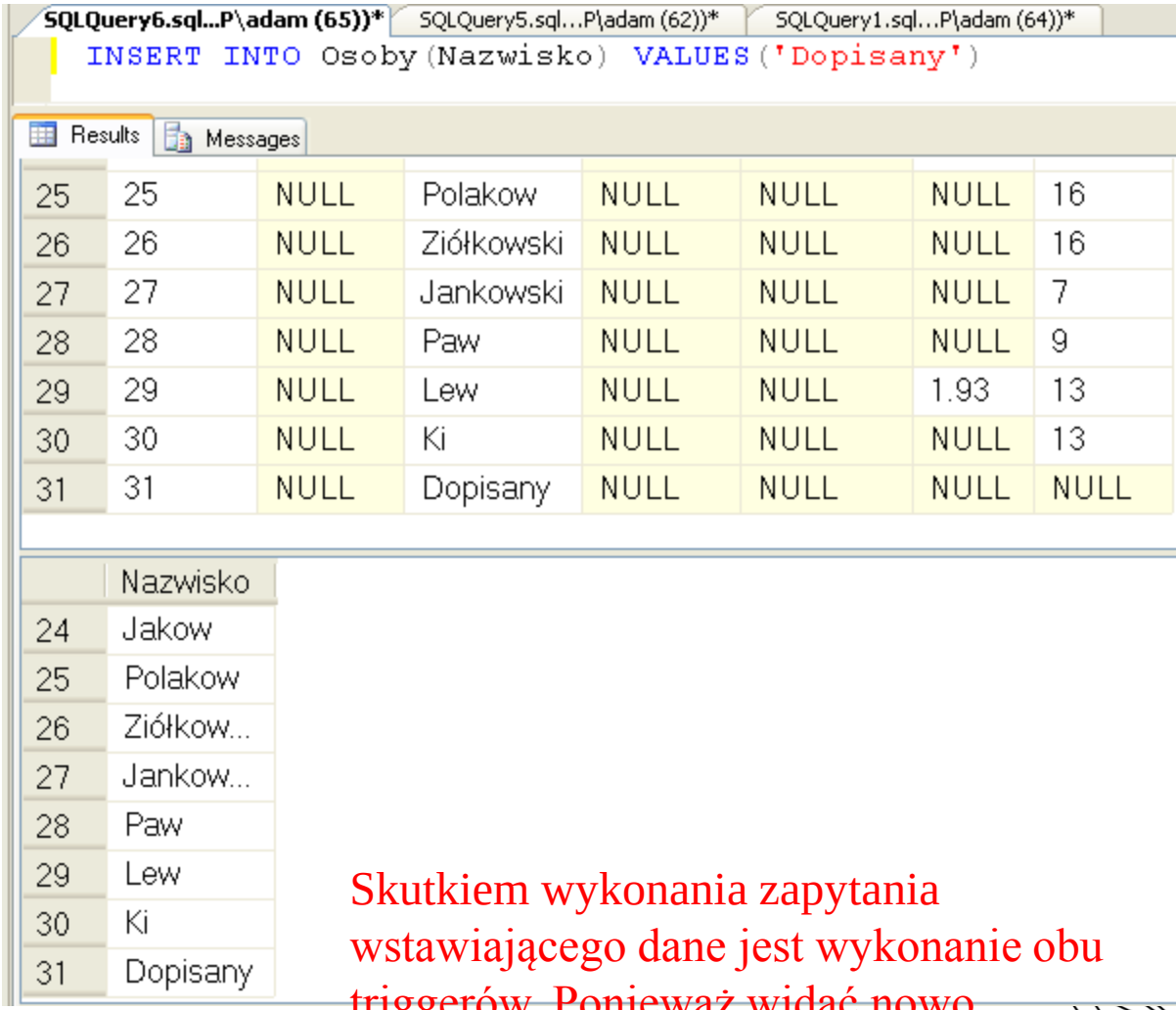
# Procedury wyzwalane
# Triggery

# Tworzenie triggerów – procedur wyzwalanych (wyzwalaczy)

```
CREATE TRIGGER selall
ON Osoby
FOR INSERT, UPDATE
AS
SELECT * FROM Osoby
```




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 25 | 25 | NULL | Polakow | NULL | NULL | NULL | 16 |
| 26 | 26 | NULL | Ziółkowski | NULL | NULL | NULL | 16 |
| 27 | 27 | NULL | Jankowski | NULL | NULL | NULL | 7 |
| 28 | 28 | NULL | Paw | NULL | NULL | NULL | 9 |
| 29 | 29 | NULL | Lew | NULL | NULL | 1.93 | 13 |
| 30 | 30 | NULL | Ki | NULL | NULL | NULL | 13 |
| 31 | 31 | NULL | Dopisany | NULL | NULL | NULL | NULL |

