

# Bazy danych – Semestr 2

## Zajęcia nr 4

Pisanie skryptów  
kontynuacja

# Zakres zajęć

- Pisanie skryptu do tworzenia bazy danych „Wypożyczalnia filmów” - **kontynuacja**
- Modyfikacja utworzonych tabel:
  - Check (ograniczenie, uszczegółowienie typu danej)
  - Default (wartość domyślna)
  - Identity (autonumerowanie)
- Tworzenie Perspektyw (Widoki)
- Tworzenie zmiennej RULE (reguła) i dodawanie jej do kolumn tabeli
- Tworzenie zmiennej DEFAULT (wartość domyślna) i ustawianie jej do kolumn tabeli
- Wprowadzanie/usuwanie wierszy danych do /z bazy
  - Tworzenie skryptu do wprowadzania i usuwania danych z bazy

# Modyfikacja utworzonych tabel

## Tworzenie klucza głównego w tabeli KLIENCI (inny sposób)

```
ALTER TABLE KLIENCI ADD  
  CONSTRAINT [PK_KLIENCI] PRIMARY KEY  
  (IDKLIENTA) ON [PRIMARY]
```

### Przykład

```
CREATE TABLE KLIENCI
```

```
(  
  IDKLIENTA      int          PRIMARY KEY,  
  NAZWISKO      char (30)      NOT NULL,  
  IMIE          char (15)      NOT NULL,  
  WIEK          int          NOT NULL,  
  ADRES         char (30)      NULL,  
  TELEFON       char (15)      NULL,  
  PLEC          char (1)       NOT NULL  
)
```

```
ALTER TABLE KLIENCI ADD  
  CONSTRAINT [PK_KLIENCI] PRIMARY KEY  
  (IDKLIENTA) ON [PRIMARY]
```

# Modyfikacja utworzonych tabel

Definicja kolumny np. **< Zarobki int >**  
pozwała wprowadzać wartości dopuszczalne dla danego typu  
-2<sup>31</sup> (-2,147,483,648) to 2<sup>31</sup>-1 (2,147,483,647)

Za pomocą **CHECK** można ustalić warunek (*uszczegółović typ danych*) jaki mają spełniać wartości wprowadzane do wybranych kolumn.

Warunek ten można ustalić:

- w trakcie tworzenia tabeli
- lub dla tabeli już istniejącej za pomocą **ALTER**

**CHECK - (ograniczenie typu zmiennej)**

Np.        **CHECK (ZAROBKI BETWEEN 100 AND 20000)**  
             **CHECK (KOLOR\_OCZU IN ('Z','N','C'))**

**Przykład: zapis (definicja kolumny tabeli bazy danych)**

Zarobki	int	CHECK (ZAROBKI BETWEEN 100 AND 20000)
---------	-----	---------------------------------------

# Modyfikacja utworzonych tabel

**DEFAULT** umożliwia wprowadzenie wartości domyślnej dla danej kolumny.

W ten sposób można ustalić taką domyślną wartość:

- podczas tworzenia tabeli
- lub po jej utworzeniu za pomocą polecenia ALTER.

**DEFAULT - (wartość domyślna zmiennej**

Np. DEFAULT ('brak danych')

**Przykład zapisu (definicja kolumny tabeli bazy danych)**

Zainteresowania          varchar(30)          DEFAULT('Brak danych')

# Tworzenie reguł

- W SQL SERVER 2014 można tworzyć obiekty - reguły (***RULE***)
- Kiedy reguła jest związana z kolumną (typem danych) komenda ***SP\_BINDRULE*** określa dopuszczalne wartości, które mogą być wstawiane do tej kolumny
- Regułę można stosować do wielu kolumn z różnych tabel
- Zmiana reguły powoduje zmianę dopuszczalnych wartości związanych z nią kolumn

## REGUŁA (ograniczająca typ zmiennej)

- Tworzenie reguł (RULE)

```
CREATE RULE OSOBY_ZAMIESZK AS @X IN ('Miasto','Wieś')
```

- Dowiązanie reguły do pola tabeli

```
EXEC SP_BINDRULE  
OSOBY_ZAMIESZK, 'OSOBY.TYP_ZAMIESZKANIA'
```

