**SQL Triggers**

A trigger is a procedure that executes when a database event occurs.

To create a trigger use the CREATE TRIGGER command.

Triggers can help enforce data and referential integrity in the database.

**Example**

[**#**](https://www.dofactory.com/sql/triggers#example)

This trigger recalculates the order amount after an orderitem row is changed

**CREATE TRIGGER OrderItemUpdate**

**ON**

**OrderItem**

**AFTER**

**UPDATE**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**DECLARE @OrderId INT;**

**SELECT @OrderId = INSERTED.OrderId**

**FROM INSERTED**

**UPDATE [Order]**

**SET TotalAmount = (SELECT SUM(UnitPrice \* Quantity)**

**FROM OrderItem**

**WHERE OrderId = @OrderId)**

**WHERE [Order].Id = @OrderId;**

**END**

Try it live

Once created, this trigger will automatically run after an OrderItem is updated.

**Using Triggers**

[**#**](https://www.dofactory.com/sql/triggers#using)

Triggers are procedures that automatically execute when a database event occurs.

These events include INSERT, UPDATE, and DELETE operations.

A trigger can execute AFTER or INSTEAD OF the action.

INSTEAD OF overrides the INSERT, UPDATE, and DELETE that triggered it.

[SQL Procedures](https://www.dofactory.com/sql/stored-procedures)

[SQL Functions](https://www.dofactory.com/sql/functions)

**Syntax**

[**#**](https://www.dofactory.com/sql/triggers#syntax)

Syntax to create a trigger.

**CREATE TRIGGER trigger\_name**

**ON**

**table\_name**

**AFTER | INSTEAD OF**

**[INSERT],[UPDATE],[DELETE]**

**AS**

**sql\_statements**

Syntax to change a trigger.

**ALTER TRIGGER trigger\_name**

**ON**

**table\_name**

**AFTER | INSTEAD OF**

**[INSERT],[UPDATE],[DELETE]**

**AS**

**sql\_statements**

Syntax to remove a trigger.

**DROP TRIGGER trigger\_name**

**More Examples**

[**#**](https://www.dofactory.com/sql/triggers#examples)

CREATE TRIGGER

|  |
| --- |
| **ORDERITEM** |
| Id |
| OrderId |
| ProductId |
| UnitPrice |
| Quantity |

Problem: Create a trigger that prevents changes to order items.

**CREATE TRIGGER OrderItemBlock**

**ON**

**OrderItem**

**INSTEAD OF**

**UPDATE, DELETE**

**AS**

**RAISERROR ('Cannot change existing orderitems',1,1)**

Try it live

Result:  Trigger created

ALTER TRIGGER

|  |
| --- |
| **ORDERITEM** |
| Id |
| OrderId |
| ProductId |
| UnitPrice |
| Quantity |

Problem: Change the error message in the OrderItemBlock trigger.

**ALTER TRIGGER OrderItemBlock**

**ON**

**OrderItem**

**INSTEAD OF**

**UPDATE, DELETE**

**AS**

**RAISERROR ('Illegal to change existing orders',1,1)**

Try it live

Result:  Trigger changed

DROP TRIGGER

|  |
| --- |
| **ORDERITEM** |
| Id |
| OrderId |
| ProductId |
| UnitPrice |
| Quantity |

Problem: Remove the OrderItemBlock trigger.

**DROP TRIGGER OrderItemBlock**

Try it live

Result:  Trigger dropped

RENAME TRIGGER

|  |
| --- |
| **ORDERITEM** |
| Id |
| OrderId |
| ProductId |
| UnitPrice |
| Quantity |

Problem: Rename the OrderItemBlock trigger to OrderItemPreventChange.

**DROP TRIGGER OrderItemBlock**

**CREATE TRIGGER OrderItemPreventChange**

**ON**

**OrderItem**

**INSTEAD OF**

**UPDATE, DELETE**

**AS**

**RAISERROR ('Cannot change existing orderitems',1,1)**

Try it live

Renaming a trigger is not available in SQL Server.  
To rename, drop the trigger and then re-create it with a new name.

