Podstawy Baz Danych

**Projekt: System zarządzania konferencjami**

Dokumentacja

Wiktor Łęczyński, Dawid Wełna

**1.Opis systemu**

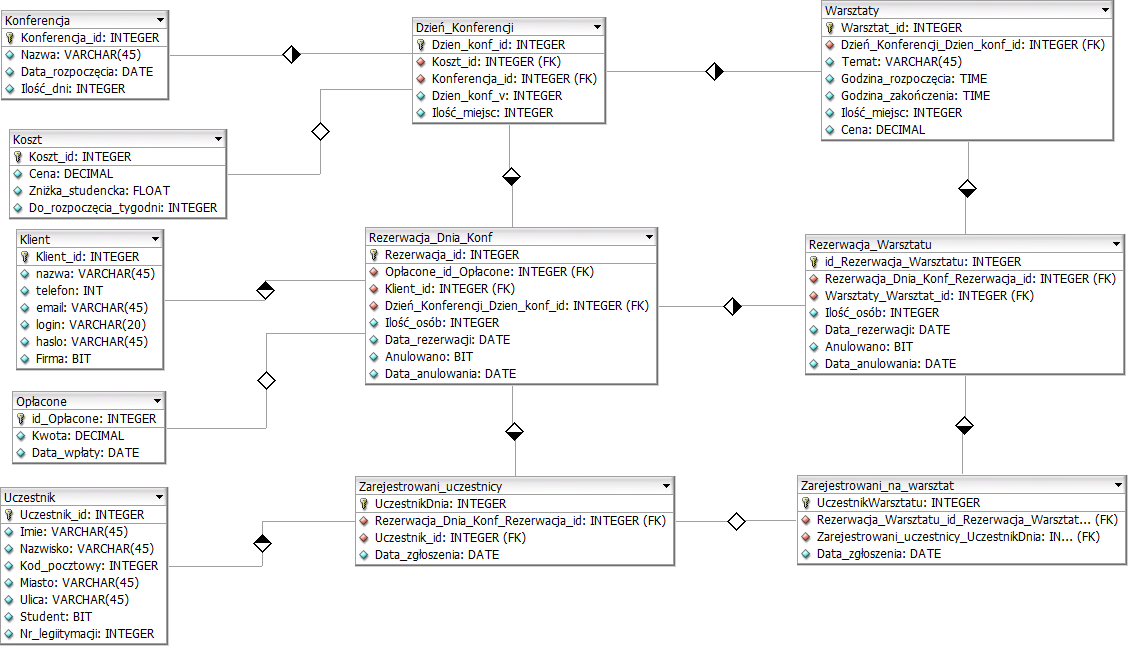
System obsługi rezerwacji konferencji . Konferencje mogą być kilkudniowe .

Klientami mogą być firmy jak i osoby .Uczestnikami sa osoby i studenci .Klient nie musi przy rezerwacji podawać Ilosci osób , jednak musi je uzupełnić 2 tyg przed terminem.

Opłata musi być zapłacona maksymalnie tydzień przed terminem.

Każda konferencja posiada wyarsztaty . Kilka wykładów mogą trwać jednoczesnie .

**2.Schemat**



**3.Opis**

**Klient**- tabela rezprezentująca klientów w bazie danych. Każdy klient posiada dane(Nazwa,telefon,email,login,haslo,firma)

Mail i Login musza być unikalne , telefon musi się składac z 9 cyfr .Email musi mieć forme emailu

CREATE TABLE Klient (

Klient\_id INTEGER NOT NULL Primary key IDENTITY(1,1),

nazwa VARCHAR(45) unique NOT NULL,

telefon INTEGER NULL Check(telefon>100000000 and telefon<999999999),

email VARCHAR(45) NOT NULL Unique CHECK (email LIKE '%\_@\_%.\_%'),

login VARCHAR(20) NOT NULL Unique ,

haslo VARCHAR(45) NOT NULL ,

Firma BIT not null default 0,

);

**Rezerwacja\_Dnia\_Konf** – tabela reprezentująca rezerwacje na jakiś dzień konferencji.

Połączona jest z tablica rezerwacji , oplacone , Klient oraz Dzień Konferencji\_Dzien\_konf\_id.

