

Bazy danych – Semestr 2

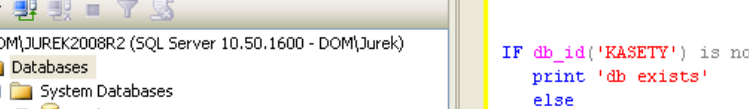
Zajęcia nr 5

Pisanie skryptów
kontynuacja

Zakres zajęć

- Pisanie skryptu do tworzenia bazy danych „Wypożyczalnia kaset wideo” - **kontynuacja**
- Procedury Wyzwalane (Wyzwalacze, Triggery)
- Funkcja Skalarna
- Funkcja Tabelarna

Sprawdzanie występowania bazy danych na serwerze



The screenshot shows the SQL Server Enterprise Manager interface. The left pane displays the server hierarchy for 'DOM\JUREK2008R2 (SQL Server 10.50.1600 - DOM\Jurek)'. Under 'Databases', the 'KASETY' database is highlighted. The right pane shows a SQL query window titled 'SQLQuery1.sql - DOM\...2)))*' containing the following T-SQL code:

```
IF db_id('KASETY') is not null
    print 'db exists'
else
    print 'brak bazy'
```

Below the query window, the 'Messages' tab is active, displaying the output: 'db exists'.

The screenshot displays the SQL Server Enterprise Manager interface on the left and the SQL Server Enterprise Query window on the right.

SQL Server Enterprise Manager (Left Panel):

- Server: **SQLSERVER**
- Database: **sys**
- System Tables: **sys.sysdatabases** (selected)
- Columns: **name**, **dbid**, **sid**
- Triggers: **sys.sysdatabases**
- Indexes: **sys.sysdatabases**
- Statistics: **sys.sysdatabases**
- sys.sysdepends
- sys.sysdevices
- sys.sysfilegroups
- sys.sysfiles
- sys.sysforeignkeys
- sys.sysfulltextcatalogs
- sys.sysindexes
- sys.sysindexkeys
- sys.syslanguages
- sys.syslockinfo
- sys.syslogins
- sys.sysmembers
- sys.sysmessages
- sys.sysobjects

SQL Server Enterprise Query (Right Panel):

Query Text:

```
select * from sys.sysdatabases
```

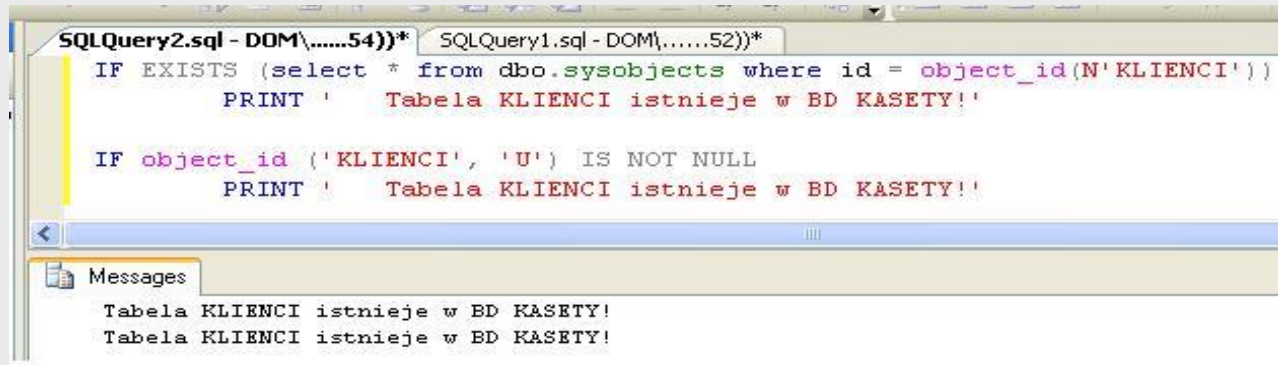
Results:

