Programming SQL Server Database Triggers and Functions

VALIDATING AND MODIFYING DATA WITH DML TRIGGERS

Ryan Booz
AUTHOR AND SPEAKER

@ryanbooz https://www.softwareandbooz.com

DML Triggers

Understanding DML Triggers

Common Use Cases

INSTEAD OF and AFTER Triggers

INSERTED and DELETED Tables

Trigger Execution Order

What is a DML Trigger?

DML

Data Manipulation Language is a vocabulary of standard T-SQL commands that **attempt to** retrieve, modify or manipulate data.

SELECT, INSERT, UPDATE, DELETE, etc.

Data is stored in relational tables

DML Triggers watch for data manipulation events

INSERTING, UPDATING, or DELETING

React to those modifications

Triggers can also be created on Views

```
INSERT INTO OrderLines
   (StockItemId, Description, UnitPrice, Quantity)
      VALUES ('111','Furry Gorilla Slippers',11.99, 8)
```

ID	StockItemId	Description	UnitPrice	Quantity
1	135	USB Food Flash Drive	9.99	25
2	23	DBA Joke Mug	15.75	15
3	460	Superhero Jacket	35.68	3
4	144	Chocolate Frogs	15.33	10

```
INSERT INTO OrderLines
   (StockItemId, Description, UnitPrice, Quantity)
      VALUES ('111','Furry Gorilla Slippers',11.99, 8)
```

ID	StockItemId	Description	UnitPrice	Quantity
1	135	USB Food Flash Drive	9.99	25
2	23	DBA Joke Mug	15.75	15
3	460	Superhero Jacket	35.68	3
4	144	Chocolate Frogs	15.33	10
5	111	Furry Gorilla Slippers	11.99	12

```
UPDATE OrderLines SET
      Quantity = Quantity+1 WHERE ID > 2
```

ID	StockItemId	Description	UnitPrice	Quantity
1	135	USB Food Flash Drive	9.99	25
2	23	DBA Joke Mug	15.75	15
3	460	Superhero Jacket	35.68	3
4	144	Chocolate Frogs	15.33	10
5	111	Furry Gorilla Slippers	11.99	12

```
UPDATE OrderLines SET
      Quantity = Quantity+1 WHERE ID > 2
```

ID	StockItemId	Description	UnitPrice	Quantity
1	135	USB Food Flash Drive	9.99	25
2	23	DBA Joke Mug	15.75	15
3	460	Superhero Jacket	35.68	4
4	144	Chocolate Frogs	15.33	11
5	111	Furry Gorilla Slippers	11.99	13

```
DELETE FROM OrderLines WHERE StockItemId=460
```

ID	StockItemId	Description	UnitPrice	Quantity
1	135	USB Food Flash Drive	9.99	25
2	23	DBA Joke Mug	15.75	15
~~3~~	~~460~~	~~Superhero Jacket~~	~~35.68~~	~~4~~
4	144	Chocolate Frogs	15.33	11
5	111	Furry Gorilla Slippers	11.99	13

INSTEAD OF vs. AFTER Triggers

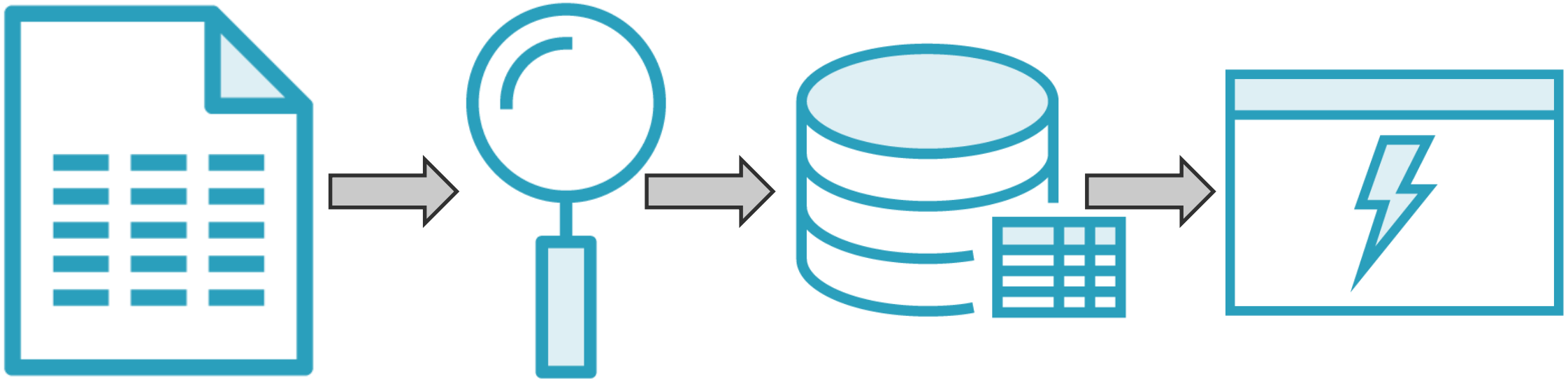
AFTER vs. INSTEAD OF Triggers

AFTER Triggers	INSTEAD OF Triggers
The default if neither type is specified	
Access to all new (INSERTED) and old (DELETED) data	Access to all new (INSERTED) and old (DELETED) data
Executed as part of the DML Transaction	Executed as part of the DML Transaction
Called AFTER all constraints have passed and data is modified	Called INSTEAD OF the actual DML, before constraints have been checked and data is modified
Data modification persists regardless of Trigger logic and action	Data is not modified unless the Trigger specifically completes the DML action

