(T29)介紹DDLTrigger搭配AllServerScope和LoginTrigger  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T29)介紹DDLTrigger搭配AllServerScope和LoginTrigger  
=======================================================================  
0. Summary

-----------

1. Database Scoped Data Definition Language (DDL) Triggers event : CREATE\_TABLE, ALTER\_TABLE , DROP\_TABLE, RENAME

1.1. DDL Trigger event: CREATE\_TABLE

1.2. Enable/Disable DDL Trigger : CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

1.3. DATABASE scoped DDL Trigger : RENAME

1.4. Clean up

-----------

2. Server Scoped Data Definition Language (DDL) Triggers event : CREATE\_TABLE, ALTER\_TABLE , DROP\_TABLE

-----------

3. TriggerExecutionOrder\_sp\_settriggerorder

3.1. sp\_settriggerorder

3.2. sp\_settriggerorder order

-----------

4. AuditTableChanges

4.1. EVENTDATA

4.2. AuditTableStructureChanges

4.3. Clean up

-----------

5. LogonTriggers\_sys.dm\_exec\_sessions\_ORIGINAL\_LOGIN()  
=======================================================================

0. Summary

1.

DDL Trigger Reference:

<https://docs.microsoft.com/en-us/sql/relational-databases/triggers/ddl-events>

Drop DDL trigger(ALL Server scope/Database scope) if it exists

<http://sqlhints.com/2016/04/03/how-to-check-if-a-trigger-exists-in-sql-server/>

Syntax

--CREATE TRIGGER [TriggerName]

--ON (All Server/Database)

--FOR [EventType1, EventType2, ...],

--AS

--BEGIN

--   ...

--END

-----------------------------------------------------------

2.

Database Scope DDL Trigger

2.1.

Create Database Scope DDL Trigger

Syntax

--CREATE TRIGGER [TriggerName]

--ON Database

--FOR [EventType1, EventType2, ...],

--AS

--BEGIN

--   ...

--END

2.1.1.

E.g.

--IF EXISTS ( SELECT  \*

--            FROM    sys.triggers

--            WHERE   name = 'trgNoCreateAlterDropTable' )

--    BEGIN

--        DROP TRIGGER trgNoCreateAlterDropTable ON DATABASE;

--    END;

--GO -- Run the previous command and begins new batch

--CREATE TRIGGER trgNoCreateAlterDropTable ON DATABASE

--    FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

--AS

--    BEGIN

--        PRINT 'Create table is prohibited.';

--        ROLLBACK;

--    END;

--GO -- Run the previous command and begins new batch

2.1.2.

Create DDL Database Triggers in SSMS

Database Name --> Programmability --> Database Triggers

-----------------------

2.2.

Enable/Sisable Trigger

Syntax

--DISABLE TRIGGER trgName ON DATABASE

Disable trgName DATABASE scope trigger.

--ENABLE TRIGGER trgName ON DATABASE

Enable trgName DATABASE scope trigger.

E.g.

--DISABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE

--ENABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE

------------------------------

2.3.

----Drop database scope trigger if it exists

--IF EXISTS ( SELECT  \*

--            FROM    sys.triggers

--            WHERE   name = 'trgNoCreateAlterDropTable' )

--    BEGIN

--        DROP TRIGGER trgNoCreateAlterDropTable ON DATABASE;

--    END;

--GO -- Run the previous command and begins new batch

-----------------------------------

2.4.

--CREATE TRIGGER trgRename ON DATABASE

--    FOR RENAME

--AS

--    BEGIN

--        PRINT 'Rename DDL trigger is fired.';

--    END;

--GO -- Run the previous command and begins new batch

--...

--sp\_rename 'TestTable', 'TestTable2';

Rename 'TestTable' to 'TestTable2'

--sp\_rename 'TestTable2.ID' , 'ID2', 'column'

Rename TestTable2.ID Column to TestTable2.ID2 Column,

the third parameter means dealing with column.

----------------------------------------------------------------------

3.

All Server Scope DDL Trigger

3.1.

Create All Server Scope DDL Trigger

Syntax

--CREATE TRIGGER [TriggerName]

--ON All Server

--FOR [EventType1, EventType2, ...],

--AS

--BEGIN

--   ...

--END

3.1.1.

E.g.

--IF EXISTS ( SELECT  \*

--            FROM    sys.server\_triggers

--            WHERE   name = 'trgNoCreateAlterDropTable2' )

--    BEGIN

--        DROP TRIGGER trgNoCreateAlterDropTable2 ON DATABASE;

--    END;

--GO -- Run the previous command and begins new batch

--CREATE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER

--    FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

--AS

--    BEGIN

--        PRINT 'Any definition change for table is not prohibited.';

--        ROLLBACK;

--    END;

--GO -- Run the previous command and begins new batch

The scope is ALL SERVER.

This is the CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE event trigger.

3.1.2.

Create DDL Database scope Triggers in SSMS

Database Name --> Programmability --> Database Triggers

Create DDL All Server scope Triggers in SSMS

Server Objects --> Triggers --> ...

-----------------------------------

3.2.

Enable/Sisable ALL SERVER scope DDL trigger

Syntax

--DISABLE TRIGGER trgName ON ALL SERVER

Disable trgName All Server scope trigger.

--ENABLE TRIGGER trgName ON ALL SERVER

Enable trgName All Server scope trigger.

E.g.

--DISABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER

--ENABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER

-----------------------------------

3.3.

Drop ALL SERVER scope trigger if it exists

Reference: <http://sqlhints.com/2016/04/03/how-to-check-if-a-trigger-exists-in-sql-server/>

--IF EXISTS ( SELECT  \*

--            FROM    sys.server\_triggers

--            WHERE   name = 'trgNoCreateAlterDropTable2' )

--    BEGIN

--        DROP TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER;

--    END;

--GO -- Run the previous command and begins new batch

-----------------------------------------------------------------------------

4.

4.1.

Even if we use sp\_settriggerorder to change the order,

server scoped triggers will always fire before any of the database scoped triggers.

4.2.

sp\_settriggerorder can set the order of server scoped DDL triggers

or database scoped triggers .

4.3.

E.g.

--EXEC sp\_settriggerorder

--    @triggername = 'trgFirstTrigger',

--    @order = 'first',

--  @stmttype = 'CREATE\_TABLE',

--    @namespace = 'DATABASE';

Set the database scoped trigger, trgFirstTrigger,

to the first order of CREATE\_TABLE event.

4.3.1.

--    @triggername = 'trgFirstTrigger',

1st parameter is the trigger name that you want to set.

4.3.2.

--@order = 'first',

--@order = 'last',

--@order = 'none',

2nd parameter is the running order.

Value can be First, Last or None.

4.3.3.

--  @stmttype = 'CREATE\_TABLE',

3rd parameter is the SQL statement that fires the trigger.

You many only put ONE statement type here.

If you want 'ALTER\_TABLE' or 'DROP\_TABLE' or other statement,

you need to "ECECUTE sp\_settriggerorder" several times

for each statement type you want to set.

4.3.4.

-- @namespace = 'DATABASE';

or

-- @namespace = 'SERVER';

4th parameter is the scope of the trigger.

Value can be DATABASE, SERVER, or NULL.

----------------------------------------------------

4.4.

If there is a database scoped DDL Triggers and a server scoped DDL triggers

handling the same event.

Here is the execution order.

4.4.1.

