Akademia Górniczo-Hutnicza
Wydział Informatyki, Elektroniki i Telekomunikacji

AGH

Podstawy Baz Danych
2016/2017

Temat projektu:

„Konferencje”

Autorzy:
Kinga Kaczmarczyk
Filip Wojas

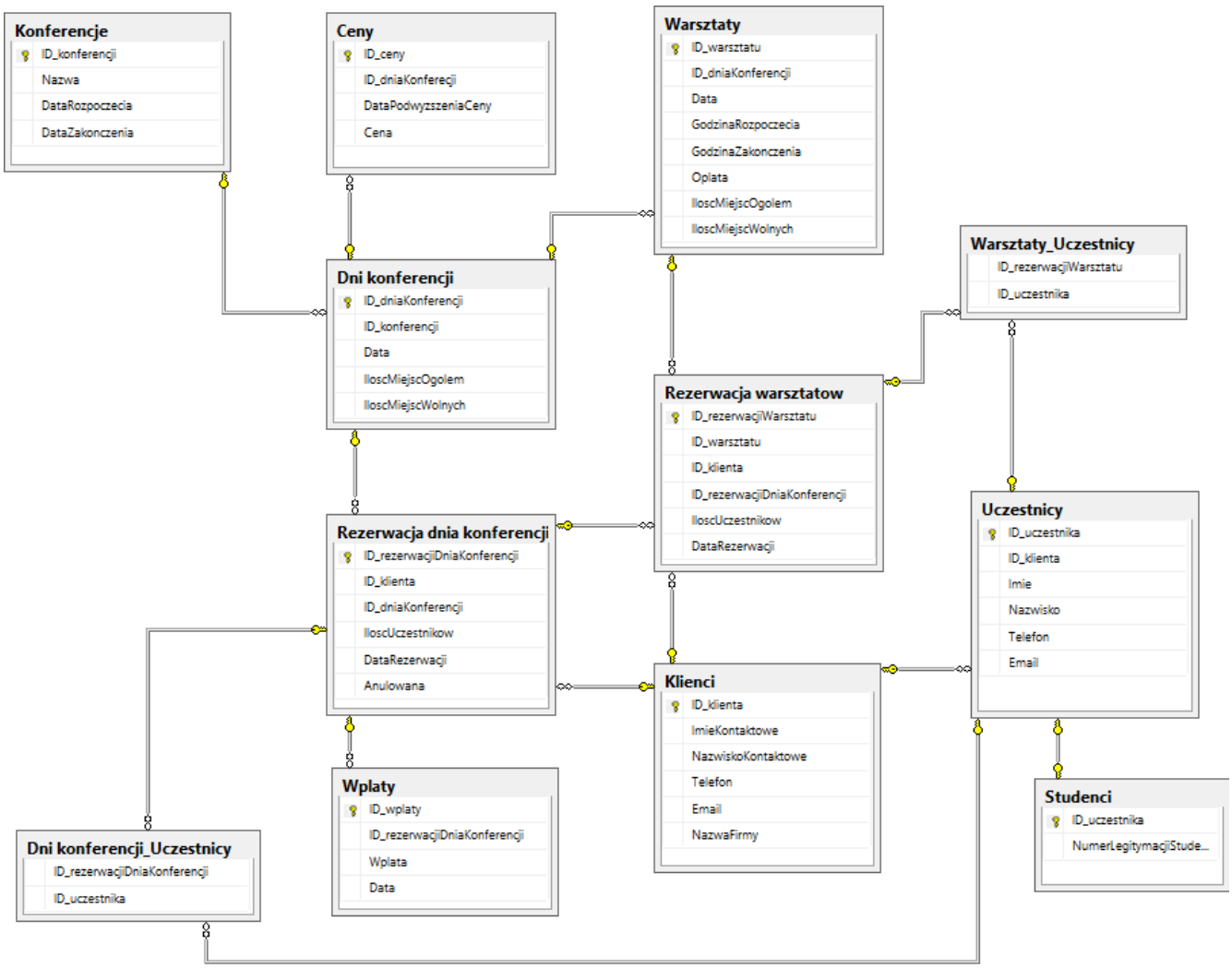
Spis treści

1. Opis projektu 2
2. Diagram Bazy Danych 2
3. Tabele 3
4. Procedury 13
5. Funkcje 21
6. Widoki 25
7. Triggery 26
8. Role 28
9. Generator 29

1. Opis projektu

W ramach projektu została zrealizowana baza danych dla firmy zajmującej się organizacją konferencji i warsztatów. Każda konferencja składa się z kilku dni, podczas których odbywają się warsztaty. W bazie danych przechowywane są między innymi informacje o rezerwacjach, dane klientów oraz zgłoszeni uczestnicy.

2. Diagram Bazy Danych



3. Tabele

- Ceny - przechowuje dane o progach cenowych. Jest połączona relacją jeden do wielu z tabelą Dni Konferencji. Dzięki tej tabeli jesteśmy w stanie dowiedzieć się jaki próg cenowy obowiązuje do jakiego terminu na dany dzień konferencji. ID_ceny to klucz główny opisujący konkretny próg cenowy, ID_dniaKonferencji to klucz obcy, dzięki niemu jesteśmy w stanie dowiedzieć się na jaki dzień konferencji obowiązuje dany próg cenowy, DataPodwyzszeniaCeny to data ostatniego dnia, kiedy próg cenowy jest jeszcze ważny, później przestaje obowiązywać dana cena tylko wyższa, Cena to cena za jedno miejsce w danych dniu konferencji, nie może być ona ujemna.

```
CREATE TABLE [dbo].[Ceny](
	[ID_ceny] [int] IDENTITY(1,1) NOT NULL,
	[ID_dniaKonferecji] [int] NOT NULL,
	[DataPodwyzszeniaCeny] [date] NOT NULL,
	[Cena] [money] NOT NULL,
 CONSTRAINT [PK_Ceny] PRIMARY KEY CLUSTERED
(
	[ID_ceny] 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].[Ceny]  WITH CHECK ADD  CONSTRAINT [FK_Ceny_Dni
konferencji] FOREIGN KEY([ID_dniaKonferecji])
REFERENCES [dbo].[Dni konferencji] ([ID_dniaKonferencji])
GO

ALTER TABLE [dbo].[Ceny] CHECK CONSTRAINT [FK_Ceny_Dni konferencji]
GO

ALTER TABLE [dbo].[Ceny]  WITH CHECK ADD  CONSTRAINT [CK_Ceny_Cena] CHECK
(([Cena]>=(0)))
GO

ALTER TABLE [dbo].[Ceny] CHECK CONSTRAINT [CK_Ceny_Cena]
GO
```