AFTER vs. INSTEAD OF Triggers

AFTER Triggers	INSTEAD OF Triggers
The default if neither type is specified	
Access to all new (INSERTED) and old (DELETED) data	Access to all new (INSERTED) and old (DELETED) data
Executed as part of the DML Transaction	Executed as part of the DML Transaction
Called AFTER all constraints have passed and data is modified	Called INSTEAD OF the actual DML, before constraints have been checked and data is modified
Data modification persists regardless of Trigger logic and action	Data is not modified unless the Trigger specifically completes the DML action

AFTER Triggers



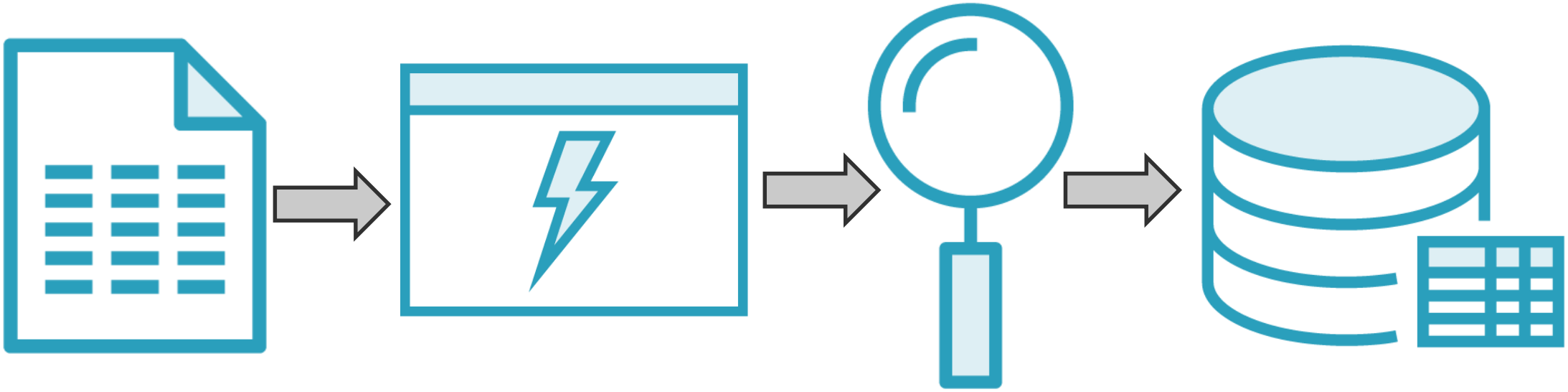
DML event that modifies data will first run all constraint checks

Data is modified in the table after constraints pass

AFTER Trigger is executed

INSERTED and DELETED data is passed to the Trigger

INSTEAD OF Triggers



DML event that modifies data will execute the INSTEAD OF Trigger **first**

The INSTEAD OF Trigger determines what happens with the DML data

Constraints are checked if the Trigger modifies data in the table

AFTER Triggers will execute if data modification is attempted by the INSTEAD OF trigger

“How do I choose between AFTER and INSTEAD OF Triggers?”

Consider an AFTER Trigger When...

Relying on constraints to ensure data is valid and saved

Modifying data or settings in other tables that rely on current data to be exist in table

There is no compelling reason to program the INSTEAD OF variant

Consider an INSTEAD OF Trigger When...

You require INSERT/UPDATE optimizations from reduced transaction overhead

Modifying known data issues that would otherwise prevent DML success

You have a complex VIEW that serves as a proxy to save data

Anatomy of DML Triggers

Anatomy of a Trigger

```
CREATE OR ALTER TRIGGER [Trigger Name]
ON [Table or View Name]
{FOR | AFTER | INSTEAD OF} INSERT
AS

    /*

        Insert business logic here…

    */

GO;
```

DML Triggers work on batches, not individual rows

```
INSERT INTO OrderLines
   (StockItemId, Description, UnitPrice, Quantity)
      VALUES ('111','Furry Gorilla Slippers',11.99, 8)
```