The server scope DDL trigger which @order = 'first'

-->

other server scope DDL triggers which @order = 'none'

-->

The server scope DDL trigger which @order = 'Last'

-->

The database scope DDL trigger which @order = 'first'

-->

other database scope DDL triggers which @order = 'none'

-->

The database scope DDL trigger which @order = 'Last'

-----------------------------------------------------------------------------

5.

--EVENTDATA()

5.1.

--EVENTDATA()

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/eventdata-transact-sql>

EventData() returns information about server or database events in XML format.

E.g.

--<EVENT\_INSTANCE>

--  <EventType>CREATE\_TABLE</EventType>

--  <PostTime>2017-10-10T04:42:27.870</PostTime>

--  <SPID>54</SPID>

--  <ServerName>N550JKL\SQL2016</ServerName>

--  <LoginName>MicrosoftAccount\[lpmplpmp01@hotmail.com](mailto:lpmplpmp01@hotmail.com)</LoginName>

--  <UserName>dbo</UserName>

--  <DatabaseName>Sample3</DatabaseName>

--  <SchemaName>dbo</SchemaName>

--  <ObjectName>TestTable</ObjectName>

--  <ObjectType>TABLE</ObjectType>

--  <TSQLCommand>

--    <SetOptions ANSI\_NULLS="ON" ANSI\_NULL\_DEFAULT="ON" ANSI\_PADDING="ON" QUOTED\_IDENTIFIER="ON" ENCRYPTED="FALSE" />

--    <CommandText>CREATE TABLE TestTable

--    (

--      ID INT ,

--      [Name] NVARCHAR(100)

--    )</CommandText>

--  </TSQLCommand>

--</EVENT\_INSTANCE>

-----------------------------------------------------

5.2.

----Drop Table if it exists

--IF ( EXISTS ( SELECT    \*

--              FROM      INFORMATION\_SCHEMA.TABLES

--              WHERE     TABLE\_NAME = 'AuditTableStructureChanges' ) )

--    BEGIN

--        TRUNCATE TABLE dbo.AuditTableStructureChanges;

--        DROP TABLE AuditTableStructureChanges;

--    END;

--GO -- Run the previous command and begins new batch

--CREATE TABLE AuditTableStructureChanges

--    (

--      EventType NVARCHAR(300) ,

--      PostTime DATETIME ,

--      --Server Process ID

--      SPID NVARCHAR(300) ,

--      ServerName NVARCHAR(300) ,

--      LoginName NVARCHAR(300) ,

--      UserName NVARCHAR(300) ,

--      DatabaseName NVARCHAR(300) ,

--      SchemaName NVARCHAR(300) ,

--      ObjectName NVARCHAR(300) ,

--      ObjectType NVARCHAR(300) ,

--      TSQLCommand NVARCHAR(MAX)

--    );

--GO

-----------------------------------------------------

----Drop ALL SERVER scope trigger if it exists

----Reference: <http://sqlhints.com/2016/04/03/how-to-check-if-a-trigger-exists-in-sql-server/>

--IF EXISTS ( SELECT  \*

--            FROM    sys.server\_triggers

--            WHERE   name = 'trgAuditTableStructureChanges' )

--    BEGIN

--        DROP TRIGGER trgAuditTableStructureChanges ON ALL SERVER;

--    END;

--GO -- Run the previous command and begins new batch

--CREATE TRIGGER trgAuditTableStructureChanges ON ALL SERVER

--    FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

--AS

--    BEGIN

--        DECLARE @EventData XML;

--        SELECT  @EventData = EVENTDATA();

--        INSERT  INTO Sample3.dbo.AuditTableStructureChanges

--                ( EventType ,

--                  PostTime ,

--                  SPID ,

--                  ServerName ,

--                  LoginName ,

--                  UserName ,

--                  DatabaseName ,

--                  SchemaName ,

--                  ObjectName ,

--                  ObjectType ,

--                  TSQLCommand

--                )

--        VALUES  ( @EventData.value('(/EVENT\_INSTANCE/EventType)[1]',

--                                   'NVARCHAR(300)') ,

--                  @EventData.value('(/EVENT\_INSTANCE/PostTime)[1]', 'DATETIME') ,

--                  @EventData.value('(/EVENT\_INSTANCE/SPID)[1]',

--                                   'NVARCHAR(300)') ,

--                  @EventData.value('(/EVENT\_INSTANCE/ServerName)[1]',

--                                   'NVARCHAR(300)') ,

--                  @EventData.value('(/EVENT\_INSTANCE/LoginName)[1]',

--                                   'NVARCHAR(300)') ,

--                  @EventData.value('(/EVENT\_INSTANCE/UserName)[1]',

--                                   'NVARCHAR(300)') ,

--                  @EventData.value('(/EVENT\_INSTANCE/DatabaseName)[1]',

--                                   'NVARCHAR(300)') ,

--                  @EventData.value('(/EVENT\_INSTANCE/SchemaName)[1]',

--                                   'NVARCHAR(300)') ,

--                  @EventData.value('(/EVENT\_INSTANCE/ObjectName)[1]',

--                                   'NVARCHAR(300)') ,

--                  @EventData.value('(/EVENT\_INSTANCE/ObjectType)[1]',

--                                   'NVARCHAR(300)') ,

--                  @EventData.value('(/EVENT\_INSTANCE/TSQLCommand)[1]',

--                                   'NVARCHAR(MAX)')

--                );

--    END;

--GO -- Run the previous command and begins new batch

==================================================

1. Database Scoped Data Definition Language (DDL) Triggers event : CREATE\_TABLE, ALTER\_TABLE , DROP\_TABLE, RENAME

--============================================================================

--T029\_01\_Database Scoped Data Definition Language (DDL) Triggers event : CREATE\_TABLE, ALTER\_TABLE , DROP\_TABLE, RENAME

--============================================================================

1.1. DDL Trigger event: CREATE\_TABLE

--============================================================================

--T029\_01\_01

--DDL Trigger event: CREATE\_TABLE

----------------------------------------------------------------------------

--T029\_01\_01\_01

--Create DDL Trigger : CREATE\_TABLE

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgNoCreateAlterDropTable' )

    BEGIN

        DROP TRIGGER trgNoCreateAlterDropTable ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

CREATE TRIGGER trgNoCreateAlterDropTable ON DATABASE

    FOR CREATE\_TABLE

AS

    BEGIN

        PRINT 'Create table is prohibited.';

        ROLLBACK;

    END;

GO -- Run the previous command and begins new batch

/\*

1.

--CREATE TRIGGER trgNoCreateAlterDropTable ON DATABASE

--    FOR CREATE\_TABLE

--AS

--    BEGIN

--        PRINT 'Create table is prohibited.';

--        ROLLBACK;

--    END;

prohibit create table in Sample3 database.

1.1.

Syntax

--CREATE TRIGGER [TriggerName]

--ON (All Server/Database)

--FOR [EventType1, EventType2, ...],

--AS

--BEGIN

--   ...

--END

1.2.

--CREATE TRIGGER trgNoCreateAlterDropTable ON DATABASE

--    FOR CREATE\_TABLE

Create a trigger which name as trgNoCreateAlterDropTable

The scope is current database.

