

# Known limitations of triggers

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER



Florin Angelescu  
Instructor

# Advantages of triggers

- Used for database integrity
- Enforce business rules directly in the database
- Control on which statements are allowed in a database
- Implementation of complex business logic triggered by a single event
- Simple way to audit databases and user actions

# Disadvantages of triggers

- Difficult to view and detect
- Invisible to client applications or when debugging code
- Hard to follow their logic when troubleshooting
- Can become an overhead on the server and make it run slower

# Finding server-level triggers

```
SELECT * FROM sys.server_triggers;
```

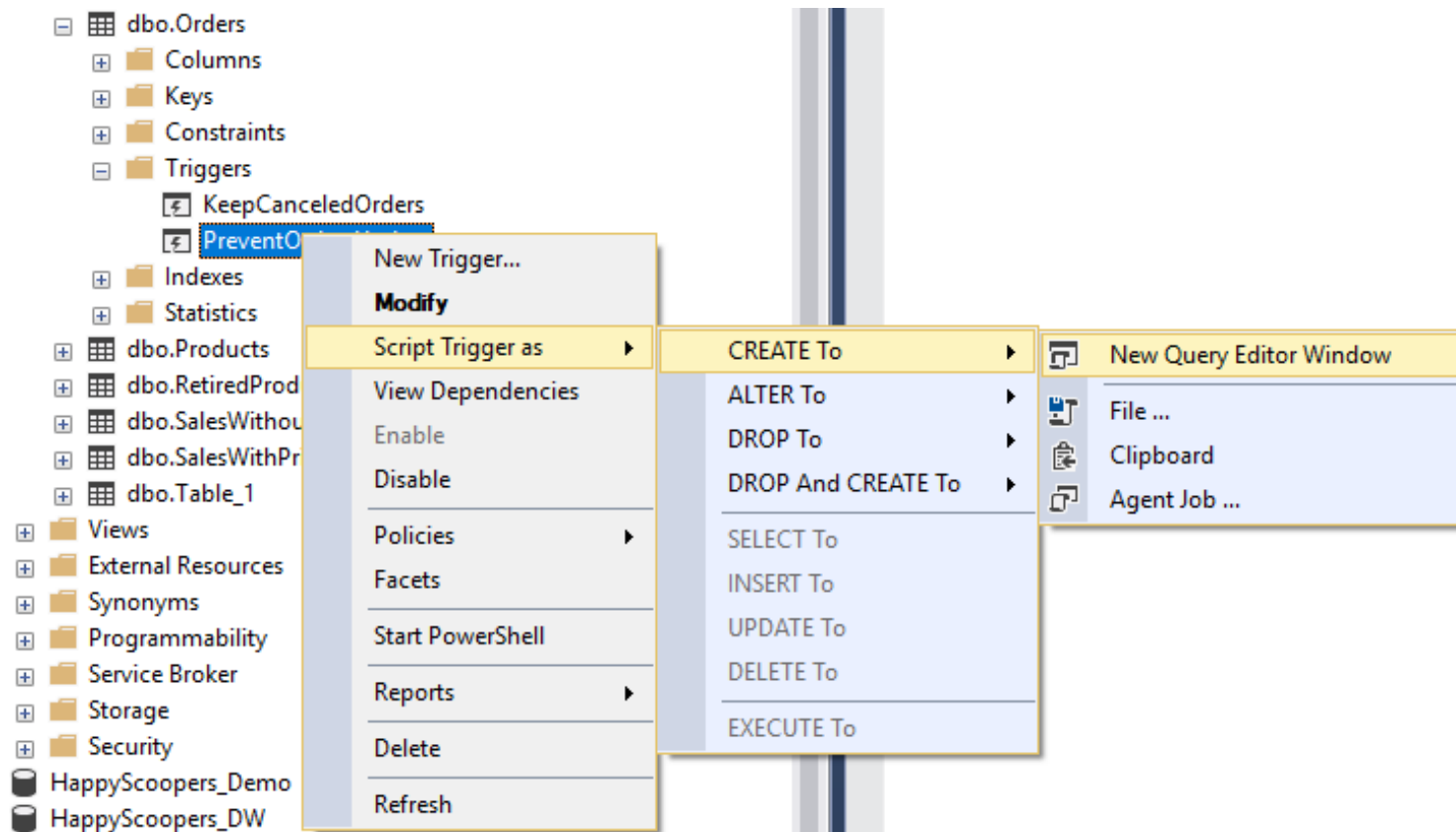
name	parent_class_desc	create_date	is_disabled	...
-----	-----	-----	-----	---
DenyNewDatabases	SERVER	2019-01-22	0	...
DenyNewLinkedServers	SERVER	2019-01-22	1	...
DenyDatabaseDelete	SERVER	2019-01-22	1	...

