BAZY DANYCH

Modyfikacja bazy danych INSERT, UPDATE, DELETE

mgr inż. Patryk Kaczmarek

patryk.kaczmarek@ansleszno.pl

Plan laboratorium

- INSERT
- UPDATE
- DELETE
- CREATE TABLE
- TRIGGER
- Deklaracja zmiennych

INSERT

Wstawia nowy rekord do tabeli:

Ogólny model polecenia

```
INSERT INTO nazwa_tabeli
(atrybut1, atrybut2, atrybut3)
VALUES (wart1, wart2, wart3)
```

INSERT - przykład

```
INSERT INTO [dbo].[ZESPOLY]
           ([ID_ZESP]
           ,[NAZWA]
           ,[ADRES])
     VALUES
           (60
           ,'ZESPÓŁ PIERWSZY'
           ,'LESZNO')
GO
```

UPDATE

Aktualizuje rekord w bazie spełniający określony warunek

Ogólny model polecenia

```
UPDATE nazwa_tabeli
SET atrybut1 = wartosc1, atrubut2 = wartosc2
WHERE atrybut = `cos`
```

UPDATE - przykład

```
UPDATE [dbo].[ZESPOLY]
   SET[ADRES] = 'LESZNO, MICKIEWICZA'
 WHERE ID_ZESP = 60
```

DELETE

Usuwa rekord spełniający warunek

Ogólny model polecenia

```
DELETE FROM nazwa_tabeli
WHERE atrybut = `cos`
```

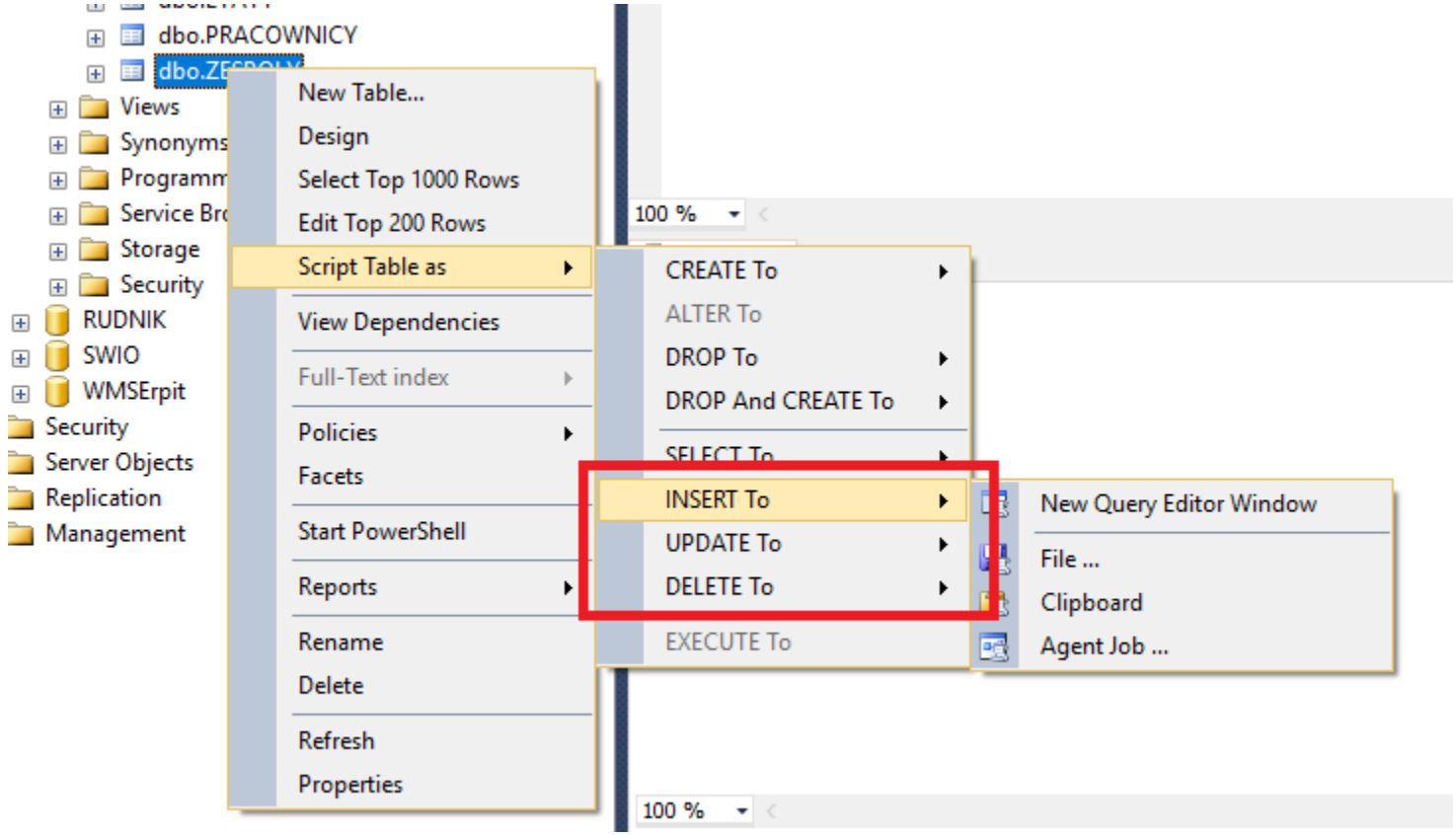
DELETE - przykład

```
DELETE FROM [dbo].[ZESPOLY]
      WHERE ID_ZESP=60
```

