Projekt Bazy Danych

Mikołaj Wróblewski, Mikita Yakimovich

Spis treści

1 OPIS FUNKCJI BAZY DANYCH

Baza danych obsługuje system informatyczny firmy organizującej konferencje. Mogą one trwać kilka dni, niekoniecznie występujących po sobie w sposób ciągły. Klientami są osoby fizyczne jak i firmy, uczestnikami mogą być tylko osoby fizyczne.

Z konferencją są związane warsztaty na które uczestnicy mogą się rejestrować.
W danej chwili użytkownik może być zarejestrowany tylko na jeden warsztat.
Warsztaty mogą być płatne, ale nie muszą.

1.1 Opis funkcji ze względu na użytkownika

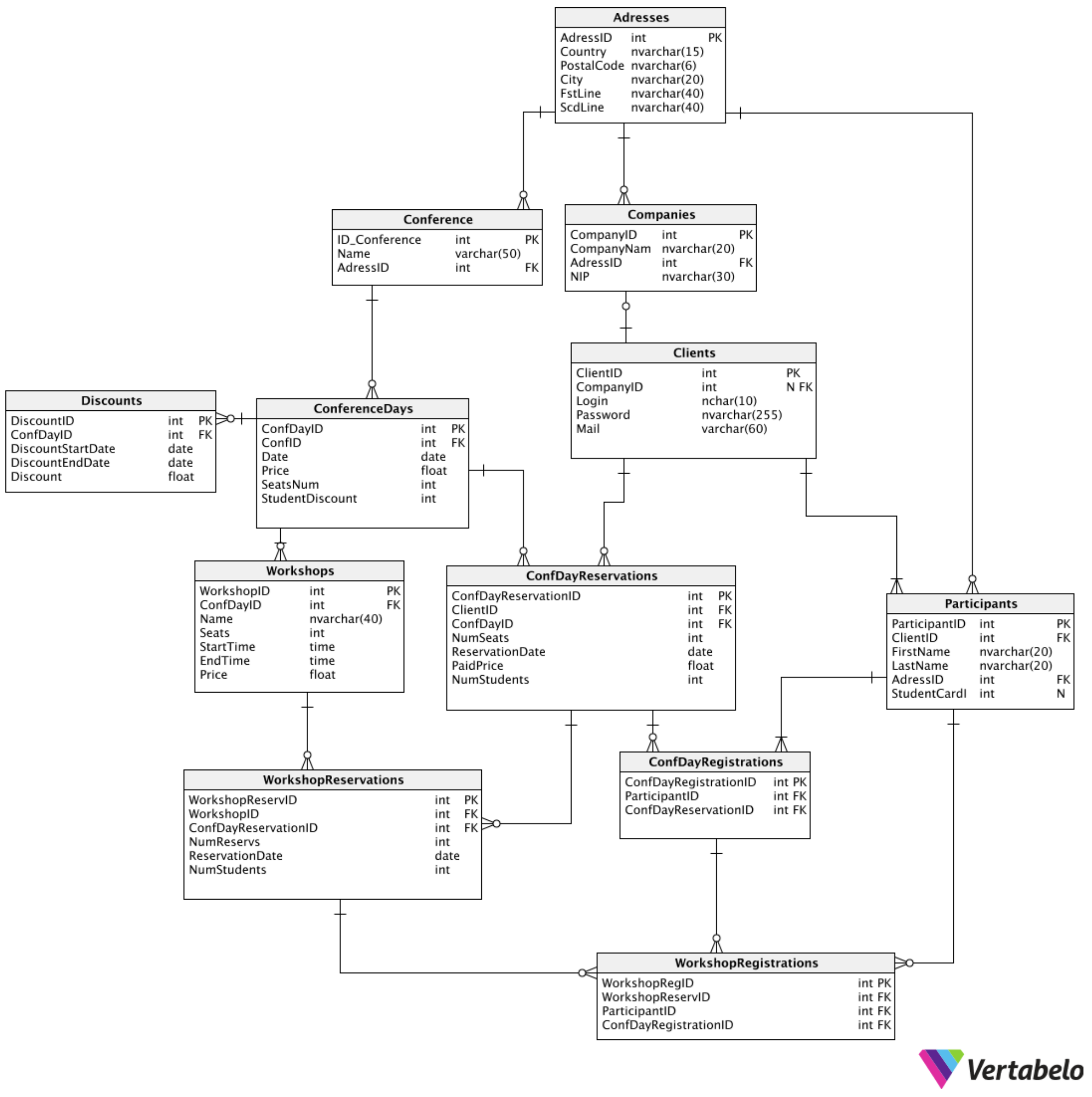
- Role definiują uprawnienia w bazie danych.

- Klient
    - Edycja swoich danych
    - Edycja swojego adresu
    - Zakładanie kont uczestnikom
    - Rezerwacja miejsc na dni konferencji
    - Anulowanie rezerwacji na konferencje
    - Usuwanie swoich uczestników z konferencji
    - Rezerwacja miejsc na warsztaty
    - Rejestrowanie uczestników na warsztaty
    - Anulowanie rezerwacji na warsztaty
    - Usuwanie swoich uczestników z warsztatów

- Uczestnik
    - Edycja swoich danych
    - Edycja swojego adresu
    - Rejestrowanie na dni konferencji
    - Rezerwowanie miejsc na warsztaty

- Organizator
    - Tworzenie konferencji
    - Tworzenie warsztatów
    - Wprowadzanie cen za konferencje
    - Wprowadzanie informacji o płatnościach
    - Sporządzenie listy identyfikatorów na dni konferencji
    - Sporządzenie listy klientów, którzy nie dokonali jeszcze płatności
    - Sporządzenie listy firm, którzy nie zarejestrowali wszystkich użytkowników
    - Sporządzenie listy z ilością rezerwacji dla klienta
    - Obliczanie należności dla danego klienta (z rozbiciem na dni)

2 Schemat bazy danych



3 Opis tabel

3.1 Adresses

Tabela zawiera infomacje o adresach klientów, miejsca gdzie odbywają się konferencje i miejsca zamieszkania uczestników. Każdy adres ma swój unikalny numer identyfikacyjny.

```
CREATE TABLE [dbo].[Adresses](
	[AdressID] [int] IDENTITY(1,1) NOT NULL,
	[Country] [nvarchar](15) NOT NULL,
	[PostalCode] [nvarchar](6) NOT NULL,
	[City] [nvarchar](20) NOT NULL,
	[FstLine] [nvarchar](40) NOT NULL,
	[ScdLine] [nvarchar](40) NOT NULL,
 CONSTRAINT [Adresses_pk] PRIMARY KEY CLUSTERED
(
       [AdressID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
       [AdressID] 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].[Adresses]  WITH CHECK ADD  CONSTRAINT [PostalCode_clength_check]
CHECK  ((len([PostalCode])>=(1) AND len([PostalCode])<=(8)))
GO

ALTER TABLE [dbo].[Adresses] CHECK CONSTRAINT [PostalCode_clength_check]
GO)
;
```

3.2 Clients

Tabela zawiera informacje dotyczące klientów indywidualnych jak i klientów, którymi są firmy. Null w polu CompanyID oznacza klienta indywidualnego.

```
CREATE TABLE [dbo].[Clients](
       [ClientID] [int] IDENTITY(1,1) NOT NULL,
       [CompanyID] [int] NULL,
       [Login] [nvarchar](10) NOT NULL,
       [Password] [nvarchar](255) NOT NULL,
       [Mail] [varchar](60) NOT NULL,
 CONSTRAINT [Clients_pk] PRIMARY KEY CLUSTERED
(
       [ClientID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
       [Mail] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
       [CompanyID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
       [Login] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
       [ClientID] 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].[Clients]  WITH CHECK ADD  CONSTRAINT [Clients_Companies] FOREIGN
KEY([CompanyID])
REFERENCES [dbo].[Companies] ([CompanyID])
GO

ALTER TABLE [dbo].[Clients] CHECK CONSTRAINT [Clients_Companies]
GO

ALTER TABLE [dbo].[Clients]  WITH CHECK ADD  CONSTRAINT [Client_mail_format_check]
CHECK  (([Mail] like '%_@_%_._%'))
GO

ALTER TABLE [dbo].[Clients] CHECK CONSTRAINT [Client_mail_format_check]
GO
;
```

3.3 Companies

Tabela zawiera informacje o firmach. Każda z firm otrzymuje unikalny numer identyfikujący ją w bazie.

```
CREATE TABLE [dbo].[Companies](
	[CompanyID] [int] IDENTITY(1,1) NOT NULL,
	[CompanyName] [nvarchar](40) NOT NULL,
	[AdressID] [int] NOT NULL,
	[NIP] [nvarchar](13) NOT NULL,
 CONSTRAINT [Companies_pk] PRIMARY KEY CLUSTERED
(
	[CompanyID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[CompanyID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[NIP] 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].[Companies]  WITH CHECK ADD  CONSTRAINT [Companies_Adresses] FOREIGN
KEY([AdressID])
REFERENCES [dbo].[Adresses] ([AdressID])
GO

ALTER TABLE [dbo].[Companies] CHECK CONSTRAINT [Companies_Adresses]
GO

ALTER TABLE [dbo].[Companies]  WITH CHECK ADD  CONSTRAINT [NIP_format_check] CHECK
(([NIP] like '[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]'))
GO

ALTER TABLE [dbo].[Companies] CHECK CONSTRAINT [NIP_format_check]
```

```
GO
```