ID	StockItemId	Description	UnitPrice	Quantity
1	135	USB Food Flash Drive	9.99	25
2	23	DBA Joke Mug	15.75	15
3	460	Superhero Jacket	35.68	3
4	144	Chocolate Frogs	15.33	10
5	111	Furry Gorilla Slippers	11.99	12

```
UPDATE OrderLines SET
      Quantity = Quantity+1 WHERE ID > 2
```

ID	StockItemId	Description	UnitPrice	Quantity
1	135	USB Food Flash Drive	9.99	25
2	23	DBA Joke Mug	15.75	15
3	460	Superhero Jacket	35.68	4
4	144	Chocolate Frogs	15.33	11
5	111	Furry Gorilla Slippers	11.99	13

```
DELETE FROM OrderLines WHERE StockItemId=460
```

ID	StockItemId	Description	UnitPrice	Quantity
1	135	USB Food Flash Drive	9.99	25
2	23	DBA Joke Mug	15.75	15
~~3~~	~~460~~	~~Superhero Jacket~~	~~35.68~~	~~4~~
4	144	Chocolate Frogs	15.33	11
5	111	Furry Gorilla Slippers	11.99	13

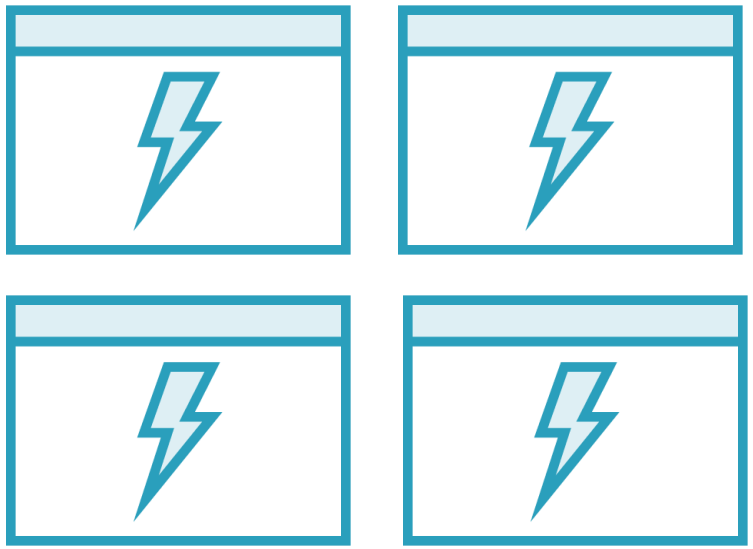
Triggers should NOT return data

```
DECLARE @newId INT;

SELECT * FROM INSERTED i
INNER JOIN T1 ON T1.A = i.A

SELECT * INTO #Changed FROM
(SELECT * FROM INSERTED
    EXCEPT
  SELECT * FROM DELETED)
ChangedData;


INSERT INTO Table1 (ID)
SELECT ID FROM INSERTED;

IF UPDATE(Column1)
BEGIN
    /* Do Something */
END;
```

- DECLARE variables to use
- JOIN to other tables
- Select ONLY the rows with modified data to use later in the trigger
- Modify another table
- Only do something if a specific column had an attempted update

Tables can have one
INSTEAD OF Trigger for
each DML event

Multiple AFTER Triggers can be defined on a table for each DML event



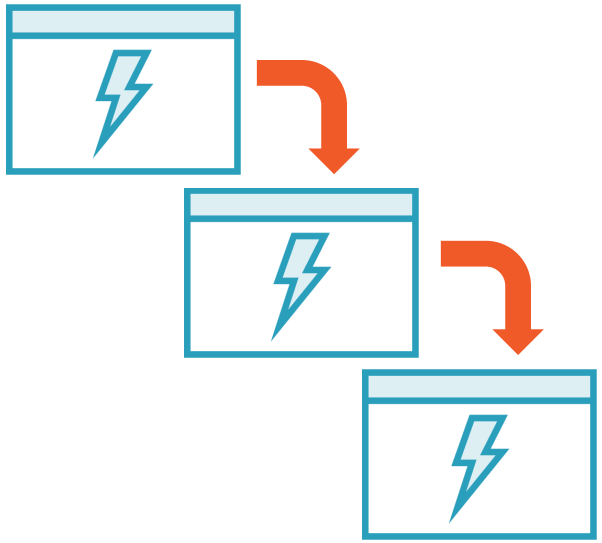
Separate business needs that might not exist in each client DB

Naming can help bring visibility into what a Trigger is doing

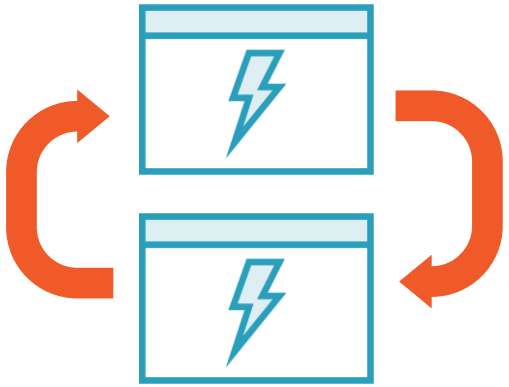
- TI_OrderLines
- TI_OrderLines_VerifyActive
- TD_OrderLines
- TD_OrderLines_LogDeletion

Triggers can be nested by default, up to 32 levels

Execution of Nested Triggers



Nested Triggers are enabled by default



Direct Recursive Triggers are supported with settings

All nested Triggers execute in the same transaction

When one fails, the entire transaction is rolled back

AFTER Trigger Execution Order

AFTER Trigger Execution Order

DML actions can have one
FIRST and one LAST trigger.

