Introduction

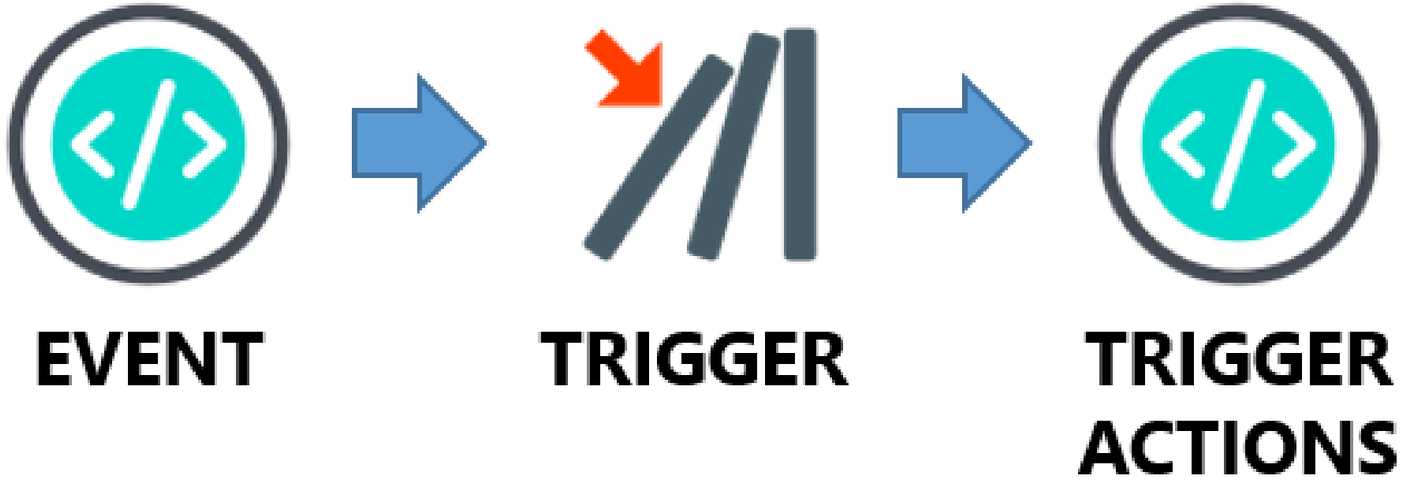
BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER

Florin Angelescu
Instructor

datacamp

What is a trigger?

- Special type of stored procedure
- Executed when an event occurs in the database server



Types of trigger (based on T-SQL commands)

- Data Manipulation Language (DML) triggers
  - `INSERT`, `UPDATE` or `DELETE` statements
- Data Definition Language (DDL) triggers
  - `CREATE`, `ALTER` or `DROP` statements
- Logon triggers
  - `LOGON` events

Types of trigger (based on behavior)

- `AFTER` trigger
  - The original statement executes
  - Additional statements are triggered
- Examples of use cases
  - Rebuild an index after a large insert
  - Notify the admin when data is updated

Types of trigger (based on behavior)

- `INSTEAD OF` trigger
  - The original statement is prevented from execution
  - A replacement statement is executed instead
- Examples of use cases
  - Prevent insertions
  - Prevent updates
  - Prevent deletions
  - Prevent object modifications
  - Notify the admin

Trigger definition (with AFTER)

```
-- Create the trigger by giving it a descriptive name
CREATE TRIGGER ProductsTrigger
-- The trigger needs to be attached to a table
ON Products
-- The trigger behavior type
AFTER INSERT
-- The beginning of the trigger workflow
AS
-- The action executed by the trigger
PRINT ('An insert of data was made in the Products table.');
```

Trigger definition (with INSTEAD OF)

```
-- Create the trigger by giving it a descriptive name
CREATE TRIGGER PreventDeleteFromOrders
-- The trigger needs to be attached to a table
ON Orders
-- The trigger behavior type
INSTEAD OF DELETE
-- The beginning of the trigger workflow
AS
-- The action executed by the trigger
PRINT ('You are not allowed to delete rows from the Orders table.');
```

AFTER vs. INSTEAD OF

```
CREATE TRIGGER MyFirstAfterTrigger
ON Table1
-- Triggered after
-- the firing event (UPDATE)
AFTER UPDATE
AS
{trigger_actions_section};
```

```
CREATE TRIGGER MyFirstInsteadOfTrigger
ON Table2
-- Triggered instead of
-- the firing event (UPDATE)
INSTEAD OF UPDATE
AS
{trigger_actions_section};
```

Let's practice!

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER

How DML triggers are used

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER

Florin Angelescu
Instructor

Why should we use DML triggers?

