AGH

AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA
W KRAKOWIE

Podstawy Baz Danych:

System zarządzania konferencjami

Przemysław Lechowicz, Hubert Wolak

Styczeń 2020

SPIS TREŚCI

1. Założenia projektu:

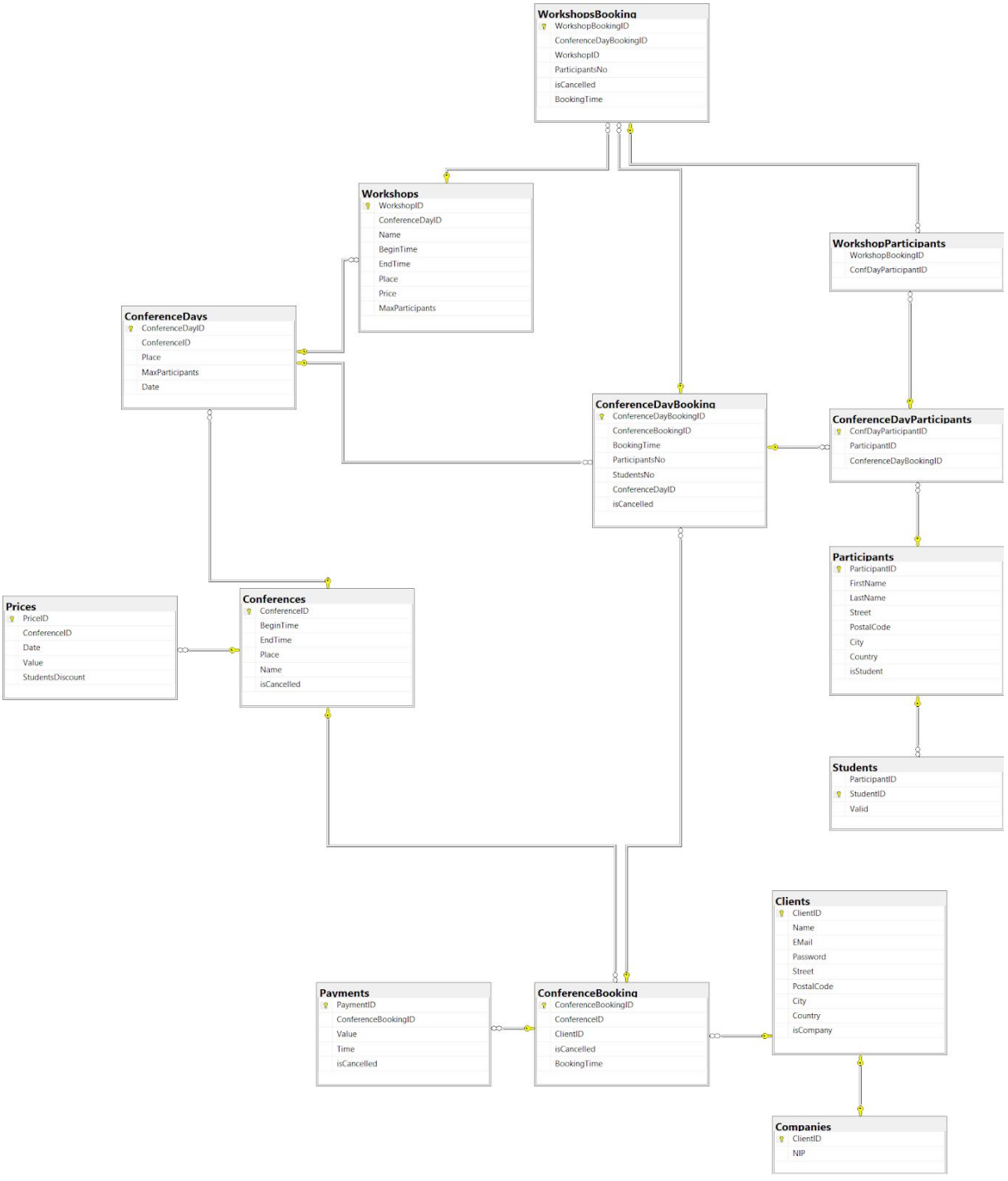
Baza danych obsługuje system informatyczny firmy organizującej konferencje głównie dla studentów i pracowników firm branży IT. Konferencje widniejącą w bazie danych, mogą trwać kilka dni, niekoniecznie tworzących ciągły przedział (np. wyłączając weekendy). Klientami przedsiębiorstwa są zarówno firmy, które zgłaszają uczestników, jak i osoby fizyczne. Uczestnikami mogą być tylko osoby fizyczne.

Z konferencją mogą być związane warsztaty na które uczestnicy konferencji mogą się dodatkowo rejestrować. Wymogiem uczestnictwa w warsztatach danego dnia jest udział w konferencji, która odbyła się tego dnia. Uczestnik może być zarejestrowany maksymalnie na jedne zajęcia, które odbywają się w tym samym czasie, co jest sprawdzane przez bazę danych.

Konferencje mogą być płatne i darmowe. Opłaty za rezerwacje uczestnictwa w konferencji mogą być progowe – cena może się zmieniać każdego dnia.

Opłaty za uczestnictwo w warsztatach są stałe. Istnieje możliwość uzyskania zniżki dla studentów, która może być różna w zależności od konferencji i warsztatów.

2. Schemat


WorkshopsBooking
WorkshopBookingID
ConferenceDayBookingID
WorkshopID
ParticipantsNo
isCancelled
BookingTime
Workshops
WorkshopID
ConferenceDayID
Name
BeginTime
EndTime
Place
Price
MaxParticipants
WorkshopParticipants
WorkshopBookingID
ConfDayParticipantID
ConferenceDays
ConferenceDayID
ConferenceID
Place
MaxParticipants
Date
ConferenceDayBooking
ConferenceDayBookingID
ConferenceBookingID
BookingTime
ParticipantsNo
StudentsNo
ConferenceDayID
isCancelled
ConferenceDayParticipants
ConfDayParticipantID
ParticipantID
ConferenceDayBookingID
Participants
ParticipantID
FirstName
LastName
Street
PostalCode
City
Country
isStudent
Prices
PriceID
ConferenceID
Date
Value
StudentsDiscount
Conferences
ConferenceID
BeginTime
EndTime
Place
Name
isCancelled
Students
ParticipantID
StudentID
Valid
Clients
ClientID
Name
EMail
Password
Street
PostalCode
City
Country
isCompany
Payments
PaymentID
ConferenceBookingID
Value
Time
isCancelled
ConferenceBooking
ConferenceBookingID
ConferenceID
ClientID
isCancelled
BookingTime
Companies
ClientID
NIP

3. Tabele

3.1. ConferenceDayBooking

Tabela przechowuje dane dotyczące rezerwacji na konkretny dzień konferencji. Zawiera:

3.1.1. **ConferenceDayBookingID** - identyfikator rezerwacji dnia konferencji

3.1.2. **ConferenceBookingID** - identyfikator powiązanej rezerwacji konferencji

3.1.3. **BookingTime** - data i godzina dokonania rezerwacji

3.1.4. **ParticipantsNo** - zadeklarowana liczba zarezerwowanych miejsc, musi być większa od 0

3.1.5. **StudentsNo** - zadeklarowana liczba zarezerwowanych miejsc przez studentów, musi być większa lub równa **ParticipantsNo**

3.1.6. **ConferenceDayID** - identyfikator powiązanego z rezerwacją dnia konferencji

3.1.7. **isCancelled** - flaga określająca, czy rezerwacja jest aktywna (0) lub anulowana (1), domyślnie 0

```
CREATE TABLE [dbo].[ConferenceDayBooking]
(
   [ConferenceDayBookingID] [int]IDENTITY (1,1)NOT NULL,
   [ConferenceBookingID]    [int]              NOT NULL,
   [BookingTime]            [datetime]         NOT NULL,
   [ParticipantsNo]         [int]              NOT NULL,
   [StudentsNo]             [int]              NOT NULL,
   [ConferenceDayID]        [int]              NOT NULL,
   [isCancelled]            [bit]              NOT NULL DEFAULT 0,
   CONSTRAINT [PK_ConferenceDayBooking] PRIMARY KEY CLUSTERED
       (
        [ConferenceDayBookingID] ASC
           ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
           ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
ALTER TABLE [dbo].[ConferenceDayBooking]
   WITH CHECK ADD CONSTRAINT
[FK_ConferenceDayBooking_ConferenceBooking] FOREIGN
       KEY ([ConferenceBookingID])
       REFERENCES [dbo].[ConferenceBooking]
([ConferenceBookingID])
```

```
ALTER TABLE [dbo].[ConferenceDayBooking]
   CHECK CONSTRAINT [FK_ConferenceDayBooking_ConferenceBooking]
ALTER TABLE [dbo].[ConferenceDayBooking]
   WITH CHECK ADD CONSTRAINT
[FK_ConferenceDayBooking_ConferenceDays] FOREIGN
       KEY ([ConferenceDayID])
       REFERENCES [dbo].[ConferenceDays] ([ConferenceDayID])
ALTER TABLE [dbo].[ConferenceDayBooking]
   CHECK CONSTRAINT [FK_ConferenceDayBooking_ConferenceDays]
ALTER TABLE [dbo].[ConferenceDayBooking]
   WITH CHECK ADD CONSTRAINT [CK_ParticipantsNo] CHECK
       (([ParticipantsNo] > (0)))
ALTER TABLE [dbo].[ConferenceDayBooking]
   CHECK CONSTRAINT [CK_ParticipantsNo]
ALTER TABLE [dbo].[ConferenceDayBooking]
   WITH CHECK ADD CONSTRAINT [CK_ParticipantsStudentsNo] CHECK
       (([ParticipantsNo] >= [StudentsNo]))
ALTER TABLE [dbo].[ConferenceDayBooking]
   CHECK CONSTRAINT [CK_ParticipantsStudentsNo]
ALTER TABLE [dbo].[ConferenceDayBooking]
   WITH CHECK ADD CONSTRAINT [CK_StudentsNo] CHECK (([StudentsNo]
>= (0)))
ALTER TABLE [dbo].[ConferenceDayBooking]
   CHECK CONSTRAINT [CK_StudentsNo]
```

3.2. Participants

Tabela participants to baza wszystkich uczestników konferencji. Zawiera:

3.2.1. **ParticipantID** - identyfikator uczestnika
3.2.2. **FirstName** - imię
3.2.3. **LastName** - nazwisko
3.2.4. **Street** - ulica
3.2.5. **PostalCode** - kod pocztowy, postaci XX-XXX, XXXXXX, XXXXXXX, gdzie X to cyfra
3.2.6. **City** - miejscowość
3.2.7. **Country** - kraj
3.2.8. **isStudent** - flaga określająca czy uczestnik jest studentem

```
CREATE TABLE [dbo].[Participants]
(
   [ParticipantID] [int] IDENTITY (1,1) NOT NULL,
   [FirstName]     [nvarchar](50)       NOT NULL,
   [LastName]      [nvarchar](50)       NOT NULL,
   [Street]        [nvarchar](50)       NULL,
   [PostalCode]    [varchar](10)        NULL,
   [City]          [nvarchar](50)       NULL,
   [Country]       [nvarchar](50)       NULL,
   [isStudent]     [bit]                NOT NULL DEFAULT 0,
   CONSTRAINT [PK_Participants] 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]
) ON [PRIMARY]

ALTER TABLE [dbo].[Participants]
WITH CHECK ADD CONSTRAINT [CK_Participants_PostalCode]
CHECK (([PostalCode] LIKE '[0-9][0-9]-[0-9][0-9][0-9]'
OR [PostalCode] LIKE '[0-9][0-9][0-9][0-9][0-9]'
OR [PostalCode] LIKE '[0-9][0-9][0-9][0-9][0-9][0-9]'))
```

3.3. Student

Tabela students przechowuje informacje o studentach uczestniczących w konferencjach. Zawiera:

3.3.1. **ParticipantID** - identyfikator uczestnika
3.3.2. **StudentID** - identyfikator studenta
3.3.3. **Valid** - data ważności legitymacji studenckiej

```
CREATE TABLE [dbo].[Student]
(
   [ParticipantID] [INT]      NOT NULL,
   [StudentID]     [CHAR](10) NOT NULL,
   [Valid]         [Date]     NOT NULL,
   CONSTRAINT [UQ_Student_ParticipantID] 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]
ALTER TABLE [dbo].[Student]
   WITH CHECK ADD CONSTRAINT [FK_Participants_Student] FOREIGN KEY
([ParticipantID])
       REFERENCES [dbo].[Participants] ([ParticipantID])
ALTER TABLE [dbo].[Student]
   CHECK CONSTRAINT [FK_Participants_Student]
```

3.4. WorkshopsBooking

Tabela WorkshopsBooking specyfikuje ile osób i na jaki warsztat chce zgłosić klient, który dokonał już rezerwacji na dany dzień konferencji. Zawiera:

3.4.1. **WorkshopBookingID** - identyfikator rezerwacji warsztatu

3.4.2. **ConferenceDayBookingID** - identyfikator powiązanej z nią rezerwacji dnia konferencji

3.4.3. **WorkshopID** - identyfikator warsztatu

3.4.4. **ParticipantsNo** - zadeklarowana liczba uczestników, musi być wieksza od 0

3.4.5. **BookingTime** - czas rezerwacji

3.4.6. **isCancelled** - flaga określająca, czy rezerwacja jest aktywna (0) lub anulowana (1), domyślnie 0

```
CREATE TABLE [dbo].[WorkshopsBooking]
(
   [WorkshopBookingID]      [int]IDENTITY (1,1)NOT NULL,
   [ConferenceDayBookingID] [int]              NOT NULL,
   [WorkshopID]             [int]              NOT NULL,
   [ParticipantsNo]         [int]              NOT NULL,
   [isCancelled]            [bit]              NOT NULL DEFAULT 0,
   [BookingTime]            [datetime]         NOT NULL,
   CONSTRAINT [PK_WorkshopsBooking] PRIMARY KEY CLUSTERED
       (
        [WorkshopBookingID] ASC
           ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
```

```
            ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
ALTER TABLE [dbo].[WorkshopsBooking]
   WITH CHECK ADD CONSTRAINT
[FK_WorkshopsBooking_ConferenceDayBooking]
       FOREIGN KEY ([ConferenceDayBookingID])
           REFERENCES [dbo].[ConferenceDayBooking]
([ConferenceDayBookingID])
ALTER TABLE [dbo].[WorkshopsBooking]
   CHECK CONSTRAINT [FK_WorkshopsBooking_ConferenceDayBooking]
ALTER TABLE [dbo].[WorkshopsBooking]
   WITH CHECK ADD CONSTRAINT [FK_WorkshopsBooking_Workshops]
FOREIGN
       KEY ([WorkshopID])
       REFERENCES [dbo].[Workshops] ([WorkshopID])
ALTER TABLE [dbo].[WorkshopsBooking]
   CHECK CONSTRAINT [FK_WorkshopsBooking_Workshops]
ALTER TABLE [dbo].[WorkshopsBooking]
   WITH CHECK ADD CONSTRAINT [CK_ParticipantsNo_WorkshopsBooking]
       CHECK (([ParticipantsNo] > (0)))
ALTER TABLE [dbo].[WorkshopsBooking]
   CHECK CONSTRAINT [CK_ParticipantsNo_WorkshopsBooking]
```

3.5. WorkshopParticipants

Tabela participants to baza wszystkich uczestników warsztatu. Zawiera:

3.5.1. **WorkshopBookingID** - identyfikator powiązanej rezerwacji warsztatu

3.5.2. **ConferenceDayParticipantID** - identyfikator uczestnika dnia konferencji

```
CREATE TABLE [dbo].[WorkshopParticipants]
(
   [WorkshopBookingID]    [int] NOT NULL,
   [ConferenceDayParticipantID] [int] NOT NULL
) ON [PRIMARY]
ALTER TABLE [dbo].[WorkshopParticipants]
   WITH CHECK ADD CONSTRAINT
       [FK_WorkshopParticipants_ConferenceDayParticipants] FOREIGN
KEY ([ConferenceDayParticipantID])
```

```
            REFERENCES [dbo].[ConferenceDayParticipants]
([ConferenceDayParticipantID])
ALTER TABLE [dbo].[WorkshopParticipants]
   CHECK CONSTRAINT
[FK_WorkshopParticipants_ConferenceDayParticipants]
ALTER TABLE [dbo].[WorkshopParticipants]
   WITH CHECK ADD CONSTRAINT
       [FK_WorkshopParticipants_WorkshopsBooking] FOREIGN KEY
([WorkshopBookingID])
            REFERENCES [dbo].[WorkshopsBooking]
([WorkshopBookingID])
ALTER TABLE [dbo].[WorkshopParticipants]
   CHECK CONSTRAINT [FK_WorkshopParticipants_WorkshopsBooking]
```

3.6. Workshops

Tabela workshops przechowuje listę wszystkich warsztatów odbywających się ramach konferencji. Zawiera:

3.6.1. **WorkshopID** - identyfikator warsztatu
3.6.2. **ConferenceDayID** - identyfikator powiązanego z warsztatem dnia konferencji
3.6.3. **Name** - nazwę warsztatu
3.6.4. **BeginTime** - godzinę rozpoczęcia warsztatu
3.6.5. **EndTime** - godzinę zakończenia warsztatu
3.6.6. **Place** - lokalizację warsztatu
3.6.7. **Price** - cenę brutto warsztatu dla jednego uczestnika
3.6.8. **MaxParticipants** - limit uczestników, musi być większy od 0

```
CREATE TABLE [dbo].[Workshops]
(
   [WorkshopID]      [int] IDENTITY (1,1) NOT NULL,
   [ConferenceDayID] [int]                NOT NULL,
   [Name]            [nvarchar](50)       NOT NULL,
   [BeginTime]       [time](7)            NOT NULL,
   [EndTime]         [time](7)            NOT NULL,
   [Place]           [nvarchar](50)       NULL,
   [Price]           [money]              NULL,
   [MaxParticipants] [int]                NOT NULL,
   CONSTRAINT [PK_Workshops] 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]
) ON [PRIMARY]
ALTER TABLE [dbo].[Workshops]
   WITH CHECK ADD CONSTRAINT [FK_Workshops_ConferenceDays] FOREIGN
       KEY ([ConferenceDayID])
       REFERENCES [dbo].[ConferenceDays] ([ConferenceDayID])
ALTER TABLE [dbo].[Workshops]
   CHECK CONSTRAINT [FK_Workshops_ConferenceDays]
ALTER TABLE [dbo].[Workshops]
   WITH CHECK ADD CONSTRAINT [CK_ParticipansLimit_Workshops] CHECK
       (([MaxParticipants] > (0)))
ALTER TABLE [dbo].[Workshops]
   CHECK CONSTRAINT [CK_ParticipansLimit_Workshops]
```

3.7. ConferenceDays

Tabela ConferenceDays zawiera informacje o poszczególnych dniach danej konferencji. Zawiera:

3.7.1. **ConferenceDayID** - identyfikator konkretnego dnia konferencji

3.7.2. **ConferenceID** - identyfikator tej konferencji, do której należy dany dzień

3.7.3. **Place** - miejsce, w którym odbywa się dzień konferencji, określona bardziej szczegółowo od lokalizacji związanej z konferencją

3.7.4. **MaxParticipants** - limit uczestników dnia konferencji, musi być większe od 0

3.7.5. **date** - oraz datę dnia konferencji

```
CREATE TABLE [dbo].[ConferenceDays]
(
   [ConferenceDayID] [int] IDENTITY (1,1) NOT NULL,
   [ConferenceID]    [int]                NOT NULL,
   [Place]           [nvarchar](50)       NULL,
   [MaxParticipants] [int]                NOT NULL,
   [Date]            [date]               NOT NULL,
   CONSTRAINT [PK_ConferenceDays] PRIMARY KEY CLUSTERED
```

```
       (
        [ConferenceDayID] ASC
           ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
           ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
ALTER TABLE [dbo].[ConferenceDays]
   WITH CHECK ADD CONSTRAINT [FK_ConferenceDays_Conferences]
FOREIGN
       KEY ([ConferenceID])
       REFERENCES [dbo].[Conferences] ([ConferenceID])
ALTER TABLE [dbo].[ConferenceDays]
   CHECK CONSTRAINT [FK_ConferenceDays_Conferences]
ALTER TABLE [dbo].[ConferenceDays]
   WITH CHECK ADD CONSTRAINT [CK_MaxParticipants] CHECK
       (([MaxParticipants] > (0)))
ALTER TABLE [dbo].[ConferenceDays]
   CHECK CONSTRAINT [CK_MaxParticipants]
```