All other triggers are executed
in an unspecified order.

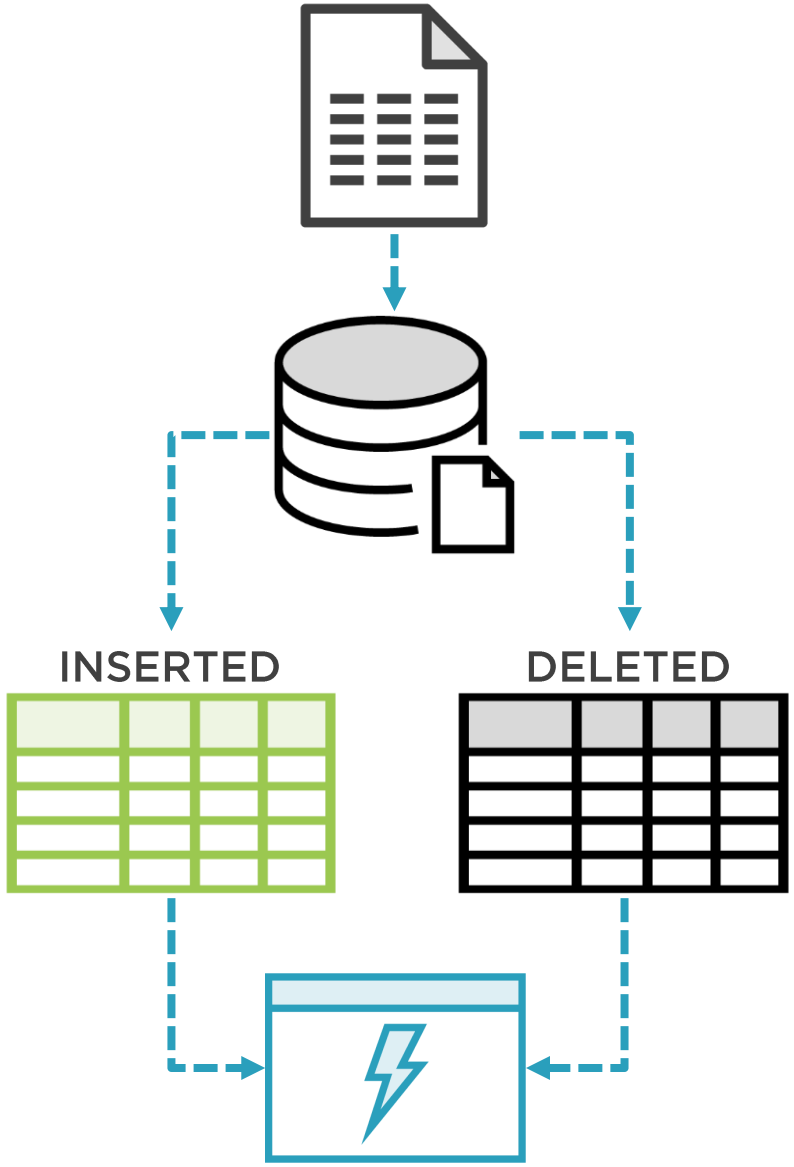
Any modification of a FIRST or LAST trigger will reset the order to NONE.

```
CREATE OR ALTER TRIGGER...

ALTER TRIGGER...
```

Modifying any Trigger specified as FIRST or LAST will reset the order to NONE

Utilizing the INSERTED and DELETED Tables

“How do I interact with the data that was modified?”



When a trigger is executed, virtual tables are passed in containing the modified data

INSERTED contains the “new” data

DELETED contains the “old” data

Update Triggers have both virtual tables

```
INSERT INTO OrderLines
   (StockItemId, Description, UnitPrice, Quantity)
      VALUES ('111','Furry Gorilla Slippers',11.99, 8)
```

ID	StockItemId	Description	UnitPrice	Quantity
1	135	USB Food Flash Drive	9.99	25
2	23	DBA Joke Mug	15.75	15
3	460	Superhero Jacket	35.68	3
4	144	Chocolate Frogs	15.33	10
5	111	Furry Gorilla Slippers	11.99	12

INSERTED Table

StockItemId	Description	UnitPrice	Quantity
111	Furry Gorilla Slippers	11.99	12

DELETED Table

StockItemId	Description	UnitPrice	Quantity

```
UPDATE OrderLines SET
      Quantity = Quantity+1 WHERE ID > 2
```

