Temat: Ewidencja płyt z filmami,

Kolber Maciej, grupa 3a,

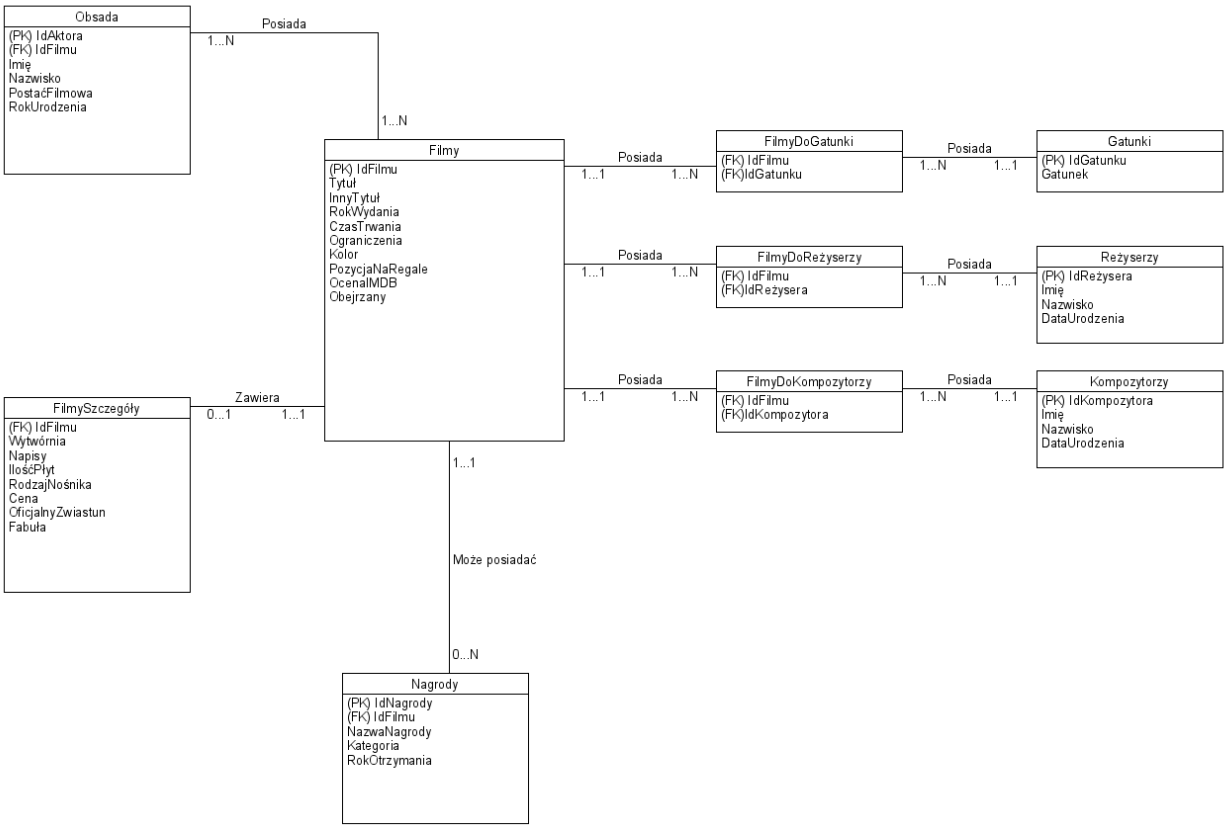
1. Cel, wymagania

Celem systemu bazodanowego jest przechowanie informacji o posiadanych i magazynowanych filmach, tak aby klient miał wgląd do informacji na temat interesującej go pozycji.