W tabeli tej znajduja się kolumny ilosc osob , wartosc ta musi być wieksza od 0

Data\_rezerwacji która zawiera date rezerwacji, Anulowano jeśli się rowna 1 to w Data anulowano znajduje się data anulowania

CREATE TABLE Rezerwacja\_Dnia\_Konf (

Rezerwacja\_id INTEGER NOT NULL Primary key IDENTITY(1,1),

Opłacone\_id\_Opłacone Integer null,

Klient\_id INTEGER NOT NULL ,--

Dzień\_Konferencji\_Dzien\_konf\_id INTEGER NOT NULL, --

Ilość\_osób INTEGER NOT NULL check(ilość\_osób>0),

Data\_rezerwacji DATE NOT NULL,

Anulowano BIT NOT NULL default 0,

Data\_anulowania DATE NULL,

);

**Oplacone –** tabela reprezentująca opłate , w kolumnie Kwota znajduje się kwota wieksza od 0 , Data wpłaty znajduje się data

CREATE TABLE Opłacone (

id\_Opłacone INTEGER NOT NULL Primary key IDENTITY(1,1),

Kwota DECIMAL NOT NULL CHECK (Kwota > 0),

Data\_wpłaty DATE NOT NULL,

);

**Zarejestrowani\_uczestnicy**-tabela reprezentująca zarejestrowanych uczestików . Połączona z tablica RezerwacjaDniaKonf\_Rezerwacja , Uczestnik . Zawiera Date zgloszenia.

CREATE TABLE Zarejestrowani\_uczestnicy (

Uczestnik\_Dnia INTEGER NOT NULL Primary key IDENTITY(1,1),

Rezerwacja\_Dnia\_Konf\_Rezerwacja\_id INTEGER NOT NULL, --

Uczestnik\_id INTEGER NOT NULL ,--

Data\_zgłoszenia DATE NULL, );

**Uczestnik-**tabela reprezentująca uczestników . Połączona z tablicą Student .

Tabela zawiera kolumny Imie ,Nazwisko, Kod Pocztowy, Miasto , Ulica oraz Student.

Jeśli student jest rowne 1 to w polu Nr\_legitymacji jest nr legitymacji studenta

CREATE TABLE Uczestnik (

Uczestnik\_id INTEGER NOT NULL Primary key IDENTITY(1,1),

Imie VARCHAR(45) NOT NULL,

Nazwisko VARCHAR(45) NOT NULL,

Kod\_pocztowy INTEGER NULL Check(Kod\_pocztowy>10000 and Kod\_pocztowy<99999),

Miasto VARCHAR(45) NULL,

Ulica VARCHAR(45) NULL,

Student BIT NOT NULL default 0,

Nr\_legitymacji INTEGER NULL Check(Nr\_legitymacji>0 )

);

**Zarejestrowani\_na\_warsztat**-tablica reprezentująca zarejestrowanych na warsztat .

Połączona z tablica Rezerwacja\_Warsztatu oraz Zarejstowani\_uczestnicy . Zawiera Date zgloszenia

CREATE TABLE Zarejestrowani\_na\_warsztat (

Uczestnik\_Warsztatu INTEGER NOT NULL Primary key IDENTITY(1,1),

Rezerwacja\_Warsztatu\_id\_Rezerwacja\_Warsztatu INTEGER NOT NULL, --

Zarejestrowani\_uczestnicy\_Uczestnik\_Dnia INTEGER NOT NULL , --

Data\_zgłoszenia DATE NULL,

);

**Rezerwacja\_Warsztatu**- tablica reprezentująca rezerwacje warszatu . Połączona z tabela Warsztatów i Rezerwacja Warsztów . Zawiera kolumny Ilosc osob (wieksze od 0) , Anulowano , Jeśli Anulowano =1 to Data anulowania zawiera Date anulowania.