3.4 ConfDayRegistrations

Tabela w której obecność świadczy o rejestracji uczestnika na dany dzień konferencji. Łączy tabele Participant i ConfDayReservations.

```
CREATE TABLE [dbo].[ConfDayRegistrations](
	[ConfDayRegistrationID] [int] IDENTITY(1,1) NOT NULL,
	[ParticipantID] [int] NOT NULL,
	[ConfDayReservationID] [int] NOT NULL,
 CONSTRAINT [ConfDayRegistrations_pk] PRIMARY KEY CLUSTERED
(
	[ConfDayRegistrationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
 CONSTRAINT [Participant_ConfDayReservationID_unique] UNIQUE NONCLUSTERED
(
	[ConfDayReservationID] ASC,
	[ParticipantID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[ConfDayRegistrationID] 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].[ConfDayRegistrations]  WITH CHECK ADD  CONSTRAINT
[ConfDayRegistration_Participants] FOREIGN KEY([ParticipantID])
REFERENCES [dbo].[Participants] ([ParticipantID])
GO

ALTER TABLE [dbo].[ConfDayRegistrations] CHECK CONSTRAINT
[ConfDayRegistration_Participants]
GO

ALTER TABLE [dbo].[ConfDayRegistrations]  WITH CHECK ADD  CONSTRAINT
[ConfDayReservation_ConfDayRegistration] FOREIGN KEY([ConfDayReservationID])
REFERENCES [dbo].[ConfDayReservations] ([ConfDayReservationID])
GO

ALTER TABLE [dbo].[ConfDayRegistrations] CHECK CONSTRAINT
[ConfDayReservation_ConfDayRegistration]
GO
```

3.5 ConfDayReservations

Tabela zawierająca informację o dokonanej rezerwacji przez klienta na liczbę miejsc określoną w polu NumSeats. Posiada bit anulowania, gdyż po anulowaniu danej rezerwacji chcemy trzymać o niej informacje, a także opłaconą kwotę.

```
CREATE TABLE [dbo].[ConfDayReservations](
	[ConfDayReservationID] [int] IDENTITY(1,1) NOT NULL,
	[ClientID] [int] NOT NULL,
	[ConfDayID] [int] NOT NULL,
	[NumSeats] [int] NOT NULL,
	[ReservationDate] [datetime] NOT NULL,
	[PaidPrice] [float] NOT NULL,
	[NumStudents] [int] NOT NULL,
	[Cancelled] [bit] NULL,
 CONSTRAINT [ConfDayReservations_pk] PRIMARY KEY CLUSTERED
(
	[ConfDayReservationID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[ConfDayReservationID] 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].[ConfDayReservations] ADD  DEFAULT ((0)) FOR [PaidPrice]
GO

ALTER TABLE [dbo].[ConfDayReservations] ADD  DEFAULT ((0)) FOR [NumStudents]
GO

ALTER TABLE [dbo].[ConfDayReservations] ADD  DEFAULT ((0)) FOR [Cancelled]
GO

ALTER TABLE [dbo].[ConfDayReservations]  WITH CHECK ADD  CONSTRAINT
[ConfDayReservation_Clients] FOREIGN KEY([ClientID])
REFERENCES [dbo].[Clients] ([ClientID])
GO

ALTER TABLE [dbo].[ConfDayReservations] CHECK CONSTRAINT [ConfDayReservation_Clients]
GO

ALTER TABLE [dbo].[ConfDayReservations]  WITH CHECK ADD  CONSTRAINT
[ConfDayReservation_ConferenceDays] FOREIGN KEY([ConfDayID])
REFERENCES [dbo].[ConferenceDays] ([ConfDayID])
GO

ALTER TABLE [dbo].[ConfDayReservations] CHECK CONSTRAINT
[ConfDayReservation_ConferenceDays]
GO

ALTER TABLE [dbo].[ConfDayReservations]  WITH CHECK ADD  CONSTRAINT
[NumSeats_num_ckeck] CHECK  (([NumSeats]>(0)))
GO

ALTER TABLE [dbo].[ConfDayReservations] CHECK CONSTRAINT [NumSeats_num_ckeck]
GO

ALTER TABLE [dbo].[ConfDayReservations]  WITH CHECK ADD  CONSTRAINT
[NumStudents_not_more_then_NumSeats_check] CHECK  (([NumStudents]<=[NumSeats]))
GO

ALTER TABLE [dbo].[ConfDayReservations] CHECK CONSTRAINT
[NumStudents_not_more_then_NumSeats_check]
GO
```

3.6 Conference

Tabela posiadająca informacje o zniżce studenckiej na daną konfencję, a także nazwę danej konferencji. Posiada też informacje o adresie, pod którym się odbywa.

```
CREATE TABLE [dbo].[Conference](
	[ConferenceID] [int] IDENTITY(1,1) NOT NULL,
	[Name] [varchar](50) NOT NULL,
	[AdressID] [int] NOT NULL,
	[StudentDiscount] [int] NOT NULL,
 CONSTRAINT [Conference_pk] PRIMARY KEY CLUSTERED
(
	[ConferenceID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[ConferenceID] 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].[Conference] ADD  DEFAULT ((0)) FOR [StudentDiscount]
GO

ALTER TABLE [dbo].[Conference]  WITH CHECK ADD  CONSTRAINT [Conference_Adresses]
FOREIGN KEY([AdressID])
REFERENCES [dbo].[Adresses] ([AdressID])
GO

ALTER TABLE [dbo].[Conference] CHECK CONSTRAINT [Conference_Adresses]
GO
```

3.7 ConferenceDays

Tabela posiadająca informacje o odbywających się konferencjach, a także o cenie. Posiada też informacje o zniżce studenckiej i liczbę miejsc.

```
CREATE TABLE [dbo].[ConferenceDays](
	[ConfDayID] [int] IDENTITY(1,1) NOT NULL,
	[ConfID] [int] NOT NULL,
	[Date] [date] NOT NULL,
	[Price] [float] NOT NULL,
	[SeatsNum] [int] NOT NULL,
	[StudentDiscount] [float] NULL,
 CONSTRAINT [ConferenceDays_pk] PRIMARY KEY CLUSTERED
(
	[ConfDayID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[ConfDayID] 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].[ConferenceDays] ADD  DEFAULT ((0)) FOR [StudentDiscount]
GO

ALTER TABLE [dbo].[ConferenceDays]  WITH CHECK ADD  CONSTRAINT
[Konferencje_Dni_Konferencje] FOREIGN KEY([ConfID])
REFERENCES [dbo].[Conference] ([ConferenceID])
GO

ALTER TABLE [dbo].[ConferenceDays] CHECK CONSTRAINT [Konferencje_Dni_Konferencje]
GO

ALTER TABLE [dbo].[ConferenceDays]  WITH CHECK ADD  CONSTRAINT [SeatsNum_not_zero]
CHECK  (([SeatsNum]>(0)))
GO

ALTER TABLE [dbo].[ConferenceDays] CHECK CONSTRAINT [SeatsNum_not_zero]
GO
```

3.8 Discounts

Tabela posiadająca informację o zniżkach obowiązujących w danym przedziale czasowym.

```
CREATE TABLE [dbo].[Discounts](
	[DiscountID] [int] IDENTITY(1,1) NOT NULL,
	[ConfDayID] [int] NOT NULL,
	[DiscountStartDate] [date] NOT NULL,
	[DiscountEndDate] [date] NOT NULL,
	[Discount] [float] NOT NULL,
 CONSTRAINT [Discounts_pk] PRIMARY KEY CLUSTERED
(
	[DiscountID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[DiscountID] 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].[Discounts] ADD  DEFAULT ((0)) FOR [Discount]
GO

ALTER TABLE [dbo].[Discounts]  WITH CHECK ADD  CONSTRAINT [Discounts_ConferenceDays]
FOREIGN KEY([ConfDayID])
REFERENCES [dbo].[ConferenceDays] ([ConfDayID])
GO

ALTER TABLE [dbo].[Discounts] CHECK CONSTRAINT [Discounts_ConferenceDays]
GO
```

```
ALTER TABLE [dbo].[Discounts]  WITH CHECK ADD  CONSTRAINT [Discount_format_check]
CHECK  (([Discount]>=(0) AND [Discount]<=(1)))
GO

ALTER TABLE [dbo].[Discounts] CHECK CONSTRAINT [Discount_format_check]
GO

ALTER TABLE [dbo].[Discounts]  WITH CHECK ADD  CONSTRAINT
[Start_End_Discount_Date_check] CHECK  (([DiscountStartDate]<[DiscountEndDate]))
GO

ALTER TABLE [dbo].[Discounts] CHECK CONSTRAINT [Start_End_Discount_Date_check]
GO
```