- Zdjęcie dowiązania pola z reguły

```
EXEC SP_UNBINDRULE 'OSOBY.TYP_ZAMIESZKANIA'
```

- Usunięcie reguły

```
DROP RULE OSOBY_ZAMIESZK
```

# Tworzenie zmiennych domyślnych

- W SQL SERVER 2014 można tworzyć obiekty – zmienna domyślna (**DEFAULT**)
- Kiedy zmienna domyślna jest związana z kolumną (typem danych) komendą **SP\_BINDEFULT** można do tabeli wprowadzać wiersze danych z argumentem domyślnym dla tej kolumny
- Zmienną domyślną DEFAULT można stosować do wielu kolumn z różnych tabel

## ZMIENNA DOMYŚLNA

- Tworzenie zmiennej domyślnej (DEFAULT)

```
CREATE DEFAULT BRAK_INF AS 'Brak informacji'
```

- Dowiązanie zmiennej domyślnej do pola tabeli

```
EXEC SP_BINDEFULT BRAK_INF, 'OSOBY.STAN_CYWILNY'
```

- Zdjęcie dowiązania pola ze zmiennej domyślnej

```
EXEC SP_UNBINDEFULT 'OSOBY.STAN_CYWILNY'
```

- Usuwanie zmiennej domyślnej

```
DROP DEFAULT BRAK_INF
```

# Przykładowy Skrypt SQL

```
CREATE DATABASE TEST
GO
USE TEST
GO
PRINT 'Tworzenie Tabeli BD TEST : '
PRINT '-----'
-- ***** Tworzenie Tabeli: OSOBY
-- -----
IF EXISTS (select * from dbo.sysobjects where id = object_id(N'OSOBY'))
BEGIN
    PRINT '    Tabela OSOBY istnieje w BD KASETY!'
END
ELSE
BEGIN
    PRINT '    Tworze tabele OSOBY W BD TEST'
    CREATE TABLE OSOBY
    (
        ID                int                PRIMARY KEY,
        NAZWISKO          char (30)         NOT NULL,
        IMIE              char (15)         NOT NULL,
        ADRES             char (30)         NULL DEFAULT ('Brak Danych'),
        STAN_CYWILNY      char (30)         NULL,
        TELEFON_DOM       char (15)         NULL,
        TELEFON_PRACA     char (15)         NULL,
        ZAROBKI           decimal(8,2) CHECK (ZAROBKI BETWEEN 100 AND 20000),
        PLEC_OSOBY        char (1)          NOT NULL,
        KOLOR_OCZU        char(1)          NOT NULL CHECK (KOLOR_OCZU IN ('Z','N','C')) DEFAULT ('Z')
    )
END
GO
PRINT '-----'
PRINT '***** Tworzenie reguły (RULE) PLEC_XYZ'
GO
CREATE RULE PLEC_XYZ AS @X IN ('M','K')
GO
-- -----
PRINT '***** Dowiązanie kolumny KLIENCI.PLEC do reguły PLEC_XYZ'
EXEC SP_BINDRULE PLEC_XYZ,'OSOBY.PLEC_OSOBY'
GO
PRINT '-----'
PRINT '***** Tworzenie zmiennej domyślnej (DEFAULT) o nazwie BRAK_INF'
GO
CREATE DEFAULT BRAK_INF AS 'Brak informacji'
GO
-- -----
PRINT '***** Dowiązanie kolumny OSOBY.STAN_CYWILNY zmiennej domyślnej BRAK_INF'
EXEC SP_BINDEFULT BRAK_INF,'OSOBY.STAN_CYWILNY'
GO
PRINT '-----'
/*
EXEC SP_UNBINDEFULT 'OSOBY.STAN_CYWILNY'
DROP DEFAULT BRAK_INF
EXEC SP_UNBINDRULE 'OSOBY.PLEC_OSOBY'
DROP RULE PLEC_XYZ
USE KASETY_01
GO
DROP DATABASE TEST
*/
```