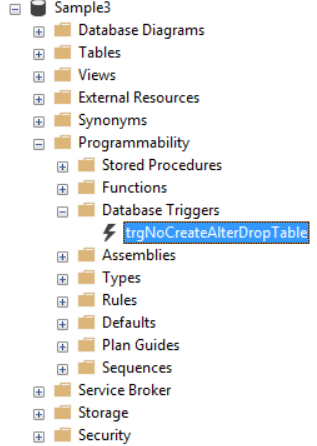
This is the CREATE\_TABLE event trigger.

1.3.

Create DDL Database Triggers in SSMS

Database Name --> Programmability --> Database Triggers

\*/



----------------------------------------------------------------------------

--T029\_01\_01\_02

--Test DDL Trigger CREATE\_TABLE

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TestTable ( ID INT PRIMARY KEY );

GO -- Run the previous command and begins new batch

/\*

Output Error Message

--Create table is prohibited.

--Msg 3609, Level 16, State 2, Line 88

--The transaction ended in the trigger. The batch has been aborted.

\*/

Company name

Description automatically generated with low confidence

1.2. Enable/Disable DDL Trigger : CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

--============================================================================

--T029\_01\_02

--Enable/Disable DDL Trigger : CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

----------------------------------------------------------------------

--T029\_01\_02\_01

--Create DDL Trigger : CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

ALTER TRIGGER trgNoCreateAlterDropTable ON DATABASE

    FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

AS

    BEGIN

        PRINT 'Any change for table is not prohibited.';

        ROLLBACK;

    END;

GO -- Run the previous command and begins new batch

/\*

1.

--ALTER TRIGGER trgNoCreateAlterDropTable ON DATABASE

--    FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

--AS

--    BEGIN

--        PRINT 'Any change for table is not prohibited.';

--        ROLLBACK;

--    END;

prohibit create/Alter/Drop table in Sample3 database.

1.1.

Syntax

--CREATE TRIGGER [TriggerName]

--ON (All Server/Database)

--FOR [EventType1, EventType2, ...],

--AS

--BEGIN

--   ...

--END

1.2.

--ALTER TRIGGER trgNoCreateAlterDropTable ON DATABASE

--    FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

Alter a trigger which name as trgNoCreateAlterDropTable

The scope is current database.

This is the CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE event trigger.

1.3.

Create DDL Database Triggers in SSMS

Database Name --> Programmability --> Database Triggers

\*/

----------------------------------------------------------------------

--T029\_01\_02\_02

--Enable/Disable DDL Trigger : CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

/\*

Enable/Sisable Trigger

Syntax

--DISABLE TRIGGER trgName ON DATABASE

Disable trgName DATABASE scope trigger.

--ENABLE TRIGGER trgName ON DATABASE

Enable trgName DATABASE scope trigger.

E.g.

--DISABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE

--ENABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE

\*/

----------------------------------------

--T029\_01\_02\_02\_01

--Disable the Trigger

DISABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE;

GO -- Run the previous command and begins new batch

Graphical user interface, application, table

Description automatically generated

----------------------------------------

--T029\_01\_02\_02\_02

--Test the trigger

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TestTable ( ID INT PRIMARY KEY );

GO -- Run the previous command and begins new batch

--Create table successfully..

----------------------------------------

--T029\_01\_02\_02\_03

--Test the trigger

DROP TABLE TestTable;

--Drop table successfully..

----------------------------------------

--T029\_01\_02\_02\_04

--Enable the Trigger

ENABLE TRIGGER trgNoCreateAlterDropTable ON DATABASE;

GO -- Run the previous command and begins new batch

----------------------------------------

--T029\_01\_02\_02\_04

--Test the trigger

CREATE TABLE TestTable ( ID INT PRIMARY KEY );

GO -- Run the previous command and begins new batch

-- Error : Create table un-successfully

A picture containing text

Description automatically generated

----------------------------------------

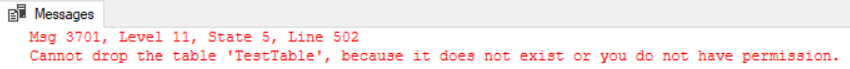
--T029\_01\_02\_02\_05

--Test the trigger

DROP TABLE TestTable;

GO -- Run the previous command and begins new batch

--Error : Drop table un-successfully.



----------------------------------------------------------------------

--T029\_01\_02\_03

--Drop DATABASE scoped DDL Trigger : CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgNoCreateAlterDropTable' )

    BEGIN

        DROP TRIGGER trgNoCreateAlterDropTable ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

1.3. DATABASE scoped DDL Trigger : RENAME

--============================================================================

--T029\_01\_03

--DATABASE scoped DDL Trigger : RENAME

----------------------------------------------------------------------

--T029\_01\_03\_01

--Create DATABASE scoped DDL Trigger : RENAME

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgRename' )

    BEGIN

        DROP TRIGGER trgRename ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

CREATE TRIGGER trgRename ON DATABASE

    FOR RENAME

AS

    BEGIN

        PRINT 'Rename DDL trigger is fired.';

    END;

GO -- Run the previous command and begins new batch

/\*

1.

--CREATE TRIGGER trgRename ON DATABASE

--    FOR RENAME

--AS

--    BEGIN

--        PRINT 'Rename DDL trigger is fired.';

--    END;

Rename DDL trigger will be fired when rename events happen.

1.1.

Syntax

--CREATE TRIGGER [TriggerName]

--ON (All Server/Database)

--FOR [EventType1, EventType2, ...],

--AS

--BEGIN

--   ...

--END

1.2.

--CREATE TRIGGER trgRename ON DATABASE

--    FOR RENAME

CREATE a trigger which name as trgRename

The scope is current database.

This is the RENAME event trigger.

1.3.

Create DDL Database Triggers in SSMS

Database Name --> Programmability --> Database Triggers

\*/

----------------------------------------------------------------------

--T029\_01\_03\_02

--Test DATABASE scoped DDL Trigger : RENAME

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TestTable ( ID INT );

GO -- Run the previous command and begins new batch

--Rename 'TestTable' to 'TestTable2'

EXECUTE sp\_rename 'TestTable', 'TestTable2';

GO -- Run the previous command and begins new batch

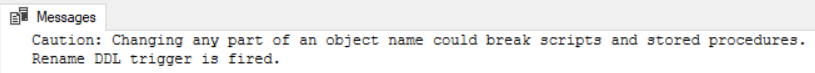
/\*

Output message

--Caution: Changing any part of an object name could break scripts and stored procedures.

--Rename DDL trigger is fired.

\*/



--Rename TestTable2.ID Column to TestTable2.ID2 Column,

--the third parameter means dealing with column.

EXECUTE sp\_rename 'TestTable2.ID', 'ID2', 'column';

GO -- Run the previous command and begins new batch

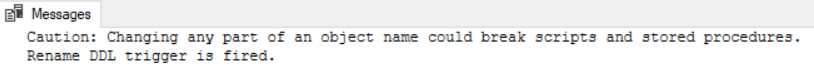
/\*

Output message

--Caution: Changing any part of an object name could break scripts and stored procedures.

--Rename DDL trigger is fired.