3.9 Participants

Tabela zawierająca informacje o uczestniku.

```
CREATE TABLE [dbo].[Participants](
	[ParticipantID] [int] IDENTITY(1,1) NOT NULL,
	[ClientID] [int] NOT NULL,
	[FirstName] [nvarchar](20) NOT NULL,
	[LastName] [nvarchar](20) NOT NULL,
	[AdressID] [int] NOT NULL,
	[StudentCardID] [int] NULL,
 CONSTRAINT [Participants_pk] PRIMARY KEY CLUSTERED
(
	[ParticipantID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[StudentCardID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[ParticipantID] 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].[Participants] ADD  DEFAULT ('Kamil') FOR [FirstName]
GO

ALTER TABLE [dbo].[Participants] ADD  DEFAULT ('Sokolowski') FOR [LastName]
GO

ALTER TABLE [dbo].[Participants] ADD  DEFAULT ((0)) FOR [StudentCardID]
GO

ALTER TABLE [dbo].[Participants]  WITH CHECK ADD  CONSTRAINT [Participants_Adresses]
FOREIGN KEY([AdressID])
REFERENCES [dbo].[Adresses] ([AdressID])
GO
```

```
ALTER TABLE [dbo].[Participants] CHECK CONSTRAINT [Participants_Adresses]
GO

ALTER TABLE [dbo].[Participants]  WITH CHECK ADD  CONSTRAINT [Participants_Clients]
FOREIGN KEY([ClientID])
REFERENCES [dbo].[Clients] ([ClientID])
GO

ALTER TABLE [dbo].[Participants] CHECK CONSTRAINT [Participants_Clients]
GO

ALTER TABLE [dbo].[Participants]  WITH CHECK ADD  CONSTRAINT [FirstName_Format] CHECK
((NOT [FirstName] like '%[^a-zA-Z,.\-\ ]%'))
GO

ALTER TABLE [dbo].[Participants] CHECK CONSTRAINT [FirstName_Format]
GO

ALTER TABLE [dbo].[Participants]  WITH CHECK ADD  CONSTRAINT [LastName_Format] CHECK
((NOT [LastName] like '%[^a-zA-Z,.\-\ ]%'))
GO

ALTER TABLE [dbo].[Participants] CHECK CONSTRAINT [LastName_Format]
GO
```

3.10 WorkshopRegistrations

Tabela posiadająca informacje o rejestracjach na warsztaty dla poszczególnych uczestników.

```
CREATE TABLE [dbo].[WorkshopRegistrations](
	[WorkshopRegID] [int] IDENTITY(1,1) NOT NULL,
	[WorkshopReservID] [int] NOT NULL,
	[ParticipantID] [int] NOT NULL,
	[ConfDayRegistrationID] [int] NOT NULL,
 CONSTRAINT [WorkshopRegistrations_pk] PRIMARY KEY CLUSTERED
(
	[WorkshopRegID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[WorkshopRegID] 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].[WorkshopRegistrations]  WITH CHECK ADD  CONSTRAINT
[WorkshopRegistrations_ConfDayRegistrations] FOREIGN KEY([ConfDayRegistrationID])
REFERENCES [dbo].[ConfDayRegistrations] ([ConfDayRegistrationID])
GO

ALTER TABLE [dbo].[WorkshopRegistrations] CHECK CONSTRAINT
[WorkshopRegistrations_ConfDayRegistrations]
```

```
GO

ALTER TABLE [dbo].[WorkshopRegistrations]  WITH CHECK ADD  CONSTRAINT
[WorkshopRegistrations_Participants] FOREIGN KEY([ParticipantID])
REFERENCES [dbo].[Participants] ([ParticipantID])
GO

ALTER TABLE [dbo].[WorkshopRegistrations] CHECK CONSTRAINT
[WorkshopRegistrations_Participants]
GO

ALTER TABLE [dbo].[WorkshopRegistrations]  WITH CHECK ADD  CONSTRAINT
[WorkshopRegistrations_WorkshopReservations] FOREIGN KEY([WorkshopReservID])
REFERENCES [dbo].[WorkshopReservations] ([WorkshopReservID])
GO

ALTER TABLE [dbo].[WorkshopRegistrations] CHECK CONSTRAINT
[WorkshopRegistrations_WorkshopReservations]
GO
```

3.11 WorkshopReservations

Tabela posiadająca informacje o rezerwacjach na warsztaty, ma liczbę uczestników, liczbę uczestników będących studentami, a także bit anulowania.

```
CREATE TABLE [dbo].[WorkshopReservations](
	[WorkshopReservID] [int] IDENTITY(1,1) NOT NULL,
	[WorkshopID] [int] NOT NULL,
	[ConfDayReservationID] [int] NOT NULL,
	[NumReservs] [int] NOT NULL,
	[ReservationDate] [datetime] NOT NULL,
	[NumStudents] [int] NOT NULL,
	[Cancelled] [bit] NULL,
 CONSTRAINT [WorkshopReservations_pk] PRIMARY KEY CLUSTERED
(
	[WorkshopReservID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[WorkshopReservID] 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].[WorkshopReservations] ADD  DEFAULT ((0)) FOR [NumStudents]
GO

ALTER TABLE [dbo].[WorkshopReservations] ADD  DEFAULT ((0)) FOR [Cancelled]
GO

ALTER TABLE [dbo].[WorkshopReservations]  WITH CHECK ADD  CONSTRAINT
[WorkshopReservations_ConfDayReservations] FOREIGN KEY([ConfDayReservationID])
REFERENCES [dbo].[ConfDayReservations] ([ConfDayReservationID])
GO

ALTER TABLE [dbo].[WorkshopReservations] CHECK CONSTRAINT
[WorkshopReservations_ConfDayReservations]
```

```
GO

ALTER TABLE [dbo].[WorkshopReservations]  WITH CHECK ADD  CONSTRAINT
[WorkshopReservations_Workshops] FOREIGN KEY([WorkshopID])
REFERENCES [dbo].[Workshops] ([WorkshopID])
GO

ALTER TABLE [dbo].[WorkshopReservations] CHECK CONSTRAINT
[WorkshopReservations_Workshops]
GO

ALTER TABLE [dbo].[WorkshopReservations]  WITH CHECK ADD  CONSTRAINT
[NumReservs_check] CHECK  (([NumReservs]>(0)))
GO

ALTER TABLE [dbo].[WorkshopReservations] CHECK CONSTRAINT [NumReservs_check]
GO

ALTER TABLE [dbo].[WorkshopReservations]  WITH CHECK ADD  CONSTRAINT
[NumSrudent_not_greater_then_Reservs_ckeck] CHECK  (([NumStudents]<=[NumReservs]))
GO

ALTER TABLE [dbo].[WorkshopReservations] CHECK CONSTRAINT
[NumSrudent_not_greater_then_Reservs_ckeck]
GO
```

3.12 Workshops

Tabela określająca warsztaty. Posiada unikalne pole WorkshopID, nazwę warsztatu, informację o tym, jakiego dnia się odbywają, liczbę miejsc i czas. Posiada także cenę poszczególnego warsztatu.

```
CREATE TABLE [dbo].[Workshops](
	[WorkshopID] [int] IDENTITY(1,1) NOT NULL,
	[ConfDayID] [int] NOT NULL,
	[Name] [nvarchar](40) NOT NULL,
	[Seats] [int] NOT NULL,
	[StartTime] [datetime] NOT NULL,
	[EndTime] [datetime] NOT NULL,
	[Price] [float] NOT NULL,
 CONSTRAINT [Workshops_pk] PRIMARY KEY CLUSTERED
(
	[WorkshopID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
UNIQUE NONCLUSTERED
(
	[WorkshopID] 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].[Workshops]  WITH CHECK ADD  CONSTRAINT [Workshops_ConferenceDays]
FOREIGN KEY([ConfDayID])
REFERENCES [dbo].[ConferenceDays] ([ConfDayID])
GO
```

```
ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [Workshops_ConferenceDays]
GO

ALTER TABLE [dbo].[Workshops]  WITH CHECK ADD  CONSTRAINT [Start_End_time_check] CHECK
(([StartTime]<[EndTime]))
GO

ALTER TABLE [dbo].[Workshops] CHECK CONSTRAINT [Start_End_time_check]
GO
```

4 Widoki

4.1 Widok wyświetlający liczbę rezerwacji dla danego klienta na dane dni konferencji

```
CREATE VIEW [dbo].[v_ClientConfDayReservationCount]
             AS
             SELECT CD.ConfDayID AS 'ConfDayID' , C.ClientID AS 'ClientID', COUNT(*)
AS 'Number of reservations'
              from ConfDayReservations as CDRv
             INNER JOIN ConferenceDays as CD on CDRv.ConfDayID = CD.ConfDayID
             INNER JOIN dbo.Clients AS C ON CDRv.ClientID = C.ClientID
             GROUP BY C.ClientID, CD.ConfDayID
GO
```

4.2 Widok wyświetlający jakie opłaty uiścił dany klient

```
CREATE VIEW [dbo].[v_ClientPayments]
             AS
             SELECT Login,  ISNULL(CompanyName, 'Private person') AS 'Company name',
SUM(PaidPrice) AS 'Paid price' FROM
             dbo.Clients
             INNER JOIN dbo.ConfDayReservations ON Clients.ClientID =
ConfDayReservations.ClientID
             LEFT OUTER JOIN dbo.Companies ON Companies.CompanyID=Clients.CompanyID
             GROUP BY Login, CompanyName
GO
```

4.3 Widok wyświetlający klientów, którzy nie uiścili opłat

```
CREATE VIEW [dbo].[v_ClientsWhoDidntPay]
             AS
             SELECT Login AS 'Login', ISNULL(CompanyName, 'Personal client') AS
'Company',  dbo.f_PriceToPayPerDay(Clients.ClientID, ConfDayReservationID) - PaidPrice
AS 'Price to pay'
             FROM dbo.Clients
             INNER JOIN dbo.ConfDayReservations ON ConfDayReservations.ClientID =
Clients.ClientID
             LEFT OUTER JOIN dbo.Companies ON Companies.CompanyID = Clients.CompanyID
             WHERE dbo.f_PriceToPayPerDay(Clients.ClientID, ConfDayReservationID)
>PaidPrice AND
             DATEDIFF(DAY, ReservationDate,  GETDATE()) > 7
             AND Cancelled = 1
GO
```

4.4 Wyświetlanie sumarycznej liczy rezerwacji klientów

Widok przygotowany do posortowania klientów względem aktywności.

```
CREATE VIEW [dbo].[v_MostActiveClients]
              AS
              SELECT Clients.ClientID AS 'ClientID', Login AS 'Login', COUNT(*)  AS
'Number of Reservations' FROM dbo.Clients
              INNER JOIN dbo.ConfDayReservations ON ConfDayReservations.ClientID =
Clients.ClientID
              GROUP BY Clients.ClientID, Login

GO
```