# Wstawianie danych (1)

```
INSERT INTO OSOBY VALUES (1,'KOWAL','JAN',NULL,NULL,NULL,NULL,200,'M','N')
```

```
INSERT INTO OSOBY VALUES  
(2,'ROBOT','ANNA',DEFAULT,DEFAULT,'(22)123456',NULL,1000,'K',DEFAULT)
```

```
INSERT INTO OSOBY  
(ID,NAZWISKO,IMIE,ADRES,STAN_CYWILNY,TELEFON_DOM,TELEFON_PRACA,ZAROBKI,  
PLEC_OSOBY,KOLOR_OCZU)  
VALUES (3,'KOT','ROMAN',' ',' ',200,'M','C')
```

**IDENTITY - (automatyczne numerowanie)** Np. IDENTITY (1,1)

```
CREATE TABLE OSOBY_1
```

```
(  
  ID                int                PRIMARY KEY IDENTITY(1,1),  
  NAZWISKO          char (30)          NOT NULL,  
  IMIE              char (15)          NOT NULL,  
  ADRES             char (30)          NULL      DEFAULT ('Brak Danych'),  
  STAN_CYWILNY      char (30)          NULL,  
  TELEFON_DOM       char (15)          NULL,  
  TELEFON_PRACA     char (15)          NULL,  
  ZAROBKI            decimal(8,2)      CHECK (ZAROBKI BETWEEN 100 AND 20000),  
  PLEC_OSOBY        char (1)           NOT NULL,  
  KOLOR_OCZU        char(1)           NOT NULL CHECK (KOLOR_OCZU IN ('Z','N','C')) DEFAULT ('Z')  
)
```

## WSTAWIANIE DANYCH

```
INSERT INTO OSOBY_1 VALUES ('KOWAL','JAN',NULL,NULL,NULL,NULL,200,'M','C')
```

```
INSERT INTO OSOBY_1 VALUES ('ROBOT','ANNA',DEFAULT, NULL,'(22)123456',NULL,1000,'K',DEFAULT)
```

```
SELECT * FROM OSOBY_1
```

## KASOWANIE DANYCH:

```
DELETE FROM OSOBY_1
```

```
TRUNCATE TABLE OSOBY_1
```

# Wstawianie i usuwanie danych (2)

**Wstawianie danych (wierszy) do tabel bazy danych „Wypożyczalnia kaset Video”**

**INSERT INTO** Kraj (idkraj,kraiprod) values (1,'POLSKA')

SELECT '1 - Wpisałem KRAJ' as operacja,COUNT(\*) FROM Kraj

**INSERT INTO** Rodzaj (idrodzaj,rodzajfil) values (1, 'KOMEDIA')

**INSERT INTO** Klienci (idklienta,nazwisko,imie,adres,telefon,plec)

values (1, 'KOWALSKI','JAN','KOCHANOWSKIEGO 21','1234567','M')

**INSERT INTO** Rezyser (idrezyser,nazwisko,imie) values (1, 'WAJDA','ANDRZEJ')

**INSERT INTO** Filmy (idfilmu,tytul,idrezyser,cena,kolor) values (1,'Kanał',1,9,'C')

**INSERT INTO** Kasety (idkasety,idfilmu,status) values (101,1,'W')

**INSERT INTO** Wypo (idklienta,idkasety,dataw,dataz) values (1,101,'1997-12-22',NULL)

**Usuwanie danych (wierszy) z tabel bazy danych „Wypożyczalnia kaset Video”**

DELETE FROM Filkra

GO

SELECT '1 - Usunąłem Filkra' as operacja

DELETE FROM Filrodz

DELETE FROM Wypo

DELETE FROM Klienci

DELETE FROM Kasety

DELETE FROM Filmy

DELETE FROM Rezyser

DELETE FROM Kraj

DELETE FROM Rodzaj

SELECT 'KONIEC usuwania danych z bazy danych'

