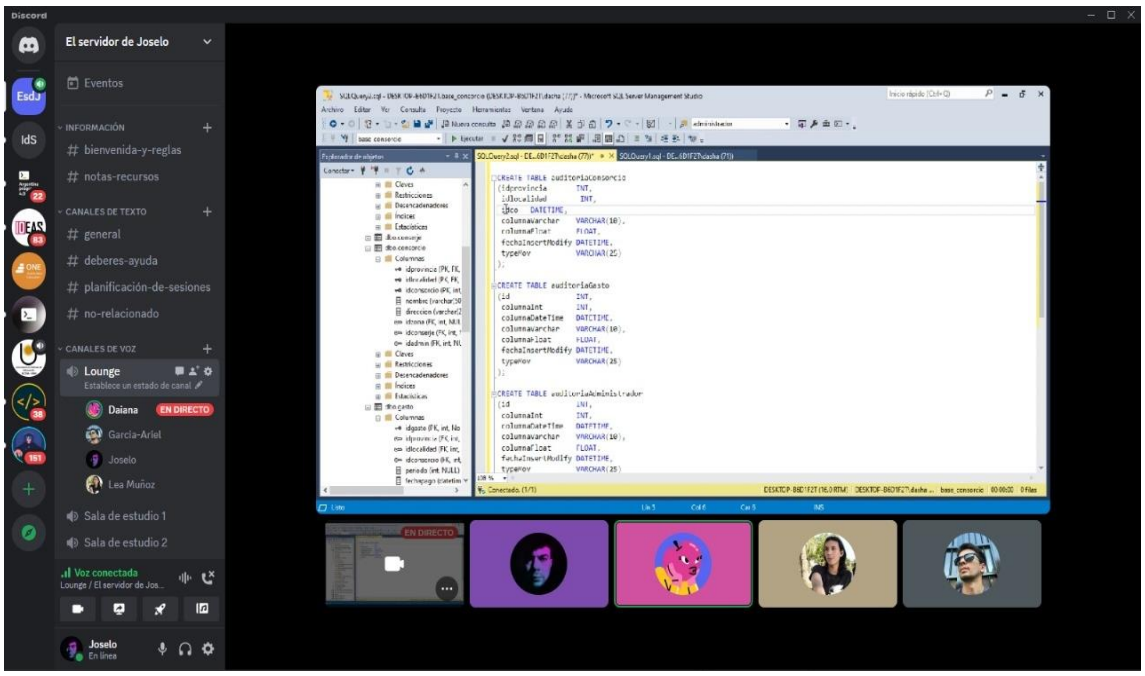
Las herramientas y procedimientos utilizados en este trabajo son los siguientes:

Herramienta Principal: Para llevar a cabo la implementación de los triggers y ejemplificar su funcionamiento, se ha empleado el Gestor de Base de Datos SQL Server Management Studio.

Origen de los Datos: Los datos y ejemplos utilizados se extraen de una base de datos ya creada y cargada, denominada "base_consorcio". Estos datos proporcionan un contexto realista para ejemplificar los triggers y su aplicación en situaciones prácticas.

Esta metodología se ha seguido para proporcionar ejemplos concretos y prácticos de cómo funcionan los triggers en un entorno de base de datos. La implementación de triggers de auditorías y la restricción de DELETE en la tabla Administrador son ejemplos concretos que ilustran el uso y la utilidad de los triggers en la gestión de bases de datos y que se harán ver en el desarrollo del mismo.

Además, para llevar a cabo el desarrollo del presente trabajo de investigación se han realizado reuniones grupales para la elaboración del Script de los Triggers solicitados y las tablas auxiliares necesarias, para ellos se dispuso un día y horario determinado para concertar una reunión virtual por videollamada a través de la plataforma **Discord**.



Capítulo IV: Desarrollo del Tema/ Resultados

La seguridad y la integridad de los datos son de vital importancia en la actualidad. La creciente cantidad de información almacenada en sistemas de bases de datos ha generado la necesidad de implementar mecanismos efectivos de auditoría. La auditoría de bases de datos es una práctica que permite rastrear y registrar cambios en los datos, identificar actividades sospechosas y garantizar la transparencia.

