LAB 9_Trigger

A. Tutorial.

A trigger is a special type of stored procedure that automatically executes when an event occurs in the database server. DML triggers execute when a user tries to modify data through a data manipulation language (DML) event. DML events are INSERT, UPDATE, or DELETE statements on a table or view. These triggers fire when any valid event is fired, regardless of whether or not any table rows are affected.

Syntax:

```
CREATE [ OR ALTER ] TRIGGER [ schema_name . ]trigger_name
ON { table | view }
[ WITH <dml_trigger_option> [ ,...n ] ]
{ FOR | AFTER | INSTEAD OF }
{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }
[ WITH APPEND ]
[ NOT FOR REPLICATION ]
AS { sql_statement  [ ; ] [ ,...n ] | EXTERNAL NAME <method specifier [ ; ] > }

<dml_trigger_option> ::=
    [ ENCRYPTION ]
    [ EXECUTE AS Clause ]

<method_specifier> ::=
    assembly_name.class_name.method_name
```

Arguments:

```
OR ALTER
```

Applies to: Azure SQL Database, SQL Server (starting with SQL Server 2016 (13.x) SP1).

Conditionally alters the trigger only if it already exists.

schema_name

Is the name of the schema to which a DML trigger belongs. DML triggers are scoped to the schema of the table or view on which they are created. *schema_name* cannot be specified for DDL or logon triggers.

trigger_name

Is the name of the trigger. A *trigger_name* must comply with the rules for identifiers, except that *trigger_name* cannot start with # or ##.

table | view

Is the table or view on which the DML trigger is executed and is sometimes referred to as the trigger table or trigger view. Specifying the fully qualified name of the table or view is optional. A view can be referenced only by an INSTEAD OF trigger. DML triggers cannot be defined on local or global temporary tables.

DATABASE

Applies the scope of a DDL trigger to the current database. If specified, the trigger fires whenever *event_type* or *event_group* occurs in the current database.

ALL SERVER

Applies to: SQL Server 2008 through SQL Server 2017.

Applies the scope of a DDL or logon trigger to the current server. If specified, the trigger fires whenever *event_type* or *event_group* occurs anywhere in the current server.

WITH ENCRYPTION

Applies to: SQL Server 2008 through SQL Server 2017.

Obfuscates the text of the CREATE TRIGGER statement. Using WITH ENCRYPTION prevents the trigger from being published as part of SQL Server replication. WITH ENCRYPTION cannot be specified for CLR triggers.

EXECUTE AS

Specifies the security context under which the trigger is executed. Enables you to control which user account the instance of SQL Server uses to validate permissions on any database objects that are referenced by the trigger.

This option is required for triggers on memory-optimized tables.

FOR | AFTER

AFTER specifies that the DML trigger is fired only when all operations specified in the triggering SQL statement have executed successfully. All referential cascade actions and constraint checks also must succeed before this trigger fires.

AFTER is the default when FOR is the only keyword specified.

AFTER triggers cannot be defined on views.

INSTEAD OF

Specifies that the DML trigger is executed *instead of* the triggering SQL statement, therefore, overriding the actions of the triggering statements. INSTEAD OF cannot be specified for DDL or logon triggers.

At most, one INSTEAD OF trigger per INSERT, UPDATE, or DELETE statement can be defined on a table or view. However, you can define views on views where each view has its own INSTEAD OF trigger.

INSTEAD OF triggers are not allowed on updatable views that use WITH CHECK OPTION. SQL Server raises an error when an INSTEAD OF trigger is added to an updatable view WITH CHECK OPTION specified. The user must remove that option by using ALTER VIEW before defining the INSTEAD OF trigger.

{ [DELETE] [,] [INSERT] [,] [UPDATE] }

Specifies the data modification statements that activate the DML trigger when it is tried against this table or view. At least one option must be specified. Any combination of these options in any order is allowed in the trigger definition.

For INSTEAD OF triggers, the DELETE option is not allowed on tables that have a referential relationship specifying a cascade action ON DELETE. Similarly, the UPDATE option is not allowed on tables that have a referential relationship specifying a cascade action ON UPDATE.

Example:

```
CREATE TRIGGER reminder1
```

```
ON Sales.Customer
AFTER INSERT, UPDATE
AS RAISERROR ('Notify Customer Relations', 16, 10);
GO
```

B. Exercise

A. Use the database WorldEvents.

1. Create a trigger to write any country changes made into a separate table.
2. create a trigger to run after anyone changes a country.
3. Create a trigger to stop anyone deleting events which happened in the UK (country number 7). To do this, create a trigger which operates on the tblEvent table.

B. Use the database SoftUni.

4. Create another table – Logs (LogId, AccountId, OldSum, NewSum). Add a trigger to the Accounts table that enters a new entry into the Logs table every time the sum on an account changes.

Example

LogId	AccountId	OldSum	NewSum
1	1	123.12	113.12
…	…	…	…

5. Create another table – **NotificationEmails**(Id, Recipient, Subject, Body). Add a trigger to logs table and **create new email whenever new record is inserted in logs table.** The following data is required to be filled for each email:

   Recipient – AccountId

   Subject – "Balance change for account: **{AccountId}**"

   Body - "On **{date}** your balance was changed from **{old}** to **{new}.**"

Example

Id	Recipient	Subject	Body
1	1	Balance change for account: 1	On Sep 12 2016 2:09PM your balance was changed from 113.12 to 103.12.
…	…	…	…

6. Users should not be allowed to buy items with higher level than their level. Create a trigger that restricts that. Add bonus cash of 50000 to users: **baleremuda, loosenoise, inguinalself, buildingdeltoid, monoxidecos** in the game **"Bali".**

   There are two groups of items that you should buy for the above users in the game. First group is with **id between 251 and 299 including**. Second group is with **id between 501 and 539 including.**

   Take off cash from each user for the bought items.

   Select all users in the current game with their items. Display **username**, **game name**, **cash** and **item name**. Sort the result by username alphabetically, then by item name alphabetically.

Output

Username	Name	Cash	Item Name
baleremuda	Bali	4****.**	Iron Wolves Doctrine
baleremuda	Bali	4****.**	Irontoe Mudsputters
…	…	…	…
buildingdeltoid	Bali	3****.**	Alabaster Gloves
…	…	…	…