# Wstawianie i usuwanie danych (3)

## Wstawianie losowych danych (wierszy) do tabel bazy danych

```
CREATE TABLE KRAJ
```

```
(  
  NAZWA_KRAJU varchar(30) PRIMARY KEY,  
)
```

```
GO
```

```
INSERT INTO KRAJ VALUES('POLSKA');
```

```
INSERT INTO KRAJ VALUES('ROSJA');
```

```
INSERT INTO KRAJ VALUES('USA');
```

```
INSERT INTO KRAJ VALUES('WIELKA BRYTANIA');
```

```
INSERT INTO KRAJ VALUES('NIEMCY');
```

```
INSERT INTO KRAJ VALUES('UKRAINA');
```

```
GO
```

```
CREATE TABLE OSOBA NAZWISKO LOSOWO KRAJ
```

```
(ID INT IDENTITY PRIMARY KEY,
```

```
  NAZWISKO varchar(30),
```

```
  KRAJ varchar(30)
```

```
)
```

```
GO
```

```
INSERT into OSOBA NAZWISKO LOSOWO KRAJ values ('KOT',(SELECT TOP 1 * FROM KRAJ ORDER BY NEWID()))
```

```
INSERT into OSOBA NAZWISKO LOSOWO KRAJ values ('LIS',(SELECT TOP 1 * FROM KRAJ ORDER BY NEWID()))
```

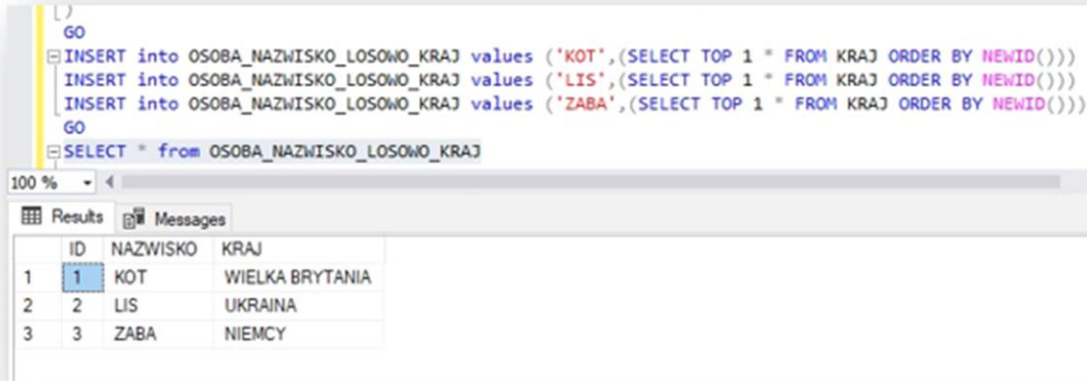
```
INSERT into OSOBA NAZWISKO LOSOWO KRAJ values ('ZABA',(SELECT TOP 1 * FROM KRAJ ORDER BY NEWID()))
```

```
GO
```

```
SELECT * from OSOBA NAZWISKO LOSOWO KRAJ
```

```
DELETE FROM OSOBA NAZWISKO LOSOWO KRAJ
```

```
TRUNCATE TABLE OSOBA NAZWISKO LOSOWO KRAJ
```



# Wstawianie i usuwanie danych (4)

## Wstawianie unikatowych rekordów danych do tabel bazy danych

```
DROP TABLE NOWA
GO
DROP TABLE NOWA1
GO
DECLARE @T1 TABLE (imie varchar(20))
DECLARE @T2 TABLE (nazwisko varchar(20))
DECLARE @T3 TABLE (miasto varchar(20))
insert INTO @T1 VALUES ('Jan'),('Anna'),('Kamil')
SELECT * FROM @T1
INSERT INTO @T2 VALUES ('Kot'),('Lis'),('Ptak')
SELECT * FROM @T2
INSERT INTO @T3 VALUES ('Warszawa'),('Kraków')
SELECT * FROM @T3
SELECT * into NOWA FROM @T1 CROSS JOIN @T2 CROSS JOIN @T3
ALTER table NOWA ADD ID INT IDENTITY(1,1)
GO
SELECT ID,imie,nazwisko,miasto into NOWA1 from NOWA
SELECT * FROM NOWA1
```

```
DROP TABLE NOWA
GO
DROP TABLE NOWA1
GO
DECLARE @T1 TABLE (imie varchar(20))
DECLARE @T2 TABLE (nazwisko varchar(20))
DECLARE @T3 TABLE (miasto varchar(20))
insert INTO @T1 VALUES ('Jan'),('Anna'),('Kamil')
SELECT * FROM @T1
INSERT INTO @T2 VALUES ('Kot'),('Lis'),('Ptak')
SELECT * FROM @T2
```