# Finding database and table triggers

```
SELECT * FROM sys.triggers;
```

name	parent_class_desc	create_date	is_disabled	is_instead_of_trigger
TrackRetiredProducts	OBJECT_OR_COLUMN	2019-04-19	0	0
PreventOrdersUpdate	OBJECT_OR_COLUMN	2019-04-22	0	1
TrackDatabaseTables	DATABASE	2019-04-23	0	0
KeepCanceledOrders	OBJECT_OR_COLUMN	2019-04-27	0	0
DiscountsPreventDelete	OBJECT_OR_COLUMN	2019-04-27	0	1
PreventNewDiscounts	OBJECT_OR_COLUMN	2019-04-27	0	1
PreventTableDeletion	DATABASE	2019-04-27	0	0

# Viewing a trigger definition (option 1)



```
CREATE TRIGGER PreventOrdersUpdate
ON Orders
INSTEAD OF UPDATE
AS
RAISERROR ( 'Updates on "Orders" table
            are not permitted.
            Place a new order
            to add new products.', 16, 1 );
```

# Viewing a trigger definition (option 2)

```
SELECT definition
FROM sys.sql_modules
WHERE object_id = OBJECT_ID ('PreventOrdersUpdate');
```

```
| definition |
|-----|
| CREATE TRIGGER PreventOrdersUpdate |
| ON Orders |
| INSTEAD OF UPDATE |
| AS |
| RAISERROR ('Updates on "Orders" table are not permitted. |
|           Place a new order to add new products.', 16, 1); |
```

# Viewing a trigger definition (option 3)

```
SELECT OBJECT_DEFINITION (OBJECT_ID ('PreventOrdersUpdate'));
```

```
| (No column name) |  
|-----|  
| CREATE TRIGGER PreventOrdersUpdate |  
| ON Orders |  
| INSTEAD OF UPDATE |  
| AS |  
| RAISERROR ('Updates on "Orders" table are not permitted. |  
|           Place a new order to add new products.', 16, 1); |
```



# Viewing a trigger definition (option 4)

```
EXECUTE sp_helptext @objname = 'PreventOrdersUpdate';
```

```
| Text |
|-----|
| CREATE TRIGGER PreventOrdersUpdate |
| ON Orders |
| INSTEAD OF UPDATE |
| AS |
| RAISERROR ('Updates on "Orders" table are not permitted. |
|           Place a new order to add new products.', 16, 1); |
```

# Triggers best practice

Tips:

- well-documented database design
- simple logic in trigger design
- avoid overusing triggers

# Let's practice!

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER

# Use cases for AFTER triggers (DML)

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER



Florin Angelescu  
Instructor

# Keeping a history of row changes

```
SELECT * FROM Customers;
```

Customer	ContractID	Address	PhoneNo
Every Fruit	ABF138256334	2522 Consectetuer St.	1-307-717-2294
eFruits	691C37BC3CED	1908 Fames Street	1-854-241-5573
Healthy Choices	435ADE342265	2826 Mauris Rd.	1-369-765-1647
Health Mag	73F6095C6930	1080 Aliquet. St.	1-634-676-3716
Fruit Mania	5CC27CBC78BA	311 In Avenue	1-790-501-4629

# Keeping a history of row changes

```
SELECT * FROM CustomersHistory;
```

Customer	ContractID	Address	PhoneNo	ChangeDate
Every Fruit	ABF138256334	2522 Consectetuer St.	1-307-717-2294	2017-05-03
eFruits	691C37BC3CED	1908 Fames Street	1-854-241-5573	2017-10-23
Healthy Choices	435ADE342265	2826 Mauris Rd.	1-369-765-1647	2018-02-10
Health Mag	73F6095C6930	1080 Aliquet. St.	1-634-676-3716	2018-03-03
Fruit Mania	5CC27CBC78BA	311 In Avenue	1-790-501-4629	2018-09-15

# Keeping a history of row changes

## Customers

Customer	ContractID	Address	PhoneNo
-----	-----	-----	-----
eFruits	691C37BC3CED	1908 Fames Street	1-854-241-6000

## CustomersHistory

Customer	ContractID	Address	PhoneNo	ChangeDate
-----	-----	-----	-----	-----
eFruits	691C37BC3CED	1908 Fames Street	1-854-241-5573	2017-10-23
eFruits	691C37BC3CED	1908 Fames Street	1-854-241-6000	2019-05-12

# Keeping a history of row changes

```
CREATE TRIGGER CopyCustomersToHistory
ON Customers

AFTER INSERT, UPDATE

AS

    INSERT INTO CustomersHistory (Customer, ContractID, Address, PhoneNo)
    SELECT Customer, ContractID, Address, PhoneNo, GETDATE()
    FROM inserted;
```



