Michał Krzyżowski, grupa 1b, informatyka bazy danych 2

23.05.2020

BD 2 projekt – Lista przebojów

Spis treści

[1. Cel i wymagania 1](#_Toc41147434)

[2. DZE 3](#_Toc41147435)

[3. Model relacyjny 4](#_Toc41147436)

[4. Normalizacja: 4](#_Toc41147437)

[5. Zasad poprawności danych: 4](#_Toc41147438)

[6. Definicja schematu bazy danych – utworzenie bazy danych: 5](#_Toc41147439)

[7. 8. Definicja niedeklaratywnych mechanizmów sprawdzania poprawności danych 9](#_Toc41147440)

[9. Kod wspomagający aplikacje użytkową 12](#_Toc41147441)

[10 Wprowadzenie przykładowych danych 14](#_Toc41147442)

# Cel i wymagania

Celem systemu jest przechowywanie podstawowych informacji dotyczących przebojów muzycznych w sposób, który umożliwia intuicyjną modyfikację i intuicyjny przegląd danych.

Wymagania:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lp | Opis wymagania | Waga (0-1) | Źródło | Miara | Uwagi |
| 1. | Możliwość dodawania nowych utworów do bazy i możliwość edycji informacji na ich temat. | 1 | Internet, albumy muzyczne |  |  |
| 2. | Przy wprowadzaniu nowej informacji do bazy każdy wymagany atrybut musi zostać uzupełniony | 1 | Internet, albumy muzyczne |  |  |
| 3. | Możliwość dodanie/ edycji wykonawcy wraz z informacjami o nim do bazy. | 0.75 | Internet, biografie |  |  |
| 4. | Możliwość dodania/ edycji albumu wraz z informacjami o nim do bazy. | 0.25 | Internet, albumy muzyczne |  |  |
| 5. | Możliwość dodania/ edycji gatunku i informacji o nim do bazy. | 0.75 | Internet, albumy muzyczne, encyklopedie |  |  |
| 6. | Możliwość dodania/ edycji nagrody i informacji na jej temat do bazy | 0.25 | Internet |  |  |
| 7. | Użytkownik: Administrator – dostęp do wszystkich części bazy, pełne prawo edytowania, tworzenia usuwania rekordów, wystawiania ocenAdministratora, brak uprawnień do wystawiania ocenAdministratorki. | 1 |  |  |  |
| 8. | Użytkownik: Gość – dostęp do wszystkich części bazy, bez prawa edycji, tworzeni i usuwania rekordów, brak uprawnień do wystawiania ocen. | 1 |  |  |  |
| 9. | Użytkownik: Administratorka – dostęp do wszystkich części bazy, pełne prawo edycji, tworzenia, usuwania rekordów, brak uprawnień do wystawiania ocenyAdministratora, uprawnienie do wystawienia ocenyAdministratorki. | 1 |  |  |  |
| 10. | Możliwość dodanie/ edycji ocen wraz z danymi odnośnie werdyktu. | 0.25 | Administrator,  administratorka |  |  |
| 11 | Daty wprowadzane do Bazy muszą mieć sens logiczny – jeżeli jakaś rzecz nie mogła się wydarzyć przed dniem dzisiejszy powinna możliwość dodania takiej daty zostać zablokowana | 0.1 |  |  |  |

# DZE

Należy

Wykonuje

Zawiera

Otrzymuje

\*<M> - oznacza atrybut obowiązkowy

1..N

0..N

1..1

N..M

1..1

1..N

0..N

1..N

Gatunek

**NazwaGatunku <M>**  
DataPowstania  
MiejscePowstania  
Instrumenty

Utwory

**IdUtworu <M>**  
NazwaUtworu <M>  
DataWykonania  
OcenaUtworu <M>  
OpisUtworu  
Długość  
OcenaAdministratora  
OcenaAdministratorki  
Komentarz

Nagrody

**IdNagrody <M>**  
NazwaNagrody  
Kategoria  
Pierwsze wręczenie

Wykonawcy

**IdWykonawcy <M>**  
Imię <M>  
Nazwisko <M>  
DataUrodzenia  
TypGłosu

Album

**IdAlbumu <M>**  
NazwaAlbumu <M>  
DataWydania  
RozpoczęcieNagrań  
Wydawnictwo

# Model relacyjny





# Normalizacja:

1NF – mówi o atomowości danych, każde pole ma przechowywać jedną informacje. Wydaje mi się iż moja baza danych posiada atomowe wartości – nie widzę danych, które wymagałyby rozbicia na mniejsze. Każda encja posiada też swój klucz główny, który jednoznacznie identyfikuje wartości w tabeli.