CREATE TABLE Rezerwacja\_Warsztatu (

id\_Rezerwacja\_Warsztatu INTEGER NOT NULL Primary key IDENTITY(1,1),

Rezerwacja\_Dnia\_Konf\_Rezerwacja\_id INTEGER NOT NULL,

Warsztaty\_Warsztat\_id INTEGER NOT NULL, --

Ilość\_osób INTEGER NOT NULL check(Ilość\_osób>0),

Data\_rezerwacji DATE NOT NULL,

Anulowano BIT NOT NULL default 0,

Data\_anulowania DATE NULL,

);

**Warszaty** –tablica reprezentująca warstaty . Polaczona jest z tablica Dzien Konferencji . Zawiera Temat , Godzine rozpoczecia , Godzine zakończenia , Ilość miejsc (wieksze od 0) oraz Cene

CREATE TABLE Warsztaty (

Warsztat\_id INTEGER NOT NULL Primary key IDENTITY(1,1),

Dzień\_Konferencji\_Dzien\_konf\_id Integer NOT NULL,

Temat VARCHAR(45) NOT NULL,

Godzina\_rozpoczęcia TIME NOT NULL,

Godzina\_zakończenia TIME NOT NULL,

Ilość\_miejsc INTEGER NULL check(Ilość\_miejsc>0),

Cena DECIMAL NULL check (Cena>0),

);

**Dzień\_Konferencji** – tablica reprezentująca Dni konferencji . Połączona jest z tablica Koszt oraz Konferencja . Zawiera Dzien konferencji oraz ilość miejsc

CREATE TABLE Dzień\_Konferencji (

Dzien\_konf\_id INTEGER NOT NULL Primary key IDENTITY(1,1),

Konferencja\_id INTEGER NOT NULL,

Koszt\_id INTEGER NOT NULL,

dzien\_konf\_v INTEGER NOT NULL check(dzien\_konf\_v > 0 and dzien\_konf\_v < 10),

ilość\_miejsc INTEGER NOT NULL check(ilość\_miejsc>0),

);

**Konferencja**- tablica reprezentujaca konferencje . Zawiera nazwe , Date rozpoczecia , Ilość dni

CREATE TABLE Konferencja (

Konferencja\_id INTEGER NOT NULL Primary key IDENTITY(1,1),

Nazwa VARCHAR(45) Not NULL,

Data\_rozpoczęcia DATE NOT NULL,

Ilość\_dni INTEGER NOT NULL check(ilość\_dni>0),

);

**Koszt**-tablica zawierajaca koszty . Zawiera Cena , zniżke studencka , do rozpoczecia

CREATE TABLE Koszt (

Koszt\_id INTEGER NOT NULL Primary key IDENTITY(1,1),

Cena DECIMAL NOT NULL CHECK (Cena >= 0),

Zniżka\_studencka FLOAT NOT NULL CHECK(Zniżka\_studencka >= 0 and Zniżka\_studencka <= 100),

Do\_rozpoczęcia\_tygodni INTEGER NULL,

);

**4. Procedury**

**DodajKlienta** – procedura dodająca Klienta

CREATE PROCEDURE [dbo].[DodajKlienta]

@nazwa VARCHAR(45),

@telefon VARCHAR(45),

@email VARCHAR(45),

@login VARCHAR(45),

@haslo VARCHAR(45),

@Firma BIT

AS

BEGIN

INSERT INTO [Klient](nazwa, telefon, email, login, haslo, Firma)

VALUES (@nazwa, @telefon, @email, @login, @haslo, @Firma)

END

**DodajKonferencje** – procedura dodająca Konferencje

CREATE PROCEDURE [dbo].[DodajKonferencje]

@nazwa VARCHAR(45),

@dataRoz DATE,

@iloscDni INTEGER

AS

BEGIN

INSERT INTO [Konferencja](nazwa, Data\_rozpoczęcia, Ilość\_dni)

VALUES (@nazwa, @dataRoz, @iloscDni)

END

**DodajDzienKonferencji** – procedura dodajaca dzien konferencji

CREATE PROCEDURE [dbo].[DodajDzienKonf]