	name	dbid	sid
1	master	1	0x01
2	tempdb	2	0x01
3	model	3	0x01
4	msdb	4	0x01
5	ReportServer\$JUREK2008R2	5	0x010500000000000005150000005729C
6	ReportServer\$JUREK2008R2TempDB	6	0x010500000000000005150000005729C
7	AAA	7	0x010500000000000005150000005729C
8	KASETY	8	0x010500000000000005150000005729C
9	KASETY_NOWA	9	0x010500000000000005150000005729C
10	WULKANIZACJA	10	0x010500000000000005150000005729C
11	MOJA	11	0x010500000000000005150000005729C
12	kasety_nnn	12	0x010500000000000005150000005729C
13	BBB	13	0x010500000000000005150000005729C
14	Z502_03	14	0x010500000000000005150000005729C
15	ppp	15	0x010500000000000005150000005729C
16	UUU	16	0x010500000000000005150000005729C

Uwaga do zajęć poprzednich

Sprawdzanie występowania tabeli w bazie danych

- Oba poniższe zapisy mają tę samą funkcjonalność
 - IF EXISTS (select * from dbo.sysobjects where id = object_id(N'KLIENCI'))
PRINT ' Tabela KLIENCI istnieje w BD KASETY!'
 - IF object_id ('KLIENCI', 'U') IS NOT NULL
PRINT ' Tabela KLIENCI istnieje w BD KASETY!'
 - 'KLIENCI' : parametr **Name** tabeli sysobject;
 - 'U' parametr **Type** tabeli sysobject **oznaczający tabelę!!**

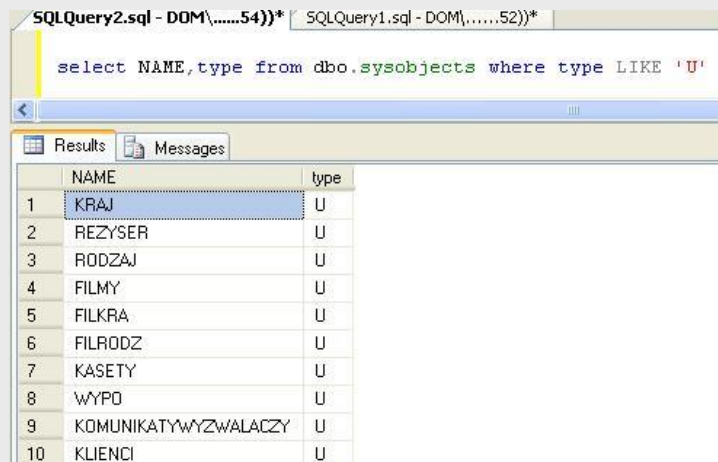


The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'SQLQuery2.sql - DOM\.....54))' and 'SQLQuery1.sql - DOM\.....52))'. The active tab is 'SQLQuery2.sql', which contains the following T-SQL code:

```
IF EXISTS (select * from dbo.sysobjects where id = object_id(N'KLIENCI'))  
    PRINT ' Tabela KLIENCI istnieje w BD KASETY!'  
  
IF object_id ('KLIENCI', 'U') IS NOT NULL  
    PRINT ' Tabela KLIENCI istnieje w BD KASETY!'
```

Below the query window, the 'Messages' pane displays the output of the query:

```
Tabela KLIENCI istnieje w BD KASETY!  
Tabela KLIENCI istnieje w BD KASETY!
```



The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'SQLQuery2.sql - DOM\.....54))' and 'SQLQuery1.sql - DOM\.....52))'. The active tab is 'SQLQuery2.sql', which contains the following T-SQL code:

```
select NAME,type from dbo.sysobjects where type LIKE 'U'
```