- Dni konferencji – tabela zawiera szczegóły na temat konkretnego dnia danej konferencji. Posiada jako klucz główny własne, auto-inkrementowane ID_dniaKonferencji, klucz obcy ID_konferencji, Data oraz IloscMiejscOgolem i IloscMiejscWolnych, które są większe od 0.

```
CREATE TABLE [dbo].[Dni konferencji](
	[ID_dniaKonferencji] [int] IDENTITY(1,1) NOT NULL,
	[ID_konferencji] [int] NOT NULL,
	[Data] [date] NOT NULL,
	[IloscMiejscOgolem] [int] NOT NULL,
	[IloscMiejscWolnych] [int] NOT NULL,
 CONSTRAINT [PK_Dni konferencji] PRIMARY KEY CLUSTERED
(
	[ID_dniaKonferencji] 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].[Dni konferencji]  WITH CHECK ADD  CONSTRAINT [FK_Dni
konferencji_Konferencje] FOREIGN KEY([ID_konferencji])
REFERENCES [dbo].[Konferencje] ([ID_konferencji])
GO

ALTER TABLE [dbo].[Dni konferencji] CHECK CONSTRAINT [FK_Dni
konferencji_Konferencje]
GO

ALTER TABLE [dbo].[Dni konferencji]  WITH CHECK ADD  CONSTRAINT [CK_Dni
konferencji_IloscMiejscOgolem] CHECK  (([IloscMiejscOgolem]>(0)))
GO

ALTER TABLE [dbo].[Dni konferencji] CHECK CONSTRAINT [CK_Dni
konferencji_IloscMiejscOgolem]
GO

ALTER TABLE [dbo].[Dni konferencji]  WITH CHECK ADD  CONSTRAINT [CK_Dni
konferencji_IloscMiejscWolnych] CHECK  (([IloscMiejscWolnych]>(0)))
GO

ALTER TABLE [dbo].[Dni konferencji] CHECK CONSTRAINT [CK_Dni
konferencji_IloscMiejscWolnych]
GO

ALTER TABLE [dbo].[Dni konferencji]  WITH CHECK ADD  CONSTRAINT [CK_Miejsca]
CHECK  (([IloscMiejscWolnych]<=[IloscMiejscOgolem]))
GO

ALTER TABLE [dbo].[Dni konferencji] CHECK CONSTRAINT [CK_Miejsca]
GO
```

- Dni konferencji_Uczestnicy – Tabela zawiera listę uczestników dnia konferencji. Łączonym kluczem głównych jest tutaj ID_uczestnika i

ID_rezerwacjiDniaKonferencji.

```
CREATE TABLE [dbo].[Dni konferencji_Uczestnicy](
	[ID_rezerwacjiDniaKonferencji] [int] NOT NULL,
	[ID_uczestnika] [int] NOT NULL
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Dni konferencji_Uczestnicy]  WITH CHECK ADD  CONSTRAINT
[FK_Dni konferencji_Uczestnicy_Rezerwacja dnia konferencji] FOREIGN
KEY([ID_rezerwacjiDniaKonferencji])
REFERENCES [dbo].[Rezerwacja dnia konferencji]
([ID_rezerwacjiDniaKonferencji])
GO

ALTER TABLE [dbo].[Dni konferencji_Uczestnicy] CHECK CONSTRAINT [FK_Dni
konferencji_Uczestnicy_Rezerwacja dnia konferencji]
GO

ALTER TABLE [dbo].[Dni konferencji_Uczestnicy]  WITH CHECK ADD  CONSTRAINT
[FK_Dni konferencji_Uczestnicy_Uczestnicy] FOREIGN KEY([ID_uczestnika])
REFERENCES [dbo].[Uczestnicy] ([ID_uczestnika])
GO

ALTER TABLE [dbo].[Dni konferencji_Uczestnicy] CHECK CONSTRAINT [FK_Dni
konferencji_Uczestnicy_Uczestnicy]
GO
```

- Klienci – tabela zawiera informacje na temat wszystkich firm (lub prywatnych osób), które chcą zgłosić uczestników na konferencję. Kluczem głównym, z autoinkrementacją jest ID_klienta. Dane szczegółowe to: ImięKontaktowe, NazwiskoKontaktowe, Telefon (9 cyfr), Email (w formie % @ %. %) oraz ewentualnie NazwaFirmy (jeśli klientem jest osoba prywatna NazwaFirmy zostaje null).

```
CREATE TABLE [dbo].[Klienci](
	[ID_klienta] [int] IDENTITY(1,1) NOT NULL,
	[ImieKontaktowe] [varchar](255) NOT NULL,
	[NazwiskoKontaktowe] [varchar](255) NOT NULL,
	[Telefon] [int] NOT NULL,
	[Email] [varchar](255) NOT NULL,
	[NazwaFirmy] [varchar](255) NULL,
 CONSTRAINT [PK_Klienci] PRIMARY KEY CLUSTERED
(
	[ID_klienta] 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

SET ANSI_PADDING OFF
GO

ALTER TABLE [dbo].[Klienci]  WITH CHECK ADD  CONSTRAINT [CK_Klienci_Email]
CHECK  (([Email] like '%_@_%._%'))
GO

ALTER TABLE [dbo].[Klienci] CHECK CONSTRAINT [CK_Klienci_Email]
GO

ALTER TABLE [dbo].[Klienci]  WITH CHECK ADD  CONSTRAINT [CK_Klienci_Telefon]
CHECK  (([Telefon] like '[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'))
GO

ALTER TABLE [dbo].[Klienci] CHECK CONSTRAINT [CK_Klienci_Telefon]
GO
```

- Konferencje – tabela przechowuje informacje na temat poszczególnych konferencji. Zawiera ID_konferencji z autoinkrementacja będące kluczem głównym, Nazwa, DataRozpoczecia i DataZakonczenia (przy czym warunek jest by zaczynała się zanim się skończy).

```
CREATE TABLE [dbo].[Konferencje](
	[ID_konferencji] [int] IDENTITY(1,1) NOT NULL,
	[Nazwa] [varchar](255) NOT NULL,
	[DataRozpoczecia] [date] NOT NULL,
	[DataZakonczenia] [date] NOT NULL,
 CONSTRAINT [PK_Konferencje] PRIMARY KEY CLUSTERED
(
	[ID_konferencji] 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

SET ANSI_PADDING OFF
GO

ALTER TABLE [dbo].[Konferencje]  WITH CHECK ADD  CONSTRAINT
[CK_Konferencje_Daty] CHECK  (([DataRozpoczecia]<=[DataZakonczenia]))
GO

ALTER TABLE [dbo].[Konferencje] CHECK CONSTRAINT [CK_Konferencje_Daty]
GO
```