@KonferencjaId INTEGER,

@dzienKonf INTEGER,

@iloscMiejsc INTEGER

AS

BEGIN

INSERT INTO [Dzień\_Konferencji](Konferencja\_Id, dzien\_konf\_v, ilość\_miejsc)

VALUES (@KonferencjaId, @dzienKonf, @iloscMiejsc)

END

**DodajUczestnika** – procedura dodajaca uczestnika

CREATE PROCEDURE [dbo].[DodajUczestnika]

@Imie VARCHAR(45),

@Nazwisko VARCHAR(45),

@Kod\_pocztowy INTEGER,

@Miasto VARCHAR(45),

@Ulica VARCHAR(45),

@Student BIT,

@Nr\_leg INTEGER

AS

BEGIN

INSERT INTO [Uczestnik](Imie, Nazwisko, Kod\_pocztowy, Miasto, Ulica, Student, Nr\_legitymacji)

VALUES (@Imie, @Nazwisko, @Kod\_pocztowy, @Miasto, @Ulica, @Student, @Nr\_leg)

END

**dodajOplate** - procedura dodaje informacje o uiszczonej opłacie

CREATE PROCEDURE [dbo].[dodajOplate]

@Kwota DECIMAL

AS

BEGIN

INSERT INTO [Opłacone](Kwota, Data\_wpłaty)

VALUES(@Kwota, getDate())

END

**dodajKoszt** - procedura dodaje informacje o kosztach i progach cenowych w zależności od czasu rejestracji.

CREATE PROCEDURE [dbo].[dodajKoszt]

@Cena DECIMAL(5,1),

@Zniżka\_studencka FLOAT,

@do\_rozpoczecia INTEGER

AS

BEGIN

INSERT INTO [Koszt](Cena, Zniżka\_studencka, Do\_rozpoczęcia\_tygodni)

VALUES(@Cena, @Zniżka\_studencka, @do\_rozpoczecia)

END

**dodajWarsztat** – dodaje informacje o warsztatach

CREATE PROCEDURE [dbo].[dodajWarsztat]

@Warsztat\_id INTEGER,

@Dzien\_konf\_id INTEGER,

@Temat VARCHAR(45),

@Godzina\_rozpoczęcia TIME,

@Godzina\_zakończenia TIME,

@Ilość\_miejsc INTEGER,

@Cena DECIMAL

AS

BEGIN

INSERT INTO [Warsztaty](Warsztat\_id, Dzień\_Konferencji\_Dzien\_konf\_id, Temat, Godzina\_rozpoczęcia, Godzina\_zakończenia,

Ilość\_miejsc, Cena)

VALUES (@Warsztat\_id, @Dzien\_konf\_id, @Temat, @Godzina\_rozpoczęcia, @Godzina\_zakończenia, @Ilość\_miejsc, @Cena)

END

**ZarejestrujUczestnikaNaDzien –** procedura umożliwiająca rejestracje uczestnika na dzień konferencji

CREATE PROCEDURE [dbo].[zarejestrujUczestnikaNaDzien]

@Uczestnik\_id INTEGER,

@Rezerwacja\_id INTEGER

AS

BEGIN

INSERT INTO [Zarejestrowani\_uczestnicy](Rezerwacja\_Dnia\_Konf\_Rezerwacja\_id, Uczestnik\_id, Data\_zgłoszenia)

VALUES (@Rezerwacja\_id, @Uczestnik\_id, getDate())

END

**zarejestrujUczestnikaNaWarsztat** - procedura umożliwiająca rejestracje uczestnika na warsztat

CREATE PROCEDURE [dbo].[zarejestrujUczestnikaNaWarsztat]

@Uczestnik\_id INTEGER,

@idRezerwacjaWarsztatu INTEGER

AS

BEGIN

DECLARE @UczestnikDnia INT = (SELECT Uczestnik\_Dnia FROM Zarejestrowani\_uczestnicy WHERE (@Uczestnik\_id = Uczestnik\_id))

IF (@UczestnikDnia = NULL)

BEGIN

RAISERROR('Ten uczestnik nie jest jeszcze zarejestrowany na konferencje. Nie można zarejestrować go na wybrany warsztat.', 16, -1);