\*/



1.4. Clean up

--============================================================================

--T029\_01\_04

--Clean up

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgNoCreateAlterDropTable' )

    BEGIN

        DROP TRIGGER trgNoCreateAlterDropTable ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgRename' )

    BEGIN

        DROP TRIGGER trgRename ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable2' ) )

    BEGIN

        TRUNCATE TABLE TestTable2;

        DROP TABLE TestTable2;

    END;

GO -- Run the previous command and begins new batch

==================================================

2. Server Scoped Data Definition Language (DDL) Triggers event : CREATE\_TABLE, ALTER\_TABLE , DROP\_TABLE

--============================================================================

--T029\_02\_Server Scoped Data Definition Language (DDL) Triggers event : CREATE\_TABLE, ALTER\_TABLE , DROP\_TABLE

--============================================================================

--============================================================================

--T029\_02\_01

--Create Server Scoped DDL Triggers event : CREATE\_TABLE, ALTER\_TABLE , DROP\_TABLE

IF EXISTS ( SELECT  \*

            FROM    sys.server\_triggers

            WHERE   name = 'trgNoCreateAlterDropTable2' )

    BEGIN

        DROP TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER;

    END;

GO -- Run the previous command and begins new batch

CREATE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER

    FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

AS

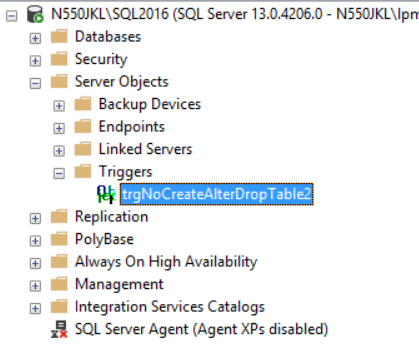
    BEGIN

        PRINT 'Any definition change for table is not prohibited.';

        ROLLBACK;

    END;

GO -- Run the previous command and begins new batch



/\*

1.

--IF EXISTS ( SELECT  \*

--            FROM    sys.server\_triggers

--            WHERE   name = 'trgNoCreateAlterDropTable2' )

--    BEGIN

--        DROP TRIGGER trgNoCreateAlterDropTable2 ON DATABASE;

--    END;

--GO -- Run the previous command and begins new batch

--CREATE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER

--    FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

--AS

--    BEGIN

--        PRINT 'Any definition change for table is not prohibited.';

--        ROLLBACK;

--    END;

prohibit create/Alter/Drop table in ALL SERVER.

1.1.

Syntax

--CREATE TRIGGER [TriggerName]

--ON (All Server/Database)

--FOR [EventType1, EventType2, ...],

--AS

--BEGIN

--   ...

--END

1.2.

--CREATE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER

--    FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

CREATE a trigger which name as trgNoCreateAlterDropTable2

The scope is ALL SERVER.

This is the CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE event trigger.

1.3.

Create DDL Database scope Triggers in SSMS

Database Name --> Programmability --> Database Triggers

Create DDL All Server scope Triggers in SSMS

Server Objects --> Triggers --> ...

\*/

--============================================================================

--T029\_02\_02

--Disable/Enable Server Scoped DDL Triggers event : CREATE\_TABLE, ALTER\_TABLE , DROP\_TABLE

--Disable/Enable ALL SERVER scope DDL trigger

DISABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER;

GO -- Run the previous command and begins new batch

ENABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER;

GO -- Run the previous command and begins new batch

/\*

Enable/Sisable ALL SERVER scope DDL trigger

Syntax

--DISABLE TRIGGER trgName ON ALL SERVER

Disable trgName All Server scope trigger.

--ENABLE TRIGGER trgName ON ALL SERVER

Enable trgName All Server scope trigger.

E.g.

--DISABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER

--ENABLE TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER

\*/

--============================================================================

--T029\_02\_03

--Drop Server Scoped DDL Triggers event : CREATE\_TABLE, ALTER\_TABLE , DROP\_TABLE

IF EXISTS ( SELECT  \*

            FROM    sys.server\_triggers

            WHERE   name = 'trgNoCreateAlterDropTable2' )

    BEGIN

        DROP TRIGGER trgNoCreateAlterDropTable2 ON ALL SERVER;

    END;

GO -- Run the previous command and begins new batch

3. TriggerExecutionOrder\_sp\_settriggerorder

--============================================================================

--T029\_03\_TriggerExecutionOrder\_sp\_settriggerorder

--============================================================================

3.1. sp\_settriggerorder

--============================================================================

--T029\_03\_01

--sp\_settriggerorder

----------------------------------------------------------------------

--T029\_03\_01\_01

--Create two CREATE\_TABLE triggers

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgFirstTrigger' )

    BEGIN

        DROP TRIGGER trgFirstTrigger ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgSecondTrigger' )

    BEGIN

        DROP TRIGGER trgSecondTrigger ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

CREATE TRIGGER trgSecondTrigger ON DATABASE

    FOR CREATE\_TABLE

AS

    PRINT 'This is the second CREATE\_TABLE trigger';

GO

CREATE TRIGGER trgFirstTrigger ON DATABASE

    FOR CREATE\_TABLE

AS

    PRINT 'This is the first CREATE\_TABLE trigger';

GO

----------------------------------------------------------------------

--T029\_03\_01\_02

--Test the trigger

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

--Create a simple test table

CREATE TABLE TestTable ( ID INT );

GO

/\*

Output as the following

--This is the second CREATE\_TABLE trigger

--This is the first CREATE\_TABLE trigger

\*/

--============================================================================

--T029\_03\_01

--sp\_settriggerorder

----------------------------------------------------------------------

--T029\_03\_01\_01

--Create two CREATE\_TABLE triggers

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgFirstTrigger' )

    BEGIN

        DROP TRIGGER trgFirstTrigger ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgSecondTrigger' )

    BEGIN

        DROP TRIGGER trgSecondTrigger ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

CREATE TRIGGER trgSecondTrigger ON DATABASE

    FOR CREATE\_TABLE

AS

    PRINT 'This is the second CREATE\_TABLE trigger';

GO

CREATE TRIGGER trgFirstTrigger ON DATABASE

    FOR CREATE\_TABLE

AS

    PRINT 'This is the first CREATE\_TABLE trigger';

GO

----------------------------------------------------------------------

--T029\_03\_01\_02

--Test the trigger

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

--Create a simple test table

CREATE TABLE TestTable ( ID INT );

GO

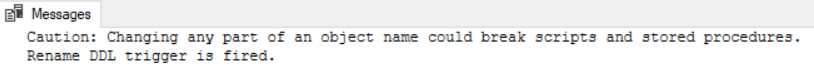
/\*

Output as the following

--This is the second CREATE\_TABLE trigger

--This is the first CREATE\_TABLE trigger

\*/



----------------------------------------------------------------------

--T029\_03\_01\_03

--sp\_settriggerorder

EXEC sp\_settriggerorder @triggername = 'trgFirstTrigger', @order = 'first',

       --@order = 'last',

       --@order = 'none',

    @stmttype = 'CREATE\_TABLE', @namespace = 'DATABASE';

       --@namespace = 'SERVER';

/\*

1.

--EXEC sp\_settriggerorder

--     @triggername = 'trgFirstTrigger',

--     @order = 'first',

--  @stmttype = 'CREATE\_TABLE',

--     @namespace = 'DATABASE';

Set the database scoped trigger, trgFirstTrigger,

to the first order of CREATE\_TABLE event.

1.1.

--     @triggername = 'trgFirstTrigger',

1st parameter is the trigger name that you want to set.

1.2.

--@order = 'first',

--@order = 'last',

--@order = 'none',

2nd parameter is the running order.

Value can be First, Last or None.

1.3.

--  @stmttype = 'CREATE\_TABLE',

3rd parameter is the SQL statement that fires the trigger.

You many only put ONE statement type here.