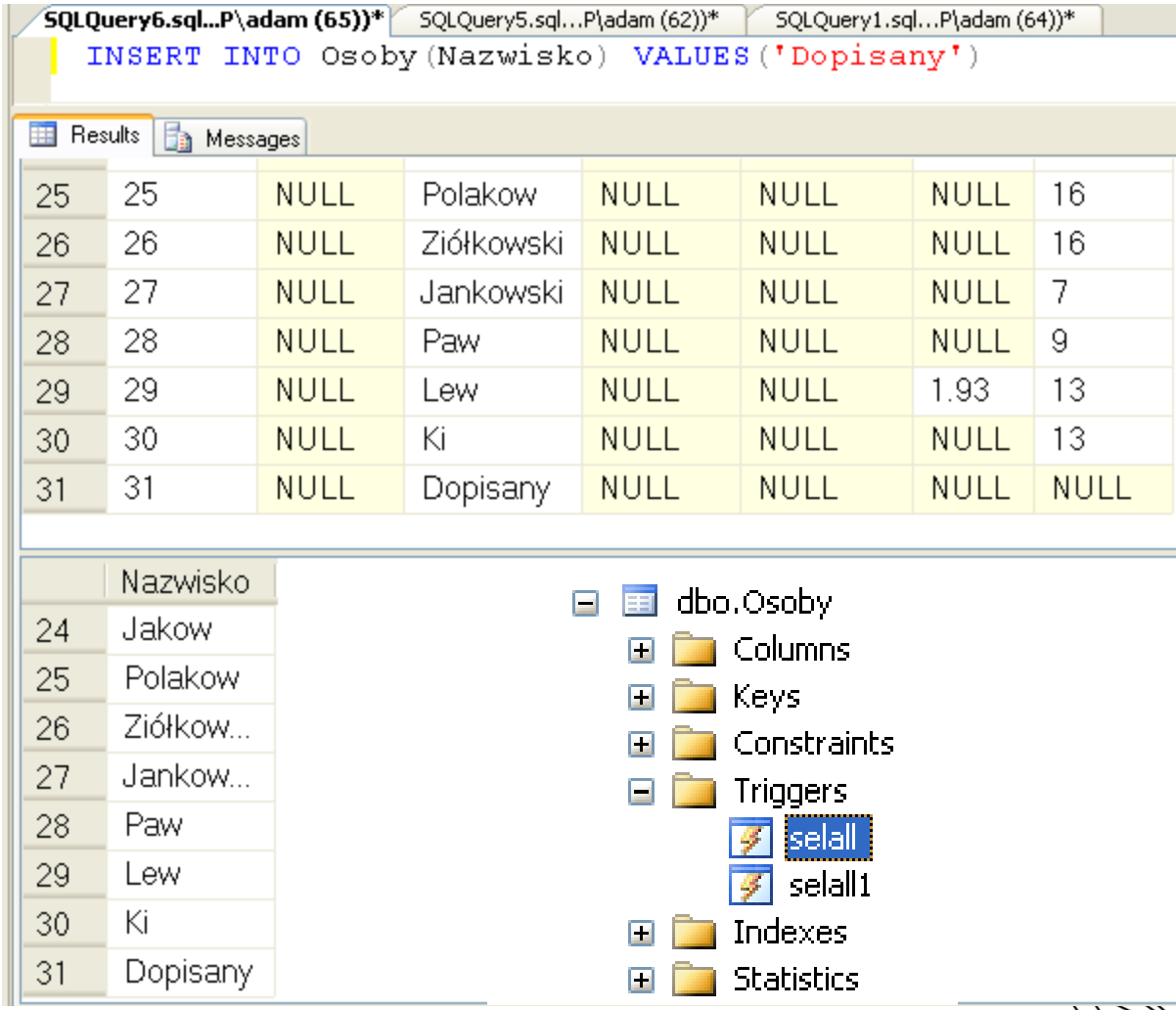
| | Nazwisko |
|---|---|
| 24 | Jakow |
| 25 | Polakow |
| 26 | Ziółkow... |
| 27 | Jankow... |
| 28 | Paw |
| 29 | Lew |
| 30 | Ki |
| 31 | Dopisany |

```
CREATE TRIGGER selall1
ON Osoby
FOR INSERT
AS
SELECT Nazwisko FROM Osoby
```

Skutkiem wykonania zapytania wstawiającego dane jest wykonanie obu triggerów. Ponieważ widać nowo wpisywane nazwisko triggery wykonują się po zdarzeniu

# Tworzenie triggerów – procedur wyzwalanych (wyzwalaczy)

```
DROP TRIGGER selall
GO
CREATE TRIGGER selall
ON Osoby
AFTER INSERT, UPDATE
AS
SELECT * FROM Osoby
```




```
DROP TRIGGER selall1
GO
CREATE TRIGGER selall1
ON Osoby
AFTER INSERT
AS
SELECT Nazwisko FROM Osoby
```

Zmiana FOR na AFTER nie zmienia sposobu działania

# Tworzenie triggerów – procedur wyzwalanych (wyzwalaczy)

```
CREATE TRIGGER selall
ON Osoby
AFTER INSERT, UPDATE, DELETE
AS
SELECT * FROM DELETED
SELECT * FROM INSERTED
```

| IdOsoby | IdDzialu | Nazwisko | Imie | RokUrodz | wzrost | IdSzefa |
|---|---|---|---|---|---|---|

```
INSERT INTO Osoby(Nazwisko) VALUES('Dopisany')
```

| IdOsoby | IdDzialu | Nazwisko | Imie | RokUrodz | wzrost | IdSzefa |
|---|---|---|---|---|---|---|
| 32 | NULL | Dopisany | NULL | NULL | NULL | NULL |

| IdOsoby | IdDzialu | Nazwisko | Imie | RokUrodz | wzrost | IdSzefa |
|---|---|---|---|---|---|---|
| 32 | NULL | Dopisany | NULL | NULL | NULL | NULL |
| 31 | NULL | Dopisany | NULL | NULL | NULL | NULL |

```
UPDATE Osoby SET Nazwisko='Dodany' WHERE Nazwisko='Dopisany‘
```

| IdOsoby | IdDzialu | Nazwisko | Imie | RokUrodz | wzrost | IdSzefa |
|---|---|---|---|---|---|---|
| 32 | NULL | Dodany | NULL | NULL | NULL | NULL |
| 31 | NULL | Dodany | NULL | NULL | NULL | NULL |

| IdOsoby | IdDzialu | Nazwisko | Imie | RokUrodz | wzrost | IdSzefa |
|---|---|---|---|---|---|---|
| 32 | NULL | Dodany | NULL | NULL | NULL | NULL |
| 31 | NULL | Dodany | NULL | NULL | NULL | NULL |

```
DELETE FROM Osoby WHERE Nazwisko= 'Dodany'
```

| IdOsoby | IdDzialu | Nazwisko | Imie | RokUrodz | wzrost | IdSzefa |
|---|---|---|---|---|---|---|