100 %

Results		Messages	
imie			
1	Jan		
2	Anna		
3	Kamil		
nazwisko			
1	Kot		
2	Lis		
3	Ptak		
miasto			
1	Warszawa		
2	Kraków		

	ID	imie	nazwisko	miasto
1	1	Jan	Kot	Warszawa
2	2	Jan	Kot	Kraków
3	3	Anna	Kot	Warszawa
4	4	Anna	Kot	Kraków
5	5	Kamil	Kot	Warszawa
6	6	Kamil	Kot	Kraków
7	7	Jan	Lis	Warszawa
8	8	Jan	Lis	Kraków
9	9	Anna	Lis	Warszawa
10	10	Anna	Lis	Kraków
11	11	Kamil	Lis	Warszawa
12	12	Kamil	Lis	Kraków
13	13	Jan	Ptak	Warszawa
14	14	Jan	Ptak	Kraków
15	15	Anna	Ptak	Warszawa
16	16	Anna	Ptak	Kraków
17	17	Kamil	Ptak	Warszawa
18	18	Kamil	Ptak	Kraków

# Tworzenie UNIQUE (unikatowy klucz dla wielu kolumn)

## Tworzenie UNIQUE (unikatowy klucz dla wielu kolumn)

Np.

```
CREATE TABLE COS
```

```
(
```

```
    ID1    INT
```

```
    NOT NULL,
```

```
    ID2    INT
```

```
    NOT NULL
```

```
    UNIQUE (ID1,ID2)
```

```
)
```

```
GO
```

```
INSERT INTO COS values (1,1)
```

```
INSERT INTO COS values (1,1)
```

Próba ponownego  
wprowadzenie tej samej  
kombinacji (w tym wypadku  
dwuwymiarowych) danych

**Nie wykona się !!!!!**

```
CREATE TABLE COS
(
    ID1 INT NOT NULL,
    ID2 INT NOT NULL,
    UNIQUE (ID1,ID2)
)
GO

INSERT INTO COS values (1,1)
INSERT INTO COS values (1,1)
```

(1 row affected)

Completion time: 2020-11-13T13:10:18.9728217+01:00

```
CREATE TABLE COS
(
    ID1 INT NOT NULL,
    ID2 INT NOT NULL,
    UNIQUE (ID1,ID2)
)
GO

INSERT INTO COS values (1,1)
INSERT INTO COS values (1,1)
```

Msg 2627, Level 14, State 1, Line 42  
Violation of UNIQUE KEY constraint 'UQ\_\_COS\_\_18DE73E12AAC4400'. Cannot insert duplicate key in object 'dbo.COS'. The duplicate key value is (1, 1).  
The statement has been terminated.

Completion time: 2020-11-13T13:11:24.7831556+01:00



# Tworzenie widoków

Widoki służą do optymalizacji zapytań na bardzo dużej ilości danych

## Przykład 1

```
CREATE VIEW WIDOK_KRAJE_ILOSC
```

```
AS
```

```
SELECT TOP 100 PERCENT KRAJ.KRAJPROD, COUNT(KRAJ.KRAJPROD) AS ILE
```

```
FROM FILMY INNER JOIN
```

```
FILKRA ON FILMY.IDFILMU = FILKRA.IDFILMU INNER JOIN
```

```
KRAJ ON FILKRA.IDKRAJ = KRAJ.IDKRAJ
```

```
GROUP BY KRAJ.KRAJPROD
```

```
ORDER BY KRAJ.KRAJPROD
```

```
GO
```