5. Funkcje

5.1 Obliczanie wolnych miejsc na konferencję

```
CREATE FUNCTION [dbo].[ConfDayFreeSeats](@CDayID int)
              RETURNS int
              AS
              BEGIN
                     DECLARE @Seats AS int
                     SET @Seats = (
                            SELECT SeatsNum
                            FROM ConferenceDays
                            WHERE ConfDayID = @CDayID
                     )

                     DECLARE @Taken AS int
                     SET @Taken = (
                            SELECT SUM( NumSeats)
                            FROM ConfDayReservations
                            WHERE ConfDayID = @CDayID and Cancelled = 0
                     )
                     IF @Taken is null
                     BEGIN
                            SET @Taken = 0
                     END
                     RETURN (@Seats - @Taken)
              END
GO
```

5.2 Obliczanie czy z danej rezerwacji można wykonać rejestrację na konferencję

```
CREATE FUNCTION [dbo].[ConfDayReservationFreeSeats](@ReservationID int)
              RETURNS int
              AS
              BEGIN
                     DECLARE @Seats AS int
                     SET @Seats = (
				SELECT NumSeats
				FROM ConfDayReservations
				WHERE ConfDayReservationID = @ReservationID
			)

			DECLARE @Taken AS int
			SET @Taken = (
				SELECT COUNT(*)
				FROM ConfDayRegistrations
				WHERE ConfDayReservationID = @ReservationID
			)
			IF @Taken is null
			BEGIN
				SET @Taken = 0
			END
			RETURN (@Seats - @Taken)
		END
GO
```

5.3 Obliczanie wolnych miejsc na warsztaty

```
CREATE FUNCTION [dbo].[WorkshopFreeSeats](@WorkshopID int)
		RETURNS int
		AS
		BEGIN
			DECLARE @Seats AS int
			SET @Seats = (
				SELECT Seats
				FROM Workshops
				WHERE WorkshopID = @WorkshopID
			)

			DECLARE @Taken AS int
			SET @Taken = (
				SELECT SUM(NumReservs)
				FROM WorkshopReservations
				WHERE WorkshopID = @WorkshopID and Cancelled = 0
			)
			IF @Taken is null
			BEGIN
				SET @Taken = 0
			END
			RETURN (@Seats - @Taken)
		END
GO
```

5.4 Obliczanie czy z danej rezerwacji warsztatu można wykonać rejestrację

```
CREATE FUNCTION [dbo].[WorkshopReservationFreeSeats](@ReservationID int)
		RETURNS int
		AS
		BEGIN
			DECLARE @Seats AS int
			SET @Seats = (
                        SELECT NumReservs
                        FROM WorkshopReservations
                        WHERE WorkshopReservID = @ReservationID
                )

                DECLARE @Taken AS int
                SET @Taken = (
                        SELECT COUNT(*)
                        FROM WorkshopRegistrations
                        WHERE WorkshopReservID = @ReservationID
                )
                IF @Taken is null
                BEGIN
                        SET @Taken = 0
                END
                RETURN (@Seats - @Taken)
        END
GO
```

5.5 Obliczanie kwoty do zapłaty za jeden dzień

```
CREATE FUNCTION [dbo].[f_PriceToPayPerDay](@ClientID int, @ConfDayReservationID int)
        RETURNS float
        AS
        BEGIN

                Declare @toPayWorkshop float =
                (Select SUM (Ws.Price*((NumSeats-CDRv.NumStudents)
+(CDRv.NumStudents)*(1-(StudentDiscount)))*(1-WR.Cancelled))
                 from ConfDayReservations as CDRv
                inner join ConferenceDays as CD on CDRv.ConfDayID = CD.ConfDayID
                inner join Workshops as Ws on Ws.ConfDayID = CD.ConfDayID
                inner join WorkshopReservations as WR on WR.ConfDayReservationID =
CDRv.ConfDayReservationID
                where CDRv.ClientID = @ClientID and CDRv.Cancelled = 0 and
WR.Cancelled = 0)


                Declare @Discount float = (select Ds.Discount from
ConfDayReservations as CDRv
                inner join ConferenceDays as CD on CDRv.ConfDayID = CD.ConfDayID
                inner join Discounts as Ds on CD.Date between Ds.DiscountStartDate
and Ds.DiscountEndDate
                where CDRv.ConfDayReservationID = @ConfDayReservationID)

                IF @Discount IS NULL
                BEGIN

                        RETURN ( @toPayWorkshop +
                        (Select SUM (CD.Price*((NumSeats-CDRv.NumStudents)
+(CDRv.NumStudents)*(1-(StudentDiscount)))*(1-CDRv.Cancelled))
                         from ConfDayReservations as CDRv
                        inner join ConferenceDays as CD on CDRv.ConfDayID =
CD.ConfDayID
                        where CDRv.ClientID = @ClientID and ConfDayReservationID =
@ConfDayReservationID and CDRv.Cancelled = 0)
                        )
                END
                ELSE
                        RETURN ( @toPayWorkshop +
                        (Select SUM ((1-@Discount)*CD.Price*((NumSeats-
CDRv.NumStudents) +(CDRv.NumStudents)*(1-(StudentDiscount)))*(1-CDRv.Cancelled))
```

```
                                     from ConfDayReservations as CDRv
                                    inner join ConferenceDays as CD on CDRv.ConfDayID =
CD.ConfDayID
                                    where CDRv.ClientID = @ClientID and ConfDayReservationID =
@ConfDayReservationID and CDRv.Cancelled = 0)
                                    )
                              RETURN 0.0
                        END
GO
```

5.6 Obliczanie sumarycznej kwoty do zapłaty

```
CREATE FUNCTION [dbo].[f_PriceToPay](@ClientID int)
                        RETURNS float
                        AS
                        BEGIN

                              DECLARE @ToPay float = (select
Sum(dbo.f_PriceToPayPerDay(@ClientID, ConfDayReservations.ConfDayReservationID))
                              from ConfDayReservations where
ConfDayReservations.ClientID=@ClientID AND ConfDayReservations.Cancelled=0)

                              RETURN (@toPay)
                        END
GO
```

5.7 Zwracanie tabeli zawierającej uczestników danego dnia konferencji

```
CREATE FUNCTION [dbo].[f_ConfDayParticipants] (@ConfDayID int)
                        RETURNS TABLE
                        AS

                              RETURN (SELECT dbo.Participants.FirstName,
dbo.Participants.LastName, dbo.Participants.StudentCardID as 'StudentCard ID'
                                          FROM dbo.ConferenceDays
                                          INNER JOIN dbo.ConfDayReservations ON
ConfDayReservations.ConfDayID = ConferenceDays.ConfDayID
                                          INNER JOIN dbo.ConfDayRegistrations ON
ConfDayRegistrations.ConfDayReservationID = ConfDayReservations.ConfDayReservationID
                                          INNER JOIN dbo.Participants ON
Participants.ParticipantID = ConfDayRegistrations.ParticipantID
                                          WHERE ConferenceDays.ConfDayID = @ConfDayID)
GO
```

5.8 Zwracanie tabeli zawierającej uczestników danego warsztatu

```
CREATE FUNCTION [dbo].[f_WorkshopParticipants] (@WorkshopID int)
                        RETURNS TABLE
                        AS

                              RETURN (SELECT dbo.Participants.FirstName,
dbo.Participants.LastName, dbo.Participants.StudentCardID as 'StudentCard ID'
                                          FROM dbo.Workshops
                                          INNER JOIN dbo.WorkshopReservations ON
WorkshopReservations.WorkshopID = Workshops.WorkshopID
```

```
                                  inner JOIN dbo.WorkshopRegistrations ON
WorkshopRegistrations.WorkshopReservID = WorkshopReservations.WorkshopReservID
                                  INNER JOIN dbo.Participants ON
Participants.ParticipantID = WorkshopRegistrations.ParticipantID
                                  WHERE Workshops.WorkshopID = @WorkshopID)
GO
```

6. Procedury dodające dane

6.1 Procedura dodająca/modyfikująca adres

Procedura dodaje nowy adres, jeżeli już taki istnieje, aby zachować integralność bazy danych.

```
CREATE PROCEDURE [dbo].[AddAdress]
       @AdressID int OUT,
       @Country nvarchar (15),
       @PostalCode nvarchar (6),
       @City nvarchar (20),
       @FstLine nvarchar (40),
       @ScdLine nvarchar (40)
AS BEGIN
       SET NOCOUNT ON
              BEGIN TRY
                     BEGIN TRANSACTION

                            IF @Country is null or LTRIM(@Country) = ''
                            THROW 2500, 'Country cant be null or blank!!!', 1

                            IF @PostalCode is null or LTRIM(@PostalCode) = ''
                            THROW 2500, 'PostalCode cant be null or blank!!!', 1

                            IF @City is null or LTRIM(@City) =''
                            THROW 2500, 'City cant be null or blank!!', 1

                            IF @FstLine is null or LTRIM(@FstLine) = ''
                            THROW 2500, 'First Line cant be null!', 1

                            IF @ScdLine is null
                            SET @ScdLine = ''

                            SELECT @AdressID = AdressID
                            From Adresses
                            WHERE FstLine = @FstLine AND ScdLine = @ScdLine AND City =
@City AND PostalCode = @PostalCode

                            IF @AdressID is null
                                   or (select count(pr.AdressID)
                                          from (select AdressID
                                                        from company
                                                        union all
                                                        select AdressID
                                                        from Participants) as pr
                                          where AdressID = @AdressID ) > 1
                                   BEGIN
```

```
                                        INSERT INTO Adresses (FstLine, ScdLine, City,
PostalCode)
                                        VALUES (@FstLine, @ScdLine, @City,
@PostalCode)
                                        SET @AdressID = SCOPE_IDENTITY()
                                  END

                        COMMIT TRANSACTION
                END TRY

                BEGIN CATCH
                        ROLLBACK TRANSACTION
                        THROW
                END CATCH

END

GO
```