ID	StockItemId	Description	UnitPrice	Quantity
1	135	USB Food Flash Drive	9.99	25
2	23	DBA Joke Mug	15.75	15
3	460	Superhero Jacket	35.68	4
4	144	Chocolate Frogs	15.33	11
5	111	Furry Gorilla Slippers	11.99	13

INSERTED Table

ID	StockItemId	Description	UnitPrice	Quantity
3	460	Superhero Jacket	35.68	4
4	144	Chocolate Frogs	15.33	11
5	111	Furry Gorilla Slippers	11.99	13

DELETED Table

ID	StockItemId	Description	UnitPrice	Quantity
3	460	Superhero Jacket	35.68	3
4	144	Chocolate Frogs	15.33	10
5	111	Furry Gorilla Slippers	11.99	12

```
DELETE FROM OrderLines WHERE StockItemId=460
```

ID	StockItemId	Description	UnitPrice	Quantity
1	135	USB Food Flash Drive	9.99	25
2	23	DBA Joke Mug	15.75	15
~~3~~	~~460~~	~~Superhero Jacket~~	~~35.68~~	~~4~~
4	144	Chocolate Frogs	15.33	11
5	111	Furry Gorilla Slippers	11.99	13

INSERTED Table

ID	StockItemId	Description	UnitPrice	Quantity

DELETED Table

ID	StockItemId	Description	UnitPrice	Quantity
3	460	Superhero Jacket	35.68	3

Less is more!

Limit the work performed in Triggers

- Check that rows are actually modified
- Check for modifications of specific columns
- Find differences between the old and new data
- Keep transactions short lived
- Utilize Service Broker for longer running modifications

```
IF (ROWCOUNT_BIG() = 0)
```

```
IF NOT EXISTS (SELECT 1
FROM INSERTED)
```

```
IF UPDATE({ColumnName})
```

```
SELECT C1,C2 FROM INSERTED
EXCEPT
SELECT C1,C2 FROM DELETED
```

- Verify that rows were actually modified, especially in UPDATE Triggers
- Further checks for MERGE statements which return a count of all rows modified
- Did the DML event attempt to update a specific column on a table
- This will be true regardless of actual data modification
- Select the actual differences between what was inserted and what was already present

```
IF (ROWCOUNT_BIG() = 0)
```

```
IF NOT EXISTS (SELECT 1
FROM INSERTED)
```

```
IF UPDATE({ColumnName})
```

```
SELECT C1,C2 FROM INSERTED
EXCEPT
SELECT C1,C2 FROM DELETED
```

- Verify that rows were actually modified, especially in UPDATE Triggers
- Further checks for MERGE statements which return a count of all rows modified
- Did the DML event attempt to update a specific column on a table
- This will be true regardless of actual data modification
- Select the actual differences between what was inserted and what was already present

```
IF (ROWCOUNT_BIG() = 0)
```

```
IF NOT EXISTS (SELECT 1
FROM INSERTED)
```

```
IF UPDATE({ColumnName})
```

```
SELECT C1,C2 FROM INSERTED
EXCEPT
SELECT C1,C2 FROM DELETED
```

- Verify that rows were actually modified, especially in UPDATE Triggers
- Further checks for MERGE statements which return a count of all rows modified
- Did the DML event attempt to update a specific column on a table
- This will be true regardless of actual data modification
- Select the actual differences between what was inserted and what was already present

```
CREATE OR ALTER TRIGGER [Sales].[TIU_OrderLines]
ON [Sales].[OrderLines]AFTER INSERT, UPDATE
AS
BEGIN

    IF (ROWCOUNT_BIG() = 0)
        RETURN;

    -- Do not print any result details from Trigger
    SET NOCOUNT ON;

    IF NOT EXISTS (SELECT 1 FROM INSERTED)
        RETURN;

    /* Trigger Logic */

END;
```