3.8. ConferenceDayParticipants

Tabela przechowuje listę uczestników związaną z konkretną rezerwacją dnia konferencji. Zawiera:

3.8.1. **ConferenceDayParticipantID** - identyfikator uczestnika dnia konferencji

3.8.2. **ParticipantID** - identyfikator uczestnika

3.8.3. **ConferenceDayBookingID** - identyfikator powiązanej z nim rezerwacji dnia konferencji

3.8.4. **isStudent** - oraz flagę określającą czy uczestnik jest studentem

```
CREATE TABLE [dbo].[ConferenceDayParticipants]
(
   [ConfDayParticipantID]   [int]IDENTITY (1,1)NOT NULL,
   [ParticipantID]          [int]              NOT NULL,
   [ConferenceDayBookingID] [int]              NOT NULL,
   [isStudent]              [bit]              NOT NULL DEFAULT 0,
   CONSTRAINT [PK_ConferenceDayParticipants] PRIMARY KEY CLUSTERED
```

```
       (
        [ConfDayParticipantID] ASC
           ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
           ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
SET ANSI_PADDING OFF
ALTER TABLE [dbo].[ConferenceDayParticipants]
   WITH CHECK ADD CONSTRAINT
       [FK_ConferenceDayParticipants_ConferenceDayBooking] FOREIGN
KEY ([ConferenceDayBookingID])
           REFERENCES [dbo].[ConferenceDayBooking]
([ConferenceDayBookingID])
ALTER TABLE [dbo].[ConferenceDayParticipants]
   CHECK CONSTRAINT
[FK_ConferenceDayParticipants_ConferenceDayBooking]
ALTER TABLE [dbo].[ConferenceDayParticipants]
   WITH CHECK ADD CONSTRAINT
[FK_ConferenceDayParticipants_Participants]
       FOREIGN KEY ([ParticipantID])
           REFERENCES [dbo].[Participants] ([ParticipantID])
ALTER TABLE [dbo].[ConferenceDayParticipants]
   CHECK CONSTRAINT [FK_ConferenceDayParticipants_Participants]
```

3.9. Conferences

Tabela Conferences to tabela, w której przechowywane są podstawowe informacje o konferencjach. Zawiera:

3.9.1. **ConferenceID** - identyfikator konferencji
3.9.2. **Place** - lokalizację konferencji
3.9.3. **Name** - nazwę
3.9.4. **BeginTime** - data rozpoczęcia konferencji
3.9.5. **EndTime** - data zakończenia konferencji

```
CREATE TABLE [dbo].[Conferences]
(
   [ConferenceID] [int] IDENTITY (1,1) NOT NULL,
```

```
    [BeginTime]    [date]              NULL,
    [EndTime]      [date]              NULL,
    [Place]        [nvarchar](50)      NULL,
    [Name]         [nvarchar](50)      NOT NULL,
    [isCancelled]  [bit]               NOT NULL DEFAULT 0,
    CONSTRAINT [PK_Conferences] 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]
) ON [PRIMARY]
```

3.10. Prices

Tabela przechowuje ceny uczestnictwa w konferencji przy rezerwacji w danym dniu. Zawiera:

3.10.1. **PriceID** - identyfikator ceny
3.10.2. **ConferenceID** - identyfikator powiązanej konferencji
3.10.3. **Date** - datę obowiązywania
3.10.4. **Value** - wartość ceny brutto
3.10.5. **StudentsDiscount** - oraz zniżkę studencką w postaci liczbowej

```
CREATE TABLE [dbo].[Prices]
(
    [PriceID]          [int] IDENTITY (1,1) NOT NULL,
    [ConferenceID]     [int]                NOT NULL,
    [Date]             [date]               NOT NULL,
    [Value]            [money]              NOT NULL,
    [StudentsDiscount] [decimal](3, 2)      NOT NULL,
    CONSTRAINT [PK_Prices] PRIMARY KEY CLUSTERED
        (
         [PriceID] ASC
            ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
            ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
ALTER TABLE [dbo].[Prices]
    WITH CHECK ADD CONSTRAINT [FK_Prices_Conferences] FOREIGN KEY
```

```
([ConferenceID])
       REFERENCES [dbo].[Conferences] ([ConferenceID])
ALTER TABLE [dbo].[Prices]
   CHECK CONSTRAINT [FK_Prices_Conferences]
ALTER TABLE [dbo].[Prices]
   WITH CHECK ADD CONSTRAINT [CK_StudentsDiscount] CHECK
       (([StudentsDiscount] < (1) AND [StudentsDiscount] >= (0)))
ALTER TABLE [dbo].[Prices]
   CHECK CONSTRAINT [CK_StudentsDiscount]
```

3.11. ConferenceBooking

Tabela ConferenceBooking to zbiór informacji o rezerwacjach na konferencje dokonanych przez klientów. Zawiera:

3.11.1. **ConferenceBookingID** - identyfikator rezerwacji
3.11.2. **ConferenceID** - identyfikator konferencji, której dotyczy
3.11.3. **ClientID** - identyfikator klienta składającego rezerwację
3.11.4. **BookingTime** - czas rezerwacji
3.11.5. **isCancelled** - flagę określającą czy rezerwacja jest aktywna, czy anulowana, domyślnie 0

```
CREATE TABLE [dbo].[ConferenceBooking]
(
   [ConferenceBookingID] [int] IDENTITY (1,1) NOT NULL,
   [ConferenceID]        [int]                NOT NULL,
   [ClientID]            [int]                NOT NULL,
   [isCancelled]         [bit]                NOT NULL DEFAULT 0,
   [BookingTime]         [datetime]           NOT NULL,
   CONSTRAINT [PK_ConferenceBooking] PRIMARY KEY CLUSTERED
       (
        [ConferenceBookingID] ASC
           ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
           ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
ALTER TABLE [dbo].[ConferenceBooking]
   WITH CHECK ADD CONSTRAINT [FK_ConferenceBooking_Clients]
```

```
FOREIGN
       KEY ([ClientID])
       REFERENCES [dbo].[Clients] ([ClientID])
ALTER TABLE [dbo].[ConferenceBooking]
   CHECK CONSTRAINT [FK_ConferenceBooking_Clients]
ALTER TABLE [dbo].[ConferenceBooking]
   WITH CHECK ADD CONSTRAINT [FK_ConferenceBooking_Conferences]
FOREIGN
       KEY ([ConferenceID])
       REFERENCES [dbo].[Conferences] ([ConferenceID])
ALTER TABLE [dbo].[ConferenceBooking]
   CHECK CONSTRAINT [FK_ConferenceBooking_Conferences]
```

3.12. Payments

Tabela payments to lista płatności za udział w dniach konferencji i warsztatach, których dokonali klienci po złożeniu rezerwacji. Zawiera:

3.12.1. **PaymentID** - identyfikator płatności

3.12.2. **ConferenceBookingID** - identyfikator powiązanej rezerwacji konferencji

3.12.3. **Value** - zapłaconą wartość

3.12.4. **Time** - datę i godzinę płatności

3.12.5. **isCancelled** - flagę określającą, czy płatność jest aktywna (0) czy została anulowana (1),domyślnie 0

```
CREATE TABLE [dbo].[Payments]
(
   [PaymentID]           [int] IDENTITY (1,1) NOT NULL,
   [ConferenceBookingID] [int]                NOT NULL,
   [Value]               [money]              NOT NULL,
   [Time]                [datetime]           NOT NULL,
   [isCancelled]         [bit]                NOT NULL DEFAULT 0,
   CONSTRAINT [PK_Payments] PRIMARY KEY CLUSTERED
       (
        [PaymentID] ASC
           ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
           ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

```
ALTER TABLE [dbo].[Payments]
   WITH CHECK ADD CONSTRAINT [FK_Payments_ConferenceBooking]
FOREIGN
       KEY ([ConferenceBookingID])
       REFERENCES [dbo].[ConferenceBooking]
([ConferenceBookingID])
ALTER TABLE [dbo].[Payments]
   CHECK CONSTRAINT [FK_Payments_ConferenceBooking]
```

3.13. Clients

Tabela clients przechowuje informacje o klientach korzystających z usług systemu. Zawiera ona dane niezbędne do skontaktowania się z klientem . Zawiera:

3.13.1. **ClientID** - identyfikator klienta
3.13.2. **Name** - nazwę klienta
3.13.3. **Email** - adres e-mail. Adres email zawiera '@' i '.' i jest unikalny,
3.13.4. **Password** - zaszyfrowane hasło
3.13.5. **Street** - ulicę
3.13.6. **PostalCode** - kod pocztowy, postaci XX-XXX, XXXXXX, XXXXXXX, gdzie X to cyfra
3.13.7. **City** - miejscowość
3.13.8. **Country** - kraj
3.13.9. **isCompany** - flagę określającą, czy klient jest firmą (1) lub osobą prywatną (0), domyślnie 0

```
CREATE TABLE [dbo].[Clients]
(
   [ClientID]   [int] IDENTITY (1,1) NOT NULL,
   [isCompany]  [bit]                NOT NULL DEFAULT 0,
   [Name]       [nvarchar](50)       NOT NULL,
   [EMail]      [varchar](50)        NOT NULL UNIQUE,
   [Password]   [varchar](20)        NOT NULL,
   [Street]     [nvarchar](50)       NULL,
   [PostalCode] [varchar](10)        NULL,
   [City]       [nvarchar](50)       NULL,
   [Country]    [nvarchar](50)       NULL,
   CONSTRAINT [PK_Clients] 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],
    CONSTRAINT [UniqueEMail_Clients] UNIQUE NONCLUSTERED
        (
         [EMail] ASC
            ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
            ALLOW_PAGE_LOCKS = ON) ON [PRIMARY],
)
ALTER TABLE [dbo].[Clients]
    WITH CHECK ADD CONSTRAINT [CK_Clients_Email]
        CHECK ((EMail LIKE '%@%.%'))

ALTER TABLE [dbo].[Clients]
    WITH CHECK ADD CONSTRAINT [CK_Clients_PostalCode]
        CHECK (([PostalCode] LIKE '[0-9][0-9]-[0-9][0-9][0-9]'
            OR [PostalCode] LIKE '[0-9][0-9][0-9][0-9][0-9]'
            OR [PostalCode] LIKE '[0-9][0-9][0-9][0-9][0-9][0-9]'))
```

3.14. Companies

Tabela Companies zawiera informacje o firmach rezerwujących miejsca na konferencje i warsztaty. Zawiera:

3.14.1. **ClientID** - identyfikator klienta

3.14.2. **NIP** - Numer identyfikacji podatkowej, jest unikalny

```
CREATE TABLE [dbo].[Companies]
(
    [ClientID]      [INT]           NOT NULL,
    [NIP]           [CHAR](10)      NOT NULL,
    CONSTRAINT [PK_Companies] 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],
    CONSTRAINT [UQ_Companies_NIP] 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]
ALTER TABLE [dbo].[Companies]
   WITH CHECK ADD CONSTRAINT [FK_Clients_Companies] FOREIGN KEY
([ClientID])
       REFERENCES [dbo].[Clients] ([ClientID])
ALTER TABLE [dbo].[Companies]
```

4. Widoki

4.1. CanceledConferences

Widok przedstawia wszystkie anulowane konferencje.