UPDATE, DELETE - UWAGA

Jeśli nie podamy warunku WHERE będzie on spełniony dla wszystkich rekordów i zostaną zmodyfikowane wszystkie

MS SQL - Podpowiedź



CREATE TABLE

Tworzy tabele w bazie danych

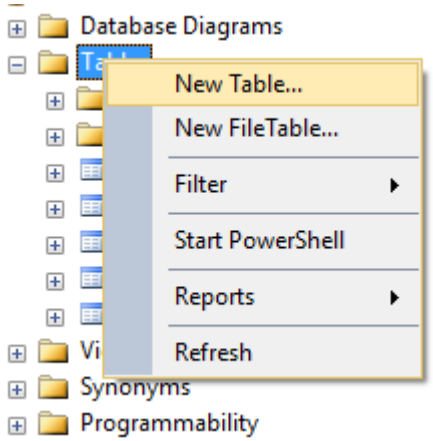
Ogólny model polecenia

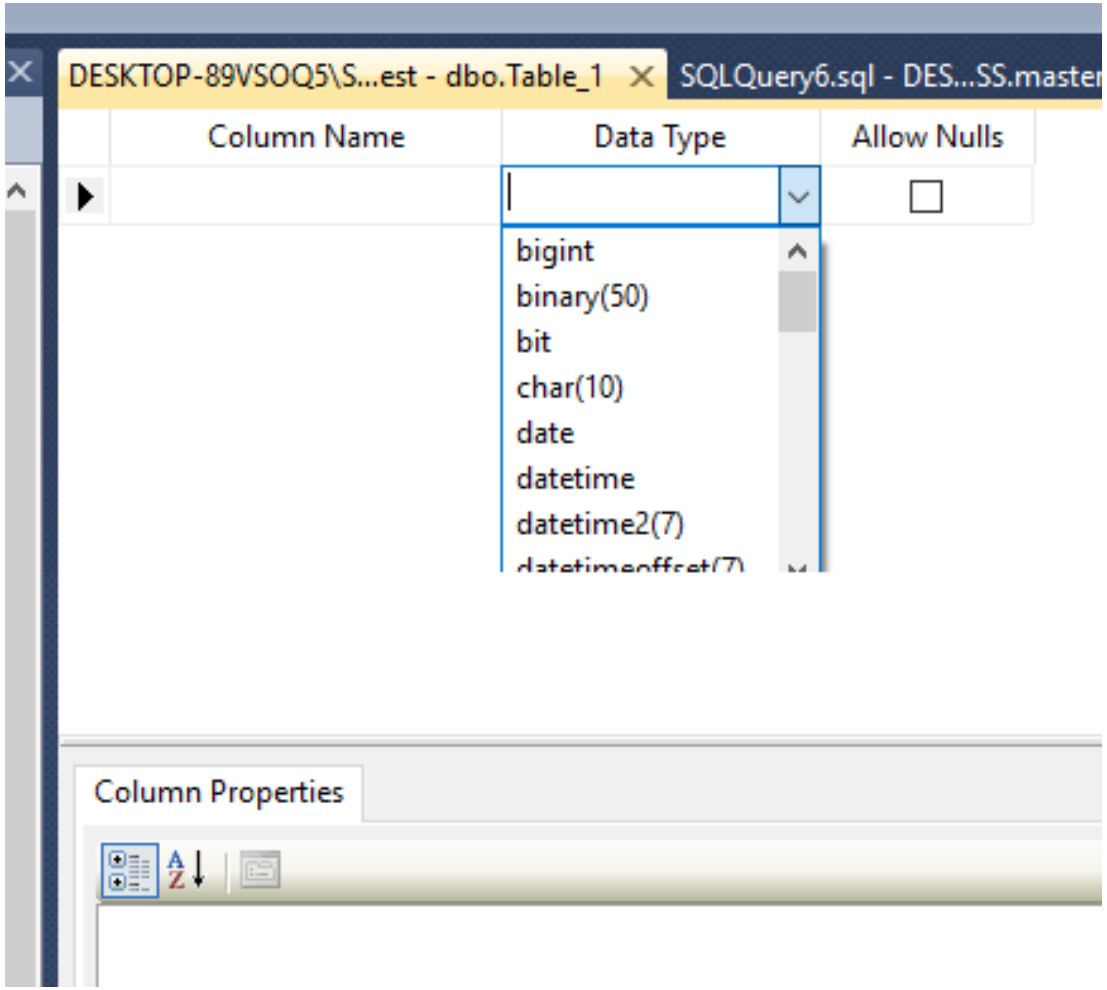
```
CREATE TABLE nazwa_tabeli
( nazwa_kolumny1 typKolumny1,
nazwa_kolumny2 typKolumny2)
```