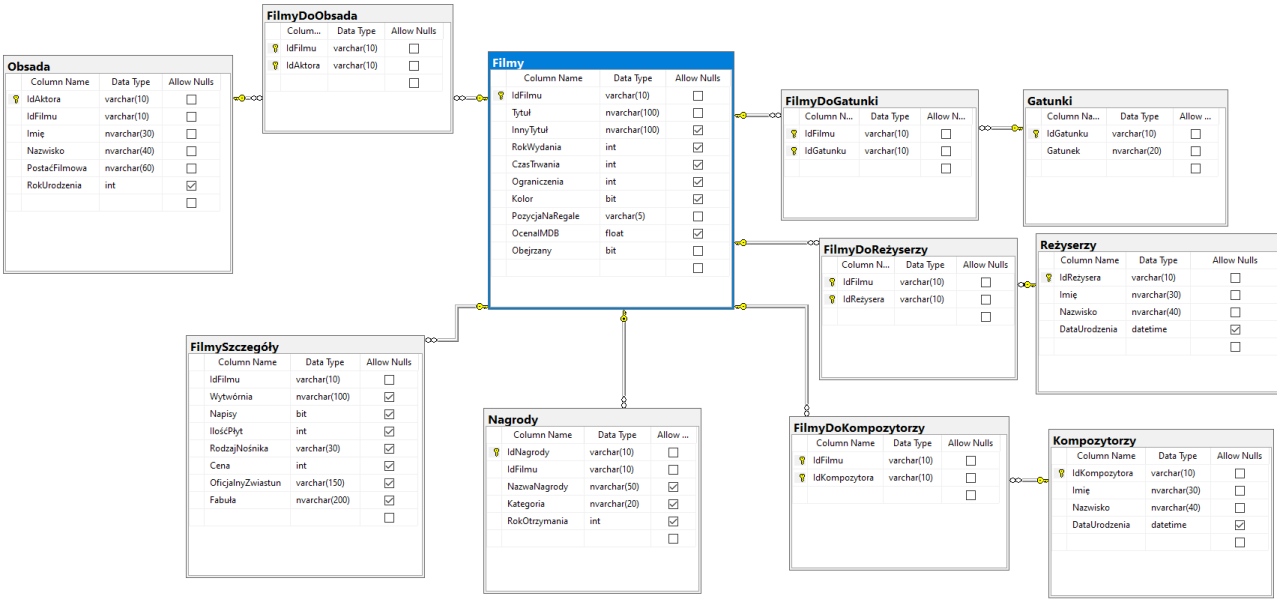
Lp.	Treść	Waga (1-10)	Źródło	Miara	Uwagi
1	Tytuł, Tytuł alternatywny(np. Spolszczony), Rok Wydania, Czas Trwania, Ograniczenia wiekowe Kolor,	10	Biblioteka IMDB		https:// www.imdb.com/
2	Pozycja na Regale, czy został już obejrzany	10	Klient		
3	Wytwórnia, Rodzaj nośnika, ilość płyt, napisy, cena, Oficjalny zwiastun, opis fabuły, Nazwa nagrody, za co otrzymał nagrodę, rok otrzymania nagrody	7	Biblioteka IMDB		
4	Imię, nazwisko aktora, Rok urodzenia aktora, Postać filmowa	8	Biblioteka IMDB		
5	Imię, nazwisko reżysera, Rok urodzenia reżysera	9	Biblioteka IMDB		
6	Imię, nazwisko kompozytora, rok urodzenia	6	Biblioteka IMDB		
7	Sprawdzenie czy klient posiada daną pozycję	10	Klient		
8	Posortowanie posiadanych filmów według najstarszego	8	Klient		
9	Posortowanie posiadanych filmów według najnowszego	8	Klient		

10	Podgląd obejrzanych filmów	10	Klient		
11	Podgląd nieobejrzanych filmów	10	Klient		
12	Ranking filmów według OcenyIMDB	9	Klient		
13	Podgląd filmów kolorowych	7	Klient		
14	Podgląd filmów czarno-białych	3	Klient		
15	Podgląd filmów dla osób poniżej 16 roku życia.	5	Klient		
16	Posortowanie według najkrótszego filmu	9	Klient		
17	Posortowanie według najdłuższego filmu	6	Klient		
18	Wyszukanie filmów w których gra interesujący nas aktor.	10	Klient		
19	Ranking filmów według ilości przyznanych mu nagród	9	Klient		
20	Podgląd filmów z danego gatunku	10	Klient		
21	Podgląd filmów według danego reżysera	10	Klient		
22	Podgląd filmów według danego kompozytora	7	Klient		
23	Możliwość dodania filmu	10	Klient		
24	Możliwość usunięcia filmu	10	Klient		
25	Możliwość zmiany pozycji na regale	10	Klient		
26	Możliwość zmiany atrybutu „Obejrzany”	10	Klient		
27	Możliwość dodania aktora/kompozytora/reżysera/nagrody	10	Klient		