2NF – W niej każda tabela powinna przechowywać dane dotyczące tylko konkretnej klasy obiektów. W pierwszych wersjach Nie istniała tabela oceny – ocena znajdowała się w tabeli utwory. Postanowiłem, że wyciągnę tę daną do osobnej tabeli. Rozważałem tu usunięcie kolumny komentarze, z tabeli Oceny jako że byłaby to redundancja związana z kolumną Opis z klasy utwory. Uznałem jednak, że komentarz jest opinią, a opis jest faktem, czyli tyczy się tak naprawdę innych rzeczy.

3NF - zakłada że każdy niekluczowy argument jest bezpośrednio zależny tylko od klucza głównego, a nie od innej kolumny. Nie widzę w mojej Bazie takiego przypadku – więć zakładam iż moja baza jest w 3-ciej postaci normalnej.

# Zasad poprawności danych:

Data powstania gatunku nie powinna być późniejsza niż obecna data, podobnie jak data wydania albumu, data wykonania (tabela utworu), data urodzenia/ powstania (w tabeli wykonawcy), pierwsze wręczenie (tabela nagrody), dataOtrzymania (encja otrzymanie nagrody).

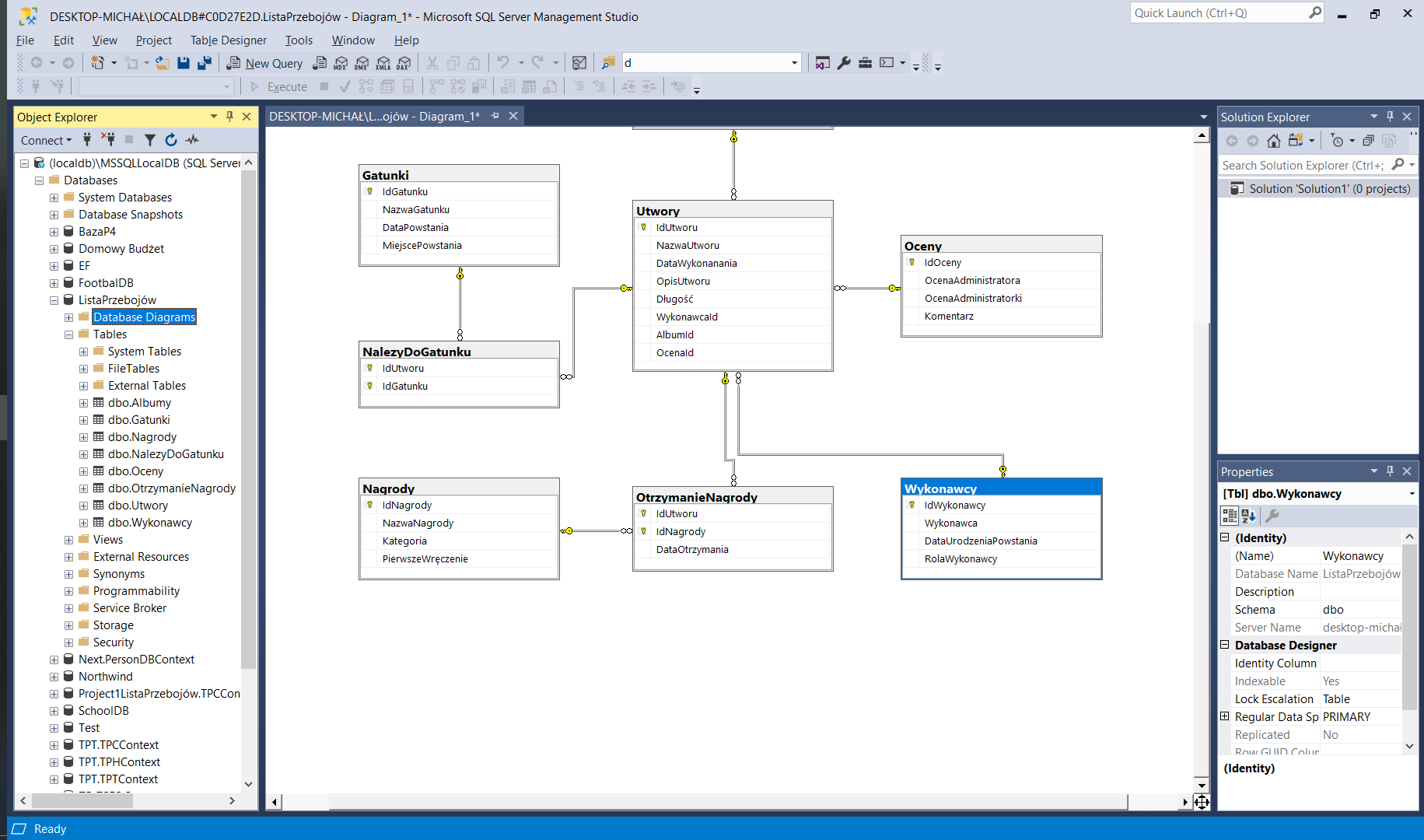
Data wydania albumu musi być późniejsza niż data rozpoczęcia nagrań.