6.2 Procedura dodająca firmę

```
CREATE PROCEDURE [dbo].[AddCompany] (
        @CompanyID int,
        @CompanyName int,
        @AdressID int,
        @NIP nvarchar(12)
)
AS BEGIN
        SET NOCOUNT ON


                IF @CompanyID is null
                        THROW 2503, 'CompanyID cant be null!', 1

                IF @CompanyName is null
                        THROW 2503, 'CompanyName cant be null!', 1

                IF @AdressID is null
                        THROW 2503, 'AdressID cant be null!', 1

                IF @NIP is null
                        THROW 2503, 'NIP cant be null!', 1

                INSERT INTO Companies (CompanyID, CompanyName, AdressID, NIP)
                VALUES (@CompanyID, @CompanyName, @AdressID, @NIP)

END

GO
```

6.3 Dodawanie klienta firmowego

```
CREATE PROCEDURE [dbo].[AddCompanyClient] (
        @ClientID int = NULL OUT,
        @CompanyID int = NULL OUT,
        @Login nchar(10),
        @Password nvarchar(255),
        @Mail varchar(60),

        @CompanyName int,
        @AdressID int,
        @NIP nvarchar(12)
)

AS BEGIN
SET NOCOUNT ON
        BEGIN TRY
                BEGIN TRANSACTION

                                IF @Login is null
                                        THROW 2502, 'Login cant be null!', 1

                                IF @Password is null
                                        THROW 2502, 'Password cant be null!', 1

                                IF @Mail is null
                                        THROW 2502, 'Password cant be null!', 1

                                INSERT Clients (Login, Password, Mail)
                                VALUES (@Login, @Password, @Mail)
                                SET @ClientID = SCOPE_IDENTITY();
                                SET @CompanyID = SCOPE_IDENTITY();

                                EXEC AddCompany @CompanyID, @CompanyName, @AdressID, @NIP

                COMMIT TRANSACTION

        END TRY

        BEGIN CATCH
                ROLLBACK TRANSACTION
                THROW
        END CATCH
END

GO
```

6.4 Dodawanie klienta indywidualnego

```
CREATE PROCEDURE [dbo].[AddIndividualClient] ( --klient może być zarówno firmą, jak i
osobą indywidualną - rozbijamy.
-- dodatkowo od razu dodajemy participanta
        @ClientID int = NULL OUT,
        @Login nchar(10),
        @Password nvarchar(255),
        @Mail varchar(60),

--uzywane do dodania participanta
        @ParticipantID int = NULL OUT,
     @FirstName nvarchar(20),
     @LastName nvarchar(20),
     @StudentCardID int,

 --uzywane do dodania adresu
        @AdressID int,
     @Country nvarchar(15),
     @PostalCode nvarchar(6),
     @City nvarchar(20),
     @FstLine nvarchar(40),
     @ScdLine nvarchar(40)
)
        AS
        BEGIN
        SET NOCOUNT ON
               BEGIN TRY
                      BEGIN TRANSACTION

                             IF @ClientID is null
                                    THROW 2500, 'ClientID cant be null!', 1

                             IF @Login is null
                                    THROW 2500, 'Login cant be null!', 1

                             IF @Password is null
                                    THROW 2500, 'Password cant be null!', 1

                             IF @Mail is null
                                    THROW 2500, 'Mail cant be null!', 1

                             INSERT INTO Clients(Login, Password, Mail, CompanyID)
                             VALUES (@Login, @Password, @Mail, NULL)
                             SET @ClientID = SCOPE_IDENTITY();

                             EXEC AddParticipant @FirstName, @LastName, @AdressID,
@StudentCardID, @ClientID;

                      COMMIT TRANSACTION
               END TRY

               BEGIN CATCH
                      ROLLBACK TRANSACTION
                      THROW
               END CATCH
        END
GO
```

6.5 Dodawanie zniżek

```
CREATE PROCEDURE [dbo].[AddDiscount]
       @ConfDayID int,
       @DiscountStartDate date,
       @DiscountEndDate date,
       @Discount float,

       @DiscountID int = NULL OUT
AS
BEGIN
       SET NOCOUNT ON
       BEGIN TRY
              BEGIN TRANSACTION


              IF @ConfDayID IS NULL
                     THROW 14,'@ConfDayID is null in AddDiscount', 1

              IF @DiscountStartDate IS NULL
```

```
                        THROW 14,'@DiscountStartDate is null in AddDiscount', 1

                IF @DiscountEndDate IS NULL
                        THROW 14,'@DiscountEndTime is null in AddDiscount', 1

                IF @ConfDayID IS NULL
                        THROW 14,'@Discount is null in AddDiscount', 1

                IF @Discount < 0
                        THROW 14,'@Discount must be > 0  in AddDiscount', 1

                IF @DiscountStartDate > @DiscountEndDate
                        THROW 14,'@DiscountStartDate > @DiscountEndDate in AddDiscount', 1

                DECLARE @ConferenceDate date = (select Date from ConferenceDays where
ConferenceDays.ConfDayID = @ConfDayID)

                IF @DiscountStartDate > @ConferenceDate or @DiscountEndDate >
@ConferenceDate
                        THROW 14,'@DiscountStartDate > @ConferenceDate or @DiscountEndDate
> @ConferenceDate in AddDiscount', 1

                INSERT Discounts(ConfDayID, DiscountStartDate, DiscountEndDate, Discount)
                VALUES (@ConfDayID, @DiscountStartDate, @DiscountEndDate, @Discount)
                SET @DiscountID =  SCOPE_IDENTITY();

                COMMIT TRANSACTION
        END TRY

        BEGIN CATCH
                ROLLBACK TRANSACTION
                THROW
        END CATCH

END
GO
```

6.6 Dodawanie konferencji

```
CREATE PROCEDURE [dbo].[AddConference]
        @Name varchar,
        @AdressID int,
        @ConferenceID int = NULL OUT
AS
BEGIN
        SET NOCOUNT ON
        BEGIN TRY
                BEGIN TRANSACTION


                IF @AdressID IS NULL
                        THROW 14,'Musisz podac adres', 1

                IF @Name IS NULL
                        THROW 14, 'Musisz podac nazwe konferencji', 1


                INSERT Conference (Name, AdressID)
                VALUES (@Name, @AdressID)
                SET @ConferenceID =  SCOPE_IDENTITY();

                COMMIT TRANSACTION
        END TRY

        BEGIN CATCH
                ROLLBACK TRANSACTION
                THROW
        END CATCH

END
GO
```

6.7 Dodawanie dnia konferencji

```
CREATE PROCEDURE [dbo].[AddConferenceDay]
        @ConfID int,
        @Date date,
        @Price float,
        @SeatNums int,
        @StudentDiscount float,
        @ConfDayID int = NULL OUT
AS
BEGIN
        SET NOCOUNT ON
        BEGIN TRY
                BEGIN TRANSACTION


                IF @ConfID IS NULL
                        THROW 14,'conf_id is null', 1

                IF @Date IS NULL
                        THROW 14, 'Musisz podac dzien konferencji', 1

                IF @Price IS NULL
                        THROW 14, 'Musisz podac cene dnia konferencji', 1

                IF @SeatNums IS NULL
                        THROW 14, 'Musisz podac liczbe miejsc w dniu konferencji
konferencji', 1

                IF @StudentDiscount IS NULL OR @StudentDiscount <0
                        THROW 14, '@StudentDiscount IS NULL OR @StudentDiscount < 0 in
AddConferenceDay', 1

                INSERT ConferenceDays (ConfID, Date, Price, SeatsNum, StudentDiscount)
                VALUES (@ConfDayID, @Date, @Price, @SeatNums,@StudentDiscount)
                SET @ConfDayID =  SCOPE_IDENTITY();

                COMMIT TRANSACTION
        END TRY

        BEGIN CATCH
                ROLLBACK TRANSACTION
                THROW
        END CATCH

END
```

```
GO
```

6.8 Dodawanie uczestników

```
CREATE PROCEDURE [dbo].[AddParticipant] (
    @FirstName nvarchar(20),
    @LastName nvarchar(20),
    @AdressID int,
    @StudentCardID int,
    @ClientID int,
       @ParticipantID int = NULL OUT
)
       AS
       BEGIN
       SET NOCOUNT ON


       IF @FirstName is null
              THROW 2501, 'First name cant be null!', 1

       IF @LastName is null
              THROW 2501, 'LastName cant be null!', 1

       IF @AdressID is null
              THROW 2501, 'AdressID cant be null!', 1

       IF @ClientID is null
              THROW 2501, 'ClientID cant be null', 1


       INSERT INTO Participants (ClientID, FirstName, LastName, AdressID,
StudentCardID)
       VALUES (@ClientID, @FirstName, @LastName, @AdressID, @StudentCardID)
       SET @ParticipantID = SCOPE_IDENTITY();
END
GO
```