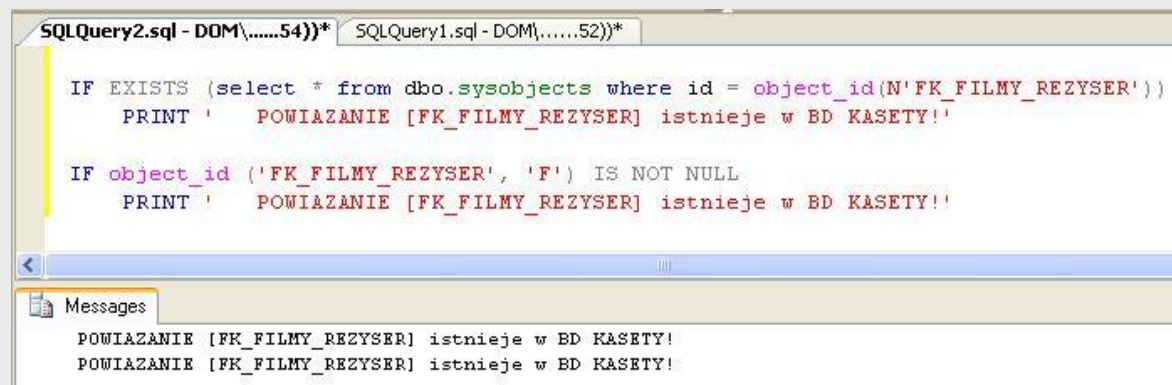
Below the query window, the 'Results' pane displays the output of the query as a table with 10 rows and 2 columns: 'NAME' and 'type'.

	NAME	type
1	KRAJ	U
2	REZYSER	U
3	RODZAJ	U
4	FILMY	U
5	FILKRA	U
6	FILRODZ	U
7	KASETY	U
8	WYPO	U
9	KOMUNIKATYWYZWALACZY	U
10	KLIENCI	U

Uwaga do zajęć poprzednich

Sprawdzanie występowania powiązań tabel w bazie danych

- Oba poniższe zapisy mają tę samą funkcjonalność
 - IF EXISTS (select * from dbo.sysobjects where id = object_id(N'FK_FILMY_REZYSER'))
PRINT ' POWIAZANIE [FK_FILMY_REZYSER] istnieje w BD KASETY!'
 - IF object_id ('FK_FILMY_REZYSER', 'F') IS NOT NULL
PRINT ' POWIAZANIE [FK_FILMY_REZYSER] istnieje w BD KASETY!'

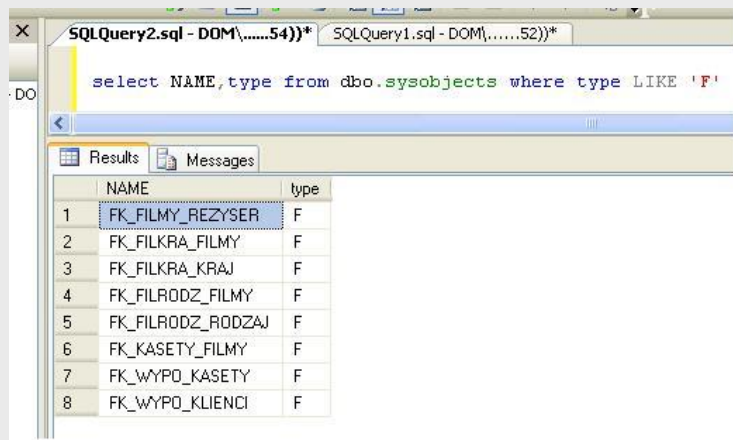


The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'SQLQuery2.sql - DOM\.....54)))*' and 'SQLQuery1.sql - DOM\.....52)))*'. The active tab contains the following SQL code:

```
IF EXISTS (select * from dbo.sysobjects where id = object_id(N'FK_FILMY_REZYSER'))  
    PRINT ' POWIAZANIE [FK_FILMY_REZYSER] istnieje w BD KASETY!'  
  
IF object_id ('FK_FILMY_REZYSER', 'F') IS NOT NULL  
    PRINT ' POWIAZANIE [FK_FILMY_REZYSER] istnieje w BD KASETY!'
```

Below the query window, the 'Messages' pane displays the output of the query:

```
POWIAZANIE [FK_FILMY_REZYSER] istnieje w BD KASETY!  
POWIAZANIE [FK_FILMY_REZYSER] istnieje w BD KASETY!
```



The screenshot shows a SQL Server Enterprise Manager window with two tabs: 'SQLQuery2.sql - DOM\.....54)))*' and 'SQLQuery1.sql - DOM\.....52)))*'. The active tab contains the following SQL code:

```
select NAME,type from dbo.sysobjects where type LIKE 'F'
```