Przy dodawaniu utworu do bazy jego nazwa, nie może być polem pusty. Tak samo do każdego utworu powinien być przypisany album, wykonawca, gatunki (-ek).

Analogiczna zasada dla innych tabel – dodając nowy album do bazy, należy podać jego nazwę (pole to nie może pozostać puste), wprowadzając nowy gatunek należy pamiętać o podaniu nazwy gatunku, dodając wykonawcę także wymagana jest nazwa wykonawcy/ zespołu. Przy dodawaniu utworu należy pamiętać o wpisaniu w tabeli Oceny wartości dla ocena administratora.

# Definicja schematu bazy danych – utworzenie bazy danych:

Skrypt sqlowi znajduje się w pliku – zadanie6.sql, a także poniżej:



* [NACIŚNIJ TUTAJ ABY PRZESKOCZYĆ SKRYPT 🡨](#_8._Definicja_niedeklaratywnych)

USE [ListaPrzebojów]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Albumy] Script Date: 21.05.2020 20:59:26 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Albumy](

[IdAlbum] [int] NOT NULL,

[NazwaAlbumu] [nvarchar](50) NOT NULL,

[DataWydania] [datetime] NULL,

[RozpoczęcieNagrań] [datetime] NULL,

[Wydawnictwo] [nvarchar](100) NULL,

CONSTRAINT [PK\_Albumy] PRIMARY KEY CLUSTERED

(

[IdAlbum] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Gatunki] Script Date: 21.05.2020 20:59:26 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Gatunki](

[IdGatunku] [int] NOT NULL,

[NazwaGatunku] [nvarchar](20) NOT NULL,

[DataPowstania] [datetime] NULL,

[MiejscePowstania] [nvarchar](50) NULL,

CONSTRAINT [PK\_Gatunki] PRIMARY KEY CLUSTERED

(

[IdGatunku] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Nagrody] Script Date: 21.05.2020 20:59:26 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Nagrody](

[IdNagrody] [int] NOT NULL,

[NazwaNagrody] [nvarchar](50) NULL,

[Kategoria] [varchar](50) NULL,

[PierwszeWręczenie] [datetime] NULL,

CONSTRAINT [PK\_Nagrody] PRIMARY KEY CLUSTERED

(

[IdNagrody] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[NalezyDoGatunku] Script Date: 21.05.2020 20:59:26 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[NalezyDoGatunku](

[IdUtworu] [int] NOT NULL,

[IdGatunku] [int] NOT NULL,

CONSTRAINT [PK\_NalezyDoGatunku] PRIMARY KEY CLUSTERED

(

[IdUtworu] ASC,

[IdGatunku] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Oceny] Script Date: 21.05.2020 20:59:26 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Oceny](

[IdOceny] [int] NOT NULL,

[OcenaAdministratora] [char](2) NOT NULL,

[OcenaAdministratorki] [char](2) NULL,

[Komentarz] [varchar](100) NULL,

CONSTRAINT [PK\_Oceny] PRIMARY KEY CLUSTERED

(

[IdOceny] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[OtrzymanieNagrody] Script Date: 21.05.2020 20:59:26 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[OtrzymanieNagrody](

[IdUtworu] [int] NOT NULL,

[IdNagrody] [int] NOT NULL,

[DataOtrzymania] [datetime] NULL,

CONSTRAINT [PK\_OtrzymanieNagrody] PRIMARY KEY CLUSTERED

(

[IdUtworu] ASC,

[IdNagrody] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Utwory] Script Date: 21.05.2020 20:59:26 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Utwory](