END

INSERT INTO [Zarejestrowani\_na\_warsztat](Rezerwacja\_Warsztatu\_id\_Rezerwacja\_Warsztatu, Zarejestrowani\_uczestnicy\_Uczestnik\_Dnia, Data\_zgłoszenia)

VALUES (@idRezerwacjaWarsztatu, @UczestnikDnia, getDate())

END

**zarezerwujMiejscaNaDzien –** procedura dzięki której dokonujemy rezerwacji dnia konferencji.

CREATE PROCEDURE [dbo].[zarezerwujMiejscaNaDzien]

@IloscOs INTEGER,

@idKlienta INTEGER,

@idDnia INTEGER

AS

BEGIN

INSERT INTO [Rezerwacja\_Dnia\_Konf](Klient\_id, Dzień\_Konferencji\_Dzien\_konf\_id, Ilość\_osób, Data\_rezerwacji)

VALUES (@idKlienta, @idDnia, @IloscOs, getDate())

END

**zarezerwujMiejscaNaWarsztat -** procedura dzięki której dokonujemy rezerwacji warsztatu.

CREATE PROCEDURE [dbo].[zarezerwujMiejscaNaWarsztat]

@IloscOs INTEGER,

@idWar INTEGER,

@rezDniaid INTEGER

AS

BEGIN

INSERT INTO [Rezerwacja\_Warsztatu](Warsztaty\_Warsztat\_id, Ilość\_osób, Rezerwacja\_Dnia\_Konf\_Rezerwacja\_id)

VALUES(@IloscOs, @idWar, @rezDniaid)

END

**anulowanieRezerwacjiWarsztatu** – procedura za pomocą której zostaje anulowana rezerwacja warszatu

CREATE PROCEDURE [dbo].[anulowanieRezerwacjiWarsztatu]

@Rezerwacja\_id INTEGER

AS

BEGIN

if( (SELECT Anulowano FROM Rezerwacja\_Warsztatu WHERE @Rezerwacja\_id = id\_Rezerwacja\_Warsztatu) = 0)

BEGIN

UPDATE Rezerwacja\_Warsztatu

SET Anulowano = 1,

Data\_anulowania = GETDATE()

WHERE id\_Rezerwacja\_Warsztatu = @Rezerwacja\_id

END

ELSE RAISERROR('Warsztat był już anulowany.', 16, -1);

END

**anulowanieRezerwacjiDnia -** procedura za pomocą której zostaje anulowana rezerwacja na dzień konferencji.

CREATE PROCEDURE [dbo].[anulowanieRezerwacjiDnia]

@Rezerwacja\_id INTEGER

AS

BEGIN

if( (SELECT Anulowano FROM Rezerwacja\_Dnia\_Konf WHERE @Rezerwacja\_id = Rezerwacja\_id) = 0)

BEGIN

UPDATE Rezerwacja\_Dnia\_Konf

SET Anulowano = 1,

Data\_anulowania = GETDATE()

WHERE Rezerwacja\_id = @Rezerwacja\_id

END

ELSE RAISERROR(‘Rezerwacja była już anulowana.', 16, -1);

END

**analujNieoplaconeRezerwacje –** procedura służąca do anulowania nieopłaconych jeszcze rezerwacji.

CREATE PROCEDURE [dbo].[anulujNieoplaconeRezerwacje]

AS

BEGIN

DECLARE tab CURSOR LOCAL FOR (SELECT rdk.Data\_rezerwacji, rdk.Rezerwacja\_id FROM Rezerwacja\_Dnia\_Konf as rdk

LEFT JOIN Opłacone as o ON rdk.Opłacone\_id\_Opłacone = o.id\_Opłacone

WHERE rdk.Opłacone\_id\_Opłacone is NULL)

DECLARE @Rezerwacja\_id INTEGER, @dataRezerwacji date

OPEN tab

FETCH NEXT FROM tab INTO @dataRezerwacji, @Rezerwacja\_id

WHILE @@FETCH\_STATUS = 0

BEGIN

IF(DATEDIFF(day,@dataRezerwacji,GETDATE()) > 7)