If you want 'ALTER\_TABLE' or 'DROP\_TABLE' or other statement,

you need to "ECECUTE sp\_settriggerorder" several times

for each statement type you want to set.

1.4.

-- @namespace = 'DATABASE';

or

-- @namespace = 'SERVER';

4th parameter is the scope of the trigger.

Value can be DATABASE, SERVER, or NULL.

\*/

----------------------------------------------------------------------

--T029\_03\_01\_04

--Test the trigger

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

--Create a simple test table

CREATE TABLE TestTable ( ID INT );

GO

/\*

--This is the first CREATE\_TABLE trigger

--This is the second CREATE\_TABLE trigger

\*/

Text

Description automatically generated

----------------------------------------------------------------------

--T029\_03\_01\_05

--sp\_settriggerorder

--When @order = 'none', trigger is fired in random order

EXEC sp\_settriggerorder @triggername = 'trgFirstTrigger',

       --@order = 'first',

       --@order = 'last',

    @order = 'none', -- Clean up

    @stmttype = 'CREATE\_TABLE', @namespace = 'DATABASE';

       --@namespace = 'SERVER';

/\*

1.

--EXEC sp\_settriggerorder

--     @triggername = 'trgFirstTrigger',

--     @order = 'none',

--  @stmttype = 'CREATE\_TABLE',

--     @namespace = 'DATABASE';

Set the database scoped trigger, trgFirstTrigger,

to the none order of CREATE\_TABLE event.

When @order = 'none', trigger is fired in random order.

1.1.

--     @triggername = 'trgFirstTrigger',

1st parameter is the trigger name that you want to set.

1.2.

--@order = 'first',

--@order = 'last',

--@order = 'none',

2nd parameter is the running order.

Value can be First, Last or None.

1.3.

--  @stmttype = 'CREATE\_TABLE',

3rd parameter is the SQL statement that fires the trigger.

You many only put ONE statement type here.

If you want 'ALTER\_TABLE' or 'DROP\_TABLE' or other statement,

you need to "ECECUTE sp\_settriggerorder" several times

for each statement type you want to set.

1.4.

-- @namespace = 'DATABASE';

or

-- @namespace = 'SERVER';

4th parameter is the scope of the trigger.

Value can be DATABASE, SERVER, or NULL.

\*/

----------------------------------------------------------------------

--T029\_03\_01\_06

--Drop the trigger

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

--Create a simple test table

CREATE TABLE TestTable ( ID INT );

GO

/\*

Output as the following

--This is the second CREATE\_TABLE trigger

--This is the first CREATE\_TABLE trigger

\*/

Text

Description automatically generated

----------------------------------------------------------------------

--T029\_03\_01\_07

--Clean up

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgFirstTrigger' )

    BEGIN

        DROP TRIGGER trgFirstTrigger ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgSecondTrigger' )

    BEGIN

        DROP TRIGGER trgSecondTrigger ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

----------------------------------------------------------------------

--T029\_03\_01\_03

--sp\_settriggerorder

EXEC sp\_settriggerorder @triggername = 'trgFirstTrigger', @order = 'first',

       --@order = 'last',

       --@order = 'none',

    @stmttype = 'CREATE\_TABLE', @namespace = 'DATABASE';

       --@namespace = 'SERVER';

/\*

1.

--EXEC sp\_settriggerorder

--     @triggername = 'trgFirstTrigger',

--     @order = 'first',

--  @stmttype = 'CREATE\_TABLE',

--     @namespace = 'DATABASE';

Set the database scoped trigger, trgFirstTrigger,

to the first order of CREATE\_TABLE event.

1.1.

--     @triggername = 'trgFirstTrigger',

1st parameter is the trigger name that you want to set.

1.2.

--@order = 'first',

--@order = 'last',

--@order = 'none',

2nd parameter is the running order.

Value can be First, Last or None.

1.3.

--  @stmttype = 'CREATE\_TABLE',

3rd parameter is the SQL statement that fires the trigger.

You many only put ONE statement type here.

If you want 'ALTER\_TABLE' or 'DROP\_TABLE' or other statement,

you need to "ECECUTE sp\_settriggerorder" several times

for each statement type you want to set.

1.4.

-- @namespace = 'DATABASE';

or

-- @namespace = 'SERVER';

4th parameter is the scope of the trigger.

Value can be DATABASE, SERVER, or NULL.

\*/

----------------------------------------------------------------------

--T029\_03\_01\_04

--Test the trigger

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

--Create a simple test table

CREATE TABLE TestTable ( ID INT );

GO

/\*

--This is the first CREATE\_TABLE trigger

--This is the second CREATE\_TABLE trigger

\*/

----------------------------------------------------------------------

--T029\_03\_01\_05

--sp\_settriggerorder

--When @order = 'none', trigger is fired in random order

EXEC sp\_settriggerorder @triggername = 'trgFirstTrigger',

       --@order = 'first',

       --@order = 'last',

    @order = 'none', -- Clean up

    @stmttype = 'CREATE\_TABLE', @namespace = 'DATABASE';

       --@namespace = 'SERVER';

/\*

1.

--EXEC sp\_settriggerorder

--     @triggername = 'trgFirstTrigger',

--     @order = 'none',

--  @stmttype = 'CREATE\_TABLE',

--     @namespace = 'DATABASE';

Set the database scoped trigger, trgFirstTrigger,

to the none order of CREATE\_TABLE event.

When @order = 'none', trigger is fired in random order.

1.1.

--     @triggername = 'trgFirstTrigger',

1st parameter is the trigger name that you want to set.

1.2.

--@order = 'first',

--@order = 'last',

--@order = 'none',

2nd parameter is the running order.

Value can be First, Last or None.

1.3.

--  @stmttype = 'CREATE\_TABLE',

3rd parameter is the SQL statement that fires the trigger.

You many only put ONE statement type here.

If you want 'ALTER\_TABLE' or 'DROP\_TABLE' or other statement,

you need to "ECECUTE sp\_settriggerorder" several times

for each statement type you want to set.

1.4.

-- @namespace = 'DATABASE';

or

-- @namespace = 'SERVER';

4th parameter is the scope of the trigger.

Value can be DATABASE, SERVER, or NULL.

\*/

----------------------------------------------------------------------

--T029\_03\_01\_06

--Drop the trigger

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

--Create a simple test table

CREATE TABLE TestTable ( ID INT );

GO

/\*

Output as the following

--This is the second CREATE\_TABLE trigger

--This is the first CREATE\_TABLE trigger

\*/

----------------------------------------------------------------------

--T029\_03\_01\_07

--Clean up

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgFirstTrigger' )

    BEGIN

        DROP TRIGGER trgFirstTrigger ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgSecondTrigger' )

    BEGIN

        DROP TRIGGER trgSecondTrigger ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

3.2. sp\_settriggerorder order

--============================================================================

--T029\_03\_02

--sp\_settriggerorder order

/\*

1.

Even if we use sp\_settriggerorder to change the order,

server scoped triggers will always fire before any of the database scoped triggers.

2.

sp\_settriggerorder can set the order of server scoped DDL triggers

or database scoped triggers.

3.

If there is a database scoped DDL Triggers and a server scoped DDL triggers

handling the same event.

Here is the execution order.