[IdUtworu] [int] NOT NULL,

[NazwaUtworu] [nvarchar](50) NOT NULL,

[DataWykonanania] [datetime] NULL,

[OpisUtworu] [nvarchar](100) NULL,

[Długość] [char](5) NULL,

[WykonawcaId] [int] NOT NULL,

[AlbumId] [int] NOT NULL,

[OcenaId] [int] NOT NULL,

CONSTRAINT [PK\_Utwory] PRIMARY KEY CLUSTERED

(

[IdUtworu] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Wykonawcy] Script Date: 21.05.2020 20:59:26 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Wykonawcy](

[IdWykonawcy] [int] NOT NULL,

[Wykonawca] [nvarchar](50) NOT NULL,

[DataUrodzeniaPowstania] [date] NULL,

[RolaWykonawcy] [varchar](50) NULL,

CONSTRAINT [PK\_Wykonawcy] PRIMARY KEY CLUSTERED

(

[IdWykonawcy] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[NalezyDoGatunku] WITH CHECK ADD CONSTRAINT [FK\_NalezyDoGatunku\_Gatunki] FOREIGN KEY([IdGatunku])

REFERENCES [dbo].[Gatunki] ([IdGatunku])

GO

ALTER TABLE [dbo].[NalezyDoGatunku] CHECK CONSTRAINT [FK\_NalezyDoGatunku\_Gatunki]

GO

ALTER TABLE [dbo].[NalezyDoGatunku] WITH CHECK ADD CONSTRAINT [FK\_NalezyDoGatunku\_Utwory] FOREIGN KEY([IdUtworu])

REFERENCES [dbo].[Utwory] ([IdUtworu])

GO

ALTER TABLE [dbo].[NalezyDoGatunku] CHECK CONSTRAINT [FK\_NalezyDoGatunku\_Utwory]

GO

ALTER TABLE [dbo].[OtrzymanieNagrody] WITH CHECK ADD CONSTRAINT [FK\_OtrzymanieNagrody\_Nagrody] FOREIGN KEY([IdNagrody])

REFERENCES [dbo].[Nagrody] ([IdNagrody])

GO

ALTER TABLE [dbo].[OtrzymanieNagrody] CHECK CONSTRAINT [FK\_OtrzymanieNagrody\_Nagrody]

GO

ALTER TABLE [dbo].[OtrzymanieNagrody] WITH CHECK ADD CONSTRAINT [FK\_OtrzymanieNagrody\_Utwory] FOREIGN KEY([IdUtworu])

REFERENCES [dbo].[Utwory] ([IdUtworu])

GO

ALTER TABLE [dbo].[OtrzymanieNagrody] CHECK CONSTRAINT [FK\_OtrzymanieNagrody\_Utwory]

GO

ALTER TABLE [dbo].[Utwory] WITH CHECK ADD CONSTRAINT [FK\_Utwory\_Albumy] FOREIGN KEY([AlbumId])

REFERENCES [dbo].[Albumy] ([IdAlbum])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Utwory] CHECK CONSTRAINT [FK\_Utwory\_Albumy]

GO

ALTER TABLE [dbo].[Utwory] WITH CHECK ADD CONSTRAINT [FK\_Utwory\_Oceny] FOREIGN KEY([OcenaId])

REFERENCES [dbo].[Oceny] ([IdOceny])

GO

ALTER TABLE [dbo].[Utwory] CHECK CONSTRAINT [FK\_Utwory\_Oceny]

GO

ALTER TABLE [dbo].[Utwory] WITH CHECK ADD CONSTRAINT [FK\_Utwory\_Wykonawcy] FOREIGN KEY([WykonawcaId])

REFERENCES [dbo].[Wykonawcy] ([IdWykonawcy])

GO

ALTER TABLE [dbo].[Utwory] CHECK CONSTRAINT [FK\_Utwory\_Wykonawcy]

GO

Plik z zadaniem można znaleźć także na repozytorium:

<https://github.com/Jarverr/ProjektBazyDanych>

# 8. Definicja niedeklaratywnych mechanizmów sprawdzania poprawności danych

Wyzwalacz mający za cel sprawdzać, czy aby data urodzenia nie jest późniejsza niż data dzisiejszego dnia:

USE [ListaPrzebojów]

GO

/\*\*\*\*\*\* Object: Trigger [dbo].[tooLateDataErrorWykonawcy] Script Date: 23.05.2020 16:05:33 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER TRIGGER [dbo].[tooLateDataErrorWykonawcy] ON [dbo].[Wykonawcy]

FOR INSERT

NOT FOR REPLICATION

AS

IF(SELECT COUNT(\*) FROM inserted)>=1

BEGIN

IF((SELECT DataUrodzeniaPowstania from inserted)>GETDATE())

BEGIN

RAISERROR('Data urodzenia/ powstania zespołu nie może być późniejsza niż data dzisiejsza',16,1)

ROLLBACK TRANSACTION

END

END

Kolejny trigger opisujący podobnie jak poprzedni co ma się wydarzyć gdy data wydania utworu będzie późniejsza od daty dzisieijszej:

USE [ListaPrzebojów]

GO

/\*\*\*\*\*\* Object: Trigger [dbo].[DateIsNotCorrect] Script Date: 23.05.2020 16:05:08 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER TRIGGER [dbo].[DateIsNotCorrect] ON [dbo].[Utwory]

FOR INSERT

NOT FOR REPLICATION

AS

IF (SELECT COUNT(\*) FROM inserted)=1

BEGIN

IF(Select YEAR(i.DataWykonanania)from inserted i)>=YEAR(GETDATE())

BEGIN

RAISERROR('Ta data wykonania jeszcze nie została osiągnięta przez czas',16,1)

ROLLBACK TRANSACTION

END

END

Analogicznie dla daty w tabeli nagrody

USE [ListaPrzebojów]

GO

/\*\*\*\*\*\* Object: Trigger [dbo].[tooLateDataErrorNagrody] Script Date: 23.05.2020 16:02:51 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER TRIGGER [dbo].[tooLateDataErrorNagrody] ON [dbo].[Nagrody]

FOR INSERT

NOT FOR REPLICATION

AS

IF(SELECT COUNT(\*) FROM inserted)>=1

BEGIN

IF((SELECT PierwszeWręczenie from inserted)>GETDATE())

BEGIN

RAISERROR('Data pierwszęgo wręczenia nie może być późniejsza niż data dzisiejsza',16,1)

ROLLBACK TRANSACTION

END

END

Podoba zasada odnośnie dat w tabeli gatunki:

USE [ListaPrzebojów]

GO

/\*\*\*\*\*\* Object: Trigger [dbo].[tooLateDataErrorGatunki] Script Date: 23.05.2020 16:03:06 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER TRIGGER [dbo].[tooLateDataErrorGatunki] ON [dbo].[Gatunki]

FOR INSERT

NOT FOR REPLICATION

AS

IF(SELECT COUNT(\*) FROM inserted)>=1

BEGIN

IF((SELECT DataPowstania from inserted)>GETDATE())

BEGIN

RAISERROR('Data powstania nie może być późniejsza niż data dzisiejsza',16,1)

ROLLBACK TRANSACTION

END

END

Dla albumów musi sprawdzić dwie daty:

USE [ListaPrzebojów]

GO

/\*\*\*\*\*\* Object: Trigger [dbo].[tooLateDataErrorAlbums] Script Date: 23.05.2020 16:04:03 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER TRIGGER [dbo].[tooLateDataErrorAlbums] ON [dbo].[Albumy]

FOR INSERT

NOT FOR REPLICATION

AS

IF(SELECT COUNT(\*) FROM inserted)>=1

BEGIN

IF((SELECT DataWydania from inserted)>GETDATE())

BEGIN

RAISERROR('Data wydania nie może być późniejsza niż data dzisiejsza',16,1)

ROLLBACK TRANSACTION

END

IF((SELECT RozpoczęcieNagrań from inserted)>GETDATE())

BEGIN

RAISERROR('Data rozpoczęcia nagrań nie może być późniejsza niż data dzisiejsza',16,1)

ROLLBACK TRANSACTION

END

END

Wartości będące zakazanymi jako puste, zostały zdefiniowane w czasie tworzenia bazy danych – poprzez dodania do tworzonej kolumny dopisku NOT NULL (podobnie jak albumy każda inna tabela dostała podobną właściwość dla odpowiednich kolumn).