Los triggers de auditoría de bases de datos son elementos que permiten la monitorización y el registro automatizado de ciertos eventos en una base de datos, tales como update, delete o insert permitiendo su trazabilidad

A continuación, se intentará explicar de la manera más sencilla y clara posible la implementación de triggers sobre las tablas consorcio, gasto y administrador, tablas que corresponde a la base de datos "base_consorcio".

Para empezar con el desarrollo del tema, luego de haber buscado información e interiorizarnos, comenzamos con la elaboración del código necesario para la implementación de los TRIGGERS solicitados, cuyo Script es un producto del aporte en simultáneo de todos los integrantes del grupo.

Para comenzar se creó una tabla auditoriaConsorcio:

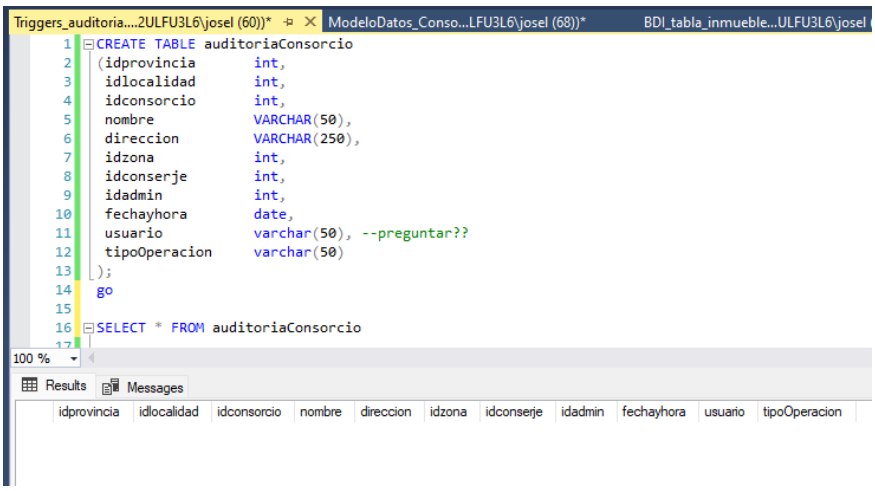


Figura 1 - Código de creación de la tabla auditoriaConsorcio

Ésta tabla guardará los datos de los registros que han de sufrir modificaciones luego de que ello ocurra.

Una vez creada la tabla mencionada en la base de datos creamos el TRIGGER trg_auditConsorcio_update que actuará como un disparador cada vez que se intente realizar una actualización en algún registro de la tabla consorcio en la base de datos base_consorcio cuyo objetivo principal es el de registrar en la tabla auditoriaConsorcio los valores antiguos de la fila que está siendo modificada.

```
CREATE TRIGGER trg_auditConsorcio_update
ON dbo.consorcio
AFTER UPDATE
AS
BEGIN
    -- Registrar los valores antes de la modificación en una tabla auxiliar
    INSERT INTO auditoriaConsorcio
    SELECT *, GETDATE(), ORIGINAL_LOGIN(), 'Update'
    FROM deleted;
END;
go
```

Figura 2 - Código de creación del TRIGGER auditoriaConsorcio – Operación Update

El TRIGGER se ejecutará después de que se haya realizado la actualización en la tabla realizando una inserción en la tabla auditoriaConsorcio de los datos antiguos, la fecha de la actualización el nombre de usuario que realizó la operación. A continuación, en primer lugar se realizará una consulta (select) para obtener los datos de los registros de la tabla consorcio.

```
144 SELECT * FROM consorcio
```