BEGIN

exec anulowanieRezerwacjiDnia @Rezerwacja\_id

END

FETCH NEXT FROM tab INTO @Rezerwacja\_id, @dataRezerwacji

END

CLOSE tab

DEALLOCATE tab

END

**5.Triggery**

**ZabronRejestracjiNaDwaTygPrzed** - trigger, który sprawdza czy data rezerwacji nie jest na mniej niż 2 tygodnie przed rozpoczeciem

CREATE TRIGGER [dbo].[ZabronRejestracjiNaDwaTygPrzed]

ON [dbo].[Rezerwacja\_Dnia\_Konf]

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @Rezerwacja\_id INTEGER = (SELECT Rezerwacja\_id FROM Inserted)

DECLARE @DataRez DATE = (SELECT Data\_rezerwacji FROM Inserted)

DECLARE @DataRozp DATE = (SELECT Data\_rozpoczęcia FROM Konferencja as k

JOIN Dzień\_Konferencji as dk ON k.Konferencja\_id = dk.Konferencja\_id

JOIN Rezerwacja\_Dnia\_Konf as rdk ON rdk.Dzień\_Konferencji\_Dzien\_konf\_id = dk.Dzien\_konf\_id

WHERE rdk.Rezerwacja\_id = @Rezerwacja\_id)

IF((DAY(@DataRozp) - DAY(@DataRez)) < 14)

BEGIN

RAISERROR ('Za poźno aby zarezerwować.', 16, -1);

ROLLBACK TRANSACTION

END

END

**placesOfWorkshopLessThanConfDay** - czy liczba miejsc na warsztaty tego dnia nie jest wieksza niz na konferencje odbywającą się tego dnia

CREATE TRIGGER [dbo].[placesOfWorkshopLessThanConfDay]

ON [dbo].[Warsztaty]

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @miejscaWar INTEGER = (SELECT Ilość\_miejsc FROM INSERTED);

DECLARE @DzienKonfId INTEGER = (SELECT Dzień\_Konferencji\_Dzien\_konf\_id FROM inserted)

DECLARE @miejscaKonf INTEGER = (SELECT Ilość\_miejsc FROM Dzień\_Konferencji WHERE

(@DzienKonfId = Dzień\_Konferencji.Dzien\_konf\_id))

IF( @miejscaWar > @miejscaKonf )

BEGIN

RAISERROR ('W warsztacie nie może uczestniczyć więcej osób niż w konferencji.', 16, -1);

ROLLBACK TRANSACTION

END

END

**nieMożnaRezerwowaćnaWarsztaty** - TRIGGER SPRAWDZAJACY CZY CZY LICZBA OSOB NA WARSZTAT NIE PRZEKRACZA LICZBY OSOB NA DZIEN KONFERENCJI

CREATE TRIGGER [dbo].[nieMożnaRezerwowaćnaWarsztaty]

ON [dbo].[Rezerwacja\_Warsztatu]

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @RezerwacjaKonfId INTEGER = (SELECT Rezerwacja\_Dnia\_Konf\_Rezerwacja\_id FROM INSERTED)

DECLARE @liczbaOsDzien INTEGER = (SELECT rdk.Ilość\_osób FROM Rezerwacja\_Dnia\_Konf as rdk WHERE (@RezerwacjaKonfId = rdk.Rezerwacja\_id))

DECLARE @liczbaOsWar INTEGER = (SELECT Ilość\_osób FROM INSERTED)

if(@liczbaOsWar > @liczbaOsDzien)

BEGIN

RAISERROR ('Nie można zarezerwoać więcej miejsc na warsztat niż na dany dzień konferencji', 16, -1);

ROLLBACK TRANSACTION

END

END

**nieMoznaRejestrowacNaWarsztatyRownoczesne** - uczestnik nie powinien rejestrowac sie na 2 warsztaty trwajace w tym samym czasie.