CREATE TABLE - przykład

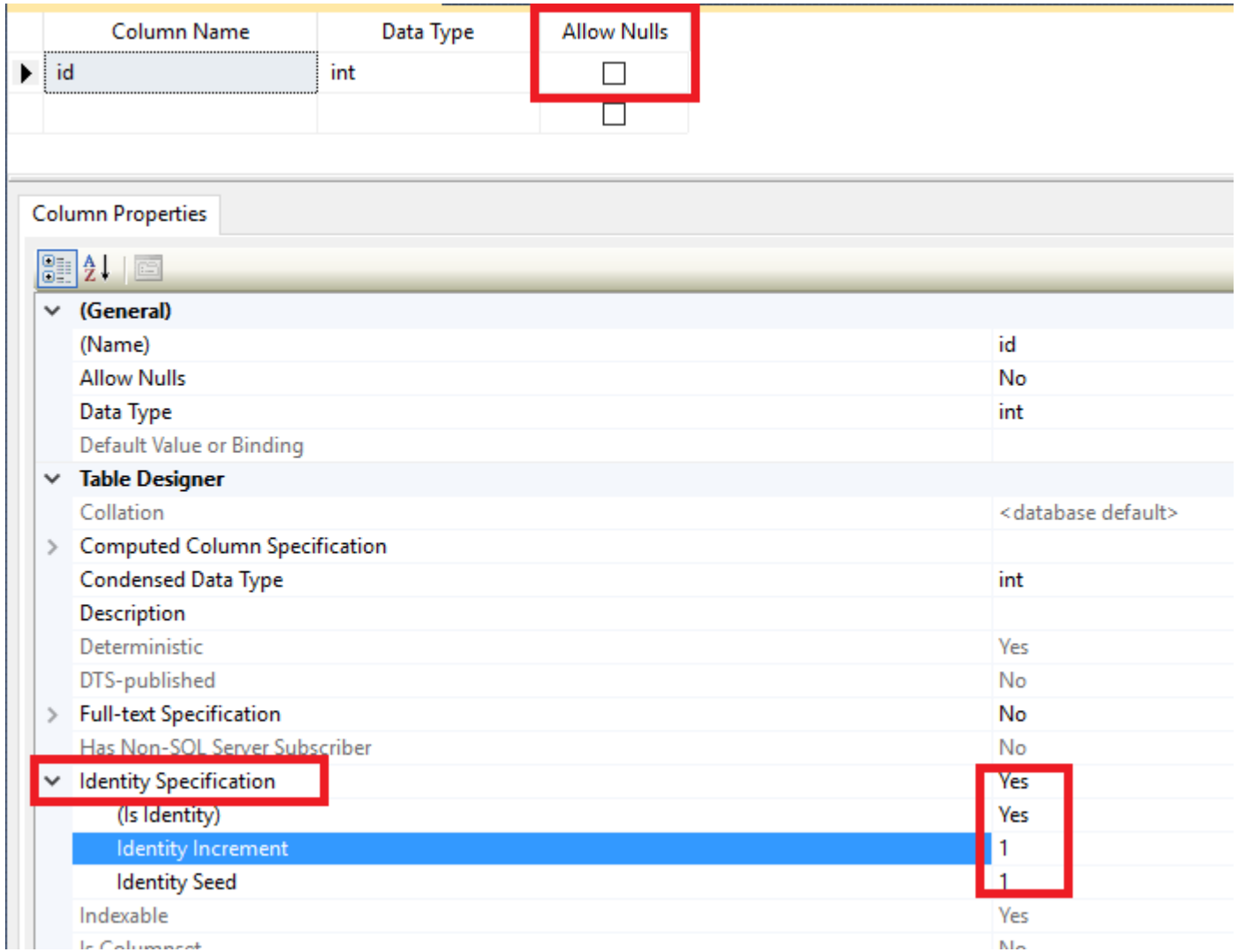
```
CREATE TABLE [dbo].[INF_XXXXX](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [imie] [varchar](100) NOT NULL,
    [nazwisko] [varchar](100) NOT NULL,
    [plec] [varbinary](1) NOT NULL,
    [data_urodzenia] [date] NOT NULL
)
```