100 %

Results | Messages

	idprovincia	idlocalidad	idconsorcio	nombre	direccion	idzona	idconserje	idadmin
1	1	1	1	EDIFICIO-111	PARAGUAY Nº 630	5	100	1
2	1	2	2	EDIFICIO-122	Bº 250 VIV SEC 2 MZ 4 CSA Nº 2	5	99	2
3	2	48	1	EDIFICIO-2481	SAN LUIS Nº 1035, 4º piso, Dpto c	4	98	3
4	2	55	2	EDIFICIO-2552	REMEDIOS DE ESCALADA Nº 5353	3	97	4
5	3	16	1	EDIFICIO-3161	Bº VENEZUELA, GR.4, MZ.21,C.2	2	96	5
6	3	20	2	EDIFICIO-3202	LAVALLE Nº 1386	3	95	6
7	3	25	3	EDIFICIO-3253	JOSE G OMEZ Nº 770	4	94	7
8	3	27	4	EDIFICIO-3274	MARIANO MORENO Nº 1626, 2º piso, Dpto D	4	93	8
9	3	29	5	EDIFICIO-3295	MZ1 PC6 C26 Bº SANTA BARBARA	5	92	9
10	4	21	1	EDIFICIO-4211	COMANDANTE FONTANA Nº 174, .º piso, Dpto -	6	91	10
11	4	36	2	EDIFICIO-4362	SAN MARTIN Nº 2195, PAº piso, Dpto 1	4	90	11
12	4	37	3	EDIFICIO-4373	PADRE BORGATTI Y MORENO	1	89	12
13	5	3	1	EDIFICIO-531	JULIO A. ROCA Nº 1281	6	88	13
14	5	4	2	EDIFICIO-542	Bº L.SECA, 119VIV. STOR A, CASA 3	2	87	14
15	5	12	3	EDIFICIO-5123	BELGRANO Nº 2481	4	86	15

Query executed successfully. | LAPTOP-2ULFU3L6\SQLEXPRESS ... | LAPTOP-2ULFU

Figura 3 - Consulta de la tabla consorcio

A continuación, ejecutamos un update para comprobar el funcionamiento del TRIGGER, y podemos comprobar que el registro que ha sido modificado y se han guardado en la tabla auditoriaConsorcio los valores antiguos del mencionado registro.

```
149 update consorcio set direccion = 'Paraguay 387 - Planta Baja' where idadmin = 1
150 SELECT * FROM consorcio
```

100 %

Results | Messages

	idprovincia	idlocalidad	idconsorcio	nombre	direccion	idzona	idconserje	idadmin
1	1	1	1	EDIFICIO-111	Paraguay 387 - Planta Baja	5	100	1

Figura 4 - Update en tabla consorcio

```
148
149 SELECT * FROM auditoriaConsorcio
```

100 %

Results | Messages

	idprovincia	idlocalidad	idconsorcio	nombre	direccion	idzona	idconserje	idadmin	fechayhora	usuario	tipoOperacion
1	1	1	1	EDIFICIO-111	Paraguay 387 - Planta Baja	5	100	1	2023-10-30	LAPTOP-2ULFU3L6\josel	Update

Figura 5 - Actualización de la tabla auditoriaConsorcio

Continuando con el avance de nuestro trabajo desarrollamos el código para del TRIGGER que se ejecutará cuando un usuario intente realizar una operación de eliminación sobre un registro de la tabla consorcio:

```
30
31 CREATE TRIGGER trg_auditConsorcio_delete
32 ON dbo.consorcio
33 AFTER DELETE
34 AS
35 BEGIN
36         -- Registrar los valores antes de la eliminación en una tabla auxiliar
37         INSERT INTO auditoriaConsorcio
38         SELECT * , GETDATE(), SUSER_NAME(), 'Delete'
39         FROM deleted;
40 END;
41 go
42
```

Figura 6 - Código de creación del TRIGGER auditConsorcio – Operación Delete.