```
CREATE VIEW CancelledConferences AS
SELECT dbo.Conferences.Name, dbo.Conferences.BeginTime,
dbo.Conferences.EndTime, dbo.Conferences.Place
FROM dbo.Conferences
WHERE dbo.Conferences.isCancelled = 1
```

4.2. CanceledWorkshops

Widok przedstawia wszystkie anulowane warsztaty. Dla każdego warsztatu wyświetlane są: nazwa warsztatu, daty rozpoczęcia i zakończenia warsztatu oraz miejsce, w którym warsztat miał się odbyć

```
CREATE VIEW CancelledWorkshops AS
SELECT Workshops.Name, Workshops.BeginTime, Workshops.EndTime,
Workshops.Place
FROM Workshops
WHERE Workshops.IsCancelled = 1
```

4.3. ClientsWithMostReservations

Widok przedstawia klientów, którzy dokonali największej liczby rezerwacji. Dla każdego klienta wyświetlane są: jego unikalne ID w systemie, jego nazwa i łączna liczba rezerwacji: na konferencje, na dni konferencji, na warsztaty i suma wszystkich trzech.

```
CREATE VIEW ClientsWithMostReservations AS
SELECT a.ClientID,
      Clients.Name,
      ConferenceBookingsCount,
      ConferenceDayBookingsCount,
      WorkshopsBookingsCount,
      ConferenceBookingsCount + ConferenceDayBookingsCount +
WorkshopsBookingsCount AS AllBookings
```

```
FROM (SELECT ClientID,
COUNT(ConferenceBooking.ConferenceBookingID) AS
ConferenceBookingsCount
     FROM ConferenceBooking
     GROUP BY ClientID) AS a
        JOIN
    (SELECT ClientID,
COUNT(ConferenceDayBooking.ConferenceDayBookingID) AS
ConferenceDayBookingsCount
     FROM ConferenceDayBooking
               LEFT JOIN ConferenceBooking ON
ConferenceDayBooking.ConferenceBookingID =

ConferenceBooking.ConferenceBookingID
     GROUP BY ClientID) AS b
    ON a.ClientID = b.ClientID
        JOIN
    (SELECT ConferenceBooking.ClientID,
COUNT(WorkshopsBooking.WorkshopBookingID) AS
WorkshopsBookingsCount
     FROM WorkshopsBooking
               LEFT JOIN ConferenceDayBooking ON
WorkshopsBooking.ConferenceDayBookingID =

ConferenceDayBooking.ConferenceDayBookingID
               LEFT JOIN ConferenceBooking ON
ConferenceDayBooking.ConferenceBookingID =

ConferenceBooking.ConferenceBookingID
     GROUP BY ConferenceBooking.ClientID) AS c
        INNER JOIN Clients on c.ClientID = Clients.ClientID
                   ON b.ClientID = c.ClientID
```

4.4. FutureConferences

Widok zwraca listę konferencji, które odbędą się w przyszłości (uwzględnia również konferencje obecnie trwające). Dla każdej konferencji wyświetla jej nazwę, ramy czasowe, lokalizację, całkowitą liczbę miejsc oraz liczbę miejsc już zarezerwowanych.

```
CREATE VIEW FutureConferences AS
```

```
SELECT Conferences.Name,
       Conferences.BeginTime,
       Conferences.EndTime,
       Conferences.Place,
       SUM(ConferenceDays.MaxParticipants) AS MaxParticipants,
       (SELECT SUM(daySum)
        FROM (SELECT SUM(ConferenceDayBooking.ParticipantsNo) AS
daySum, ConferenceDayID
              FROM ConferenceDayBooking
              WHERE ConferenceDayBooking.isCancelled = 0
              GROUP BY ConferenceDayID) AS s
                 JOIN ConferenceDays ON s.ConferenceDayID =
ConferenceDays.ConferenceDayID
                 JOIN Conferences AS c ON c.ConferenceID =
ConferenceDays.ConferenceID
        WHERE Conferences.ConferenceID = c.ConferenceID
        GROUP BY c.ConferenceID)           AS Reserved
FROM Conferences
         LEFT OUTER JOIN ConferenceDays ON Conferences.ConferenceID
=

ConferenceDays.ConferenceID
WHERE (Conferences.EndTime >= GETDATE())
GROUP BY Conferences.ConferenceID, Conferences.Name,
Conferences.BeginTime, Conferences.EndTime,
         Conferences.Place
```

4.5. UnpaidCompanyBookings

Widok zwraca listę firm, które nie opłaciły ani nie anulowały swoich rezerwacji. Dla każdej firmy wyświetla jej nazwę, identyfikator rezerwacji, której dotyczy brak płatności, email oraz nazwę konferencji.

```
CREATE VIEW UnpaidCompanyBookings AS
SELECT Clients.Name,
       ConferenceBooking.ConferenceBookingID,
       Clients.EMail    AS [Client Email],
       Conferences.Name AS [Conference Name]
FROM (ConferenceBooking LEFT JOIN Payments P on
```

```
ConferenceBooking.ConferenceBookingID = P.ConferenceBookingID)
        INNER JOIN Clients ON Clients.ClientID =
ConferenceBooking.ConferenceBookingID
        INNER JOIN Conferences ON Conferences.ConferenceID =
ConferenceBooking.ConferenceID
WHERE P.Value IS NULL
 AND ConferenceBooking.isCancelled = 0
 AND Clients.isCompany = 1
```

4.6. UnpaidPersonBookings

Widok zwraca listę osób fizycznych, które nie opłaciły ani nie anulowały swoich rezerwacji. Dla każdego klienta wyświetla jego nazwę, identyfikator rezerwacji, której dotyczy brak płatności, email oraz nazwę konferencji.

```
CREATE VIEW UnpaidPersonBookings AS
select Clients.Name,
     ConferenceBooking.ConferenceBookingID,
     Clients.EMail    AS [Client Email],
     Conferences.Name AS [Conference Name]
from (ConferenceBooking left join Payments P on
ConferenceBooking.ConferenceBookingID = P.ConferenceBookingID)
       INNER JOIN Clients ON Clients.ClientID =
ConferenceBooking.ConferenceBookingID
       INNER JOIN Conferences ON Conferences.ConferenceID =
ConferenceBooking.ConferenceID
Where P.Value IS NULL
AND ConferenceBooking.isCancelled = 0
AND Clients.isCompany = 0
```

4.7. ParticipantsInFutureConferences

Widok zwraca listę uczestników dnia konferencji dla przyszłych i obecnie trwających dni konferencji. Dla każdego uczestnika wyświetla jego numer, imię, nazwisko, nazwę konferencji oraz datę dnia konferencji.

```
CREATE VIEW ParticipantsInFutureConferences AS
```

```
SELECT DISTINCT Participants.ParticipantID,
                Participants.FirstName,
                Participants.LastName,
                ConferenceDays.Date,
                Conferences.Name As [Conference Name]
FROM Participants
        JOIN ConferenceDayParticipants ON
Participants.ParticipantID =
                  ConferenceDayParticipants.ParticipantID
        JOIN ConferenceDayBooking ON
ConferenceDayBooking.ConferenceDayBookingID =

ConferenceDayParticipants.ConferenceDayBookingID AND

ConferenceDayBooking.isCancelled = 0
        JOIN ConferenceDays ON
ConferenceDayBooking.ConferenceDayID =
                                ConferenceDays.ConferenceDayID
        JOIN Conferences ON ConferenceDays.ConferenceID =
Conferences.ConferenceID
WHERE (ConferenceDays.Date >= CAST(GETDATE() AS DATE))
```

4.8. FutureConferencesDays

Widok zwraca listę rezerwacji dni konferencji wraz z wraz z flagą isCancelled, datą konferencji, zadeklarowaną liczbą uczestników, studentów oraz liczbą uczestników, których dane zostały już uzupełnione (z podziałem na ogólną liczbę uczestników i liczbę studentów). Widok uwzględnia tylko te konferencje, które odbędą się w ciągu następnego miesiąca.

```
CREATE VIEW FutureConferencesDays AS
SELECT ConferenceDayBooking.ConferenceDayBookingID,
      ConferenceDayBooking.isCancelled,
      ConferenceDays.Date,
      ConferenceDayBooking.ParticipantsNo,
      ConferenceDayBooking.StudentsNo,
      COUNT(ConferenceDayParticipants.ConfDayParticipantID)
AS TotalCount,
      SUM(CASE WHEN ConferenceDayParticipants.isStudent = 1 THEN 1
ELSE 0 END) AS StudentCount
```

```
FROM ConferenceDayBooking
        LEFT JOIN ConferenceDayParticipants ON
ConferenceDayBooking.ConferenceDayBookingID =

ConferenceDayParticipants.ConferenceDayBookingID
        JOIN ConferenceDays ON ConferenceDays.ConferenceDayID =

ConferenceDayBooking.ConferenceDayID
WHERE ConferenceDays.Date <= DATEADD(day, 30, CAST(GETDATE() AS
date))
 AND ConferenceDays.Date >= CAST(GETDATE() AS date)
GROUP BY ConferenceDayBooking.ConferenceDayBookingID,
ConferenceDayBooking.isCancelled, ConferenceDays.Date,
        ConferenceDayBooking.ParticipantsNo,
ConferenceDayBooking.StudentsNo
```

4.9. FutureWorkshops

Widok zwraca listę rezerwacji warsztatów wraz z flagą isCancelled, nazwą, datą i czasem warsztatu, zadeklarowaną liczbą uczestników oraz liczbą uczestników, których dane zostały już uzupełnione. Widok uwzględnia tylko te warsztaty, które odbędą się w ciągu następnego miesiąca.