- Rezerwacja dnia konferencji – tabela zawiera informacje na temat rezerwacji dnia konferencji złożonych jednorazowo przez danego klienta. Zawiera autoinkrementowany klucz główny ID_rezerwacjiDniaKonferencji, ID_klienta i ID_dniaKonferencji jako klucze obce, IloscUczestnikow (ma być ona większa od 0), DataRezerwacji oraz bitowa wartość Anulowana (1 jest rezerwacja, 0 jest anulowana).

```
CREATE TABLE [dbo].[Rezerwacja dnia konferencji](
	[ID_rezerwacjiDniaKonferencji] [int] IDENTITY(1,1) NOT NULL,
	[ID_klienta] [int] NOT NULL,
	[ID_dniaKonferencji] [int] NOT NULL,
	[IloscUczestnikow] [int] NOT NULL,
	[DataRezerwacji] [date] NOT NULL,
	[Anulowana] [bit] NOT NULL,
 CONSTRAINT [PK_Rezerwacja dnia konferencji] PRIMARY KEY CLUSTERED
(
	[ID_rezerwacjiDniaKonferencji] 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].[Rezerwacja dnia konferencji] ADD  CONSTRAINT
[DF_Rezerwacja dnia konferencji_Anulowana]  DEFAULT ((0)) FOR [Anulowana]
GO

ALTER TABLE [dbo].[Rezerwacja dnia konferencji]  WITH CHECK ADD  CONSTRAINT
[FK_Rezerwacja dnia konferencji_Dni konferencji] FOREIGN
KEY([ID_dniaKonferencji])
REFERENCES [dbo].[Dni konferencji] ([ID_dniaKonferencji])
GO

ALTER TABLE [dbo].[Rezerwacja dnia konferencji] CHECK CONSTRAINT
[FK_Rezerwacja dnia konferencji_Dni konferencji]
GO

ALTER TABLE [dbo].[Rezerwacja dnia konferencji]  WITH CHECK ADD  CONSTRAINT
[FK_Rezerwacja dnia konferencji_Klienci] FOREIGN KEY([ID_klienta])
REFERENCES [dbo].[Klienci] ([ID_klienta])
GO

ALTER TABLE [dbo].[Rezerwacja dnia konferencji] CHECK CONSTRAINT
[FK_Rezerwacja dnia konferencji_Klienci]
GO

ALTER TABLE [dbo].[Rezerwacja dnia konferencji]  WITH CHECK ADD  CONSTRAINT
[CK_Rezerwacja dnia konferencji_IloscUczestnikow] CHECK
(([IloscUczestnikow]>=(0)))
```

```
GO

ALTER TABLE [dbo].[Rezerwacja dnia konferencji] CHECK CONSTRAINT
[CK_Rezerwacja dnia konferencji_IloscUczestnikow]
GO
```

- Rezerwacja warsztatów - tabela zawiera informacje na temat rezerwacji warsztatów złożonych jednorazowo przez danego klienta. Zawiera autoinkrementowany klucz główny ID_rezerwacjiWarsztatu, ID_klienta, ID_warsztatu i ID_rezerwacjiDniaKonferencji jako klucze obce, IloscUczestnikow (ma być ona większa od 0).

```
CREATE TABLE [dbo].[Rezerwacja warsztatow](
	[ID_rezerwacjiWarsztatu] [int] IDENTITY(1,1) NOT NULL,
	[ID_warsztatu] [int] NOT NULL,
	[ID_klienta] [int] NOT NULL,
	[ID_rezerwacjiDniaKonferencji] [int] NOT NULL,
	[IloscUczestnikow] [int] NOT NULL,
	[DataRezerwacji] [date] NOT NULL,
 CONSTRAINT [PK_Rezerwacja warsztatow] PRIMARY KEY CLUSTERED
(
	[ID_rezerwacjiWarsztatu] 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].[Rezerwacja warsztatow]  WITH CHECK ADD  CONSTRAINT
[FK_Rezerwacja warsztatow_Klienci] FOREIGN KEY([ID_klienta])
REFERENCES [dbo].[Klienci] ([ID_klienta])
GO

ALTER TABLE [dbo].[Rezerwacja warsztatow] CHECK CONSTRAINT [FK_Rezerwacja
warsztatow_Klienci]
GO

ALTER TABLE [dbo].[Rezerwacja warsztatow]  WITH CHECK ADD  CONSTRAINT
[FK_Rezerwacja warsztatow_Rezerwacja dnia konferencji] FOREIGN
KEY([ID_rezerwacjiDniaKonferencji])
REFERENCES [dbo].[Rezerwacja dnia konferencji]
([ID_rezerwacjiDniaKonferencji])
GO

ALTER TABLE [dbo].[Rezerwacja warsztatow] CHECK CONSTRAINT [FK_Rezerwacja
warsztatow_Rezerwacja dnia konferencji]
GO

ALTER TABLE [dbo].[Rezerwacja warsztatow]  WITH CHECK ADD  CONSTRAINT
[FK_Rezerwacja warsztatow_Warsztaty] FOREIGN KEY([ID_warsztatu])
```

```
REFERENCES [dbo].[Warsztaty] ([ID_warsztatu])
GO

ALTER TABLE [dbo].[Rezerwacja warsztatow] CHECK CONSTRAINT [FK_Rezerwacja
warsztatow_Warsztaty]
GO

ALTER TABLE [dbo].[Rezerwacja warsztatow]  WITH CHECK ADD  CONSTRAINT
[CK_Rezerwacja warsztatow_IloscUczestnikow] CHECK
(([IloscUczestnikow]>(0)))
GO

ALTER TABLE [dbo].[Rezerwacja warsztatow] CHECK CONSTRAINT [CK_Rezerwacja
warsztatow_IloscUczestnikow]
GO
```

- Studenci – tabela zawiera informacje na temat uczestników będących studentami. W tabeli można znaleźć ID_uczestnika, czyli klucz główny oraz NumerLegitymacji.

```
CREATE TABLE [dbo].[Studenci](
	[ID_uczestnika] [int] NOT NULL,
	[NumerLegitymacjiStudenckiej] [varchar](255) NOT NULL,
 CONSTRAINT [PK_Studenci] PRIMARY KEY CLUSTERED
(
	[ID_uczestnika] 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

SET ANSI_PADDING OFF
GO

ALTER TABLE [dbo].[Studenci]  WITH CHECK ADD  CONSTRAINT
[FK_Studenci_Uczestnicy] FOREIGN KEY([ID_uczestnika])
REFERENCES [dbo].[Uczestnicy] ([ID_uczestnika])
GO

ALTER TABLE [dbo].[Studenci] CHECK CONSTRAINT [FK_Studenci_Uczestnicy]
GO
```