El código que se ejecutará después de la operación de borrado en la tabla consorcio, registrará los valores antiguos que han sido eliminados en la tabla auditoriaConsorcio creada y descripta con anterioridad.

```
154
155 DELETE FROM consorcio WHERE idprovincia =1 AND idlocalidad=1 AND idconsorcio=1
156 SELECT * FROM auditoriaConsorcio
157
```

100 %

Results | Messages

	idprovincia	idlocalidad	idconsorcio	nombre	direccion	idzona	idconserje	idadmin	fechayhora	usuario	tipoOperacion
1	1	1	1	EDIFICIO-111	PARAGUAY Nº 630	5	100	1	2023-10-30	LAPTOP-2ULFU3L6\josel	Update
2	1	1	1	EDIFICIO-111	MISIONES 387 - Planta Baja	5	100	1	2023-10-30	LAPTOP-2ULFU3L6\josel	Delete

Figura 7 - Prueba de borrado de un registro en tabla consorcio y registro en tabla auditoriaConsorcio

Ahora nos ocuparemos de crear la tabla auditoriaGasto y también de desarrollar un TRIGGER que actúe como disparador ante modificaciones que se realicen dicha tabla.

```
43
44  CREATE TABLE auditoriaGasto
45   (idgasto                INT,
46    idprovincia            INT,
47    idlocalidad            int,
48    idconsorcio            int,
49    periodo                int,
50    fechapago              DATE,
51    idtipogasto            int,
52    importe                decimal(8,2),
53    fechayhora             date,
54    usuario                varchar(50),
55    tipoOperacion          varchar(50)
56   );
57   Go
58
59  CREATE TRIGGER trg_auditGasto_update
60   ON dbo.gasto
61   AFTER UPDATE
62   AS
63  BEGIN
64          -- Registrar los valores antes de la modificación en una tabla auxiliar
65          INSERT INTO auditoriaGasto
66          SELECT  * , GETDATE(), SUSER_NAME(), 'Update'
67          FROM deleted;
68   END;
69   go
70
```

Figura 8 - Tabla auditoriaGasto y TRIGGER auditGasto – Operación Update

El TRIGGER desarrollado realiza la acción de registrar en la tabla auxiliar auditoriaGasto, los datos antiguos después de realizarse el Update en la tabla gasto, así como también la fecha en que se realizó la modificación, el nombre de usuario de quien lo realizó y el tipo de operación.

Realizamos una prueba del código desarrollado, para ello primero efectuamos una consulta sobre los registros guardados en la tabla gasto, seguidamente hicimos un update sobre el importe del registro con idgasto = 11

```
154
155  SELECT * FROM GASTO
156
```

100 %

Results | Messages

	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe
1	11	1	2	2	3	2013-03-28 00:00:00.000	2	9723.16
2	12	1	2	2	2	2013-02-19 00:00:00.000	4	3696.11
3	13	1	2	2	9	2013-09-14 00:00:00.000	3	1238.12
4	14	1	2	2	9	2013-09-14 00:00:00.000	3	3846.92
5	15	1	2	2	5	2013-05-19 00:00:00.000	3	55294.59
6	16	1	2	2	4	2013-04-01 00:00:00.000	5	828.13
7	17	1	2	2	5	2013-05-02 00:00:00.000	2	9346.01

Figura 9 - SELECT de la tabla gasto

Confirmada la actualización del registro en la tabla gasto ejecutamos una consulta sobre la tabla auditoriaGasto, cuyos resultados fueron los siguientes:

```
156
157  update gasto set importe = '14879' where idgasto = 11
158
159  SELECT * FROM auditoriaGasto
160
```

100 %

Results | Messages

	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe	fechayhora	usuario	tipoOperacion
81	11	1	2	2	3	2013-03-28	2	9723.16	2023-10-30	LAPTOP-2ULFU3L6\josel	Update

Figura 10 - Prueba de modificación en tabla gasto y registro en auditoriaConsorcio.

Seguidamente se crea el trigger para la operación de DELETE en la tabla de gasto con nombre tgr_auditGasto_delete.

```
CREATE TRIGGER trg_auditGasto_delete
ON dbo.gasto
AFTER DELETE
AS
BEGIN
    -- Registrar los valores antes de la eliminación en una tabla auxiliar
    INSERT INTO auditoriaGasto
    SELECT * , GETDATE(),SUSER_NAME(), 'Delete'
    FROM deleted;
END
GO
```

Figura 11 - Código de creación del trigger de la tabla gasto para la operación DELETE

Como se puede ver en el código lo que realiza el disparador es insertar dentro de la tabla auxiliar auditoriaGasto todos los datos del registro que se elimina y además se guarda la fecha y hora, el nombre del usuario de la base de dato que se encuentre actualmente, y la leyenda 'Delete' para saber de qué tipo de operación se trata.