## Wywołanie komendy SELECT z wykorzystaniem widoku

```
SELECT *
```

```
FROM WIDOK_KRAJE_ILOSC
```

**Uwaga:**

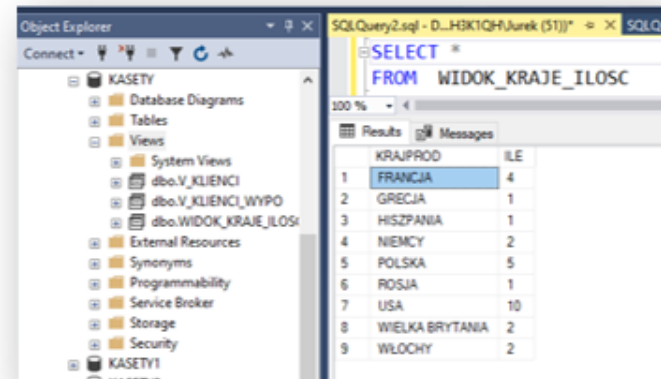
**Przy tworzeniu widoków nie należy używać klauzuli ORDER BY !**

**Jeżeli potrzebujemy posortowane dane, należy operację tę wykonać posługując się już stworzonym widokiem.**

```
SELECT *
```

```
FROM WIDOK_KRAJE_ILOSC
```

```
ORDER BY KRAJPROD DESC
```



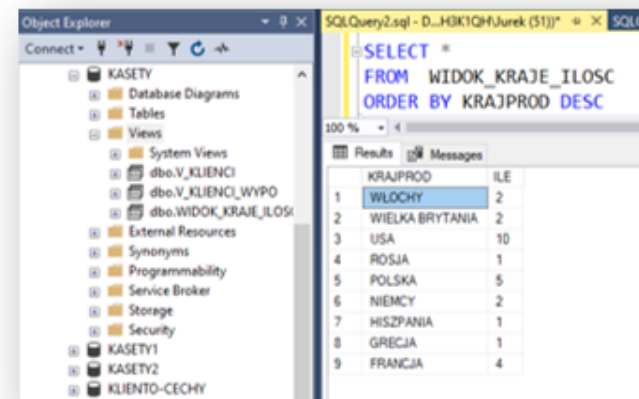
Object Explorer

Connect

SQLQuery2.sql - D:\HBK1QH\Jurek (STJ)\* - SQL

```
SELECT *  
FROM WIDOK_KRAJE_ILOSC
```

	KRAJPROD	ILE
1	FRANCJA	4
2	GRECJA	1
3	HISZPANIA	1
4	NIEMCY	2
5	POLSKA	5
6	ROSJA	1
7	USA	10
8	WIELKA BRYTANIA	2
9	WŁOCHY	2



Object Explorer

Connect

SQLQuery2.sql - D:\HBK1QH\Jurek (STJ)\* - SQL

```
SELECT *  
FROM WIDOK_KRAJE_ILOSC  
ORDER BY KRAJPROD DESC
```

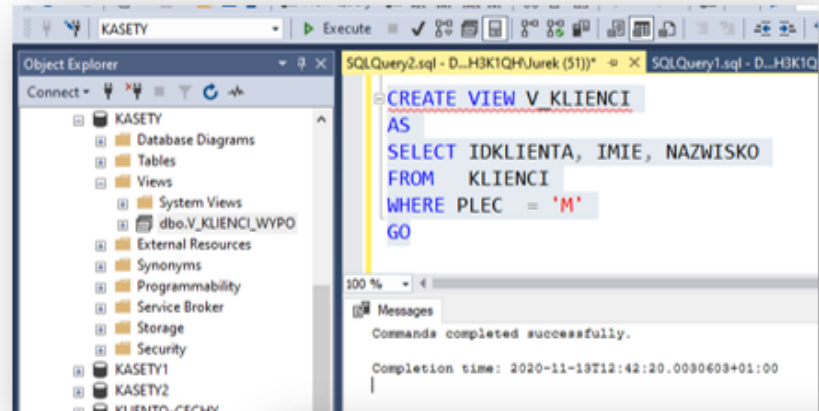
	KRAJPROD	ILE
1	WŁOCHY	2
2	WIELKA BRYTANIA	2
3	USA	10
4	ROSJA	1
5	POLSKA	5
6	NIEMCY	2
7	HISZPANIA	1
8	GRECJA	1
9	FRANCJA	4