CREATE TABLE [dbo].[Albumy](

[IdAlbum] [int] NOT NULL,

[NazwaAlbumu] [nvarchar](50) NOT NULL,

[DataWydania] [datetime] NULL,

[RozpoczęcieNagrań] [datetime] NULL,

[Wydawnictwo] [nvarchar](100) NULL,

CONSTRAINT [PK\_Albumy] PRIMARY KEY CLUSTERED

(

[IdAlbum] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

W pliku zadanie7.sql (oraz poniżej) znajduje się kod dodatkowy, który prezentuje co byłoby wymagane w razie potrzeby modyfikacji własności null na not null dla tabel:

ALTER TABLE dbo.Utwory ALTER COLUMN [NazwaUtworu] [nvarchar] NOT NULL

GO

ALTER TABLE dbo.Gatunki ALTER COLUMN [NazwaGatunku] [nvarchar] NOT NULL

GO

ALTER TABLE dbo.Albumy ALTER COLUMN [NazwaAlbumu] [nvarchar] NOT NULL

GO

ALTER TABLE dbo.Wykonawcy ALTER COLUMN Wykonawca [nvarchar] NOT NULL

GO

# Kod wspomagający aplikacje użytkową

Poniżej kod, który ma wspomóc tworzenie aplikacji użytkowej (kod znajduje się także w pliku zadanie9.sql)

Funkcja która wyświetla utwory o ocenie większej niż 5:

DROP FUNCTION IF EXISTS dbo.f\_OcenaAdministratoraWiekszaNiz5

GO

CREATE FUNCTION f\_OcenaAdministratoraWiekszaNiz5 ()

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM Utwory INNER JOIN Oceny on oceny.IdOceny=Utwory.OcenaId

where OcenaAdministratora>5

);

GO

Funkcja wyświetlająca oceny na poziomie 10:

DROP FUNCTION IF EXISTS dbo.f\_NajlepszeUtworyOcenaAdministratoraRowna10

GO

CREATE FUNCTION f\_NajlepszeKawałki ()

RETURNS TABLE

AS

RETURN

(

SELECT \* FROM Utwory INNER JOIN Oceny on oceny.IdOceny=Utwory.OcenaId

where OcenaAdministratora=10

);

GO

Funkcja wyświetlająca wszystkie dane z bazy danych:

DROP FUNCTION IF EXISTS dbo.f\_WszystkieDane

GO

CREATE FUNCTION f\_WszystkieDane ()

RETURNS TABLE

AS

RETURN

(

SELECT u.IdUtworu,u.AlbumId,u.WykonawcaId,u.OcenaId,ndo.IdGatunku,ng.IdNagrody,

u.NazwaUtworu,u.DataWykonanania,u.Długość,u.OpisUtworu,

o.OcenaAdministratora,o.OcenaAdministratorki,o.Komentarz,

a.NazwaAlbumu,a.RozpoczęcieNagrań,a.DataWydania,a.Wydawnictwo,

w.Wykonawca,w.DataUrodzeniaPowstania,w.RolaWykonawcy,

g.NazwaGatunku,g.DataPowstania,g.MiejscePowstania,

n.NazwaNagrody,n.PierwszeWręczenie,n.Kategoria

FROM Utwory u JOIN oceny o ON u.IdUtworu=o.IdOceny

JOIN Albumy a ON a.IdAlbum=u.AlbumId

JOIN Wykonawcy w ON w.IdWykonawcy=u.WykonawcaId

JOIN NalezyDoGatunku ndo ON ndo.IdUtworu=u.IdUtworu

JOIN Gatunki g ON g.IdGatunku=ndo.IdGatunku

JOIN OtrzymanieNagrody ng ON ng.IdUtworu=u.IdUtworu

JOIN Nagrody n ON n.IdNagrody=ng.IdNagrody

);

GO

Funkcja wyświetlająca utwory, które otrzymały jakąkolwiek nagrodę:

DROP FUNCTION IF EXISTS dbo.f\_UtworyZNagrodami

GO

CREATE FUNCTION f\_UtworyZNagrodami ()

RETURNS TABLE

AS

RETURN

(

SELECT u.IdUtworu,u.NazwaUtworu,u.DataWykonanania,u.Długość,u.OpisUtworu,n.NazwaNagrody,onag.DataOtrzymania FROM Utwory u INNER JOIN OtrzymanieNagrody onag on onag.IdUtworu=u.IdUtworu