- Uczestnicy – tabela zawiera informacje na temat wszystkich uczestników konferencji i warsztatów. Każda osoba ma swoje własne, autoinkrementowane ID_uczestnika, klucz obcy ID_klienta, Imie, Nazwisko, Telefon (9 cyfr) oraz Email (musi być w formie %_@_%._%).

```
CREATE TABLE [dbo].[Uczestnicy](
	[ID_uczestnika] [int] IDENTITY(1,1) NOT NULL,
	[ID_klienta] [int] NOT NULL,
	[Imie] [varchar](255) NOT NULL,
	[Nazwisko] [varchar](255) NOT NULL,
	[Telefon] [int] NOT NULL,
	[Email] [varchar](255) NOT NULL,
 CONSTRAINT [PK_Uczestnicy] PRIMARY KEY CLUSTERED
(
	[ID_uczestnika] 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

SET ANSI_PADDING OFF
GO

ALTER TABLE [dbo].[Uczestnicy]  WITH CHECK ADD  CONSTRAINT
[FK_Uczestnicy_Klienci] FOREIGN KEY([ID_klienta])
REFERENCES [dbo].[Klienci] ([ID_klienta])
GO

ALTER TABLE [dbo].[Uczestnicy] CHECK CONSTRAINT [FK_Uczestnicy_Klienci]
GO

ALTER TABLE [dbo].[Uczestnicy]  WITH CHECK ADD  CONSTRAINT
[CK_Uczestnicy_Email] CHECK  (([Email] like '%_@_%._%'))
GO

ALTER TABLE [dbo].[Uczestnicy] CHECK CONSTRAINT [CK_Uczestnicy_Email]
GO

ALTER TABLE [dbo].[Uczestnicy]  WITH CHECK ADD  CONSTRAINT
[CK_Uczestnicy_Telefon] CHECK  (([Telefon] like '[0-9][0-9][0-9][0-9][0-
9][0-9][0-9][0-9][0-9]'))
GO

ALTER TABLE [dbo].[Uczestnicy] CHECK CONSTRAINT [CK_Uczestnicy_Telefon]
GO
```

- Warsztaty – tabela zawiera informacje na temat wszystkich warsztatów. klucz główny, autoinkrementowany to ID_warsztatu. Ponadto klucz obcy to ID_dniaKonferencji, Data, GodzinaRozpoczecia i GodzinaZakonczenia (gdzie ma się wcześniej zacząć niż zakończyć), Oplata (większa od 0) oraz IloscMiejscOgolem i IloscMiejscWolnych.

```
CREATE TABLE [dbo].[Warsztaty](
	[ID_warsztatu] [int] IDENTITY(1,1) NOT NULL,
```

```
	[ID_dniaKonferencji] [int] NOT NULL,
	[Data] [date] NOT NULL,
	[GodzinaRozpoczecia] [time](7) NOT NULL,
	[GodzinaZakonczenia] [time](7) NOT NULL,
	[Oplata] [money] NULL,
	[IloscMiejscOgolem] [int] NOT NULL,
	[IloscMiejscWolnych] [int] NOT NULL,
 CONSTRAINT [PK_Warsztaty] PRIMARY KEY CLUSTERED
(
	[ID_warsztatu] 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].[Warsztaty]  WITH CHECK ADD  CONSTRAINT [FK_Warsztaty_Dni
konferencji] FOREIGN KEY([ID_dniaKonferencji])
REFERENCES [dbo].[Dni konferencji] ([ID_dniaKonferencji])
GO

ALTER TABLE [dbo].[Warsztaty] CHECK CONSTRAINT [FK_Warsztaty_Dni
konferencji]
GO

ALTER TABLE [dbo].[Warsztaty]  WITH CHECK ADD  CONSTRAINT
[CK_Warsztaty_Godzina] CHECK  (([GodzinaRozpoczecia]<=[GodzinaZakonczenia]))
GO

ALTER TABLE [dbo].[Warsztaty] CHECK CONSTRAINT [CK_Warsztaty_Godzina]
GO

ALTER TABLE [dbo].[Warsztaty]  WITH CHECK ADD  CONSTRAINT
[CK_Warsztaty_IloscMiejscOgolem] CHECK  (([IloscMiejscOgolem]>(0)))
GO

ALTER TABLE [dbo].[Warsztaty] CHECK CONSTRAINT
[CK_Warsztaty_IloscMiejscOgolem]
GO

ALTER TABLE [dbo].[Warsztaty]  WITH CHECK ADD  CONSTRAINT
[CK_Warsztaty_IloscMiejscWolnych] CHECK  (([IloscMiejscWolnych]>(0)))
GO

ALTER TABLE [dbo].[Warsztaty] CHECK CONSTRAINT
[CK_Warsztaty_IloscMiejscWolnych]
GO

ALTER TABLE [dbo].[Warsztaty]  WITH CHECK ADD  CONSTRAINT
[CK_Warsztaty_Miejsca] CHECK  (([IloscMiejscWolnych]<=[IloscMiejscOgolem]))
GO
```

```
ALTER TABLE [dbo].[Warsztaty] CHECK CONSTRAINT [CK_Warsztaty_Miejsca]
GO

ALTER TABLE [dbo].[Warsztaty]  WITH CHECK ADD  CONSTRAINT
[CK_Warsztaty_Oplata] CHECK  (([Oplata]>=(0)))
GO

ALTER TABLE [dbo].[Warsztaty] CHECK CONSTRAINT [CK_Warsztaty_Oplata]
GO
```

- Warsztaty_Uczestnicy - Tabela zawiera listę uczestników warsztatów. Łączonym kluczem głównych jest tutaj ID_uczestnika i ID_rezerwacjiWarsztatu.

```
CREATE TABLE [dbo].[Warsztaty_Uczestnicy](
       [ID_rezerwacjiWarsztatu] [int] NOT NULL,
       [ID_uczestnika] [int] NOT NULL
) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Warsztaty_Uczestnicy]  WITH CHECK ADD  CONSTRAINT
[FK_Warsztaty_Uczestnicy_Rezerwacja warsztatow] FOREIGN
KEY([ID_rezerwacjiWarsztatu])
REFERENCES [dbo].[Rezerwacja warsztatow] ([ID_rezerwacjiWarsztatu])
GO

ALTER TABLE [dbo].[Warsztaty_Uczestnicy] CHECK CONSTRAINT
[FK_Warsztaty_Uczestnicy_Rezerwacja warsztatow]
GO

ALTER TABLE [dbo].[Warsztaty_Uczestnicy]  WITH CHECK ADD  CONSTRAINT
[FK_Warsztaty_Uczestnicy_Uczestnicy] FOREIGN KEY([ID_uczestnika])
REFERENCES [dbo].[Uczestnicy] ([ID_uczestnika])
GO

ALTER TABLE [dbo].[Warsztaty_Uczestnicy] CHECK CONSTRAINT
[FK_Warsztaty_Uczestnicy_Uczestnicy]
GO
```

- Wpłaty – tabela zawiera listę wpłat. ID_wpalaty to główny klucz, autoinkrementowany. Posiada także klucz obcy ID_rezerwacjiDniaKonferencji, Wplata (zawsze większa od 0) oraz Data zapłacenia.

```
CREATE TABLE [dbo].[Wplaty](
       [ID_wplaty] [int] IDENTITY(1,1) NOT NULL,
```

```
	[ID_rezerwacjiDniaKonferencji] [int] NOT NULL,
	[Wplata] [money] NOT NULL,
	[Data] [date] NOT NULL,
 CONSTRAINT [PK_Wplaty] PRIMARY KEY CLUSTERED
(
	[ID_wplaty] 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].[Wplaty]  WITH CHECK ADD  CONSTRAINT [FK_Wplaty_Rezerwacja
dnia konferencji] FOREIGN KEY([ID_rezerwacjiDniaKonferencji])
REFERENCES [dbo].[Rezerwacja dnia konferencji]
([ID_rezerwacjiDniaKonferencji])
GO

ALTER TABLE [dbo].[Wplaty] CHECK CONSTRAINT [FK_Wplaty_Rezerwacja dnia
konferencji]
GO

ALTER TABLE [dbo].[Wplaty]  WITH CHECK ADD  CONSTRAINT [CK_Wplaty_Wplata]
CHECK  (([Wplata]>(0)))
GO

ALTER TABLE [dbo].[Wplaty] CHECK CONSTRAINT [CK_Wplaty_Wplata]
GO
```