Common Use Cases for DML Triggers

Common Use Cases – DML Triggers

Log information about DML events on data

Common Use Cases – DML Triggers

Log information about DML events on data

Insert or Modify data in another table

Common Use Cases – DML Triggers

Log information about DML events on data

Insert or Modify data in another table

Default Business Logic

Common Use Cases – DML Triggers

Log information about DML events on data

Insert or Modify data in another table

Default Business Logic

Mitigate impacts of schema changes using Views

Common Use Cases – DML Triggers

Log information about DML events on data

Insert or Modify data in another table

Default Business Logic

Mitigate impacts of schema changes using Views

Referential Integrity in legacy systems

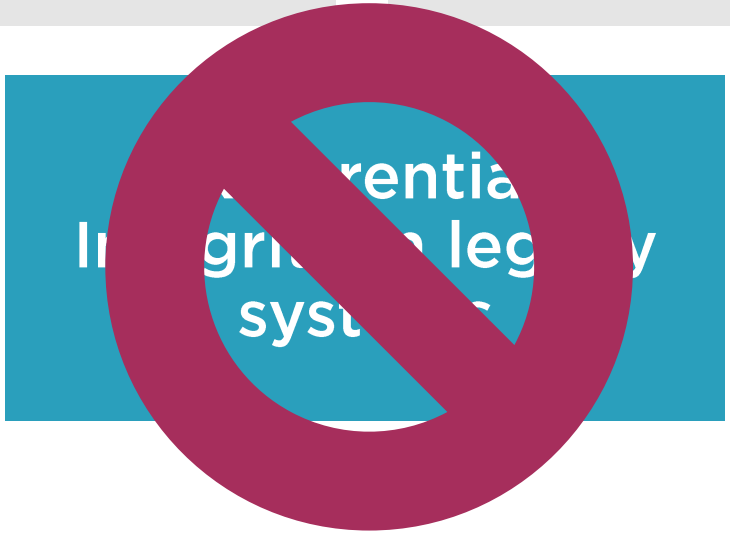
Common Use Cases – DML Triggers

Log information about DML events on data

Insert or Modify data in another table

Default Business Logic

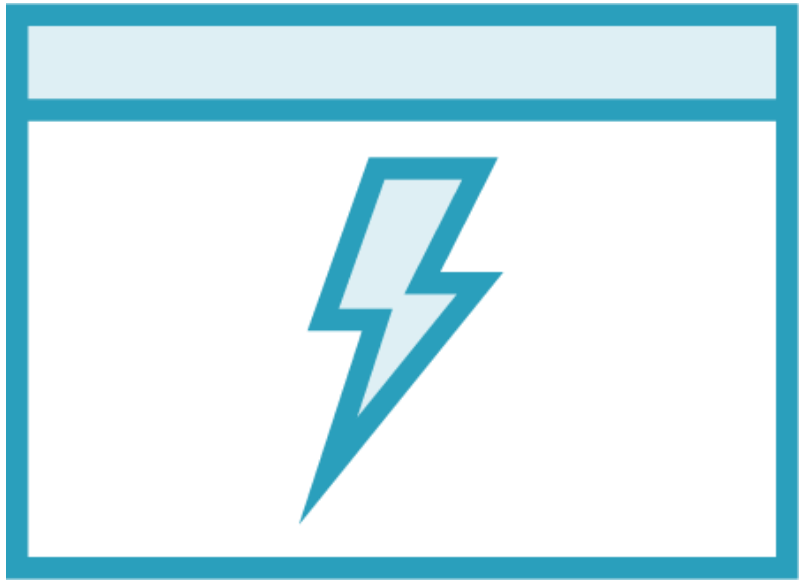
Mitigate impacts of schema changes using Views



Demo

INSERT AFTER Trigger

TI_Orders_AFTER



The DML action causes the Trigger to execute, not inserted data

All constraints have passed before the trigger is executed

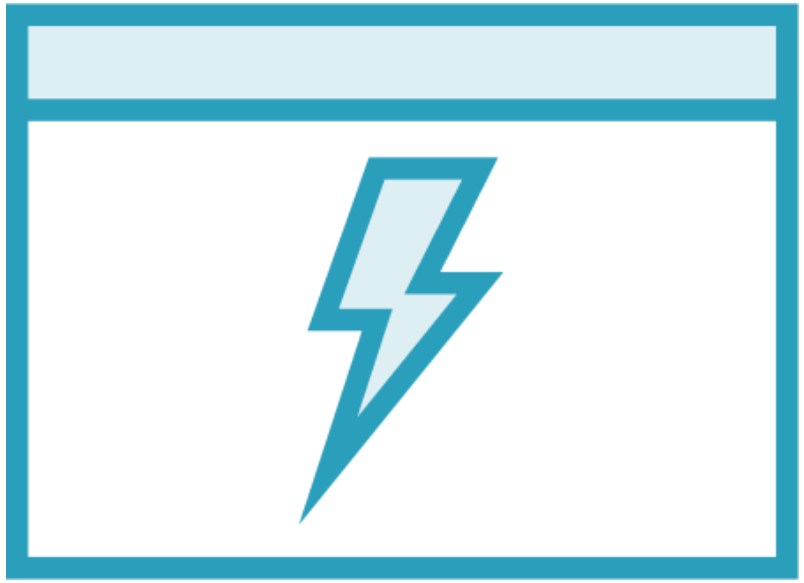
Any errors that cause a ROLLBACK will effect the entire DML action