INNER JOIN Nagrody n on n.IdNagrody=onag.IdNagrody

where onag.IdUtworu IS NOT NULL

);

GO

Funkcja wyświetlająca wykonawcę i jego albumy:

DROP FUNCTION IF EXISTS dbo.f\_WykonawcyIIchAlbumy

GO

CREATE FUNCTION f\_WykonawcyIIchAlbumy ()

RETURNS TABLE

AS

RETURN

(

SELECT w.Wykonawca, a.NazwaAlbumu, a.RozpoczęcieNagrań, a.DataWydania FROM Wykonawcy w inner join Utwory u on u.WykonawcaId=w.IdWykonawcy

INNER JOIN Albumy a on a.IdAlbum=u.AlbumId

);

GO

Funkcja wyświetlająca gatunki, w których tworzył/ tworzyli wykonawcy:

DROP FUNCTION IF EXISTS dbo.f\_wykonawcyIIchGatunkui

GO

CREATE FUNCTION f\_wykonawcyIIchGatunkui ()

RETURNS TABLE

AS

RETURN

(

SELECT w.Wykonawca, g.NazwaGatunku, g.MiejscePowstania, g.DataPowstania FROM Wykonawcy w inner join Utwory u on u.WykonawcaId=w.IdWykonawcy

INNER JOIN NalezyDoGatunku a on a.IdUtworu=u.IdUtworu

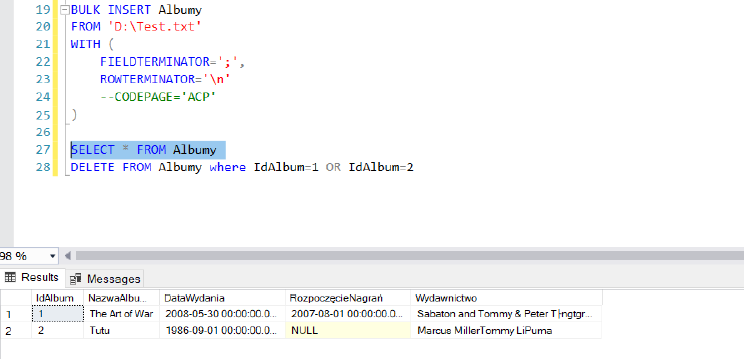
INNER JOIN Gatunki g on g.IdGatunku=a.IdGatunku

);

GO

# Wprowadzenie przykładowych danych

Dane wprowadzić można poprzez polecenie BULK INSERT np.



Pliki zawierające przykładowe dane znajdują pod nazwami Albumy.txt, Wykonawcy.txt etc. Przy dodawaniu ustawione są też dla utworu odpowiednie id albumu, id wykonawcy itd. Dla tego utwory zalecane aby dodać na końcu by referencja mogła powstać.

Dodatkowo przygotowany jest kod wstawiający w pliku zadanie10.sql:

USE ListaPrzebojów

GO

--GATUNKI

INSERT INTO Gatunki VALUES (1,'Metal','19650101','USA/UK'),(2,'Jazz','19100101','Nowy Orlean');

--ALBUMY

INSERT INTO Albumy VALUES (1,'Tha art of War','20080530','20070801','Black Lodge Records'),(2,'Tutu','19860901',Null,'Marcus MillerTommy LiPuma');

--NAGRODY

INSERT INTO Nagrody VALUES(1,'Grammis Award for Hard Rock/Metal of the Year',NULL,'19900221'),(2,'Nagroda Grammy','Najlepsza solowa improwizacja jazzowa','19590101');

--WYKONAWCY

INSERT INTO Wykonawcy VALUES (1,'Sabaton','19990101','Zespol'),(2,'Miles Davis','19260526','trebacz, lider, kompozytor')

--OCENY

INSERT INTO Oceny VALUES (1,9,3,'Dobry kawalek o mestwie i wojnie'),(2,9,7,'Stare przeboje');

--UTWORY

INSERT INTO Utwory VALUES ( 1,'40:1','20080520',NULL,'4:11',1,1,1),(2,'Tutu','19860901','Stary jazzowy kawalek, z trabka','5:18',2,2,2);

INSERT INTO NalezyDoGatunku VALUES (1,1),(2,2)

INSERT INTO OtrzymanieNagrody VALUES (1,1),(2,2)

Pliki znaleźć można pod linkiem : <https://github.com/Jarverr/ProjektBazyDanych> (repozytorium).