The server scope DDL trigger which @order = 'first'

-->

other server scope DDL triggers which @order = 'none'

-->

The server scope DDL trigger which @order = 'Last'

-->

The database scope DDL trigger which @order = 'first'

-->

other database scope DDL triggers which @order = 'none'

-->

The database scope DDL trigger which @order = 'Last'

\*/

----------------------------------------------------------------------

--T029\_03\_02\_01

--Create Database Scope Create\_Table Trigger

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgDatabaseScopeCreateTableTrigger' )

    BEGIN

        DROP TRIGGER trgDatabaseScopeCreateTableTrigger ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

CREATE TRIGGER trgDatabaseScopeCreateTableTrigger ON DATABASE

    FOR CREATE\_TABLE

AS

    BEGIN

        PRINT 'Database Scope CREATE\_TABLE Trigger';

    END;

GO -- Run the previous command and begins new batch

----------------------------------------------------------------------

--T029\_03\_02\_02

--Create Server Scope Create\_Table Trigger

IF EXISTS ( SELECT  \*

            FROM    sys.server\_triggers

            WHERE   name = 'trgServerScopeCreateTableTrigger' )

    BEGIN

        DROP TRIGGER trgServerScopeCreateTableTrigger ON ALL SERVER;

    END;

GO -- Run the previous command and begins new batch

CREATE TRIGGER trgServerScopeCreateTableTrigger ON ALL SERVER

    FOR CREATE\_TABLE

AS

    BEGIN

        PRINT 'Server Scope CREATE\_TABLE Trigger';

    END;

GO -- Run the previous command and begins new batch

----------------------------------------------------------------------

--T029\_03\_02\_03

--Test the trigger

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

--Create a simple test table

CREATE TABLE TestTable ( ID INT );

GO -- Run the previous command and begins new batch

/\*

Output as the following

--Server Scope CREATE\_TABLE Trigger

--Database Scope CREATE\_TABLE Trigger

\*/

Text

Description automatically generated

----------------------------------------------------------------------

--T029\_03\_02\_04

--sp\_settriggerorder order

EXEC sp\_settriggerorder @triggername = 'trgDatabaseScopeCreateTableTrigger',

    @order = 'first', @stmttype = 'CREATE\_TABLE', @namespace = 'DATABASE';

/\*

1.

Even if we use sp\_settriggerorder to change the order,

server scoped triggers will always fire before any of the database scoped triggers.

2.

sp\_settriggerorder can set the order of server scoped DDL triggers

or database scoped triggers .

\*/

----------------------------------------------------------------------

--T029\_03\_02\_05

--Test the trigger

--If Table exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

--Create a simple test table

CREATE TABLE TestTable ( ID INT );

GO

/\*

Output as the following

--Server Scope CREATE\_TABLE Trigger

--Database Scope CREATE\_TABLE Trigger

\*/

Text

Description automatically generated

----------------------------------------------------------------------

--T029\_03\_02\_06

--Clean up

IF EXISTS ( SELECT  \*

            FROM    sys.triggers

            WHERE   name = 'trgDatabaseScopeCreateTableTrigger' )

    BEGIN

        DROP TRIGGER trgDatabaseScopeCreateTableTrigger ON DATABASE;

    END;

GO -- Run the previous command and begins new batch

IF EXISTS ( SELECT  \*

            FROM    sys.server\_triggers

            WHERE   name = 'trgServerScopeCreateTableTrigger' )

    BEGIN

        DROP TRIGGER trgServerScopeCreateTableTrigger ON ALL SERVER;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

4. AuditTableChanges

--============================================================================

--T029\_04\_AuditTableChanges

--============================================================================

4.1. EVENTDATA

--===========================================================================

--T029\_04\_01

--EVENTDATA()

------------------------------------------------

--T029\_04\_01\_01

--EVENTDATA()

IF EXISTS ( SELECT  \*

            FROM    sys.server\_triggers

            WHERE   name = 'trgAuditTableStructureChanges' )

    BEGIN

        DROP TRIGGER trgAuditTableStructureChanges ON ALL SERVER;

    END;

GO -- Run the previous command and begins new batch

CREATE TRIGGER trgAuditTableStructureChanges ON ALL SERVER

    FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

AS

    BEGIN

        SELECT  EVENTDATA();

    END;

GO -- Run the previous command and begins new batch

/\*

--EVENTDATA()

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/eventdata-transact-sql>

EventData() returns information about server or database events in XML format.

E.g.

--<EVENT\_INSTANCE>

--  <EventType>CREATE\_TABLE</EventType>

--  <PostTime>2017-10-10T04:42:27.870</PostTime>

--  <SPID>54</SPID>

--  <ServerName>N550JKL\SQL2016</ServerName>

--  <LoginName>MicrosoftAccount\[lpmplpmp01@hotmail.com](mailto:lpmplpmp01@hotmail.com)</LoginName>

--  <UserName>dbo</UserName>

--  <DatabaseName>Sample3</DatabaseName>

--  <SchemaName>dbo</SchemaName>

--  <ObjectName>TestTable</ObjectName>

--  <ObjectType>TABLE</ObjectType>

--  <TSQLCommand>

--    <SetOptions ANSI\_NULLS="ON" ANSI\_NULL\_DEFAULT="ON" ANSI\_PADDING="ON" QUOTED\_IDENTIFIER="ON" ENCRYPTED="FALSE" />

--    <CommandText>CREATE TABLE TestTable

--    (

--      ID INT ,

--      [Name] NVARCHAR(100)

--    )</CommandText>

--  </TSQLCommand>

--</EVENT\_INSTANCE>

\*/

------------------------------------------------

--T029\_04\_01\_02

--Test EVENTDATA()

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TestTable

    (

      ID INT ,

      [Name] NVARCHAR(100)

    );

GO -- Run the previous command and begins new batch

Graphical user interface, text

Description automatically generated with medium confidence

Graphical user interface, text, application, email

Description automatically generated

/\*

Output

--<EVENT\_INSTANCE>

--  <EventType>CREATE\_TABLE</EventType>

--  <PostTime>2017-10-10T08:01:28.670</PostTime>

--  <SPID>54</SPID>

--  <ServerName>N550JKL\SQL2016</ServerName>

--  <LoginName>MicrosoftAccount\[lpmplpmp01@hotmail.com](mailto:lpmplpmp01@hotmail.com)</LoginName>

--  <UserName>dbo</UserName>

--  <DatabaseName>Sample3</DatabaseName>

--  <SchemaName>dbo</SchemaName>

--  <ObjectName>TestTable</ObjectName>

--  <ObjectType>TABLE</ObjectType>

--  <TSQLCommand>

--    <SetOptions ANSI\_NULLS="ON" ANSI\_NULL\_DEFAULT="ON" ANSI\_PADDING="ON" QUOTED\_IDENTIFIER="ON" ENCRYPTED="FALSE" />

--    <CommandText>CREATE TABLE TestTable

--    (

--      ID INT ,

--      [Name] NVARCHAR(100)

--    )</CommandText>

--  </TSQLCommand>

--</EVENT\_INSTANCE>

\*/

4.2. AuditTableStructureChanges

--===========================================================================

--T029\_04\_02

--AuditTableStructureChanges

------------------------------------------------

--T029\_04\_02\_01

--Create AuditTableStructureChanges

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'AuditTableStructureChanges' ) )

    BEGIN

        TRUNCATE TABLE dbo.AuditTableStructureChanges;

        DROP TABLE AuditTableStructureChanges;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE AuditTableStructureChanges