```
CREATE VIEW FutureWorkshops AS
SELECT WorkshopsBooking.WorkshopBookingID,
      WorkshopsBooking.isCancelled,
      Workshops.Name,
      ConferenceDays.Date,
      Workshops.BeginTime,
      Workshops.EndTime,
      WorkshopsBooking.ParticipantsNo,
      COUNT(WorkshopParticipants.ConfDayParticipantID) AS
fullParticipants
FROM WorkshopsBooking
        LEFT JOIN WorkshopParticipants ON
WorkshopsBooking.WorkshopBookingID =

WorkshopParticipants.WorkshopBookingID
        JOIN Workshops ON WorkshopsBooking.WorkshopID =
```

```
Workshops.WorkshopID
        JOIN ConferenceDays ON ConferenceDays.ConferenceDayID =
Workshops.ConferenceDayID
WHERE ConferenceDays.Date <= DATEADD(day, 30, CAST(GETDATE() AS
date))
 AND ConferenceDays.Date >= CAST(GETDATE() AS date)
GROUP BY WorkshopsBooking.WorkshopBookingID,
WorkshopsBooking.isCancelled, Workshops.Name,
        ConferenceDays.Date, Workshops.BeginTime,
Workshops.EndTime, WorkshopsBooking.ParticipantsNo
```

4.10. WorkshopOccupancy

Widok prezentuje identyfikator warsztatu, jego nazwę, ilość wszystkich miejsc, ilość zajętych miejsc oraz procentowe zapełnienie dostępnych miejsc na warsztatach

```
CREATE VIEW WorkshopOccupancy AS
Select WB1.WorkshopID,
      W1.name,
      W1.MaxParticipants,
      Sum(WB1.ParticipantsNo) AS EmptySeats,
      CAST(CONVERT(DECIMAL(5, 2), (Sum(WB1.ParticipantsNo)) /
CONVERT(DECIMAL(5, 2), W1.MaxParticipants)) *
           100 AS Varchar)    AS FillPercent
from WorkshopsBooking AS WB1
        INNER JOIN Workshops W1 on WB1.WorkshopID = W1.WorkshopID
GROUP BY WB1.WorkshopID, W1.MaxParticipants, w1.name
```

4.11. ConferenceOccupancy

Widok prezentuje identyfikator konferencji, jej nazwę, ilość wszystkich miejsc, ilość zajętych miejsc oraz procentowe zapełnienie dostępnych miejsc na konferencji

```
CREATE VIEW ConferenceOccupancy AS
SELECT ConferenceDays.ConferenceID,
      C.Name,
      sum(ConferenceDays.MaxParticipants) AS MaxParticipants,
```

```
      sum(CDB.ParticipantsNo)             AS TakenSeats,
      CONVERT(DECIMAL(15, 8), (Sum(CDB.ParticipantsNo)) /
CONVERT(DECIMAL(15, 8), ConferenceDays.MaxParticipants) * 100) AS
FillPercent
from ConferenceDays
        INNER JOIN ConferenceDayBooking CDB on
ConferenceDays.ConferenceDayID = CDB.ConferenceDayID
        INNER JOIN Conferences C on ConferenceDays.ConferenceID =
C.ConferenceID
GROUP BY ConferenceDays.ConferenceID, C.Name,
ConferenceDays.MaxParticipants
```

4.12. BookingsToCancell

Widok zwraca te rezerwacje, które powinny zostać anulowane, ponieważ zostały nieopłacone, a do rozpoczęcia konferencji pozostało mniej niż 7 dni.

```
CREATE VIEW BookingsToCancell AS
SELECT C.Name AS [Client name], C2.Name AS [Conference Name],
C2.BeginTime AS [Conference Begin Time]
from ConferenceBooking
        LEFT JOIN Payments P on
ConferenceBooking.ConferenceBookingID = P.ConferenceBookingID
        INNER JOIN Clients C on ConferenceBooking.ClientID =
C.ClientID
        INNER JOIN Conferences C2 on
ConferenceBooking.ConferenceID = C2.ConferenceID
WHERE P.Time is null
 AND ABS(DATEDIFF(Day, GETDATE(), C2.BeginTime)) < 7
```

5. Procedury

5.1. Procedury dotyczące klientów i uczestników konferencji

5.1.1. AddNewClient

Procedura, która dodaje nowego klienta. Jako argumenty przyjmuje nazwę klienta, adres e-mail, który nie pojawił się jeszcze w bazie, oraz hasło (Argumenty wymagane), dane teleadresowe oraz NIP (argumenty opcjonalne). Bit isStudent przybiera wartość zależną od argumentu NIP.

```
CREATE PROCEDURE AddNewClient @Name nvarchar(50),
```

```
                                @EMail varchar(50),
                                @Password varchar(100),
                                @Street nvarchar(50) = null,
                                @PostalCode nvarchar(10) = null,
                                @City nvarchar(20) = null,
                                @Country nvarchar(20) = null,
                                @NIP varchar(10) = null
AS
BEGIN
    SET NOCOUNT ON
    IF (@NIP IS NOT NULL)
        Begin
            INSERT INTO Clients (Name, EMail, Password, Street,
PostalCode, City, Country, isCompany)
            VALUES (@Name, @EMail, '*********', @Street,
@PostalCode, @City, @Country, 1)
            INSERT INTO Companies (ClientID, NIP)
            VALUES ((SELECT ClientID
                     from Clients
                     WHERE Email = @Email), @NIP)
        end
    ELSE
        BEGIN
            INSERT INTO Clients (Name, EMail, Password, Street,
PostalCode, City, Country, isCompany)
            VALUES (@Name, @EMail, '*********', @Street,
@PostalCode, @City, @Country, 0)
        end
END
```

5.1.2. UpdateClientData

Procedura, która zmienia klienta. Zmienia wartości tylko tych pól, które zostały podane jako argumenty.

```
CREATE PROCEDURE UpdateClientData    @EMail varchar(50),
                                @Name nvarchar(50) = NULL,
                                @Password varchar(100) = NULL,
                                @Street nvarchar(50) = NULL,
                                @PostalCode varchar(10) = NULL,
```

```
                                    @City nvarchar(50) = NULL,
                                    @Country nvarchar(50) = NULL,
                                    @NIP varchar(10) = NULL
AS
BEGIN
    SET NOCOUNT ON;
    IF @Name IS NOT NULL
        BEGIN
            UPDATE Clients SET Name = @Name WHERE EMail = @EMail
        END
    IF @Password IS NOT NULL
        BEGIN
            UPDATE Clients SET Password = @Password WHERE EMail =
@EMail
        END
    IF @Street IS NOT NULL
        BEGIN
            UPDATE Clients SET Street = @Street WHERE EMail =
@EMail
        END
    IF @PostalCode IS NOT NULL
        BEGIN
            UPDATE Clients SET PostalCode = @PostalCode WHERE EMail
= @EMail
        END
    IF @City IS NOT NULL
        BEGIN
            UPDATE Clients SET City = @City WHERE EMail = @EMail
        END
    IF @Country IS NOT NULL
        BEGIN
            UPDATE Clients SET Country = @Country WHERE EMail =
@EMail
        END
    IF @NIP IS NOT NULL
        BEGIN
            UPDATE Clients SET isCompany = 1 WHERE EMail = @EMail
            IF ((SELECT Companies.ClientID
                 from Companies
                          INNER JOIN Clients C on
Companies.ClientID = C.ClientID
```

```
                WHERE C.EMail = @EMail) IS NOT NULL)
            BEGIN
                UPDATE Companies
                SET NIP = @NIP
                WHERE ClientID = (SELECT Companies.ClientID
                                  from Companies
                                           INNER JOIN Clients C
on Companies.ClientID = C.ClientID
                                  WHERE C.EMail = @EMail)
            end
        ELSE
            BEGIN
                INSERT INTO Companies (ClientID, NIP)
                VALUES ((SELECT ClientID
                         from Clients
                         WHERE Email = @Email), @NIP)
            end
    END
END
```

5.1.3. AddParticipant

Procedura, która dodaje uczestnika konferencji. Jako argumenty przyjmuje imię i nazwisko (argumenty wymagane) oraz dane teleadresowe (argumenty opcjonalne).

```
CREATE PROCEDURE AddParticipant @FirstName nvarchar(50),
                                @LastName nvarchar(50),
                                @Street nvarchar(50) = null,
                                @PostalCode nvarchar(10) = null,
                                @City nvarchar(50) = null,
                                @Country nvarchar(50) = null,
                                @StudentID nvarchar(50) = null,
                                @valid date = null
AS
BEGIN
   SET NOCOUNT ON
   IF @StudentID IS NOT NULL
       BEGIN
           INSERT INTO Participants (FirstName, LastName, Street,
```

```
PostalCode, City, Country, isStudent)
            VALUES (@FirstName, @LastName, @Street, @PostalCode,
@City, @Country, 1)
            INSERT INTO Student (ParticipantID, StudentID, Valid)
            VALUES ((select ParticipantID
                     from Participants
                     WHERE FirstName = @FirstName
                       AND LastName = @LastName
                       And Street = @Street
                       AND PostalCode = @PostalCode
                       AND City = @City
                       And Country = @Country), @StudentID, @valid)
        end
    ELSE
        BEGIN
            INSERT INTO Participants (FirstName, LastName, Street,
PostalCode, City, Country, isStudent)
            VALUES (@FirstName, @LastName, @Street, @PostalCode,
@City, @Country, 0)
        end

END
```

5.1.4. AddParticipantToConferenceDay

Procedura, która dodaje istniejącego w bazie uczestnika do dnia konferencji. Jako argumenty przyjmuje identyfikator uczestnika oraz identyfikator rezerwacji dnia konferencji

```
CREATE PROCEDURE AddParticipantToConferenceDay @ParticipantID int,
                                               @ConferenceDayBookingID int,
AS
BEGIN
    SET NOCOUNT ON
    INSERT INTO ConferenceDayParticipants (ParticipantID,
ConferenceDayBookingID)
    VALUES (@ParticipantID, @ConferenceDayBookingID)
END
```

5.2. Procedury dotyczące konferencji

5.2.1. AddConference

Procedura, która dodaje konferencję. Jako argumenty przyjmuje nazwę konferencji, datę rozpoczęcia i zakończenia oraz lokalizację konferencji.