CREATE TRIGGER [dbo].[nieMoznaRejestrowacNaWarsztatyRownoczesne]

ON [dbo].[Zarejestrowani\_na\_warsztat]

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @isNull INTEGER = (SELECT Zarejestrowani\_uczestnicy\_Uczestnik\_Dnia

FROM INSERTED

JOIN Rezerwacja\_Warsztatu as rw1 ON rw1.id\_Rezerwacja\_Warsztatu =INSERTED.Rezerwacja\_Warsztatu\_id\_Rezerwacja\_Warsztatu

JOIN Warsztaty as war ON rw1.Warsztaty\_Warsztat\_id = war.Warsztat\_id

WHERE Zarejestrowani\_uczestnicy\_Uczestnik\_Dnia IN (SELECT Zarejestrowani\_uczestnicy\_Uczestnik\_Dnia

FROM Zarejestrowani\_na\_warsztat AS znw JOIN Rezerwacja\_Warsztatu as rw2 ON znw.Rezerwacja\_Warsztatu\_id\_Rezerwacja\_Warsztatu = rw2.id\_Rezerwacja\_Warsztatu

JOIN Warsztaty as war2 ON rw2.Warsztaty\_Warsztat\_id = war2.Warsztat\_id

WHERE (Zarejestrowani\_uczestnicy\_Uczestnik\_Dnia = znw.Zarejestrowani\_uczestnicy\_Uczestnik\_Dnia

and rw2.Warsztaty\_Warsztat\_id <> rw1.Warsztaty\_Warsztat\_id and (

(war.Godzina\_rozpoczęcia < war2.Godzina\_zakończenia and war.Godzina\_zakończenia > war2.Godzina\_rozpoczęcia) OR (war.Godzina\_rozpoczęcia < war2.Godzina\_rozpoczęcia and war.Godzina\_zakończenia > war2.Godzina\_zakończenia) OR (war.Godzina\_rozpoczęcia < war2.Godzina\_rozpoczęcia and war.Godzina\_zakończenia < war2.Godzina\_zakończenia) OR

(war.Godzina\_rozpoczęcia > war2.Godzina\_rozpoczęcia and war.Godzina\_zakończenia < war2.Godzina\_zakończenia)

))))

IF(@isNull is not NULL or @isNull != 0)

BEGIN

RAISERROR ('Nie można zarejestrowac tego uczestnika, gdyż on już jest zarejestrowany na warsztat odbywający się w tym samym czasie', 16, -1);

ROLLBACK TRANSACTION

END

END

**godzinaRozpMniejszaOdGodzinyZak** - trigger sprawdzający czy godzina rozpoczęcia warsztatu jest przed godziną zakończenia oraZ czy warsztat trwa przynajmniej 30 minut

CREATE TRIGGER [dbo].[godzinaRozpMniejszaOdGodzinyZak]

ON [dbo].[Warsztaty]

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @rozp TIME = (SELECT Godzina\_rozpoczęcia FROM INSERTED)

DECLARE @zak TIME = (SELECT Godzina\_zakończenia FROM INSERTED)

IF(DATEDIFF(MINUTE, @ROZP, @ZAK) < 30)

BEGIN

RAISERROR ('Godzina rozpoczęcia musi byc mniejsza od godziny zakończenia oraz warsztat powinien

trwać przynajmniej 30 minut.', 16, -1);

ROLLBACK TRANSACTION

END

END

**6.Widoki**

Najdrozszy\_Warsztat- pokazuje najdrozszy wyklad

Create view Najdrozszy\_Warsztat as

Select Top 1 \* from Warsztaty

order by Cena

Najdrozsza\_Konferencja-pokazuje najdrozsza konferencje

Create view Najdrozsza\_Konferencja as

Select Top 1 \* from Koszt

order by Cena desc

Napopularniejszy\_Dzien-pokazuje najpopularniejszy warsztat

Create view Napopularniejszy\_Dzien as

Select TOP 1 \* from Rezerwacja\_Warsztatu

order by Ilość\_osób desc

Najblizsza\_Konf-pokazuje najblizsza konferencje

Create view Najblizsza\_Konf as

select TOP 1 \* from Konferencja

where Data\_rozpoczęcia>GETDATE()

