**1. Create DDL triggers.**

**Create CREATE trigger on objects: DATABASE, TABLE, VIEW, LOGIN, USER. etc. Objects creation and functional test of the trigger.**

**• <PostTime> The time at which the trigger was fired**

**• <SPID> The ID of the database process causing the trigger to fire**

**• <EventType>**

**The type of event that caused the trigger to fire, such as CREATE\_TABLE or UPDATE\_STATISTICS**

CREATE TRIGGER UpdStats

ON DATABASE FOR UPDATE\_STATISTICS

AS DECLARE @data XML

DECLARE @posttime NVARCHAR(24)

DECLARE @database NVARCHAR(100)

DECLARE @targetobject NVARCHAR(100)

SET @data = eventdata()

SET @posttime = CONVERT(NVARCHAR(24), @data.query('data(//PostTime)'))

SET @database = CONVERT(NVARCHAR(100), @data.query('data(//DatabaseName)'))

SET @targetobject = CONVERT(NVARCHAR(100), @data.query('data(//TargetObjectName)'))

PRINT @posttime PRINT @database PRINT @targetobject

**Trigger select:**

SELECT name FROM sys.triggers

name

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

tr\_insert\_1

tr\_delete1

tr\_update1

tr1update

tr\_vdis

Toylnventory\_UPDATE

tr\_insert2

UpdStats

(8 row(s) affected)

SELECT definition FROM sys.sql\_modules

WHERE [object\_id]=(SELECT [object\_id]

FROM sys.triggers

WHERE name='trigger\_name')

Definition

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

(0 row(s) affected)

**Use the ALTER TRIGGER command to change the definition of a DDL trigger. The syntax for changing a DDL trigger is very similar to amending a standard trigger. You can:**

**• Change the triggering event.**

**• Encrypt the trigger.**

**• Modify the Transact-SQL statements that make up the body of the trigger.**

**Trigger removal:**

DROP TRIGGER UpdStats ON DATABASE

Command(s) completed successfully.

**Table creation (audit):**

CREATE TABLE dbo.ddl

(Command NVARCHAR(1000),

PostTime NVARCHAR(50),

HostName NVARCHAR(100),

LoginName NVARCHAR(100))

**Trigger creation:**

CREATE TRIGGER ddl

ON DATABASE

FOR DDL\_DATABASE\_LEVEL\_EVENTS

AS DECLARE @data XML

DECLARE @cmd NVARCHAR(1000)

DECLARE @posttime NVARCHAR(24)

DECLARE @spid NVARCHAR(6)

DECLARE @hostname NVARCHAR(100)

DECLARE @loginname NVARCHAR(100)

SET @data=eventdata()

SET @cmd=CONVERT(NVARCHAR(100), @data.query('data(//TSQLCommand//CommandText)'))

SET @posttime=CONVERT(NVARCHAR(24), @data.query('data(//PostTime)'))

SET @spid=CONVERT(NVARCHAR(6), @data.query('data(//SPID)'))

SET @hostname=HOST\_NAME()

SET @loginname = SYSTEM\_USER

**Data insert:**

INSERT INTO dbo.ddl (Command,PostTime,HostName,LoginName)

VALUES('insert', '11/26/09', '7777', 'student')

**Data select:**

SELECT \* FROM dbo.ddl

**insert 11/26/09 7777 student**

UPDATE STATISTICS Discipline

CREATE TABLE dbo.Test(col INT)

DROP TABLE dbo.Test

SELECT \* FROM dbo.ddl

DROP TRIGGER ddl

ON DATABASE

DROP TRIGGER ddl

ON DATABASE

**Event Natifications:**

CREATE EVENT NOTIFICATION ddltable1

ON DATABASE FOR DDL\_TABLE\_VIEW\_EVENTS

TO SERVICE 'EventService', '111'

**EVENTDATE:**

Trigger creation using **EVENTDATE:**

CREATE TRIGGER ev

ON DATABASE

FOR CREATE\_TABLE

AS

PRINT 'CREATE TABLE'

SELECT EVENTDATA().value

('(/EVENT\_INSTANCE/TSQLCommand/CommandText)[1]','nvarchar(max)')

RAISERROR ('New tables cannot be created in this database.', 16, 1)

ROLLBACK

CREATE TABLE NewTable (Column1 int);

DROP TRIGGER ev

ON DATABASE

CREATE TABLE log1

(PostTime datetime,

DB\_User nvarchar(100),

Event nvarchar(100),

TSQL nvarchar(2000));

CREATE TRIGGER log

ON DATABASE

FOR DDL\_DATABASE\_LEVEL\_EVENTS

AS

DECLARE @data XML

SET @data = EVENTDATA()

INSERT log1

(PostTime, DB\_User, Event, TSQL)

VALUES

(GETDATE(),

CONVERT(nvarchar(100), CURRENT\_USER),

@data.value('(/EVENT\_INSTANCE/EventType)[1]', 'nvarchar(100)'),

@data.value('(/EVENT\_INSTANCE/TSQLCommand)[1]', 'nvarchar(2000)') ) ;

--Test the trigger.

CREATE TABLE TestTable (a int)

SELECT \* FROM TestTable

DROP TABLE TestTable ;

SELECT \* FROM log1

--Drop the trigger.

DROP TRIGGER log

ON DATABASE

--Drop table ddl\_log.

DROP TABLE log1