Para probar el disparador primero se crea un registro nuevo de gasto que se eliminará luego.

```
INSERT INTO gasto VALUES (1,1,1,2,GETDATE(), 1, 156215.20);
SELECT * FROM gasto;
```

Figura 12 - Código de inserción de un gasto nuevo.

7...	7998	24	17	6	8	2017-08-18 00:00:00.000	5	107.52
7...	7999	24	17	6	1	2017-01-01 00:00:00.000	2	1424.34
8...	8000	24	17	6	7	2017-07-02 00:00:00.000	5	869.17
8...	8002	1	1	1	2	2023-10-28 21:13:22.373	1	156215.20

Figura 13 - Resultado luego de la inserción que se visualiza en el id 8002.

Luego, se elimina el registro que se había ingresado anteriormente asignado con el idgasto nro 8002.

```
DELETE FROM gasto WHERE idgasto = 8002;
```

Figura 14 - Código de eliminación del gasto 8002

Results Messages

	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe
7...	7996	24	17	6	4	2017-04-08 00:00:00.000	5	195.62
7...	7997	24	17	6	9	2017-09-02 00:00:00.000	5	871.30
7...	7998	24	17	6	8	2017-08-18 00:00:00.000	5	107.52
7...	7999	24	17	6	1	2017-01-01 00:00:00.000	2	1424.34
8...	8000	24	17	6	7	2017-07-02 00:00:00.000	5	869.17

	idgasto	idprovincia	idlocalidad	idconsorcio	periodo	fechapago	idtipogasto	importe	fechayhora	usuario	tipoOperacion
1	8002	1	1	1	2	2023-10-30	1	156215.20	2023-10-30	LeaMunoz	Delete

Figura 15 - -Resultado luego de la eliminación visualizándo las tablas gasto y auditoriaGasto

Y como se puede comprobar, el disparador realiza la función requerida, guardando los datos del registro eliminado en la tabla auditoriaGasto(egistro de tabla inferior).

A continuación, se procede a la creación de disparadores para la tabla administrador, en el cual se debe resguardar los datos que se han modificado y evitar la eliminación de algún registro. Por lo tanto, en primer lugar se crea una tabla auxiliar a la que llamaremos auditoriaAdministrador que contendrá las mismas columnas que la tabla administrador, además una columna para la fecha y hora, otra para el nombre de usuario de la base de datos y otro para el tipo de operación.

```
CREATE TABLE auditoriaAdministrador
(idadmin                INT,
 apeynom                VARCHAR(50),
 viveahi                VARCHAR(1),
 tel                    VARCHAR(20),
 sexo                   VARCHAR(1),
 fechanac               DATETIME,
 fechayhora             DATE,
 usuario                VARCHAR(50),
 tipoOperacion          VARCHAR(50)
);
GO
```

Figura 16 - Código de creación de una tabla auxiliar auditoriaAdministrador para resguardar los datos actualizados de la tabla administrador.

Luego, se procede a la creación del disparador para la operación de UPDATE, que llevará el nombre de tgr_auditAdmin_update.

```
CREATE TRIGGER trg_auditAdmin_update
ON dbo.administrador
AFTER UPDATE
AS
BEGIN
        -- Registrar los valores antes de la modificación en una tabla auxiliar
        INSERT INTO auditoriaAdministrador
        SELECT  * , GETDATE(), SUSER_NAME(), 'Update'
        FROM deleted;
END;
GO
```

Figura 17 - Código de creación del trigger de la tabla administrador para la operación UPDATE

El código indica que, al igual que los disparadores anteriores, al momento de actualizar algún dato de la tabla el disparador insertará dentro de la tabla auxiliar auditoriaAdministrador todos los datos del registro que se actualiza y además se guarda la fecha y hora, el nombre del usuario de la base de datos y la leyenda 'Update'.