order by Data\_rozpoczęcia

Anulowane\_Warsztaty-pokazuje anulowane warsztaty

Create view Anulowane\_Warsztaty as

Select \* from Rezerwacja\_Warsztatu

where Anulowano=1

Anulowane\_Dni-pokazuje anulowane dni konferencje

Create view Anulowane\_Dni as

Select \* from Rezerwacja\_Dnia\_Konf

where Anulowano=1

Najdluzsza\_konf-pokazuje najdluzsza konferencje

Create view Najdluzsza\_konf as

select top 1 \* from Konferencja

order by Ilość\_dni desc

Uczestnik\_nieStudent-pokazuje uczestnikow ,którzy nie sa studentami

Create view Uczestnik\_nieStudent as

Select \* from Uczestnik

where Student=0

Uczestnik\_Student-pokazuje studentow

Create view Uczestnik\_Student as

Select \* from Uczestnik

where Student=1

Klient\_Firma-pokazuje Firme

Create view Klient\_Firma as

Select \* from Klient

where Firma=1

Klient\_nieFirma-pokazuje klientów którzy nie są firmami

Create view Klient\_nieFirma as

Select \* from Klient

where Firma=0

Nie\_Oplacone-pokazuje wszystkie nie oplacone konferencje

Create view Nie\_Oplacone as

Select \* from Rezerwacja\_Dnia\_Konf

where Opłacone\_id\_Opłacone is null

GO

**7.Generator**

Uzytkownik uzytkownik = new Uzytkownik(zapis);

uzytkownik.generuj();

for (int i = 1; i < 73; i++) {

Konferencje konf = new Konferencje(i, new Date(data \* 1000), zapis);

ileDni = konf.generuj();

for (int j = 1; j < ileDni + 1; j++) {

List<Integer> lista = new ArrayList<>();

iloscWarsztatow = generator.nextInt(3) + 2;

Koszt koszt = new Koszt(dzienKonf, zapis);

kwota = koszt.generuj();

Dzień dzien = new Dzień(i, 200, j, zapis, dzienKonf);

dzienKonf = dzien.generuj();

Oplacone oplacone = new Oplacone(rezerwacjanr, kwota, new Date(

data \* 1000), zapis);

oplacone.generuj();

Rezerwacja\_dzien rezerwacja\_dzien = new Rezerwacja\_dzien(dzienKonf-1,

data, zapis, rezerwacjanr);

datarezerwaci = rezerwacja\_dzien.generuj();

rezerwacjanr = rezerwacja\_dzien.generujnr();

Zarejestrowani\_uzytkownicy zarejestrowani = new Zarejestrowani\_uzytkownicy(

(rezerwacjanr-1), datarezerwaci, UczestnikDnia, zapis);

lista = zarejestrowani.generuj();

UczestnikDnia = UczestnikDnia + lista.size();

for (int k = 1; k < iloscWarsztatow + 1; k++) {

Warsztaty warsztaty = new Warsztaty(dzienKonf-1, Warsztatyid,

zapis);

Warsztatyid = warsztaty.generuj();

Rezerwacja\_warsztatu rezerwacja\_warsztatu = new Rezerwacja\_warsztatu(

Warsztatyid-1, datarezerwaci, (rezerwacjanr-1), zapis);

ileOsob = rezerwacja\_warsztatu.generuj();

for (int l = 1; l < ileOsob; l++) {

Zarejestrowani\_warsztat zare = new Zarejestrowani\_warsztat(

Warsztatyid-1, lista, datarezerwaci,

UczestnikWarsztatu, zapis);

UczestnikWarsztatu = zare.generator();

}

}

}

dni = generator.nextInt(5) + 12;

data = data + dni \* 24 \* 3600;

}

Generator zostal wygenerowany w jezyku java.

**8.Role**

-administrator-posiada pełny dotęp do bazy

-obsluga-dostęp do widoków i procedur informacyjnych

-Klient-może dodac Konferencje/warsztat oraz anulować je

-Uczestnik-może się dodac na dana konferencje /warsztat