# Triggery – procedury wyzwalane (wyzwalacze)

```
CREATE TRIGGER up
ON Osoby
FOR INSERT, UPDATE
AS
UPDATE Osoby SET Nazwisko=UPPER(Nazwisko),  Imie =UPPER(Imie)
        PRINT 'Wykonano'
GO
```

**Wykonuje się dla wszystkich rekordów bez względu na liczbę modyfikowanych lub wstawianych rekordów**

```
UPDATE Osoby SET Nazwisko=LOWER(Nazwisko) WHERE IdOsoby >10
UPDATE Osoby Set RokUrodz=0 WHERE RokUrodz IS NULL
INSERT INTO Osoby(Nazwisko)  VALUES('Nowy')
```

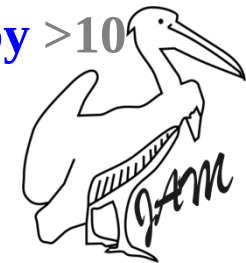
# Triggery – procedury wyzwalane (wyzwalacze)

## Z zastosowaniem systemowych tabel tymczasowych

```
CREATE TRIGGER up
ON Osoby
FOR INSERT, UPDATE
AS
UPDATE Osoby SET Nazwisko=UPPER(Nazwisko),  Imie =UPPER(Imie)
                    WHERE IdOsoby IN (SELECT IdOsoby FROM INSERTED)
PRINT 'Wykonano'
GO
```

Wykonuje się tylko dla modyfikowanych lub wstawianych rekordów

```
UPDATE Osoby SET Nazwisko=LOWER(Nazwisko) WHERE IdOsoby >10
UPDATE Osoby Set RokUrodz=0 WHERE RokUrodz IS NULL
INSERT INTO Osoby(Nazwisko)  VALUES('Nowy')
```



# Triggery – procedury wyzwalane (wyzwalacze)

## Z zastosowaniem systemowych tabel tymczasowych

```
CREATE TRIGGER up
ON Osoby
FOR INSERT, UPDATE
AS
IF UPDATE(Nazwisko) OR UPDATE(Imie)
BEGIN
UPDATE Osoby SET Nazwisko=UPPER(Nazwisko),  Imie =UPPER(Imie)
                    WHERE IdOsoby IN (SELECT IdOsoby FROM INSERTED)
PRINT 'Wykonano‘
PRINT COLUMNS_UPDATED ( )
END
ELSE
 PRINT 'Ominięto‘
PRINT COLUMNS_UPDATED ( )
GO

UPDATE Osoby SET Nazwisko=LOWER(Nazwisko)
WHERE IdOsoby >10
UPDATE Osoby Set RokUrodz=0 WHERE RokUrodz IS NULL
INSERT INTO Osoby(Nazwisko)  VALUES('Nowy')
```

**Wykonuje się tylko wtedy gdy modyfikowana jest jedna z dwóch kolumn wykonuje się również dla INSERT gdy nie puste wartości**

→ **0x04**

→ **0x10**

Wykonano
0x04
0x04

Ominięto
0x10

Wykonano
0x3F
0x3F

Wykonano
000100
000100

Ominięto
010000

Wykonano
111111
111111

# Triggery – procedury wyzwalane (wyzwalacze)
## Z zastosowaniem systemowych tabel tymczasowych

```
UPDATE Osoby SET Nazwisko=LOWER(Nazwisko)
WHERE IdOsoby >10
```

Wykonano
0x04
0x04

Wykonano
000100
000100

```
UPDATE Osoby Set RokUrodz=0 WHERE RokUrodz IS NULL
```

Ominięto
0x10

Ominięto
010000

```
INSERT INTO Osoby(Nazwisko)  VALUES('Nowy')
```

Wykonano
0x3F
0x3F

Wykonano
111111
111111

| | | |
|---|---|---|
| IdOsoby | 0x01 | 000001 |
| IdDzialu | 0x02 | 000010 |
| Nazwisko | 0x04 | 000100 |
| Imie | 0x08 | 001000 |
| RokUrodz | 0x10 | 010000 |
| Wzrost | 0x20 | 100000 |

# Triggery – procedury wyzwalane (wyzwalacze)

```
DROP TRIGGER upd
GO
CREATE TRIGGER upd  ON Osoby  FOR INSERT, UPDATE
AS
DECLARE            @max_v int, @act_v int
SELECT   @max_v = MAX(o.RokUrodz) FROM Osoby o, INSERTED i
WHERE o.IdOsoby <> i.IdOsoby
SELECT   @act_v = i.RokUrodz FROM INSERTED i
PRINT columns_updated()
PRINT @max_v
PRINT @act_v
if @act_v > @max_v
 BEGIN
     RAISERROR ('Wartość powinna być mniejsza równa %d a jest równa %d.',17,127,
@max_v, @act_v)
     ROLLBACK TRANSACTION
 END
GO
INSERT INTO Osoby(RokUrodz)
VALUES(1999)
```

**Nie zezwala na wprowadzenie osoby młodszej niż najmłodza istniejąca w tabeli**

```
Warning: Null value is eliminated by an aggregate
or other SET operation.
0x0C
1982
1999
Msg 3609, Level 16, State 1, Procedure up, Line 7
The transaction ended in the trigger. The batch
has been aborted.
Msg 50000, Level 17, State 127, Procedure upd,
Line 12
Wartość powinna być mniejsza równa 1982 a jest
równa 1999.
```

# Triggery – procedury wyzwalane (wyzwalacze)

## Zamiast

```
CREATE TRIGGER zamiast
ON Osoby
INSTEAD OF DELETE
AS
PRINT 'Zakaz kasowania'
GO
DELETE FROM Osoby
```

Nie zezwala na usuwanie wierszy z tabeli

```
Zakaz kasowania

(23 row(s) affected)
```