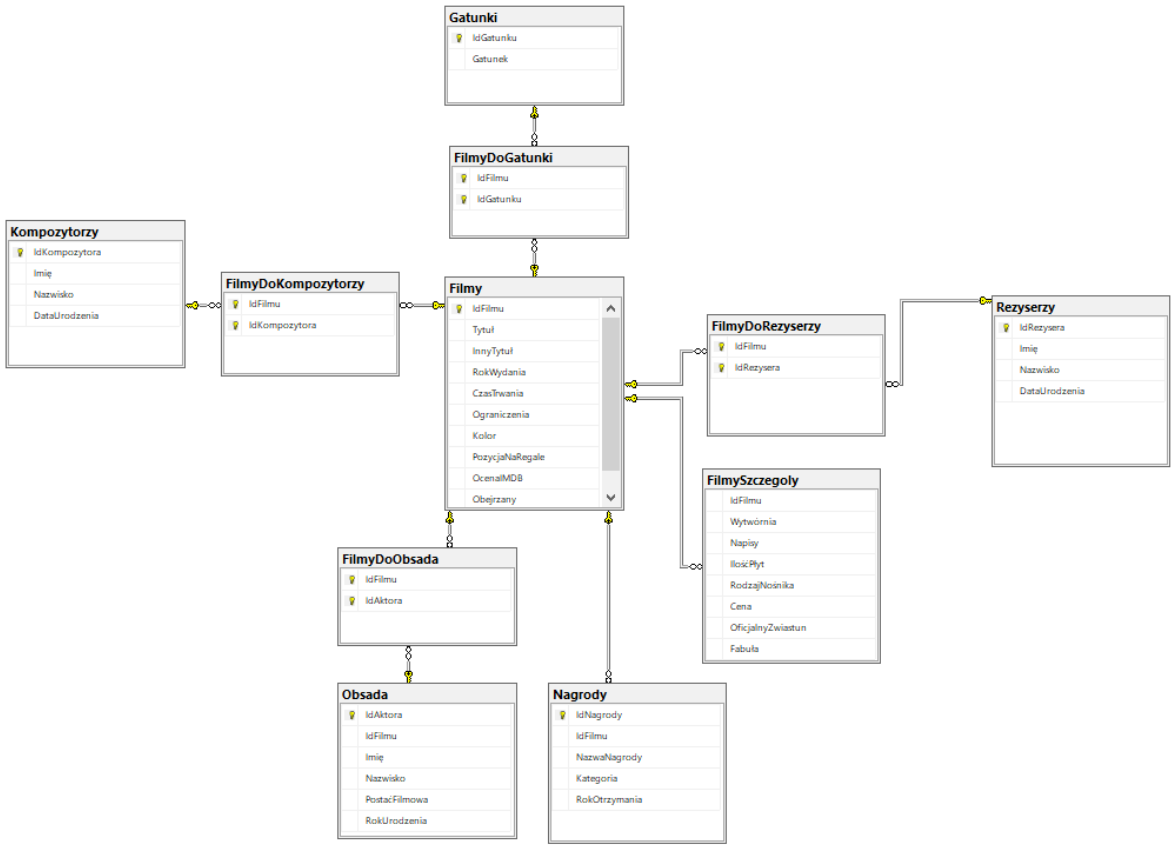
2. Definicja DZE



3. Transformacja DZE do modelu relacyjnego



4. Normalizacja



5. Definicja zasad poprawności danych

Wartości kolumny zaczynającej się od ‘Id’ muszą zawierać wartości unikatowe dla każdej tabeli w której jest kluczem głównym.

Tabela **Filmy**
- Kolumna **Tytuł** jest wymagana i zawiera łańcuch znaków.
- Kolumna **RokWydania** zawiera wartość INT i nie może być ujemna. Nie może być większa niż obecny rok.
- Kolumna **CzasTrwania** zawiera wartość INT która zawiera czas trwania określony w minutach. Nie może być ujemna.
- Kolumna **Ograniczenia** określa minimalny wiek odbiorcy, nie może być ujemna. Jest to INT
- Kolumna **Kolor** określa czy film jest kolorowy czy czarno-biały. Dla wartości 1 film jest kolorowy, dla 0 – czarno biały.
- Kolumna **PozycjaNaRegale** Określa położenie filmu na odpowiednio przygotowanych i ponumerowanych półkach. Jest to łańcuch znaków. Jest Wymagana.
- Kolumna **OcenaIMDB** jest typem float i przyjmuje wartości z przedziału 0.0 – 10.0. Wszelkie inne wartości są błędem.
- Kolumna Obejrzany przyjmuje wartość prawda/fałsz (1/0) i jest wymagana.