6.9 Dodawanie warsztatów

```
CREATE PROCEDURE [dbo].[AddWorkshop]
       @ConfDayID int,
       @Name nvarchar,
       @Seats int,
       @StartTime DateTime,
       @EndTime DateTime,
       @Price float,

       @WorkshopID int = NULL OUT
AS
BEGIN
       SET NOCOUNT ON
       BEGIN TRY
              BEGIN TRANSACTION


              IF @StartTime IS NULL
                     THROW 14,'@StartTime is null', 1
```

```
		IF @EndTime IS NULL
			THROW 14,'@EndTime is null', 1

		IF @EndTime < @StartTime
			THROW 14,'Workshop @EndTime < @StartTime', 1

		IF @Seats < 0
			THROW 14,'Workshop @Seats < 0', 1

		IF @Price < 0
			THROW 14,'Workshop @Price < 0', 1

		IF @ConfDayID IS NULL
			THROW 14,'@ConfDayID is null', 1

		IF @Name IS NULL
			THROW 14, 'Musisz podac nazwe warsztatu', 1

		IF @Price IS NULL
			THROW 14, 'Musisz podac cene konferencji', 1

		IF @Seats IS NULL
			THROW 14, 'Musisz podac liczbe miejsc na warsztacie (@Seats is
null)', 1

		INSERT Workshops(ConfDayID, Name, Seats, StartTime, EndTime, Price)
		VALUES (@ConfDayID, @Name, @Seats, @StartTime, @EndTime, @Price)
		SET @WorkshopID =  SCOPE_IDENTITY();

		COMMIT TRANSACTION
	END TRY

	BEGIN CATCH
		THROW
		ROLLBACK TRANSACTION
	END CATCH

END
GO
```

6.10 Dodawanie rezerwacji na konferencje

```
CREATE PROCEDURE [dbo].[AddConfDayReservation]
	@ClientID int,
	@ConfDayID int,
	@NumSeats int,
	@ReservationDate datetime,
	@PaidPrice float,
	@NumStudents int,

	@ConfDayReservationID int = NULL OUT
AS
BEGIN
	SET NOCOUNT ON
	BEGIN TRY
		BEGIN TRANSACTION


		IF @ClientID IS NULL
			THROW 14,'@ClientID is null in AddConfDayReservation', 1
		IF @ConfDayID IS NULL
			THROW 14,'@ConfDayID is null in AddConfDayReservation', 1

		IF @NumSeats IS NULL
			THROW 14,'@NumSeats is null in AddConfDayReservation', 1
		IF @ReservationDate IS NULL
			THROW 14,'@ReservationDate is null in AddConfDayReservation', 1

		IF @PaidPrice IS NULL
			THROW 14,'@PaidPrice is null in AddConfDayReservation', 1

		IF @NumStudents IS NULL
			THROW 14,'@NumStudents is null in AddConfDayReservation', 1

		IF @NumStudents > @NumSeats
			THROW 14,'@NumStudents > @NumSeats in AddConfDayReservation', 1

		IF @PaidPrice < 0
			THROW 14,'@PaidPrice < 0  in AddConfDayReservation', 1

		IF dbo.ConfDayFreeSeats(@ConfDayID) < @NumSeats
			THROW 14,'ConfDayFreeSeats(@ConfDayID) < @NumSeats in 
AddConfDayReservation', 1

		INSERT ConfDayReservations (ClientID, ConfDayID, NumSeats, 
ReservationDate, PaidPrice, NumStudents)
		VALUES (@ClientID, @ConfDayID, @NumSeats, @ReservationDate, @PaidPrice, 
@NumStudents)
		SET @ConfDayReservationID =  SCOPE_IDENTITY();

		COMMIT TRANSACTION
	END TRY

	BEGIN CATCH
		ROLLBACK TRANSACTION
		THROW
	END CATCH

END
GO
```

6.11 Dodawanie rejestracji na konferencje

```
CREATE PROCEDURE [dbo].[AddConfDayRegistration]
	@ParticipantID int,
	@ConfDayReservationID int,

	@ConfDayRegistrationID int = NULL OUT
AS
BEGIN
	SET NOCOUNT ON
	BEGIN TRY
		BEGIN TRANSACTION


		IF @ParticipantID IS NULL
			THROW 14,'@ParticipantID is null in AddConfDayRegistration', 1

		IF @ConfDayReservationID IS NULL
```

```
				THROW 14,'@ConfDayReservationID is null in 
AddConfDayRegistration', 1

			Declare @ClientID int = (Select TOP 1 ClientID from Participants where 
Participants.ParticipantID=@ParticipantID)

			IF (SELECT Count (*) from Clients
				Inner Join Participants on Participants.ClientID = 
Clients.ClientID
				inner join ConfDayReservations on ConfDayReservations.ClientID = 
Participants.ClientID
				where Participants.ParticipantID = @ParticipantID
				and ConfDayReservations.ConfDayReservationID = 
@ConfDayReservationID) = 0
				THROW 14, 'Nie istnieje takiego polaczenia @ParticipantID - 
@ConfDayReservationID w AddConfDayRegistration',1

			DECLARE @PriceToPay float = dbo.f_PriceToPayPerDay (@ClientID, 
@ConfDayReservationID)

			IF @PriceToPay >= (select PaidPrice from ConfDayReservations where 
ConfDayReservationID=@ConfDayReservationID)
				THROW 14,'Nie zostala dokonana wplata, uczestnik nie moze byc 
zarejestrowany na dany dzien konferencji.',1

			INSERT ConfDayRegistrations(ParticipantID, ConfDayReservationID)
			VALUES (@ParticipantID, @ConfDayReservationID)
			SET @ConfDayRegistrationID =  SCOPE_IDENTITY();

			COMMIT TRANSACTION
		END TRY

		BEGIN CATCH
			THROW
			ROLLBACK TRANSACTION
		END CATCH

END
GO
```

6.12 Dodawanie płatności

```
CREATE PROCEDURE [dbo].[AddPayment]
		@ConfDayReservationID int,
		@Payed float
AS
BEGIN
		SET NOCOUNT ON
		BEGIN TRY
			BEGIN TRANSACTION
			if @ConfDayReservationID IS NULL
			Throw 14, '@ConfDayReservationID IS NULL in AddPayment', 1

			if @Payed IS NULL
			Throw 14, '@Payed IS NULL in AddPayment', 1

			if @Payed <0
			Throw 14, '@Payed <0 in AddPayment', 1

			Declare @paidprice float = (select PaidPrice from ConfDayReservations 
where ConfDayReservationID = @ConfDayReservationID)
```

```
			UPDATE ConfDayReservations set PaidPrice = @paidprice+@Payed where
ConfDayReservationID = @ConfDayReservationID

			COMMIT TRANSACTION
		END TRY

		BEGIN CATCH
			ROLLBACK TRANSACTION
			THROW
		END CATCH

END
GO
```

6.13 Dodawanie rezerwacji na warsztaty

```
CREATE PROCEDURE [dbo].[AddWorkshopReservation]
		@WorkshopID int,
		@ConfDayReservationID int,
		@NumReservs int,
		@ReservationDate datetime,
		@NumStudents float,

		@WorkshopReservationID int = NULL OUT
AS
BEGIN
		SET NOCOUNT ON
		BEGIN TRY
			BEGIN TRANSACTION


			IF @WorkshopID IS NULL
				THROW 14,'@WorkshopID is null in AddWorkshopReservation', 1

			IF @ConfDayReservationID IS NULL
				THROW 14,'@ConfDayReservationID is null in
AddWorkshopReservation', 1

			IF @NumReservs IS NULL or @NumReservs < 1
				THROW 14,'@NumReservs IS NULL or @NumReservs < 1 in
AddWorkshopReservation', 1

			IF @ReservationDate IS NULL
				THROW 14,'@ReservationDate is null in AddWorkshopReservation', 1

			DECLARE @ConfDate date = (select ConferenceDays.Date from Workshops
								inner join ConferenceDays on
ConferenceDays.ConfDayID = Workshops.ConfDayID
								where
Workshops.WorkshopID=@WorkshopID )

			IF @ReservationDate  > @ConfDate
				THROW 14, '@ReservationDate  > @ConfDate in
AddWorkshopReservation', 1


			IF @NumStudents > @NumReservs
				THROW 14,'@NumStudents > @NumReservs in AddWorkshopReservation', 1
		IF @NumStudents < 0
			THROW 14,'@NumStudents < 0 in AddWorkshopReservation', 1

		IF @NumReservs > dbo.WorkshopFreeSeats(@WorkshopID)
			THROW 14,'@NumReservs > dbo.WorkshopFreeSeats(@WorkshopID) in
AddWorkshopReservation', 1

		Declare @NumConfSeatsReserv int = (select NumSeats from
ConfDayReservations where ConfDayReservationID = @ConfDayReservationID)

		If @NumConfSeatsReserv < @NumReservs
			THROW 14,'@NumConfSeatsReserv < @NumReservs in
AddWorkshopReservation', 1

		INSERT WorkshopReservations(WorkshopID, ConfDayReservationID, NumReservs,
ReservationDate, NumStudents)
		VALUES (@WorkshopID, @ConfDayReservationID, @NumReservs,
@ReservationDate, @NumStudents)
		SET @WorkshopReservationID =  SCOPE_IDENTITY();

		COMMIT TRANSACTION
	END TRY

	BEGIN CATCH
		THROW
		ROLLBACK TRANSACTION
	END CATCH

END
GO
```