# Triggery – procedury wyzwalane (wyzwalacze)

## Zamiast

```
CREATE TRIGGER zamiast
ON Osoby
INSTEAD OF DELETE
AS
DECLARE @ile int
SELECT @ile=COUNT(IdZarobku) FROM Zarobki
        WHERE IdOsoby IN
                (SELECT IdOsoby FROM  DELETED)

IF (@ile >0)
PRINT 'Zakaz kasowania'
ELSE
DELETE FROM Osoby WHERE Idosoby IN
                (SELECT IdOsoby FROM  DELETED)
GO
```

Nie zezwala na usuwanie wierszy z tabeli dla osób, które mają jakiekolwiek zarobki

# Hierarchia triggerów DML

```
WITH Hierarchia AS (
SELECT hierarchyid::GetRoot() AS Wezel, type, type_name, parent_type,  0 AS
Level, CAST('' AS varchar(max)) AS Sciezka FROM sys.trigger_event_types
WHERE parent_type  IS NULL
UNION ALL
SELECT CAST( Wezel.ToString() +
CAST(ROW_NUMBER() OVER (PARTITION BY
sys.trigger_event_types.parent_type
ORDER BY sys.trigger_event_types.parent_type) AS varchar(30)) + '/' AS
hierarchyid),
sys.trigger_event_types.type,
sys.trigger_event_types.type_name,
sys.trigger_event_types.parent_type, Level+1, CAST(Sciezka +'|'AS varchar(max))
FROM sys.trigger_event_types JOIN Hierarchia
ON sys.trigger_event_types.parent_type = Hierarchia.type )
SELECT CAST(Wezel AS varchar(12)), Sciezka + REPLICATE('_',2*LEVEL) +
CAST (type as varchar(6)) +' '+
type_name+ ' ' +CAST(parent_type as varchar(6))+ CHAR(9)+ CAST(Level AS
varchar(1))
FROM Hierarchia ORDER BY Wezel
```

# Hierarchia triggerów DML

```
/           NULL
/2/         |__10016 DDL_DATABASE_LEVEL_EVENTS 10001          1
/2/1/       ||____241 RENAME 10016                            2
/2/2/       ||____10017 DDL_TABLE_VIEW_EVENTS 10016           2
/2/2/1/     |||______10018 DDL_TABLE_EVENTS 10017             3
/2/2/1/1/   ||||________21 CREATE_TABLE 10018                 4
/2/2/1/2/   ||||________22 ALTER_TABLE 10018                  4
/2/2/1/3/   ||||________23 DROP_TABLE 10018                   4
/2/2/2/     |||______10019 DDL_VIEW_EVENTS 10017              3
/2/2/2/1/   ||||________41 CREATE_VIEW 10019                  4
/2/2/2/2/   ||||________42 ALTER_VIEW 10019                   4
/2/2/2/3/   ||||________43 DROP_VIEW 10019                    4
/2/2/3/     |||______10020 DDL_INDEX_EVENTS 10017             3
/2/2/3/1/   ||||________24 CREATE_INDEX 10020                 4
/2/2/3/2/   ||||________25 ALTER_INDEX 10020                  4
/2/2/3/3/   ||||________26 DROP_INDEX 10020                   4
/2/2/3/4/   ||||________206 CREATE_XML_INDEX 10020           4
/2/2/3/5/   ||||________213 ALTER_FULLTEXT_INDEX 10020       4
/2/2/3/6/   ||||________224 CREATE_FULLTEXT_INDEX 10020      4
/2/2/3/7/   ||||________235 DROP_FULLTEXT_INDEX 10020        4
/2/2/3/8/   ||||________274 CREATE_SPATIAL_INDEX 10020       4
```

# Triggery na bazie danych

```
DROP TRIGGER Db_schemat ON DATABASE
GO
CREATE TRIGGER Db_schemat
ON DATABASE
FOR
-- CREATE_TABLE
-- CREATE_TABLE, ALTER_TABLE
-- DDL_TABLE_EVENTS
-- DDL_TABLE_VIEW_EVENTS
  DDL_DATABASE_LEVEL_EVENTS
AS
SELECT 'Zmiana bazy danych ' + user
```

```
CREATE Table xxx
(id int)
Zmiana bazy danych dbo

ALTER Table xxx
ADD pole varchar(3)
Zmiana bazy danych dbo

ALTER Table xxx
DROP COLUMN pole
Zmiana bazy danych dbo

DROP Table xxx
Zmiana bazy danych dbo

CREATE VIEW xxx
AS SELECT Nazwisko FROM Osoby
Zmiana bazy danych dbo
```

# Pełna struktura EVENT_INSTANCE znajduje się w

## http://schemas.microsoft.com/sqlserver/2006/11/eventdata/events.xsd

```
- <xs:complexType name="EVENT_INSTANCE_ADD_SIGNATURE_SCHEMA_OBJECT">
      - <xs:sequence>asic
 Envelope
        <xs:element name="EventType" type="SSWNAMEType" />
        <xs:element name="PostTime" type="xs:string" />
        <xs:element name="SPID" type="xs:int" />
Server Scoped DDL
        <xs:element name="ServerName" type="PathType" />
        <xs:element name="LoginName" type="SSWNAMEType" />
Server Scoped DDL
        <xs:element name="ServerName" type="PathType" />
        <xs:element name="LoginName" type="SSWNAMEType" />
DB Scoped DDL
        <xs:element name="UserName" type="SSWNAMEType" />
Main Body
        <xs:element name="DatabaseName" type="SSWNAMEType" />
        <xs:element name="SchemaName" type="SSWNAMEType" />
        <xs:element name="ObjectName" type="SSWNAMEType" />
        <xs:element name="ObjectType" type="SSWNAMEType" />
        <xs:element name="CounterSignature" type="xs:boolean" />
        <xs:element name="TSQLCommand" type="EventTag_TSQLCommand" />
        </xs:sequence>
        </xs:complexType>
      - <xs:complexType name="EVENT_INSTANCE_ALTER_APPLICATION_ROLE">
      - <xs:sequence>
Basic Envelope
        <xs:element name="EventType" type="SSWNAMEType" />
```