Below the query window, the 'Results' pane displays the output of the query in a table format:

	NAME	type
1	FK_FILMY_REZYSER	F
2	FK_FILKRA_FILMY	F
3	FK_FILKRA_KRAJ	F
4	FK_FILRODZ_FILMY	F
5	FK_FILRODZ_RODZAJ	F
6	FK_KASETY_FILMY	F
7	FK_WYPO_KASETY	F
8	FK_WYPO_KLIENCI	F

Uwaga do zajęć poprzednich

Sprawdzanie występowania Reguł w bazie danych

- Oba poniższe zapisy mają tę samą funkcjonalność
 - IF object_id ('PLEC_R','R') IS NOT NULL
 - IF EXISTS (SELECT name FROM sys.objects WHERE name = 'PLEC_R' AND type = 'R')

Sprawdzanie występowania Wartości domyślnych w bazie danych

- IF OBJECT_ID ('B_D','D') IS NOT NULL

Sprawdzanie występowania Widoków w bazie danych

- IF OBJECT_ID ('V_KLIENCI_WYPO','V') IS NOT NULL

Sprawdzanie występowania Wyzwalaczy w bazie danych

- IF OBJECT_ID ('PO_AKT_RODZAJ','TR') IS NOT NULL – dotyczy Triggerów

Sprawdzanie występowania Funkcji skalarnej w bazie danych

- IF OBJECT_ID ('OSOBY_IM','FN') IS NOT NULL – dotyczy Funkcji Skalarnej

Sprawdzanie występowania Funkcji Tabelarnej w bazie danych

- IF OBJECT_ID ('KASETY1','IF') IS NOT NULL – dotyczy Funkcji Tabelarnej

Procedury wyzwalane (ang.Triggers)

- **Procedury wyzwalane** (typ procedury przechowywanej), są przechowywane na serwerze bazy danych i wykonywane automatycznie w wyniku zaistnienia określonego zdarzenia.
- W bazie danych MS SQL Server 2014 istnieją trzy typy wyzwalaczy:
 - **Wyzwalacze DML** – uruchamiają się gdy użytkownik próbuje modyfikować dane przy pomocy zdarzeń języka DML (Data Manipulating Language). Są to polecenia INSERT, DELETE i UPDATE na tablicy lub widoku.
 - **Wyzwalacze DDL** – uruchamiają się gdy wywoływane są polecenia języka DDL (Data Definition Language), takie jak polecenia CREATE, ALTER, DROP a także niektóre procedury systemowe.
 - **Wyzwalacze Logon** – uruchamiają się podczas zdarzenia LOGON, gdy użytkownik nawiązuje sesję z bazą danych.

Procedury wyzwalane (ang.Triggers)

- **Wyzwalacze DML** mogą być inicjowane zdarzeniami modyfikującymi wiersze **tabeli** lub **widoku** polecenia:
 - ✓ dodawania (INSERT)
 - ✓ modyfikacji (UPDATE)
 - ✓ lub usuwania danych (DELETE)
- W SQL Server 2014 można wyróżnić dwa rodzaje procedur wyzwalanych:
 - ✓ Wyzwalacze „**AFTER**” – wykonywane natychmiast po poleceniach INSERT, UPDATE, DELETE
 - ✓ Wyzwalacze „**INSTEAD OF**” – wykonywane są zamiast poleceń INSERT, UPDATE, DELETE
- Wyzwalacze uruchamiają się niezależnie od tego czy dane polecenia mają wpływ na wiersze w tabeli lub widoku.
- **Dla tabel** można wywoływać wyzwalacze „**po**” i „**zamiast**” danego polecenia (AFTER i INSTEAD OF),
- **Dla widoków** można tylko wywoływać wyzwalacze „**zamiast**” danego polecenia (INSTEAD OF).
- Wyzwalacze skojarzone są z tabelami oraz operacjami na nich wykonywanymi. Zapewniają spójność danych.
- Podczas pisania wyzwalaczy bardzo pomocne są dane przechowywane w tabelach DELETED oraz INSERTED, które przechowują ostatnio dodane lub usunięte wiersze.