Demo

Prevent incorrect data from being inserted based on business constraints

INSTEAD OF Trigger

TI_Orders_INSTEAD



INSTEAD OF Triggers act on behalf of the DML action

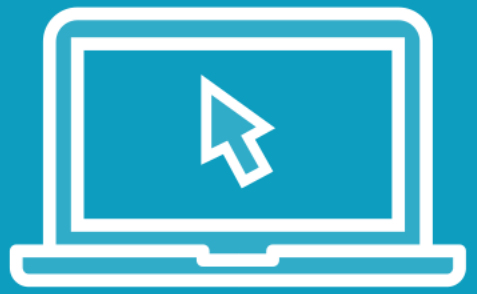
Execute before any AFTER trigger

No constraints have been checked

Allowed on VIEW's to proxy data to tables

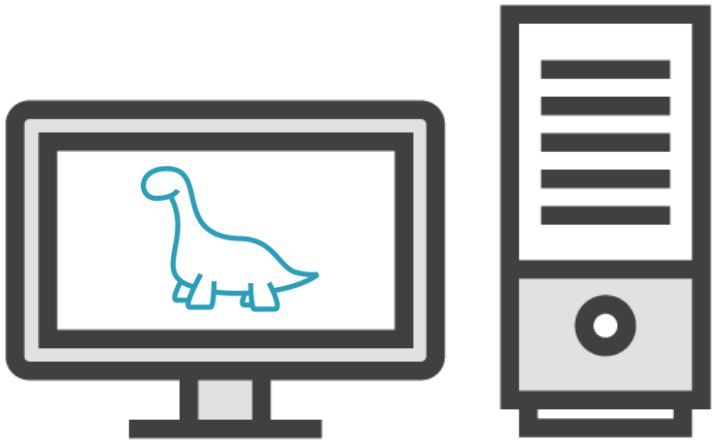
Any errors that cause a ROLLBACK will effect the entire DML action

Demo



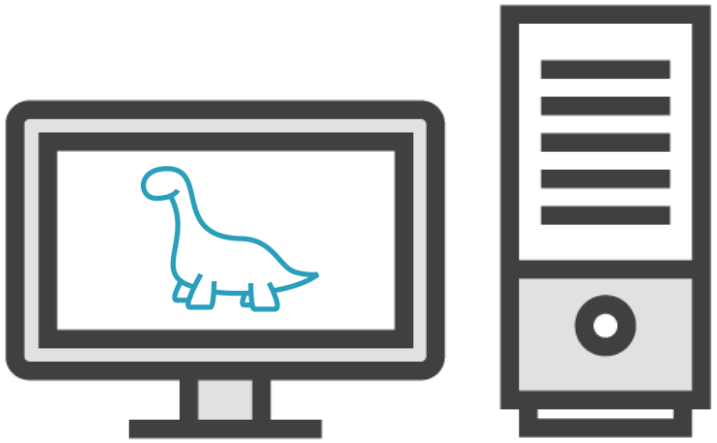
Correct data before it is INSERTED

UPDATE data thru a VIEW



CustomerID	125
SalesPersonID	10
OrderDate	01/01/2019
PurchaseOrderNumber	ABC123
Comments	Big Sale!

CustomerID	210
SalesPersonID	25
OrderDate	01/01/2019
PurchaseOrderNumber	ZXY456
Comments	Bigger Sale!
ExpectedDeliveryDate	01/08/2019



CustomerID	10
AddressLine1	100 Jones Road
AddressLine2	Suite 200
City	Bell
State	CA
Postal Code	90210

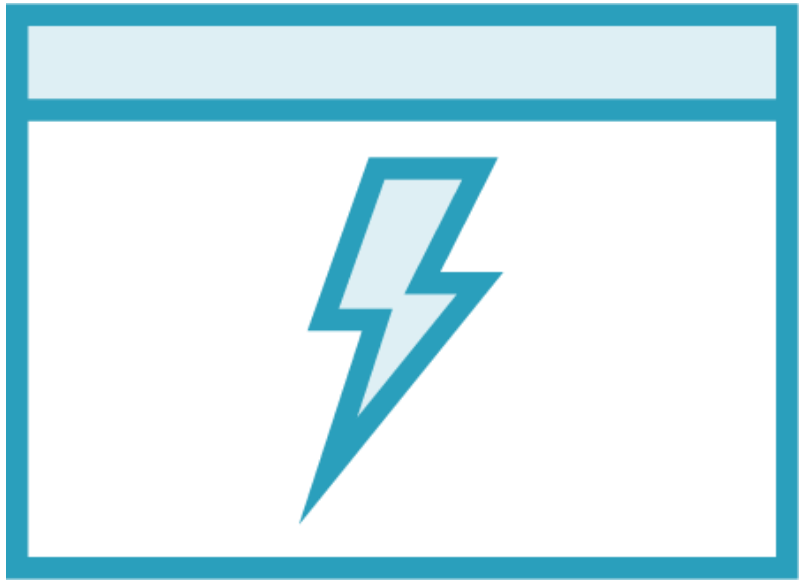
Customer
AddressLine1
AddressLine2
PostalCode
CityID