Tabela **FilmySzczegóły**
- Kolumna **IlośćPłyt** przyjmuje wartość INT. Nie może być ujemna.
- Kolumna **Cena** podobnie jak kolumna **IlośćPły**t przyjmuje wartość INT i nie może być ujemna.
- Kolumna **OficjalnyZwiastun** przyjmuje łańcuch znaków który będzie linkiem do zewnętrznego serwisu udostępniającego zwiastun.

Tabela **Nagrody**
- Kolumna **RokOtrzymania** jest typem INT, wartość nie może być ujemna ani większa niż obecny rok.

Tabela Obsada
- Kolumna **RokUrodzenia** jest typem INT. Nie może być większa niż obecny rok oraz nie może być ujemna.

Tabele **Kompozytorzy** i **Reżyserzy**
- Kolumna **DataUrodzenia** przyjmuje typ DATE. Nie może być on dalszy niż data dnia dzisiejszego.

6. Definicja schematu bazy danych, utworzenie bazy danych

Kod tworzący bazę danych będzie dołączony w pliku o nazwie „Generowanie BD.sql” Zawiera on tabele oraz relację.

Dostępny również na Github:
https://github.com/Konserwa203/Bazy-Danych → „Generowanie BD.sql”

7+8 Definicja i implementacja niedeklaratywnych mechanizmów sprawdzania poprawności danych

Wyzwalacze:
1) Nie pozwala na wprowadzenie ujemnej ceny filmu.

```
ALTER TRIGGER Cena_Powyzej_0 ON FilmySzczegoly
For Insert AS BEGIN
Declare @cena AS MONEY
SET @CENA = ( SELECT CENA FROM INSERTED )
       IF(@cena < 0)
              BEGIN
              RAISERROR('Cena nie może byc ujemna!',14,1)
       END
END
```

2) Nie pozwala na wprowadzenie ujemnej ilości płyt.

```
ALTER TRIGGER Plyty_Ilość_Nieujemna ON FilmySzczegoly
FOR INSERT AS BEGIN
 DECLARE @plyty AS INTEGER
 SET @plyty = ( SELECT IlośćPłyt FROM INSERTED )
     IF(@plyty < 0)
         BEGIN
         RAISERROR('Nie można mieć ujemnej ilości płyt!',14,1)
     END
 END
```

3) Nie pozwala na wprowadzenie daty urodzin reżysera na dzień dalszy niż dzień dzisiejszy.

```
Create Trigger Data_Nie_Pozniej_Niz_Dzisiaj on Rezyserzy
FOR INSERT AS BEGIN
 DECLARE @DATA AS DATE
 SET @DATA = (SELECT DataUrodzenia FROM inserted)
     if(@DATA > GETDATE())
     BEGIN
     RAISERROR('Nie można ustawić daty urodzin na dzień dalszy niż dzisiaj',14,1)
     END
 END
```

4) Nie pozwala na zmianę daty urodzin reżysera na dzień dalszy niż dzisiejszy.

```
ALTER TRIGGER Data_Nie_Pozniej_Niz_Dzisiaj_Update on Rezyserzy
Instead of update AS BEGIN
DECLARE @DATA AS DATE
SET @DATA = (SELECT DataUrodzenia FROM inserted)
    if(@DATA >= GETDATE())
    BEGIN
    RAISERROR('Nie można ustawić daty urodzin na dzień dalszy niż dzisiaj',14,1)
    END

    if(@DATA < GETDATE())
    Begin
    Update Rezyserzy set DataUrodzenia = @DATA
    END
END
```