# Triggery na bazie danych

```
DROP TRIGGER Db_schemat ON DATABASE
GO
CREATE TRIGGER Db_schemat
ON DATABASE
            FOR  DDL_DATABASE_LEVEL_EVENTS
AS
SELECT 'Polecenie '+ EVENTDATA().value(
'(/EVENT_INSTANCE/TSQLCommand/CommandText)[1]','nvarchar(max)')  + '
wykonane przez - '+ user
```

```
CREATE Table xxx
(id int)
GO
ALTER Table xxx
ADD pole varchar(3)
GO
ALTER Table xxx
DROP COLUMN pole
GO
DROP Table xxx
GO
CREATE VIEW xxx
AS SELECT Nazwisko FROM Osoby
GO
DROP VIEW xxx
```

```
Polecenie CREATE Table xxx
(id int)
 wykonane przez - dbo

Polecenie ALTER Table xxx
ADD pole varchar(3)
 wykonane przez - dbo

Polecenie ALTER Table xxx
DROP COLUMN pole
 wykonane przez - dbo

Polecenie DROP Table xxx
 wykonane przez - dbo

Polecenie CREATE VIEW xxx
AS SELECT Nazwisko FROM Osoby
 wykonane przez - dbo
```

# Triggery na bazie danych

```
DROP TRIGGER Db_schemat ON DATABASE
GO
CREATE TRIGGER Db_schemat
ON DATABASE
            FOR  DDL_DATABASE_LEVEL_EVENTS
AS
SELECT 'Polecenie '+ EVENTDATA().value(
'(/EVENT_INSTANCE/EventType)[1]','nvarchar(max)')  + ' wykonane przez - '+ user
```

```
CREATE Table xxx
(id int)
GO
ALTER Table xxx
ADD pole varchar(3)
GO
ALTER Table xxx
DROP COLUMN pole
GO
DROP Table xxx
GO
CREATE VIEW xxx
AS SELECT Nazwisko FROM Osoby
GO
DROP VIEW xxx
```

Polecenie CREATE_TABLE wykonane przez - dbo

Polecenie ALTER_TABLE wykonane przez - dbo

Polecenie ALTER_TABLE wykonane przez - dbo

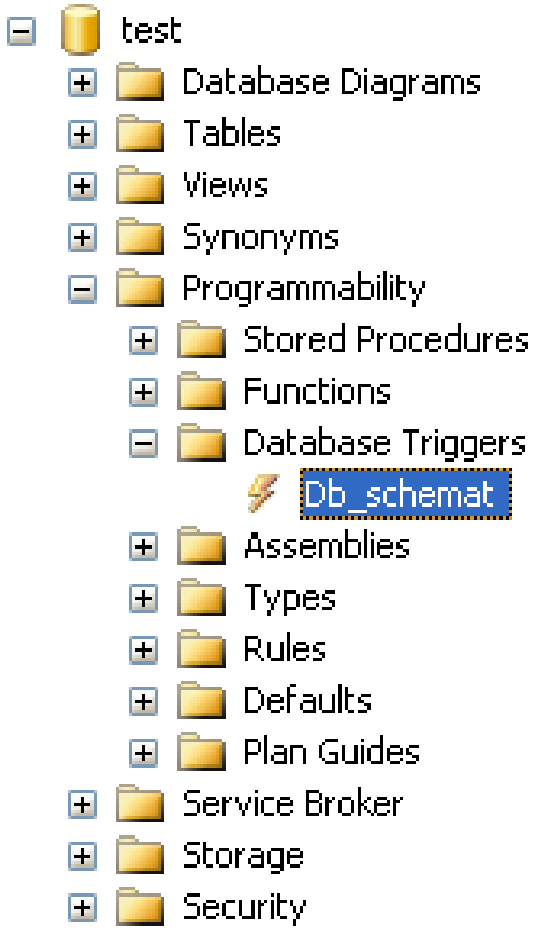
Polecenie DROP_TABLE wykonane przez - dbo

Polecenie CREATE_VIEW wykonane przez - dbo

Polecenie DROP_VIEW wykonane przez - dbo

# Triggery na bazie danych

```
DROP TRIGGER Db_schemat ON DATABASE
GO
CREATE TRIGGER Db_schemat
ON DATABASE
            FOR  DDL_DATABASE_LEVEL_EVENTS
AS
SELECT 'Polecenie '+ EVENTDATA().value(
'(/EVENT_INSTANCE/EventType)[1]','nvarchar(max)'
)  + ' wykonane przez - '+ user
```




# Struktura XML *EVENT_INSTANCE* zwracana przez funkcję EVENTDATA().value dla większości poleceń

```
<EVENT_INSTANCE>
  <EventType>type</EventType>
  <PostTime>date-time</PostTime>
  <SPID>spid</SPID>
  <ServerName>name</ServerName>
  <LoginName>name</LoginName>
  <UserName>name</UserName>
  <DatabaseName>name</DatabaseName>
  <SchemaName>name</SchemaName>
  <ObjectName>name</ObjectName>
  <ObjectType>type</ObjectType>
  <TSQLCommand>command</TSQLCommand>
</EVENT_INSTANCE>
```

**Zawsze**

**Często**

```
CREATE TABLE Audyt_ddl
(
IdAudyt_ddl int IDENTITY(1,1) PRIMARY KEY,
EventType varchar(30),
PostTime varchar(30),
SPID varchar(30),
ServerName varchar(30),
LoginName varchar(30),
UserName varchar(30),
DatabaseName varchar(30),
SchemaName varchar(30),
ObjectName varchar(30),
ObjectType varchar(30),
TSQLCommand varchar(max)
)
```