Cities
CityID
CityName
StateProvinceID

StateProvince
StateProvinceID
StateAbbreviation

DELETE AFTER Trigger

TD_Orders_AFTER



Access to the DELETED table

Data has already been deleted from the table, therefore joins will not work

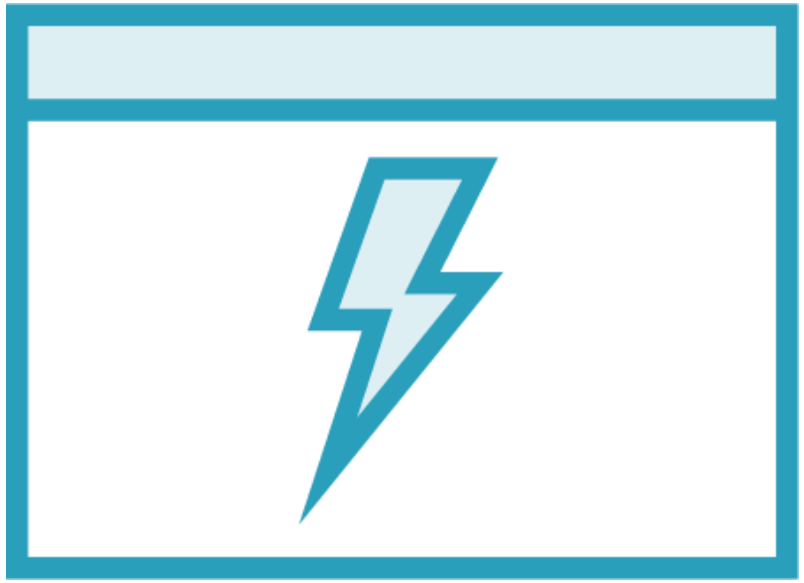
Demo

Prevent the deletion of data based on business constraints

Log data deletion through a custom audit table

UPDATE AFTER Trigger

TU_Orders_AFTER



Access to both INSERTED and DELETED virtual tables

All rows that had a modification will exist in both tables, regardless of data change

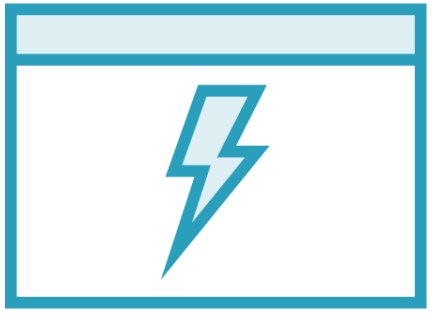
Save work by only acting on truly modified data

Demo

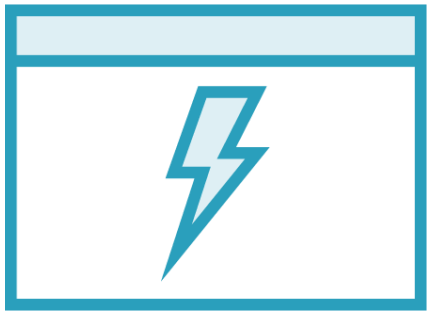
Log data modifications into a custom audit table

AFTER Trigger Order

FIRST



LAST



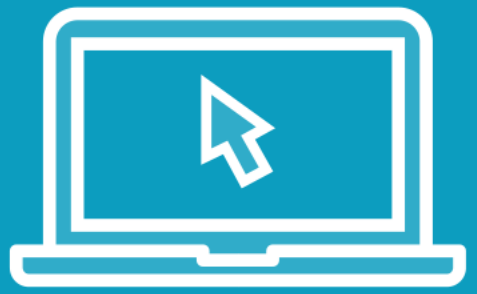
DML AFTER Trigger order is not guaranteed

Allowed to set one FIRST and one LAST of each DML action type

When one trigger modifies data that another relies on

Snapshots at the beginning or end of Trigger modifications

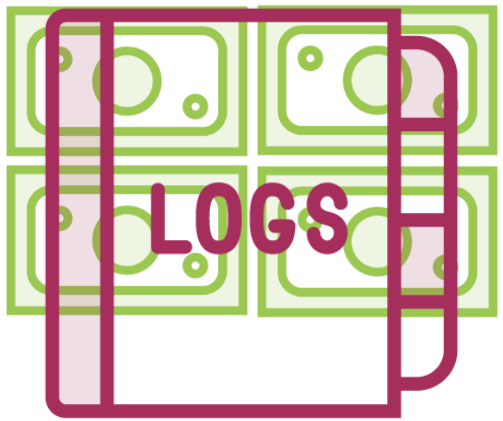
Demo



Use ‘sp_settriggerorder’ to specify FIRST and LAST trigger

Log data from another table on UPDATE before it is modified by another trigger

Sales Person 1



Sales Person 2

Summary

- What DML Triggers are
- AFTER vs. INSTEAD OF
- How to use the INSERTED and DELETED virtual tables
- Common use cases
- Logging data modifications
- Setting DML Trigger Order