```
CREATE PROCEDURE AddConference @Name nvarchar(50),
                               @BeginTime date,
                               @EndTime date,
                               @Place nvarchar(50)
AS
BEGIN
   SET NOCOUNT ON;
   INSERT INTO Conferences (BeginTime, EndTime, Place, Name)
   VALUES (@BeginTime, @EndTime, @Place, @Name)
END
```

5.2.2. CancelConference

Procedura, która anuluje konferencę. Jako argument przyjmuje identyfikator konferencji

```
CREATE PROCEDURE CancelConference @ConferenceID int
AS
BEGIN
   SET NOCOUNT ON
   UPDATE Conferences SET isCancelled = 1 Where ConferenceID =
@ConferenceID
END
```

5.2.3. AddConferenceDay

Procedura, która dodaje dzień konferencji. Jako argumenty przyjmuje identyfikator konferencji, lokalizację, limit uczestników oraz datę. Dzień konferencji może zostać dodany tylko wtedy, gdy jego data pokrywa się z czasem trwania konferencji, do której należy.

```
CREATE PROCEDURE AddConferenceDay @ConferenceID int,
                                  @Place nvarchar(50),
```

```
                                    @MaxParticipants int,
                                    @Date date
AS
BEGIN
    SET NOCOUNT ON
    DECLARE @ConfStart date = (SELECT BeginTime FROM Conferences
WHERE ConferenceID = @ConferenceID)
    DECLARE @ConfEnd date = (SELECT EndTime FROM Conferences WHERE
ConferenceID = @ConferenceID)
    DECLARE @Allowed int = (SELECT Count(*)
                            FROM ConferenceDays
                            WHERE ConferenceID = @ConferenceID
                              AND Date = @Date)
    IF (@Date >= @ConfStart AND @Date <= @ConfEnd AND @Allowed = 0)
        BEGIN
            INSERT INTO ConferenceDays (ConferenceID, Place,
MaxParticipants, Date)
            VALUES (@ConferenceID, @Place, @MaxParticipants, @Date)
        END
    ELSE
        BEGIN
            RAISERROR ('Wrong date',-1,-1)
        END
END
```

5.2.4. UpdateConferenceDay

Procedura, która zmienia dzień konferencji. Zmienia wartości tylko tych pól, które zostały podane jako argumenty.

```
CREATE PROCEDURE UpdateConferenceDay @ConferenceDayID int,
                                    @Place nvarchar(50) = NULL,
                                    @MaxParticipants int = NULL
AS
BEGIN
  SET NOCOUNT ON
  IF @Place IS NOT NULL
   BEGIN
        UPDATE ConferenceDays SET Place = @Place Where
```

```
ConferenceDayID = @ConferenceDayID
    end
        IF @MaxParticipants IS NOT NULL
    BEGIN
         UPDATE ConferenceDays SET MaxParticipants =
@MaxParticipants Where ConferenceDayID = @ConferenceDayID
    end
END
```

5.3. Procedury dotyczące rezerwacji konferencji

5.3.1. AddConferenceBooking

Procedura, która dodaje rezerwację. Jako argumenty przyjmuje identyfikator konferencji oraz klienta.

```
CREATE PROCEDURE AddConferenceBooking @ConferenceID int,
                                      @ClientID     int
AS
BEGIN
    INSERT INTO ConferenceBooking (ConferenceID, ClientID,
isCancelled, BookingTime)
    VALUES (@ConferenceID, @ClientID, 0, GETDATE())
END
```

5.3.2. CancelConferenceBooking

Procedura anuluje rezerwację konferencji, jako argument przyjmuje identyfikator rezerwacji konferencji.

```
CREATE PROCEDURE CancelConfBooking @ConferenceBookingID int
AS
BEGIN
    SET NOCOUNT ON;
    UPDATE ConferenceBooking SET isCancelled = 1 WHERE
ConferenceBookingID = @ConferenceBookingID
END
```

5.3.3. AddConferenceDayBooking

Procedura dodająca nową rezerwację dnia konferencji. Jako argumenty przyjmuje identyfikator rezerwacji konferencji, liczbę uczestników (łącznie ze studentami), liczbę studentów oraz identyfikator dnia konferencji. Procedura dodaje rezerwację tylko wtedy, gdy liczba rezerwowanych miejsc jest mniejsza lub równa liczbie miejsc dostępnych do rezerwacji.

```
CREATE PROCEDURE AddConferenceDayBooking @ConferenceBookingID int,
                                         @ParticipantsNo int,
                                         @StudentsNo int,
                                         @ConferenceDayID int
AS
BEGIN
   DECLARE @MaxParticipants int = (SELECT MaxParticipants
                                   FROM ConferenceDays
                                   WHERE ConferenceDayID =
                                         @ConferenceDayID)
   DECLARE @AlreadyBooked int = (SELECT
isnull(SUM(ParticipantsNo), 0)
                                 FROM ConferenceDayBooking
                                 WHERE ConferenceDayID =
@ConferenceDayID)
   DECLARE @Allowed int = (SELECT
                           Count(*)
                            FROM ConferenceBooking
                                    JOIN Conferences ON
ConferenceBooking.ConferenceID
                                = Conferences.ConferenceID
                                    JOIN ConferenceDays ON
ConferenceDays.ConferenceID =

Conferences.ConferenceID
                           WHERE ConferenceDayID =
@ConferenceDayID
                             AND ConferenceBookingID =
                                 @ConferenceBookingID)
   IF (@ParticipantsNo + @AlreadyBooked <= @MaxParticipants AND
@Allowed = 1)
       BEGIN
           INSERT INTO ConferenceDayBooking (ConferenceBookingID,
```

```
BookingTime, ParticipantsNo, StudentsNo,
                                                ConferenceDayID,
isCancelled)
            VALUES (@ConferenceBookingID, GETDATE(),
@ParticipantsNo, @StudentsNo, @ConferenceDayID, 0)
        END
    ELSE
        BEGIN
            RAISERROR ('ParticipantsNo over MaxParticipants',-1,-1)
        END
END
```

5.3.4. CancelConferenceBooking

Procedura, która anuluje rezerwację dnia konferencji. Jako argument przyjmuje identyfikator dnia konferencji.

```
CREATE PROCEDURE CancelConfereceBooking @ConferenceBookingID int
AS
BEGIN
    SET NOCOUNT ON
    UPDATE ConferenceBooking SET isCancelled = 1 WHERE
ConferenceBookingID = @ConferenceBookingID
END
```

5.4. Procedury dotyczące płatności

5.4.1. AddPrice

Procedura, która dodaje cenę dla konferencji w podanym zakresie dni. Jako argumenty przyjmuje zakres dni, identyfikator konferencji, cenę oraz zniżkę studencką. Płatność zostaje dodana tylko gdy data końca zakresu jest mniejsza bądź równa dacie konferencji, której dotyczy

```
CREATE PROCEDURE AddPrice @from date,
                          @to date,
                          @ConferenceID int,
                          @Value money,
                          @StudentsDiscount decimal(3, 2)
```

```
AS
BEGIN
    SET NOCOUNT ON;
    DECLARE @ConfDate date = (SELECT EndTime FROM Conferences WHERE
ConferenceID = @ConferenceID)
    IF (@from < @to) AND (@to <= @ConfDate)
        BEGIN
            WHILE @from <= @to
                BEGIN
                    INSERT INTO Prices (ConferenceID, Date, Value,
StudentsDiscount)
                    VALUES (@ConferenceID, @from, @Value,
@StudentsDiscount)
                    SET @from = (SELECT DATEADD(day, 1, @from))
                END
        END
    ELSE
        BEGIN
            RAISERROR ('From >= To',-1,-1)
        END
END
```

5.4.2. AddPayment

Procedura, która dodaje płatność za konferencję. Jako argumenty przyjmuje identyfikator rezerwacji konferencji oraz wartość płatności

```
CREATE PROCEDURE AddPayment @ConferenceBookingID int,
                            @Value money
AS
BEGIN
    SET NOCOUNT ON;
    INSERT INTO Payments (ConferenceBookingID, Value, Time,
isCancelled)
    VALUES (@ConferenceBookingID, @Value, GETDATE(), 0)
END
```

5.4.3. CancelPayment

Procedura, która anuluje płatność. Jako argumenty przyjmuje identyfikator płatności

```
CREATE PROCEDURE CancelPayment @PaymentID int
AS
BEGIN
   SET NOCOUNT ON;
   UPDATE Payments SET isCancelled = 1 WHERE PaymentID =
@PaymentID
END
```

5.5. Procedury dotyczące warsztatów

5.5.1. AddWorkshop

Procedura, która dodaje warsztat. Jako argumenty przyjmuje identyfikator dnia konferencji, nazwę warsztatu, jego czas trwania, lokalizację, cenę oraz limit uczestników

```
CREATE PROCEDURE AddWorkshop @ConferenceDayID int,
                             @Name nvarchar(50),
                             @BeginTime time(7),
                             @EndTime time(7),
                             @Place nvarchar(50),
                             @Price money,
                             @MaxParticipants int
AS
BEGIN
   SET NOCOUNT ON
   INSERT INTO Workshops (ConferenceDayID, Name, BeginTime,
EndTime, Place, Price,
                          MaxParticipants)
   VALUES (@ConferenceDayID, @Name, @BeginTime, @EndTime, @Place,
@Price, @MaxParticipants)
END
```

5.5.2. UpdateWorkshop

Procedura, która zmienia warsztat. Modyfikacji ulegają tylko te pola, które są podane jako argumenty

```
CREATE PROCEDURE UpdateWorkshop @WorkshopID int,
                            @Name nvarchar(50),
                            @Place nvarchar(50),
```

```
                                @Price money,
                                @MaxParticipants int
AS
BEGIN
    SET NOCOUNT ON
    IF @Name IS NOT NULL
        BEGIN
            UPDATE Workshops SET Name = @Name Where WorkshopID =
@WorkshopID
        end
    IF @Place IS NOT NULL
        BEGIN
            UPDATE Workshops SET Place = @Place Where WorkshopID =
@WorkshopID
        end
    IF @Price IS NOT NULL
        BEGIN
            UPDATE Workshops SET Price = @Price Where WorkshopID =
@WorkshopID
        end
    IF @MaxParticipants IS NOT NULL
        BEGIN
            UPDATE Workshops SET MaxParticipants = @MaxParticipants
Where WorkshopID = @WorkshopID
        end
END
```

5.5.3. AddWorkshopBooking

Procedura, która dodaje rezerwację na warsztat. Jako argumenty przyjmuje identyfikator rezerwacji dnia konferencji, identyfikator warsztatu oraz liczbę uczestników. Procedura dodaje rezerwację tylko wtedy, gdy liczba rezerwowanych miejsc jest mniejsza lub równa liczbie miejsc dostępnych do rezerwacji oraz warsztat odbywa się podczas dnia konferencji, którego dotyczy rezerwacja dnia.