Procedury wyzwalane (ang.Triggers)

Składnia tworzenia procedury wyzwalanej

```
CREATE TRIGGER <Nazwa wyzwalacza>  
ON <TABELA>  
<Typ wyzwalacza> <Lista poleceń>  
AS  
<Instrukcja poleceń>
```

Procedury wyzwalane (ang.Triggers)

Przykład

1. Utworzyć tabelę w której będą zapisywane akcje użytkowników bazy na danych (wstaw, zmodyfikuj usuń)

```
CREATE TABLE KOMUNIKATYWYZWALACZY
(
    IDW                int                IDENTITY(1,1),
    TABELA              char(30),
    KOLUMNA             char(15),
    OPERACJA            char(10),
    STARA_WART          char(50),
    NOWA_WART           char(50),
    CZAS                smalldatetime      DEFAULT (GETDATE()),
    UZYTKOWNIK          char(20)  DEFAULT (USER)
)
```

Procedury wyzwalane (ang.Triggers)

Przykład

2a. Utworzyć przykładowy wyzwalacz (trigger)

```
CREATE TRIGGER POAKTUALIZACJI_RODZAJ
```

Nazwa wyzwalacza

```
ON RODZAJ
```

tabela

```
AFTER UPDATE
```

Lista poleceń

```
AS
```

```
INSERT INTO KOMUNIKATYWYZWALACZY (TABELA, KOLUMNA, OPERACJA,  
    STARA_WART, NOWA_WART)
```

```
VALUES ('RODZAJ','dowolna','UPDATE','Stary rodzaj','Nowy rodzaj')
```

```
GO
```

Typ wyzwalacza

Procedury wyzwalane (ang.Triggers)

Przykład

2b. Utworzyć przykładowe trigger

```
CREATE TRIGGER FUNKCJE_UPDATE_KLIENCI
ON KLIENCI
AFTER UPDATE
AS
IF UPDATE(NAZWISKO)
INSERT INTO KOMUNIKATYWYZWALACZY (TABELA, KOLUMNA, OPERACJA,
    STARA_WART, NOWA_WART)
VALUES ('KLIENCI', 'Nazwisko', 'UPDATE', 'Stare nazwisko', 'Nowe nazwisko')
IF UPDATE(IMIE)
INSERT INTO KOMUNIKATYWYZWALACZY (TABELA, KOLUMNA, OPERACJA,
    STARA_WART, NOWA_WART)
VALUES ('KLIENCI', 'imie', 'UPDATE', 'Stare imię', 'Nowe imię')
```

Procedury wyzwalane (ang.Triggers)

Przykład

2c. Utworzyć przykładowe wyzwalacze

```
CREATE TRIGGER PO_AKT_KRAJ
ON KRAJ
AFTER UPDATE
AS
IF UPDATE (KRAJPROD)
  DECLARE @kraj_OLD varchar(15),@kraj_NEW varchar(15)
  SELECT @kraj_OLD=krajprod
  FROM DELETED
  SELECT @kraj_NEW=krajprod
  FROM INSERTED
  INSERT INTO KOMUNIKATYWYZWALACZY (TABELA, KOLUMNA, OPERACJA, STARA_WART, NOWA_WART)
  VALUES ('KRAJ','KRAJPROD','UPDATE' ,@kraj_OLD ,@kraj_NEW)
GO
```

```
CREATE TRIGGER PO_WSTAW_KRAJ
ON KRAJ
AFTER INSERT
AS
IF UPDATE (KRAJPROD)
  DECLARE @kraj_NEW varchar(15)
  SELECT @kraj_NEW=krajprod
  FROM INSERTED
  INSERT INTO KOMUNIKATYWYZWALACZY (TABELA KOLUMNA OPERACJA STARA_WART NOWA_WART)
  VALUES ('KRAJ','KRAJPROD','INSERT' ,NULL ,@kraj_NEW)
GO
```

Procedury wyzwalane (ang.Triggers)