9. Implementacja kodu wspomagającego aplikację użytkową

a) Procedura sprawdzająca czy klient posiada dany tytuł, jeżeli tak to wyświetla podstawowe informacje o konkretnym filmie.

```
ALTER PROCEDURE [dbo].[procedure_Posiadany] @Tytuł nvarchar(100)
AS
SELECT * FROM Filmy
where @Tytuł = Tytuł
```

b) Procedura wyświetlająca filmy w których gra dany aktor.

```
ALTER PROCEDURE [dbo].[procedure_Aktor] @Imię nvarchar(30), @Nazwisko nvarchar(40)
AS
SELECT Tytuł, InnyTytuł, OcenaImdb, Kolor, Obejrzany,RokWydania, PozycjaNaRegale FROM
Filmy
Inner Join FilmyDoObsada on Filmy.IdFilmu = FilmyDoObsada.IdFilmu
Inner Join Obsada on FilmyDoObsada.IdAktora = Obsada.IdAktora

Where ((Imię = @Imię) AND (Nazwisko = @Nazwisko))
```

c) Procedura zawierająca najdłuższe filmy posortowane według czasu trwania.

```
CREATE proc proc_SortujNajdluzsze
AS
SELECT  Tytuł, InnyTytuł, OcenaImdb, CzasTrwania,Kolor, Obejrzany,RokWydania, PozycjaNaRegale
FROM Filmy
ORDER BY CzasTrwania desc
```

d) Procedura zawierająca najkrótsze filmy posortowane według czasu trwania.

```
CREATE proc proc_SortujNajkrotsze
AS
SELECT  Tytuł, InnyTytuł, OcenaImdb, CzasTrwania,Kolor, Obejrzany,RokWydania, PozycjaNaRegale
FROM Filmy
ORDER BY CzasTrwania asc
```

e) Procedura zawierająca ranking filmów według oceny IMDB

```
CREATE proc proc_Ranking_Filmow
AS
SELECT Tytuł, InnyTytuł, OcenaImdb, Kolor, Obejrzany,RokWydania, PozycjaNaRegale
FROM Filmy
ORDER BY OcenaIMDB desc
```

f) Procedura zawierająca najnowsze filmy.

```
CREATE proc proc_Najnowsze_Filmy
AS
SELECT Tytuł, InnyTytuł, OcenaImdb, Kolor, Obejrzany,RokWydania, PozycjaNaRegale
FROM Filmy
ORDER BY RokWydania desc
```

g) Procedura zawierająca najstarsze filmy.

```
CREATE proc proc_Najstarsze_Filmy
AS
SELECT Tytuł, InnyTytuł, OcenaImdb, Kolor, Obejrzany,RokWydania, PozycjaNaRegale
FROM Filmy
ORDER BY RokWydania asc
```

h) Procedura umożliwiająca podgląd filmów z konkretnego gatunku

```
ALTER PROC proc_Gatunek @Gatunek nvarchar(20)
AS
SELECT Filmy.IdFilmu, Tytuł, InnyTytuł, CzasTrwania, RokWydania, OcenaIMDB,Obejrzany, PozycjaNaRegale, Gatunek
FROM
    Filmy
    INNER JOIN FilmyDoGatunki ON Filmy.IdFilmu = FilmyDoGatunki.IdFilmu
    INNER JOIN Gatunki ON FilmyDoGatunki.IdGatunku = Gatunki.IdGatunku
WHERE Gatunek = @Gatunek
```

i) Procedura umożliwiająca podgląd filmów według reżysera

```
Create PROC proc_Filmy_Wedlug_Rezyserow @Imię nvarchar(30), @Nazwisko nvarchar(40)
AS
SELECT Filmy.IdFilmu, Tytuł, InnyTytuł, CzasTrwania, RokWydania, OcenaIMDB,Obejrzany, PozycjaNaRegale,
Rezyserzy.IdRezysera,Imię AS [Imię Reżysera], Nazwisko as [Nazwisko Reżysera]
FROM
    Filmy
    INNER JOIN FilmyDoRezyserzy ON Filmy.IdFilmu = FilmyDoRezyserzy.IdFilmu
    INNER JOIN Rezyserzy ON FilmyDoRezyserzy.IdRezysera = Rezyserzy.IdRezysera
WHERE Rezyserzy.Imię = @Imię AND Rezyserzy.Nazwisko = @Nazwisko
```

j) Procedura umożliwiająca podgląd filmów według kompozytora

```
Create PROC proc_Filmy_Wedlug_Kompozytora @Imię nvarchar(30), @Nazwisko nvarchar(40)
AS
SELECT Filmy.IdFilmu, Tytuł, InnyTytuł, CzasTrwania, RokWydania, OcenaIMDB,Obejrzany, PozycjaNaRegale,
Kompozytorzy.IdKompozytora,Imię AS [Imię Kompozytora], Nazwisko as [Nazwisko Kompozyrora]
FROM
    Filmy
    INNER JOIN FilmyDoKompozytorzy ON Filmy.IdFilmu = FilmyDoKompozytorzy.IdFilmu
    INNER JOIN Kompozytorzy ON FilmyDoKompozytorzy.IdKompozytora = Kompozytorzy.IdKompozytora
WHERE Kompozytorzy.Imię = @Imię AND Kompozytorzy.Nazwisko = @Nazwisko
```