# Table auditing using triggers

```
CREATE TRIGGER OrdersAudit
ON Orders
AFTER INSERT, UPDATE, DELETE
AS
    DECLARE @Insert BIT = 0, @Delete BIT = 0;
    IF EXISTS (SELECT * FROM inserted) SET @Insert = 1;
    IF EXISTS (SELECT * FROM deleted) SET @Delete = 1;
    INSERT INTO [TablesAudit] ([TableName], [EventType], [UserAccount], [EventDate])
    SELECT 'Orders' AS [TableName]
        ,CASE WHEN @Insert = 1 AND @Delete = 0 THEN 'INSERT'
              WHEN @Insert = 1 AND @Delete = 1 THEN 'UPDATE'
              WHEN @Insert = 0 AND @Delete = 1 THEN 'DELETE'
              END AS [Event]
        ,ORIGINAL_LOGIN()
        ,GETDATE();
```

# Notifying users

```
CREATE TRIGGER NewOrderNotification
ON Orders
AFTER INSERT
AS
    EXECUTE SendNotification @RecipientEmail = 'sales@freshfruit.com'
        ,@EmailSubject = 'New order placed'
        ,@EmailBody = 'A new order was just placed.';
```

# Let's practice!

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER

# Use cases for INSTEAD OF triggers (DML)

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER



Florin Angelescu  
Instructor

# General use of INSTEAD OF triggers

- Prevent operations from happening
- Control database statements
- Enforce data integrity

# Triggers that prevent changes

```
CREATE TRIGGER PreventProductChanges
ON Products
INSTEAD OF UPDATE
AS
    RAISERROR ('Updates of products are not permitted.
               Contact the database administrator if a change is needed.', 16, 1);
```

# Triggers that prevent and notify

```
CREATE TRIGGER PreventCustomersRemoval
ON Customers
INSTEAD OF DELETE
AS
    DECLARE @EmailBodyText NVARCHAR(50) =
        (SELECT 'User "' + ORIGINAL_LOGIN() +
            '" tried to remove a customer from the database.');
```

```
RAISERROR ('Customer entries are not subject to removal.', 16, 1);
```

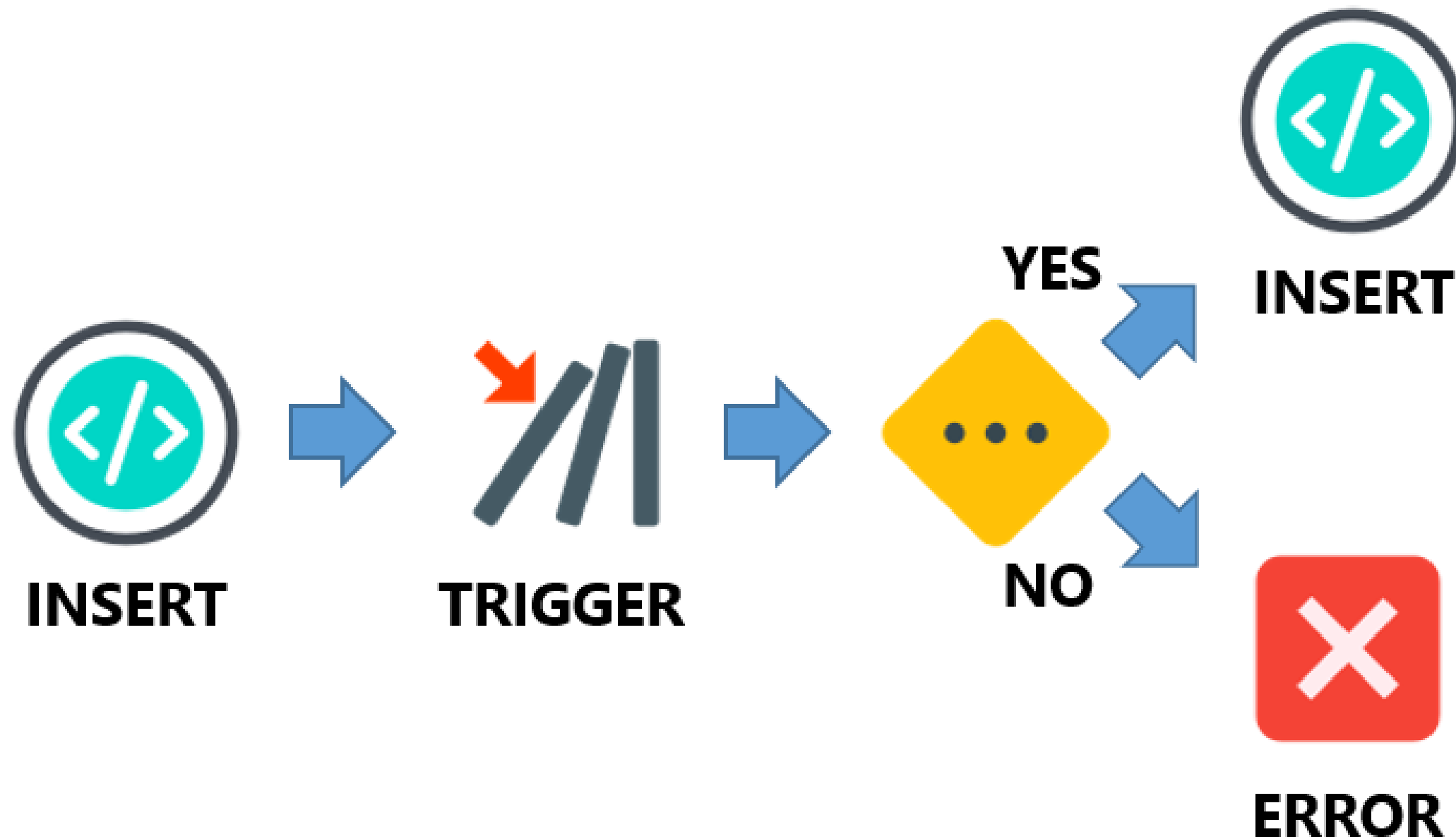
```
EXECUTE SendNotification @RecipientEmail = 'admin@freshfruit.com'
    ,@EmailSubject = 'Suspicious database behavior'
    ,@EmailBody = @EmailBodyText;
```