Przykład

2d. Utworzyć przykładowe triggery

```
CREATE TRIGGER PO_USUN_KRAJ
ON KRAJ
AFTER DELETE
AS
IF UPDATE (KRAJPROD)
  DECLARE @kraj_OLD varchar(15)
  SELECT @kraj_OLD=krajprod
  FROM DELETED
  INSERT INTO KOMUNIKATYWYZWALACZY
    (TABELA,KOLUMNA,OPERACJA,STARA_WART,NOWA_WART)
  VALUES ('KRAJ','KRAJPROD','DELETE',@kraj_OLD,NULL)
GO
```

Procedury wyzwalane (ang.Triggers)

Przykład: Wykonanie

IDKRAJ	KRAJPROD
.....

IDW	TABELA	KOLUMNA	OPERACJA	STARA_WART	NOWA_WART	CZAS	UŻYTKOWNIK
1	KRAJ	KRAJPROD	INSERT	NULL	POLSKA	2009-11-26 19:25:00	dbo
2	KRAJ	KRAJPROD	UPDATE	POLSKA	POLSKA1	2009-11-26 19:27:00	dbo
3	KRAJ	KRAJPROD	DELETE	POLSKA1	NULL	2009-11-26 19:29:00	dbo

Procedury wyzwalane (ang.Triggers)

Przykład

Napisać trigger, który w tabeli WYPO w kolumnie KWOTA wyliczy ile klient ma zapłacić za wypożyczenie kasety (DATAZ - DATAW) * CENA - uwaga Cena filmu

```
CREATE TRIGGER tr_update_wypo
ON WYPO
AFTER UPDATE
AS
DECLARE @idklienta int,
        @idkasety int,
        @dataw smalldatetime,
        @dataz_OLD smalldatetime,
        @dataz_NEW smalldatetime,
        @cenak decimal(6,2)

IF UPDATE (dataz)
BEGIN
    SELECT @idklienta=idklienta,
           @idkasety=idkasety,
           @dataw=dataw,
           @dataz=dataz
    FROM DELETED
    SELECT @dataz_NEW=dataz
    FROM INSERTED

    SELECT @cenak=CENA
    FROM KASETY join FILMY on kasety.idfilmu=filmy.idfilmu
    WHERE idkasety=@idkasety

    UPDATE wypo
    SET kwota=DATEDIFF(day,@dataw,@dataz_NEW)*@cenak
    WHERE idklienta=@idklienta and
           idkasety=@idkasety and
           dataw=@dataw
END
GO
```


Funkcje skalarne

Zwracają skalarne –jednowartościowe wyniki np. ciągi znaków lub liczby (nie zwraca tabeli)

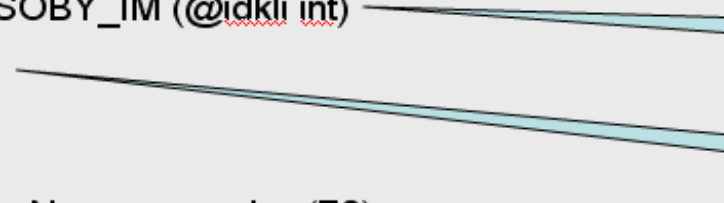
Składnia Tworzenia Funkcji skalarnej

```
CREATE FUNCTION nazwa_funkcji (lista parametrów)  
RETURNS typ_danych  
[ AS ]  
BEGIN  
    instrukcje transact SQL  
    RETURN wartość zwracana  
END
```

Funkcje skalarne - przykład

Napisać funkcję dla której parametrem na wejściu jest **IDklienta**
a na wyjściu otrzymamy: **Nazwisko, imię i adres klienta**

```
CREATE FUNCTION OSOBY_IM (@idkli int)
RETURNS nvarchar(70)
AS
BEGIN
    DECLARE @PelnaNazwa nvarchar(70)
    SELECT @PelnaNazwa = nazwisko + '-' + imię + ' ' + adres
    FROM Klienci
    WHERE idklienta = @idkli
    RETURN (@PelnaNazwa)
END
GO
```