Nuevamente, para probar que el disparador funciona correctamente, primero se crea un nuevo administrador.

```
INSERT INTO administrador VALUES ('Gomez María', 'N', '3794568542', 'F', '19900615');

SELECT * FROM administrador;
```

Figura 18 - Código de inserción de un registro en la tabla administrador.

172	172	SAUCEDO FAUSTINA	S	3678235689	F	1987-07-23 00:00:00.000
173	173	GONZALEZ LIDIA ESTER	N	3671235689	F	1984-11-20 00:00:00.000
174	174	ARAUJO GUILLERMO JOSE	S	3672235689	M	1985-10-13 00:00:00.000
175	175	Gomez Maria	N	3794568542	F	1990-06-15 00:00:00.000
176	176	Gomez Maria	N	3794568542	F	1990-06-15 00:00:00.000

Figura 19 - Resultado luego de insertar 2 veces el registro anterior (fila 175 y 176).

Como se puede notar, se ingresó dos veces el mismo registro por lo que se procede a modificar el último.

```
UPDATE administrador
SET apeynom = 'Fernandez Laura', tel = '3794611462', fechnac = '19840418'
WHERE idadmin = 176;

SELECT * FROM administrador;
SELECT * FROM auditoriaAdministrador;
```

Figura 20 - Código de actualización de los datos del registro 176 de la tabla administrador, cambiando nombre, teléfono y fecha de nacimiento

	idadmin	apeynom	viveahi	tel	sexo	fechnac
169	169	TALAVERA CONSTANCIA	N	3655235689	F	1986-01-02 00:00:00.000
170	170	VARGAS MIGUEL	S	3676235689	M	1986-03-15 00:00:00.000
171	171	MACIEL PAULINA	N	3677235689	F	1970-03-18 00:00:00.000
172	172	SAUCEDO FAUSTINA	S	3678235689	F	1987-07-23 00:00:00.000
173	173	GONZALEZ LIDIA ESTER	N	3671235689	F	1984-11-20 00:00:00.000
174	174	ARAUJO GUILLERMO JOSE	S	3672235689	M	1985-10-13 00:00:00.000
175	175	Gomez María	N	3794568542	F	1990-06-15 00:00:00.000
176	176	Fernandez Laura	N	3794611462	F	1984-04-18 00:00:00.000

	idadmin	apeynom	viveahi	tel	sexo	fechanac	fechayhora	usuario	tipoOperacion
1	176	Gomez María	N	3794568542	F	1990-06-15 00:00:00.000	2023-10-28	LeaMunoz	Update

Figura 21 - Resultado de las tablas administrador y auditoriaAdministrador luego de la actualización.

El disparador realiza la función requerida correctamente, guardando los datos del registro modificado en la tabla auditoriaAdministrador (Figura 21-2do registro de tabla inferior).

En cuanto al disparador para la operación DELETE, éste difiere de los demás ya que no debe permitir la eliminación de un registro de administrador. Por lo tanto, la creación del trigger con el nombre tgr_auditoriaAdmin_delete se realiza de la siguiente manera:

```
CREATE TRIGGER trg_auditAdmin_delete
ON dbo.administrador
AFTER DELETE
AS
BEGIN
    -- Emitir un mensaje y prevenir la operación de eliminación
    RAISERROR('La eliminación de registros en la tabla Administrador no está permitida.', 16, 1);
    ROLLBACK;
END
GO
```

Figura 22 - Código de creación del trigger de la tabla administrador para la operación DELETE

Cuando se intente eliminar un registro lo que realizará el disparador será, en primer lugar, emitir un mensaje de error que rece “La eliminación de registros de la tabla Administrador no está permitida”, luego ejecutará un rollback que revertirá la eliminación que se intentó realizar.