```
CREATE PROCEDURE AddNewWorkshopBooking @ConferenceDayBookingID
int,
                                       @WorkshopID int,
                                       @ParticipantsNo int
```

```
AS
BEGIN
    DECLARE @MaxParticipants AS int
    DECLARE @AlreadyBooked AS int
    SET @MaxParticipants = (SELECT MaxParticipants FROM Workshops
WHERE WorkshopID = @WorkshopID)
    SET @AlreadyBooked = (SELECT isnull(SUM(ParticipantsNo), 0)
                          FROM WorkshopsBooking
                          WHERE WorkshopID = @WorkshopID)
    DECLARE @Allowed AS int
    SET @Allowed = (SELECT
                    TOP 1
                    1
                    FROM ConferenceDayBooking
                             JOIN ConferenceDays ON
                        ConferenceDayBooking.ConferenceDayID =
ConferenceDays.ConferenceDayID
                             JOIN Workshops ON
                        Workshops.ConferenceDayID =
ConferenceDays.ConferenceDayID
                    WHERE WorkshopID = @WorkshopID
                      AND ConferenceDayBookingID =
@ConferenceDayBookingID)
    IF (@MaxParticipants >= @AlreadyBooked + @ParticipantsNo AND
@Allowed = 1)
        BEGIN
            INSERT INTO WorkshopsBooking (ConferenceDayBookingID,
WorkshopID, ParticipantsNo,
                                          isCancelled, BookingTime)
            VALUES (@ConferenceDayBookingID, @WorkshopID,
@ParticipantsNo, 0, GETDATE())
        END
    ELSE
        BEGIN
            RAISERROR ('ParticipantsNo over MaxParticipants', -1,
-1)
        END
END
```

5.5.4. CancelWorkshopBooking

Procedura, która anuluje rezerwację warsztatu. Jako argument przyjmuje identyfikator warsztatu.

```
CREATE PROCEDURE CancelWorkshopBooking @WorkshopBookingID int
AS
BEGIN
    SET NOCOUNT ON
  UPDATE WorkshopsBooking SET isCancelled = 1 WHERE
WorkshopBookingID = @WorkshopBookingID
END
```

5.5.5. AddParticipantToWorkshop

Procedura, która dodaje uczestnika dnia konferencji do warsztatu. Jako argumenty przyjmuje identyfikator uczestnika oraz warsztatu. By uczestnik został dodany, musi być uczestnikiem dnia konferencji, do którego należy warsztat oraz w czasie trwania warsztatu nie może być zarejestrowany na inny warsztat.

```
CREATE PROCEDURE AddParticipantToWorkshop @ParticipantID int,
                                          @WorkshopBookingID int
AS
BEGIN
    SET NOCOUNT ON
    DECLARE @ConferenceDayBookingID int = (SELECT
ConferenceDayBookingID
                                           FROM WorkshopsBooking
                                           WHERE WorkshopBookingID
= @WorkshopBookingID)
    DECLARE @ConferenceDayParticipantID int =(SELECT
ConfDayParticipantID
                                              FROM
ConferenceDayParticipants
                                              WHERE
ConferenceDayBookingID = @ConferenceDayBookingID
                                                AND ParticipantID =
@ParticipantID)
    DECLARE @WorkshopID int = (SELECT WorkshopID
                               FROM WorkshopsBooking
```

```
                                WHERE WorkshopBookingID =
                                      @WorkshopBookingID)
   DECLARE @Allowed int = (SELECT Count(*)
                           FROM WorkshopParticipants AS wp
                                JOIN WorkshopsBooking AS wb ON
wp.WorkshopBookingID = wb.WorkshopBookingID
                                JOIN Workshops AS w ON
w.WorkshopID = wb.WorkshopID
                           WHERE wp.ConfDayParticipantID IN
                                 (SELECT ConfDayParticipantID
                                  FROM ConferenceDayParticipants
                                  WHERE ParticipantID =
@ParticipantID)
                             AND (w.BeginTime < (SELECT
Workshops.EndTime
                                                 FROM Workshops
                                                 WHERE
Workshops.WorkshopID = @WorkshopID)
                                  OR w.EndTime > (SELECT
Workshops.BeginTime
                                                  FROM Workshops
                                                  WHERE
Workshops.WorkshopID = @WorkshopID)))
   IF @Allowed = 0
       BEGIN
           INSERT INTO WorkshopParticipants (WorkshopBookingID,
ConfDayParticipantID)
           VALUES (@WorkshopBookingID,
@ConferenceDayParticipantID)
       END
   ELSE
       BEGIN
           RAISERROR ('The participant takes part in another
Workshop at the time',-1,-1)
       END
END
```

6. Triggery

6.1. ConferenceDayParticipantsCheckTrigger

Trigger, który sprawdza czy liczba uczestników powiązanych z rezerwacją dnia konferencji nie przekracza liczby zadeklarowanej.

```
CREATE TRIGGER ConferenceDayParticipantsCheckTrigger
   ON ConferenceDayParticipants
   FOR INSERT AS
BEGIN
   DECLARE @ConferenceDayBookingID int = (SELECT
ConferenceDayBookingID FROM inserted)
   DECLARE @ParticipantsCount int = (SELECT COUNT(*)
                                     FROM
ConferenceDayParticipants
                                     WHERE ConferenceDayBookingID
=

@ConferenceDayBookingID)
   DECLARE @ParticipantsNo int = (SELECT ParticipantsNo
                                  FROM ConferenceDayBooking
                                  WHERE ConferenceDayBookingID =
                                        @ConferenceDayBookingID)
   IF (@ParticipantsNo < @ParticipantsCount)
       BEGIN
           Throw 52000, 'Participants number over ParticipantsNo
from ConferenceDayBooking', 1
           ROLLBACK TRANSACTION
       END
END
```

6.2. WorkshopParticipantsCheckTrigger

Trigger, który sprawdza czy liczba uczestników powiązanych z rezerwacją warsztatu nie przekracza liczby zadeklarowanej.

```
CREATE TRIGGER WorkshopParticipantsCheckTrigger
   ON WorkshopParticipants
   FOR INSERT AS
```

```
BEGIN
   DECLARE @WorkshopBookingID int = (SELECT WorkshopBookingID FROM
inserted)
   DECLARE @ParticipantsCount int = (SELECT COUNT(*)
                                     FROM WorkshopParticipants
                                     WHERE WorkshopBookingID =
                                           @WorkshopBookingID)
   DECLARE @ParticipantsNo int = (SELECT ParticipantsNo
                                  FROM WorkshopsBooking
                                  WHERE WorkshopBookingID =
                                        @WorkshopBookingID)
   IF (@ParticipantsNo < @ParticipantsCount)
       BEGIN
           THROW 52000, 'Participants number over ParticipantsNo
from WorkshopsBooking', 1
           ROLLBACK TRANSACTION
       END
END
```

6.3. CancelConferenceTrigger

Trigger, który ustawia flagę isCancelled dla rezerwacji konferencji powiązanych z konferencji, gdy anulowana jest konferencja.

```
CREATE TRIGGER CancelConferenceTrigger
   ON Conferences
   FOR UPDATE AS
BEGIN
   IF UPDATE(isCancelled)
       BEGIN
           DECLARE @ConferenceID int = (SELECT ConferenceID FROM
inserted)
           IF (SELECT isCancelled FROM inserted) = 1
               BEGIN
                   UPDATE ConferenceBooking SET isCancelled = 1
WHERE ConferenceBookingID = @ConferenceID
               END
       END
END
```

6.4. CancellConferenceBookingTrigger

Trigger, który ustawia flagę isCancelled dla rezerwacji dni konferencji powiązanych z rezerwacją konferencji, gdy anulowana jest rezerwacja konferencji

```
CREATE TRIGGER CancelConferenceBookingTrigger
   ON ConferenceBooking
   FOR UPDATE AS
BEGIN
   IF UPDATE(isCancelled)
       BEGIN
           DECLARE @ConferenceBookingID int = (SELECT
ConferenceBookingID FROM inserted)
           IF (SELECT isCancelled FROM inserted) = 1
               BEGIN
                   UPDATE ConferenceDayBooking SET isCancelled = 1
WHERE ConferenceBookingID = @ConferenceBookingID
               END
       END
END
```

.

6.5. CancelWorkshopBookingTrigger

Trigger, który ustawia flagę isCancelled dla rezerwacji dni warsztatów powiązanych z rezerwacją dnia konferencji, gdy anulowana jest rezerwacja dnia konferencji.