Wywołanie **Uwaga** (należy podać nazwę właściciela np. dbo lub Z502_11)

```
SELECT dbo.OSOBY_IM(1)
GO
Lub SELECT Z502_11.OSOBY_IM(1)
GO
```

Funkcje tabelarne

Zwracają tabelę z danymi

Składnia Tworzenia Funkcji tabelarnej

```
CREATE FUNCTION nazwa_funkcji (lista parametrów)  
    ( [ { @parameter_name [AS] scalar_parameter_data_type [= default] } [ ,...n ] ] )  
RETURNS TABLE  
    [ AS ]  
  
RETURN (instrukcja select)
```

Usuwanie funkcji

```
DROP FUNCTION nazwa
```

Funkcje tabelarne - przykład

Napisać funkcję dla której parametrem na wejściu jest **idfilmu**
a na wyjściu otrzymamy: **tytuł** filmu i **idkasety** (może być ich kilka)
na których ten film się znajduje

```
CREATE FUNCTION KASETY1 (@TYP INT)
RETURNS TABLE
AS
    RETURN (SELECT idkasety,tytul
            FROM Kasety
            join Filmy on Kasety.idfilmu= Filmy.idfilmu
            WHERE Filmy.idfilmu=@TYP)
```



Wywołanie

```
SELECT * FROM dbo.KASETY1(4)
GO
```

Procedury wyzwalane (ang.Triggers)

Przykład: Wykonanie

STACJA LOKALNA 1

ID	STACJA	DATA	TEMPERATURA	CIŚNIENIE	STAN
			11	1001	

insert into DANE (temperatura,cisnienie,data)
values (11,1001,'2016-11-25 10:00')

ID	STACJA	DATA	TEMPERATURA	CIŚNIENIE	STAN
1	Stacja_1	2016-11-25 9:00	10	1000	P
2	Stacja_1	2016-11-25 10:00	11	1001	P

Trigger stacji lokalnej

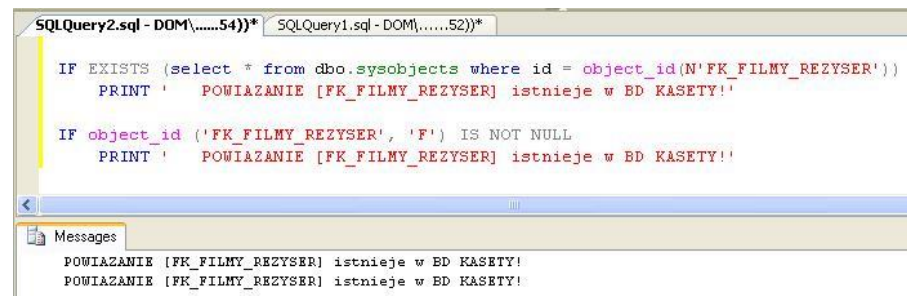
Trigger stacji centralnej

STACJA CENTRALNA

ID	STACJA	DATA	TEMPERATURA	CIŚNIENIE
1	Stacja_1	2016-11-25 9:00	10	1000
2	Stacja_1	2016-11-25 10:00	11	1001

Bazy danych – laboratorium

Dziękuję za uwagę!!!!!!
Teraz ćwiczenie



The screenshot shows a SQL query window with two tabs: 'SQLQuery2.sql - DOM\.....54))*' and 'SQLQuery1.sql - DOM\.....52))*'. The active tab contains the following SQL code:

```
IF EXISTS (select * from dbo.sysobjects where id = object_id(N'FK_FILMY_REZYSER'))  
    PRINT '    POWIAZANIE [FK_FILMY_REZYSER] istnieje w BD KASETY!'  
  
IF object_id ('FK_FILMY_REZYSER', 'F') IS NOT NULL  
    PRINT '    POWIAZANIE [FK_FILMY_REZYSER] istnieje w BD KASETY!'
```

Below the code, a 'Messages' window displays the results of the query:

```
POWIAZANIE [FK_FILMY_REZYSER] istnieje w BD KASETY!  
POWIAZANIE [FK_FILMY_REZYSER] istnieje w BD KASETY!
```