Result:  Trigger dropped; trigger created

**Types of Triggers**

[**#**](https://www.dofactory.com/sql/triggers#types)

Triggers fall into 3 categories:

|  |  |
| --- | --- |
| TRIGGER | DESCRIPTION |
| DML | React to DML (Data Manipulation Language) queries, such as, INSERT, UPDATE, and DELETE |
| DDL | React to DDL (Data Definition Language) queries, such as, CREATE, ALTER, and DROP |
| Logon | Triggers when a LOGON event is detected |

DML are the triggers presented above.  
DDL and Logon Triggers are discussed next.

**DDL Triggers**

[**#**](https://www.dofactory.com/sql/triggers#ddl)

DDL Triggers respond to changes made to the database schema.

Events include CREATE, ALTER, DROP, GRANT, DENY, REVOKE actions.

A common use is to prevent certain changes to the data model.

Another use is to audit data model changes and log these into a log-table.

**DDL Trigger Syntax**

[**#**](https://www.dofactory.com/sql/triggers#ddl-syntax)

Syntax to create a DDL trigger.

**CREATE TRIGGER trigger\_name**

**ON**

**DATABASE | ALL SERVER**

**FOR**

**event\_type | event\_group**

**AS**

**sql\_statement**

There are literally dozens of event\_types and event\_groups.

Examples of event\_type include: CREATE\_TABLE or DROP\_INDEX.

Examples of event\_group include: DDL\_TABLE\_EVENTS or DDL\_DATABASE\_SECURITY\_EVENTS.

**DDL Trigger Examples**

[**#**](https://www.dofactory.com/sql/triggers#ddl-examples)

CREATE DDL TRIGGER

Problem: Log all database index modifications to a log table.

**CREATE TRIGGER TriggerIndexChange**

**ON**

**DATABASE**

**FOR**

**CREATE\_INDEX, ALTER\_INDEX, DROP\_INDEX**

**AS**

**BEGIN**

**SET NOCOUNT ON;**

**INSERT INTO IndexLog (EventData, ChangedBy)**

**VALUES (EVENTDATA(), USER)**

**END**

Try it live

This trigger assumes a log table named IndexLog has been created.

The EVENTDATA() function returns information about the database event.

Anytime an index changes, this trigger will add a new entry to the log table.

CREATE DDL TRIGGER with ROLLBACK

Problem: Prevent users from making table schema changes. Rollback any attempt.

**CREATE TRIGGER TriggerTableRestrictions**

**ON**

**DATABASE**

**FOR**

**CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE**

**AS**

**BEGIN**

**PRINT 'No table modifications can be made'**

**ROLLBACK**

**END**

Try it live

This trigger responds to all table schema changes.

An error message displays and the table change is rolled back (undone).

**Logon Triggers**

[**#**](https://www.dofactory.com/sql/triggers#logon)

A logon trigger fires when a new database connection is established.

Common use cases include:

* Auditing logon activity
* Restricting number of concurrent login sessions
* Restricting the total number of connections
* Restricting login by hostname or IP address

**Logon Triggers Syntax**

[**#**](https://www.dofactory.com/sql/triggers#logon-syntax)

Syntax to create a Logon trigger.

**CREATE TRIGGER trigger\_name**

**ON**

**ALL SERVER**

**FOR**

**LOGON**

**AS**

**BEGIN**

**sql\_statement**

**END**

**Trigger Use Cases**

[**#**](https://www.dofactory.com/sql/triggers#use-cases)

These are some use cases where triggers can be very helpful:

Enforce referencial integrity across the database

Event logging and auditing

Generate derived or calculated column values instantly

Replicate tables synchronously

Impose security authorizations

Prevent invalid transactions

**Trigger Considerations**

[**#**](https://www.dofactory.com/sql/triggers#considerations)

Things to consider before using triggers.

Triggers can negatively affect performance.

Triggers are difficult to debug and can cause hard to solve bugs.

Debugging nested and recursive triggers is extremely hard.

Triggers are easily forgotten, leaving developers scratching their heads.

As a general rule it is best to use triggers sparingly.

**List all Triggers**

[**#**](https://www.dofactory.com/sql/triggers#list)

This query will list all triggers in the database.

**SELECT \***

**FROM sys.triggers**

**WHERE type = 'TR';**

Triggers are easily forgotten, so a list of triggers can be useful when trying to identify a mysterious bug in the database.