# Tworzenie widoków

Widoki służą do optymalizacji zapytań na bardzo dużej ilości danych

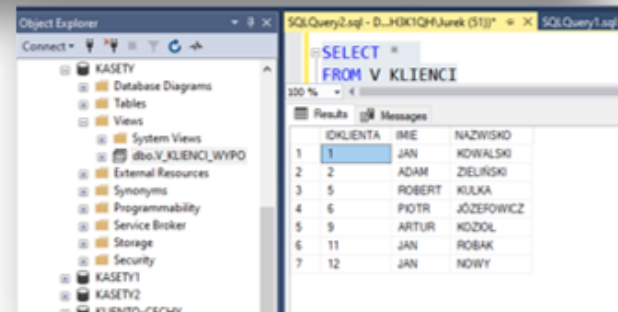
## Przykład 2

```
CREATE VIEW V_KLIENCI
AS
SELECT IDKLIENTA, IMIE, NAZWISKO
FROM KLIENCI
WHERE PLEC = 'M'
GO
```



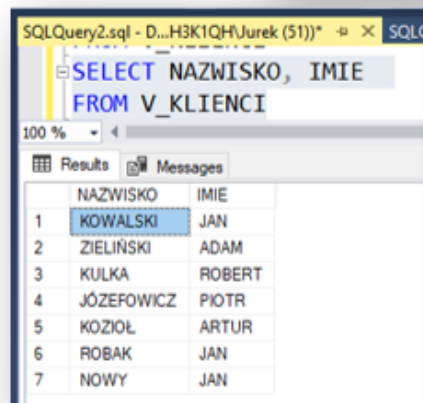
## Wywołanie komendy SELECT z wykorzystaniem widoku

```
SELECT *
FROM V_KLIENCI
```



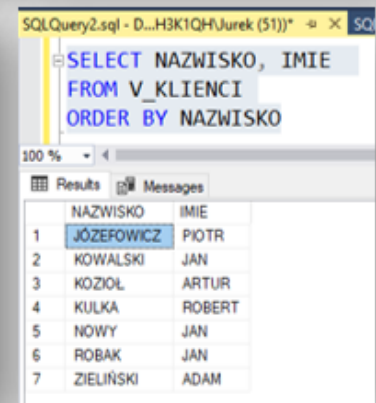
## Projekcja (wybrane kolumny z widoku)

```
SELECT NAZWISKO, IMIE
FROM V_KLIENCI
```



## Projekcja (wybrane kolumny z widoku z sortowaniem wyniku)

```
SELECT NAZWISKO, IMIE
FROM V_KLIENCI
ORDER BY NAZWISKO
```



# Funkcje (1)

Funkcje DATY	Parametry	Zwracana wartość
DATEADD	składnik daty, liczba, data	Nowa data powstała przez dodanie określonej liczby (liczba) do wartości składnik daty w parametrze data
GETDATE		Bieżąca data i godzina systemowa
MONTH	DATA	Liczba całkowita – numer miesiąca
YEAR	DATA	Rok
DAY	DATA	Dzień

## Przykłady

```
SELECT year(getdate())
```

```
GO
```

```
SELECT day(getdate())
```

```
GO
```

```
SELECT month(getdate())
```

```
GO
```

```
SELECT dateadd(day,2,getdate())
```

```
GO
```

```
SELECT dateadd(month,2,getdate())
```

```
GO
```

```
declare @log_bledow varchar(70)
```

```
select @log_bledow = CONCAT('Godzina: ',current_timestamp, ' błąd krytyczny')
```

```
select @log_bledow
```

```
select (CONCAT('Godzina: ',current_timestamp))
```

```
select current_timestamp
```

```
SELECT year(getdate())
GO
SELECT day(getdate())
GO
SELECT month(getdate())
GO
SELECT dateadd(day,2,getdate())
GO
SELECT dateadd(month,2,getdate())
GO
```

```
declare @log_bledow varchar(70)
select @log_bledow = CONCAT('Godzina: ',current_timestamp, ' błąd krytyczny')
select @log_bledow

select (CONCAT('Godzina: ',current_timestamp))
select current_timestamp
```