4. Procedury

- AnulujRezerwacje – anulowanie rezerwacji np. gdy chce tego klient. Dostaje ID_rezerwacjiDniaKonferencji i zmienia Anulowana na 0.

```
CREATE PROCEDURE [dbo].[AnulujRezerwacje]

	@param int -- idrezerwacji

AS
BEGIN

	update [dbo].[Rezerwacja dnia konferencji]
	set [dbo].[rezerwacja dnia konferencji].Anulowana=0
	where [dbo].[Rezerwacja dnia
konferencji].ID_rezerwacjiDniaKonferencji=@param

END
```

```
GO
```

- AnulujRezerwacje – anuluje rezerwacje, gdy minął czas na opłacenie jej.

```
CREATE PROCEDURE [dbo].[AnulujRezerwacjeNieZaplacono]


AS
BEGIN

	SET NOCOUNT ON;

	declare curs cursor local for (select [Rezerwacja dnia
konferencji].ID_rezerwacjiDniaKonferencji, [Rezerwacja dnia
konferencji].DataRezerwacji, Ceny.Cena
from
[Rezerwacja dnia konferencji]
inner
join [Dni konferencji] on [Dni konferencji].ID_dniaKonferencji=[Rezerwacja
dnia konferencji].ID_dniaKonferencji
inner
join Ceny on Ceny.ID_dniaKonferecji=[Dni konferencji].ID_dniaKonferencji
left
outer join wplaty on ([Rezerwacja dnia
konferencji].ID_rezerwacjiDniaKonferencji=Wplaty.ID_rezerwacjiDniaKonferencj
i) and (wplaty.wplata < ceny.cena))

	declare @idrezerwacji int, @datarezerwacji date


	open curs
		fetch next from curs into @idrezerwacji, @datarezerwacji
		while @@fetch_status = 0
		begin
			if(datediff(day,@datarezerwacji,getdate()) > 7)
			begin
				exec [dbo].[anulujRezerwacje] @idrezerwacji
			end
			fetch next from curs into @idrezerwacji,
@datarezerwacji
		end
	close curs

END

GO
```

- dodajCene – dodaje cenę i jej progi dla poszczególnych dni konferencji.

```
CREATE PROCEDURE [dbo].[dodajCene]
	@ID_dniaKonferencji int,
	@DataPodwyzszeniaCeny date,
	@Cena money
AS
BEGIN
	SET NOCOUNT ON;
	INSERT INTO Ceny VALUES
(@ID_dniaKonferencji,@DataPodwyzszeniaCeny,@Cena)
END

GO
```

- dodajDzienKonferencji – dodaje dzień konferencji.

```
CREATE PROCEDURE [dbo].[dodajDzienKonferencji]
	@ID_konferencji int,
	@Data date,
	@IloscMiejscOgolem int -- to jest ilosc miejsc ogolem i przy dodawanu
dnia konferencji takze ilosc miejsc wolnych
AS
BEGIN
	SET NOCOUNT ON;

	declare @DataRozpoczecia date
	declare @DataZakonczenia date

	set @DataRozpoczecia = (select DataRozpoczecia from Konferencje where
ID_konferencji=@ID_konferencji)
	set @DataZakonczenia = (select DataZakonczenia from Konferencje where
ID_konferencji=@ID_konferencji)

	if (@DataRozpoczecia > @Data) or (@DataZakonczenia < @Data)
		THROW 50001,'Nie zgadza sie data dnia konferencji',1

	INSERT INTO [Dni konferencji] VALUES
(@ID_konferencji,@Data,@IloscMiejscOgolem,@IloscMiejscOgolem)
END

GO
```

- dodajKlienta – dodaje klienta.

```
CREATE PROCEDURE [dbo].[dodajKlienta]
```

```
	@ImieKontaktowe varchar(255),
	@NazwiskoKontaktowe varchar(255),
	@Telefon int,
	@Email varchar(255),
	@NazwaFirmy varchar(255)
AS
BEGIN
	SET NOCOUNT ON;
	INSERT INTO Klienci
VALUES(@ImieKontaktowe,@NazwiskoKontaktowe,@Telefon,@Email,@NazwaFirmy)
END

GO
```

- dodajKonferencje – dodaje konferencje

```
CREATE PROCEDURE [dbo].[dodajKonferencje]
	@Nazwa varchar(255),
	@DataRozpoczecia date,
	@DataZakonczenia date
AS
BEGIN
	SET NOCOUNT ON;
	INSERT INTO Konferencje VALUES
(@Nazwa,@DataRozpoczecia,@DataZakonczenia)
END

GO
```

- dodajRezerwacjeDniaKonferencji – dodaje rezerwacje dnia konferencji, sprawdza od razu czy tyle uczestników może jeszcze zostać dopisanych.

```
CREATE PROCEDURE [dbo].[dodajRezerwacjeDniaKonferencji]
	@ID_klienta int,
	@ID_dniaKonferencji int,
	@IloscUczestnikow int
AS
BEGIN
	SET NOCOUNT ON;

	-- sprawdzanie czy jest wystarczajaco miejsca w dniu konferencji
	declare @IloscMiejscWolnych int
	declare @IloscMiejscOgolem int

	set @IloscMiejscWolnych =
dbo.podajIloscMiejsc(@ID_dniaKonferencji,'DzienKonferencji','wolnych')
	set @IloscMiejscOgolem =
dbo.podajIloscMiejsc(@ID_dniaKonferencji,'DzienKonferencji','ogolem')
```

```
	if @IloscMiejscWolnych + @IloscUczestnikow > @IloscMiejscOgolem
		THROW 50001,'Nie wystarczajaco miejsc na ten dzien
konferencji',1

	UPDATE [Dni konferencji] set
IloscMiejscWolnych=dbo.podajIloscWolnychMiejscDniaKonferencji(@ID_dniaKonfer
encji) where ID_dniaKonferencji=@ID_dniaKonferencji

	INSERT INTO [Rezerwacja dnia konferencji] VALUES
(@ID_klienta,@ID_dniaKonferencji,@IloscUczestnikow,getdate(),1) -- 1 oznacza
ze nie anulowana
END

GO
```