# Triggery na bazie danych

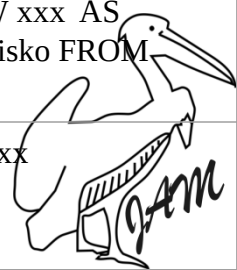
# Triggery na bazie danych

```
DROP TRIGGER Db_schemat ON DATABASE
GO
CREATE TRIGGER Db_schemat
ON DATABASE
        FOR  DDL_DATABASE_LEVEL_EVENTS
AS
INSERT INTO Audyt_ddl VALUES(
EVENTDATA().value('(/EVENT_INSTANCE/EventType)[1]','varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/PostTime)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/SPID)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/ServerName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/LoginName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/UserName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/DatabaseName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/SchemaName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/ObjectName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/ObjectType)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/TSQLCommand)[1]',' varchar(max)')
)
```

```
CREATE Table xxx
(id int)
GO
ALTER Table xxx
ADD pole varchar(3)
GO
ALTER Table xxx
DROP COLUMN pole
GO
DROP Table xxx
GO
CREATE VIEW xxx
AS SELECT Nazwisko FROM Osoby
GO
DROP VIEW xxx
```

# Triggery na bazie danych

| IdAudyt_ddl | EventType | PostTime | SPID | ServerName | LoginName | UserName | DatabaseName | SchemaName | ObjectName | ObjectType | TSQLCommand |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CREATE_TABLE | 2010-11-14T16:29:59.620 | 55 | PC22431 | sa | dbo | test | dbo | xxx | TABLE | CREATE Table xxx  (id int) |
| 2 | ALTER_TABLE | 2010-11-14T16:29:59.887 | 55 | PC22431 | sa | dbo | test | dbo | xxx | TABLE | ALTER Table xxx  ADD pole varchar(3) |
| 3 | ALTER_TABLE | 2010-11-14T16:29:59.917 | 55 | PC22431 | sa | dbo | test | dbo | xxx | TABLE | ALTER Table xxx  DROP COLUMN pole |
| 4 | DROP_TABLE | 2010-11-14T16:29:59.933 | 55 | PC22431 | sa | dbo | test | dbo | xxx | TABLE | DROP Table xxx |
| 5 | CREATE_VIEW | 2010-11-14T16:29:59.933 | 55 | PC22431 | sa | dbo | test | dbo | xxx | VIEW | CREATE VIEW xxx  AS SELECT Nazwisko FROM Osoby |
| 6 | DROP_VIEW | 2010-11-14T16:29:59.933 | 55 | PC22431 | sa | dbo | test | dbo | xxx | VIEW | DROP VIEW xxx |



# Triggery na serwerze

```
DROP TRIGGER  dla_logowania ON ALL SERVER

GO

CREATE TRIGGER dla_logowania
ON ALL SERVER
FOR LOGON
AS
PRINT 'Nastąpiło logowanie'
--SELECT EVENTDATA().value(
--'(/EVENT_INSTANCE/TSQLCommand/CommandText)[1]','varchar(max)')
GO
```

***Próba wykonania SELECT powoduje wygenerowanie komunikatu o błędzie natomiast PRINT nie pozostawia żadnego śladu na konsoli***

# Triggery na serwerze

```
USE master
GO
DROP TABLE Logowanie
GO
CREATE TABLE Logowanie
(
IdLogowanie int IDENTITY(1,1) PRIMARY KEY,
EventType varchar(30),
PostTime varchar(30),
SPID varchar(30),
ServerName varchar(30),
LoginName varchar(30),
UserName varchar(30),
DatabaseName varchar(30),
SchemaName varchar(30),
ObjectName varchar(30),
ObjectType varchar(30),
TSQLCommand varchar(max)
)
GO

GRANT INSERT ON Logowanie TO public
```

W przypadku stworzenia tabeli w innej bazie niż master należy dodać dodatkowe uprawnienia do tej bazy dla roli public

# Triggery na serwerze

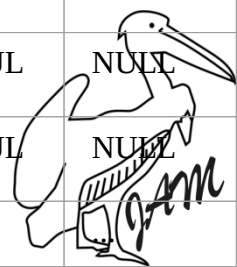
```
DROP TRIGGER  dla_logowania ON ALL SERVER
GO
CREATE TRIGGER dla_logowania
ON ALL SERVER
FOR LOGON
AS
INSERT INTO master.dbo.Logowanie VALUES(
EVENTDATA().value('(/EVENT_INSTANCE/EventType)[1]','varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/PostTime)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/SPID)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/ServerName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/LoginName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/UserName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/DatabaseName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/SchemaName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/ObjectName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/ObjectType)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/TSQLCommand)[1]',' varchar(max)')
)

GO
```

Po zalogowaniu jako inny i sa oraz krótkim czasie bezczynnym zawartość tabeli ma postać

# Triggery na serwerze

| IdLogowanie | EventType | PostTime | SPID | ServerName | LoginName | UserName | DatabaseName | SchemaName | ObjectName | ObjectType | TSQLCommand |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 6 | LOGON | 2010-11-17T10:58:34.290 | 52 | PC22431 | ZARZĄDZANIENT\SYSTEM | NULL | NULL | NULL | NULL | NULL | NULL |
| 7 | LOGON | 2010-11-17T10:58:34.303 | 55 | PC22431 | inny | NULL | NULL | NULL | NULL | NULL | NULL |
| 8 | LOGON | 2010-11-17T10:58:35.087 | 52 | PC22431 | ZARZĄDZANIENT\SYSTEM | NULL | NULL | NULL | NULL | NULL | NULL |
| 9 | LOGON | 2010-11-17T10:58:35.460 | 57 | PC22431 | inny | NULL | NULL | NULL | NULL | NULL | NULL |
| 10 | LOGON | 2010-11-17T10:58:35.477 | 57 | PC22431 | inny | NULL | NULL | NULL | NULL | NULL | NULL |
| 11 | LOGON | 2010-11-17T10:58:35.507 | 57 | PC22431 | inny | NULL | NULL | NULL | NULL | NULL | NULL |
| 12 | LOGON | 2010-11-17T10:58:44.107 | 52 | PC22431 | ZARZĄDZANIENT\SYSTEM | NULL | NULL | NULL | NULL | NULL | NULL |
| 13 | LOGON | 2010-11-17T10:58:45.130 | 52 | PC22431 | ZARZĄDZANIENT\SYSTEM | NULL | NULL | NULL | NULL | NULL | NULL |
| 14 | LOGON | 2010-11-17T10:58:52.747 | 57 | PC22431 | sa | NULL | NULL | NULL | NULL | NULL | NULL |
| 15 | LOGON | 2010-11-17T10:58:54.137 | 58 | PC22431 | sa | NULL | NULL | NULL | NULL | NULL | NULL |
| 16 | LOGON | 2010-11-17T10:58:54.170 | 58 | PC22431 | sa | NULL | NULL | NULL | NULL | NULL | NULL |
| 17 | LOGON | 2010-11-17T10:58:54.130 | 58 | PC22431 | sa | NULL | NULL | NULL | NULL | NULL | NULL |
| 18 | LOGON | 2010-11-17T10:58:54.463 | 52 | PC22431 | ZARZĄDZANIENT\SYSTEM | NULL | NULL | NULL | NULL | NULL | NULL |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |



# Triggery na serwerze

Aby wyeliminować cykliczne logowania oraz pozbyć się pustych wpisów należy zmienić tabelę i trigger

```
USE master
GO
DROP TABLE Logowanie
GO
CREATE TABLE Logowanie
(
IdLogowanie int IDENTITY(1,1) PRIMARY KEY,
EventType varchar(30),
PostTime varchar(30),
SPID varchar(30),
ServerName varchar(30),
LoginName varchar(30)
)
GO
GRANT INSERT ON Logowanie TO public
```

# Triggery na serwerze

Aby wyeliminować cykliczne logowania oraz pozbyć się pustych wpisów należy zmienić tabelę i trigger

```
DROP TRIGGER  dla_logowania ON ALL SERVER
GO
CREATE TRIGGER dla_logowania
ON ALL SERVER
FOR LOGON
AS
DECLARE @kto AS varchar(30)
SET @kto= EVENTDATA().value('(/EVENT_INSTANCE/LoginName)[1]',' varchar(30)')
IF( @kto <>'ZARZĄDZANIE NT\SYSTEM')
INSERT INTO master.dbo.Logowanie VALUES(
EVENTDATA().value('(/EVENT_INSTANCE/EventType)[1]','varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/PostTime)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/SPID)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/ServerName)[1]',' varchar(30)'),
@kto
)
GO
```

# Triggery na serwerze

Skutek po przełączeniu użytkowników

| IdLogowanie | EventType | PostTime | SPID | ServerName | LoginName |
|---|---|---|---|---|---|
| 1 | LOGON | 2010-11-17T11:23:18.517 | 55 | PC22431 | inny |
| 2 | LOGON | 2010-11-17T11:23:19.687 | 56 | PC22431 | inny |
| 3 | LOGON | 2010-11-17T11:23:19.700 | 56 | PC22431 | inny |
| 4 | LOGON | 2010-11-17T11:23:19.717 | 56 | PC22431 | inny |
| 5 | LOGON | 2010-11-17T11:23:38.060 | 56 | PC22431 | sa |

# Triggery na serwerze

Puste wpisy wynikają z tego, że struktura EVENT_INSTANCE dla logowania ma postać

```
<EVENT_INSTANCE>
    <EventType>event_type</EventType>
    <PostTime>post_time</PostTime>
    <SPID>spid</SPID>
    <ServerName>server_name</ServerName>
    <LoginName>login_name</LoginName>
    <LoginType>login_type</LoginType>
    <SID>sid</SID>
    <ClientHost>client_host</ClientHost>
    <IsPooled>is_pooled</IsPooled>
</EVENT_INSTANCE>
```

# Triggery na serwerze

```
USE master
GO
DROP TABLE Autoryzacja
GO
CREATE TABLE Autoryzacja
(
IdAutoryzacja int IDENTITY(1,1) PRIMARY KEY,
EventType varchar(30),
PostTime varchar(30),
SQLInstance varchar(30),
SPID varchar(30),
ServerName varchar(30),
LoginName varchar(30),
LoginType varchar(30),
SID varchar(30),
ObjectName  varchar(30),
ObjectType varchar(30),
DefaultLanguage varchar(30),
DefaultDatabase  varchar(30)
)
```

# Triggery na serwerze

```
DROP TRIGGER  dla_loginow ON ALL SERVER
GO
CREATE TRIGGER dla_loginow
ON ALL SERVER
FOR DDL_LOGIN_EVENTS
AS
INSERT INTO master.dbo.Autoryzacja VALUES(
EVENTDATA().value('(/EVENT_INSTANCE/EventType)[1]','varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/PostTime)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/SQLInstance)[1]','varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/SPID)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/ServerName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/LoginName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/LoginType)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/SID)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/ObjectName)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/ObjectType)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/DefaultLanguage)[1]',' varchar(30)'),
EVENTDATA().value('(/EVENT_INSTANCE/DefaultDatabase)[1]',' varchar(30)')
)
GO
```

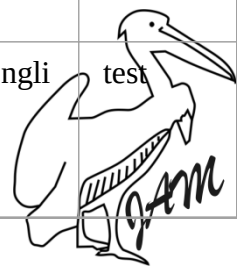
# Triggery na serwerze

```
sp_addlogin 'dodany', 'kajak', 'test'
GO
SELECT *  FROM master.dbo.Autoryzacja
```

| IdAutoryzacja | EventType | PostTime | SQLInstance | SPID | ServerName | LoginName | LoginType | SID | ObjectName | ObjectType | DefaultLanguage | DefaultDatabase |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CREATE_LOGIN | 2010-11-17T13:26:53.457 | NULL | 56 | PC22431 | sa | SQLLogin | 3A9RNtEDGEac51e5G+lFLQ== | dodany | LOGIN | us_english | test |

```
GO
sp_droplogin 'dodany‘
GO
SELECT *  FROM master.dbo.Autoryzacja
```

| IdAutoryzacja | EventType | PostTime | SQLInstance | SPID | ServerName | LoginName | LoginType | SID | ObjectName | ObjectType | DefaultLanguage | DefaultDatabase |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | CREATE_LOGIN | 2010-11-17T13:26:53.457 | NULL | 56 | PC22431 | sa | SQLLogin | 3A9RNtEDGEac51e5G+lFLQ== | dodany | LOGIN | us_english | test |
| 2 | DROP_LOGIN | 2010-11-17T13:26:54.283 | NULL | 56 | PC22431 | sa | SQLLogin | 3A9RNtEDGEac51e5G+lFLQ== | dodany | LOGIN | us_english | test |