Results Messages

(No column name)

Godzina: Nov 13 2020 12:02PM błąd krytyczny

(No column name)

Godzina: Nov 13 2020 12:02PM

(No column name)

2020-11-13 12:02:10.490

Results Messages

(No column name)

2020

(No column name)

13

(No column name)

11

(No column name)

2020-11-15 11:52:12.197

(No column name)

2021-01-13 11:52:12.240



# Funkcje (2)

Funkcje Agregujące	Parametry	Zwracana wartość
AVG	Kolumna tabeli	Średnia wartość
COUNT	Kolumna tabeli	Liczba (ilość)
MAX	Kolumna tabeli	Maksymalna wartość
MIN	Kolumna tabeli	Minimalna wartość
SUM	Kolumna tabeli	Suma wartości

# Funkcje (3)

Funkcje Operujące na metodach	Parametry	Zwracana wartość
<b>DB_NAME</b>		Nazwa bazy
<b>DB_ID</b>		ID bazy

Funkcje zabezpieczające	Parametry	Zwracana wartość
<b>HAS_DBACCESS</b>	Nazwa bazy danych	Czy użytkownik ma dostęp do bazy danych o podanej nazwie?
<b>IS_MEMBER</b>	<u>Grupa rola</u>	Określa czy bieżący użytkownik jest członkiem grupy lub ma przypisaną rolę
<b>USER_ID</b>	użytkownik	Numer identyfikacyjny
<b>USER</b>		Nazwa bieżącego użytkownika

## Przykłady

```
SELECT user,
       user_id(),
       getdate(),
       has_dbaccess('master') as [dostęp do bazy]
```

Query: `SELECT user, user_id(), getdate(), has_dbaccess('master') as [dostęp do bazy] GO`

	(No column name)	(No column name)	(No column name)	dostęp do bazy
1	dbo	1	2020-11-13 12:10:23.927	1

```
SELECT db_name(),
       db_id(),
       SCHEMA_NAME(),
       IS_ROLEMEMBER('guest'),
       IS_SRVROLEMEMBER('sysadmin')
```

Query: `select db_name(), db_id(), SCHEMA_NAME(), IS_ROLEMEMBER('guest'), IS_SRVROLEMEMBER('sysadmin')`

	(No column name)	(No column name)	(No column name)	(No column name)	(No column name)
1	master	1	dbo	0	1

```
select db_name(),
       db_id(),
       SCHEMA_NAME(),
       IS_ROLEMEMBER('db_owner'),
       IS_SRVROLEMEMBER('sysadmin')
```

Query: `select db_name(), db_id(), SCHEMA_NAME(), IS_ROLEMEMBER('db_owner'), IS_SRVROLEMEMBER('sysadmin')`

	(No column name)	(No column name)	(No column name)	(No column name)	(No column name)
1	master	1	dbo	1	1

# Funkcje (4)

Funkcje Operujące na metodach	Parametry	Zwracana wartość
COL_LENGTH		Długość kolumny
COL_NAME		Nazwa kolumny

**Zmień znaki na „duże” lub „małe”**

SELECT upper(nazwisko) from klienci

SELECT lower(nazwisko) from klienci

	(No column name)	(No column name)
1	KOWALSKI	kowalski
2	ZIELINSKI	zielinski
3	WOJEWODZIŃSKA	wojewodzinska
4	ZIMA	zima
5	KULKA	kulka
6	JÓZEFOWICZ	jozefowicz
7	PAJACZKOWSKA	pajaczowska
8	BOGUCA	bogucka
9	KOZIOŁ	kozioł
10	KALINOWSKA	kalinowska
11	ROBĄK	robak
12	NOWY	nowy

**Długość kolumn**

SELECT col\_length('klienci', 'nazwisko')

SELECT col\_length('kraj', 'krajprod')

	(No column name)
1	30

	(No column name)
1	15

**Zwraca nazwę kolumny**

SELECT COL\_NAME(OBJECT\_ID('kraj'), 2)

	(No column name)
1	KRAJPROD