k) Procedura umożliwiająca dodanie nowego filmu.

```
Create PROC proc_Dodaj_Nowy_Film
@IdFilmu varchar(10),
@Tytuł nvarchar(100),
@InnyTytuł nvarchar(100) = NULL,
@RokWydania int = NULL,
@CzasTrwania int = NULL,
@Ograniczenia int = NULL,
@Kolor bit = NULL,
@PozycjaNaRegale varchar(5),
@OcenaIMDB float = NULL,
@Obejrzany bit
AS
Insert Into Filmy
VALUES(@IdFilmu, @Tytuł, @InnyTytuł, @RokWydania,@CzasTrwania,@Ograniczenia,@Kolor,@PozycjaNaRegale,@OcenaIMDB,@Obejrzany)
```

l) Procedura umożliwiająca usunięcie filmu.

```
Create PROC proc_Usun_Film
@IdFilmu varchar(10),
@Tytuł nvarchar(100)
AS
DELETE FROM Filmy
WHERE @IdFilmu = IdFilmu AND @Tytuł = Tytuł
```

m) Procedura umożliwiająca zmianę pozycji na regale dla wybranego filmu.

```
Create PROC proc_Zmien_Pozycje_Na_Regale
@IdFilmu varchar(10),
@Tytuł nvarchar(100),
@PozycjaNaRegale varchar(5)
AS
Update Filmy
SET PozycjaNaRegale = @PozycjaNaRegale
WHERE @IdFilmu = IdFilmu AND @Tytuł = Tytuł
```

n) Procedura umożliwiająca zmianę wartości Obejrzane

```
Create PROC proc_Zmiana_Obejrzany
@IdFilmu varchar(10),
@Tytuł nvarchar(100),
@Obejrzany bit
AS
Update Filmy
SET Obejrzany = @Obejrzany
WHERE @IdFilmu = IdFilmu AND @Tytuł = Tytuł
```

o) Procedura umożliwiająca dodanie nowego aktora.

```
Create PROC proc_Dodaj_Aktora
@IdAktora varchar(10),
@IdFilmu varchar(10),
@Imię nvarchar(30),
@Nazwisko nvarchar(40),
@PostaćFilmowa nvarchar(60),
@RokUrodzenia int = NULL
AS
Insert Into Obsada
VALUES(@IdAktora,@IdFilmu,@Imię,@Nazwisko,@PostaćFilmowa,@RokUrodzenia)
```

p) Procedura umożliwiająca dodanie nowego kompozytora.

```
Create PROC proc_Dodaj_Kompozytora
@IdKompozytora varchar(10),
@Imię nvarchar(30),
@Nazwisko nvarchar(40),
@DataUrodzenia date = NULL
AS
Insert Into Kompozytorzy
VALUES(@IdKompozytora,@Imię,@Nazwisko,@DataUrodzenia)
```

q)Procedura umożliwiająca dodanie nowego reżysera.

```
Create PROC proc_Dodaj_Rezysera
@IdRezysera varchar(10),
@Imię nvarchar(30),
@Nazwisko nvarchar(40),
@DataUrodzenia date = NULL
AS
Insert Into Rezyserzy
VALUES(@IdRezysera,@Imię,@Nazwisko,@DataUrodzenia)
```

r) Procedura umożliwiająca dodanie nowej nagrody.

```
Create PROC proc_Dodaj_Nagrode
@IdNagrody varchar(10),
@IdFilmu varchar(10),
@NazwaNagrody nvarchar(50) = NULL,
@Kategoria nvarchar(40) = NULL,
@RokOtrzymania int = NULL
AS
Insert Into Nagrody
VALUES(@IdNagrody,@IdFilmu,@NazwaNagrody,@Kategoria,@RokOtrzymania)
```

10. Wprowadzenie przykładowych danych

Kod dodający dane do bazy danych będzie dołączony w pliku o nazwie „Dane.sql”

Dostępny również na Github:
https://github.com/Konserwa203/Bazy-Danych → „Dane.sql”