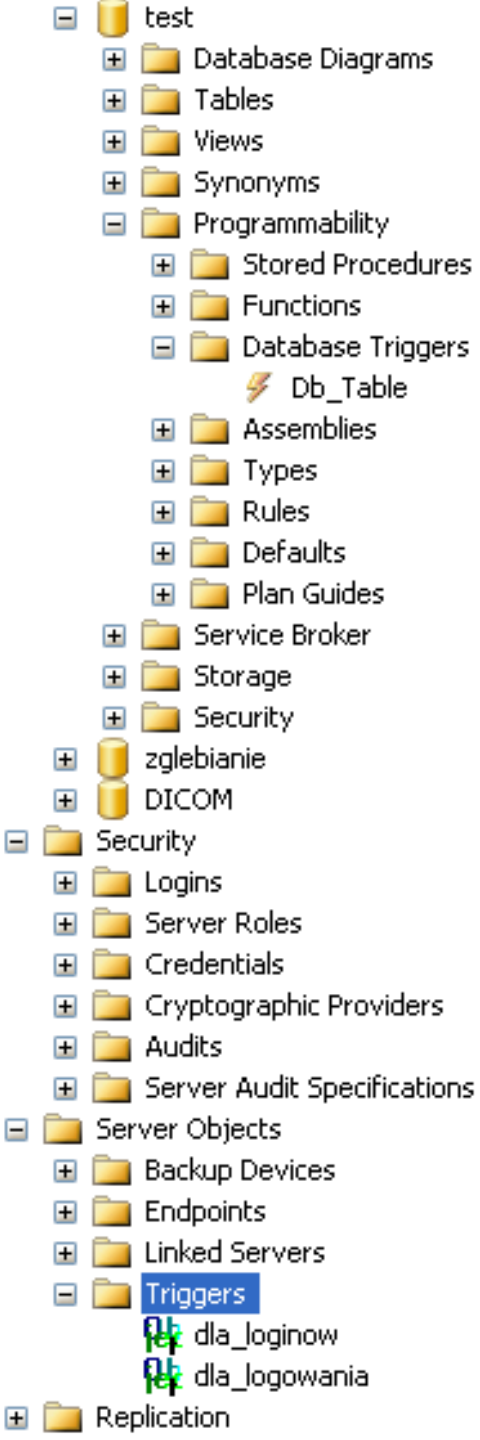
# Tworzenie triggera dla potrzeb audytu

```
CREATE TABLE ddl_log (PostTime datetime, DB_User nvarchar(100),
Event nvarchar(100), TSQL nvarchar(2000));
GO
CREATE TRIGGER log
ON DATABASE
FOR DDL_DATABASE_LEVEL_EVENTS
AS
DECLARE @data XML
SET @data = EVENTDATA()
INSERT ddl_log
   (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)')
    ) ;
GO
```

# Tworzenie triggera dla potrzeb audytu

```
CREATE VIEW xxx
AS SELECT Nazwisko FROM Osoby
GO
DROP VIEW xxx
```

| PostTime | DB_ User | Event | TSQL |
|---|---|---|---|
| 2010-11-17 17:30:27.673 | dbo | CREATE_ TABLE | CREATE TABLE ddl_log (PostTime datetime, DB_User nvarchar(100), Event nvarchar(100), TSQL nvarchar(2000)); |
| 2010-11-17 17:33:20.683 | dbo | CREATE_ VIEW | CREATE VIEW xxx AS SELECT Nazwisko FROM Osoby |
| 2010-11-17 17:33:20.697 | dbo | DROP_VI EW | DROP VIEW xxx |




# Tworzenie triggera dla potrzeb audytu

Różne miejsca
przechowywania triggerów
dla bazy danych i dla serwera

# Wyłączanie i włączanie triggera

**DISABLE TRIGGER { [ *schema* . ] *trigger_name* [ ,...*n* ] | ALL }**
**ON { *object_name* | DATABASE | ALL SERVER } [ ; ]**

**ENABLE TRIGGER { [ *schema_name* . ] *trigger_name* [ ,...*n* ] | ALL }**
**ON { *object_name* | DATABASE | ALL SERVER } [ ; ]**

# Wyłączanie i włączanie triggera

```
DISABLE TRIGGER Db_schemat  ON DATABASE
GO

DISABLE TRIGGER ALL ON DATABASE
GO

ENABLE TRIGGER Db_schemat ON DATABASE
GO

ENABLE Trigger ALL ON DATABASE
GO
```

Nie ma różnicy w graficznej prezentacji wyłączonego triggera
Powtórzenie polecenia ENABLE lub DISABLE dla tego samego triggera nie powoduje komunikatu o błędzie

# Wyłączanie i włączanie triggera

```
DISABLE TRIGGER dla_loginow  ON ALL SERVER
GO

DISABLE TRIGGER ALL ON ALL SERVER
GO

ENABLE TRIGGER dla_loginow  ON ALL SERVER
GO

ENABLE Trigger ALL ON ALL SERVER
GO
```

Nie ma różnicy w graficznej prezentacji wyłączonego triggera
Powtórzenie polecenia ENABLE lub DISABLE dla tego samego triggera nie powoduje komunikatu o błędzie

# Następnie kursory