# Triggers with conditional logic

```
CREATE TRIGGER ConfirmStock
ON Orders
INSTEAD OF INSERT
AS
    IF EXISTS (SELECT * FROM Products AS p
               INNER JOIN inserted AS i ON i.Product = p.Product
               WHERE p.Quantity < i.Quantity)
        RAISERROR ('You cannot place orders when there is no product stock.', 16, 1);
    ELSE
        INSERT INTO dbo.Orders (Customer, Product, Quantity, OrderDate, TotalAmount)
        SELECT Customer, Product, Quantity, OrderDate, TotalAmount FROM inserted;
```



# Triggers with conditional logic



# Let's practice!

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER

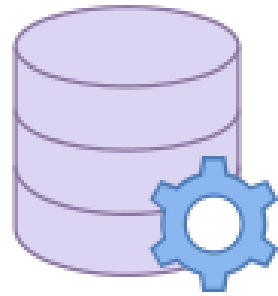
# Use cases for DDL triggers

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER



Florin Angelescu  
Instructor

# DDL trigger capabilities



## Database level

CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

CREATE\_VIEW, ALTER\_VIEW, DROP\_VIEW

CREATE\_INDEX, ALTER\_INDEX, DROP\_INDEX

ADD\_ROLE\_MEMBER, DROP\_ROLE\_MEMBER

CREATE\_STATISTICS, DROP\_STATISTICS



## Server level

CREATE\_DATABASE, ALTER\_DATABASE,  
DROP\_DATABASE

GRANT\_SERVER, DENY\_SERVER, REVOKE\_SERVER

CREATE\_CREDENTIAL, ALTER\_CREDENTIAL,  
DROP\_CREDENTIAL

# Database auditing

```
CREATE TRIGGER DatabaseAudit
ON DATABASE
FOR DDL_TABLE_VIEW_EVENTS
AS
    INSERT INTO [DatabaseAudit] ([EventType], [Database], [Object],
                                [UserAccount], [Query], [EventTime])
    SELECT
        EVENTDATA().value(' (/EVENT_INSTANCE/EventType)[1]', 'NVARCHAR(50)'),
        EVENTDATA().value(' (/EVENT_INSTANCE/DatabaseName)[1]', 'NVARCHAR(50)'),
        EVENTDATA().value(' (/EVENT_INSTANCE/ObjectName)[1]', 'NVARCHAR(100)'),
        EVENTDATA().value(' (/EVENT_INSTANCE/LoginName)[1]', 'NVARCHAR(100)'),
        EVENTDATA().value(' (/EVENT_INSTANCE/TSQLCommand/CommandText)[1]', 'NVARCHAR(MAX)'),
        EVENTDATA().value(' (/EVENT_INSTANCE/PostTime)[1]', 'DATETIME');
```

# Database auditing

EventType	Database	Object	UserAccount	Query	EventTime	
CREATE_TABLE	FreshFruit	Sales	XXX	CREATE TABLE [Sales]...	2019-05-13	
CREATE_TABLE	FreshFruit	Employees	XXX	CREATE TABLE [Employ...	2019-05-13	

# Preventing server changes

```
CREATE TRIGGER PreventDatabaseDelete
ON ALL SERVER
FOR DROP_DATABASE
AS
    PRINT 'You are not allowed to remove existing databases.';
ROLLBACK;
```

# Let's practice!

BUILDING AND OPTIMIZING TRIGGERS IN SQL SERVER