- dodajRezerwacjeWarsztatu – dodaje rezerwacje warsztatu tak ja powyższa.

```
CREATE PROCEDURE [dbo].[dodajRezerwacjeWarsztatu]
	@ID_warsztatu int,
	@ID_klienta int,
	@ID_rezerwacjiDniaKonferencji int,
	@IloscUczestnikow int
AS
BEGIN
	SET NOCOUNT ON;

	-- sprawdzanie czy jest wystarczajaco miejsca w dniu konferencji
	declare @IloscMiejscWolnych int
	declare @IloscMiejscOgolem int

	set @IloscMiejscWolnych =
dbo.podajIloscMiejsc(@ID_warsztatu,'Warsztat','wolnych')
	set @IloscMiejscOgolem =
dbo.podajIloscMiejsc(@ID_warsztatu,'Warsztat','ogolem')

	if @IloscMiejscWolnych + @IloscUczestnikow > @IloscMiejscOgolem
		THROW 50001,'Nie wystarczajaco miejsc na ten warsztat',1

	UPDATE Warsztaty set
IloscMiejscWolnych=dbo.podajIloscWolnychMiejscWarsztatu(@ID_warsztatu) where
ID_warsztatu=@ID_warsztatu


	INSERT INTO [Rezerwacja warsztatow]
VALUES(@ID_warsztatu,@id_klienta,@id_rezerwacjiDniaKonferencji,@IloscUczestn
ikow,getdate())
END
```

```
GO
```

- dodajStudenta – dodaje studenta.

```
CREATE PROCEDURE [dbo].[dodajStudenta]
	@ID_uczestnika int,
	@NumerLegitymacjiStudenckiej varchar(255)
AS
BEGIN
	SET NOCOUNT ON;
	INSERT INTO Studenci VALUES
(@ID_uczestnika,@NumerLegitymacjiStudenckiej)
END

GO
```

- dodajUczestnika – dodaje uczestnika.

```
CREATE PROCEDURE [dbo].[dodajUczestnika]
	@ID_klienta int,
	@Imie varchar(255),
	@Nazwisko varchar(255),
	@Telefon int,
	@Email varchar(255)
AS
BEGIN
	SET NOCOUNT ON;
	INSERT INTO Uczestnicy VALUES
(@ID_klienta,@Imie,@Nazwisko,@Telefon,@Email)
END

GO
```

- dodajWarsztat – dodaje nowy warsztat.

```
CREATE PROCEDURE [dbo].[dodajWarsztat]
	@ID_dniaKonferencji int,
	@Data date,
	@GodzinaRozpoczecia time(7),
	@GodzinaZakonczenia time(7),
	@Oplata money,
	@IloscMiejscOgolem int
AS
BEGIN
	SET NOCOUNT ON;
```

```
	declare @DataDniaKonferencji date

	set @DataDniaKonferencji = (select Data from [Dni konferencji] where
ID_dniaKonferencji=@ID_dniaKonferencji)

	if @Data <> @DataDniaKonferencji
		THROW 50001,'Nie zgadza sie data warsztatu. Musi byc taka sama
jak data dnia konferencji',1

	INSERT INTO Warsztaty
VALUES(@ID_dniaKonferencji,@Data,@godzinaRozpoczecia,@godzinaZakonczenia,@Op
lata,@IloscMiejscOgolem,@IloscMiejscOgolem)
END


GO
```

- dodajWplate – dodaje nowa wpłate.

```
CREATE PROCEDURE [dbo].[dodajWplate]
	@ID_rezerwacjiDniaKonferencji int,
	@Wplata money
AS
BEGIN
	SET NOCOUNT ON;
	INSERT INTO Wplaty
VALUES(@ID_rezerwacjiDniaKonferencji,@Wplata,getdate())

END

GO
```

- Generuj_identyfikatory – generuje identyfikatory.

```
CREATE PROCEDURE [dbo].[Generuj_identyfikatory]

@param INT
AS
BEGIN
SET NOCOUNT ON;
SELECT dbo.Uczestnicy.Imie, dbo.Uczestnicy.Nazwisko, dbo.Klienci.NazwaFirmy
AS 'Firma'
FROM dbo.[Rezerwacja dnia konferencji] INNER JOIN
dbo.Klienci ON dbo.[Rezerwacja dnia konferencji].ID_klienta =
dbo.Klienci.ID_klienta INNER JOIN
dbo.uczestnicy ON dbo.klienci.ID_klienta = dbo.Uczestnicy.ID_klienta
WHERE dbo.[Rezerwacja dnia konferencji].ID_rezerwacjiDniaKonferencji=@param
AND dbo.[Rezerwacja dnia konferencji].Anulowana IS NULL
```

```
END
GO
```

- przydzielUczestnikaDoDniaKonferencji – łaczy uczestnika z rezerwacja dnia konferencji w tabeli.

```
CREATE PROCEDURE [dbo].[przydzielUczestnikaDoRezerwacjiDniaKonferencji]
	@ID_rezerwacjiDniaKonferencji int,
	@ID_uczestnika int
AS
BEGIN
	SET NOCOUNT ON;
	INSERT INTO [Dni konferencji_Uczestnicy] VALUES
(@ID_rezerwacjiDniaKonferencji, @ID_uczestnika)
END

GO
```

- przydzielUczestnikaDoWarsztatu – łączy uczestnika z rezerwacją warsztatu.

```
CREATE PROCEDURE [dbo].[przydzielUczestnikaDoRezerwacjiWarszatu]
	@ID_rezerwacjiWarsztatu int,
	@ID_uczestnika int
AS
BEGIN
	SET NOCOUNT ON;

	declare @ID_rezerwacjiDniaKonferencji int = ( select
ID_rezerwacjiDniaKonferencji from [Rezerwacja warsztatow] where
ID_rezerwacjiWarsztatu = @ID_rezerwacjiWarsztatu)
	declare @ID int = (select ID_uczestnika from [Dni
konferencji_Uczestnicy] where
(ID_rezerwacjiDniaKonferencji=@ID_rezerwacjiDniaKonferencji) and
(ID_uczestnika=@ID_uczestnika))

	if @ID is null
		throw 52000, 'Uczestnik nie zostal przydzielony do
odpowiedniego dnia konferencji',1

	INSERT INTO Warsztaty_Uczestnicy VALUES
(@ID_rezerwacjiWarsztatu,@ID_uczestnika)
END

GO
```

- rozszerzIloscMiejscDniaKonferencji – zwiększa ilość miejsc ogółem dla danego dnia konferencji.

```
CREATE PROCEDURE [dbo].[rozszerzIloscMiejscDniaKonferencji]
	@ID_dniaKonferencji int,
	@IloscMiejscOgolem int
AS
BEGIN
	SET NOCOUNT ON;

	if(@IloscMiejscOgolem <= (select IloscMiejscOgolem from [Dni
konferencji] where ID_dniaKonferencji=@ID_dniaKonferencji))
		throw 52000, 'Mozna tylko zwiekszyc ilosc miejsc',1

	UPDATE [Dni konferencji] SET IloscMiejscOgolem=@IloscMiejscOgolem
WHERE ID_dniaKonferencji=@ID_dniaKonferencji
END

GO
```