    (

      EventType NVARCHAR(300) ,

      PostTime DATETIME ,

         --Server Process ID

      SPID NVARCHAR(300) ,

      ServerName NVARCHAR(300) ,

      LoginName NVARCHAR(300) ,

      UserName NVARCHAR(300) ,

      DatabaseName NVARCHAR(300) ,

      SchemaName NVARCHAR(300) ,

      ObjectName NVARCHAR(300) ,

      ObjectType NVARCHAR(300) ,

      TSQLCommand NVARCHAR(MAX)

    );

GO -- Run the previous command and begins new batch

Graphical user interface, text

Description automatically generated with medium confidence

------------------------------------------------

--T029\_04\_02\_02

--Alter trgAuditTableStructureChanges

IF EXISTS ( SELECT  \*

            FROM    sys.server\_triggers

            WHERE   name = 'trgAuditTableStructureChanges' )

    BEGIN

        DROP TRIGGER trgAuditTableStructureChanges ON ALL SERVER;

    END;

GO -- Run the previous command and begins new batch

CREATE TRIGGER trgAuditTableStructureChanges ON ALL SERVER

    FOR CREATE\_TABLE, ALTER\_TABLE, DROP\_TABLE

AS

    BEGIN

        DECLARE @EventData XML;

        SELECT  @EventData = EVENTDATA();

        INSERT  INTO [Sample].dbo.AuditTableStructureChanges

                ( EventType ,

                  PostTime ,

                  SPID ,

                  ServerName ,

                  LoginName ,

                  UserName ,

                  DatabaseName ,

                  SchemaName ,

                  ObjectName ,

                  ObjectType ,

                  TSQLCommand

                )

        VALUES  ( @EventData.value('(/EVENT\_INSTANCE/EventType)[1]',

                                   'NVARCHAR(300)') ,

                  @EventData.value('(/EVENT\_INSTANCE/PostTime)[1]', 'DATETIME') ,

                  @EventData.value('(/EVENT\_INSTANCE/SPID)[1]',

                                   'NVARCHAR(300)') ,

                  @EventData.value('(/EVENT\_INSTANCE/ServerName)[1]',

                                   'NVARCHAR(300)') ,

                  @EventData.value('(/EVENT\_INSTANCE/LoginName)[1]',

                                   'NVARCHAR(300)') ,

                  @EventData.value('(/EVENT\_INSTANCE/UserName)[1]',

                                   'NVARCHAR(300)') ,

                  @EventData.value('(/EVENT\_INSTANCE/DatabaseName)[1]',

                                   'NVARCHAR(300)') ,

                  @EventData.value('(/EVENT\_INSTANCE/SchemaName)[1]',

                                   'NVARCHAR(300)') ,

                  @EventData.value('(/EVENT\_INSTANCE/ObjectName)[1]',

                                   'NVARCHAR(300)') ,

                  @EventData.value('(/EVENT\_INSTANCE/ObjectType)[1]',

                                   'NVARCHAR(300)') ,

                  @EventData.value('(/EVENT\_INSTANCE/TSQLCommand)[1]',

                                   'NVARCHAR(MAX)')

                );

    END;

GO -- Run the previous command and begins new batch

------------------------------------------------

--T029\_04\_02\_03

--Create TestTable to test trgAuditTableStructureChanges

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

CREATE TABLE TestTable

    (

      ID INT ,

      [Name] NVARCHAR(100)

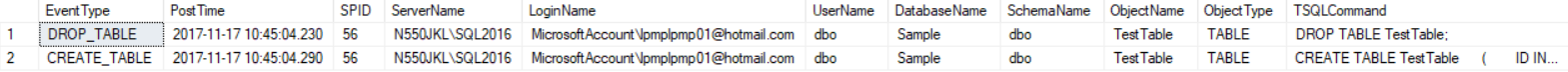
    );

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    dbo.AuditTableStructureChanges;

GO -- Run the previous command and begins new batch



------------------------------------------------

--T029\_04\_02\_04

--Alter TestTable to test trgAuditTableStructureChanges

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        ALTER TABLE TestTable

        ALTER COLUMN [Name] NVARCHAR(150);

    END;

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    dbo.AuditTableStructureChanges;

GO -- Run the previous command and begins new batch



------------------------------------------------

--T029\_04\_02\_05

--Drop TestTable to test trgAuditTableStructureChanges

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

SELECT  \*

FROM    dbo.AuditTableStructureChanges;

GO -- Run the previous command and begins new batch



4.3. Clean up

--===========================================================================

--T029\_04\_03

--Clean up

IF EXISTS ( SELECT  \*

            FROM    sys.server\_triggers

            WHERE   name = 'trgAuditTableStructureChanges' )

    BEGIN

        DROP TRIGGER trgAuditTableStructureChanges ON ALL SERVER;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'TestTable' ) )

    BEGIN

        TRUNCATE TABLE dbo.TestTable;

        DROP TABLE TestTable;

    END;

GO -- Run the previous command and begins new batch

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.TABLES

              WHERE     TABLE\_NAME = 'AuditTableStructureChanges' ) )

    BEGIN

        TRUNCATE TABLE dbo.AuditTableStructureChanges;

        DROP TABLE AuditTableStructureChanges;

    END;

GO -- Run the previous command and begins new batch

==================================================

5. LogonTriggers\_sys.dm\_exec\_sessions\_ORIGINAL\_LOGIN()

--============================================================================

--T029\_05\_LogonTriggers\_sys.dm\_exec\_sessions\_ORIGINAL\_LOGIN()

--============================================================================

--===========================================================================

--T029\_05\_01

--sys.dm\_exec\_sessions and ORIGINAL\_LOGIN()

SELECT  ORIGINAL\_LOGIN();

GO -- Run the previous command and begins new batch



/\*

ORIGINAL\_LOGIN()

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/original-login-transact-sql>

Returns the name of the login that connected to the instance of SQL Server.

Output

--MicrosoftAccount\[XXXXXX@hotmail.com](mailto:XXXXXX@hotmail.com)

\*/

--===========================================================================

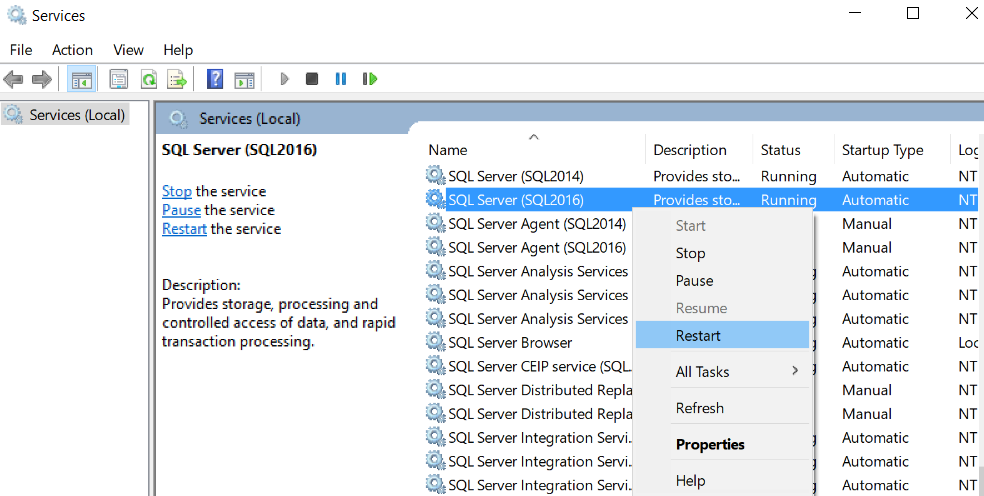
--T029\_05\_02

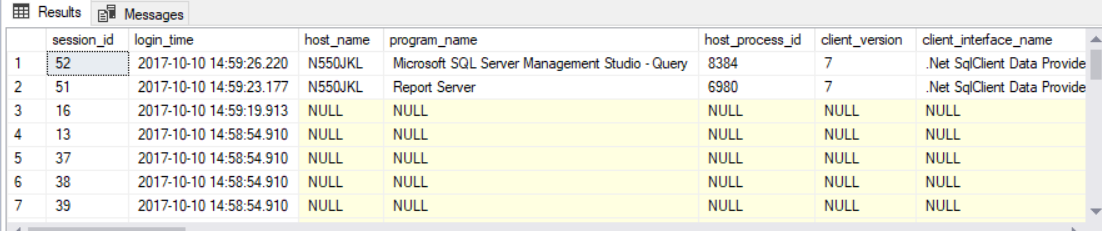
SELECT  \*

FROM    sys.dm\_exec\_sessions

ORDER BY login\_time DESC;