6.14 Dodawanie rejestracji na warsztaty

```
CREATE PROCEDURE [dbo].[AddWorkshopRegistration]
	@WorkshopReservID int,
	@ParticipantID int,
	@ConfDayRegistrationID int,
	@WorkShopRegistrationID int = NULL OUT
AS
BEGIN
	SET NOCOUNT ON
	BEGIN TRY
		BEGIN TRANSACTION


		IF @ParticipantID IS NULL
			THROW 14,'@ParticipantID is null in AddWorkshopRegistration', 1

		IF @WorkshopReservID IS NULL
			THROW 14,'@WorkshopReservID is null in AddWorkshopRegistration', 1

		IF @ConfDayRegistrationID IS NULL
			THROW 14,'@ConfDayRegistrationID is null in
AddWorkshopRegistration', 1


		Declare @ClientID int = (Select TOP 1 ClientID from Participants where
Participants.ParticipantID=@ParticipantID)

		IF (SELECT Count (*) from Clients
                    Inner Join Participants on Participants.ClientID =
Clients.ClientID
                    inner join ConfDayReservations on ConfDayReservations.ClientID =
Participants.ClientID
                    inner join WorkshopReservations on
WorkshopReservations.ConfDayReservationID=ConfDayReservations.ConfDayReservationID
                    inner join ConfDayRegistrations on
ConfDayReservations.ConfDayReservationID = ConfDayRegistrations.ConfDayReservationID
                    where Participants.ParticipantID = @ParticipantID
                    and ConfDayRegistrations.ConfDayRegistrationID =
@ConfDayRegistrationID
                    and WorkshopReservations.WorkshopReservID = @WorkshopReservID
                    ) = 0
                    THROW 14, 'Nie istnieje takiego polaczenia @ParticipantID -
@WorkshopReservID w AddWorkshopRegistration',1

             if dbo.WorkshopReservationFreeSeats(@WorkshopReservID) < 1
                    THROW 14, 'WorkshopReservationFreeSeats(@WorkshopReservID) = 0 in
AddWorkshopRegistration', 1

             INSERT WorkshopRegistrations(WorkshopReservID,
ParticipantID,ConfDayRegistrationID)
             VALUES (@WorkshopReservID, @ParticipantID,@ConfDayRegistrationID)
             SET @WorkShopRegistrationID =  SCOPE_IDENTITY();

             COMMIT TRANSACTION
      END TRY

      BEGIN CATCH
             ROLLBACK TRANSACTION
             THROW
      END CATCH

END
GO
```

7. Procedury modyfikujące dane

7.1 Anulowanie rezerwacji dnia konferencji

```
CREATE PROCEDURE [dbo].[CancelClientConfDayReservation] (
       @ConfDayReservationID int
)

AS BEGIN

       SET NOCOUNT ON

       BEGIN TRY

              BEGIN TRANSACTION

                            IF @ConfDayReservationID is null
                            THROW 2500, 'ConfDayReservationID cant be null!', 1

                            UPDATE ConfDayReservations
                                   SET Cancelled = 1
```

```
                                WHERE ConfDayReservationID = @ConfDayReservationID

                                --tutaj wszystkie rezerwacje na warszaty w danym dniu
zostana anulowane
                                --a na dodatek usuniete wszystkie rejestracje na warsztaty
w danym dniu

                                EXEC CancelClientWorkshopReservations
@ConfDayReservationID;

                                DELETE FROM ConfDayRegistrations
                                WHERE ConfDayReservationID = @ConfDayReservationID

                COMMIT TRANSACTION

        END TRY

        BEGIN CATCH

                ROLLBACK TRANSACTION
                THROW

        END CATCH

END

GO
```

7.2 Anulowanie rezerwacji warsztatu klienta

```
CREATE PROCEDURE [dbo].[CancelClientWorkshopReservation] (
        @ConfDayReservationID int,
        @WorkshopID int
)

AS BEGIN
        SET NOCOUNT ON

                BEGIN TRY
                        BEGIN TRANSACTION

                                IF @ConfDayReservationID is null
                                THROW 2500, 'ClientID cant be null!', 1



                                UPDATE WorkshopReservations
                                        SET Cancelled = 1
                                WHERE ConfDayReservationID = @ConfDayReservationID and
WorkshopID = @WorkshopID


                                DELETE WRS from WorkshopRegistrations WRS
                                        left join WorkshopReservations w on
w.WorkshopReservID = WRS.WorkshopReservID
                                WHERE w.ConfDayReservationID = @ConfDayReservationID and
w.WorkshopID = @WorkshopID
```

```
                        COMMIT TRANSACTION

                END TRY

                BEGIN CATCH

                        ROLLBACK TRANSACTION
                        THROW

                END CATCH
END

GO
```

7.3 Anulowanie wszystkich rezerwacji warsztatów dla klienta z danego dnia

```
CREATE PROCEDURE [dbo].[CancelClientWorkshopReservations] (
        @ConfDayReservationID int
)

AS BEGIN
        SET NOCOUNT ON

                BEGIN TRY
                        BEGIN TRANSACTION

                                IF @ConfDayReservationID is null
                                THROW 2500, 'ConfDayReservationID cant be null!', 1


                                UPDATE WorkshopReservations
                                        SET Cancelled = 1
                                WHERE ConfDayReservationID = @ConfDayReservationID


                                DELETE WRS from WorkshopRegistrations WRS
                                WHERE WRS.WorkshopReservID = (select wr.WorkshopReservID
from WorkshopReservations wr
                                        inner join ConfDayReservations on
ConfDayReservations.ConfDayReservationID = @ConfDayReservationID
                                )

                        COMMIT TRANSACTION

                END TRY

                BEGIN CATCH

                        ROLLBACK TRANSACTION
                        THROW

                END CATCH
END

GO
```

7.4 Anulowanie jednej rejestracji na warsztat

```
CREATE PROCEDURE [dbo].[CancelParticipantWorkshopRegistration] (
	@ParticipantID int,
	@WorkshopRegID int
)

AS BEGIN

	SET NOCOUNT ON
	BEGIN TRY
		BEGIN TRANSACTION

					IF @ParticipantID is null
						THROW 2500, 'ParticipantID cant be null!', 1

					DELETE b From WorkshopRegistrations b
					LEFT JOIN Participants on b.ParticipantID =
@ParticipantID and b.WorkshopRegID = @WorkshopRegID


		COMMIT TRANSACTION
	END TRY

	BEGIN CATCH
		ROLLBACK TRANSACTION
		THROW
	END CATCH
END

GO
```

7.5 Anulowanie rezerwacji wszystkich warsztatów przez uczestnika

```
CREATE PROCEDURE [dbo].[CancelParticipantWorkshopRegistrations] (
	@ParticipantID int
)

AS BEGIN

	SET NOCOUNT ON
	BEGIN TRY
		BEGIN TRANSACTION

					IF @ParticipantID is null
						THROW 2500, 'ParticipantID cant be null!', 1


					DELETE b From WorkshopRegistrations b
					where b.ParticipantID = @ParticipantID

		COMMIT TRANSACTION
	END TRY

	BEGIN CATCH
		ROLLBACK TRANSACTION
		THROW
	END CATCH
END

GO
```

7.6 Modyfikowanie danych klienta

```sql
CREATE PROCEDURE [dbo].[ModifyClient] (
	@ClientID int,
	@CompanyID int,
	@Login nvarchar (20) ,
	@Password nvarchar (255),
	@Mail nvarchar (60)
)

AS BEGIN
	SET NOCOUNT ON

		BEGIN TRY
			BEGIN TRANSACTION

				IF @ClientID is null
				THROW 2500,'ClientID cant be null!', 1

				IF @Login is null
				THROW 2500, 'Login cant be null!', 1

				IF @Password is null
				THROW 2500, 'Password cant be null!', 1

				IF @Mail is null
				THROW 2500, 'Mail cant be null!', 1

				UPDATE Clients
				SET
					CompanyID = @CompanyID,
					Login = @Login,
					Password = @Password,
					Mail = @Mail
				WHERE ClientID = @ClientID

				IF @@ROWCOUNT = 0
					THROW 2500, 'You provided incorrect ClientID, Client
with such ID doesnt exist!', 1

			COMMIT TRANSACTION
		END TRY

		BEGIN CATCH
			ROLLBACK TRANSACTION
			THROW
		END CATCH
END

GO
```

7.7 Modyfikowanie danych firmy

```sql
CREATE PROCEDURE [dbo].[ModifyCompany] (
	@CompanyID int,
	@CompanyName nvarchar (20),
	@AdressID int
)
```

```
AS BEGIN
	SET NOCOUNT ON

	BEGIN TRY
		BEGIN TRANSACTION

			IF @CompanyID is null
			THROW 2500, 'CompanyID cant be null!', 1

			IF @CompanyName is null
			THROW 2500, 'CompanyName cant be null!', 1

			IF @AdressID is null
			THROW 2500, 'AdressID cant be null!', 1

			UPDATE Companies
			SET
				CompanyName = @CompanyName,
				AdressID = @AdressID

			WHERE CompanyID = @CompanyID

			IF @@ROWCOUNT = 0
				THROW 2500, 'You provided incorrect CompanyID, Company with 
such ID doesnt exist!', 1

		COMMIT TRANSACTION

	END TRY

	BEGIN CATCH
		ROLLBACK TRANSACTION
		THROW
	END CATCH

END

GO
```

7.8 Modyfikowanie Konferencji

```
CREATE PROCEDURE ModifyConferances (
	@ConferenceID int,
	@Name varchar (50),
	@AdressID int,
	@StudentDiscount int
)

AS BEGIN
	SET NOCOUNT ON

	BEGIN TRY
		BEGIN TRANSACTION

			IF @ConferenceID is null
			THROW 2500, 'ConferenceID cant be null!', 1
```

```
				IF @Name is null
				THROW 2500, 'Name cant be null!', 1

				IF @AdressID is null
				THROW 2500, 'AdressID cant be null!', 1

				UPDATE Conference
				SET
					Name = @Name,
					AdressID = @AdressID,
					StudentDiscount = @StudentDiscount
				WHERE ConferenceID = @ConferenceID

			COMMIT TRANSACTION
		END TRY

		BEGIN CATCH
			ROLLBACK TRANSACTION
			THROW
		END CATCH

END

GO
```