```
CREATE TRIGGER CancelWorkshopBookingTrigger
   ON ConferenceDayBooking
   FOR UPDATE AS
BEGIN
   IF UPDATE(isCancelled)
       BEGIN
           DECLARE @ConferenceDayBookingID int = (SELECT
ConferenceDayBookingID FROM inserted)
           IF (SELECT isCancelled FROM inserted) = 1
```

```
                BEGIN
                    UPDATE WorkshopsBooking SET isCancelled = 1
                    WHERE ConferenceDayBookingID =
@ConferenceDayBookingID
                END
        END
END
```

6.6. ConferenceDayWithinConference

Trigger, który blokuje dodanie dnia konferencji, poza datą rozpoczęcia i zakończenia konferencji

```
CREATE TRIGGER ConferenceDayWithinConference
   ON ConferenceDays
   AFTER INSERT
   AS
BEGIN
   SET NOCOUNT ON
   IF EXISTS(
            SELECT *
            FROM inserted as i
                     join Conferences AS c ON c.ConferenceID =
i.ConferenceID
            WHERE i.Date > C.EndTime OR i.Date <C.BeginTime
        )
        BEGIN
            ; THROW 50001 , 'Conference day(s) is(are) outside of
conference duration .' ,1
        end
end
```

6.7. BookPastConference

Trigger, który uniemożliwia dodanie rezerwacji dnia konferencji, poza zakończeniu tego dnia.

```
CREATE TRIGGER BookPastConference
   ON ConferenceDayBooking
   AFTER INSERT
   AS
BEGIN
   SET NOCOUNT ON
   IF EXISTS(
            SELECT *
            FROM inserted as i
                     join ConferenceDayBooking AS c ON
c.ConferenceBookingID = i.ConferenceDayBookingID
                     join ConferenceDays CD on c.ConferenceDayID =
CD.ConferenceDayID
            WHERE i.BookingTime > CD.Date
       )
       BEGIN
           ; THROW 50001 , 'Booking is outside of conference day
duration .' ,1
       end
end
```

6.8.

7. Funkcje

7.1. Conference_days

Funkcja zwraca wpisy w tabeli conferenceDays odpowiadające dniom konferencji, której id jest podane.

```
CREATE FUNCTION conference_days(
   @conferenceId INT
)
   RETURNS @days TABLE
                 (
                     conferenceDayId INT,
                     date              DATE,
                     MaxParticipants   INT
```

```
                    )
AS
BEGIN
    INSERT INTO @days
    SELECT ConferenceDayID, Date, MaxParticipants
    FROM ConferenceDays
    WHERE ConferenceID = @conferenceId
    RETURN
END
```

7.2. PriceOnDay

Funkcja zwraca koszt konferencji przy zamówieniu złożonym w danym dniu

```
CREATE FUNCTION PriceOnDate(@conferenceId INT,
                            @date DATE)
    RETURNS INT
AS
BEGIN
    RETURN (
        SELECT
        top 1
        Prices.Value
        FROM Prices
        WHERE Prices.ConferenceID = @conferenceId
          and Prices.Date = @date
        ORDER BY Prices.Date
    )
END
```

7.3. AvailablePlaces

Funkcja zwraca ilość wolnych miejsc na dany dzień konferencji.

```
CREATE FUNCTION availablePlaces(@conferenceDayId INT
)
    RETURNS INT
AS
BEGIN
    RETURN (
```

```
        SELECT ConferenceDays.MaxParticipants -
SUM(CDB.ParticipantsNo)
        FROM ConferenceDays
                LEFT JOIN ConferenceDayBooking CDB
                        on ConferenceDays.ConferenceDayID =
CDB.ConferenceDayID AND CDB.isCancelled = 0
        WHERE ConferenceDays.ConferenceDayID = @conferenceDayId
        GROUP BY ConferenceDays.ConferenceDayID,
ConferenceDays.MaxParticipants
    )
END
```

7.4. BeginTimeWorkshop

Funkcja zwraca datę i godzinę rozpoczęcia warsztatu.

```
CREATE FUNCTION BeginTimeWorkshop(@WorshopID INT
)
    RETURNS datetime
AS
BEGIN
    RETURN (
        SELECT CAST(Workshops.BeginTime AS DATETIME) + CAST(CD.Date
AS DATETIME)
        FROM Workshops
                INNER JOIN ConferenceDays CD on
Workshops.ConferenceDayID = CD.ConferenceDayID
        WHERE WorkshopID = @WorshopID
    )
END
```

7.5. EndTimeWorkshop

Funkcja zwraca datę i godzinę zakończenia warsztatu.

```
CREATE FUNCTION EndTimeWorkshop(@WorshopID INT
)
    RETURNS datetime
AS
BEGIN
    RETURN (
```

```
        SELECT CAST(Workshops.EndTime AS DATETIME) + CAST(CD.Date
AS DATETIME)
        FROM Workshops
                INNER JOIN ConferenceDays CD on
Workshops.ConferenceDayID = CD.ConferenceDayID
        WHERE WorkshopID = @WorshopID
    )
END
```

7.6. AreWorkshopAtTheSameTime

Funkcja zwraca bit, oznajmujący czy dane dwa warsztaty nachodzą na siebie

```
CREATE FUNCTION AreWorkshopAtTheSameTime(@Workshop1 int,
                                         @Workshop2 int)
    RETURNS bit
AS
BEGIN
    DECLARE @Begin1 datetime = dbo.BeginTimeWorkshop(@Workshop1);
    DECLARE @End1 datetime = dbo.EndTimeWorkshop(@Workshop1);
    DECLARE @Begin2 datetime =dbo.BeginTimeWorkshop(@Workshop2);
    DECLARE @End2 datetime = dbo.EndTimeWorkshop(@Workshop2);
    IF (@Begin1 < @Begin2 AND @Begin2 < @End1) RETURN 1
    IF (@Begin2 < @Begin1 AND @Begin1 < @End2) RETURN 1
    IF (@Begin1 >= @Begin2 AND @End2 >= @End1) RETURN 1
    IF (@Begin2 >= @Begin1 AND @End1 >= @End2) RETURN 1
    RETURN 0
end
```

8. Indeksy

8.1. Indeks ConferenceBookingConferenceIDIndex

```
CREATE NONCLUSTERED INDEX ConferenceBooking-ConferenceIDIndex ON
ConferenceBooking
    (
     ConferenceID ASC
```

```
       ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
   ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.2. Indeks ConferenceBookingClientIDIndex

```
CREATE NONCLUSTERED INDEX ConferenceBookingClientIDIndex ON
ConferenceBooking
(
[ClientID] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.3. Indeks ConferenceDayBookingConferenceBookingIDIndex

```
CREATE NONCLUSTERED INDEX ConferenceDayBookingConfBookingIDIndex
ON ConferenceDayBooking
   (
    ConferenceBookingID ASC
       ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
   ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.4. Indeks ConferenceDayBookingConferenceDayIDIndex

```
CREATE NONCLUSTERED INDEX ConferenceDayBookingConferenceDayIDIndex
ON ConferenceDayBooking
   (
    ConferenceDayID ASC
       ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
   ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.5. Indeks ConferenceDayParticipantsParticipantIDIndex

```
CREATE NONCLUSTERED INDEX
ConferenceDayParticipantsParticipantIDIndex ON
ConferenceDayParticipants
   (
    ParticipantID ASC
       ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
   ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.6. Indeks ConferenceDayParticipantsConferenceDayBookingIDIndex

```
CREATE NONCLUSTERED INDEX
ConferenceDayParticipantsConferenceDayBookingIDIndex ON
   ConferenceDayParticipants
       (
        ConferenceDayBookingID ASC
           ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
   ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.7. Indeks StudentsStudentID

```
CREATE NONCLUSTERED INDEX StudentsStudentID ON Student
   (
    StudentID ASC
       ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
   ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.8. Indeks PaymentsConferenceBookingIDIndex

```
CREATE NONCLUSTERED INDEX PaymentsConferenceBookingIDIndex ON
Payments
   (
```

```
    ConferenceBookingID ASC
       ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
   ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.9. Indeks WorkshopParticipantsWorkshopBookingIDIndex

```
CREATE NONCLUSTERED INDEX
WorkshopParticipantsWorkshopBookingIDIndex ON
   WorkshopParticipants
       (
        WorkshopBookingID ASC
           ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
   ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.10. Indeks WorkshopParticipantsConferenceDayParticipantIDIndex

```
CREATE NONCLUSTERED INDEX
WorkshopParticipantsConferenceDayParticipantIDIndex ON
   WorkshopParticipants
       (
        ConferenceDayParticipantID ASC
           ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
   ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.11. Indeks WorkshopsConferenceDayIDIndex

```
CREATE NONCLUSTERED INDEX WorkshopsConferenceDayIDIndex ON
Workshops
   (
    ConferenceDayID ASC
       ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
   ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.12. Indeks WorkshopsBookingConferenceDayBookingIDIndex

```
CREATE NONCLUSTERED INDEX
WorkshopsBookingConferenceDayBookingIDIndex ON WorkshopsBooking
    (
     ConferenceDayBookingID ASC
        ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
    ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.13. Indeks WorkshopsBookingWorkshopIDIndex

```
CREATE NONCLUSTERED INDEX WorkshopsBookingWorkshopIDIndex ON
WorkshopsBooking
    (
     WorkshopID ASC
        ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
    ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

8.14. Indeks CompaniesClientIDIndex

```
CREATE NONCLUSTERED INDEX CompaniesClientIDIndex ON Companies
    (
     ClientID ASC
        ) WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF,
SORT_IN_TEMPDB = OFF, DROP_EXISTING = OFF,
    ONLINE = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON)
```

9. Role

W systemie proponujemy zdefiniowanie następujących ról:

- Administrator systemu ma nieograniczony dostęp do bazy danych, pełna kontrola nad systemem.

- Pracownik firmy zajmującej się organizacją konferencji ma dostęp do wszystkich procedur i widoków, możliwość modyfikowania danych w tabeli.
- Organizator konferencji ma dostęp do funkcji umożliwiających dodanie nowej konferencji, dostęp do funkcji i widoków zwracających informacje o wolnych miejscach, płatnościach, a także o uczestnikach biorących udział w konferencjach i warsztatach dla organizowanej konferencji.
- Klient ma dostęp do procedur powiązanych z tworzeniem nowych rezerwacji, anulowaniem i usuwaniem rezerwacji na konferencje i warsztaty, a także dodawania klientów i uczestników.

Powyżej opisane role nie zostały uwzględnione w implementacji.

10. Generator

W projekcie do wygenerowania potrzebnych danych wykorzystany został program Red Gate SQL Data Generator (http://www.redgate.com/products/sql-development/sql-data-generator). Podczas generowania danych wyłączone zostały triggery ConferenceDayParticipantsCheckTrigger oraz WorkshopParticipantsCheckTrigger odpowiedzialne za sprawdzanie, czy liczba uczestników nie przekracza liczby zadeklarowanej.