GO -- Run the previous command and begins new batch





/\*

1.

Go to Service Manager and restart the SQL server service

2.

--sys.dm\_exec\_sessions

Reference:

<https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-sessions-transact-sql>

sys.dm\_exec\_sessions returns one row

per authenticated session on SQL Server.

\*/

--===========================================================================

--T029\_05\_03

--sys.dm\_exec\_sessions

SELECT  is\_user\_process ,

        original\_login\_name ,

        session\_id

FROM    sys.dm\_exec\_sessions

WHERE   is\_user\_process = 1

        AND original\_login\_name = ORIGINAL\_LOGIN()

ORDER BY login\_time DESC;

GO -- Run the previous command and begins new batch

--Return all the sessions which is using by current user.

SELECT  COUNT(\*)

FROM    sys.dm\_exec\_sessions

WHERE   is\_user\_process = 1

        AND original\_login\_name = ORIGINAL\_LOGIN();

GO -- Run the previous command and begins new batch

--Return the number of the sessions which is using by current user.

Graphical user interface

Description automatically generated

/\*

Go to Service Manager and restart the SQL server service

1.

ORIGINAL\_LOGIN()

Reference:

<https://docs.microsoft.com/en-us/sql/t-sql/functions/original-login-transact-sql>

Returns the name of the login that connected to the instance of SQL Server.

Output

--MicrosoftAccount\[XXXXXX@hotmail.com](mailto:XXXXXX@hotmail.com)

2.

--sys.dm\_exec\_sessions

sys.dm\_exec\_sessions returns one row

per authenticated session on SQL Server.

2.1.

Go to Service Manager and restart the SQL server service

2.2.

E.g.

--SELECT  is\_user\_process ,

--        original\_login\_name ,

--           session\_id

--FROM    sys.dm\_exec\_sessions

--WHERE   is\_user\_process = 1

--        AND original\_login\_name = ORIGINAL\_LOGIN()

--ORDER BY login\_time DESC;

Return all the sessions which is using by current user.

2.3.

--SELECT  COUNT(\*)

--FROM    sys.dm\_exec\_sessions

--WHERE   is\_user\_process = 1

--        AND original\_login\_name = ORIGINAL\_LOGIN()

Return the number of the sessions which is using by current user.

2.4.

Columns in sys.dm\_exec\_sessions

2.4.1.

--sys.dm\_exec\_sessions

Reference:

<https://docs.microsoft.com/en-us/sql/relational-databases/system-dynamic-management-views/sys-dm-exec-sessions-transact-sql>

sys.dm\_exec\_sessions returns one row

per authenticated session on SQL Server.

2.4.2.

--is\_user\_process=1

0 if the session is a system session.

Otherwise, it is 1. Is not nullable.

2.4.3.

--original\_login\_name

SQL Server login name that the client used to create this session.

2.4.4.

--session\_id

The current connection ID,

If using SSMS, you may find the session on the bottom bar.

\*/

--===========================================================================

--T029\_05\_04

--sys.dm\_exec\_sessions

--Drop ALL SERVER scope trigger if it exists

--Reference: <http://sqlhints.com/2016/04/03/how-to-check-if-a-trigger-exists-in-sql-server/>

IF EXISTS ( SELECT  \*

            FROM    sys.server\_triggers

            WHERE   name = 'trgLogonAuditTriggers' )

    BEGIN

        DROP TRIGGER trgLogonAuditTriggers ON ALL SERVER;

    END;

GO -- Run the previous command and begins new batch

CREATE TRIGGER trgLogonAuditTriggers ON ALL SERVER

    FOR LOGON

AS

    BEGIN

        DECLARE @LoginName NVARCHAR(100);

        SET @LoginName = ORIGINAL\_LOGIN();

        IF ( SELECT COUNT(\*)

             FROM   sys.dm\_exec\_sessions

             WHERE  is\_user\_process = 1

                    AND original\_login\_name = @LoginName

           ) > 3

            BEGIN

                PRINT 'LoginName : ' + @LoginName + '. The connections can not more than 3';

                ROLLBACK;

            END;

    END;

GO -- Run the previous command and begins new batch

Graphical user interface, text, application

Description automatically generated

/\*

Go to Service Manager and restart the SQL server service

If there are more than 3 sessions/connections by current user,

then we block it by rollback.

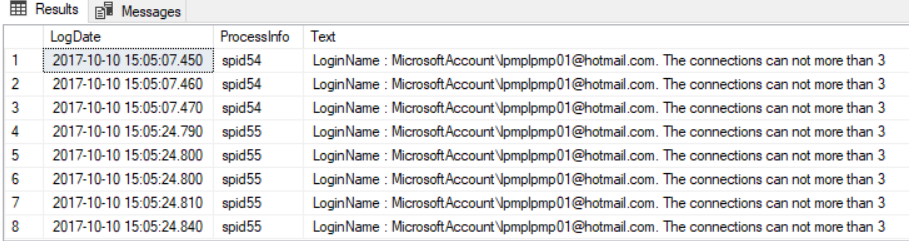
\*/

--===========================================================================

--T029\_05\_05

--The system store procedure to read the Error Log.

Execute sp\_readerrorlog 0, 1, 'The connections can not more than 3'



/\*

1.

--sp\_readerrorlog

1.1.

Reference:

<https://www.mssqltips.com/sqlservertip/1476/reading-the-sql-server-log-files-using-tsql/>

1.1.1.

1st parameter: The session Number.

Value of error log file you want to read:

0 = current, 1 = Archive #1, 2 = Archive #2, etc...

1.1.2.

2nd parameter: Log File Type:

1 or NULL = error log, 2 = SQL Agent log

1.1.3.

3rd parameter:

Search string 1:

String one you want to search for

1.1.4.

4th parameter:

Search string 2:

String two you want to search for to further refine the results

1.2.

E.g.

Execute sp\_readerrorlog 0, 1, 'The connections can not more than 3'

Searching the Text 'The connections can not more than 3'

in the Error log in Current log file

\*/

--===========================================================================

--T029\_05\_06

--Clean up

IF EXISTS ( SELECT  \*

            FROM    sys.server\_triggers

            WHERE   name = 'trgLogonAuditTriggers' )

    BEGIN

        DROP TRIGGER trgLogonAuditTriggers ON ALL SERVER;

    END;

GO -- Run the previous command and begins new batch