CREATE TABLE – nie musimy ręcznie





CREATE TABLE – Unikalne pole ID



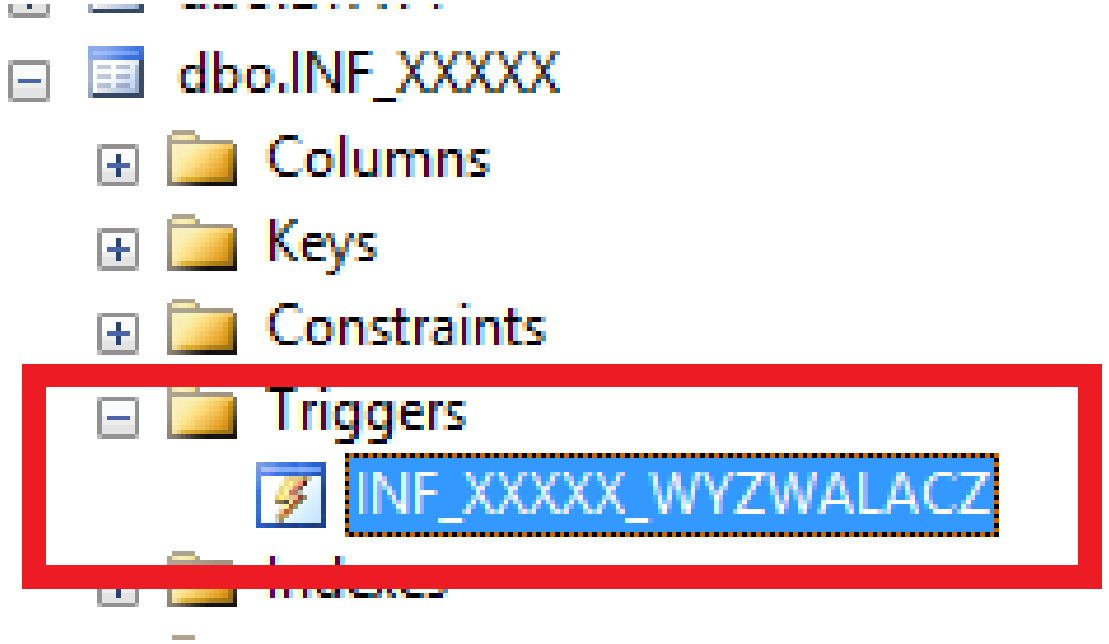
TRIGGER - wyzwalacz

Są to instrukcje, które pozwalają nam na wykonywanie pewnych operacji w trakcie wywołania pewnych instrukcji. Wyzwalacze pozwalają na zmianę wartości kolumn w odpowiedzi na instrukcję INSERT, UPDATE, DELETE wykonaną na tabeli.

TRIGGER - tworzenie

```
CREATE TRIGGER nazwa_wyzwalacza
ON nazwa_tabeli
FOR akcja
AS
„co ma się stać”
```

TRIGGER - podgląd



TRIGGER – podgląd globalny

```
SELECT
     sysobjects.name AS "Nazwa wyzwalacza"
    ,OBJECT_NAME(parent_obj) AS "Tabela"
    ,OBJECTPROPERTY( id, 'ExecIsUpdateTrigger') AS "UPDATE"
    ,OBJECTPROPERTY( id, 'ExecIsDeleteTrigger') AS "DELETE"
    ,OBJECTPROPERTY( id, 'ExecIsInsertTrigger') AS "INSERT"
FROM sysobjects
WHERE sysobjects.type = 'TR' AND sysobjects.name like
'%INF_XXXXX%';
```

Komunikat w TRIGGER

Trigger wykonuje się po wykonaniu operacji, więc przed wyświetleniem komunikatu musimy cofnąć transakcje bazy danych ROLLBACK.

Wyświetlenie komunikatu:

```
RAISERROR(message, severity, state)
```

Gdzie:
Message - dowolny tekst (komunikat błędu)
Severity - liczba z przedziału 0-25 (przy czym użytkownik może używać wartości z przedziału 0-18
State - liczba z przedziału 1-127

Komunikat w TRIGGER - przykład

```
BEGIN
ROLLBACK
RAISERROR('Komunikat',1,2)
END
```

MS SQL – deklaracja zmiennych

Potrzebne przy tworzeniu bardziej zaawansowanego TRIGGERA !

```
DECLARE @nazwa_zmiennej typ;
```

```
DECLARE @zmienna varchar(50);
```

Przypisanie wartości w trigger INSERT

```
@zmienna = (SELECT atrybut FROM inserted)
```