- rozszerzIloscMiejscWarsztatu – zwiększa ilość miejsc ogółem dla warsztatu.

```
CREATE PROCEDURE [dbo].[rozszerzIloscMiejscWarsztatu]
	@ID_warsztatu int,
	@IloscMiejscOgolem int
AS
BEGIN
	SET NOCOUNT ON;

	if(@IloscMiejscOgolem <= (select IloscMiejscOgolem from Warsztaty
where ID_warsztatu=@ID_warsztatu))
		throw 52000, 'Mozna tylko zwiekszyc ilosc miejsc',1

	UPDATE Warsztaty SET IloscMiejscOgolem=@IloscMiejscOgolem WHERE
@ID_warsztatu=@ID_warsztatu
END

GO
```

5. Funkcje

- podajCeneDniaDlaDaty – wprowadza datę i ID_dniaKonferencji i wyświetla jaką cenę trzeba będzie zapłacić w tym dniu.

```
CREATE FUNCTION [dbo].[podajCeneDniaKonferencjiDlaDaty]
(
	@Data date,
	@ID_dniaKonferencji int
)
RETURNS money
AS
BEGIN
	declare @Cena money = ( select top 1 Cena from Ceny where
(ID_dniaKonferecji=@ID_dniaKonferencji) and
(datediff(day,@Data,DataPodwyzszeniaCeny)>=0) order by DataPodwyzszeniaCeny)

	return @Cena
END

GO
```

- podajIloscMiejsc – podaje ilość miejsc wolnych lub ogółem dla danego ID i rodzaju (dzień konferencji, warsztat).

```
CREATE FUNCTION [dbo].[podajIloscMiejsc]
(
	@ID int,
	@Rodzaj varchar(255),
	@WolneCzyOgolem varchar(255)
)
RETURNS int
AS
BEGIN

	declare @IloscMiejsc int

	if @Rodzaj = 'DzienKonferencji'
		if @WolneCzyOgolem = 'wolne'
			set @IloscMiejsc = (select sum(IloscMiejscWolnych) from
[Dni konferencji] where ID_dniaKonferencji=@ID)
		else
			set @IloscMiejsc = (select sum(IloscMiejscOgolem) from
[Dni konferencji] where ID_dniaKonferencji=@ID)
	else
		if @WolneCzyOgolem = 'wolne'
			set @IloscMiejsc = (select sum(IloscMiejscWolnych) from
Warsztaty where ID_warsztatu=@ID)
		else
			set @IloscMiejsc = (select sum(IloscMiejscOgolem) from
Warsztaty where ID_warsztatu=@ID)

	return @IloscMiejsc
END
```

```
GO
```

- podajIloscWolnychMiejscDniaKonferencji – wyświetla ilość miejsc wolnych w dniu konferencji, ale z szukania a nie z liczby w tabeli.

```
CREATE FUNCTION [dbo].[podajIloscWolnychMiejscDniaKonferencji]
(
	@ID_dnia int
)
RETURNS int
AS
BEGIN

	declare @IloscMiejscOgolem int
	declare @IloscMiejscZajetych int

	set @IloscMiejscOgolem = (select IloscMiejscOgolem from [Dni
konferencji] where ID_dniaKonferencji=@ID_dnia)
	set @IloscMiejscZajetych = (select sum(IloscUczestnikow) from
[Rezerwacja dnia konferencji] where ID_dniaKonferencji=@ID_dnia and
Anulowana = 1)

	if @IloscMiejscZajetych is null
		set @IloscMiejscZajetych =0

	return (@IloscMiejscOgolem - @IloscMiejscZajetych )
END

GO
```

- podajIloscWolnychMiejscWarsztatu - wyświetla ilość miejsc wolnych dla warsztatu, ale z szukania a nie z liczby w tabeli.

```
CREATE FUNCTION [dbo].[podajIloscWolnychMiejscWarsztatu]
(
	@ID_warsztatu int
)
RETURNS int
AS
BEGIN

	declare @IloscMiejscOgolem int
	declare @IloscMiejscZajetych int

	set @IloscMiejscOgolem = (select IloscMiejscOgolem from Warsztaty
where ID_warsztatu=@ID_warsztatu)
	set @IloscMiejscZajetych = (select sum(IloscUczestnikow) from
[Rezerwacja warsztatow] where ID_warsztatu=@ID_warsztatu )
```

```
	if @IloscMiejscZajetych is null
		set @IloscMiejscZajetych =0

	return (@IloscMiejscOgolem - @IloscMiejscZajetych )
END

GO
```

- znajdz_cene – podaje cenę do zapłacenia dla klienta.

```
CREATE FUNCTION [dbo].[znajdz_cene]
(

	@param int
)
RETURNS int
AS
BEGIN

	DECLARE @suma int

	set @suma = isnull((select top 1 ceny.cena
	from [Rezerwacja dnia konferencji]
	inner join [Dni konferencji] on [Dni
konferencji].ID_dniaKonferencji=[Rezerwacja dnia
konferencji].ID_dniaKonferencji
	join ceny on [Dni
konferencji].ID_dniaKonferencji=Ceny.ID_dniaKonferecji
	where [Rezerwacja dnia
konferencji].ID_rezerwacjiDniaKonferencji=@param
	and [Rezerwacja dnia konferencji].DataRezerwacji >
ceny.DataPodwyzszeniaCeny
	order by Ceny.DataPodwyzszeniaCeny desc),
	isnull ((select top 1 Ceny.Cena
	from [Rezerwacja dnia konferencji]
	inner join [Dni konferencji] on [Dni
konferencji].ID_dniaKonferencji=[Rezerwacja dnia
konferencji].ID_dniaKonferencji
	join ceny on [Dni
konferencji].ID_dniaKonferencji=Ceny.ID_dniaKonferecji
	where [Rezerwacja dnia
konferencji].ID_rezerwacjiDniaKonferencji=@param
	and [Rezerwacja dnia konferencji].DataRezerwacji >
ceny.DataPodwyzszeniaCeny
	order by Ceny.DataPodwyzszeniaCeny desc),0))
	return @suma
END

GO
```

6. Widoki

- najpopularniejsze warsztaty – pokazuje najpopularniejsze warsztaty

```
CREATE VIEW [dbo].[najpopularniejsze warsztaty]
AS
SELECT        TOP (100) PERCENT dbo.Warsztaty.ID_warsztatu,
SUM(dbo.[Rezerwacja warsztatow].IloscUczestnikow) AS [Ilość chętnych]
FROM            dbo.[Rezerwacja warsztatow] INNER JOIN
                         dbo.Warsztaty ON dbo.[Rezerwacja
warsztatow].ID_warsztatu = dbo.Warsztaty.ID_warsztatu
GROUP BY dbo.Warsztaty.ID_warsztatu
ORDER BY [Ilość chętnych] DESC

GO
```

- NajpoularniejszeKonferencje – pokazuje najpopularniejsze konferencje

```
CREATE VIEW [dbo].[NajpopularniejszeKonferencje]
AS
SELECT        TOP (100) PERCENT dbo.Konferencje.ID_konferencji,
dbo.Konferencje.Nazwa, SUM(dbo.[Rezerwacja dnia
konferencji].IloscUczestnikow) AS [Ilość chętnych]
FROM            dbo.[Dni konferencji] INNER JOIN
                         dbo.Konferencje ON dbo.[Dni
konferencji].ID_konferencji = dbo.Konferencje.ID_konferencji INNER JOIN
                         dbo.[Rezerwacja dnia konferencji] ON dbo.[Dni
konferencji].ID_dniaKonferencji = dbo.[Rezerwacja dnia
konferencji].ID_dniaKonferencji
WHERE        (dbo.[Rezerwacja dnia konferencji].Anulowana IS NULL)
GROUP BY dbo.Konferencje.ID_konferencji, dbo.Konferencje.Nazwa
ORDER BY [Ilość chętnych] DESC

GO
```

- PokazFirmy – pokazuje wszystkich klientów-firmy.

```
CREATE VIEW [dbo].[PokazFirmy]
AS
SELECT        ImieKontaktowe, NazwiskoKontaktowe, Telefon, Email,
UlicaINumerBudynku, Miasto, Kraj, ID_klienta, NazwaFirmy
FROM            dbo.Klienci
WHERE        (NazwaFirmy IS NOT NULL)
```

```
GO
```

- PokazOsobPrywatne – pokazuje wszystkich klientów-osoby prywatne.

```
CREATE VIEW [dbo].[PokazOsobyPrywatne]
AS
SELECT        ID_klienta, ImieKontaktowe, NazwiskoKontaktowe, Telefon,
Email, UlicaINumerBudynku, Miasto, Kraj
FROM            dbo.Klienci
WHERE        (NazwaFirmy IS NULL)

GO
```

7. Triggery

- czy_konferencja_w_przeszlosci – sprawdza czy konferencja zaczyna się w przeszlosci, jak tak to wyskakuje blad

```
CREATE TRIGGER [dbo].[czy_konferencje_w_przeszlosci]
   ON  [dbo].[Konferencje]
   AFTER insert, update
AS
BEGIN
       DECLARE @DataR date = (SELECT DataRozpoczecia FROM inserted)
    IF((DATEDIFF(day,GETDATE(),@DataR) <= 0))
       BEGIN
       ;THROW 52000,'Nie mozna dodac konferencji w przeszlosci',1
       ROLLBACK TRANSACTION
       END
       END

GO
```

- limit_osob_warsz – sprawdza limit osób na warsztat.

```
create trigger [dbo].[limit_osob_warsz]
on [dbo].[Rezerwacja warsztatow]
after insert, update as
BEGIN
if exists (select 'TAK'
       from inserted inner join
       [Rezerwacja warsztatow] on inserted.ID_warsztatu=[Rezerwacja
warsztatow].ID_warsztatu inner join
       Warsztaty on inserted.ID_warsztatu=Warsztaty.ID_warsztatu
```

```
        group by Warsztaty.ID_warsztatu, Warsztaty.IloscMiejscWolnych
        having sum([rezerwacja
warsztatow].IloscUczestnikow)>Warsztaty.IloscMiejscWolnych
)
begin
raiserror ('Nie mozna wpisac wiecej osob na ten warsztat', 16, 1);
rollback transaction;
end
end


GO
```

- czy_osoba_moze_warsztat – sprawdza, czy osoba może zapisać się na warsztat, czyli czy jest zapisana na dzień konferencji w dniu warsztatu.

```
CREATE TRIGGER [dbo].[czy_osoba_moze_warsztat]
on [dbo].[Warsztaty_Uczestnicy]
after insert, update as
begin
if exists (select *
from inserted as Warsztaty_Uczestnicy inner join Warsztaty on
Warsztaty_Uczestnicy.ID_uczestnika = warsztaty.id_warsztatu
inner join [Rezerwacja warsztatow] on [Rezerwacja
warsztatow].id_warsztatu=Warsztaty.id_warsztatu
inner join [Rezerwacja dnia konferencji] on [Rezerwacja
warsztatow].id_rezerwacjidniakonferencji=[Rezerwacja dnia
konferencji].id_rezerwacjidniakonferencji
where Warsztaty.id_dniakonferencji!=[Rezerwacja dnia
konferencji].id_dniakonferencji
)
begin
raiserror ('Osoba nie jest zapisana na konferencje na której jest ten
warsztat.', 16, 1)
rollback transaction
end
end

GO
```

- pilnuj_warsztatu – sprawdza czy warsztat, na który zapisujemy się nie jest w godzinach warsztatu, na który już jesteśmy zapisani.

```
create trigger [dbo].[pilnuj_warsztatu]
on [dbo].[Warsztaty_Uczestnicy]
after insert, update as
```

```
begin
if (select count(*)
from inserted as warsztaty_uczestnicy inner join [Rezerwacja warsztatow] on
	[Rezerwacja warsztatow].ID_rezerwacjiWarsztatu =
Warsztaty_Uczestnicy.ID_rezerwacjiWarsztatu inner join
	Warsztaty on Warsztaty.ID_warsztatu=[Rezerwacja
warsztatow].ID_warsztatu inner join
	[Dni konferencji] on [Dni
konferencji].ID_dniaKonferencji=Warsztaty.ID_dniaKonferencji inner join
	[Rezerwacja dnia konferencji] on [Rezerwacja dnia
konferencji].ID_dniaKonferencji=[Dni konferencji].ID_dniaKonferencji inner
join
	[Dni konferencji_Uczestnicy] on [Rezerwacja dnia
konferencji].ID_rezerwacjiDniaKonferencji=[Dni
konferencji_Uczestnicy].ID_rezerwacjiDniaKonferencji inner join
	[Dni konferencji] as DK2 on [Dni konferencji].Data=DK2.Data inner
join
	Warsztaty as W_tegodnia on W_tegodnia.ID_dniaKonferencji =
DK2.ID_dniaKonferencji
	where (Warsztaty.GodzinaRozpoczecia<W_tegodnia.GodzinaRozpoczecia and
Warsztaty.GodzinaZakonczenia>W_tegodnia.GodzinaZakonczenia)
	)>1
Begin
raiserror ('Ta osoba jest już zapisana na inny warsztat w tych godzinach',
16, 1)
ROLLBACK TRANSACTION
END
end

GO
```

8. Role

W pracy z bazą powinny być wyróżnione następujące role:

- administrator – dostaje dostęp do wszystkiego i zarządza bazą także od strony edytora
- właściciel – dostaje dostęp do wszystkich danych, tabel, widoków
- pracownik firmy konferencyjnej – dostaje dostęp do konferencji, warsztatów i może dodawać klientów
- klient – dostaje możliwość zapisania się na konferencję czy warsztaty, a także wniesienie opłaty

9. Generator

Dane do bazy wygenerowaliśmy za pomocą programu Redgate, który został stworzony właśnie do tego celu. Konfiguracja danych w nim jest bardzo wygodna. Bazę wypełniliśmy danymi odpowiadającymi około 3-letniej działalności firmy.