De igual manera se procede a realizar la prueba del disparador intentando eliminar el registro ingresado anteriormente.

```
14  DELETE FROM administrador WHERE idadmin = 176;
15
```

```
Messages
Msg 50000, Level 16, State 1, Procedure trg_auditAdmin_delete, Line 8 [Batch Start Line 13]
La eliminación de registros en la tabla Administrador no está permitida.
Msg 3609, Level 16, State 1, Line 14
The transaction ended in the trigger. The batch has been aborted.
```

Figura 23 - Código de eliminación del registro 176 de la tabla administrador y mensaje de error lanzado por el trigger.

```
14  DELETE FROM administrador WHERE idadmin = 176;
15
16  SELECT * FROM administrador;
```

Results | Messages

	idadmin	apeynom	viveahi	tel	sexo	fechnac
165	165	ARMOA DEMETRIA	N	3651235689	F	1992-08-15 00:00:00.000
166	166	LUGO DE R. RAMONA	S	3652235689	F	1975-10-18 00:00:00.000
167	167	LOPEZ ELOYSA	N	3653235689	F	1989-06-08 00:00:00.000
168	168	LOPEZ CONRADO	S	3654235689	M	1983-10-26 00:00:00.000
169	169	TALAVERA CONSTANCIA	N	3655235689	F	1986-01-02 00:00:00.000
170	170	VARGAS MIGUEL	S	3676235689	M	1986-03-15 00:00:00.000
171	171	MACIEL PAULINA	N	3677235689	F	1970-03-18 00:00:00.000
172	172	SAUCEDO FAUSTINA	S	3678235689	F	1987-07-23 00:00:00.000
173	173	GONZALEZ LIDIA ESTER	N	3671235689	F	1984-11-20 00:00:00.000
174	174	ARAUJO GUILLERMO JOSE	S	3672235689	M	1985-10-13 00:00:00.000
175	175	Gomez María	N	3794568542	F	1990-06-15 00:00:00.000
176	176	Fernandez Laura	N	3794611462	F	1984-04-18 00:00:00.000

Figura 24 - Consulta de la tabla administrador en la que se visualiza el registro que se intentó eliminar anteriormente.

Luego, se puede comprobar que el disparador funciona correctamente ya que al ejecutar la eliminación del registro lanza el error que se había programado y al visualizar la tabla de administradores se puede ver que sigue ahí (Figura 24).

Capítulo V: Conclusiones

En resumen, esta investigación demuestra la importancia y el funcionamiento de los disparadores en la gestión de bases de datos. Los disparadores, que consisten en conjuntos de declaraciones SQL que se activan automáticamente en respuesta a eventos específicos, desempeñan un papel importante en la automatización de tareas y la mejora de la integridad y seguridad de los datos, así como en la optimización de la gestión de bases de datos.

Hemos explorado muchos aspectos diferentes de los activadores, desde su estructura y tipos hasta sus ventajas y desventajas y los ejemplos prácticos de implementación de disparadores en bases de datos de SQL Server demostraron cómo utilizar estos elementos para registrar cambios, mantener la trazabilidad y aplicar restricciones en tablas específicas.

Esto es necesario para garantizar que los datos sigan siendo precisos y seguros, especialmente en entornos donde la integridad de los datos es crítica.

En un mundo donde la cantidad de datos almacenados sigue creciendo, la capacidad de automatizar tareas, garantizar la integridad de los datos y rastrear cambios se vuelve esencial, por lo que los triggers son una herramienta poderosa para lograr estos objetivos. Sin embargo, también es importante comprender las desventajas potenciales, como la complejidad y la posible sobrecarga del servidor, para utilizarlos de manera efectiva.

Por último, esta investigación ha proporcionado una visión sólida de los triggers en bases de datos y su relevancia en la gestión de datos. Los ejemplos prácticos presentados demuestran cómo los triggers pueden

mejorar la eficiencia y la seguridad de las bases de datos, lo que los convierte en una herramienta valiosa para cualquier profesional de la gestión de datos.

Capítulo VI: Bibliografía

[1] M. Winand, SQL PerformanceExplicated, M. Winand, 2012.

[2] D. Petkovic, Microsoft SQL Server 2016: A Beginner's Guide, McGraw-Hill Education, 2016.