7.9 Modyfikowanie uczestnika

```
CREATE PROCEDURE ModifyParticipant (
	@ParticipantID int,
	@FirstName nvarchar (20),
	@LastName nvarchar (20),
	@AdressID int,
	@StudentCardID int
)

AS BEGIN
	SET NOCOUNT ON

		BEGIN TRY
			BEGIN TRANSACTION

				IF @ParticipantID is null
					THROW 2500, 'ParticipantID cant be null', 1

				IF @FirstName is null
					THROW 2500, 'FirstName cant be null', 1

				IF @LastName is null
					THROW 2500, 'LastName cant be null', 1

				IF @AdressID is null
					THROW 2500, 'AdressID cant be null', 1

				IF @StudentCardID is null
					THROW 2500, 'StudentCardID cant be null', 1

				UPDATE Participants
				SET
```

```
                                        FirstName = @FirstName,
                                        LastName = @LastName,
                                        AdressID = @AdressID,
                                        StudentCardID = @StudentCardID
                                WHERE ParticipantID = @ParticipantID

                                IF @@ROWCOUNT = 0
                                        THROW 2500, 'You provided incorrect ParticipantID,
Participant with such ID doesnt exist!', 1

                                COMMIT TRANSACTION
                END TRY

                BEGIN CATCH
                        ROLLBACK TRANSACTION
                        THROW
                END CATCH

END

GO
```

7.10 Modyfikowanie rezerwacji warsztatu

```
CREATE PROCEDURE [dbo].[ModifyWorkshopReservation]
        @WorkshopReservID int,
        @WorkshopID int,
        @NumReservs int,
        @NumStudents float

AS
BEGIN
        SET NOCOUNT ON
        BEGIN TRY
                BEGIN TRANSACTION


                IF @WorkshopID IS NULL
                        THROW 14,'@WorkshopID is null in AddWorkshopReservation', 1


                IF @NumReservs IS NULL or @NumReservs < 1
                        THROW 14,'@NumReservs IS NULL or @NumReservs < 1 in
AddWorkshopReservation', 1

                IF @NumStudents > @NumReservs
                        THROW 14,'@NumStudents > @NumReservs in AddWorkshopReservation', 1

                IF @NumStudents < 0
                        THROW 14,'@NumStudents < 0 in AddWorkshopReservation', 1

                IF @NumReservs > dbo.WorkshopFreeSeats(@WorkshopID)
                        THROW 14,'@NumReservs > dbo.WorkshopFreeSeats(@WorkshopID) in
AddWorkshopReservation', 1

                Declare @ConfDayReservationID int = (select ConfDayReservationID from
WorkshopReservations where WorkshopReservations.WorkshopReservID = @WorkshopReservID)

                Declare @NumConfSeatsReserv int = (select NumSeats from
ConfDayReservations where ConfDayReservationID = @ConfDayReservationID)
```

```
              If @NumConfSeatsReserv < @NumReservs
                     THROW 14,'@NumConfSeatsReserv < @NumReservs in
AddWorkshopReservation', 1

              IF dbo.WorkshopFreeSeats(@WorkshopID)<@NumReservs - (select NumReservs
from WorkshopReservations where WorkshopReservID=@WorkshopReservID)
                     THROW 14, 'Za malo wolnych miejsc na warsztacie w
AddWorkshopReservation',1

              UPDATE WorkshopReservations SET
                     NumReservs=@NumReservs,
                     ReservationDate=GETDATE(),
                     NumStudents=@NumStudents
              WHERE WorkshopReservID=WorkshopReservID


              COMMIT TRANSACTION
       END TRY

       BEGIN CATCH
              ROLLBACK TRANSACTION
              THROW
       END CATCH

END
GO
```

8. Triggery

8.1 Walidacja rejestracji na konferencje

```
CREATE TRIGGER ConferenceRegistrationsValidate

ON ConfDayRegistrations

AFTER INSERT, UPDATE

AS BEGIN

       DECLARE @ConfDayRegistrationID int, @ConfDayReservationID int

       IF (SELECT COUNT (*) FROM INSERTED) = 0
              RETURN
       IF (SELECT COUNT (*) FROM INSERTED) = 1

              BEGIN

                     SELECT @ConfDayRegistrationID = ConfDayRegistrationID,
                            @ConfDayReservationID = ConfDayReservationID
                     FROM INSERTED

                     IF NOT EXISTS(
                            SELECT ConfDayReservationID from ConfDayReservations
                            where ConfDayReservationID = @ConfDayReservationID
                     )
```

```
                    BEGIN

                          RAISERROR ('THERE IS NO RESERVATION ASSOCIATED WITH THIS 
REGISTRATION!!',2500, 1)
                          ROLLBACK TRANSACTION

                    END

                    IF dbo.ConfDayReservationFreeSeats(@ConfDayReservationID) <= 0
                    BEGIN
                          RAISERROR ('YOU DONT HAVE ENOUGH PLACES RESERVED!!',2500, 
1)
                          ROLLBACK TRANSACTION
                    END

              END
       ELSE

              BEGIN
                          RAISERROR('YOU CANT INSERT OR UPDATE MORE THAN ONE RECORD 
AT THE SAME TIME!!!',2500,1)
                          ROLLBACK TRANSACTION
              END
END


ALTER TABLE ConfDayRegistrations ENABLE TRIGGER ConferenceRegistrationsValidate
```

8.2 Walidacja rejestracji na warsztaty

```
IF OBJECT_ID ('WorkshopsRegistrationValidate') is not null
DROP TRIGGER WorkshopsRegistrationValidate;
GO


CREATE TRIGGER WorkshopsRegistrationValidate
-- przy dodawaniu rejestracji na warsztat sprawdza, czy participant nie
-- zarejestrowal sie na inny warsztat w tym czasie
ON WorkshopRegistrations

AFTER INSERT, UPDATE

AS BEGIN
       DECLARE @WorkshopRegID int

       IF (select count(*) from INSERTED) = 0
              RETURN
       IF (select count(*) from INSERTED) = 1

              BEGIN

                    SELECT @WorkshopRegID = WorkshopRegID
                    FROM INSERTED

                          DECLARE
                          @EndTime datetime,
                          @StartTime datetime,
                          @WorkshopReservID int,
                          @ParticipantID int,
                          @WorkshopID int

                          SELECT
```

```
                           @WorkshopID = W.WorkshopID,
                           @StartTime = W.StartTime,
                           @EndTime = W.EndTime,
                           @WorkshopReservID = WR.WorkshopReservID,
                           @ParticipantID = WR.ParticipantID

                           FROM WorkshopRegistrations WR
                                  JOIN WorkshopReservations WV on WV.WorkshopReservID
= WR.WorkshopReservID
                                  JOIN Workshops W on W.WorkshopID = WV.WorkshopID
                           WHERE WR.WorkshopRegID = @WorkshopRegID

                           IF EXISTS
                            (
                                  SELECT W.WorkshopID
                                  FROM Workshops W
                                  WHERE ((W.StartTime BETWEEN @StartTime AND @EndTime)
OR (W.EndTime BETWEEN @StartTime AND @EndTime))
                                  AND W.WorkshopID IN (SELECT W.WorkshopID
                                                                            FROM
Workshops W
                                                                            JOIN
WorkshopReservations WR on W.WorkshopID = WR.WorkshopID
                                                                            JOIN
WorkshopRegistrations WRs on WR.WorkshopReservID = WRs.WorkshopReservID
                                                                      WHERE
WRs.ParticipantID = @ParticipantID and WRs.WorkshopRegID <> @WorkshopRegID

                                         )
                           )

                           BEGIN

                           RAISERROR ('THIS PARTICIPANT HAS ANOTHER WORKSHOP AT THIS
TIME!',2500, 1)
                           ROLLBACK TRANSACTION

                           END

                           IF EXISTS (
                                  SELECT WorkshopID from WorkshopReservations w
                                  JOIN WorkshopRegistrations wr on wr.ParticipantID =
@ParticipantID
                                  where WorkshopID = @WorkshopID and
w.WorkshopReservID <> @WorkshopReservID
                           )

                           BEGIN

                                  RAISERROR ('THIS PARTICIPANT IS ALREADY REGISTERED
ON THIS WORKSHOP!',2500, 1)
                                  ROLLBACK TRANSACTION

                           END

                           IF NOT EXISTS (
                                  SELECT WR.ConfDayRegistrationID
                                  FROM WorkshopRegistrations WR
                                         JOIN ConfDayRegistrations CDR on
CDR.ConfDayRegistrationID = WR.ConfDayRegistrationID
```

```
                                            WHERE WR.WorkshopRegID = @WorkshopRegID and
WR.ParticipantID = @ParticipantID
                                    )
                                    BEGIN

                                    RAISERROR ('THIS PARTICIPANT IS NOT REGISTERED ON
CONFERENCE!!',2500,1)
                                    ROLLBACK TRANSACTION

                                    END

                  END
            ELSE
                  BEGIN

                  RAISERROR ('You cant insert more than one record ar once!',2500, 1)
                  ROLLBACK TRANSACTION

                  END

END

GO

ALTER TABLE WorkshopRegistrations ENABLE TRIGGER WorkshopsRegistrationValidate
```