- Initiating actions when manipulating data
- Preventing data manipulation
- Tracking data or database object changes
- User auditing and database security

Deciding between AFTER and INSTEAD OF

```
CREATE TRIGGER MyFirstAfterTrigger
ON Table1
-- Triggered after
-- the firing event (UPDATE)
AFTER UPDATE
AS
{trigger_actions_section};
```

```
CREATE TRIGGER MyFirstInsteadOfTrigger
ON Table2
-- Triggered instead of
-- the firing event (UPDATE)
INSTEAD OF UPDATE
AS
{trigger_actions_section};
```

Deciding between AFTER and INSTEAD OF

AFTER trigger

Initial event fires the trigger

Initial event executes

The trigger actions execute

INSTEAD OF trigger

Initial event fires the trigger

Initial event is not executed anymore

The trigger actions execute

AFTER trigger usage example

- Data is inserted into a sales table
- Start a data cleansing procedure
- Generate a table report with the procedure results
- Notify the database administrator

```
CREATE TRIGGER SalesNewInfoTrigger
ON Sales
AFTER INSERT
AS
EXEC sp_cleansing @Table = 'Sales';
EXEC sp_generateSalesReport;
EXEC sp_sendnotification;
```

INSTEAD OF trigger usage example

Brand	Model	Power	Stock
Ecco	Standard	30W	30
Miry	Buma	45W	0
Lume	Ultra	50W	0

```
CREATE TRIGGER BulbsStockTrigger
ON Bulbs
INSTEAD OF INSERT
AS
```

- The power changes for some models

INSTEAD OF trigger usage example

Brand	Model	Power	Stock
Ecco	Standard	30W	30
Miry	Buma	50W	100
Lume	Ultra	52W	100

- The power changes for some models
- Update only the products with no stock

```
CREATE TRIGGER BulbsStockTrigger
ON Bulbs
INSTEAD OF INSERT
AS
IF EXISTS (SELECT * FROM Bulbs AS b
INNER JOIN inserted AS i
           ON b.Brand = i.Brand
          AND b.Model = i.Model
WHERE b.Stock = 0)
BEGIN
    UPDATE b
    SET b.Power = i.Power,
        b.Stock = i.Stock
    FROM Bulbs AS b
    INNER JOIN inserted AS i
                ON b.Brand = i.Brand
                  AND b.Model = i.Model
    WHERE b.Stock = 0
END
```

INSTEAD OF trigger usage example

Brand	Model	Power	Stock
Ecco	Standard	30W	30
Miry	Buma	50W	100
Lume	Ultra	52W	100
Ecco	Standard	35W	100

- The power changes for some models
- Update only the products with no stock
- Add new rows for the products with stock

```
-- First part was truncated for spacing reasons
IF EXISTS (SELECT * FROM Bulbs AS b
INNER JOIN inserted AS i
           ON b.Brand = i.Brand
          AND b.Model = i.Model
WHERE b.Stock = 0)
BEGIN
    UPDATE b
    SET b.Power = i.Power,
        b.Stock = i.Stock
    FROM Bulbs AS b
    INNER JOIN inserted AS i
                ON b.Brand = i.Brand
                  AND b.Model = i.Model
    WHERE b.Stock = 0
END
ELSE
    INSERT INTO Bulbs
    SELECT * FROM inserted;
```

Practice questions

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER

Trigger alternatives

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER

Florin Angelescu
Instructor

Triggers vs. stored procedures

Triggers

- Fired automatically by an event

```
-- Will fire an INSERT trigger
INSERT INTO Orders [...];
```

- Don't allow parameters or transactions
- Cannot return values as output

Stored procedures

- Run only when called explicitly

```
-- Will run the stored procedure
EXECUTE sp_DailyMaintenance;
```

- Accept input parameters and transactions
- Can return values as output

Triggers vs. stored procedures

Triggers

Used for:

- auditing
- integrity enforcement

Stored procedures

Used for:

- general tasks
- user-specific needs

Triggers vs. computed columns

Triggers

- calculate column values
- use columns **from other tables** for calculations
- `INSERT` or `UPDATE` used to calculate

```
-- Used in the trigger body
[...]
UPDATE
SET TotalAmount = Price * Quantity
[...]
```

Computed columns

- calculate column values
- use columns **only from the same table** for calculations
- calculation defined when creating the table

```
-- Column definition
[...]
TotalAmount AS Price * Quantity
[...]
```

Example of a computed column

```
CREATE TABLE [SalesWithPrice]
(
    [OrderID] INT IDENTITY(1,1),
    [Customer] NVARCHAR(50),
    [Product] NVARCHAR(50),
    [Price] DECIMAL(10,2),
    [Currency] NVARCHAR(3),
    [Quantity] INT,
    [OrderDate] DATE DEFAULT (GETDATE()),
    [TotalAmount] AS [Quantity] * [Price]
);
```

Using a trigger as a computed column

```
CREATE TRIGGER [SalesCalculateTotalAmount]
ON [SalesWithoutPrice]
AFTER INSERT
AS
    UPDATE [sp]
    SET [sp].[TotalAmount] = [sp].[Quantity] * [p].[Price]
    FROM [SalesWithoutPrice] AS [sp]
    INNER JOIN [Products] AS [p] ON [sp].Product = [p].[Product]
    WHERE [sp].[TotalAmount] IS NULL;
```

Let's compare them in practice!

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER